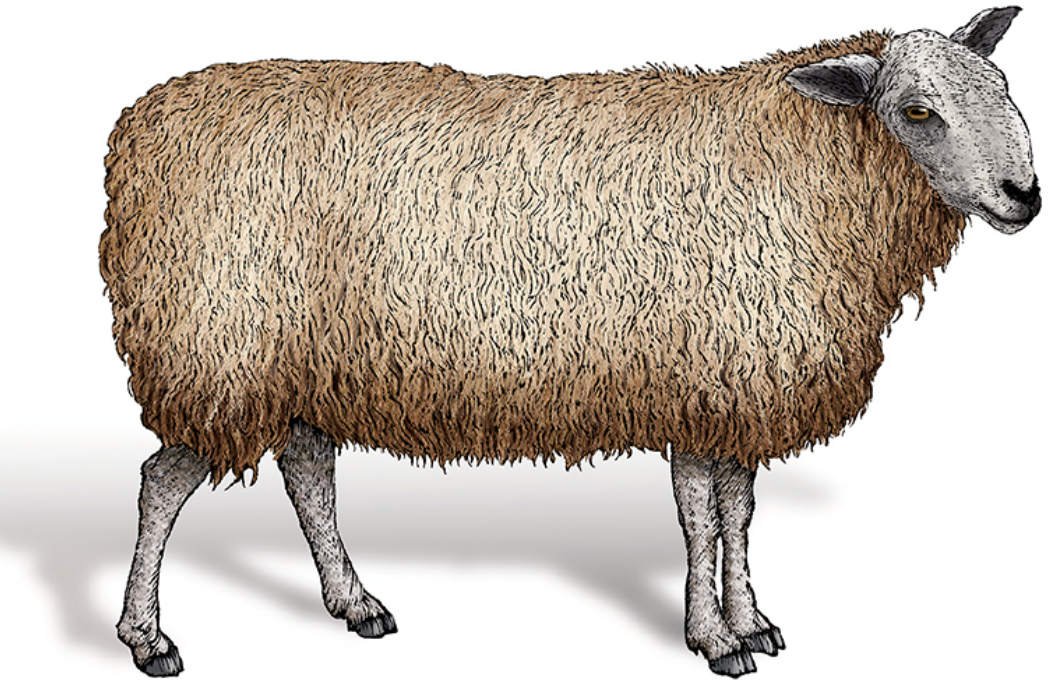
O'REILLY®

Практический анализ временных рядов

Прогнозирование со статистикой и машинное обучение



Эйлин Нильсен

Практический анализ временных рядов

Прогнозирование со статистикой и машинное обучение

Practical Time Series Analysis

Prediction with Statistics and Machine Learning

Aileen Nielsen

Beijing · Boston · Farnham · Sebastopol · Tokyo O'REILLY®

Практический анализ временных рядов

Прогнозирование со статистикой и машинное обучение

Эйлин Нильсен

ДИАЛЕКТИКА

Москва · Санкт-Петербург
2021

ББК 32.813.5
Н66
УДК 004.8

ООО "Диалектика"

Зав. редакцией *С.Н. Тригуб*

Перевод с английского и редакция докт. физ.-мат. наук *Д.А. Клюшина*

При участии *И.В. Василенко*

По общим вопросам обращайтесь в издательство "Диалектика" по адресу:
info.dialektika@gmail.com, http://www.dialektika.com

Нильсен, Эйлин.

Н66 Практический анализ временных рядов: прогнозирование со статистикой и машинное обучение. : Пер. с англ. — СПб. : ООО "Диалектика", 2021. — 544 с. : ил. — Парал. тит. англ.

ISBN 978-5-907365-04-9 (рус.)

ББК 32.813.5

Научно-популярное издание

Эйлин Нильсен

Практический анализ временных рядов: прогнозирование со статистикой и машинное обучение

Подписано в печать 15.12.2020. Формат 70x100/16
Усл. печ. л. 43,9. Уч.-изд. л. 29,2
Тираж 300 экз. Заказ № 8669.

Отпечатано в АО "Первая Образцовая типография"
Филиал "Чеховский Печатный Двор"
142300, Московская область, г. Чехов, ул. Полиграфистов, д. 1
Сайт: www.chpd.ru, E-mail: sales@chpd.ru, тел. 8 (499) 270-73-59

ООО "Диалектика", 195027, Санкт-Петербург, Магнитогорская ул., д. 30, лит. А, пом. 848

ISBN 978-5-907365-04-9 (рус.)

ISBN 978-1-492-04165-8 (англ.)

Оглавление

Содержание

Об авторе

Эйлин Нильсен — разработчик программного обеспечения и специалист по анализу данных из Нью-Йорка. Она является общепризнанным специалистом по временны*м* рядам и другим структурированным данным, принимающим участие в самых разных проектах: от стартапов в сфере здравоохранения, политических кампаний и обустройства исследовательских лабораторий до биржевой торговли в крупных финансовых компаниях. В настоящее время она разрабатывает приложения прогнозирования, основанные на нейронных сетях.

Колофон

Животное на обложке книги — *голубомордая овца*. Эта британская порода, также известная как *голубомордый лейстер* или *лейстерская овца*, была выведена в XVIII веке в г. Дишли, графство Лестершир. Сегодня эту специализированную породу овец разводят фермеры по всей Великобритании и Канаде.

Голубомордые лейстеры ценятся белой шерстью, из которой получают тонкую, блестящую и высококачественную пряжу высокой упругости. Их темные глаза сильно контрастируют со светло-голубой мордой. Овцы этой породы отличаются хорошей выносливостью, имеют длинные спины, крепкие плечи, широкие морды и темные копыта. Шерсть на ногах и мордах почти отсутствует.

Взрослая особь голубомордой овцы весит от 70 до 115 кг и в холке достигает высоты 85–90 см. Как и большинство овец других пород, голубомордый лейстер требует регулярного выпаса на целинных, залежных или горных пастбищах. Среднее стадо насчитывает около 20 голов.

Лейстерские овцы высоко ценятся заводчиками за способность скрещиваться с другими породами для производства гибридных ягнят на убой.

Многие из животных на обложках книг издательства O'Reilly находятся под угрозой исчезновения, и поэтому очень важно сохранить их в естественной среде обитания.

Иллюстрация на обложке создана Карен Монтгомери (Karen Montgomery) на основании черно-белой гравюры из энциклопедического словаря *Meyers Kleines Lexicon*.

Введение

Погода, фондовые рынки и сердцебиение. Все это описывается временны*ми* рядами. Чаще всего для построения прогноза для некой последовательности данных прибегают к тщательному анализу имеющихся временны*х* рядов.

Добро пожаловать в мир анализа временны*х* рядов! Если вы держите в руках эту книгу, то, вероятно, уже заметили, что временны*е* ряды окружают нас повсюду. В последнее время они становятся важной или даже неотъемлемой частью экосистем больших данных. На сегодняшний день установка всевозможных датчиков и устройств наблюдения, генерирующих высокочастотные последовательности сигналов, стала обыденным явлением — все они требуют представления и обработки в виде временны*х* рядов. Временны*е* ряды невероятно удобны тем, что при правильном анализе позволяют не только рассмотреть тенденции в существующих наборах данных, но и составить прогноз будущих изменений. В этой книге вы познакомитесь с основными принципами работы с временны*ми* рядами, обеспечивающими решение указанных и некоторых других задач.

Временны*е* ряды востребованы в широком спектре научных и инженерных дисциплин. Временны*м* рядом можно представить все что угодно — от журнала торговых сделок до последовательности данных о проводимости наноэлектронной системы и записи человеческой речи в цифровом формате. В этой книге мы будем исходить из того, что временны*е* ряды могут содержать наборы данных самых разнообразных типов. Заметим, что описанными далее методами можно обрабатывать и анализировать любые наборы данных, имеющие упорядоченную структуру в некотором, не обязательно временно*м*, направлении. Традиционно временны*е* ряды применяются для хранения вполне ожидаемых последовательностей данных, например сведений об изменениях складских запасов и погодных условий. Однако не менее успешно их можно использовать и для анализа более экзотичной информации, в частности данных, получаемых спектрографом при анализе винного материала. В последнем случае ось “времени” фактически заменяется осью частоты. Временны*е* ряды используются повсеместно!

Для кого предназначена эта книга

Эта книга рассчитана на читателей двух категорий. Первая, ощутимо бо́льшая, категория — это специалисты по анализу и обработке данных, которым по долгу службы приходится работать с временны́ми рядами, но делают они это не очень часто. Это могут быть как ветераны отрасли, так и начинающие аналитики. Опытным специалистам материал первых глав покажется знакомым, но это не значит, что им можно пренебречь, — здесь описаны самые современные методы обработки данных и рассмотрены важные особенности управления временны́ми рядами. Аналитикам с небольшим рабочим опытом желательно проработать все без исключения главы книги предельно внимательно, несмотря на их тематическую независимость друг от друга.

Вторая категория читателей — руководители отделов по обработке и анализу данных в компаниях с интенсивным внутренним сбором информации. Если вы относитесь к этой группе читателей, то должны быть в курсе технологических решений, применяемых для обработки временны́х рядов, хотя вам и не приходится заниматься программированием самостоятельно. Для вас эта книга будет полезна тем, что обозначит область применения временны́х рядов в существующих или создаваемых заново алгоритмах сбора и анализа данных. Назначение этой книги — помочь вам разобраться в технологиях, призванных упростить обработку существующих ресурсов данных.

Ожидаемый уровень подготовки

Что касается программирования, предполагается, что вам знакомы языки R и Python, особенно такие их фундаментальные пакеты, как `NumPy`, `Pandas`, `scikit-learn` (в Python) и `data.table` (в R). Конечно, приведенные в книге примеры программных кодов можно читать и безо всякой подготовки, но в этом случае вам потребуется сделать небольшое отступление и самостоятельно ознакомиться с этими пакетами. В наибольшей степени это касается пакета `data.table` языка R, который не пользуется широкой популярностью, но является чрезвычайно производительным инструментом с просто фантастической скоростью обработки данных.

В соответствующих главах книги приведено краткое описание всех основных пакетов, включающее примеры их практического использования, а также детальный обзор выполняемых операций. Кроме того, в них вы найдете ссылки на ресурсы, содержащие более полные сведения о часто используемых программных библиотеках.

Что касается статистического анализа и машинного обучения, вы должны быть знакомы со следующими научными и предметными областями.

Основы математической статистики

Включает описание таких понятий, как дисперсия, корреляция и распределение вероятностей.

Машинное обучение

Предполагает знакомство с методами кластеризации и деревьями решений.

Нейронные сети

Требуется понимание того, что они собой представляют и как обучаются.

В книге приведен только краткий обзор обозначенных выше понятий, поэтому, если вы с ними встречаетесь впервые, прежде чем перейти к изучению отдельных глав, познакомьтесь с ними поближе. В большинстве случаев в соответствующих главах содержатся ссылки на бесплатные ресурсы, включающие полное описание рассматриваемых тем или методов.

Почему именно эта книга

Я написал эту книгу по трем причинам.

Во-первых, временны*е* ряды, хотя и являются важным аспектом, не относятся к стандартным средствам анализа данных. Это прискорбно как из-за того, что временны*е* ряды становятся все более и более доступными для обработки, так и потому, что они однозначно отвечают на вопросы, на которые не могут ответить срезы данных. Аналитик, не знакомый с фундаментальными принципами анализа временны*х* рядов, не использует большинство своих данных. Я надеюсь, что эта книга сможет заполнить существующий и важный пробел.

Во-вторых, перед тем, как приступить к написанию книги, я не нашел ни одного полноценного руководства по анализу данных временны*х* рядов в разрезе современной науки о данных. В Интернете есть много отличных ресурсов, содержащих описание основных принципов использования временны*х* рядов, в которых рассматриваемые темы чаще всего представляются в формате классических математических учебников по статистическому анализу. Кроме того, доступно много отличных блогов, посвященных традиционным статистическим методам, их роли в технологиях машинного обучения и анализу временны*х* рядов с помощью нейронных сетей. Однако мне не удалось найти ни одного ресурса, на котором все эти темы рассматривались бы неотрывно друг от друга, как взаимосвязанные понятия. Цель этой книги — представить на ваше суждение такой ресурс: всеобъемлющий, включающий описание современных технологий и рассматривающий практические примеры использования временны*х* рядов в анализе и моделировании данных. Опять

же, я надеюсь, что эта книга сможет заполнить имеющийся важный пробел в науке о данных.

В-третьих, временны*е* ряды — это интересная тема со специфической проблематикой. Задачи описания утечек данных, прогнозирования и нахождения причинных связей особенно интересны в контексте анализа временны*х* рядов, как и многие методы обработки данных, упорядоченных вдоль некоторой временно*й* оси. Всеобъемлющий обзор обозначенных тем, систематизация и поиск взаимосвязей между ними — еще одна причина написания этой книги.

Структура книги

Эта книга организована следующим образом.

История

В главе 1 описана история развития представлений о временны*х* рядах от древних греков до современности. Познакомившись с ней, вы поймете, насколько важную и сложную дисциплину мы изучаем в этой книге.

Все о данных

В главах 2–5 рассматриваются задачи получения, очистки, моделирования и хранения данных временны*х* рядов. Эти главы связаны между собой контекстом предварительной обработки данных, которая предшествует непосредственному анализу временны*х* рядов. Такие темы редко обсуждаются на узкоспециализированных ресурсах, но чрезвычайно важны при рассмотрении большинства конвейеров данных. Распознавание и очистка данных отнимают очень много времени и усилий, которые тратятся большинством аналитиков при исследовании временны*х* рядов.

Модели, макеты, модели

В главах 6–10 описаны всевозможные модели, применяемые для анализа временны*х* рядов. Первые две главы посвящены статистическим методам, основанным на стандартных статистических моделях, таким как ARIMA и пространство состояний Байеса. В последующих главах рассматриваются более современные подходы, предполагающие анализ данных временны*х* рядов с помощью алгоритмов машинного обучения и нейронных сетей. В них основное внимание уделяется задачам обработки и размещения данных, в которых временны*е* ряды используются для подгонки моделей, изначально не учитывающих время, таких как деревья решений.

Анализ моделей

В главах 11 и 12 рассматриваются понятия точности и производительности моделей, определяющие область применения модели и факторы, которые

должны учитываться после выполнения нескольких первых проходов моделирования временны*х* рядов.

Решение практических задач

В главах 13–15 приведено описание некоторых тематических задач по анализу медицинских, финансовых и статистических данных.

Наиболее популярные решения

Главы 16 и 17 включают краткое описание самых популярных технологий, применяемых при решении задач анализа временны*х* рядов и прогнозирования. В главе 16 приведен обзор различных пакетов автоматизированного анализа временны*х* рядов (зачастую с открытым исходным кодом), как разработанных крупными технологическими компаниями, так и полученных в рамках отдельных академических исследований. Эти инструменты постоянно совершенствуются, чтобы соответствовать постоянно возрастающим запросам на создание автоматизированных систем прогнозирования, основанных на анализе временны*х* рядов. В главе 17 обсуждаются трудности подобного прогнозирования, связанные с ростом размера экосистем больших данных, и рассматриваются средства обработки временны*х* рядов, заимствованные у таких экосистем.

В общем случае я рекомендовал бы сначала внимательно ознакомиться с материалом главы и только после этого приступать к изучению примеров приведенных в ней программных кодов. Обычно в каждой главе представлено сразу несколько новых концепций — на их проработку у вас уйдет достаточно много времени и усилий. Кроме того, в большинстве случаев код реализации описанных моделей не очень сложный, и вашим основным приобретением станет не сама программа, а понимание принципов ее построения и дальнейшего использования. Знание интерфейсов прикладного программирования большинства вспомогательных пакетов станет еще одним, не менее важным, приобретением, овладеть которым будет намного легче, если вы обратите внимание на концепции.

Книга рассчитана на последовательное изучение, начиная с первой и заканчивая последней главой. Помните, что материал более поздних глав основан на концепциях, рассмотренных в начальных главах книги, и если их опустить, то дальнейшее изучение может оказаться очень затруднительным. Конечно, я старался сделать каждую главу как можно более законченной, чтобы позволить читателям, обладающим определенными знаниями в рассматриваемых дисциплинах, пропускать уже знакомые им темы и акцентироваться исключительно на новом материале.

Принятые условные обозначения

В этой книге широко используются следующие обозначения.

Курсив
: Указывает новые термины и важные сведения.

`Моноширинный шрифт`
: Используется для листингов программ, а также в абзацах для ссылки на элементы программ, такие как имена переменных или функций, базы данных, типы данных, переменные среды, операторы и ключевые слова. Им представлены URL-адреса, адреса электронной почты, имена и расширения файлов.

`Моноширинный полужирный шрифт`
: Демонстрирует команды или другой текст, который должен быть набран пользователем вручную.

`Моноширинный курсив`
: Демонстрирует текст, который должен быть заменен предоставленными пользователем значениями или значениями, определенными контекстом.

Советы содержат сведения, позволяющие упростить выполняемую работу. Они позволяют взглянуть на излагаемый материал немного под другим углом зрения.

Примечания предоставляют дополнительную, вспомогательную информацию, которая может оказаться полезной, но немного выделяется из потока излагаемых сведений.

В предупреждениях приводятся сведения, которые призваны привлечь внимание читателя. Они позволяют узнать, о чем не следует забывать в процессе работы.

Примеры исходных кодов

Дополнительный материал (примеры кода, упражнения и т.п.) доступен для скачивания по такому адресу:

`http://www.williamspublishing.com/Books/978-5-907365-04-9.html`

Книга призвана помочь в решении стоящих перед вами задач, поэтому не бойтесь использовать все доступные для загрузки данные в собственных целях. Любые файлы, которые вы найдете на указанном сайте, можно смело применять в других проектах или ссылаться на них в учебных пособиях. При этом совсем не обязательно ставить в известность их авторов, за исключением случаев коммерческого использования больших текстовых фрагментов книги. В частности, использование приведенного в книге кода при создании частных проектов вполне допустимо, а вот его продажа и распространение на любом из носителей категорически запрещены. Подобным образом разрешается бесплатно использовать фрагменты кода из книги в собственных программах, но никак не включать их в руководства или справочные пособия к собственным коммерческим продуктам.

По возможности ссылайтесь на материалы, взятые из этой книги, в своих проектах, хотя это необязательное требование. Если вы не знаете, нужно ли получать специальное разрешение на использование программных кодов, приведенных в книге, в вашем конкретном случае, не поленитесь связаться с нами по электронной почте:

`permissions@oreilly.com`

Благодарности

Огромное спасибо обоим техническим рецензентам этой книги, профессорам Робу Хиндману и Дэвиду Стоффере. Оба были исключительно любезны, выделив время на изучение этой книги и предоставив подробные и необычайно полезные советы по ее наполнению. Вне всяких сомнений, после рецензирования ими книга стала заметно лучше. Я благодарен Робу за обнаружение недоработок в первоначальном проекте, неоценимую помощь в построении альтернативных методологий и предоставление интересных источников временны*х* рядов. Отдельное спасибо Дэвиду за здоровый скепсис в отношении чрезмерной автоматизации разрабатываемых инструментов и своевременное указание на явную переоценку возможностей многих из них. Спасибо за предоставленную помощь, ценные советы и глубокий анализ описанных в книге программных решений.

Хотелось бы поблагодарить редактора из O'Reilly Джеффа Блейла, который был самым оптимистичным, воодушевляющим и полезным партнером проекта за последний год. Я также должен выразить огромную благодарность техническому редактору Кэти Тозер, которая удивительно терпеливо занималась подготовкой этой книги к печати и помогала решать любые проблемы, связанные с ее производством. Спасибо Рэйчел Монаган за тщательное и отличное редактирование. Специальной благодарности заслуживает Ребекка Демарест за создание множества иллюстраций и всестороннюю помощь в художественном оформлении. Спасибо также Джонатану Хасселу, который взялся за сопровождение этого проекта и необычайно удивительным образом убедил издательство O'Reilly опубликовать его. Наконец, огромная благодарность Николь Таш, которая была первой, кто предложил мне сотрудничество с издательством O'Reilly. Всем сотрудникам издательства O'Reilly спасибо за поддержку и всестороннюю помощь в реализации проекта.

Я искренне благодарен читателям, которые поддерживают со мной обратную связь и указывают на найденные ошибки и недочеты, в том числе Вэнфей Тонг, Ричарду Краюнусу, Заку Богарту, Гейбу Фернандо, Лауре Кеннеди, Стивену Финкельштейн, Лиане Хиттс и Джейсону Гринбергу. Их вклад необычайно ценен как для проекта, так и для меня лично. Спасибо за внимательное изучение книги и своевременные отзывы.

Я благодарю мою маму (и образец для подражания), Элизабет, за пожизненную любовь, поддержку и дисциплину. Я также выражаю признательность отцу Джону и тете Клэр за любовь и поддержку моего образования на протяжении многих лет. Больше всех я хочу поблагодарить мужа Ивана и сына Эдмунда Хиллари за проявленное терпение и незыблемую веру в меня. Надеюсь, они смогут простить меня за отсутствие в кругу семьи во время написания книги.

Все ошибки, найденные в этой книге, принадлежат только моей руке. Связаться со мной можно по такому адресу:

aileen.a.nielsen@gmail.com.

Ждем ваших отзывов!

Вы, читатель этой книги, и есть главный ее критик. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересны любые ваши замечания в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам электронное письмо либо просто посетить наш сайт и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится ли вам эта книга, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

Отправляя письмо или сообщение, не забудьте указать название книги и ее авторов, а также свой обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию новых книг.

Наши электронные адреса:

E-mail: `info.dialektika@gmail.com`
WWW: `http://www.dialektika.com`

ГЛАВА 1

Временны*е* ряды: обзор и краткая история

Временны*е* ряды и их анализ становятся все более востребованными структурами данных ввиду их стремительного распространения, что вызвано самыми разными факторами, включая Интернет вещей, переход к электронному документообороту и совершенствование технологий умных городов. В ближайшие годы спрос на технологии обработки данных, представленных временны*ми* рядами, вполне закономерно будет только увеличиваться.

По мере возрастания запроса на средства непрерывного мониторинга и сбора данных потребность в инструментах анализа временны*х* рядов с использованием как статистических, так и машинных методов обучения будет только увеличиваться. В действительности наиболее перспективные модели основаны на обеих технологиях. Именно поэтому мы остановимся на рассмотрении каждой из них как можно подробнее. В круг наших интересов попадают самые разные методы изучения временны*х* рядов, востребованных в научных исследованиях, бизнес-анализе и даже прогнозировании поведения человека — каждая из дисциплин предполагает сбор и обработку больших массивов временны*х* данных.

Начнем с определения. *Анализ временных рядов* предполагает извлечение важной сводной и статистической информации из точек данных, заданных в хронологическом порядке. Это операция позволяет не только изучить прошлые состояния, но и спрогнозировать положение будущих точек данных. В этой книге анализ временны*х* рядов выполняется с помощью нескольких подходов, основанных как на статистических моделях столетней давности, так и на самых современных наработках, в которых основная роль отведена нейронным сетям.

Ни один из рассматриваемых далее методов не развивался отдельно от остальных и не разрабатывался из чисто теоретического интереса. Инновации в технологиях анализа временны*х* рядов являются результатом постоянного совершенствования инструментов сбора, записи и визуализации данных. Далее мы кратко остановимся на истории появления знаний о временны*х* рядах в различных дисциплинах.

История возникновения знаний о временны*х* рядах

Анализ временны*х* рядов часто сводится к задаче о причинности: как прошлое повлияло на будущее? Иногда такие задачи (и их решения) рассматриваются исключительно в пределах некой дисциплины и не выделяются в отдельную область знаний о временны*х* рядах. Результатом такого подхода стало то, что в исследование принципов анализа временны*х* рядов внесли вклад очень многие дисциплины.

В этом разделе мы рассмотрим несколько примеров исторических данных и способов анализа временны*х* рядов в следующих дисциплинах.

- Медицина.
- Прогноз погоды.
- Экономика.
- Астрономия.

Как мы увидим, темпы развития указанных дисциплин и вклады, вносимые каждой из областей знаний, тесно связаны с характером исследуемых временны*х* рядов.

Временны*е* ряды в медицине

Медицина — это область знаний, основанная на постоянном анализе информации, которая на протяжении нескольких веков самым тщательным образом изучала временны*е* ряды данных с целью извлечения наиболее ценных сведений. Рассмотрим несколько примеров исторических временны*х* рядов в медицине и проследим за эволюцией инструментов их обработки во времени.

Медицина начала задумываться об использовании математики для прогнозирования возможных ситуаций удивительно поздно, и это несмотря на то, что прогнозы являются неотъемлемой частью медицинской практики. Тому есть сразу несколько объяснений. Статистика и вероятностное представление об окружающем мире — это очень молодые области знаний, которые оставались неизведанными человечеством в течение многих веков существования медицинской науки. Кроме того, большинство врачей практиковали в полной изоляции, без какого бы то ни было обмена профессиональным опытом со своими коллегами и общей потребности в ведении записей о здоровье как отдельных пациентов, так и всего населения вверенного им региона. Следовательно, даже если бы врачи располагали достаточными познаниями в области статистической обработки данных, они, вероятнее всего, не имели бы в своем распоряжении достаточного количества данных, чтобы сделать определенные выводы.

Но не будем поддавать лишней критике врачей прошлого, а всего лишь примем за должное приведенное выше объяснение для понимания того, почему одно из первых упоминаний временны*х* рядов в области здравоохранения пришло к нам отнюдь не от медицинских работников, а от продавца шляп. Если задуматься, то в этом нет ничего удивительного: в прошлые века продавец городских шляп, скорее всего, имел больше опыта в ведении делопроизводства и искусстве выявления тенденций, чем врач.

Новатором был Джон Граунт, лондонский галантерейщик XVII века. Граунт предпринял исследование записей о смерти, которые хранились в лондонских приходах с начала 1500-х годов. При этом он создал дисциплину, которая называется демографией. В 1662 году он опубликовал книгу *Natural and Political Observations ... Made upon the Bills of Mortality* (Естественные и политические наблюдения над списками умерших) (рис. 1.1).

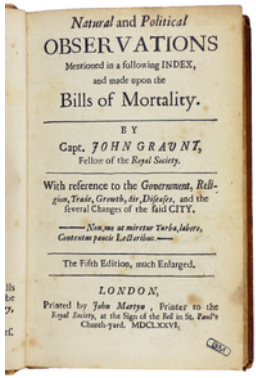

Natural and *Political*
OBSERVATIONS
Mentioned in a following INDEX,
and made upon the
Bills of Mortality.
BY
Capt. *JOHN GRAUNT*,
Fellow of the *Royal Society*.
With reference to the *Government*, *Religion*, *Trade*, *Growth*, *Air*, *Diseases*, and the several Changes of the said CITY.
——*Non me ut miretur Turba, laboro, Contentus paucis Lectoribus.*——
The Fifth Edition, much Enlarged.
LONDON,
Printed by *John Martyn*, Printer to the *Royal Society*, at the Sign of the Bell in St. *Paul's* Church-yard. MDCLXXVI.

Рис. 1.1. *Актуарные таблицы Джона Граунта — один из первых примеров анализа временных рядов применительно к медицинским данным.*

Источник: Википедия `https://ru.wikipedia.org/wiki/Граунт_Джон`

В этой книге Граунт представил первые *таблицы дожития*, которые сейчас относятся к классу *актуарных таблиц*. В них рассчитывается вероятность того, что человек определенного возраста не доживет до своего следующего дня рождения. Но Джон Граунт известен не только как первый человек, который разработал и опубликовал таблицы дожития, но и как первый документально подтвержденный статистический исследователь в медицине. Его таблицы дожития выглядели примерно так, как показано в табл. 1.1, взятой из материалов лекций по статистической математике Университета Райса (`https://perma.cc/HU6A-9W22`).

Таблица 1.1. Образец таблиц дожития Джона Граунта

Возраст, лет	Доля умерших за период времени	Доля доживших до начала периода
0–6	0,36	1,0
7–16	0,24	0,64
17–26	0,15	0,40
27–36	0,09	0,25

К сожалению, математический подход Граунта к решению задачи о дожитии до определенного возраста не был по достоинству оценен современниками. В то время в научных кругах уже начало формироваться представление о взаимосвязанном и управляемом данными мире — с национальными государствами, аккредитацией научных сотрудников, профессиональными сообществами, научными журналами и, несколько позже, государственным медицинским учетом. Медицина продолжала акцентироваться на физиологии, а не на статистике.

Развитие медицины как дисциплины обусловливалось вполне понятными причинами. Во-первых, изучение анатомии и физиологии небольшого числа субъектов обеспечило основные достижения в медицине на протяжении многих веков, и поэтому большинство врачей (и даже ученых) изо всех сил придерживались выработанных ранее профессиональных традиций. А так как изучение физиологии позволило добиться более чем очевидных успехов, заинтересованности в переходе к новым научным направлениям не проявлял почти никто. Во-вторых, у врачей было очень мало средств и возможностей для составления подробных таблиц и повсеместного обмена информацией, что, в свою очередь, сделало статистические методы менее полезными, чем клинические наблюдения.

Общие принципы анализа временны*х* рядов были разработаны позже, чем остальные инструменты статистической математики, поскольку такой анализ более требователен к системам учета. Записи должны быть связаны во времени и, предпочтительно, собираться через одинаковые промежутки времени. По этой причине временны*е* ряды начали использоваться в эпидемиологических исследованиях совсем недавно и не сразу, а только по мере создания достаточной государственной и научной инфраструктур, обеспечивающих достоверность и долгосрочность хранения данных временны*х* записей.

Аналогичным образом персонифицированная медицинская помощь, основанная на анализе временны*х* рядов, все еще остается новой и сложной задачей, поскольку создание согласованных во времени наборов данных сопряжено с целым рядом трудностей. Даже в небольших тематических исследованиях, основанных

на конкретных данных, организация и координация действий группы людей остается чрезвычайно трудным и дорогостоящим занятием. Если же такие исследования проводятся в течение длительных периодов времени, то они часто становятся каноническими в своих областях и многократно (порой чрезмерно тщательно) изучаются, потому что полученные данные могут послужить основой для решения самых разных задач, даже несмотря на извечные проблемы с финансированием и просчетами в управлении.[1]

Медицинские инструменты

Анализ временны*х* рядов применительно к отдельным пациентам имеет гораздо более раннюю и успешную практику, чем история их использования для исследования общего здоровья населения. Впервые анализом временны*х* рядов в медицине стали заниматься после изобретения в 1901 году первых аппаратов регистрации электрокардиограммы (ЭКГ), которые позволяли диагностировать сердечные заболевания по электрическим сигналам, проходящим через сердце (рис. 1.2). Следующим медицинским устройством с регистрацией данных в виде временны*х* рядов стал аппарат записи электроэнцефалограммы (ЭЭГ), позволяющий неинвазивно измерять электрические импульсы в головном мозге. Впервые в медицинских исследованиях он был задействован в 1924 году, предоставив врачам более широкие возможности по сбору и анализу временны*х* рядов, применяемых в индивидуальной диагностике (рис. 1.3).

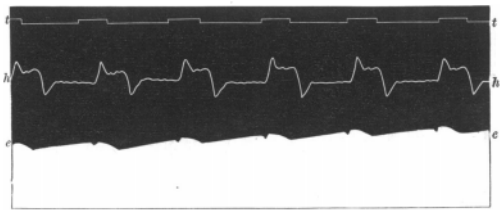


Fig. 1. Man. Heart led off to electrometer from front and back of chest (front to Hg; back to H_2SO_4).
e.e. electrometer. *h.h.* cardiograph. *t.t.* time in seconds.

Рис. 1.2. *Одна из первых ЭКГ, приведенная в оригинальной статье A Demonstration on Man of Electromotive Changes Accompanying the Heart's Beat (Демонстрация человеком электродвижущих изменений, сопровождающих сердцебиение),* `https://perma.cc/ZGB8-3C95` *доктора медицины Августа Д. Уоллера и датируемая 1877 годом. Ранние ЭКГ было очень сложно регистрировать и описывать, поэтому прошло еще несколько десятилетий, прежде чем они стали использоваться в повсеместной врачебной практике*

[1] Примерами могут служить исследования British Doctors Study и Nurses Health Study.

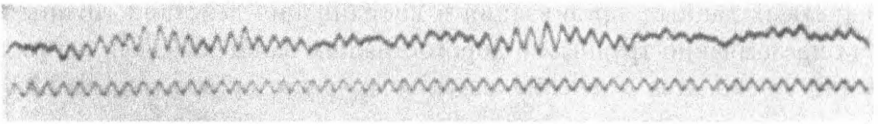


***Рис. 1.3.** Первая ЭЭГ человека (1924 г.).*

Источник: Википедия `https://ru.wikipedia.org/wiki/Электроэнцефалография`

Оба аппарата регистрации временны*х* рядов были частью более обширной программы по совершенствованию сферы медицинского обслуживания населения, основанной на идеях и технологиях, которые появились во время второй промышленной революции.

Классификация временны*х* рядов ЭКГ и ЭЭГ по-прежнему относится к актуальным исследовательскими задачами, поскольку служит очень важной практической цели — оценке риска внезапного сердечного приступа. Такие измерения служат богатым источником данных, но основная трудность их сбора заключается в том, что они, как правило, проводятся только для пациентов с заболеваниями сердца. Аппараты ЭКГ и ЭЭГ не применяются для получения длительных временны*х* рядов, позволяющих судить об общем здоровье и поведении человека, поскольку исследования практически никогда не проводятся в течение длительных периодов ремиссии или до того, как у пациента возникнет заболевание.

К счастью, с точки зрения анализа данных мы переходим в эпоху, когда ЭКГ и подобные им исследования становятся общепринятой практикой сбора информации, представленной временны*ми* рядами. С появлением носимых датчиков и умных медицинских устройств такие измерения стали проводиться многими здоровыми людьми автоматически или с минимальным их участием, что сделало сбор качественных наборов последовательно поступающих данных как от больных, так и от здоровых людей не только возможным, но и непрерывным процессом. Такие данные резко контрастируют с медицинскими временны*ми* рядами прошлого столетия, которые собирались исключительно для людей, имеющих серьезные заболевания, а доступ к ним был сильно ограничен.

Как показывают последние тенденции, сильное влияние на здравоохранение оказывают наиболее влиятельные игроки других отраслей, среди которых социальные сети, финансовые организации и крупные торговые ретейлеры.[2] Вероятнее всего, все они заинтересованы в использовании больших наборов данных для оптимизации сферы здравоохранения угодным им способом. В свою очередь,

[2] См., например, публикации "Darrell Etherington, Amazon, JPMorgan and Berkshire Hathaway to Build Their OwnHealthcare Company" TechCrunch, January 30, 2018 (`https://perma.cc/S789-EQGW`); "Christina Farr, Facebook Sent a Doctor on a Secret Mission to Ask Hospitals to Share Patient Data," CNBC, April 5, 2018 (`https://perma.cc/65GF-M2SJ`).

отрасль здравоохранения пополняется не только новыми игроками, но и новыми технологиями. Персонифицированная медицина, основанная на анализе ДНК, стала возможной в том числе благодаря инструментам сбора и обработки временны*х* рядов. Учитывая возможности современных медицинских технологий и объемы подлежащих обработке данных, в ближайшие годы анализу временны*х* рядов будет уделяться все более и более пристальное внимание, особенно со стороны коммерческих организаций, оказывающих сильное финансовое давление на сферу здравоохранения. Хочется верить, что методы анализа временны*х* рядов сослужат хорошую службу всем без исключения заинтересованным сторонам.

Прогноз погоды

По понятным причинам составление прогнозов погоды стало востребованной задачей еще в древности. Древнегреческий философ Аристотель изучал погоду в отдельном трактате "Метеорология", а его идеи о причинах и закономерностях в изменении погоды оставались доминирующими вплоть до эпохи Возрождения. В эту эпоху ученые только начали собирать данные о погоде с помощью недавно изобретенных приборов, подобных барометру, которые предназначались для регистрации изменений состояния атмосферы. Такие приборы позволяли получать временны*е* ряды значений с ежедневными или даже часовыми интервалами. Записи хранились в разных источниках, преимущественно — в частных дневниках и муниципальных учетных журналах. На протяжении веков такой подход оставался единственным используемым европейской цивилизацией для отслеживания погодных условий.

Отрасль метеорологии подверглась полной реорганизации и документальной формализации только в 1850-х годах с назначением Роберта Фитцроя главой нового правительственного департамента записи и публикации данных о погоде в Королевском адмиралтействе.[3] Именно Фитцрой ввел понятие "прогноз погоды". В то время его нещадно критиковали за низкую точность прогнозов, но сегодня считается, что он намного опередил свое время, поставив науку во главе методов их составления. Он постановил печатать прогнозы погоды в газете — это были первые прогнозы погоды, публикуемые лондонской *The Times*. В наши дни Фитцрой считается родоначальником научного подхода к прогнозированию погоды.

В конце XIX века, спустя сотни лет после того, как регистрация изменений в состоянии атмосферы стала обыденным явлением, был изобретен телеграф, который позволил сопоставлять атмосферные условия, зафиксированные в разных местах земного шара в одно и то же время и представленные временны*ми*

[3]Роберт Фитцрой был капитаном корабля *Beagle*, совершившего кругосветное плавание, в котором принимал участие Чарльз Дарвин. Это путешествие сыграло важную роль в формировании теории эволюции и роли в ней естественного отбора.

рядами данных. К 1870-м годам такой подход стал общепринятым во многих странах, что привело к получению первых значимых наборов данных, позволяющих судить об изменении местной погоды по данным в других географических точках.

На рубеже XIX–XX веков предпринимались неоднократные попытки компилирования полученных ранее наборов данных и составления по ним прогнозов погоды, рассчитываемых предельно точным математическим способом. Такие расчеты выполнялись вручную, требовали колоссальных усилий и давали плохие результаты. В то время физические и химические процессы, происходящие в атмосфере, были изучены наукой достаточно хорошо, но описывались слишком большим количеством законов, которые было очень сложно учитывать все вместе. Результирующая система уравнений была настолько сложной, что сама только попытка приняться за ее решение считалась сродни безумству.

Только спустя несколько десятилетий непрерывных исследований уравнения, описывающие учитываемые в прогнозах погоды физические и химические явления, удалось упростить настолько, что их стало возможно решить с заданными точностью и скоростью. Такой подход в полной мере применяется в современных моделях прогнозирования погодных условий — они основаны на оптимальной комбинации подлежащих учету физических принципов и исторически выверенных эвристических правил.

В настоящее время метеорологические ведомства многих стран проводят точные измерения погодных условий на сотнях или даже тысячах станций, разбросанных по всему миру, а получаемые прогнозы основаны в том числе на данных о точном местонахождении метеостанций и измерительного оборудования. Как бы там ни было, корнями современный подход уходит к скоординированным наборам данных 1870-х годов, а еще раньше — к практике ведения дневников погодных условий в эпоху Возрождения.

К сожалению, недостаточная точность прогнозов становится причиной частой критики научного подхода в отслеживании погодных изменений. Дошло даже до того, что сомнению подвергаются временны*е* ряды данных, по которым строятся прогнозы. Политизированными оказываются не только суждения о правильности получения временны*х* рядов средней температуры, но и предельно точные данные, получаемые с помощью многократно выверенных исследовательских методик, например о направлении движения ураганов (`https://perma.cc/D9GG-FND2`).

Экономические прогнозы

Экономические и финансовые показатели уже давно представляются временны*ми* рядами и являются предметом пристального изучения профильными

специалистами. Во все времена наибольший интерес представляла задача прогнозирования экономических показателей по уже известным значениям. Такие прогнозы позволяют не только оптимизировать прибыль компаний, но и предотвратить экономические и социальные катастрофы. Давайте обсудим наиболее важные события в истории экономического прогнозирования.

Экономическое прогнозирование выросло из беспокойства, вызванного эпизодическими финансовыми кризисами в Соединенных Штатах и Европе в конце XIX и начале XX веков. В то время как практикующие экономисты, так и исследователи были убеждены, что мировая экономика, подобно погоде, подвержена циклическим изменениям. Считалось, что при правильных расчетах можно делать очень точные прогнозы и предотвращать возможные экономические кризисы.

На начальных этапах при построении экономических прогнозов использовалась такая же терминология, как и при построении прогнозов погоды. Это получилось непреднамеренно, но, как ни странно, идея прижилась. В начале XX века построение экономических прогнозов и предсказание погодных условий выполнялись по схожим схемам. Вполне ожидаемо они не отличались низкой точностью получаемых результатов. Так чаяниями экономистов была создано новое, очень перспективное исследовательское направление, давшее толчок к развитию государственных учреждений и частных организаций, занимающихся постоянным мониторингом экономической ситуации. Ранние усилия по экономическому прогнозированию привели к разработке экономических показателей и общедоступных таблиц их временного изменения, успешно используемых по сегодняшний день. Некоторые из них мы даже будем рассматривать в этой книге.

В настоящее время в США и многих других странах сбором и публикацией экономических данных занято несколько тысяч государственных исследовательских организаций и архивов. В их задачу входит максимально точная регистрация исторических экономических данных и их предоставление для широкого доступа (рис. 1.4). Такая практика оказала положительное влияние на экономический рост и позволила предотвратить частые экономические катастрофы, неизбежно возникающие в периоды спада. Более того, общедоступность больших массивов экономических данных существенно улучшила деловую активность, позволив крупным производителям, поставщикам услуг и владельцам малого бизнеса лучше ориентироваться в будущих рыночных условиях. Такая существующая по сей день инфраструктура возникла из предположения о цикличности экономических процессов, которая, как считалось, была главной причиной повторяющихся финансовых кризисов. Ее основным предназначением был анализ исторических временны*х* рядов экономических данных.

Б*о*льшая часть собираемых правительством экономических данных, наиболее часто освещаемых в новостных лентах, отражает общее экономическое благосостояние населения. Например, одним из постоянно отслеживаемых

экономических показателей является количество людей, обратившихся за пособием по безработице. К не менее важным также относятся показатели валового внутреннего продукта и общие налоговые поступления, полученные за определенный год.

BUSINESS CYCLE REFERENCE DATES		DURATION IN MONTHS			
Peak	Trough	Contraction	Expansion	Cycle	
Quarterly dates are in parentheses		*Peak to Trough*	*Previous trough to this peak*	*Trough from Previous Trough*	*Peak from Previous Peak*
	December 1854 (IV)	--	--	--	--
June 1857(II)	December 1858 (IV)	18	30	48	--
October 1860(III)	June 1861 (III)	8	22	30	40
April 1865(I)	December 1867 (I)	32	46	78	54
June 1869(II)	December 1870 (IV)	18	18	36	50
October 1873(III)	March 1879 (I)	65	34	99	52
March 1882(I)	May 1885 (II)	38	36	74	101
March 1887(II)	April 1888 (I)	13	22	35	60
July 1890(III)	May 1891 (II)	10	27	37	40
January 1893(I)	June 1894 (II)	17	20	37	30
December 1895(IV)	June 1897 (II)	18	18	36	35
June 1899(III)	December 1900 (IV)	18	24	42	42
September 1902(IV)	August 1904 (III)	23	21	44	39

Рис. 1.4. *Правительство США финансирует большое количество государственных учреждений и аффилированных некоммерческих организаций, ответственных за сбор демографической статистики и отслеживание экономических показателей*

Источник: National Bureau of Economic Research `https://www.nber.org/cycles/cyclesmain.html`

Благодаря стремлению к экономическому прогнозированию правительства стран получили в свое распоряжение очень много важных сведений о состоянии экономики и повысили эффективность сбора налогов. Накопленная информация часто используется при разработке экономических теорий и финансовых приложений, а также в качестве источника данных во многих других дисциплинах, в частности в науке о данных. Обработка исторических временны́х рядов экономических величин позволила если не предотвратить, то по крайней мере серьезно сгладить очередные финансовые кризисы. Кроме того, временны́е ряды начали подробно рассматриваться в экономических учебниках, настаивающих на цикличности всех происходящих в экономике процессов.

Экономические рынки

Вернемся к нашему историческому экскурсу. Как только усилия правительств по сбору экономических данных увенчались успехом, коммерческие организации

начали активно использовать накопленную информацию в собственных целях. Со временем товарные и фондовые биржи стали переходить на более современное техническое оснащение. Приобрели популярность календари предсказываемых событий. Их издание стало возможным благодаря постоянному совершенствованию инструментов автоматической обработки данных и как следствие лучшей информированности участников рынка. Впервые в истории предположения о ценах начали строить на основе данных, а не интуитивных соображений.

Доступ к подробным историческим записям повысил спрос на построение прогнозов исключительно математическим способом, предполагающим анализ данных статистическими методами (в последнее время с помощью алгоритмов машинного обучения). Первые исследователи выполняли все необходимые математические расчеты вручную, тогда как современные применяют для этих целей очень сложные и запатентованные инструменты анализа временных рядов.

Одним из пионеров *машинной, или алгоритмической, торговли,* в которой анализ временных рядов выполняется с помощью компьютерных алгоритмов, стал Ричард Деннис. Будучи миллионером и человеком, который всего добился сам, Деннис превратил обычных трейдеров, именуемых не иначе как "черепахами", в первоклассных биржевых игроков, обучив их нескольким простым правилам, предельно точно указывающим, как, когда и чем нужно торговать, чтобы добиться успеха. Эти правила были сформулированы в 1970–1980-х годах и отражали точку зрения самых современных эвристических алгоритмов того времени, которые, как известно, считались наиболее перспективными для использования в системах искусственного интеллекта.

Современные алгоритмические трейдеры пользуются несколько видоизмененными правилами, чтобы подстроиться под реалии полностью автоматизированной биржевой торговли. Несмотря на то что они обеспечивают меньшую прибыльность, алгоритмические трейдеры продолжают успешно наращивать доход и оборотный капитал, находясь в постоянном поиске новых инструментов, обеспечивающих преимущество в острой борьбе с конкурентами.

Астрономия

Астрономия издавна опиралась на сведениях об изменении размера, траекторий движения и поведения небесных тел. Во все времена астрономы были самыми лучшими специалистами по анализу временных рядов как того, что касается калибровки приборов, так и изучения наблюдаемых объектов. В качестве примера одного из самых длительных периодов сбора данных астрономических временных рядов можно привести исследования пятен на Солнце, проводимые в Древнем Китае начиная с VIII столетия до нашей эры, что делает их одними

из наиболее хорошо изученных астрономических явлений за всю историю наблюдений.

Стоит заметить, что самые захватывающие астрономические достижения прошлого столетия связаны с анализом временны*х* рядов. Открытие переменных звезд (применяемые современной наукой для определения межгалактических расстояний) и наблюдение за происходящими событиями, такими как появление сверхновых звезд (улучшают наше понимание временны*х* изменений Вселенной), стало результатом непрерывного сбора временны*х* последовательностей данных, состоящих из значений длины и интенсивности электромагнитных волн. Временны*е* ряды оказали фундаментальное влияние на понимание происходящих во Вселенной явлений и возможности по ее исследованию.

Стоит заметить, что такой мониторинг астрономических явлений позволил астрономам регистрировать многие события даже в момент их возникновения (`https://perma.cc/2TNK-2TFW`).

За последние несколько десятилетий количество общедоступных астрономических данных, снабженных временны*ми* метками, резко увеличилось благодаря широкому распространению более совершенных телескопов, автоматически собирающих огромное количество сведений о наблюдаемых космических объектах и явлениях. В некоторых случаях астрономы даже называют временны*е* ряды не иначе как "потоп данных".

Начало анализа временны*х* рядов

Джордж Бокс, один из первых специалистов по статистической обработке данных, принимающий участие в разработке одной из популярных моделей временны*х* рядов и известный своим устойчивым прагматизмом, как-то заметил: "Все модели неправильные, и только некоторые из них полезны".

Бокс сделал это заявление в ответ на распространенное мнение о том, что моделирование временны*х* рядов — это не больше чем подгонка модели под имеющиеся данные. Как он впоследствии объяснил, идея о том, что реальный мир можно описать любой моделью, не имеет под собой никаких оснований. Бокс сделал это заявление в 1978 году, что достаточно поздно для дисциплины анализа временны*х* рядов, которая на тот момент уже существовала, хотя и считалась достаточно молодой.

Стоит заметить, что одно из главных открытий Джорджа Бокса, принесших ему известность (метод Бокса–Дженкинса, считающийся фундаментальным для дисциплины анализа временны*х* рядов), было сделано только в 1970 году.[4] Интересно

[4] Метод Бокса–Дженкинса считается каноническим при подборе наилучших параметров для моделей ARMA и ARIMA, применяемых в прогнозировании временных рядов (см. главу 6).

то, что впервые этот метод был представлен общественности не в научном журнале, а в учебнике по статистической математике.

Первоначальная модель Бокса–Дженкинса применялась к набору значений, указывающих концентрацию углекислого газа, выбрасываемого газовой печью. Хотя газовой печью уже давно никого не удивишь, набор из 300 записей, который использовался в демонстрации метода, сегодня выглядит, мягко говоря, несколько урезанным. Конечно, в 1970-х годах исследователям были доступны намного бо́льшие наборы данных, но не стоит забывать, что в то время не существовало таких языков программирования, как как R, Python и даже C++, и работать с такими последовательностями значений было невероятно сложно. У исследователей были веские причины сосредоточиться на небольших наборах данных и методах, требующих небольших вычислительных мощностей.

Эволюция дисциплин анализа временны́х рядов и прогнозирования начиналась так же, как и компьютеров, из набора простых инструментов, с помощью которым которых нужно было обработать как можно бо́льшие наборы данных. В процессе их развития перед исследователями не только постоянно возникали все более и более сложные вопросы, но и открывались новые, недоступные ранее возможности. В статье *A brief history of time series forecasting competitions*, посвященной истории развития представлений об анализе временны́х рядов, профессором Робом Хайндманом (`https://perma.cc/32LJ-RFJW`) приведено несколько подтверждений того, что технологии прогнозирования развивались синхронно с компьютерными технологиями.

Профессор Хайндман считает, что "самое раннее нетривиальное исследование точности прогноза временны́х рядов" было представлено в докторской диссертации, которая защищалась в 1969 году в Ноттингемском университете — всего за год до публикации метода Бокса-Дженкинса. Оба события можно считать началом эры соперничества в разработке алгоритмов прогнозирования временны́х рядов. Первые соревнования по прогнозированию проводились в начале 1970-х годов всего на ста наборах данных.[5] Весьма неплохо для того времени, но не нужно забывать, что в случае крайней необходимости все вычисления можно было выполнить вручную.

К концу 1970-х годов соревнования проводились над массивом из около 1000 наборов данных — весьма впечатляющее увеличение размера. Заметьте, что та эпоха была также знаменательна разработкой первого коммерческого микропроцессора, накопителя на гибких дисках, первых персональных компьютеров Apple и языка программирования Pascal. Вне всяких сомнений, некоторые из этих нововведений внесли свою лепту в развитие технологий анализа временны́х рядов. Соревнование по прогнозированию временны́х рядов в конце 1990-х годов проводилось на 3000

[5]Точнее, на ста отдельных наборах данных, извлеченных из разных частей разных временны́х рядов разной длины.

наборах данных. Хотя это очень большой массив информации для сбора и анализа, обработка которого требует серьезных усилий и немалой изобретательности, он все еще намного меньше действительных временны*х* рядов, с которыми всем нам приходится иметь дело каждый день. Именно так — временны*е* ряды окружают нас повсюду, и скоро ими можно будет представить все на свете!

Быстрый рост объемов обрабатываемых данных обусловлен прогрессом в развитии вычислительной техники, демонстрируемым в последние несколько десятилетий. Он все еще описывается законом Мура, который гласит, что вычислительная мощность компьютерного оборудования увеличивается во времени по экспоненциальному закону. Наряду с увеличением производительности со временем аппаратное обеспечение становится дешевле и доступнее для использования в самых разных устройствах, начиная с мобильных устройств, "напичканных" всевозможными датчиками, и заканчивая серверами в огромных центрах обработки данных, составляющих ресурсный базис современного Интернета. Появление носимой электроники, алгоритмов машинного обучения и графических процессоров произвело настоящую революцию в технологиях обработки и анализа данных, подлежащих изучению.[6]

Несомненно, дальнейшее увеличение производительности оборудования положительно скажется на скорости и точности обработки временны*х* рядов, поскольку анализ данных требует существенных аппаратных мощностей. Стоит ожидать, что сохранение тенденции к увеличению вычислительной мощности и размера хранилищ информации приведет к заметному повышению скорости обработки данных временны*х* рядов.

Истоки статистического анализа временны*х* рядов

Статистика — очень молодая наука. Ее развитие и эволюция методов анализа временны*х* рядов всегда сильно зависели от доступности и объема обрабатываемых данных. Появление анализа временны*х* рядов как отдельной дисциплины неотрывно связано не только с развитием теории вероятностей, но и с появлением устойчивых национальных государств, в которых ведение всестороннего учета стало достижимой необходимостью. Ранее мы уже останавливались на рассмотрении этого утверждения в контексте смежных дисциплин. Сейчас же давайте поговорим о временны*х* рядах как об отдельной дисциплине.

[6] Учитывая то количество мобильных устройств, которые все мы носим с собой каждый день, а также количество создаваемых ими временны*х* меток (при совершении покупок, регистрации в беспроводных сетях, посещении сайтов, совершении звонков по телефону или просто перемещению по городу согласно указаниям GPS-навигатор) можно с уверенностью утверждать, что среднестатистический американец генерирует несколько тысяч временны*х* точек данных в год.

Одним из первых признаков возникновения анализа временны*х* рядов как предметной области является применение моделей авторегрессии к реальным данным. Впервые эта задача была выполнена в 1920-х годах. Физик-экспериментатор Удни Юл, лектор по статистической математике Кембриджского университета, применил авторегрессионную модель для описания данных о солнечных пятнах, предлагая новый способ представления данных в противоположность используемому в ранних моделях, где основной акцент делался на учете частоты их пульсаций. Юл отметил, что авторегрессионная модель не согласуется с моделью, предполагающей периодичность изменения размера пятен.

> Если целью спектрального анализа, выполняемого по данным, которые описывают определенное физическое явление, является нахождение одной или нескольких периодичных зависимостей, то большинство исследователей, как мне кажется, заведомо исходят из предположения о том, что такие зависимости скрываются за случайными флуктуациями, которые никоим образом не влияют на истинные данные описывающих их функций... Я убежден, что такое предположение нужно рассматривать исключительно как одну из возможных гипотез, а не принимаемую по умолчанию истину.

Идея Юла о том, что сама модель предопределяет результат, была оригинальной, но, скорее всего, возникла под сторонним влиянием. Будучи физиком-экспериментатором, которому в прошлом довелось поработать в Германии (оплоте сторонников квантовой теории), Юл наверняка знал о последних разработках, доказывающих вероятностный характер квантовых процессов. Он также признавал опасность выдвижения слишком большого количества предположений о модели до ее появления, как это делалось в классической физике до возникновения квантовой механики.

Упорядоченность, формализация и предсказуемость современного мира, воочию наблюдаемые после второй мировой войны, обеспечиваются в том числе решением большого количества практических задач по анализу временны*х* рядов. Экономические задачи, требующие анализа временны*х* рядов, имеют первостепенную важность, хотя и не представляют большого научного интереса. Среди них стоит выделить прогнозирование спроса, оценку стоимости сырьевых ресурсов и хеджирование производственных затрат. В производственном секторе методы прогнозирования считались работоспособными, если давали справедливый результат, и отвергались, если результат оказывался неудовлетворительным. А так как производственный сектор получает доступ к большим наборам данных раньше, чем научные организации, зачастую многие теоретические наработки и недостаточно хорошо исследованные методы находят практическое применение задолго до полного изучения.

Анализ временных рядов с помощью машинного обучения

С момента первого применения методов машинного обучения для анализа временных рядов произошло ни много, ни мало — несколько десятилетий. В часто цитируемой статье 1969 года *The Combination of Forecasts* рассматривается принцип объединения прогнозов в противоположность выбору единственно возможного — наиболее эффективного с точки зрения получаемого результата. Традиционные статистические методы полностью отвергают такой подход в отличие от ансамблевых методов, которые стали золотым стандартом решения в самых разных задачах прогнозирования. Ансамбль не предполагает превосходство или просто выделение одной модели прогнозирования из всех остальных моделей.

Если быть точными, то методы анализа временных рядов нашли практическое применение в технологии машинного обучения еще в 1980-х годах и использовались в широком спектре сценариев.

- Специалисты по компьютерной безопасности предложили задействовать их для обнаружения аномалий при выявлении взломов и вмешательств.
- Динамическое изменение масштаба времени — один из доминирующих методов оценки схожести временных рядов — получил распространение с появлением высокопроизводительного вычислительного оборудования, позволяющего достаточно быстро вычислить "временное расстояние", например, между различными аудиозаписями.
- Были изобретены рекурсивные нейронные сети, которые показали свою полезность для извлечения шаблонов из поврежденных данных.

На сегодняшний день методы анализа и прогнозирования временных рядов далеки от совершенства — они все еще основаны на традиционных статистических принципах и простых алгоритмах машинного обучения, таких как ансамбли деревьев и линейная аппроксимация. Научные прорывы в предсказании будущего все еще впереди!

Дополнительные источники

- Общая история развития методов анализа и прогнозирования временных рядов

  Kenneth F. Wallis, "Revisiting Francis Galton's Forecasting Competition," Statistical Science 29, no. 3 (2014): 420–24, `https://perma.cc/FJ6V-8HUY`

  Историческое описание статистического метода оценки веса убитого быка по его живому весу, измеренному по прибытии на уездную ярмарку.

G. Udny Yule, "On a Method of Investigating Periodicities in Disturbed Series, with Special Reference to Wolfer's Sunspot Numbers," Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character 226 (1927): 267–98, https://perma.cc/D6SL-7UZS

Оригинальная статья Удни Юла, рассматривающая одну из первых попыток практического применения метода авторегрессионного скользящего среднего. Описывает способ исключения предположения о периодичности изменений из модели, применяемой для анализа предположительно периодического явления.

J.M. Bates and C. W. J. Granger, "The Combination of Forecasts," Organizational Research Quarterly 20, No. 4 (1969): 451–68, https://perma.cc/9AEE-QZ2J

Оригинальный документ, описывающий использование ансамбля для прогнозирования временных рядов. Утверждение о том, что усреднение моделей позволяет добиться лучшего прогноза, чем поиск идеальной модели, было новым и неприемлемым для многих специалистов по статистике, придерживающихся традиционных взглядов.

Jan De Gooijer and Rob Hyndman, "25 Years of Time Series Forecasting," International Journal of Forecasting 22, no. 3 (2006): 443–73, https://perma.cc/ 84RG-58BU

Подробный статистический отчет об исследованиях по прогнозированию временных рядов в XX веке.

Rob Hyndman, "A Brief History of Time Series Forecasting Competitions," Hyndsight blog, April 11, 2018, https://perma.cc/32LJ-RFJW

Сжатый отчет о соревнованиях по прогнозированию временных рядов за последние 50 лет, в котором указаны места проведения, имена участников и полученные ими результаты.

- История развития методов анализа временных рядов в отдельных предметных областях

NASA, "Weather Forecasting Through the Ages," Nasa.gov, February 22, 2002, https://perma.cc/8GK5-JAVT

История появления прогнозов погоды от NASA. Особое внимание уделяется целевым задачам и исследовательским успехам в XX веке.

Richard C. Cornes, "Early Meteorological Data from London and Paris: Extending the North Atlantic Oscillation Series," PhD diss., School of Environmental Sciences, University of East Anglia, Norwich, UK, May 2010, https://perma.cc/NJ33-WVXH

Докторская диссертация, содержащая увлекательный отчет об изменении погодных условий в двух наиболее важных городах Европы.

Включает обширный список мест измерения и данные о характере погоды в прошлом, представленные в формате временны*х* рядов.

Dan Mayer, "A Brief History of Medicine and Statistics," in Essential Evidence-Based Medicine (Cambridge, UK: Cambridge University Press, 2004), `https://perma.cc/WKU3-9SUX`

В этой главе книги Майера показано, насколько сильно социальные и политические факторы влияют на связь между медициной и статистикой, что необходимо учитывать при обучении будущих врачей, которым по долгу службы приходится работать со статистическими данными.

Simon Vaughan, "Random Time Series in Astronomy", Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 371, no. 1984 (2013): 1–28, `https://perma.cc/J3VS-6JYB`

Воган оценивает применимость самых разных принципов анализа и прогнозирования временны*х* рядов в астрономии и предупреждает об опасности повторного изобретения их астрономами или отказа от использования в них традиционных статистических методов.

ГЛАВА 2

Распознавание и обработка временны*х* рядов

В этой главе мы обсудим проблемы, которые могут возникнуть при предварительной обработке временны*х* рядов. Некоторые из таких проблем хорошо знакомы опытным аналитикам, но к ним не относятся трудности выбора временны*х* меток. Как и в любой другой задаче анализа данных, очистка и предварительная обработка данных являются залогом правильной расстановки временны*х* меток. Каким бы хорошим ни был метод, он не будет работать с неупорядоченными данными.

Большинству аналитиков приходится либо находить, выравнивать, очищать и сглаживать собственные данные для последующего анализа в виде временны*х* рядов, либо заняться решением совершенно иных задач. При подготовке данных приходится решать множество вопросов: от объединения разрозненных столбцов и повторной выборки нерегулярных или отсутствующих данных до выравнивания временны*х* рядов с разными временны*ми* осями. Эта глава поможет вам создать содержательный и правильно подготовленный набор временны*х* рядов.

В ней рассматриваются следующие операции по поиску и очистке временны*х* рядов.

- Поиск временны*х* рядов в онлайн-хранилищах
- Обнаружение и извлечение временны*х* рядов из источников, изначально не предназначенных для хранения временны*х* рядов
- Решение типичных проблем обработки временны*х* рядов, в частности трудностей расстановки временны*х* меток

Изучив эту главу, вы научитесь обнаруживать и подготавливать источники временны*х* рядов к последующему анализу.

Источники временны*х* рядов

Ответ на вопрос, где можно найти временной ряд и как его очистить от лишних данных, зависит от преследуемой цели.

- Поиск подходящего набора данных для обучения или тестирования
- Создание набора временны*х* рядов из существующих данных, которые не имеют явно выраженной временно*й* ориентации

В первом случае уже собранные кем-то наборы данных с известными контрольными показателями используются для проверки правильности проведения анализа. Чаще всего они представлены конкурсными наборами данных (такими, как Kaggle) или наборами данных, хранящихся в репозиториях. Получив такой набор, вам требуется дополнительно подготовить его — очистить для собственных целей даже несмотря на то, что тот прошел некую предварительную обработку.

Во втором случае для получения информативных временны*х* рядов вам нужно распознать в имеющемся наборе данных значения, снабженные временны*ми* метками, преобразовать их в ряды, очистить и выровнять с другими данными, имеющими временны*е* метки. Я буду называть данные, извлеченные из таких источников, *полученными временными рядами* (это мой собственный, а не технический термин).

Далее мы обсудим как подготовленные наборы данных, так и полученные временны*е* ряды.

Подготовленные наборы данных

Лучший способ освоить аналитические методы или приемы моделирования — применить их к разным наборам данных и посмотреть, к каким результатам это приведет и как их можно использовать в собственных целях. Проще всего проводить такое обучение на заранее подготовленных наборах данных.

Несмотря на то что временны*е* ряды встречаются повсюду, найти необходимые данные далеко не всегда так просто, как того хотелось бы. Если вы часто оказываетесь в таком положении, обратитесь за помощью к распространенным репозиториям временны*х* рядов. Далее мы рассмотрим несколько наиболее примечательных вариантов.

Репозиторий UCI Machine Learning

Репозиторий *UCI Machine Learning* (`https://perma.cc/M3XC-M9HU`) (рис. 2.1) содержит около 80 временны*х* рядов, начиная с ежечасных показателей качества воздуха в итальянском городе и журналов доступа к файлам Amazon и заканчивая записями об активности, рационе питания и уровне сахара в крови больных диабетом. Это очень разнотипные сведения — просмотр файлов показывает, что они отражают различные способы временно*й* регистрации данных, но все они представлены временны*ми* рядами.

Рис. 2.1. *В репозитории UCI Machine Learning хранятся временные ряды, снабженные подробным описанием*

Рассмотрим первый набор данных (`https://perma.cc/8E7D-ESGM`), относящийся к категории Time Series (Временные ряды) репозитория UCI Machine Learning, в котором приведены сведения о прогулах на рабочем месте (рис. 2.2).

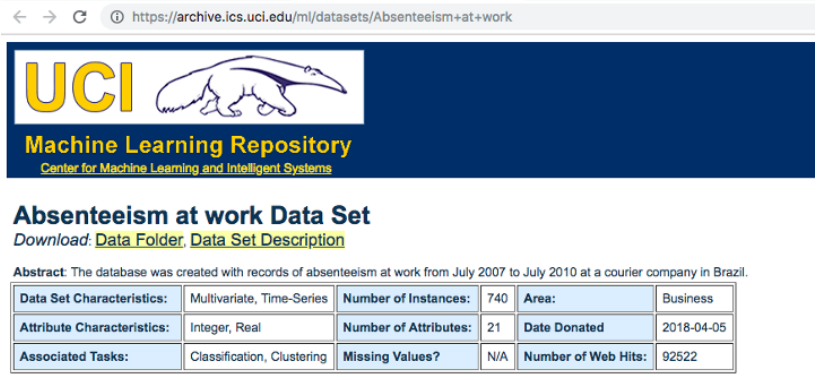


Рис. 2.2. *Данные об отсутствии сотрудников на рабочем месте — первый из хранящихся в репозитории UCI Machine Learning временных рядов*

Беглый просмотр данных показывает, что в наборе данных присутствуют такие столбцы с временны́ми данными, как Month of absence (Месяц отсутствия), Day of the week (День отсуствия) и Seasons (Времена года), но нет столбца, указывающего год события. Кроме того, в отдельном столбце указываются имена сотрудников с продублированными в некоторых временны́х индексах показателями. Наконец, в нескольких дополнительных столбцах приводятся важные сведения о сотрудниках.

Набор данных может показаться довольно сложным для обработки, поскольку сначала нам необходимо определить, все ли данные были получены за один год или они собирались несколько раз с 1 по 12 месяцы в течение нескольких лет. Необходимо также решить, как должно подсчитываться общее количество прогулов — по временны́м индексам или идентификаторам сотрудников, перечисленных в наборе данных. В первом случае мы получим всего один временной ряд, тогда как во втором случае у нас будет сразу несколько временны́х рядов с перекрывающимися временны́ми метками. Способ представления данных зависит от вопроса, ответ на который мы хотим получить.

Сравните набор данных о прогулах на рабочем месте с другим набором данных, приведенным в начале списка Time Series, — о знаках австралийского языка жестов (`https://perma.cc/TC5E-Z6H4`), полученных и описанных с помощью контроллера Nintendo PowerGlove (рис. 2.3). Данные хранятся в виде набора больших CSV-файлов, каждый из которых помещается в отдельную папку, название которой отражает имя человека, демонстрирующего описываемый файлом знак.

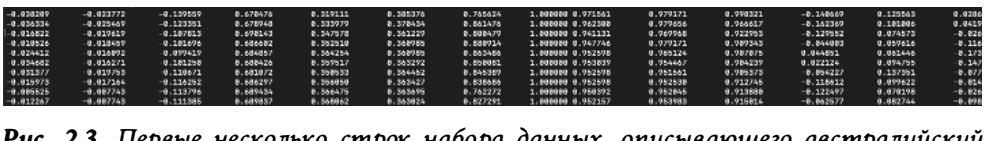


***Рис. 2.3.** Первые несколько строк набора данных, описывающего австралийский язык жестов. Как видите, набор включает огромное количество значений*

В этом наборе данных столбцы не имеют названий, а их значения не содержат временны́х меток. Тем не менее он представляет собой временной ряд, поскольку на оси отмечены временны́е шаги, совершаемые только в одном направлении (вперед), независимо от точного времени совершения событий. Заметьте, что здесь распознавание временны́х рядов не требует использования единиц измерения времени; достаточно выделить общую последовательность, а не точные временны́е индексы. В этом случае все, что вас интересует, — это порядок следования событий и подтверждение того, что измерения проводились через регулярные промежутки времени, о чем можно узнать из описания данных.

Проводя анализ обоих наборов данных, мы успели заметить, что их обработка сопряжена с самыми разными трудностями. Перечислим наиболее важные проблемы, с которые нам довелось встречаться.

- Неполный набор временных меток
- Временная ось может быть направлена горизонтально или вертикально
- Различные концепции представления времени

Репозиторий UEA and UCR Time Series Classification

Хранилище *UEA and UCR Time Series Classification* — это относительно новый проект, предоставляющий на ваше суждение стандартные наборы временных рядов, обычно применяемых при изучении методов и алгоритмов классификации данных. Репозиторий также включает сильно различающиеся наборы данных. В этом легко убедиться на примере всего двух из них.

Первый рассматриваемый нами набор данных относится к задаче классификации движений в йоге (`https://perma.cc/U6MU-2SCZ`). Задача классификации заключается в распознавании различий между двумя сериями фотографий, отснятых при смене поз йоги, принимаемых разными актерами. Впоследствии все полученные снимки преобразуются в одномерные ряды данных. Данные сохраняются в файле формата CSV, в котором крайний слева столбец отводится для подписей данных, а в остальных столбцах представлены отдельные временные шаги. Ось времени направлена слева направо вдоль строк, а не сверху вниз — вдоль строк.

Графики двух произвольно выбранных временных рядов для актеров каждого пола показаны на рис. 2.4.

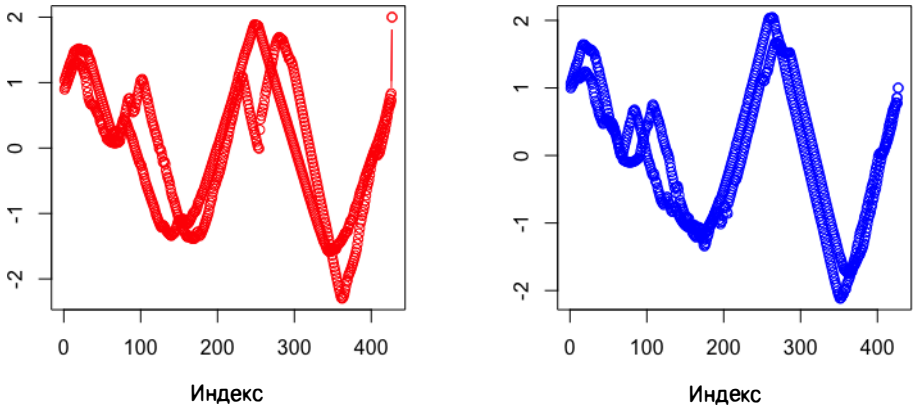


Рис. 2.4. *Повторяющиеся графики, описывающие движения при смене поз в йоге актерами мужского и женского полов. Движения каждого актера представлены двумя временными рядами. На оси* ***x*** *явные временные метки не указаны. Их абсолютные значения не столь важны, как равномерность размещения точек данных вдоль оси* ***x****, как в данном случае*

Одномерный и многомерный временные ряды

Рассматриваемые до сих пор наборы данных представлялись *одномерными* временными рядами, т.е. характеризовались всего одной изменяемой во времени переменной.

Многомерные временные ряды включают несколько таких переменных, измеренных в каждой временной метке. Они особенно богаты информацией, потому что часто переменные взаимосвязаны и демонстрируют временные зависимости друг от друга. Мы познакомимся с многомерными временными рядами немного позже.

Следующим рассмотрим набор данных (`https://perma.cc/CJ7A-SXFD`), в котором сорта вин классифицированы по регионам в соответствии с формой их спектрограммы. Что же делает такой набор данных пригодным для анализа методами временных рядов? В нашем случае спектрограмма представляет собой график зависимости интенсивности поглощения от длины световых волн. Как видите, задача классификации временных рядов не предполагает рассмотрение временного измерения. Однако метод анализа временных рядов применим и здесь, поскольку значения на оси *x* уникальны, равномерно распределены и заданы через строго заданные интервалы. Заметьте, что в этом методе, в отличие от метода перекрестного анализа, учитывается дополнительный фактор, определяющий способ упорядочивания значений вдоль оси *x*, например время, длина волны и т.д. Пример графика "временного" ряда, в котором временное измерение отсутствует, приведен на рис. 2.5. Как видите, оси времени на нем действительно нет, но набор данных выглядит более чем упорядоченным, поэтому к нему можно смело применять стандартные методы обработки временных рядов.

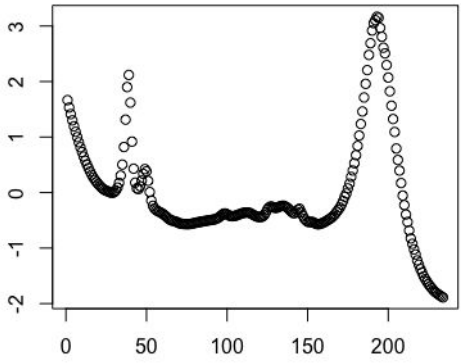


Рис. 2.5. *Образец спектра вина, взятый из набора данных. Пики на кривой указывают длины волн, характеризующихся особенно большими уровнями поглощения. Длины волн равномерно задаются вдоль оси* ***x****, тогда как ось* ***y****, также имеющая линейный масштаб, определяет уровень поглощения. Для сравнения таких графиков подходят любые методы анализа временных рядов*

Наборы временных рядов, предоставляемые правительственными организациями

Правительство США собирает данные, представленные временными рядами, на протяжении десятилетий, а в отдельных случаях даже столетий. Например, Национальные центры информации об окружающей среде (National Centers for Environmental Information — NOAA) (`https://perma.cc/EA5R-TP5L`) регулярно публикуют данные о температуре и осадках, собираемые с интервалом 15 минут со всех метеостанций США. Статистическое управление (Bureau of Labor Statistics) (`https://www.bls.gov/`) предоставляет ежемесячные сведения об изменении уровня безработицы во всех регионах страны. Центры по контролю и профилактике заболеваний (Centers for Disease Control and Prevention) (`https://perma.cc/Y6KG-T948`) издают еженедельные отчеты о количестве переболевших гриппом в текущем сезоне. Федеральный резервный банк Сент-Луиса (`https://fred.stlouisfed.org`) предоставляет в открытый доступ необычайно полезный набор экономических временных рядов.

Обращаясь к данным правительственных организаций в процессе обучения методам анализа временных рядов, лучше всего использовать их только для знакомства и построения информативных графиков. Детальное изучение таких наборов данных может вызывать серьезные трудности, как как они описывают чрезвычайно сложные задачи. Например, многие экономисты посвящают всю свою карьеру разработке моделей предсказания уровня безработицы, но получаемые ими результаты очень часто расходятся с официально опубликованными.

Для правительственных организаций составление точных прогнозов, которые сводятся к решению сложных вычислительных задач, является не только социально выгодным, но и жизненно необходимым занятием. К решению таких задач привлекаются многие высококвалифицированные исследователи, но пока что им не удалось достичь действительно значимых результатов. Работать над непростыми задачами очень увлекательно, но обучаться на них невероятно сложно.

Временные ряды в данных, предоставляемых правительственными организациями

На государственных сайтах, подобных упомянутым выше, нетрудно найти превеликое множество актуальных и многообещающих наборов данных. Собираемые правительственными организациями данные могут послужить прекрасным источником необычайно интересных временных рядов. Например, используя их, можно построить параллельные временные ряды с экономическими и климатическими показателями или отследить взаимосвязь между различного рода правонарушениями и структурой государственных расходов. Для выполнения таких

задач вам придется научиться извлекать и комбинировать данные из нескольких совершенно разных наборов.

По целому ряду причин обращаться к временным рядам, полученным с сайтов правительственных организаций, следует крайне предусмотрительно. Принятые в них условные обозначения, названия столбцов и даже их назначение часто изменяются без предупреждения и специального оглашения. Решения о запуске и завершении правительственных проектов часто принимаются спонтанно под влиянием политических, финансовых и других не менее непредсказуемых факторов. Кроме того, данные на правительственных сайтах могут представляться в самых неожиданных форматах, а сами наборы, в отличие от предоставляемых коммерческим сектором, не отличаются хорошей структурой и последовательностью. Все это сильно усложняет дальнейшее использование предоставляемых в широкий доступ временных рядов.

Дополнительные источники данных

Формат книги не позволяет подробно описать все основные источники временных рядов, но вам обязательно нужно познакомиться с приведенными далее репозиториями.

CompEngine
`https://comp-engine.org/`

Эта самоорганизующееся хранилище данных содержит более 25 тысяч баз временных рядов, которые насчитывают почти 140 миллионов отдельных точек данных. Назначение этого хранилища данных и связанного с ним программного обеспечения, снабженного отдельным веб-интерфейсом, состоит в организации и сопровождении системы сравнительного анализа временных рядов (highly comparative time-series analysis — hctsa). Такой анализ проводится с целью выявления в исходных наборах данных всех возможных типов временных зависимостей, специфичных для указанной предметной области.

Пакеты Mcomp и M4comp2018 языка R
`https://cran.r-project.org/web/packages/Mcomp/index.html`
`https://github.com/carlanetto/M4comp2018`

Эти пакеты языка R включают наборы данных, предоставляемые участникам конкурса *M*, проводимого в 1982 году (1001 временной ряд), а также конкурсов *M3* (3003 временных ряда) и *M4* (100 000 временных рядов), проводимых соответственно в 2000 и 2018 годах. Упомянутые выше соревнования по анализу временных рядов уже рассматривались нами в главе 1 в контексте попыток профессора Роба Хайндмана получить точные методы прогнозирования временных рядов. Дополнительные наборы данных, предоставляемые участникам в других соревнованиях по прогнозированию временных

рядов, включены в пакет tscompdata (https://github.com/robjhyndman/tscompdata). Наконец, более специализированные наборы временных рядов можно найти во вспомогательных пакетах языка R, доступных для загрузки из раздела Time Series Data (Данные временных рядов) репозитория *CRAN* (https//perma.cc/2694-D79K).

Полученные временные ряды

Концепция полученного временного ряда рассматривалась нами в начале главы. Она заключается том, что такой временной ряд извлекается из самых разных источников данных. Если быть предельно точным, то такой временной ряд строится из отдельных точек данных, достаточных для его получения, без каких-либо принимаемых в расчет допущений. Ярким примером является объединение временных рядов с данными о транзакциях некого клиента, извлекаемых из общей реляционной базы данных транзакций компании. В таком случае временной ряд может быть построен только потому, что в базе данных хранятся временные значения, которые можно использовать в качестве временных меток или их аналогов. Из исходного набора данных всегда можно извлечь совершенно иные временные ряды, например временной ряд общего для компании дневного объема транзакций или общих недельных поступлений от клиентов женского пола. Более того, подобным образом можно получить временной ряд нескольких переменных, например представляющий сведения об общих недельных поступлениях от всех клиентов в возрасте до 18 лет, общих недельных поступлений от клиентов женского пола старше 65 лет и общих недельных расходах на рекламу. В таком временном ряду на каждом временном шаге указываются сразу три индикатора, т.е. он представляет данные трех переменных.

Поиск временного ряда в структурированных данных, который в явном виде в них не содержится, выполняется очень просто, если такие данные снабжены временными метками. Ниже приведено несколько примеров задания временных меток в стандартных базах данных.

Временные метки событий

Если в данных присутствуют временные метки, то есть все основания утверждать, что на их основе можно создать временной ряд. Вы получите временной ряд даже в случае регистрации одного только времени доступа к рабочим файлам без указания любой другой информации. В этом случае вы можете сопоставить каждую из временных меток с ее разностью с более поздней временной меткой так, что результирующий временной ряд будет состоять из значений времени на оси x и разности между временными метками на оси y. Вы можете пойти дальше, вычисляя среднее значения таких разностей, суммируя их за длительные периоды или просто регистрируя для других целей.

Вневременные измерения, в которых время заменяется другим измерением

В некоторых случаях время не задается явным образом, но учитывается структурой набора данных. Например, данные всегда можно рассматривать как последовательность значений, отстоящих друг от друга на определенном расстоянии, которое задается экспериментально выверенным параметром, таким как смещение датчика, движущегося с известной скоростью. Если вам удастся сопоставить одну из имеющихся переменных времени, то вы получите самый настоящий временной ряд. С другой стороны, если значения одной из осей выражаются через известные соотношения с расстоянием и другой характеристикой, определяющей порядок размещения данных на другой оси (например, как в случае длин волн на рассматриваемых ранее спектрограммах сортов вин), то в вашем распоряжении находятся полноценные временные ряды.

Физические данные

Сбором физических показателей занимаются во многих научных дисциплинах, будь то медицина, сурдология или метеорология. Но если раньше такие данные представлялись вручную собранными записями физических сигналов, регистрируемых с помощью аналоговых устройств, то в настоящее время они сразу сохраняются в цифровом формате. Данные современных временных рядов могут храниться в совершенно неочевидных и даже необычных форматах, например в виде графических файлов или векторов одного из полей базы данных.

Получение набора временных рядов из таблиц

Наиболее типичный пример полученного временного ряда — это ряд, извлеченный из набора данных о состояниях и событиях, хранящегося в реляционной базе данных. Такой способ получения временных рядов также является наиболее востребованным, поскольку в традиционных реляционных базах данных по-прежнему хранится невероятно большое количество важной информации.

Представьте, что вы состоите в крупной благотворительной организации. Перед вами стоит задача определить факторы, которые можно проанализировать с помощью временных рядов.

- Реакция держателей электронных почтовых ящиков на получение новых писем с течением времени. Просматривались письма или нет?
- История членства. Были ли в истории организации периоды, когда ее ряды покидали отдельные члены?
- История платежей. Можно ли спрогнозировать время внесения добровольных взносов определенным членом организации?

Вы можете проанализировать имеющиеся данные, рассмотрев их с позиции временны*х* рядов.

- Сгенерируйте двумерную гистограмму, отображающую количество просмотров электронных писем в зависимости от времени, отдельно для каждого члена организации, чтобы понять, появляется ли у кого-то из них усталость от работы с электронной почтой. (Подробно о роли двумерных гистограмм в анализе временны*х* рядов рассказано в главе 3.)
- Проверьте, прописаны *ли* для членов вашей организации стандартные поведенческие шаблоны (правила), которым они должны следовать в ключевых ситуациях. Например, существует *ли* правило, указывающее, при каких обстоятельствах (проступках) член организации должен покинуть ее ряды (возможно, не ответил на три письма подряд)? В ходе анализа временны*х* рядов такую задачу можно решить, установив связь между статусом членства в организации и внешними действиями. (К обсуждению методов анализа временны*х* рядов в пространстве состояний мы вернемся в главе 7.)
- Задачу прогнозирования платежей легко можно свести к задаче прогнозирования временны*х* рядов. (О классическом статистическом подходе к прогнозированию мы поговорим в главе 4.)

Как видите, самая обычная реляционная база данных может стать источником большого количества задач, решаемых с помощью временны*х* рядов. Чаще всего разработка структуры таких баз данных ведется без учета того, что они когда-нибудь смогут использоваться в качестве источников временны*х* рядов. В подобных случаях временны*е* ряды приходится извлекать из отдельных таблиц и других разрозненных источников.

Учебная задача: получение временны*х* рядов

Если вам посчастливилось заполучить несколько связанных источников данных, то выровняйте их, учитывая возможные различия в уровнях масштабирования значений и устранив несоответствия во временны*х* метках. В качестве наглядного примера давайте создадим несколько наборов данных для упомянутой выше некоммерческой организации. Предположим, в вашем распоряжении находятся данные, приведенные в табл. 2.1–2.3.

Таблица 2.1. Год вступления в организацию и текущий статус членства

Идентификатор	Год вступления в организацию	Статус
1	2017	Золотой
2	2018	Серебряный
3	2016	Недействительный

Таблица 2.2. Недельное количество писем, отправленных члену организации, которые были им открыты

Идентификатор	Неделя	Количество открытых писем
2	2017-01-08	3
2	2018-01-15	2
1	2016-01-15	1

Таблица 2.3. Данные о добровольных взносах

Идентификатор	Метка времени	Сумма взноса, долл.
2	2017-05-22 11:27:49	1000
2	2017-04-13 09:19:02	350
1	2018-01-01 00:15:45	25

Работать с данными в простой табличной форме вам придется чаще всего. Имея такие данные, вы можете ответить на многие вопросы, например, о том, как количество открытых членом организации писем соотносится с общей суммой добровольных взносов.

Вы также можете ответить на вопросы временного характера, например внесет ли пожертвование участник вскоре после вступления в организацию или через какое-то время. Однако, не переводя эти данные в более удобный для временных рядов формат, вы не сможете распознать однозначно поведение, которое поможет в предсказании того, когда именно будет сделан взнос (скажем, на основании того, просматривались последние письма или нет).

Для анализа данных методами временных рядов нужно представить их в правильном формате. И тут мы сталкиваемся с рядом проблем, требующих обязательного решения.

Сначала в имеющихся данных необходимо определить временные оси. В предыдущих таблицах обозначены три уровня временного масштабирования.

- Годовой статус участника.
- Еженедельный подсчет открытых писем.
- Точные временные метки совершения добровольных взносов.

Кроме того, нужно удостовериться, что данные означают именно то, что, по вашему мнению, должны означать. Допустим, что вы хотите удостовериться, что текущий статус определяется его годовым, а не другим, более краткосрочным статусом. Один из способов ответить на этот вопрос — проверить, есть ли у членов организации более одной записи.

```
## Python
>>> YearJoined.groupby('memberId').count().
                    groupby('memberStats').count()
```

```
1000
```

Здесь мы видим, что все 1000 членов имеют только один статус, поэтому для всех них показатель `YearJoined`, скорее всего, будет указывать год вступления в организацию, а по нему будет определяться текущий статус или статус на момент вступления в организацию. Такое подробное изучение данных крайне важно для понимания области использования переменной статуса — планируя поддавать данные более глубокому анализу, необходимо внимательно исследовать их вместе с тем, кто досконально разбирается в их структуре. Если бы текущий статус члена организации применялся для анализа прошлых данных, то это считалось бы *упреждением*, поскольку вы вводили бы дополнительные элементы в модель временного ряда, которые на тот момент еще не были известны. Вот почему нельзя использовать переменную статуса, такую как `YearJoined`, в модели данных, не зная точно, когда именно она была определена.

Что такое упреждение

Термин *упреждение* используется в анализе временных рядов для обозначения любых знаний о будущем. Вы не должны располагать такими знаниями при разработке, обучении или оценке модели. Упреждение — это способ узнать из данных что-то о будущем раньше, чем полагается.

Упреждением называется принцип, согласно которому информация о том, что произойдет в будущем, может распространиться во времени и повлиять на поведение модели в прошлом. Например, выбирая гиперпараметры для модели, вы можете неоднократно протестировать ее на своем наборе данных, затем выбрать лучшую модель и начать анализ с самого начала, чтобы проверить и эту модель. Но такой подход будет неправильным, потому что вы выбрали модель в тот момент времени, когда уже знали, что произойдет в следующий момент времени, а это самое настоящее упреждение.

К сожалению, не существует автоматизированного метода или статистического теста проверки данных на упреждение, поэтому вы должны предельно внимательно изучать их перед построением модели и дальнейшим анализом.

Если посмотреть на таблицу учета электронных писем, то легко заметить, что данные столбца `week` (неделя) могут указываться либо за недельные периоды, либо на моменты времени, определенные недельными временными метками. При этом совершенно понятно, что они должны отображать информацию, собранную на протяжении недельных интервалов времени, поэтому мы должны

рассматривать указанные временны*е* значения как недельные периоды, а не как временны*е* метки, отстоящие одна от другой на недельное расстояние.

Далее нам предстоит оценить некоторые важные характеристики данных. Например, крайне важно понимать, как формируется недельный отчет. Хотя у нас нет планов по реструктуризации таблицы, если в рассматриваемой предметной области неделя интерпретируется необычным образом, то это нужно обязательно учитывать при изучении данных. В процессе анализа человеческой деятельности календарная неделя обычно начинается с воскресенья и заканчивается в субботу или длится с понедельника по воскресенье. Все остальные варианты календарных схем, например такие, в которых рабочая неделя начинается 1 января, считаются нестандартными.

Вы также можете спросить, существуют ли пустые недели? Иначе говоря, есть ли в таблице недели, в которые участник не открыл ни одного электронного письма? Это важно знать при построении модели данных, основанной на временны*х* зависимостях. Пустые недели должны обязательно включаться и учитываться в модели, поскольку они устанавливают такие же точки данных, как недели со значащими данными.

```
## Python
>>> emails[emails.EmailsOpened < 1]

Empty DataFrame
Columns: [EmailsOpened, member, week]
Index: []
```

Здесь существуют две возможности: либо нулевые значения не регистрируются вовсе, либо члены организации открывают в течение недели хотя бы одно электронное письмо. Любой, кто работал с электронной почтой, знает, что заставить людей открывать электронные письма сложно, поэтому гипотеза о том, что члены организации всегда открывают не менее одного письма в неделю, весьма несостоятельна. Решить проблему можно, просмотрев историю работы с электронной почтой одного пользователя.

```
## Python
>>> emails[emails.member == 998]
       EmailsOpened member week
25464   1           998 2017-12-04
25465   3           998 2017-12-11
25466   3           998 2017-12-18
25467   3           998 2018-01-01
25468   3           998 2018-01-08
25469   2           998 2018-01-15
25470   3           998 2018-01-22
25471   2           998 2018-01-29
25472   3           998 2018-02-05
```

```
25473   3           998 2018-02-12
25474   3           998 2018-02-19
25475   2           998 2018-02-26
25476   2           998 2018-03-05
```

Как видите, отдельные недели не указаны в списке. После 18 декабря 2017 года до конца года событий, связанных с электронной почтой, не зарегистрировано.

В этом можно удостовериться, подсчитав количество еженедельных наблюдений между первым и последним событиями для рассматриваемого члена организации. Сначала рассчитаем срок его членства в неделях.

```
## Python
>>> (max(emails[emails.member == 998].week) -
                        min(emails[emails.member == 998].week)).days/7
25.0
```

Затем для этого члена организации нужно определить количество недель, для которых заданы учетные данные.

```
## Python
>>> emails[emails.member == 998].shape
(24, 3)
```

Мы насчитали 24 строки, а их должно быть 26. Это указывает на то, что для этого члена организации не приведены данные за несколько недель. Между прочим, мы могли выполнить такой расчет для всех членов организации сразу, обратившись к групповым операциям, но в нашем примере целесообразно ограничиться всего одним членом организации.

Почему 26 строк

Вас может удивить, что мы говорим о 26, а не о 25 строках, как того требует обозначенная выше операция вычитания. Поступив иначе, мы совершим ошибку, и вот почему. Работая с временны*ми* рядами, после вычитания точек данных нужно спросить себя, нужно ли добавлять единицу к полученному результату, чтобы учесть начальное смещение? Иначе говоря, нужно ли вычитать позицию, которая учитывается в данных?

Рассмотрим следующий пример. Допустим, в модели данных приведена информация для 7, 14, 21 и 28 апреля. Необходимо определить общее количество учитываемых точек данных. Вычитание 7 из 28 и деление на 7 дает 21/7, что равно 3. При этом вполне очевидно, что данные должны представляться четырьмя точками. В самом начале вычиталась точка 7 апреля, но ее нужно вернуть обратно в модель, поэтому правильный расчет — это разница между первым и последним днями, разделенная на 7 плюс 1 для учета вычтенной даты начала.

Вернемся к задаче заполнения пробелов для получения полного набора данных. Нам удалось выяснить, что в наборе действительно есть недостающие данные — для некоторых недель они просто не указаны. Мы не можем точно идентифицировать все пропущенные недели, поскольку некоторые из них могли предшествовать самой ранней зарегистрированной дате или следовать после самой последней даты списка. Однако мы можем заполнить пропущенные значения между начальной и конечной временны*ми* точками, в которых член организации открывал хотя бы одно электронное письмо.

Намного легче заполнить все недостающие недели для всех участников сразу, используя функции индексации пакета `Pandas`. Можно сгенерировать объект `Multiindex` из фрейма данных `Pandas`, который представляет все комбинации недель и членов организации, т.е. их декартово произведение.

```
## Python
>>> complete_idx = pd.MultiIndex.from_product((set(emails.week),
                                               set(emails.member)))
```

Используем полученный индекс для переиндексации исходной таблицы и заполнения пропущенных значений — в нашем случае нулем, предполагая, что если в строке ничего не записано, значит, записывать было нечего. Мы также сбрасываем индекс, чтобы заносить информацию о членах организации и неделях в отдельные столбцы, а затем присваиваем имена этим столбцам.

```
## Python
>>> all_email = emails.set_index(['week', 'member']).
                               reindex(complete_idx, fill_value = 0).
                               reset_index()
>>> all_email.columns = ['week', 'member', 'EmailsOpened']
```

Еще раз изучим данные для члена организации с индексом 998.

```
## Python
>>> all_email[all_email.member == 998].sort_values('week')
   week    member  EmailsOpened
2015-02-09  998     0
2015-02-16  998     0
2015-02-23  998     0
2015-03-02  998     0
2015-03-09  998     0
```

Библиотека Pandas языка Python

`Pandas` — пакет анализа фреймов данных в языке программирования Python, который широко используется в анализе данных. Одно только название указывает на пригодность этого пакета для анализа временны*х* рядов: слово “Pandas” полу-

чено как сокращение термина "panel data" (панельные данные). Таким термином временны*е* ряды обозначаются в социологии.

Библиотека `Pandas` основана на таблицах данных, индексируемых по строкам и столбцам. Ею поддерживаются SQL-подобные операции, такие как группировка, выбор строки и индексация по ключам. Она также снабжена специальными функциями работы с временны*ми* рядами, такими как индексация по временны*м* периодам, прореживание и группирование данных по временны*м* точкам.

Если вы не знакомы с пакетом `Pandas`, я настоятельно рекомендую ознакомиться с его основными возможностями, описанными, например, в официальной документации (`https://perma.cc/7R9B-2YPS`).

Обратите внимание на большое количество нулей в начале временно*го* ряда. Вероятнее всего, они указаны для временны*х* периодов, в которых рассматриваемый член организации не был включен в список почтовой рассылки. Существует не так много видов анализа, в которых пустые недели являются значащими и требуют обязательного учета — особенно те недели, которые предшествовали моменту открытия членом организации первого письма корпоративной почты. Располагая сведениями о такой дате (дате начала получения писем из рассылки), мы сможете предельно точно определить момент временно*й* отсечки данных. Для этого внимательно изучим имеющиеся данные. Для каждого члена организации определим срезы `start_date` и `end_date`, группируя `DataFrame` электронной почты для каждого члена организации и выбирая максимальное и минимальное значения недели.

```
## Python
>>> cutoff_dates = emails.groupby('member').week.
                          agg(['min', 'max']).reset_index)
>>> cutoff_dates = cutoff_dates.reset_index()
```

Удалим из `DataFrame` строки, которые не вносят заметного вклада в хронологию событий, а именно — все нулевые строки перед первыми ненулевыми строками каждого члена организации.

```
## Python
>>> for _, row in cutoff_dates.iterrows():
>>> member       = row['member']
>>> start_date   = row['min']
>>> end_date     = row['max']
>>> all_email.drop(
            all_email[all_email.member == member]
            [all_email.week < start_date].index, inplace=True)
>>> all_email.drop(all_email[all_email.member == member]
            [all_email.week > end_date].index, inplace=True)
```

Операция < или <=?

Здесь используются операции < и > строгого неравенства, потому что объекты `start_date` и `end_date` включают значащие точки данных, и мы отбрасываем, а не сохраняем данные в коде. В этом случае нам нужно включить контрольные недели в анализ, потому что они устанавливают первую и последнюю значащие точки данных.

Вам следует убедить специалистов по сбору данных и администраторов баз данных хранить информацию в хронологическом порядке, наглядно показав, как и зачем создаются временны*е* метки. Чем больше проблем получится решить заранее, тем с меньшим количеством трудностей вы столкнетесь впоследствии.

После очистки учетных данных об активности при работе с почтой можно переходить к решению последующих задач. В частности, нужно попытаться определить связь между частотой изучения электронных писем и датой добровольных взносов, совершаемых всеми членами организации. Давайте попробуем ответить на такие вопросы.

- Просуммируем значения переменной `DonationAmount` по неделям, чтобы получить сопоставимые периоды времени. Теперь мы сможем проанализировать, соотносятся *ли* взносы каким-либо образом с частотой открытия электронных писем членами организации.
- Будем рассматривать переменную `EmailOpened` за предыдущую неделю как предиктор для переменной `DonationAmount` для текущей недели. Очень важно исходно рассматривать именно предыдущую неделю, поскольку `EmailsOpened` указывает сводную статистику за всю неделю. Например, если требуется предсказать вероятность оплаты взносов в среду, а переменная `EmailsOpened` обозначает количество открытых электронных писем с понедельника по воскресенье, то использование информации за эту же неделю потенциально приведет к включению в расчет действий, совершаемых впоследствии (например, открытия письма в пятницу — уже после внесения пожертвований).

Построение полученного временн*о*го ряда

Теперь рассмотрим, как можно связать данные о работе с почтой и оплатой взносов. Можно перестроить данные о взносах, чтобы превратить их в недельные временны*е* ряды, однозначно сопоставляемые с данными об активности членов организации при изучении почты. Нас будут интересовать еженедельные взносы, поэтому мы будем суммировать данные для временны*х* меток, относящихся к общим недельным периодам. Более одного взноса в неделю — событие маловероятное, поэтому в большинстве случаев еженедельные суммы пожертвований будут отражать взносы индивидуальных плательщиков.

```
## Python
>>> donations.timestamp = pd.to_datetime(donations.timestamp)
>>> donations.set_index('timestamp', inplace = True)
>>> agg_don = donations.groupby('member').apply(
            lambda df: df.amount.resample("W-MON").sum().dropna())
```

В этом коде мы сначала преобразуем строковый символ в соответствующий класс данных с временны*ми* метками, чтобы в дальнейшем применить функцию индексации по дате пакета `Pandas`. Тогда при проведении повторной выборки фрейма данных в качестве индекса можно будет использовать временны*е* метки. Наконец, в получаемых для каждого члена организации фреймах данных мы группируем и суммируем взносы за недельные периоды времени, отбрасываем недели, в которых не оплачивались взносы, а затем складываем их вместе.

Обратите внимание, что повторная выборка данных начинается с недели, к которой относится связанная дата, чтобы в точности сопоставить их с недельными датами в таблице данных о работе членов организации с электронной почтой. Также обратите внимание, что неделя, начинающаяся с понедельника, — вполне разумный выбор.

Получив выборки данных о взносах и работе с электронной почтой, имеющие одинаковую периодичность, объединим их. Библиотека `Pandas` позволяет решить эту задачу очень просто, поскольку ранее все недели уже были привязаны к одному и тому же дню. Пройдем по всем элементам и объединим их фреймы данных.

```
## Python
>> for member, member_email in all_email.groupby('member'):
>>> member_donations = agg_donations[agg_donations.member
                                     == member]

>>> member_donations.set_index('timestamp', inplace = True)
>>> member_email.set_index ('week', inplace = True)

>>> member_email = all_email[all_email.member == member]
>>> member_email.sort_values('week').set_index('week')

>>> df = pd.merge(member_email, member_donations, how = 'left',
                  left_index = True,
                  right_index = True)
>>> df.fillna(0)

>>> df['member'] = df.member_x
>>> merged_df = merged_df.append(df.reset_index()
                   [['member', 'week', 'emailsOpened', 'amount']])
```

На данный момент мы располагаем выровненными данными о работе с почтой и взносах каждого члена организации. Для каждого из них учитываются

только значащие недели и отбрасываются те, которые предшествовали вступлению в организацию или следуют после ухода из нее.

Мы можем рассматривать работу с электронной почтой как переменную "статуса" по отношению к поведению внесения пожертвований, но чтобы избежать упреждения, данные события должны рассматриваться с недельным смещением. Нам нужно построить модель, в которой прогнозирование следующих взносов члена организации основано на данных о работе с электронной почтой за прошлую неделю. В этом случае события оплаты можно попробовать сопоставить с событиями просмотра полученных почтовых сообщений. Все, что нам требуется, — это поставить в соответствие текущие недельные взносы с данными о количестве открытых письмах за предыдущую неделю. Для этого нужно сдвинуть выровненные и очищенные данные из обеих таблиц на соответствующее количество недель. Задача смещения оплаты взносов на неделю вперед легко выполняется с помощью оператора сдвига, но не забывайте, что эту операцию нужно задействовать сразу ко всем членам организации.

```
## Python
>>> df = merged_df[merged_df.member == 998]
>>> df['target'] = df.amount.shift(1)
>>> df = df.fillna(0)
>>> df
```

Рекомендуем сохранять результат сдвига в новом столбце, а не перезаписывать исходные данные, особенно в случаях отказа от смещения вперед временных меток в данных о суммах взносов. Мы сместим их на одну неделю вперед отдельно, используя все ту же встроенную функцию сдвига пакета `Pandas`. Ничто не запрещает сдвинуть временные метки на неделю назад, задавая отрицательные значения параметров. Обычно предикторов больше, чем целей, поэтому чаще сдвигаются цели. Ниже показан результат выполнения кода.

```
amount  emailsOpened  member   week        target
0       1             998    2017-12-04    0
0       3             998    2017-12-11    0
0       3             998    2017-12-18    0
0       0             998    2017-12-25    0
0       3             998    2018-01-01    0
50      3             998    2018-01-08    0
0       2             998    2018-01-15   50
```

Итак, после учета пропущенных строк данные члена организации с идентификатором 998 стали насчитывать 26 строк. Набор данных стал полным и очистился от ненужной информации.

Еще раз кратко опишем принципы обработки временных рядов, использованные нами при реструктуризации данных.

1. *Изменение детализации данных* в соответствии с задачей. Часто данные содержат более детальную временную информацию, чем требуется.
2. *Предотвращение упреждения* — отказ от данных с временными метками, которые влияют на доступность данных.
3. *Учет всех релевантных временных периодов*, даже если в них "ничего не происходило". Нуль так же информативен, как и любое другое число.
4. *Избегание упреждения* — отказ от данных с временными метками, которые предоставляют информацию, о которой пока не должно быть известно.

До сих пор мы работали с необработанными полученными временными рядами, сопоставляя данные о взносах членов организации и об их работе с электронной почтой, которые собирались с одинаковой частотой и за одинаковые временные периоды. Заметьте, что мы не прибегали к тщательной очистке этих данных или к полному их изучению перед дальнейшим анализом. Этим мы займемся в главе 3.

Трудности описания временных меток

Временные метки очень важны в анализе временных рядов. По временным меткам можно экстраполировать ряд интересных признаков, таких как время суток или день недели. Изучение таких признаков сильно помогает в изучении исходных наборов данных, особенно касающихся поведения людей. Тем не менее временные метки относятся к неоднозначно представляемым данным. Далее мы обсудим некоторые трудности задания временных меток.

Получение временных меток

Первая задача, которую предстоит решить при изучении временных меток, заключается в определении того, каким процессом, как и когда они были сгенерированы. Очень часто происходящее событие не совпадает с регистрируемым событием. Например, в ходе работы исследователь может быстро внести новые данные в записную книжку, а добавить их в основной файл журнала формата CSV несколько позже (в конце рабочего дня). Какое время при этом будет указано в записи — добавления данных в записную книжку или в CSV-файл? Например, пользователь мобильного приложения может совершать действия, автоматически регистрируемые в журнале даже тогда, когда его телефон находится в оффлайн-режиме. На сервер данные журнала будут загружены после подключения телефона к сети, т.е. с некоторой временной задержкой. Что в таком случае отражают временные метки, передаваемые вместе с данными: время совершения пользователем действий, записи их приложением, загрузки метаданных

на сервер или запрос к серверу на обновление данных приложением (или любого другого события по передаче целевых данных)?

В первую очередь, временны*е* метки помогают выяснить, что именно описывают данные, но они оказываются бесполезными при отсутствии надлежащего их описания. Изучая временны*е* метки, вам нужно попытаться как можно лучше определить время регистрации событий.

Проиллюстрируем трудности описания временны*х* меток на конкретном примере. Представьте, что вы просматриваете данные, предоставленные мобильным приложением, которое помогает избавиться от лишнего веса, и наблюдаете на экране телефона записи, подобные приведенным в табл. 2.4.

Таблица 2.4. Записи о потребленных блюдах, отображаемые приложением

Время	Блюдо
Пн, 7 апреля, 11:14:32	Блины
Пн, 7 апреля, 11:14:32	Сэндвич
Пн, 7 апреля, 11:14:32	Пицца

Конечно, существует вероятность, что пользователь смог съесть блины, сэндвич и пиццу за один присест, но ведь есть более реалистичные сценарии. Чтобы правильно оценить ситуацию, нужно ответить на ряд вопросов. Время приема пищи указано пользователем или оно генерируется автоматически? Предлагает ли приложение устанавливать время по шаблону, предполагающему переназначение или игнорирование значения по умолчанию? Ответы на эти вопросы могли бы объяснить одинаковые временны*е* метки правдоподобнее, чем предположение, что пользователь, пытающийся похудеть, съедает блины, сэндвич и пиццу за время всего одного ланча.

Но даже если в 11:14 у пользователя были все эти блюда, где именно он так славно покушал? Какое время указывается в записях приложения: местное или глобальное? Даже если принять маловероятное предположение, что пользователь съел все блюда за один присест, из одних этих записей можно мало что почерпнуть о временно*м* аспекте события. Например, мы не знаем, был ли это завтрак, обед, ужин или просто перекус. Чтобы предоставить пользователю полезную информацию о питании, нужно знать время суток, которое обозначает указанное в записях время, а для этого нужно располагать сведениями о часовом поясе места его пребывания.

Лучший способ ответить на эти вопросы — изучить код, отвечающий за сбор и хранение данных, или попросить помощи у разработчиков приложения, написавших этот код. После ознакомления со всеми доступными техническими аспектами данных вам нужно самостоятельно протестировать приложение, чтобы убедиться в том, что данные собираются именно так, как предполагается. Чем

лучше вы изучите поток регистрируемых данных, тем меньше вероятность того, что у вас возникнут неправильные вопросы о назначении временны*х* меток — вам не будет казаться, что они обозначают не то, что должны обозначать на самом деле.

Вы и только вы отвечаете за правильность трактовки данных. Люди, от которых к вам поступают данные, не знают, каким образом вы собираетесь их анализировать. Постарайтесь как можно лучше понять, как и когда именно генерируются временны*е* метки. Если данные получены из мобильного приложения, то загрузите и запустите это приложение, добейтесь возникновения в нем необходимого события в нескольких правдоподобных сценариях и внимательно ознакомьтесь с получаемыми данными. Пообщавшись с разработчиками, отвечающими за написание кода обработки потоков данных, вы будете удивлены тем, как приложение регистрирует действия пользователей. В одном приложении сложно учесть не только несколько механизмов отсчета времени, но и все непредвиденные обстоятельства, поэтому большинство наборов данных выравниваются по одной временно*й* оси. И вы должны иметь четкое представление о том, как именно это делается.

Целесообразность расстановки временны*х* меток

Если иметь дело со старыми или недокументированными потоками данных, у вас не получится полноценно исследовать их и пообщаться с теми, кто их сопровождает. Чтобы понять назначение временны*х* меток в таких данных, придется ограничиться умозрительным исследованием.

- Внимательно изучая данные, как это делалось в предыдущем примере, можно выдвинуть первоначальные гипотезы о предназначении временны*х* меток. Просто просмотрите записи для нескольких пользователей, чтобы узнать, основаны они на одном и том же шаблоне (записи с одинаковыми временны*ми* метками и неадекватным количеством блюд как для одного приема пищи) или представляют действительные, хотя и несколько необычные данные.
- Проанализировав агрегированные данные, вы сможете лучше понять, что обозначают или на что указывают временны*е* метки. В предыдущем примере нам осталось ответить на такие вопросы.
  - В каком формате заданы временны*е* метки: локального часового пояса или глобального, всемирного времени?
  - Сопоставляются ли временны*е* метки с действиями пользователя или внешними событиями, например с подключением приложения к серверу?

Местное или всемирное время

Большинство временных меток задано в формате всемирного скоординированного времени (UTC) или локального часового пояса, что зависит от физического местоположения сервера, а не самого пользователя. При этом хранить временные данные в формате локального временного пояса кажется нелогичным. Тем не менее вы должны рассмотреть оба варианта, потому что оба они встречаются на практике очень часто.

Если все временные значения представлены в формате местного времени, то нам будет намного проще отследить разницу в дневном и ночном поведении пользователей. В частности, не стоит ожидать от них большой ночной активности, ведь большинство из них в это время спит. В примере мобильного приложения для похудения ночью должен регистрироваться заметно меньший объем потребленных блюд, поскольку в большинстве стран люди не имеют устойчивой привычки есть посреди ночи.

Если бы во временных метках однозначно не указывался день недели, то можно было бы предположить, что они представлены в универсальном временном формате, а пользовали, данные о которых занесены в клиентскую базу, проживают в разных временных поясах. В подобном случае экстраполяция данных отдельных пользователей, снабженных временными метками, которые представлены в формате местного времени без указания часового пояса, представляется невероятно сложной вычислительной задачей. Она предполагает полный анализ данных всех пользователей, детальное изучение временных меток каждого из них и добавление в них точных сведений о действительных часовых поясах. Заметим, что наряду с высоким уровнем сложности такой комплекс мер характеризуется низкой точностью.

Даже если вам не удастся определить часовые пояса мест проживания пользователей, временные метки все равно придется переписать в формате всемирного времени. Для начала попробуйте определить временные периоды наиболее интенсивного использования серверов вашим приложением, ориентируясь по тому, в какое время суток и дни недели пользователи чаще всего передают данные о своей диете. Вам также может понадобиться рассчитать разницу между временем приема пищи пользователями — здесь не нужно беспокоиться о часовых поясах, поскольку для каждого пользователя временные значения указываются в абсолютном формате. Последняя операция оказывается полезной в том числе при генерировании признаков.

```
# Python
>>> df['dt'] = df.time - df.time.shift(-1)
```

Здесь столбец `dt` представляет признак, который можно использовать в дальнейшем анализе. В частности, по разнице между временными точками можно

попробовать оценить часовой пояс каждого пользователя. Определите время суток, которому соответствуют большие значения столбца `dt`. Скорее всего, они обозначают ночной период времени у рассматриваемого пользователя. Благодаря такому подходу можно приблизительно оценить временны*е* пояса всех пользователей, избегая более комплексного анализа.

Действия пользователей и сетевые события

Перейдем ко второму нерешенному вопросу нашей короткой выборки данных. Нам предстоит выяснить, какие событие обозначают имеющиеся временны*е* метки: времени действительного приема пищи пользователем или загрузки введенных им данных на сервер.

Чтобы ответить на второй вопрос, нужно прибегнуть к таким же рассуждениям, как при определении часового пояса места пребывания пользователя. Проанализировав столбец признака `dt` (показан выше), нужно найти в нем нулевые кластеры, а затем попытаться выяснить, на что они указывают — на отдельные пользовательские действия или на сетевые события. Также попробуйте определить периодичность изменения признака `dt` в разные дни. Для пользовательского поведения периодичность является более характерным критерием, чем для работы программного обеспечения или сетевого оборудования.

Подводя итог, перечислим задачи, которые нужно решить, чтобы получить представление о структуре набора данных, даже не располагая информацией о том, как в нем генерируются временны*е* метки.

- Исследуйте временны*е* расстояния между отдельными временны*ми* метками, чтобы получить представление о периодичности приема пищи для каждого пользователя или ввода им данных о своей диете (в зависимости от рабочей гипотезы о регистрируемом событии: совершаемое пользователем действие или сетевое событие).
- Опишите общее поведение пользователей, чтобы определить время наибольшей загруженности серверов в течение суток (24-часового рабочего цикла).

Часовые пояса

В наиболее благоприятном сценарии данные снабжаются временны*ми* метками, заданными в формате UTC, который по умолчанию применяется в большинстве баз данных и других систем хранения информации. Однако в отдельных ситуациях вы можете столкнуться с данными, в которые временны*е* метки задаются в формате, отличном от UTC.

- Временные метки не связываются с объектами данных, для которых они генерируются, как в случае API-вызовов. Такие вызовы основаны на строковых объектах, а не на объектах времени.
- Данные собираются вручную небольшими местными организациями, в которых сведения о часовом поясе не играют никакой роли, например электронные таблицы, создаваемые бизнес-аналитиками или биологами, работающими в полевых условиях.

Осмысленная шкала времени

Временное разрешение анализируемых данных устанавливается, исходя из полного понимания изучаемых процессов и особенностей сбора данных.

Например, представьте, что вы просматриваете данные о ежедневных продажах, понимая, что во многих случаях менеджеры сдают точные отчеты только в конце рабочей недели, а в конце каждого рабочего дня оценивают основные учетные показатели только приблизительно. Ошибка между фактическими и указываемыми в отчетах данными может оказаться более чем существенной. Чаще всего она обусловливается трудностями учета возврата товара и неизбежными искажениями восприятия менеджерами собственных возможностей. Вы можете изменить масштабирование данных об объемах продаж с ежедневного на еженедельное, чтобы уменьшить или усреднить эту систематическую ошибку. В противном случае вам придется построить модель, учитывающую ошибки восприятия действительных данных в разные дни недели. Например, может случиться так, что менеджеры систематически переоценивают свою результативность за понедельник, указывая ее в пятничных отчетах.

Психологическое дисконтирование во времени

Дисконтирование во времени — это частное проявление феномена, известного как *психологическая дистанция*, отражающего общую тенденцию к более оптимистичной (и менее реалистичной) оценке событий, относящихся к более отдаленным моментам времени. Дисконтирование во времени говорит о том, что данные, полученные в более отдаленном прошлом, будут систематически недооценены по сравнению с данными, полученными относительно недавно. Этот феномен выделяется из общей проблемы потери воспоминаний, предопределяя неслучайную ошибку. Не забывайте о его существовании, анализируя собранные вручную данные, запись которых велась не тогда, когда происходили регистрируемые события.

Следующее ограничение, требующее детального рассмотрения, носит сугубо физический характер. Например, уровень глюкозы в крови человека изменяется с некой предельной скоростью, поэтому, просматривая набор данных об уровнях глюкозы в крови, собранных в течение нескольких секунд, обязательно усредните их — рассматривать их как отдельные точки данных малопродуктивно. В противном случае вы будете собирать не уровни исследуемого показателя, а значения ошибки устройства, представленные частыми флуктуационными измерениями регистрируемых данных.

Людям известно о течении времени

Всякий раз, проводя медицинские исследования, помните о том, что биологические процессы сильно зависят от восприятия человеком времени. Например, недавние исследования показали, что манипулирование скоростью течения времени на циферблате часов, которые показывают пациенту, вызывает синхронные изменения в скорости изменения уровня глюкозы в крови.

Очистка данных

В этом разделе мы рассмотрим наиболее распространенные недостатки наборов данных временны*х* рядов.

- Отсутствующие данные.
- Изменение частоты временно*г*о ряда (так называемая повышающая и понижающая дискретизация).
- Сглаживание данных.
- Решение проблемы сезонности в данных.
- Предотвращение непреднамеренного упреждения.

Обработка недостающих данных

Отсутствующие данные встречаются на удивление часто. Например, пропущенные данные в медицинских временны*х* рядах могут объясняться множеством причин.

- Пациент не провел необходимое измерение
- Пациент признан здоровым, и в проведении дополнительных измерений нет необходимости
- О пациенте забыли или не уделили ему достаточного внимания

- На момент проведения измерений медицинское устройство было технически неисправным
- Произошла ошибка ввода данных

Подытоживая вышесказанное, можно утверждать, что пропущенные данные в анализе временны*х* рядов встречаются чаще, чем в перекрестном анализе, поскольку полноценные продольные выборки данных получить очень сложно. Неполные временны*е* ряды встречаются настолько часто, что специалистами по анализу были разработаны специальные методы решения проблемы пропущенных записей.

Ниже перечислены наиболее распространенные способы устранения пропущенных данных во временны*х* рядах.

Восполнение

Заполнение недостающих данных на основе наблюдений по всему набору данных.

Интерполяция

Недостающие значения получаются на основе соседних точек данных. Интерполяция также может быть формой восполнения.

Удаление периодов времени пропущенных данных

Отказ от использования периодов времени, в которых отсутствуют значащие данные.

Первыми мы рассмотрим методы восполнения и интерполяции, не забыв проиллюстрировать их на практических примерах. Заметьте, что оба метода не вызывают сокращения набора данных, чего нельзя сказать о методе удаления периодов пропущенных данных, который приводит к уменьшению размера модели данных. Сохранять данные или отказываться от проблемных периодов времени — зависит от сценария использования данных и от того, можно ли пожертвовать такими периодами времени в угоду целостности набора и модели данных.

Подготовка набора данных к восполнению отсутствующих значений

В дальнейших изысканиях мы будем опираться на ежемесячные данные о безработице (`https://data.bls.gov/timeseries/LNS14000000`), которые публикуются правительством США начиная с 1948 года (находятся в свободном доступе). Взяв их за основу, мы сгенерируем два новых набора данных. В первом из них отсутствующие данные удалялись из набора случайным образом, а во втором — были представлены только месяцами с самым высоким уровнем безработицы за всю историю сбора данных. Таким образом, мы сможем провести два независимых исследования и увидеть, как восполняются временны*е* ряды, в которых пропуски данных носят случайный и систематический характер.

В следующем примере мы переходим к использованию языка программирования R. В дальнейшем материале мы будем переключаться между языками R и Python без специальных упреждений. Предполагается, что у вас есть определенный опыт работы с фреймами данных и матрицами как в R, так и в Python.

```
## R
> require(zoo)                  ## zoo - временной ряд
> require(data.table)           ## data.table - высокопроизводительный
                                ## фрейм данных

> unemp <- fread("UNRATE.csv")
> unemp[, DATE := as.Date(DATE)]
> setkey(unemp, DATE)

> ## Генерирование набора данных со случайными пропущенными значениями
> rand.unemp.idx <- sample(1:nrow(unemp), .1*nrow(unemp))
> rand.unemp <- unemp[-rand.unemp.idx]

> ## Генерирование набора данных с пропущенными значениями
> ## для высоких уровней безработицы,
> high.unemp.idx <- which(unemp$UNRATE > 8)
> num.to.select <- .2 * length(high.unemp.idx)
> high.unemp.idx <- sample(high.unemp.idx,)
> bias.unemp <- unemp[-high.unemp.idx]
```

Чтобы создать набор данных с отсутствующими записями на основе таблиц данных с только что удаленными строками, нужно считать отсутствующие даты и значения NA, для чего лучше всего обратиться к инструментам *скользящего объединения* пакета `data.table`.

Пакет `data.table` языка R

Пакет `data.table` языка R — это высокопроизводительная, но сильно недооцененная альтернатива более популярного пакета `data.frame`. Многие из вас уже знакомы с набором пакетов `tidyverse` (`https://perma.cc/E4S8-RUHN`). В отличие от них, пакет `data.table` — это автономная библиотека, созданная поверх `data.frame`. Она имеет много полезных функций анализа временных рядов. Вероятно, это связано с тем, что ее создавали разработчики финансового программного обеспечения, которые испытывали острую нехватку высокопроизводительных инструментов языка R для обработки финансовых данных, чаще всего представленных временными рядами.

Хотя пакет `data.table` несколько сложнее `data.frame`, я считаю его сильно недооцененным относительно возможностей работы с временными рядами в языке R. Его инструменты применяются во многих дальнейших примерах. Сейчас же да-

вайте кратко рассмотрим различия в принципах обработки данных пакетами `data.table` и `data.frame`.

```
dt[, new.col := old.col + 7]
```

В этом примере предполагается, что исходный объект `data.table` включает столбец с именем `old.col`. Последней инструкцией создается новый столбец на месте, что означает добавление нового столбца в таблицу `data.table` без полного ее копирования в новый объект, что приводит к сокращению потребляемой памяти. Вскоре вы поймете, что оператор `:=` используется в рассматриваемых далее примерах необычайно часто.

Чтобы получить доступ к подмножеству столбцов объекта `data.table`, их список передается в качестве аргумента индексации столбцов — второго аргумента структуры `data.table`.

```
dt[, .(col1, col2, col3)]
```

Эта инструкция возвращает подвыборку объекта `data.table`, включающую только объекты `col1`, `col2` и `col3`. Обратите внимание на оператор `.()`, в который заключены названия столбцов. Он представляет собой сокращенную запись оператора `list()`.

Для выборки строк, а не столбцов, их нужно определить в первом аргументе структуры `data.table`, например так.

```
dt [col1 <3 & col2> 5]
```

Инструкция возвращает все столбцы таблицы `data.table`, для которых выполнены оба логических условия. И конечно, выборку можно определять комбинированными условиями для строк и столбцов, как показано далее.

```
dt [col1 <3 & col2> 5, (col1, col2, col3)]
```

Наконец, высокую производительность и скорость обработки данных показывают операции группирования. Например, чтобы подсчитать количество строк в каждой группе из заданных в столбце `col1`, нужно использовать следующий код.

```
dt [, .N, col1]
```

Над сгруппированными объектами можно выполнять множественные операции. Для этого потребуется снова создать список и даже назвать столбцы.

```
dt [,. (total = .N, mean = mean(col2), col1]
```

Эта инструкция возвращает общее количество строк в группах, указанных в столбце `col1`, и среднее значение `col2` для каждой группы.

Пакет `data.table` находится в активной разработке и является бесценным инструментом анализа временных рядов, особенно полученных из больших наборов данных. Я настоятельно рекомендую ознакомиться с официальной документацией к нему (`https://perma.cc/3HEB-NE6A`).

```
## R
> all.dates <- seq(from = unemp$DATE[1], to = tail(unemp$DATE, 1),
                                                    by = "months")
> rand.unemp = rand.unemp[J(all.dates), roll=0]
> bias.unemp = bias.unemp[J(all.dates), roll=0]
> rand.unemp[, rpt := is.na(UNRATE)]
```

В скользящем объединении генерируется последовательность всех дат, которые должны оставаться доступными между начальной и конечной точками набора данных. Такой подход позволяет включать в набор данных строки с отсутствующими данными — обозначенные как NA.

Теперь, когда у нас есть наборы данных с отсутствующими значениями, рассмотрим несколько способов заполнения таких отсутствующих значений числами.

- Замена предыдущим значением.
- Скользящее среднее.
- Интерполяция.

Будем сравнивать производительность обозначенных методов на наборах как со случайно отсутствующими, так и с систематически пропускаемыми данными. Поскольку оба набора данных получены из общего полного набора данных, нам будет намного проще проверить правдоподобность восполнения значений, избегая неправильных результатов. Конечно, решая реальные задачи, вы не будете располагать контрольным набором значений для проверки восполнения данных.

Скользящее объединение

Скользящее объединение в data.table работает во многом подобно объединению данных в SQL, но только в масштабе временны*х* рядов. Несмотря на то что таблицы далеко не всегда согласуются по заданным в них временны*м* меткам, функция скользящего соединения обрабатывает их весьма точно.

```
## R
> ## Фрагмент data.table с данными о датах взносов
> donations <- data.table(
> amt = c(99, 100, 5, 15, 11, 1200),
> dt = as.Date(c("2019-2-27", "2019-3-2", "2019-6-13",
>                "2019-8-1", "2019-8-31", "2019-9-15"))
> )
> ## Информация о каждой информационной кампании
> publicity <- data.table(
>                identifier = c("q4q42", "4299hj", "bbg2"),
>                dt = as.Date(c("2019-1-1",
>                               "2019-4-1",
>                               "2019-7-1")))
```

```
> ## Установка первичного ключа для каждого пакета data.table
> setkey(publicity, "dt")
> setkey(donations, "dt")

> ## Нужно снабдить данные о взносах в рамках
> ## последней кампании временными метками.
> ## Для этого параметру roll присваивается значение TRUE
> publicity[donations, roll = True]
```

Результат превосходит всякие ожидания.

```
## R
> publicity[donations, roll = TRUE]
  identifier         dt   amt
1:      q4q42 2019-02-27    99
2:      q4q42 2019-03-02   100
3:     4299hj 2019-06-13     5
4:       bbg2 2019-08-01    15
5:       bbg2 2019-08-31    11
6:       bbg2 2019-09-15  1200
```

Теперь каждый взнос связывается с идентификатором, обозначающим информационную кампанию, которая наиболее вероятно предшествовала благотворительным взносам.

Замена предыдущим значением

Один из самых простых способов восполнения пропущенных значений — заполнить их известными предыдущими значениями. Такой метод получил название *замена предыдущим значением*. Он основан на простом принципе и не требует дополнительных математических вычислений и сложных логических рассуждений. Просто представьте, что в ходе анализа имеющихся данных вы сдвигаете время на шаг вперед — недостающие значения оказываются заполненными зарегистрированными ранее данными. При таком подходе недостающие значения восполняются преимущественно предыдущими известными измерениями.

Восполнение предыдущими значениями легко выполняется с помощью функции `na.locf` пакета `zoo`.

```
## R
> rand.unemp[, impute.ff := na.locf(UNRATE, na.rm = FALSE)]
> bias.unemp[, impute.ff := na.locf(UNRATE, na.rm = FALSE)]
>
> ## Вывод графика упрощенного набора данных
> unemp[350:400, plot (DATE, UNRATE,
                       col = 1, lwd = 2, type = 'b')]
> rand.unemp[350:400, lines(DATE, impute.ff,
                       col = 2, lwd = 2, lty = 2)]
> rand.unemp[350:400][rpt == TRUE, points(DATE, impute.ff,
                       col = 2, pch = 6, cex = 2)]
```

Результирующий график выглядит достаточно естественно, за исключением точек, в которых отсутствующие данные замещены повторяющимися значениями (рис. 2.6). Тем не менее, как показано на графике, значения, полученные подстановкой предыдущими данными, обычно сильно не отклоняются от истинных значений.

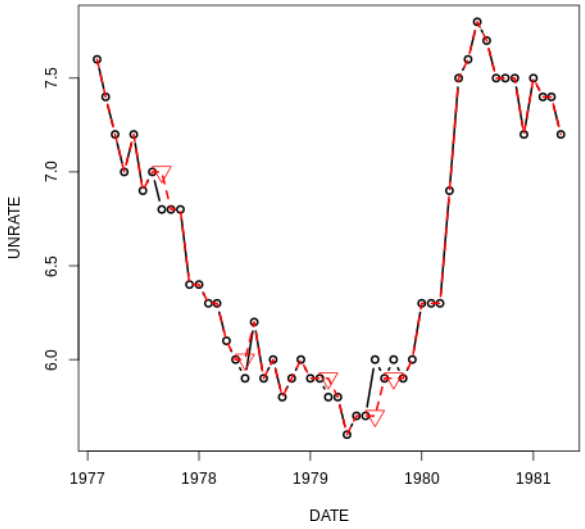


Рис. 2.6. *Исходный временной ряд (сплошная линия) и временной ряд, построенный методом восполнения пропущенных данных (пунктирная линия). Точки, полученные заменой предыдущими значениями, отмечены символами перевернутого треугольника*

Для более наглядного изучения полученного результата давайте построим сравнительные графики временных рядов, основанных на исходном и восполненном наборах значений. На них большинство значений должны совпадать абсолютно точно в силу общности их данных. На рис. 2.7 видно, что большинство точек данных полностью совпадает. На нем также показаны отдельные, разбросанные вокруг общей линии точки данных, но они не демонстрируют систематического смещения.

Замена следующим значением

Подобно замене недостающих данных предыдущими значениями для восполнения отсутствующих точек данных можно выбрать обратное направление течения времени и использовать в их качестве следующие известные значения. Тем не менее такой подход — это своего рода упреждение, поэтому он справедлив только в случаях отказа от использования полученных наборов данных в прогнозировании ситуаций. Такой способ восполнения обусловливается исключительно предметной областью и целесообразностью заполнения данных в обратном, а не прямом направлении.

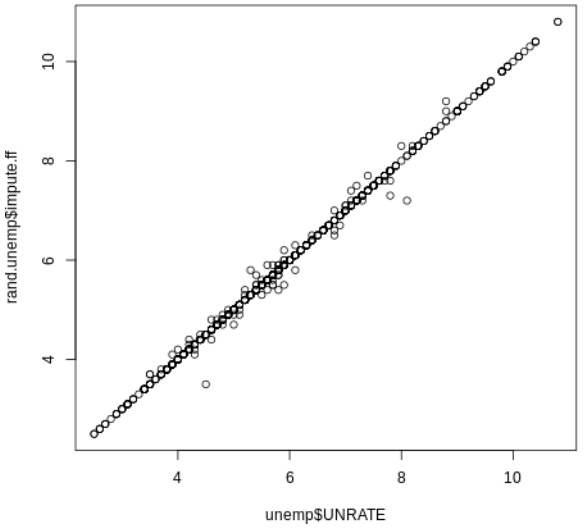


Рис. 2.7. *График истинного уровня безработицы и временного ряда с восполненными значениями. Как видите, замена отсутствующих данных предыдущими точками не вносит систематических искажений в исходный набор данных*

В некоторых ситуациях замена пропущенного значения предыдущим является наилучшим способом восполнения отсутствующих данных, даже при доступности других, более совершенных методов. Например, в медицинских задачах пропущенное значение часто указывает на то, что врач не счел необходимым провести дополнительные измерения, вероятно, посчитав состояние пациента вполне ожидаемым. В подобных медицинских исследованиях замена отсутствующего значения последним известным будет вполне обоснованной, поскольку она базируется на убеждении медицинского работника в ненадобности пересмотра последних измерений.

Метод замены пропущенного значения предыдущим обладает целым рядом преимуществ: он применяется к последовательным потокам данных, не требует каких бы то ни было вычислений и не вносит систематических искажений в набор значений. Вскоре мы рассмотрим наглядный пример.

Скользящее среднее

Восполнить данные можно *по методу скользящего среднего* или медианы. Этот метод во многом подобен методу замены отсутствующих значений предыдущими. В обоих случаях будущие значения предсказываются по известным предыдущим значениям (восполнение можно рассматривать как специальную форму предсказания). Однако метод скользящего среднего основан на данных *нескольких* последних временны*х* точек.

Существует много ситуаций, когда скользящее среднее лучше подходит для целевой задачи, чем замена пропущенного значения предыдущим. Например, если данные зашумлены и у вас есть основания полагать, что значение отдельной пропущенной точки данных очень близко к среднему уровню значений, то для его определения следует использовать метод скользящего среднего, а не замены предыдущим значением. При замене данных предыдущими значениями в них вносится случайный шум, тогда как усреднение позволяет частично от него избавиться.

Чтобы избежать упреждения, усреднение нужно выполнять исключительно для временны*х* точек, предшествующих пропущенному значению. Следовательно, метод можно реализовать следующим образом.

```
## R
> ## Скользящее среднее без упреждения
> rand.unemp[, impute.rm.nolookahead := rollapply(c(NA, NA, UNRATE), 3,
>              function(x) {
>                          if (!is.na(x[3])) x[3] else mean(x, na.rm = TRUE)
>                          })]
> bias.unemp[, impute.rm.nolookahead := rollapply(c(NA, NA, UNRATE), 3,
>              function(x) {
>                          if (!is.na(x[3])) x[3] else mean(x, na.rm = TRUE)
>                          })]
```

Здесь отсутствующие значения представляются средними значениями всех предшествующих им значений (поскольку мы индексируем окончательное значение и используем это значение, чтобы определить, отсутствует ли оно и как его заменить).

Скользящее среднее не обязательно рассчитывается через среднее арифметическое предыдущих данных. Например, экспоненциально взвешенное скользящее среднее позволяет придать больший вес последним значениям и меньший вес — более отдаленным. С другой стороны, среднее геометрическое лучше использовать во временны*х* рядах, демонстрирующих сильную последовательную корреляцию или представленных последовательностью постоянно увеличивающихся значений.

При восполнении пропущенных данных методом скользящего среднего обязательно определитесь с тем, нужно ли принимать в расчет будущие значения, повышая риск возникновения упреждения. Если упреждение данным не грозит, то вычисление скользящего среднего лучше проводить как для прошлых, так и для будущих временны*х* значений, максимально расширяя объем учитываемой информации и повышая точность прогнозирования значений. В подобных случаях нужно указывать диапазон или окно данных, рассматриваемых функцией `rollapply()` пакета `zoo`.

```
## R
> ## Скользящее среднее с упреждением
> rand.unemp[, complete.rm := rollapply(c(NA, UNRATE, NA), 3,
>              function(x) {
>                          if (!is.na(x[2]))
>                              x[2]
>                          else
>                              mean(x, na.rm = TRUE)
> })]
```

Как указывалось выше, вычисление скользящего среднего одновременно по данным прошлых и будущих временны́х точек прекрасно подходит при решении задач визуализации и сбора полных, непрерывных наборов данных, но оказывается несостоятельным при составлении прогнозов.

Результаты, полученные при вычислении скользящего среднего только по будущим значениям, а также расчета его одновременно по прошлым и будущим значениям, приведены на рис. 2.8.

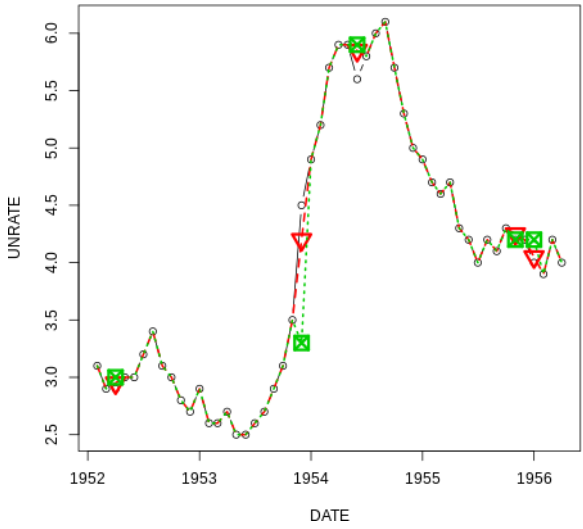


Рис. 2.8. *Точечной линией показан график временного ряда, восполненного по методу скользящего среднего без упреждения, а пунктирной линией — график временного ряда, данные которого восполнены по методу скользящего среднего с упреждением. Небольшими квадратиками обозначены восполненные без упреждения значения, а перевернутыми треугольниками — точки данных, полученные с упреждением*

Восполненный по методу скользящего среднего набор данных характеризуется низким уровнем дисперсии. Это необходимо учитывать при расчете точности вычислений, коэффициента детерминаций набора данных и других

типов ошибок. Ваш расчет может привести к переоценке точности модели, что является частой проблемой, возникающей в ходе анализа временных рядов.

Подбор набора данных для восполнения недостающих значений

В процессе перекрестного анализа восполнение набора данных за счет включения в него отсутствующих точек, представленных средним или медианой, является общепринятой практикой. Хотя данный метод можно применять и по отношению к временным рядам, в большинстве случаев этого избегают. При вычислении среднего значения всего набора данных неизбежно учитывают будущие временные точки... а это упреждение!

Интерполяция

Интерполяция — это метод определения значений отсутствующих точек данных, основанный на геометрических ограничениях поведения данных. Например, в линейной интерполяции отсутствующие данные получаются в результате линейной аппроксимации известных соседних точек.

Наибольший интерес представляет линейная интерполяция, позволяя учитывать поведение системы во времени. Например, если известно, что система изменяется линейно во времени, то для аппроксимации отсутствующих данных лучше всего применять именно линейные зависимости. В байесовском понимании такой подход позволяет получать восполняемые значения с учетом априорных сведений о них.

Как и в случае скользящего среднего, интерполяция может основываться как на прошлых, так и на будущих значениях или учитывать только одно из направлений. Остаются справедливыми стандартные предостережения: интерполяцию по будущим значениям стоит выполнять только в том случае, когда возникающее упреждение не оказывает заметного влияния на получаемый результат.

Ниже показано, как выполняется интерполяция данных с учетом прошлых и будущих временных точек (рис. 2.9).

```
## R
> ## Линейная интерполяция
> rand.unemp[, impute.li := na.approx(UNRATE)]
> bias.unemp[, impute.li := na.approx(UNRATE)]
>
> ## Полиномиальная интерполяция
> rand.unemp[, impute.sp := na.spline(UNRATE)]
> bias.unemp[, impute.sp := na.spline(UNRATE)]
>
> use.idx = 90:120
> unemp[use.idx, plot(DATE, UNRATE, col = 1, type = 'b')]
```

```
> rand.unemp[use.idx, lines(DATE, impute.li, col = 2, lwd = 2, lty = 2)]
> rand.unemp[use.idx, lines(DATE, impute.sp, col = 3, lwd = 2, lty = 3)]
```

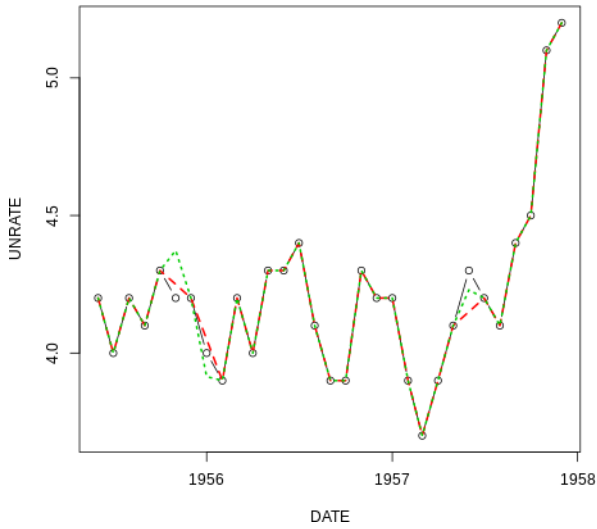


Рис. 2.9. *Пунктирная линия соответствует линейной интерполяции, а точечная линия — сплайновой (полиномиальной)*

Существует много ситуаций, в которых линейная (или сплайновая) интерполяция оказывается наиболее предпочтительным методом восполнения. Например, она будет справедливой в задачах исследования средней еженедельной температуры воздуха, имеющей тенденцию к повышению или понижению в зависимости от времени года, или ежегодных данных об объемах продаж быстро растущего предприятия. Если из года в год объем продаж предприятия увеличивается в одно и то же количество раз, то для восполнения отсутствующих данных лучше всего обратиться к линейной интерполяции. Иначе говоря, линейная интерполяция учитывает тенденции в данных, чего не скажешь об описанном выше методе скользящего среднего. В частности, если в данных просматривается тенденция к увеличению значения, то при использовании метода скользящего среднего восполняемые значения систематически недооценивались бы.

Однако учтите, что существует много ситуаций, в которых линейная (или сплайновая) интерполяция не находит достойного применения. Например, такому восполнению не подлежат пропущенные в наборе метеорологических данных сведения об осадках — изменение их количества происходит далеко не по линейному закону. Точно так при восполнении нескольких временных точек в наборе данных о суточном режиме сна нужно предельно настороженно отнестись к идее экстраполяции данных между двумя известными значениями. Например, одна из

таких временны*х* точек может относиться к бессонной ночи, а следующая — обозначать 30-минутную дремоту. Правильно оценить недостающие данные по ним будет крайне сложно.

Сравнение методов

Рассмотрев несколько различных методов восполнения, можно протестировать их на готовых наборах данных и сравнить полученные результаты.

Ранее мы создали два неполных набора данных, в одном из которых отсутствующие значения выбирались случайным образом, а в другом — в наиболее неудовлетворительных точках, соответствующих высокому уровню безработицы. Сравнивая методы с позиции точности получаемых результатов, несложно заметить, что у них разные среднеквадратические ошибки.

```
## R
> sort(rand.unemp[ , lapply(.SD, function(x) mean((x - unemp$UNRATE)^2,
>              na.rm = TRUE)),
>          .SDcols = c("impute.ff", "impute.rm.lookahead",
>                      "impute.rm.nolookahead", "impute.li",
>                      "impute.sp")])
impute.li  impute.rm.lookahead  impute.sp  impute.ff  impute.rm.nolookahead
0.0017     0.0019               0.0021     0.0056     0.0080
> sort(bias.unemp[ , lapply(.SD, function(x) mean((x - unemp$UNRATE)^2,
>               na.rm = TRUE)),
> .          SDcols = c("impute.ff", "impute.rm.lookahead",
>                      "impute.rm.nolookahead", "impute.li",
>                      "impute.sp")])
impute.sp  impute.li  impute.rm.lookahead  impute.rm.nolookahead  impute.ff
0.0012     0.0013     0.0017               0.0030                0.0052
```

Помните, что во многих рассмотренных выше методах возникает упреждение. Единственные методы, в которых оно не наблюдается, — замены пропущенного значения предыдущим и скользящего среднего без упреждения (в отличие от скользящего среднего с упреждением). Совершенно неудивительно, что они обладают разными среднеквадратичными ошибками, а методы без упреждения не столь точны, как остальные.

Последние замечания

Выше рассмотрены простейшие и наиболее часто используемые методы восполнения отсутствующих данных временны*х* рядов. Восполнение данных остается важной областью исследований дисциплины анализа данных. Чем важнее принимаемые решения, тем тщательнее нужно изучать возможные причины пропуска данных и последствия их восполнения. Ниже приведено несколько действенных советов по заполнению отсутствующих данных.

- Невозможно доказать, что в наборе отсутствуют действительно случайные данные, и тем более маловероятно, что их исключение из набора предопределяется естественными причинами.
- Иногда, но далеко не всегда, отсутствие отдельного измерения можно объяснить моделью. Широкие наборы данных со многими показателями — наилучшее средство для исследования закономерностей в пропущенных данных рассматриваемой модели. Тем не менее они не очень часто подвергаются анализу методами анализа временны*х* рядов.
- Чтобы понять причины и масштабы неопределенности, вызванной восполнением набора данных отсутствующими значениями, необходимо проанализировать несколько сценариев и пообщаться с как можно б*о*льшим количеством людей, вовлеченных в процесс сбора данных.
- Способ включения в набор недостающих данных должен учитывать их назначение. Будьте предельно внимательны и постарайтесь избежать упреждения. В противном случае определите, насколько сильно упреждение влияет на точность получаемых результатов.

Понижение и повышение частоты дискретизации

Очень часто связанные временны*е* ряды, полученные из разных источников данных, имеют неодинаковую частоту дискретизации. Это одна из многих причин, по которым вам придется изменять частоту дискретизации рядов данных, подвергаемых анализу. Разумеется, вы не можете изменить фактическую частоту сбора информации, но вам вполне по силам изменить частоту указания временны*х* меток в исходном наборе. Такая операция называется *понижением* или *повышением частоты дискретизации* временны*х* меток.

Нам уже доводилось сокращать временны*е* данные в разделе “Получение набора временны*х* рядов из таблиц”. Давайте еще раз рассмотрим эту задачу, но на этот раз предельно детально.

Понижение частоты дискретизации — это выбор подмножества данных, временны*е* метки которых имеют меньшую частоту, чем в исходном временно*м* ряду. Повышение частоты дискретизации позволяет получить наборы данных, которые якобы собирались чаще, чем на самом деле.

Понижение частоты дискретизации

Каждый раз, понижая частоту выборки данных, вы уменьшаете частоту дискретизации. Чаще всего это делается в следующих случаях.

Исходное разрешение данных слишком большое. Существует целый ряд причин, по которым высокая детализация данных не имеет смысла. Например, вы измеряете некий показатель чаще, чем того требует модель данных. Предположим, у вас есть набор данных о температуре воздуха, измеряемой каждую секунду. Исторический опыт (и здравый смысл) подсказывает, что измерения выполнялись слишком часто, и, скорее всего, набор данных включает слишком много повторяющейся информации, приводящей к дополнительной нагрузке на системы хранения и обработки данных. Более того, погрешность измерения может быть сопоставимой с посекундной вариацией температуры воздуха. В хранении чрезмерно больших и неинформативных наборов данных очень мало смысла. Уменьшение частоты дискретизации в таких наборах данных с регулярной выборкой сводится к выбору всех *n*-х элементов.

Выделение сезонных данных. Вместо того чтобы учитывать сезонность данных в общем временном ряду, образуйте частичный временной ряд, включающий данные только одного сезона. В следующем примере показан процесс понижения частоты дискретизации при образовании частичного временного ряда январских измерений, образованного из первоначального временного ряда ежемесячных данных. Как видите, теперь данные выбираются с годовой частотой.

```
## R
> unemp[seq.int(from = 1, to = nrow(unemp), by = 12)]
DATE         UNRATE
1948-01-01    3.4
1949-01-01    4.3
1950-01-01    6.5
1951-01-01    3.7
1952-01-01    3.2
1953-01-01    2.9
1954-01-01    4.9
1955-01-01    4.9
1956-01-01    4.0
1957-01-01    4.2
```

Сравнение с более низкочастотными данными. Уменьшить частоту дискретизации данных может понадобиться при сопоставлении их с данными с заметной меньшей частотой выборки. В подобных случаях данные сначала подвергаются агрегации и только затем из них удаляются определенные временные точки. Агрегация может заключаться в такой простой операции, как усреднение или суммирование, или же представлять более сложное действие, подобное вычислению взвешенного среднего, в котором измеряемая величина характеризуется другими весовыми значениями в разных временных точках. Ранее в наборе данных о добровольных взносах нам уже доводилось выполнять понедельное суммирование значений, поскольку именно такая общая сумма взносов представляет интерес для дальнейшего анализа.

Напротив, в наборах экономических данных интерес будут представлять среднегодовые значения. В следующем примере мы будем рассчитывать среднюю величину вместо скользящей средней, подчеркивая актуальность веса среднеарифметического, а не последнего годичного значения (обратите внимание на различия в восполнении данных). В процессе группировки даты приводятся к строковому формату — обработка значений года является прекрасным примером использования SQL-подобных команд для обработки временны*х* рядов.

```
## R
> unemp[, mean(UNRATE), by = format(DATE, "%Y")]
format   V1
1948    3.75
1949    6.05
1950    5.21
1951    3.28
1952    3.03
1953    2.93
1954    5.59
1955    4.37
1956    4.13
1957    4.30
```

Повышение частоты дискретизации

Повышение частоты выборки — это намного больше, чем операция, обратная понижению частоты дискретизации. Такое преобразование имеет отдельный смысл: в реальном мире уменьшение частоты измерений воспринимается вполне естественно, а вот повышение частоты выборки без увеличения количества измерений представляется безосновательной попыткой получения готового результата. Процитирую автора популярного пакета `XTS`, предназначенного для обработки временны*х* рядов, языка R (`https://perma.cc/83E9-4N79`).

> Невозможно преобразовать ряд низкой периодичности в ряд более высокой периодичности — например, еженедельные данные в ежедневные или ежедневные данные в 5-минутные, — поскольку это потребует магии.

Однако существуют законные способы размечать данные с большей частотой, чем принято по умолчанию. Достаточно помнить об ограничениях, которые необходимо при этом учитывать. Помните, что вы всего лишь увеличиваете количество временны*х* меток, а не пополняете ряд новыми данными.

Обсудим несколько ситуаций, когда повышение частоты дискретизации все же имеет здравый смысл.

Нерегулярные временны*е* ряды. Одна из распространенных причин повышать частоту дискретизации заключается в необходимости получения регулярного ряда данных из нерегулярного временно*г*о ряда. Такое преобразование

невозможно без увеличения частоты дискретизации, поскольку исходные данные конвертируются в данные, которые были собраны с заведомо большей частотой выборки. Если это ваш случай, то для повышения частоты дискретизации нужно воспользоваться функцией скользящего объединения языка R, как это делалось при заполнении отсутствующих значений экономического набора данных.

```
## R
> all.dates <- seq(from = unemp$DATE[1], to = tail(unemp$DATE, 1),
>                  by = "months")
> rand.unemp = rand.unemp[J(all.dates), roll=0]
```

Входные данные имеют разную частоту выборки. Иногда наряду с высокочастотными данными в модель нужно передать низкочастотную информацию, выровненную по частоте выборки исходных данных. Как и ранее, старайтесь избегать упреждения, но не забывайте, что, пока известные состояния остаются истинными и в данные не добавлены новые состояния, можно абсолютно безопасно увеличивать частоту выборки низкочастотных данных. Пусть нам абсолютно точно известно, что большинство вакантных рабочих мест появляется первого числа каждого месяца. Возникает соблазн использования значения месячного уровня безработицы, указанного в отчете, в качестве показателя *всего* месяца (это не будет упреждением, потому что уровень безработицы рассматривается неизменным в течение месяца).

```
## R
> daily.unemployment = unemp[J(all.dates), roll = 31]
> daily.unemployment
  DATE        UNRATE
1948-01-01     3.4
1948-01-02     3.4
1948-01-03     3.4
1948-01-04     3.4
1948-01-05     3.4
```

Понимание динамики изменения временны*х* рядов. Располагая базовыми познаниями о временно*м* поведении исследуемой величины, задачу увеличения частоты дискретизации можно свести к проблеме отсутствия данных. В этом случае к набору данных можно применять любые описанные ранее методы. Наиболее вероятным способом получения новых точек данных выступает интерполяция, но не забудьте убедиться в том, что выбранная модель данных полностью оправдывает такой выбор.

Как обсуждалось ранее, повышать или понижать частоту дискретизации может понадобиться даже в самом чистом наборе данных, поскольку эта операция выполняется почти всегда там, где приходится сравнивать показатели, изучаемые в разных временны*х* масштабах. Примите к сведению, что пакет `Pandas` обладает невероятно удобной функцией увеличения и уменьшения частоты дискретизации — методом `resample()`.

Сглаживание данных

Сглаживание данных выполняется по разным причинам, но чаще всего реальные временны*е* ряды сглаживаются перед непосредственным анализом, особенно перед визуализацией, призванной наглядно проиллюстрировать историю их изменения. В этом разделе мы обсудим причины сглаживания данных, а также изучим наиболее распространенный метод сглаживания временны*х* рядов — экспоненциальное сглаживание.

Цель сглаживания

Несмотря на то что обнаружение выбросов в данных является серьезной темой для рассмотрения, если у вас есть веские основания полагать, что данные требуют сглаживания, можете прибегнуть к методу скользящего среднего, чтобы исключить из набора резкие скачки показаний, ошибки измерения или и то, и другое вместе. Даже если скачки проявляются очень точно, они могут не относиться к измеряемой величине, а представлять ошибки измерительного оборудования. В таких случаях данные нужно подвергнуть обязательному сглаживанию.

Операция сглаживания данных тесно связана с задачей восполнения недостающих значений. Не удивительно, что в них обоих часто применяются одинаковые методы обработки данных. Например, вы можете сгладить данные по методу скользящего среднего как с упреждением, так и без него, — различие состоит только в выборе положения исследуемой точки относительно окна измерений, в котором рассчитывается сглаженное значение.

Перед тем как приступить к сглаживанию данных, ответьте на ряд вопросов.

- Зачем нужно сглаживание? Сглаживание может служить ряду целей.

  Подготовка данных

  Необработанные данные не подходят для дальнейшей обработки? Например, вы можете знать, что слишком высокие значения маловероятны или нарушают определенные физические законы, но вам нужно обязательно учитывать их в дальнейшем анализе. Сглаживание — самое простое решение.

  Получение данных признаков

  На практике часто приходится строить выборки, извлекаемые из общих данных множественных характеристик людей, изображений или других объектов, которые включают значения всего нескольких признаков. Таким образом, полный набор данных сжимается до нескольких измерений, включая данные только отдельных характеристик. Генерация признаков особенно важна в машинном обучении.

Прогнозирование

Простейшей формой прогнозирования в некоторых процессах является операция возврата к среднему, которая основана на составлении прогнозов для сглаженных данных признаков.

Визуализация

Хотите выявить сигнал в том, что представляется зашумленной диаграммой рассеяния? Если хотите, то как вы собираетесь это сделать?

- Как сглаживание или отсутствие сглаживания влияет на получаемые результаты?
  - — Включаются ли в модель зашумленные и некоррелированные данные, и как сглаживание помогает бороться с ними?
  - — Необходимо ли сглаживать данные модели, описывающей реальное производство? Если да, то нужно воспользоваться методом сглаживания, который не приводит к упреждению.
  - — Рассматриваете ли вы некие общие принципы сглаживания или каждый раз подбираете метод по набору гиперпараметров? В последнем случае как удостовериться в том, что используемая форма перекрестной проверки не позволяет будущим данным оказывать влияние на получаемые результаты?

Что такое гиперпараметры

Термин *гиперпараметры* звучит устрашающе, но на практике это числа, которые используются для настройки или установки точности статистической модели или модели машинного обучения. Чтобы оптимизировать модель, необходимо протестировать ее на нескольких наборах гиперпараметров, которые часто подбираются по своего рода сетке значений. Поиск по сетке заключается в построении всех возможных комбинаций гиперпараметров в пространстве значений и проверке всех возможных комбинаций.

Например, представим, что изучаемый алгоритм принимает гиперпараметры А и Б, о которых известно, что разумное значение параметра А равно 0, 1 или 2, а у параметра Б оно представлено значением 0,7, 0,75 или 0,8. Тогда в нашем распоряжении находится сетка из девяти возможных комбинаций гиперпараметров — сочетание каждого возможного значения параметра А (их три) с каждым возможным значением параметра Б (их тоже три) предоставляет в общей сложности $3 \cdot 3 = 9$ возможностей. Как видите, настройка гиперпараметров может быть очень трудоемкой задачей.

Экспоненциальное сглаживание

При сглаживании данных обрабатывать все временные точки одинаково приходится крайне редко. В частности, относительно недавние данные могут быть более информативными, и в этом случае экспоненциальное сглаживание является наиболее приемлемым решением. В отличие от скользящего среднего, которое мы рассматривали ранее, где каждая точка, в которой отсутствовали данные, могла быть восполнена средним значением окружающих ее точек, экспоненциальное сглаживание более чувствительно ко времени и придает относительно недавним точкам больший вес, чем более отдаленным по времени. Таким образом, для данного окна измерений ближайший момент времени оказывается наиболее весомым, а вес точки, находящейся на большем временном удалении, уменьшается по экспоненциальному закону (отсюда и название метода).

Экспоненциальное сглаживание работает следующим образом. Для заданного периода времени t сглаженное значение ряда данных S_t определяется по такой формуле:

$$S_t = d \cdot S_{t-1} + (1 - d) \cdot x_t.$$

Давайте представим, как изменяется ситуация с переходом к другой временной точке. Сглаженное значение в момент времени $t - 1$ рассчитывается по аналогичной формуле:

$$S_{t-1} = d \cdot S_{t-2} + (1 - d) \cdot x_{t-1}.$$

Подставив ее в исходную формулу, мы получим более сложное выражение вычисления сглаженного значения в момент времени t:

$$d \cdot (d \cdot S_{t-2} + (1 - d) \cdot x_{t-1}) + (1 - d) \cdot x_t.$$

Те из вас, кто имеет математическое образование, легко распознают в этой записи ряд вида

$$d^3 \cdot x_{t-3} + d^2 \cdot x_{t-2} + d \cdot x_{t-1}.$$

Фактически именно благодаря такой форме записи экспоненциальное скользящее среднее считается легко разрешимым методом. Более подробная информация широко доступна в Интернете и приведена в многочисленных учебниках (см. издания, упомянутые в конце главы).

Принципы сглаживания данных будут проиллюстрированы на примере, написанном на Python, поскольку его пакет `Pandas` включает один из самых широких наборов функций сглаживания. Функции сглаживания также широко применяются в языке R — они представлены как в стандартной библиотеке, так и в сторонних пакетах, специально предназначенных для обработки временных рядов.

Хотя рассматриваемые ранее примеры основывались на данных об уровнях безработицы в США, в текущем примере будет использован совершенно другой набор данных, описывающий пассажирские авиаперевозки (восходит к Боксу и широко известной книге Дженкинса о временны*х* рядах). Исходный набор данных представляет собой отчет о количестве пассажиров (тысяч человек), ежемесячно перевозимых авиакомпанией, представленный с помесячной разбивкой.

```
## Python
>>> air
        Date  Passengers
0    1949-01         112
1    1949-02         118
2    1949-03         132
3    1949-04         129
4    1949-05         121
5    1949-06         135
6    1949-07         148
7    1949-08         148
8    1949-09         136
9    1949-10         119
10   1949-11         104
```

Для сглаживания данных о количестве перевозимых пассажиров применяется функция `ewma()` пакета `Pandas`, в которой учитываются разные параметры затухания.

```
## Python
>>> air['Smooth.5'] = pd.ewma(air, alpha = .5).Passengers
>>> air['Smooth.9'] = pd.ewma(air, alpha = .9).Passengers
```

Легко заметить, что значение параметра `alpha`, определяющего *коэффициент сглаживания*, показывает, насколько сильно изменилось значение до его текущей величины по сравнению с вычисленным средним значением. Чем больше значение `alpha`, тем сильнее приближение к текущей величине. Заметьте, что функции сглаживания библиотеки `Pandas` принимают несколько параметров, подставляемых в одну и ту же расчетную формулу, но задающих несколько способов интерпретации экспоненциального скользящего среднего.[1]

```
## Python
>>> air
      Date   Passengers    Smooth.5    Smooth.9
0  1949-01         112  112.000000  112.000000
```

[1] В реляционных базах данных информация традиционно хранится в таблицах. Например, для представления сведений о транзакциях клиентов в такую базу данных обычно добавляются таблица с полной информацией о клиентах, включая уникальный идентификатор, и таблица с данными транзакций, в которой одно из полей отводится для хранения идентификаторов клиентов.

```
1  1949-02      118    116.000000    117.454545
2  1949-03      132    125.142857    130.558559
3  1949-04      129    127.200000    129.155716
4  1949-05      121    124.000000    121.815498
5  1949-06      135    129.587302    133.681562
6  1949-07      148    138.866142    146.568157
7  1949-08      148    143.450980    147.856816
8  1949-09      136    139.718200    137.185682
9  1949-10      119    129.348974    120.818568
10 1949-11      104    116.668295    105.681857
```

Однако в случае данных с долгосрочным трендом простое экспоненциальное сглаживание не позволяет добиться хороших результатов в прогнозировании. Сглаживание данных с ярко выраженным трендом, а также характеризующихся не только им, но и сезонностью, выполняется с помощью двух специальных экспоненциальных методов — Холта и Холта–Уинтерса.

Кроме того, существует много других не менее распространенных методов сглаживания. Например, фильтры Калмана сглаживают данные, представляя процессы временны*х* рядов в виде комбинации динамических состояний и погрешностей измерения. Или LOESS (Locally Estimated Scatter Plot Smoothing — сглаживание диаграммы рассеяния методом локальной оценки) — это непараметрический метод локального сглаживания данных. Эти и другие методы представляют более сложные способы сглаживания, но характеризуются заметно более высокой вычислительной сложностью. Примечательно, что в методе фильтрации Калмана и LOESS учитываются как предыдущие, так и последующие значения. Помните об этом, применяя эти методы в практических задачах — они точно не подойдут для прогнозирования временны*х* рядов.

Сглаживание является широко используемой формой прогнозирования, и вы можете использовать сглаженные временны*е* ряды (полученные без упреждения) для проверки того, стоит ли применять более сложные методы для получения точного прогноза.

Выбор начальной точки сглаживания

Одна из важных причин использования функций экспоненциального сглаживания, включенных в пакет `Pandas`, заключается в сложности выбора точки начала сглаженного ряда. Пусть первая точка в ряду данных представлена значением 3, а вторая — значением 6. Также пусть коэффициент сглаживания (который определяет весовой вклад текущего значения в среднее значение) равен 0,7, тогда вторая сглаженная точка ряда рассчитывается как

$$3 \cdot 0{,}7 + 6 \cdot (1 - 0{,}7) = 3{,}9.$$

На самом деле это неправильный результат. А все потому, что при умножении первого значения (3) на коэффициент сглаживания (0,7) неявно предполагается, что число 3 включает весовые вклады всех предыдущих значений, уходящих в бесконечность, а не просто еще одну только что измеренную точку данных. Следовательно, используемый нами коэффициент сглаживания приписывает значению 3 излишне большой весовой вклад относительно значения 6 так, если бы число 3 представляло очень длинный набор других значений, чего на самом деле не наблюдается.

Методика определения точной вычислительной формулы слишком сложна для такой короткой врезки, но вы можете поискать подробную информацию о ней в дополнительных источниках, приведенных в конце этой книги. На данный момент я ограничусь лишь общим замечанием о том, что числу 6 следует приписать несколько больший вес, а числу 3 — немного меньший, чем тот, который задан выше. По мере прохода по временному ряду старые значения будут более точно соответствовать фактическому коэффициенту дисконтирования 0,7. Насколько быстро это будет проявляться, можно узнать из дополнительных источников, описанных в конце раздела.

Сезонные данные

Сезонность проявляется в данных как повторяющееся поведение, характеризуемое постоянной частотой. Такое поведение может описываться одновременно несколькими частотами. Например, человеческое поведение имеет выраженную дневную (завтрак, обед и ужин в одно и то же время каждый день), недельную (понедельник — день тяжелый) и годичную (Новый год — выходной) сезонности. Физические системы также обладают заметной сезонностью, например Земля делает оборот вокруг Солнца за строго определенный период времени (год).

Распознавание и работа с сезонными данными является неотъемлемой частью процесса моделирования. С другой стороны, обработка сезонных данных выступает одной из форм очистки данных, как показано в докладе об уровне безработицы в США (`https://perma.cc/GX6J-QJG9`). Действительно, многие правительственные статистические данные, в частности экономического характера, очищаются от сезонной составляющей перед публикацией в официальных источниках.

Чтобы увидеть, к чему приводит сглаживание сезонных данных, вернемся к рассмотрению привычного набора данных о количестве пассажиров, перевозимых авиакомпанией. Даже быстрое изучение данных наталкивает на мысль о том, что такие данные — сезонные. Чтобы убедиться в этом, достаточно построить правильный график.

Обратите внимание на разницу между графиком, построенным с помощью стандартной функции языка R с настройками по умолчанию (точки данных обозначены кружочками; рис. 2.10), и им же, но полученным при передаче функции специального аргумента (точки данных соединены линиями; рис. 2.11).

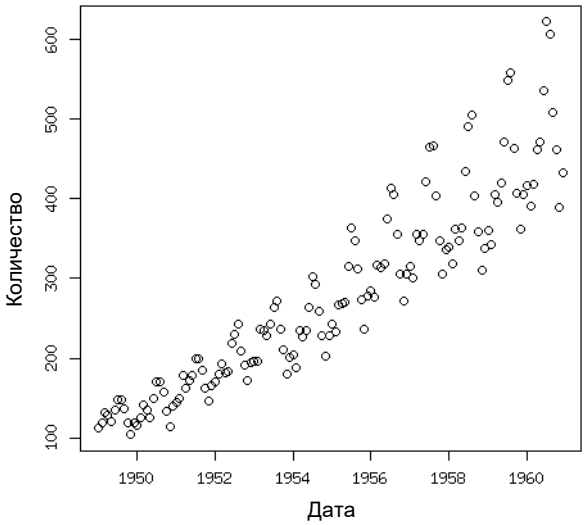


Рис. 2.10. *Постоянно возрастающее среднее и большая дисперсия данных не позволяют рассмотреть сезонность изменения данных на диаграмме рассеяния*

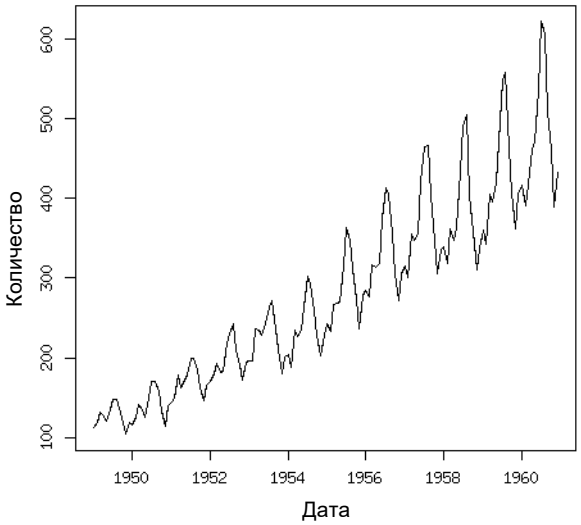


Рис. 2.11. *Линейный график демонстрирует сезонность данных как нельзя лучше*

Изучая график, построенный на языке R с настройками по умолчанию, вы вряд ли сможете обнаружить сезонную направленность в данных. Вне всякого сомнения, вы сделаете это после полного анализа данных, возможно, при построении графика автокорреляции (см. главу 3) или при проведении других исследований.

Люди — заложники привычек

Человеческую жизнь также можно описать сезонными поведенческими шаблонами разной периодичности (часовой, недельный, лето-зима и т.д.).

Диаграмма рассеяния действительно предоставляет определенную информацию в более понятной, чем линейный график, форме. В нашем случае на ней прекрасно просматривается постоянный рост дисперсии и среднего значения, представленных расширяющимся облаком точек данных, устремленным к правому верхнему углу области построения. Данные имеют выраженный тренд, поэтому, скорее всего, будут подвержены логарифмическому или разностному преобразованию в зависимости от требований избранной модели. Такие наборы данных также характеризуются тенденцией к постоянному росту дисперсии. Подробно о преобразовании сезонных данных в рамках выбранной модели мы поговорим в последующих главах, посвященных непосредственно моделированию, и не будем больше возвращаться к этому вопросу в настоящей главе.

Из линейного графика можно узнать не только о сезонности данных, но и о *типе* такой сезонности. На линейном графике четко просматривается не только сезонность изменений данных, но и мультипликативность этого эффекта. По мере увеличения средних значений сезонные колебания (расстояние от пика до впадины) увеличиваются.

Исходные данные легко разложить на сезонную, трендовую и остаточную составляющие с помощью единственной команды языка R, как показано ниже.

```
## R
> plot(stl(AirPassengers, "periodic"))
```

Полученный график предельно точно отражает исходные данные (рис. 2.12). Чтобы получить исходный ряд, достаточно объединить данные сезонной, трендовой и остаточной составляющих. Заметьте, что в данном разложении учитывается аддитивность, а не мультипликативность сезонных изменений данных, поскольку остаточная компонента вносит наибольший вклад в начале и конце временного ряда. Похоже на то, что график сезонной составляющей, показанный на рис. 2.12, отображает изменения не абсолютной, а средней сезонной дисперсии.

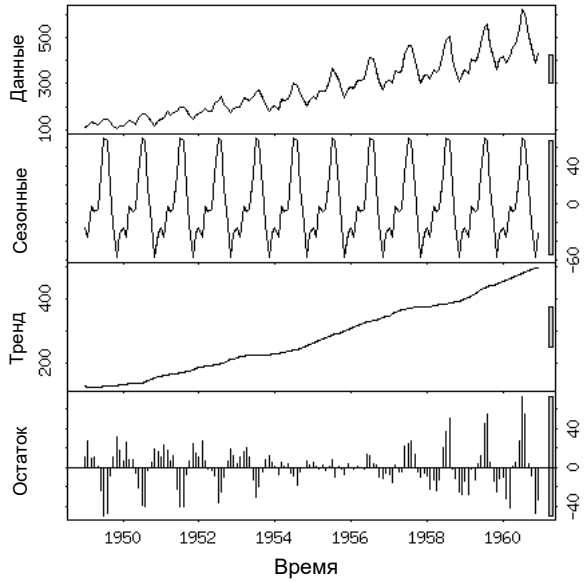


Рис. 2.12. *Разложение исходного временного ряда на сезонную, трендовую и остаточную составляющие. Заметьте, что на осях y графиков откладываются совершенно разные величины. Также обратите внимание на серые столбцы у правого края каждого графика. Эти столбцы характеризуются одинаковым абсолютным размером (в единицах измерения соответствующих осей y) и указывают на различия в масштабировании отдельных составляющих*

Чтобы получить базовое представление о том, как выполняется такое разложение, обратимся к официальной документации языка R.

> Сезонная составляющая выделяется с помощью функции LOESS, сглаживающей сезонные подряды данных (ряд значений для января...). При установке параметра `s.window = "periodic"` сглаживание заменяется операцией вычисления среднего значения. Для нахождения тренда сезонная составляющая удаляется, а оставшиеся данные сглаживаются. Из сезонной составляющей удаляется общий уровень, а полученный остаток добавляется к трендовой составляющей. Этот процесс повторяется несколько раз. Остаточный ряд данных представлен значениями, полученными при вычитании сезонных и трендовых компонент из исходных данных.

Упомянутая выше функция LOESS реализует необычайно сложный вычислительный метод сглаживания точек данных, основанный на динамическом изменении положения измерительного окна при определении сглаженного значения каждой точки данных. (Я надеюсь, вы осознаете высокую вероятность возникновения упреждения при использовании этой функции!)

Сезонные и циклические данные

Сезонные временны*е* ряды — это временны*е* ряды, в которых поведение данных повторяется в течение фиксированного периода времени. Существует несколько типов сезонности, характеризующихся разными периодами измерения данных, таких как 24-часовая (суточная) сезонность или 12-месячная (годичная) сезонность, которые проявляются в большинстве временны*х* рядов, описывающих человеческое поведение.

Циклические временны*е* ряды также описывают повторяющееся поведение, но переменной периодичности. Типичным примером будет циклическая финансовая активность, включающая периоды роста и спада фондового рынка, которые имеют неопределенную продолжительность. Точно так же циклическое, но не сезонное поведение демонстрируют вулканы. Мы умеем прогнозировать приблизительные периоды их активности, но не очень точно, и они постоянно изменяются с течением времени.

Часовые пояса

Часовые пояса — обременительный, трудноразрешимый и неустранимый фактор, который приходится учитывать в ходе анализа временны*х* рядов. Вот почему вы никогда не должны использовать собственное решение. С момента изобретения часовые пояса только то и делают, что усложняют обработку временны*х* данных, а с появлением персональных компьютеров таких сложностей стало несравнимо больше. На то существует много причин.

- Положение часовых поясов определяется политическими и социальными факторами.
- Не существует единого способа передачи информации о часовом поясе между разными языками программирования или по протоколу HTTP.
- Не существует единого протокола именования часовых поясов или определения даты начала и окончания перехода на летнее время.
- Переход на летнее время в некоторых часовых поясах иногда происходит дважды в год!

Большинство языков программирования получают информацию о часовом поясе от операционной системы. К сожалению, встроенная функция автоматического определения даты и времени в языке Python, `datetime.now()`, не возвращает временну*ю* метку, учитывающую часовой пояс. Отчасти это сделано преднамеренно. Некоторые решения, принятые в стандартной библиотеке, поддерживают запрет на обработку информации о часовом поясе, относящейся к модулю

datetime (такая информация изменяется очень часто), и объекты datetime как с включенной информацией о часовом поясе, так и без нее. При этом сравнение объектов времени с данными часового пояса и без них вызовет ошибку TypeError.

Некоторые исследователи утверждают, что большинство программных библиотек Python написаны в предположении, что tzinfo == None. Несмотря на недоказуемость, скорее всего, это действительно так. Многие также сообщают о сложностях сериализации объектов, снабженных временными метками с указанием часового пояса, поэтому, если вы планируете подвергать ей свои данные, заранее удостоверьтесь в справедливости или невозможности выполнения этой операции.[2]

Давайте посмотрим, как обрабатываются данные часовых поясов в языке Python. Чаще всего для этой цели применяются инструменты библиотек datetime, pytz и dateutil. Кроме того, многие удобные функции управления объектами с данными о часовых поясах содержатся в библиотеке Pandas — они основаны на средствах последних двух библиотек.

Ниже рассмотрены только наиболее важные функции управления данными часовых поясов.

Во-первых, примите к сведению, что функция now() модуля datetime не предоставляет информацию о часовом поясе, хотя и учитывает его при определении (правильного) времени. Также обратите внимание на различия в данных, возвращаемых функциями now() и utcnow().

```
## Python
>>> datetime.datetime.utcnow()
datetime.datetime(2018, 5, 31, 14, 49, 43, 187680)

>>> datetime.datetime.now()
datetime.datetime(2018, 5, 31, 10, 49, 59, 984947)
>>> # Компьютер не возвращает время в формате UTC,
>>> # хотя часовой пояс не указан

>>> datetime.datetime.now(datetime.timezone.utc)
datetime.datetime(2018, 5, 31, 14, 51, 35, 601355,
                  tzinfo=datetime.timezone.utc)
```

Чтобы получить правильный результат, достаточно перейти к формату с указанием часового пояса, но это нестандартное поведение. Для управления часовыми поясами в языке Python создается объект часового пояса, например western, представляющий тихоокеанский часовой пояс США.

[2] Вы можете установить такие параметры, как коэффициент сглаживания, полупериод, интервал или центр масс. Подробности приведены в официальной документации (https://perma.cc/4265-4U8L).

```
## Python
>>> western = pytz.timezone('US/Pacific')
>>> western.zone
'US/Pacific'
```

Впоследствии такие объекты применяются для преобразования часового пояса в данные с помощью функции `localize()`.

```
## Python
>>> ## API предоставляет два способа вычисления времени с учетом
>>> ## часового пояса — "локализацию" и приведение времени к формату
>>> ## другого часового пояса.
>>> # В нашем случае выполняется "локализация"
>>> loc_dt = western.localize(datetime.datetime(2018, 5, 15, 12, 34, 0))
datetime.datetime(2018, 5, 15, 12, 34,
          tzinfo=<DstTzInfo 'US/Pacific' PDT-1 day, 17:00:00 DST>)
```

Однако не забывайте, что передача часового пояса непосредственно в конструктор `datetime()` часто приводит к совершенно неожиданному результату.

```
## Python
>>> london_tz = pytz.timezone('Europe/London')
>>> london_dt = loc_dt.astimezone(london_tz)

>>> london_dt
datetime.datetime(2018, 5, 15, 20, 34,
tzinfo=<DstTzInfo 'Europe/London' BST+1:00:00 DST>)

>>> f = '%Y-%m-%d %H:%M:%S %Z%z'
>>> datetime.datetime(2018, 5, 12, 12, 15, 0,
                      tzinfo = london_tz).strftime(f)
'2018-05-12 12:15:00 LMT-0001'
>>> ## Как подчеркивается в документации к модулю pytz,
>>> ## задание параметра tzinfo в функции инициализации
>>> ## datetime.datetime не всегда приводит к получению желаемого
>>> ## результата, как в случае г. Лондона.
>>> ## Согласно документации к библиотеке pytz, метод работает только
>>> ## в данных с часовыми поясами без перехода на летнее время
```

Это важно учитывать, например, при вычислении разницы между двумя моментами времени. Первый пример из следующих трех описывает поведение программы в случае сбоя.

```
## Python
>>> # Как правило, данные времени хранятся в формате UTC
>>> # и преобразовываются в другие форматы только при выводе
>>> # для изучения людьми. Кроме того, к часовым поясам часто
>>> # применяются арифметические операции
>>> event1 = datetime.datetime(2018, 5, 12, 12, 15, 0,
                               tzinfo = london_tz)
>>> event2 = datetime.datetime(2018, 5, 13, 9, 15, 0,
                               tzinfo = western)
```

```
>>> event2 - event1
>>> ## Разница между моментами времени определяется неправильно,
>>> ## поскольку часовые пояса помечены не так, как нужно
>>> event1 = london_tz.localize(
                datetime.datetime(2018, 5, 12, 12, 15, 0))
>>> event2 = western.localize(
                datetime.datetime(2018, 5, 13, 9, 15, 0))
>>> event2 - event1
>>> event1 = london_tz.localize(
                (datetime.datetime(2018, 5, 12, 12, 15, 0))).
                    astimezone(datetime.timezone.utc)
>>> event2 = western.localize(
                    datetime.datetime(2018, 5, 13, 9, 15, 0)).
                    astimezone(datetime.timezone.utc)
>>> event2 - event1
```

Библиотека `pytz` включает списки наиболее распространенных часовых поясов и часовых поясов разных стран мира.

```
## Python
## Ознакомьтесь основными часовыми поясами pytz.common_timezones
>>> pytz.common_timezones
(длинный список...)
## или часовыми поясами стран мира
>>> pytz.country_timezones('RU')
['Europe/Kaliningrad', 'Europe/Moscow', 'Europe/Simferopol',
'Europe/Volgograd', 'Europe/Kirov', 'Europe/Astrakhan',
'Europe/Saratov', 'Europe/Ulyanovsk', 'Europe/Samara',
'Asia/Yekaterinburg', 'Asia/Omsk', 'Asia/Novosibirsk',
'Asia/Barnaul', 'Asia/Tomsk', 'Asia/Novokuznetsk',
'Asia/Krasnoyarsk', 'Asia/Irkutsk', 'Asia/Chita',
'Asia/Yakutsk', 'Asia/Khandyga', 'Asia/Vladivostok',
'Asia/Ust-Nera', 'Asia/Magadan', 'Asia/Sakhalin',
'Asia/Srednekolymsk', 'Asia/Kamchatka', 'Asia/Anadyr']
>>>
>>> pytz.country_timezones('fr')
>>> ['Europe/Paris']]
```

Особенно проблематичным является вопрос учета летнего времени. Некоторые используемые человечеством форматы времени учитываются компьютером дважды (возврат к “зимнему” времени осенью), тогда как другие вообще не учитываются (перевод времени вперед весной).

```
## Python
>>> ## Часовые пояса
>>> ambig_time = western.localize(
                    datetime.datetime(2002, 10, 27, 1, 30, 00)).
                       astimezone(datetime.timezone.utc)
>>> ambig_time_earlier = ambig_time - datetime.timedelta(hours=1)
>>> ambig_time_later = ambig_time + datetime.timedelta(hours=1)
```

```
>>> ambig_time_earlier.astimezone(western)
>>> ambig_time.astimezone(western)
>>> ambig_time_later.astimezone(western)

>>> # Вывод результатов
datetime.datetime(2002, 10, 27, 1, 30,
            tzinfo=<DstTzInfo 'US/Pacific' PDT-1 day, 17:00:00 DST>)
datetime.datetime(2002, 10, 27, 1, 30,
            tzinfo=<DstTzInfo 'US/Pacific' PST-1 day, 16:00:00 STD>)
datetime.datetime(2002, 10, 27, 2, 30,
            tzinfo=<DstTzInfo 'US/Pacific' PST-1 day, 16:00:00 STD>)
>>> # Последние две временные метки идентичны — это недопустимо!

>>> ## В данном случае необходимо использовать параметр is_dst, чтобы
>>> ## обозначить переход на летнее время
>>> ambig_time = western.localize(
       datetime.datetime(2002, 10, 27, 1, 30, 00), is_dst = True).
          astimezone(datetime.timezone.utc)
>>> ambig_time_earlier = ambig_time - datetime.timedelta(hours=1)
>>> ambig_time_later = ambig_time + datetime.timedelta(hours=1)
>>> ambig_time_earlier.astimezone(western)
>>> ambig_time.astimezone(western)
>>> ambig_time_later.astimezone(western)

datetime.datetime(2002, 10, 27, 0, 30,
            tzinfo=<DstTzInfo 'US/Pacific' PDT-1 day, 17:00:00 DST>)
datetime.datetime(2002, 10, 27, 1, 30,
            tzinfo=<DstTzInfo 'US/Pacific' PDT-1 day, 17:00:00 DST>)
datetime.datetime(2002, 10, 27, 1, 30,
            tzinfo=<DstTzInfo 'US/Pacific' PST-1 day, 16:00:00 STD>)
## Повторяющиеся временные метки отсутствуют, по крайней мере
## до момента учета смещения относительно UTC
```

Трудности учета часового пояса могут не касаться решаемых вами задач, поэтому полезность приведенных выше сведений будет зависеть от характера обрабатываемых данных. Безусловно, существуют ситуации, в которых ошибка равнозначна катастрофе (скажем, экипажу дальномагистрального авиалайнера, который совершает рейс, проходящий временную точку перехода на летнее время, будет крайне неприятно осознавать, что в переданном прогнозе погоды такое событие никак не учитывается).

Предотвращение упреждения

Упреждение включается в процесс моделирования очень легко, особенно с помощью векторизованных интерфейсов функциональной обработки данных, написанных на языках R и Python. Достаточно сместить рассматриваемое значение

в неправильном временном направлении, сместить его дальше или ближе, чем необходимо, или получить не совсем "честные" результаты неким иным способом, предполагающим принятие в расчет информации, которая стала доступной раньше, чем должна учитываться моделью данных.

К сожалению, не существует единого критерия, позволяющего однозначно обнаруживать упреждения, — в конце концов, все усилия по анализу временны*х* рядов сводятся к моделированию неизвестных ситуаций. В детерминированных системах, подчиняющихся строго заданным динамическим законам, отличить очень хорошую модель от модели с упреждением очень трудно — до тех пор, пока модель не будет запущена в обработку, вы не поймете, что в ней отсутствуют данные или получаемые результаты не отражают того, что наблюдалось во время обучения.

Лучший способ предотвратить неприятные ситуации — постоянно быть бдительным. Всякий раз, перемещая во времени, сглаживая, восполняя данные и повышая частоту их дискретизации, спрашивайте себя, знаете ли вы заранее что-либо о рассматриваемом моменте времени. Не забывайте, что работаете не столько с временны*ми* точками, сколько с временны*ми* диапазонами. Вам нужно научиться определять действительные временны*е* задержки между рассматриваемыми событиями и моментами времени получения данных об этом событии. Например, если ваша организация занимается еженедельным скреппингом Twitter с целью сбора данных о потребительских настроениях, то вам необходимо принимать в расчет такую периодичность при построении наборов данных для обучения и тестирования. Аналогично, если вы переобучаете модели не чаще раза в месяц, обязательно выясните, какие временны*е* модели должны применяться к собранным наборам данным. Например, вы не можете обучить модель на июльских данных, а затем протестировать ее на июльских данных, потому что в реальной ситуации вы не сможете обучить эту модель вовремя, поскольку обучение отнимает много времени.

Ниже приведен список замечаний, призванных помочь в получении правильных наборов данных. Принимайте их к сведению как при построении модели данных, так и при последующем ее анализе.

- При сглаживании или восполнении отсутствующих данных обязательно выясните, как именно выполняется целевая операция — не приводит ли она к упреждению. И не только выясните, но и проверьте — всесторонне протестируйте ее, как мы делали это ранее, на разных наборах данных и внимательно изучите полученные восполненные и сглаженные временны*е* ряды. Можно ли считать, что они лишены упреждения? Если нельзя, будет ли оправданным их практическое использование?
- Постройте весь процесс на основе небольшого набора данных (объекта `data.table`, состоящего из нескольких строк, или нескольких записей

с небольшим количеством временных точек в любом другом формате). Проведите выборочные проверки на каждом этапе и выясните, не была ли перенесена важная информация в неподходящее для нее место.

- Определите задержку получения каждого типа данных относительно собственной временной метки. Например, если временная метка указывает время измерения данных, а не их загрузки на сервер, обязательно учтите это в последующем анализе. Разные столбцы одного фрейма данных могут характеризоваться разными задержками. Чтобы решить эту проблему, можете либо выровнять задержки отдельно в каждом фрейме данных, либо (лучше и более практично) положиться на самую большую задержку, применив ее сразу ко всем данным. Несмотря на то что чрезмерное упрощение модели не сулит ничего хорошего, столь радикальный шаг послужит хорошей отправной точкой на пути к пошаговой отмене ограничений!
- Проверяйте данные на наличие ошибок учета времени и выполняйте перекрестные проверки. Этот метод будет обсуждаться в главе 11, но помните, что рандомизация обучающих наборов и наборов для тестирования в случае временных рядов не работает должным образом. Будущая информация не должна использоваться в моделях, основанных на данных из прошлого.
- Преднамеренно внесите упреждение в модель и проследите за ее поведением. Протестируйте модель на упреждении разных уровней сложности, чтобы получить представление о его влиянии на точность получаемых результатов. Определившись с точностью модели, вы сможете понять, как она будет вести себя в реальных условиях — лишенная каких бы то ни было будущих данных. Помните, что многие задачи анализа временных рядов чрезвычайно сложны для решения, и модель с упреждением может казаться великолепным инструментом до тех пор, пока вы не поймете, что имеете дело с набором сильно зашумленных данных или данных с низким уровнем сигнала.
- Понемногу выделяйте признаки, особенно подлежащие дальнейшей обработке. Один из симптомов упреждения проявляется в том, что какой-то признак оказывается неожиданно хорошим, и тому нет достойного объяснения. В первых строках списка причин такого поведения должно значиться: "упреждение".

Обработка и очистка данных, снабженных временными метками, может быть утомительным и скрупулезным занятием.

Наибольшую опасность, подстерегающую вас при очистке и обработке данных, представляет упреждение! Оно допустимо только в случаях умышленного включения в модель будущих данных, но такие ситуации встречаются очень редко.

Дополнительные источники

- Недостающие данные

  Steffen Moritz et al., "Comparison of Different Methods for Univariate Time Series Imputation in R," unpublished research paper, October 13, 2015, `https://perma.cc/M4LJ-2DFB`

  В этом подробном отчете за 2015 год рассмотрены актуальные методы восполнения одномерных временных рядов. В отдельную категорию задач выделены однофакторные временные ряды, поскольку многие современные методы их восполнения основаны на изучении распределения ковариат, недоступных в одномерных временных рядах. В статье описаны преимущества использования и производительность основных пакетов R, а также эмпирические результаты исследований, полученные на различных наборах данных.

  James Honaker and Gary King, "What to Do About Missing Values in Time-Series Cross-Section Data," American Journal of Political Science 54, no. 2 (2010): 561–81, `https://perma.cc/8ZLG-SMSX`

  Здесь рассматриваются методологические рекомендации по обработке пропущенных данных во временных рядах с широкими наборами ковариат.

  Leo Belzile, "Notes on Irregular Time Series and Missing Values," n.d. `https://perma.cc/8LHP-92FP`

  Автор работы рассматривает задачи восполнения недостающих данных на примере нерегулярных наборов данных и приводит обзор наиболее популярных пакетов R.

- Часовые пояса

  Tom Scott, The Problem with Time & Timezones, Computerphile video, December 30, 2013, `https://oreil.ly/iKHkp`

  В этом 10-минутном видео на YouTube, насчитывающем более 1,5 миллиона просмотров, изложены основные трудности принятия в расчет часовых поясов при обработке временных данных, особенно в контексте веб-приложений.

  Wikipedia, "Time Zone," `https://perma.cc/J6PB-232C`

  Короткая захватывающая история об истории учета времени человечеством до начала прошлого столетия и о потребности в координировании мирового времени по мере развития технологий (начиная с эпохи паровых двигателей). Здесь также приведено несколько забавных карт часовых поясов.

Declan Butler, "GPS Glitch Threatens Thousands of Scientific Instruments," Nature, April 3, 2019, `https://perma.cc/RPT6-AQBC`

Статья не имеет непосредственного отношения к часовым поясам, но описывает общие трудности снабжения данных временны*ми* метками. В ней рассмотрена одна из последних проблем такого рода, причиной которой стал сбой в системе глобального позиционирования. Система снабжала данные временны*ми* метками, номер недели в которых представлялся двоичными 10-значными номерами с началом отсчета 6 января 1980 года. Такая учетная система охватывает только 1024 недели (2 в степени 10). Во второй раз предел был превышен в апреле 2019 года. Устройства, которые были не способны обходить такое ограничение, сбрасывались в начальную точку отсчета и генерировали неверные операционные и научные данные. В статье описываются основные трудности регистрации временны*х* рядов научным оборудованием, исходно не рассчитанным на передачу данных в ограниченных временны*х* форматах.

- Сглаженные и сезонные данные

Rob J. Hyndman and George Athanasopoulos, "Exponential Smoothing," in Forecasting: Principles and Practices, 2nd ed. (Melbourne: OTexts, 2018), `https://perma.cc/UX4K-2V5N`

В этой обзорной статье к академическому учебнику Хиндмана и Афанасопулоса рассматриваются основные методы экспоненциального сглаживания временны*х* рядов, приведена их классификация и описаны расширения экспоненциального сглаживания в приложениях прогнозирования.

David Owen, "The Correct Way to Start an Exponential Moving Average," Forward Motion blog, January 31, 2017, `https://perma.cc/ZPJ4-DJJK`

Как подчеркивалось ранее в одном из замечаний, экспоненциальное скользящее среднее вычисляется относительно просто, если абстрагироваться от проблематики выбора начальной точки. Нам нужно убедиться в том, что в скользящем среднем учитывается новая информация — при его вычислении должен рассматриваться как можно более полный временно*й* диапазон сбора данных. Если мы не будем принимать во внимание это обстоятельство, то в последующих вычислениях получим скользящее среднее, отражающее "бесконечность" предыдущего набора точек данных и недооценивающее новую информацию. Подробнее об этом и о правильных вычислительных решениях можно узнать в этом блоге.

Avner Abrami, Aleksandr Arovkin, and Younghun Kim, "Time Series Using Exponential Smoothing Cells," unpublished research paper, last revised September 29, 2017, `https://perma.cc/2JRX-K2JZ`

Это написанная простым языком статья об анализе временных рядов, в которой описывается идея применения простого экспоненциального сглаживания для получения "ячеек экспоненциального сглаживания".

- Анализ функциональных данных

Jane-Ling Wang, Jeng-Min Chiou, and Hans-Georg Muller, "Review of Functional Data Analysis," Annual Reviews of Statistics and its Application, 2015, `https://perma.cc/3DNT-J9EZ`

В этой статье, посвященной статистическим вычислениям, приведен краткий обзор наиболее важных математических методов анализа функциональных данных. В ней также рассматриваются популярные инструменты визуализации получаемых результатов.

Shahid Ullah and Caroline F. Finch, "Applications of Functional Data Analysis: A Systematic Review," BMC Medical Research Methodologyv, 13, no. 43 (2013), `https://perma.cc/VGK5-ZEUX`

В этом обзоре применяется междисциплинарный подход к исследованию недавно опубликованных функциональных данных, подвергнутых предварительному анализу. Авторы утверждают, что методы анализа функциональных данных имеют гораздо более широкое применение, чем принято считать, и приводят аргументы в пользу того, что биологические и медицинские науки могли бы выиграть от более широкого использования таких методов при изучении собираемых ими временных рядов.

ГЛАВА 3

Методы исследования временны*х* рядов

Исследовательским методам анализа временны*х* рядов посвящены два раздела. В первом из них мы будем рассматривать общие инструменты анализа данных, часто применяемые для исследования временны*х* рядов. В частности, вы узнаете об изучении временны*х* рядов с помощью гистограмм, графиков и функций группировки.

Второй раздел посвящен специальным методам анализа временных рядов — разработанным исключительно для обработки временны*х* рядов и применимых только к данным с временно*й* связью (в отличие от данных с перекрестными связями).

Общие методы исследования временны*х* рядов

Свои изыскания начнем с изучения принципов адаптации наиболее распространенных методов анализа данных к временны*м* рядам. Этот процесс аналогичен обработке данных, лишенных временно*й* зависимости.

Вам предстоит научиться распознавать столбцы, диапазоны значений и логические взаимосвязи между ними. В процессе анализа временны*х* рядов вам нужно будет ответить на такие же вопросы, как и при исследовании любого другого набора данных.

- Существуют ли в наборе данных взаимосвязанные столбцы?
- Каково среднее значение изучаемой величины? Какова его дисперсия?

Чтобы ответить на них, вы можете обратиться к общеизвестным методам и инструментам — графикам, сводным статистикам, гистограммам и диаграммам рассеяния. Кроме того, необходимо предельно точно определить временны*е* характеристики набора данных, ответив на следующие вопросы.

- Изменяется ли диапазон доступных значений с изменением временно*го* периода или другой подвергаемой анализу характеристики?
- Являются ли данные однородными и логически взаимосвязанными или же предполагают временны*е* изменения в способах измерения или в поведении?

Чтобы правильно ответить на поставленные вопросы, в ходе анализа данных с помощью упомянутых выше инструментов — гистограмм, диаграмм рассеяния и сводных статистик — нужно обязательно учитывать временны*е* изменения исследуемых величин. Следовательно, в дальнейших статистических вычислениях время должно быть представлено отдельной осью на графиках и группой — в групповых операциях — на гистограммах и диаграммах рассеяния.

Оставшуюся часть раздела мы посвятим нескольким исследовательским методам анализа временны*х* рядов. В процессе дальнейших изысканий мы постараемся определить, можно ли применять к временны*м* рядам традиционные, не учитывающие временны*е* изменения в данных, аналитические методы напрямую или только после соответствующей модификации. Для более наглядной демонстрации будем исследовать целевые методы на данных европейских фондовых рынков, получаемых с помощью базовых инструментов языка R.

Операции группировки

Во временны*х* рядах, как и в данных, лишенных временны*х* связей, можно выделить отдельные группы значений, поэтому перед анализом такие данные можно подвергнуть самым разным операциям группировки. Например, в перекрестных данных операции группировки позволяют определить средние значения для значений возраста, пола или места проживания респондентов. В анализе временны*х* рядов востребованы групповые операции, позволяющие рассчитывать основные статистические показатели наборов, например среднемесячные или недельные медианы. Более того, правильно проведенная группировка значений поможет проанализировать наборы данных, обладающих одновременно не только временны*ми*, но и другими типами связей. Например, анализируя медицинский набор данных, вы сможете определить такие важные характеристики, как среднемесячное количество калорий, потребленных пациентами обоих полов, а также среднее за неделю время сна в каждой возрастной группе. Существует большое количество способов использования операций группировки данных для определения связей временны*х* рядов с остальными данными набора.

- Вы можете анализировать данные разных периодов отдельно, если знаете, что между ними нет однозначных зависимостей. Применяйте исследовательские методики для определения таких разрозненных временны*х* периодов в исходном наборе данных.
- Данные одного временно*г*о ряда (в частности, о поступлении добровольных взносов) зачастую становятся более понятными при разделении на несколько параллельных временны*х* рядов (например, о поступлении взносов от отдельных респондентов).

Построение графиков

Рассмотрим типичный временной ряд, представленный в языке R набором данных `EuStockMarkets`. Чтобы понять его, внимательно изучим заголовок набора данных.

```
## R
> head(EuStockMarkets)
         DAX  SMI    CAC    FTSE
[1,] 1628.75 1678.1 1772.8 2443.6
[2,] 1613.63 1688.5 1750.5 2460.2
[3,] 1606.51 1678.6 1718.0 2448.2
[4,] 1621.04 1684.1 1708.1 2470.4
[5,] 1618.16 1686.6 1723.1 2484.7
[6,] 1610.61 1671.6 1714.3 2466.8
```

Это встроенный набор данных, относящийся к стандартной библиотеке языка R, который содержит сведения о ежедневных ценах закрытия четырех основных европейских фондовых индексов, указанных с 1991 по 1998 год. Значения в нем приведены только для торговых (биржевых) дней.

Этот набор данных подготовлен лучше, чем все рассматриваемые ранее, — он правильно собран и отформатирован. Нам не нужно беспокоиться о пропущенных значениях, часовых поясах или неправильных измерениях; можем сразу же переходить к исследованию данных.

Наши первые шаги по изучению данных будут такими же, как при анализе рядов, лишенных выраженной временной зависимости, хотя ничто не запрещает рассматривать временные ряды как перекрестные данные, обладающие временными связями. Чтобы получить представление об исследуемых величинах, представим их на отдельных графиках. Сравните такой подход с принятым при визуализации перекрестных данных, в котором каждый отдельный график представляет зависимость рассматриваемого признака от индекса, указывающего только порядок сбора или получения значений с сервера (иначе говоря, произвольный порядок), но не предоставляющий сведений о самих данных. Однако в случае временных рядов графики (рис. 3.1) становятся весьма информативным инструментом анализа данных.

```
## R
> plot(EuStockMarkets)
```

Обратите внимание на то, что перед визуализацией с помощью простой команды `plot()` данные автоматически разделяются на несколько временных рядов. И все благодаря использованию в языке R встроенного объекта `mts` (в противоположность объекту `ts`, обеспечивающего визуализацию только одного временного ряда).

```
## R
> class(EuStockMarkets)
[1] "mts"    "ts"     "matrix"
```

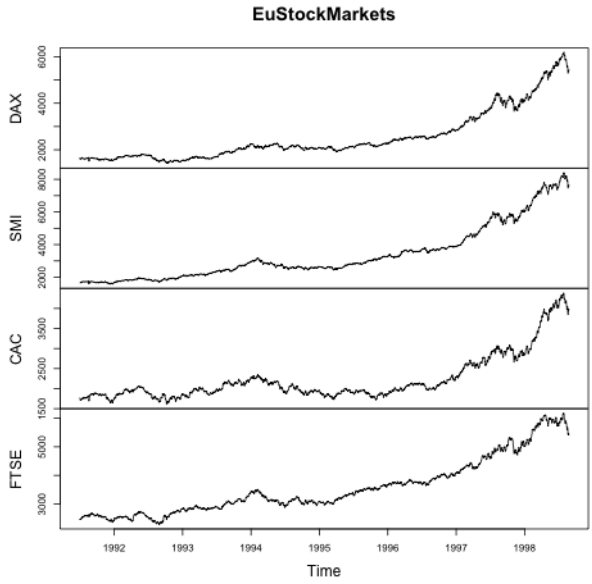


***Рис. 3.1.** Простой график временного ряда*

Объекты `ts` и производные им классы применяются во многих популярных программных пакетах. Такие объекты снабжаются встроенными средствами визуализации, подобные продемонстрированному в предыдущем примере, в котором информативный график с несколькими областями построения создается в результате вызова единственной функции — `plot()`. Объекты `ts` также включают несколько полезных функций.

- `frequency()`. Определяет годовую частоту данных.

```
## R
> frequency(EuStockMarkets)
[1] 260
```

- `start()` и `end()`. Устанавливают начальную и конечную временны*е* метки ряда.

```
## R
> start(EuStockMarkets)
[1] 1991 130
> end(EuStockMarkets)
[1] 1998 169
```

- `window()`. Задает временной период ряда данных.

```
## R
> window(EuStockMarkets, start = 1997, end = 1998)
Time Series:
Start = c(1997, 1)
```

```
End = c(1998, 1)
Frequency = 260 DAX SMI CAC FTSE
1997.000 2844.09 3869.8 2289.6 4092.5
1997.004 2844.09 3869.8 2289.6 4092.5
1997.008 2844.09 3869.8 2303.8 4092.5
...
1997.988 4162.92 6115.1 2894.5 5168.3
1997.992 4055.35 5989.9 2822.9 5020.2
1997.996 4125.54 6049.3 2869.7 5018.2
1998.000 4132.79 6044.7 2858.1 5049.8
```

У класса `ts` есть свои преимущества и недостатки. Как упоминалось ранее, `ts` и производные классы используются во многих пакетах обработки временных рядов данных. Кроме того, полезной оказывается функция автоматической настройки параметров печати. Тем не менее индексация значений в таких объектах может оказаться непростой задачей, а процесс доступа к группам данных, реализуемый функцией `window()`, слишком сложным. Далее в этой книге вы познакомитесь с несколькими способами хранения и доступа к данным временных рядов,— не спешите с выбором наиболее приемлемого варианта.

Гистограммы

Продолжим сравнительный анализ инструментов исследования данных с явно и неявно выраженной временной зависимостью. Давайте построим гистограмму временного ряда подобно тому, как это делается в ходе анализа любых других данных. Кроме того, для более полного понимания временного ряда построим гистограмму разностей исходных данных, чтобы получить временную зависимость (рис. 3.2).

```
## R
> hist( EuStockMarkets[, "SMI"], 30)
> hist(diff(EuStockMarkets[, "SMI"], 30))
```

В анализе временных рядов функция `hist()`, применимая к разностям значений, представляет намного больший интерес, чем она же, заданная для необработанных данных. В конце концов, при изучении временных рядов (особенно финансовых) намного чаще требуется отслеживать изменения исследуемой величины, а не ее абсолютное значение. В наибольшей степени это касается визуализации данных, поскольку гистограмма данных, имеющих выраженный тренд, не несет большой информативной нагрузки.

Обратите внимание на гистограмму данных разностного временного ряда. Если исходные графики котировок акций, приведенные в предыдущем разделе, более чем наглядно демонстрируют уверенный рост цен, то гистограмма разностей исходных данных представляет финансовую картину в таком виде, в каком она анализируется биржевыми игроками. Гистограмма разности указывает на то, что значения ряда

данных со временем увеличиваются (положительные значения разности) и уменьшаются (отрицательные значения разности) примерно на одну и ту же величину. Значения фондовых индексов не показывают временной рост и спад на одну и ту же величину, поскольку имеют тенденцию к постоянному увеличению. Однако из гистограммы разностей видно, что причиной такой тенденции является лишь небольшое смещение в пользу положительных, а не отрицательных различий.

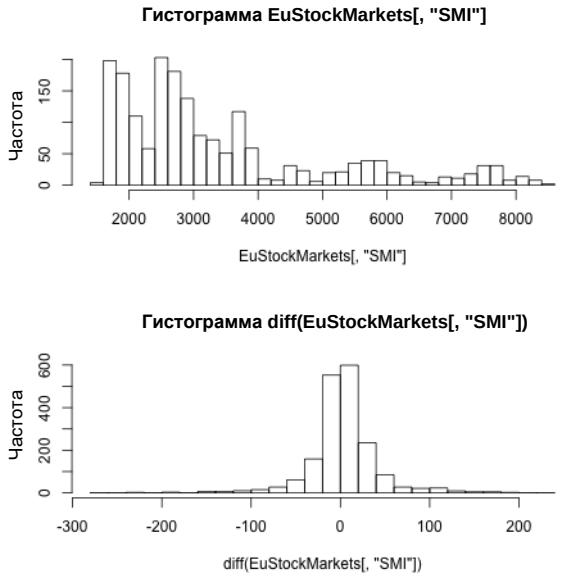


Рис. 3.2. *Необработанные данные (вверху) характеризуются большим разбросом значений и не подчиняются нормальному распределению. Этого следует ожидать, учитывая наличие тренда в исходных данных. Для его удаления и приведения данных к нормальному распределению нужно работать с разностями значений (внизу)*

Приведенный выше пример как нельзя лучше указывает на важность выбора правильной временной шкалы при выборке, обработке и анализе данных. Понимание того, демонстрируют акции высокую прибыль (долгосрочный график) или слабый рост (гистограмма разностей), напрямую зависит от типа временной шкалы — ориентируемся мы на сиюминутную выгоду или инвестируем в более долгосрочной перспективе? Если вы работаете в финансовой организации, занимающейся биржевой торговлей, то для изучения годовой прибыли лучше анализировать краткосрочные изменения, обращая особое внимание на моменты времени, когда разница цен на акции представляется отрицательными значениями. Являясь корпоративным инвестором — представляя крупное производственное предприятие или медицинское учреждение, — анализируйте ситуацию в долгосрочной перспективе, ориентируясь на рост доходности. Последний сценарий предполагает

свои трудности: как выбрать временной отрезок, достаточный для максимизации прибыли, но не настолько большой, чтобы не дождаться ее получения? Описанные выше вопросы стимулируют финансовую индустрию к всестороннему исследованию временны*х* рядов и прогнозированию будущих данных.

Диаграммы рассеяния

Традиционный подход к изучению диаграмм рассеяния настолько же полезен при изучении данных временны*х* рядов, как и других типов данных. Мы можем использовать диаграммы рассеяния для определения взаимосвязи между ценами двух акций в отдельные моменты времени, а также отслеживания их временны*х* изменений.

В следующем примере рассматриваются оба варианта (рис. 3.3).

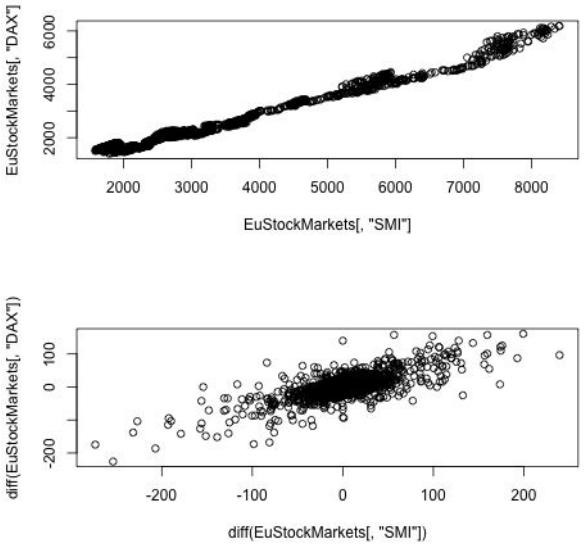


Рис. 3.3. *Простые диаграммы рассеяния двух сильно коррелирующих фондовых индексов. У нас есть веские причины с недоверием относиться к представленным на них данным*

- Цена двух разных акций с течением времени
- Разница (ежедневное изменение) цен двух акций с течением времени, вычисленная с помощью функции `diff()`

```
## R
> plot(      EuStockMarkets[, "SMI"],        EuStockMarkets[, "DAX"])
> plot(diff(EuStockMarkets[, "SMI"]), diff(EuStockMarkets[, "DAX"]))
```

Как было показано выше, абсолютные значения менее информативны, чем разница значений соседних временных точек. Это побуждает к построению второй диаграммы рассеяния разностей исходных данных. Она указывает на сильную корреляцию данных, но последняя не настолько велика, как кажется. (Чтобы понять, почему, внимательно изучите раздел "Ложные корреляции" далее.)

Выраженные корреляции на рис. 3.3 представляют исключительно информационный интерес: даже если они являются истинными (есть основания сомневаться в этом), мы не сможем монетизировать полученную информацию на биржевых торгах. К тому времени, когда станет известно о росте или падении цены одной из акций, цена коррелирующей с ней акции успеет показать уверенный рост или падение, поскольку корреляция между ценами акций рассматривается в одни и те же моменты времени. Нам же нужно попытаться выяснить, можно ли по текущему изменению цены одной из акций предсказать будущее изменение цены другой акции. Чтобы понять это, перед анализом диаграммы рассеяния нужно сместить один из рядов разностей цен акций на шаг назад. Внимательно изучите следующий код; обратите внимание на построение диаграммы временных рядов разностей исходных данных, один из которых смещен относительно второго на единичный временной шаг (рис. 3.4).

```
## R
> plot(lag(diff(EuStockMarkets[, "SMI"]), 1),
           diff(EuStockMarkets[, "DAX"]))
```

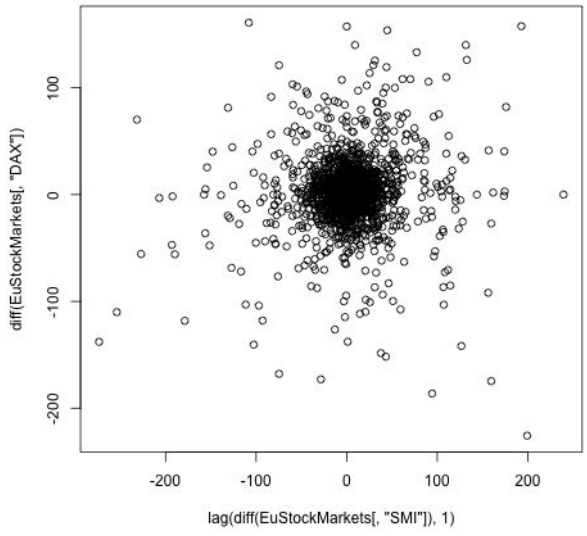


Рис. 3.4. *Корреляция между фондовыми индексами перестала проявляться после временного смещения одного из рядов. Это свидетельствует о невозможности предсказания индекса DAX по индексу SMI*

Приведенный выше код легко читается благодаря правильному выравниванию строк. Писать длинные строки нечитаемого кода, особенно на функциональном языке программирования, таком как R или Python, достаточно просто, но вам следует избегать их везде, где только возможно!

Полученный результат позволяет сделать нескольких важных выводов.

- Данные временны*х* рядов можно анализировать с помощью общих исследовательских методов, но в таком анализе мало толку. Нам нужно модифицировать общие методы под измененный формат данных.
- Часто именно взаимосвязь между данными в разных точках или временно*е* их изменение лучше всего описывает поведение данных.
- График на рис. 3.4 показывает, с какими трудностями приходится сталкиваться биржевым трейдерам. Если вы пассивный инвестор, склонный к выжиданию, то постараетесь извлечь выгоду из долгосрочного возрастающего тренда. Однако, попытавшись построить точный прогноз будущих изменений, вы столкнетесь с серьезными затруднениями!

Функция `lag()` языка R не позволяет смещать данные в произвольном направлении. С ее помощью данные сдвигаются только вперед во времени. Об этом нужно помнить, рассматривая сценарии, в которых направление смещения играет важную роль, — в разных задачах более предпочтительными могут быть сдвиги во времени как вперед, так и назад.

Специальные методы исследования временны*х* рядов

Несколько описанных далее методов анализа данных основаны на изучении взаимосвязи значений временно*го* ряда в разных временны*х* точках. Если вам не доводилось встречаться с ними раньше, то вы, скорее всего, не работали с временны*ми* рядами. В оставшейся части этой главы мы рассмотрим несколько принципов и связанных с ними методов классификации временны*х* рядов.

В последующем материале главы активно используются следующие понятия.

Стационарность

Что означает для временно*го* ряда быть стационарным и как проверяется стационарность временно*го* ряда.

Автокорреляция

Что такое корреляция временно*го* ряда с самим собой и как можно ее использовать для описания динамических изменений временно*го* ряда.

Ложная корреляция

Что такое ложная корреляция и когда она возникает.

Нам предстоит изучить следующие инструменты.

- Функции скользящего и расширяющегося окон
- Функции автокорреляции
  - — Функция общей автокорреляции
  - — Функция частичной автокорреляции

Сначала мы рассмотрим понятия стационарности, а затем перейдем к изучению связанных с ней методов анализа временных рядов — автокорреляции и ложной корреляции. Но перед тем как перейти к их изучению, давайте обсудим область применения этих методов.

Первый вопрос, на который нужно ответить при изучении временного ряда, заключается в том, какую систему он описывает: стабильную или изменчивую. Уровень стабильности, или *стационарности*, важен для оценки того, как долгосрочное поведение системы в прошлом отражает ее будущее долгосрочное поведение. После оценки уровня "стабильности" (здесь этот термин не имеет строго технического определения) временного ряда мы пытаемся определить, присуще ли ему динамическое поведение (например, сезонные изменения). Иначе говоря, мы ищем автокорреляции — в отдаленных и последних значениях, — по которым можно предсказать будущие данные. Наконец, получив представление о динамическом поведении системы, необходимо убедиться в том, что оно не основано на причинно-следственных связях, т.е. попробовать найти *ложные корреляции*.

Определение стационарности

Многие традиционные статистические модели анализа временных рядов основаны на предположении о том, что временные ряды являются стационарными. Вообще говоря, стационарный временной ряд — это ряд, который характеризуется неизменными во времени статистическими показателями, в частности средним значением и дисперсией. Такое определение кажется относительно простым.

Тем не менее стационарность может быть сложной концепцией, особенно применительно к рядам реального масштаба времени. Но полагаясь на одну только интуицию, слишком просто обмануть себя. Поэтому мы сначала рассмотрим понятие стационарности интуитивно и формально, а затем перейдем к обсуждению строгих методов проверки системы на стационарность и способов практического ее применения.

Интуитивный подход

Стационарный временной ряд — это ряд, описывающий систему в устойчивом состоянии. Иногда трудно сказать, что именно это означает, и намного легче показать, что данные не являются стационарными, чем доказать их стационарность. Простым примером нестационарных данных является набор сведений о пассажирах, перевозимых авиакомпанией, который рассматривался нами в главе 2 и графически показан на рис. 3.5 (широко доступен для загрузки со многих интернет-ресурсов и представлен пакетом `AirPassengers` в языке R).

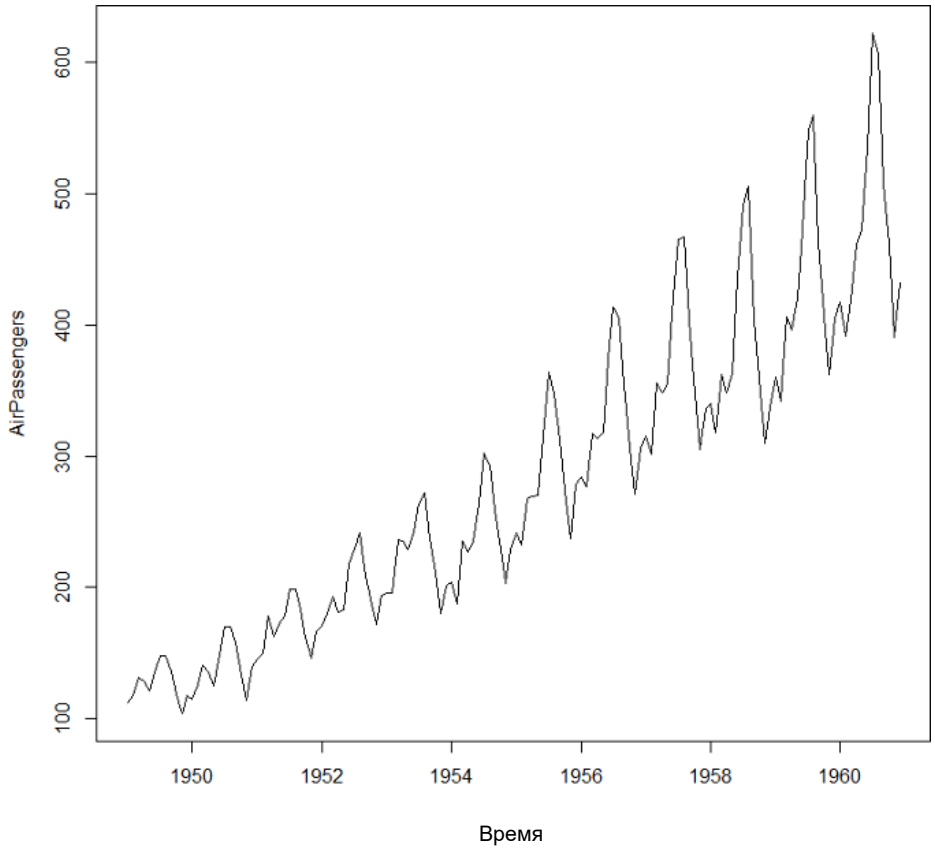


Рис. 3.5. *Данные о пассажирских авиаперевозках являются примером нестационарного временного ряда. Со временем и среднее значение, и дисперсия изменяются. Кроме того, в данных проявляется сезонность, которая присуща только нестационарным процессам*

На нестационарность процесса указывает сразу несколько признаков. Во-первых, среднее значение ряда увеличивается со временем, а не остается постоянным. Во-вторых, расстояние между пиком и впадиной ежегодно увеличивается,

что указывает на возрастание дисперсии процесса с течением времени. В-третьих, процесс демонстрирует сильное сезонное поведение, не характерное для стационарных данных.

Определение стационарности и расширенный тест Дики–Фуллера

Простое определение стационарного процесса заключается в следующем: процесс считается стационарным, если для всех возможных смещений k распределение $y_t, y_{t+1}, ..., y_{t+k}$ не зависит от t.

Статистические тесты на стационарность часто сводятся к вопросу о существовании единичного корня: является ли число 1 решением характеристического уравнения процесса.[1] В общем случае линейный временной ряд является нестационарным, если у него существует единичный корень, хотя отсутствие единичного корня не указывает на стационарность. Ответ на вопрос о стационарности данных все еще остается сложной задачей, а определение того, имеет ли процесс единичный корень, остается актуальной областью исследований.

Тем не менее простое интуитивное представление о том, что такое единичный корень, можно получить из примера, описывающего процесс случайного блуждания:

$$y_t = f \cdot y_{t-1} + e_t.$$

В этом процессе значение временного ряда в текущий момент времени является функцией его значения в предшествующий момент времени и некоторой случайной ошибки. Если f равно 1, то ряд имеет единичный корень, "убегает" и не будет стационарным. Заметьте, нестационарность ряда не указывает на наличие в нем тренда. Случайное блуждание является хорошим примером нестационарного временного ряда, который не имеет основного тренда.[2]

Определение стационарности процесса выполняется согласно специальным *критериям проверки гипотез*. Наиболее часто используемым критерием оценки стационарности временных рядов является расширенный тест Дики–Фуллера (Augmented Dickey-Fuller — ADF). В этом тесте в качестве основной гипотезы принимается предположение о существовании единичного корня. В зависимости от результатов тестирования нулевая гипотеза может быть отклонена при определенном уровне значимости. Это означает, что при данном уровне значимости наличие единичного корня не выступает подтверждением гипотезы.

[1] Не волнуйтесь, если на данный момент вам не понятна эта концепция. При необходимости вы можете детально познакомиться с ней в публикациях, указанных в конце главы.

[2] Может показаться, что у временного ряда, описывающего процесс случайного блуждания, есть тренд, но это очень спорное утверждение, особенно для временных рядов, представляющих биржевые котировки.

Заметьте, что тесты на стационарность основаны на понимании того, изменяется ли среднее значение ряда. При этом дисперсия вычисляется с помощью преобразований, а не формальных методик. Таким образом, проверка стационарности сводится к оценке интегрированности временного ряда. Интегрированный ряд порядка d — это ряд, разности порядка d которого образуют стационарный временной ряд.

Формула теста Дики-Фуллера выглядит следующим образом:

$$Dy_t = y_t - y_{t-1} = (f - 1) \cdot y_{t-1} + e_t.$$

Тогда проверка $f = 1$ сводится к определению t-критерия гипотезы о равенстве нулю запаздывающего параметра y_{t-1}. Тест ADF предполагает большее число временных смещений, а базовая модель учитывает динамику более высокого порядка, которую можно описать таким рядом разностей:

$$y_t - f_1 \cdot y_{t-1} - f_2 \cdot y_{t-2} \ldots = e_t.$$

Как видите, для получения ряда разностей нужно выполнить более сложные вычисления, а ожидаемое распределение, используемое для проверки нулевой гипотезы, несколько отличается от применяемого в тесте Дики–Фуллера. Тест ADF является наиболее широко освещенным методом проверки на стационарность в литературе, посвященной анализу временных рядов.

К сожалению, упомянутые выше тесты не позволяют однозначно решить задачу стационарности по целому ряду причин.

- Эти тесты имеют крайне низкую мощность при различении корней, *близких* к единице, от единичных корней.
- При небольшом размере выборок довольно часто встречаются ложные срабатывания для единичных корней.
- В большинстве тестов не выполняются все виды проверок временного ряда на нестационарность. Например, в некоторых случаях особое внимание уделяется проверке постоянности среднего значения или дисперсии (но не обоих показателей). В некоторых тестах анализируются только распределения общего характера. Следовательно, перед использованием теста нужно познакомиться с его ограничениями и убедиться, что они соответствуют представлениям об анализируемых наборах данных.

Альтернативная нулевая гипотеза: тест KPSS

В то время как тест ADF проверяет нулевую гипотезу о единичном корне, тест Квятковского–Филипса–Шмидта–Шина (KPSS — Kwiatkowski-Philips-Schmidt-Shin) выдвигает нулевую гипотезу о стационарном процессе. В отличие от ADF, тест KPSS

невозможно выполнить с помощью базовых инструментов языка R, хотя он все еще широко применяется на практике. Тесты различаются областью назначения и принципами использования — рассмотрение этих вопросов выходит за рамки материала книги, но вы можете изучить их на специализированных сайтах.

Практический подход

Критерий стационарности находит широкое практическое применение. Во-первых, стационарные процессы описываются большим количеством моделей, например традиционными моделями с заданной силой связи и классическими статистическими моделями. Подробно эти классы моделей описываются в главе 6.

В более широком смысле точность модели нестационарного временного ряда будет меняться по мере изменения временных характеристик данных. Иными словами, в такой модели, призванной оценить среднее значение временного ряда с непостоянными средним и дисперсией, ошибки выборки и вычисления будут изменяться со временем, что делает значения модели недостоверными.

Часто временной ряд можно сделать в достаточной степени стационарным с помощью относительно простых преобразований. Наибольшую популярность приобрели логарифмическое преобразование и преобразование по закону квадратного корня, особенно в рядах с изменяемой во времени дисперсией. Точно так же для удаления из ряда тренда применяются разности. Иногда разности между членами ряда приходится вычислять более одного раза. Однако, если порядок разностей слишком большой (более трех), то маловероятно, что проблему стационарности все же удастся решить с их помощью.

Логарифмирование или извлечение квадратного корня

Хотя вычислять квадратный корень намного проще, чем логарифм, обратите внимание на оба варианта. Оцените диапазон данных и необходимость сжатия больших значений, не отвлекаясь на преждевременную оптимизацию (чрезмерную пессимизацию) вычислительного алгоритма и кода.

Стационарность — не единственное предположение, на котором строятся модели прогнозирования. Другим распространенным предположением является нормальность распределения входных переменных или прогнозируемой величины. В таких моделях востребованными оказываются совсем другие преобразования, например преобразование Box Cox, которое реализовано в пакете `forecast` языка R и в пакете `scipy.stats` языка Python. Такие преобразование позволяет привести ассиметричные данные (с распределением, отличным от нормального) к нормальному распределению. Однако возможность преобразования данных

не предполагает необходимость выполнения этой операции. Тщательно изучите расстояния между точками данных в исходном наборе данных и убедитесь, что в процессе преобразования не будет утеряна важная информация.

Преобразования основываются на предположениях

Может показаться, что предлагаемые преобразования не основаны на предположениях (функции `log()` и `sqrt()` кажутся предельно простыми), но так ли это в действительности?

Как уже отмечалось, логарифмическое преобразование и преобразование по закону квадратного корня обычно используются для уменьшения временной дисперсии ряда. И они основаны на определенных предположениях. Во-первых, данные ряда должны быть положительными. Во-вторых, если вы сдвинете данные до вычисления квадратного корня или логарифма, то увеличите ошибку выборки или же решите, что ею можно пренебречь. И в-третьих, извлекая квадратный корень или рассчитывая логарифм, вы делаете большие значения менее отличными друг от друга, эффективно "ужимая" расстояние между большими, но не меньшими значениями и уменьшая величину выбросов. Такое преобразование уместно далеко не во всех случаях.

Применение оконных функций

Рассмотрим наиболее важные графики временных рядов, которые нам придется исследовать в ходе первичного анализа наборов данных.

Скользящее окно

Общепринятой в анализе временных рядов является оконная функция. Она представляется функцией любого типа и применяется для агрегации данных с целью последующего сжатия (понижения частоты дискретизации; см. главу 2) или сглаживания (также см. главу 2).

Наряду с рассмотренными ранее областями применения, операции сглаживания и агрегации по окнам позволяют добиться более информативной визуализации данных.

Для вычисления скользящего среднего и проведения некоторых других линейных вычислений над рядом точек данных применяется функция `filter()` стандартной библиотеки языка R.

```
## R
> ## Вычисление скользящего среднего с помощью функции filter()
> x <- rnorm(n = 100, mean = 0, sd = 10) + 1:100
> mn <- function(n) rep(1/n, n)
```

```
> plot(x, type = 'l', lwd = 1)
> lines(filter(x, mn( 5)), col = 2, lwd = 3, lty = 2)
> lines(filter(x, mn(50)), col = 3, lwd = 3, lty = 3)
```

При выполнении кода создается график, показанный на рис. 3.6.

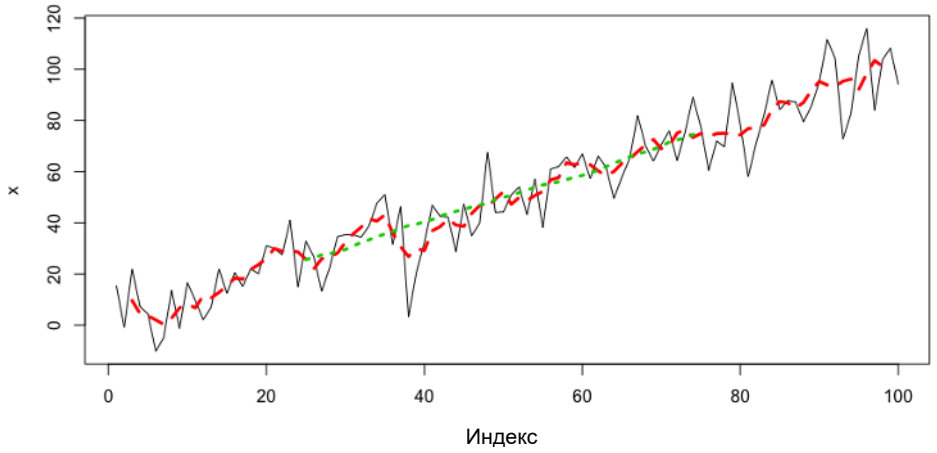


Рис. 3.6. *Две кривые, построенные в процессе сглаживания скользящего среднего. Их можно использовать для выявления тренда в сильно зашумленных данных или определения уровня отклонения от линейной зависимости, рассматриваемого как шум*

Если точки данных в окне нужно описать нелинейным образом, то придется отказаться от функции `filter()`, поскольку она основана исключительно на линейных способах преобразованиях данных. Рассмотрите функции пакета `zoo`, в частности функцию `rollapply()`, результат использования которой показан на рис. 3.7.

```
## R
> ## Настраиваемые оконные функции
> require(zoo)

> f1 <- rollapply(zoo(x), 20, function(w) min(w),
>                 align = "left", partial = TRUE)
> f2 <- rollapply(zoo(x), 20, function(w) min(w),
>                 align = "right", partial = TRUE)
> plot (x,           lwd = 1,          type = 'l')
> lines(f1, col = 2, lwd = 3, lty = 2)
> lines(f2, col = 3, lwd = 3, lty = 3)
```

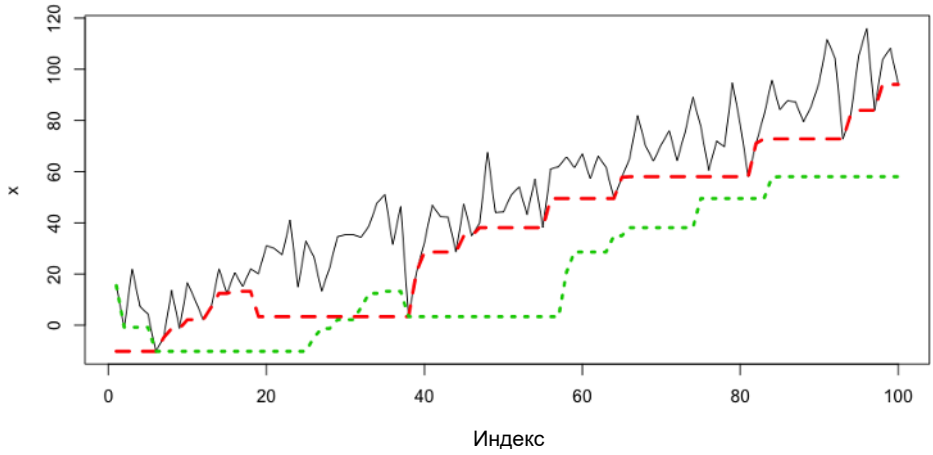


Рис. 3.7. *Минимум скользящего окна выровнен либо по левому краю (длинные штрихи), либо по правому (короткие штрихи). Выравнивание по левому краю позволяет учитывать события в будущем, тогда как выравнивание по правому — только в прошлом. Это важно понимать во избежание упреждения. Тем не менее иногда выравнивание по левому краю может быть даже полезным, позволяя применить знания о будущем для правильного анализа имеющихся данных. В отдельных случаях правильных результатов не позволяет добиться даже упрежденный просмотр данных. Если знание будущих значений не помогает в анализе временного ряда, то это означает, что рассматриваемая величина не информативна по своей сути и от ее исследования нужно отказаться*

В функциях пакета `zoo` следует использовать объекты пакета. При непосредственной передаче числовых векторов в функцию `rollapply()` аргумент `align` в ней не учитывается. Чтобы убедиться в этом, удалите оболочку `zoo()` у аргумента `x` в предыдущем коде. Вы получите идентичные кривые вследствие скрытой ошибки, которая чрезвычайно опасна при анализе временных рядов, поскольку может привести к непреднамеренному упреждению. Приведенный пример наглядно показывает важность обязательной проверки чистоты кода в процессе анализа данных. К сожалению, скрытая ошибка не является чем-то необычным во многих популярных пакетах не только языка R, но и языках описания сценариев. Будьте предельно внимательны при их использовании!

У вас есть возможность настраивать собственные оконные функции для эффективного ограничения исследуемых зависимостей. Начните с изучения и изменения кода уже существующих функций, включенных в такие популярные пакеты, как `zoo`. Это наиболее оправданный подход, поскольку даже в одномерном

временном ряду существует большое количество факторов, требующих специального внимания — значения NA, а также начальные и конечные точки ряда, у которого меньше точек, чем требует размер окна.

Инструменты языка R для работы с временными рядами

Ранее в этой главе мы рассмотрели базовый класс `ts` языка R, применяемый для обработки временных рядов. В этом разделе мы воспользовались для этих целей объектами пакета `zoo`. Кроме того, в дальнейшем мы будем активно использовать пакет `xts`. Ниже приведены краткие замечания по использованию объектов пакетов `zoo` и `xts`, расширяющих возможности класса `ts`.

- Объект класса `ts` предназначен для работы с равномерно распределенными полными временными рядами — в нем отсутствуют данные временных индексов, но содержатся сведения о начальной и конечной точках, а также о частоте выборки временного ряда.
- Объекты класса `ts` поддерживают периодически повторяющиеся временные циклы, например годы или месяцы.
- В объектах пакета `zoo` временные метки хранятся в специальном атрибуте `index`, поэтому их можно применять для хранения неравномерно распределенных или непериодических временных рядов.
- На графиках объекты пакета `zoo` можно откладывать вдоль как горизонтальной, так и вертикальной осей.
- Данные объекта пакета `zoo` можно представить вектором или матрицей.
- Объекты пакета `xts` являются расширением объектов пакета `zoo` с большим количеством параметров.

Расширяющиеся окна

В анализе временных рядов расширяющиеся окна используются несколько реже скользящих окон, поскольку имеют ограниченную область применения. Расширяющиеся окна уместно использовать только при подведении сводной статистики устойчивых процессов, которые не подвержены сильным временным изменениям и не демонстрируют заметных колебаний. Расширяющееся окно не имеет строго заданного постоянного размера. Имея в самом начале минимально возможный размер, с течением времени оно расширяется, включая все точки данных до указанного момента времени.

Расширяющееся окно обеспечивает большую достоверность оценки статистических показателей во времени за счет "расширения" диапазона исследуемых значений временного ряда. Заметьте, что такой подход имеет место только при

справедливости гипотезы о стационарности исходной системы. Он позволяет получать и оценивать сводную статистику в режиме реального времени — по мере включения во временной ряд новых данных.

Внимательно изучив инструменты стандартной библиотеки языка R, вы найдете, что в ней уже реализованы функции управления расширяющимися окнами, такие как `cummax()` и `cummin()`. Более того, функцию `cumsum()` можно легко приспособить к вычислению накопительного среднего. На следующем графике вы можете познакомиться с расширяющимися окнами `max` и `mean` (рис. 3.8).

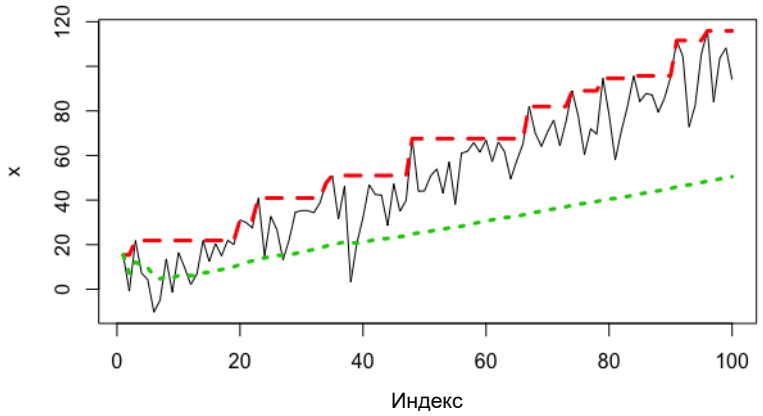


Рис. 3.8. *Расширяющиеся окна* `max` *(длинные штрихи) и* `mean` *(короткие штрихи). В первом случае расширяющееся окно отражает глобальный максимум до текущего момента времени, представленный монотонной функцией. Благодаря "долгосрочной памяти" кривая расширяющегося окна* `mean` *расположена ниже графика скользящего среднего (см. рис. 3.7) — основной тренд в расширяющемся среднем выражен слабее. Хорошо это или плохо, зависит от исходных предположений о поведении исследуемой системы*

```
## R
> # Расширяющиеся окна
> plot(x, type = 'l', lwd = 1)
> lines(cummax(x), col = 2, lwd = 3, lty = 2)                # max
> lines(cumsum(x)/1:length(x), col = 3, lwd = 3, lty = 3)  # mean
```

Для задания расширяющегося окна с произвольными настройками можно обратиться к функции `rollapply()`, используемой ранее при описании пользовательского скользящего окна. Отличие состоит в том, что в данном случае предстоит указать последовательность размеров окна, а не скаляр. При выполнении следующего кода создается график, идентичный показанному на рис. 3.8, но теперь полученный с помощью функции `rollapply()`, а не встроенных инструментов языка R.

```
## R
> plot(x, type = 'l', lwd = 1)
> lines(rollapply(zoo(x), seq_along(x), function(w) max(w),
>                         partial = TRUE, align = "right"),
>           col = 2, lwd = 3, lty = 2)
> lines(rollapply(zoo(x), seq_along(x), function(w) mean(w),
>                         partial = TRUE, align = "right"),
>           col = 3, lwd = 3, lty = 3)
```

Пользовательские оконные функции

Необходимость в применении пользовательских оконных функций возникает только в случаях, когда обойтись без них очень сложно или просто невозможно, например при исследовании временных рядов с известными поведенческими закономерностями или анализе данных с помощью строго заданных вычислительных методик.

Предположим, что требуется задать окно для исследования отдельного признака — наиболее информативного в исследуемой системе. Нам предстоит узнать, что именно описывают исследуемые данные: *монотонное* изменение (например, уровня сахара в крови) или беспорядочные скачки, подразумевающие инструментальный шум, а не тренд. Для исследования такого поведения данных лучше написать пользовательскую функцию и применить ее в сценариях со скользящим и/или расширяющимся окном.

Самокорреляция и ее поиск

В основе понятия самокорреляции временного ряда лежит представление о том, что значения временного ряда в отдельные моменты времени могут коррелировать с его значениями в другие моменты времени. Заметьте, что термин "самокорреляция" не имеет строго технического определения и применяется для описания общей идеи.

Прекрасным примером самокорреляции служит взаимосвязь между значениями временных рядов, представляющих годовые данные о дневной температуре воздуха. Допустим, что при сравнении температур воздуха 15 мая и 15 августа каждого года была выявлена следующая взаимосвязь между данными: более жаркая погода 15 мая имеет тенденцию к корреляции с более жаркой (или с более холодной) погодой 15 августа. Если такая взаимосвязь действительно существует, то выявленный факт, указывающий на определенную закономерность в поведении температуры, можно использовать при составлении долгосрочных прогнозов погоды. С другой стороны, факт отсутствия корреляции (нулевая корреляция) также позволяет сделать определенные выводы: сведения о температуре 15 мая не позволяют выдвигать какие бы то ни было предположения о диапазоне температур 15 августа.

От описанного выше простого примера мы перейдем к изучению автокорреляции — общему случаю самокорреляции, лишенному привязки к конкретному моменту времени. В частности, автокорреляция сводится к решению задачи поиска взаимосвязи между любыми двумя точками общего временного ряда, расположенными на строго заданном расстоянии друг от друга. Далее мы остановимся на рассмотрении этого вопроса более подробно, а затем перейдем к изучению понятия частичной автокорреляции.

Автокорреляционная функция

Изучение автокорреляции начнем с ее определения в англоязычной Википедии.

> Автокорреляция, также известная как корреляция рядов, — это корреляция сигнала с его смещенной во времени копией как функция смещения. В нестрогом понимании автокорреляция описывает сходство между наблюдениями как функцию временного сдвига между ними.

А теперь перепишем это определение простым языком. Автокорреляция дает представление о линейной взаимосвязи точек данных, полученных в разные моменты времени, как о функции разницы времени их получения.

Проще всего *функция автокорреляции* (AutoCorrelation Function — ACF) воспринимается в графическом виде. Для построения графика функции автокорреляции можно обратиться к базовым инструментам языка R (рис. 3.9).

```
## R
> x <- 1:100
> y <- sin(x x pi /3)
> plot(y, type = "b")
> acf(y)
```

Корреляция в детерминированной системе

Ряд значений функции синуса описывает полностью детерминированную систему с известной последовательностью входных данных. Тем не менее корреляция в ней не равна единице. Почему? Изучите этот ряд и попробуйте самостоятельно разобраться в том, как вычисляется функция ACF. Понимание того, увеличивается или уменьшается последующее значение, зависит от места в цикле, с которого начинается анализ данных. Для определения направления изменения значений достаточно отследить несколько точек подряд (функция ACF является однозначной мерой корреляции). Если же это не представляется возможным, то корреляция будет меньше единицы, так как для большинства точек данных последующие значения представлены не одним, а несколькими значениями. Важно понимать, что корреляция, отличная от единицы, не указывает на зашумленность или вероятностность поведения временного ряда.

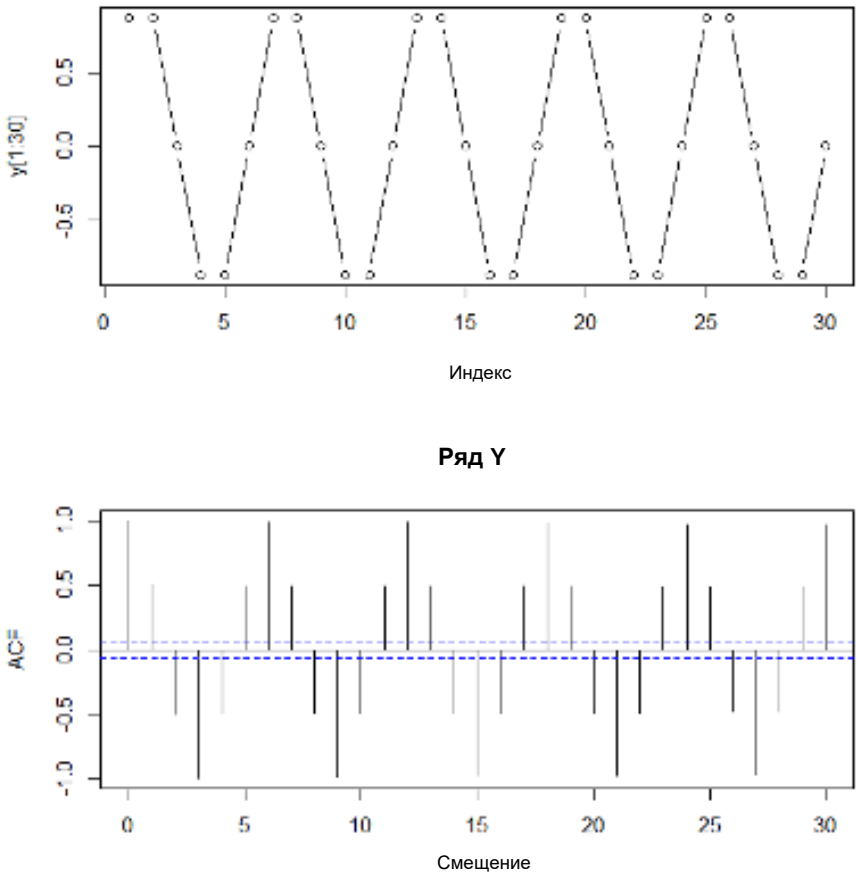


Рис. 3.9. *Графики функции синуса и его автокорреляционной функции*

Автокорреляционная функция показывает, что точки с нулевым временным сдвигом имеют единичную корреляцию (верно для любого временного ряда), тогда как корреляция точек данных с единичным сдвигом составляет 0,5. При этом точки, разделенные двойным временным сдвигом, характеризуются корреляцией, равной –0,5, и т.д.

Вычислить функцию ACF несложно. Попробуем сделать это самостоятельно, обратившись к функции `shift()` пакета `data.table`.

```
## R
> cor(y, shift(y, 1), use = "pairwise.complete.obs")
[1] 0.5000015
> cor(y, shift(y, 2), use = "pairwise.complete.obs")
[1] -0.5003747
```

Проведенные выше расчеты соответствуют результатам, проиллюстрированным на рис. 3.9.[3] Хотя написание собственного кода вычисления функции ACF не вызывает особых затруднений, все же лучше использовать для этих целей проверенное решение, например представленное функцией `acf()` языка R. Используя ее, вы получаете в свое распоряжение следующие возможности.

- Автоматическое построение графиков с важными подписями.
- Правильное определение (в большинстве случаев, но не всегда) максимального количества сдвигов, для которых рассчитываются автокорреляционная функция, а также возможность их переопределения вручную.
- Изящный способ обработки многомерных временных рядов.

Также остановимся на важных математических особенностях функции ACF.

- Автокорреляция периодической функции имеет ту же периодичность, что и исходный процесс. Вы можете увидеть это на приведенных выше графиках для функции синуса.
- Автокорреляция суммы периодических функций — это сумма автокорреляций для каждой функции в отдельности. Вы можете легко проиллюстрировать это утверждение на примере, реализуемом с помощью простого кода.
- Все временные ряды имеют единичную автокорреляцию для нулевого временного сдвига.
- Автокорреляция белого шума будет приблизительно равняться 0 при любых временных сдвигах, за исключением нулевого.
- Автокорреляция симметрична относительно отрицательных и положительных временных сдвигов, поэтому рассчитывать ее в явном виде имеет смысл только в одном из вариантов (для положительных сдвигов). Чтобы доказать это, постройте автокорреляционную функцию, рассчитанную указанным способом вручную.
- Статистическое правило определения значимой ненулевой оценки автокорреляции основано на выборе критической области с границами $\pm 1{,}96\sqrt{n}$. В этом правиле в расчет принимаются достаточно большой размер выборки и конечное значение дисперсии процесса.

[3] Выборочные корреляции, рассчитанные с помощью функций `cor()` и `acf()` языка R, будут несколько отличаться, поскольку в них используются разные делители. Дополнительная информация приведена на сайте StackExchange по адресу `https://perma.cc/M7V6-HN5Y`.

Частичная автокорреляция

Функция частичной автокорреляции (Partial Autocorrelation Function — PACF) немного сложнее автокорреляционной функции. Частичная автокорреляция временного ряда для заданного сдвига — это корреляция временного ряда с самим собой в этом смещении с учетом всей доступной информации между указанными двумя моментами времени.

С благоразумностью звучания определения нельзя не согласиться. Но что же означает фраза "учет всей доступной информации между указанными двумя моментами времени"? А означает она то, что для получения точного результата нужно вычислить ряд условных корреляций и вычесть их из общей корреляции. Вычисление функции PACF — это нетривиальная задача, и для ее решения придумано большое количество методик. Мы не будем обсуждать их в этой главе, но вы можете найти их описание в документации к соответствующим инструментам языков R и Python.

Как и ACF, функцию PACF легче изучать графически, чем концептуально. Ее важность в анализе данных также становится более очевидной при визуализации в отдельной области построения (рис. 3.10).

```
## R
> y <- sin(x * pi /3)
> plot(y[1:30], type = "b")
> pacf(y)
```

Функция PACF для временного ряда функции синуса сильно контрастирует с функцией ACF. Функция PACF наглядно показывает, какие точки данных являются информативными, а какие выступают гармониками более коротких периодов времени.

Для периодического процесса без шума, например описываемого функцией синуса с периодом T, то же самое значение ACF будет наблюдаться в периоды времени T, $2T$, $3T$ и так далее вплоть до бесконечности. Функция ACF не в состоянии отсеять такие избыточные корреляции. С другой стороны, функция PACF показывает, какие корреляции для заданных сдвигов являются истинными, а какие — избыточными. Эти сведения оказываются неоценимыми при выборе длинного окна для достаточно большого количества данных в требуемом временном масштабе.

Критическая область для функции PACF рассчитывается так же, как и для функции ACF. Она имеет границы $\pm 1{,}96\sqrt{n}$. Все временные сдвиги, значения PACF для которых попадают в обозначенную такими границами область, фактически равны нулю.

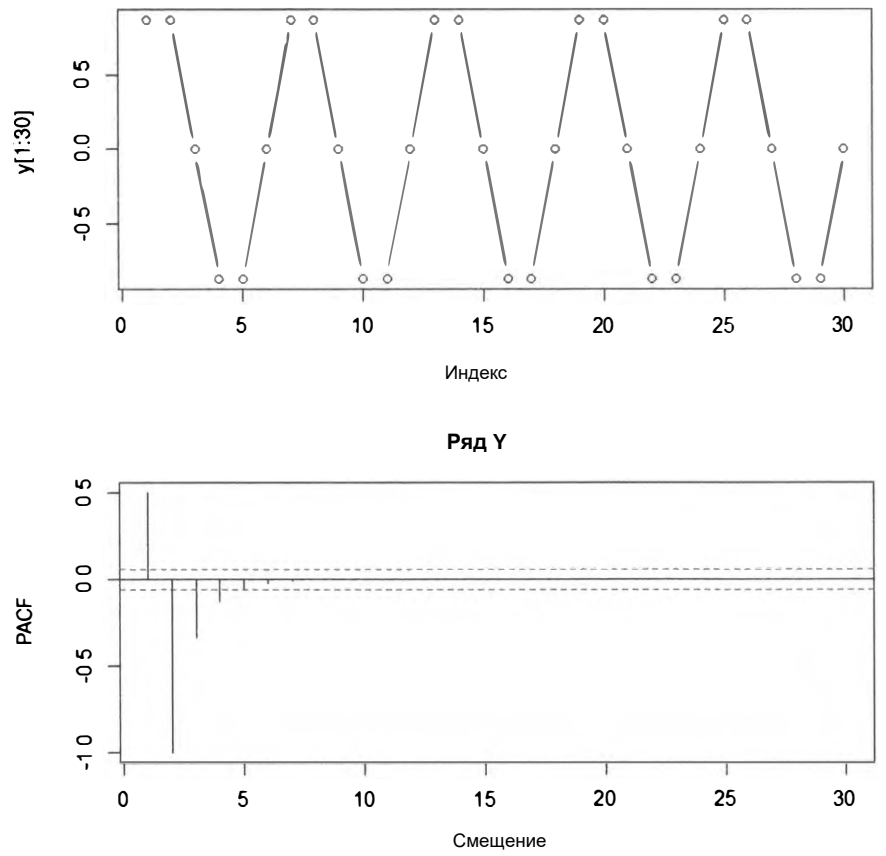


Рис. 3.10. *График функции PACF для периодического процесса, лишенного шума*

До настоящего момента нами рассматривались примеры только полностью бесшумных одночастотных процессов. Давайте изучим более сложный пример. Представим сигнал суммой двух синусоидальных функций в условиях нулевого, слабого и сильного шума.

Для начала построим графики синусоид, лишенных шума, по отдельности (рис. 3.11).

```
## R
> y1 <- sin(x × pi /3)
> plot(y1, type = "b")
> acf (y1)
> pacf(y1)

> y2 <- sin(x × pi /10)
> plot(y2, type = "b")
> acf (y2)
> pacf(y2)
```

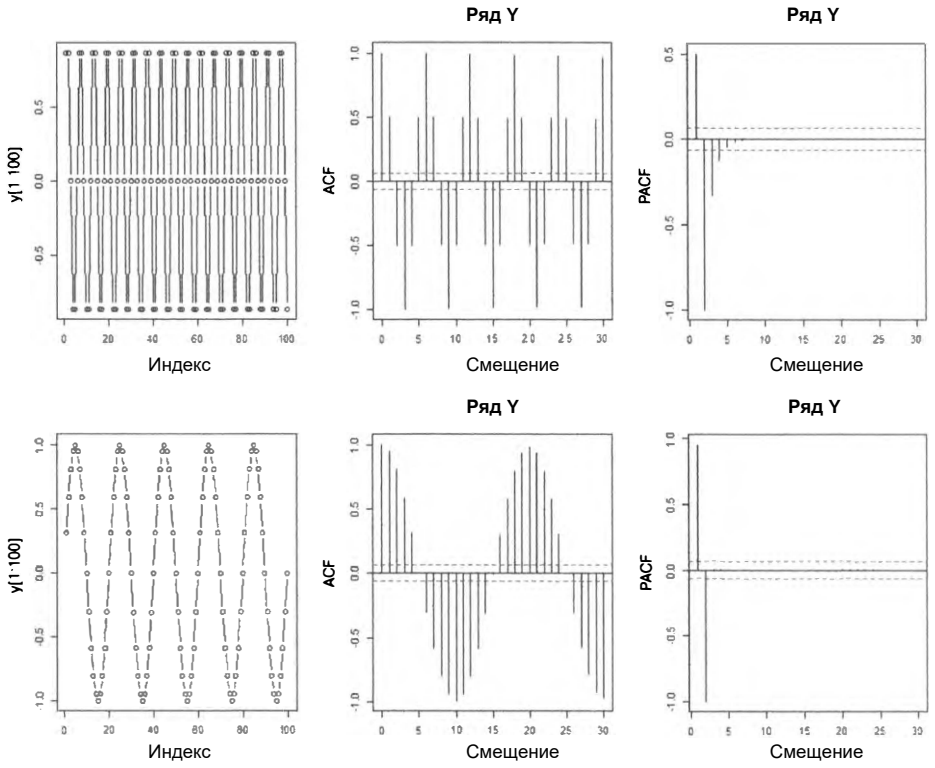


Рис. 3.11. *Графики двух синусоид, а также их ACF и PACF*

Значение автокорреляционной функции для стационарных данных должно быстро падать к нулю. У нестационарных данных значение ACF при единичном сдвиге всегда будет положительным и большим.

Просуммируем два ряда и построим такие же графики для полученного результата (рис. 3.12).

```
## R
> y <- y1 + y2
> plot(y, type = "b")
> acf (y)
> pacf(y)
```

Легко заметить, что график ACF полностью подчиняется упомянутому выше правилу: ACF суммы периодических рядов представляется суммой ACF отдельных рядов. В этом можно наглядно убедиться, изучив последовательность чередующихся положительных и отрицательных участков ACF, коррелирующих с более медленно осциллирующими значениями ACF. Внутри плавных волн отчетливо просматриваются более частые колебания высокочастотной составляющей ACF.

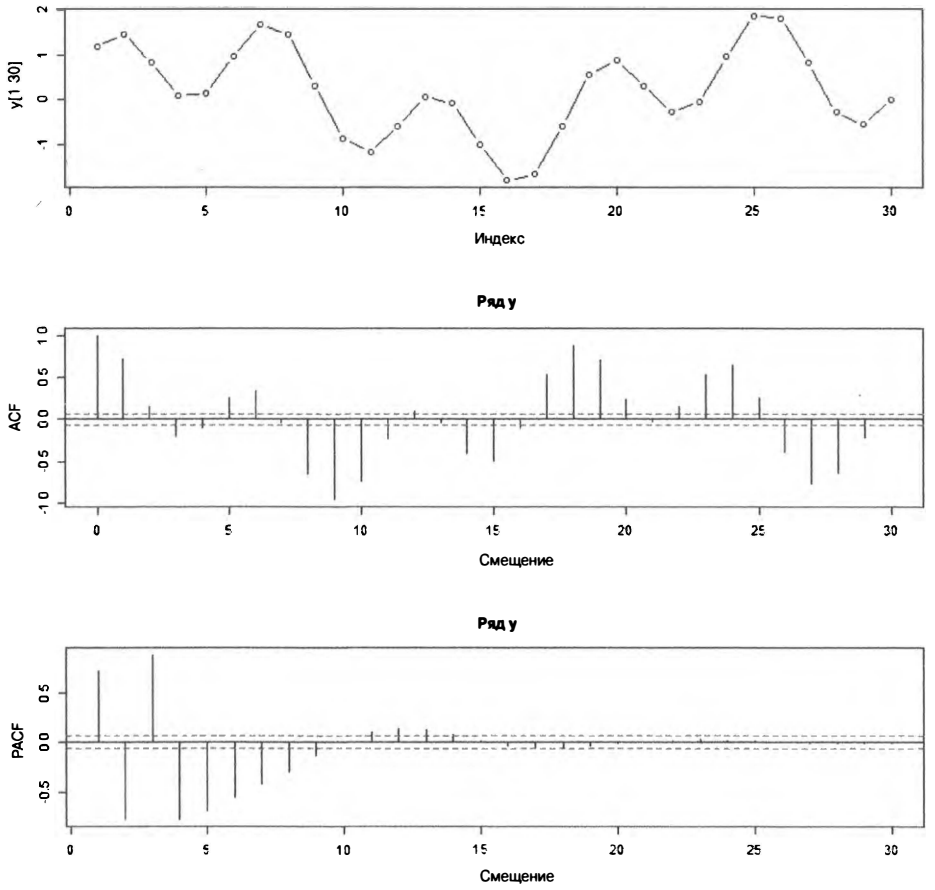


Рис. 3.12. *Графики суммы двух синусоид, а также их ACF и PACF*

При этом PACF суммы не представлена суммой PACF ее слагаемых. Она достаточно простая для понимания, но чрезвычайно сложная для вычисления или прогнозирования. В нашем случае анализ PACF показывает, что частичная автокорреляция сильнее проявляется в ряду суммы синусоид, чем в любом из исходных рядов. Таким образом, корреляция между точками, сдвинутыми на определенный временной интервал, с учетом значений точек между ними сильнее проявляется в ряду суммы сигналов, чем в исходных рядах. Это вызвано существованием в нем двух разных периодов — любая рассматриваемая точка данных в меньшей степени определяется значениями соседних точек, поскольку ее положение в пределах двухпериодного цикла теперь менее фиксировано из-за колебаний на разных частотах.

Рассмотрим аналогичную ситуацию, но на этот раз с некоторым шумом (рис. 3.13).

```
## R
> noise1 <- rnorm(100, sd = 0.05)
> noise2 <- rnorm(100, sd = 0.05)

> y1 <- y1 + noise1
> y2 <- y2 + noise2
> y <- y1 + y2

> plot(y1, type = 'b')
> acf (y1)
> pacf(y1)

> plot(y2, type = 'b')
> acf (y2)
> pacf(y2)

> plot(y, type = 'b')
> acf (y)
> pacf(y)
```

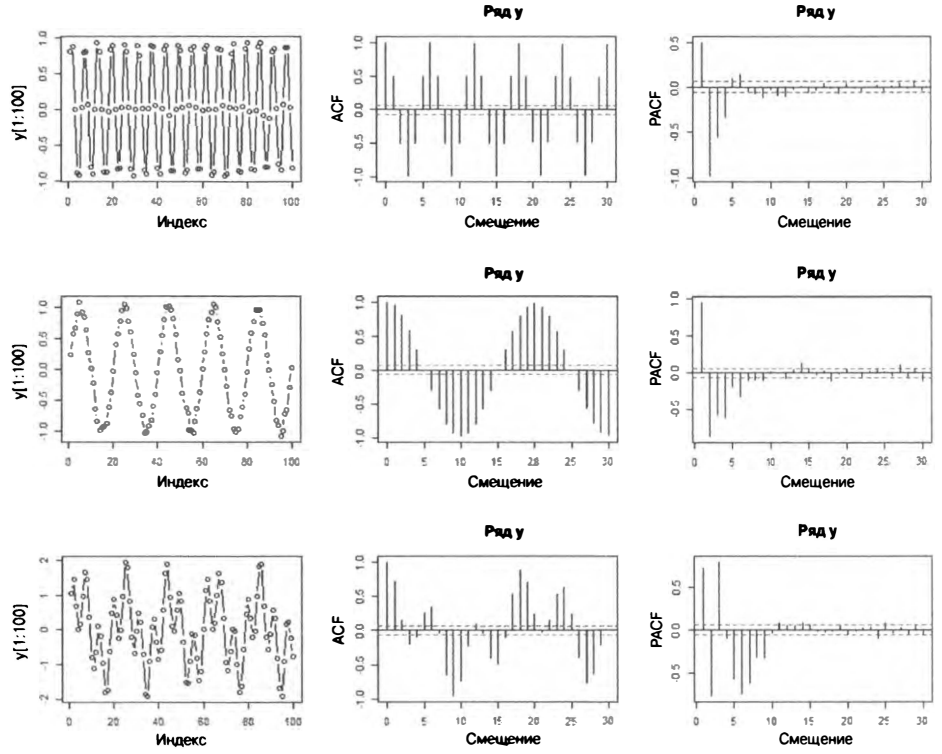


Рис. 3.13. *Графики двух зашумленных синусоид, их суммы, а также ACF и PACF*

Наконец, добавим во временные ряды больше шума, чтобы еще сильнее исказить исходные синусоиды. (Пример кода их построения не приводится умышленно — он такой же, как в предыдущем примере, но с заметно большим значением параметра `sd` функции `rnorm`.) Как можно видеть, шум создает дополнительные трудности при интерпретации данных, особенно функции PACF. Единственное различие между графиками, приведенными на рис. 3.14 и предыдущих рисунках, — это большее значение `sd` для переменных `noise`.

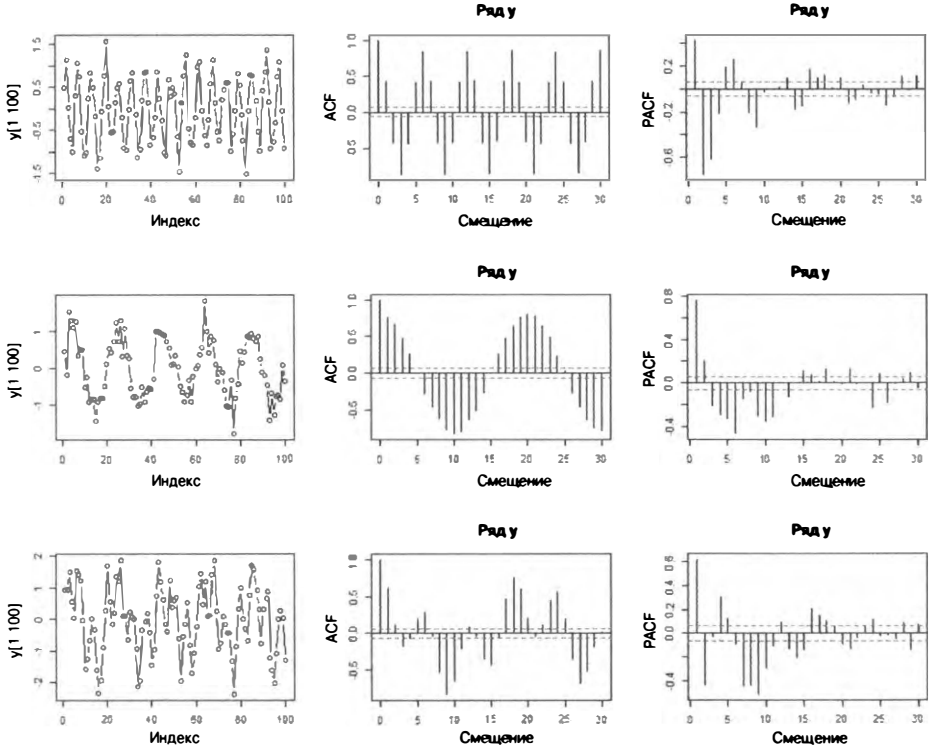


Рис. 3.14. Графики двух сильно зашумленных синусоид, их суммы, а также ACF и PACF

Нестационарные данные

Посмотрим, как будут выглядеть функции ACF и PACF для временного ряда с трендом, но лишенного периодических изменений (рис. 3.15).

```
## R
> x <- 1:100
> plot(x)
> acf (x)
> pacf(x)
```

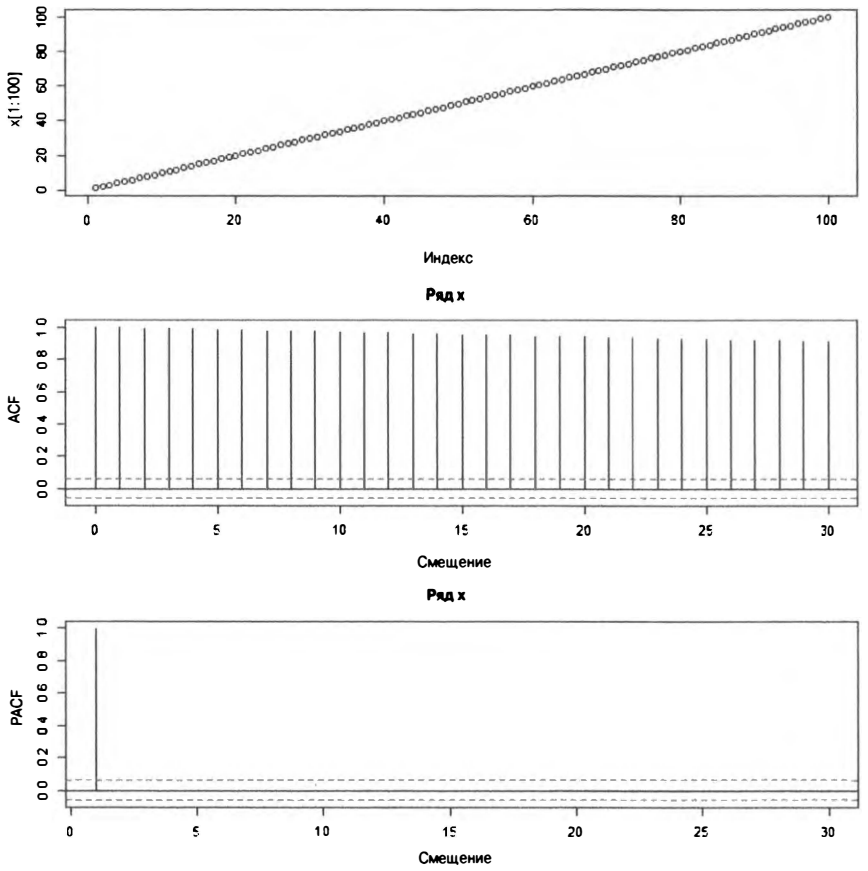


Рис. 3.15. *Графики процесса с линейным трендом, а также его ACF и PACF*

В данном случае функция ACF совершенно не информативна. Она имеет одно и то же значение для всех возможных сдвигов, что указывает на одинаковую корреляцию на любых временных интервалах. Непонятно, как это нужно трактовать, и больше похоже на случай с совершенно бессмысленным результатом.

К счастью, PACF не столь сложна для понимания и предоставляет более важную информацию — единственная значимая корреляция PACF проявляется при единичном сдвиге. Почему так? Да потому, что всю необходимую информацию о ряде на определенный момент времени можно получить по известной предыдущей точке данных. И связано это с тем, что следующая временная точка сдвинута всего на 1 относительно предыдущей точки.

В заключение давайте изучим функции ACF и PACF, рассчитанные для реального набора данных. В качестве примера воспользуемся набором данных `AirPassengers` (рис. 3.16). Учитывая приведенные выше рассуждения, попробуйте

выяснить причину большого количества "критических" значений в ACF (ответ: наличие тренда) и существование их в PACF для больших временных сдвигов (ответ: годовой сезонный цикл различается даже в процессе с трендом).

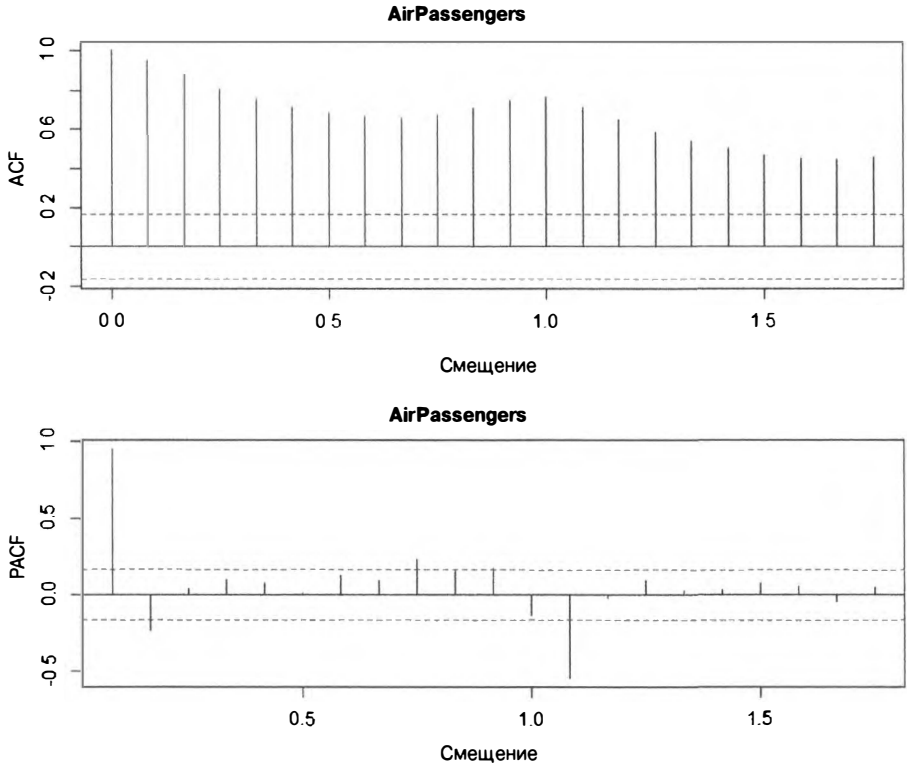


Рис. 3.16. *Графики ACF и PACF для данных набора* `AirPassenger`

Ложные корреляции

Те, кто плохо знаком с временными рядами, обычно применяют для анализа данных стандартные исследовательские методики, заключающиеся в изучении графиков зависимости двух переменных и их корреляции. Только приступив к исследованию временных рядов и научившись различать взаимосвязи в данных, вы, скорее всего, будете уделять корреляции слишком много внимания. Занимаясь исследованием данных, вы будете постоянно находить самые разные взаимосвязи и искренне удивляться тому, насколько сильно могут коррелировать данные в, казалось бы, хорошо изученной системе. Рано или поздно удивление сменится сожалением о столь запоздалом переходе к изучению методов анализа временных рядов, открывающих совершенно новые карьерные возможности.

Принявшись наверстывать упущенное и прочитав несколько учебников по анализу временных рядов (наиболее предпочтительный сценарий), вы поделитесь

полученными результатами с более опытными коллегами, которые непременно укажут на ошибочность многих ваших предположений (не самый лучший сценарий). Но обязательно найдутся и те, кто скажет, что данные коррелируют очень сильно. На этой волне вам будет казаться, что взаимосвязь просматривается между абсолютно любыми двумя значениями. Еще больше проблем возникает при перезапуске анализа с другими наборами переменных и обнаружении в них поразительно высоких корреляций. Наконец, в какой-то момент вам станет ясно, что настолько высокие корреляции в данных просто невозможны.

Описанный выше сценарий концептуально повторяет историю развития такой важной сегодня дисциплины, как эконометрика. О цикличности экономических процессов экономисты начали задумываться еще в начале XIX века. В качестве причины возникновения таких циклов выбирались самые невероятные внешние факторы, включая появление солнечных пятен (11-летний цикл) или недавно изученные метеорологические изменения (4-летний цикл атмосферных осадков). Они неизменно получали сильно коррелирующие величины даже в отсутствие причинно-следственной гипотезы, объясняющей взаимосвязи в исследуемых данных.

Наряду с этим большинство экономистов и специалистов по статистике, будучи истинными учеными, скептически оценивали полученные результаты. Удни Юл описал проблематику ложных корреляций в статье *Why Do We Sometimes Get Nonsense Correlations?* (Почему мы получаем бессмысленные корреляции?) (`https://www.jstor.org/stable/2341482`). Вскоре она была выделена в отдельную область исследований, которая не только доставляет ученым головную боль, но и радует новыми открытиями. Ложные корреляции остаются важной проблемой, требующей самого пристального изучения и опровержения. Наиболее горячие дискуссии они вызывают в судебных разбирательствах, в которых одна из сторон настаивает на существовании каких-то отношений, а другая всячески их опровергает. Подобным образом попытки дискредитации гипотезы об изменении климата аргументируются проявлением ложных корреляций в данных о выбросах углерода и повышении средней общемировой температуры, что объясняется наличием в них очевидных трендов (спорное утверждение).

Со временем специалисты по экономике выяснили, что ложным корреляциям сильно подвержены данные с явно выраженным трендом. Это можно объяснить следующим образом: динамический временной ряд содержит больше информации, чем стационарный, поэтому у его точек больше возможностей для совместного перемещения.

Кроме трендов, ложные корреляции могут порождаться некоторыми другими общими факторами, описывающими поведение временны*х* рядов.

- Сезонность. Простым примером будет ложная корреляция между объемами продаж хот-догов и количеством утопленников в летний сезон.

- Смещение уровня или наклона графика, вызванное временным изменением режима (приводит к гантелеобразному распределению с бессмысленно высокой корреляцией).
- Кумулятивное суммирование исследуемых величин (этот прием часто применяется в определенных отраслях для представления корреляции в более выгодном свете, чем она проявляется).

Коинтеграция

Коинтеграция описывает действительно существующую взаимосвязь между временными рядами. Часто приводимый пример — поведение пьяного пешехода и его собаки. Рассматриваемые по отдельности траектории их движения могут выглядеть как полностью случайные и независимые. На самом деле они никогда не отдаляются слишком далеко друг от друга.

В случае коинтеграции также имеет место сильная корреляция данных. Трудность ее распознавания заключается в определении того, связаны ли между собой процессы или вы наблюдаете ложную корреляцию, — сильная корреляции данных характерна для обоих случаев. Основное различие состоит в том, что в случае ложной корреляции взаимосвязь никак не проявляется, тогда как коинтегрированные временные ряды тесно связаны друг с другом.

Большое количество замечательных примеров ложных корреляций приведено в одном хорошо известном блоге (а теперь уже и книге), и я хотел бы поделиться с вами одним из приведенных в нем примеров (рис. 3.17). Всякий раз, получив предпосылки к распознаванию в данных сильных взаимосвязей, постарайтесь найти убедительные тому противоречия, подобные явно выраженным трендам.

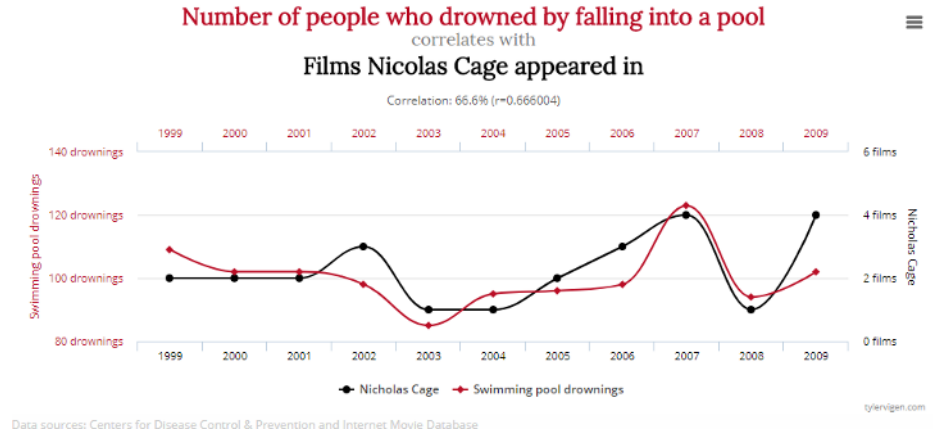


Рис. 3.17. *Некоторые ложные корреляции выглядят на удивление убедительно. График заимствован с сайта Tyler Vigen, посвященного проблематике ложной корреляции (*`https://perma.cc/6UYH-FPBX`*)*

Основные способы визуализации

Графики играют основополагающую роль в детальном анализе временных рядов. Вы наверняка захотите изобразить данные ряда вдоль временной оси — предпочтительно таким образом, чтобы получить исчерпывающую информацию о наборе данных, такую как поведение исследуемой переменной или общее временное распределение точек данных.

Ранее в этой главе мы уже рассматривали отдельные методы графического представления данных, знакомые каждому аналитику, например в виде графика временной зависимости изучаемых величин или диаграммы рассеяния значений столбцов во времени. В последнем разделе, посвященном графическим инструментам анализа данных, описываются некоторые другие способы визуализации, которые хорошо показали себя при исследовании поведения временных рядов.

Далее рассматриваются принципы визуализации разного уровня сложности.

- Одномерная визуализация предназначается для ознакомления с общим временным распределением наборов данных, из которых извлекаются анализируемые временные ряды (глава 2).
- Двумерная визуализация (гистограмма) позволяет отследить временные изменения исследуемых величин при проведении большого количества параллельных измерений (проведение одинаковых измерений в течение многих лет или повторного многократного сбора данных одной и той же величины).
- Трехмерная визуализация, в которой время может представлять сразу два или даже ни одного измерения, но все равно (неявно) принимать участие в анализе данных.

Одномерная визуализация

В случае проведения измерений для множества изучаемых участников (пользователей, членов организации и т.д.) несколько временных рядов рассматриваются параллельно. Всегда неплохо сопоставить их в графическом виде, заключив данные отдельных участников в соответствующие временные рамки. На этом этапе объектом исследования выступают не отдельные временные точки, а временные диапазоны. Предметом анализа становятся временные интервалы. Несмотря на широкий выбор инструментов, мы будем проводить анализ с помощью пакета `timevis` языка R. Нам предстоит изучить подмножество данных о добровольных взносах, подготовленное в главе 2 (рис. 3.18).

```
## R
> require(timevis)
> donations <- fread("donations.csv")
> d         <- donations[, .(min(timestamp), max(timestamp)), user]
```

```
> names(d)  <- c("content", "start", "end")
> d         <- d[start != end]
> timevis(d[sample(1:nrow(d), 20)])
```

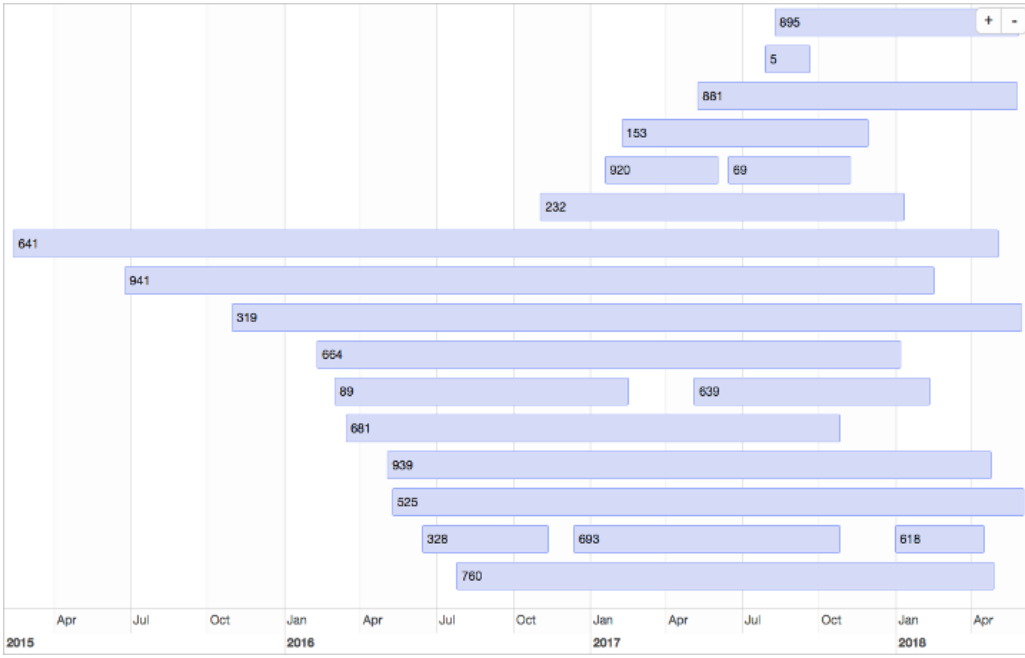


Рис. 3.18. *Диаграмма Гантта случайной выборки данных позволяет узнать о периодах "занятости" членов организации/плательщиков взносов*

Диаграмма, показанная на рис. 3.18, показывает, что членов благотворительной организации можно охарактеризовать так называемыми периодами "занятости". Кроме того, из нее можно получить представление о суммах благотворительных взносов, вносимых плательщиком на протяжении всего времени членства в организации.

Диаграммы Гантта активно используются в течение последнего столетия, — чаще всего при решении задач управления проектами. Они начали применяться независимо сразу во многих отраслях, а их общая идея становится понятной уже при знакомстве с представленной на них информацией. Несмотря на то что исходно диаграммы Гантта были предназначены для управления проектами, их можно эффективно использовать в анализе временных рядов, представляющих данные многих независимых участников (в противоположность одному измеряемому процессу). Диаграмма на рис. 3.18 однозначно отвечает на вопрос о перекрытии платежей отдельных пользователей в течение всей истории внесения

добровольных пожертвований. Получить представление о распределении взносов по одним табличным данным необычайно сложно.

Двумерная визуализация

Теперь давайте обратимся к данным пакета `AirPassengers`, чтобы исследовать их сезонные и трендовые изменения, отказавшись от представления о линейности течения времени. В частности, будем отображать время сразу на нескольких осях. Конечно, существует общая ось времени со всем нам привычным укладом смены часов, дней и годов, но ничто не запрещает выделить отдельную временную ось только для дней недели или времени суток. В такой способ мы акцентируем внимание на поведении данных в строго заданные временны*е* интервалы — дни недели или месяцы года. Реализуемый подход позволяет предельно точно отслеживать сезонные изменения, что очень сложно сделать в ходе анализа данных в линейном, хронологическом масштабе времени.

Следующий код извлекает данные из объекта `AirPassengers` и представляет их в виде матрицы.

```
## R
> t(matrix(AirPassengers, nrow = 12, ncol = 12))
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12]
[1,]  112  118  132  129  121  135  148  148  136  119  104  118
[2,]  115  126  141  135  125  149  170  170  158  133  114  140
[3,]  145  150  178  163  172  178  199  199  184  162  146  166
[4,]  171  180  193  181  183  218  230  242  209  191  172  194
[5,]  196  196  236  235  229  243  264  272  237  211  180  201
[6,]  204  188  235  227  234  264  302  293  259  229  203  229
[7,]  242  233  267  269  270  315  364  347  312  274  237  278
[8,]  284  277  317  313  318  374  413  405  355  306  271  306
[9,]  315  301  356  348  355  422  465  467  404  347  305  336
[10,] 340  318  362  348  363  435  491  505  404  359  310  337
[11,] 360  342  406  396  420  472  548  559  463  407  362  405
[12,] 417  391  419  461  472  535  622  606  508  461  390  432
```

Заметьте, что в процессе преобразования данные представляются в формате, принятом в объекте `ts`.

Столбцовая и строчная организация данных

По умолчанию данные в языке R имеют столбцовую организацию (`https://perma.cc/L4BH-DKB8`) в противоположность строчному способу хранения данных, принятому в пакете `NumPy` языка Python (и в большинстве реляционных баз данных). Знать способ хранения данных, принятый по умолчанию в используемом языке программирования, важно не только для правильной их визуализации, но и для эффективного управления памятью, а также для получения быстрого доступа к любым хранящимся в ней данным.

Данные годов откладываются на отдельных осях, каждая из которых отражает помесячные изменения исследуемой величины в течение рассматриваемого года (рис. 3.19).

```
# R
> colors <- c("green", "red", "pink", "blue",
>             "yellow", "lightsalmon", "black", "gray",
>              "cyan", "lightblue", "maroon", "purple")
> matplot(matrix(AirPassengers, nrow = 12, ncol = 12),
>         type = 'l', col = colors, lty = 1, lwd = 2.5,
>         xaxt = "n", ylab = "Passenger Count")
> legend("topleft", legend = 1949:1960, lty = 1, lwd = 2.5,
>         col = colors)
> axis(1, at = 1:12, labels = c("Jan", "Feb", "Mar", "Apr",
>                               "May", "Jun", "Jul", "Aug",
>                               "Sep", "Oct", "Nov", "Dec"))
```

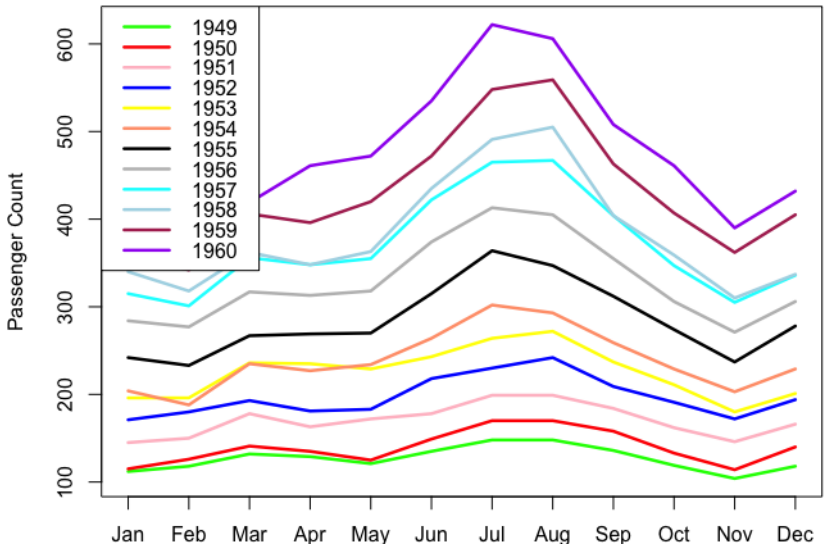


Рис. 3.19. *Помесячный учет для каждого года*[4]

Более простой способ построения такого же графика заключается в использовании пакета `forecast` (рис. 3.20).

```
## R
> require(forecast)
> seasonplot(AirPassengers)
```

[4] Чтобы детально познакомиться с исходным рисунком, загрузите его из репозитория на GitHub (`https://github.com/PracticalTimeSeriesAnalysis/BookRepo`) или постройте самостоятельно.

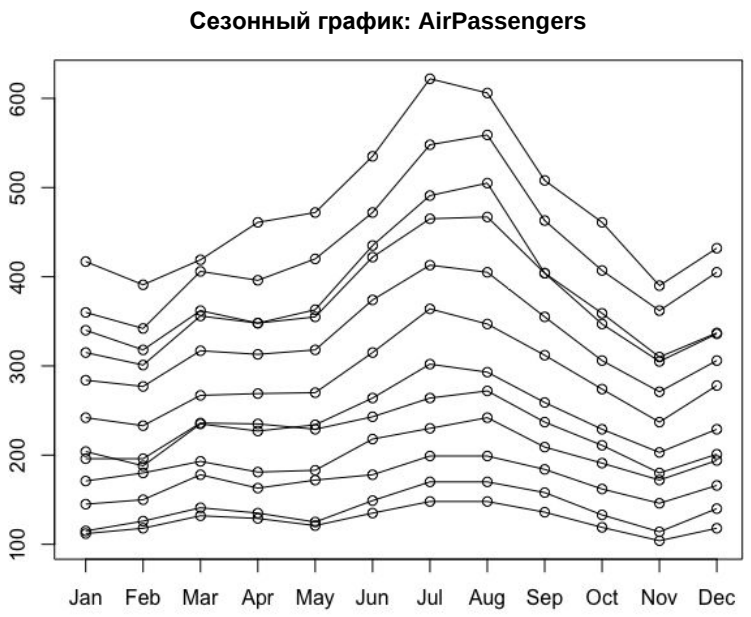


Рис. 3.20. *Результат построения предыдущего графика с помощью функции* `seasonplot()`

На оси *x* откладываются месяцы всех подлежащих анализу годов. В каждом году максимальное количество перевозимых авиакомпанией пассажиров регистрируется в июле или августе (7- и 8-й месяцы). Кроме того, в марте (3-й месяц) почти каждого года наблюдается локальный пик пассажирских перевозок. Таким образом, из графика можно узнать много важных сведений о сезонном характере пассажирских авиаперевозок.

Кривые разных лет редко пересекаются. Наблюдается настолько устойчивый рост, что в разные годы количество пассажиров, перевозимых в одном и том же месяце, редко бывает одинаковым. В общей тенденции есть несколько исключений, но такие наблюдения случаются не в пиковые месяцы. На основании одних только этих результатов можно составить прогноз по росту объемов пассажирских перевозок авиакомпании.

Альтернативный график годовых изменений отдельно для каждого месяца не менее стандартен, хотя и не столь информативен (рис. 3.21).

```
## R
> months <- c("Jan", "Feb", "Mar", "Apr", "May", "Jun",
>              "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")
> matplot(t(matrix(AirPassengers, nrow = 12, ncol = 12)),
>           type = 'l', col = colors, lty = 1, lwd = 2.5)
> legend("left", legend = months,
>         ol = colors, lty = 1, lwd = 2.5)
```

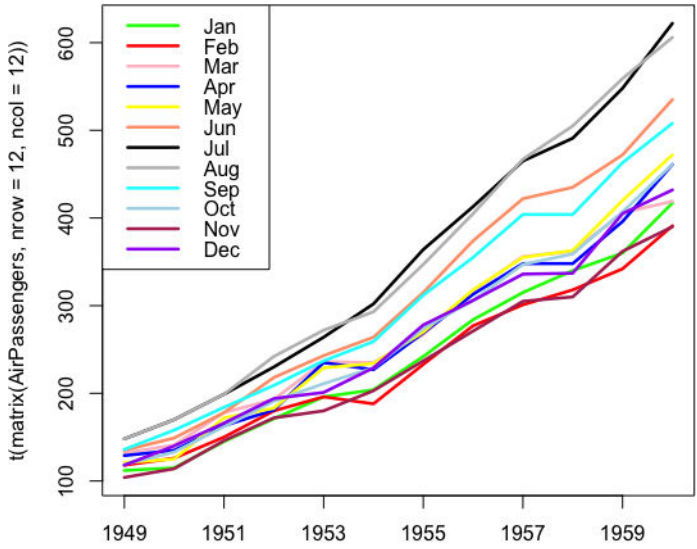


Рис. 3.21. *Годовые изменения для каждого месяца*[5]

С увеличением значения года наблюдается тенденция к росту объемов пассажирских перевозок; иначе говоря, увеличивается скорость роста. Наибольший рост наблюдается в течение двух месяцев — июля и августа. Аналогичные выводы можно сделать, представив данные в графическом виде с помощью простой функции визуализации, включенной в состав пакета прогнозирования `forecast` (рис. 3.22).

```
## R
> monthplot(AirPassengers)
```

Из двух последних графиков можно сделать два очевидных вывода.

- При построении графиков данных временны*х* рядов можно использовать сразу несколько временны*х* осей. В одном из случаев отдельные временны*е* оси представляют помесячную разбивку для каждого года (с января по декабрь), а в другом наборы данных для каждого года отображаются на единой временно*й* оси (с первого по последний/двенадцатый год).
- Изучая результаты визуализации данных временны*х* рядов, можно получить намного больше полезных сведений и составить более точные прогнозы, чем в ходе анализа линейных графиков.

[5] Чтобы детально познакомиться с исходным рисунком, загрузите его из репозитория на GitHub (`https://github.com/PracticalTimeSeriesAnalysis/BookRepo`) или постройте самостоятельно.

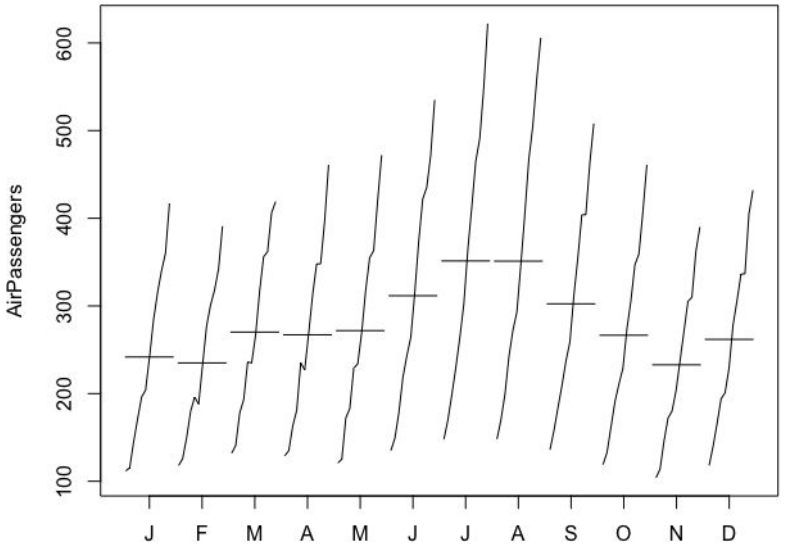


Рис. 3.22. *Используем функцию* `monthplot()` *для помесячного изучения годовых объемов пассажирских перевозок*

Далее рассмотрим правильно построенную двумерную гистограмму. В контексте анализа временны*х* рядов на одной из осей двумерной гистограммы откладывается время (или величина, его заменяющая), а вторая ее ось отводится для отображения изучаемой величины. Набор временны*х* графиков, отображенных вместе на одном из предыдущих рисунков, близок к тому, чтобы быть двумерной гистограммой, — для полного соответствия в них следует внести определенные изменения.

- Данные должны суммироваться как по времени, так и по количеству пассажиров.
- Нужно отобразить как можно больше данных. Двумерная гистограмма малоинформативна, если не отображает данные сразу всех кривых — будучи отображенными по отдельности, они не несут важных сведений. Только в случае представления полного набора данных двумерная гистограмма становится важным источником информации.

Сначала мы проиллюстрируем возможности двумерных гистограмм на небольшом наборе данных, а затем перейдем к анализу более содержательного примера. Напишем пользовательскую функцию построения двумерной гистограммы "с нуля".

```
## R
> hist2d <- function(data, nbins.y, xlabels) {
```

```
>    ## Создание равномерных интервалов на оси y, включающих
>    ## минимальные и максимальные значения
>    ymin = min(data)
>    ymax = max(data) * 1.0001
>    ## Ленивый способ вычислений, не требующий учета
>    ## включаемых/исключаемых точек

> ybins = seq(from = ymin, to = ymax, length.out = nbins.y + 1)

> ## Создание нулевой матрицы подходящего размера
> hist.matrix = matrix(0, nrow = nbins.y, ncol = ncol(data))

>    ## Данные представляются в виде матрицы, в которой каждая строка
>    ## соответствует отдельной точке данных
>    for (i in 1:nrow(data)) {
>       ts = findInterval(data[i, ], ybins)
>       for (j in 1:ncol(data)) {
>          hist.matrix[ts[j], j] = hist.matrix[ts[j], j] + 1
>       }
>    }
>    hist.matrix
> }
```

Построим гистограмму в формате тепловой карты.

```
## R
> h = hist2d(t(matrix(AirPassengers, nrow = 12, ncol = 12)), 5, months)
> image(1:ncol(h), 1:nrow(h), t(h), col = heat.colors(5),
> axes = FALSE, xlab = "Время", ylab = "Частота")
```

Полученная диаграмма (рис. 3.23) не очень удобна для анализа.

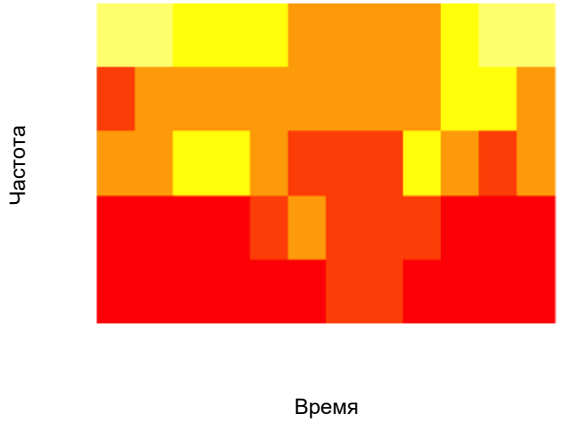


Рис. 3.23. *Тепловая карта, построенная по двумерной гистограмме, для данных набора* `AirPassenger`

Эта диаграмма не информативна, потому что в ней не хватает данных. На ней представлены данные всего 12 кривых, разделенных на 5 сегментов. Однако более важной проблемой является нестационарность данных. Наряду с этим гистограмма имеет смысл только в случае отображения стационарных данных. В нашем случае в данных просматривается тренд, который будет мешать распознаванию сезонных изменений.

Теперь давайте рассмотрим набор данных, включающий бо́льшее количество выборок и не "испорченный" трендом. Воспользуемся подмножеством набора данных `FiftyWords`, заимствованного из архива UCR Time Series Classification (`https://perma.cc/Y982-9FPS`). Этот набор данных включает представление 50 разных слов, записанных в виде одномерных временны́х рядов, каждый из которых имеет одинаковую длину.

Подмножество данных, используемых для построения графика, показанного на рис. 3.24, взято из набора, который применялся при рассмотрении общих вопросов классификации временны́х рядов (`https://oreil.ly/M6T-u`). Для самостоятельного выполнения этого упражнения вы можете скачать это подмножество по указанной ссылке.

```
## R
> require(data.table)

> words <- fread(url.str)
> w1 <- words[V1 == 1]

> h = hist2d(w1, 25, 1:ncol(w1))

> colors <- gray.colors(20, start = 1, end = .5)
> par(mfrow = c(1, 2))
> image(1:ncol(h), 1:nrow(h), t(h),
>     col = colors, axes = FALSE, xlab = "Время", ylab = "Проецируемое
      значение")
> image(1:ncol(h), 1:nrow(h), t(log(h)),
>     col = colors, axes = FALSE, xlab = "Время", ylab = "Проецируемое
      значение")
```

График в правой части рис. 3.24 намного четче, поскольку на нем цветовая насыщенность точек определяется логарифмическим, а не линейным масштабом измерения.

В этом подходе применяется такой же принцип, как и при логарифмировании значений временны́х рядов, позволяющий уменьшить дисперсию набора данных, амплитуду выбросов и расстояние между ними. Логарифмическое преобразование отображаемых данных позволяет сжать шкалу до масштаба, в котором большая часть диапазона отводится для значащих, а не редко расположенных точек данных.

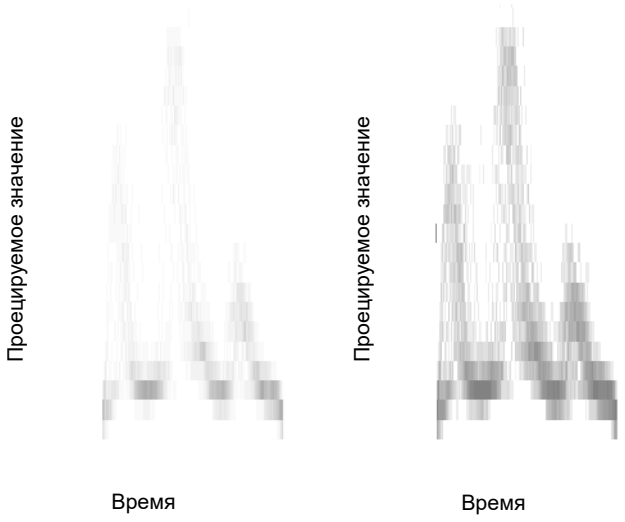


Рис. 3.24. *Двумерная гистограмма проецирования звуков отдельного слова. В левой части график имеет линейную шкалу измерений, а в правой построен по логарифмической шкале*

Наш способ визуализации данных заметно отличается от принятого в готовых решениях, поэтому давайте изучим их более внимательно. Чтобы воспользоваться преимуществами готовых решений, необходимо предварительно изменить формат представления исходного набора данных, поскольку они предполагают отображение на гистограмме пар значений x – y. В отличие от написанного собственноручно решения, готовые решения не предназначены для непосредственной обработки временных рядов. Тем не менее в них включены отличные функции визуализации данных (рис. 3.25).

```
## R
> w1 <- words[V1 == 1]

> ## Представление данных в виде пар координат в соответствии
> ## с требованиями функций построения двумерных гистограмм
> names(w1) <- c("type", 1:270)
> w1        <- melt(w1, id.vars = "type")
> w1        <- w1[, -1]
> names(w1) <- c("Время", "Значение")

> plot(hexbin(w1))
```

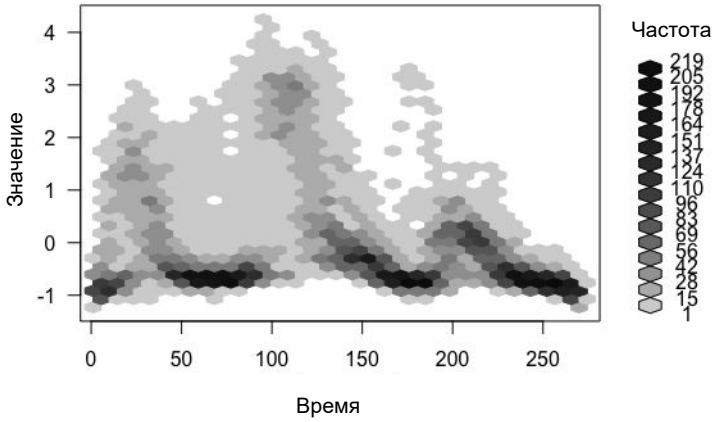


***Рис. 3.25.** Альтернативная двумерная гистограмма для таких же исходных данных*

Трехмерная визуализация

Инструменты трехмерной визуализации данных не включены в стандартную библиотеку языка R, но содержатся в специальных программных пакетах. Представленные далее быстрые графики созданы с помощью пакета `plotly`, который выбран мною из-за возможности вращения в среде RStudio и экспортирования во всевозможные веб-интерфейсы. Кроме того, загрузка и установка пакета `plotly` обычно не вызывают особых затруднений, чего нельзя сказать о большинстве пакетов визуализации.

Как и ранее, воспользуемся данными пакета `AirPassengers`. Отобразим их в трех измерениях, два из которых отводятся для данных времени (месяц и год), а еще одно — для исследуемых значений.

```
## R
> require(plotly)
> require(data.table)

> months = 1:12
> ap = data.table(matrix(AirPassengers, nrow = 12, ncol = 12))
> names(ap) = as.character(1949:1960)
> ap[, month := months]
> ap = melt(ap, id.vars = 'month')
> names(ap) = c("month", "year", "count")

> p <- plot_ly(ap, x = ~month, y = ~year, z = ~count,
>              color = ~as.factor(month)) %>%
>   add_markers() %>%
>   layout(scene = list(xaxis = list(title = 'Месяц'),
>                       yaxis = list(title = 'Год'),
>                       zaxis = list(title = 'AirPassenger')))
```

Представленная в трехмерном виде диаграмма помогает лучше понять общую структуру данных. Вам уже доводилось видеть подобную диаграмму рассеяния, но расширение до трех измерений делает ее более информативной, чем двумерная гистограмма, основанная на неполном наборе данных (рис. 3.26 и 3.27).

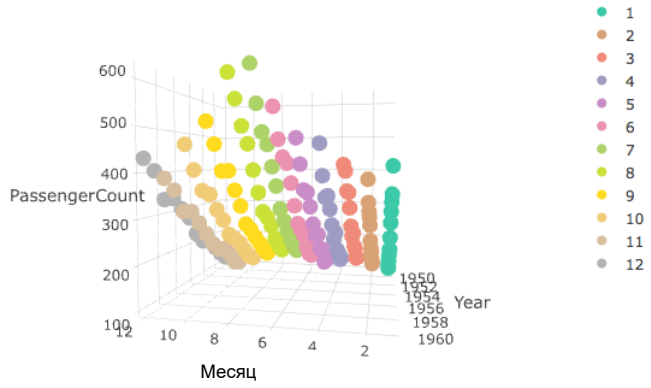


Рис. 3.26. *Трехмерная диаграмма рассеяния, основанная на данных набора* `AirPassanger`*, демонстрирует ярко выраженный сезонный характер*[6]

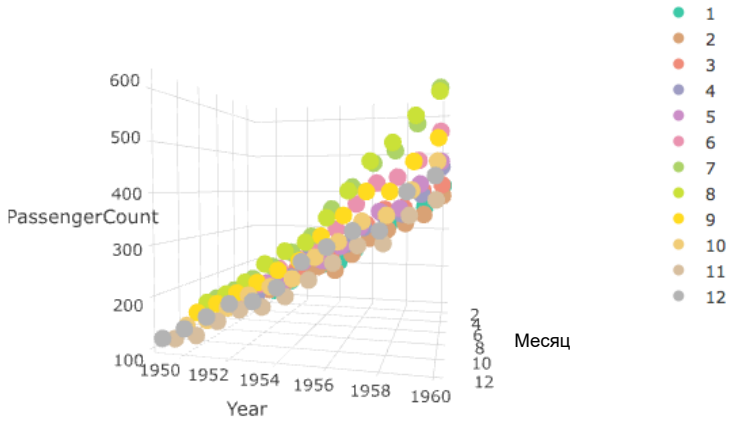


Рис. 3.27. *Представление этих же данных с другого ракурса позволяет убедиться в существовании возрастающего тренда от одного года к другому. Обязательно выполните код на своем компьютере, чтобы повернуть трехмерную диаграмму самостоятельно*[7]

[6]Чтобы детально познакомиться с исходным рисунком, загрузите его из репозитория на GitHub (`https://github.com/PracticalTimeSeriesAnalysis/BookRepo`) или постройте самостоятельно.

[7]Чтобы детально познакомиться с исходным рисунком, загрузите его из репозитория на GitHub (`https://github.com/PracticalTimeSeriesAnalysis/BookRepo`) или постройте самостоятельно.

Совсем не обязательно откладывать значения времени сразу на двух осях. Например, их можно указывать только на одной оси, а две другие использовать для позиционирования (установки двумерных координат). Таким способом можно визуализировать двумерное случайное блуждание, лишь немного изменив исходный код примера демонстрации возможностей пакета `plotly` (рис. 3.28 и 3.29).

```
## R
> file.location <- 'https://raw.githubusercontent.com/plotly/datasets/
                                                    master/\_3d-line-plot.csv'
> data <- read.csv(file.location)
> p <- plot_ly(data, x = ~x1, y = ~y1, z = ~z1,
>                type = 'scatter3d', mode = 'lines',
>                line = list(color = '#1f77b4', width = 1))
```

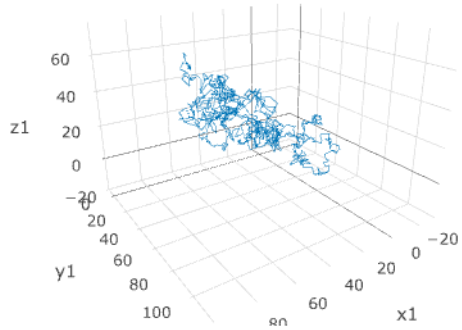


Рис. 3.28. *Одна из возможных точек зрения на двумерное блуждание во времени*

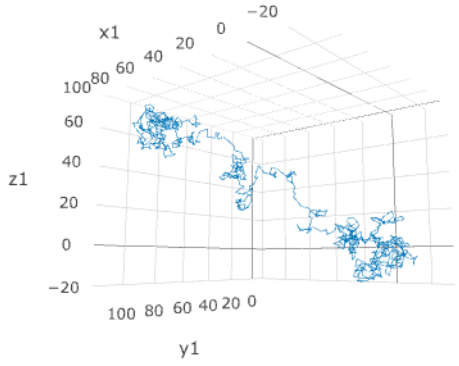


Рис. 3.29. *Еще одна точка зрения на такое же случайное блуждание оказывается более информативной. Не поленитесь выполнить код примера самостоятельно*

Интерактивный характер диаграммы является ключевым фактором в анализе данных. Отдельные перспективы могут как вводить в заблуждение, так и освещать данные в выгодном свете. Зачастую истинный характер данных остается неизведанным до поворота диаграммы.

Хорошим упражнением будет генерирование зашумленных данных о сезонных перемещениях в двух измерениях и последующая их визуализация таким же способом, как при случайном блуждании. Следует ожидать существенных различий между полученным результатом и визуализацией случайного блуждания. Такие пакеты, как `plotly`, позволяют легко экспериментировать с представлением данных и получать всестороннюю визуальную обратную связь.

Дополнительные источники

- Ложные корреляции

  Ai Deng, "A Primer on Spurious Statistical Significance in Time Series Regressions," Economics Committee Newsletter 14, no. 1 (2015), `https://perma.cc/9CQR-RWHC`

  Эта техническая статья, описывающая ложные корреляции и причины их появления в данных, станет полезной для выработки навыков практического понимания того, когда и при каких условиях проблемы подобного рода будут возникать в собственных наборах данных. Статья написана на очень доступном языке.

  Tyler Vigen, Spurious Correlations (New York: Hachette, 2015), `https://perma.cc/YY6R-SKWA`

  Коллекция временных рядов с разными корреляциями является важным источником информации для каждого специалиста по анализу данных.

  Antonio Noriega and Daniel Ventosa-Santaularia, "Spurious Regression Under Broken-Trend Stationarity," Journal of Time Series Analysis 27, no. 5 (2006): 671–84, `https://perma.cc/V993-SF4F`

  Авторы рассматривают как теоретические аспекты, так и результаты моделирования, чтобы показать, что сдвиги уровня или тренда независимо сгенерированных наборов случайных данных проявляются в виде ложных корреляций.

  C.W.J. Granger and P. Newbold, "Spurious Regressions in Econometrics," Journal of Econometrics 2, no. 2 (1974): 111–20, `https://perma.cc/M8TE-AL6U`

  Эта статья по эконометрике удостоена Нобелевской премии за описание изначальных трудностей выявления ложных корреляций и обоснование надежного способа определения взаимосвязей во временных рядах.

- Исследовательские методы анализа данных

  David R. Brillinger and Mark A. Finney, "An Exploratory Data Analysis of the Temperature Fluctuations in a Spreading Fire," Environmetrics 25, no. 6 (2014): 443–53, `https://perma.cc/QB3D-APKM`

  Яркий пример анализа одних и тех же лабораторных данных, имеющих географическую и временную привязку, всевозможными научными и правительственными организациями.

  Robert H. Shumway and David S. Stoffer, "Time Series Regression and Exploratory Data Analysis," in Time Series Analysis and Its Applications with R Examples (New York: Springer, 2011), `https://perma.cc/UC5B-TPVS`

  Глава, посвященная исследовательским методам анализа данных, из канонического труда по анализу временных рядов для аспирантов.

- Визуализация данных

  Christian Tominski and Wolfgang Aigner, "The TimeViz Browser," `https://perma.cc/94ND-6ZA5`

  Потрясающий каталог примеров (и исходного кода) визуализации самых разных временных рядов, освещенных как в научных публикациях, так и в отраслевых работах.

  Oscar Perpinan Lamigueiro, "GitHub Repository for Displaying Time Series, Spatial, and Space-time Data with R," `https://perma.cc/R69Y-5JPL`

  Содержит примеры реализации на языке R задач визуализации временных рядов, в том числе наборов геопространственных данных.

  Myles Harrison, "5 Ways to Do 2D Histograms in R," R-bloggers, September 1, 2014, `https://perma.cc/ZCX9-FQQY`

  Практическое руководство по многочисленным инструментам пакетов языка R построения и цветового форматирования двумерных гистограмм. Наряду со стандартными средствами в нем детально описывается пакет `tidyquant`, применяемый для визуализации данных фондового рынка — очень важного источника временных рядов.

- Тренды

  Halbert White and Clive W.J. Granger, "Considerations of Trends in Time Series," Journal of Time Series Econometrics 3, no. 1 (2011), `https://perma.cc/WF2H-TVTL`

  Относительно недавняя научная статья, показывающая, что, хотя тренды часто встречаются в данных, даже традиционная статистика не имеет четкого набора определений для их описания. В доступном стиле авторы приводят статистическую информацию о присутствующих в нестационарных данных трендах и рекомендации по улучшению статистических методов их обработки.

ГЛАВА 4

Моделирование временных рядов

До настоящего момента нами обсуждались только вопросы сбора и извлечения временных рядов, а также их обработки. В этой главе речь пойдет о создании временных ряды в результате моделирования.

Нам предстоит обсудить три основных вопроса. Во-первых, сравнить принципы моделирования временных рядов с таковыми в других способах моделирования данных, отмечая особенности процесса, вызванные необходимостью учета течения времени. Во-вторых, изучить несколько программных примеров моделирования. И в-третьих, обсудить важные тенденции моделирования временных рядов.

Основная часть этой главы будет посвящена примерам кода, отвечающего за генерацию временных рядов разных типов. Среди них выделяются следующие наиболее важные ситуации.

- Моделирование поведения членов благотворительной организации, заключающегося в изучении почтовых сообщений и внесении добровольных пожертвований в течение всего срока членства в организации. Пример напрямую связан с данными, которые были описаны в главе 2.
- Моделирование событий в таксопарке, насчитывающем 1000 автомобилей, каждый из которых описывается с разным временем начала смены и количеством перевозимых пассажиров в течение рабочего дня.
- Пошаговое моделирование состояния твердого магнитного тела при заданной температуре и размерах, описываемого строгими физическими законами.

Каждый пример подчиняется одному из следующих способов моделирования временных рядов.

Эвристическое моделирование

Мы сами решаем, как должен работать мир, обеспечиваем его "разумность" и самостоятельно пишем код управления таким миром.

Моделирование дискретных событий

Создаем объекты системы, подчиняющиеся определенным правилам, а затем активизируем их, чтобы посмотреть, как такая система будет эволюционировать с течением времени.

Моделирование физических систем

Применяем физические законы, чтобы отслеживать поведение системы с течением времени.

Моделирование временных рядов — это важная аналитическая задача, примеры выполнения которой неоднократно рассматриваются в последующих главах при изучении конкретных моделей.

Особенности моделирования временных рядов

Моделирование данных выступает разновидностью анализа данных, которому редко обучают, но который находит широкое применение при работе с временными рядами. Это следует из одного из недостатков временных данных: никакие две точки данных в одном и том же временном ряду не могут быть точно сопоставимыми, поскольку они относятся к разному времени. Как только мы задумаемся о том, *что могло бы произойти в данный момент времени*, сразу же перейдем в область моделирования.

Моделирование может описываться как простым, так и сложным процессами. С одной стороны, смоделированные данные можно встретить в любом учебнике по статистике, например при изучении случайного блуждания. Обычно они генерируются в виде накопительных сумм некого случайного процесса (получаемого, например, с помощью пакета `rnorm` языка R) или периодической функции (в частности, синусоиды). С другой стороны, многие ученые и инженеры делают карьеру, изучая и описывая технологии моделирования данных временных рядов. Моделирование временных рядов остается активной областью исследований (необычайно требовательной к производительности вычислительного оборудования) во многих отраслях и дисциплинах.

- Метеорология.
- Финансы.
- Эпидемиология.
- Квантовая химия.
- Физика плазмы.

В некоторых из представленных случаев фундаментальные правила поведения системы хорошо изучены, но описать их бывает трудно из-за высокой сложности уравнений (например, в метеорологии, квантовой химии, физике плазмы). Кроме того, далеко не всегда в системе известны все исследуемые величины, а сделать точный прогноз не представляется возможным из-за стохастической нелинейной природы изучаемых процессов (например, в финансах и эпидемиологии).

Моделирование и прогнозирование

Моделирование и прогнозирование — схожие задачи. В обоих случаях сначала нужно сформулировать гипотезу о параметрах и поведении базовой системы, а затем экстраполировать имеющиеся данные для получения новых точек.

Наряду с этим нужно четко понимать различия между принципами моделирования и прогнозирования.

- Иногда качественные наблюдения проще обрабатывать методами моделирования, а не прогнозирования.
- Моделирование выполняется в определенном масштабе, что позволяет увидеть множество альтернативных сценариев, в то время как прогнозы составляются предельно точно.
- Моделирование — не такое рискованное занятие, как прогнозирование. Моделирование не влияет на события реальной жизни и не вызывает потерь важных ресурсов — в первых сеансах моделирования вы можете проявлять максимум фантазии и изобретательности. Конечно, рано или поздно вам захочется обосновать модель точно так же, так это делается в прогнозировании.

Моделирование с помощью программного кода

Далее мы рассмотрим три примера программ моделирования временны*х* рядов. Изучая их, постарайтесь понять, насколько большими могут быть наборы данных, используемые в качестве источников "временны*х* рядов", генерируемых в процессе моделирования. Не забывайте, что периодичность временно*й* выборки может устанавливаться предельно точно и зависеть от действий человека (например, день и время совершения добровольных пожертвований) или же носить неустановленный характер и задаваться с помощью неопределенных утверждений (*n*-й шаг физического моделирования).

В этом разделе нам предстоит рассмотреть три следующих примера моделирования.

- Моделирование искусственного набора данных для проверки гипотезы о поведении членов организации, в частности о том, что их отношение к электронным письмам, полученным от благотворительной организации, коррелирует (или нет) с готовностью вносить благотворительные пожертвования. Это яркий пример жесткого генерирования исходных табличных данных и определения взаимосвязей между данными, устанавливаемых в программном коде с помощью циклов `for` и подобных им структур.

- Моделирование синтезированного набора данных, используемого для изучения совокупного поведения водителей в таксопарке с учетом рабочего графика и количества перевозимых пассажиров в зависимости от времени суток. В этом наборе данных акцент делается на объектно-ориентированных атрибутах языка Python и генераторах, которые оказываются незаменимыми при тестировании и изучении поведения системы.
- Моделирование физического поведения магнитного материала, заключающегося в переориентировании отдельных магнитных элементов, которые вначале пребывают в беспорядке, но в конечном итоге объединяются в хорошо упорядоченную систему. В этом примере показано, что принципы моделирования временны*х* рядов могут основываться на физических законах, предопределяющих временной масштаб и другие характеристики модели.

Самостоятельная работа

При написании программы моделирования необходимо помнить о логических правилах, определяющих поведение системы. Ниже рассматривается пример программы, в которой выполняется жесткая проверка логической целостности данных (например, из системы исключаются события, которые противоречат ее логике).

В начале кода проверяется состояние членства в организации — устанавливается количество членов благотворительной организации и моментов времени вступления в нее. Каждый из членов организации участника получает определенный статус.

```
## Python
>>> ## Статус членов организации
>>> years        = ['2014', '2015', '2016', '2017', '2018']
>>> memberStatus = ['bronze', 'silver', 'gold', 'inactive']

>>> memberYears  = np.random.choice(years, 1000,
>>>                 p = [0.1, 0.1, 0.15, 0.30, 0.35])
>>> memberStats = np.random.choice(memberStatus, 1000,
>>>                 p = [0.5, 0.3, 0.1, 0.1])
>>> yearJoined = pd.DataFrame({'yearJoined': memberYears,
>>>                 'memberStats': memberStats})
```

В приведенных выше строках кода сделано несколько предположений и реализовано несколько важных правил моделирования процесса. В них задаются вероятности вступления в организацию в каждом анализируемом году. Кроме того, здесь предполагается, что статус члена организации никак не связан с годом вступления в организацию. В реальном мире ситуация выглядит несколько иначе — более точным будет предположение о существовании слабой связи

между этими переменными, особенно в случаях заинтересованности организации в сохранении своих членов.

Далее в коде создается таблица со сведениями об открытии электронных писем членами организации в течение каждой недели. В данном случае считается, что организации свойственно следующее поведение: рассылка всем членам по три электронных письма в неделю. Кроме того, в коде определяются возможные поведенческие реакции членов организации на получение почтовых сообщений.

- Никогда не открывает электронную почту
- Максимальный уровень взаимодействия (постоянно открывает электронную почту)
- Увеличение или уменьшение уровня взаимодействия

Мы можем усложнить поведенческие правила, корректируя отдельные параметры модели в зависимости от принятия старых или создания новых предположений о ненаблюдаемых процессах, влияющих на данные.

```
## Python
>>> NUM_EMAILS_SENT_WEEKLY = 3

>>> ## Определение функций для разных поведенческих шаблонов
>>> def never_opens(period_rng):
>>>     return []
>>> def constant_open_rate(period_rng):
>>>    n, p = NUM_EMAILS_SENT_WEEKLY, np.random.uniform(0, 1)
>>>    num_opened = np.random.binomial(n, p, len(period_rng))
>>>    return num_opened
>>> def increasing_open_rate(period_rng):
>>>    return open_rate_with_factor_change(period_rng,
>>>                                        np.random.uniform(1.01,
>>>                                                          1.30))
>>> def decreasing_open_rate(period_rng):
>>>    return open_rate_with_factor_change(period_rng,
>>>                                        np.random.uniform(0.5,
>>>                                                          0.99))
>>> def open_rate_with_factor_change(period_rng, fac):
>>>     if len(period_rng) < 1 :
>>>        return []
>>>     times = np.random.randint(0, len(period_rng),
>>>                               int(0.1 * len(period_rng)))
>>>     num_opened = np.zeros(len(period_rng))
>>>     for prd in range(0, len(period_rng), 2):
>>>         try:
>>>             n, p = NUM_EMAILS_SENT_WEEKLY, np.random.uniform(0,
>>>                                                              1)
>>>             num_opened[prd:(prd + 2)] = np.random.binomial(n, p,
>>>                                                            2)
```

```
>>>             p = max(min(1, p * fac), 0)
>>>         except:
>>>             num_opened[prd] = np.random.binomial(n, p, 1)
>>>     for t in range(len(times)):
>>>         num_opened[times[t]] = 0
>>>     return num_opened
```

Выше определены функции моделирования четырех различных типов поведения.

Пользователи, никогда не открывающие отправленные им электронные письма

```
never_opens()
```

Пользователи, открывающие примерно одинаковое количество писем каждую неделю

```
constant_open_rate()
```

Пользователи, открывающие меньшее количество электронных писем каждую неделю

```
decreasing_open_rate()
```

Пользователи, открывающие большее количество электронных писем каждую неделю

```
increasing_open_rate()
```

Поведение членов организации, которые с течением времени становятся более активными или, наоборот, более пассивными, описывается подобным образом с помощью функции `open_rate_with_factor_change()`, основанной на функциях `increasing_open_rate()` и `decreasing_open_rate()`.

Далее нам нужно определиться с моделью внесения добровольных пожертвований. Она не должна быть слишком простой, иначе это приведет к неправильному пониманию описываемых процессов. Другими словами, в модель внесения пожертвований нужно включить текущие предположения о поведении членов организации, а затем проверить, соответствуют ли они модели и наблюдениям, представленным реальными данными. Ниже приведен код описания процесса внесения пожертвований, находящегося в слабой зависимости от количества писем, открываемых членом организации в течение недели.

```
## Python
>>> ## Внесение пожертвований
>>> def produce_donations(period_rng, member_behavior, num_emails,
>>>                       use_id, member_join_year):
>>>     donation_amounts = np.array([0, 25, 50, 75, 100, 250, 500,
>>>                                  1000, 1500, 2000])
>>>     member_has = np.random.choice(donation_amounts)
>>>     email_fraction = num_emails /
>>>                      (NUM_EMAILS_SENT_WEEKLY * len(period_rng))
```

```
>>>     member_gives = member_has * email_fraction
>>>     member_gives_idx = np.where(member_gives
>>>                                 >= donation_amounts)[0][-1]
>>>     member_gives_idx = max(min(member_gives_idx,
>>>                                len(donation_amounts) - 2),
>>>                            1)
>>>     num_times_gave = np.random.poisson(2) *
>>>                          (2018 - member_join_year)
>>>     times = np.random.randint(0, len(period_rng), num_times_gave)
>>>     dons = pd.DataFrame({'member' : [],
>>>                          'amount' : [],
>>>                          'timestamp': []})

>>> for n in range(num_times_gave):
>>>     donation = donation_amounts[member_gives_idx
>>>                   + np.random.binomial(1, .3)]
>>>     ts = str(period_rng[times[n]].start_time
>>>              + random_weekly_time_delta())
>>>     dons = dons.append(pd.DataFrame(
>>>                 {'member' : [use_id],
>>>                  'amount' : [donation],
>>>                  'timestamp': [ts]}))
>>>
>>> if dons.shape[0] > 0:
>>>     dons = dons[dons.amount != 0]
>>>     ## Нулевые пожертвования не указываются, поскольку
>>>     ## в реальной базе данных они не регистрируются
>>>
>>> return dons
```

Чтобы сделать модель более реалистичной, в коде реализуются следующие важные предположения.

- Общая сумма пожертвований зависит от общего срока членства в организации.
- Каждому из членов организации назначается уровень благосостояния, с помощью которого обыгрывается предположение о том, что сумма денежного взноса напрямую зависит от суммы, которую человек готов выделить на добровольные пожертвования.

Поскольку поведение наших участников строго фиксируется по временны*м* меткам, каждому члену организации нужно определить неделю и день внесения пожертвований. Напишем вспомогательную функцию выбора случайного времени в течение недели.

```
## Python
>>> def random_weekly_time_delta():
>>>     days_of_week = [d for d in range(7)]
```

```
>>>     hours_of_day = [h for h in range(11, 23)]
>>>     minute_of_hour = [m for m in range(60)]
>>>     second_of_minute = [s for s in range(60)]
>>>     return pd.Timedelta(str(np.random.choice(days_of_week))
>>>                         + " days" ) +
>>>         pd.Timedelta(str(np.random.choice(hours_of_day))
>>>                         + " hours" ) +
>>>         pd.Timedelta(str(np.random.choice(minute_of_hour))
>>>                         + " minutes") +
>>>         pd.Timedelta(str(np.random.choice(second_of_minute))
>>>                         + " seconds")
```

Легко заметить, что в приведенном выше коде обрабатываются только часы временной метки в диапазоне от 11 до 23 (`hours_of_day = [h for h in range (11, 23)]`). Предполагается, что все члены организации проживают в смежных часовых поясах или даже в общем часовом поясе, поэтому мы не обрабатываем часы вне указанного диапазона. Таким способом мы заметно ограничиваем базовую модель поведения пользователей.

Итак, мы ожидаем увидеть унифицированное поведение наших пользователей, как если бы они все находились в одном или нескольких смежных часовых поясах. Кроме того, считается, что людям свойственно переводить пожертвования в дневное время — с позднего утра до позднего вечера, но не ночью и не ранним утром, сразу же после пробуждения.

Наконец, нам нужно объединить все компоненты модели и смоделировать поведение определенного количества членов организации и соответствующих событий, возникающих только вследствие получения членства в организации, таким образом, чтобы обеспечить некоторое (но не пренебрежимо малое) влияние событий просмотра электронной почты на события внесения пожертвований.

```
## Python
>>> behaviors                = [never_opens,
>>>                             constant_open_rate,
>>>                             increasing_open_rate,
>>>                             decreasing_open_rate]
>>> member_behaviors = np.random.choice(behaviors, 1000,
>>>                                     [0.2, 0.5, 0.1, 0.2])
>>> rng = pd.period_range('2015-02-14', '2018-06-01', freq = 'W')
>>> emails = pd.DataFrame({'member' : [],
>>>                        'week' : [],
>>>                        'emailsOpened': []})
>>> donations = pd.DataFrame({'member' : [],
>>>                           'amount' : [],
>>>                           'timestamp': []})
>>> for idx in range(yearJoined.shape[0]):
>>>     ## Генерирование случайного момента получения членства
>>>     ## в организации
>>>     join_date = pd.Timestamp(yearJoined.iloc[idx].yearJoined) +
```

```
>>>                 pd.Timedelta(str(np.random.randint(0, 365)) +
>>>                                  ' days')
>>>     join_date = min(join_date, pd.Timestamp('2018-06-01'))
>>>
>>>     ## До вступления в организацию временные метки не предусмотрены
>>>     member_rng = rng[rng > join_date]
>>>
>>>     if len(member_rng) < 1:
>>>        continue
>>>
>>>     info = member_behaviors[idx](member_rng)
>>>     if len(info) == len(member_rng):
>>>       emails = emails.append(pd.DataFrame(
>>>            {'member': [idx] * len(info),
>>>             'week': [str(r.start_time) for r in member_rng],
>>>             'emailsOpened': info}))
>>>       donations = donations.append(
>>>            produce_donations(member_rng, member_behaviors[idx],
>>>                              sum(info), idx, join_date.year))
```

Теперь нужно изучить временное поведение доноров, чтобы понять, как можно использовать эту информацию в дальнейшем анализе или прогнозировании. Отобразим общую сумму ежемесячных добровольных пожертвований, вычисленных из набора данных (рис. 4.1).

```
## Python
>>> df.set_index(pd.to_datetime(df.timestamp), inplace = True)
>>> df.sort_index(inplace = True)
>>> df.groupby(pd.Grouper(freq='M')).amount.sum().plot()
```

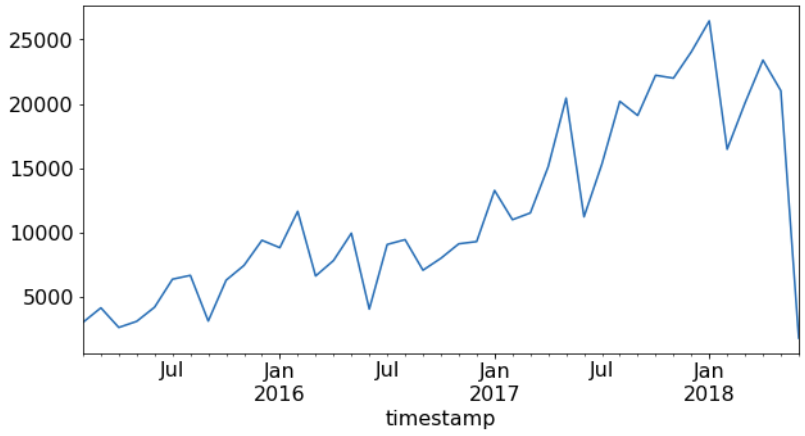


Рис. 4.1. *Общая сумма ежемесячных взносов, учитываемых в наборе данных*

Как видите, объем пожертвований и количество открытых электронных писем увеличились с 2015 по 2018 год. Это неудивительно, поскольку за этот же период у организации прибавилось членов, о чем свидетельствует общее количество новых членов, присоединившихся к организации за учитываемые годы. Фактически одно из допущений нашей модели заключалось в том, что мы должны сохранять информацию о каждом члене неопределенный длительный период времени после его вступления в организацию. Мы не предусмотрели возможность выхода из организации, хотя и учли, что ее члены могут потерять интерес к благотворительной деятельности (отказ от просмотра электронных писем). Однако даже в этом случае за ними сохраняется возможность внесения пожертвований. Предположение о бесконечно продолжающемся членстве в организации (и коррелированном поведении пожертвований) прекрасно отображено на рис. 4.1. Таким образом, нам следует доработать код, чтобы исключить из него нереалистичный сценарий неограниченного членства и срока внесения пожертвований.

В нашем примере мы несколько отходим от классических принципов моделирования временны*х* рядов и занимаемся скорее генерированием табличных данных. Но не нужно забывать и том, что нашей конечной целью было получение временны*х* рядов. Поэтому генерирование табличных данных выполнялось с учетом следующих факторов и предположений.

- Общее количество временны*х* рядов.
- Моделируемые временны*е* тренды.
  - — В случае электронной почты мы будем придерживаться трех трендов: постоянной, растущей и падающей частот просмотра электронной почты.
  - — Внесение добровольных пожертвований считается шаблонным поведением, коррелирующим с количеством электронных писем, просматриваемым членом организации за анализируемый период времени. Это порождает упреждение, но, поскольку данные генерируются, а не собираются, таким способом описывается лояльность к организации — чем чаще просматриваются электронные письма, тем больше частота внесения пожертвований.
- События просмотра электронных писем и внесения пожертвований должны приниматься в расчет только после получения членства в организации.
- Данные не должны уходить в будущее в угоду их конечным потребителям, даже несмотря на то, что в моделировании такие данные всегда приветствуются.

Конечно, полученный нами код не идеален. Он громоздкий и не реализует модель в задуманном виде. Более того, поскольку логика модели проверялась только на программном уровне, в ней могут встречаться ошибки пограничного

характера (возникновения событий в нелогической последовательности). Для предотвращения ошибок такого типа, прежде чем приступить к разработке модели, нужно выработать общие показатели и стандарты ее достоверности.

Итак, нам требуется программное обеспечение, которое сделает модель логичной и целостной. За помощью обратимся к генераторам языка Python.

Создание самоуправляемой среды моделирования

При решении многих задач от вас может потребоваться установить правила для уже имеющейся системы и понаблюдать за их соблюдением. Таким способом, например, можно проанализировать цели использования приложения широким кругом независимых пользователей или протестировать внутреннюю систему принятия решений на устойчивость к внешним воздействиям. Иными словами, вы можете отслеживать вклад отдельных агентов в совокупные показатели с течением времени. Язык Python особенно хорошо подходит для решения такого рода задач благодаря поддержке генераторов. Переходя от анализа данных к написанию программного обеспечения, имеет смысл обратиться к Python, даже если вы чувствуете себя более комфортно, программируя на R.

Генераторы позволяют создать серию независимых (или зависимых!) объектов и активизировать их для изучения имеющихся возможностей без написания слишком громоздкого кода отслеживания происходящих изменений.

В следующем примере мы будем моделировать поведение таксопарка. Попробуем представить, как функционирует таксопарк, в котором рабочие смены начинаются в разное время. Для этого создадим определенное количество виртуальных такси, отправим их в виртуальный город и заставим отчитываться о проделанной работе.

Такое моделирование представляет собой исключительно сложную задачу. В демонстрационных целях признаем, что будем создавать более простую среду, чем та, которая, как нам кажется, отражает действительность (не забывайте о том, что все модели неправдоподобны в той или иной степени). Начнем с изучения понятия генератора в языке Python.

Рассмотрим метод, используемый мною для получения идентификационного номера такси.

```
## Python
>>> import numpy as np

>>> def taxi_id_number(num_taxis):
>>> arr = np.arange(num_taxis)
>>> np.random.shuffle(arr)
>>> for i in range(num_taxis):
>>> yield arr[i]
```

Тем, кто не знаком с генераторами, будет интересно ознакомиться с результатом выполнения кода.

```
## Python
>>> ids = taxi_id_number(10)
>>> print(next(ids))
>>> print(next(ids))
>>> print(next(ids))
```

На экран выводится последовательность чисел.

```
7
2
5
```

Числа повторяются до тех пор, пока их не станет 10, после чего реализуется выход из цикла `for` по исключению `StopIteration`.

Функция `taxi_id_number()` создает разовые объекты, не зависящие друг от друга, но сохраняющие сведения о состоянии ее выполнения. Это функция-генератор. Вы можете рассматривать генераторы как небольшие объекты, снабженные специальными переменными состояния, которые оказываются незаменимыми при создании множества схожих объектов с собственными значениями таких переменных.

В задаче моделирования таксопарка разделим подвижной состав такси на смены, индикация которых выполняется с помощью генератора. Предполагается, что в середине дня на маршруты выезжает больше машин, чем в вечерние или ночные смены, определяемые разными вероятностями начала смены в указанные моменты времени.

```
## Python
>>> def shift_info():
>>>     start_times_and_freqs = [(0, 8), (8, 30), (16, 15)]
>>>     indices =               np.arange(len(start_times_and_freqs))
>>>     while True:
>>>         idx =   np.random.choice(indices, p = [0.25, 0.5, 0.25])
>>>         start = start_times_and_freqs[idx]
>>>         yield (start[0], start[0] + 7.5, start[1])
```

Обратите внимание на переменную `start_times_and_freqs`. С ее определения начинается моделирование временного ряда. В нем указываем, что вероятность выхода такси на маршрут зависит от времени суток. Кроме того, каждому времени суток сопоставляется свое среднее количество поездок.

Перейдем к написанию более сложного генератора, в котором предыдущие генераторы используются для установки индивидуальных параметров и расписания такси.

```
## Python
>>> def taxi_process(taxi_id_generator, shift_info_generator):
>>>     taxi_id = next(taxi_id_generator)
```

```
>>>     shift_start, shift_end, shift_mean_trips =
>>>                                 next(shift_info_generator)
>>>     actual_trips = round(np.random.normal(loc = shift_mean_trips,
>>>                                           scale = 2))
>>>     average_trip_time = 6.5 / shift_mean_trips * 60
>>>     # Среднее время поездки в минутах
>>>     between_events_time = 1.0 / (shift_mean_trips - 1) * 60
>>>     # Город, в котором такси освобождается от клиента
>>>     time = shift_start
>>>     yield TimePoint(taxi_id, 'start shift', time)
>>>     deltaT = np.random.poisson(between_events_time) / 630
>>>     time += deltaT
>>>     for i in range(actual_trips):
>>>        yield TimePoint(taxi_id, 'pick up ', time)
>>>        deltaT = np.random.poisson(average_trip_time) / 60
>>>        time += deltaT
>>>        yield TimePoint(taxi_id, 'drop off ', time)
>>>        deltaT = np.random.poisson(between_events_time) / 60
>>>        time += deltaT
>>>     deltaT = np.random.poisson(between_events_time) / 60
>>>     time += deltaT
>>>     yield TimePoint(taxi_id, 'end shift ', time)
```

Здесь с помощью генераторов сначала определяются идентификационный номер такси, время начала смены и среднее количество поездок.

После этого такси отправляется по маршруту, выполняя по сформированному для него расписанию определенное количество поездок, каждая из которых совершается для следующего клиента по вызову функции `next()` из генератора. По сути, этот генератор создает временны*е* ряды данных для отдельного такси.

Генератор такси возвращает объекты класса `TimePoint`, которые определяются следующим образом.

```
## Python
>>> from dataclasses import dataclass

>>> @dataclass
>>> class TimePoint:
>>>     taxi_id: int
>>>     name: str
>>>     time: float

>>>     def __lt__(self, other):
>>>         return self.time < other.time
```

Для упрощения кода в нем используется декоратор `dataclass()` (требуется Python 3.7). Настоятельно рекомендуется ознакомиться с возможностями этого вспомогательного инструмента языка Python всем заинтересованным специалистам по анализу и обработке данных, применяющим его в своих проектах.

Магические методы в Python

Магические (или дандер) методы в языке Python, имена которых начинаются и заканчиваются двойным подчеркиванием, представляют собой встроенные методы, характерные для целевого класса. Такие методы вызываются автоматически при обращении к соответствующему объекту. Существуют реализации таких методов, которые могут быть переопределены при ручном задании для класса. Существует ряд причин, по которым вы можете захотеть сделать это. Например, в предыдущем коде нужно, чтобы сравнение объектов класса `TimePoints` выполнялось только по времени, но не по атрибуту `taxi_id` или `name`.

Термин "дандер" (dunder) происходит от "double under" (двойное подчеркивание).

Кроме автоматически сгенерированного инициализатора класса `TimePoint`, нам понадобятся два других, более сложных метода: `__lt__` (сравнение объектов класса `TimePoints`) и `__str__` (вывод объектов класса `TimePoints`, не реализованный здесь). Сравнение необходимо, поскольку сгенерированные в процессе моделирования временны*е* точки объединяются в общую структуру данных, в которых они хранятся в определенном порядке — в очереди по приоритету. *Очередь по приоритету* — это абстрактный тип данных, в который объекты можно добавлять в любом порядке, но в котором они выстраиваются в порядке, устанавливаемом их приоритетом.

Абстрактный тип данных

Абстрактный тип данных — это вычислительная модель, определяемая поведением, которая характеризуется заранее оговоренным набором действий и входных данных, а также результатами применения таких действий к определенным наборам данных.

Пример общеизвестного абстрактного типа данных — FIFO ("первым пришел — первым обслужен"). В нем объекты извлекаются из структуры данных в том же порядке, в котором были в нее добавлены. Способы достижения такого принципа зависят от методов реализации, а не от его определения.

Итак, мы располагаем классом моделирования для настройки и запуска генераторов такси. Это нестандартный класс, поскольку обладает специальными функциями — в том числе заданными в инициализаторе — представления входных данных в виде понятной структуры данных и дальнейшей их обработки. Заметьте, что при этом единственной общедоступной является функция `run()`.

```
## Python
>>> import queue
>>> class Simulator:
>>>    def __init__(self, num_taxis):
>>>        self._time_points = queue.PriorityQueue()
>>>        taxi_id_generator = taxi_id_number(num_taxis)
>>>        shift_info_generator = shift_info()
>>>        self._taxis = [taxi_process(taxi_id_generator,
>>>                                    shift_info_generator) for
>>>                                          i in range(num_taxis)]
>>>        self._prepare_run()

>>> def _prepare_run(self):
>>>     for t in self._taxis:
>>>        while True:
>>>            try:
>>>                e = next(t)
>>>                self._time_points.put(e)
>>>            except:
>>>                break

>>> def run(self):
>>>    sim_time = 0
>>>    while sim_time < 24:
>>>        if self._time_points.empty():
>>>          break
>>>        p = self._time_points.get()
>>>        sim_time = p.time
>>>        print(p)
```

Сначала в коде создается точно такое количество генераторов, какое требуется для моделирования заданного числа машин такси. Затем совершается проход по каждому такси, связанному с классом `TimePoints`, в результате которого все объекты класса `TimePoints` располагаются в порядке приоритетности. В пользовательском классе, таком как `TimePoint`, приоритет объекта определяется в его инициализаторе `__lt__` сравнением текущей временной точки с начальной точкой. Таким образом, при размещении согласно приоритетности объекты класса `TimePoints` упорядочиваются по времени их создания.

Процесс моделирования запускается с помощью следующих команд.

```
## Python
>>> sim = Simulator(1000)
>>> sim.run()
```

Ниже показан результат моделирования (в вашем случае он будет другим, так как мы не установили начальное значение — при каждом перезапуске кода оно выбирается случайным образом).

```
id: 0539 name: drop off   time: 23:58
id: 0318 name: pick up    time: 23:58
id: 0759 name: end shift  time: 23:58
id: 0977 name: pick up    time: 23:58
id: 0693 name: end shift  time: 23:59
id: 0085 name: end shift  time: 23:59
id: 0351 name: end shift  time: 23:59
id: 0036 name: end shift  time: 23:59
id: 0314 name: drop off   time: 23:59
```

Начальное значение генератора случайных чисел

Создавая код, отвечающий за генерирование случайных чисел, вам может понадобиться сделать его полностью воспроизводимым (например, планируя проводить модульное тестирование, которое обычно выполняется со случайными настройками, или отладку кода со суженным диапазоном изменяемых параметров). Чтобы обеспечить генерирование случайных чисел в одном и том же (неслучайном) порядке, следует определить начальное значение процесса. Это общепринятая настройка — возможность установки начального значения представлена в большинстве языков программирования.

Для простоты отображения мы округлили временны*е* точки до ближайшей минуты, хотя могли остановиться на более точных значениях. Используемое временно*е* разрешение зависит от преследуемых целей.

- Если нужно учитывать влияние машин такси на общегородской трафик, то лучше остановиться на почасовых временны*х* точках.
- Создавая мобильное приложение вызова такси, нужно исходить из соображений вычислительной нагрузки на сервер. Скорее всего, придется довольствоваться минутной или более высокой разрешающей способностью, чтобы не перегружать сетевую и вычислительную инфраструктуру.

Ранее нами было принято решение регистрировать объекты класса `TimePoints` каждого такси в момент их "возникновения". Иначе говоря, мы сообщаем о начале поездки в такси (посадка пассажира), не указывая момента ее окончания, хотя в его определении нет ничего сложного. Это один из способов сделать временной ряд более реалистичным в том понимании, что регистрация событий в нем ведется в режиме реального времени.

Обратите внимание, что, как и в предыдущем случае, моделирование все еще не приводит к получению временны*х* рядов. Но в нашем распоряжении находится журнал событий, который может послужить основой для временно*го* ряда. Существует сразу несколько возможностей.

- Запись результатов моделирования в файл CSV или базу данных временных рядов.
- Подключение к нашей модели одной из онлайн-моделей, обеспечивающих разработку конвейера потоковой обработки данных в режиме реального времени.
- Сохранение выходных данных в файл или базу данных с последующей обработкой (с риском получения упреждения) для представления в более удобном формате, например совмещение точек начала и окончания поездок, чтобы получить представление об их длительности в разное время суток.

Наряду с проверкой гипотезы о динамичности системы управления таксопарком моделирование таких данных позволяет ответить на некоторые другие вопросы. В частности, синтезированные временные ряды могут пригодиться в следующих ситуациях.

- Тестирование метрик различных моделей прогнозирования по известной динамической модели
- Построение конвейера реальных данных, который в их отсутствие (ожидание) отлаживается на синтезированных данных

Используя генераторы и другие преимущества объектно-ориентированного программирования, вы получаете в свое распоряжение более совершенные инструменты анализа временных рядов. Рассмотренный ваше пример показывает, что генераторы позволяют упростить исходную модель, а также заметно улучшить программный код.

Преимущества моделирования, основанного на использовании агентов

Продемонстрированное в этом разделе решение оказалось верным, но в нем проводится слишком много проверок данных на соответствие логическим правилам, принятым в системе. Если целью моделирования дискретных событий, основанных на действиях дискретных акторов, является получение источников данных для временных рядов, то рассмотрите возможность его выполнения с помощью специальных модулей, в первую очередь — с помощью модуля `SimPy` (`https://simpy.readthedocs.io/en/latest`), снабженного удобным программным интерфейсом и обладающего достаточной гибкостью для моделирования процессов, описанных в этом разделе.

Моделирование физических процессов

В еще одном сценарии моделирования поведение системы определяется физическими законами, управляемыми по собственному усмотрению. Это не обязательно должны быть физические системы, — рассмотренные далее принципы справедливы для систем многих других типов.

- Специалисты, занимающиеся количественным финансовым анализом, утверждают, что поведение рынков прекрасно описывается "физическими" законами. Это же утверждение справедливо для экономических систем, хотя и в несколько иных временны*х* масштабах.
- Психологи настаивают на том, что общесоциальные принципы принятия решений основаны на "психофизических" принципах, которые могут послужить основой для "физических" правил модели, описывающей ожидаемую человеческую реакцию на различные поведенческие ситуации, изменяющиеся во времени.
- Биологи отслеживают временны*е* изменения во многих экосистемах, подверженных самым разным воздействиям.

Рассмотрим простой пример моделирования физической системы, заключающийся в изучении поведения магнитного материала. Для описания такой системы воспользуемся моделью, которая в статистической физике известна как модель Изинга.[1] Сразу заметим, что будем изучать поведение упрощенной модели Изинга. В ней предполагается, что в исходном состоянии отдельные компоненты магнитного материала имеют случайную направленность. Нам нужно инициализировать такую систему и посмотреть, как она будет эволюционировать к состоянию, в котором все магнитные компоненты выстраиваются в одном направлении под действием известных физических законов, устанавливаемых всего несколькими строками кода.

Мы также изучим принципы моделирования таких систем по методу Монте-Карло с использованием марковских цепей (MCMC — Markov Chain Monte Carlo), обсудим особенности метода и тонкости его применения в рамках модели Изинга.

Моделирование по методу Монте-Карло с использованием марковских цепей

Суть метода Монте-Карло состоит в описании процесса, представленного точной математической моделью, с помощью вероятностных характеристик.

[1]Модель Изинга является хорошо изученной математической моделью статистической физики, описывающей поведение магнитных материалов. В Интернете можно найти продолжение ее описания и огромное количество примеров кода ее реализации в прикладных задачах.

Марковские цепи являются полезным дополнением к методу Монте-Карло, в наибольшей степени востребованным в моделировании временных рядов. Сам по себе метод Монте-Карло позволяет узнать только распределение ряда данных, но не отследить его изменение во времени. Именно здесь в действие вступают марковские цепи. С их помощью вычисляются вероятности перехода между состояниями, — учитывая их, рассматриваем отдельные "шаги" процесса, а не просто вычисляем общий интеграл. В результате мы получаем возможность проводить моделирование временных рядов, а не просто вычисление интеграла.

В физике метод MCMC используется, например, для изучения влияния квантовых переходов в отдельных молекулах на совокупные ансамблевые измерения, проводимые в системе в разные моменты времени. В нашем случае система описывается следующими правилами.

1. В марковском процессе вероятность перехода к будущему состоянию зависит только от текущего состояния (не от прошлых состояний).
2. Система имеет физическое ограничение, выражаемое через распределение Больцмана для энергии: $T_{ij}/T_{ji} = e^{-b(E_j - E_i)}$. Его нужно воспринимать как отдельный принцип реализации, даже не пытаясь понять теоретическое обоснование (конечно, если вы не физик).

Моделирование по методу MCMC реализуется следующим образом.

1. Случайным образом выбираем начальное состояние каждого отдельного узла (элемента) решетки.
2. На каждого временном шаге выбираем отдельный узел решетки и изменяем его спин (направление магнитного поля).
3. Согласно известным физическим законам рассчитываем, насколько изменилась энергия элемента при переключении спина. Существует всего несколько возможностей.
   - Изменение энергии выражается отрицательным числом — осуществляется переход в более низкое энергетическое состояние, которое является наиболее предпочтительным вариантом. В этом случае состояние сохраняется и выполняется переход к следующему временному шагу.
   - Изменение энергии представляется неотрицательным значением — вероятность такого перехода рассчитывается как $e^{(-\text{изменение энергии})}$, что в полной мере соответствует правилу 2.

Продолжайте выполнять шаги 2 и 3 до завершения процесса — получения наиболее вероятного состояния для проводимого совокупного измерения.

Рассмотрим детали реализации модели Изинга. Представим материал, состоящий из множества объектов, расположенных в узлах двумерной сетки, каждый из которых обладает магнитным моментом (спином), характеризующимся направлением поля вверх или вниз. В начальный момент времени направление поля у всех объектов выбирается случайным образом. Нам нужно описать эволюцию системы таких объектов из случайного состояния в наиболее упорядоченное состояние, происходящую при низкой температуре.[2]

Инициализируем систему следующим образом.

```
## Python
>>> ### НАСТРОЙКА
>>> ## Физическая модель
>>> N = 5 # ширина сетки
>>> M = 5 # высота сетки
>>> ## Установка температуры
>>> temperature = 0.5
>>> BETA = 1 / temperature
```

Для случайной инициализации стартового блока обратимся к служебным методам.

```
>>> def initRandState(N, M):
>>>     block = np.random.choice([-1, 1], size = (N, M))
>>>     return block
```

Кроме того, нам нужно вычислить энергию выравнивания центрального блока относительно его соседей.

```
## Python
>>> def costForCenterState(state, i, j, n, m):
>>>     centerS = state[i, j]
>>>     neighbors = [((i + 1) % n, j), ((i - 1) % n, j),
>>>                  (i, (j + 1) % m), (i, (j - 1) % m)]
>>>     ## Обратите внимание на использование аргумента % n,
>>>     ## указывающего на периодичность условия.
>>>     ## Игнорируйте его, если в нем нет необходимости, -
>>>     ## это всего лишь физическое ограничение для двумерных
>>>     ## систем, которые можно описать через поверхность тора
>>>     interactionE = [state[x, y] * centerS for (x, y) in neighbors]
>>>     return np.sum(interactionE)
```

Определим намагниченность всего блока в данном состоянии.

```
## Python
>>> def magnetizationForState(state):
>>> return np.sum(state)
```

[2]Модель Изинга чаще используется для описания равновесных состояний ферромагнитных материалов, а не для изучения временны*х* аспектов их перехода в состояние равновесия. Тем не менее мы будем рассматривать временны*е* ряды, описывающие эволюцию системы во времени.

Этапы моделирования процесса по методу MCMC, рассматриваемому ранее, реализуются следующим образом.

```
## Python
>>> def mcmcAdjust(state):
>>>     n = state.shape[0]
>>>     m = state.shape[1]
>>>     x, y = np.random.randint(0, n), np.random.randint(0, m)
>>>     centerS = state[x, y]
>>>     cost = costForCenterState(state, x, y, n, m)
>>>     if cost < 0:
>>>       centerS *= -1
>>>     elif np.random.random() < np.exp(-cost * BETA):
>>>       centerS *= -1
>>>     state[x, y] = centerS
>>>     return state
```

Для запуска процесса моделирования нам потребуются функции регистрации данных и перенастройки модели MCMC.

```
## Python
>>> def runState(state, n_steps, snapsteps = None):
>>>     if snapsteps is None:
>>>        snapsteps = np.linspace(0, n_steps, num = round(n_steps /
>>>                                                        (M * N * 100)),
>>>           dtype = np.int32)
>>>     saved_states = []
>>>     sp = 0
>>>     magnet_hist = []
>>>     for i in range(n_steps):
>>>        state = mcmcAdjust(state)
>>>        magnet_hist.append(magnetizationForState(state))
>>>        if sp < len(snapsteps) and i == snapsteps[sp]:
>>>           saved_states.append(np.copy(state))
>>>           sp += 1
>>>     return state, saved_states, magnet_hist
```

Запуск моделирования

```
## Python
>>> ### МОДЕЛИРОВАНИЕ
>>> init_state = initRandState(N, M)
>>> print(init_state)
>>> final_state = runState(np.copy(init_state), 1000)
```

Чтобы сделать определенные выводы по результатам моделирования, нужно сравнить начальное и конечное состояния системы.

На рис. 4.2 показано случайно сгенерированное распределение начальных состояний. Хотя можно ожидать более равномерного их распределения, не забывайте, что при вероятностном подходе получить распределение состояний

в идеальном шахматном порядке почти невозможно. Попробуйте перегенерировать начальную сетку несколько раз, чтобы убедиться в том, что строгое "случайное" или абсолютное шахматное распределение крайне маловероятны. Тем не менее обратите внимание на то, что в начальный момент времени в каждом из двух возможных состояний находится примерно половина узлов. Примите к сведению, что любые шаблоны начальных состояний, которые только вам удастся распознать, будут всего лишь плодом воображения — человеческому мозгу свойственно находить известные ему картины везде, даже там, где их нет.

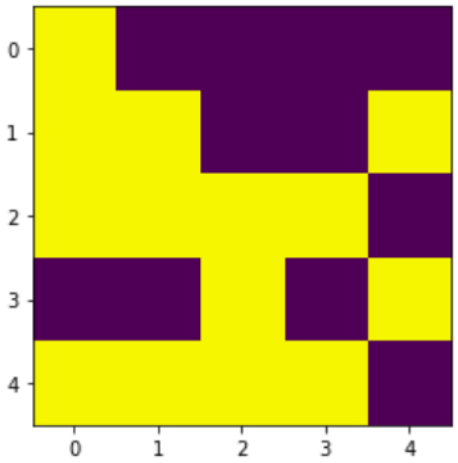


Рис. 4.2. *Начальное состояние сетки 5×5 узлов моделируемого ферромагнитного материала, в которой ориентация полей вверх или вниз выбирается случайно и с одинаковой вероятностью*

Передадим начальное состояние в функцию `runState()`, выполним моделирование для 1000 временны*х* шагов и изучим конечный результат, показанный на рис. 4.3.

Мы получили распределение состояний через 1000 временны*х* шагов. Результат моделирования позволяет сделать как минимум два интересных вывода. Во-первых, доминирующие состояния изменились на противоположные через 1000 шагов. Во-вторых, в числовом выражении текущее доминирующее состояние не так сильно, как противоположное доминирующее состояние в начальный момент времени. Это указывает на то, что под воздействием температуры узлы могут выходить из доминирующего состояния, даже если предыдущее состояние оказывается более предпочтительным. Чтобы лучше понять динамику процесса, нужно анализировать обобщенные характеристики, такие как намагниченность, или отслеживать временны*е* изменения в двумерных данных по специально созданным анимационным роликам.

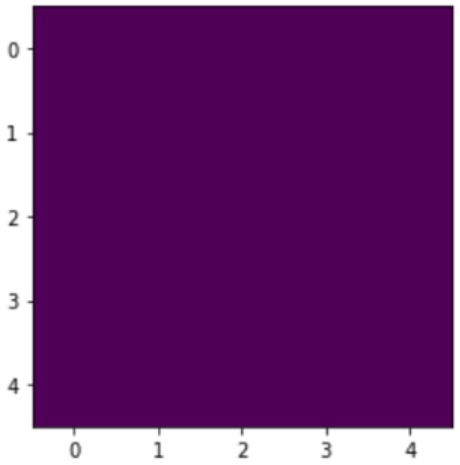


Рис. 4.3. *Конечное низкотемпературное состояние, полученное при моделировании 1000 шагов*

Определим намагниченность материала в разные моменты времени для большого количества независимых сеансов моделирования (рис. 4.4).

```
## Python
>>> Каждый временной ряд представляет отдельный результат
>>> results = []
>>> for i in range(100):
>>>     init_state = initRandState(N, M)
>>>     final_state, states, magnet_hist = runState(init_state, 1000)
>>>     results.append(magnet_hist)
>>>
>>> ## Построение графиков каждого ряда для оценки их перекрываемости
>>> for mh in results:
>>> plt.plot(mh,'r', alpha=0.2)
```

Изучение кривых намагничивания — это только один из способов отслеживания эволюции системы во времени. В качестве альтернативного варианта можно проводить исследование двумерных временных рядов, каждый из которых представляет данные состояния всей системы в отдельный момент времени. Или же можно проследить на каждом этапе моделирования за изменением других важных сводных характеристик, например энтропии или общей энергии. Заметьте, что в нашей модели намагниченность и энтропия являются связанными величинами, поскольку выступают функциями распределения состояний узлов решетки, но определяют совершенно разные физические характеристики.

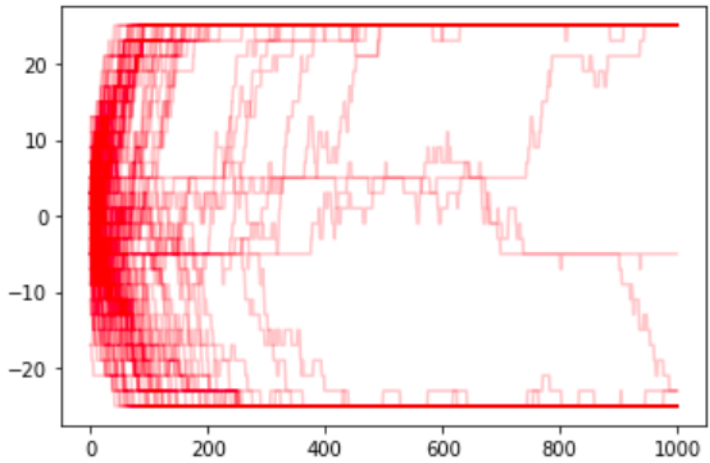


Рис. 4.4. *Сто независимых сеансов моделирования процесса изменения намагниченности системы при низкой температуре со случайным распределением состояний начальной сетки*

Полученные результаты можно использовать в дальнейших исследованиях подобно тому, как это было сделано при моделировании поведения таксопарка (несмотря на сильные различия в моделях). Повышенный интерес вызывают следующее задачи.

- Настройки конвейера данных на основе сгенерированных в результате моделирования временны*х* рядов
- Тестирование методов машинного обучения на синтезированных данных для изучения возможности их применения к собираемым данным до очистки и предварительной обработки
- Изучение важных характеристик физической модели для лучшего понимания базовой системы

Замечания по моделированию

Мы рассмотрели несколько примеров моделирования сильно различающихся систем, поведение которых изменяется во времени. Нашей основной задачей было построение моделей данных, описывающих поведение потребителей (внесение пожертвований членами благотворительной организации), городской инфраструктуры (службы такси) и замкнутой физической системы (упорядочение случайно направленного магнитного поля). После детального изучения этих примеров вы почувствуете себя комфортно, рассматривая программный код моделирования более сложных процессов, а также используя полученные результаты для решения практических задач.

Получив в свое распоряжение некий набор данных, вы, скорее всего, выдвигали предположения о его структуре и возможности практического использования, но не имели возможности их проверить. Моделирование данных позволяет получить ответы на очень многие вопросы. Результаты моделирования подтверждают или опровергают самые смелые предположения о данных, основанные на умозрительных рассуждениях и изучении количественных характеристик. Анализ полученных результатов открывает новые возможности не только в моделировании временны*х* рядов, но и в других дисциплинах науки о данных.

Статистическое моделирование

Статистическое моделирование — это традиционный способ изучения временны*х* рядов. Оно выполняется в задачах с известными динамическими характеристиками стохастической системы и позволяет оценить неизвестные параметры или понять, как сделанные предположения влияют на процесс оценки таких параметров (наглядный пример приведен в последующих главах). Для многих физических систем статистическое моделирование оказывается предпочтительнее других методов оценки.

Статистическое моделирование временны*х* рядов также находит широкое применение, например, в задачах нахождения количественных показателей оценки точности модели. Точность традиционных статистических моделей, таких как ARIMA (глава 6), рассчитывается согласно строго заданным аналитическим формулам. Это означает, что при описании систем с помощью известных статистических методов численную оценку дисперсии и точности модели можно получить без проведения большого количества сеансов моделирования.

Модели глубокого обучения

Моделирование временны*х* рядов методами глубокого обучения — это новая, но весьма многообещающая дисциплина науки о данных. С помощью технологий глубокого обучения во временны*х* рядах можно обнаружить сложные динамические процессы, зачастую скрытые для понимания даже опытных специалистов по анализу данных. Это же является недостатком технологии — чаще всего специалисты по анализу данных не имеют ни малейшего представления о принципах изучения обнаруженных динамических процессов.

Алгоритмы глубокого обучения предоставляют широкие возможности по защите исследуемых данных. Например, с помощью методов глубокого обучения можно синтезировать временны*е* ряды неоднородных данных, впоследствии обрабатываемых медицинскими приложениями, на основе реальных временны*х* рядов, уязвимых к утечке конфиденциальной информации. Такой защищенный набор данных, если он действительно будет получен, станет неоценимым источником медицинских сведений, лишенным недостатков исходных временны*х* рядов.

Дополнительные источники

Cristobal Esteban, Stephanie L. Hyland, and Gunnar Ratsch, "Real-Valued (Medical) Time Series Generation with Recurrent Conditional GANs," unpublished manuscript, last revised December 4, 2017, `https://perma.cc/Q69W-L44Z`

Авторы работы описывают принципы использования генеративно-состязательных сетей для синтезирования реалистично выглядящих разнородных медицинских временных рядов. Это пример использования методов глубокого обучения для получения этически выверенных, юридически обоснованных и (хочется верить) защищенных от утечек наборов медицинских данных, которые могут широко использоваться при разработке медицинских приложений, основанных на алгоритмах машинного и глубокого обучения.

Gordon Reikard and W. Erick Rogers, "Forecasting Ocean Waves: Comparing a Physics-based Model with Statistical Models," Coastal Engineering 58 (2011): 409–16, `https://perma.cc/89DJ-ZENZ`

В этой статье проводится сравнение двух совершенно разных моделей описания системы — статистической и физической. Авторы работы приходят к выводу, что в задачах прогнозирования выбор модели определяется временной шкалой анализируемых данных. Хотя статья посвящена преимущественно прогнозированию, рассматриваемые в ней вопросы имеют самое непосредственное отношение к моделированию, основанному на тех же принципах, что и прогнозирование.

Wolfgang Hardle, Joel Horowitz, and Jens-Peter Kreiss, "Bootstrap Methods for Time Series," International Statistical Review / Revue Internationale de Statistique 71, no. 2 (2003): 435–59, `https://perma.cc/6CQA-EG2E`

Критический взгляд на недостатки статистических методов моделирования данных с временными зависимостями от 2005 года. В характерной для научных сотрудников манере авторы объясняют, почему бутстрап-методы анализа демонстрируют низкую эффективность в процессе анализа временных рядов по сравнению с данными других типов. В статье также рассматриваются наиболее перспективные (на момент ее написания) методы анализа временных данных. Технологии моделирования не сильно изменились с тех пор, и статья остается актуальной и сегодня.

ГЛАВА 5

Хранение временны*х* рядов

Очень часто ценность данных временно*г*о ряда возрастает с увеличением срока их давности — данные, получаемые в режиме реального времени, оказываются менее полезными, чем собранные много лет назад. Чтобы иметь возможность обрабатывать временны*е* ряды в будущем, нужно побеспокоиться об их хранении.

Правильное хранилище данных характеризуется высокой надежностью и простотой доступа и не требует значительных инвестиций в вычислительное оборудование. В этой главе мы обсудим характеристики наборов данных, на которые следует обращать наиболее пристальное внимание при передаче временны*х* рядов на хранение. Мы также обсудим преимущества реляционных и нереляционных баз данных и некоторых других файловых форматов.

Разработка универсального решения, отвечающего за хранение временны*х* рядов, является сложной задачей, поскольку они бывают самых разных типов, каждый из которых имеет свой формат хранения, чтения/записи и обработки данных. Заметьте, что одни данные сохраняются и обрабатываются с заданной периодичностью, тогда как другие востребованы в течение предельно короткого периода времени, после чего их можно безбоязненно удалить.

Перечислим наиболее распространенные сценарии использования хранилищ временны*х* рядов с разными форматами запроса, чтения и записи данных.

1. Временной ряд используется для хранения данных о производительности промышленного оборудования. Сведения о производительности необходимо хранить годами, но чем старше данные, тем меньше деталей они должны включать. Следовательно, вам нужен формат хранения, в котором данные автоматически урезаются и сокращаются по мере устаревания информации.

2. Имея доступ к удаленному хранилищу временны*х* рядов с открытым исходным кодом, вам удобнее работать с локальной копией данных в силу существующих ограничений на доступ к сетевому окружению. В удаленном репозитории временны*е* ряды хранятся в виде отдельных файлов, доступных для загрузки с веб-сервера, но для большего удобства их лучше свести в единую базу данных. В конечной базе данных информация должна представляться в неизменном виде и храниться неограниченный период времени, так как она представляет резервную копию удаленного репозитория.

3. Вы создали собственный временной ряд данных, собранных из многих источников с разными временны*ы*ми масштабами, подвергая их предварительной обработке и форматированию. Сбор и обработка заняли много времени и сил. Вы хотели бы сохранить данные в их окончательном формате, чтобы избежать повторной предварительной обработки. Наряду с этим не стоит пренебрегать сохранением копии необработанных данных на случай возникновения необходимости в их анализе в исходном виде. Вы ожидаете частого пересмотра обработанных и необработанных данных вследствие разработки новых алгоритмов машинного обучения, усовершенствования старых и создания новых моделей, а также пополнения наборов новыми свежими данными. Вы не планируете сокращать или очищать данные в хранилище, даже если это требуется в конечных задачах их анализа.

Как видите, рассмотренные выше сценарии сильно различаются требованиями к системе.

Масштабирование и производительность

В первом сценарии требуется решение, включающее сценарии автоматического удаления старых данных. Здесь не стоит беспокоиться о масштабировании системы, как в случае больших данных, поскольку в хранилище помещаются только небольшие наборы данных. Наряду с этим в остальных сценариях ожидается, что хранилище будет заполняться исключительно большими наборами данных: неизменных (сценарий 2) или регулярно пополняемых (сценарий 3).

Случайный и последовательный доступ к данным

Во втором сценарии предполагается одинаковая частота обращения ко всем значениям набора, так как данные всего временно*г*о ряда имеют одинаковый "возраст" хранения и представляют одну и ту же величину. Напротив, в остальных сценариях (особенно в первом, учитывая предыдущее замечание) ожидается, что последние данные будут востребованы чаще более ранних данных.

Сценарии автоматизации

Первый сценарий просто-таки напрашивается на использование средств автоматизации, тогда как во втором сценарии от автоматизации можно полностью отказаться (данные остаются неизменными). Третий сценарий предполагает наименьшую автоматизацию, но наибольший уровень выборки и обработки всех, а не только последних, данных. В первом сценарии нужно использовать решение для хранения данных, поддерживающее работу со сценариями автоматизации и хранимыми процедурами, тогда как в третьем сценарии оно должно обеспечивать свободную перенастройку инструментов обработки данных.

Мы рассмотрели всего три ограниченных сценария, но даже они дают представление о многих аспектах разработки универсальных решений для хранения данных временны*х* рядов.

В реальных задачах хранения данных вы сможете адаптировать имеющиеся готовые решения и не заниматься поиском инструментов, удовлетворяющих требованиям всех возможных сценариев. Как бы там ни было, вам придется выбирать из небольшого числа форматов, которые можно представить следующим списком возможностей.

- Реляционные базы данных (SQL).
- Нереляционные базы данных (NoSQL).
- Файлы обычных форматов.

В этой главе мы обсудим все три возможности, а также рассмотрим преимущества и недостатки каждой из них. Конечно, формат, в котором будут храниться данные, в наибольшей степени зависит от избранного сценария хранения временны*х* рядов, и в этой главе вы узнаете об основных аспектах, на которые нужно обращать внимание при его выборе.

Сначала мы обсудим требования к хранилищам данных, на которые нужно обращать самое пристальное внимание, а затем перейдем к сравнению реляционных и нереляционных баз данных и рассмотрению наиболее популярных решений по хранению временны*х* рядов. В конце главы мы займемся настройкой политик, отвечающих за умышленное устаревание и удаление более старых данных временны*х* рядов.

Определение требований

Определяясь с технологиями хранения временны*х* рядов, постарайтесь ответить на несколько вопросов.

- *Как много данных временного ряда подлежит хранению? Насколько быстро увеличивается их объем?* Решение для хранения нужно выбирать, исходя из скорости увеличения объема хранимых данных. Администраторы баз данных, переходящие к работе с временны*ми* рядами, содержащими данные о финансовых транзакциях, искренне удивляются тому, насколько сильно они увеличиваются в размере со временем.
- *Исследованию подлежит непрерывный поток восполняемых данных (например, сетевой трафик) или дискретный набор значений (например, почасовые данные о воздушных перевозках в дни государственных праздников США за последние 10 лет)?* Если временны*е* ряды собираются из непрерывного потока, то анализу, скорее всего, будут подлежать только последние данные. С

другой стороны, если анализируемые данные представлены конечными временны*ы*ми рядами, то более отдаленные во времени значения могут представлять даже больший интерес, чем последние. В последнем варианте наиболее предпочтительным будет случайный порядок доступа к данным.

- *Данные собираются через регулярные или нерегулярные временные интервалы?* При равномерном сборе данных можно заранее определить общее количество временны*х* точек и частоту выборки. Если же данные собираются через нерегулярные временны*е* интервалы, то будьте готовы к непредсказуемому порядку выборки данных, который характеризуется периодами простоя и сбора разной длительности.
- *Сбор данных осуществляется в течение бесконечного периода времени или у исследовательского проекта имеется строго оговоренная дата окончания?* В случае существования конечной даты сбора данных всегда можно рассчитать общий объем хранимых данных. Как это ни странно, в этом сценарии самым трудным оказывается прекратить сбор данных в строго оговоренную дату. Нередки ситуации, когда временны*е* ряды определенного типа продолжают собираться даже после завершения всех необходимых исследований!
- *Каково назначение временных рядов? Нужно ли визуализировать их в реальном времени? Будут ли данные подвергаться предварительной обработке нейронной сетью в несколько тысяч проходов? Доступны ли сегментированные данные широкому кругу мобильных пользователей?* Тип доступа к данным (последовательный или свободный), их формат и время ожидания в полной мере определяются избранным сценарием хранения.
- *Каким образом отбраковываются данные и понижается частота их выборки? Как предотвращается их бесконечный рост? Каким должен быть жизненный цикл отдельной точки данных во временном ряду?* Сохранить данные всех событий за весь бесконечный период времени просто невозможно. Именно поэтому лучше заранее выработать принципы автоматического удаления больше ненужных данных, чем заниматься их очисткой вручную. Чем эффективнее будет ваше решение, тем проще вам будет определиться с форматом хранения данных. Подробнее об этом рассказано в следующем разделе.

Ответив на такие вопросы, вы сможете определиться со многими важными аспектами хранения — данные будут записываться в необработанном или в обработанном виде, располагаться в памяти в соответствии со временем сбора или с некой другой характеристикой и нужно ли их представлять в специальном формате, упрощающем чтение и запись. Сценарии хранения могут изменяться в процессе принятия решений, поэтому не ленитесь их пересматривать для каждого нового набора данных.

Сегментированными называются данные, являющиеся частью общей структуры данных, которая разделяется на мелкие, более удобные в управлении части, зачастую хранящиеся на разных серверах.

Оперативные и хранимые данные

Выбирая оптимальный способ хранения, не забывайте о жизненном цикле данных. Чем реалистичнее вы подойдете к оценке фактического сценария, тем меньший объем данных вам придется хранить и тем быстрее определитесь с оптимальной системой хранения, поскольку вряд ли сможете превысить зарезервированный объем в предельно короткие сроки. Очень часто исследователи, боясь потерять важные данные, записывают временны*е* ряды повторно, что совершенно неоправданно: иметь большой объем неразрешимых данных — не намного лучше, чем заниматься сбором данных в течение неприлично большого временно*го* периода.

Краткосрочные временны*е* ряды, например с данными о производительности системы, предназначенные только для ознакомления, обычно не подлежат хранению, по крайней мере в течение долгого периода времени. Такой подход в наибольшей степени проявляется при использовании данных, управляемых событиями, когда интерес представляют не отдельные события, а общие статистические показатели процесса.

Представьте, что с помощью специального серверного программного обеспечения собираются и сохраняются сведения о количестве времени, затрачиваемого каждым мобильным устройством для полной загрузки веб-страницы. В качестве результата вы получите неравномерные временны*е* ряды, выглядящие примерно так, как показано в табл. 5.1.

Таблица 5.1. Временны*е* ряды, собираемые веб-сервером

Временна*я* метка	Время загрузки страницы, с
5 апреля, 2018 22:22:24	23
5 апреля, 2018 22:22:28	15
5 апреля, 2018 22:22:41	14
5 апреля, 2018 22:23:02	11

Сами по себе временны*е* метки событий загрузки страницы не представляют ценности для анализа сразу по многим причинам. Поэтому подвергать анализу лучше всего средние значения величин (например, среднее время загрузки страницы за минуту). Но результаты статистических исследований усредненных показателей будут актуальны только непродолжительный период времени. Предположим, что требуется изучить данные, собранные сервером в течение ночи, чтобы

убедиться в его высокой производительности. Упростите исходный временной ряд до единственной точки данных, в которой будет указываться усредненное за все время наблюдений (12 часов) значение показателя. Полученный результат несет больше полезной информации, чем исходно собранные данные (табл. 5.2).

Таблица 5.2. Упрощенные точки данных для периодов наблюдений

Временной период	Наиболее популярное время доступа	Количество загрузок	Среднее время загрузки, с	Максимальное время загрузки, с
5 апреля 2018 года 20:00–8:00	23:00	3470	21	45

Как видите, в таком сценарии можно отказаться от долговременного хранения данных всех регистрируемых событий. Вместо этого используйте для хранения такое решение, которое обеспечивает их сохранность в течение небольшого периода времени — до представления в окончательном, наиболее информативном формате. Тем самым вы избежите неоправданно быстрого роста объема данных, помещаемых на хранение. Вместо 3470 отдельных событий, которые по отдельности не представляют особого интереса, долговременному хранению будут подвергаться компактные данные с заведомо важной информацией. Старайтесь всегда, когда только возможно, сокращать объем хранящихся данных за счет усреднения и исключения из временного ряда дублирующих значений.

Далее мы рассмотрим несколько возможностей по сокращению данных без потери важной информации.

Медленно изменяющиеся переменные

При хранении переменных состояния рассмотрите возможность записи только тех точек данных, в которых они претерпевают изменения. Например, если вы снимаете показания термометра через каждые пять минут, то кривая изменения температуры будет выглядеть как пошаговая функция, особенно если вы округляете измеренные значения до ближайшего градуса. В таком случае можно отказаться от хранения повторяющихся значений, сэкономив место для более значимых данных.

Зашумленные, высокочастотные данные

Если данные зашумлены, то нет никаких оснований дорожить каждой отдельно рассматриваемой точкой данных. Рассмотрите возможность частичного объединения точек данных перед записью в хранилище — высокий уровень шума делает любое отдельное наблюдение малоинформативным. Конечно, все зависит от рассматриваемой задачи, и вам нужно убедиться в том, что при необходимости исходный шум, присутствующий в данных, все еще можно оценить.

Устаревшие данные

Чем старше данные, тем меньше вероятность того, что они будут привлекаться к анализу временны*х* рядов, разве что в самом общем виде. Всякий раз, собираясь сохранять новый набор временны*х* рядов, попытайтесь понять, в какой момент времени они станут неактуальными.

- Существует ли у данных естественный срок годности?
- Если нет, то по возможности ознакомьтесь с результатами исследований аналитического отдела и посмотрите, насколько давние данные принимаются в них в расчет? Например, узнайте, когда в последний раз сценарий запрашивал самые старые значения временны*х* рядов из репозитория GitHub.

Автоматизировав операцию удаления данных без нарушения процедуры анализа временны*х* рядов, вы уменьшите сложность их хранения и дальнейшей обработки, нивелировав проблему масштабируемости или медленно обрабатываемых запросов к непомерно раздутым наборам данных.

Правовые аспекты

Если вы работаете в крупной организации, которая занимается обработкой больших массивов данных, предоставляемых сторонним организациям (например, регулирующим органам), то при выборе средств хранения обязательно учитывайте юридические аспекты решаемой задачи. Постарайтесь оценить, как часто будут обрабатываться запросы на получение данных, какой объем данных передается при их выполнении и регулирует ли нормативная документация формат хранения и передачи данных.

Изучение правовых аспектов при выборе средств хранения данных может показаться излишней мерой предосторожности, но последние исследования, проводимые специалистами по обработке данных в Европе и США, свидетельствуют о том, что наибольший интерес у аналитиков вызывают наборы данных, доступные для использования в машинном обучении (подпадающие под действие норм общего регламента по защите данных Европейского союза).

До настоящего момента мы рассматривали только общие требования к решениям, предназначенным для хранения временны*х* рядов. Вы также познакомились с особенностями получения и анализа временны*х* рядов как основного фактора определения формата хранения данных. Теперь можно переходить к рассмотрению двух наиболее распространенных вариантов хранения временны*х* рядов: в базах данных и в файлах.

Хранение временны*х* рядов в базах данных

База данных является интуитивно понятным и знакомым инструментом хранения данных практически для любого специалиста по анализу и обработке информации. При этом базы данных прекрасно подходят для хранения не только реляционных данных, но и временны*х* рядов. В наибольшей степени они подходят для решения задач, в которых к системам хранения данных выдвигаются следующие, давно ставшие классическими, требования.

- Масштабирование данных на нескольких серверах.
- Выполнение операций чтения/записи с низкой задержкой.
- Поддержка функции вычисления часто определяемых статистических показателей (например, среднего значения в запросах группировки, где операция группировки может применяться в том числе к временны*м* величинам).
- Снабжение инструментами сопровождения и устранения неполадок, которые можно использовать для настройки производительности системы и анализа узких мест.

Из всех возможных вариантов хранения временны*х* рядов (`https://perma.cc/K994-RXE9`) существуют веские причины отказаться от использования отдельных файлов в пользу баз данных, особенно при работе с новыми наборами данных. В базах данных, особенно в нереляционных, данные находятся в гибком состоянии. Наличие строгой структурной организации является еще одним преимуществом баз данных: можно существенно ускорить запуск проектов с данными временны*х* рядов по сравнению с решениями, основанными на отдельных файлах. Но даже если вы предпочитаете хранить данные временны*х* рядов в файлах (в отличие от большинства исследователей), база данных поможет вам определиться со структурой файлов, количество которых неизбежно увеличивается по мере усложнения проекта и пополнения временны*х* рядов новыми данными.

В оставшейся части этого раздела мы рассмотрим преимущества реляционных баз данных перед NoSQL-решениями, а затем обсудим самые популярные в настоящее время типы баз данных, предназначенные для хранения временны*х* рядов.

Хорошая новость заключается в стремительном развитии графовых СУБД (`https://perma.cc/RQ79-5AX7`), в настоящее время являющихся наиболее перспективной технологией управления базами данных, и есть все основания полагать, что в ближайшем будущем появится несколько достойных внимания решений.

Реляционные и нереляционные базы данных

Технологии SQL и NoSQL привлекают не менее пристальное внимание, чем базы данных временны*х* рядов. Многие опытные администраторы баз данных

настаивают на том, что SQL — единственно верная технология управления базами данных и что не существует данных, которые нельзя описать правильно подобранным набором реляционных таблиц. Тем не менее при попытках масштабирования SQL-решений при хранении больших временны*х* рядов часто наблюдается падение производительности, что подталкивает к переходу на нереляционные технологии управления базами данных, особенно при поиске открытого решения, обеспечивающего свободное масштабирование данных при сборе временны*х* рядов для бесконечного временно*г*о горизонта.

При работе с временны*м*и рядами могут применяться как реляционные, так и нереляционные технологии управления базами данных, но оба случая сопряжены с целым рядом трудностей при управлении базами данных временны*х* рядов. Для правильного их применения нужно понять, чем временны*е* ряды отличаются от других типов данных, для управления которыми изначально разрабатывались СУБД указанных типов.

Данные, с которыми работают СУБД

Для лучшего понимания принципов несоответствия реляционных баз данных потребностям в хранении временны*х* рядов достаточно изучить историю разработки решений, основанных на технологии SQL. Исходно SQL-решения предназначались для обработки транзакционных данных — достаточных для полного описания дискретного события. В транзакции обрабатываются данные атрибутов, которые отражены во многих первичных ключах, таких как назначение, имя клиента, дата исполнения и стоимость транзакции. Обратите внимание, что время также может выступать первичным ключом, указывая на одно из многих, но не привилегированное направление измерения информации.

Существуют две важные особенности транзакционных данных, которые отличают их от данных временны*х* рядов.

- Точки данных часто обновляются.
- Доступ к данным осуществляется случайным образом, так как порядок их обработки никак не регламентируется.

Характеристики временны*х* рядов

Если данные временно*г*о ряда описывают историю некого процесса, то транзакция указывает только конечное состояние системы. Следовательно, данные временны*х* рядов обычно не требуют обновления, что делает операции записи данных со случайным доступом крайне маловероятными.

Таким образом, требования к производительности инструментов, которые были ключевыми на протяжении многих десятилетий разработки СУБД, оказываются несущественными при работе с временны*м*и рядами. Фактически, выдвигая требования к системам хранения временны*х* рядов, мы преследуем несколько

иные цели, поскольку управление такими хранилищами построено на принципах, отличных от принятых в реляционных базах данных. Для лучшего понимания обозначим ключевые требования к средствам хранения временны*х* рядов.

- Операции записи преобладают над операциями чтения.
- Данные записываются, считываются и обновляются не случайным образом, а во временно*м* порядке.
- Одновременное считывание данных выполняется намного чаще, чем в случае проведения транзакций.
- Кроме времени, допускается использовать (если они заданы) всего несколько первичных ключей.
- Массовое удаление данных выполняется чаще, чем удаление отдельных точек данных.

Большинство из этих функций поддерживается NoSQL-решениями — многие нереляционные базы данных общего назначения предлагают многое из того, что требуется реализовать в хранилище временны*х* рядов — в первую очередь, из того, что касается превалирования операций записи над операциями чтения. Концептуально технологии NoSQL хорошо согласуются с требованиями, которые выдвигаются к средствам хранения временны*х* рядов, изначально отражая многие специальные возможности, например заполнения полей данных не для всех точек данных. Гибкость структуры NoSQL оказывается естественной для временны*х* рядов. По правде говоря, стремительно растущий интерес к нереляционным технологиям в большой степени вызван потребностью в системах управления и хранения временны*х* рядов.

По этой же причине готовые NoSQL-решения показывают лучшие результаты в операциях записи, чем реляционные СУБД. На рис. 5.1 приведены графики сравнения производительностей операций добавления (записи) новых точек временны*х* рядов в стандартные реляционную базу данных и базу данных NoSQL.

Трудности выбора

Может показаться, что я пытаюсь подтолкнуть вас к варианту использования нереляционных решений, но у реляционных баз данных тоже есть важные преимущества. Изучая имеющиеся данные, помните о главном аспекте, который нужно учитывать как в реляционных базах данных и NoSQL-решениях, так и при хранении временны*х* рядов в старом добром текстовом файле: одновременно запрашиваемые данные должны храниться одном и том же месте. Это самый важный фактор, независимо от предпочитаемого решения.

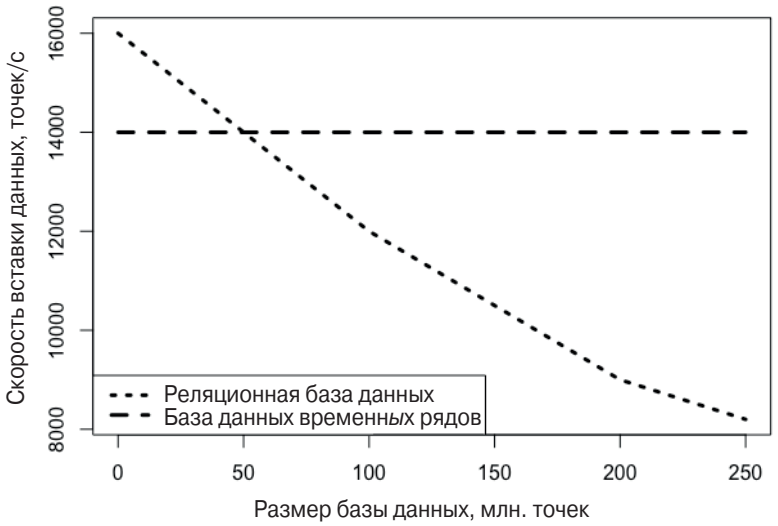


Рис. 5.1. *Главная особенность базы данных временны**х** рядов состоит в неизменности скорости добавления новых точек данных при увеличении ее общего размера. Это важнейшее преимущество перед традиционными реляционными СУБД*

На многих тематических конференциях и в блогах NoSQL-решения описываются как хранилища данных, в полной мере отвечающие требованиям высокой скорости записи, низкой частоты обновления и последовательной организации данных. Наряду с этим реляционные базы данных также относятся к оправданным и часто востребованным решениям для хранения временны*х* рядов. В частности, внеся незначительные архитектурные изменения в традиционные базы данных и внутреннюю структуру памяти, можно преодолеть большинство недостатков с сохранением большинства преимуществ реляционных СУБД (`https://perma.cc/3ZUQ-B5WC`). Например, такое простое и, казалось бы, очевидное решение, как структурирование внутренней памяти реляционных таблиц, обеспечивающее учет времени, позволяет разительно увеличить общую производительность системы.

В конечном счете преимущества и недостатки как реляционных, так и нереляционных баз данных сильно зависят от их реализации и не настолько концептуальны и важны, как предполагают технологии, лежащие в их основе. Допустим, что выбор конкретной реализации одной из технологий определяют исследуемые данные. Рассматривая атрибуты временны*х* рядов и возможные сценарии реализации, помните о следующих важных ограничениях.

*Преимущества хранения временны**х** рядов в реляционных базах данных*

- При хранении в реляционной базе данных временной ряд можно легко связать с соответствующими перекрестными данными этой же базы данных.

- Иерархические данные временны*х* рядов естественным образом встраиваются в реляционные таблицы. Образованная структура позволяет предельно точно группировать связанные временны*е* ряды и четко обозначать иерархические отношения, в то время как в NoSQL-решении эта задача представляется намного сложнее и требует более системного подхода в выполнении.
- При создании временны*х* рядов на основе транзакционных данных, хранящихся в реляционной базе данных, самым оправданным решением будет сохранять временны*е* ряды в этой же базе данных, упрощая их дальнейшую проверку, наполнение перекрестными связями и т.д.

*Преимущества хранения временны**х** рядов в нереляционных базах данных*

- Высокая скорость записи.
- Прекрасно подходят для разработки функциональных и высокопроизводительных решений в системах, в которых мало что известно о будущих данных.
- Начинающим специалистам по анализу данных нереляционные базы данных кажутся более производительным, готовым к использованию решением, не способным к образованию малопригодной производственной схемы.

Популярные базы данных временны*х* рядов и файловые решения

Настало время обсудить наиболее популярные решения для хранения временны*х* рядов. Вам предстоит узнать о возможностях, которые предоставляются в дополнение к традиционным базам данных. Обратите внимание, что обсуждаемые далее решения относятся к нерегулярно задействуемым технологиям. Те из них, которые пользуются популярностью в этом году, могут оказаться маловостребованными в следующем году. Таким образом, дальнейшие рассуждения следует воспринимать не как строгие технологические рекомендации, а скорее как набор советов, помогающих ориентироваться в текущем состоянии рынка программных решений, предназначенных для хранения данных.

Базы данных для хранения временны*х* рядов и инструменты мониторинга

Сначала давайте познакомимся со специальными инструментами хранения и мониторинга временны*х* рядов. В частности, рассмотрим базу временны*х* рядов InfluxDB и специальный инструмент, отвечающий за мониторинг производительности системы, который также может использоваться как решение для хранения временны*х* рядов, — Prometheus. Преимущества каждого инструмента в полной мере отражают их назначение и способ использования.

InfluxDB. Согласно описанию проекта на GitHub база данных InfluxDB исходно ориентирована на хранение временны*х* рядов.

> InfluxDB — это база данных временны*х* рядов с открытым исходным кодом. Ее лучше всего использовать для сбора статистических показателей, регистрации событий и результатов аналитических исследований.

В InfluxDB данные представлены временны*ми* рядами. Точка данных в InfluxDB состоит из следующих компонентов.

- Временн*ая* метка.
- Подпись, описывающая измеряемую величину.
- Одно или несколько полей "ключ/значение" (например, `temperature=25.3`).
- Пары "ключ/значение", содержащие теги метаданных.

Как учитывающая время база данных InfluxDB автоматически расставляет временны*е* метки для всех точек данных, лишенных их. Кроме того, в InfluxDB используются SQL-подобные запросы, например такие.

```
SELECT * FROM access_counts WHERE значение> 10000
```

База данных InfluxDB характеризуется следующими преимуществами.

- Поддержка инструментов хранения данных, которые позволяют легко автоматизировать операции назначения и удаления устаревших данных.
- Высокая скорость получения и высокая степень сжатия данных.
- Возможность добавления меток к временны*м* рядам, которые соответствуют определенному критерию, для быстрой индексации.
- Полноправное подключение к TICK (`https://www.influxdata.com/time-series-platform/`) — платформе для сбора, хранения, мониторинга и отображения временны*х* рядов.

Существует много других баз данных, ориентированных на хранение временны*х* рядов. На данный момент InfluxDB является наиболее популярной из них, поэтому с ней вы, скорее всего, и столкнетесь в первую очередь. Ее функции стандартны для специализированных баз своего типа и отражают наиболее востребованные возможности по хранению временны*х* рядов.

Как база данных InfluxDB основана на *технологии "проталкивания"* (push-технология) — в базу данных заносятся данные, предложенные ей. Такой подход заметно отличается от представленного далее в описании базы данных Prometheus.

С учетом рассмотренных спецификаций InfluxDB является полнофункциональным решением, основанным на общепринятых принципах. Это означает, что при ее использовании вы получаете в свое распоряжение как стандартные SQL-инструменты, так и специализированные функции, отвечающие за сбор

и контроль размера временны*х* рядов. Кроме того, InfluxDB включает команды быстрой записи данных, часто востребованные при сборе данных временны*х* рядов непосредственно в процессе их создания.

Prometheus. Программное обеспечение Prometheus (`https://github.com/prometheus/prometheus`) описывается как "система мониторинга и база данных временны*х* рядов", работающая по протоколу HTTP. В описании недвусмысленно указывается его основное функциональное назначение: сначала — мониторинг и только потом — хранение данных. Главная особенность Prometheus заключается в том, что она основана *на pull-технологии*, — параметры извлечения данных временно*г*о ряда, а также сериализации легко поддаются проверке и настройке.

Система Prometheus — это ценный инструмент на случай непредвиденных ситуаций благодаря атомарной и самодостаточной структуре. Однако поддержка технологии pull не предполагает обновление данных с абсолютной точностью и своевременностью. А так как Prometheus, в первую очередь, предназначена для оперативного мониторинга производительности, она точно не подойдет для систем, в которых первостепенную важность имеет 100-процентная точность данных.

Запросы в Prometheus пишутся на языке функциональных выражений PromQL.

```
access_counts> 10000
```

Кроме того, система Prometheus снабжена программным интерфейсом PromQL, включающим команды выполнения многих распространенных задач обработки временны*х* рядов, в том числе такие сложные, как прогнозирование (`predict_linear()`) и вычисление скорости роста временно*г*о ряда за единицу времени (`rate()`). А еще он обеспечивает агрегацию показателей по периодам времени. В Prometheus акцент сделан на мониторинге и анализе временны*х* рядов, а не на обслуживании базы данных, поэтому по сравнению с InfluxDB в нем намного меньше команд автоматической обработки данных.

Систему Prometheus можно смело использовать для хранения временны*х* рядов, особенно в потоковых приложениях и в случаях, когда превыше всего ценится доступность данных. Она не самая простая для обучения благодаря использованию специального языка написания сценариев, а также необычной архитектуре и не характерному для базы данных программному интерфейсу. Тем не менее она широко используется и нравится многим разработчикам.

Нереляционные базы данных

В то время как базы данных, специально предназначенные для временны*х* рядов, имеют много преимуществ, для их хранения можно попробовать использовать нереляционные базы данных общего назначения. Структура таких баз данных основана на документах, а не таблицах, а сами они обычно включают много явных функций обработки временны*х* рядов данных.

Несмотря на это гибкость состояния нереляционных баз данных прекрасно сочетается с обработкой временны*х* рядов, особенно в новых проектах, в которых скорость сбора данных и количество входных каналов могут изменяться в течение всего срока хранения набора данных. Например, временной ряд может изначально включать всего один канал данных, но постепенно расширяться за счет включения большего количества типов данных, снабженных временно*й* разметкой. В дальнейшем, ввиду бесполезности, некоторые из входных каналов всегда можно отключить.

Сохранение подобных данных в реляционной таблице представляет собой трудно решаемую задачу сразу по нескольким причинам и приводит к включению большого количества значений `NaN`, указывающих на недоступность данных. В нереляционной базе данных, чтобы избежать загромождения хранилища данных множеством значений `NaN`, достаточно отключить соответствующие каналы.

В качестве примера популярной и производительной нереляционной базы данных можно привести MongoDB. Превыше всего она ценится именно как база данных временны*х* рядов, а также за активную поддержку IoT-технологий на архитектурном и программном уровнях. Она включает функции агрегации высокого уровня, которые обеспечивают агрегацию информации по времени и группировку данных по временны*м* признакам, а также предлагает множество инструментов автоматического разделения данных по принятым в человеческом мире временны*м* отрезкам — дням недели или месяца.

- `$dayOfWeek`
- `$dayOfMonth`
- `$hour`

Кроме того, разработчики системы MongoDB приложили значительные усилия для документирования ее функций обработки временны*х* рядов.[1] Полный набор документации и акцент на временны*х* рядах однозначно указывают на то, что пользователям стоит ожидать появления более удобных функций управления временны*ми* зависимостями.

Однако наиболее важной из всех функциональных особенностей MongoDB является гибкость изменения схемы со временем. Она позволяет избежать многих неприятностей при работе с быстро эволюционирующими наборами временны*х* рядов, характеризующихся нерегулярным способом сбора данных.

[1] См., например, публикации “Schema Design for Time Series Data in MongoDB” (`https://perma.cc/E8XL-R3SY`), “Time Series Data and MongoDB: Part 1 — An Introduction” (`https://perma.cc/A2D5-HDFB`) и “Time Series Data and MongoDB: Part 2 — Schema Design Best Practices” (`https://perma.cc/2LU7-YDHX`). Противоположная точка зрения изложена в статье “How to store time-series data in MongoDB, and why that is a bad idea” (`https://perma.cc/T4KM-Z2B4`).

Например, проследим за тем, как могут изменяться способы сбора временны*х* рядов при запуске одного из возможных стартапов в области здравоохранения.

1. Стартап начинается с выпуска приложения для измерения артериального давления и обеспечивает сбор всего двух показателей: систолического и диастолического артериального давлений, которые рекомендуется измерять пользователям несколько раз в день.
2. Однако пользователи желают получить советы по здоровому образу жизни и готовы предоставить вам больше данных. Вы получаете все необходимые сведения о пользователях от дат их рождения до ежемесячных показателей веса, количества ежечасно пройденных шагов и расходованных калорий. Очевидно, что эти разные типы данных собираются с совершенно разными скоростями (частотами).
3. Вскоре вы понимаете, что некоторые из типов данных не представляют для приложения никакой ценности, и поэтому прекращаете их сбор.
4. Но пользователи не просматривают выводимые им сообщения, в том числе о ненадобности отдельных показателей, и вы вынуждены продолжать сбор в том числе ненужных данных…
5. И затем правительство одного из крупнейших рынков вашей целевой аудитории принимает закон о защите персональных данных о состоянии здоровья пользователей, и вам необходимо либо удалить все данные, либо зашифровать их — в базу данных нужно добавить еще одно, зашифрованное, поле.
6. Изменения продолжаются до бесконечности.

Если в вашем проекте важна гибкость в запросах и структурной организации данных, подобная описанной выше, то можете смело обращаться к услугам нереляционной базы данных, обеспечивающей разумный баланс между общей гибкостью и временно*й* точностью системы.

Еще одно преимущество нереляционной базы данных общего назначения, в отличие от базы данных, ориентированной на хранение временны*х* рядов, заключается в возможности быстрой интеграции в систему данных, не относящихся к временны*м* рядам, для образования перекрестных связей между наборами данных. Иногда реляционная база данных общего назначения — это просто удачное сочетание высокопроизводительных решений и SQL-подобных инструментов, полученных без понимания того, как правильно оптимизировать реляционную схему под задачи управления временны*ми* рядами.

В этом разделе мы рассмотрели решения, основанные на нереляционных СУБД[2] и некоторые другие широко используемые технологии. Несмотря на то что доминирующие на сегодняшний момент технологии будут непрерывно совершенствоваться, общие принципы их построения, особенности использования и конкурентные преимущества останутся неизменными. В качестве краткого итога перечислим достоинства упомянутых ранее технологий баз данных.

- Высокая скорость чтения или записи (либо и то, и другое).
- Гибкая структура данных.
- Технологии передачи данных push и pull.
- Автоматизированные инструменты сбора данных.

Базы данных, ориентированные на хранение временны*х* рядов, предлагают наиболее высокую степень автоматизации операций по обработке временны*х* рядов, но при этом характеризуются меньшей гибкостью и предоставляют меньше возможностей по интегрированию перекрестных данных, чем нереляционные базы данных общего назначения.

По большому счету базы данных характеризуются б*о*льшей гибкостью, чем хранилища из плоских файлов, но это означает, что базы данных менее оптимизированы и производительны, чем файловые решения, о которых мы поговорим далее.

Файловые решения

В общем понимании база данных — это программное обеспечение, в котором удачно объединяются сценарии автоматизации и хранилище данных. А по своей сути это обычный файл, обернутый в специальное программное обеспечение, которое делает его максимально безопасным и простым в использовании.

Иногда имеет смысл отказаться от внешней оболочки и взять ответственность за хранение данных на себя. Хотя это не очень распространено в бизнес-приложениях, так часто поступают в научных программах и некоторых типах промышленного программного обеспечения (например, в приложениях высокочастотной биржевой торговли), в которых производительность имеет первостепенное значение. В таких ситуациях разработка гораздо более сложного конвейера данных, обеспечивающего распределение памяти для хранения данных, открытие, чтение и закрытие файлов, их защита и так далее выполняются специалистами по анализу данных вручную. Разумеется, это намного более сложная задача, чем написание запросов к базе данных.

[2] Рассмотрение принципов оптимизации реляционных баз данных для обработки временны*х* рядов выходит за рамки материала этой книги ввиду высокой их сложности. Этому вопросу посвящены публикации, упомянутые в конце главы.

Решение на основе плоских файлов будет неплохим вариантом при выполнении любого из следующих требований.

- Формат данных выбран правильно, позволяя собирать всю необходимую информацию за достаточно долгий период времени.
- Скорость обработки данных зависит от скорости их ввода-вывода, поэтому нет смысла тратить время и усилия на ее повышение.
- Свободный доступ к данным не требуется; достаточно считывать данные последовательно.

Далее мы кратко остановимся на некоторых популярных решениях, основанных на плоских файлах. Не забывайте, что вы всегда можете создать собственный формат хранения файлов (`https://perma.cc/DU98-FWW9`), хотя это довольно сложно. Так поступать стоит только в случае владения языком программирования, отличающимся высокой производительностью, например C++ или Java.

Если ваша система достаточно зрелая и производительная для работы с плоской файловой системой, то такая реализация будет обладать рядом неоспоримых преимуществ, даже несмотря на высокую трудоемкость процесса переноса данных из базы данных в отдельные файлы.

- Формат плоского файла не зависит от системы. Для обмена данными просто сделайте файлы общедоступными. При этом не нужно просить коллегу открывать удаленный доступ к базе данных или создавать зеркальную базу данных.
- Временн*ые* затраты на ввод (вывод) данных в обычном файле несравнимо меньшие, чем при работе с базой данных, поскольку определяются простой операцией чтения файла (записи в файл) без необходимости поиска и извлечения из базы данных.
- Порядок, в котором должны считываться данные, указывается в самом формате файла, что менее характерно для баз данных. Такой подход повышает скорость последовательного считывания данных, что благосклонно сказывается на производительности, например, обучения алгоритмов глубокого обучения.
- Конечные данные занимают в памяти гораздо меньший объем, чем при использовании базы данных, благодаря возможности максимально увеличить степень сжатия. Вы можете предельно точно настроить степень сжатия данных, чтобы добиться оптимального баланса между потребностями в минимизации размера хранилища данных и скорости ввода-вывода данных. Высокая степень сжатия позволяет уменьшить объем данных, но понижает скорость операций ввода-вывода.

Библиотека NumPy

Если данные временны*х* рядов носят исключительно числовой характер, то для их хранения можно смело применять массивы библиотеки `NumPy` языка Python. Такие массивы могут иметь самые разные типы данных, а их относительная производительность освещается в большом количестве публикаций. Самостоятельно оценку производительности операций с различными типами массивов `NumPy` можно выполнить с помощью решения, представленного в репозитории `array_storage_benchmark` (`https://perma.cc/ZBS7-PR56`) на GitHub.

Главный недостаток массива `NumPy` заключается в невозможности хранения в нем сразу нескольких типов данных. Это означает, что вы не можете автоматически сохранять в ней разнородные данные временны*х* рядов и должны придумать способ приведения их к единственному типу данных в необработанном или обработанном виде (это ограничение можно обойти). Еще один важный недостаток массивов `NumPy` обусловливается тем фактом, что в нем не предусмотрена возможность добавления подписей к строкам и столбцам, поэтому, например, нет простого способа временно*й* разметки отдельно каждой строки массива.

Преимущество использования массива `NumPy` состоит в поддержке большого количества форматов сохранения, включая сжатый двоичный формат, который позволяет сэкономить место и обеспечивает быстрый ввод-вывод по сравнению с традиционными базами данных. С точки зрения анализа и хранения данных массив `NumPy` является необычайно производительной структурой данных.

Библиотека Pandas

Для более простой разметки данных или сохранения разнородных данных временны*х* рядов (или и того, и другого) обратитесь к более структурно сложному и гибкому фрейму данных пакета `Pandas`. Он станет незаменимым при работе с временны*ми* рядами, состоящими из большого количества разнородных данных, в том числе включающих счетчики событий (целочисленные значения), характеристики состояния (числа с плавающей точкой) и подписи (текстовые строки или программный код). В таких случаях фреймы данных пакета `Pandas` (“pandas” происходит от словосочетания “panel data” (панельные данные), представляющего привычный формат в большинстве случаев использования) оказываются наиболее естественным решением.

Фреймы данных `Pandas` находят широкое применение в анализе временны*х* рядов — результаты оценки их производительности при хранении данных разных форматов приведены на тематических площадках (`https://perma.cc/BNJ5-EDGM`).

Разнозначные инструменты языка R

Собственный формат хранения объектов языка R представлен классами `.Rds` и `.Rdata`. В них данные представлены в битовой форме, что обусловливает более высокую, чем в текстовом формате, степень сжатия и скорость выполнения операций ввода-вывода. В этом понимании объекты языка R во многом напоминают фреймы данных `Pandas` языка Python. Более того, как в R, так и в Python фреймы данных можно сохранять в независимом от языка битовом формате `feather` (`https://perma.cc/4C3J-TBK8`). В R наибольшую производительность будут показывать, конечно же, собственные двоичные форматы. В табл. 5.3 приводятся сравнительные характеристики разных файловых форматов.

Таблица 5.3. Сравнение размера и производительности файловых форматов

Название	Относительный размер	Относительное время загрузки
`.RDS`	1x	1x
`feather`	2x	1x
`CSV`	3x	20x

Как видите, собственный формат выступает явным победителем как по объему хранящихся данных, так и по скорости операций вводы-вывода. Наряду с этим аутсайдером выступает текстовый формат, отличающийся более низкими, чем в двоичном формате, оценками производительности.

Пакет `Xarray`

При работе с многомерными временны*ы*ми рядами обратите внимание на одно из промышленных решений для хранения данных — в частности, на пакет `Xarray`, выделяющийся рядом полезных возможностей.

- Именованные размерности.
- Векторизованные математические операции, подобные применяемым в `NumPy`.
- Групповые операции, как в `Pandas`
- Функциональные особенности, свойственные базам данных, включая команды индексации по временны*м* диапазонам
- Различные форматы хранения файлов

`Xarray` — это структура данных, которая поддерживает множество функций обработки данных временны*х* рядов, в том числе индексации и повторной временно*й* выборки, интерполяции данных и работы с отдельными компонентами даты и времени. Пакет `Xarray` разрабатывался как высокопроизводительный

научный вычислительный инструмент и остается крайне недооцененным решением для анализа временны*х* рядов.

`Xarray` представляется отдельной структурой данных в языке Python, обеспечивающей несколько вариантов хранения временны*х* рядов. В ней возможны как сериализация, так и представление данных в еще одном двоичном формате — `netCDF` , — универсальном формате обмена научными данными, поддерживаемом многими платформами и языками. Для тех, кто ищет эффективный инструмент для работы с временны*м*и рядами в языке Python, пакет `Xarray` будет очень благоразумным выбором.

Как вы могли заметить, существует большое количество способов хранения данных временны*х* рядов в плоских файлах. Одни из них предполагают использование ассоциированных функций (`Xarray`), а другие — работают только с упрощенными, исключительно числовыми наборами данных (`NumPy`). При переносе информации из конвейера базы данных в конвейер, основанный на использовании отдельных файлов, неизбежно будут возникать определенные трудности, связанные с необходимостью упрощения данных и переписывания сценариев — формализация логики базы данных в явных сценариях ETL (Extract-Transform-Load — извлечение-преобразование-загрузка). В случаях, когда первостепенное значение имеет производительность, перемещение базы данных в отдельные файлы окажется наиболее правильным шагом по оптимизации, позволяющим существенно снизить задержки в обработке данных. Давайте посмотрим, в каких именно случаях такой шаг оказывается наиболее оправданным.

- Чувствительное к временны*м* задержкам прогнозирование, выполняемое для программного обеспечения, ориентированного на пользователя
- Операции ввода-вывода с частым повторным доступом к данным, например в задачах обучения алгоритмов глубокого обучения

С другой стороны, удобство использования, масштабируемость и гибкость базы данных оказываются более важными, чем производительность, факторами во многих приложениях. В результате оптимальным считается решение для хранения данных, при разработке которого учитываются как характер и назначение данных, так и способ их обработки.

Дополнительные источники

- Базы данных временны*х* рядов

  Jason Moiron, "Thoughts on Time-Series Databases," jmoiron blog, June 30, 2015, `https://perma.cc/8GDC-6CTX`

  Популярный пост в блоге 2015 года, дающий представление о раннем периоде разработки баз данных временны*х* рядов и культе сбора

данных “на все случаи жизни”. Содержит подробный обзор решений для хранения временны*х* рядов и типичных сценариев их реализации. Будет полезен для начинающих специалистов по администрированию и разработке баз данных.

Preetam Jinka, “List of Time Series Databases,” Misframe blog, April 9, 2016, `https://perma.cc/9SCQ-9G57`

Длинный, регулярно обновляемый список баз данных временны*х* рядов, предоставляющих сведения о наиболее популярных в настоящий момент времени решениях. В описании каждой базы данных проводится сравнение ее с ближайшими конкурентами и предшественниками.

Peter Zaitsev, “Percona Blog Poll: What Database Engine Are You Using to Store Time Series Data?” Percona blog, February 10, 2017, `https://perma.cc/5PXF-BF7L`

Опрос инженеров баз данных, проведенный в 2017 году, показал, что реляционные базы данных занимают доминирующе положение на рынке, причем 35% опрашиваемых указали, что используют такие базы данных для хранения временны*х* рядов. Лидерами опроса стали такие системы, как ElasticSearch, InfluxDB, MongoDB и Prometheus.

Rachel Stephens, “The State of the Time Series Database Market,” RedMonk, April 3, 2018, `https://perma.cc/WLA7-ABRU`

Недавний отчет, основанный на изучении данных и выполненный специалистом по анализу данных, включает описание наиболее популярных решений для хранения временны*х* рядов методом эмпирического исследования активности сервисов GitHub и Stack Overflow. Отчет указывает на высокий уровень специализации решений для хранения временны*х* рядов, что обусловлено широким выбором сценариев реализации, а также тенденцией к разработке преимущественно узкопрофильных баз данных.

Prometheus.io, “Prometheus Documentation: Comparison to Alternatives,” n.d., `https://perma.cc/M83E-NBHQ`

Чрезвычайно подробный и полный список различий в функциональных особенностях системы Prometheus и других решений для хранения временны*х* рядов. Его можно использовать в качестве краткого справочника по структурам данных и средствам хранения, наиболее часто применяемым в альтернативных решениях. Прекрасный источник сведений о наиболее востребованных технологиях, а также преимуществах и недостатках баз данных, предназначенных для хранения временны*х* рядов.

- Адаптация баз данных общего назначения

  Gregory Trubetskoy, "Storing Time Series in PostreSQL Efficiently," Notes to Self blog, September 23, 2015, `https://perma.cc/QP2D-YBTS`

  > Старое, но до сих пор актуальное сообщение в блоге, в котором описываются общие принципы хранения временны*х* рядов в базе данных Postgres, обеспечивающей высокую скорость обработки данных. В нем Трубецкой указывает на недостатки "наивного" подхода, основанного на размещении значений и временны*х* меток в отдельных столбцах, и дает практические рекомендации по работе с массивами Postgres.

  Josiah Carlson, "Using Redis as a Time Series Database," InfoQ, January 2, 2016, `https://perma.cc/RDZ2-YM22`

  > Статья содержит подробные советы по работе и примеры использования системы Redis в качестве базы данных временны*х* рядов. Несмотря на то что Redis исходно разрабатывалась как решение для хранения временны*х* рядов, работа в ней сопряжена с рядом трудностей и требует использования специальных структур данных, которые можно применять во многих других решениях для хранения временны*х* рядов. Статью можно рассматривать как краткое справочное руководство по использованию баз данных Redis для хранения временны*х* рядов, а также для демонстрации принципов выполнения операций над временны*ми* рядами с помощью общепринятых инструментов.

  Mike Freedman, "Time Series Data: Why (and How) to Use a Relational Database Instead of NoSQL," Timescale blog, April 20, 2017, `https://perma.cc/A6CU-6XTZ`

  > Пост в блоге одного из разработчиков базы данных TimescaleDB, в котором описаны способы построения реляционной базы данных временны*х* рядов, включая изменение схемы распределения памяти под задачи управления временны*ми* данными. Разработчики TimescaleDB утверждают, что основная проблема использования традиционных реляционных баз данных для хранения временны*х* рядов заключается в низком уровне масштабируемости, предопределяющем недостаточную производительность операций передачи и извлечения данных из памяти при выполнении запросов, связанных с обработкой временны*х* значений. TimescaleDB предлагает перенастраивать схемы распределения памяти и отображения объектов в памяти таким образом, чтобы отразить временну*ю* природу данных и сократить обмен разрозненными данными.

Hilmar Buchta, "Combining Multiple Tables with Valid from/to Date Ranges into a Single Dimension," Oraylis blog, November 17, 2014, `https://perma.cc/B8CT-BCEK`

Несмотря на "заумный" заголовок, статья будет полезной всем, кто не считает себя экспертом по реляционным базам данных. В ней описан эффективный способ обработки данных, которые рассматриваются как действительные только в течение определенного промежутка времени, и показано, как правильно объединять такие данные, содержащиеся сразу в нескольких таблицах. Это необычайно распространенная и на удивление сложная задача. В рассматриваемом примере изучаются таблицы распределения кадровых ресурсов, в том числе для действительных дат трудоустройства сотрудника, для определенного отдела, для офиса с определенным расположением и для служебного автомобиля. Весьма вероятной задачей было бы определить статус сотрудника в определенный день — выходил ли он на работу, и если да, то в каком отделе и офисе, а также имел ли он в своем распоряжении служебный автомобиль? Звучит как интуитивно понятное решение, но использовать реляционные таблицы для получения ответа на поставленную задачу очень сложно. Тем не менее в этой статье он рассмотрен достаточно детально.

ГЛАВА 6

Статистические модели для временных рядов

В этой главе мы изучим некоторые линейные статистические модели, предназначенные для описания временных рядов. Эти модели основаны на линейной регрессии, но учитывают корреляции, возникающие между точками данных в одном и том же временном ряду, в отличие от стандартных методов, применяемых к перекрестным данным, в которых предполагается, что каждая точка данных в выборке не зависит от остальных точек данных.

Нам предстоит обсудить следующие модели.

- Модели авторегрессии (AR) и скользящего среднего (MA), а также интегрированная модель авторегрессии скользящего среднего (Autoregressive Integrated Moving Average — ARIMA)
- Векторная авторегрессия (VAR)
- Иерархические модели

Эти модели традиционно использовались в прогнозировании временных рядов и находят частое применение в самых разных задачах как научного толка, так и отраслевого моделирования.

Почему не линейная регрессия

Как специалист по анализу данных вы, вероятнее всего, знакомы с понятием линейной регрессии. Если нет, то примите к сведению, что у нее весьма ограниченная область применения: *с помощью моделей линейной регрессии обрабатываются независимые и равномерно распределенные данные.* Как мы подробно обсуждали в предыдущих главах, это утверждение не выполняется для временных рядов. Во временных рядах наблюдается сильная корреляция у близких по времени точек данных. На самом деле при отсутствии временных корреляций такие данные вряд ли можно будет применять для решения традиционных задач обработки временных рядов, таких как прогнозирование или наблюдение за динамическими временными процессами.

Зачастую в руководствах по анализу временных рядов создается ложное представление о том, что линейная регрессия не имеет практической ценности при изучении временных рядов. В них настаивается на том, что простые модели линейной регрессии попросту не применимы к такого рода данным. К счастью, это совсем не так. Самая обычная модель линейной регрессии по методу наименьших квадратов может применяться к данным временного ряда при соблюдении следующих условий.

Предположения о поведении временных рядов

- Временной ряд имеет линейную зависимость от своих предикторов.
- Ни одна входная переменная не является постоянной во времени и не коррелирует абсолютным образом с другой входной переменной. Это условие вытекает из требования традиционной линейной регрессии к независимым переменным при учете временного характера измерения данных.

Предположения об ошибке

- Для каждого момента времени ожидаемое значение ошибки, учитывая все объясняющие переменные для всех периодов времени (вперед и назад), равно нулю.
- Ошибка в любой заданный период времени не связана с входными данными в любой период времени в прошлом или будущем. Таким образом, график автокорреляционной функции ошибок не будет демонстрировать какую-либо закономерность.
- Дисперсия ошибки не зависит от времени.

Если предположения верны, то с помощью модели линейной регрессии по методу наименьших квадратов можно получить несмещенную оценку коэффициентов с учетом входных данных даже для данных временных рядов.[1] В этом случае выборочные отклонения оценок рассчитываются по тем же математическим формулам, что и в стандартной модели линейной регрессии. Поэтому, если ваши данные соответствуют только что перечисленным допущениям, можете смело применять к ним модель линейной регрессии, что, несомненно, позволит сформировать интуитивно понятное представление о поведении исследуемых временных рядов.

[1] Обратите внимание, что оценки по обычному методу наименьших квадратов являются несмещенными даже в случае, когда некоторые из этих условий не выполняются. Например, если ошибки являются коррелированными и гетероскедастичными (имеют неодинаковые дисперсии), то обычный метод наименьших квадратов все еще может давать несмещенную оценку коэффициентов, хотя возникают проблемы, связанные с эффективностью этой оценки. Дополнительная информация об эффективности оценки приведена в Википедии.

Описанные выше требования к данным временны*х* рядов аналогичны таковым в стандартной модели линейной регрессии, применяемым к перекрестным данным. В них всего лишь сделан акцент на временно*м* характере зависимостей, проявляющихся в наборе данных.

Не стоит считать модель линейной регрессии палочкой-выручалочкой. Ниже приведены некоторые из последствий применения этой модели к данным, не соответствующим указанным ранее предположениям.

- Коэффициенты не обеспечивают минимизацию ошибки модели.
- *p*-значения критериев, по которым определяется неравенство коэффициентов нулю, рассчитываются неправильно, поскольку основаны на ложных предположениях. Это также означает, что неправильно оценивается и значимость коэффициентов.

Модели линейной регрессии отличаются простотой и доступностью в реализации, но использование неправильной модели, конечно же, просто недопустимо!

Справедливо задаться вопросом, не слишком ли высокие требования выдвигаются специалистами по анализу временны*х* рядов, из-за чего они не могут воспользоваться моделью линейной регрессии? Практикующие аналитики время от времени делают определенные послабления в предположениях о действительности модели. Такой подход оказывается плодотворным только при полном понимании всех его возможных недостатков.

Важность соблюдения предположений о справедливости модели сильно зависит от предметной области. Иногда модель применяется с полным пониманием того, что исходные предположения не будут выполняться, но негативными последствиями такого шага можно пренебречь ввиду несравнимо большей выгоды от получаемого результата. Например, в высокочастотной торговле линейные регрессионные модели довольно популярны по ряду причин, хотя никто не верит, что биржевые данные в полной мере соответствуют всем стандартным предположениям.[2]

[2] В данном случае существуют некоторые смягчающие факторы, которые оправдывают использование стандартной модели линейной регрессии. Во-первых, некоторые специалисты считают, что на достаточно коротких масштабах времени движения финансовых рынков не коррелируют друг с другом. Во-вторых, ввиду высокой вычислительной эффективности модель линейной регрессии позволяет получить быстрый результат: даже не будучи точной в своих допущения, она будет считаться хорошей моделью в отрасли, для которой характерна высокая скорость изменения данных. В-третьих, финансовым организациям, использующим эти модели на практике, удается зарабатывать немалые деньги, поэтому их нельзя считать такими уж неправильными.

Что такое несмещенная оценка

Если оценка не является завышенной или заниженной, она считается несмещенной. Это, как правило, хорошо, хотя вы должны знать о *дилемме смещения и дисперсии*, которая свойственна как статистическим задачам, так задачам машинного обучения, в которых оценки параметров с более низким смещением, как правило, имеют более высокую дисперсию. Дисперсия оценки параметра отражает, насколько изменчивой будет оценка для разных выборок данных.

Если модель линейной регрессии хорошо подходит для решаемой вами задачи прогнозирования, то рассмотрите возможность использования функции `tslm()` пакета `forecast` (`https://perma.cc/TR6C-4BUZ`). Она специально предназначена для расчета обычной линейной регрессии по данным временны*х* рядов.

Статистические методы обработки временны*х* рядов

Рассмотрим статистические методы, разработанные специально для временны*х* рядов. Сначала изучим методы, применяемые к одномерным временны*м* рядам, начиная с простейшей авторегрессионной модели, в которой утверждается, что будущие значения временно*го* ряда являются функцией его прошлых значений. Затем перейдем к описанию более сложных моделей, завершая экскурс знакомством с векторной авторегрессионной моделью для многомерных временны*х* рядов и некоторыми более специализированными моделями анализа временны*х* рядов — GARCH и иерархического моделирования.

Авторегрессионные модели

Авторегрессионная модель (AR) опирается на интуитивное предположение о том, что прошлое предсказывает будущее и, следовательно, описывает процесс временно*го* ряда так, что значение в момент времени *t* выступает функцией значений ряда в более ранние моменты времени.

Описание авторегрессионных моделей не будет полным, если не позволить вам получить представление о том, как воспринимают такие модели и их свойства специалисты по статистике. Чтобы понять это, нужно начать с довольно длинного теоретического обоснования. Вы всегда можете его пропустить, если вам не интересны математические принципы определения свойств статистических моделей, предназначенных для описания временны*х* рядов.

Алгебраический расчет ограничений авторегрессионных процессов

Авторегрессия напоминает интуитивную попытку аппроксимации временно*го* ряда, не располагая никакой другой информацией, кроме данных самого

временного ряда. Именно это и подразумевает ее название: регрессия по прежним значениям для предсказания будущих значений.

Простейшая авторегрессионная модель, AR(1), описывает данные временного ряда следующим образом:

$$y_t = b_0 + b_1 y_{t-1} + e_t.$$

Значение ряда в момент времени t является функцией постоянной b_0, значения ряда на предыдущем временном шаге, умноженного на другую константу b_1 ($b_1 y_{t-1}$), и погрешности e_t, которая также изменяется со временем. Предполагается, что погрешность имеет постоянную дисперсию и нулевое среднее значение. Обозначаем авторегрессионный член, который учитывает только непосредственно предшествующий момент времени, как AR(1), поскольку он включает в себя упреждение при сдвиге на один шаг.

Кстати, модель AR(1) описывается формулой, идентичной применяемой в простой модели линейной регрессии, но с одной объясняющей переменной. Иначе говоря, она имеет такой вид:

$$Y = b_0 + b_1 x + e.$$

Можно рассчитать как ожидаемое значение y_t, так и его дисперсию по заданным значениям y_{t-1} при известных значениях b_0 и b_1 (уравнение 6.1).[3]

Уравнение 6.1

$$E(y_t \mid y_{t-1}) = b_0 + b_1 y_{t-1} + e_t,$$
$$Var(y_t \mid y_{t-1}) = Var(e_t) = Var(e)$$

Что означает символ "|"

В статистике применяются определения условных вероятностей, условных математических ожиданий и иных величин, зависящих от других переменных. Такая зависимость обозначается с помощью знака "|", который читается как "при условии". Так, например, P($A|B$) следует понимать как "вероятность A при условии B".

Почему мы использовали этот символ в предыдущем случае? Нам нужно определить ожидаемое значение процесса y_t в момент времени t — по его значению в предыдущий момент времени. Таким способом обозначается тот факт, что нам известна информация до момента времени $t - 1$. Если мы не располагаем требуемой информацией, то ожидаемое значение y_t можно вычислить согласно общим правилам. Однако, если нам удастся узнать значение на один шаг раньше, мы сможем точнее определить ожидаемое значение в момент времени t.

[3] В дальнейшем нумеруются только уравнения, на которые добавлены ссылки.

Обобщая последнее выражение, можно записать формулу вычисления текущего значения авторегрессионного процесса как функцию p последних значений, получая процесс AR(p).

Перейдем к более традиционной форме записи, в которой коэффициенты авторегрессии обозначаются символами ϕ:

$$y_t = \phi_0 + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \ldots + \phi_p y_{t-p} + e_t.$$

Как было показано в главе 3, понятие стационарности является ключевым в анализе временных рядов — оно учитывается во многих моделях временных рядов, включая авторегрессионные модели.

Условия стационарности AR-модели устанавливаются исходя из его общего определения. В уравнении 6.2 мы продолжим рассматривать простейшую модель AR(1).

Уравнение 6.2

$$y_t = \phi_0 + \phi_1 y_{t-1} + e_t.$$

Давайте предположим, что процесс является стационарным, а затем "посмотрим назад", чтобы увидеть, как это отражается на коэффициентах. Предположение о стационарности указывает на то, что ожидаемое значение процесса должно быть всегда одинаковым. Мы можем представить y_t через уравнение процесса AR(1):

$$E(y_t) = \mu = E(y_{t-1}).$$

По определению e_t имеет нулевое математическое ожидание. Кроме того, значения ϕ являются константами, поэтому их ожидаемые значения будут постоянными. Левая часть уравнения 6.2 сокращается до такого выражения:

$$E(y_t) = E(\phi_0 + \phi_1 y_{t-1} + e_t),$$

$$E(y_t) = \mu.$$

Правая часть принимает следующий вид:

$$\phi_0 + \phi_1 \mu + 0.$$

Впоследствии все выражение упрощается до вида

$$\mu = \phi_0 + \phi_1 \mu.$$

В результате получаем уравнение 6.3.

Уравнение 6.3

$$\mu = \frac{\phi_0}{1 - \phi_1}.$$

Таким образом описывается взаимосвязь между математическим ожиданием и соответствующими коэффициентами процесса AR(1).

Выполним подобные действия для определения влияния на коэффициенты ϕ условия неизменности дисперсии и ковариации. Начнем с подстановки значения ϕ_0 в уравнение 6.2.

Уравнение 6.4

Подставив $\phi_0 = \mu(1 - \phi_1)$ в уравнение 6.2, получим

$$y_t = \phi_0 + \phi_1 y_{t-1} + e_t,$$

$$t = (\mu - \mu\phi_1) + \phi_1 y_{t-1} + e_t,$$

$$y_t - \mu = \phi_1(y_{t-1} - \mu) + e_t.$$

Изучив уравнение 6.4, легко заметить, что в его левой и правой частях содержатся схожие выражения, а именно: $y_t - \mu$ и $y_{t-1} - \mu$. Учитывая это, можно сделать вывод о стационарности временного ряда, поскольку значение в момент времени $t - 1$ такое же, как и в момент времени t. Перепишем уравнение 6.4 для предыдущей временной точки, представив его в виде уравнения 6.5.

Уравнение 6.5

$$y_{t-1} - \mu = \phi_1(y_{t-2} - \mu) + e_{t-1}.$$

Подставим полученное выражение в уравнение 6.4:

$$y_t - \mu = \phi_1(\phi_1(y_{t-2} - \mu) + e_{t-1}) + e_t.$$

Для большей ясности перепишем уравнение так.

Уравнение 6.6

$$y_t - \mu = e_t + \phi_1(e_{t-1} + \phi_1(y_{t-2} - \mu)).$$

Обратите внимание на то, что уравнение 6.6 можно преобразовать в несколько иную форму, проведя рекурсивные подстановки для члена $y_{t-2} - \mu$, но на этот раз выразив его не через значение y_{t-1}, а через y_{t-2}. Выполнив такую замену, можно получить более понятный результат:

$$y_t - \mu = e_t + \phi_1(e_{t-1} + \phi_1(e_{t-2} + \phi_1(y_{t-3} - \mu))) = e_t + \phi e_{t-1} + \phi^2 e_{t-2} + \phi^3 e_{t-3} +$$
$$+ \text{(следующие подставляемые выражения)}.$$

В общем случае формула принимает такой вид:

$$y_t - \mu = \sum_{i=1}^{\infty} \phi_1^i e_{t-i}.$$

Проще говоря, разность между y_t и математическим ожиданием процесса является линейной функцией остаточных членов. Полученный результат можно использовать для вычисления математического ожидания $E[(y_t - \mu)e_{t+1}] = 0$ при условии независимости значений e_t при разных значениях t. Отсюда можно сделать вывод, что ковариация y_{t-1} и e_t равна 0, как и предполагалось. Мы можем применить такой же подход для вычисления дисперсии y_t, возведя в квадрат следующее уравнение:

$$y_t - \mu = \phi_1(y_{t-1} - \mu) + e_t,$$
$$var(y_t) = \phi_1^2 var(y_{t-1}) + var(e_t).$$

Вследствие стационарности дисперсия должна быть одинаковой в обеих частях уравнения, или $var(y_t) = var(y_{t-1})$, поэтому

$$var(y_t) = \frac{var(e_t)}{1 - \phi_1^2}.$$

Учитывая тот факт, что по определению дисперсия больше или равна нулю, величина ϕ_1^2 должна быть меньше единицы, чтобы обеспечить положительное значение в правой части предыдущего уравнения. Следовательно, для стационарного процесса должно выполняться условие $-1 < \phi_1 < 1$. Это необходимое и достаточное условие слабой стационарности.

Слабая и сильная стационарность

Говоря о стационарности, мы подразумеваем слабую стационарность. А она предполагает инварианты по времени только математического ожидания и дисперсии процесса.

Сильная стационарность требует, чтобы распределение случайных величин, описывающих процесс, оставалось неизменным во времени. Например, оно требует, чтобы у статистических распределений y_1, y_2, y_3 и y_{101}, y_{102}, y_{103} совпадали все параметры, а не только два первых показателя (математическое ожидание и дисперсия).

Забавный факт: ожидается, что сильная стационарность полностью включает случай слабой стационарности, но на практике это не совсем так. Сильная стационарность заключает в себе слабую стационарность, когда процесс можно описать терминами математического ожидания и дисперсии, но если он не характеризуется математическим ожиданием или дисперсией, то все еще может оставаться сильно стационарным, даже не являясь слабо стационарным. Классическим примером будет распределение Коши, которое имеет неопределенное математическое ожидание и бесконечную дисперсию. Процесс Коши не удовлетворяет требованиям слабой стационарности, но в полной мере соответствует требованиям сильной стационарности.

Мы изучили процесс AR(1) как самый простой авторегрессионный процесс. Наряду с этим на практике приходится иметь дело с более сложными моделями. Можно вывести равнозначные условия стационарности для процесса произвольного порядка AR(p), и существует очень много изданий, в которых показано, как это делается. Если вы хотите хорошо разобраться в этом вопросе, обратитесь к источникам, приведенным в конце главы. Главный вывод, который нужно сделать из приведенного выше обсуждения, — моделирование временных рядов не требует сложных алгебраических и статистических вычислений, а стационарность — это не просто условие возможности построения графика модели, а математическая концепция, которая отражает особенности рассматриваемой статистической модели.

Распределение — это статистическая функция, описывающая для всех возможных значений вероятность того, что конкретное значение будет сгенерировано процессом. Хотя вам, возможно, не доводилось встречаться с формальным определением этого термина, вы наверняка сталкивались с ним на практике. Например, рассмотрим *колоколообразную кривую*, описывающую распределение вероятностей того, что большинство наблюдений близко к среднему значению и характеризуется одинаковым отклонением по обеим сторонам. В статистике оно известно как *нормальное*, или *гауссово, распределение*.

Выбор параметров модели AR(p)

Чтобы оценить уместность модели AR для своих данных, начните с построения графика процесса и его *функции частичной автокорреляции* (Partial Autocorrelation Function — PACF). Функция PACF процесса AR срезается к нулю за пределами порядка p процесса AR(p), тем самым явным образом обозначая порядок процесса.

С другой стороны, функция автокорреляции (Autocorrelation Function — ACF) процесса AR не является столь информативной, хотя и имеет характерную форму, демонстрируя экспоненциальное падение с увеличением временного сдвига.

Рассмотрим некоторые фактические данные о прогнозировании спроса, заимствованные из хранилища UCI Machine Learning (`https://perma.cc/B7EQ-DNLU`).

Сначала изобразим исходные данные на графике в хронологическом порядке (рис. 6.1). Поскольку мы моделируем AR-процесс, построим функцию PACF, чтобы установить порядок процесса, по которому выполняется срез (рис. 6.2).

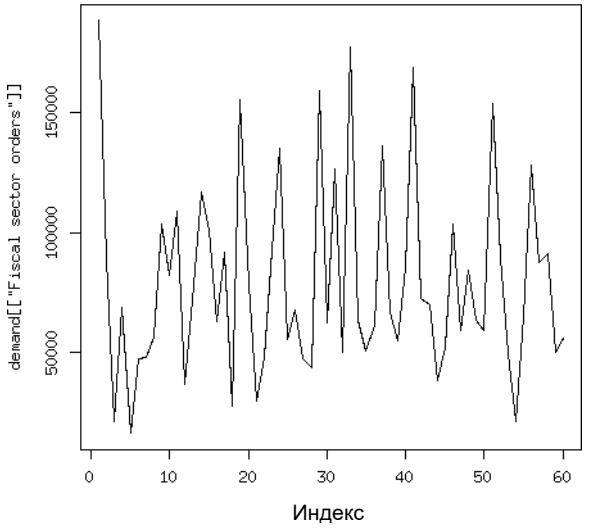


Рис. 6.1. *Ежедневное количество банковских переводов*

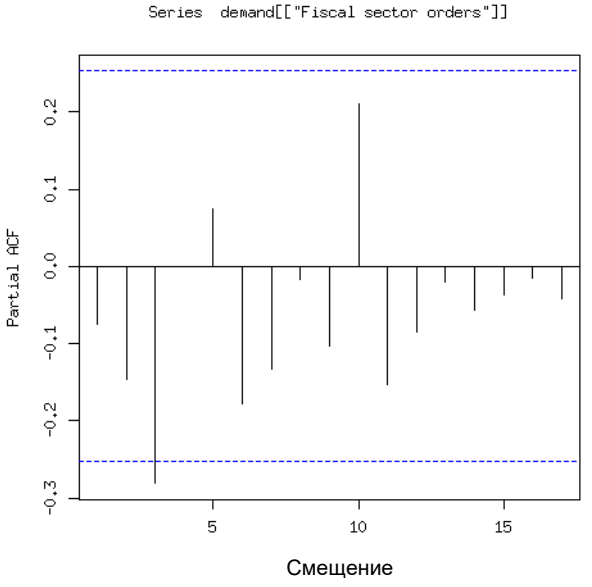


Рис. 6.2. *Функция частичной автокорреляции исходного временного ряда банковских переводов, показанного на рис. 6.1*

Мы можем видеть, что значение PACF пересекает порог значимости 5% при сдвиге, равном 3. Это согласуется с результатами, возвращаемыми функцией `ar()` пакета `stats` в языке R.

```
## R
> fit <- ar(demand[["Banking orders (2)"]], method = "mle")
> fit

Call:
ar(x = demand[["Banking orders (2)"]], method = "mle")
Coefficients:
     1          2          3
-0.1360    -0.2014    -0.3175
```

В документации к функции `ar()` указано, что возвращаемый порядок определяется (для параметров по умолчанию, которые нами не изменялись) по *информационному критерию Акаике* (AIC). Это очень важный момент: визуальная оценка, выполненная при изучении функции PACF, полностью согласуется с полученной в результате минимизации информационного критерия. Это два разных способа определения порядка модели, но в данном случае они возвращают согласованный результат.

Информационный критерий Акаике

Значение AIC для модели рассчитывается как $AIC = 2k - 2\ln L$, где k — количество параметров модели, а L — значение максимального правдоподобия функции. Как правило, требуется уменьшить сложность модели (значение параметра k), одновременно увеличивая правдоподобие/точность подгонки модели (параметра L). Поэтому предпочтение отдается моделям с меньшими значениями AIC перед моделями с более высокими значениями AIC.

Функция правдоподобия — это мера того, насколько вероятен конкретный набор параметров для функции относительно других параметров этой функции для заданных данных. Представьте, что вы строите модель линейной регрессии значений y от x для следующего набора данных.

x	y
1	1
2	2
3	3

Если применить к таким данным модель $y = b \cdot x$, то функция правдоподобия покажет, что оценка $b = 1$ гораздо более вероятна, чем $b = 0$. Функции правдоподобия можно рассматривать как инструмент определения наиболее вероятных истинных параметров модели для заданного набора данных.

Обратите внимание на то, что функция `ar()` также возвращает коэффициенты модели. При необходимости коэффициенты можно ограничить. Например, изучая функцию PACF, задайтесь вопросом, действительно ли нужно включить

коэффициент для члена lag – 1 или его лучше обнулить, учитывая, что соответствующее значение PACF значительно ниже уровня значимости. В этом случае можно также использовать функцию `arima()` пакета `stats`.

Ниже описана процедура вызова функции аппроксимации процесса AR(3) при установке параметра порядка `c(3, 0, 0)`, где 3 указывает порядок AR-компоненты (компоненты других порядков будут описаны в последующих разделах при рассмотрении примеров разностной модели и модели скользящего среднего).

```
## R
> est <- arima(x = demand[["Banking orders (2)"]],
>              order = c(3, 0, 0))
> est

Call:
arima(x = demand[["Banking orders (2)"]], order = c(3, 0, 0))

Coefficients:
         ar1      ar2     ar3    intercept
      -0.1358  -0.2013  -0.3176  79075.350
s.e.   0.1299   0.1289   0.1296   2981.125

sigma^2 estimated as 1.414e+09: log likelihood = -717.42,
                                aic = 1444.83
```

Чтобы внедрить в модель априорное знание или мнение, нужно положить коэффициент равным нулю. Например, чтобы ограничить модель нулевым значением члена lag – 1, нужно использовать следующий код.

```
## R
> est.1 <- arima(x = demand[["Banking orders (2)"]],
>                order = c(3, 0, 0),
>                fixed = c(0, NA, NA, NA))
> est.1

Call:
arima(x = demand[["Banking orders (2)"]],
      order = c(3, 0, 0),
      fixed = c(0, NA, NA, NA))

Coefficients:
      ar1      ar2      ar3   intercept
        0  -0.1831 -0.3031   79190.705
s.e.    0   0.1289  0.1298    3345.253

sigma^2 estimated as 1.44e+09: log likelihood = -717.96,
                               aic = 1443.91
```

Передавая значение в векторной форме как фиксированный параметр функции `arima`, нужно установить его равным `0`, а не `NA`, чтобы правильно задать ограничение.

```
## R
> fixed <- c(0, NA, NA, NA)
```

Теперь проверим точность модели на обучающих данных для оценки соответствия модели исходному набору данных. Мы можем сделать это двумя способами. Начнем с построения функции ACF для невязок (т.е. ошибок), чтобы увидеть, существует ли модель самокорреляции, которую не учитывает наша модель.

Построить график невязок довольно просто — достаточно воспользоваться результатом функции `arima()` (рис. 6.3).

```
## R
> acf(est.1$residuals)
```

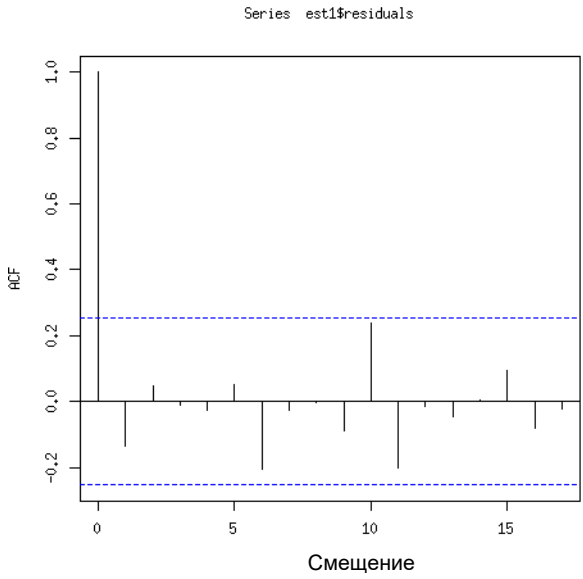


Рис. 6.3. *ACF для невязок в модели AR(3) при аппроксимации с обнуленным параметром lag – 1*

Ни одно из значений функции ACF не пересекает порог значимости. Конечно, мы не должны слепо ориентироваться на пороговое значение при определении или опровержении значимости, но оно оказывается полезным при оценке модели, которая считается правдоподобной по другим критериям.

В данном случае нам не удалось обнаружить самокорреляцию среди невязок (т.е. ошибок). В случае ее выявления нам пришлось бы пересмотреть исходную модель, рассмотрев возможность включения в нее дополнительных членов для учета большей части автокорреляции невязок.

Другой распространенный тест — это *критерий Льюнга–Бокса* (Ljung-Box), проверяющий случайность временного ряда. Точнее говоря, этот критерий устанавливает справедливость нулевой и альтернативной гипотез.

- H0: данные не демонстрируют последовательную корреляцию.
- H1: данные демонстрируют последовательную корреляцию.

Такой тест обычно применяется к AR-моделям (и в более общем смысле — к моделям ARIMA), а точнее — к невязкам оценки соответствия модели, а не самой модели.

```
## R
> Box.test(est.1$residuals, lag = 10, type = "Ljung", fitdf = 3)

Box-Ljung test

data: est.1$residuals
X-squared = 9.3261, df = 7, p-value = 0.2301
```

Применим тест Льюнга–Бокса к модели `est.1`, чтобы оценить точность аппроксимации. Мы не можем отвергнуть нулевую гипотезу о том, что данные не демонстрируют последовательную корреляцию. Это подтверждает результат, полученный при изучении графика ACF для невязок.

Прогнозирование с помощью процесса AR(*p*)

В следующих разделах будет показано, как составлять прогнозы с помощью AR-процессов. Сначала рассмотрим случай прогнозирования на один временной шаг вперед, а затем обсудим повышение уровня сложности прогнозирования на несколько шагов вперед. Хорошая новость заключается в том, что с точки зрения программирования обе задачи решаются одинаково, но в последнем случае объем вычислений намного больше.

Прогнозирование на один шаг вперед. Сначала рассмотрим случай прогнозирования на один шаг вперед с помощью известной (оценочной) AR-модели. В этом случае мы располагаем всей необходимой информацией.

Продолжим работать с моделью на основе данных о банковских переводах с коэффициентом lag – 1, равным 0 (подбирается так же, как в модели `est.1`).

Построим прогноз, используя функцию `fitted()` пакета `forecast`. Ниже приведен полный код решения.

```
## R
> require(forecast)
> plot(demand[["Banking orders (2)"]], type = 'l')
> lines(fitted(est.1), col = 3, lwd = 2) ## используем пакет forecast
```

Результатом будет график, показанный на рис. 6.4.

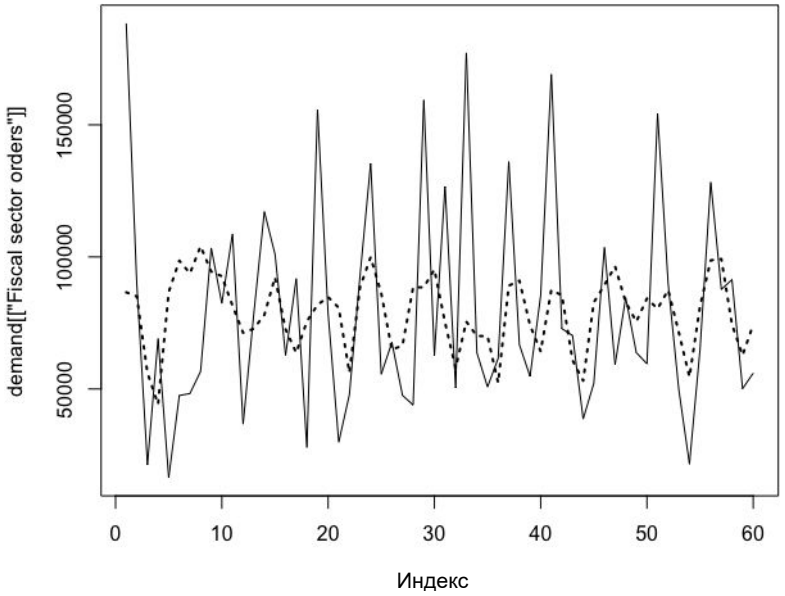


Рис. 6.4. *Исходный временной ряд и его аппроксимация (пунктирная линия)*

AR(*p*)-модели и функции скользящего окна

При прогнозировании нашей модели AR(3) можно отказаться от встроенной функции пакета `forecast` — вместо нее можно применять описанную ранее функцию `rollapply()`, позволяющую составлять более точные прогнозы. Эта оконная функция прекрасно справляется с моделированием AR-процессов. Чтобы построить прогноз в каждой точке, нужно заимствовать коэффициенты из функции `ar()` и использовать их в качестве весов входного вектора, представляющего значения отдельных сдвигов. Оставим эту задачу на ваше рассмотрение в качестве домашнего задания.

Теперь давайте задумаемся о точности прогноза. Вычислив корреляцию между прогнозируемым и фактическим значениями, получим значение 0,29. В некоторых случаях это достаточно неплохой результат, но не забывайте, что разностный метод часто опровергает существование, казалось бы, устойчивых связей в таких данных в пользу случайных зависимостей.

В частности, это имеет место в не полностью стационарных данных, для аппроксимации которых применяется очень точная модель, маскирующая нераспознанный тренд, который следовало найти и изучить моделированием.

Будем рассчитывать разности как между значениями ряда, так и между прогнозируемыми значениями, чтобы увидеть, насколько хорошо модель предсказывает переход от одного периода времени к другому. Даже после вычисления разностей

результат прогнозирования и исходные данные демонстрируют схожие закономерности, что указывает на правдоподобность используемой модели (рис. 6.5).

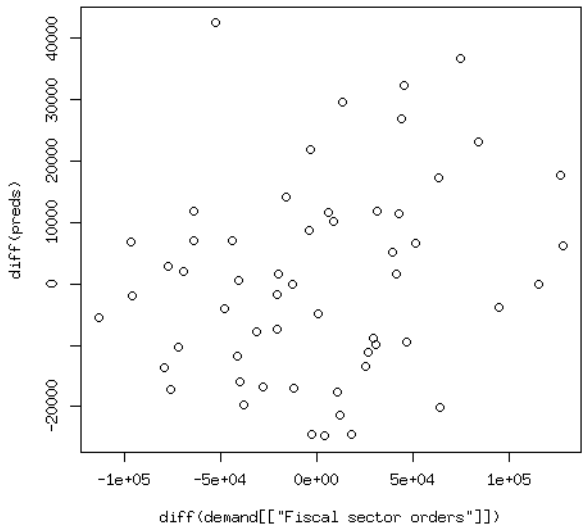


Рис. 6.5. *Разности между значениями исходного временного ряда и прогнозируемыми значениями, демонстрирующие корректность модели*

Кроме того можно проверить, предсказывается ли одно и то же движение в одно и то же время, построив график разностного ряда и изучив его корреляцию. Действительно, на графике просматривается слабая корреляция, которую можно подтвердить, рассчитав математически. Модель работает даже при прогнозировании изменений от одного временного шага к следующему.

Сопоставив графики исходных и прогнозируемых значений, можно заметить, что основное различие заключается в меньшей изменчивости результатов прогнозирования по сравнению с исходными данными. Модель может правильно предсказать направление движения в будущем, но не величину такого изменения от одного периода времени к другому. Это не проблема, а скорее особенность прогнозирования, которая заключается в том, что прогнозы рассчитываются как математические ожидания соответствующих распределений, и поэтому обязательно имеют меньшую изменчивость, чем выборочные данные.

Об этом свойстве статистических моделей иногда забывают ввиду доступности средств быстрой визуализации, которые, как правило, демонстрируют более стабильное будущее, чем наблюдается в действительности. Прибегнув к визуализации данных, не поленитесь объяснить аудитории, что именно изображено на графике. В нашем случае прогноз говорит о более стабильном будущем, чем оно является на самом деле.

Прогнозирование на много шагов вперед. До настоящего момента мы составляли прогноз всего на шаг вперед. Тем не менее часто требуется предсказать данные для более отдаленного периода времени. Сначала давайте представим, как можно сделать прогноз на два, а не на один, шага вперед. Очевидным будет решение, в котором сначала делается прогноз на один шаг вперед, а затем на основе полученного значения y_t строится прогноз значения y_{t+1}.

Обратите внимание на то, что в нашей текущей модели переход от прогноза на один шаг вперед к прогнозу на два шага вперед фактически не требует выполнения таких действий, поскольку значение y_{t-1} не принимает участия в прогнозировании значения y_t. Все, что нужно знать для составления прогноза на два временны*х* шага вперед, нам известно, и в угадывании данных нет особой необходимости. В действительности мы можем обойтись тем же рядом значений, что и при составлении прогноза на шаг вперед, — новых источников ошибок или изменчивости данных не наблюдается.

Однако если требуется прогнозировать более отдаленное будущее, то в качестве входных данных придется использовать сгенерированные будущие значения. Посмотрим, как можно спрогнозировать значение y_{t+3}. Используемая модель предполагает расчет коэффициентов, зависящих от значений y_{t+1} и y_t. Таким образом, сначала нам нужно предсказать оба этих значения — y_{t+1} и y_t, — а затем по очереди использовать полученные оценочные значения для прогнозирования значения y_{t+3}. Как и прежде, воспользуемся для этого функцией `fitted()` пакета `forecast`. С точки зрения программирования такая задача не намного сложнее составления прогноза на один шаг вперед. Как упоминалось ранее, прогноз также можно получить с помощью метода `rollapply()`, но он имеет более высокую вычислительную сложность и подверженность ошибкам.

Воспользуемся функцией `fitted()` с дополнительным параметром `h` для установки горизонта прогнозирования. Напомним, что объект `est.1` представляет процесс AR(3) с коэффициентом lag – 1 (время минус 1), приравненным к нулю.

```
## R
> fitted(est.1, h = 3)
```

Простота прогнозирования на несколько временны*х* шагов вперед позволяет составлять многоэтапные прогнозы для самых разных горизонтов. В следующем примере мы рассмотрим разницу в прогнозах для разных временны*х* горизонтов, полученных для общей базовой модели. (Обратите внимание, что на отображаемые данные сильное влияние оказывают операции округления и сокращения количества десятичных разрядов, поэтому лучше всего анализировать будущее по дисперсии оценочных значений в зависимости от временно*г*о горизонта.)

Как показано на рис. 6.6, дисперсия спрогнозированных значений уменьшается с увеличением временно*г*о горизонта. Причина тому, подчеркивающая важное ограничение модели, заключается в уменьшающемся влиянии фактических

данных по мере продвижения по временному горизонту — при расчете коэффициентов для входных данных учитывается только небольшое количество предыдущих временных точек (lag – 3 в используемой модели, т.е. в текущий момент времени – 3). Один из выводов, которые нужно сделать при исследовании полученных результатов — в более отдаленном периоде времени следует говорить о безусловном прогнозе, т.е. о не зависящем от данных. По мере роста временного горизонта прогноз будущих значений приближается к математическому ожиданию ряда, и следовательно, дисперсия как ошибки, так и прогнозных значений будет стремиться к нулю, поскольку прогнозные значения стремятся к безусловному математическому ожиданию.

```
## R
> var(fitted(est.1, h = 3), na.rm = TRUE)
[1] 174,870,141
> var(fitted(est.1, h = 5), na.rm = TRUE)
[1] 32,323,722
> var(fitted(est.1, h = 10), na.rm = TRUE)
[1] 1,013,396
> var(fitted(est.1, h = 20), na.rm = TRUE)
[1] 1,176
> var(fitted(est.1, h = 30), na.rm = TRUE)
[1] 3.5
```

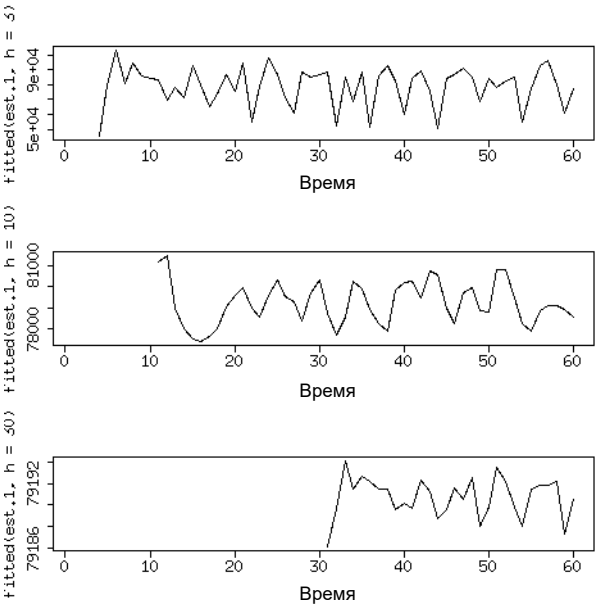


Рис. 6.6. *Графики спрогнозированных будущих значений. Диапазон значений, отложенных вдоль оси* ***y****, сужается по мере продвижения в будущее время — процесс все точнее описывается одним только математическим ожиданием. Горизонт прогнозирования увеличивается сверху вниз и установлен равным соответственно 3, 10 и 30*

Прогнозы на более длительные периоды времени в будущем строятся преимущественно на математическом ожидании процесса, что вполне обосновано. В какой-то момент времени в отдаленном будущем текущие данные перестанут предоставлять информацию о будущих изменениях, а прогнозирование будет сводиться к преимущественному изучению таких базовых статистических характеристик процесса, как математическое ожидание.

Из этого можно сделать важный вывод о том, что модели AR (а также MA, ARMA и ARIMA, о которых вы узнаете в последующих разделах) лучше всего подходят для краткосрочного прогнозирования. Они теряют предсказательную силу для отдаленных горизонтов в будущем.

При рассмотрении остальных моделей мы будет придерживаться такого же плана действий, как и в случае авторегрессионной модели, хотя и с меньшей степенью детализации процесса. Полное описание рассматриваемых далее методик приведено в узкоспециализированных учебниках по анализу временны*х* рядов.

Модели скользящего среднего

Модель скользящего среднего (Moving Average — MA) предназначена для описания процессов, в которых значение в каждой отдельной временно*й* точке является функцией недавних компонентов "ошибок", каждая из которых не зависит от других. Давайте рассмотрим модель скользящего среднего согласно такой же процедуре, что и при изучении авторегрессионной модели.

Сходство моделей AR и MA

Во многих случаях процесс MA можно представить через AR-процесс бесконечного порядка. И наоборот, очень часто AR-процесс представляется как процесс MA бесконечного порядка. Подробно об этом рассказано в статьях, описывающих обратимость процесса MA (`https://perma.cc/GJ6B-YASH`), теорему представления Вольда (`https://perma.cc/B3DW-5QGB`) и двойственность процессов MA/AR в целом (`https://perma.cc/K78H-YA6U`). Математические принципы, лежащие в их основе, выходят за рамки материала, рассматриваемого в этой книге!

Определение модели

Модель скользящего среднего похожа на авторегрессионную модель, за исключением того, что в ней члены линейного уравнения представлены текущими и предыдущими ошибками, а не текущими и предыдущими значениями самого процесса. Таким образом, модель MA порядка q записывается как

$$y_t = \mu + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \ldots + \theta_q e_{t-q}.$$

Не путайте модель MA со скользящим средним. Это не одно и то же. Если вы знаете, как аппроксимировать процесс методом скользящего среднего, то можете сравнить полученный результат с приближением с помощью модели MA для базового временного ряда. Попробуйте выполнить это упражнение самостоятельно.

Экономисты рассматривают компоненты ошибок как возмущения системы, в то время как инженеры-электротехники интерпретируют их как серию импульсов, а саму модель — как фильтр с конечной импульсной характеристикой, предопределяющий воздействие отдельного импульса на систему конечным периодом времени. Важно не определение, а сама концепция, согласно которой текущее состояние процесса подвержено влиянию многих независимых прошлых событий, вносящих в него индивидуальные вклады.

Оператор сдвига назад

Оператор сдвига назад, или оператор задержки, применяется к отдельным точкам временных рядов, смещая их на один шаг назад.

$$B^k y_t = y_{t-k}$$

Оператор сдвига назад помогает упростить запись модели временных рядов. Например, модель MA можно переписать следующим образом.

$$y_t = \mu + (1 + \theta_1 B + \theta_2 B^2 + \ldots + q_q B^q)e_t.$$

Модели MA без наложения на их параметры каких-либо ограничений являются слабостационарными. Это связано с конечностью и инвариантностью математического ожидания и дисперсии процесса MA во времени, поскольку предполагается, что ошибки являются независимыми, одинаково распределенными и имеют нулевое математическое ожидание.

$$E(y_t = \mu + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} \ldots + \theta_q e_{t-q}) =$$
$$= E(\mu) + \theta_1 \cdot 0 + \theta_2 \cdot 0 + \ldots = \mu.$$

Для вычисления дисперсии процесса мы воспользуемся тем фактом, что члены e_t являются независимыми и одинаково распределенными, а также положениями общего статистического свойства, утверждающего, что дисперсия суммы двух случайных величин равна сумме их индивидуальных дисперсий и двукратной ковариации. Ковариация между независимыми и одинаково распределенными переменными равна нулю. Это приводит к следующему выражению.

$$Var(y_t) = \left(1 + \theta_1^2 + \theta_2^2 + \ldots + \theta_q^2\right)\sigma_e^2.$$

Таким образом, математическое ожидание и дисперсия процесса MA постоянны во времени, независимо от значений параметров.

Подбор параметров процесса MA(*q*)

Построим модель MA на тех же данных, которые применялись при описании модели AR, используя функцию ACF для определения порядка процесса MA (рис. 6.7). Перед тем как продолжить, попробуйте разобраться в том, как работает модель MA, и объяснить, почему для определения порядка процесса применяется функция ACF, а не PACF.

```
## R
> acf(demand[["Banking orders (2)"]])
```

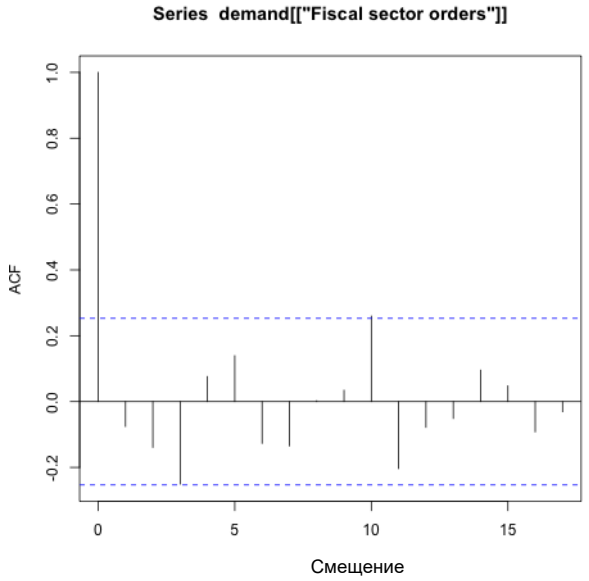


Рис. 6.7. *Применим функцию ACF к временному ряду о банковских переводах, чтобы определить порядок модели MA*

Шаблоны ACF и PACF различаются в процессах MA и AR

В отличие от авторегрессионного процесса, который характеризуется медленно убывающей ACF, процессу MA свойственно резкое обнуление ACF при любом значении порядка, превышающем *q*. Это связано с тем, что авторегрессионный процесс зависит от предыдущих членов, а они учитывают предыдущие импульсы в системе, в то время как модель MA, учитывающая импульсы непосредственно по их значениям, обладает механизмом предотвращения бесконечного влияния.

Наиболее значимые величины наблюдаются в сдвигах 3 и 9, поэтому будем подгонять модель MA именно к ним. Нужно проявить осторожность, чтобы случайно не обнулить не те коэффициенты, для чего выводим их на экран.

```
##R
> ma.est = arima(x = demand[["Banking orders (2)"]],
                 order = c(0, 0, 9),
                 fixed = c(0, 0, NA, rep(0, 5), NA, NA))
> ma.est
Call:
arima(x = demand[["Banking orders (2)"]], order = c(0, 0, 9),
      fixed = c(0, 0, NA, rep(0, 5), NA, NA))

Coefficients:
     ma1 ma2     ma3  ma4  ma5  ma6  ma7  ma8      ma9 intercept
       0   0 -0.4725    0    0    0    0    0  -0.0120  79689.81
s.e.   0   0  0.1459    0    0    0    0    0   0.1444   2674.60

sigma^2 estimated as 1.4e+09: log likelihood = -717.31,
                              aic = 1442.61
```

Кроме того, нужно проверить точность аппроксимации, как это делалось для модели AR, построив график ACF для невязок модели. В качестве дополнительной, дублирующей проверки точности модели применим к невязкам тест Льюнга-Бокса для оценки общей стохастичности в каждой аппроксимации. Обратите внимание, что при передаче данных в функцию `Box.test()` необходимо указать количество степеней свободы — число параметров модели, подлежащих свободной оценке — не ограниченных определенным значением. В нашем случае свободными параметрами считаются сдвиг, а также члены MA3 и MA9.

```
## R
> Box.test(ma.est$residuals, lag = 10, type = "Ljung", fitdf = 3)

Box-Ljung test

data: ma.est$residuals
X-squared = 7.6516, df = 7, p-value = 0.3643
```

Мы не можем отвергнуть нулевую гипотезу об отсутствии временной корреляции между точками невязок. При всем этом график ACF невязок не демонстрирует никакой временной корреляции (пусть это будет упражнением для домашнего выполнения).

Прогнозирование с помощью процесса MA(*q*)

Повторим прогноз на тех же исходных данных, придерживаясь методик, которые были продемонстрированы для AR-процесса, и полагаясь на метод `fitted()` пакета `forecast`.

```
## R
> fitted(ma.est, h=1)
Time Series:
Start = 1
End = 60
Frequency = 1
[1]   90116.64  80626.91 74090.45  38321.61  74734.77 101153.20 65930.90
[8]  106351.80 104138.05 86938.99 102868.16  80502.02  81466.01  77619.15
[15] 100984.93  81463.10 61622.54  79660.81  88563.91  65370.99 104679.89
[22]  48047.39  73070.29 115034.16  80034.03 70052.29  70728.85  90437.86
[29]  80684.44  91533.59 101668.18  42273.27 93055.40  68187.65  75863.50
[36]  40195.15  82368.91 90605.60   69924.83 54032.55  90866.20  85839.41
[43]  64932.70  43030.64 85575.32   76561.14 82047.95  95683.35  66553.13
[50]  89532.20  85102.64 80937.97   93926.74 47468.84  75223.67 100887.60
[57]  92059.32  84459.85 67112.16   80917.23
```

Модели MA характеризуются сильной реверсивностью математического ожидания, поэтому прогнозы быстро сходятся к математическому ожиданию процесса. Это обосновано тем, что процесс считается функцией белого шума.

Результатом прогнозирования вне пределов диапазона модели, определяемого ее порядком, неизбежно будет математическое ожидание процесса, что исходит из его определения. Рассмотрим модель MA(1):

$$y_t = \mu + \theta_1 e_{t-1} + e_t.$$

Прогнозирование следующего временного шага сводится к оценке значения y_{t+1}, равного $\mu + \theta_1 y_t + e_t$. При предсказании двух следующих временных шагов получим следующую оценку:

$$E(y_{t+2} = \mu + e_{t+2} + \theta_1 e_{t+1}) = \mu + 0 + \theta_1 \cdot 0 = \mu.$$

Для процесса MA(1) невозможно обосновать предсказание вне пределов единственного следующего шага, подобно тому, как для всего процесса MA(q) невозможно получить более обоснованное прогнозное значение вне пределов следующих q шагов, чем математическое ожидание процесса. Под *обоснованным* предсказанием подразумевается получение прогноза, на который влияют последние наблюдения.

Общепринятая форма записи — со знаком "минус"

Обратите внимание, что математическое определение модели MA обычно записывается не так, как показано выше. Обычно перед коэффициентами q указывается знак "минус". В нашем случае это объясняется другой интерпретацией модели MA, состоящей в том, что модель MA — это AR-модель с ограниченными параметрами. Эта формулировка в результате довольно сложных преобразований приводит к получению отрицательных коэффициентов q.

Мы можем убедиться в этом, составив прогноз на 10 временны*х* шагов вперед с помощью только что полученной модели MA(9).

```
## R
> fitted(ma.est, h=10)
Time Series:
Start = 1
End = 60
Frequency = 1
 [1]       NA       NA          NA NA NA          NA NA NA
 [9]       NA       NA 79689.81 79689.81 79689.81 79689.81 79689.81 79689.81
[17] 79689.81 79689.81 79689.81 79689.81 79689.81 79689.81 79689.81 79689.81
[25] 79689.81 79689.81 79689.81 79689.81 79689.81 79689.81 79689.81 79689.81
[33] 79689.81 79689.81 79689.81 79689.81 79689.81 79689.81 79689.81 79689.81
[41] 79689.81 79689.81 79689.81 79689.81 79689.81 79689.81 79689.81 79689.81
[49] 79689.81 79689.81 79689.81 79689.81 79689.81 79689.81 79689.81 79689.81
[57] 79689.81 79689.81 79689.81 79689.81
```

Попытавшись предсказать 10 следующих временны*х* шагов, мы рассчитываем среднее значение для каждого временно*г*о шага. Но это можно сделать без использования сложной статистической модели!

Во всем должен проявляться здравый смысл. Если вы применяете модель, не понимая, как она работает, то будете совершить большую глупость, например, каждый день отправляя своему начальнику один и тот же прогноз, полученный по неподходящей модели, на построение которой было затрачено много времени и ресуров.

Интегрированная модель авторегрессии скользящего среднего

После отдельного изучения моделей AR и MA перейдем к модели интегрированной модели регрессии скользящего среднего (ARIMA — Autoregressive Integrated Moving Average), в которой учитывается тот факт, что одни и те же временны*е* ряды могут проявлять базовую динамику процессов как AR, так и MA. Уже один этот факт приводит к модели ARMA, но мы расширим AR до модели ARIMA, предполагающей вычисление разностей, учет трендов и обработку нестационарных временны*х* рядов.

Модели ARIMA по-прежнему обеспечивают точность, близкую к современному уровню, особенно на небольших наборах данных, на которых более сложные модели машинного обучения или глубокого обучения выглядят не наилучшим образом. Однако даже модели ARIMA подвержены опасности чрезмерной подгонки, несмотря на их относительную простоту.

Вычисление разностей

Как обсуждалось ранее в книге, вычисление разностей — это операция преобразования временного ряда значений во временной ряд изменений значений во времени. Чаще всего это достигается путем вычисления парных разностей значений для смежных временных точек, так что значение разностного ряда в момент времени t представляется как разница между значениями в моменты времени t и $t - 1$ исходного ряда. Однако при необходимости разности можно вычислять для любых временных сдвигов.

Определение модели

Вам может показаться странной аппроксимация одних и те же данных с помощью моделей AR и MA без указания причин такого решения. С таким несколько раздражающим подходом вы можете столкнуться во многих учебниках по анализу временных рядов. Одни авторы, превозмогая лень, стараются использовать при описании разных моделей несколько различающиеся наборы данных, в то время как другие нарочито отказываются от такого подхода. Ранее мы не проверяли, какая из моделей лучше аппроксимирует исходные данные, но в процессе их изучения выяснили, что в разных условиях существуют предпосылки к использованию для описания данных каждой из моделей (AR и MA). Возникает вопрос: не стоит ли объединить их в одну модель?

Чтобы понять, какая из моделей — AR, MA или ARMA — лучше всех подходит для анализа временных рядов, обратимся к табл. 6.1.

Таблица 6.1. Лучшая модель для описания временного ряда

Вид графика	AR(p)	MA(q)	ARMA
ACF	Медленно убывает	Резко обрывается после lag = q	Без явных обрывов
PACF	Резко обрывается после lag = p	Медленно убывает	Без явных обрывов

Совмещение моделей AR и MA приводит к получению модели авторегрессионного скользящего среднего (ARMA — Autoregressive Moving Average), которая применяется в случаях, когда ни AR, ни MA не позволяет описать наблюдаемую динамику временного ряда с достаточной степенью точности. Довольно часто при определении порядковых статистик процессов AR и MA (PACF и ACF соответственно) можно получить ненулевые значения, указывающие на порядок модели AR или MA. Их можно успешно применять в модели ARMA.

Теорема Вольда

Теорема Вольда утверждает, что каждый ковариационно-стационарный временной ряд можно представить как сумму двух временных рядов — детерминированного и стохастического. Основываясь на этой теореме, можно показать, что стационарный процесс можно вполне обоснованно аппроксимировать моделью ARMA, хотя найти подходящую модель зачастую очень трудно.

Перейдем к рассмотрению математической формы записи модели ARMA, традиционно предваряя коэффициенты процесса MA знаками "минус":

$$y_t = \phi_0 + \Sigma\phi_i r_{t-i} + e_t - \Sigma\theta_i e_{t-i}.$$

Модель ARMA не единственно возможная

Если у компонентов AR и MA представленного выше уравнения наблюдаются общие множители, то модель ARMA(p, q) можно свести к другому набору параметров. В общем случае необходимо избегать такого рода вырожденных ситуаций. Старайтесь всегда выбирать самую простую модель ARMA. Дальнейшее обсуждение выходит за рамки материала книги, но с ним вы можете детально познакомиться в дополнительных источниках, приведенных в конце главы.

Один из способов выявления ситуаций, в которых могут наблюдаться такого рода трудности, — запись уравнения модели ARMA через оператор запаздывания (обсуждался ранее) и его полиномиальное разложение для поиска общих множителей в компонентах AR и MA, которые можно сократить, чтобы получить эквивалентную, но более простую модель. Рассмотрим пример. Один из способов представления уравнения модели ARMA состоит в разнесении компонентов AR и MA по разные стороны знака равенства.

Рассмотрим следующую модель ARMA:

$$y_t = 0{,}5y_{t-1} + 0{,}24y_{t-2} + e_t + 0{,}6e_{t-1} + 0{,}09e_{t-2}.$$

Уравнение можно переписать так, что члены модели AR (коэффициенты x) будут размещаться по одну, а члены MA — по другую сторону равенства:

$$y_t - 0{,}5y_{t-1} - 0{,}24y_{t-2} = e_t + 0{,}6e_{t-1} + 0{,}09e_{t-2}.$$

Теперь перепишем его с использованием оператора запаздывания. Так, например, y_{t-2} будет заменено на L^2y:

$$y_t - 0{,}5Ly_t - 0{,}24L^2y_t = e_t + 0{,}6Le_t + 0{,}09L^2e_t.$$

Выделив общие множители у члена y_t в левой и члена e_t в правой части уравнения, разложим его на множители:

$$(1 + 0{,}3L)(1 - 0{,}8L)y_t = (1 + 0{,}3L)(1 + 0{,}3L)e_t.$$

Обратите внимание на то, что при разложении на множители значения, на которые действует оператор запаздывания, записываются справа от него, чтобы выделять значения y_t, находящиеся справа от полиномиальных множителей, включающих оператор L, а не слева, что может показаться более естественным.

Выполнив факторизацию, легко обнаружить общий множитель в левой и правой частях уравнения, а именно $(1 + 0{,}3L)$, который нужно сократить:

$$(1 - 0{,}8L)y_t = (1 + 0{,}3L)e_t.$$

В результате выполнения описанных выше операций мы понизили порядок модели — теперь она включает значения только для момента времени $t - 1$, но не $t - 2$. Это указывает на то, что даже модель, которая не кажется слишком сложной, такая как ARMA(2, 2), может быть эквивалентна более простой модели, но это становится очевидным только после ее детального изучения.

Также не забывайте о возможности непосредственной аппроксимации моделей. Реальные наборы данных часто описываются уравнениями, в которых обе части численно близки, но не совпадают. Например, в левой части уравнения может располагаться такой множитель, как $(1 - 0{,}29L)$, а в его правой части — множитель $(1 - 0{,}33L)$. В подобных случаях стоит воспользоваться близостью множителей для уменьшения сложности модели.

Стационарность процесса ARMA обусловливается стационарностью его компонента AR и описывается тем же характеристическим уравнением, которое используется для определения стационарности модели AR.

От модели ARMA легко перейти к модели ARIMA. Различие между этими моделями проявляется в их названиях — аббревиатура "ARIMA" включает дополнительную букву "I", обозначающую понятие integrated (*интегрированная*), которое показывает, сколько раз следует вычислить разность, чтобы получить стационарный временной ряд.

Модели ARIMA находят более широкое практическое применение, чем модели AR, MA и ARMA, особенно в научных исследованиях и прогнозировании. Поиск в базе Google Scholar показывает, что ARIMA используется для решения самых разных задач прогнозирования, включая такие:

- прогнозирование потока авиапассажиров, прибывающих на Тайвань;
- прогнозирование объемов энергопотребления Турции по разным видам топлива;

- прогнозирование ежедневных объемов продаж овощного рынка Индии;
- прогнозирование потребности в пунктах неотложной помощи на Западном побережье США.

Важно отметить, что порядок вычисления разностей не должен быть слишком большим. В общем случае значение каждого параметра ARIMA(p, d, q) нужно минимизировать, чтобы избежать необоснованной сложности модели и чрезмерной подгонки к выборочным данным. Существует универсальное эмпирическое правило: следует предельно осторожно относиться к значениям d больше 2 и значениям p и q больше 5. Кроме того, наибольшим должен быть параметр p или q, и меньший из них должен быть относительно небольшим. Впрочем, эти рекомендации носят сугубо практический характер и не имеют точного математического обоснования.

Выбор параметров

Модель ARIMA описывается тремя параметрами (p, d, q). Значения p, d и q выбираются согласно имеющемуся набору данных.

Приведем несколько общеизвестных примеров модели ARIMA, описанных в Википедии.

- ARIMA(0, 0, 0) — модель белого шума.
- ARIMA(0, 1, 0) — модель случайного блуждания, а ARIMA(0, 1, 0) с ненулевой константой — модель случайного блуждания со смещением.
- ARIMA(0, 1, 1) — модель экспоненциального сглаживания.
- ARIMA(0, 2, 2) — то же, что и линейный метод Хольта, в котором модель экспоненциального сглаживания расширяется на данные с трендом. В результате ее можно использовать для прогнозирования данных, включающих тренд.

Порядок модели выбирается, исходя из знаний о предметной области, значений оцениваемых критериев (таких, как AIC) и сведений о поведении функций PACF и ACF для исследуемого процесса (см. табл. 6.1). Давайте попробуем настроить модель ARIMA как вручную, используя сведения о функциях PACF и ACF, так и автоматически — с помощью программных инструментов выбора параметров модели, — в частности функции `auto.arima()` пакета `forecast`.

Подгонка модели вручную. Для подбора параметров ARIMA вручную применяются вычислительные методики, в которых основным оцениваемым фактором является *простота модели*. Одна из самых старых, но до сих пор популярных из них — это модель Бокса–Дженкинса, представляющая собой итерационный многошаговый процесс.

1. Используйте исследуемые данные, имеющиеся графики и знания о предметной области для выбора класса модели, подходящей для описания данных.
2. Оцените параметры модели на обучающих данных.
3. Оцените точность модели на основе обучающих данных и подберите параметры модели так, чтобы устранить недостатки, выявленные при оценке точности.

Рассмотрим следующий пример подгонки модели. Для начала нам нужно определиться с исходными данными. Для лучшего понимания, а также для сверки полученного результата с правильным ответом заимствуем данные из процесса ARMA.

```
## R
> require(forecast)
> set.seed(1017)
> ## Порядок модели ARIMA скрыт преднамеренно
> y = arima.sim(n = 1000, list(ar = ###, ma = ###))
```

Не нужно подсматривать порядок созданной модели; давайте сосредоточимся на самостоятельном его определении. Сначала необходимо построить временной ряд, изучить его и убедиться в том, что он стационарный (рис. 6.8). Затем нужно проанализировать ACF и PACF для ряда *y* (рис. 6.9) и сравнить полученные сведения с данными, приведенными в табл. 6.1.

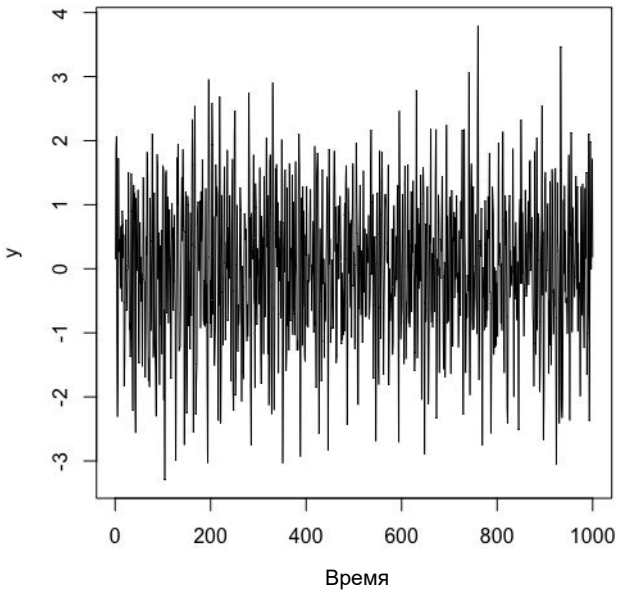


Рис. 6.8. *График временного ряда*

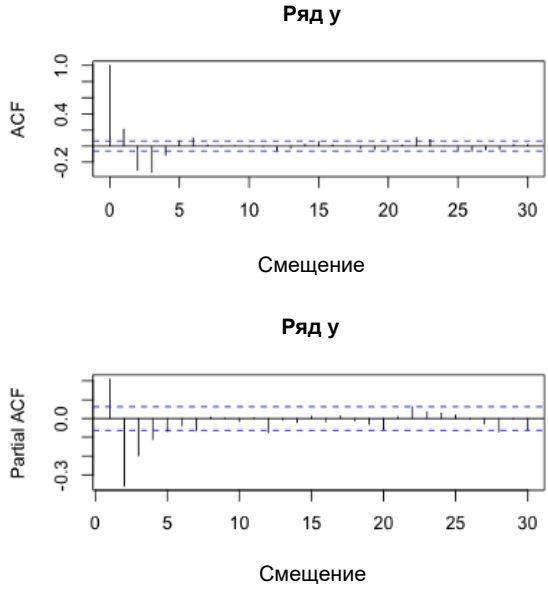


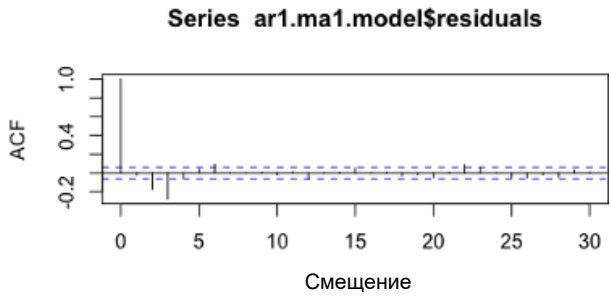
Рис. 6.9. *Функции ACF и PACF для временного ряда*

Как легко заметить, ни ACF, ни PACF, не имеют явно выраженного резкого обрыва, что позволяет (см. табл. 6.1) описать исследуемый процесс с помощью модели ARMA. Начнем подгонку с изучения относительно простой модели ARIMA(1, 0, 1), в которой не вычисляются разности значений ряда и не учитываются тенденции (рис. 6.10).

```
## R
> ar1.ma1.model = Arima(y, order = c(1, 0, 1))
> par(mfrow = c(2,1))
> acf(ar1.ma1.model$residuals)
> pacf(ar1.ma1.model$residuals)
```

На рис. 6.10 просматриваются необычайно сильные невязки, особенно в графике PACF, указывая на то, что модель не полностью описывает авторегрессионное поведение данных. Таким образом, нам нужно включить в модель компонент AR более высокого порядка. В следующем коде тестируется модель ARIMA(2, 0, 1) и строятся графики ACF и PACF для невязок улучшенной, более сложной модели (рис. 6.11).

```
## R
> ar2.ma1.model = Arima(y, order = c(2, 0, 1))
> plot(y, type = '1')
> lines(ar2.ma1.model$fitted, col = 2)
> plot(y, ar2.ma1.model$fitted)
> par(mfrow = c(2,1))
> acf(ar2.ma1.model$residuals)
> pacf(ar2.ma1.model$residuals)
```



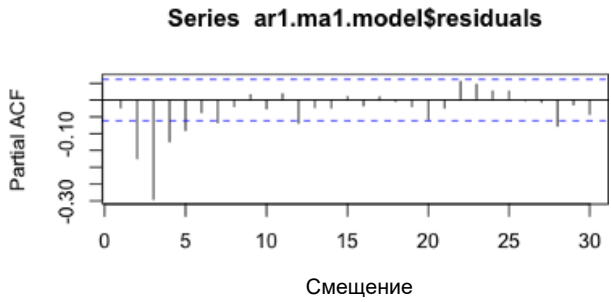


Рис. 6.10. *ACF и PACF невязок модели ARIMA(1, 0, 1)*

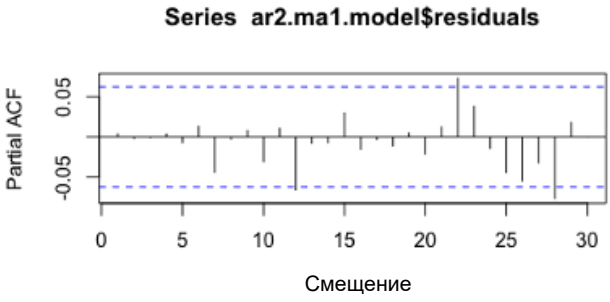


Рис. 6.11. *ACF и PACF невязок модели ARIMA(2, 0, 1)*

Невязки на рис. 6.11 уже не демонстрируют больших отклонений ни в ACF, ни в PACF. Учитывая требование получить как можно более простую модель ARIMA и избежать чрезмерной ее подгонки, дальнейшие изыскания можно прекратить, поскольку здесь невязки не демонстрируют поведения, которое нужно учитывать с помощью дополнительного авторегрессионного, скользящего среднего или разностного компонента модели. Вы всегда можете выполнить такое упражнение самостоятельно, рассмотрев возможность построения более сложных моделей. Хотя и без демонстрации графиков здесь, я все же протестировала более сложную модель с помощью следующего кода. Полученный результат показал, что точность модели заметно не повысилась для тех же исходных данных, а значения ACF или PACF не уменьшились на большую величину, чем в предыдущей модели. При желании убедитесь в этом самостоятельно.

```
## R
> ar2.ma2.model = Arima(y, order = c(2, 0, 2))
> plot(y, type = 'l')
> lines(ar2.ma2.model$fitted, col = 2)
> plot(y, ar2.ma2.model$fitted)
> par(mfrow = c(2,1))
> acf(ar2.ma2.model$residuals)
> pacf(ar2.ma2.model$residuals)
>
> ar2.d1.ma2.model = Arima(y, order = c(2, 1, 2))
> plot(y, type = 'l')
> lines(ar2.d1.ma2.model$fitted, col = 2)
> plot(y, ar2.d1.ma2.model$fitted)
> par(mfrow = c(2,1))
> acf(ar2.d1.ma2.model$residuals)
> pacf(ar2.d1.ma2.model$residuals)
```

Ниже приведен еще один, быстрый, способ сравнения моделей, который заключается в изучении корреляции спрогнозированных значений и фактических данных временного ряда.

```
## R
> cor(y, ar1.ma1.model$fitted)
[1] 0.3018926
> cor(y, ar2.ma1.model$fitted)
[1] 0.4683598
> cor(y, ar2.ma2.model$fitted)
[1] 0.4684905
> cor(y, ar2.d1.ma2.model$fitted)
[1] 0.4688166
```

Существенное улучшение наблюдается при переходе от модели ARIMA(1, 0, 1) к модели ARIMA(2, 0, 1) (первых два результата) — корреляция уменьшается с 0,3 до 0,47. Наряду с этим последующее увеличение сложности модели не вызывает

заметного повышения корреляции. Это еще раз подтверждает полученный ранее результат — модель ARIMA(2, 0, 1), скорее всего, достаточно хорошо описывает поведение данных, и нет необходимости добавлять в нее дополнительные компоненты AR или MA для лучшей подгонки.

Как бы там ни было, понять, насколько хорошо была выполнена подгонка, можно, сравнивая исходные коэффициенты (показанные здесь, хотя и скрытые ранее) с только что подобранными коэффициентами.

```
## R
## Исходные коэффициенты
> y = arima.sim(n = 1000, list(ar = c(0.8, -0.4), ma = c(-0.7)))
> ar2.ma1.model$coef
        ar1          ar2          ma1    intercept
0.785028320 -0.462287054 -0.612708282 -0.005227573
```

Здесь наблюдаем хорошее соответствие между подобранными и исходными коэффициентами, используемыми для моделирования данных.

Настройка модели ARIMA вручную — это более сложный процесс, чем описано выше. На протяжении десятилетий практикующие специалисты по анализу данных занимались выработкой хорошо зарекомендовавших себя эмпирических правил выявления проблем подгонки модели, вызванных существованием слишком большого количества членов определенного вида, слишком высоким порядком вычисления разницы и т.п. Все они достаточно хорошо рассмотрены в руководстве, написанном профессором Университета штата Пенсильвания, которое доступно для просмотра онлайн (`https://perma.cc/P9BK-764B`).

Подгонка модели ARIMA вручную часто подвергается вполне обоснованной критике. Такая настройка — это довольно плохо регламентированный процесс, в котором точность получаемого результата зависит преимущественно от квалификации специалиста по анализу данных. Он может отнимать много времени, что зависит от выбранного способа получения конечного результата. Тем не менее это удачное и эффективное решение, которое на протяжении десятилетий использовалось при прогнозировании реальных процессов, но оно не настолько точное, как того хотелось бы.

Автоматическая подгонка модели. В настоящее время мы можем отказаться от пошаговой подгонки модели вручную в пользу автоматического способа ее подюора. Выбор целевой модели осуществляется на основе самых разных информационных критериев (например, AIC, кратко описанного выше) с помощью функции `auto.arima()` пакета `forecast`:

```
## R
> est = auto.arima(demand[["Banking orders (2)"]],
          stepwise = FALSE, ## это медленное решение, но оно
                            ## обеспечивает более полный поиск
          max.p = 3, max.q = 9)
```

```
> est
Series: demand[["Banking orders (2)"]]
ARIMA(0,0,3) with non-zero mean

Coefficients:
          ma1       ma2       ma3      mean
      -0.0645   -0.1144   -0.4796 79914.783
s.e.   0.1327    0.1150    0.1915  1897.407

sigma^2 estimated as 1.467e+09: log likelihood=-716.71
AIC=1443.42 AICc=1444.53 BIC=1453.89
```

В этом примере функции передаются входные данные, выбор которых основан на результатах предыдущих исследований. В частности, мы указали наиболее подходящие с нашей точки зрения максимальные значения порядков процессов AR и MA, но программой была выбрана более простая модель, чем предполагалось, — полностью лишенная AR-составляющей. Тем не менее такая модель хорошо описывает исследуемые временны*е* ряды, и для подбора более точной модели нам потребуется слишком долго и упорно изучать исходные данные, что в рассматриваемом случае совершенно неоправданно. Обратите внимание, что подогнанная вручную в соответствии с критерием AIC модель MA, описанная в предыдущем разделе, работает несколько лучше, чем автоматически выбранная модель, но при изучении графиков разница в их точности оказывается несущественной.

В качестве домашнего задания выполните задачи по отображению на общем графике смоделированных и спрогнозированных значений и определению невязок автоматически выбранной модели. Убедитесь в том, что невязки такой модели не нуждаются в устранении путем увеличения порядка одной из составляющих. Приведенный выше код ничем не отличается от используемого при описании моделей AR и MA ранее. Приведенные выше замечания в полной мере справедливы для случая прогнозирования данных — мы умышленно не будем останавливаться на рассмотрении этой темы, поскольку автоматически выбранная модель ARIMA мало чем отличается от модели MA, обсуждавшейся и применяемой для получения прогнозов ранее.

Давайте вкратце изучим результат выполнения функции `auto.arima()` для модели, полученной методом ручной подгонки в предыдущем разделе.

```
## R
> auto.model = auto.arima(y)
> auto.model
Series: y
ARIMA(2,0,1) with zero mean

Coefficients:
         ar1     ar2     ma1
      0.7847 -0.4622 -0.6123
s.e. 0.0487  0.0285  0.0522
```

```
sigma^2 estimated as 1.019: log likelihood=-1427.21
AIC=2862.41 AICc=2862.45 BIC=2882.04
```

Здесь в функции `auto.arima()` не используются необязательные параметры, указывающие, с каких значений начинать выбор модели, и гарантирующие схождение к однажды полученному решению. Как видите, в отдельных случаях совершенно разные методы приводят к одному и тому же результату. В ручном способе при построении более сложных моделей мы ориентировались на функции ACF и PACF невязок более простых моделей, тогда как автоматический выбор, заключающийся в использовании функции `auto.arima()`, основан на задаче минимизации значения AIC. Конечно, учитывая факт получения исходных данных из процесса ARIMA, наш случай заведомо более простой, чем приходится рассматривать в реальных задачах. Нет никакой гарантии того, что на практике автоматически выбранная модель будет обеспечивать такие же результаты, как и подогнанная вручную.

Собираясь постоянно использовать функцию `auto.arima()` или любой другой инструмент автоматического выбора модели в анализе данных временны*х* рядов, обязательно изучите документацию к ней, протестируйте ее на искусственных данных, а также изучите опыт работы с ней других специалистов по анализу данных. Существует несколько общеизвестных сценариев, в которых функции такого рода работают не так, как ожидалось, но опытным специалистам известны обходные решения подобных проблем. В целом функция `auto.arima()` является хорошим, но далеко не идеальном инструментом.[4] Подробное описание возможностей функции `auto.arima()` приведено в тематической главе онлайн-учебника, написанной разработчиком функции профессором Робом Хиндманом.

Выше продемонстрированы два разных способа настройки параметров модели: подгонка вручную методом Бокса-Дженкинса и автоматический выбор с помощью инструментов пакета `forecast`. Здесь мнения практикующих специалистов по анализу данных кардинально разделились: часть из них твердо убеждена в истинности ручную пошагового процесса, а остальные не менее яростно отстаивают автоматические инструменты настройки модели. Дискуссия все еще не завершена! В долгосрочной перспективе, вследствие стремительного роста популярности больших данных, инструменты автоматического исследования и подгонки моделей будут играть главную роль в анализе временны*х* рядов больших наборов данных.

Векторная авторегрессия

В реальном мире часто приходится иметь дело сразу с несколькими параллельными временны*ми* рядами, предположительно связанными друг с другом. Ранее было показано, как правильно очищать и выравнивать параллельные данные, а в

[4] См. примеры в блоге Роба Хиндмана (`https://perma.cc/9DH6-LGNW`) и на сайте Stack Overflow (`https://perma.cc/2KM3-Z4R4`).

этом разделе мы рассмотрим принципы использования и область применения таких временны*х* рядов. Для этого нам понадобится сгенерировать модель AR(p) сразу для нескольких переменных. Такая модель должна учитывать факт влияния каждой из переменных на остальные и, в свою очередь, испытывать влияние со стороны остальных переменных — в модели отсутствует привилегированная величина y, выделяемая из остальных величин x. Напротив, подгонка модели выполняется на основе полностью симметричных данных. Обратите внимание на то, что в случае нестационарных данных анализ выполняется для разностей, а не исходных значений, как в рассматриваемых ранее моделях.

Экзогенные и эндогенные величины

С точки зрения статистических исследований переменные модели, которые оказывают влияние друг на друга, называются *эндогенными* — их значения описываются поведением модели. В противоположность им *экзогенные* переменные не находят объяснения в модели — они не описываются предположениями модели, а потому их значения принимаются такими, какими они есть, и не подвержены динамическим изменениям.

Поскольку каждый временной ряд предположительно предсказывает все остальные, а также себя, каждая переменная будет описываться одним уравнением. Пусть у нас есть три временны*х* ряда — обозначим их значения в момент времени t как $y_{1,t}$, $y_{2,t}$ и $y_{3,t}$. Тогда уравнения векторной авторегрессии (VAR) порядка 2 (с учетом двух временны*х* сдвигов) можно записать следующим образом:

$$y_{1,t} = \phi_{01} + \phi_{11,1}y_{1,t-1} + \phi_{12,1}y_{2,t-1} + \phi_{13,1}y_{3,t-1} + \phi_{11,2}y_{1,t-2} + \phi_{12,2}y_{2,t-2} + \phi_{13,1}y_{3,t-2};$$
$$y_{2,t} = \phi_{02} + \phi_{21,1}y_{1,t-1} + \phi_{22,1}y_{2,t-1} + \phi_{23,1}y_{3,t-1} + \phi_{21,2}y_{1,t-2} + \phi_{22,2}y_{2,t-2} + \phi_{23,1}y_{3,t-2};$$
$$y_{3,t} = \phi_{03} + \phi_{31,1}y_{1,t-1} + \phi_{32,1}y_{2,t-1} + \phi_{33,1}y_{3,t-1} + \phi_{31,2}y_{1,t-2} + \phi_{32,2}y_{2,t-2} + \phi_{33,2}y_{3,t-2}.$$

Умножение матриц

Обладая некоторыми познаниями в линейной алгебре, вы легко заметите, что предыдущие три уравнения намного проще выразить в матричной форме записи. Поступив так, вы представите модель VAR в похожем на модель AR виде. В матричной форме записи уравнения модели VAR имеют вид

$$y = \phi_0 + \phi_1 y_{t-1} + \phi_2 y_{t-2},$$

где y и ϕ_0 — матрицы размером 3×1, а остальные ϕ — матрицы размером 3×3.

Даже в таком простом примере видно, что количество параметров модели возрастет очень быстро. Например, в модели с p сдвигами и N переменными

каждая переменная может принимать $1 + pN$ значений. Располагая N значениями для прогнозирования, мы получим $N + pN^2$ переменных — число переменных растет со скоростью $O(N^2)$ относительно количества исследуемых временны*х* рядов. Следовательно, использовать такую модель стоит не для анализа любых данных, представленных в виде временны*х* рядов, а только в случаях, когда есть веские основания предполагать, что между временны*ми* рядами существуют некие взаимосвязи.

Модели VAR чаще всего применяются в эконометрике. Они часто подвергаются критике за отсутствие строгого обоснования, за исключением гипотезы о взаимном влиянии исследуемых величин. Именно по этой причине очень сложно оценить точность модели. Тем не менее модели VAR все еще востребованы, например, при оценке влияния переменных друг на друга. Их часто применяют для прогнозирования поведения переменных, когда знания предметной области не позволяют выявить тип существующих взаимоотношений. Иногда они также используются для определения того, насколько сильно расхождения в прогнозах обусловливаются "естественными" причинами.

Рассмотрим простой пример. Располагая известными данными об идентификационных номерах клиентов (UCI), спрогнозируем количество банковских переводов (2) не только по их собственным значениям, но и по значениям второго столбца набора данных (обратите внимание: прогнозирование второго столбца набора данных выполняется ввиду предположения о симметричности исследуемых величин). Сведения о переводах извлекаются для сектора управления дорожным движением. Тем самым обеспечивается относительная независимость источника информации по сравнению с предыдущим случаем, когда в обработку принимались данные о переводах для фискального сектора. Также предположим, что в каждом столбце содержатся все необходимые сведения о проводимых платежах, позволяющие определить направление будущих изменений.

Для определения используемых параметров обратимся к инструментам пакета `vars`, в частности к его методу `VARselect()`.

```
## R
> VARselect(demand[, 11:12, with = FALSE], lag.max=4,
            +           type="const")

$selection
AIC(n) HQ(n) SC(n) FPE(n)
   3     3     1      3

$criteria

              1           2           3           4
AIC(n) 3.975854e+01 3.967373e+01 3.957496e+01 3.968281e+01
HQ(n)  3.984267e+01 3.981395e+01 3.977126e+01 3.993521e+01
```

```
SC(n)  3.997554e+01 4.003540e+01 4.008130e+01 4.033382e+01
FPE(n) 1.849280e+17 1.700189e+17 1.542863e+17 1.723729e+17
```

Как видим, функция предоставляет широкий выбор информационных критериев. Обратите внимание на использование атрибута `const`, обеспечивающего учет компонента ненулевого математического ожидания. Вместо него можно было задать дрейфовый компонент, их оба или ни одного, хотя вариант "`const`" кажется наиболее подходящим для наших данных. Начнем подгонку модели с учета трех сдвигов и посмотрим, какой результат будет получен.

```
## R
> est.var <- VAR(demand[, 11:12, with = FALSE], p=3, type="const")
> est.var

> par(mfrow = c(2, 1))
> plot(demand$'Banking orders (2)', type = "l")
> lines(fitted(est.var)[, 1], col = 2)
> plot(demand$'Banking orders (3)',
>      type = "l")
> lines(fitted(est.var)[, 2], col = 2)
> par(mfrow = c(2, 1))
> acf(demand$'Banking orders (2)' - fitted(est.var)[, 1])
> acf(demand$'Banking orders (3)' -
>     fitted(est.var)[, 2])
```

Результатом выполнения кода будут графики, показанные на рис. 6.12 и 6.13.

Функции ACF недостаточно четко указывают на отсутствие автокорреляции в невязках, как того хотелось бы, поэтому дополнительно проведем *тест портманто* (Portmanteau test) для определения последовательной корреляции, воспользовавшись методом `vars` пакета `serial.test()`. Этот тест выполняется так же, как и тест на последовательную корреляцию в одномерных данных, с которым вам уже доводилось встречаться.

```
## R
> serial.test(est.var, lags.pt = 8, type="PT.asymptotic")

Portmanteau Test (asymptotic)

data: Residuals of VAR object est.var
Chi-squared = 20.463, df = 20, p-value = 0.4293
```

Столь высокое значение p не позволяет отвергнуть нулевую гипотезу об отсутствии последовательной корреляции в невязках. Это еще раз доказывает состоятельность модели в описании данных.

В ходе анализа одномерных данных мы изучили несколько моделей, пока не пришли к ARMA и ARIMA. Справедливым будет вопрос о существовании модели VARIMA для многомерных временных рядов. Если она существует, то такая модель вряд ли найдет широкое применение ввиду высокой точности и сложности модели VAR. Стоит заметить, что на практике — в подавляющем большинстве производственных и научных решений — применяется модель VAR, а не VARIMA.

Родственным классом векторной авторегрессионной модели выступает CVAR (Cointegrated Vector Autoregression), или модель коинтегрированной векторной авторегрессии. Она находит применение в случаях, когда нестационарными являются отдельные временные ряды, но линейная их комбинация остается стационарной для исходных значений, а не только разностей.

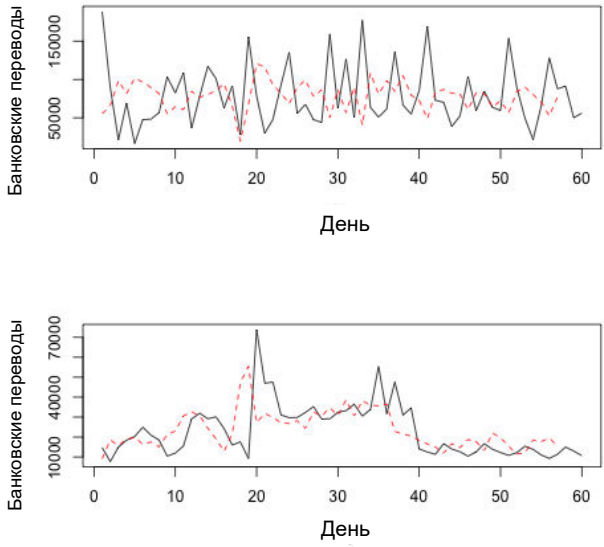


Рис. 6.12. *Вверху показаны фактические (сплошная линия) и прогнозные (пунктирная линия) данные о количестве банковских переводов (2), а внизу приведены такие же графики для банковских переводов (3). По всей видимости, верхний график демонстрирует более точный прогноз, поскольку в нем "приглушенная" кривая прогнозных значений довольно плавно повторяет кривую реальных данных, в то время как на нижнем графике прогнозные данные предсказывают события несколько раньше, чем они происходят. Это говорит о том, что ряд банковских переводов (2) опережает ряд банковских переводов (3). Следовательно, ряд банковских переводов (2) можно использовать для прогнозирования переводов, производимых сектором управления дорожным движением, но не наоборот — по крайней мере не с такой высокой точностью*

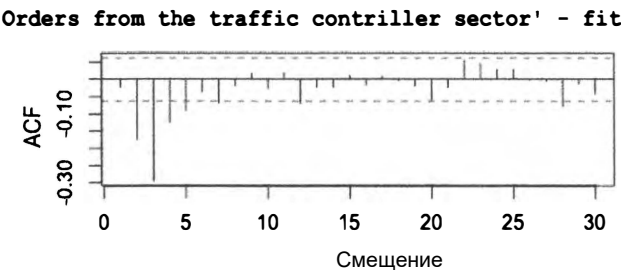


Рис. 6.13. *Автокорреляционные функции невязок обоих временных рядов. В каждом временном ряду наблюдается значительная автокорреляция ошибок в смещении 3, которую невозможно полностью учесть в модели*

Виды статистических моделей

Существует много других видов статистических моделей, предназначенных для описания временных рядов. Некоторые из них расширяют модель ARIMA, в то время как в остальных принимаются другие базовые предположения о временной динамике, не такие, как в ARIMA. В этом разделе мы кратко обсудим некоторые из наиболее часто используемых и известных статистических моделей, применяемых для анализа временных рядов.

Сезонная модель ARIMA

Сезонная модель ARIMA (SARIMA) предполагает мультипликативную сезонность. Исходя из такого определения модель SARIMA можно описать как ARIMA(p, d, q)(P, D, Q)m. В ней постулируется, что сезонное поведение можно рассматривать как процесс ARIMA, где m указывает количество временных шагов в сезонном цикле. Важно знать, что модель SARIMA предполагает, что данные в смежных временных точках оказывают влияние друг на друга либо в пределах одного сезона, либо в разные сезоны, и такое поведение можно описать отдельно с помощью таких же методов, которые применяются в модели ARIMA.

Подгонка модели SARIMA выполняется несколько сложнее, чем модели ARIMA именно из-за необходимости учета сезонных изменений. К счастью, функция `auto.arima()` пакете `forecast` справляется с этой задачей так же просто, как со стандартной задачей подбора модели ARIMA. Как указывалось ранее, не обладая достаточными познаниями, лучше обратиться к автоматическому способу выбора параметров модели. Ручной подгонкой автоматически подобранной модели можно заниматься только в случае глубокого понимания всех тонкостей процесса.

ARCH, GARCH и родственные им модели

Под аббревиатурой "ARCH" скрывается название "Autoregressive Conditional Heterosckedasticity" (Авторегрессионная условная гетероскедастичность). За редким исключением эта модель используется только в финансовой отрасли. Но она часто описывается в руководствах по анализу временны*х* рядов, поэтому заслуживает отдельного упоминания. Этот отдельный класс моделей происходит из наблюдений, утверждающих, что цены на акции не характеризуются постоянной дисперсией, а сама дисперсия оказывается авторегрессионной по отношению к более ранним дисперсиям (например, дни высокой волатильности на фондовой бирже следуют кластерами). В таких моделях авторегрессионным процессом выступает дисперсия, а не сами исходные данные.

Модели иерархических временны*х* рядов

Иерархические временны*е* ряды широко распространены в реальном мире, хотя и не представляются в явном виде как таковые. Можно легко представить ситуации, при описании которых они будут востребованы.

- Общий месячный спрос на продукцию компании в денежном выражении, впоследствии пересчитываемый в единицах складского учета (SKU).
- Еженедельные данные об общих электоральных предпочтениях населения с разбивкой по демографическим категориям (перекрывающимся или не перекрывающимся), например женщины в сравнении с мужчинами или латиноамериканцы в сравнении с афроамериканцами.
- Общее количество туристов, прибывающих в ЕС ежедневно, по сравнению с количеством туристов, прибывающих отдельно в каждую страну — член ЕС.

Одним из самых удобных инструментов обработки иерархических временны*х* рядов считается пакет `hts` языка R. Он может применяться как для визуализации данных иерархических временны*х* рядов, так и для прогнозирования.

Прогнозы создаются на основе самых разных методик, поддержка которых исторически была включена в этот пакет.

- После получения прогнозов самого низкого уровня (наиболее индивидуализированные) они обобщаются для создания прогнозов более высокого уровня.
- После получения прогнозов самого высокого уровня составляются прогнозы более низкого уровня, для чего рассчитывается долевое участие каждого из них в общей совокупности. Эта методика обычно менее точная при низкоуровневом прогнозировании, хотя ее можно несколько улучшить, спрогнозировав изменение пропорций совокупности во времени.
- Придерживаясь "среднего" прогноза (в моделях с многоуровневой иерархией), получить лучшие прогнозы для внутренних уровней иерархии с помощью оптимальных методик. Для получения остальных прогнозов опускаться или подниматься по иерархии относительно таких точных прогнозов, задействовав наиболее подходящие модели.

По большому счету, золотой стандарт использования пакета `hts` заключается в построении прогнозов для любых уровней иерархии, независимо от остальных ее уровней, с последующим обобщением для согласования метода, описанного Хиндманом (`https://perma.cc/G4EG-6SMP`).

Многие из рассматриваемых выше статистических моделей часто применяются в задачах прогнозирования временны*х* рядов самых разных уровней. Анализ иерархических взаимоотношений между отдельными составляющими процесса обычно выносится в отдельный этап моделирования.

Преимущества и недостатки статистических методов анализа временны*х* рядов

Принимая решение об использовании одной из описанных выше статистических моделей для анализа временны*х* рядов, обращайте внимание на ее преимущества и недостатки.

Преимущества

- Простота и прозрачность модели предполагают использование понятных параметров.
- Описание модели простыми математическими выражениями позволяет рассматривать ее свойства с помощью предельно строгих статистических методов.
- Применяя статистические модели к относительно небольшим наборам данных, можно получить такие же хорошие результаты, как и в случае больших данных.

- Простые модели и их модификации работают очень хорошо даже в сравнении со сложными моделями, применяемыми в машинном обучении. Таким образом, вы получаете хорошую точность без опасности переобучения.
- Правильно подобранные автоматизированные методики определения порядка модели и оценки их параметров существенно упрощают прогнозирование.

Недостатки

- Относительная простота модели не позволяет ей иметь высокую точность при обработке больших наборов данных. Работая с очень большими наборами данных, лучше обратиться к сложным моделям машинного обучения и алгоритмам нейронных сетей.
- Статистические модели "заточены" на оценку математического ожидания распределения, а не самого распределении. Конечно, они позволяют рассчитать дисперсию выборки и другие статистические показатели, указывающие на неопределенность прогнозов, но на базовом уровне модель располагает лишь ограниченными возможностями по выявлению неопределенности во всех комбинациях параметров, которые рассматриваются при выборе модели.
- По определению такие модели не созданы для описания динамики нелинейных процессов и плохо подходят для анализа данных с преимущественно нелинейными зависимостями.

Дополнительные источники

- Классические учебники

  Rob J. Hyndman and George Athanasopoulos, Forecasting: Principles and Practice, 2nd ed. (Melbourne: OTexts, 2018), `https://perma.cc/9JNK-K6US`

  Очень практичный и доступный для бесплатной загрузки учебник, в котором рассматриваются основы предварительной обработки данных временных рядов и использования таких данных для построения прогнозов на языке R. Акцент делается на быстром обучении наиболее эффективным методам прогнозирования временных рядов для решения практических задач.

  Ruey Tsay, Analysis of Financial Time Series (Hoboken, NJ: John Wiley & Sons, 2001)

  В этом классическом учебнике описаны различные модели временных рядов (отдельная глава посвящена разработке моделей AR, MA

и ARIMA с приложениями к историческим биржевым котировкам). В нем рассматриваются сложные примеры моделирования, выполняемого на языке R. Книга включает материал средней сложности относительно рассматриваемых статистических и общематематических понятий, но вполне подойдет для всех, кто обладает начальными познаниями в дисциплинах математического анализа и статистики.

Robert H. Shumway, Time Series Analysis and Its Applications (NY, NY: Springer-International, 2017)

Еще один классический учебник, написанный на математическом языке и включающий материал повышенной сложности. Перед его изучением лучше познакомиться с работами Цая (Tsay) — эта книга включает расширенное обсуждение теоретических принципов построения статистических моделей временны*х* рядов. В ней также рассматривается широкий круг практических задач по анализу данных, основанных на некоторых научных, но чаще финансовых источниках данных. Этот учебник намного сложнее для изучения, чем упомянутые выше, но не из-за информативной насыщенности, а не потому, что требует от читателей более высокого уровня теоретической подготовки (на самом деле это не так, но такое впечатление складывается при беглом изучении).

- Общие руководства

Robert Nau, "Summary of Rules for Identifying ARIMA Models," course notes from Fuqua School of Business, Duke University, `https://perma.cc/37BY-9RAZ`

Подробные рекомендации по выбору трех основных параметров модели ARIMA. Знакомясь с ними, обратите внимание на нарочитую простоту описываемых моделей.

National Insitute of Standards and Technology (NIST), "Box-Jenkins Models," in NIST/SEMATECH e-Handbook of Statistical Methods (Washington, DC: NIST, US Department of Commerce, 2003), `https://perma.cc/3XSC-Y7AG`

В этом разделе интерактивного справочника NIST приведены пошаговые инструкции по реализации метода Бокса-Дженкинса, широко используемого при подгонке параметров модели ARIMA. В нем описаны чрезвычайно полезные источники данных, предоставляемые Национальным институтом стандартов и технологии (NIST) для обучения принципам статистического анализа временны*х* рядов как важной составляющей общей методологии знакомства с передовыми технологиями обработки данных.

Rob J. Hyndman, "The ARIMAX Model Muddle," Hyndsight blog, October 4, 2010, `https://perma.cc/4W44-RQZB`

Краткий пост в блоге известного специалиста по прогнозированию Роба Хиндмана, в котором он рассматривает возможность включения независимых переменных в модель ARIMA и альтернативные методики построения моделей VAR, предназначенных для обработки многомерных временных рядов.

Richard Hardy, "Cross Validation: Regularization for ARIMA Models," question posted on Cross Validated, StackExchange, May 13, 2015, `https://perma.cc/G8NQ-RCCU`

В авторегрессионных процессах высокого порядка и в случаях анализа многомерных величин с помощью VAR часто прибегают к предварительному упорядочению данных, что позволяет существенно повысить точность моделирования. Этот пост на сайте Cross Validated Q&A содержит некоторые общие замечания по рассматриваемому вопросу, ссылки на примеры реализации и результаты научных исследований по смежным темам.

ГЛАВА 7

Модели пространства состояний для временны*х* рядов

Модели пространства состояний аналогичны статистическим моделям, которые рассматривались в предыдущей главе, но имеют *б*ольшую "практическую" ценность. Они применяются при решении таких примечательных инженерных задач, как учет погрешности измерений и получение оценок на основе априорных знаний или убеждений.

Рассматриваемые здесь модели определяют системы, в которых истинное состояние не может быть измерено напрямую, но может быть выведено из результатов измерений. Модели пространства состояний основываются на знаниях о динамическом поведении системы, например о временно*м* развитии истинного состояния системы, определяемом внутренними процессами и внешними воздействиями.

Ранее вам, скорее всего, не доводилось математически рассчитывать модели пространства, хотя в повседневной жизни они встречаются повсеместно. Например, понаблюдаем за водителем автомобиля, "виляющим" в дорожном потоке. Постараемся определить его возможные маневры, чтобы избежать попадания с ним в ДТП. Если водитель находится под воздействием алкоголя, то вам следует обратиться в полицию, но если в его поведении не наблюдается повторяющихся шаблонов, то вам придется реагировать на возможные угрозы самостоятельно. В следующие несколько секунд или минут вам нужно обновлять собственную модель пространства для этого водителя, чтобы понять, как правильно поступать в будущем.

Классический пример ситуации, описываемой моделью пространства состояний, — задача о запускаемой в космос ракете. Нам известны законы Ньютона, применяемые для описания динамики системы, и мы можем отследить движение ракеты в любой момент времени. Нам также известно, что GPS-датчики и другие инструменты отслеживания местоположения ракеты имеют некоторую погрешность измерения, которую можно определять количественно и нужно обязательно учитывать в выполняемых расчетах. Наконец, нужно понимать, что практически невозможно учесть все действующие на нашу ракету силы, поскольку система характеризуется большим количеством неизвестных параметров — нам

нужно обеспечить ее устойчивость к самым разным источникам воздействия, таким как, например, солнечный ветер. Как оказалось, достижения в статистических и инженерных дисциплинах, накопленные за последние 50 лет, позволяют решать подобные задачи достаточно легко.

К разработке моделей пространства состояний и возникновению интереса к решаемым с их помощью задачам привели два разных исторических свершения. Во-первых, примерно в середине XX века мы вступили в эпоху механистической автоматизации. Человечеством были созданы ракеты и космические корабли, навигационные системы для подводных лодок и множество других средств автоматизации, работа которых основана на оценке состояния системы, которое не поддается измерению. Тогда исследователи задумались об оценке состояния системы с помощью специально разработанных инструментов анализа в пространстве состояний, позволяющих минимизировать ошибки измерения и другие виды неопределенности. В результате появились первые методы (анализа) пространства состояний.

Во-вторых, параллельно с методами пространства состояний развивались технологии ведения учетной документации и соответствующих вычислительных алгоритмов. Все это привело к получению гораздо более крупных наборов данных, используемых в качестве источников временны*х* рядов, чем те, которые рассматривались нами ранее. Извлекаемые из них временны*е* ряды имеют несравнимо большую длину, или плотность, временны*х* точек. С увеличением количества доступных для изучения наборов данных возникла необходимость в разработке более информационно емких инструментов их обработки, предполагающих моделирование пространства состояний.

В этой главе мы рассмотрим наиболее часто используемые методы пространства состояний.

- Фильтр Калмана для линейной гауссовой модели
- Скрытые марковские модели
- Байесовский структурный временной ряд

Рассматриваемые методы хорошо описаны, доступны для реализации и обладают строго очерченной областью применения. Для каждого из них мы выработаем интуитивно понятную математическую модель и обсудим данные, подходящие для обработки с помощью того или иного метода. Наконец, мы приведем примеры кодов реализации методов пространства состояний.

В каждом случае важно научиться сопоставлять наблюдения с состояниями, в которых были проведены такие наблюдения. Оценивая основное состояние по наблюдениям, мы можем выделить следующие исследовательские процессы и этапы.

Фильтрация

Использование измерения в момент времени t для обновления оценки состояния в момент времени t.

Прогнозирование

Использование измерения в момент времени $t - 1$ для составления прогноза ожидаемого состояния в момент времени t (что позволяет также спрогнозировать измерение в момент времени t).

Сглаживание

Использование измерения в течение определенного периода времени, который включает в себя момент t, а также периоды до и после него, для оценки того, каково было истинное состояние в момент времени t.

Механика этих операций часто схожа, чего не скажешь о получаемых результатах. Фильтрация — это способ сопоставления последней информации с предыдущими данными при обновлении оценки состояния. Прогнозирование — это предсказание возможного состояния без получения сведений о будущем. Сглаживание — это использование информации о будущем и прошлом для получения наилучшей оценки состояния в данный момент времени.

Модели пространства состояний: преимущества и недостатки

Модели пространства состояний могут использоваться как в детерминированных, так и в стохастических приложениях и применяться как к непрерывным, так и к дискретным данным.[1]

Уже одно это дает некоторое представление об их полезности и невероятной гибкости. Гибкость моделей пространства состояний предопределяет как достоинства, так и недостатки этого класса моделей.

У моделей пространства состояний много сильных сторон. Они позволяют моделировать наиболее интересные данные временны*х* рядов: динамические процессы и состояния, которые генерируют анализируемые данные с шумом, а не одни только данные с шумом. С помощью модели пространства состояний вводится модель *причинности* в процесс моделирования, чтобы объяснить, что, в первую очередь, порождает процесс. Такой подход оказывается оправданным в случаях существования обоснованных теорий или надежных знаний о работе

[1] В этой книге анализируются дискретные данные, которые чаще всего рассматриваются в реальных приложениях.

системы либо когда необходимо использовать модель для детального исследования общей динамики известной системы.

Модель пространства состояний позволяет изменять коэффициенты и параметры во времени, т.е. определять поведение системы во времени. Заметьте, что при использовании моделей пространства состояний условие стационарности на данные не накладывается. Это сильно отличается от ситуаций, рассматриваемых в главе 6, в которых предполагалось, что устойчивый процесс моделируется только одним набором не изменяющихся во времени коэффициентов.

Тем не менее у модели пространства состояний есть и ряд недостатков, а иногда они даже рассматриваются в качестве сильной составляющей модели.

- Поскольку модели пространства состояний очень гибкие, существует множество параметров, которые можно установить, и многие формы, которые они могут принимать. Это означает, что свойства конкретной модели пространства состояний часто оказываются недостаточно изученными. При построении модели пространства состояний, адаптированной к исследуемым данным временны*х* рядов, вы вряд ли найдете учебники по статистике или научные статьи, в которых она уже рассматривалась. Таким образом, вы оказываетесь в менее определенной ситуации, пытаясь понять, как работает модель, или определить, где были совершены ошибки.
- Модели пространства состояний могут быть очень сложными в вычислительном отношении, поскольку включают много параметров. Кроме того, слишком большое количество параметров для некоторых типов моделей пространства состояний может сделать вас зависимым от переобучения, особенно при недостаточности данных.

Фильтр Калмана

Фильтр Калмана — это хорошо исследованный и популярный метод для включения новой информации из временно*г*о ряда и ее разумного объединения с ранее известной информацией для оценки основного состояния. Одно из первых применений фильтра Калмана произошло во время миссии Apollo 11: когда инженеры НАСА поняли, что встроенные вычислительные элементы не позволят использовать другие, более ресурсоемкие методы оценки положения, они выбрали этот фильтр. Как вы увидите в этом разделе, преимущества фильтра Калмана заключаются в том, что его относительно легко вычислить, и он не требует хранения прежних данных для составления текущих оценок или будущих прогнозов.

Обзор

Вычислительные методики, применяемые для описания фильтра Калмана, способны озадачить начинающих специалистов по обработке данных — не только исходя из их высокой сложности, но и потому, что в них приходится отслеживать значительное количество величин — это итеративный, отчасти замкнутый процесс со многими взаимосвязанными величинами. По этой причине здесь мы не будем выводить уравнения фильтра Калмана, а всего лишь приведем общее их описание, чтобы понять, как они работают.[2]

Начнем с изучения линейной гауссовой модели, в которой утверждается, что состояние и наблюдения имеют следующую динамику:

$$x_t = Fx_{t-1} + Bu_t + w_t,$$

$$y_t = Ax_t + v_t.$$

Как видите, состояние в момент времени t является функцией состояния на предыдущем временном шаге (Fx_{t-1}), внешнего воздействия (Bu_t) и стохастической составляющей (w_t). Аналогично измерение в момент времени t является функцией состояния в момент времени t и члена стохастической ошибки, т.е. ошибки измерения.

Давайте представим, что x_t — это реальное положение космического корабля, а y_t — положение, измеряемое с помощью некого измерительного устройства (датчика). Пусть v_t — ошибка измерения такого датчика. Тогда основное уравнение фильтра Калмана будет показывать обновленную оценку с учетом новой информации для времени t:

$$\hat{x}_t = K_t y_t + (1 - K_t)\hat{x}_{t-1}.$$

Здесь мы переходим к этапу фильтрации, на котором принимается решение о том, как измерение в момент времени t влияет на обновленную оценку состояния в момент времени t. Не забывайте, что мы рассматриваем ситуацию, в которой выступаем простыми наблюдателями, и, делая выводы о состоянии, не можем быть в них уверены. Выше показано, что величина K_t задает баланс между старой информацией ($\hat{x}_{t-1}$) и новой информацией (y_t) нашей оценки.

Чтобы перейти к более подробному описанию, нам нужно определиться с используемой терминологией. Величина P_t обозначает оценки ковариации нашего состояния (это может быть скаляр или матрица, в зависимости от размерности состояния — многомерные состояния более распространены). P_t^- — это оценка для t до учета измерения в момент времени t.

[2] Настоятельно рекомендуется познакомиться с многочисленными альтернативными описаниями фильтра Калмана на сайте Mathematics StockExchange (`https://perma.cc/27RK-YQ52`).

Кроме того, величиной R мы будем определять дисперсию ошибки измерения, т.е. дисперсию v_t, которая также может представляться либо скаляром, либо ковариационной матрицей в зависимости от размерности измерений. Как правило, в реальных системах параметр R хорошо известен, поскольку описывает известные физические свойства конкретного датчика или измерительного устройства. Соответствующее ему значение w_t, которое выражается через Q, определено менее точно и подлежит уточнению в процессе моделирования.

Начать исследование стоит с процесса, в котором известны или оцениваются значения x и P в момент времени 0. Продвигаясь вперед по временной шкале, мы будем последовательно чередовать фазы прогнозирования и обновления так, чтобы каждая следующая фаза прогнозирования предшествовала последующим этапам обновления/фильтрации и т.д.

Прогноз:

$$\hat{x}_t^- = F\hat{x}_{t-1} + Bu_t,$$
$$\hat{P}_t^- = FP_{t-1}F^T + Q.$$

Фильтрация:

$$\hat{x}_t = \hat{x}_t^- + K_t\left(y_t + A\hat{x}_t^-\right),$$
$$P_t = \left(I - K_tA\right)P_t^-,$$

где K_t, матрица коэффициентов усиления фильтра Калмана равна

$$K_t = P_t^-A^T\left(AP_t^-A^T + R\right)^{-1}.$$

Существует много вариантов визуализации такого рекурсивного процесса. Иногда его разбивают на множество этапов (чаще всего — на четыре или пять). Однако самый простой способ его описания состоит в получении прогнозных значений в момент времени t без измерения y_t (прогноз) и проведения вычислительных этапов для момента времени t уже после того, как измерение y_t станет известным (фильтрация).

Для выполнения таких действий нам понадобятся следующие значения.

- Оценки для R и Q — ковариационные матрицы значений ошибок измерения (легко вычисляются) и стохастичности состояния (обычно оцениваются) соответственно.
- Оценки или известные значения состояния $\hat{x}_0$ в момент времени 0 (оценивается по значению y_0).

- Представление о том, какие силы будут воздействовать на систему в момент времени t и как они повлияют на состояние, т.е. матрица B и значение u_t.
- Представление о динамике системы, определяющей переход состояний от одного временного шага к другому, а именно F.
- Понимание зависимости измерения от состояния системы, а именно H.

Положение

Существует много способов получения уравнений фильтра Калмана, в том числе с вероятностной точки зрения в терминах математического ожидания, минимизации наименьших квадратов или оценки максимального правдоподобия. Все они широко освещены в специализированной литературе, и вы легко найдете их, выполнив поиск на тематических сайтах в Интернете.

Код реализации фильтра Калмана

Рассмотрим классический вариант использования фильтра Калмана: попробуем отследить объект, подчиняющийся законам Ньютона, с помощью датчиков, передающих показания с некой ошибкой. Сгенерируем временные ряды, основываясь на ньютоновских законах движения тел, согласно которым положение объекта является функцией его скорости и ускорения. Несмотря на непрерывность физического процесса, движение будет отслеживаться по данным дискретных измерений. В самом начале создадим ряд со значениями ускорения, а затем предположим, что начальные положение и скорость объекта равны 0. Хотя это не совсем реалистично, будем предполагать, что ускорение изменяется мгновенно в начале каждого временного шага и остается постоянным в течение всей его длительности.

```
## R
## Ракета движется 100 временных шагов
ts.length <- 100

## Движение является ускоренным
a <- rep(0.5, ts.length)

## Начальное положение и скорость равны 0
x <- rep(0, ts.length)
v <- rep(0, ts.length)
for (ts in 2:ts.length) {
    x[ts] <- v[ts - 1] * 2 + x[ts - 1] + 1/2 * a[ts - 1] ^ 2
    x[ts] <- x[ts] + rno:rm(1, sd = 20) ## стохастическая компонента
    v[ts] <- v[ts - 1] + 2 * a[ts - 1]
}
```

Если вы не помните законы Ньютона, то обязательно повторите их, чтобы не принимать на веру все дальнейшие выкладки (в первую очередь, касающиеся вычисления значений `x[ts]` и `v[ts]`).

Динамические характеристики движения, которое описывается заданными ранее параметрами, показаны на графиках, изображенных на рис. 7.1.

```
## R
par(mfrow = c(3, 1))
plot(x,            main = "Положение",     type = 'l')
plot(v,            main = "Скорость",      type = 'l')
plot(acceleration, main = "Ускорение", type = 'l')
```

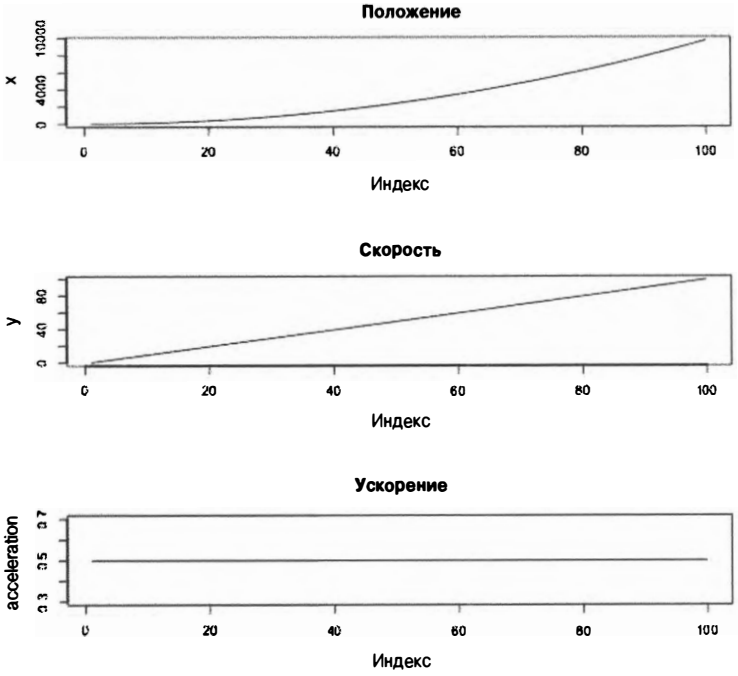


Рис. 7.1. *Положение, скорость и ускорение ракеты*

Предполагается, что переменные полностью описывают состояние, но единственные доступными данные — это сведения о положении объекта, и они поступают с зашумленного датчика. В следующем коде положение такого датчика задается переменной x, а измеренные значения соотносятся с фактическими данными о положении объекта так, как показано на рис. 7.2.

```
## R
z <- x + rnorm(ts.length, sd = 300)
plot(x, ylim = range(c(x, z)))
lines(z)
```

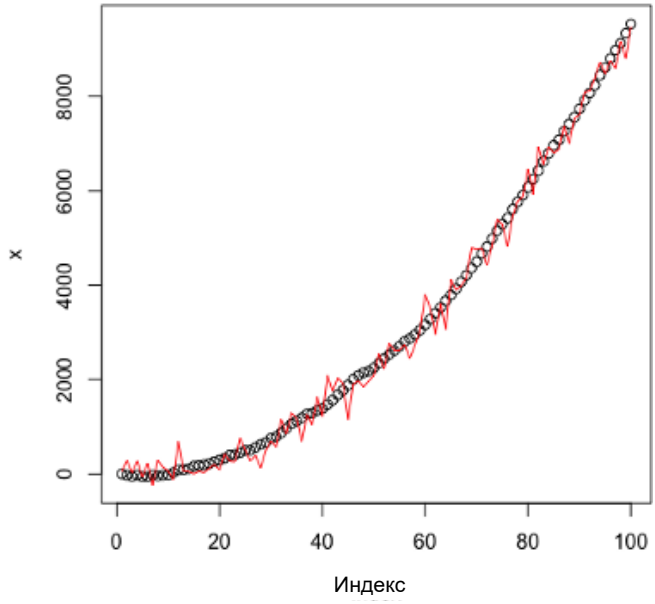


Рис. 7.2. *Действительное положение (точки) и зашумленные измерения (линии). Обратите внимание на то, что положение* x *не задается идеальной параболой из-за шума, привнесенного в уравнение состояния*

Как видно на рис. 7.1, движение характеризуется постоянным ускорением (нижний график), обеспечивающим линейное увеличение скорости (средний график) и изменение координаты по параболической траектории (верхний график). Если такое поведение объекта кажется вам непонятным, то либо примите его как должное, либо повторите курс механики в любом учебнике по физике.

Применим фильтр Калмана. Сначала напишем общую функцию, отражающую параметризацию системы и результаты, полученные ранее в этом разделе.

```
## R
kalman.motion <- function(z, Q, R, A, H) {
    dimState = dim(Q)[1]

    xhatminus <- array(rep(0, ts.length * dimState),
                       c(ts.length, dimState))
    xhat      <- array(rep(0, ts.length * dimState),
                       c(ts.length, dimState))

    Pminus <- array(rep(0, ts.length * dimState * dimState),
              c(ts.length, dimState, dimState))
    P      <- array(rep(0, ts.length * dimState * dimState),
              c(ts.length, dimState, dimState))
```

```
    K <- array(rep(0, ts.length * dimState),
          c(ts.length, dimState)) # коэффициент усиления Калмана

    # Начальное предположение о равенстве нулю всех параметров
    xhat[1, ] <- rep(0, dimState)
    P[1, , ] <- diag(dimState)

    for (k in 2:ts.length) {
       # Обновление времени
       xhatminus[k, ] <- A %*% matrix(xhat[k - 1, ])
       Pminus[k, , ] <- A %*% P[k - 1, , ] %*% t(A) + Q

       K[k, ] <- Pminus[k, , ] %*% H %*%
                             solve(t(H) %*% Pminus[k, , ] %*% H + R)
       xhat[k, ] <- xhatminus[k, ] + K[k, ] %*%
                             (z[k]- t(H) %*% xhatminus[k, ])
       P[k, , ] <- (diag(dimState)-K[k,] %*% t(H)) %*% Pminus[k, , ]
    }

    ## Возвращение прогноза и сглаженного значения
    return(list(xhat = xhat, xhatminus = xhatminus))
}
```

Теперь применим эту функцию для определения одного только положения (но не ускорения и скорости) ракеты.

```
## R
## Параметры шума
R <- 10^2 ## Дисперсия измерений. Предопределяется физическими
          ## ограничениями, накладываемыми на измерительное
          ## оборудование. Согласуется с шумом, добавленным к x,
          ## в представленном выше коде
Q <- 10   ## Дисперсия процесса. Обычно это гиперпараметр,
          ## обеспечивающий наибольшую точность

## Динамические параметры
A <- matrix(1) ## x_t = A * x_t-1 (как предыдущее значение x влияет
                                   на последующее значение x)
H <- matrix(1) ## y_t = H * x_t   (перевод состояния в измерение)
## Прогонка данных через фильтр Калмана
xhat <- kalman.motion(z, diag(1) * Q, R, A, H)[[1]]
```

Представим реальные, измеренные и прогнозируемые положения на общем графике (рис. 7.3).

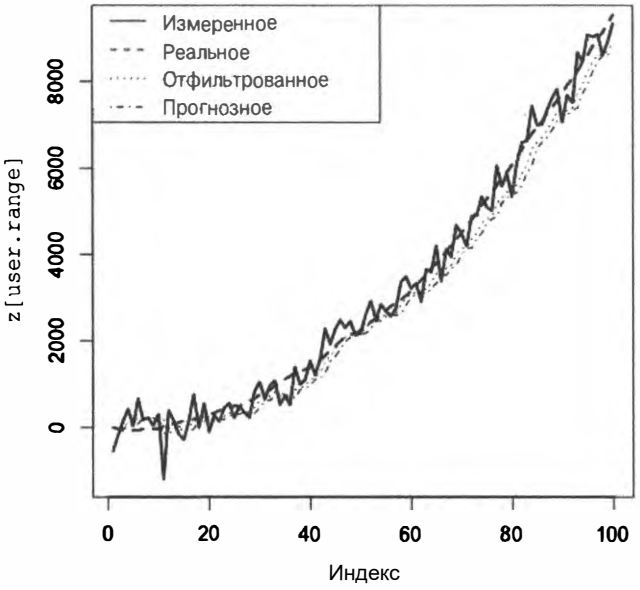


Рис. 7.3. *Основные параметры: измеренное, реальное/истинное положение, а также отфильтрованная оценка (наилучшая оценка в момент t с учетом измерений) и прогноз (наилучшая оценка в момент t с учетом заданной динамики системы и измерений, выполненных до момента времени t – 1, исключая момент t) положения*

Фильтр Калмана удаляет большую часть шума из ошибки измерения. Насколько много — зависит от значения *R*, параметра шума измерения, который отражает способность фильтра взвешивать последнее значение по сравнению с более ранними значениями. Как видим, фильтр вполне удовлетворительно справляется с прогнозированием данных. В частности, между прогнозными и фактическими данными не наблюдается запаздывание, а это указывает, что прогноз текущего значения строится на основе последнего значения.

Мы рассмотрели самый простой пример фильтра Калмана. Он хорошо исследован и находит широкое применение в различных приложениях, особенно в системах с неплохо изученной внутренней динамикой. Это делает его идеальным инструментом для решения задач, подобных простому движению ракеты, с достаточно точно изученными процессами изменения системы.

Обратите внимание на то, что в этом простом примере возможности и преимущества фильтра Калмана раскрыты не полностью. В общем случае он оказывается полезным при решении задач с несколькими типами измерений — разных величин или параллельных измерений одного и того же показателя несколькими устройствами. Существует множество вариантов фильтра Калмана, имеющих большое прикладное значение в самых разных дисциплинах и областях знаний.

Как было показано выше, одно из главных преимуществ фильтра Калмана заключается в его рекурсивности. Это устраняет необходимость в просмотре всех предыдущих точек данных в каждой итерации процесса. Скорее, на каждом временном шаге вся информация из предыдущих временных шагов оптимальным образом включается в несколько оценочных параметров, а именно — в последние оценки состояния и ковариации. К достоинствам метода стоит отнести взвешенное обновление данных, когда сводные статистические значения исследуемых показателей оптимально сопоставляются только с последними данными. Такой подход делает фильтр Калмана наиболее востребованным в приложениях, где наибольшую ценность представляют вычислительная производительность и скорость обработки данных. Во многих случаях он прекрасно описывает динамику реальных систем, представляемых с помощью марковских процессов (хранящими сведения только о предыдущих состояниях) и функции базового состояния, которое может быть измерено только с некоторой ошибкой.

Существует много полезных модификаций рассмотренного выше фильтра Калмана. Один из наиболее распространенных его вариантов отличается адаптацией к сглаживанию — возможностью получить наилучшую оценку истинного состояния в момент времени t на основе данных как до, так и после времени t. Математические уравнения и код реализации такого подхода во многом схожи с представленными выше. Еще один вариант — расширенный фильтр Калмана (Extended Kalman Filter — EKF), в котором фильтр Калмана адаптируется к данным процессов с нелинейной динамикой. Его достаточно просто реализовать благодаря включению специальных инструментов в различные пакеты языков R и Python.

Сложность фильтра Калмана составляет $O(T)$ относительно длины временного ряда и $O(d^2)$ — относительно d, размерности состояния. Это указывает на то, что нужно отказываться от переопределения состояния в случаях, когда более упорядоченная спецификация уже работает хорошо. Однако именно линейность по отношению к длине временных рядов делает фильтр Калмана широко используемым в реальных производственных задачах и гораздо более популярным, чем другие фильтры, предназначенные для моделирования пространств состояний временных рядов.

Скрытые марковские модели

Скрытые марковские модели (Hidden Markov Models — HMM) — это особенно полезное и интересное средство моделирования временных рядов, представляющее в анализе временных рядов редкий случай обучения без учителя, в котором обучение ведется в отсутствие обозначенного правильного ответа. Модели HMM обосновываются исходя из интуитивных соображений, подобных используемым при описании фильтра Калмана ранее в этой главе, — предположении о том, что

наблюдаемые переменные могут быть не самыми информативными для рассматриваемой системы. Как и в случае фильтра Калмана, примененного к линейной гауссовой модели, будем полагать, что процесс имеет состояния, а наблюдения предоставляют информацию о таких состояниях. Как и прежде, нам нужно получить представление о том, как переменные состояния связаны с проведенными наблюдениями. В случае применения HMM утверждается, что процесс нелинейный и характеризуется скачками между дискретными состояниями.

Обзор

Модель HMM описывает систему с непосредственно не наблюдаемыми состояниями. Система описывается марковским процессом, не хранящим сведений обо всех предыдущих состояниях — при расчете вероятности будущих событий учитываются только текущие состояния системы. Таким образом, сведения о предыдущих состояниях системы оказываются менее полезными, чем знание текущего состояния системы.

Марковские процессы часто описываются в матричном изложении. Например, рассмотрим систему, которая характеризуется состояниями *A* и *B*, между которыми возможны переходы. В любом из состояний система с большей статистической вероятностью будет оставаться в прежнем состоянии, а не переходить в другое состояние на любом отдельном временном шаге. Такая система может описываться следующей матрицей вероятностей.

	A	*B*
A	0,7	0,3
B	0,2	0,8

Представим, что наша система находится в состоянии *A*, а именно — (1, 0). (Состояние *B* описывается как (0, 1).) В таком случае вероятность того, что система останется в состоянии *A*, составляет 0,7, тогда как вероятность изменения состояния равна 0,3. Заметьте, что здесь совершенно не важно, в каких состояниях система находилась до последнего момента времени. Именно такие процессы называются марковскими.

Скрытая марковская модель представляет собой систему такого же типа, за исключением того, что в ней нельзя напрямую сделать вывод о состоянии системы, исходя из наблюдений. Вместо этого наблюдения выступают подсказками в предположениях о состоянии системы (рис. 7.4).

Обратите внимание на то, что в действительных приложениях состояния обычно создают перекрывающиеся выходы, поэтому далеко не всегда понятно, какое из состояний за какой выход отвечает. Например, мы собираемся применить модель HMM к данным, подобным представленным на рис. 7.5. Это данные,

смоделированные для системы с четырьмя состояниями, но простой анализ графика временных рядов не позволяет определить количество состояний, их границы и области переходов.

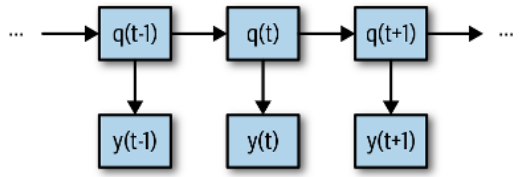


Рис. 7.4. *Процесс для модели HMM. Фактические состояния системы в текущий момент времени представлены как x(t), а наблюдаемые данные в этот же момент времени заданы как y(t). Только x(t) имеет влияние на y(t). Иначе говоря, значение x(t – 1) не привносит никакого вклада в предсказание y(t) при известном x(t). Аналогично при предсказании x(t + 1) учитывается только x(t), а знания x(t – 1) отбрасываются. Таким образом работают любые марковские процессы*

Рис. 7.5. *Смоделированный временной ряд для процесса с четырьмя состояниями. По его графику не совсем понятно, что состояний четыре, — на нем не обозначены ни их границы, ни переходы*

Тем не менее модели HMM находят применение в следующих прикладных задачах.

- Определение смены режима финансовых рынков (`https://perma.cc/JRT2-ZDVJ`).
- Классификация, прогнозирование и исправление данных в последовательностях ДНК (`https://perma.cc/4V4A-53TZ`).
- Распознавание стадий сна по данным ЭКГ (`https://perma.cc/G37Y-XBQH`).

Обучение модели

Мы утверждаем, что существует состояние, которое невозможно измерить напрямую, и во многих наборах данных, к которым можно было бы применить метод, невозможно определить их визуально. Тогда как нужно действовать, чтобы распознать скрытые состояния, не обладая априорными знаниями о них? Ответ: проходом по состояниям. Не существует волшебной палочки для получения наиболее вероятной последовательности скрытых состояний для объяснения наблюдений, но всегда можно провести их оценку, правильно описав систему.

В модели HMM утверждается, что система полностью описывается при наличии следующей информации.

- *Вероятность перехода* от $x(t)$ к $x(t + 1)$. Задается с помощью матрицы, аналогичной описанной выше и устанавливающей вероятности переходов между состояниями *A* и *B*. Размер такой матрицы зависит от количества гипотетических состояний.
- *Вероятность эмиссии*, или вероятность наблюдения $y(t)$ при заданном значении $x(t)$.
- Начальное состояние системы.

Рассмотрим конкретный случай, в котором обозначим переменные, принимающие участие в определении и обучении HMM-процесса.

- $Q = q_1, q_2, \ldots, q_N$ различных состояний системы.
- $A = a_{i,j} = a_{1,1}, a_{1,2}, ..., a_{N,N}$ — матрица вероятностей перехода, определяющая переход на любом заданном временном шаге из состояния *i* в состояние *j*.
- $O = o_1, o_2, \ldots, o_T$ — последовательность наблюдений, отобранных из этого процесса в порядке следования, т.е. временного ряда наблюдений.
- $b_{i(ot)}$ — вероятность эмиссии (вероятность наблюдения данного значения, o_t, если состояние q_i).
- $p = p_1, p_2, \ldots, p_N$ — начальные распределения вероятностей, а именно — вероятность того, что система имеет начальное состояние $q_1, q_2, \ldots, q_N$ соответственно.

Однако в реальных данных обычно ни одна из этих переменных не определена. Чаще всего изначально известна только фактическая последовательность наблюдаемых значений $y_1, y_2, \ldots, y_t$.

Алгоритм Баума–Уэлча

Для оценки параметров скрытой марковской модели воспользуемся алгоритмом Баума–Уэлча. Он сводится к решению сложной задачи — оценке значений

всех параметров, подробно описанных в предыдущем разделе. Это очень многогранная задача. Разобьем ее на следующие подзадачи.

- Определение вероятностей эмиссии для каждого возможного скрытого состояния и перехода из каждого возможного скрытого состояния в другое возможное скрытое состояние. Используем алгоритм Баума–Уэлча.
- Определение наиболее вероятного скрытого состояния на каждом временном шаге с учетом всей истории наблюдений. Используем алгоритм Витерби (описан ниже).

Это родственные задачи, каждая из которых достаточно сложна и требует для решения больших вычислительных затрат. Более того, они связаны одна с другой. Для решения двух взаимосвязанных задач — оценки параметров и максимизации вероятности — можно использовать алгоритм максимизации математического ожидания для перехода между этими двумя этапами, пока не будет найдено приемлемое решение.

Чтобы применить алгоритм Баума–Уэлча, первым делом нужно определить функцию правдоподобия, которая представляет вероятность наблюдения имеющейся последовательности при заданных гипотетических параметрах. В нашем случае такими предполагаемыми параметрами будут математические параметры для каждого постулируемого состояния.

Например, предположим, что состояния описываются гауссовыми выходными данными с различными средними значениями и стандартными отклонениями в наблюдаемых значениях, зависящими от состояния. Рассмотрим модель с двумя такими состояниями, которую будем описывать в терминах μ_1, σ_1, μ_2 и σ_2, где $\mu_{u=i}$ обозначает среднее значение i-го состояния, а σ_i — стандартное отклонение i-го состояния. С их помощью рассчитываются вероятности эмиссии, все вместе обозначаемые как θ. Кроме того, нам нужно обозначить последовательность состояний x_1, x_2, ..., x_t (все вместе X_t), которые нами не наблюдаются, но давайте представим, что они все же существуют.

Функция правдоподобия такой системы должна описывать вероятность наблюдения существующей последовательности при заданных параметрах вероятности эмиссии (т.е. вероятности наблюдения в строго заданном состоянии) и последовательности скрытых состояний как интеграла по всем возможным X_t, таких что:

$$p(y_1, y_2, \ldots, y_t \mid \mu_1, \sigma_1, \mu_2, \sigma_2, \ldots, \mu_N, \sigma_{Nt}) = p(y_1, y_2, \ldots, y_t \mid \mu_1, \sigma_1, \mu_2, \sigma_2, \ldots, \mu_N, \sigma_N).$$

Однако это сложная задача по нескольким причинам, включая факт ее экспоненциального усложнения с увеличением количества временных шагов, что указывает на невозможность проведения исчерпывающего анализа

по перекрывающимся значениям. Следовательно, нам нужно упростить задачу, обратившись к следующему EM-алгоритму.

1. Произвольно инициализируем переменные вероятности эмиссии.
2. Вычисляем вероятность каждого возможного X_t, учитывая значения вероятности эмиссии.
3. Используем эти значения X_t, чтобы получить лучшую оценку переменных вероятности эмиссии.
4. Повторяем пп. 2 и 3 до тех пор, пока не будет достигнута сходимость.

Более неформально это означает, что при случайном размещении двух распределений мы будем рассматривать каждый временной шаг и определять на каждом из них вероятность наблюдения определенного состояния (например, состояния *A* или *B* на временном шаге *t*). Назначив предполагаемое состояние для каждого временного шага, используем эти метки для переоценки вероятностей эмиссии (обнуляя лучшие среднее и стандартное отклонение для состояния). Затем процесс нужно повторить заново, используя недавно обновленные переменные вероятности эмиссии для улучшения оценки траектории X_t.

Не забывайте, что найти оптимальный набор параметров в таком способе использования EM-алгоритма удается далеко не всегда. Может понадобиться много попыток, прежде чем вы добьетесь успеха, — продолжительность операции зависит от конкретных данных и выбранной модели.

Алгоритм Витерби

После оценки параметров процесса HMM, например с помощью алгоритма Баума–Уэлча, нужно выполнить следующую, не менее важную задачу, заключающуюся в определении наиболее вероятного ряда состояний, основанных на временном ряду наблюдаемых значений.

В отличие от алгоритма Баума–Уэлча алгоритм Витерби гарантированно предоставляет наилучшее решение поставленной выше задачи. А все потому, что он относится к алгоритмам динамического программирования, предназначенным для полного и эффективного изучения диапазона возможных совпадений методом сохранения решений, полученных для отдельных участков пути, что позволяет отказаться от повторного пересчета всех возможных вариантов в случае удлинения пути (рис. 7.6).

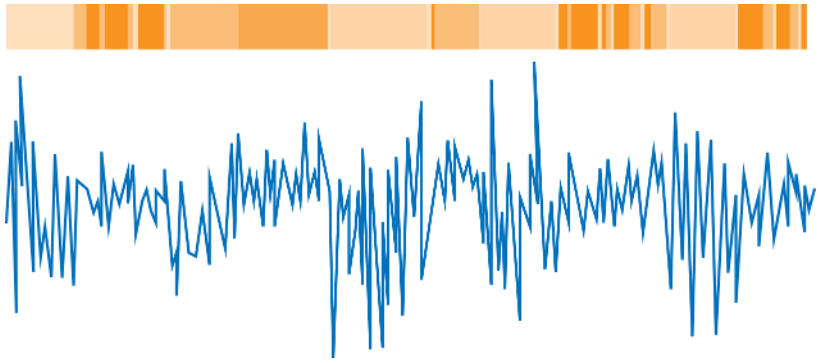


Рис. 7.6. *Алгоритм Витерби ищет все возможные пути, которые могут объяснить наблюдаемый временной ряд, причем отдельный путь указывает на состояние, которое наблюдалось в каждый момент времени*

Динамическое программирование

Специалисты по анализу данных не всегда располагают полным набором алгоритмов, но при изучении временных рядов точно не смогут обойтись без базовых решений, в которых предположения и повторное рассмотрение выступают необычайно важными техниками исследования упорядоченных во времени данных.

Проще всего объяснить, что такое динамическое программирование, на уже ставшем классическим примере последовательности чисел Фибоначчи. Представьте, что вам требуется вычислить восьмое число Фибоначчи. Проще всего получить его, зная шестое и седьмое числа последовательности Фибоначчи. А чтобы получить их, нужны четвертое и пятое числа Фибоначчи и т.д. Таким образом, решение более сложных задач по вычислению последовательностей Фибоначчи основывается на результатах более простых таких задач. Следовательно, при вычислении чисел Фибоначчи их нужно как-то сохранять, чтобы иметь возможность использовать в последующих, еще более сложных задачах. По этой причине динамическое программирование также известно как *мемоизация*.

Динамическое программирование применяется для решения задач со следующими отличительными признаками.

- Решение задачи размера N можно основать на решении задачи размера $N - 1$. По этой причине *мемоизация* решений более ранних, простых задач помогает более эффективно решать более поздние, заведомо более сложные задачи.
- Задачи имеют явно выраженный порядок масштабирования от более простой к более сложной.
- Всегда можно выделить базовую задачу, подлежащую простому расчету.

Код обучения модели HMM

Хотя процесс обучения модели HMM очень сложен, он реализуется с помощью специализированных пакетов языка R очень просто. В следующем примере используются инструменты пакета depmixS4. Вначале нужно получить надлежащий временной ряд. Воспользуемся следующим кодом.

```
## R
## Обратите внимание: здесь задается начальное значение
## При одном и том же начальном значении числа должны совпадать
set.seed(123)

## Параметры распределения для всех четырех состояний рынка,
## подлежащих моделированию
bull_mu <- 0.1
bull_sd <- 0.1

neutral_mu <- 0.02
neutral_sd <- 0.08

bear_mu <- -0.03
bear_sd <- 0.2

panic_mu <- -0.1
panic_sd <- 0.3

## Представление параметров векторами для упрощения индексации
mus <- c(bull_mu, neutral_mu, bear_mu, panic_mu)
sds <- c(bull_sd, neutral_sd, bear_sd, panic_sd)

## Константы, описывающие генерируемые временные ряды
NUM.PERIODS <- 10
SMALLEST.PERIOD <- 20
LONGEST.PERIOD <- 40

## Определение рыночных дней случайным образом.
## Каждый такой день соответствует отдельному состоянию рынка
days <- sample(SMALLEST.PERIOD:LONGEST.PERIOD, NUM.PERIODS,
               replace = TRUE)

## Генерирование временного ряда состояний рынка для заданного
## количества дней и его добавление к общему временному ряду
returns <- numeric()
true.mean <- numeric()
for (d in days) {
   idx = sample(1:4, 1, prob = c(0.2, 0.6, 0.18, 0.02))
   returns <- c(returns, rnorm(d, mean = mus[idx], sd = sds[idx]))
   true.mean <- c(true.mean, rep(mus[idx], d))
}
```

В предыдущем коде моделируется процесс биржевой торговли на бычьих, медвежьих, нейтральных и панических рынках. В нем устанавливается случайное количество дней, для которых сохраняются состояния рынка, а также определяются переменные, описывающие распределение вероятности эмиссии для каждого состояния (_mu и _sd, хранящие значения, которые подлежат измерению в заданном состоянии).

Чтобы получить представление о сгенерированном кодом временном ряде и частотности каждого состояния, нужно понять, сколько дней в выборке соответствует каждому значению переменной true.mean, по которой отслеживаются состояния.

```
## R
> table(true.mean)
true.mean
-0.03 0.02 0.1
  155  103  58
```

Невероятно, но факт! Несмотря на намерение включить четыре состояния в моделируемый ряд, их было добавлено только три. Скорее всего, это связано с очень низкой вероятностью включения (0,02) четвертого состояния. Мы видим, что наименее вероятное состояние даже не было выбрано для добавления в ряд. Таким образом, далеко не всегда известно, что для заданного временного ряда учитываются не все возможные состояния, что еще раз свидетельствует о высокой сложности алгоритма модели HMM и сложности его обучения. Как бы там ни было, дальнейший анализ будет выполнен для группы из четырех состояний, чтобы увидеть, к какому результату это приведет.[3]

Можно переходить к обучению HMM. Результирующая модель HMM будет представлять временные ряды апостериорных вероятностей для каждого состояния для любого количества указанных состояний. В соответствии с более ранним описанием EM-алгоритма от нас требуется указать всего одно значение — число предполагаемых состояний. Остальные параметры будут определяться по мере выполнения вычислительных этапов.

Как это часто бывает, при работе с программными пакетами наиболее сложная часть анализа на самом деле оказывается очень простой в реализации и представлении в коде. В нашем случае используется пакет depmixS4[4] языка R. Обучение

[3]Обратите внимание, что еще одна проблема с моделируемыми данными заключается в том, что мы не создали матрицу вероятности перехода из одного состояния в другое для управления потоком скрытого состояния. По сути, мы предположили, что состояние с большей вероятностью останется таким, как есть, в течение многих дней подряд, а затем с равной вероятностью сменится любым другим состоянием. Мы упустили формальную спецификацию и отказались от использования матрицы переходов для упрощения кода.

[4]Название этого пакета заимствовано у альтернативного названия "HMM" — Dependent Mixture Models (Модель зависимых смесей).

модели проводится дважды. Во-первых, с помощью функции `depmix()`, в которой устанавливается ожидаемое распределение, указываются число состояний и входные данные, которые будут использоваться при обучении. Далее в действие вступает функция `fit()`, которая принимает в качестве входных данных спецификацию модели. И только после этого применяется функция `posterior()`, генерирующая апостериорное распределение подписей состояний с учетом соответствия данных. Начиная с этого момента модель считается обученной, и нам остается только решить задачу по разметке данных, чтобы оценить параметры, описывающие распределения состояний и вероятности переходов.

```
## R
require(depmixS4)
hmm.model <- depmix(returns ~ 1, family = gaussian(),
                    nstates = 4, data=data.frame(returns=returns))
model.fit <- fit(hmm.model)
post_probs <- posterior(model.fit)
```

Этим кодом создается модель `hmm.model`, в которой в качестве наблюдаемого указывается вектор `returns`. В нем определяется число состояний (`4`) и соответствие вероятности эмиссии нормальному распределению (см. параметр `family`). Модель обучается с помощью функции `fit()`, а для вычисления апостериорных вероятностей применяется функция `posterior()`. Апостериорные вероятности определяют правдоподобие состояний в данное время для параметров модели, определенных в процессе обучения.

Теперь можно переходить к визуализации состояний на одном графике с измеренными значениями (рис. 7.7).

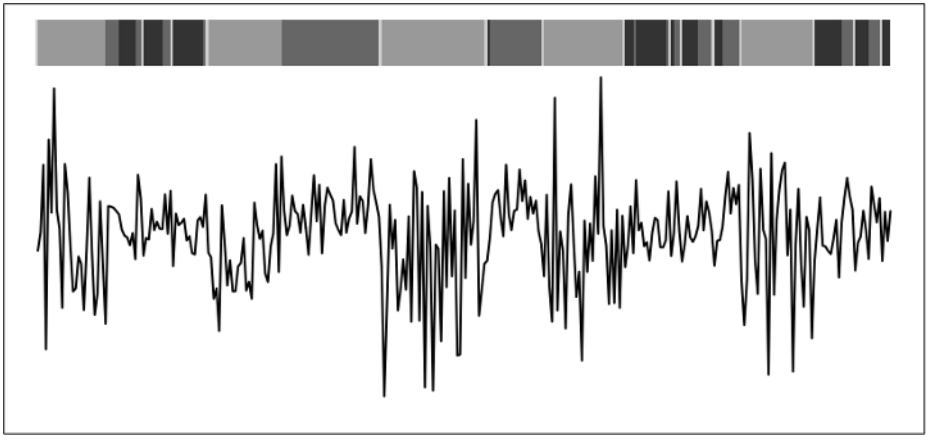


Рис. 7.7. *Для обозначения состояний используется цветовая шкала, а реальные значения представлены на линейном графике, обозначенном сплошной черной линией. Вертикальной белой линией обозначен узкий временной срез, в котором (в соответствии с оценкой) процесс переходит в редкое четвертое состояние*

```
## R
plot(returns, type = 'l', lwd = 3, col = 1,
     yaxt = "n", xaxt = "n", xlab = "", ylab = "",
     ylim = c(-0.6, 0.6))

lapply(0:(length(returns) - 1, function (i) {
  ## Добавление прямоугольника соответствующего цвета,
  ## обозначающего состояние в заданный момент времени
  rect(i,-0.6,(i + 1),0.6,
    col = rgb(0.0,0.0,0.0,alpha=(0.2 * post_probs$state[i + 1])),
    border = NA)
}
```

Сведения о предполагаемых параметрах распределения хранятся в отдельных атрибутах. При их просмотре помните об исходных настройках генерации данных.

```
bull_mu    <- 0.1
bull_sd    <- 0.1

neutral_mu <- 0.02
neutral_sd <- 0.08

bear_mu    <- -0.03
bear_sd    <-  0.2

panic_mu   <- -0.1
panic_sd   <-  0.3
```

Попытавшись сравнить состояния, фактически обозначенные в данных (режим паники на бирже исключен из данных), можно получить представление о корреляции между следующими группами.

```
> attr(model.fit, "response")
[[1]]
[[1]][[1]] <- среднее значение близко к режиму паники, но этот
              режим в выборке не представлен, поэтому четвертому
              состоянию назначены большие отрицательные значения
Model of type gaussian (identity), formula: returns ~ 1
Coefficients:
(Intercept)
-0.09190191
sd 0.03165587

[[2]]
[[2]][[1]] <- соответствует режиму медвежьего рынка
Model of type gaussian (identity), formula: returns ~ 1
Coefficients:
(Intercept)
-0.05140387
sd 0.2002024
```

```
[[3]]
[[3]][[1]] <- соответствует режиму бычьего рынка
Model of type gaussian (identity), formula: returns ~ 1
Coefficients:
(Intercept)
0.0853683
sd 0.07115133

[[4]]

[[4]][[1]] <- соответствует режиму нейтрального рынка
Model of type gaussian (identity), formula: returns ~ 1
Coefficients:
(Intercept)
-0.0006163519
sd 0.0496334
```

Одна из возможных причин, по которым обучение не привело к хорошему согласованию с базовыми скрытыми состояниями, заключается в отказе от использования правильной матрицы переходов, необходимой для обучения. В результате переходы между состояниями не были марковскими, и это оказало пагубное влияние на обучение. Кроме того, мы пытались приблизить относительно короткий временной ряд с небольшим количеством переходов между состояниями, в то время как модели HMM лучше работают на более длинных временны*х* рядах с б*о*льшим количеством переходов. Я бы рекомендовал придумать более реалистичные искусственные данные для тестирования предлагаемой модели HMM. Помните, что в большинстве практических задач приходится работать с ненаблюдаемыми состояниями, поэтому перед тем как взяться за более амбициозные проекты, постарайтесь разобраться в факторах, вносящих ограничения в точность модели, в предельно простых случаях (с искусственными данными).

Модели HMM подходят для анализа многих видов данных. Они использовались для моделирования поведения финансовых рынков в фазе роста и рецессии, определения стадии скручивания белка в клетках и описания перемещения людей (до появления глубокого изучения). На сегодняшний день они остаются востребованными — чаще в задачах исследования динамического поведения систем, чем прогнозирования. Кроме того, модели HMM предоставляют больше сведений о процессах, чем точечные оценки или прогнозы. В такую модель легко включить априорные знания или предположения, например, указав количество состояний, используемых для ее обучения. Тем самым обеспечиваются преимущества статистических методов, но сохраняется возможность параметризации априорных знаний о системе.

Математические принципы и уравнения, используемые для расчета моделей HMM, хорошо изучены и доступны для понимания. На ваше рассмотрение предлагается большое количество программных инструментов и численных алгоритмов оптимизации, используемых при обучении моделей HMM по данным. Вы также познакомитесь с методами динамического программирования, с которыми нужно быть на "ты" каждому специалисту по анализу данных или программисту.

Как и фильтры Калмана, модели HMM могут использоваться для решения задач самых разных типов. На самом деле количество задач логического вывода, связанных с HMM-системами, увеличивается с повышением сложности дискретных состояний, каждое из которых имеет собственную вероятность эмиссии. Перечислим некоторые из задач логического вывода, с которыми вы можете столкнуться при использовании моделей HMM.

- Определение наиболее вероятного описания состояний, производящих ряд наблюдений. Включает в себя оценку вероятностей эмиссии этих состояний, а также матрицы переходов, которая показывает вероятность перехода процесса из одного состояния в другое. Мы проделали это выше, хотя и не задавали вероятности перехода в явном виде.
- Определение наиболее вероятной последовательности состояний с учетом ряда наблюдений и описания состояний, а также вероятностей их эмиссии и переходов. Мы также выполнили эту задачу в предыдущем упражнении. Иногда ее называют "наиболее вероятное объяснение", и для ее решения обычно применяется алгоритм Витерби.
- Фильтрация и сглаживание. В этой ситуации фильтрация будет соответствовать оценке скрытого состояния последнего временного шага с учетом последнего наблюдения. Сглаживание будет соответствовать определению наиболее вероятного распределения скрытого состояния на конкретном временном шаге с учетом наблюдений до, во время и после этого временного шага.

Байесовский структурный временной ряд

Байесовский структурный временной ряд (Bayesian structural time series — BSTS) связан с линейной гауссовой моделью, которую мы ранее использовали в фильтрах Калмана. Основное различие состоит в том, что байесовские структурные временные ряды позволяют использовать уже существующие компоненты для построения более сложных моделей, которые отражают известные факты или интересные гипотезы о системе. С их помощью можно разработать структуру модели, провести обучение по имеющимся данным для оценки параметров модели и посмотреть, насколько хорошо модель описывает и предсказывает поведение системы.

Модель BSTS базируется на более сложных математических принципах, чем те, которые использовались в линейной гауссовой модели, которая рассматривалась

при изучении фильтра Калмана. Ниже приведен только краткий их обзор, а также пример реализации в программном коде.

Обучение модели BSTS выполняется в четыре этапа, следующих в таком порядке.

1. Определение структурной модели, в частности задание априорных вероятностей.
2. Применение фильтра Калмана для обновления оценок состояния на основе наблюдаемых данных.
3. Применение метода "пик-плато" для выбора переменных в структурной модели.[5]
4. Усреднение по байесовской модели для объединения результатов с целью составления прогноза.

В следующем примере мы сосредоточимся только на шагах 1 и 2, в которых определяется гибкость модели, основанной на существующих модульных компонентах, а затем обучим ее на имеющихся данных с помощью байесовского метода, обновляющего оценку параметров с течением времени.

Код реализации байесовских структурных временных рядов

В дальнейшей работе будем использовать популярный и невероятно производительный пакет `bsts`, разработанный Google, и открытый набор данных, полученный с ресурса `OpenEI.org`.

Отобразим исходные данные на графике, чтобы получить представление о том, что именно нам необходимо смоделировать (рис. 7.8).

```
## R
elec = fread("electric.csv")

require(bsts)
n = colnames(elec)[9]
par(mfrow = c(2, 1))
plot(elec[[n]][1:2000])
plot(elec[[n]][1:96)

## Как указывалось ранее, для правильного анализа временных рядов
## важно правильно подобрать временную шкалу
```

[5]Познакомиться с методом "пик–плато" можно в Википедии. Его математика довольно сложная, и мы не будем останавливаться на ее детальном рассмотрении. Метод "пик–плато" наиболее востребован в системах с большим количеством входных данных, которые нужно описать упрощенной моделью с небольшим количеством переменных.

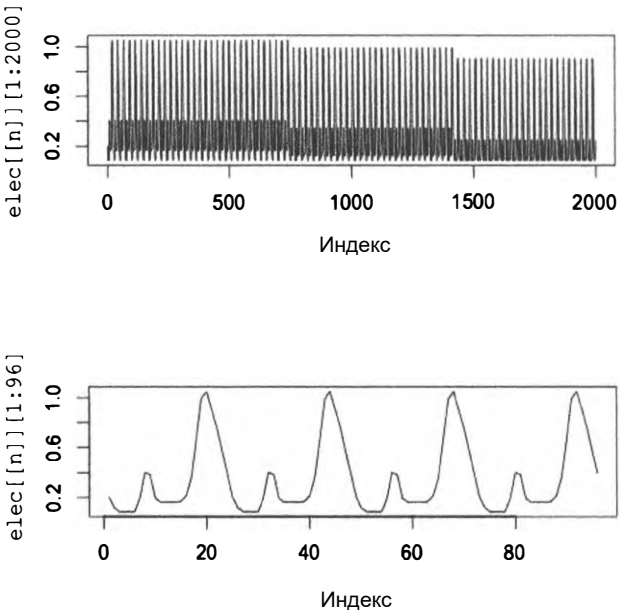


Рис. 7.8. *Полный временной ряд (вверху) моделируемых данных (2000 измерений в час) и более короткое, понятное подмножество этих данных (внизу). График лучше всего подходит для изучения дневных шаблонов*

Просмотр данных позволяет определиться с принципами моделирования. На графике достаточно четко просматривается дневной шаблон, и существует вероятность получения подобного шаблона для недельного изменения данных. Такие шаблоны отражают сезонное поведение данных, которое нужно отразить в модели. Кроме того, в ней нужно учесть присутствующий в данных тренд, представляющий нестационарное поведение, наблюдаемое в верхней области построения рис. 7.8.

```
## R
ss <- AddLocalLinearTrend(list(), elec[[n]])
ss <- AddSeasonal(ss, elec[[n]], nseasons = 24, season.duration = 1)
ss <- AddSeasonal(ss, elec[[n]], nseasons = 7, season.duration = 24)
```

Локальный линейный тренд модели отображает тот факт, что среднее значение и наклон линии тренда описываются случайным блужданием.[6]

Сезонная составляющая модели представлена двумя аргументами, один из которых указывает количество сезонов, а другой — продолжительность сезона. Первая добавляемая в модель сезонная составляющая определяет дневной цикл. В нее нужно включать сезоны почасового изменения данных — каждый длительностью один час. Вторая сезонная составляющая представляет недельный цикл. Нам нужно добавить в нее сезоны для каждого дня недели, каждый из которых длится 24 часа.

[6]Дополнительная информация приведена на сайте `https://perma.cc/2N77-ALJ4`.

Вы можете задаться вопросом, действительно ли нужно начинать наблюдения в 12:01 в понедельник (или определить неделю другим способом). В данном случае согласованность данных важнее, чем назначение сезонной подписи первому дню недели. На самом деле в наблюдаемой повторяющейся структуре анализ сезонности можно выполнять при абсолютно любом способе разбивки данных на 24 часовые интервалы.

Ниже приведена наиболее сложная в вычислительном отношении часть кода. Преимущество пакета `bsts` заключается в возможности проведения вычислений по методу Монте-Карло с использованием марковских цепей (MCMC — Markov Chain Monte Carlo).

```
## R
model1 <- bsts(elec[[n]],
               state.specification = ss,
               niter = 100)
plot(model1, xlim = c(1800, 1900)
```

Всегда можно проверить сезонные компоненты. Например, сезонная составляющая для дней недели проявляется следующим образом (рис. 7.9 и 7.10).

```
## R
plot(model1, "seasonal", nseasons = 7, season.duration = 24)
```

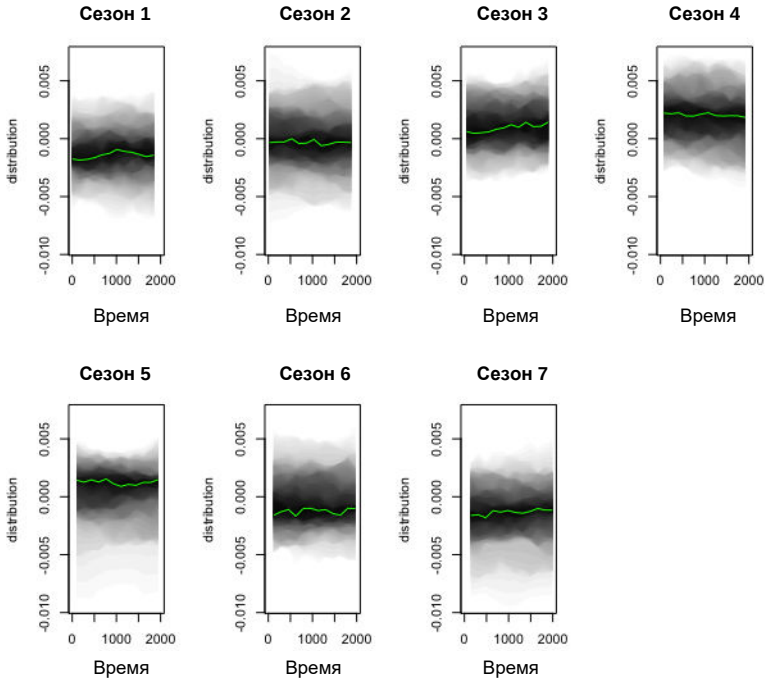


Рис. 7.9. *В недельных сезонах проявляется различие во вкладах по дням недели. Распределение параметров является устойчивым по времени для каждого дня недели*

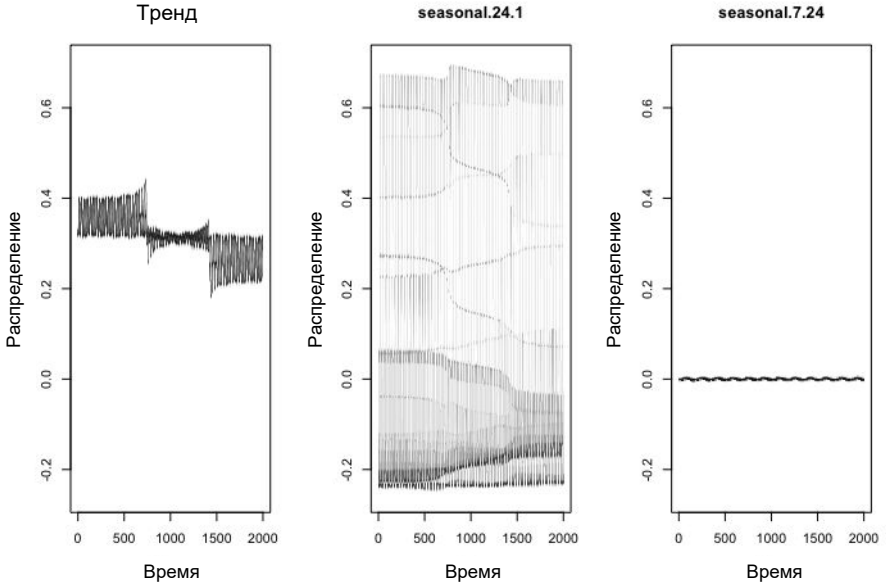


Рис. 7.10. *Распределение вкладов каждого тренда, а также сезонных составляющих — дневных и недельных. Для получения прогнозных значений нужно просуммировать вклады всех трех компонентов*

Недельная сезонная составляющая демонстрирует хорошую устойчивость, в то время как дневная сезонность, показанная на среднем графике рис. 7.10, сильно зависит от времени суток (вероятнее всего, привязана к световому дню). На рис. 7.10 также проявляется локальный линейный тренд, показывающий общее снижение потребления электроэнергии.

Наконец, построим прогноз апостериорного распределения, отображаемого в графическом виде (рис. 7.11). Обратите внимание, что модель настолько гибка, что позволяет указать количество временных горизонтов, подлежащих предсказанию. Помните, что временной горизонт указывается на почасовой основе — прогнозирование вперед на 24 временных горизонта может показаться амбициозным шагом, но на самом деле будет выполняться всего на один день. В нашем примере прогнозирование проводится для 72 периодов времени, что определяется контекстом задачи.

```
pred <- predict(model1, horizon = 24, quantiles = c(0.05, 0.95))
plot(pred, plot.original = 72)
```

В пакете `bsts` и байесовском моделировании структурных временных рядов есть немало возможностей, которые подлежат отдельной параметризации.

- Возможность указывать нестандартные априорные распределения.
- Выбор регрессоров с помощью метода "пик-плато".
- Усреднение байесовской модели.

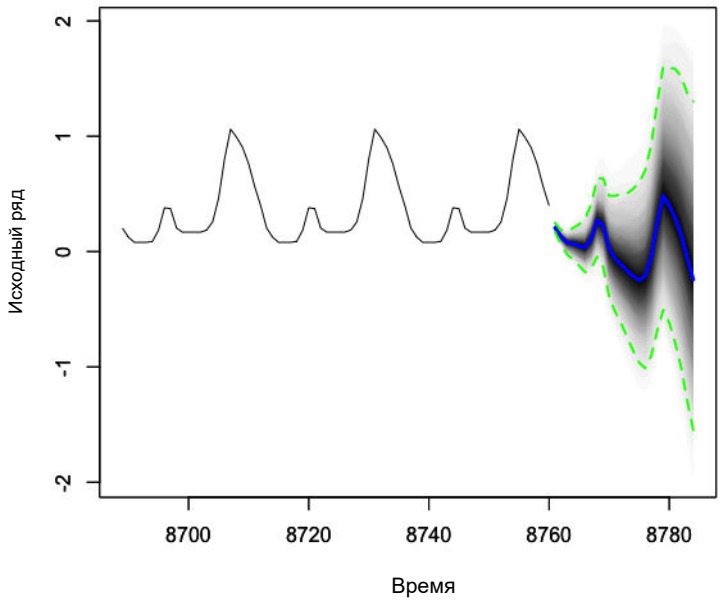


Рис. 7.11. *Исходные данные за временной период 72 часа, дополненные прогнозными значениями для последующих 24 часов с выделенными квантилями 5% и 95%. Заметьте, что разброс прогнозных значений возрастает по мере увеличения периода прогнозирования*

Всю необходимую информацию о реализации этих возможностей средствами пакета `bsts` вы найдете в его документации.

Вы получили только поверхностное представление о задачах, возлагаемых на модели BSTS. Ниже перечислено несколько важных особенностей, характеризующих сильные стороны технологии BSTS.

- Модели BSTS позволяют выполнять моделировании с учетом любых априорных распределений. Стандартная линейная гауссова модель, которую мы рассмотрели при обсуждении фильтра Калмана, является лишь одним из вариантов классического априорного распределения, в то время как модель BSTS работает с множеством других вариантов (например, асимметричными априорными вероятностями).
- В BSTS разрешается самостоятельно выбирать переменные.
- Модели BSTS можно комбинировать с методами усреднения байесовских моделей, что позволяет устранить неопределенность, связанную с выбором модели.

Хотя в текущем примере данные возможности не учитывались, все они доступны для реализации с помощью пакета `bsts`. Вы можете найти много соответствующих примеров в Интернете.

Дополнительные источники

- Фильтры Калмана и линейные гауссовы модели пространства состояний

  Greg Welch and Gary Bishop, "An Introduction to the Kalman Filter," technical report, University of North Carolina at Chapel Hill, 1995, `https://perma.cc/ZCU8-MXEF`

  Вводный курс по фильтру Калмана, содержащий базовые сведения и описание принципов матричных вычислений, применяемых при его расчете. Включает рассмотрение расширенного фильтра Калмана и наиболее распространенных сценариев его практического применения — описания нелинейных процессов или учета нелинейных ошибок измерения.

  R.E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," Transactions of the ASME—Journal of Basic Engineering 82, Series D (1960): 35– 45, `https://perma.cc/GNC4-YLEC`

  Статья 1960 года, включающая исходное описание фильтра Калмана. Для ее понимания достаточно базовых познаний в статистике и математическом анализе. Представляет интерес с исторической точки зрения, поскольку содержит упоминание об исходном назначении фильтра Калмана и мотивационном контексте его создания.

  R. Labbe, "Kalman and Bayesian Filters in Python," GitHub repository, `https://perma.cc/CMU5-Y94A`

  Репозиторий GitHub, содержащий десятки примеров использования фильтров Калмана и связанных с ними общих методов фильтрации. Структурирован как учебник: включает рабочие упражнения, книгу в формате PDF и сборник заданий с примерами их решения.

  Marie Auger-Methe et al., "State-Space Models' Dirty Little Secrets: Even Simple Linear Gaussian Models Can Have Estimation Problems," Scientific Reports 6, no. 26677 (2016), `https://perma.cc/9D8V-Z7KJ`

  В этой статье освещается случай подверженности простых линейных гауссовых моделей, подобных применяемым при изучении фильтра Калмана, заведомо ошибочной спецификации, особенно в случаях относительно больших ошибок измерения значений временного ряда. В материале статьи акцент делается на решении экологических задач, но рассматриваемая в ней проблематика будет близкой для множества других дисциплин, основанных на управлении данными. В ней предлагаются совершенно иные подходы, обладающие собственными преимуществами, по сравнению с принятыми в исходном методе.

- Скрытые марковские модели

  Andrew Moore, "Hidden Markov Models," lecture notes, School of Computer Science, Carnegie Mellon University, `https://perma.cc/K3HP-28T8`

  Лекционные материалы с обзорным описанием методов HMM, включающие иллюстрации алгоритмов оценки и примеры практического применения HMM в робототехнических приложениях.

  Dan Klein, "Artificial Intelligence: Hidden Markov Model," lecture notes, University of California Berkeley, `https://perma.cc/V7U4-WPUA`

  Еще один справочник по методам HMM. В нем описана роль HMM в технологиях оцифровки речи, а также алгоритмах искусственного интеллекта в стратегических играх.

  user34790, "What Is The Difference Between the Forward-Backward and Viterbi Algorithms?" question posted on Cross Validated, StackExchange, July 6, 2012, `https://perma.cc/QNZ5-U3CN`

  Пост на сайте StackExchange, вызвавший активное обсуждение и предлагающий интересные решения, которые находят широкое применение в алгоритмах оценки, используемых в методах HMM. Его материал поможет разобраться в принципах применения HMM для изучения данных временных рядов даже в случаях отсутствия заинтересованности в детальном изучении алгоритмов моделирования.

- Байесовские структурные временные ряды

  Mark Steel, "Bayesian Time Series Analysis," in Macroeconometrics and Time Series Analysis, ed. Steven N. Durlauf and Lawrence E. Blume (Basingstoke, UK: Palgrave Macmillan, 2010), 35–45, `https://perma.cc/578D-XCVH`

  Статья предлагает всестороннее описание байесовских принципов анализа временных рядов и включает краткое обсуждение сильных и слабых сторон рассматриваемых методов.

  Steven Scott and Hal Varian, "Predicting the Present with Bayesian Structural Time Series," unpublished paper, June 28, 2013, `https://perma.cc/4EJX-6WGA`

  Документ Google, основанный на данных экономических временных рядов и являющийся результатом решения задачи прогнозирования применимо к данным за указанный период времени, которые представляются с различными задержками. В частности, авторами предпринята попытка спрогнозировать уровень безработицы по данным поисковых запросов Google — задачи, в которой сведения о безработице публикуются периодически, а поисковые запросы обрабатываются непрерывно.

Фактически такое "текущее прогнозирование", несмотря на временные задержки в сборе данных, выполняется для настоящего момента времени. Задача решается с помощью комбинации байесовских структурных временных рядов и ансамблевых методов.

Jennifer Hoeting et al., "Bayesian Model Averaging: A Tutorial," Statistical Science 14, no. 4 (1999): 382–401, `https://perma.cc/BRP8-Y33X`

В этой статье приведено детально описание принципов усреднения байесовской модели, находящих применение в самых разных методах. В ней показано, что усреднение байесовской модели позволяет избавиться от неопределенности, вызванной выбором неоптимальной модели. Хорошо проработанные примеры позволяют предельно точно оценить неопределенность в составляемых прогнозах.

ГЛАВА 8

Генерация и выбор признаков для временных рядов

В предыдущих двух главах мы рассмотрели методы анализа временных рядов, в ходе которого обучение модели выполняется с использованием всех точек данных временного ряда. В рамках подготовки к изучению методов анализа временных рядов с помощью технологий машинного обучения в следующей главе нам предстоит познакомиться с принципами генерации и выбора признаков в временных рядах. Если вы еще не знакомы с подобными концепциями, то узнаете из этой главы много нового и интересного. Это интуитивно понятный процесс, знаменующий творческий подход к анализу данных.

Генерация признаков — это процесс поиска количественного способа инкапсуляции наиболее важных характеристик данных временного ряда в виде нескольких числовых значений и категориальных подписей. С помощью набора признаков можно сжимать необработанные данные временного ряда в более короткое представление (небольшой пример приведен далее). Например, существует очень простой способ получения признаков — анализ временного ряда по среднему значению и количеству временных шагов в нем. Это один из способов описания временного ряда без пошагового прохода по всем необработанным данным.

Целью генерации признаков является сжатие как можно большего количества информации о полном временном ряде в нескольких показателях или, как вариант, использование этих показателей для определения наиболее важной информации о временном ряде и отбрасывания остальных факторов. Наибольшее распространение признаки получили в алгоритмах машинного обучения, большинство из которых разрабатывались для данных, не имеющих родственного отношения с временными рядами, но могут быть успешно применены к их обработке при условии представления временного ряда в виде правильно структурированных входных данных. В частности, в этой главе речь пойдет о пакетах, предназначенных для автоматического генерирования часто используемых признаков временных рядов, что избавит нас от необходимости их изобретения или собственноручной реализации.

После получения в свое распоряжение предположительно актуальных признаков нужно убедиться в их полезности для модели. Хотя вам вряд ли удастся сгенерировать много бесполезных признаков вручную, рано или поздно вы все же столкнетесь с ними, особенно при автоматическом получении для временных рядов, используемых в машинном обучении, с помощью специального кода. По этой причине всегда нужно проверять избранные признаки на пригодность, чтобы иметь возможность отказаться от их применения до проведения анализа данных.

При разработке модели машинного обучения не предполагалось применять для работы с временными рядами, а потому они плохо адаптируются к приложениям их автоматического исследования. Один из способов заставить такие модели работать с временными данными заключается в описании их признаками. Например, представляя одномерный временной ряд не последовательностью чисел, задающей направление прохода по данным процесса, а набором признаков, к ним можно применить любые методы обработки перекрестных данных.

В этой главе мы сначала рассмотрим очень простой пример генерации признаков для коротких временных рядов. Затем перейдем к изучению пакетов генерации признаков для временных рядов, представленных в языках R и Python. И наконец рассмотрим пример кода автоматической генерации и выбора признаков. Изучив эту главу, вы получите все необходимые познания о предварительной обработке данных временных рядов, востребованные при знакомстве с алгоритмами машинного обучения, описанными в главе 9.

Вводный пример

Представьте, что утренние, полуденные и вечерние температуры на прошлой неделе были такими, как показано в табл. 8.1.

Внимательно изучив данные, можно увидеть элементы периодичности (ежедневный цикл), а также тенденцию к общему повышению температуры. Но в базе данных невозможно сохранить изображение графика, а большинство методов, принимающих изображения в качестве входных данных, очень ресурсоемкие и стремятся привести их к более сжатым метрикам. Таким образом, нам нужно самостоятельно определить сводные показатели. Вместо представления 21 числа из табл. 8.1 временным рядом данные можно описать всего несколькими словами и числами.

- Дневная периодичность
- Тенденция к росту (этот факт определяется количественным путем по наклону графика)
- Средние значения для каждого утра, полудня и вечера

Таблица 8.1. Температура на предыдущей неделе

Время	Температура, °F
Полдень понедельника	52
Вечер понедельника	15
Утро вторника	37
Полдень вторника	52
Вечер вторника	15
Утро среды	37
Полдень среды	54
Вечер среды	16
Утро четверга	39
Полдень четверга	51
Вечер четверга	12
Утро пятницы	41
Полдень пятницы	55
Вечер пятницы	20
Утро субботы	43
Полдень субботы	58
Вечер субботы	22
Утро воскресенья	46
Полдень воскресенья	61
Вечер воскресенья	35

Таким образом, мы свели 21-точечный временной ряд к нескольким числам — довольно сильное сжатие без потери детальных сведений. Это простейший случай генерации признаков. Последующий *подбор признаков* влечет за собой удаление любых признаков, недостаточно информативных для того, чтобы быть включенными в модель. Критерии выбора "правильных" признаков зависят от их дальнейшего назначения.

Общие принципы выбора признаков

Как и в случае с любыми другими аспектами анализа данных, при определении признаков для временных рядов некого набора данных необходимо помнить о конечной цели анализа, а также о том, что чрезмерные усилиях, прилагаемые

для генерации как можно большего количества признаков, могут привести к переобучению модели, а не к осмысленным результатам.

Наилучшим подходом является разработка набора потенциально полезных признаков на этапах исследования и очистки временных рядов. В ходе визуализации и анализа данных вы научитесь наблюдать различия между временными рядами одного набора данных или разными временными периодами одного и того же временного ряда, постепенно вырабатывая концепции и метрики, на которых основывается разметка или прогнозирование временных рядов. Здесь вам станут полезными любые базовые знания о системе или сведения о рабочих гипотезах, которые требуется учитывать в последующем анализе.

В последующем материале вам предстоит познакомиться с концепциями, которые следует принимать во внимание при генерировании признаков временных рядов.

Природа временного ряда

Принимая решение о генерировании тех или иных временных рядов, необходимо помнить об их основных атрибутах, определенных на этапе исследования и очистки данных.

Стационарность

Стационарность является одним из важнейших факторов. Многие признаки временных рядов выбираются исходя из стационарности данных и оказываются бесполезными, если исходные временные ряды не оказываются стационарными или по крайней мере эргодическими. Например, использование среднего значения временного ряда в качестве признака будет разумным только тогда, когда оно имеет смысл, т.е. в случае стационарных числовых последовательностей. Среднее значение не имеет особого смысла при анализе нестационарных временных рядов, поскольку в их случае оно будет представлять результат обработки слишком большого числа запутанных процессов — трендов или сезонных циклов.

Стационарные и эргодические временные ряды

Эргодический временной ряд — это ряд, который одинаково представляет каждая (достаточно большая) его подвыборка. Это более слабое свойство, чем стационарность, требующая, чтобы подвыборки имели одинаковые среднее значение и дисперсию. Эргодичность предполагает "эквивалентность" по содержанию информации в каждом срезе временного ряда без сохранения одинаковых статистических параметров (среднее значение и дисперсия). Детальное описание этого понятия приведено на сайте StackExchange (`https://perma.cc/5GW4-ZENE`).

Длина временного ряда

Еще один фактор, который следует учитывать при генерации признаков, — длина временного ряда. Некоторые признаки прекрасно работают в стационарных временных рядах, но становятся нестабильными по мере увеличения их длины, например минимальное и максимальное значения ряда. Для одного и того же базового процесса более длинный временной ряд, скорее всего, будет представлять более интенсивные максимальное и минимальное значения, чем те, которые наблюдаются в коротких временных рядах, созданных общим процессом, — только потому, что сбор его данных сопряжен с более широкими возможностями.

Знания предметной области

Знания предметной области должны выступать ключевым критерием при генерации признаков для временного ряда. Далее в этой главе описано несколько примеров того, как знания о предметной области влияют на генерацию ключевых признаков временных рядов, а в этом разделе мы рассмотрим только общие концепции.

Например, обрабатывая временные ряды физических показателей, необходимо определить количество признаков, которые имеют смысл для временной шкалы изучаемой системы, а также убедиться, что на выбранные признаки не оказывают чрезмерного влияния неустранимые критерии, обусловленные, скажем, ошибками измерения датчиков, а не свойствами базовой системы.

Например, представьте, что нужно обработать данные некого финансового рынка. Чтобы обеспечить финансовую стабильность, для рынка устанавливается максимальное изменение цены для каждого отдельного дня. Если цена меняется слишком сильно, то рынок закрывается. Вам решать, нужно ли в такой системе использовать признак, описывающий максимальную цену за выбранный день.

Независимые факторы

При выборе признаков немаловажную роль играют вычислительные ресурсы и связанные с ними средства хранения данных. Кроме того, большое значение имеет преследуемая вами цель. Признаки генерируются для дальнейшего использования в модели, что позволяет отказаться от дальнейшего хранения большого объема исходных данных? Или же они рассчитываются только для однократного анализа, а исходные данные подлежат обязательному хранению?

Цель, преследуемая при генерации признаков временных рядов, влияет на выбор их количества, а также на важность каждого из них. Она часто зависит от общего размера анализируемого набора данных. В небольшом наборе данных принимаемые решения обычно имеют незначительные последствия, но в случае очень крупных временных рядов задача расчета неправильно подобранных признаков часто сопряжена с напрасными ресурсными затратами — в первую

очередь, заключающимися в использовании дорогого вычислительного оборудования и времени, потраченного на написание программного кода.

Таким образом, изучив все требующие учета факторы, попробуйте составить список признаков и протестировать их на небольшом наборе данных, чтобы понять, насколько быстро или медленно выполняется анализ данных. Если небольшой набор данных анализируется слишком медленно, то необходимо сократить временной ряд, прежде чем проводить полный анализ. Аналогичным образом, прежде чем приступать к анализу полного набора данных, рассмотрите возможность проверки вычислительно затратных признаков на подмножестве исходных данных.

Источники признаков

Процесс выбора признаков для данных временных рядов ограничен только воображением исследователей, их навыками в написании кода и познаниями предметной области. Для их получения нужно придумать достаточно общий, но в то же время точный вычислительный способ количественного описания поведения временных рядов. Некоторые простые часто используемые признаки временных рядов представляют сводные статистические показатели, используемые в других приложениях.

- Среднее значение и дисперсия
- Максимальное и минимальное значения
- Разница между последним и первым значениями

Кроме того, отдельные важные признаки легче определить визуально, чем вычислить.

- Количество локальных максимумов и минимумов
- Гладкость временного ряда
- Периодичность и автокорреляция временных рядов

В общих случаях необходимо разработать реализационные определения, включающие описания всех возможных способов идентификации часто используемых признаков. Это позволит создать собственную программную библиотеку кодов генерации признаков, но вы всегда можете обратиться к сторонним библиотекам генерации признаков для данных временных рядов, особенно при работе с более требовательным к вычислительным ресурсам методикам. Постарайтесь найти как можно лучше реализацию с надежным и эффективным кодом.

Далее мы перейдем к рассмотрению библиотек получения признаков временных рядов, акцентируя внимание на извлечении широкого спектра признаков, генерируемых с помощью автоматизированных инструментов.

Библиотеки генерации признаков временных рядов с открытым исходным кодом

Исследователями было предпринято много попыток автоматизации процесса генерации признаков временных рядов, широко востребованных в задачах, характерных для разных предметных областей и дисциплин.

Модуль `tsfresh` в языке Python

Одним из наиболее успешных инструментов автоматической генерации признаков в языке Python считается модуль `tsfresh`, обеспечивающий создание большого универсального набора признаков. Чтобы получить представление о количестве поддерживаемых им признаков (`https://perma.cc/2RCC-DJLR`), рассмотрим наиболее общие их категории.

Описательные статистики

Они основаны на традиционных методологиях статистического исследования временных рядов, которые изучались в главе 6, в том числе:

- оценочное значение расширенного критерия Дики–Фуллера;
- коэффициент AR(k);
- автокорреляция для запаздывания, k.

Физические показатели нелинейности и сложности системы

Эта категория включает в себя следующие функции.

- `c3()`. Через нее рассчитывается ожидаемое значение $L^2(X^2)L(X)X$ (L — оператор запаздывания). Была предложена как мера нелинейности временного ряда.
- `cid_ce()`. Вычисляет квадратный корень суммы значений от 0 до $n - 2 \times$ *запаздывание* $(x_i - x_{i+1})^2$. Предложена в качестве меры сложности временного ряда.
- `friedrich_coefficients()`. Возвращает коэффициенты модели, предназначенной для описания *сложного* нелинейного движения.

Сжатие истории

К этой категории относятся такие функции.

- Сумма значений во временном ряду, которые встречаются более одного раза.
- Длина самой длинной последовательной подпоследовательности значений, которые больше или меньше среднего значения.

- Самое раннее появление во временном ряду минимального или максимального значения.

Модули, подобные `tsfresh`, призваны сэкономить рабочее время и выбрать наиболее подходящие реализации инструментов подбора признаков. Они также указывают на оптимальные способы описания данных, с которыми вам не приходилось встречаться ранее. Такие модули предоставляют много неоспоримых преимуществ при анализе данных с помощью надежных, хорошо проверенных инструментов с открытым исходным кодом.

- Не стоит изобретать новые методы при работе со стандартными признаками. Используя библиотеку общего назначения, можно быть уверенным в том, что другие пользователи будут подвергать сомнению только точность метода, но не код его реализации и используемые в нем инструменты.
- Подобные общедоступные пакеты являются не обычными библиотеками, а скорее фреймворками вычисления признаков. Например, `tsfresh` включает отдельный класс для вычисления признаков, который можно использовать для расширения возможностей библиотеки в собственных проектах с сохранением всех ее исходных преимуществ.
- Библиотека предназначена для взаимодействия с целевыми пользователями признаков, среди которых наиболее важным является пакет `sklearn`, позволяющий применять их в моделях машинного обучения.

Библиотека `tsfresh` имеет строгую техническую реализацию в том понимании, что многие полученные с ее помощью признаки вычисляются в результате анализа научных данных, собранных экспериментальным путем.

Платформа анализа временных рядов `cesium`

Не столь обширный, но при этом более доступный каталог автоматически генерируемых признаков создается с помощью библиотеки `cesium` (`http://cesium-ml.org/docs/index.html`). В настоящее время в документации упоминается следующий список поддерживаемых признаков (`http://cesium-ml.org/docs/feature_table.html`). Далее в этой главе мы выберем из него и проверим несколько, как нам кажется, наиболее интересных вариантов. Несмотря на то что общие категории признаков детально рассмотрены в исходном коде (`https://perma.cc/8HX4-MXBU`), мы все же подробно изучим следующие из них.

- Признаки, которые описывают общее распределение значений данных без учета временных связей между ними. Эта категория включает самые разнообразные признаки, которые, тем не менее, не учитывают временной контекст данных.

- Сколько локальных пиков в гистограмме данных?
- Какой процент точек данных находится в фиксированном окне значений, близких к медиане данных?

- Признаки, которые описывают распределение данных во времени.
  - Признаки, которые описывают распределение времени между измерениями и вычисляют статистику, аналогичную рассмотренной выше, но учитывающую распределение временны*х* разностей, а не значений данных.
  - Признаки, которые вычисляют вероятность того, что следующее наблюдение произойдет в течение *n* временны*х* шагов с учетом наблюдаемого распределения.
- Признаки, которые измеряют периодичность поведения данных в пределах временно*г*о ряда. Часто эти признаки связаны с периодограммой Ломба–Скарлга.

Периодограммы

Периодограмма временно*г*о ряда — это оценка величины вклада различных базовых частот во временной ряд. Идеальная периодограмма позволяет преобразовать временной ряд из фазового пространства "время–значение", с которым приходится работать чаще всего, в частотно-энергетическое пространство, в котором временной ряд рассматривается с точки зрения количества повторяющихся процессов, описываемых в различные моменты времени. Если вы не знакомы с такой концепцией, то ознакомьтесь с результатами ее визуализации, представленными на рис. 8.1.

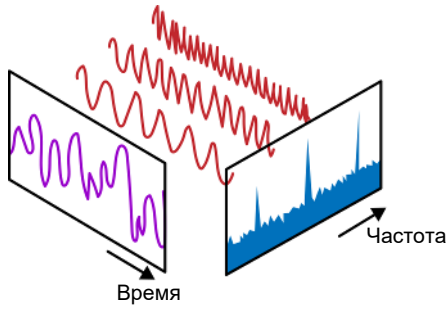


Рис. 8.1. *Данные одной и той же временно*й *области, в которой они выглядят как сумма синусоидального временно*г*о ряда. На оси частоты показано, что вся спектральная масса данных сосредоточена в окрестности трех частот, характеризующих синусоидальные процессы, порождающие данный временной ряд*

Существует много способов расчета периодограмм. Один из основных применяемых методов — *преобразование Фурье* автокорреляционной функции. Более интуитивно понятный подход к созданию периодограммы заключается в аппроксимации исходного набора синусоидальных кривых различной частоты по методу наименьших квадратов.

Именно второй вариант лег в основу широко известного метода Ломба–Скаргла, который создает периодограмму по нерегулярной выборке временных рядов в отличие от традиционной периодограммы, которая основана на регулярной выборке временных рядов. Ломб, а затем Скаргл, разработали методы изучения нерегулярных временных рядов и установили, что статистические свойства периодограммы, построенной для таких данных, совпадают с таковыми в традиционной периодограмме, построенной по регулярным выборкам временных рядов. Достижения Ломба и Скаргла позволили решить многие задачи естествознания, неизбежно требующие анализа нерегулярных временных рядов, в таких исследовательских областях, как климатология, астрофизика и геология.

Упомянутые выше признаки могут быть вычислены либо для всего временного ряда, либо только для скользящих или расширяющихся окон. Исходя из полученных в предыдущих главах знаний о механизмах выбора признаков для скользящих или расширяющихся окон, вы можете самостоятельно написать код реализации соответствующих вычислительных алгоритмов. У вас достаточно навыков, чтобы изучить документацию и разобраться в принципах работы библиотеки. В нашем примере функции скользящего окна используются для суммирования, а не очистки данных. Заметьте, что одни и те же методы анализа временных рядов применяются во многих различных, но одинаково полезных случаях.

Помимо функций генерации признаков, библиотека `cesium` включает много дополнительных инструментов. Например, она предоставляет веб-интерфейс для функций генерации признаков, а также команды интегрирования с инструментами пакета `sklearn`.

Протестировав все оговоренные пакеты на своих данных, можно заметить, что генерация признаков для временных рядов отнимает очень много времени. По этой причине необходимо заранее определиться с их количеством в анализируемых данных и также выяснить, в каких случаях автоматическая генерация признаков имеет смысл, а когда лучше отдать предпочтение ручному способу определения признаков.

Зачастую автоматизированное создание признаков с помощью программных библиотек сопряжено с большими вычислительными затратами и, учитывая их большое количество, такой подход не представляет ощутимых преимуществ. В каждой отдельной области знаний можно легко определить признаки, не представляющие интереса, определяемые шумом или не несущие информативной нагрузки. Не стоит тратить время и ресурсы на их вычисление — они совершенно бесполезны в последующем анализе и не добавят ясности в изучение данных. Несомненно, пакеты автоматической генерации признаков имеют определенную практическую ценность, но должны использоваться разумно, а не во всех без разбора наборах данных.

Пакет `tsfeatures` языка R

Библиотека `tsfeatures`, разработанная Робом Хиндманом и его коллегами, представляет программный пакет для языка R, предназначенный для генерации большого количества часто используемых и удобных признаков временных рядов. В документации к пакету упомянут ряд полезных функций.

- Функции `acf_features()` и `pacf_features()`, каждая из которых вычисляет количество связанных значений с учетом общей важности роли автокорреляции в поведении ряда. Согласно документации функция `acf_features()` возвращает следующие данные: "Вектор из 6 значений: первый коэффициент автокорреляции и сумма квадратов первых 10 коэффициентов автокорреляции исходного ряда, ряда первых разностей и ряда вторых разностей. Для сезонных данных также возвращается коэффициент автокорреляции для первого сезонного смещения".
- Функции `lumpiness()` и `stable()`, указанные как определяемые окнами, и `max_level_shift()` и `max_var_shift()` — определяемые скользящими окнами. В каждой из вариантов к значениям, измеренным в перекрывающихся (скользящих) или непересекающихся (мозаичных) окнах временного ряда, рассчитываются разности и статистики измерения вариативности.
- Функции `unitroot_kpss()` (`https://perma.cc/WF3Y-7MDJ`) и `unitroot_pp()` (`https://perma.cc/54XY-4HWJ`).

В пакете `tsfeatures` реализованы наработки различных академических проектов по изучению признаков временных рядов. Он является результатом консолидации проектов по совершенствованию инструментов генерации признаков временных рядов, востребованных в различных областях знаний. В частности, пакет включает следующие функции.

- Функция `compengine()` вычисляет признаки временных рядов, разработанных в рамках проекта comp-engine.org, которые эффективно

применяются для решения широкого спектра задач по анализу временных рядов во многих прикладных областях.[1]

- Ряд функций заимствован из пакета `hctsa`, исходно отвечающего за сравнительный анализа временных рядов в Matlab. Приведем только некоторые из них: `autocorr_features()`, `firstmin_ac()`, `pred_features()` и `trev_num()`. Описание этих и других функций можно найти в документации.

Документация к пакету `tsfeatures` (`https://perma.cc/Y8E9-9XCK`) включает наглядные демонстрации с инструкциями по использованию и описанием результатов применения каждой функции генерации признаков, а также список специальной литературы, в которой вопросы статистического и машинного обучения, основанного на поддерживаемых пакетом признаках временного ряда, рассматриваются более детально.

Примеры признаков, характерных для предметных областей

Другим источником вдохновения могут стать специфичные для выбранной предметной области признаки, определенные для различных временных рядов. Часто такие признаки вырабатывались в течение десятилетий и либо являются результатом опыта, даже если они не совсем понятны, либо основываются на научных знаниях о том, как работает система.

Далее мы рассмотрим примеры признаков, свойственные временным рядам двух важных предметных областей: финансовой и медицинской.

Технические индикаторы фондового рынка

Технические индикаторы фондового рынка, вероятно, являются наиболее широко документированными и формализованными наборами показателей, используемых для анализа временных рядов в отдельных предметных областях. Специалисты по экономике, которые изучают данные временных рядов в течение последнего столетия, располагают обширным списком признаков, используемых для количественной оценки финансовых временных рядов

[1] Если вы заинтересованы в более глубоком изучении рассматриваемых принципов, то обратите самое пристальное внимание на описание признаков набора временных рядов Catch22 (`https://perma.cc/57AG-V8NP`), которые находят широкое применение при решении самых разных задач, связанных с анализом и обработкой данных временных рядов. Сокращение набора признаков с более чем 4000 до 22 позволяет уменьшить время вычислений в 1000 раз — снижение точности классификации при этом составляет всего 7%. Также не лишне познакомиться с конвейером, который применялся для исследования данных и выбора в начальном наборе относительно независимых, но все же точных признаков.

и составления биржевых прогнозов. Даже не будучи специалистом по финансам, вы будете поражены размером этого списка — он как нельзя лучше показывает, что анализ данных в отдельно взятой предметной области можно выполнять на основе довольно большого количества информативных и очень специфических признаков.

Ниже приведен сильно сокращенный вариант списка таких признаков, но даже он позволяет понять, насколько специфичны и важны указанные показатели для финансовой отрасли. Учитывая их высокую сложность, не удивительно, что многие специалисты по финансам потратили на их изучение всю свою карьеру, пытаясь понять, как по таким "индикаторам" можно предсказать рост и падение финансовых рынков.

Индекс относительной силы (Relative Strength Index — RSI)

Этот показатель вычисляется как $100 - 100/(1 + RS)$, где RS — это отношение средней прибыли за период роста цен к среднему убытку за период снижения цен. Входным параметром в таком анализе выступает период изучения роста и снижения цен, так что для разных периодов просмотра можно получить разные значения RSI. Трейдеры разработали эмпирические правила определения пороговых значений RSI, указывающих на недооценку или переоценку активов относительно реальной стоимости. RSI также известен как "индикатор импульса", поскольку показывает направление движения финансового актива.

Схождение/расхождение скользящих средних (Moving Average Convergence/Divergence — MACD)

Этот индикатор основывается на трех временных рядах.

- Временной ряд MACD состоит из значений разности кратковременной (быстрой) и долгосрочной (медленной) экспоненциальными скользящими средними актива.
- Временной ряд "средних" представлен значениями экспоненциальных скользящих средних временного ряда MACD.
- Временной ряд "дивергенции" — это разность между временным рядом MACD и временным рядом "средних". Это значение, которое обычно используется для составления финансовых прогнозов. Другие временные ряды признака (MACD и "средних") обычно нужны только для получения временного ряда "дивергенции".

Денежный поток по Чайкину (Chaikin Money Flow — CMF)

Этот индикатор измеряет направление изменения цены. Чтобы вычислить его, необходимо выполнить следующие операции.

— Рассчитать множитель денежного потока:
((*Close* – *Low*) – (*High* – *Close*))/(*High* – *Low*).

— Вычислить объем денежных средств, который представляет собой дневной объем торгов, умноженный на множитель денежного потока.

— Просуммировать этот объем денежных средств за определенное количество дней и разделить его на объем за то же количество дней. Этот индикатор является "осциллятором", который задается в диапазоне значений от –1 до 1. Он указывает на "давление покупки" или "давление продажи", т.е. показатель направления рынка.

Как можно видеть из приведенного выше краткого описания технических признаков, вычисляемых из простых наборов финансовых данных, существует много способов описания временны*х* рядов. Финансовые рынки являются особенно богатым и хорошо изученным источником данных, который успешно применяется для генерации признаков.[2]

Медицинские временны*е* ряды

Здравоохранение — это еще одна область, в которой признаки временны*х* рядов имеют специфические для предметной области значения и даже названия. Как обсуждалось в главе 1, данные о состоянии здоровья выступают источником для огромного количества временны*х* рядов. Один из наиболее выразительных примеров — данные электрокардиограмм (ЭКГ) (рис. 8.2). Их чтение — это целая наука, в которой признаки устанавливаются врачами и в дальнейшем используются для чтения временны*х* рядов. Если вы планируете подбирать признаки для алгоритмов машинного обучения по данным ЭКГ, то вам, безусловно, следует начать с изучения уже известных вариантов, с описанием и назначением которых вас познакомит любой медицинский сотрудник должной квалификации.

Точно так же, анализируя временны*е* ряды с данными высокого разрешения об уровне глюкозы в крови, важно научиться распознавать шаблоны, применяемые для представления ежедневных данных, а также принципы их описания и маркировки медицинскими работниками (рис. 8.3).

[2] Обширный каталог признаков, которые можно использовать в методах машинного обучения, применяемых для анализа финансовых временны*х* рядов, приведен в блоге Kaggle (`https://perma.cc/Q84C-44XD`). К сожалению, описанный в нем код не получил широкого признания, хотя и выступает отличным примером исчерпывающе правильного подхода к выбору релевантных признаков в рассматриваемой предметной области.

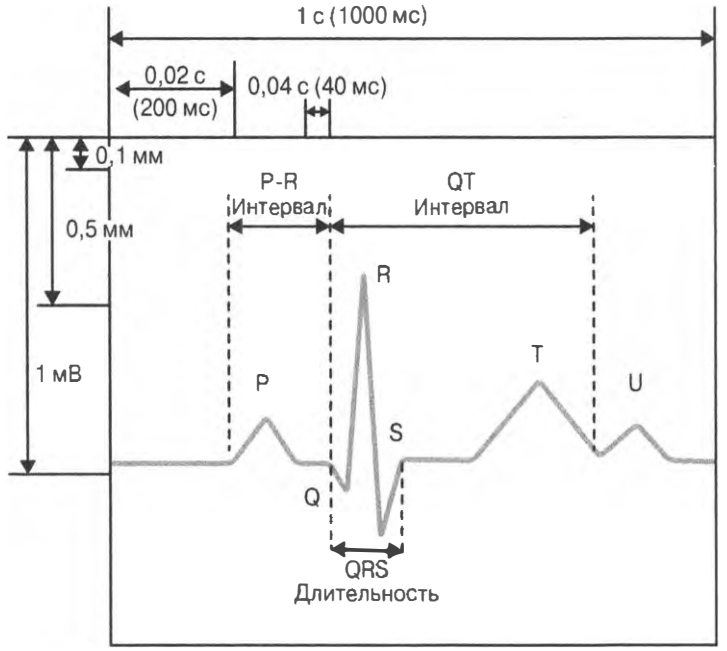


Рис. 8.2. *Примеры признаков медицинских временных рядов, которые используются при чтении ЭКГ*

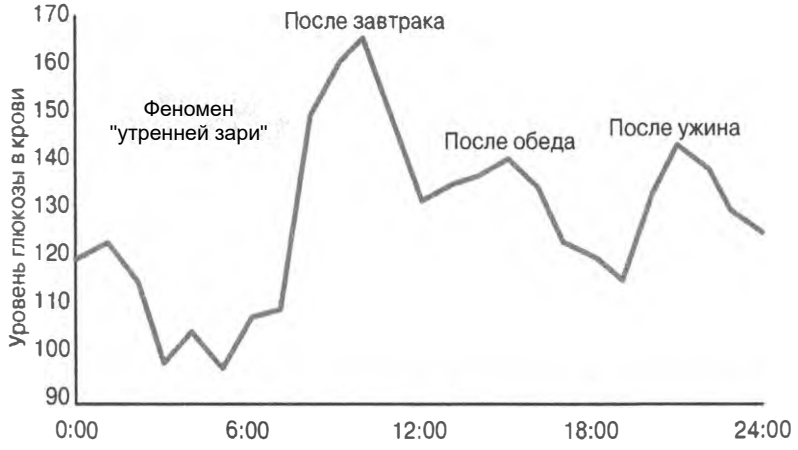


Рис. 8.3. *Временной ряд ежедневных данных об уровне глюкозы в крови характеризуется четырьмя признаками, по которым медицинские работники судят о состоянии больного. Один из них не связан с режимом питания и известен под названием феномен "утренней зари". Он оказывает непосредственное влияние на остальные признаки*

Оба временных ряда, показанных на рис. 8.2 и 8.3, являются хорошими примерами данных, которые достаточно точно описываются величинами локальных максимумов или расстояний между ними. Следовательно, используя правильные инструменты библиотек `tsfresh` и `cesium` и учитывая особенности заданной предметной области, можно легко подобрать соответствующие признаки для целевых медицинских временных рядов.

Отбор автоматически сгенерированных признаков

Предположим, вам удалось автоматически сгенерировать большое количество признаков, описывающих набор данных большого временного ряда. Скорее всего, вам не удастся оценить их все при первом же просмотре данных, поэтому всегда не лишне дополнить функцию автоматической *генерации* признаков возможностью их автоматического *отбора*. К одним из лучших инструментов отбора признаков относится алгоритм FRESH, который реализован в описанном ранее пакете `tsfresh`. Этот алгоритм выбирает релевантные признаки на основе масштабируемых критериев проверки гипотез.

Необходимость в разработке алгоритма FRESH была вызвана постоянно увеличивающимся объемом данных временных рядов, зачастую подлежащих распределенному хранению с целью последующего распараллеливания вычислений. Алгоритм оценивает значимость каждого входного признака по отношению к целевой переменной методом вычисления *p-значения* для каждого из них. После этого *p*-значения каждого признака оцениваются в совокупности с помощью *процедуры Бенджамини–Екутиели*, учитывающей такие входные параметры, как уровень допустимых погрешностей, и указывающей, какие из признаков следует сохранить для дальнейшего использования. Процедура *Бенджамини–Екутиели* — это метод ограничения количества ложных срабатываний, обнаруженных во время проверки гипотез, используемый для получения *p*-значений на начальном этапе алгоритма FRESH.

Самостоятельная реализация таких процедур — это довольно сложная задача, но воспользовавшись пакетом `tsfresh`, ее можно решить всего в несколько строк кода. В следующем примере использован шаблон кода, приведенный в документации к модулю. Вначале нужно загрузить данные временных рядов, относящиеся к сбоям в работе робота.

```
## Python
>> from tsfresh.examples.robot_execution_failures import
                    download_robot_execution_failures,
                    load_robot_execution_failures
>> download_robot_execution_failures()
>> timeseries, y = load_robot_execution_failures()
```

Признаки извлекаются без дополнительного указания, поэтому пакет автоматически вычисляет их все. В этом смысле он поступает вразрез с предостережениями, приведенными ранее в этой главе, оставаясь достаточно беспечным относительно экономии вычислительных ресурсов. В тестовом наборе данных не слишком много точек данных, но вы, вероятнее всего, откажетесь от полной их обработки, предварительно не сократив до разумного количества.

```
## Python
>> from tsfresh import extract_features
>> extracted_features = extract_features(timeseries,
                                         column_id = "id",
                                         column_sort = "time")
```

Хотя пакет `tsfresh` позволяет указывать вычисляемые признаки, в настоящем примере мы будем рассматривать все возможные варианты. Заметьте, что всегда можно указать параметры для тех признаков, которые должны приниматься в расчет при вычислении. Все возможности детально описаны и проиллюстрированы в документации.

Проведя полное извлечение для данных — в нашем случае включенных в пакет `tsfresh` — можно получить огромное количество признаков .

```
## Python
>> extracted_features.columns
Index(['F_x__abs_energy', 'F_x__absolute_sum_of_changes',
       'F_x__agg_autocorrelation__f_agg_"mean"',
       'F_x__agg_autocorrelation__f_agg_"median"',
       'F_x__agg_autocorrelation__f_agg_"var"',
       'F_x__agg_linear_trend__f_agg_"max"__chunk_len_10__attr_
                                                    "intercept"',
       'F_x__agg_linear_trend__f_agg_"max"__chunk_len_10__attr_
                                                     "rvalue"',
       'F_x__agg_linear_trend__f_agg_"max"__chunk_len_10__attr_
                                                      "slope"',
       'F_x__agg_linear_trend__f_agg_"max"__chunk_len_10__attr_
                                                      "stderr"',
       'F_x__agg_linear_trend__f_agg_"max"__chunk_len_50__attr_
                                                      "intercept"',
       ...
       'T_z__time_reversal_asymmetry_statistic__lag_1',
       'T_z__time_reversal_asymmetry_statistic__lag_2',
       'T_z__time_reversal_asymmetry_statistic__lag_3',
       'T_z__value_count__value_-inf', 'T_z__value_count__value_0',
       'T_z__value_count__value_1', 'T_z__value_count__value_inf',
       'T_z__value_count__value_nan', 'T_z__variance',
        'T_z__variance_larger_than_standard_deviation'],
     dtype='object', name='variable', length=4764)
```

Нами получено 4764 столбца значений. Это количество признаков гораздо больше того, которое можно вычислить вручную, но при работе с реальными наборами данных на их вычисление уйдет заметно больше времени. Решившись на развертывание настолько большого набора признаков, постарайтесь реалистично оценить как вычислительную мощность рабочего оборудования, так собственные способности и скорость анализа полученных результатов. Помните, что выбросы данных во временны*х* рядах могут оказаться особенно сложными и неудобными для последующего анализа. Обязательно убедитесь, что избранные вами признаки лишены их.

Несмотря на способность алгоритма FRESH учитывать зависимость между признаками, разобраться в принципах его работы зачастую весьма непросто. Следовательно, во многих случаях можно обратиться к более традиционному и простому для понимая методу — *рекурсивного удаления признаков* (Recursive Feature Elimination — RFE). Часто метод RFE применяется как дополнение алгоритма FRESH, позволяя упростить понимание различий между признаками, подобранными и отвергнутыми алгоритмом FRESH.

Метод RFE основан на поэтапном подходе к выбору признаков, в рамках которого элементы постепенно исключаются из более полной модели, что приводит к постепенному ее упрощению до размера с минимальным набором элементов, количество которых определяется в самом начале процедуры выбора.

Этот метод известен как обратный отбор, потому что вы начинаете с наиболее сложной модели и возвращаетесь "обратно" к более простой модели. В противоположность ему в прямом отборе признаки добавляются постепенно, пока не будет получено максимальное возможное их количество или достигнуто предельное значение заранее оговоренного критерия остановки.

Метод RFE можно использовать как для выбора признаков, так и для оценки их важности. В следующем эксперименте объединим 10 случайно выбранных признаков из списка, предоставленного алгоритмом FRESH, с 10 случайно выбранными элементами из списка признаков, отклоненных алгоритмом FRESH.

```
## R
>> x_idx = random.sample(range(len(features_filtered.columns)), 10)
>> selX = features_filtered.iloc[:, x_idx].values
>> unselected_features = list(set(extracted_features.columns)
                         .difference(set(features_filtered.columns)))
>> unselected_features = random.sample(unselected_features, 10)
```

```
>> unsel_x_idx = [idx for (idx, val) in enumerate(
          extracted_features.columns) if val in unselected_features]
>> unselX = extracted_features.iloc[:, unsel_x_idx].values
>> mixed_X = np.hstack([selX, unselX])
```

Применим метод RFE к этому набору из 20 признаков, чтобы получить представление об их ранжировании (важности) для набора данных и модели, используемой в методе RFE.

```
>> svc = SVC(kernel="linear", C=1)
>> rfe = RFE(estimator=svc, n_features_to_select=1, step=1)
>> rfe.fit(mixed_X, y)
>> rfe.ranking_
array([ 9, 12,  8,  1, 2,  3,  6,  4, 10, 11,
       16,  5, 15, 14, 7, 13, 17, 18, 19, 20])
```

Итак, мы получили относительные рейтинги 20 признаков, "прогнанных" через алгоритм RFE. Ожидается, что первые 10 признаков, выбранных алгоритмом FRESH, будут иметь более высокий рейтинг, чем те, что были отвергнуты им. В значительной степени это так, но не полностью. Например, легко заметить, что во второй половине массива с рейтингами отвергнутых признаков, указаны 5- и 7-й наиболее важные признаки из всех 20. За исключением этого в отсутствие идеального совпадения полученные результаты согласуются с исходными предположениями.

Как видите, метод RFE можно использовать для дальнейшего отбора единожды извлеченных признаков. Кроме того, он прекрасно подходит для проверки работоспособности алгоритма FRESH при точной настройке его входных параметров или количества генерируемых признаков.

Обратите внимание, что алгоритм FRESH по умолчанию работает без указания параметров, поэтому количество генерируемых признаков — это лучшее средство, с помощью которого можно повлиять на его производительность. Другой не менее важный параметр, который приходится устанавливать в алгоритме FRESH — `fdr_level`, — определяет процент ненужных признаков, которые отвергаются по завершении процедуры. По умолчанию этот параметр равен `.05`, но можно задать намного большее значение, чтобы повысить качество отбора признаков, особенно при их генерировании в большом количестве без предварительного учета того, попадают ли они в область рассмотрения.

Заключение

В этой главе мы обсудили причины, побуждающие исследователей обращаться к признакам в анализе данных, а также рассмотрели простой пример использования признаков для преобразования небольшого временного ряда

в ограниченный набор чисел, не уступающий по информативности исходному набору данных. Кроме того, были рассмотрены два программных модуля Python, предназначенных для автоматического выбора и генерации признаков в данных временных рядах, которые способны извлечь несколько тысяч признаков из одного временного ряда. Так как в дальнейшей работе будут полезны не все автоматически полученные признаки, их последующий отбор выполнялся с использованием специальных методик, позволяющих отсеять зашумленные и неинформативные варианты, — только после этого они становятся пригодными к передаче в конвейер анализа данных.

Генерация признаков позволяет достигнуть ряда целей.

- Предоставить сводные данные о временных рядах в формате, позволяющем применять к ним алгоритмы машинного обучения, большинство из которых предназначены для обработки наборов признаков для одной точки данных, а не всего временного ряда.
- Сжать данные временных рядов таким образом, чтобы свести временные наблюдения к нескольким числам и качественным показателям. Такой подход применяется не только в анализе, но и в хранении временных рядов в более кратком и удобочитаемом формате в случаях передачи на хранение полного временного ряда.
- Сформировать набор показателей для описания и определения сходства данных, которые получены или измерены в различных условиях. Обобщая данные в более широком плане, можно добиться их лучшего сопоставления, что представляется очень сложной задачей при прочих условиях.

В главе 9 мы будем использовать сгенерированные признаки для подготовки к передаче данных в алгоритмы машинного обучения, работа которых заключается в классификации и прогнозировании по признакам временных рядов, а не по необработанным данным базовых временных рядов.

Дополнительные источники

- Анализ временных рядов по признакам

  Ben D. Fulcher, "Feature-Based Time-Series Analysis," eprint arXiv:1709.08055, 2017, `https://perma.cc/6LZ6-S3NC`

  Обзорная статья, написанная на доступном языке и принадлежащая перу разработчика пакета `tsfresh` в Matlab. В ней приведена обширная таксономия различных видов признаков, реализуемых для данных временных рядов. Кроме того, в ней наглядно проиллюстрированы

категории всех обсуждаемых признаков. Акцент сделан на описании важности использования автоматически сгенерированных признаков в исследовании данных — с расширением на их применение в анализе временных рядов.

- Отбор признаков

  Maximilian Christ et al., "Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh—A Python package)," Neurocomputing 307 (2018): 72–7, `https://oreil.ly/YDBM8`

  В этой статье приведен обзор пакета `tsfresh` для языка Python и алгоритма FRESH, рассматриваемых в этой главе. Приведены сведения о вычислительной эффективности отдельных признаков и рассмотрены некоторые шаблоны их использования.

  Maximilian Christ et al., "Distributed and Parallel Time Series Feature Extraction for Industrial Big Data Applications," paper presented at ACML Workshop on Learning on Big Data (WLBD), Hamilton, NZ, November 16 2016, `https://arxiv.org/pdf/1610.07717.pdf`

  Это подробное техническое описание алгоритма FRESH содержит более детальную информацию о том, как работает алгоритм и почему он находит применение в промышленных временных рядах, а также обширный список справочных материалов в разделе ссылок.

- Признаки для отдельных предметных областей

  Wikipedia, "Technical Analysis for Financial Markets," `https://perma.cc/8533-XFSZ`

  Статья в Википедии, в которой приведена история разработки методов технического анализа финансовых рынков, а также содержится описание большого количества популярных индикаторов.

  Amjed S. Al-Fahoum and Ausilah A. Al-Fraihat, "Methods of EEG Signal Features Extraction Using Linear Analysis in Frequency and Time-Frequency Domains," ISRN Neuroscience 2014, no. 730218 (2014), `https://perma.cc/465U-QT53`

  Здесь рассматривается пример тестирования стандартных признаков временных рядов данных ЭЭГ (электроэнцефалограмм). Существует целая академическая индустрия генерации признаков для описания медицинских временных рядов, и этот документ включает доступное описание принципов выполнения этой задачи.

Juan Bautista Cabral et al., "From FATS to Feets: Further Improvements to an Astronomical Feature Extraction Tool Based on Machine Learning, Astronomy and Computing 25 (2018), `https://perma.cc/8ZEM-Y892`

В этой статье приведен обзор последнего издания пакета Python, предназначенного для извлечения признаков из данных астрономических временных рядов. Рассчитанная на специалистов по астрономии, знакомых с целевым программным пакетом, она будет полезной всем, кто планирует использовать специальное программное обеспечение для генерации признаков, а также для понимания вызовов и задач, стоящих перед специалистами по анализу астрономических временных рядов.

Alvin Rajkomar et al., "Supplementary Information for Scalable and Accurate Deep Learning for Electronic Health Records," npj Digital Medicine 1, no. 18 (2018), `https://perma.cc/2LKM-326C`

Комментарии к весьма интересной статье, опубликованной Google, в которой проиллюстрированы результаты успешного применения методов глубокого обучения для анализа электронных медицинских записей. Содержат подробную информацию о принципах приведения медицинских записей к количественным входным признакам, которые доступны для обработки нейронной сетью. Если вы не знакомы с технологиями глубокого обучения, то приступайте к изучению этого документа только после изучения главы 10. Кроме того, перед знакомством с ним неплохо получить представление о вложениях пространств для глубокого обучения (`https://perma.cc/3KAZ-9A3Y`)

- Периодограммы

"The Periodogram," lecture notes, Eberly College of Science, Pennsylvania State University, `https://perma.cc/5DRZ-VPR9`

Заметки из лекционного курса по прикладному анализу временных рядов Университета Penn State, знакомящие студентов с понятием периодограммы, принципами ее интерпретации и построения с помощью инструментов языка R.

Jacob T. VanderPlas, "Understanding the Lomb-Scargle Periodogram," Astrophysical Journal Supplement Series 236, no. 1 (2018), `https://arxiv.org/pdf/1703.09824.pdf`

Эта обширная статья, знакомящая читателей с общими концепциями оценки периодических зависимостей и, в частности, с методом Ломба–Скаргла.

ГЛАВА 9

Машинное обучение в анализе временных рядов

В этой главе мы рассмотрим несколько примеров применения методов машинного обучения к анализу временных рядов. Это относительно новая дисциплина науки о данных, но она демонстрирует многообещающие результаты. В отличие от статистических моделей, которые мы изучали в главах 7 и 8, методы машинного обучения, с которыми нам предстоит познакомиться, изначально не разрабатывались для анализа временных рядов, но оказались очень полезными для работы с ними.

Переход к технологиям машинного обучения задуман как продолжение изысканий по составлению прогнозов, проводимых в предыдущих главах. До настоящего момента наше внимание было сконцентрировано на статистических моделях прогнозирования временных рядов. При разработке таких моделей мы сформулировали основную теорию описания динамики временного ряда и статистических способов представления шума и неопределенности в его поведении. Впоследствии на основе рассчитанной гипотетической динамики процесса были составлены прогнозы и оценена степень их неопределенности. При таком подходе идентификация и оценка параметров модели требовали предельно тщательной проработки — они должны наилучшим образом описывать динамику изменения данных.

В этой главе рассматриваются вычислительные методики, в которых не учитываются характеристики базового процесса и правила, описывающие его поведение. Вместо этого в них поведение системы описывается согласно шаблонам, которые распознаются с помощью методов, сходных с применяемыми для предсказания целевой величины, например классификационных подписей в данных временного ряда. В ней мы также рассмотрим методы обучения без учителя на примере кластеризованных данных временных рядов.

Здесь для описания принципов прогнозирования и классификации применяются древовидные модели, а кластеризация представлена как специальная форма классификации. В методах классификации деревьев крайне важно определиться с признаками временного ряда, поскольку в них, в отличие, например, от ARIMA, не учитываются временные изменения в поведении данных.

В случае кластеризации и метрической классификации будет показано, что в качестве входных данных можно использовать как признаки, так и исходные

временны*е* ряды. Чтобы подать на вход сам временной ряд, нужно рассчитать метрику расстояния, известную как *динамическая трансформация времени*, которая в случае применения к временны*м* рядам позволяет сохранить в данных полный хронологический набор информации, не прибегая к ее представлению ограниченным набором признаков.

Классификация временны*х* рядов

В этом разделе рассматривается пример представления признаками необработанных временны*х* рядов данных электроэнцефалограммы (ЭЭГ), которые в дальнейшем подвергаются обработке с помощью алгоритмов машинного обучения. После извлечения признаков из временно*го* ряда ЭЭГ для классификации данных обратимся к деревьям решений.

Выбор и генерация признаков

В предыдущей главе мы посвятили много времени, описывая цели, преследуемые при генерации признаков временны*х* рядов. Кроме того, в ней рассматривался небольшой пример получения признаков для данных временны*х* рядов с помощью пакета `tsfresh`. В этой главе признаки для временны*х* рядов будут генерироваться с помощью другого, также обсуждаемого ранее, программного пакета `cesium`.

Одно из главных преимуществ пакета `cesium` заключается в его снабжении множеством полезных наборов временны*х* рядов, включая данные ЭЭГ, первоначально полученных в исследовательской работе 2001 года. В ней детально описаны все операции по подготовке данных. Нам же достаточно знать, что все пять категорий временны*х* рядов ЭЭГ из встроенного набора данных представляют собой фрагменты равной длины, вырезанные из непрерывных временны*х* отсчетов избранных ЭЭГ. В частности, среди них вы найдете следующие временны*е* ряды.

- ЭЭГ-записи здоровых людей с открытыми и закрытыми глазами (две отдельные категории)
- ЭЭГ-записи пациентов с эпилепсией в периоды без судорог для двух областей мозга, не связанных с проявлением судорог (две отдельные категории)
- Внутричерепная запись ЭЭГ во время судорог (одна категория)

Набор данных загружается с помощью специальной функции, также включенной в пакет `cesium`.

```
## Python
>>> from cesium import datasets
>>> eeg = datasets.fetch_andrzejak()
```

Всегда полезно предварительно просмотреть несколько примеров анализируемых данных, чтобы получить представление о том, как такие временны́е ряды лучше всего классифицировать.

```
# Python
>>> plt.subplot(3, 1, 1)
>>> plt.plot(eeg["measurements"][0])
>>> plt.legend(eeg['classes'][0])
>>> plt.subplot(3, 1, 2)
>>> plt.plot(eeg["measurements"][300])
>>> plt.legend(eeg['classes'][300])
>>> plt.subplot(3, 1, 3)
>>> plt.plot(eeg["measurements"][450])
>>> plt.legend(eeg['classes'][450])
```

На графиках заметны отчетливые различия между классами ЭЭГ (рис. 9.1). И это не удивительно: графики ЭЭГ указывают на активность разных частей мозга при выполнении разных действий — как у здоровых субъектов, так и у пациентов с эпилепсией.

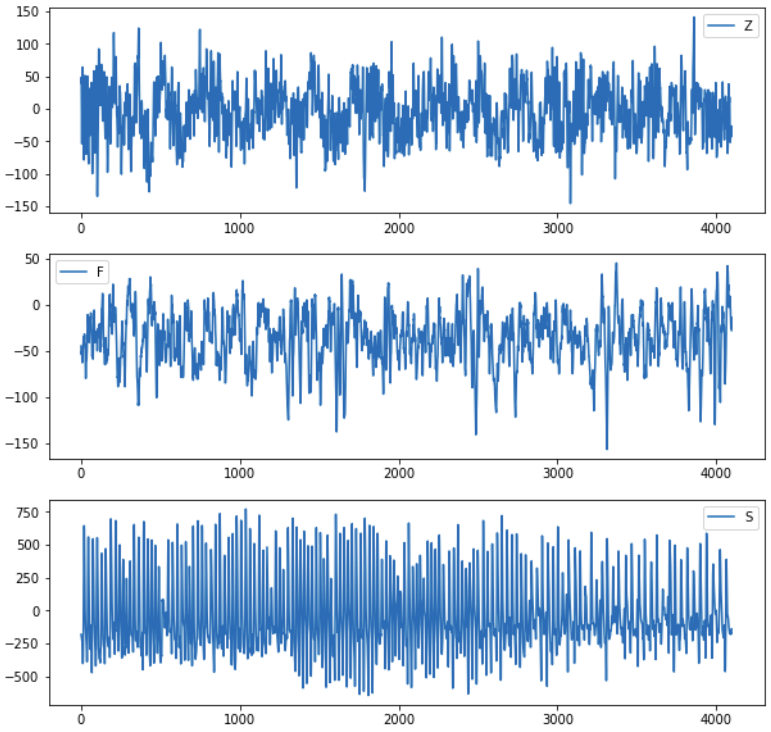


Рис. 9.1. *Графики трех случайных выборок из набора данных ЭЭГ. Это выборки независимых измерений, а не результаты исследований, проводимых одновременно в разных частях одного и того же мозга. Представленные данные получены для разных пациентов в разное время*

Результат визуализации может послужить руководством к генерации признаков. В частности, классы *Z* и *F*, скорее всего, содержат меньше искаженных данных, чем класс *S*. Кроме того, как видно по оси *y*, каждый класс характеризуется своим, отличным от остальных, диапазоном значений. Это указывает на то, что в качестве признака может использоваться амплитуда. Помимо того, графики различаются не только амплитудой значений, но и общим распределением точек, которые различаются у всех трех классов. Таким образом, в дальнейшем анализе мы будем опираться на эти и некоторые другие признаки, после чего напишем код их генерации.

Визуализация и машинное обучение

Данные ЭЭГ отлично визуализируются на двумерной гистограмме. В частности, такая гистограмма предельно точно представляет данные каждого типа временны́х рядов и подтверждает (или опровергает) репрезентативность исследуемых признаков. В нашем случае она позволяет установить, будет ли амплитуда или наклон прямой устойчиво различаться для разных классов.

Кроме того, нелишне построить одномерные гистограммы — по одной на класс. Также рассмотрите возможность проведения дополнительных исследований, заключающихся в ядерной оценке плотности для каждого класса.

Визуально ядерная оценка плотности схожа с гистограммой, но строится так, чтобы обладать специальными свойствами, такими как непрерывность и гладкость, эффективно приводя необработанные данные гистограммы к плавному, непараметрическому оценочному распределению для базовой переменной. Википедия предлагает хорошее описание ядерной оценки плотности, которая может быть выполнена с помощью таких программных инструментов, как пакет `scikit-learn` в языке Python или функция `density()` пакета `stats` в языке R.

Сгенерируем признаки с помощью пакета `cesium`.

```
## Python
>>> from cesium import featurize.featurize_time_series as ft
>>> features_to_use = ["amplitude",
>>>                    "percent_beyond_1_std",
>>>                    "percent_close_to_median",
>>>                    "skew",
>>>                    "max_slope"]
>>> fset_cesium = ft(times = eeg["times"],
>>>                  values = eeg["measurements"],
>>>                  errors = None,
>>>                  features_to_use = features_to_use,
>>>                  scheduler = None)
```

Этот код создает признаки, представленные в таблице на рис. 9.2 (снимок экрана виртуального ноутбука Jupyter).

```
fset_cesium.head()
```

feature	amplitude	percent_beyond_1_std	percent_close_to_median	skew	max_slope
channel	0	0	0	0	0
0	143.5	0.327313	0.505004	0.032805	11107.796610
1	211.5	0.290212	0.640469	-0.092715	20653.559322
2	165.0	0.302660	0.515987	-0.004100	13537.627119
3	171.5	0.300952	0.541128	0.063678	17008.813559
4	170.0	0.305101	0.566268	0.142753	13016.949153

Рис. 9.2. *Числовые значения признаков, сгенерированных для первых нескольких выборок анализируемого набора данных*

Заметьте, что многие значения не нормализованы. Это следует учитывать при использовании методов, которые требуют передачи исключительно нормализованных входных данных.

Нужно четко понимать, на что именно указывают признаки, и уметь сопоставлять ожидания с результатами, полученными с помощью пакета `cesium`. Чтобы проиллюстрировать важность проверки полученных результатов и отсутствия противоречий со здравым смыслом, давайте проанализируем параметр `percent_beyond_1_std` для единственной выборки временного ряда.

```
## Python
>>> np.std(eeg_small["measurements"][0])
40.411
>>> np.mean(eeg_small["measurements"][0])
-4.132
>>> sample_ts = eeg_small["measurements"][0]
>>> sz = len(sample_ts)
>>> ll = -4.13 - 40.4
>>> ul = -4.13 + 40.4
>>> quals = [i for i in range(sz) if sample_ts[i] < ll or
                                    sample_ts[i] > ul ]
>>> len(quals)/len(ser)
0.327 ## Проверка сгенерированного признака (см. рис. 9.2)
```

Эргодические признаки

При выборе признаков временного ряда из автоматически сгенерированных вариантов обращайте внимание только на *эргодические* признаки. Они предполагают, что каждое из измерений будет сходиться к устойчивому значению по мере сбора большего

количества данных для одного и того же процесса. Примером, где такое условие не выполняется, может быть случайное блуждание, в котором измерение среднего значения для процесса не имеет смысла, а потому не будет эргодическим. Среднее значение случайного блуждания не будет сходиться к конкретному значению.

В данных ЭЭГ, представленных на графике, различные подвыборки из любого заданного временного ряда явно сопоставимы, а сам ряд слабо стационарен, поэтому сгенерированные признаки имеют смысл.

Необходимо иметь возможность проверки используемых признаков. Это вопрос ответственного анализа. Не стоит предоставлять алгоритму информацию, которую нельзя понять, объяснить и проверить.

Не злоупотребляйте библиотеками, предназначенными для генерации признаков. Написать собственный код генерации признаков совсем не трудно. Если вы работаете с данными предметной области, в которой определенный набор признаков используется неоднократно и в одних и тех же комбинациях, то постарайтесь написать собственный код реализации алгоритма (даже если исходно использовали инструменты программных библиотек).

Вы сможете оптимизировать собственное решение намного лучше, чем разработчики равнозначного универсального пакета. Например, располагая несколькими признаками, основанными на вычислении среднего значения временного ряда, можно создать код однократного вычисления такого значения и не дублировать его для каждого отдельно рассчитываемого признака.

Деревья принятия решений

Деревья принятия решений имитируют принятые у людей способы принятия решений — пошагово и в высшей степени нелинейно. Они отражают принципы принятия нами сложных решений: одно за раз, изучая то, как текущая переменная влияет на полученное решение, затем учитывается следующий вариант — очень похоже на блок-схему.

С большой вероятностью вам уже доводилось работать с деревьями принятия решений или вы знакомы с их общей концепцией. Если это не так, то поставьте изучение материала главы на паузу, сделайте перерыв и ознакомьтесь с основными справочными сведениями о деревьях принятия решений (`https://perma.cc/G9AA-ANEN`).

На рис. 9.3 показан простой пример дерева принятия решений, которое можно использовать для оценки веса тела любого человека.

Рис. 9.3. *В этом простом дереве ряд логических ветвей используется для предсказания веса человека. Это грубая модель, но она иллюстрирует нелинейный и изменчивый подход, который можно реализовать в регрессионной задаче с помощью простой модели дерева принятия решений*

Существует множество примеров человеческого поведения, следующего принципам дерева принятия решений при анализе данных временных рядов. Например, независимые трейдеры на фондовом рынке также получают доступ к техническим индикаторам, но, скорее всего, изучают их пошагово в некой иерархически выверенной манере, подобно тому, как это принято в дереве нелинейных решений — сначала определяя направление движения рынка по одному техническому индикатору, а затем, получив один из вариантов ответа, исследуя для него временные изменения волатильности. По всей видимости, они удерживают в памяти некое подобие древовидной структуры принятия решений, которую они используют, чтобы составлять достаточно точные прогнозы о направлении движения рынка.

Подобным образом медицинские сотрудники читают ЭЭГ и ЭКГ. Сначала они ищут хотя бы один из известных признаков, и только затем переходят к поиску следующего, последовательно учитывая все возможные факторы. При наличии только одного признака и отсутствии другого диагноз будет отличаться от утверждаемого в противном случае и, следовательно, другого прогноза относительно состояния пациента.

Далее рассмотрим признаки, сгенерированные для данных ЭЭГ, применяемых в качестве входных данных в двух разных методах дерева принятия решений — случайного леса и градиентного бустинга деревьев, — каждый из которых используется в задачах классификации. Наша задача будет заключаться в классификации исследуемых данных ЭЭГ исключительно на основе признаков, сгенерированных из необработанных данных.

Случайный лес

Случайный лес — это модель, в которой используется не одно, а множество деревьев принятия решений. Результатом усреднения выходных данных таких деревьев будет классификация или регрессия данных. Случайные леса основаны на понятии "мудрости толпы", где "толпа" представлена большим количеством простых моделей, ни одна из которых сама по себе не может быть достаточно хороша, но все вместе они часто превосходят даже самое точное, но единственное дерево решений.

Идея собрать коллекцию моделей для составления точного прогноза, а не просто найти единственную "лучшую" модель, была высказана еще в 1969 году в статье *Combination of Forecasts* (Комбинации прогнозов) — исследовательской работе двух именитых статистов, Дж. М. Бейтса и С. В. Дж. Грейнджер. В этом документе показано, что объединение двух отдельных прогнозов данных о пассажирах авиакомпании может привести к получению модели с более низкой среднеквадратичной ошибкой, что оказывается совершенно неожиданным и в то же время неубедительным результатом. Тем не менее молодое поколение аналитиков, часто прибегающих при исследовании данных к услугам алгоритмов машинного обучения (в противоположность статистическим методам), наоборот, считают такой подход интуитивно понятным, а не обескураживающим. Очень быстро случайный лес деревьев стал наиболее частым решением, применяемым к большинству задач прогнозирования.

Случайный лес строится в соответствии с параметрами, определяющими количество деревьев для обучения, а также максимально допустимую глубину этих деревьев. Впоследствии обучение каждого отдельного дерева выполняется на случайной выборке данных и их признаков. Деревья, как правило, параметризуются так, чтобы получить максимально простую структуру и избежать эффекта переобучения, а модель должна уметь усреднять многие другие модели, ни одна из которых не является особенно хорошей, но все они достаточно общие, чтобы избежать "ловушек", таящихся в данных.

Как упоминалось ранее, признаки, рассчитанные для каждой выборки временного ряда, вводятся в модель в качестве результатов обучения. Теоретически можно придумать способы ввода в модель не вычисленных признаков, а необработанных временных рядов, но их разработка связана с рядом трудностей.

- Временные ряды неодинаковой длины обрабатывать очень сложно.
- Использование большого количества входных данных (равного или близкого к числу временных шагов) приведет к созданию невероятно сложных и дорогостоящих для вычисления, а также обучения моделей.
- Если предположить, что ни один временной шаг не является важным (поскольку каждый временной шаг соответствует хотя бы одному признаку),

то модель будет характеризоваться высоким уровнем шума и низким уровнем полезного сигнала для обучения, что будет проявляться на каждом шаге ввода данных.

Таким образом, случайные леса не являются хорошим инструментом для исследования данных временных рядов в необработанном виде, но становятся полезными после сжатия данных в сводные признаки. Перечислим несколько подтверждающих факторов.

- С точки зрения эффективности/вычислительных затрат принято считать, что очень длинные временные ряды всегда можно представить набором признаков и подобрать для них модель с разумной точностью.
- Очень важно добиться того, чтобы случайный лес понижал риск переобучения. Как обсуждалось ранее, переобучение является особенно частой проблемой анализа временных рядов из-за наличия выраженной взаимосвязи между методами обучения и прогнозирования. Преднамеренно упрощенная методология позволяет устранить некоторые из потенциальных трудностей.
- Случайные леса могут быть особенно подходящими для данных временных рядов, для которых не предусмотрена рабочая модель или гипотеза о механике базового процесса.

Специалистам по анализу удалось добиться заметных успехов только в развертывании случайных лесов в задачах классификации временных рядов, но не в прогнозирования данных. В противоположность им рассматриваемый в следующем разделе метод, градиентный бустинг деревьев, получил широкое распространение при решении задач обоих типов.

Градиентный бустинг деревьев

Бустинг является еще одним способом построения ансамбля предикторов. Бустинг, предполагающий последовательный способ построения модели, основан на предположении, что более поздние модели исправляют ошибки более ранних моделей, а несоответствие данных в более ранних моделях должно в большей степени учитываться и устраняться более поздними моделями.

Градиентный бустинг деревьев оказался особенно успешным методом исследования временных рядов, о чем свидетельствует его популярность среди участников соревнований по анализу данных, проводимых в течение нескольких последних лет.

Библиотека `XGBoost` строит деревья принятия решений последовательным образом так, что каждое следующее дерево стремится предсказать остатки комбинации предыдущих деревьев. Так, например, первое дерево, построенное

библиотекой XGBoost, будет пытаться сопоставить данные напрямую (категория или числовое значение). Второе дерево будет предсказывать истинное значение за вычетом прогнозируемого значения. Третье дерево будет предсказывать истинное значение минус предсказанное значение первого дерева минус предсказание второго дерева об остатках первого дерева.

Однако библиотека XGBoost, пытаясь минимизировать остатки прогнозируемых остатков прогнозируемых остатков, и так до бесконечности, не стремится создать неисчислимое количество моделей. Алгоритм XGBoost минимизирует функцию потерь, которая также включает штрафной член для сложности модели, и именно этот штрафной член ограничивает количество получаемых деревьев. Кроме того, количество производимых деревьев можно указать напрямую.

За последние несколько лет появилось много извещений о больших успехах использования пакета XGBoost не только в методах машинного обучения, но и методах анализа временных рядов. В частности, они поступали с соревнований Kaggle и отраслевых конференций по машинному обучению.

Баггинг (или, более формально, *бустстрэп-агрегирование*) — это метод обучения моделей, при котором для каждой отдельной модели в ансамбле создаются случайно сгенерированные обучающие наборы. Случайные леса, как правило, используют методологию баггинга при обучении моделей.

Бустинг, как отмечалось выше, относится к методу обучения моделей, в которых ансамбль состоит из последовательно обучаемых моделей, каждая из которых направлена на исправление ошибки, допущенной ее предшественницей. Градиентный бустинг лежит в основе обучения деревьев.

Пример кода

В случае применения случайных лесов модель машинного обучения проще кодировать с помощью инструментов библиотеки XGBoost, чем разбираться в том, как она работает. В следующем примере мы будем обучать модель случайного леса и использовать градиентный бустинг для классификации исходных данных ЭЭГ на основе предварительно сгенерированных признаков.

Для разделения данных на обучающий набор и набор тестирования воспользуемся пакетом sklearn.

```
## Python
>>> from sklearn.model_selection import train_test_split
>>> X_train, X_test, y_train, y_test = train_test_split(
        fset_cesium.values, eeg["classes"], random_state=21)
```

Начнем с создания классификатора, основанного на случайном лесе. Как видите, модель классификации данных ЭЭГ реализуется невероятно просто.

```
## Python
>>> from sklearn.ensemble import RandomForestClassifier
>>> rf_clf = RandomForestClassifier(n_estimators = 10,
>>>                                 max_depth    =  3,
>>>                                 random_state = 21)
>>> rf_clf.fit(X_train, y_train)
```

Теперь нужно определить точность данных за пределами выборки, вызвав метод объекта `Classifier`.

```
## Python
>>> rf_clf.score(X_test, y_test)
0.616
```

Всего несколько строк кода реализуют модель намного лучше и быстрее, чем это может сделать человеческий мозг (лишенный медицинского образования). Помните, что благодаря правильному выбору признаков модель работает со сводной статистикой, а не с полными данными ЭЭГ.

Код классификатора `XGBoost` также предельно прост и выразителен.

```
## Python
>>> import xgboost as xgb
>>> xgb_clf = xgb.XGBClassifier(n_estimators   = 10,
>>>                             max_depth      =  3,
>>>                             random_state   = 21)
>>> xgb_clf.fit(X_train, y_train)
>>> xgb_clf.score(X_test, y_test)
0.648
```

Легко заметить, что модель классификатора `XGBoost` работает немного лучше, чем модель случайного леса. Она также обучается немного быстрее, что показывает следующий эксперимент по расчету времени обучения каждой модели.

```
## Python
>>> start = time.time()
>>> xgb_clf.fit(X_train, y_train)
>>> end = time.time()
>>> end - start
0.0189

## Случайный лес
>>> start = time.time()
>>> rf_clf.fit(X_train, y_train)
>>> end = time.time()
>>> end - start
0.027
```

Скорость проведения вычислений существенно повысилась: модель случайного леса рассчитывается на 50% дольше, чем `XGBoost`. Хотя это не окончательный тест, он все же указывает на преимущество `XGBoost`, особенно при работе с большими наборами данных. Убедитесь в том, что это преимущество увеличится при использовании больших наборов данных с большим количеством выборок и признаков.

Это не тестирование производительности!

Можно высказать много критических замечаний по поводу предыдущего кода. Для начала заметим, что существуют более простые варианты реализации методов тестирования производительности. Кроме того, время обучения необходимо усреднять, а код запускать на реальных данных в не менее реальных условиях обучения, а не на примитивном примере, существующем исключительно в виртуальном ноутбуке Jupyter.

Помимо того, низкая скорость работы алгоритма общего назначения на неких данных совсем не означает, что алгоритм имеет низкую производительность. Зачастую общедоступные программные модули исходно пишутся так, чтобы, в первую очередь, обеспечить прозрачность кода и только затем — достойную производительность. В таких случаях алгоритм действительно работает медленнее, чем мог бы. В остальных случаях алгоритмы отлаживаются под как можно более широкий круг задач, тогда как вам требуется более узкоспециализированная реализация, т.е. стандартные реализации алгоритмов нужно изменять в соответствии с поставленными целями.

Вопросы производительности, характерные для временны*х* рядов, подробно рассматриваются в главе 12.

Возникает справедливый вопрос о том, существует ли в рассматриваемом наборе гиперпараметров что-то, что предоставляет `XGBoost` преимущество над случайным лесом. Например, что будет, если использовать менее сложные деревья, установив меньшую глубину? Или что произойдет при включении в модель меньшего количества деревьев решений? Проверить эти возможности довольно просто, и мы снова видим, что алгоритм `XGBoost` стремится сохранить свои преимущества.

Например, если мы допустим одинаковое количество деревьев решений в ансамбле, но понизим сложность за счет уменьшения глубины деревьев, то увидим, что модель с градиентным бустингом оказывается более точной, чем модель случайного леса.

```
## Python
>>> ## Тестирование того же количества менее сложных деревьев (10)
>>> ## (max_depth = 2)
```

```
>>>
>>> ## XGBoost
>>> xgb_clf = xgb.XGBClassifier(n_estimators = 10,
                               max_depth    =  2,
                               random_state = 21)
>>> xgb_clf.fit(X_train, y_train)
>>> xgb_clf.score(X_test, y_test)
0.616

>>> ## Случайный лес
>>> rf_clf = RandomForestClassifier(n_estimators = 10,
                                    max_depth    =  2,
                                    random_state = 21)
>>> rf_clf.fit(X_train, y_train)
>>> rf_clf.score(X_test, y_test)
0.544
```

Это утверждение оказывается верным даже при еще большем уменьшении сложности дерева.

```
>>> ## Тестирование того же количества менее сложных деревьев (10)
>>> ## (max_depth = 1)

>>> ## XGBoost
>>> xgb_clf = xgb.XGBClassifier(n_estimators = 10,
                               max_depth    =  1,
                               random_state = 21)
>>> xgb_clf.fit(X_train, y_train)
>>> xgb_clf.score(X_test, y_test)
0.632

>>> ## Случайный лес
>>> rf_clf = RandomForestClassifier(n_estimators = 10,
                                    max_depth    =  1,
                                    random_state = 21)
>>> rf_clf.fit(X_train, y_train)
>>> rf_clf.score(X_test, y_test)
0.376
```

Есть несколько возможных причин, объясняющих эффективность и предполагаемое преимущество градиентного бустинга деревьев над случайными лесами. Важно помнить, что нам наверняка не известно, насколько полезными окажутся признаки, избранные для классификации. Этот пример показывает, в каких ситуациях бустинг (градиентный бустинг дерева) оказывается предпочтительнее баггинга (случайного леса). Бустинг с большей вероятностью игнорирует бесполезные признаки, поскольку всегда учитывает полный набор признаков и выделяет наиболее релевантные из них, в то время как отдельные деревья, возникающие при баггинге, основываются на менее значимых признаках.

Этот пример также указывает на то, что наибольшая эффективность бустинга достигается при работе с признаками, сгенерированными специальными программными пакетами, описанными в предыдущей главе. Не боясь создания для временных рядов нескольких сотен признаков — гораздо большего количества, чем можно осмыслить, — смело обращайтесь к бустингу, позволяющему избежать действительно катастрофических результатов.

Градиентный бустинг дерева эффективнее всего работает с большими наборами данных, включая длинные временные ряды. Точность прогнозирования и скорость обучения напрямую зависят от избранного способа реализации. Помимо `XGBoost`, обязательно обратите внимание на библиотеки `LightGBM` и `CatBoost`. Как известно, последние пакеты работают заметно быстрее, чем `XGBoost`, хотя и с ощутимо меньшей точностью вневыборочного тестирования.

Классификация и регрессия

В предыдущих примерах рассмотрены принципы классификации временных рядов с использованием алгоритмов случайных лесов и градиентного бустинга. Не менее эффективно они применяются для прогнозирования временных рядов.

Многие специалисты по статистике считают, что в прогнозировании данных машинное обучение позволяет добиться не намного больших, если не меньших, успехов, чем традиционный статистический подход в анализе временных рядов. Тем не менее за последние несколько лет градиентный бустинг деревьев стал опорным камнем технологий прогнозирования, часто превосходя традиционные статистические модели особенно при обработке больших наборов данных — как в соревнованиях по прогнозированию, так и в промышленных приложениях. Тем не менее при его использовании приходится уделять много внимания настройке параметров моделей, а также подготовке признаков временных рядов.

Одно из неоспоримых преимуществ моделей градиентного бустинга заключается в способности автоматического отсеивания нерелевантных или зашумленных признаков и выделении наиболее важных из них. Однако одного только этого недостаточно для получения самых передовых по характеристикам моделей. Даже для, казалось бы, автоматического метода, такого как градиентный бустинг деревьев, выходные данные могут оказаться не намного лучшими по качеству входных данных. На этом фоне самым важным способом улучшения модели по-прежнему будет предоставление качественных и хорошо протестированных признаков для входных данных.

Существует много вариантов улучшения текущей модели. В частности, можно обратиться к специальным функциям пакета `XGBoost` для определения степеней важности признаков. Это поможет разделить полезные и неиспользуемые признаки и впоследствии расширить набор данных, добавив первые к тем, которые были признаны полезными ранее. Кроме того, можно выполнить поиск по сетке

гиперпараметров, чтобы более точно настроить параметры модели. Наконец, всегда можно более тщательно изучить необработанные временны*е* ряды размеченных данных, попытавшись распознать в них ошибочные данные, которые невозможно представить текущим набором признаков. Стоит задуматься о добавлении признаков, которые наилучшим образом описывают ошибочно распознанные данные, что приведет к дополнительному их уточнению.

Кластеризация

Главная идея кластеризации состоит в отнесении точек данных с похожими характеристиками к разным группам, представляющим отдельные цели для анализа. Такой подход оказывается верным не только для наборов временны*х* рядов, но и для любых других видов данных.

Как и в любом другом случае, я исхожу из предположения, что вы немного знакомы с соответствующими концепциями машинного обучения — хотя бы применимо к данным, не имеющим выраженной временно*й* направленности. Если же вы не знакомы с кластеризацией, то для начала хотя бы кратко ознакомьтесь с ее основными принципами (`https://perma.cc/36EX-3QJU`).

В анализе временны*х* рядов кластеризация может применяться как для классификации, так и для прогнозирования. В первом случае алгоритмы кластеризации применяются для определения требуемого количества кластеров на этапе обучения. Затем эти кластеры используются для выделения типов временны*х* рядов и отнесения целевых выборок к соответствующей группе.

При прогнозировании кластеризация задействуется в явном виде или проявляется в опосредованной форме — для определения метрик расстояний (подробнее о них вы узнаете далее). Существует несколько вариантов построения прогнозов на будущем горизонте h, основанных на кластеризации данных и связанных с ними методов. Помните, что в подобном случае наблюдается не полный временно*й* ряд, а только его первые N шагов, по которым предсказываются значения на временно*м* шаге $N + h$. Здесь также возможно несколько вариантов действий.

Один из них заключается в составлении прогноза по типичному поведению данных, характерному для рассматриваемого класса. Сначала на основе первых N временны*х* шагов определяется кластер, к которому относится выборка временно*го* ряда, а затем прогнозируется поведение по принадлежности к такому кластеру. В частности, на этом этапе нужно отследить изменения в значениях временны*х* рядов между временны*ми* шагами N и $N + h$. Обратите внимание, что первоначальная кластеризация на основе их первых N шагов выполняется для всех временны*х* рядов, а не для всех частей временно*го* ряда, чтобы избежать упреждения.

Еще один вариант — предсказать будущее поведение выборочного временно*го* ряда по поведению его ближайшего соседа (или соседей) в выборочном

пространстве. В этом сценарии находятся ближайшие соседи выборки временного ряда, для которых известна полная траектория, на основе показателей из первых N временных шагов, а затем усредняется поведение таких ближайших соседей в момент $N + h$, чтобы получить прогноз для текущей выборки.

В случае классификации и прогнозирования наиболее важным фактором выступает способ оценки сходства между временными рядами. Кластеризация может быть выполнена для различных расстояний, и способам их определения в задачах высокой размерности посвящено большое количество исследований. Например, каково расстояние между двумя соискателями? Каким будет правильное расстояние между двумя выборками показателей крови? Такие задачи давно изучаются в статическом анализе данных — нам предстоит разобраться в них в контексте анализа временных рядов.

В нашем распоряжении есть два больших класса метрик расстояния (схожести), применяемых в методах кластеризации по отношению к данным временных рядов.

Расстояние, основанное на признаках

Создайте признаки для временных рядов и обработайте их как координаты, для которых необходимо рассчитать данные. Это не полностью решает проблему выбора метрики расстояния, но сводит ее к стандартной задаче определения расстояния в наборе статических данных.

Расстояние на основе необработанных данных временного ряда

Найдите способ определения "близости" временных рядов, предпочтительно таким образом, чтобы учитывать разные временные масштабы, различное количество измерений и другие вероятные отличия в выборках временных рядов.

Далее мы будем применять оба способа определения метрики расстояния к набору данных временного ряда, в котором каждая выборка представляет проекцию рукописного слова из двумерного изображения в одномерный временной ряд.

Генерация признаков из данных

Ранее мы обсудили способы создания и выбора признаков. Далее нам предстоит оценить расстояние между наборами данных временных рядов на основе сходства их признаков.

В идеальном сценарии, используя дерево для оценки важности признаков, из набора данных отсеиваются малозначимые и неинтересные временные ряды. При этом такие признаки не применяются в методиках расчета расстояния, поскольку могут ошибочно указывать на различия между двумя временными рядами, в то время как самом деле они не свидетельствуют о сходстве целевых классов в задаче классификации или результатах прогнозирования.

Как и в случае с набором данных ЭЭГ, анализ данных стоит начать с изучения отдельных примеров классов и определения очевидных различий во временной структуре наборов данных временных рядов.

В качестве исходных данных будем рассматривать подмножество набора FiftyWords (`https://oreil.ly/yadNp`), доступного для загрузки из репозитория UEA and UCR Time Series Classification. Этот набор данных был создан авторами статьи о кластеризации рукописных слов в исторических документах (`https://oreil.ly/01UJ8`) еще в 2003 году. В этой статье авторы предложили использовать своеобразные "профили слов" для масштабирования двумерного изображения рукописного слова в одномерную кривую, представленную одинаковым количеством измерений независимо от длины слова. Набор данных в репозитории не совпадает с описанным в документе, но имеет схожую структурную организацию. Целью оригинальной работы было разработка метода унифицированного обозначения всех схожих или идентичных слов в документе, что позволяет в дальнейшем обозначить их цифровой подписью (подвиг, который в наши дни "элементарно" осуществляется нейронной сетью, прошедшей 20-летний путь постоянного усовершенствования).

Таким образом, в этом примере анализ данных основан на работе с *профилями проекции слов*, в которых термин "проекция" указывает на преобразование изображения из двумерного вида в одномерный, и это очень важный этап, поскольку размерность данных имеет первостепенную важность в анализе временных рядов. Обратите внимание, что ось времени не содержит временных меток, а представляет последовательность написанных слов слева направо. Тем не менее структура таких данных полностью повторяет принятую для временных рядов (упорядоченные и равномерно распределенные значения), поэтому в дальнейшем они будут описываться терминами *время* и *временной*, хотя это и не совсем верно. В рассматриваемом сценарии это не имеет принципиального значения.

Примеры представления разных слов приведены на рис. 9.4.[1]

Рассматривая полученные данные, и особенно анализируя очевидные закономерности, проявляющиеся при визуальном изучении графиков, так же, как в данных ЭЭГ, можно определиться с важными для дальнейшего анализа признаками, например указывающими высоту и расположение пиков, а также с отдельными их характеристиками, в частности с крутизной и формой вершины.

Многие из таких признаков становятся более значимыми при распознавании изображений, чем временных рядов, и могут использоваться в качестве основы для генерации признаков в других задачах. Зачастую визуализация данных, выполняемая очень просто, может предоставить намного больше понятной

[1]Обратите внимание, что информация о фактическом содержании каждого "слова" остается недоступной и не особенно актуальна при сборе всего исходного набора данных. Идея заключается в том, чтобы обозначить одинаковые слова одной и той же подписью, чтобы в дальнейшем сократить усилия по разметке документов.

и полезной информации, чем утомительное изучение длинных последовательностей значений временных рядов. Таким образом, определяясь с признаками, не пренебрегайте изучением графического представления данных. Один только этот факт указывает на сложность и важность операции генерации признаков в анализе временных рядов. В некоторых случаях различия в классах временных рядов проявляются достаточно сильно, но описать их в коде оказывается далеко не так просто, как хотелось бы. Или вдруг выясняется, что написание такого кода оказывается чрезвычайно затратным занятием.

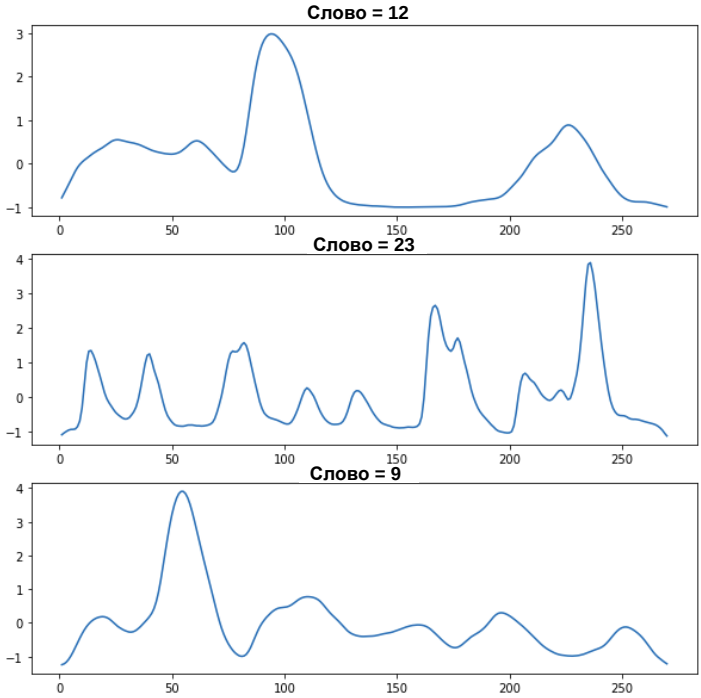


Рис. 9.4. *Профили проекции трех разных слов (12, 23 и 9) сильно различаются между собой. На них просматриваются признаки, по которым различаются слова: временное расположение (ось* ***x****) самого большого или второго по величине пика, количество локальных пиков, общий диапазон значений и средняя выпуклость кривых*

В последнее десятилетие самым перспективным инструментом классификации изображений стало глубокое обучение. При наличии достаточного количества данных можно обучить классификатор глубокого обучения на изображениях наших графиков (подробности приведены в главе 10). Но на данный момент мы не будем акцентировать свое внимание на нем, а обсудим сложности более простых вариантов реализации. В частности, генерировать признаки, определяющие местоположение каждого пика, достаточно сложно, поскольку обнаружение пиков требует значительных программных и вычислительных ресурсов и больше сродни искусству.

Воспользуемся для визуальной оценки данных одномерной гистограммой, построенной либо для всех, либо только для отдельных примеров классов. Такой подход оказывается менее затратным в вычислительном плане при нахождении пиков или поиске других критериев, представляющих целевые формы, которые проявляются во временных рядах. В следующем примере на графике отображаются те же члены класса, которые были представлены на предыдущем графике, но теперь в сопровождении одномерных гистограмм (рис. 9.5).

```
## Python
>>> plt.subplot(3, 2, 1)
>>> plt.plot(words.iloc[1, 1:-1])
>>> plt.title("Word = " + str(words.word[1]), fontweight = 'bold')
>>> plt.subplot(3, 2, 2)
>>> plt.hist(words.iloc[1, 1:-1], 10)
>>> plt.subplot(3, 2, 3)
>>> plt.plot(words.iloc[3, 1:-1])
>>> plt.title("Word = " + str(words.word[3]), fontweight = 'bold')
>>> plt.subplot(3, 2, 4)
>>> plt.hist(words.iloc[3, 1:-1], 10)
>>> plt.subplot(3, 2, 5)
>>> plt.plot(words.iloc[5, 1:-1])
>>> plt.title("Word = " + str(words.word[11]), fontweight = 'bold')
>>> plt.subplot(3, 2, 6)
>>> plt.hist(words.iloc[5, 1:-1], 10)
```

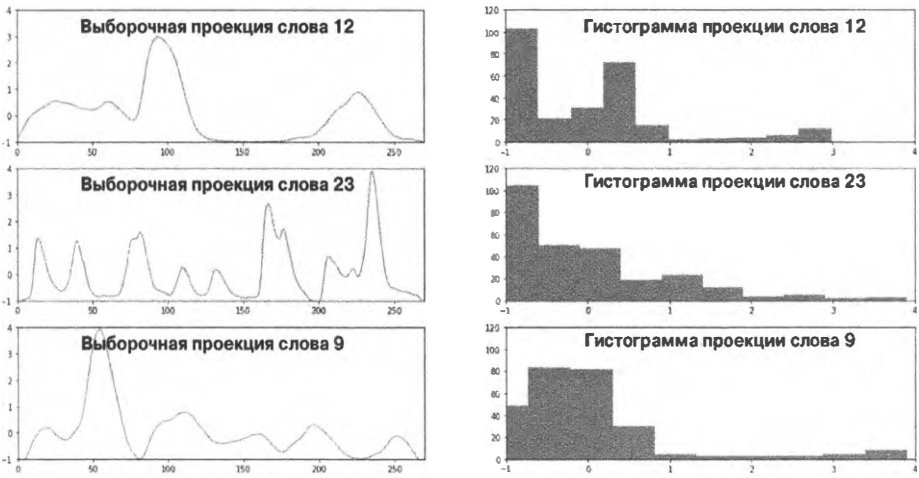


Рис. 9.5. *Еще один способ количественного описания классов при поиске полезных признаков. В частности, гистограмма отдельных экземпляров классов показывает, что такие характеристики гистограммы, как количество локальных пиков, асимметрия и эксцесс, могут оказаться хорошей и полезной заменой отдельных атрибутов кривых временных рядов, будучи простыми для визуального анализа, но чрезвычайно сложными в программной реализации*

Нужно убедиться, что рассматриваемые примеры не являются выбросами в сравнении с другими экземплярами этих же слов. По этой причине построим двумерную гистограмму представления двух слов, чтобы получить представление об их отдельных вариантах (рис. 9.6).

```
## Python
>>> x = np.array([])
>>> y = np.array([])
>>>
>>> w = 12
>>> selected_words = words[words.word == w]
>>> selected_words.shape
>>>
>>> for idx, row in selected_words.iterrows():
>>>     y = np.hstack([y, row[1:271]])
>>>     x = np.hstack([x, np.array(range(270))])
>>>
>>> fig, ax = plt.subplots()
```

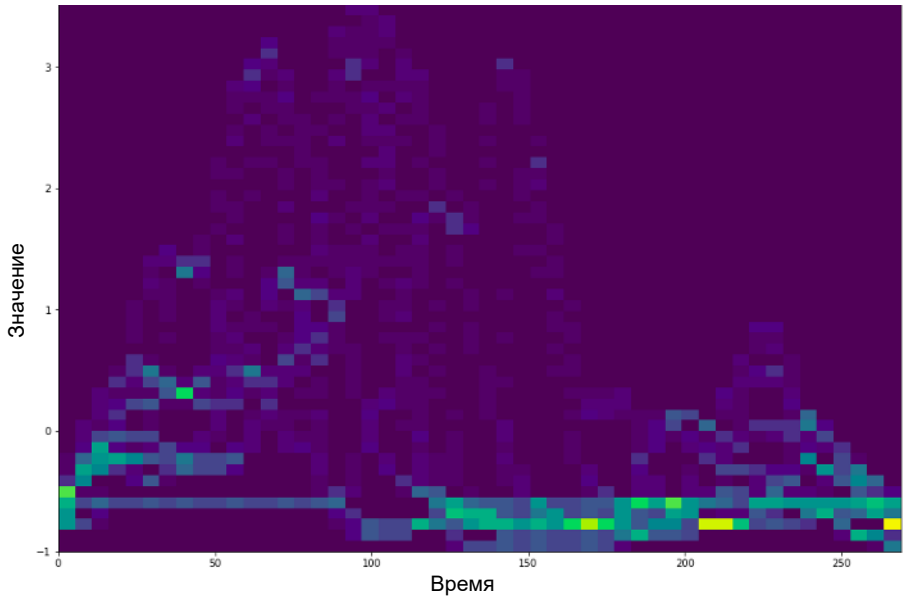


Рис. 9.6. *Двумерная гистограмма одномерных проекций слова 12. На оси* ***y*** *откладываются значения на отдельных временных шагах, а ось* ***x*** *включает 270 временных шагов для каждой выборки временного ряда/проекции слова*

На рис. 9.6 показана двумерная гистограмма для всех экземпляров слова 12 в наборе данных. Хотя отдельная кривая на рис. 9.5 показывает, что нужно сосредоточиться на поиске двух больших пиков, которые доминируют во временном ряду, гистограмма показывает, что, вероятнее всего, общим для большинства

членов этого класса будет плоский участок между пиками, расположенными между временными шагами от 120 до 200.

Двумерную гистограмму также можно использовать для определения точки отсечения для максимального пика в заданном классе, который, по-видимому, находится между временными шагами 50 и 150, — всегда можно написать функцию, которая ответит на вопрос "достигается ли максимальное значение между точками 50 и 150?"

Подобным образом построим еще одну двумерную гистограмму — на этот раз для слова 23, профиль которого характеризуется большим количеством маленьких выпуклостей (см. пример на рис. 9.5), признаки которых также сложно поддаются программному вычислению (рис. 9.7).

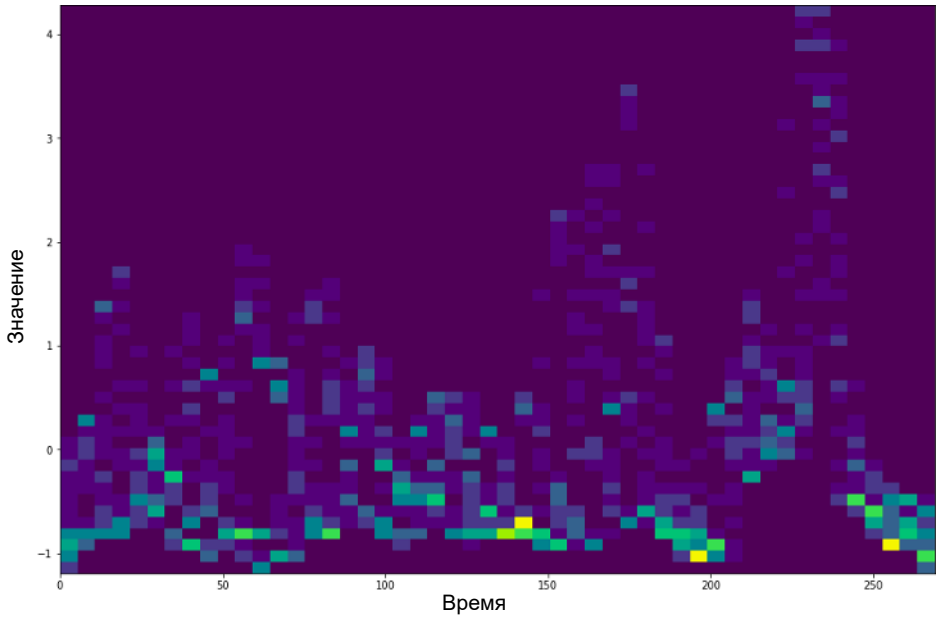


Рис. 9.7. *Двумерная гистограмма одномерных проекций слова 23. На оси* ***y*** *откладываются значения на отдельных временных шагах, а ось* ***x*** *включает 270 временных шагов для каждой выборки временного ряда/проекции слова*

Неудивительно наблюдать сильно "размазанную" гистограмму для слова 23 (рис. 9.7), учитывая тот факт, что даже в примере, представленном графиками на рис. 9.5, проявляется достаточно много признаков, чтобы ожидать сильного размазывания для выборок с неточно совпадающими признаками. Тем не менее мы также видим, что максимальное значение этого класса относится к непересекающемуся диапазону временных шагов, в отличие от слова 12. В этом классе максимальное значение наблюдается после временного момента 150, что вполне

резонно, учитывая, что два наибольших пика в профиле слова 23 относились именно к этому диапазону. Двумерная гистограмма подтверждает, что более ранний пик не такой высокий, как последующий, предлагая другие способы квантования формы этого временного ряда, чтобы отличать его от других классов.

Двумерные гистограммы помогают оценить изменчивость признака в отдельном классе, позволяя отойти от учета единственного экземпляра класса при выборе признаков.

В данном случае набор признаков сгенерирован по форме проекции слова и дополнительному набору признаков, полученных из формы гистограммы проекции слова (одномерные сводные данные проецируются в другой тип одномерных данных, по которым и определяются признаки). Таким образом, учитываются большие "мазки", проявляющиеся на двумерных гистограммах и указывающих на наличие пиков, положение которых не отличается особой устойчивостью. Генерация второй характеристической формы для каждой проекции слова по гистограмме, а не исходной проекции слова, может оказаться более надежным способом. Гистограммы показывают, какие виды значений появляются во временном ряду, но не характеризуют их точное расположение, что крайне важно для нашего анализа, учитывая, что пики в проекциях не имеют стабильных временных мест расположения.

Сначала сгенерируем признаки для временного ряда длиной 270 временных точек. В нашем случае функция генерации признаков имеет сокращенное название, что сделано исключительно в целях улучшения читабельности кода.

```
from cesium import featurize.featurize_time as ft

## Python
>>> word_vals          = words.iloc[:, 1:271]
>>> times              = []
>>> word_values        = []
>>> for idx, row in word_vals.iterrows():
>>>     word_values.append(row.values)
>>>     times.append(np.array([i for i in
               range(row.values.shape[0])]))
>>>
>>> features_to_use = ['amplitude',
>>>                    'percent_beyond_1_std',
>>>                    'percent_close_to_median']
>>> featurized_words = ft(times           = times,
>>>                       values          = word_values,
>>>                       errors          = None,
>>>                       features_to_use = features_to_use,
>>>                       scheduler       = None)
```

Теперь построим гистограммы и используем их в качестве временного ряда, для которого генерируются признаки.[2]

```
## Python
>>> ## Генерирование признаков по гистограммам
>>> times = []
>>> hist_values = []
>>> for idx, row in words_features.iterrows():
>>>     hist_values.append(np.histogram(row.values,
>>>                                     bins=10,
>>>                                     range=(-2.5, 5.0))[0] + .0001)
>>>                                     ## обработка нулей вызывает
                                        ## дополнительные сложности
>>>     times.append(np.array([i for i in range(9)]))
>>>
>>> features_to_use = ["amplitude",
>>>                    "percent_close_to_median",
>>>                    "skew"
>>>                   ]
>>>
>>> featurized_hists = ft(times           = times,
>>>                       values          = hist_values,
>>>                       errors          = None,
>>>                       features_to_use = features_to_use,
>>>                       scheduler       = None)
```

Легко удостовериться, что во всех гистограммах учитывается одинаковое количество интервалов, относящихся к общему диапазону значений, представляемому параметрами, которые передаются функции `np.histogram()`. Обеспечив общность диапазона значений, образующих единую "временную" ось, гистограммы можно сравнивать непосредственно при генерации признаков временных рядов. Если отказаться от проведения такого согласования, то сгенерированные признаки не обязательно будут релевантными при сравнении одной гистограммы с другой.

Наконец, объединим оба источника признаков.

```
## Python
>>> features = pd.concat([featurized_words.reset_index(drop=True),
>>>                       featurized_hists],
>>>                       axis=1)
```

[2] При построении гистограмм мы будем придерживаться того же принципа, что и в случае визуализации проекций слов, в которых на оси *x* откладывается не время, а другая равномерно и последовательно упорядоченная величина, которая, тем не менее, рассматривается как время. В нашем примере при генерировании признаков будем рассматривать ось *x* как временную.

Временные метрики расстояния

При проведении кластерного анализа нужно определиться с метриками расстояния. Ориентируясь по признакам временных рядов, как показано выше, можно придерживаться самых разных стандартных методик, что характерно для кластерного анализа большинства перекрестных данных. Если вы не знакомы с принципами выбора метрик расстояния в подобных случаях, то обязательно обратитесь к соответствующим справочным руководствам (`https://perma.cc/MHL9-2Y8A`).

В этом разделе мы сосредоточимся на задаче определения сходства временных рядов по метрикам расстояний. Одна из наиболее часто применяемых для таких целей метрик называется *динамической деформацией времени* (Dynamic Time Warping — DTW). Механизм DTW прекрасно подходит для кластеризации временных рядов, наиболее характерной особенностью которых является их общая форма, как в случае рассмотренной ранее визуализации проекций слов.

Название метода основано на методологии, которая базируется на временной "деформации" рядов данных для выравнивания вдоль временной оси с целью последующего сравнения форм. Концепцию динамического искажения времени лучше всего проиллюстрировать графически (рис. 9.8). Временная ось (x) деформируется, т.е. расширяется или сжимается так, чтобы обеспечить наилучшее выравнивание точек между двумя кривыми (т.е. двумя временными рядами) для последующего сравнения их форм.

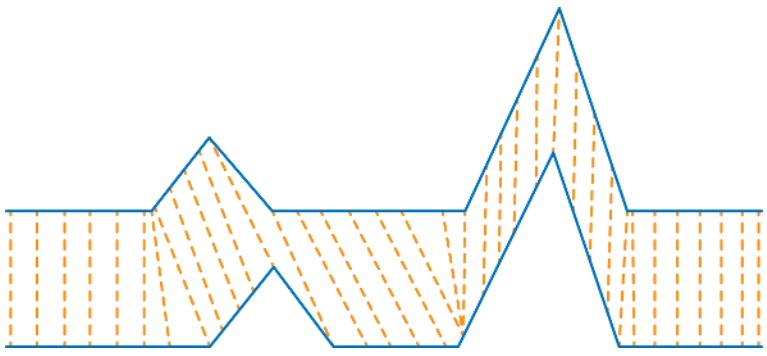


Рис. 9.8. *Механизм динамической деформации времени. Каждая точка одного временного ряда сопоставляется с точкой альтернативного временного ряда. При этом однозначного сопоставления точек не требуется, что имеет несколько важных последствий: (1) временные ряды не обязательно должны иметь одинаковую длину или даже масштабы времени, главное — их форма; (2) время не всегда движется вперед в процессе подгонки и может протекать с разной скоростью в каждом временном ряду. Под движением времени подразумевается продвижение по кривой в направлении оси* ***x***

Заметьте, что при стандартной параметризации алгоритма фактические значения на временной оси одной кривой совершенно не совпадают со значениями другой временной оси. Никто не запрещает сравнивать временной ряд с наносекундным интервалом с другим временным рядом, данные которого масштабированы для нескольких тысячелетий (скорее всего, это будет далеко не самым разумным решением). Цель этого алгоритма — сравнить одни только визуальные формы без учета временной протяженности данных. Действительно, термин "время" в данном случае рассматривается в общем концептуальном понимании как направление равномерного распределения набора точек вдоль оси x, а не в строгом физическом проявлении.

Перечислим правила работы DTW.

- Каждая точка одного временного ряда должна соответствовать как минимум одной точке другого временного ряда.
- Первый и последний индексы каждого временного ряда должны сопоставляться с таковыми в другом временном ряду.
- Масштабирование данных должно выполняться так, чтобы обеспечить прямой (вперед), а не обратный порядок движения времени. Обратного хода времени можно избежать, если исключить сопоставление точек одного временного ряда с точками другого временного ряда, которые уже были пройдены на временной оси. Однако время не обязано должно двигаться вперед равномерно. Например, два последовательных временных шага в исходном ряду могут быть деформированы (сжаты во времени) таким образом, чтобы подогнать соответствующие значения к одному и тому же месту на оси x (см. первый излом верхней кривой/сплошной линии на рис. 9.8).

Указанные правила предопределяют много разных способов временного выравнивания — выбранный нами вариант обеспечивает минимизацию расстояния между кривыми. Это расстояние, или функция стоимости, часто измеряется как сумма абсолютных разностей между совмещаемыми точками, где под абсолютной разностью подразумевают разницу между значениями точек.

Получив качественное представление о работе алгоритма DTW, давайте познакомимся с примером его программной реализации.

```
## Python
>>> def distDTW(ts1, ts2):
>>>     ## Настройка
>>>     DTW={}
>>>     for i in range(len(ts1)):
>>>         DTW[(i, -1)] = np.inf
>>>     for i in range(len(ts2)):
>>>         DTW[(-1, i)] = np.inf
>>>         DTW[(-1, -1)] = 0
```

```
>>>
>>>     ## Пошаговый расчет оптимальных значений
>>>     for i in range(len(ts1)):
>>>         for j in range(len(ts2)):
>>>             dist = (ts1[i] - ts2[j])**2
>>>             DTW[(i, j)] = dist + min(DTW[(I - 1, j)],
>>>                                      DTW[(i, j - 1)],
>>>                                      DTW[(i-1, j - 1)])
>>>             ## Это пример динамического программирования
>>>
>>>     ## Рассчитав полный путь, можно получить
>>>     ## соответствующее расстояние
>>>     return sqrt(DTW[len(ts1) - 1, len(ts2) - 1])
```

Как указано в комментариях, эта задача решается методами динамического программирования, а вычисление расстояния DTW является классической задачей динамического программирования. Мы можем пошагово отследить пути от начала каждого временного ряда до конца, и построить решение, основываясь на знаниях о предыдущих шагах при принятии последующих решений.

Существует целый ряд реализаций DTW с различными принципами повышения эффективности поиска оптимального или почти оптимального решения. Им следует уделить особо пристальное внимание, особенно при работе с большими наборами данных.

Есть и другие способы измерения расстояний между временными рядами. Ниже описаны некоторые из них.

Расстояние Фреше

Это минимальное расстояние между двумя кривыми, обеспечивающее беспрепятственное прохождении по ним при временной деформации. Эта метрика расстояния часто объясняется на примере прогулки человека и его собаки, которые движутся по двум разным траекториям, но постоянно связаны между собой поводком. Каждому из них необходимо пройти свою кривую от начала до конца, они могут двигаться с разными скоростями и изменять их в произвольных точках кривой при перемещении в одном и том же направлении. Расстояние Фреше — это самая короткая длина поводка, при которой можно пройти путь по оптимальным траекториям (при условии, что человек и его собака смогут их найти!).

Корреляция Пирсона

Измерять расстояния между двумя временными рядами можно по корреляции между ними. В отличие от других способов здесь расстояние между временными рядами минимизируется в результате максимизации корреляции между ними. Это сравнительно простой в вычислительном плане метод. Однако он требует использования во временных рядах одинакового количества

точек данных или понижения частоты дискретизации одной из выборок до таковой в другой выборке. Временная сложность вычисления корреляции составляет $O(n)$, что указывает на высокую эффективность метода с точки зрения вычислительных затрат.

Самая длинная общая подпоследовательность

Эта мера расстояния подходит для временных рядов, которые представляют собой последовательность категориальных или целочисленных значений. В таких случаях при рассмотрении сходства двух временных рядов можно определить длину самой длинной общей подпоследовательности, т.е. наибольшую длину последовательности значений, которые точно совпадают, хотя их точное расположение во временном ряду не требуется для совпадения. Как и в случае с DTW, такой подход предполагает нахождение схожих форм, а не точное совпадение временных интервалов, в которых они проявляются. Также отметим, что, как и DTW, но в отличие от корреляции Пирсона, в этом методе не требуется, чтобы временные ряды имели одинаковую длину. Родственная мера — это *расстояние редактирования*, т.е. количество изменений, которые необходимо внести в один временной ряд, чтобы сделать его идентичным другому.

Расстояние или сходство

В источниках, посвященных поиску метрик расстояний между временными рядами, для описания упомянутых выше показателей часто применяется термин "сходство". В большинстве случаев эти термины рассматриваются как взаимозаменяемые — описывающие степень подобности временных рядов между собой. Тем не менее отдельные показатели могут показаться более подходящими, например расстояние Фреше, которое вычисляется в соответствующих единицах измерения ("футы" или "кг/доллар" или любых других единицах представления данных временного ряда). Остальные метрики расстояния, в частности корреляция, единицами измерения не снабжаются.

Иногда найти простое, но удачное решение помогает творческий подход. Всегда старайтесь как можно более точно оценить стоящие перед вами цели. Обратимся к публикации на сайте Stackoverflow (`https://perma.cc/389W-68AH`), в которой рассматривается задача поиска расстояния в конкретном приложении, а именно — при классификации временных рядов для дальнейшего сопоставления найденных классов с одним из трех центроидов, полученных в предыдущем кластерном анализе. Рассматриваются три следующих класса.

- Прямая линия.
- Пик в начале временного ряда, а затем — прямая линия.
- Пик в конце временного ряда, а затем — прямая линия. Автор публикации обнаружил, что несколько стандартных метрик расстояний, включая эвклидово и DTW, не подходят для решения задачи. Метрика DTW дает слишком грубую оценку, рассматривая любые временные ряды с пиком подобно временным рядам с пиками в конце и начале ряда (именно поэтому DTW нельзя считать панацеей, даже несмотря на высокую вычислительную затратность метода!).

Один из дотошных комментаторов предложил провести преобразование, которое повысило бы точность вычисления расстояний, а именно: сравнивать не исходные временные ряды, а временные ряды накопленных сумм. В результате как эвклидовы метрики расстояния, так и DTW приводят к правильному упорядочению — временной ряд с пиком в начале показал наименьшее расстояние для прототипа своего класса, но не равное расстоянию для прототипа с пиками в начале и конце временного ряда. Это напоминает подход, принятый в одном из более ранних примеров анализа, в котором для принятия модели ARIMA достаточно правильно преобразовать временной ряд, необработанные данные которого не удовлетворяют всем необходимым условиям.

Избегайте эвклидовых метрик расстояния при сравнении временных рядов

Возможно, вы заметили, что эвклидово расстояние не упоминалось в приведенном выше обсуждении метрик подобия временных рядов. Эвклидово расстояние, как правило, плохо подходит для оценки сходства временных рядов. Чтобы понять, почему, сравним две синусоиды и прямую линию. Если синусоиды имеют одинаковый период, но разные фазы (смещены по оси времени x) или характеризуются разными амплитудами, то прямая линия будет иметь меньшее эвклидово расстояние до одной из синусоид, чем до другой. Это далеко не тот результат, который ожидалось получить, но он указывает на отдельные преимущества рассмотренных ранее метрик.

- Компенсация фазы — смещение по временной оси, которое не играет большой роли при сравнении.
- Распознавание сходства форм, а не сходства величин.

Ниже представлено простое упражнение для реализации, так что обязательно попробуйте его выполнить.

Все же, остановив свой выбор на эвклидовом расстоянии, рассмотрите возможность использования его совместно с преобразованием Фурье. В этом случае уменьшение количества измерений осуществляется за счет ограничения частоты, что подходит для большинства данных временных рядов, поскольку высокие частоты оказывают слабое влияние на общую форму и динамику временного ряда. Другой вариант заключается в аппроксимации по методу символьной агрегации (SAX — Symbolic Aggregate Approximation), разработанному в 2007 году для уменьшения размерности временных рядов перед вычислением нижней границы эвклидова расстояния.

К сожалению, не существует универсального правила для выбора метрики расстояния. Вам придется проявить недюжинные старания, чтобы найти сбалансированное решение, учитывающее следующие факторы.

- Минимизация использования вычислительных ресурсов.
- Выбор расстояния, подчеркивающего признаки временного ряда, которые лучше всего соответствуют конечной цели.
- Метрика расстояния должна учитывать и подчеркивать сильные/слабые стороны аналитических методов, используемых для сравнения временных рядов. Например, кластеризация по методу *k*-средних не предполагает попарного сравнения расстояний, но сводит к минимуму отклонения, так что в этом методе имеет смысл обратиться к эвклидовым метрикам.

Код кластеризации

Теперь, когда мы обсудили, как генерируются признаки для кластерного анализа и рассмотрели принципы измерения расстояние между временными рядами, выполним кластеризацию по выбранным признакам и согласно матрице попарных расстояний DTW с последующим сравнением результатов.

Иерархическая кластеризация нормализованных признаков

Вычислим признаки для слов, представленных в виде временных рядов, как для исходных записей, так и для их гистограмм. В дальнейшем такие признаки будут использоваться в совершенно разных масштабах, поэтому для применения к ним единой метрики расстояния их необходимо нормализовать, что является стандартной процедурой в задачах кластеризации, основанной на признаках.

```
## Python
>>> from sklearn import preprocessing
>>> feature_values = preprocessing.scale(features.values)
```

Выберем алгоритм иерархической кластеризации и проведем обучение для 50 кластеров, поскольку в дальнейшем будем сопоставлять эти кластеры с 50 словами в исходном наборе данных.

```
## Python
>>> from sklearn.cluster import AgglomerativeClustering
>>> feature_clustering = AgglomerativeClustering(n_clusters = 50,
>>>                                              linkage    = 'ward')
>>> feature_clustering.fit(feature_values)
>>> words['feature_labels'] = feature_clustering.fit_predict(p)
```

Давайте посмотрим, наблюдаются ли соответствия в подписях слов и кластеров (последние создаются независимо от оригинальных подписей слов).

```
## Python
>>> from sklearn.metrics.cluster import homogeneity_score
>>> homogeneity_score(words.word, words.feature_labels)
0.508
```

Нам повезло работать с размеченными данными; в противном случае существует вероятность сделать неправильные выводы по данным полученных кластеров. В нашем случае около половины кластеров тесно связаны с одним словом. Вернувшись назад и попробовав улучшить результат, можно прийти к следующим выводам.

- Использовано всего шесть признаков. Это очень мало, поэтому их количество лучше увеличить.
- Нужно обратиться к признакам с относительно слабой корреляцией, что в нашем случае не было сделано.
- Полезных признаков явно недостаточно. При визуальном изучении данных были подмечены признаки, которые не участвовали в анализе, например количество и расположение характерных пиков. Вероятно, нужно пересмотреть отношение к этим характеристикам, выделив их в признаки в явном или косвенном виде.
- Нужно обратиться к другим метрикам расстояний, возможно к таким, в которых отдельные признаки имеют больший вес, чем остальные, отдавая предпочтение тем признакам, которые визуально кажутся более полезными.

Иерархическая кластеризация с помощью матрицы расстояний DTW

Мы успешно завершили самую сложную часть метода прямой кластеризации временных рядов, основанного на вычислении матрицы попарных расстояний для DTW. Это очень затратная операция с точки зрения вычислительных ресурсов, поэтому полученные результаты лучше сохранить, чтобы иметь возможность вернуться к их анализу в случае любой необходимости.

```
## Python
>>> p = pairwise_distances(X, metric = distDTW)
>>> ## Вычисления выполняются долго, поэтому данные стоит сохранить
>>> with open("pairwise_word_distances.npy", "wb") as f:
    np.save(f, p)
```

Располагая матрицей попарных расстояний, можно обратиться к алгоритму иерархической кластеризации.

```
## Python
>>> from sklearn.cluster import AgglomerativeClustering
>>> dtw_clustering = AgglomerativeClustering(linkage     = 'average',
>>>                                          n_clusters  = 50,
>>>                                          affinity    = 'precomputed')
>>> words['dtw_labels'] = dtw_clustering.fit_predict(p)
```

Наконец, как и прежде, проверим соответствие подобранных кластеров известным подписям.

```
## Python
>>> from sklearn.metrics.cluster import homogeneity_score,
>>>                                      completeness_score
>>> homogeneity_score(words.word, words.dtw_labels)
0.828
>>> completeness_score(words.word, words.dtw_labels)
0.923
```

Легко заметить, что кластеризация, выполняемая с помощью DTW, работает значительно лучше, чем кластеризация на основе признаков. Однако если вы запустите код вычисления расстояния DTW на своем компьютере, особенно на стандартном ноутбуке, то увидите, насколько больше времени отнимает выполнение алгоритма DTW по сравнению с вычислением признаков. Вполне вероятно, нужно заняться улучшением алгоритма кластеризации на основе признаков, поскольку кластеризация по расстоянию DTW не предполагает простых способов усовершенствования. Уточним варианты дальнейшего улучшения методик.

- Включить в рассмотрение признаки, а также расстояние DTW. Как с программной, так и с концептуальной точек зрения, совместное использование признаков и расстояния DTW является очень сложной для реализации задачей.
- Попробовать использовать другие метрики расстояний. Как обсуждалось ранее, правильный выбор расстояния зависит от исходных данных, поставленных целей и задач анализа. Таким образом, сначала нужно предельно точно определить цель анализа слов и только затем задуматься о том, действительно ли DTW представляет лучшую из доступных метрику расстояния для достижения поставленной цели.

Дополнительные источники

- Расстояния между временными рядами и метрики сходства

  Meinard Muller, "Dynamic Time Warping," in Information Retrieval for Music and Motion (Berlin: Springer, 2007), 69–84, `https://perma.cc/R24Q-UR84`

  В одной из глав этой книги содержится обширное описание алгоритма динамической деформации времени, включая обсуждение общих приближений, призванных уменьшить вычислительную сложность метода DTW.

  Stephane Pelletier, "Computing the Frechet Distance Between Two Polygonal Curves," (lecture notes, Computational Geometry, McGill University, 2002), `https://perma.cc/5QER-Z89V`

  Набор лекционных заметок для Университета Макгилла, в которых предложено интуитивно понятное визуальное и алгоритмическое объяснение расстояния Фреше и способов его вычисления.

  Pjotr Roelofsen, "Time Series Clustering," master's thesis, Business Analytics, Vrije Universiteit Amsterdam, 2018, `https://perma.cc/K8HJ-7FFE`

  Магистерская диссертация по кластеризации временных рядов, которая начинается с важного и очень подробного обсуждения основных методов вычисления расстояний между временными рядами, включающая информацию о сложности вычислений и полезные иллюстрации, значительно облегчающие интуитивное восприятие материала.

  Joan Serra and Josep Ll. Arcos, "An Empirical Evaluation of Similarity Measures for Time Series Classification," Knowledge-Based Systems 67 (2014): 305–14, https:// `perma.cc/G2J4-TNMX`

  В этой статье проводится эмпирический анализ точности вневыборочного тестирования моделей классификации, построенных с использованием семи различных метрик сходства временных рядов: эвклидова расстояния, коэффициентов Фурье, модели AR, DTW, расстояния редактирования, измененного во времени расстояния редактирования и минимальных затрат на устранения несходства. Авторы протестировали эти показатели на 45 общедоступных наборах данных из хранилища временных рядов UCR.

- Машинное обучение для временных рядов

  Keogh Eamonn, "Introduction to Time Series Data Mining," slideshow tutorial, n.d., `https://perma.cc/ZM9L-NW7J`

Серия слайдов, посвященных предварительной обработке данных временного ряда, подвергаемых дальнейшему анализу с помощью алгоритмов машинного обучения, который основан на методах измерения сходства между временными рядами и определения мотивов, которые можно использовать для анализа и сравнения.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos, "The M4 Competition: Results, Findings, Conclusion and Way Forward," International Journal of Forecasting 34, no. 4 (2018): 802–8, `https://perma.cc/42HZ-YVUU`

В этой статье подводятся итоги конкурса *M4*, проведенного в 2018 году, в котором сравнивались различные методы прогнозирования временных рядов, включая многие ансамблевые методы, для произвольно выбранного набора из 100 тысяч временных рядов, включая данные, собранные с различными частотами (ежегодно, ежечасно и т.д.). В этом обзоре результатов конкурса авторы показывают, что как первое, так и второе места в конкурсе получены благодаря использованию комбинированных методик анализа данных, в значительной степени основанных на статистических исследованиях и алгоритмах машинного обучения. Подобные результаты указывают на важность понимания и применения как статистического анализа, так и машинного обучения в прогнозировании данных.

ГЛАВА 10

Глубокое обучение для временны*х* рядов

Глубокое обучение для временны*х* рядов — это относительно новое, но очень перспективное начинание. Будучи необычайно гибким методом, глубокое обучение как нельзя лучше подходит для анализа временны*х* рядов. Наиболее многообещающая его особенность заключается в способности моделирования сложного и нелинейного временно*г*о поведения без построения предположений о его функциональных проявлениях. Такой подход в корне изменяет ситуацию в отрасли нестатистических методов прогнозирования.

Для тех, кто не знаком с глубоким обучением, приведем его краткое описание, длиной в один абзац (конечно же, далее оно рассматривается более подробно). Глубокое обучение — это отрасль машинного обучения, основанная на анализе графа, соединяющего входные узлы со сложной структурой узлов и ребер. При переходе от одного узла к другому по ребру значение умножается на вес этого ребра, а затем, как правило, подвергается нелинейной активации с помощью специальной функции. Именно эта нелинейная функция активации делает глубокое обучение настолько интересным: она позволяет описывать очень сложные, нелинейные данные, что не удавалось сделать традиционными методами.

За последние 10 лет глубокое обучение достигло максимального развития благодаря широкой доступности и повышению производительности вычислительных устройств, а также наличию огромных объемов данных, необходимых для обучения сложных моделей. Модели глубокого обучения могут иметь миллионы параметров. Чтобы понять, как они устроены, представьте себе произвольный граф, расчет которого выполняется по всем возможным методикам умножения матриц и нелинейных преобразований, связанный с интеллектуальным механизмом оптимизации модели одномоментно по одному небольшому набору данных, веса которого пошагово корректируются таким образом, чтобы каждый раз выдавать все более и более точные результаты. Так работает глубокое обучение в двух словах.

В отличие от других предметных областей, таких как обработка изображений и анализ текстов, в прогнозировании глубокое обучение не демонстрирует поражающие воображение результаты. Тем не менее есть все основания полагать, что

в конечном счете с его участием прогнозирование выйдет на качественно новый уровень, обеспечив сокращение набора весьма однородных допущений и технических требований, свойственных традиционным моделям прогнозирования.

При использовании глубокого обучения нивелируются многие проблемы, связанные с предварительной обработкой данных.

- Отсутствует требование стационарности данных.
- Нет необходимости в развитии навыков (искусства) выбора параметров, таких как оценка сезонности и порядка модели ARIMA.
- Не нужно разрабатывать гипотезу о базовой динамике системы, без которой не обойтись модели пространства состояний.

Эти преимущества покажутся вам знакомыми в свете материала главы 9, в которой многие из них обсуждались в контексте применения машинного обучения к временны*м* рядам. По ряду причин глубокое обучение является еще более гибким средством.

- Многие алгоритмы машинного обучения являются ограниченными с точки зрения размерности и типов входных данных. Напротив, глубокое обучение является очень гибким в отношении выбора модели и природы входных данных.
- Неоднородные данные могут вызывать трудности в обработке с помощью многих широко применяемых методов машинного обучения, тогда как в моделях глубокого обучения они используются довольно часто.
- Модели машинного обучения редко создаются для анализа временны*х* рядов, тогда как глубокое обучение предлагает большую гибкость при разработке архитектур, специально предназначенных для обработки временны*х* данных.

Тем не менее глубокое обучение не является панацей в анализе временны*х* наборов данных. Хотя в глубоком обучении, применяемом к временны*м* рядам, требование стационарности не выдвигается, на практике глубокое обучение не позволяет хорошо согласовать данные с трендом, если стандартные архитектуры не модифицированы под этот тренд. Следовательно, все еще нужно предварительно обрабатывать исходные данные или подвергать модификации сам метод.

Кроме того, глубокое обучение лучше всего работает с числовыми входными данными, поступающими из разных каналов, которые масштабируются до одинаковых значений в диапазоне от –1 до 1. Это еще одно требование в пользу предварительной обработки данных, хотя в строго теоретическом изложении метода ее выполнять не обязательно. Более того, входные данные необходимо предварительно обработать таким образом, чтобы избежать упреждения, но

сообщество глубокого обучения в целом обращает на эту проблему не слишком много внимания.

Наконец, методы оптимизации глубокого обучения и моделирования для ориентированных на обработку временны*х* данных нейронных сетей (самый большой класс из которых — рекуррентные нейронные сети (Recurrent Neural Networks — RNN)) не так хорошо развиты, как равнозначные инструменты обработки изображений (самый большой класс среди которых — сверточные нейронные сети, или (Convolutional Neural Networks — CNN)). Это означает, что вы найдете значительно меньше информации по методам оптимизации и практическим правилам выбора и обучения архитектур, предназначенных для решения задач обработки временны*х* данных, в отличие от таковых, основанных на работе с "не временны*ми*" данными.

Помимо этих трудностей применения глубокого обучения к временны*м* рядам, существуют и другие проблемы. Для начала отметим, что эффективность глубокого обучения для временны*х* рядов не всегда превосходит традиционные методы прогнозирования и классификации временны*х* рядов. Действительно, прогнозирование — это предметная область и дисциплина, дальнейшее развитие которой неотрывно связано с технологиями глубокого обучения, хотя на момент написания статьи серьезных достижений в этом направлении пока не наблюдалось.

Тем не менее от использования алгоритмов глубокого обучения в анализе временны*х* рядов стоит ожидать серьезных достижений как в близкой, так и долгосрочной перспективе. Во-первых, крупные технологические компании начали применять глубокое обучение для анализа временны*х* рядов в своих службах с собственной архитектурой, которая зачастую разрабатывается с учетом отраслевых задач моделирования. Вы можете использовать такие службы в собственных целях или комбинировать их с другими методами анализа, чтобы добиться действительно хорошей производительности.

Во-вторых, вполне возможно, что ваши наборы данных временны*х* рядов в исходном виде хорошо подходят для обработки методами глубокого обучения. В целом, чем больше соотношение "сигнал/шум", тем лучше проявляет себя глубокое обучение. На данный момент ко мне приходят сообщения от многих начинающих программистов о получении удивительно хороших результатов с помощью относительно простых инструментов глубокого обучения.

Например, исследователи одного из колледжей обнаружили, что простой метод LSTM (см. далее) так же хорошо, как и перегруженные работой преподаватели, предсказывает, какие студенты могут потерпеть неудачу при сдаче экзаменов или даже бросить учебу, что позволяет руководству колледжа заранее связаться с такими студентами и предоставить специальные условия обучения. Несмотря на то, что лучшим решением все же признается увеличение количества

преподавателей-кураторов, отрадно понимать, что даже простой метод LSTM, применяемый к разнородным временны*м* рядам, включающим одни только оценки учащихся и показатели посещаемости, способен выявить слабых студентов, указывая на необходимость дополнительной их поддержки. В будущем от такого инновационного способа применения глубокого обучения для анализа временны*х* рядов стоит ожидать намного более серьезных результатов.

Помните, что каждая модель основана на своих предположениях. Модели машинного обучения, включая нейронные сети, неизменно имеют встроенные допущения как в архитектуре, так и в методах обучения. Уже тот факт, что большинство нейронных сетей лучше всего работают с входными данными, масштабированными на отрезке [–1, 1], говорит о том, что в модель заложены сильные предположения, даже если они не сформулированы явным образом.

Возможно, область прогнозирования методами нейронных сетей еще не достигла своего предела в развитии, и более глубокое понимание их теоретических основ и требований позволит добиться существенного прироста в точности.

Если вы только начинаете изучать технологии глубокого обучения, то примите к сведению, что эта глава не даст вам полного представления обо всех концепциях и программном обеспечении, необходимом для успешного начала их освоения. Тем не менее она послужит хорошей отправной точкой на этом пути. Существует множество хороших учебных пособий, руководств и даже онлайн-курсов, которые вы можете использовать для овладения методами глубокого обучения. Хорошая новость о методах глубокого обучения состоит в том, что вам не обязательно хорошо знать математику, чтобы получить общее представление о принципах их работы и реализации в программном коде. Кроме того, вам доступны многочисленные специализированные интерфейсы прикладного программирования (API), представляющие механизмы глубокого обучения. Это означает, что начинающие исследователи могут использовать такие высокоуровневые API для опробования базовых методов, и по большому счету они часто применяются даже опытными специалистами в целях экономии рабочего времени и усилий. Впоследствии, по мере возникновения необходимости в конкретных архитектурных изменениях и представлении о преимуществах низкоуровневых API, которым приходится уделять намного больше внимания, вы все больше будете склоняться к использованию именно их.

В этой главе представлен краткий обзор математических концепций, лежащих в основе глубокого обучения, а также приведены примеры кода, который вы можете использовать в собственных целях — для обработки временны*х* данных с помощью известных алгоритмов глубокого обучения.

Концепции глубокого обучения

Глубокое обучение имеет связь со многими предметными областями. В его возникновении есть определенные биологические предпосылки, поскольку разработчики — преимущественно специалисты по компьютерным наукам и вычислительной математике — постоянно задавались вопросом, смогут ли интеллектуальные устройства имитировать работу человеческого мозга, обеспечивая активацию сети нейронов в ответ на срабатывание определенных триггеров. У глубокого обучения также есть математическое обоснование, представленное теоремой об универсальной аппроксимации, которая была доказана для различных функций активации, причем многие из доказательств были выполнены в конце 1980-х и начале 1990-х годов. Наконец, быстрый рост производительности вычислительного оборудования, его повсеместная доступность и постоянно возрастающий интерес к машинному обучению стали причиной успехов, доказавших, что при достаточном количестве данных и параметров моделированию и прогнозированию подлежат очень сложные системы. Глубокое обучение еще больше расширило идею нейронной сети, позволив создать структуры, описываемые миллионами параметров и обучаемые на больших наборах данных. Они предоставили теоретическое обоснование возможности представления с помощью нейронной сети произвольной нелинейной функции с достаточной точностью. На рис. 10.1 показана простейшая нейронная сеть — многослойный *персептрон* (или полносвязная сеть).

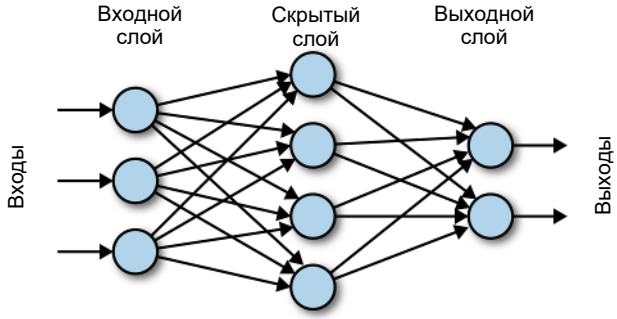


Рис. 10.1. *Простая сеть прямого распространения*

На рис. 10.1 показано, что многоканальные входы представлены в модели в виде вектора размерности d. Узлы задают входные значения, а ребра — соответствующие весовые множители. Значение каждого ребра, входящего в узел, рассчитывается через предыдущее значение, умноженное на весовой коэффициент этого ребра. Впоследствии значения ребер, поступающих в узел со всех входов, суммируются, а затем, как правило, передаются в функцию активации, устанавливающую степень нелинейности системы.

Как видите, вход включает три канала, описываемых вектором с длиной, равной 3. В сети также присутствуют четыре скрытых элемента. Следовательно, каждый из трех входов нужно умножить на различные весовые коэффициенты каждого из четырех скрытых элементов, к которым он направлен. Это означает, что нам потребуется $3 \cdot 4 = 12$ весовых коэффициентов, чтобы полностью описать систему. Кроме того, поскольку полученные произведения подлежат итоговому суммированию, все необходимые вычисления проще всего проводить в матричном виде.

При пошаговом описании алгоритм выглядел бы примерно следующим образом.

1. Входной вектор $\mathbf{X}$ включает три элемента. Веса первого слоя обозначены $\mathbf{W}_1$ и образуют матрицу размером 4×3, так что для получения значений скрытого слоя необходимо вычислить произведение $\mathbf{W}_1\mathbf{X}_1$. Его результатом будет матрица размером 4×1, но она не будет представлять выход скрытого слоя.

   $$\mathbf{W}_1\mathbf{X}_1$$

2. Нужно добавить в систему нелинейность, которая имитируется с помощью самых разных функций активации, таких как гиперболический тангенс (tanh) или сигмоидальная функция (σ). Обычно в самой функции активации также задается смещение $\mathbf{B}_1$, поэтому полный выход скрытого слоя рассчитывается следующим образом.

   $$\mathbf{H} = a(\mathbf{W}_1\mathbf{X}_1 + \mathbf{B}_1)$$

3. Нейронная сеть, изображенная на рис. 10.1, имеет два выхода для предсказания. Следовательно, нужно привести четырехмерный выход скрытого состояния к двум выходам конечного состояния. Как правило, последний слой не снабжается нелинейной функцией активации, за исключением использования для решения задачи классификации многопеременной логистической функции. Давайте полагать, что нам нужно предсказать два числа, а не вероятностные категории, что позволяет "уплотнить" последний слой — объединить четыре выхода скрытого слоя в единый результат. В нашем случае последний плотный слой представлен двумя выходами, сведенными из четырех, для чего применяется матрица $\mathbf{W}_2$ размером 2×4.

   $$\mathbf{Y} = \mathbf{W}_2\mathbf{X}$$

Надеюсь, приведенное выше описание дало вам хотя бы общее представление о работе моделей глубокого обучения. Общая идея состоит в том, чтобы иметь множество параметров и возможностей для описания нелинейных систем.

Выбор правильного числа и конфигурации параметров, правильные обучающие гиперпараметры и, конечно, разумно поставленная задача больше сродни искусству, чем науке. Секрет заключается в том, чтобы научиться настраивать параметры модели, выработать правильный подход к их инициализации и убедиться в движении модели в правильном направлении — к приемлемому решению. Это невыпуклый класс моделей, цель применения которых не сводится к нахождению глобального оптимума. Вместо этого поиск продолжается до тех пор, пока не будет найден хорошо продуманный способ упорядочения модели, представленный "достаточно правдоподобным" локальным оптимумом, который полностью соответствовал бы вашим потребностям.

Программирование нейронной сети

Понимание принципов работы нейронных сетей может оказаться несколько более простой задачей, чем изучение фреймворков и программных инструментов, применяемых для их реализации. Как вы увидите далее, несмотря на широкое разнообразие все они имеют много общего.

Символическое и императивное программирование

При программировании глубокого обучения перед вами встанет задача выбора стиля программирования: символического или императивного. В символическом программировании все отношения объявляются заранее, не подвергаясь вычислению во время объявления. Напротив, в императивном программировании вычисления выполняются по мере прохода по коду — построчно, не дожидаясь объяснения того, что произойдет позже. У каждого стиля есть свои достоинства и преимущества, и между ними нет строгого разделения. Перечислим некоторые особенности стилей программирования, о которых нужно помнить, выбирая один из вариантов.

- Код символического стиля программирования, как правило, более производительный, поскольку лишен строгого порядка выполнения и сохраняет возможность оптимизации за фреймворком.
- Код императивного стиля программирования проще для понимания и отладки.
- Фреймворки `TensorFlow` и `MXnet` тяготеют к символическому стилю программирования, в то время как фреймворк `Torch` носит более императивный характер.

Данные, символы, операции, слои и графы

Работа фреймворков глубокого обучения часто концентрируется вокруг некого представления графа и методов его построения. Идея состоит в том, чтобы описать предложенную структуру по отдельным ее компонентам и взаимоотношениям между ними. Кроме того, в них преобладает идея отрыва переменных от их фактических значений. Остановившись на ней, представим, что модель оперирует понятиями символов `A` и `B`, а также третьего символа `C`, представляемого через произведение матриц `A` и `B`.

```
# Псевдо-коде
symbol A;
symbol B;
symbol C = matmul(A, B);
```

Различие между символами и данными играет важную роль в каждой структуре из-за связи между символом и данными. Символ используется для изучения общих взаимоотношений, а данные могут иметь шум. Существует только один символ `A`, хотя у нас могут быть миллионы или даже миллиарды значений, которые может принимать переменная `A` в паре с соответствующими переменными `B` и `C`.

Популярные фреймворки

Как и во многих других высокотехнологических отраслях, карта предложений в мире глубокого обучения постоянно меняется. И так же, как в деловом мире, здесь часто происходят непродуманные слияния и поглощения, как, например, в случае совмещения `TensorFlow` и `Theano`, а также `MXNet` и `Caffe`. Ни один из фреймворков не зарекомендовал себя в качестве лидера на рынке технологий глубокого обучения для временны*х* рядов, хотя `MXNet` может оказаться лучшим выбором в ближайшем будущем, учитывая высокую заинтересованность компании Amazon в инструментах прогнозирования, подобных недавно представленной службе глубокого обучения для анализа временны*х* рядов. Как человек, которому часто приходится писать собственные итераторы данных для проектов по анализу временны*х* рядов с помощью методов глубокого обучения, я обнаружила, что проще всего выполнять эту задачу в `MXNet`.

На момент написания книги наиболее известными и широко используемыми для начинающих специалистов считались следующие библиотеки (в скобках указан разработчик).

- `TensorFlow` (Google)
- `MXNet` (Amazon)
- `Torch` (Facebook)

Тем не менее в академических и открытых источниках часто предлагаются совершенно другие, более новые решения, оптимизированные для задач отдельной предметной области или как минимум улучшающие существующую структуру. Доминирование на рынке рано и поздно заканчивается, хотя в силу выпуска технологическими гигантами оборудования, работающего исключительно с программным обеспечением их собственной разработки, лидеры рынка, скорее всего, будут удерживать свои позиции несколько дольше, чем в предыдущие годы.

Если спросить о том, зачем нужны данные, то можно получить ответ: "Чтобы совершать над ними определенные операции", например добавлять или умножать символы, представляя эти действия как операции. Кроме того, с ними можно совершать одномерные операции, такие как изменение формы символа (в частности, приведение изображения символа A из матрицы 2×4 к матрице 8×1) или передача значений функции активации, такой как tanh(A).

Поразмыслив еще, слои можно представить в виде элементов общих структур, обрабатываемых с помощью стандартных методик, например полносвязных слоев, описанных в предыдущем разделе. Принимая во внимание функцию активации и смещение, а также структуру матрицы ядра, можно сформулировать следующее утверждение.

Слой **L** = tanh(**AB** + смещение)

Во многих фреймворках такой слой выражается не с помощью нескольких операций, а через один или два слоя, в зависимости от того, достаточно ли репрезентативна комбинация операций, чтобы представить отдельный слой.

Наконец, несколько слоев можно связать между собой, подставив один в другой.

Слой **L1** = tanh(**AB** + смещение_1)

Слой **L2** = tanh(**L1D** + смещение_2)

Такой конгломерат символов, операций и слоев представляет собой граф. Граф не обязательно должен быть полностью связанным, т.е. не обязательно, чтобы все символы зависели друг от друга. Граф используется для того, чтобы разобраться, как символы зависят от других символов и от каких именно. Это важно для понимания того, как лучше настраивать веса в каждой итерации для получения лучшего решения при вычислении градиентов в методе градиентного спуска. Приятно то, что большинство современных пакетов глубокого обучения полностью автоматизируют большую часть работы. Нам не нужно указывать, что от чего зависит и как меняется градиент с каждым добавленным слоем, — все это делается фреймворком самостоятельно.

Более того, как указывалось ранее, всегда можно обратиться к высокоуровневым программным интерфейсам, в которых не нужно заниматься перемножением матриц, как показано в предыдущем примере. Для работы с полносвязными слоями, описываемыми с помощью уже известной вам процедуры суммирования произведений матриц с последующей поэлементной обработкой в функции активации, можно отказаться от выписывания всех необходимых математических выражений. Например, в пакете `MXNet` эта сложная задача реализуется с помощью следующего простого кода.

```
## Python
>>> import mxnet as mx
>>> fc1 = mx.gluon.nn.Dense(120, activation='relu')
```

Результатом его выполнения будет полносвязный слой, который преобразует входные данные в выходное измерение размером 120. Кроме того, в единственной строке кода полностью реализуются все необходимые расчеты: перемножение матриц, последующее их суммирование и поэлементный проход функцией активации.

Вы можете удостовериться в этом, сверившись с документацией к пакету (`https://perma.cc/8PQW-4NKY`) или протестировав его на примерах входных и выходных данных, для которых известны весовые коэффициенты. Удивительно, но программный интерфейс выполняет задачу полностью автоматически. От вас не требуется указывать входную форму данных (хотя она должна быть постоянной после построения графа, чтобы иметь возможность получить конечный результат). Вы не обязаны определять тип данных — по умолчанию это будет наиболее предпочтительный и часто используемый `float32` (тип `float64` излишне громоздкий для глубокого обучения). Если исходные данные в одной из выборок представлены не в одномерной форме, то они будут автоматически сглажены для соответствующего слоя так, чтобы передаваться в полносвязный/плотный слой в должном формате. Автоматизированный программный интерфейс покажется хорошим решением для начинающих исследователей, но как только вы достигнете определенного уровня квалификации, обязательно пересмотрите документацию, чтобы досконально разобраться в выполняемых операциях — пусть даже в простейшей модели глубокого обучения.

В приведенном далее примере у вас не запрашивается почти ничего из того, что приходится определять при выполнении даже самой простой задачи глубокого обучения. Здесь от вас требуется определить только входные данные, конечные цели и то, какие ошибки будут учитываться. Вам также нужно указать, как слой должен размещаться в модели, для чего применяется следующий код.

```
## Python
>>> ## Создание сети, а не отдельного слоя
>>> from mx.gluon import nn
```

```
>>> net = nn.Sequential()
>>> net.add(nn.Dense(120, activation='relu'),
>>>         nn.Dense(1))
>>> net.initialize(init=init.Xavier())
>>>
>>> ## Определение ошибок
>>> L2Loss = gluon.loss.L2Loss()
>>>
>>> trainer = gluon.Train(net.collect_params(), 'sgd',
>>>                                    {'learning_rate': 0.01})
```

Наконец, предполагая, что данные уже настроены так, как нужно, можно переходить непосредственно к этапу обучения (совершить один проход по всем данным), как показано далее.

```
## Python
>>> for data,target in train_data:
>>>     ## Вычисление градиента
>>>     with autograd.record():
>>>         out = net(data)
>>>         loss = L2Loss(output, data)
>>>     loss.backward()
>>>      ## Применение градиента для обновления параметров
>>> trainer.step(batch_size)
```

В большинстве пакетов, как и в `mxnet`, можно легко сохранить используемую модель и соответствующие параметры для последующего применения в других производственных процессах или для дополнительного обучения.

```
## Python
>>> net.save_parameters('model.params')
```

В последующих примерах мы будем использовать программный интерфейс `Module` пакета `mxnet`, чтобы продемонстрировать другие способы создания графов и обучающих моделей.

Как специалист по глубокому обучению вы в конечном итоге придете к знакомству со всеми основными пакетами и их наиболее популярными программными интерфейсами (как правило, каждый пакет имеет не менее одного высокоуровневого и одного низкоуровневого API), что позволит вам легко изучать любые примеры кодов. Знание основных пакетов глубокого обучения понадобится вам, чтобы не отставать от развития отраслевых технологий и быть в курсе тематических академических исследований, а эту информацию проще всего почерпнуть из открытых источников кода.

Автоматическое дифференцирование

Магия моделей глубокого обучения — помимо впечатляющей производительности — заключается в способности обучения чему угодно. Стандартные модели могут иметь тысячи или даже миллионы параметров, и эти параметры должны быть адаптированы к конкретной задаче. Как правило, это предполагает задействование сложных механизмов символического дифференцирования, которые предоставляются специальными фреймворками глубокого обучения. В последние десятилетия большинство математиков, физиков и исследователей других дисциплин, сталкивающихся с необходимостью оптимизации функций, полагались на методы численной оптимизации и дифференцирования, а это означает, что градиент определялся ими эмпирически (скажем, с помощью зависимости вида $(f(x + \delta) - f(x)) / \delta$). Преимуществом такого подхода было отсутствие необходимости в обучении компьютеров дифференцированию, выполняемому в явном виде, а недостаток был всего один — низкая скорость вычислений, и повысить ее не позволял даже тот факт, что все необходимые математические зависимости давно известны или специально установлены.

Автоматическое дифференцирование основано на фактической известности математических выражений, которые описывают взаимосвязь переменных, относящихся к разным слоям, а значение градиента рассчитывается согласно таким выражениям, а не более вычислительно затратным методам численного дифференцирования. В частности, в автоматическом дифференцировании используется тот факт, что в компьютерных алгоритмах, даже в большей степени, чем в математике, любые сложные математические выражения можно представить последовательностью более простых операций, правила дифференцирования которых заведомо известны, а потому к ним можно в полной мере применять правила дифференцирования сложных функций. В автоматическом дифференцировании такой подход применяется для нахождения математических выражений вычисления градиентов. Как только градиенты будут получены, можно переходить к установке весовых значений в разные моменты времени, тем самым улучшая и обучая модель.

Создание конвейера обучения

В этом разделе мы займемся моделированием одного и того же набора данных, содержащего измерения почасового потребления электроэнергии в различных населенных пунктах за последние несколько лет. Данные будут подвергаться предварительной обработке так, чтобы предоставлять сведения о потреблении электроэнергии ежечасно, что является более сложной задачей, чем прогнозирование суммарных значений, поскольку требует анализа наиболее непредсказуемых диапазонов временного ряда.

Исследование набора данных

Воспользуемся открытым репозиторием данных, в котором представлены почасовые измерения электропотребления, включенные в кодовую базу, представляющую архитектуру обновленной нейронной сети (описана далее).[1] Чтобы понять, как выглядят исходные данные, считаем их с помощью инструментов языка R и построим несколько быстрых графиков.

```
## R
> elec = fread("electricity.txt")
> elec
      V1  V2  V3  V4  V5   V6 V7   V8  V9 V10 V11 V12 V13 V14 V15 V16 V17 V18
   1: 14  69 234 415 215 1056 29  840 226 265 179 148 112 171 229 1001 49 162
   2: 18  92 312 556 292 1363 29 1102 271 340 235 192 143 213 301 1223 64 216
   3: 21  96 312 560 272 1240 29 1025 270 300 221 171 132 185 261 1172 61 197
   4: 20  92 312 443 213  845 24  833 179 211 170 149 116 151 209 813  40 173
   5: 22  91 312 346 190  647 16  733 186 179 142 170  99 136 148 688  29 144
```

Быстрый просмотр позволяет понять, с каким количеством строк и столбцов придется работать. Заметьте, что значения не снабжены временны*ми* метками, но мы знаем, что данные собираются ежечасно, хотя достоверно не известно, когда именно проводились измерения.

```
## R
> ncol(elec)
[1] 321
> nrow(elec)
[1] 26304
```

Визуализируем нескольких случайных выборок данных, чтобы получить представление о том, как они выглядят (рис. 10.2). Поскольку визуализации подлежат почасовые данные, будем считать, что суточные данные представляют 24 точки/значения.

```
## R
> elec[125:148, plot(V4,    type = 'l', col = 1, ylim = c(0, 1000))]
> elec[125:148, lines(V14,  type = 'l', col = 2)]
> elec[125:148, lines(V114, type = 'l', col = 3)]
```

Также построим график за недельный срок (рис. 10.3).

```
## R
> elec[1:168, plot(V4,    type = 'l', col = 1, ylim = c(0, 1000))]
> elec[1:168, lines(V14,  type = 'l', col = 2)]
> elec[1:168, lines(V114, type = 'l', col = 3)]
```

[1] Файл можно загрузить с репозитория поставщика на GitHub (`https://github.com/laiguokun/multivariate-time-series-data/raw/master/electricity/electricity.txt.gz`).

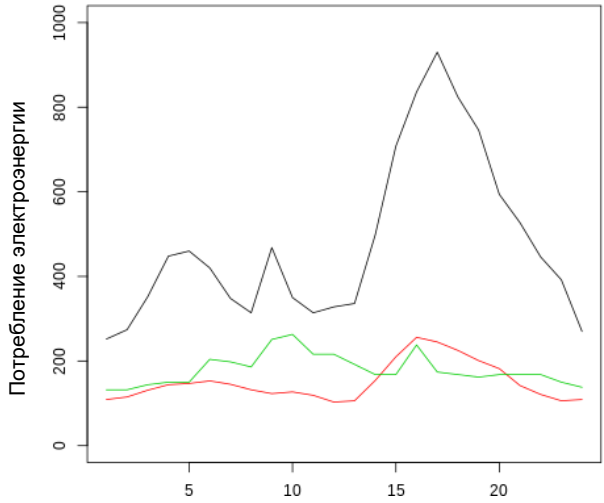


Рис. 10.2. *Выборка для трех различных населенных пунктов из 321 возможного варианта, включенного в исходный набор данных, за 24 часа. Несмотря на отсутствие информации о времени сбора показателей график представляет последовательную дневную модель*

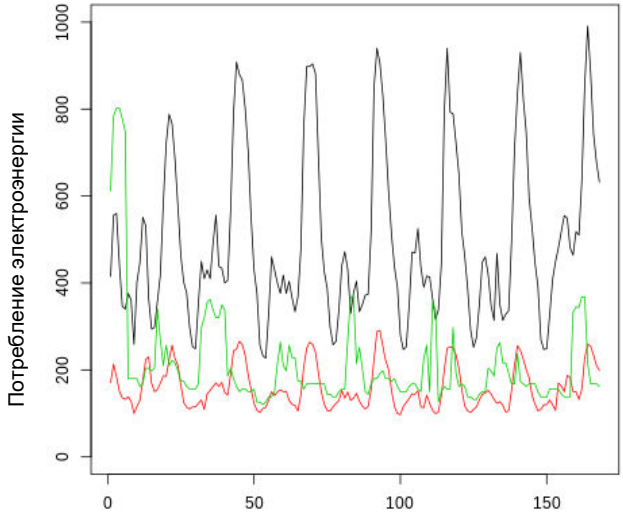


Рис. 10.3. *Полный семидневный цикл выборочных данных для тех же трех населенных пунктов. График подтверждает предположение о существовании дневного шаблона и указывает на присутствие в данных характерных ежедневных больших пиков, которые сопровождаются более мелкими выбросами*

Заметьте, что нам ничего не известно о том, как временные точки согласуются с местным временем каждого населенного пункта, но хорошо известно соотношение между ними. Тем не менее, в соответствии с общими рассуждениями, приведенными в главе 2, можно легко догадаться, какому времени суток

соответствуют приведенные графики потребления электроэнергии. На них даже можно распознать выходные дни. Но мы не будем этого делать, хотя всегда неплохо исследовать данные в привязке ко времени суток и дням недели, чтобы получить более полное представление о поведении временного набора.

Очевидно, прогноз можно построить по абсолютным значениям, но это уже было сделано как в научных работах, так и многими блогерами. Давайте лучше предсказывать поведение по разности данных. Прогнозирование разностей, а не общих значений временного ряда, как правило, носит более сложный характер, поскольку они более зашумленные, что прекрасно видно на графиках, подобных отображаемым ранее (рис. 10.4).[2]

```
## R
> elec.diff[1:168, plot( V4, type = 'l', col = 1, ylim = c(-350, 350))]
> elec.diff[1:168, lines(V14, type = 'l', col = 2)]
> elec.diff[1:168, lines(V114, type = 'l', col = 3)]
```

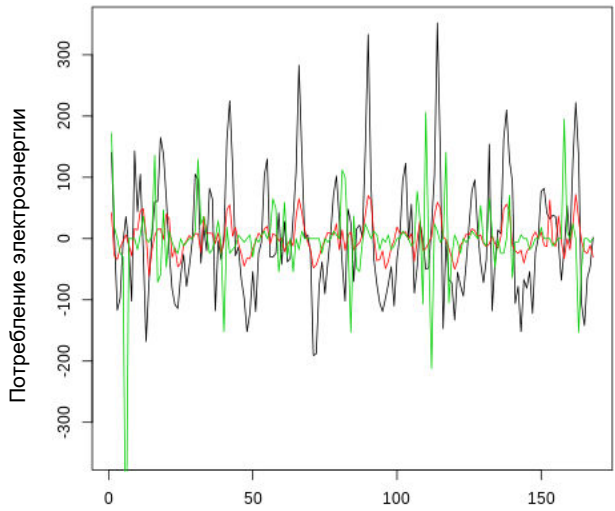


Рис. 10.4. *Недельная выборка временного ряда разностей потребляемой электроэнергии для трех населенных пунктов, представляющего ее почасовое изменение. Хотя в ряде разностей, как и в исходном ряде, наблюдается определенный шаблон, непрогнозируемые компоненты становятся более понятными, поскольку оказывают большее влияние на разностный ряд по сравнению с исходными данными*

Для того чтобы использовать традиционную статистическую модель — пространства состояний или даже машинного обучения, — на этом этапе нам пришлось бы провести более детальный анализ данных для оценки корреляции

[2] Здесь код вычисления разности не приводится, поскольку он рассматривается в главе 11 и на самом деле представлен всего несколькими простыми инструкциями.

между отдельными частями набора. Нам нужно было бы оценить, наблюдается ли в данных временной дрейф, и оценить их стационарность.

Такие же операции нужно выполнить и в глубоком обучении, чтобы подобрать подходящие модели для набора данных и определиться с тем, насколько точно они должны их описывать. Тем не менее модели глубокого обучения позволяют работать с намного более запутанными данными, а также без предварительного выполнения указанного выше статистического анализа. В производственной среде исследование данных отнимает намного больше времени, чем в приведенных далее примерах, а в этой главе мы можем сразу перейти к рассмотрению отдельных вариантов моделирования.

Этапы конвейера обучения

Как правило, сценарии моделирования с помощью нейронной сети включают нескольких стандартных этапов. Зачастую такие сценарии намного сложнее в реализации, чем в случае статистических или традиционных моделей машинного обучения, поскольку их целевые наборы данных обычно имеют заметно больший размер. Кроме того, обучение глубоких моделей выполняется пакетным способом с использованием итераторов, а не массивов полных наборов данных.

Конвейер данных включает следующие производственные этапы.

- Создание легко настраиваемого кода путем импортирования в него списка заранее оговоренных значений для параметров обучения, принятых по умолчанию. Это очень важная операция при определении слишком большого количества значений.
- Загрузка в память и предварительная обработка данных.
- Приведение данных к требуемому формату.
- Создание итераторов, соответствующих используемому модулю глубокого обучения.
- Построение графа, в котором применяются итераторы, для понимания ожидаемой формы данных; предполагается полное построение модели.
- Настройка параметров обучения, например оптимизации, скорости и длительности обучения.
- Создание системы учета как весов, так и результатов поэтапных вычислений.

Создание легко настраиваемого кода

В следующем коде показано, как можно реализовать перечисленные выше задачи. Начнем с команд импорта стандартных программных пакетов.

```
## Python
>>> from math import floor
>>>
>>> ## Пакеты для получения данных
>>> import os
>>> import argparse
>>>
>>> ## Модуль глубокого обучения
>>> import mxnet as mx
>>>
>>> ## Пакеты для обработки данных
>>> import numpy as np
>>> import pandas as pd
>>>
>>> ## Вывод результатов
>>> import perf
```

Далее нужно определиться с комбинацией задаваемых в коде переменных и настраиваемых параметров. В какой-то степени это вопрос опыта и личных предпочтений, выработанных при обучении. Не расстраивайтесь, если названия переменных покажутся вам бессмысленными, ведь многие из настраиваемых параметров применяются в компонентах нейронной сети, которые нам только предстоит обсудить. Главное — отметить настраиваемость таких параметров.

```
## Python
>>> ## Отдельные гиперпараметры, не задаваемые из командной строки
>>> DATA_SEGMENTS    = { 'tr': 0.6, 'va': 0.2, 'tst': 0.2}
>>> THRESHOLD_EPOCHS = 5
>>> COR_THRESHOLD    = 0.0005
>>>
>>> ## Настройка парсера
>>> parser = argparse.ArgumentParser()
>>>
>>> ## Формирование данных
>>> parser.add_argument('--win',      type=int,  default=24*7)
>>> parser.add_argument('--h',        type=int,  default=3)
>>>
>>> ## Спецификация моделей
>>> parser.add_argument('--model',    type=str,    default='rnn_model')
>>>
>>> ## Компоненты cnn
>>> parser.add_argument('--sz-filt',  type=str,    default=8)
>>> parser.add_argument('--n-filt',   type=int,    default=10)
>>>
>>> ## Компоненты rnn
>>> parser.add_argument('--rnn-units', type=int,   default=10)
>>>
>>> ## Параметры обучения
>>> parser.add_argument('--batch-n',   type=int,   default=1024)
```

```
>>> parser.add_argument('--lr',         type=float, default=0.0001)
>>> parser.add_argument('--drop',       type=float, default=0.2)
>>> parser.add_argument('--n-epochs',   type=int,   default=30)
>>>
>>> ## Архивирование
>>> parser.add_argument('--data-dir' , type=str,   default='../data')
>>> parser.add_argument('--save-dir' , type=str,   default=None)
```

Критически важно располагать большим количеством настраиваемых параметров, поскольку обучение модели глубокого обучения всегда предполагает поиск гиперпараметров, способных улучшить ее относительно базового уровня. Обычно настраиваемые гиперпараметры влияют на все аспекты обучения, от подготовки данных (как далеко нужно назад заглянуть во времени) до спецификации модели (насколько она сложная и какой имеет вид) и сложности обучения (насколько долго оно ведется и с какой скоростью).

В предыдущий код включены разные категории параметров. Во-первых, формирование данных зависит от процедуры получения необработанных данных — в нашем случае из CSV-файла, содержащего параллельные разности временны*х* рядов потребления электроэнергии, взятые из 321 станций. Чтобы сформировать наши данные, нам понадобится два параметра. Переменная окна — насколько далеко моделям позволено смотреть в прошлое при составлении прогнозов. Переменная горизонта — насколько далеко в будущее делается прогноз. Обратите внимание, что обе переменные относятся не к точным временны*м* значениям, таким как "5 минут", а скорее к временны*м* шагам, в соответствии с практикой, выработанной в предыдущих главах. Как и другие статистические и машинные модели обучения, нейронные сети заботятся о вычислительном представлении и не видят разницы между 5 минутами и 5 миллиардами лет при просмотре данных.

Предпоследний раздел, в котором задаются настройки обучения, обычно самый важный для оптимизации гиперпараметров, а потому наиболее часто подвергаемый правкам. С самого начала очень важно поэкспериментировать со скоростью обучения и убедиться, что выбрано удачное значение. Хорошее эмпирическое правило — начинать с 0,001, а затем переходить вверх и вниз с шагом в несколько порядков. Важно иметь не столько правильную скорость обучения, сколько правильный порядок величин.

Параметризация моделей позволяет получить различные виды сетей (например, RNN или CNN) и задать их структурные особенности. В общем, пренебрегать настройкой гиперпараметров не сто*и*т.

В представленных ниже примерах будут использованы следующие гиперпараметры, переданные сценарию из командной строки.

```
--drop=0.2 --win=96 --batch-n=128 --lr=0.001 --n-epochs=25
--data-dir=/data/elec --save-dir=/archive/results
--model=model_will_vary
```

Подготовка входных данных

Настроив параметры, нужно сообщить сценарию, где искать файл, насколько далеким должен быть прогноз и как далеко можно заглядывать в прошлое при изучении данных. Такие настройки будут важны даже на этапе предварительной обработки данных: считывании и придании правильной формы. Нам также необходимо образовать инфраструктуру обработки данными, потому что в обучении нейронных сетей используются разные методы стохастического градиентного спуска, а это значит, что оно выполняется на небольших пакетах данных за один проход, за который для обучения используются полностью все данные (хотя и не все одновременно).

Далее нам предстоит обсудить высокоуровневый процесс предоставления данных для обучения с помощью итераторов и принципы низкоуровневого формирования данных, передаваемых таким итераторам.

Формирование входных данных. В предыдущем разделе вы познакомились с построением итераторов на основе массивов `NumPy` и приняли такое их представление как должное. Далее мы обсудим принципы формирования данных, сначала — концептуально, а затем — в виде практического примера. Будем рассматривать два формата данных — NC и NTC.

Начнем изучение форматов входных данных с рабочего примера, который не имеет ничего общего с данными, приведенными в коде. Представим данные некого многомерного временного ряда со столбцами *A*, *B* и *C*.

Время	*A*	*B*	*C*
t – 3	0	–1	–2
t – 2	3	–2	–3
t – 1	4	–2	–4
t	8	–3	–9

Нам нужно построить модель, которая умеет предсказывать на один шаг вперед, и использовать для прогнозирования данные за два предыдущих момента времени. Кроме того, прогнозирование переменных *A*, *B* и *C* должно выполняться на основе данных переменных *A*, *B* и *C*. Обозначим входы символом *X* и выходы — символом *Y*.

Как предсказать *Y* в момент времени *t*? Для момента времени *t* прогноз будет представлен фактическими значениями *Y* = [*A*, *B*, *C*].

t 8 –3 –9

Как было показано, прогноз всех переменных будет делаться на основе значений переменных за два предыдущих момента времени.

A, t – 1	A, t – 2	B, t – 1	B, t – 2	C, t – 1	C, t – 2
4	3	–2	–2	–4	–3

Аналогично, чтобы предсказать Y в момент времени $t - 1$, нужно исходить из следующих целевых данных.

$t - 1$ 4 –2 –4

Ожидается, что для прогнозирования будут доступны следующие значения.

A, t – 1	A, t – 2	B, t – 1	B, t – 2	C, t – 1	C, t – 2
3	0	–2	–1	–3	-2

Получаем следующий общий формат представления входных данных для обеих точек.

Время	A, время – 1	A, время – 2	B, время – 1	B, время – 2	C, время – 1	C, время – 2
$t - 1$	3	0	–2	–1	–3	-2
t	4	3	–2	–2	–4	-3

Такой способ описания многомерной информации называется форматом данных NC, где N — отдельные выборки, а C — канал. Мы будем использовать этот формат данных для обучения полностью связанной нейронной сети, и это первый вариант метода, в котором входные данные поступают в формате CSV и преобразуются в массивы `NumPy` правильной формы и размерности.

Наряду с этим данные можно сформировать по-другому, ориентируясь на специально образованную ось времени. Обычно это делается путем представления данных в формате NTC, в котором N обозначает количество выборок, T — время, а C — каналы. В этом случае выборкой считается каждая строка исходных данных, т.е. каждый срез во времени, для которого нужно сделать прогноз (и для которого представлены данные, позволяющие это сделать). Показатель времени устанавливает, насколько далеко назад во времени будет проводиться анализ. В данном примере он равен двум шагам (и обозначается параметром `--win` в сценарии для этой главы).

В формате NTC исходные данные, с которыми мы работали ранее, для прогнозирования горизонта $t - 1$ выглядели бы примерно так.

Время	A	B	C
$t - 1$	0, 3	–1, –2	–2, –3

Или если нужно задать входные данные для обеих предыдущих выборок, то проще всего поступить следующим образом.

Время	A	B	C
$t-1$	0, 3	–1, –2	–2, –3
t	3, 4	–2, –2	–3, –4

Эти значения можно объединить с подписями, которые создавались для Y.

Время	A	B	C
$t-1$	4	–2	–4
t	8	–3	–9

Ни одно из приведенных выше NTC-представлений не точнее других, но все они обладают важной особенностью, которая выражается в явной направленности значений вдоль временной оси.

Причина, по которой используются сразу оба формата входных данных, заключается в том, что каждый из них лучше подходит для своего типа модели. Формат NTC будет применяться для передачи данных в сверточные и рекуррентные нейронные сети, которые мы подробно обсудим несколько позже.

NTC или TNC

Есть несколько особенностей, которые стоит отметить в только что описанном формате хранения данных NTC. Во-первых, он избыточный. Это ограничение можно обойти, если порядок данных не важен, используя формат TNC вместо NTC. В формате TNC можно задействовать свою "пакетную" ось для параллельного обхода различных частей данных. Так, например, в приведенной ниже последовательности чисел одновременно обучается три пакета.

56 29 56 94 10 92 52 32 19 59 88 94 6 57 73 59 95 79 97 38 65 51 27

18 77 39 4 19 60 38 73 51 65 4 96 96 6 12 62 59 21 49 65 37 64 69

36 32 48 97 33 44 63 99 10 56 75 20 61 53 71 48 41 2 58 18 4 10 17

66 64 53 24 36 23 33 38 1 17 59 11 36 43 61 96 55 21 45 44 53 26 55

99 22 10 26 25 82 54 82

Если подготовить их в формате NTC с одним каналом, то данные будут выглядеть так.

Время	A
t	56, 29, 56, 94
$t-1$	29, 56, 94, 10
$t-2$	56, 94, 10, 92

Здесь много повторений, но хорошая новость заключается в том, что алгоритм будет обучаться в реальном хронологическом порядке следования данных, так как все пакеты тесно связаны во времени. Напротив, если выбрать формат TNC (N — это число выборок на пакет) и предположить, что размер пакета равен 4 (меньше некуда), то наши данные могут выглядеть следующим образом.

Время	Элемент пакета 1	Элемент пакета 2	Элемент пакета 3	Элемент пакета 4
t	29	77	32	64
$t+1$	56	39	48	53
$t+2$	94	4	97	24

Очевидно, что в данном случае повторений данных нет. Нужно ли проводить обучение на данных, упорядоченных в хронологическом порядке, зависит от набора данных. Те из вас, кому доводилось использовать методы глубокого обучения для обработки текстов на естественных языках (NLP), посчитают этот формат удобным для многих задач NLP, но он не всегда будет столь же эффективен в задачах анализа числовых значений временны*х* рядов.

Построение итераторов. В целом, чтобы передать данные в процедуру обучения, понадобятся итераторы. Итераторы не являются чем-то уникальным для глубокого обучения или языка Python, а скорее отражают общую идею объекта, который обходит коллекцию, отслеживая свое положение и сообщая, когда проход по коллекции будет закончен. Построить итератор в случае обучения на данных, передаваемых в виде массива `NumPy`, не составляет большого труда. Если `X` и `Y` выступают массивами `NumPy`, то получить итераторы очень просто.

```
## Python
>>> ################################
>>> ## ПОДГОТОВКА ДАННЫХ          ##
>>> ################################
>>>
>>> def prepare_iters(data_dir, win, h, model, batch_n):
>>>     X, Y = prepared_data(data_dir, win, h, model)
>>>
>>>     n_tr = int(Y.shape[0] * DATA_SEGMENTS['tr'])
>>>     n_va = int(Y.shape[0] * DATA_SEGMENTS['va'])
>>>
>>>     X_tr, X_valid, X_test = X[ : n_tr],
>>>                             X[n_tr : n_tr + n_va],
>>>                             X[n_tr + n_va : ]
>>>     Y_tr, Y_valid, Y_test = Y[ : n_tr],
>>>                             Y[n_tr : n_tr + n_va],
>>>                             Y[n_tr + n_va : ]
>>>
```

```
>>>     iter_tr = mx.io.NDArrayIter(data = X_tr,
>>>                                 label = Y_tr,
>>>                                 batch_size = batch_n)
>>>     iter_val = mx.io.NDArrayIter(data = X_valid,
>>>                                 label = Y_valid,
>>>                                 batch_size = batch_n)
>>>     iter_test = mx.io.NDArrayIter(data = X_test,
>>>                                 label = Y_test,
>>>                                 batch_size = batch_n)
>>>
>>>     return (iter_tr, iter_val, iter_test)
```

В этом коде итераторы для набора данных создаются специальным методом, после чего они проходят по массивам NumPy, полученным с помощью метода `prepare_data()` (подробнее об этом — далее). Как только массивы с данными становятся доступными, они разбиваются на источники для обучения, проверки и тестирования, причем данные обучения создаются первыми, данные проверки используется для настройки гиперпараметров с обратной связью вне выборки, а данные тестирования сохраняются для полноценного тестирования.[3]

Обратите внимание, что функция инициализации итератора принимает входные данные (`data`), целевое значение (`label`) и параметр `batch_size`, отражающий количество экземпляров, которые будут использоваться в итерации для вычисления градиентов и обновления весов модели.

Формирование данных в коде

Познакомившись с двумя форматами данных, которые будут обрабатываться в коде, рассмотрим код подготовки данных.

```
## Python
>>> def prepared_data(data_dir, win, h, model_name):

>>> df = pd.read_csv(os.path.join(data_dir, 'electricity.diff.txt'),
>>> sep=',', header=0)
>>> x = df.as_matrix()
>>> ## Нормализация данных. Она вызывает упреждение, поскольку
>>> ## основана на значениях, измеренных по всему набору.
>>> ## Чтобы предотвратить упреждение, в реальных приложениях
>>> ## следует применять скользящие статистики
>>> x = (x - np.mean(x, axis = 0)) / (np.std(x, axis = 0))
```

[3] Как обсуждалось ранее, золотой стандарт заключается в обучении и продвижении моделей вперед через большое количество временных периодов, но мы будем избегать такого усложнения в нашем коде. В реальных задачах приходится постоянно проверять и тестировать полученные результаты для правильной оптимизации модели и получения полного представления о ее действительной производительности. В приведенной здесь методике вводится дополнительное смещение между данными тестирования и обучения, а это означает, что данные тестирования, используемые для оценки модели, на самом деле отражают всего один период времени из всего временного диапазона.

```
>>>
>>> if model_name == 'fc_model': ## формат данных NC
>>>     ## Возврат на один и два шага назад во входных данных
>>>     X = np.hstack([x[1:-1], x[:-h]])
>>>     Y = x[h:]
>>>     return (X, Y)
>>> else: ## Формат данных TNC
>>>       # Предварительное размещение X и Y в памяти
>>>       # X = количество экземпляров * временное окно *
>>>       # * количество каналов (NTC)
>>>       X = np.zeros((x.shape[0] - win - h, win, x.shape[1]))
>>>       Y = np.zeros((x.shape[0] - win - h, x.shape[1]))
>>>
>>>       for i in range(win, x.shape[0] - h):
>>>           ## Целевое значение/подпись рассчитывается через
>>>           ## h шагов вперед
>>>           Y[I - win] = x[i + h - 1 , :]
>>>           ## Входными являются данные из предыдущего окна
>>>           X[i-win] = x[(i - win) : i , :]
>>>
>>>       return (X, Y)
```

После считывания данных из текстового файла проводится стандартизация каждого столбца. Обратите внимание, что столбцы стандартизируются раздельно, а не однородно по всему набору данных. А все потому, что даже в нашем простом исследовании ясно видно, что данные для электростанций сильно разнятся (см. рис. 10.2 и 10.4).

```
## Python
>>> x = (x - np.mean(x, axis = 0)) / (np.std(x, axis = 0))
```

Формат данных NC. Представить данные в формате NC также не составляет большого труда.

```
## Python
>>> if model_name == 'fc_model': ## формат данных NC
>>> ## Возвращаемся на один и два шага назад в исходном вводе
>>>     X = np.hstack([x[1:-h], x[0:-(h + 1)]])
>>>     Y = x[(h + 1):]
```

Чтобы сгенерировать X входных данных, представленных для времени $t - h$, и получить прогноз для временного момента t, нужно из x удалить последние h строк (поскольку для входных данных потребуются значения подписей, которые относятся к более поздним данным, чем те, которыми мы располагаем). Затем данные смещаются обратно вдоль оси времени, чтобы получить дополнительные запаздывающие значения и убедиться в том, что массивы NumPy, представляющие разные временны*е* смещения, имеют одинаковую форму и их можно объединять между собой. Это приводит нас к предыдущей формулировке. Постарайтесь выполнить это упражнение на своем компьютере и доказать

справедливость такого подхода. Также рассмотрите возможность обобщения выражения для произвольной длительности.

Давайте посмотрим, что у нас получилось, установив контрольную точку Pdb и удостоверившись, что значения в массивах X и Y соответствуют ожидаемым аналогам в массиве x.

```
## Python
(Pdb) X[0, 1:10] == x[1, 1:10]
array([ True, True, True, True, True, True, True, True, True])
(Pdb) X[0, 322:331] == x[0, 1:10]
array([ True, True, True, True, True, True, True, True, True])
(Pdb) Y[0, 1:10] == x[4, 1:10]
array([ True, True, True, True, True, True, True, True, True])
```

Первая половина столбцов в массиве X представляет последние временны*е* точки, по которым строится прогноз, а само целевое значение/подпись прогноза расположено на три временны*х* шага вперед. Вот почему массив X[0, 1:10] должен соответствовать массиву x[1, 1:10], а Y[0, 1:10] — массиву x[4, 1:10], поскольку он должен быть на три шага впереди (для входных данных установлен горизонт, равный 3).

Может показаться, что время и выборки (индексы точек данных) часто имеют одинаковые подписи, но это не так. Стоит различать момент времени в будущем, для которого строится прогноз, текущий момент времени, для которого делается снимок входных данных для составления прогноза, и момент времени в прошлом, позволяющий собрать данные для составления прогноза. Конечно, эти значения взаимосвязаны, но нужно научиться их разделять.

Формат данных NTC. Работа с форматом NTC не вызывает особых затруднений.

```
## Python
>>> # Предварительное размещение X и Y в памяти
>>> # Формат X = кол-во экземпляров * временное окно * кол-во каналов (NTC)
>>> X = np.zeros((x.shape[0] - win - h, win, x.shape[1]))
>>> Y = np.zeros((x.shape[0] - win - h, x.shape[1]))
>>>
>>> for i in range(win, x.shape[0] - h):
...     ## Целевое значение/подпись рассчитывается через h шагов вперед
...     Y[i - win] = x[i + h - 1 , :]
...     ## Входными являются данные из предыдущего окна
...     X[i - win] = x[(i - win) : i , :]
```

В любой выборке (размерностью *N* первого измерения) рассматриваются последние win строк входных данных во всех столбцах. Так образуются три измерения. Первое измерение, по существу, задает индексы точек, а данные в таких точках представляются двумерными значениями — *время* × *каналы* (в данном случае — электростанции).

Как и прежде, установим контрольную точку Pdb для проверки нашего кода. Обратите внимание на то, как выполняется тестирование кода. Часто код для проверки формата данных более понятен, чем реальный код, потому что в нем тестирование проводится на конкретных числовых значениях.

```
## Python
(Pdb) Y[0, 1:10] == x[98, 1:10]
array([ True, True, True, True, True, True, True, True, True])
(Pdb) X.shape
(26204, 96, 321)
(Pdb) X[0, :, 1] == x[0:96, 1]
array([ True, True, True, True, True, True, True, True, True,
        True, True, True, True, True, True, True, True, True,
        True, True, True, True, True, True, True, True, True,
        True, True, True, True, True, True, True, True, True,
        True, True, True, True, True, True, True, True, True,
        True, True, True, True, True, True, True, True, True,
        True, True, True, True, True, True, True, True, True,
        True, True, True, True, True, True, True, True, True,
        True, True, True, True, True, True, True, True, True,
        True, True, True, True, True, True, True, True, True,
        True, True, True, True, True, True])
```

Заметьте, что первая подготовленная в массивах X и Y точка (т.е. первая строка), соответствует строкам 0:96 (поскольку в парсере настроено окно обратного просмотра, соответствующее 96 временны*м* шагам), а прямой временной интервал горизонта равен 3, что соответствует строке 98 (массив x заканчивается индексом 95). Не забывайте, что в срезе последнее индексное значение исключается, поэтому массив x представляет строки с 0 по 95 включительно или с 0 по 96 исключительно.

Код обработки данных подвержен ошибкам, очень запутанный и медленный. Тем не менее легко обнаружить, что при каждом следующем переписывании и улучшении он будет становиться все более эффективным. Всегда полезно тщательно протестировать код обработки данных, а затем сохранить его где-нибудь в надежном месте, чтобы избежать повторного решения описанных проблем при решении каждой следующей задачи формирования данных. Также имеет смысл добавить этот код в систему управления версиями, чтобы получить возможность отслеживать его версию при обучении конкретных моделей.

Настройка параметров обучения и создание системы учета

Особенности различных моделей мы обсудим в последующих разделах и пока что пропустим ту часть кода, которая отвечает за построение графиков, а перейдем непосредственно к рассмотрению кода обучения и ведения учета.

Ниже приведен код реализации процедуры обучения на доступных для изучения примерах.

```
## Python
>>> def train(symbol, iter_train, valid_iter, iter_test,
>>>          data_names, label_names,
>>>          save_dir):
>>>     ## Сохранение результатов обучения
>>>     if not os.path.exists(args.save_dir):
>>>        os.makedirs(args.save_dir)
>>>     printFile = open(os.path.join(args.save_dir, 'log.txt'), 'w')
>>>     def print_to_file(msg):
>>>        print(msg)
>>>        print(msg, file = printFile, flush = True)
>>>        ## Заголовок результатов архивирования
>>>        print_to_file('Epoch Training Cor Validation Cor')
>>>
>>>        ## Сохранение значений предыдущих этапов для задания
>>>        ## порога улучшения позволяет досрочно завершить процесс,
>>>        ## если обучение идет медленно
>>>        buf = RingBuffer(THRESHOLD_EPOCHS)
>>>        old_val = None
>>>
>>>        ## Шаблон mxnet
>>>        ## Значение по умолчанию - 1 GPU, индекс которого равен 0
>>>        devs = [mx.gpu(0)]
>>>        module = mx.mod.Module(symbol,
>>>                               data_names=data_names,
>>>                               label_names=label_names,
>>>                               context=devs)
>>>        module.bind(data_shapes=iter_train.provide_data,
>>>                    label_shapes=iter_train.provide_label)
>>>        module.init_params(mx.initializer.Uniform(0.1))
>>>        module.init_optimizer(optimizer='adam',
>>>                              optimizer_params={'learning_rate':
>>>                                                args.lr})
>>>
>>>        ## Обучение
>>>        for epoch in range( args.n_epochs):
>>>            iter_train.reset()
>>>            iter_val.reset()
>>>        for batch in iter_train:
>>>            # Вычисление прогнозов
>>>            module.forward(batch, is_train=True)
>>>            # Вычисление градиентов
>>>            module.backward()
>>>            # Обновление параметров
>>>            module.update()
>>>
>>>        ## Результаты обучения
>>>        train_pred = module.predict(iter_train).asnumpy()
>>>        train_label = iter_train.label[0][1].asnumpy()
>>>        train_perf = perf.write_eval(train_pred, train_label,
```

```
>>>                                 save_dir, 'train', epoch)
>>>
>>>         ## Результаты проверки
>>>         val_pred = module.predict(iter_val).asnumpy()
>>>         val_label = iter_val.label[0][1].asnumpy()
>>>         val_perf = perf.write_eval(val_pred, val_label,
>>>                                    save_dir, 'valid', epoch)
>>>
>>>         print_to_file('%d %f %f ' %
>>>                       (epoch, train_perf['COR'], val_perf['COR']))
>>>
>>>         # Если показатели улучшения не определены, то пропустить
>>>         if epoch > 0:
>>>            buf.append(val_perf['COR'] - old_val)
>>>         # Если показатели улучшения определены, то проверить их
>>>         if epoch > 2:
>>>            vals = buf.get()
>>>            vals = [v for v in vals if v != 0]
>>>            if sum([v < COR_THRESHOLD for v in vals]) == len(vals):
>>>              print_to_file('EARLY EXIT')
>>>              break
>>>       old_val = val_perf['COR']
>>>
>>>       ## Тестирование
>>>       test_pred = module.predict(iter_test).asnumpy()
>>>       test_label = iter_test.label[0][1].asnumpy()
>>>       test_perf = perf.write_eval(test_pred, test_label,
>>>                                   save_dir, 'tst', epoch)
>>>       print_to_file('TESTING PERFORMANCE')
>>>       print_to_file(test_perf)
```

Зачем нужен кольцевой буфер

Кольцевой буфер — это место для размещения значений, устроенное таким образом, что при их извлечении первый вход будет выступать первым выходом. Он имеет свойство расти только до заданного значения. Хотя производительность и объем памяти здесь не критичны, используя этот класс, можно избежать многих повторений.

В языке Python нет встроенного кольцевого буфера, но вы можете найти простые примеры его реализации в Интернете. Его удобно применять для отслеживания небольшого количества значений в порядке их получения.

Предыдущий код выполняет разнообразный набор рутинных задач. Во-первых, в нем устанавливаются значения, по которым отслеживается история показателей точности проверки, позволяющих убедиться в положительном прогрессе обучения. Если модель обучается недостаточно быстро, то не стоит продолжать нагружать графические процессоры, попусту тратя время и электроэнергию.

В шаблоне MXNet задействован программный интерфейс Module (в отличие от Gluon, с которым мы имели дело ранее).

```
## Python
>>>     ## Шаблон mxnet
>>>     ## Значение по умолчанию - 1 GPU, индекс которого равен 0
>>>     devs = [mx.gpu(0)]
>>>     module = mx.mod.Module(symbol,
>>>                            data_names=data_names,
>>>                            label_names=label_names,
>>>                            context=devs)
>>>     module.bind(data_shapes=iter_train.provide_data,
>>>                 label_shapes=iter_train.provide_label)
>>>     module.init_params(mx.initializer.Uniform(0.1))
>>>     module.init_optimizer(optimizer='adam',
>>>                           optimizer_params={'learning_rate':
>>>                                             args.lr})
```

В приведенных выше строках кода выполняются следующее операции.

1. Настройка компонента нейронной сети в виде вычислительного графа.
2. Настройка формы данных так, чтобы позволить сети получить их и оптимизировать.
3. Инициализация всех весов графа случайными значениями (это креативный процесс, а не просто случайный выбор чисел из бесконечного количества возможностей).
4. Инициализация оптимизатора, который может иметь разные представления и для которого в явном виде устанавливается начальная скорость обучения в зависимости от входных параметров.

Теперь воспользуемся итератором обучающих данных для прохода по данным во время обучения.

```
## Python
>>> for epoch in range(args.n_epochs):
>>>     iter_train.reset()
>>>     iter_val.reset()
>>>     for batch in iter_train:
>>>         module.forward(batch, is_train=True)  # вычисление прогнозов
>>>         module.backward()                     # вычисление градиентов
>>>         module.update()                       # обновление параметров
```

Оценим предсказанные результаты как для обучающего, так и для проверочного набора (в том же внешнем цикле for, что и раньше).

```
## Python
>>>     ## Результаты обучения
>>>     train_pred = module.predict(iter_train).asnumpy()
```

```
>>>      train_label = iter_train.label[0][1].asnumpy()
>>>      train_perf = evaluate_and_write(train_pred, train_label,
>>>                                      save_dir, 'train', epoch)
>>>
>>>      ## Результаты проверки
>>>      val_pred = module.predict(iter_val).asnumpy()
>>>      val_label = iter_val.label[0][1].asnumpy()
>>>      val_perf = evaluate_and_write(val_pred, val_label,
>>>                                    save_dir, 'valid', epoch)
```

Следующий цикл проверяет условия досрочного прекращения обучения.

```
## Python
>>> if epoch > 0:
>>>    buf.append(val_perf['COR'] - old_val)
>>> if epoch > 2:
>>>    vals = buf.get()
>>>    vals = [v for v in vals if v != 0]
>>>    if sum([v < COR_THRESHOLD for v in vals]) == len(vals):
>>>       print_to_file('EARLY EXIT')
>>>       break
>>> old_val = val_perf['COR']
```

Этот громоздкий код позволяет вести учет выполненных действий и проверять условия, записывая каждое последующее значение корреляции между прогнозируемыми и фактическими значениями. Если поэтапная корреляция не демонстрирует улучшений (или даже показывает ухудшение) для заданного количества повторений, то обучение прекращается.

Оценка производительности

Функция `evaluate_and_write()` фиксирует корреляцию, необработанные и оценочные значения на каждом вычислительном этапе. Она понадобится нам для тестирования в конце всего обучения.

```
## Python
>>> def evaluate_and_write(pred, label, save_dir, mode, epoch):
>>>     if not os.path.exists(save_dir):
>>>        os.makedirs(save_dir)
>>>
>>>     pred_df = pd.DataFrame(pred)
>>>     label_df = pd.DataFrame(label)
>>>     pred_df.to_csv( os.path.join(save_dir, '%s_pred%d.csv'
>>>                                  % (mode, epoch)))
>>>     label_df.to_csv(os.path.join(save_dir, '%s_label%d.csv'
>>>                                  % (mode, epoch)))
>>>
>>>     return { 'COR': COR(label,pred) }
```

В ней, в свою очередь, применяется корреляционная функция, определенная следующим образом.

```
## Python
>>> def COR(label, pred):
>>>     label_demeaned = label - label.mean(0)
>>>     label_sumsquares = np.sum(np.square(label_demeaned), 0)
>>>
>>>     pred_demeaned = pred - pred.mean(0)
>>>     pred_sumsquares = np.sum(np.square(pred_demeaned), 0)
>>>
>>>     cor_coef = np.diagonal(np.dot(label_demeaned.T, pred_demeaned)) /
>>>     np.sqrt(label_sumsquares * pred_sumsquares)
>>>
>>>     return np.nanmean(cor_coef)
```

Иногда в этом наборе данных встречаются случаи нулевой дисперсии, которые могут привести к получению значений NAN в столбце, что обусловливает выбор функции `np.nanmean()` вместо `np.mean()`.

Обратите внимание, что одна из основных возможностей, которые здесь исключаются, — сохранение весов модели в контрольных точках на протяжении всего процесса обучения. При подготовке к производству нам нужно было иметь возможность перезагрузить модель и развернуть ее, для чего используются функции `Module.save_checkpoint` (сохранение весов) и `Module.load` (загрузка модели обратно в память для продолжения обучения или разворачивания производства). Существует много вариантов обучения принципам настройки правильного конвейера глубокого обучения, здесь мы рассматриваем только основные моменты.

Сборка

Объединим компоненты конвейера в теле метода `__main__()`.

```
## Python
>>> if __name__ == '__main__':
>>>     # Аргументы командной строки парсера
>>>     args = parser.parse_args()
>>>
>>>     # Создание итераторов данных
>>>     iter_train, iter_val, iter_test = prepare_iters(
>>>         args.data_dir, args.win, args.h,
>>>         args.model, args.batch_n)
>>>
>>>     ## Подготовка символов
>>>     input_feature_shape = iter_train.provide_data[0][1]
>>>
>>>     X = mx.sym.Variable(iter_train.provide_data[0].name )
```

```
>>>     Y = mx.sym.Variable(iter_train.provide_label[0].name)
>>>
>>>     # Настройка модели
>>>     model_dict = {
>>>          'fc_model' : fc_model,
>>>          'rnn_model' : rnn_model,
>>>          'cnn_model' : cnn_model,
>>>          'simple_lstnet_model' : simple_lstnet_model
>>>     }
>>>     model = model_dict[args.model]
>>>
>>>     symbol, data_names, label_names = model(iter_train,
>>>                                             input_feature_shape,
>>>                                             X, Y,
>>>                                             args.win, args.sz_filt,
>>>                                             args.n_filt, args.drop)
>>>
>>>     ## Обучение
>>>     train(symbol, iter_train, iter_val, iter_test,
>>>           data_names, label_names, args.save_dir)
```

Здесь использована инфраструктура, созданная выше. Сначала анализируются аргументы командной строки. После этого создаются итераторы с настраиваемыми входными данными, включая горизонт прогнозирования, окно обратного просмотра, размер пакета и имя создаваемой модели. Далее создаются символы MXNet, а также записывается форма входных данных, востребованная при создании модели. Наконец, информация о модели вместе с итераторами и каталогом сохранения передается в функцию обучения, которая выполняет наиболее интересную часть: обучает модель и выводит показатели производительности.

Таким образом, мы можем наблюдать минимальный, но полностью функциональный конвейер обучения, включающий прием и изменение данных, построение и обучение модели, а также запись важных показателей для правильной ее оценки.

Кстати, метод `print_to_file()` — это всего лишь удобная оболочка для функции `print()`.

```
## Python
def print_to_file(msg):
    print(msg, file = printFile, flush = True)esfa
print_to_file(args)
```

Вам может понадобиться сохранить процесс обучения модели. Вполне возможно, что подобранные веса модели окажутся не теми, на которых завершился сеанс обучения, а получены несколько ранее. Наличие записи о ходе обучения поможет точнее настроить гиперпараметры, относящиеся как к структуре

модели, так и к процессу обучения, начиная с их количества (во избежание недо- или переобучения) и заканчивая скоростью обучения (для предотвращения слишком сильных или слабых корректировок).

На данный момент мы располагаем полным, хотя и минимальным, жизнеспособным конвейером, но нам все еще не хватает моделей для обучения. Далее мы перейдем к рассмотрению основных типов моделей, применимых в анализе временных рядов, и обучим каждую из них, чтобы оценить относительную производительность.

Сети прямого распространения

В этой главе предпринят довольно необычный прием, который заключается в использовании сетей прямого распространения для анализа временных рядов. Большинство современных задач обработки данных временных рядов основано на рекуррентных или, несколько реже, сверточных нейронных сетях. Однако мы начнем наш экскурс с нейронной сети прямого распространения, так как она имеет самую простую структуру. Существует несколько причин, по которым дальнейший материал целесообразно начинать рассматривать именно с нейронных сетей прямого распространения.

- Сети прямого распространения допускают очень простое распараллеливание, а значит, обладают высокой производительностью. Получив в свое распоряжение достаточно хорошую модель прямого распространения, вы сможете вычислить ее очень быстро.
- Сети прямого распространения прекрасно подходят для оценки сложности динамических процессов в анализируемой временной последовательности. Не все временные ряды действительно являются временными в том понимании, что не во всех них динамические изменения демонстрируют временную направленность — более ранние значения имеют определенную связь с более поздними значениями. Нейронную сеть прямого распространения желательно использовать в качестве одной из базовых моделей наряду с простой линейной моделью.
- Компоненты сетей прямого распространения часто интегрируются в более крупные и сложные архитектуры глубокого обучения временных рядов. Поэтому крайне важно понять, как они работают, даже если они не образуют целостную модель для вашего проекта.

Простой пример

Сеть прямого распространения — это простейший тип нейронной сети, и мы начнем ее изучение с примера использования архитектуры прямого распространения к некому временному ряду. В структуре стандартной нейронной сети прямого распространения нет ничего, указывающего на временные отношения, но идея заключается в том, чтобы позволить алгоритму понять, как по прошлым входным данным предсказать будущие данные. Пример сети прямого распространения приведен на рис. 10.1. Сеть прямого распространения представляет собой ряд полносвязных слоев, т.е. входные данные каждого слоя подключаются к каждому узлу графа.

Начнем изучение с простого примера с уже известным способом форматирования данных. Иначе говоря, наши входные данные `X` будут представлены двумерными значениями $N \times C$ (т.е. *выборки* × *каналы*), где временные компоненты упорядочены в каналах. Это соответствует следующему формату данных.

A, t – 2	*A, t – 3*	*B, t – 2*	*B, t – 3*	*C, t – 2*	*C, t – 3*
3	0	–2	–1	–3	–2

Напомним код, который представляет эту ветвь данных.

```
## Python
>>> if model_name == 'fc_model':
>>>     ## Первый и второй обратные шаги в исходных данных
>>>     X = np.hstack([x[1:-1], x[:-2]])
>>>     Y = x[2:]
>>>     return (X, Y)
```

Создадим полносвязную модель с помощью программного интерфейса `Module` пакета `MXNet`.

```
## Python
>>> def fc_model(iter_train, window, filter_size, num_filter, dropout):
>>>     X = mx.sym.Variable(iter_train.provide_data[0].name)
>>>     Y = mx.sym.Variable(iter_train.provide_label[0].name)
>>>
>>>     output = mx.sym.FullyConnected(data=X, num_hidden = 20)
>>>     output = mx.sym.Activation(output, act_type = 'relu')
>>>     output = mx.sym.FullyConnected(data=output, num_hidden = 10)
>>>     output = mx.sym.Activation(output, act_type = 'relu')
>>>     output = mx.sym.FullyConnected(data = output, num_hidden = 321)
>>>
>>>     loss_grad = mx.sym.LinearRegressionOutput(data = output,
>>>                                               label = Y)
>>>     return loss_grad, [v.name for v in iter_train.provide_data],
>>>                       [v.name for v in iter_train.provide_label]
```

Мы строим трехслойную полносвязную сеть. Первый слой имеет 20 скрытых элементов, а второй слой — 10. После первых двух слоев добавлен слой активации. Именно он делает модель нелинейной. Без него она представляет последовательность произведений матриц и сводится к линейной модели. Полносвязный слой имеет вид

$$\mathbf{Y} = \mathbf{X}\mathbf{W}^{\mathrm{T}} + \mathbf{b},$$

где $\mathbf{W}$ — это набор весов, соответствующих таковому в произведении матриц с исправленными размерностями, результатом которого будет вектор, учитывающий скрытые значения. Затем к нему добавляется сдвиг. Веса $\mathbf{W}$ и сдвиг $\mathbf{b}$ уточняются в ходе обучения. Заметьте, что обучение одного такого набора весов или даже серии таких весов не представляет большого интереса — это просто другой способ вычисления линейной регрессии. Однако активация, включаемая в модель после матричных операций, предопределяет разнообразие. Слой активации основан на различных нелинейных функциях, таких как tanh и ReLU, графики которых изображены на рис. 10.5 и 10.6 соответственно.

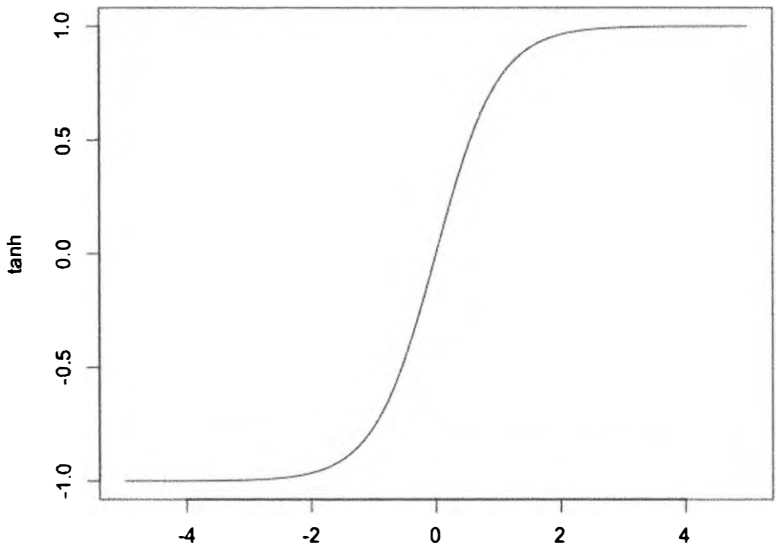


Рис. 10.5. *Функция tanh демонстрирует нелинейное поведение на небольшом интервале, а затем становится постоянной и характеризуется нулевой производной*

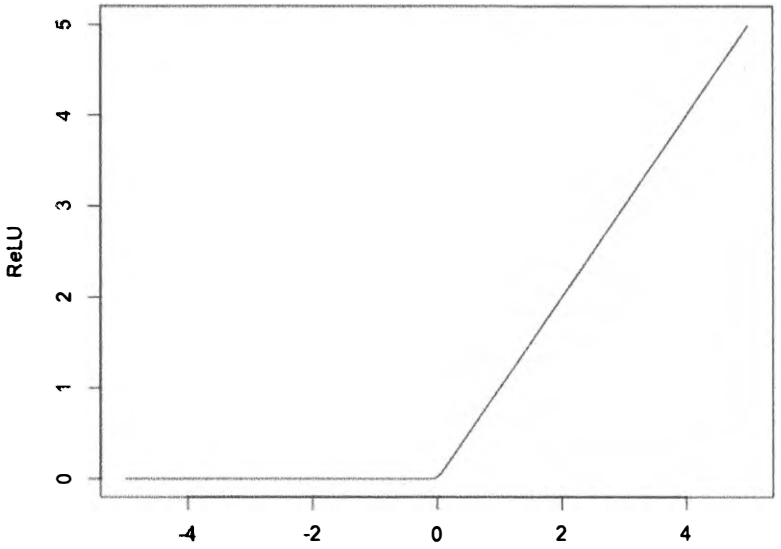


Рис. 10.6. *Функция ReLU проста для вычисления, но также привносит нелинейность*

В предыдущем коде в качестве функции активации использована функция ReLU. В общем случае имеет смысл тестировать различные функции активации (ReLU, tanh, сигмоида) на ранних стадиях построения модели и настройки гиперпараметра.

Теперь давайте обучим нашу модель и посмотрим, как она работает.

```
Epoch       Training Cor    Validation Cor
0           0.578666        0.557897
1           0.628315        0.604025
2           0.645306        0.620324
3           0.654522        0.629658
4           0.663114        0.636299
5           0.670672        0.640713
6           0.677172        0.644602
7           0.682624        0.648478
8           0.688570        0.653288
9           0.694247        0.657925
10          0.699431        0.663191
11          0.703147        0.666099
12          0.706557        0.668977
13          0.708794        0.670228
14          0.711115        0.672429
15          0.712701        0.673287
16          0.714385        0.674821
17          0.715376        0.674976
18          0.716477        0.675744
19          0.717273        0.676195
```

```
20          0.717911        0.676139
21          0.718690        0.676634
22          0.719405        0.677273
23          0.719947        0.677286
24          0.720647        0.677451

TESTING PERFORMANCE
{'COR': 0.66301745}
```

Это хорошая точность? Трудно сказать, не зная контекста задачи. Работая с моделью глубокого обучения, вы должны запустить ее в производство, только если она значительно превосходит более простые модели, рассматриваемые в предыдущих главах. Модель глубокого обучения требует больше времени для расчета и, как правило, выдвигает более высокие требования к вычислительным ресурсам, поэтому ее следует использовать только при полной оправданности производственных затрат. По этой же причине, выполняя задачи анализа временных рядов с помощью инструментов глубокого обучения, сначала попробуйте обратиться к более простым моделям, которые послужат объективным стандартом в достижении цели.

В любом случае нам удалось показать, что даже модель без явно выраженной временной направленности способна обучаться и составлять прогнозы, которые правдоподобно коррелируют с измеренными значениями.

Внимание как фактор улучшения сетей прямого распространения

Хотя сети прямого распространения не показывают хороших результатов в решении задач анализа временных рядов, исследователями ведутся работы по разработке архитектур с повышенной точностью обработки последовательных данных. Одна из рассматриваемых ими концепций основана на понятии *внимания*, которое обозначает механизмы нейронной сети, позволяющие определить ту часть последовательности, которая требует наибольшего сосредоточения и вносит наибольший вклад в получение желаемого результата.

Понятие внимания предполагает, что архитектура нейронной сети должна снабжать модель механизмами определения наиболее важной информации на каждом временном этапе. Это достигается с помощью *весов внимания*, которые настраиваются отдельно для каждого временного шага, позволяя модели комбинировать информацию с разных временных шагов. Такие веса внимания умножаются на то, что в противном случае считается выходным, либо скрытым, состоянием модели, преобразуя это скрытое состояние в *вектор контекста*, название которого указывает на то, что скрытое состояние теперь лучше учитывает весь временной контекст информации, содержащейся во временных рядах и подлежащей описанию с помощью неких временных зависимостей.

Функция Softmax()

Возможно, вы уже знакомы с функцией

$$softmax(y_i) = exp(y_i) \Big/ \sum_j exp(y_i),$$

которая служит двум целям.

- Сделать сумму выходов равной единице, чтобы представить полный набор вероятностей. Часто результат работы функции softmax() интерпретируется именно так.
- Принимая во внимание общую форму экспоненциальной функции, сделать большие входные данные (y_i) еще бо́льшими, а меньшие входные данные еще больше уменьшить. Такой подход позволяет подчеркнуть сильные активации/веса и погасить слабые сигналы.

Функция softmax() также известна как *нормализованная экспоненциальная функция* — такое название предопределяется формой ее математической записи, приведенной выше.

Исходно концепция внимания стала использоваться в рекуррентных нейронных сетях (подробнее об этом — далее), но понять ее назначение и принципы реализации проще всего на примере архитектуры прямого распространения, приведенном в исследовательской работе `https://arxiv.org/pdf/1512.08756.pdf`. В этом документе рассматривается такой способ применения нейронных сетей прямого распространения к последовательным данным, в котором реакция сети на каждый шаг последовательности передается на вход и таким образом влияет на получаемый результат. Такой подход позволяет обрабатывать последовательности переменной длины, ведь временны́е ряды переменной длины являются типичными для практических задач.

На рис. 10.7 представлен пример того, как нейронная сеть прямого распространения может использоваться для решения задачи обработки последовательных данных.

Сеть прямого распространения применяется к входам на каждом временно́м шаге отдельно, генерируя "скрытое состояние" для каждого временно́го шага h_1, ..., h_T. В результате формируется ряд скрытых состояний. Показанная в углу вторая нейронная сеть, разработанная для обучения $a(h_t)$, представляет механизм внимания. Он позволяет узнать, какой вес приписать каждому входу, причем на вход передается состояние, взятое для другого момента времени. Это позволяет механизму определить веса для каждого временно́го шага в окончательной сумме входных данных.

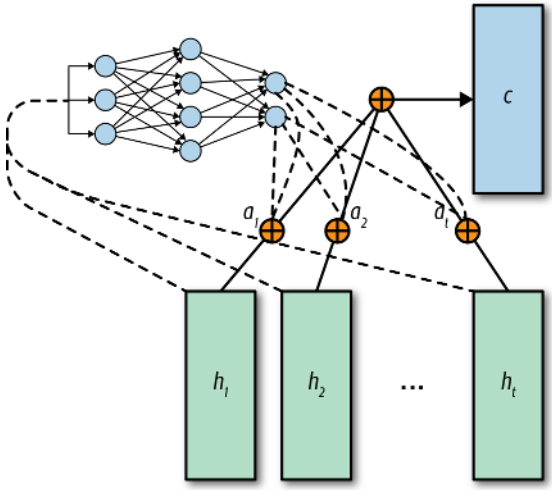


Рис. 10.7. *Механизм внимания в сети прямого распространения*

Впоследствии скрытые состояния разных временны*х* этапов объединяются с соответствующими коэффициентами внимания перед окончательной обработкой, чтобы определить целевое значение/подпись. Разработчики такой сети обнаружили, что она отлично справляется с различными задачами, которые до этого решались исключительно с помощью рекуррентных нейронных сетей, в полной мере удовлетворяющих требованиям к запоминанию более ранних входных данных и объединения их с более поздними данными.

Этот пример наглядно показывает, что модели глубокого обучения не имеют простых архитектур ввиду большого количества возможностей по адаптации даже базовой модели к целевым задачам анализа временны*х* рядов.

Как уже отмечалось, нейронные сети прямого распространения не являются основными инструментами решения задач, связанных с обработкой данных временны*х* рядов. Тем не менее они послужат хорошей отправной точкой на пути к оценке точности простых моделей. Интересно, что если не добавлять функцию активации, то можно использовать их для реализации моделей AR и VAR с помощью фреймворка `MXNet`. Иногда они, правда, применяются в чистом виде. Что более важно, существуют архитектурные решения полносвязных моделей, которые оказываются необычайно точными для определенных наборов временны*х* рядов.

Сверточные нейронные сети

Если вы знакомы с глубоким обучением, то, скорее всего, имеете представление о сверточных нейронных сетях (Convolutional Neural Networks — CNN). Большая часть выдающихся, сравнимых с возможностями человеческого мозга,

достижений, которые приписываются компьютерам, за последние годы, получены благодаря чрезвычайно запутанной и сложной сверточной архитектуре. Несмотря на все это, принципы построения свертки довольно просты и были разработаны задолго до появления глубокого обучения. Свертки давно применяются в более простых, управляемых человеком, системах обработки и распознавания изображений, например в таких простых фильтрах, как гауссово размытие. Все, кто не знакомы с устройством ядер обработки изображений и не представляют, как они пишутся, могут найти хорошее описание принципов их построения на сайте Stack Overflow (`https://perma.cc/8U8Y-RBYW`) — как с помощью высокоуровневого программного интерфейса, так и вручную с использованием инструментов библиотеки `NumPy`.

Свертка означает такое применение ядра (матрицы) к более крупной матрице, в котором она как бы "скользит" по большей матрице, образуя новую. Каждый элемент новой матрицы представляет собой сумму поэлементного произведения ядра и фрагмента большей матрицы. Такое ядро применяется неоднократно в процессе "скольжения" по матрице/изображению. Это делается для заранее оговоренного количества ядер для проявления различных признаков. Принципиальная схема сверточной сети на многослойных структурах приведена на рис. 10.8.

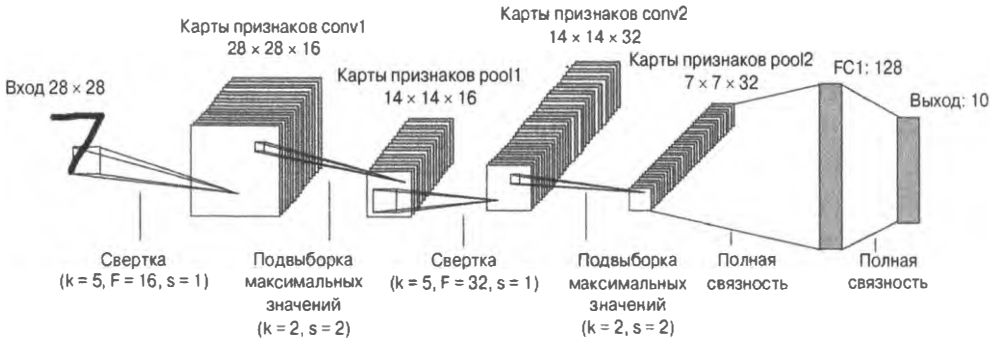


Рис. 10.8. *Сверточная сеть. Многие двумерные окна с заданным размером ядра скользят по исходному изображению, создавая множество карт признаков, основанных на получаемых при обучении весах. Часто они объединяются и обрабатываются с помощью функции активации. Процесс повторяется несколько раз на нескольких уровнях, чтобы свести множество признаков к меньшему диапазону значений, что в конечном итоге можно использовать в разного рода оценках, например в задачах классификации*

По многим причинам традиционная сверточная сеть плохо подходит для работы с временными рядами. Главная особенность сверток состоит в одинаковости обработки всех пространств, что имеет смысл при работе с изображениями,

но не с временны*ми* рядами, в которых некоторые отдельные временны*е* точки обязательно находятся ближе, чем другие. Сверточные сети обычно имеют масштабно-инвариантную структуру, что позволяет достаточно точно сказать, например, что на рисунке изображена лошадь, независимо от того, насколько она большая. Однако во временны*х* рядах по большей части масштаб, как и масштабируемые признаки, остается неизменным. Годовое сезонное колебание не должно интерпретироваться так же или описываться тем же набором признаков, что и дневное колебание, хотя в некоторых случаях это очень даже целесообразно.

Именно такая двойственная природа сверток — сильные стороны одновременно могут быть как преимуществом, так и недостатком — приводит к тому, что они все чаще выступают отдельными компонентами сети (а не отдельной сетью), предназначенной для анализа временны*х* рядов. Кроме того, это их свойство стало причиной более частого использования сверток для решения задач классификации, чем прогнозирования, хотя достойное применение находят оба варианта.

Перечислим наиболее востребованные способы использования сверточных сетей для анализа временны*х* рядов.

- Определение "характерных признаков" в истории посещений интернет-ресурсов, что помогает обнаружить аномальную активность пользователей в Интернете.
- Выявление необычных сердечных ритмов по данным ЭКГ.
- Предсказание транспортных потоков по записям с камер, установленных в нескольких местах большого города.

Применимо к одномерным временны*м* рядам свертки не могут представить ничего интересного. Их значимость проявляется при проведении анализа многоканальных временны*х* рядов, поскольку в них приходится работать в двумерных (или даже в трехмерных) пространствах, в которых время — только одна из осей.

Есть и другие архитектурные новшества, о которых стоит упомянуть в силу рассмотрения в следующем разделе двух примеров сверточных механизмлв для временны*х* рядов.

Простая сверточная модель

Сверточную модель можно применить к имеющимся данным вместо описанной выше полносвязной модели. В этом и остальных примерах мы будем работать с данными в формате NTC, которые выглядят следующим образом.

Время	*A*	*B*	*C*
$t-1$	0, 3	–1, –2	–2, –3
t	3, 4	–2, –2	–3, –4

Как видите, данные представляются в виде $N \times T \times C$. Однако сверточный слой предполагает получение данных в формате `batch_size, channel, height × width`.

```
## Python
>>> def cnn_model(iter_train, input_feature_shape, X, Y,
>>>               win, sz_filt, n_filter, drop):
>>>     conv_input = mx.sym.reshape(data=X, shape=(0, 1, win, -1))
>>>     ## Сверточная сеть работает с четырехмерными входными данными
>>>     ## (N x количествово каналов x высота x ширина)
>>>     ## в нашем случае канал один (как на черно-белом изображении,
>>>     ## где высота = время, ширина = расположение электростанций
>>>
>>>     cnn_output = mx.sym.Convolution(data=conv_input,
>>>                                     kernel=(sz_filt,
>>>                                             input_feature_shape[2]),
>>>                                     num_filter=n_filter)
>>>     cnn_output = mx.sym.Activation(data=cnn_output, act_type='relu')
>>>     cnn_output = mx.sym.reshape(mx.sym.transpose(data=cnn_output,
>>>                                                  axes=(0, 2, 1, 3)),
>>>                                 shape=(0, 0, 0))
>>>     cnn_output = mx.sym.Dropout(cnn_output, p=drop)
>>>
>>>     output = mx.sym.FullyConnected(data=cnn_output,
>>>                                    num_hidden=input_feature_shape[2])
>>>     loss_grad = mx.sym.LinearRegressionOutput(data=output, label=Y)
>>>     return (loss_grad,
>>>             [v.name for v in iter_train.provide_data],
>>>             [v.name for v in iter_train.provide_label])
```

Снова-таки, представленная модель все еще не имеет явно заданной временной направленности. При этом время все же откладывается вдоль одной из осей, что делает модель более упорядоченной.

Влияет ли временная направленность данных на точность модели? Возможно, нет.

```
0  0,330701     0,292515
1  0,389125     0,349906
2  0,443271     0,388266
3  0,491140     0,442201
4  0,478684     0,410715
5  0,612608     0,564204
6  0,581578     0,543928
7  0,633367     0,596467
8  0,662014     0,586691
9  0,699139     0,600454
10 0.692562     0.623640
11 0,717497     0,650300
12 0,710350     0,644042
13 0,715771     0,651708
```

```
14 0,717952     0,651409
15 0,712251     0,655117
16 0,708909     0,645550
17 0.696493     0.650402
18 0,695321     0,634691
19 0,672669     0,620604
20 0,662301     0,597580
21 0,680593     0,631812
22 0,670143     0,623459
23 0,684297     0,633189
24 0,660073     0,604098

TESTING PERFORMANCE
{'COR': 0.5561901}
```

Зачастую очень трудно разобраться, почему одна модель лучше, другая — хуже. На самом деле даже сами разработчики моделей нередко ошибаются, определяя причины хороших результатов модели. Существует очень мало аналитических средств, позволяющих выяснить это предельно точно, а с учетом порой запутанной структуры входного набора данных — почти невозможно. Но разве отсутствие явных преимуществ модели CNN не указывает на то, что *б*ольшая часть важной информации сосредоточена в ближайшие моменты времени? Или это отражается в количестве параметров? И как сказывается на точности модели количество используемых гиперпараметров? В реальных задачах критически важно понимать, какой уровень точности можно считать достаточным для заданных структуры модели, структуры данных и общего количества рассматриваемых параметров.

Мы наблюдаем ошибку в коде, обусловленную избранной логикой ранней остановки. Похоже, она была слишком мягкой. В данном случае задача была пересмотрена, и мне удалось заметить, что в течение ряда этапов могут наблюдаться следующие изменения в корреляции.

```
[-0,023024142, 0,03423196, -0,008353353, 0,009730637, -0,029091835]
```

Такой результат означает, что изменение в корреляции может быть очень сильным — даже отрицательным — и многократно наблюдаемым, и тем не менее точность модели понемногу улучшается. Такая мягкость оказывается не самым хорошим решением, поэтому неплохо вернуться к конвейеру и задать более жесткие условия ранней остановки. Это только один из примеров компромисса, которые вы будете постоянно искать при использовании моделей глубокого обучения. Чтобы его достичь, вам придется выполнять много пробных прогонов для построения работоспособного конвейера. Таким образом можно получить представление о параметрах, приемлемых для целевого набора данных, и о представляющих интерес моделях, учитывая, что они существенно различаются от одного набора данных к другому.

Альтернативные сверточные модели

Простая сверточная модель, с которой вы познакомились выше, работала на удивление хорошо, хотя и не содержала никаких специальных модификаций, учитывающих временную направленность данных. Далее мы обсудим два подхода к использованию сверточных архитектур для анализа временных рядов: принятые в исследовательской и производственной средах.

Почему сверточные сети столь привлекательны? На то есть множество причин. Во-первых, сверточные архитектуры — это проверенные, эффективные методы, которые хорошо изучены специалистам отрасли. Кроме того, у сверточных моделей мало параметров, так как одни и те же фильтры повторяются снова и снова, а это означает, что для обучения не требуется слишком много весов. И наконец, большие фрагменты сверточных моделей можно вычислить параллельно, т.е. они могут быть очень быстродействующими инструментами построения логических выводов.

Причинные свертки

Причинные свертки будем рассматривать на наглядных иллюстрациях, поскольку они проще всего могут продемонстрировать принципы модификации сверток для учета факторов причиной связи и течения времени. На рис. 10.9 показан пример *расширенной причинной свертки* (dilated causal convolution). Ее причинная компонента работает так, что в любой отдельно рассматриваемый сверточный фильтр попадают входы только за прошлые моменты времени. Вот почему изображение не симметрично: более ранние точки переходят в свертки, используемые в более поздние моменты времени, но не наоборот.

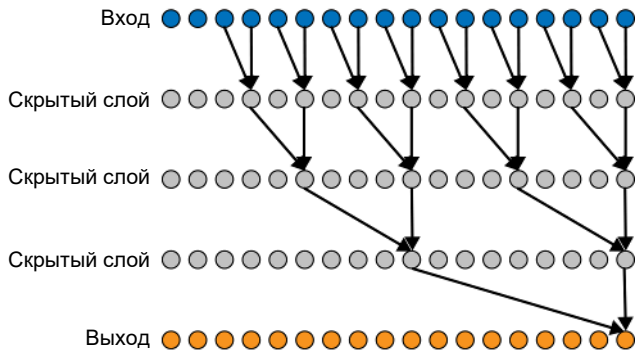


Рис. 10.9. *Граф, иллюстрирующий важную часть архитектуры WaveNet (*`https://perma.cc/Z4KZ-ZXBQ`*). Здесь показан принцип модификации архитектуры сверточной нейронной сети для задач учета поведенческих особенностей временных рядов*

Компонента "расширения" отвечает за пропуск точек в структуре сверточных фильтров, так что на каждом слое любая заданная точка попадает только в один сверточный фильтр каждого слоя. Это способствует разреженности модели и уменьшает избыточные или перекрывающиеся свертки, позволяя модели учитывать предыдущую по времени информацию и в то же время поддерживать объем вычислений на приемлемом уровне.

Приведенный пример расширенной причинной свертки основан на понятии временной причинности, разрешая только те данные, которые относятся к предыдущим моментам времени. Иначе говоря, свертки на этом изображении имеют неравные возможности: они позволяют передавать данные только из прошлого в будущее, а не наоборот. Каждая точка данных в исходном входе оказывает влияние на конечную точку. Обратите внимание, что именно здесь означает расширение: все более и более глубокие слои в свертке игнорируют все большее число точек, заданных в предыдущих слоях. В соответствии с тем, как организовано расширение, в окончательный вход включается каждая точка исходных данных, но только один раз. В принципе, для расширения это не является обязательным условием, но в данном случае задействован именно такой подход. Расширение можно организовать так, чтобы пропустить вообще все временные точки.

Хотя причинные свертки кажутся сложными для понимания и расчета, реализовать их удивительно просто. Просто заполните левую часть матрицы пустыми значениями, т.е. добавьте в нее более ранние временные отступы, приравненные к нулю, чтобы они не влияли на значение, а затем предоставьте им статус "действительных", чтобы свертка работала только в реальных границах матрицы, не затрагивая мнимые пустые ячейки. Учитывая, что свертки рассчитываются через суммирование поэлементных произведений, добавление нулей в левую часть рис. 10.9 позволит выполнить стандартную свертку так, что дополнительные нулевые ячейки не будут влиять на конечный результат.

Причинные свертки имели большой успех в производственном секторе, в частности как часть технологий компании Google, применяемой для преобразования текста в речь, а также для распознавания речи.

Преобразование временного ряда в изображения

Известно, что сверточные модели очень хорошо показали себя в анализе изображений, и для того, чтобы попытаться в полной мере применить их к временным рядам, нужно найти способ приведения данных временного ряда к изображению. Существует много способов выполнения этой задачи, один из которых интересен тем, что позволяет преобразовать в рекуррентную диаграмму даже одномерный временной ряд (рис. 10.10).

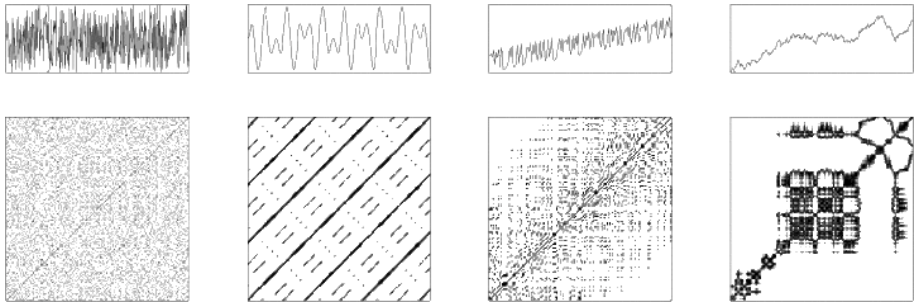


Рис. 10.10. *Наглядная визуализация четырех видов временных рядов (слева направо): (а) белый шум, (б) гармонический/сезонный ряд с двумя частотами, (в) хаотические данные с трендом, (г) авторегрессионный процесс.*

Источник: Википедия, статья Норберта Марвана (2006)
`https://perma.cc/4BV2-57T4`

Рекуррентная диаграмма — это способ описания состояний в фазовом пространстве, когда временной ряд возвращается примерно к той же фазе и состоянию, в котором он находился в более ранний момент времени. Эта диаграмма определяется через двоичную рекуррентную функцию. Рекурсия задается как $R(i, j) = 1$, если $f(i) - f(j)$ достаточно мало, и 0 — в противном случае. Такое преобразование приводит к получению двоичного черно-белого изображения, подобного показанному на рис. 10.10. Обратите внимание, что индексы i и j относятся к значениям времени, а ось времени никак не ограничена.

Несмотря на то что реализация рекуррентных диаграмм — это относительно простое занятие, в таких пакетах, как `pyts` (`https://perma.cc/4K5X-VYQR`) предусмотрена специальная функция их построения, исходный код которой широко доступен для загрузки (`https://perma.cc/VS2Z-EJ8J`) и прост в использовании.

Чтобы понять, как можно использовать описанный принцип для представления в виде изображения многомерного временного ряда, рассматривайте переменные такого ряда как отдельные цветовые каналы изображения. Это всего лишь один пример того, насколько универсальными и гибкими могут быть архитектура и методы глубокого обучения.

Рекуррентные нейронные сети

Рекуррентные нейронные сети (Recurrent Neural Networks — RNN) представляют собой широкий класс сетей, в которых одни и те же параметры применяются повторно даже при изменении входных данных с течением времени. Они очень похожи на нейронные сети прямого распространения, описанные выше, но, как бы там ни было, на RNN основаны — полностью или частично — многие

из наиболее успешных моделей, применяемых для решения научных и производственных задач, связанных с анализом последовательностей данных, обработкой языковых структур, а также с прогнозированием и классификацией временных рядов. Укажем на важные отличия RNN и сетей прямого распространения.

- RNN рассматривает временные шаги по порядку.
- RNN включает состояние, которое сохраняется неизменным от одного временного шага к другому, и именно это состояние, а также его статические параметры, определяют ответные обновления для каждой новой порции информации, получаемой на каждом шаге.
- RNN включает параметры, которые позволяют пошагово обновлять свое состояние, включая скрытое состояние.

Часто RNN описываются через парадигму развертывания, согласно которой рекуррентная архитектура основана на узлах. В такой парадигме предполагается, что одни и те же параметры используются снова и снова, а число параметров довольно мало даже для очень длинных временных последовательностей (рис. 10.11).

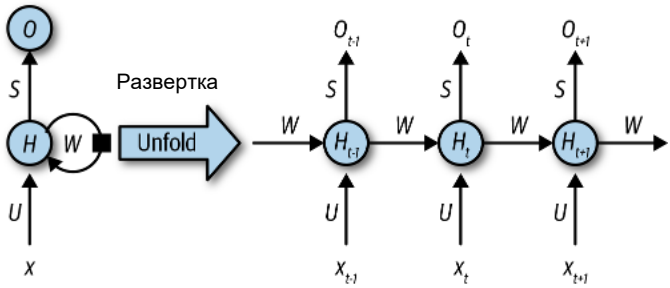


Рис. 10.11. *Развертывание рекуррентной нейронной сети на каждом временном шаге при обработке данных*

Однако самый простой способ понять, как работает RNN, — это изучить простой пример. Моя самая любимая реализация RNN — управляемый рекуррентный блок (Gated Recurrent Unit — GRU). Его математическое описание выглядит сложнее, чем код реализации, поэтому ниже рассматривается вариант написания GRU на языке Python с помощью библиотеки `NumPy`. Как видите, в коде используются две функции активации: сигмоида и гиперболический тангенс. Кроме того, все, что делает код, — перемножает и суммирует матрицы, а также выполняет поэлементное умножение матриц (произведение Адамара).

```
## Python
>>> ## Используем веса, заимствованные из TensorFlow
>>> ## https://www.tensorflow.org/api_docs/python/tf/contrib/cudnn_rnn/CudnnGRU
>>> ## Его можно настроить на использование любых других весов
```

```
>>> def calc_gru(X, weights, num_inputs, num_features):
>>>     Us = weights[:(3*num_features*num_inputs)]
>>>     Us = np.reshape(Us, [3, num_features, num_inputs])
>>>
>>>     Ws = weights[(3*num_features*num_inputs):(3*num_features*num_features +
       3*num_features*num_inputs)]
>>>     Ws = np.reshape(Ws, [3, num_features, num_features])
>>>
>>>     Bs = weights[(-6 * num_features) :]
>>>     Bs = np.reshape(Bs, [6, num_features])
>>>     s = np.zeros([129, num_features])
>>>     h = np.zeros([129, num_features])
>>>
>>>     for t in range(X.shape[0]):
>>>         z = sigmoid(np.matmul(Us[0, :, :], X[t, :]) +
>>> np.matmul(Ws[0, :, :], s[t, :]) + Bs[0, :] + Bs[3, :])
>>>         r = sigmoid(np.matmul(Us[1, :, :], X[t, :]) +
>>> np.matmul(Ws[1, :, :], s[t, :]) + Bs[1, :] + Bs[4, :])
>>>         h[t+1, :] = np.tanh(np.matmul(Us[2, :, :], X[t, :]) +
>>> Bs[2, :] +
>>> r*(np.matmul(Ws[2, :, :], s[t, :]) + Bs[5, :]))
>>>         s[t+1, :] = (1 - z)*h[t + 1, :] + z*s[t, :]
>>>
>>> return h, s
```

Механизм GRU в настоящее время является одним из наиболее широко используемых решений RNN. Это простейшая версия архитектуры с долгой кратковременной памятью (LSTM — Long Short-Time Memory), которая работает аналогичным образом. Различия между ячейками GRU и LSTM заключаются в следующем.

- У GRU есть два шлюза, а у LSTM — три. Эти шлюзы используются для определения объема разрешенной новой информации, сохраняемой старой информации и т.д. У LSTM больше шлюзов, а потому больше и параметров.
- LSTM, как правило, обладает лучшей производительностью, но GRU быстрее обучается (из-за меньшего количества параметров). Тем не менее есть свидетельства, когда GRU превосходит LSTM по производительности. Скорее всего, GRU оказываются лучше LSTM при решении неязыковых задач.

Как видите, разница в большей степени зависит от сложности соответствующей модели и производительности вычислительной системы, на которых выполняется обучение, а также от предмета прогнозирования.

Важно разобраться в принципах матричной реализации механизмов GRU и LSTM, чтобы понимать, как они работают. Познакомившись с ними, вы научитесь определять, в каких ситуациях модели не обучаются на определенных

наборах данных и почему. Зачастую в динамике проявляются неучтенные в рамках выбранного формата факторы.

Обратите внимание, что как GRU, так и LSTM помогают решить проблему, которая впервые возникла при использовании RNN, а именно — взрывающиеся и исчезающие градиенты. Из-за повторяющегося применения одних и тех же параметров часто случалось, что градиенты быстро устремлялись к нулю (что плохо) или к бесконечности (что также плохо). Это означает, что обратное распространение было трудным или даже невозможным при развертывании рекуррентной сети. Эта проблема была успешно разрешена с помощью архитектур GRU и LSTM, поскольку они, как правило, поддерживают входы и выходы узлов в диапазонах допустимых значений. Это связано как с особой формой функции активации, которая в них используется, так и со способностью шлюза обновлений к передаче (или блокированию) проходящей через него информации, что позволяет градиенту с большей вероятностью принимать правдоподобные значения, чем это происходит в обычной RNN, лишенной шлюзов.

Хотя GRU и LSTM достаточно легко написать самому, как показано выше, делать это совсем необязательно. Наиболее важная причина обратиться к готовым решениям заключается в их лучшей оптимизации за счет объединения операций матричного умножения. Наиболее эффективная и доступная сторонняя реализация, стоящая нашего внимания, представлена в библиотеке `cuDNN`, разработанной компанией NVIDIA, в которой операции умножения матриц выполняются одновременно, что крайне важно для GRU и LSTM. Использование программного интерфейса `cuDNN` позволило добиться существенного прироста производительности по сравнению с другими решениями, что было признано основным фактором повышения скорости обучения на конкурсе Kaggle. Все основные фреймворки глубокого обучения обеспечивают доступ к этой реализации, хотя в некоторых случаях (например, при работе с `tf.contrib.cudnn_rnn` в библиотеке `TensorFlow`) для ее подключения необходимо использовать специальный программный интерфейс. В других случаях, таких как MXNet, вы будете применять `cuDNN` по умолчанию, если не сделаете что-то необычное с развернутыми узлами.

Продолжение примера по прогнозированию объемов потребления электроэнергии

Теперь давайте перейдем к решению задачи по прогнозированию потребления электроэнергии с помощью RNN. Как и ранее, входные данные будут представлены в формате TNC. Это вполне ожидаемый для RNN формат, и нам не придется его изменять.

```
## Python
>>> def rnn_model(iter_train, window, filter_size, num_filter,
>>>               dropout):
>>>     input_feature_shape = iter_train.provide_data[0][1]
>>>     X = mx.sym.Variable(iter_train.provide_data[0].name)
>>>     Y = mx.sym.Variable(iter_train.provide_label[0].name)
>>>
>>>     rnn_cells = mx.rnn.SequentialRNNCell()
>>>     rnn_.add(mx.rnn.GRUCell(num_hidden=args.rnn_units))
>>>     rnn_cells.add(mx.rnn.DropoutCell(dropout))
>>>     outputs, _ = rnn_cells.unroll(length=window, inputs=X,
>>>                                   merge_outputs=False)
>>>
>>>     output = mx.sym.FullyConnected(data=outputs[-1],
>>>                                    num_hidden =
                                       input_feature_shape[2])
>>>     loss_grad = mx.sym.LinearRegressionOutput(data = output,
>>>                                    label = Y)
>>>
>>>     return loss_grad, [v.name for v in iter_train.provide_data],
>>>                       [v.name for v in iter_train.provide_label]
```

Производительность модели разочаровывает, учитывая, что она исходно предназначена для обработки временны*х* данных.

```
Epoch               Training Cor  Validation Cor
0                   0.072042      0.069731
1                   0.182215      0.172532
2                   0.297282      0.286091
3                   0.371913      0.362091
4                   0.409293      0.400009
5                   0.433166      0.422921
6                   0.449039      0.438942
7                   0.453482      0.443348
8                   0.451456      0.444014
9                   0.454096      0.448437
10                  0.457957      0.452124
11                  0.457557      0.452186
12                  0.463094      0.455822
13                  0.469880      0.461116
14                  0.474144      0.464173
15                  0.474631      0.464381
16                  0.475872      0.466868
17                  0.476915      0.468521
18                  0.484525      0.477189
19                  0.487937      0.483717
20                  0.487227      0.485799
21                  0.479950      0.478439
22                  0.460862      0.455787
23                  0.430904      0.427170
24                  0.385353      0.387026
TESTING PERFORMANCE
{'COR':0.36212805}
```

Здесь важно разобраться, почему модель не демонстрирует хорошую производительность. Разве мы не предоставили достаточно параметров для RNN? Можно ли улучшить результат за счет добавления в модель некоторых архитектурных решений, которые обычно применяются в подобных случаях, например, учесть фактор внимания? (Относится как к RNN, так и к сетям прямого распространения, рассматриваемым ранее.)

Модель достигла своих пиковых показателей в процессе обучения немного раньше, чем модели прямого распространения или сверточные модели. Это может быть вызвано как нехваткой параметров для описания набора данных, так и тем, что RNN специально приспособлена для обработки таких данных, или чем-то еще. Чтобы понять истинную причину, вам придется провести дополнительное тестирование с подбором настроек.

Изобретение автокодировщиков

Иногда вы можете столкнуться с набором данных, на котором даже очень простая модель показывает необычайно впечатляющие результаты.[4] Но зачастую добиться хорошей скорости вычисления позволяет только нестандартный подход к решению задачи. Например, на ранних этапах исследования было обнаружено, что простое улучшение RNN может существенно повысить производительность в задачах моделирования последовательностей. Хотя изначально RNN разрабатывалась для изучения языков и машинного перевода, она часто показывала хорошие результаты при решении более сложных задач, таких как прогнозирование энергопотребления или предсказание цен на акции. Эта модель, известная как *автокодировщик*, или `seq2seq`, используется очень часто, и вам следует обязательно включить ее в набор инструментов глубокого обучения для временны*х* рядов (рис. 10.12). Мы будем обращаться к ее помощи в примерах нескольких глав, в которых рассматриваются задачи анализа реальных временны*х* рядов.

Автокодировщик состоит из двух рекуррентных слоев, но не в традиционном понимании, предполагающем последовательную обработку каждого входа. В автокодировщике первый слой работает до конца. Затем его скрытое состояние передается на второй слой, который принимает это скрытое состояние и собственные выходы в качестве новых входных данных для следующего шага. Эта модель была продиктована идеей машинного перевода, согласно которой вывод прогноза на каждом временно*м* шаге не имеет смысла, поскольку в разных языках порядок слов и понятий может существенно различаться, но означать одно и то же. Предположение заключалась в том, что после обработки временно*го* ряда

[4]Ходят слухи, что в какой-то момент Google Translate работал на довольно простой семислойной сети LSTM. Однако не стоит недооценивать вклад громадных наборов данных и продуманного обучения. Не все простые модели одинаковы!

первым слоем скрытое состояние могло представлять своего рода завершенную функциональность. Ее результат передавался в новую модель, которая постепенно раскрывала смысл нового языка, комбинируя результат с собственным выводом на каждом временном шаге.

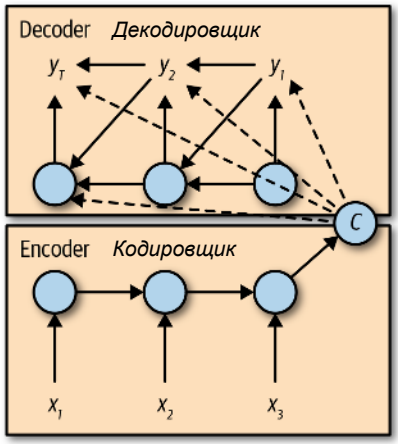


Рис. 10.12. *Автокодировщик, также известный как модель* `seq2seq`*, — популярное средство обработки языковых структур и моделирования, которое также успешно применяется для анализа временных рядов*

Как упоминалось ранее, несмотря на целевое предназначение, заключающееся в обработке языка, эта модель хорошо проявляет себя при решении традиционных задач, таких как прогнозирование одно- или многомерных временных рядов. Например, в конкурсе Kaggle по прогнозированию веб-трафика в сообщениях Википедии первое место после подбора многих гиперпараметров и архитектурных компонентов в конечном итоге заняла модель автокодировщика. Преимущество победителя, скорее всего, заключалось в продуманном обучении и поиске гиперпараметров, а также в правильной генерации и выборе полезных признаков.

Комбинированные архитектуры

Очень часто успешные промышленные и конкурсные приложения глубокого обучения, предназначенные для прогнозирования временных рядов, используют обновленную архитектуру, заключающуюся в новых способах применения традиционных механизмов LSTM или комбинировании некоторых других компонентов. Одним из примеров такого решения выступает нейронная сеть 2018 года, предложенная исследователями Университета Карнеги–Меллона. Разработчики

стремились максимально задействовать сильные стороны как сверточных, так и рекуррентных архитектур, а также некоторых других инноваций.

Они создали рекуррентную модель с пропуском слоя, которая настроена так, чтобы учитывать периодичность данных (годовую, еженедельную, ежедневную — в зависимости от характера набора данных), которую также можно задать с помощью гиперпараметра.

Оказалось, что во многих временны*х* рядах есть тренды, которые плохо моделируются нелинейными моделями глубокого обучения. Это означает, что одна только модель глубокого обучения не способна охватить существенные временны*е* масштабы, наблюдаемые в некоторых временны*х* рядах. Исследователи адаптировали архитектуру для решения подобных проблем с помощью традиционной линейной модели, а именно — модели авторегрессии, рассматриваемой в главе 6.

Итоговая модель, *модифицированная LSTNet*, комбинировала выходы модели AR и модели с традиционным рекуррентным слоем и параллельным пропуском рекуррентного слоя. Входы, подаваемые на каждый рекуррентный слой, служили выходами сверточных слоев, которые сворачивались как по оси времени, так и по оси канала (рис. 10.13).

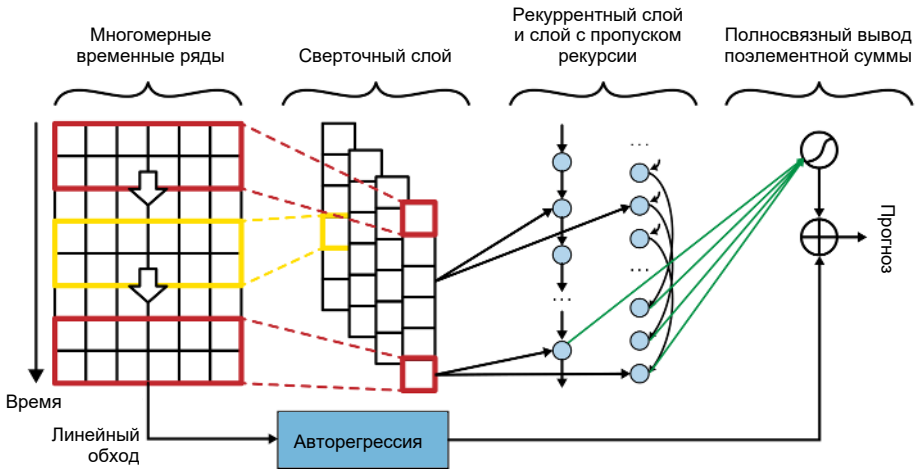


Рис. 10.13. *В модифицированной LSTNet в архитектуру нейронной сети добавлен авторегрессионный компонент (внизу). Архитектура нейронной сети содержит компонент свертки и рекуррентный элемент в последовательном порядке, причем оба компонента используют одни и те же входные данные*

Обнаружено, что для трех из четырех исследуемых наборов данных, охватывающих широкий спектр тематических областей, были получены результаты, которые намного превзошли все опубликованные до этого. Неудача наблюдалась только в случае данных о курсах валют, которые, как известно, очень сложно предсказать из-за высокого соотношения "шум/сигнал" и существования

высокоэффективного рынка, обеспечивающего быстрое рассеивание любого сигнала, что обусловлено стремлением инвесторов к нахождению граничного значения.

Источником вдохновения для следующей разработки послужила научная статья. Мы будем работать с модифицированным кодом (`https://perma.cc/3W4Y-E8E2`),[5] заимствованным из каталога примеров для пакета `MXNet`, подробно описанного разработчиком Оливером Принглом в посте, опубликованном в его блоге (`https://perma.cc/9KM2-RNPK`).

Здесь, как отмечалось ранее, применяется упрощенный код, подготовленный для репозитория `MXNet`, из которого удалены сезонные компоненты и пропуски соединений, а также используется сверточный фильтр только одного размера. Сверточный слой задается так же, как в примере `cnn_model` ранее.

```
## Python
>>> ## Для обработки данные должны быть четырех- или пятимерными
>>> conv_input = mx.sym.reshape(data=X, shape=(0, 1, win, -1))
>>>
>>> ## Сверточный компонент
>>> ## Дополнение данных до конца временного окна
>>> cnn_output = mx.sym.pad(data=conv_input,
>>>                         mode="constant",
>>>                         constant_value=0,
>>>                         pad_width=(0, 0,
>>>                                    0, 0,
>>>                                    0, sz_filt - 1,
>>>                                    0, 0))
>>> cnn_output = mx.sym.Convolution(data=cnn_output,
>>>                                 kernel=(sz_filt,
>>>                                         input_feature_shape[2]),
>>>                                 num_filter=n_filter)
>>> cnn_output = mx.sym.Activation(data=cnn_output, act_type='relu')
>>> cnn_output = mx.sym.reshape(mx.sym.transpose(data=cnn_output,
>>>                                               axes=(0, 2, 1, 3)),
>>>                              shape=(0, 0, 0))
>>> cnn_output = mx.sym.Dropout(cnn_output, p=drop)
```

RNN применяется не к исходным данным, а к результатам работы сверточного компонента.

```
## Python
>>> ## Рекуррентный компонент
>>> stacked_rnn_cells = mx.rnn.SequentialRNNCell()
>>> stacked_rnn_cells.add(mx.rnn.GRUCell(num_hidden=args.rnn_units))
>>> outputs, _ = stacked_rnn_cells.unroll(length=win,
>>>                                       inputs=cnn_output,
>>>                                       merge_outputs=False)
```

[5] Этот код можно найти в персональном репозитории Оливера Прингла (Oliver Pringle) на GitHub.

```
>>>
>>> rnn_output     = outputs[-1]
>>> n_outputs      = input_feature_shape[2]
>>> cnn_rnn_model = mx.sym.FullyConnected(data=rnn_output,
>>>                                       num_hidden=n_outputs)
```

Наконец, как показано далее, параллельно с CNN/RNN обучаются AR-модели (всего — 321 AR-модель, по одной на столбец/переменную/электростанцию). Совершается проход по станциям так, что в отдельный момент времени в модели настраивается по одной станции.

```
## Python
>>> ## AR-компонент
>>> ar_outputs = []
>>> for i in list(range(input_feature_shape[2])):
>>>     ar_series = mx.sym.slice_axis(data=X,
>>>                                   axis=2,
>>>                                   begin=i,
>>>                                   end=i+1)
>>>     fc_ar = mx.sym.FullyConnected(data=ar_series, num_hidden=1)
>>>     ar_outputs.append(fc_ar)
>>> ar_model = mx.sym.concat(*ar_outputs, dim=1)
```

Полный код выглядит следующим образом.

```
## Python
>>> def simple_lstnet_model(iter_train, input_feature_shape, X, Y,
>>>                         win, sz_filt, n_filter, drop):
>>>     ## Для обработки данные должны быть четырех- или пятимерными
>>>     conv_input = mx.sym.reshape(data=X, shape=(0, 1, win, -1))
>>>
>>>     ## Сверточный компонент
>>>     ## Дополнение данных до конца временного окна
>>>     cnn_output = mx.sym.pad(data=conv_input,
>>>                             mode="constant",
>>>                             constant_value=0,
>>>                             pad_width=(0, 0,
>>>                                        0, 0,
>>>                                        0, sz_filt - 1,
>>>                                        0, 0))
>>>     cnn_output = mx.sym.Convolution(data = cnn_output,
>>>                                     kernel = (sz_filt,
>>>                                     input_feature_shape[2]),
>>>                                     num_filter = n_filter)
>>>     cnn_output = mx.sym.Activation(data = cnn_output,
>>>                                    act_type = 'relu')
>>>     cnn_output = mx.sym.reshape(mx.sym.transpose(data = cnn_output,
>>>                                                  axes = (0, 2, 1, 3)),
>>>                                 shape=(0, 0, 0))
>>>     cnn_output = mx.sym.Dropout(cnn_output, p = drop)
>>>
```

```
>>>     ## Рекуррентный компонент
>>>     stacked_rnn_cells = mx.rnn.SequentialRNNCell()
>>>     stacked_rnn_cells.add(mx.rnn.GRUCell(num_hidden = args.rnn_units))
>>>     outputs, _ = stacked_rnn_cells.unroll(length = win,
>>>                                           inputs = cnn_output,
>>>                                           merge_outputs = False)
>>>     rnn_output = outputs[-1]
>>>     n_outputs = input_feature_shape[2]
>>>     cnn_rnn_model = mx.sym.FullyConnected(data=rnn_output,
>>>                                           num_hidden=n_outputs)
>>>     ## AR-компонент
>>>     ar_outputs = []
>>>     for i in list(range(input_feature_shape[2])):
>>>        ar_series = mx.sym.slice_axis(data=X,
>>>                                      axis=2,
>>>                                      begin=i,
>>>                                      end=i+1)
>>>        fc_ar = mx.sym.FullyConnected(data = ar_series,
                                         num_hidden = 1)
>>>        ar_outputs.append(fc_ar)
>>> ar_model = mx.sym.concat(*ar_outputs, dim=1)
>>>
>>> output = cnn_rnn_model + ar_model
>>> loss_grad = mx.sym.LinearRegressionOutput(data=output, label=Y)
>>> return (loss_grad,
>>>        [v.name for v in iter_train.provide_data],
>>>        [v.name for v in iter_train.provide_label])
```

Обратите внимание, что эта модель показывает намного лучшую производительность, чем любые другие.

```
Epoch  Training Cor   Validation Cor
0      0.256770       0.234937
1      0.434099       0.407904
2      0.533922       0.506611
3      0.591801       0.564167
4      0.630204       0.602560
5      0.657628       0.629978
6      0.678421       0.650730
7      0.694862       0.667147
8      0.708346       0.680659
9      0.719600       0.691968
10     0.729215       0.701734
11     0.737400       0.709933
12     0.744532       0.717168
13     0.750767       0.723566
14     0.756166       0.729052
15     0.760954       0.733959
16     0.765159       0.738307
17     0.768900       0.742223
18     0.772208       0.745687
19     0.775171       0.748792
```

```
20     0.777806        0.751554
21     0.780167        0.754034
22     0.782299        0.756265
23     0.784197        0.758194
24     0.785910        0.760000
TESTING PERFORMANCE
{'COR': 0.7622162}
```

Очевидно, модель работает намного производительнее, чем другие рассматриваемые нами модели, а значит, в предложенной архитектуре есть что-то, что позволяет эффективно использовать сверточные изображения для представления последовательных данных, на которых обучается рекуррентный слой. Традиционный инструмент статистического анализа — AR-модель — также вносит немалый вклад в общее повышение производительности.[6] Эта модель, безусловно, лучшая из предложенных, и она оказывается в большом отрыве от остальных решений, по крайней мере для того небольшого объема данных, который подвергался анализу. Вы, конечно же, вправе использовать и другие модели, но вам не придется долго раздумывать, чтобы определить явного лидера.

Резюме

Одно из интереснейших наблюдений, сделанных по результатам обучения, проводимого в этой главе, заключается в том, что характеристики моделей оказались далеко не такими, как ожидалось. Простая сеть прямого распространения существенно превзошла некоторые другие модели, которые концептуально имеют более сложную организацию. Однако это ни в коей мере не значит, что одни модели лучше или хуже других для предложенного набора данных по целому ряду причин.

- В каждой модели используется свое, отличное от других, количество параметров. Возможно, что разные типы моделей будут характеризоваться сильно различающейся производительностью для разного количества параметров. Также можно "поиграть" со сложностью модели, например, задавая разное количество сверточных/рекуррентных слоев или фильтров/скрытых элементов.
- Гиперпараметры моделей не настраивались. Иногда правильное задание гиперпараметров оказывает наибольшее влияние на производительность модели.
- Данные изучены недостаточно, чтобы получить хотя бы предварительное представление о том, какая модель лучше или хуже, учитывая временны*е* корреляции и корреляции между различными столбцами/электростанциями.

[6] Если сомневаетесь, то попробуйте провести обучение модели без AR-компонента. Его код удалить совершенно нетрудно.

Об улучшении навыков

Просматривая публикации в научных журналах, материалах конференций и статьях на arXiv в поисках источников сведений о технологиях глубокого обучения для временных рядов, вы будет сильно разочарованы.

- Многие исследователи временных рядов в первую очередь считают себя экспертами в предметной области и лишь во вторую — специалистами по анализу временных рядов. Это означает, что вам придется разбираться в смежных областях, таких как экономика, здравоохранение и исследование климатических изменений.
- Специалисты по глубокому обучению часто считают глубокое обучение или нейронные сети технологиями, основанными на свертках. Поэтому, попытавшись разобраться в "рекуррентных нейронных сетях", обязательно проводите поиск по указанным названиям. В большинстве работ, в названиях которых не встречается аббревиатура RNN (или схожая с ней), вы не найдете описания рекуррентных нейронных сетей.
- К сожалению, еще никто не написал модель глубокого обучения, способную выполнить поиск документов по глубокому обучению, хотя это, несомненно, был бы достойный проект.

Дополнительные источники

- Исторические документы

  Sepp Hochreiter and Jurgen Schmidhuber, "Long Short-Term Memory," Neural Computation 9, no. 8 (1997):1735–80, `https://perma.cc/AHR3-FU5H`

  В этой основополагающей работе 1997 года впервые описана архитектура с долгой кратковременной памятью (LSTM), а также предложено несколько экспериментальных тестов, которые до сих пор используются для обучения производительности нейронной сети при анализе последовательностей данных.

  Peter G. Zhang, Eddy Patuwo, and Michael Hu, "Forecasting with Artificial Neural Networks: The State of the Art," International Journal of Forecasting 14, no. 1 (1998): 35–62, `https://perma.cc/Z32G-4ZQ3`

  Здесь приведен обзор технологий анализа временных рядов и глубокого обучения на 1998 год.

- Сети RNN

Aurelien Geron, "Processing Sequences Using RNNs and CNNs," in Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition (Sebastopol: O'Reilly Media, Inc., 2019).

В этой популярной книге Орельен Жерон приводит очень много примеров применения глубокого обучения к данным временных рядов. Если вы чувствуете себя уверенно с такими инструментами, то обязательно воспользуйтесь виртуальным ноутбуком Jupyter (`https://perma.cc/D3UG-59SX`) для просмотра наглядных примеров использования различных типов моделей для обработки последовательных данных, включая некоторые упражнения с готовыми решениями.

Valentin Flunkert, David Salinas and Jan Gasthaus, "DeepAR: Probabilistic Forecasting with Autoregressive Recurrent Networks," unpublished paper, 2017, `https://perma.cc/MT7N-A2L6`

Содержит описание новаторской модели Amazon, которая была разработана с оглядкой на анализ временных рядов с данными о ее розничных продажах, которые представлены в самых разных временных масштабах и имеют различные тренды. Одним из нововведений модели была способность составлять вероятностные прогнозы, в отличие от общепринятых точечных оценок, которые, как правило, являются результатом анализа, проводимого с помощью технологий глубокого обучения.

Lingxue Zhu and Nikolay Laptev, "Deep and Confident Prediction for Time Series at Uber," paper presented at the 2017 IEEE International Conference on Data Mining Workshops (ICDMW), New Orleans, LA, `https://perma.cc/PV8R-PHV4`

В статье описывается еще один пример применения вероятностных и статистических инструментов для модификации стандартной RNN. В данном случае компания Uber предложила новую глубокую байесовскую модель, которая проводила как точечную оценку, так и оценку неопределенности, которая разворачивалась в высокопроизводительную среду.

Zhengping Che et al., "Recurrent Neural Networks for Multivariate Time Series with Missing Values," Scientific Reports 8, no. 6085 (2018), `https://perma.cc/4YM4-SFNX`

Яркий пример современного подхода к обработке медицинских временных рядов. В статье продемонстрирован пример совместного использования GRU и новаторских архитектур для учета отсутствующих данных и превращения отсутствующих данных в информативный признак. Авторы описывают нейронные сети, способные учитывать все клинические показатели, используемые в настоящее время, для прогнозирования

состояния здоровья пациентов, которые пребывают на стационарном лечении. Прекрасный пример того, как интуитивное и простое для понимания изменение простой и широко используемой архитектуры RNN (GRU) может привести к отличным результатам на хорошем наборе данных.

- Сети CNN

  Aaron van den Oord and Sander Dieleman, "WaveNet: A Generative Model for Raw Audio DeepMind," DeepMind blog, September 8, 2016, `https://perma.cc/G37Y-WFCM`

  Блог, содержащий чрезвычайно подробное и простое для понимания описание улучшенной архитектуры CNN, используемой для совершенствования технологий преобразования текста в речь, а речи — в текст для разных языков и носителей. Обновленная архитектура характеризуется повышенной производительностью и была развернута для решения других задач искусственного интеллекта, связанных с обработкой последовательностей данных, в первую очередь прогнозирования временных рядов.

- Глубокое обучение

  Vera Rimmer et al., "Automated Website Fingerprinting Through Deep Learning," paper presented at NDSS 2018, San Diego, CA, `https://perma.cc/YR2G-UJUW`

  В этом документе рассматривается способ использования алгоритмов глубокого обучения для получения данных о поведении пользователей в Интернете через снятие "отпечатков пальцев" с веб-сайтов. В частности, авторы выделили способ, который различные архитектуры нейронных сетей могут использовать для преодоления взлома систем защиты конфиденциальных данных пользователей, основанной на использовании отпечатков веб-сайтов.

  CPMP, "Second Place Solution to the Kaggle Web Traffic Forecasting Competition," Kaggle blog, 2017, `https://perma.cc/UUR4-VNEU`

  Этот пост, написанный еще до завершения конкурса, содержит размышления второго призера о разработке решения для прогнозирования, основанного на смешанной архитектуре машинного/глубокого обучения. В сообщении, опубликованном в блоге (`https://perma.cc/73M3-D7DW`), приведен также ретроспективный комментарий. Здесь рассматривается отличный пример совместного использования современных пакетов и соответствующего стиля программирования. Описание решения, занявшего первое место, вместе с обсуждением его архитектуры на основе нейронной сети (`https://perma.cc/G9DW-T8LE`) можно найти в репозитории GitHub (`https://perma.cc/K6RW-KA9E`).

ГЛАВА 11

Оценка ошибок

В предыдущих главах вы имели возможность познакомиться с некоторыми показателями, применяемыми для сравнения моделей или определения того, насколько хорошо модель выполняет возложенные на нее задачи. В этой главе вы узнаете о лучших методиках оценки точности прогнозов, в первую очередь составляемых в ходе анализа временны*х* рядов.

Те из вас, кто плохо знаком с прогнозированием временны*х* рядов, должны понимать, что стандартная перекрестная проверка плохо подходит для оценки результатов прогнозирования, и обычно такую оценку проводить не рекомендуется. Здесь вы не сможете выбирать случайные наборы данных для обучения, проверки и тестирования, к тому же подбираемые без привязки ко времени.

Но это не самая большая трудность. Приходится учитывать временну*ю* взаимосвязь между отдельными выборками даже несмотря на то, что они кажутся независимыми. Предположим, вам нужно классифицировать временны*е* ряды и вы располагаете большим количеством выборок временны*х* рядов, каждая из которых представляет отдельную точку данных. Может показаться заманчивым выбирать временны*е* ряды для обучения, проверки и тестирования случайным образом, но это большая ошибка. Недостаток такого подхода состоит в невозможности отражения в модели особенностей ее использования, а именно того, что она обучается на более ранних данных, а тестируется — на более поздних.

Нельзя допустить, чтобы в модель поступала будущая информация, потому что в действительности такие ситуации попросту невозможны. Если это произойдет, то ошибка прогнозирования, которую вы получите для своей модели, будет заведомо ниже на этапе тестирования, чем в производственном процессе, поскольку тестирование будет заключаться в перекрестной проверке модели с учетом будущей информации (получите обратную связь с выбором модели).

Рассмотрим конкретный сценарий, в котором наглядно показано, как такое может произойти. Представьте, что вы создаете программу мониторинга качества воздуха для крупных городов на Западном побережье США. Пусть в обучающий набор включены все данные за 2017 и 2018 годы для Сан-Франциско, Солт-Лейк-Сити, Денвера и Сан-Диего, а набор тестирования представлен одинаковым диапазоном данных для Лас-Вегаса, Лос-Анджелеса, Окленда и Феникса. Вы обнаружите, что модель, применяемая для определения качества воздуха,

особенно хороша в Лас-Вегасе и Лос-Анджелесе и лучше всего показывает себя в 2018 году. Замечательно!

Но как только вы попытаетесь воспроизвести процесс обучения модели на данных предыдущих десятилетий, сразу же обнаружите, что точность ее тестирования отличается от точности обучения/тестирования на других данных. Если вспомнить об аномальных лесных пожарах в Южной Калифорнии в 2018 году и заключить, что они были учтены в первоначальном тестировании/обучении, то становится понятно, что обучающий набор предоставляет своеобразное "окно в будущее". Это важная причина, чтобы избегать стандартной перекрестной проверки.

Иногда проникновение информации из будущего не создает проблем при выборе модели. Например, если вы просто хотите понять динамику временного ряда, определяя, насколько хорошо его данные подлежат прогнозированию без акцентирования внимания на точности таких прогнозов, и проверяя возможное соответствие модели имеющимся данным, то в этом случае включение будущих данных будет не только допустимо, но полезно, хотя и может привести к переобучению, чего стоит опасаться. Можно смело утверждать, что, даже получив в свое распоряжение корректный набор тестирования, лишенный обратной утечки информации из будущего, не стоит забывать об опасностях перекрестной проверки при прогнозировании временных рядов.

Закончив с общими замечаниями, рассмотрим реальный механизм разделения данных модели на обучающие, проверочные наборы и наборы для тестирования. Затем изучим общие критерии определения хорошего прогноза и прогноза настолько хорошего, насколько это только возможно. Вы также узнаете об оценке неопределенности при использовании методов прогнозирования, которые в явном виде ее не учитывают или не выделяют ошибки в получаемых результатах. В заключение рассмотрим основные трудности, с которыми вы можете столкнуться при создании модели временного ряда или ее подготовке к запуску. Их понимание позволит избежать многих недоразумений!

Проверка прогнозов: основные положения

Самое важное при составлении прогноза — убедиться, что он строится исключительно на основе заблаговременно и в полном объеме полученных данных. Следуя этому принципу, необходимо не только помнить о происходящих событиях, но и понимать, когда данные стали доступны для анализа.[1]

[1] Между прочим, моделирование с данными или без данных, доступных в определенное время, которое выполняется для демонстрации важности своевременного получения информации, может стать побуждающим фактором для инженеров и исследователей к снабжению отдельных входных пакетов большим приоритетом, но только в случаях, если можно доказать их решающее влияние на точность модели и/или успех мероприятия в целом.

Хотя это кажется достаточно простым утверждением, помните, что самая обычная предварительная обработка, такая как экспоненциальное сглаживание, может непреднамеренно привести к передаче информации с этапа обучения на этап тестирования. В этом можно легко удостовериться самостоятельно, подгоняя модель линейной регрессии сначала к авторегрессионному временному ряду, а затем — к экспоненциально сглаженному авторегрессионному временному ряду. Легко заметить, что чем сильнее сглаживаются временные ряды и чем больше период сглаживания, тем "точнее" прогнозы. А все потому, что здесь прогнозированию отводится все меньше и меньше места, поскольку большая часть получаемых значений определяется по экспоненциальному среднему предыдущих значений. Это довольно опасный подход, и он настолько предательски обманчив, что до сих пор применяется в научных работах!

Ввиду этих и других трудноразличимых опасностей просачивания будущей информации в прошлое или наоборот любая модель в обязательном порядке должна проходить тестирование на исторических данных, последовательно чередуемое с обучением, перекрестной проверкой и финальным тестированием.

При тестировании на исторических данных модель разрабатывается для одного набора или диапазона дат, а затем тщательно проверяется на доступных исторических данных, раскрывая весь потенциал возможных условий и параметров. Важно располагать весомыми причинами для тестирования на исторических данных конкретной модели, а количество рассматриваемых моделей должно быть как можно меньше. Как известно большинству специалистов по анализу данных, чем больше моделей подвергается тестированию, тем больше вероятность переоценки данных — подбора модели, в которой слишком большое внимание уделяется отдельным аспектам набора данных, вместо выделения обобщающих характеристик. К сожалению для практикующих исследователей временных рядов, это путь шаткого компромисса, который часто приводит к непредвиденным результатам при запуске модели в производство.

Но в каком виде реализуется тестирование на исторических данных? Попробуем создать структуру, подобную применяемой для перекрестной проверки, но учитывающую временную направленность данных. Общую парадигму, в которой последовательность "временной" обработки данных представляется в алфавитном порядке, можно проиллюстрировать следующим образом.

```
Обучение на данных [A]          Тестирование на данных [B]
Обучение на данных [A B]        Тестирование на данных [C]
Обучение на данных [A B C]      Тестирование на данных [D]
Обучение на данных [A B C D]    Тестирование на данных [E]
Обучение на данных [A B C D E]  Тестирование на данных [F]
```

Такая структура тестирования проиллюстрирована на рис. 11.1.

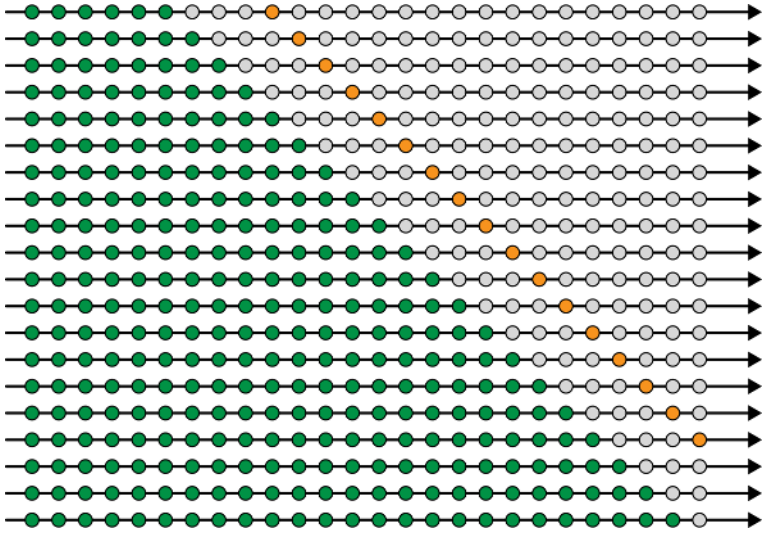


***Рис. 11.1.** Структура для тестирования модели*

Показанная выше структура позволяет перемещать окно обучающих данных, а не расширять его. В подобном случае обучение будет выполняться приблизительно так.

```
Обучение на данных [A B]  Тестирование на данных [C]
Обучение на данных [B C]  Тестирование на данных [D]
Обучение на данных [C D]  Тестирование на данных [E]
Обучение на данных [D E]  Тестирование на данных [F]
```

Какой из методов выбрать, во многом зависит от того, рассматривается ли поведение временного ряда как изменяющееся во времени. Если вы считаете, что ему свойственны временные изменения, то лучше используйте скользящее окно, чтобы во всех периодах проводить тестирование модели, обученной на наиболее актуальных данных. С другой стороны, чтобы избежать переобучения, лучше использовать расширяющееся окно, поддерживающее модель с более строгой формой, чем окно фиксированной длины.

Поскольку такой принцип разделения данных является обычным делом в обучении, в инфраструктуру языков R и Python добавлены специальные инструменты ее реализации.

- В языке Python разделения данных проще всего добиться с помощью метода `sklearn.model_selection.TimeSeriesSplit()`.
- В языке R временной проход модели вперед по схеме тестирования на исторических данных с выводом сообщений об ошибках осуществляется с помощью функции `tsCV()` пакета `forecast`.

В языках R и Python вы найдете некоторые другие пакеты, обеспечивающие выполнение равнозначных задачи. Более того, всегда можно написать собственный код функций разделения данных, реализовав в нем предположения о поведении модели тестирования в целевом проекте. Возможно, при тестировании нужно пропустить определенные периоды времени, поскольку в них наблюдается аномальная динамика, или, наоборот, увеличить точность в заданных временных интервалах.

Предположим, вы работаете с финансовыми данными. В зависимости от поставленных целей будет целесообразно исключить из рассмотрения данные за необычные периоды, например за кризисный 2008 год. Или же, обрабатывая данные розничных торговых сетей, чрезвычайно важно повысить точность модели для сезона рождественских покупок даже в ущерб точности прогноза для других сезонов, демонстрирующих более слабую динамику.

Особенности тестирования на исторических данных

Обязательно учитывайте динамику обучаемой модели при разработке структуры модели тестирования на исторических данных, особенно при обучении на данных за определенный временной диапазон.

В традиционных статистических моделях, таких как ARIMA, при выборе параметров модели все точки данных учитываются одинаково, поэтому чем большее данных, тем менее точной становится модель, параметры которой изменяются во времени. Это также справедливо для моделей машинного обучения, в которых все данные обучения учитываются в равной степени.[2]

С другой стороны, пакетные стохастические методы могут работают к весами и оценками, которые изменяются во времени. Таким образом, модели нейронных сетей, обученные с использованием типичных методов стохастического градиентного спуска, будут в некоторой степени учитывать временную природу данных, если обрабатывать их в хронологическом порядке. Последние поправки для весов градиента отражают последние данные. В большинстве случаев модели нейронных сетей для временных рядов обучаются на данных в хронологической последовательности, поскольку это чаще приводит к лучшим результатам, чем в моделях, которые обучены на данных, извлеченных в случайном порядке.

Избегайте пропусков в данных. Одна из важных особенностей временных рядов состоит в том, что точки данных автокоррелируют между собой. Следовательно, нельзя случайным образом выбирать точки во временном ряду для проверки или тестирования,

[2]Обратите внимание, что вы можете написать собственную функцию взвешивания потерь, приписывая более свежим данным более крупные веса, но для этого вам понадобятся навыки программирования выше среднего и знания методов численной оптимизации.

потому что это приведет к нарушению автокорреляционных зависимостей в данных. В результате модель не сможет правильно распознать авторегрессионный компонент в данных, что совершенно неприемлемо.

Модели пространства состояний также предоставляют возможности для адаптации с течением времени, заключающиеся в сокращении периода обучения, поскольку широкое окно не будет препятствовать росту апостериорной оценки с течением времени.

Временные снимки модели

Поскольку модели тестируются после обучения проходом по временным точкам по предыдущим данным, вам понадобится простой способ сохранения состояния модели, снабженного временной отметкой, чтобы понимать, с какого наиболее раннего момента времени можно начинать ее использовать. Тем самым вы предотвратите непреднамеренное тестирование модели на данных обучения. Это также позволит протестировать несколько разных моделей для отличных временных периодов времени на одних и тех же данных. Кроме того, так можно выяснить, влияет ли на точность модели временной период данных тестирования, выбранных для ее обучения. В конечном итоге все это поможет определиться с частотой повторного обучения моделей временных рядов в выходном коде.

Достаточно хороший прогноз

То, когда прогноз можно считать достаточно хорошим, зависит от многих факторов: от общих целей, минимального уровня точности, с которым вы согласны мириться, а также от ограничений и характера рабочих данных. Если данные характеризуются высоким соотношением "шум/сигнал", то не стоит ожидать от модели больших свершений.

Помните, что модели временного ряда не идеальны. Но вы должны сделать все возможное, чтобы заставить их быть такими же или даже более точными, чем методы вычисления систем дифференциальных уравнений, описывающих климатические изменения, советы хорошо информированного биржевого трейдера или сведения из учебника по медицине, в котором описываются принципы классификации ЭЭГ-кривых. Оценивая модель, учитывайте известные экспертные оценки прогнозирования в выбранной предметной области, поскольку они определяют верхний предел точности во многих задачах прогнозирования.

Но бывает и так, что вам хорошо известно о несовершенстве построенной модели, и потому вы хотите добиться от нее большего. Для дальнейшего ее улучшения рассмотрите следующие возможности.

Постройте график результатов работы модели для набора тестовых данных

Распределение, создаваемое моделью, должно соответствовать распределению прогнозируемых значений в отсутствие изменения режима или основной тенденции. Например, если попытаться предсказать цены акций и знать, что они растут и падают примерно с одинаковой частотой, то модель можно считать неправильной, когда она предсказывает одно только повышение цены. Конечно, иногда по вполне очевидным причинам распределения значений неизвестны, но в остальных случаях можно и нужно применять статистические сведения для сравнения выходных данных модели с фактически наблюдаемыми значениями.

Составьте график временной зависимости остатков модели

Если остатки не однородны по времени, то модель считается недостаточно обученной. Временное поведение остатков может указать на дополнительные параметры, которые необходимо включить в модель для описания ее временного поведения.

Сравните модель с простой нулевой моделью, в которой учитывается течение времени

Простая нулевая модель состоит в том, что каждый прогноз для момента времени t должен совпадать со значением в момент времени $t - 1$. Если ваша модель не опережает по точности такую простую модель, то нет этому никакого оправдания. Если простая универсальная наивная модель может превзойти созданную вами модель, значит, с вашей моделью что-то не в порядке. Уточните функцию потерь или повторно проведите предварительную обработку данных, не изменяя сетку гиперпараметров. С другой стороны, плохой результат может быть следствием сильно зашумленных данных, что означает бесполезность предложенной модели для решения возложенных на нее задач.[3]

Изучите, как модель справляется с выбросами

Во многих предметных областях под выбросами понимается именно то, что они значат. Такие события не подлежат никакому предсказанию[4], т.е. лучшее, что может предпринять модель, — просто проигнорировать выбросы, не принимая их в расчет. Если модель очень хорошо предсказывает выбросы, то это может быть признаком переобучения или неправильного подбора функции потерь. Все зависит от структуры модели и используемых вами функций потерь, но для большинства приложений понадобится модель, прогнозы

[3] Простую нулевую модель удивительно сложно превзойти.

[4] В некоторых случаях бывает нечему учиться для принятия будущих решений.

которой не столь большие, как выбросы в исходном наборе данных. Конечно, приведенные рекомендации не касаются случаев, когда стоимость предупреждения выбросов очень высока, а задача прогнозирования сводится к тому, чтобы, по возможности, предупреждать выбросы.

Исследуйте временную чувствительность

Должно ли качественно сходное поведение в связанных временны*х* рядах приводить к связанным результатам модели? Используя познания в основной динамике системы, убедитесь, что это так и что модель распознает и обрабатывает аналогичные временны*е* шаблоны одинаковым образом. Например, если один временной ряд демонстрирует тенденцию к повышению со смещением на 3 единицы в день, а в другом наблюдается тренд к повышению со смещением на 2,9 единицы в день, то вы, скорее всего, захотите удостовериться в том, что в конечном итоге прогнозы, полученные для таких рядов, были похожими. Также нужно проверить правильность ранжирования прогнозов по сравнению с исходными данными (больший дрейф должен привести к большему значению прогноза). Если это не так, то модель требует переобучения.

Это далеко не полный список способов тестирования моделей временны*х* рядов, но он послужит хорошей отправной точкой для приобретения опыта в данной предметной области.

Оценка неопределенности с помощью моделирования

Одно из преимуществ традиционного статистического подхода к анализу временны*х* рядов состоит в том, что в нем оценка неопределенности рассчитывается согласно строгим аналитическим формулам. Однако даже в таких случаях — и, само собой, в случае нестатистических методов анализа — бывает полезно рассмотреть неопределенность, обусловленную моделью прогнозирования, с помощью вычислительных методов. Один из интуитивно понятных и доступных для расчетов способов сделать это — простое моделирование.

Предположим, мы провели анализ процесса AR(1), определенного в одной из предыдущих глав. Вспомним, что процесс AR(1) можно выразить следующим образом.

$$y_t = \phi y_{t-1} + e_t.$$

После подгонки модели нужно понять, насколько изменчивой может быть оценка коэффициента ϕ. В подобных случаях, чтобы узнать это, процесс многократно моделируется по методу Монте-Карло. Такая задача легко выполняется с помощью инструментов языка R, учитывая все, что нам известно об AR-процессах из главы 6.

```
## R
> require(forecast)
>
> phi          <- 0.7
> time_steps   <- 24
> N            <- 1000
> sigma_error  <- 1
>
> sd_series <- sigma_error^2 / (1 - phi^2)
> starts    <- rnorm(N, sd = sqrt(sd_series))
> estimates <- numeric(N)
> res       <- numeric(time_steps)
>
> for (i in 1:N) {
>   errs = rnorm(time_steps, sd = sigma_error)
>   res[1] <- starts[i] + errs[1]
>
>   for (t in 2:time_steps) {
>      res[t] <- phi * tail(res, 1) + errs[t]
>   }
>   estimates <- c(estimates, arima(res, c(1, 0, 0))$coef[1])
> }
>
> hist(estimates,
>      breaks = 50)
```

В качестве результата выводится гистограмма оценки коэффициента ϕ, подобная показанной на рис. 11.2.

Кроме того, можно получить представление о диапазоне оценок и квантилей, воспользовавшись функцией `summary()`, примененной к оценкам.

```
## R
> summary(estimates1)
     Min.   1st Qu.  Median     Mean  3rd Qu.   Max.
  -0.3436   0.4909   0.6224   0.5919  0.7204  0.9331
```

Мы также можем обратиться к методам бутсрэпа, чтобы получить ответы на более сложные вопросы. Предположим, что нужно знать, какие вычислительные затраты связаны с упрощением модели относительно исходного варианта. Пусть изучаемый процесс представлен как AR(2), хотя исходно мы определили его как AR(1). Чтобы выяснить, как это сказывается на оценке модели, модифицируем предыдущий код R следующим образом.

```
> ## Допустим, что истинный процесс – AR(2),
> ## но ввиду его высокой сложности переключаемся на arima.sim
> phi_1 <- 0.7
> phi_2 <- -0.2
>
> estimates <- numeric(N)
```

```
> for (i in 1:N) {
>   res <- arima.sim(list(order = c(2,0,0),
>                         ar = c(phi_1, phi_2)),
>                         n = time_steps)
>   estimates[i] <- arima(res, c(1, 0, 0))$coef[1]
> }
>
> hist(estimates,
>      breaks = 50)
```

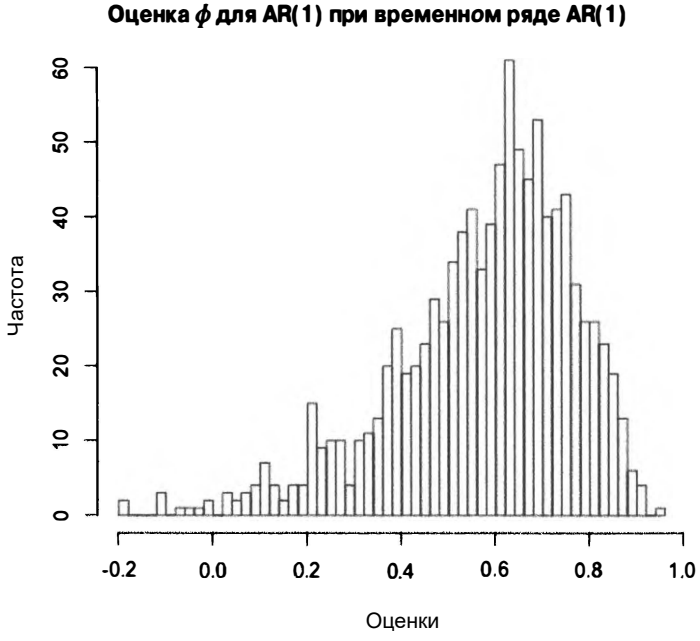


Рис. 11.2. *Распределение оценок коэффициента* ϕ

Результирующее распределение показано на рис. 11.3. Как видите, оно выглядит не столь гладким и четким для неправильно определенной модели, как в случае правильного подбора модели (рис. 11.2).

Вам может показаться, что распределения не так уж и сильно различаются, и будете правы. Это подтверждается описательными статистиками.[5]

```
## R
> summary(estimates)
   Min.  1st Qu.   Median    Mean     3rd Qu.    Max.
-0.5252  0.4766   0.6143   0.5846   0.7215     0.9468
```

[5] Будет ли плохое приближение проблемой, зависит от конкретных коэффициентов. В данном случае влияние неправильного выбора модели оказалось незначительным, но в других случаях оно может быть катастрофическим.

Легко видеть, что диапазон оценок шире для правильно определенной модели, а оценка для члена первого порядка несколько хуже, чем для правильно заданной модели, но отклонение не слишком велико.

Описанный подход позволяет решить проблемы влияния недооценки порядка модели на общую оценку ее точности. Для решения потенциальных проблем и понимания причин возникновения потенциальных ошибок, учитывающих рассматриваемые возможности, можно использовать различные сценарии моделирования.

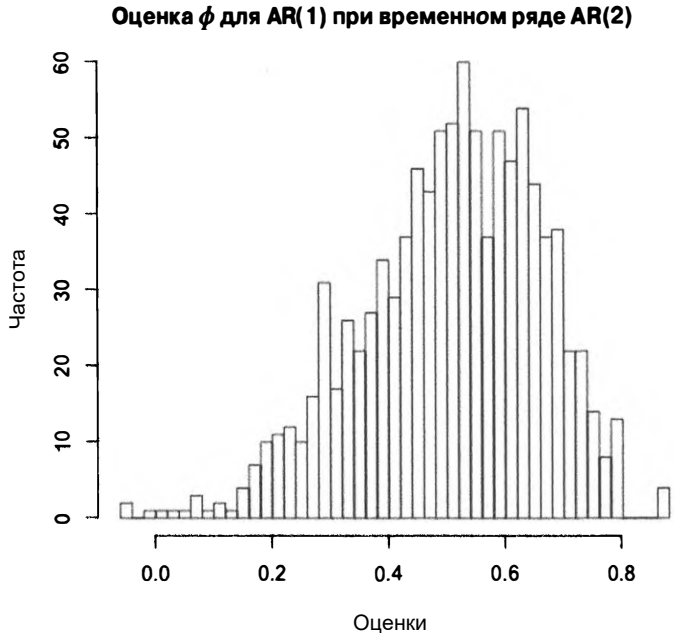


Рис. 11.3. *Оценка распределения для первого коэффициента запаздывания в модели AR(1), аппроксимирующей истинный процесс AR(2)*

Прогнозирование на несколько шагов вперед

В большинстве предыдущих глав мы занимались прогнозированием всего на шаг вперед, но при желании его можно выполнить вперед на несколько временных шагов. Необходимость в этом, наряду с другими причинами, возникает тогда, когда данные временных рядов имеют большее временное разрешение, чем значения временных рядов, для которых строится прогноз. Например, при изучении ежедневных цен на акции может возникнуть запрос на прогнозирование цен на акции на месяц вперед, чтобы разработать долгосрочную стратегию управления пенсионными вкладами. Или же, располагая ежеминутно собираемыми показаниями электрической активности мозга, может понадобиться предсказать

инсульт по крайней мере за пять минут до его возникновения, чтобы дать пользователям/пациентам возможность провести упредительную терапию. В каждом из случаев найдутся причины для построения многошаговых прогнозов.

Прогнозирование до края горизонта

Это прогнозирование выполняется так же просто, как установка целевого значения *y* для отражения горизонта прогнозирования. Например, если данные состоят из поминутных индикаторов, а вам нужен прогноз на пять минут вперед, ограничьте входные данные модели моментом времени t и обучите ее для подписи, сгенерированной для данных до момента времени $t + 5$. Тогда вы сможете выполнить прогнозирование по этим данным с помощью методов простой линейной регрессии или машинного обучения, или даже нейронной сети глубокого обучения. Выглядит это следующим образом:

модель$(X) = Y$,

где Y можно выбирать на любом временном горизонте. Таким образом, каждая из следующих моделей будет корректной в зависимости от горизонта прогнозирования (вперед на 10 или 3 шага, как показано ниже):

- *модель*$_1(X_t)$ приближает Y_{t+10};
- *модель*$_2(X_t)$ приближает Y_{t+3}.

Рекурсивное прогнозирование к отдаленным временны*м* горизонтам

Используя рекурсивный подход к подгонке различных горизонтов, вы будете строить одну модель, но готовиться к тому, чтобы направить результаты обратно на ее вход для прогнозирования более отдаленных горизонтов. Эта идея должна казаться вам знакомой, потому что ранее мы продемонстрировали, как с ее помощью реализуются стратегии многоэтапного подхода к прогнозированию по методу ARIMA. Предположим, что модель для прогнозирования на шаг вперед уже разработана в результате обучения по принципу *модель*$(X_t) = Y_{t+1}$. Чтобы подойти к горизонту на три шага вперед нужно сделать следующее:

- *модель*$(X_t) \rightarrow$ *оценка* Y_{t+1};
- *модель*(X_t с оценкой Y_{t+1}) $\rightarrow$ оценка Y_{t+2};
- *модель* (X_t с оценкой Y_{t+1} и оценкой Y_{t+2}) $\rightarrow$ оценка Y_{t+3}.

Ожидаемая ошибка оценки Y_{t+3} обязательно будет больше, чем оценка Y_{t+1}. Насколько больше? Сложно сказать. Разобраться в этом помогло бы моделирование, как было показано ранее в этой главе.

Многозадачное обучение для временных рядов

Многозадачное обучение — это общая концепция глубокого обучения, которая может применяться с особым значением для анализа временных рядов. В более общем смысле многозадачное обучение описывает идею о том, что модель может быть построена для одновременного достижения нескольких целей или для обучения обобщению при попытке одновременно предсказать несколько разных, но связанных вещей. Некоторые считают его средством регуляризации, поощряющим обобщение модели за счет обучения связанным задачам.

В контексте временных рядов многозадачное обучение можно применять, устанавливая цели для разных временных горизонтов в контексте прогнозирования. В этом случае обучение модели будет выглядеть так:

- *модель*$(X_t) = (Y_{t+1}, Y_{t+10}, Y_{t+100})$;
- *модель*$(X_t) = (Y_{t+1}, Y_{t+2}, Y_{t+3})$.

При таком обучении целесообразно подумать об использовании функции потерь: нужно ли одинаково взвешивать все прогнозы или же отдавать предпочтение одним горизонтам прогноза по сравнению с другими?

Попытавшись построить очень дальний прогноз, для обучения своей модели можно использовать многозадачные, включая краткосрочные, горизонты, которые указывали бы характерные особенности, полезные для более отдаленного горизонта, но трудно распознаваемые непосредственно в нем из-за низкого уровня отношения "сигнал/шум" в данных. Еще один сценарий многозадачного моделирования заключается в работе с несколькими временными окнами в будущем — все в одном и том же сезоне, — но, возможно, в разные моменты времени (например, в течение нескольких лет весной или в течение нескольких недель по понедельникам). Это только один из многих способов одновременного учета сезонности и тренда.

Особенности тестирования моделей

Перечислим наиболее серьезные трудности, с которыми приходится сталкиваться при оценке правильности тестирования модели в соответствии с выдвигаемыми производственными запросами.

Упреждение

Вы могли заметить, что в этой книге большое внимание уделяется упреждению — его трудно избежать, и оно может вызвать катастрофические последствия при использовании модели в производственных целях. Постарайтесь предотвратить выпуск моделей, которые по той или иной причине не прошли

производственную проверку. Это явный признак того, что из системы не было исключено упреждение. Не допускайте этого никогда!

Структурные изменения

Выявление структурных изменений отчасти зависит от суждения, отчасти — от предметной области, а отчасти — от качества данных. Динамика данных временного ряда может со временем измениться настолько, что модель, которая подходит для описания одной части временного ряда, может не подходить для другой его части. Это одна из причин важности проведения разведочного анализа — убедиться, что одна и та же модель не обучается в рамках структурных изменений, что в заданных условиях не является ни оправданным, ни разумным.

Дополнительные источники

Christoph Bergmeir, Rob J. Hyndman, and Bonsoo Koo, "A Note on the Validity of Cross-Validation for Evaluating Autoregressive Time Series Prediction," Computational Statistics & Data Analysis 120 (2018): 70–83, `https://perma.cc/YS3J-6DMD`

Авторы статьи (включая всеми уважаемого и очень плодотворного Роба Хиндмана) обсуждают случаи, когда перекрестная проверка дает хорошие результаты даже в ходе анализа временных рядов. Статья полезна тем, что дает представление о причинах возникновения ошибок при проведении перекрестной проверки, а также о моделях, в которых можно отказаться от подгонки в пользу стандартной перекрестной проверки.

Angelo Canty, "tsboot: Bootstrapping of Time Series," n.d., `https://perma.cc/MQ77-U5HL`

Описание функции `tsboot()`, включенной в состав пакета `boot`. Будет полезным для студентов, поскольку содержит описание реализаций множества общепринятых вариантов блочной выборки как разновидности бутстрэпинга. Изучение возможностей этой функции представляет хороший способ ознакомления с различными видами бутстрэпинга временных рядов, а их знание, в свою очередь, позволяет получать удобные и надежные оценки тестовых статистик и изучить неопределенность оценок даже для небольших наборов данных.

Hans R. Kunsch, "The Jackknife and the Bootstrap for General Stationary Observations," Annals of Statistics 17, no. 3 (1989): 1217–41, `https://perma.cc/5B2T-XPBC`

В этой классической и широко цитируемой статье продемонстрирован статистический подход к применению методов "складного ножа" (реже

используемых) и бутстрэпинга к временны́м рядам, исключая из них предположение о независимости распределения точек данных. Статья освещает идею выборки из временны́х рядов не отдельных точек, а блоков данных.

Christian Kleiber, "Structural Changes in (Economic) Time Series," in Complexity and Synergetics, ed. Stefan Muller et al. (Cham, Switzerland: Springer International, 2018), `https://perma.cc/7U8N-T4RC`

В этой главе содержится описание различных канонических методов определения структурных изменений во временны́х рядах с акцентом на обработке финансовых и экономических данных, хотя рассмотренные методологии можно использовать в более широких целях. Особенность приведенного в ней материала — рассмотрение большого количества пакетов языка R, содержащих инструменты для реализации популярных методов идентификации структурных изменений, и описание способов визуализации различий в результатах, получаемых с помощью таких методов для одного и того же временно́го ряда.

Robert Stambaugh, "Predictive Regressions," Journal of Financial Economics 54 (1999): 375–421, `https://perma.cc/YD7U-RXBM`

Широко цитируемая статья, в которой изложены концепции байесовской апостериорной оценки коэффициентов регрессии и оценки величин из конечных выборок, которые существенно различаются с учетом различных предположений о базовой динамике моделей финансовых временны́х рядов. Это отличный, хотя и специфичный для предметной области пример, показывающий степень влияния исходных предположений на неопределенность оценки параметров.

ГЛАВА 12

Производительность моделей временных рядов

В литературе по машинному обучению и статистическому анализу основное внимание уделяется точности моделей. Обычно точность выступает главной целью при оценке модели, но иногда — в условиях больших наборов данных или широко развернутых моделей для обслуживания большого количества клиентских приложений — огромное значение также имеет производительность вычислений.

Наборы временных рядов становятся настолько большими, что анализ не может быть выполнен вообще или не выполняется должным образом, потому что модели становятся слишком требовательными к вычислительным ресурсам. В подобных случаях многими организациями рассматриваются следующие варианты решений.

- Расширить вычислительную ресурсную базу (очень затратно как с финансовой, так и с ресурсной точки зрения).
- Выполнить проект с заниженным качеством (вследствие недостаточно точных настроек гиперпараметров, слишком большого объема данных и т.д.).
- Отказаться от выполнения проекта.[1]

Ни один из вариантов не является оптимальным, особенно при работе над проектом с новым набором данных или новыми аналитическими инструментами. Зачастую вы даже не будете знать, являются ли плохие результаты следствием плохих данных, чрезмерной сложности задачи или нехватки вычислительных ресурсов. К счастью, существуют некоторые обходные пути, позволяющие расширить аналитические возможности в случае слишком высоких требований к точности получаемых результатов или очень больших наборов данных.

В этой главе приведены некоторые соображения о способах экономии вычислительных ресурсов при обучении или тестировании конкретных моделей. По большей части такие вопросы относятся к исследованию наборов целевых данных, изучению возможностей вычислительного оборудования, а также к анализу целей, которых нужно достичь, в первую очередь в контексте точности и производительности. Вы увидите, что проблемы, подробно описанные в этой главе, имеют самое непосредственное отношение к практическим задачам. Если

[1] Да, в реальном мире это происходит постоянно.

приведенный далее материал хотя бы частично поможет вам в решении возникающих трудностей, хочется верить, что это вдохновит вас на дальнейшие свершения. Изложенные соображения становятся актуальными только после завершения первых этапов анализа и моделирования — не стоит уделять им слишком много внимания при первом же обнаружении проблем. Однако по мере выпуска модели в производственную среду или в случае расширения исследовательского проекта вам придется обращать на них самое пристальное внимание.

Общие инструменты

Одна из проблем анализа данных временны*х* рядов состоит в том, что большинство инструментов, особенно применяемых в машинном обучении, исходно рассчитывались на общие сценарии использования, а большинство наглядных примеров приведено для перекрестных данных. Существует несколько причин, по которым такие методы машинного обучения не могут показать хорошую (максимально возможную) производительность при работе с временны*ми* рядами. У каждой из задач есть свое решение несмотря на совпадение общих идей. В этом разделе как раз и пойдет речь об общих проблемах и их возможных решениях.

Модели для перекрестных данных не обрабатывают общие данные разных выборок

Во многих случаях при передаче большого количества выборок данных временны*х* рядов в алгоритм машинного обучения можно заметить, что некоторые из них включают перекрывающиеся значения. Предположим, что вам предстоит обработать следующие данные о ежемесячных объемах продаж.

Месяц	Объем продаж, шт.
Февр. 2014	9880
Март 2014	14 423
Апр. 2014	16 720
Май 2014	17 347
Июнь 2014	22 020
Июль 2014	21 340
Авг. 2014	25 973
Сент. 2014	11 210
Окт. 2014	11 583
Нояб. 2014	12 014
Дек. 2014	11 400
Янв. 2015	11 539
Февр. 2015	10 240

Вы стремитесь делать прогнозы, сопоставляя каждую форму с ближайшей соседней кривой. При этом для исходных данных подготавливается много форм. Ниже перечислены лишь некоторые точки данных, поскольку в качестве представляющих интерес форм можно использовать только кривые длиной шесть месяцев (обратите внимание, что предварительная обработка данных для нормализации или создания дополнительных признаков, представляющих интерес — скользящее среднее или сглаженные кривые, — не выполнялась).[2]

```
11221, 9880, 14423, 16720, 17347, 22020
9880, 14423, 16720, 17347, 22020, 21340
14423, 16720, 17347, 22020, 21340, 2597
```

К сожалению, все, чего удалось добиться в ходе подготовки входных данных, — это сделать наш набор в шесть раз больше, не получив никакой дополнительной информации. С точки зрения производительности такое преобразование данных считается катастрофическим, но оно часто происходит при передаче данных в различные модули машинного обучения.

Столкнувшись с такой проблемой, рассмотрите несколько решений.

Отказ от использования перекрывающихся данных

Создайте "выделенные" точки данных так, чтобы каждый отдельный месяц относился только к одной кривой. Если сделать так для предыдущих данных, то можно получить следующий результат.

```
11221, 9880, 14423, 16720, 17347, 22020
21340, 25973, 11210, 11583, 12014, 11400
```

Заметьте, это очень простое решение, поскольку сводится к изменению формы массива, а не к повторению данных.

Использование генераторной парадигмы для перебора набора данных

Генераторная парадигма перебора набора данных, т.е. повторная выборка из одной и той же структуры данных, особенно легко реализуется на языке Python и может быть выполнена на языке R, а также на некоторых других языках. Если представить, что исходные данные хранятся в одномерном массиве NumPy, то реализация такой парадигмы может выглядеть, как в следующем коде (обратите внимание, что должна использоваться структура данных машинного обучения или алгоритм, поддерживающий работу с генераторами).

```
## Python
>>> def array_to_ts(arr):
>>>     idx = 0
```

[2] При использовании моделей машинного и глубокого обучения, обсуждавшихся в предыдущих главах, обычно извлекается много выборок временны*х* рядов из одного большого временно*г*о ряда.

```
>>>   while idx + 6 <= arr.shape[0]:
>>>     yield arr[idx:(idx+6)]
```

Заметьте, что код моделирования данных не должен вызывать переструктурирования набора данных — это крайне важное требование как для этапа обучения, так и производственного использования. При обучении это позволит разместить больше примеров обучения в памяти, а на производственном уровне — построить больше прогнозов (или классификаций) на меньшим количестве обучающих ресурсов для перекрывающихся данных. Если вам приходится часто составлять прогнозы для одного и того же сценария использования, то, вероятнее всего, вы работаете с перекрывающимися данными, а рассмотренные выше проблема и способы ее решения окажутся как нельзя кстати.

Задержка между измерением данных и прогнозом в моделях без предварительной обработки

Обычно модели машинного обучения не предусматривают предварительной подготовки данных и не проводят предварительные расчеты для их части до получения всех данных. Однако в случае временны*х* рядов это очень распространенный сценарий.

Используя модель в чувствительном к временны*м* задержкам приложении, например в программном обеспечении постановки медицинских диагнозов, оценки местоположения транспортных средств или прогнозирования цены на акции, вы можете обнаружить, что задержка вычисления прогноза до момента, когда станут доступны все данные, слишком велика. В таком случае постарайтесь понять, можно ли заранее выполнить часть вычислений, предусмотренных моделью. Вот несколько примеров того, как это можно сделать.

- Используя рекуррентную нейронную сеть с несколькими каналами информации в течение ста различных временны*х* шагов, можно предварительно вычислить/развернуть нейронную сеть для первых 99 временны*х* шагов. Затем, когда, наконец, появится последняя точка данных, нужно будет сделать только последний подход к вычислению произведения матриц (и операций, заданных в функции активации), а не сто. Теоретически это может уменьшить время отклика в сто раз.
- Работая с моделью AR(5), вы можете предварительно вычислить сумму, за исключением последнего члена. Напомним, что процесс AR(5) описывается следующим образом: если вы собираетесь сделать прогноз, то понадобятся известные значения y_{t-4}, y_{t-3}, y_{t-2} и y_{t-1}, а это означает, что, зная y_t, можно заранее рассчитать все, кроме члена $ph\, i_0 \tilde{A} y_t$.

$$y_{t+1} = ph\, i_4 \cdot y_{t-4} + ph\, i_3 \cdot y_{t-3} + ph\, i_2 \cdot y_{t-2} + ph\, i_1 \cdot y_{t-1} + ph\, i_0 \cdot y_t.$$

- Если вы используете модель кластеризации для нахождения ближайших соседей по методу суммирования признаков временного ряда (среднего значения, стандартного отклонения, максимума, минимума и т.д.), то можете вычислить эти функции с помощью временного ряда с одной точкой данных и запустить модель с этим временным рядом, чтобы определить несколько ближайших соседей. Как только появится окончательное значение, нужно будет обновить признаки и перезапустить анализ, используя только ближайших соседей, найденных в первом раунде анализа. В конечном счете такой подход требует несколько больше вычислительных ресурсов, но приводит к существенному уменьшению задержки между проведением окончательных измерений и получением прогноза.

Во многих случаях задержка модели может быть больше, чем время отклика сети или другие временные факторы, поэтому к предварительному вычислению стоит прибегать только в случаях, когда время обратной связи чрезвычайно важно и вы уверены, что вычисление модели существенно влияет на время между получением приложением всей необходимой информации и выводом конечного прогноза.

Форматы хранения данных: преимущества и недостатки

Одним из самых недооцениваемых критериев, определяющих производительность модели временных рядов в случае как обучения, так и производственного использования, выступает способ хранения данных. Укажем только наиболее распространенные ошибки.

- *Хранение данных в формате с построчным вводом, даже если временные ряды формируются путем обхода столбцов.* Приводит к тому, что точки данных, смежные во времени, оказываются несмежными в памяти.
- *Хранение данных и непосредственный анализ необработанных данных.* Крайне важно передавать в модель предварительно обработанные данные и данные с уменьшенной выборкой.

Обсудим детально факторы хранения данных, позволяющие ускорить обучение и вычисление модели.

Хранение данных в двоичном формате

Очень заманчиво хранить данные в текстовом файле, разделенном запятыми, например, формата CSV. Стоит заметить, что данные предоставляются в таком формате очень часто, что делает его выбор вполне естественным решением.

Такой формат файлов оказывается удобным для визуального анализа и позволяет проверять данные при конвейерном выводе. Наконец, такие данные легко транслируются на разные платформы.[3]

Однако компьютеру читать текстовые файлы не так легко, как кажется. Работая с наборами данных, которые настолько велики, что не помещаются в памяти во время обучения, вы будете испытывать дополнительные трудности с вводом-выводом и связанной с ними обработкой данных, зависящей от формата файла. Храня данные в двоичном формате, вы можете существенно сократить задержку, связанную с выполнением операций ввода-вывода, несколькими способами.

- Если данные представлены в двоичном формате, то пакет обработки данных гарантированно их понимает. Нет необходимости специальным образом считывать файлы формата CSV и преобразовывать их в фреймы данных. При вводе данных фрейм данных создается автоматически.
- Если данные представлены в двоичном виде, они могут быть сжаты гораздо сильнее, чем в CSV или другом текстовом формате. Это означает, что операции ввода-вывода выполняются быстрее, поскольку файл требует меньше физической памяти для размещения его содержимого.

Двоичные (бинарные) форматы хранения данных хорошо поддерживаются как языком R, так и языком Python. В языке R обратите внимание на функции `save()` и `load()` для `data.table`. В языке Python используйте модуль `picle` и заметьте, что библиотеки Pandas (`pd.DataFrame.load()` и `pd.DataFrame.save()`) и NumPy (`np.load()` и `np.save()`) выступают оберточными пакетами для инструментов преобразования объектов в бинарный вид, которые применяются к определенному набору объектов.

Предварительная обработка данных для облегчения прохода по ним

Рассмотренная далее рекомендация, в первую очередь, относится к разделу "Модели для перекрестных данных не обрабатывают общие данные разных выборок". В этом случае также задумайтесь о предварительной обработке данных и убедитесь, что вы делаете это правильно с использованием скользящего окна для создания нескольких тестовых выборок.

В качестве примера рассмотрим задачу нормализации или вычисления скользящего среднего как операцию по предварительной обработке данных. Если вы будете выполнять ее для каждого временного окна, то это приведет к повышению

[3] Впрочем, не стоит забывать о проблеме кодировки Unicode, по-разному интерпретируемой платформами и устройствами. Нельзя наверняка утверждать, что данные становятся понятными только потому, что сохранены в текстовых файлах.

точности модели (хотя, согласно моему опыту, выигрыш оказывается незначительным). Однако у такого подхода есть ряд недостатков.

- Требует больше вычислительных ресурсов для многократного предварительного вычисления признаков на перекрывающихся данных только для того, чтобы получить очень схожие числа.
- Нужно хранить многочисленные снимки перекрывающихся данных, незначительно различающихся методом предварительной обработки.
- Невозможно оптимально использовать скользящее окно.

Изменение метода анализа для повышения производительности

Многие из нас слишком привыкли к определенному набору аналитических инструментов, а также к сопутствующему набору программного обеспечения и практическим правилам обучения моделей. Мы также склонны оценивать потребности только один раз и не пересматривать их, выясняя вычислительную стоимость возможных вариантов модели.

Данные временны*х* рядов, часто используемые для быстрого прогноза, порождают особенно острую потребность в моделях, которые можно быстро адаптировать и передать в производство. Модели должны быть хорошо адаптируемыми, чтобы их можно было обновлять по мере поступления новых данных, и они должны работать быстро, чтобы у заказчиков прогнозов для таких моделей оставалось как можно больше времени для совершения действий, продиктованных итогами прогнозирования. По этой же причине иногда может потребоваться изменить свои ожидания (пересмотрев результаты сопутствующего анализа), чтобы ускорить вычислительные процессы в задачах анализа и прогнозирования.

Использовать все данные необязательно

Один из важнейших принципов упрощения методик анализа данных основан на понимании того, что не все значения во временно*м* ряду одинаково важны. Как правило, более отдаленные данные имеют меньший вес. При этом данные в "исключительные" времена оказываются менее важными для моделей, разрабатываемых для данных рядовых времен.

Есть много способов уменьшить количество данных, используемых для обучения модели. Хотя многие из вариантов уже обсуждались в этой книге, вам будет полезно рассмотреть их, особенно в контексте влияния на производительность модели.

Понижающая дискретизация

Часто бывает так, что при прогнозировании используются данные, которые реже включаются в одно и то же окно просмотра назад. Это один из способов уменьшения количества данных за счет мультипликативного коэффициента. Обратите внимание, что в зависимости от используемого аналитического подхода в вашем распоряжении находятся некоторые другие возможности, например понижающая дискретизация, уровень которой зависит от глубины просмотра данных.

Обучение только по последним данным

Хотя машинное обучение любит много данных, существуют модели временных рядов, в которых статистические или даже глубокие методы обучения показывают лучшие результаты, если сосредоточить их обучение на последних, а не на всех имеющихся данных. Это также помогает уменьшить объем входных данных, исключая из них малоинформативные для выбранной модели значения.

Уменьшение окна просмотра, используемого для прогнозирования

Во многих моделях временных рядов производительность повышается, хотя и незначительно, по мере изучения данных все дальше и дальше в прошлом. Вам нужно определить баланс между точностью и производительностью. Может случиться так, что в память загружается намного больше данных, чем на самом деле необходимо для получения заданного уровня производительности.

Сложные модели не всегда лучшие

При выборе аналитической модели всегда интересно и поучительно самостоятельно испытать последние технологические разработки. Однако остается открытым вопрос, компенсирует ли более изощренная модель дополнительные вычислительные затраты?

Почти все вычислительные достижения в области машинного обучения за последние годы связаны с повышением вычислительной мощности для решения все более и более усложняющихся задач. Конечно, во многих задачах, подобных распознаванию изображений, в которых правильного результата можно добиться почти со стопроцентной точностью, такой подход вполне оправдан.

С другой стороны, в таких задачах, как прогнозирование временных рядов, точность выполнения которых обусловливается строгими физическими или математическими ограничениями, не лишне убедиться, что выбор более сложной модели далеко не всегда означает автоматическое улучшение производительности без учета преимуществ/недостатков. Постарайтесь оценить, будет ли оправданным повышение точности за счет увеличения времени вычисления прогноза и обучения, а также привлечения дополнительных вычислительных ресурсов.

Может случиться так, что менее ресурсоемкий метод с немного меньшей точностью окажется гораздо более выгодным, чем причудливая модель, которая всего лишь незначительно лучше простой версии.

Если вы практикующий специалист по анализу данных, то такой компромисс между сложностью/точностью и увеличением времени/вычислительных ресурсов должен стать предметом для анализа, который можно рассматривать как дополнительный гиперпараметр настройки модели. Ваша задача — добиться компромиссных решений, а не предположить, что об этом позаботится инженер, снабжающий вас данными. Специалисты на верхних или нижних уровнях конвейера данных не смогут выбрать модель вместо вас, поэтому вы должны определиться с выбором, взвесив все "за" и "против" как с инженерной, так и с научной точек зрения.

Кратко об альтернативных инструментах

Ознакомившись с предыдущими вариантами, рассмотрим возможность изменения базовой части кода, в частности отказавшись от медленных языков программирования, таких как Python и R. Есть несколько способов сделать это.

- Используйте языки C++ и Java. Даже если вы не работали с ними ранее, использование даже простых их инструментов поможет ускорить низкопроизводительные части конвейера настолько, что сделает сложные для реализации задачи быстрорешаемыми. В частности, язык C++ значительно эволюционировал в том, что касается удобства использования и возможностей стандартных библиотек по обработке данных. Синтаксис STL и C++ 17 теперь предлагает множество опций, вполне сопоставимых с имеющимися в языке Python, для работы с наборами временны*х* рядов в различных структурах данных. Даже если вы испытываете давнюю неприязнь к C++ и Java, постарайтесь преодолеть ее и вернуться к их использованию.[4]
- В Python представлено несколько пакетов, с помощью которых код Python можно компилировать в код C или C++, повышая время его выполнения. Это особенно полезная практика для повторяющихся фрагментов кода со множеством циклов `for`, которые медленно рассчитываются в языке Python и показывают намного большую производительность в языках C и C++ без необходимости изощренного перепроектирования — цели можно достичь, реализуя тот же код на более быстром языке. Ускорить "медленные" фрагменты кода на языке Python помогут модули `Numba` и `Cython`.
- Аналогично в языке R подобная функциональная особенность реализуется с помощью пакета `Rcpp`.

[4] Для этих языков характерна крутая кривая обучения, но как только вы разберетесь в базовой инфраструктуре средств компиляции, их знание станет серьезным подспорьем в анализе временны*х* рядов.

Дополнительные источники

- Оценка производительности моделей

  Anthony Bagnall et al., "The Great Time Series Classification Bake Off: An Experimental Evaluation of Recently Proposed Algorithms," Data Mining and Knowledge Discovery 31, no. 3 (2017): 606–60, `https://perma.cc/T76B-M635`

  В этой статье описано множество экспериментов по оценке производительности современных методологий классификации временных рядов, основанных на тестировании их на широком спектре общедоступных наборов данных. Вычислительная сложность наборов данных в конечном итоге варьируется чуть больше, чем реальная производительность опробованных методов. Как подчеркивают авторы, выбор модели, оптимальной для конкретного набора данных без перебора всех возможных вариантов, считается искусством и остается открытой проблемой. С точки зрения вычислительных ресурсов урок, который следует здесь усвоить, состоит в том, что вычислительная сложность должна в значительной степени учитывать методологические решения. Если у вас нет убедительных причин использования сложного и ресурсоемкого алгоритма, остановитесь на более простом варианте.

- Построение упрощенных моделей

  Yoon Kim and Alexander M. Rush, "Sequence Level Knowledge Distillation," in Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, ed. Jian Su, Kevin Duh, and Xavier Carreras (Austin, TX: Association for Computational Linguistics, 2016), 1317–27, `https://perma.cc/V4U6-EJNU`

  Статья следует общей концепции "дистилляции" в последовательном обучении применимо к задачам машинного перевода. Концепция дистилляции находит применение в широком спектре предметных областей. Идея состоит в том, что сначала разрабатывается сложная модель, которая обучается на основе исходных данных, а затем более простая модель обучается на результатах сложной модели. Использование результатов сложной модели вместо исходных данных уменьшает количество шума и упрощает задачу обучения. Хотя такая методика не сокращает время обучения, она позволяет построить модель, которая работает быстрее и требует меньше ресурсов при запуске в производство.

ГЛАВА 13

Медицинские приложения

В этой главе мы рассмотрим принципы анализа временны*х* рядов в контексте отрасли здравоохранения на примере двух тематических исследований: прогнозирования уровня заболеваемости гриппом и уровня глюкозы в крови. Оба приложения относятся к чрезвычайно важным проблемам здравоохранения. Кроме того, в обоих случаях задачи считаются нерешенными и активно исследуются в академических научных кругах и специалистами по здравоохранению.

Прогнозирование заболеваемости гриппом

Прогнозирование уровня заболеваемости гриппом по неделям в заданном географическом регионе является давней и острой проблемой. Специалисты по инфекционным болезням и сотрудники служб национальной безопасности согласны с тем, что инфекционные заболевания представляют значительный риск для благосостояния людей (`https://perma.cc/ZDA8-AKX6`). Это особенно актуально для гриппа, который поражает уязвимых людей во всем мире, ежегодно забирая сотни жизней, в основном очень молодых и очень старых людей. С точки зрения здравоохранения и национальной безопасности крайне важно разработать точные модели описания распространения заболеваемости гриппом в определенном сезоне. Модели прогнозирования распространения гриппа будут полезны как для изучения вируса, так и для помощи исследователям в построении общих теорий географического распространения инфекционных заболеваний.

Изучение заболеваемости гриппом на примере мегаполиса

Рассмотрим наборы данных, представленные в еженедельных отчетах по заболеваемости гриппом в различных административных регионах Франции за период с 2004 по 2013 год. Постараемся спрогнозировать уровень заболеваемости гриппом в Иль-де-Франс, столичном регионе Парижа. Изучаемые данные доступны для загрузки с сайта Kaggle[1] (`https://perma.cc/W9VQ-UUJC`).

[1] Хотя это не свободно распространяемый набор данных, доступ к нему можно получить, зарегистрировавшись для участия в конкурсе.

Исследование и частичная предварительная очистка данных

Приступим к ознакомлению с необработанными данными, сначала рассмотрев их в табличной форме.

```
## R
> flu = fread("train.csv")
> flu[, flu.rate := as.numeric(TauxGrippe)]
> head(flu)
     Id   week  region_code   region_name TauxGrippe flu.rate
1: 3235 201352           42   ALSACE               7        7
2: 3236 201352           72   AQUITAINE            0        0
3: 3237 201352           83   AUVERGNE            88       88
4: 3238 201352           25   BASSE-NORMANDIE     15       15
5: 3239 201352           26   BOURGOGNE            0        0
6: 3240 201352           53   BRETAGNE            67       67
```

Кроме того, выполним базовую проверку качества данных, например проведем поиск значений NA в представляющих интерес переменных. На данный момент совершенно не важно, как они были получены, но их обязательно нужно учесть.

```
## R
> nrow(flu[is.na(flu.rate)]) / nrow(flu)
[1] 0.01393243
> unique(flu[is.na(flu.rate)]$region_name)
[1] "HAUTE-NORMANDIE"        "NORD-PAS-DE-CALAIS"      "PICARDIE"
[4] "LIMOUSIN"               "FRANCHE-COMTE"           "CENTRE"
[7] "AUVERGNE"               "BASSE-NORMANDIE"         "BOURGOGNE"
[10] "CHAMPAGNE-ARDENNE"     "LANGUEDOC-ROUSSILLON"    "PAYS-DE-LA-LOIRE"
[13] "CORSE"
```

Общее количество точек данных со значениями NA не очень велико. Кроме того, интересующий нас регион, Иль-де-Франс, не входит в список регионов со значениями NA.

Проведем некоторую очистку данных, разделяя недельную и годовую части столбца временны*х* меток (на данный момент значения времени представлены символами, а не числами).

```
## R
> flu[, year := as.numeric(substr(week, 1, 4))]
> flu[, wk := as.numeric(substr(week, 5, 6))]
> ## Со стилистической точки зрения иметь два столбца для недели нежелательно
```

Добавим столбец класса Date для лучшего отображения временно*й* оси.

```
## R
> flu[, date:= as.Date(paste0(as.character(flu$week), "1"), "%Y%U%u")]
```

Эта строка кода немного сложнее. Чтобы преобразовать комбинации месяца и недели в даты, добавляем компонент, указывающий день. Для этого

используется функция paste0(), которая определяет каждую дату как первый день недели, вставляя "1" в строку, которая обозначает год и неделю года (одну из 52 недель в году — подробнее об этом мы поговорим чуть ниже).[2] Обратите внимание на обозначения %U и %u в строке указания формата: они устанавливают время в соответствии с неделей года и днем недели, что является довольно необычным для формата времени.[3]

Подберем данные, относящиеся непосредственно к Парижу, и отсортируем их по дате.[4]

```
## R
## Сосредоточимся на Париже
> paris.flu = flu[region_name == "ILE-DE-FRANCE"]
> paris.flu = paris.flu[order(date, decreasing = FALSE)]

> paris.flu[, .(week, date, flu.rate)]
        week         date   flu.rate
   1: 200401   2004-01-05         66
   2: 200402   2004-01-12         74
   3: 200403   2004-01-19         88
   4: 200404   2004-01-26         26
   5: 200405   2004-02-02         17
---
518:  201350   2013-12-16         13
519:  201351   2013-12-23         49
520:  201352   2013-12-30         24
521:  200953         <NA>        145
522:  200453         <NA>         56
```

Количество строк должно вас удивить. Если в году 52 недели, и мы располагаем данными за 10 лет, то почему в наборе всего 522 строки? Стоило бы ожидать получения 52 недели · 10 лет = 520 строк. Точно так откуда взялись две точки со значениями NA? Эти факты можно объяснить, внимательно изучив исходные данные. Судя по всему, 53-я неделя была как в 2004, так и в 2009 году. Через каждые несколько лет год состоит из 53 недель, так что 52 — это не ошибка, а скорее особенность системы григорианского календаря (https://perma.cc/4ETJ-88QR).

[2] Правильный выбор первого дня недели (с вариантами от 1 до 7) зависит от того, с какой даты началась регистрация уровня заболеваемости гриппом, но по предоставленным данным это определить невозможно. Для простоты остановимся на первом дне.

[3] Соответствующая строка форматирования для недели года и дня недели может зависеть от используемой операционной системы. Представленное решение выполнялось на macOS, в то время как в Linux нужно применять немного другое форматирование.

[4] Если бы мы работали с большим набором данных, то должны были сделать это в качестве первого шага, чтобы избежать вычислительно затратных операций, таких как преобразование строк в объекты Date для нерелевантных данных.

Теперь удостоверимся, что данные охватывают полную и регулярную выборку диапазона дат, предварительно убедившись, что каждый год имеет одинаковое количество точек данных.[5]

```
## R
> paris.flu[, .N, year]
      year  N
  1:  2004 53
  2:  2005 52
...
   9:  2012  52
 10: 2013  52
> paris.flu[, .N, wk]
     wk   N
  1:  1  10
  2:  2  10
  3:  3  10
...
 51: 51  10
 52: 52  10
 53: 53   2
     wk   N
```

Как видите, данные соответствуют ожиданиям: каждый год (кроме двух только что обсуждаемых) состоит из 52 недель, а каждая подпись недели включает 10 точек данных — по одной на каждый год (кроме недели 53).

Рассмотрев временны*е* метки данных, проверим фактические значения временно*г*о ряда (до этого мы имели дело только с временно*й* индексацией). Проявляются ли в данных тенденция и сезонность? Давайте выясним это (рис. 13.1).

```
## R
> paris.flu[, plot(date, flu.rate,
>                  type = "l", xlab = "Date",
>                  ylab = "Flu rate")]
```

Из этого простого графика видно, что данные демонстрируют сезонные изменения (скорее всего, вы об этом догадывались по собственному опыту). График указывает на сильную сезонную составляющую, но показывает наличия временно*г*о дрейфа сезонности.

Сезонное поведение осложняет описание 53-й недели. Для получения сезонной модели нужно выразить такое поведение через количество недель года, а сами сезоны не могут иметь переменные размеры (это то, что отличает сезоны от цикла, как описано в главе 3). Несмотря на то что к решению проблемы 53-й

[5]Это не единственная необходимая проверка, обеспечивающая регулярность и полноту выборки данных. С другими критериями вам придется ознакомиться самостоятельно.

недели можно было подойти более творчески, ограничимся простейшим вариантом сокрытия ее данных.

```
## R
> paris.flu <- paris.flu[week != 53]
```

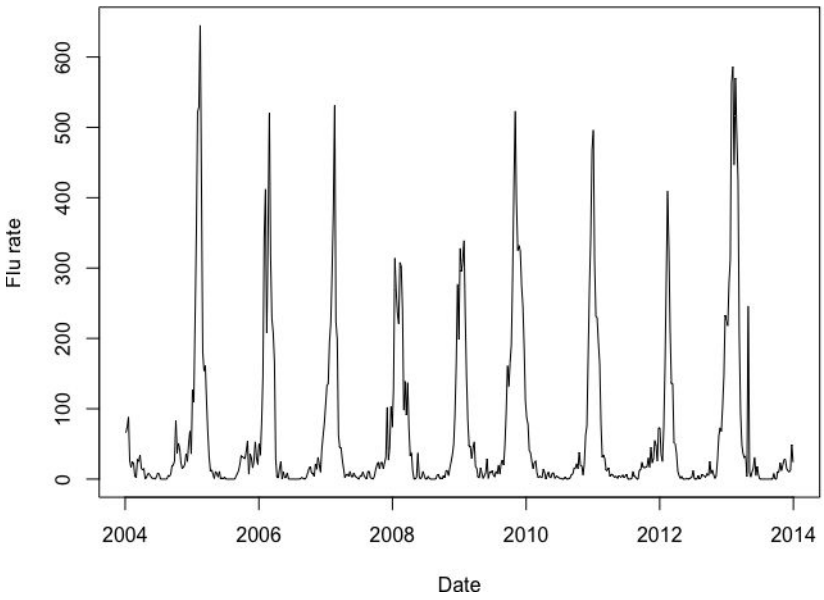


Рис. 13.1. *Построив график данных о заболеваемости гриппом в Париже, можем наблюдать его сезонный характер*

Будет ли существенной проблемой удаление точек данных, зависит от набора данных и вопроса, на который требуется получить ответ. Самостоятельно рассмотрите другие возможности по подбору данных, включающих 53-ю неделю. Есть множество способов сделать это. Один из них — объединение данных 53- и 52-й недель с последующим их усреднением. Другой способ — использовать модель, которая учитывает циклическое поведение без привязки к длине годичного цикла. Третий вариант заключается в приспособлении модели машинного обучения к такому подписыванию данных, при котором сезонность модели рассматривается в качестве входного признака.

Обучение сезонной модели ARIMA

Учитывая сильно выраженную сезонность данных, давайте попробуем описать их с помощью сезонной модели ARIMA. Ввиду еженедельного сбора периодичность данных равна 52. Нам нужна как можно более экономная модель, т.е. модель без слишком большого количества параметров, потому что при 520 точках данных временной ряд нельзя считать особенно длинным.

Наш временной ряд является хорошим примером того, что может пойти не так, если слишком сильно полагаться на автоматические настройки. Например, можно было сначала рассмотреть вопрос о вычислении разностей значений, после чего стоило построить графики автокорреляции для уровня заболеваемости гриппом и автокорреляции для разностей уровней заболеваемости гриппом, которые показаны на рис. 13.2.

```
## R
> acf(paris.flu$flu.rate, )
> acf(diff(paris.flu$flu.rate, 52))
```

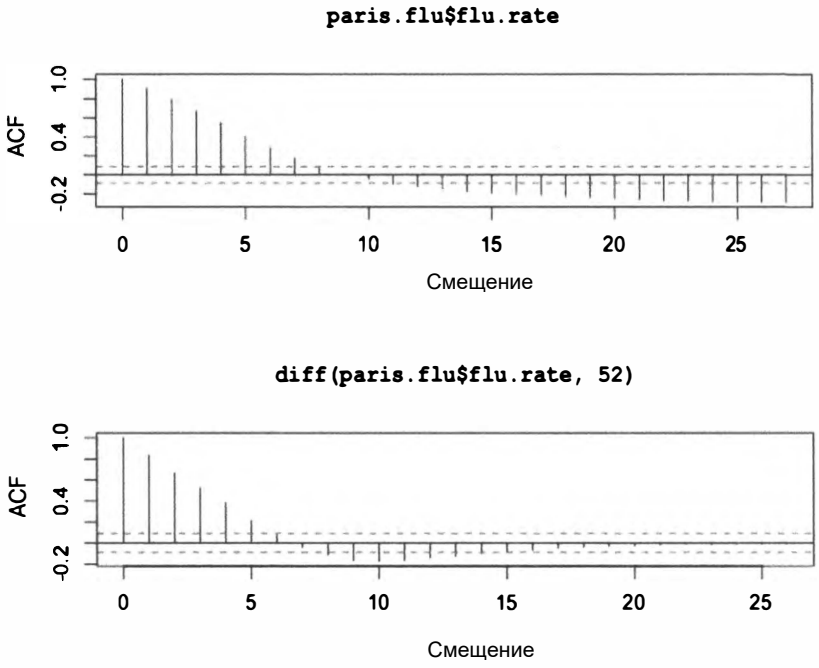


Рис. 13.2. *Графики автокорреляционной функции для уровня заболеваемости гриппом в Париже и для разностей уровней заболеваемости гриппом*

Проявив небрежность, можно было бы поздравить себя с решением проблемы стационарности для исходного временного ряда сразу же после визуализации разностей его значений. Но на самом деле они вообще ни о чем не говорят, ведь мы рассматриваем еженедельные данные, для которых наблюдается сильная сезонность. Почему же тогда это никак не проявляется на графике автокорреляции? В функции `acf()` мы воспользовались параметрами по умолчанию, но они не задают достаточного смещения, чтобы показать влияние сезонной компоненты данных, которые начинаются с временной точки 52 (один год). Давайте вызовем функцию `acf()` повторно, правильно задав соответствующее окно (рис. 13.3).

```
## R
> acf(paris.flu$flu.rate, lag.max = 104)
> acf(diff(paris.flu$flu.rate, 52), lag.max = 104)
```

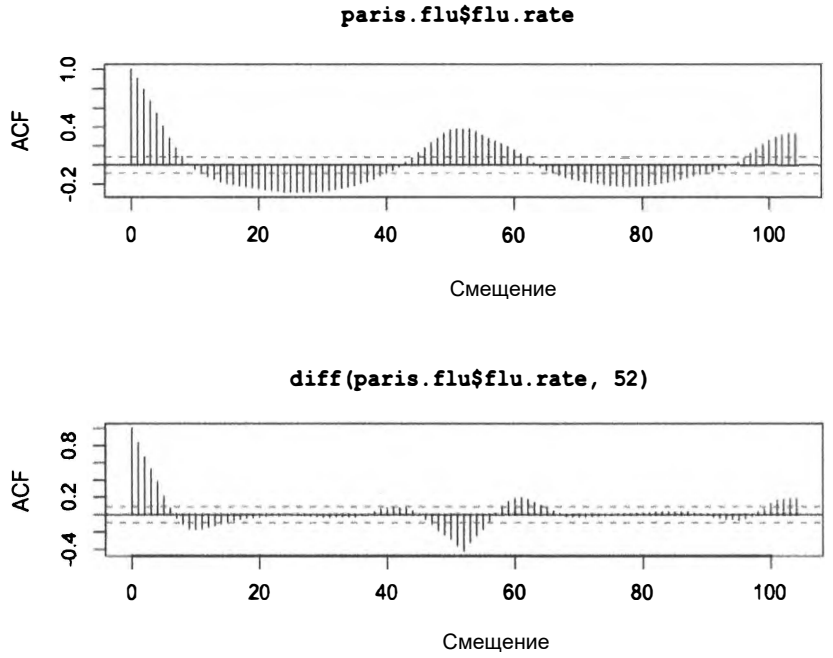


Рис. 13.3. *Графики автокорреляционной функции для уровня заболеваемости гриппом в Париже и для разностей значений уровня заболеваемости гриппом*

Результат автокорреляции временных рядов оказывается более реалистичным. Как видите, существуют значительные автокорреляции при различных смещениях, и этому у людей, живущих в климате с четырьмя сезонами, конечно же, есть свое объяснение (по крайней мере, учитывая личный опыт). Показатели заболеваемости гриппом будут демонстрировать сильную корреляцию с соседними неделями, т.е. близкими ко времени их измерения.

Учитывая сезонность, уровень заболеваемости гриппом также будет сильно коррелировать с данными за периоды времени, отстающие примерно на 52 или около 104 шагов, предопределяющих годичную сезонность. Но показатели заболеваемости гриппом также имеют довольно сильную связь с периодами времени, демонстрирующими отставание для промежуточных значений, таких как полгода (26 недель), поскольку подобные смещения также связаны с сезонными различиями и вполне предсказуемыми погодными изменениями. Например, мы знаем, что через каждые полгода значение уровня заболеваемости, скорее всего, сильно изменится. Если в какой-то момент оно высокое, то через полгода

станет низким, и наоборот, опять же, из-за сезонности. Такое поведение показано на верхнем графике на рис. 13.3.

Перейдем к рассмотрению ряда разностей, показанного на нижнем графике рис. 13.3. Легко заметить, что в значительной части автокорреляция временных рядов была уменьшена. Тем не менее остается некоторая автокорреляция в диапазоне значений, не только в 52 или 104 неделе (один или два года), но и в некоторых промежуточных значениях.

Хотя всегда возникает соблазн продолжать рассчитывать последовательности разностей, не забывайте, что реальные данные никогда не будут идеально соответствовать модели SARIMA. Поэтому вместо этого поищем более разумный способ моделирования данных. Попробуем пересмотреть разности в зависимости от времени года или использовать другой подход, вычислив разности для линейного времени. В графическом виде каждая из возможностей проиллюстрирована на рис. 13.4.

```
## R
> plot(diff(diff(paris.flu$flu.rate, 52), 52))
> plot(diff(diff(paris.flu$flu.rate, 52), 1))
```

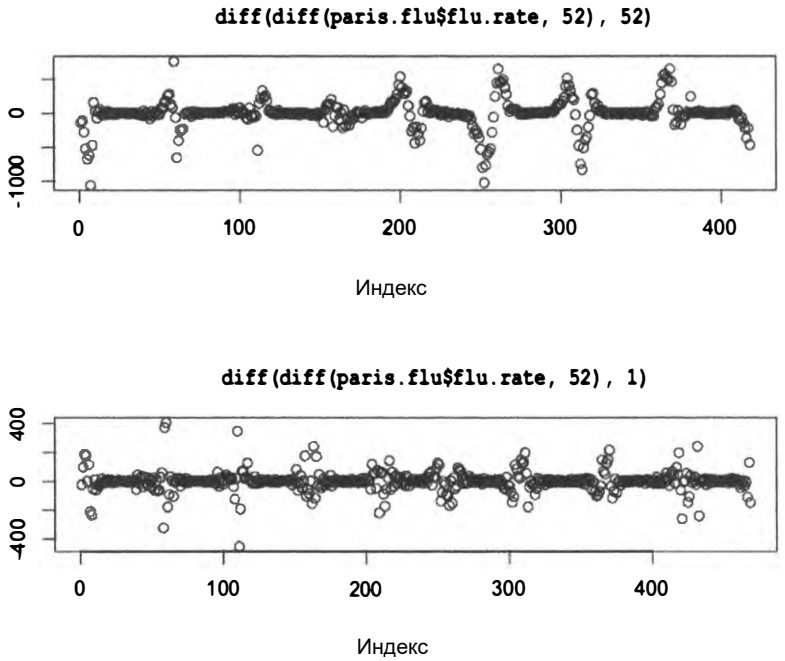


Рис. 13.4. *Графики двух вариантов дифференцирования наших временных рядов, дающие представление о сезонном характере данных*

Хотя ни один из результатов не является идеальным, наиболее удовлетворительным оказывается последний вариант — стандартная разность первого порядка для разностей исходных сезонных данных.

Решение о подгонке или подборе параметров является субъективным выбором исследователя, как и вопрос о применении тестов. В нашем случае основой анализа послужила сезонность, которую мы четко наблюдали. Тем не менее не нужно допускать чрезмерного усложнения или запутывания модели. Следовательно, в модели SARIMA(p, d, q), (P, D, Q) будем использовать параметры $d = 1$ и $D = 1$, а затем подбирать параметры AR и MA, как для настроек p и q стандартной модели ARIMA. Результаты соответствующих вычислений визуализируются с помощью следующего кода (рис. 13.5).

```
## R
> par(mfrow = c(2, 1))
> acf  (diff(diff(paris.flu$flu.rate, 52), 52), lag.max = 104)
> pacf(diff(diff(paris.flu$flu.rate, 52), 1), lag.max = 104)
```

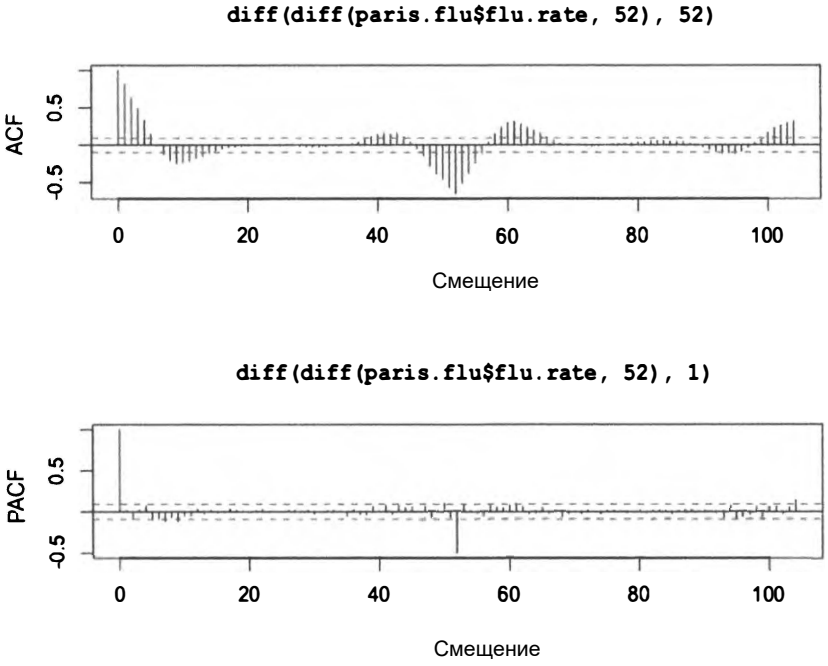


Рис. 13.5. *График функции частичной автокорреляции для выбранного разностного ряда*

Мы располагаем ограниченным набором данных, поэтому преимущество следует отдать более простой модели. Модель PACF предполагает приемлемость модели AR(2), поэтому для более точного моделирования данных будем применять модель SARIMA(2, 1, 0), (0, 1, 0).

Желательно разобраться в том, как работает такая модель, чтобы понять, как обучать ее по новым данным, когда они станут доступными. Именно так работает большинство моделей, построенных для реальных систем. Иначе говоря, как

бы вела себя модель, описывающая распространение гриппа, которое началось несколько лет назад, каждую неделю обрабатывая только текущие (доступные на тот момент) данные? Чтобы ответить на этот вопрос, нужно обучить и оценить модель следующим образом.

```
## R
> ## Обучение модели ARIMA
> ## Оценка на две недели вперед
> first.fit.size <- 104
> h              <- 2
> n              <- nrow(paris.flu) - h - first.fit.size
>
> ## Стандартные параметры обучения
> ## Сопутствующие настройки, например коэффициентов
> first.fit <- arima(paris.flu$flu.rate[1:first.fit.size],
>                    order = c(2,1,0), seasonal =
>                    list(order = c(0,1,0), period = 52))
> first.order <- arimaorder(first.fit)
>
> ## Выделение памяти для хранения прогнозов и коэффициентов
> fit.preds <- array(0, dim = c(n, h))
> fit.coefs <- array(0, dim = c(n, length(first.fit$coef)))
>
> ## После первого обучения окно расширяется на одну неделю за шаг.
> ## Модель переобучается с сохранением коэффициентов и прогнозов.
> ## Внимание! Этот цикл выполняется очень долго
> for (i in (first.fit.size + 1):(nrow(paris.flu) - h)) {
> ## Построение прогноза для постоянно расширяющегося окна
> data.to.fit = paris.flu[1:i]
> fit = arima(data.to.fit$flu.rate, order = first.order[1:3],
>             seasonal = first.order[4:6])
> fit.preds[i - first.fit.size, ] <- forecast(fit, h = 2)$mean
> fit.coefs[i - first.fit.size, ] <- fit$coef
> }
```

Визуализируем результаты, полученные для скользящего окна (рис. 13.6).

```
## R
> ylim <- range(paris.flu$flu.rate[300:400],
>               fit.preds[, h][(300-h):(400-h)])
> par(mfrow = c(1, 1))
> plot(paris.flu$date[300:400], paris.flu$flu.rate[300:400],
>      ylim = ylim, cex = 0.8,
> xlab = "Date", ylab = "Flu rate")
> lines(paris.flu$date[300:400], fit.preds[, h][(300-h):(400-h)],
>       col = 2, type = "l",
>       lty = 2, lwd = 2)
```

На графике представлен полезный прогноз, но он также иллюстрирует некоторые ограничения модели. Он недостаточно реалистичен, что подтверждается

тем фактом, что модель прогнозирует отрицательные динамику заболеваемости гриппом. А все потому, что в модели ARIMA нет ограничений на установку уровня заболеваемости гриппом в отрицательные значения. Следовательно, такие ограничения нужно определять в ходе предварительной обработки данных — до начала обучения модели.

Кроме того, модель оказывается более чувствительной к выбросам, чем ожидалось. Начало 2013 года является временной точкой, когда модель в несколько раз переоценивает уровень заболеваемости гриппом. Когда речь идет о распределении ресурсов, имеющих решающее значение для борьбы с инфекционными заболеваниями, такая оценка абсолютно неприемлема.

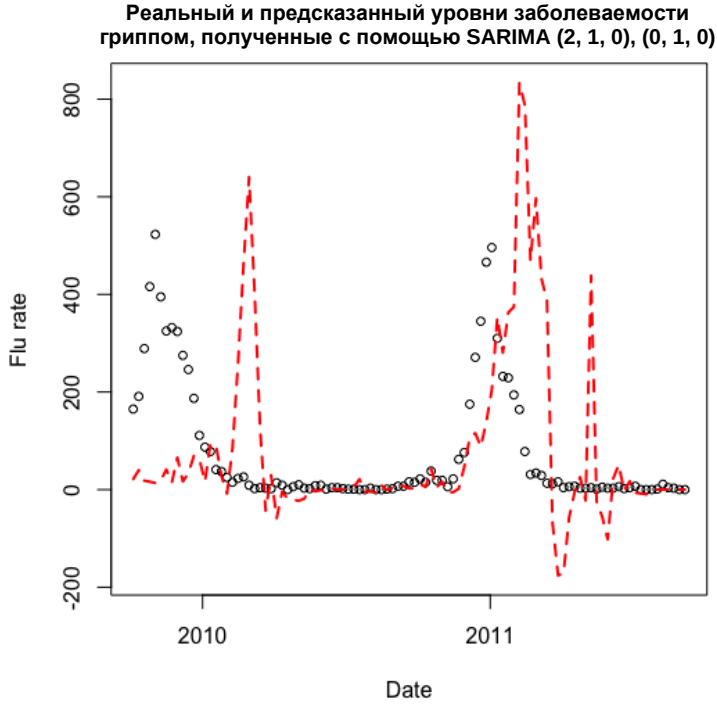


Рис. 13.6. *Сравнение реальных уровней заболеваемости гриппом (точки) с прогнозами, полученными по модели SARIMA (пунктирная кривая). Прогнозы, составленные этой простой моделью, могут использоваться при планировании противоэпидемических мероприятий*

Наконец, наша модель показывает более экстремальные значения на пиках, чем те, которые были когда-либо измерены в указанном году. Следствием такого прогноза может быть неправильное перераспределение ресурсов без учета фактических потребностей, что не приведет к хорошим результатам при борьбе с заболеванием, особенно если объем ресурсов сильно ограничен. Эти вопросы

обязательно должны учитываться при разработке модели, которая основывается как на реальных ресурсных ограничениях, так и на теоретическом анализе данных.

Теперь, когда мы рассмотрели соответствие базовой модели ARIMA этой проблеме, давайте рассмотрим другие возможности моделирования процесса.

Альтернативная модель ARIMA: экзогенные гармонические регрессоры вместо сезонности

Учитывая невысокую точность модели SARIMA, которая обсуждалась в предыдущем разделе, в нее нужно внести ряд изменений. Ниже рассматриваются две модификации, которые никак не связаны одна с другой и могут применяться независимо.[6]

Во-первых, в нашу модель нужно добавить ограничения на прогнозирование отрицательных значений. Одним из способов сделать это является логарифмическое преобразование данных — прогноз составляется по логарифмам значений временного ряда, а не по самим значениям.

Впоследствии для знакомства с "истинным" рядом, включающим фактически измеренные числа, к прогнозу нужно будет применить экспоненциальное преобразование, чтобы представить его в реальных единицах измерения.

Во-вторых, неплохо бы найти более прозрачный способ обработки сезонных данных. Хотя мы описали нашу сезонную модель ARIMA как простую, на самом деле модель, имеющая показатель сезонной рекуррентности, равный 52, далеко не так проста, как кажется. Сезонные модели ARIMA лучше работают на более коротких сезонных циклах и заметно хуже — на более длинных сезонных циклах (52 — длинный сезонный цикл).

На следующем этапе к данным применяется метод *динамической гармонической регрессии*. В этом подходе мы будем рассчитывать ряд Фурье, описывающий периодичность в данных, а затем используем его в качестве экзогенного регрессора наряду с параметрами модели ARIMA.[7] Поскольку ряд Фурье экстраполируется вперед во времени (вследствие его периодического характера), этим можно воспользоваться для предварительного вычисления значений, которые, как ожидается, будут получены при расчете прогноза.

Преимущество такой модели заключается в том, что степени свободы модели можно использовать для объяснения основного поведения в дополнение

[6] Обычно они изучаются отдельно, но здесь рассматриваются совместно.

[7] Ряд Фурье — это метод разложения функций на последовательность синусов и косинусов. Ранее мы вкратце останавливались на описании рядов Фурье. Если вы с ними не знакомы, то обязательно выделите время на их изучение. Существует много хорошо зарекомендовавших себя способов подгонки рядов Фурье к любым временным рядам, и они широко реализованы в пакетах R и Python. У Ричи Винка (Ritchie Vink) есть краткое руководство по методологии Фурье (`https://perma.cc/7HJJ-HC2T`).

к сезонному поведению, а не для того, чтобы придавать сезонному поведению большую значимость.

Динамическая гармоническая регрессия имеет несколько недостатков. Во-первых, считается, что система характеризуется регулярным поведением, которое повторяется с точно таким же интервалом. Во-вторых, предполагается, что сезонное поведение не меняется — период и амплитуда сезонного поведения остаются прежними. Такие ограничения были свойственны модели SARIMA, но она демонстрирует большую гибкость при описании влияния амплитуды сезонных изменений на временные данные.

Ниже приведен пример реализации динамической гармонической регрессии с помощью инструментов языка R, аналогичного тому, как это делалось ранее.

```
## R
> ## Выделение памяти для векторов коэффициентов и аппроксимации
> fit.preds     <- array(0, dim = c(n, h))
> fit.coefs     <- array(0, dim = c(n, 100))
>
> ## Экзогенные регрессоры
> ## Компоненты ряда Фурье, аппроксимирующего данные
> flu.ts        <- ts(log(paris.flu$flu.rate + 1) + 0.0001,
>                     frequency = 52)
> ## Добавление небольшого смещения, поскольку маленькие и нулевые
> ## значения могут вызвать трудности при расчете аппроксимации
> exog.regressors <- fourier(flu.ts, K = 2)
> exog.colnames <- colnames(exog.regressors)
>
> ## Понедельное обучение модели с расширением окна обучающих данных
> for (i in (first.fit.size + 1):(nrow(paris.flu) - h)) {
>   data.to.fit <- ts(flu.ts[1:i], frequency = 52)
>   exogs.for.fit <- exog.regressors[1:i,]
>   exogs.for.predict <- exog.regressors[(i + 1):(i + h),]
>
>   fit <- auto.arima(data.to.fit,
>                     xreg = exogs.for.fit,
>                     seasonal = FALSE)
>
>    fit.preds[i - first.fit.size, ] <- forecast(fit, h = h,
>                                    xreg = exogs.for.predict)$mean
>    fit.coefs[i - first.fit.size, 1:length(fit$coef)] = fit$coef
> }
```

Здесь в код внесено несколько корректировок из предыдущего раздела. Во-первых, в нем используется объект `ts`,[8] при создании которого сезонность временного ряда обозначается в явном виде (52 недели).

На этом этапе изменяется логическая структура данных, чтобы обеспечить получение только положительных прогнозов уровня заболеваемости гриппом.

[8] Эти объекты кратко рассмотрены в главе 3.

```
## R
> flu.ts = ts(log(paris.flu$flu.rate + 1) + 0.0001, ## эпсилон
> frequency = 52)
```

Заметьте, что мы добавляем небольшое числовое смещение (+ 0,0001), потому что числовая аппроксимация плохо работает с нулевыми и даже очень маленькими значениями. С помощью этой строки кода выполняется только одна из двух корректировок (т.е. логарифмическое преобразование значений для обеспечения неотрицательности показателей заболеваемости гриппом).

Следующим этапом будет генерирование экзогенных гармонических регрессоров (получение приближения Фурье), которые будут применяться в качестве параметров сезонности в модели SARIMA. Задача выполняется с помощью функции `fourier()` пакета `forecast`.

```
## R
> exog.regressors <- fourier(flu.ts, K = 2)
> exog.colnames <- colnames(exog.regressors)
```

Сначала генерируются вспомогательные гармонические ряды для всего исходного временного ряда, а затем в конце цикла он перегруппировывается с помощью расширяющегося окна.

Гиперпараметр `K` указывает количество отдельных пар "синус/косинус", включаемых в приближение, каждая пара которого представляет отдельную гармонику. В общем случае параметр `K` задается большим для большей продолжительности сезонного периода и меньшим — для меньшей продолжительности. В более содержательном примере можно было бы предметно рассмотреть, как использовать информационный критерий для настройки параметра `K`, но здесь мы будем использовать `K` = 2 как наиболее оптимальный вариант.

Все, что нам осталось сделать, — это создать новые модели, учитывающие только что подобранные экзогенные компоненты Фурье. Их обучение выполняется таким образом, что параметр `xreg` принимает приближение временного ряда рядом Фурье в качестве дополнительных регрессоров, которые участвуют в обучении наряду со стандартными параметрами ARIMA.

```
## R
> fit <- auto.arima(data.to.fit,
>                   xreg = exogs.for.fit,
>                   seasinal = FALSE)
```

Учитывая решение об использовании динамической гармонической регрессии, устанавливаем параметр сезонности, указывающим на отсутствие избыточных сезонных параметров, равным `FALSE`.

Регрессоры также должны подключаться при прогнозировании, т.е. нам нужно указать их в коде вычисления прогноза.

```
## R
> fit.preds[i - first.fit.size, ] <- forecast(fit, h = 2h,
>                                   xreg = exogs.for.predict)$mean
```

График точности модели выводится (рис. 13.7) с помощью следующего кода.

```
## R
> ylim = range(paris.flu$flu.rate)
> plot(paris.flu$date[300:400], paris.flu$flu.rate[300:400],
>      ylim = ylim, cex = 0.8,
> xlab = "Date", ylab = "Flu rate")
> lines(paris.flu$date[300:400], exp(fit.preds[, h][(300-h):(400-h)]),
>       col = 2, type = 'l',
> lty = 2, lwd = 2)
```

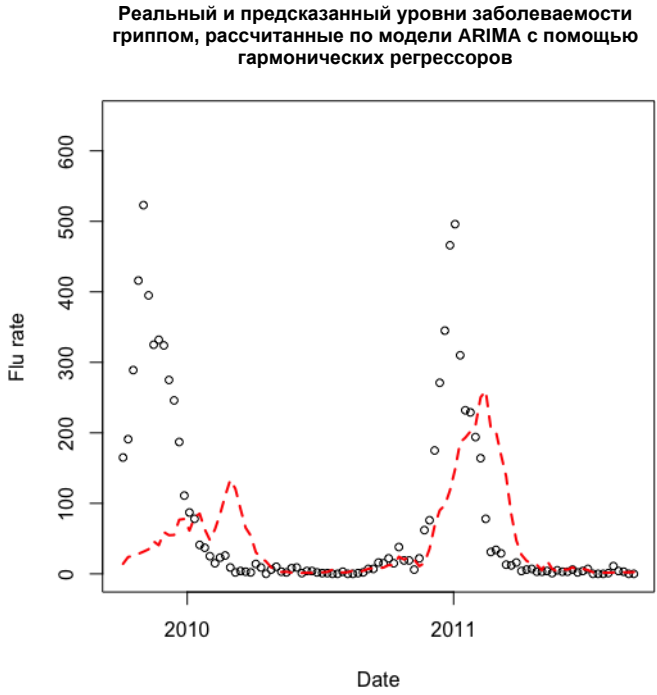


Рис. 13.7. *Графики реальных (точки) и прогнозируемых (пунктирная линия) значений заболеваемости гриппом по модели ARIMA с использованием динамической гармонической регрессии*

С одной стороны, наш прогнозируемый ряд больше не имеет отрицательных значений. С другой стороны, есть много неутешительных свидетельств низкой точности модели. Очевидно, что многие из прогнозов совершенно не верны. Пики далеки от правильных значений и расположены в неправильных местах.

В одном из объяснений выявленных проблем утверждается, что регулярная сезонность не является хорошим критерием описания сезонности

распространения гриппа.[9] По данным Центра по контролю и профилактике заболеваний (https://perma.cc/58SP-B3YH), в США грипп может достигать пика в каждый зимний сезон, начиная с декабря и заканчивая мартом. Мы наблюдаем это по данным тестирования. Рассмотрим следующий код и график, представленный на рис. 13.8, на котором пики проявляются в данных для тестирования.

```
## R
> plot(test$flu.rate)
> which(test$flu.rate > 400)
Error in which(res$flu.rate > 400) : object 'res' not found
> which(test$flu.rate > 400)
[1]  1  59 108 109 110 111 112 113
> abline(v = 59)
> which.max(test$flu.rate)
```

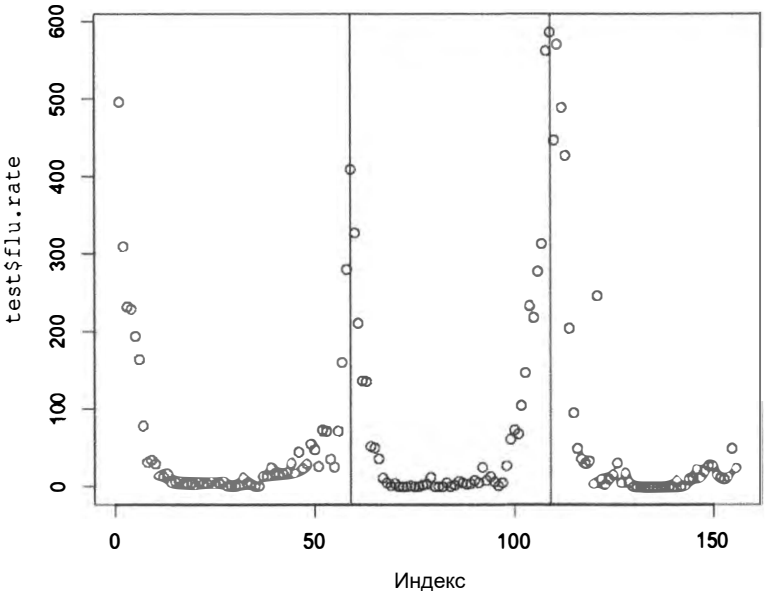


Рис. 13.8. *Диаграмма тестовых значений заболеваемости гриппом с явными пиками*

Пики наблюдаются в индексах 59 и 109, т.е. на расстоянии 50 (а не 52) недель один от другого. Кроме того, пик в индексе 59 достигнут не менее чем через 59

[9] Чтобы проверить, подходит ли такое объяснение для данной ситуации, нужно протестировать альтернативные искусственные наборы данных и убедиться, что ошибки модели вызываются циклическим, а не сезонным характером рядов, что наблюдается в наших исходных данных. Следовательно, здесь модель динамической гармонической регрессии с экзогенными регрессорами применять нельзя.

недель после предыдущего пика, а возможно и позднее, поскольку на графике не представлены полные сведения о пиках в окрестности нулевого индекса.

При расстоянии от пика до пика, превышающем 59 недель в одном случае и 50 — в другом, можно говорить о довольно слабой годовой изменчивости данных. Динамическая гармоническая регрессия не принимает этот факт во внимание. Она оценивает более ярко выраженные сезонные изменения, чем модель SARIMA, потому что последняя предполагает изменение сезонного поведения со временем. Это несоответствие между допущениями и данными в значительной степени объясняет низкую точность исходной модели — использование плохой модели фактически привлекло внимание к важной особенности данных, которая не была замечена ранее.

Несмотря на низкую точность альтернативная модель оказалась полезной по нескольким причинам. Она указала на необходимость принятия в расчет физических ограничений системы еще на этапе предварительной обработки данных, в данном случае обусловливающих их логарифмическое преобразование. Кроме того, вы узнали о важности рассмотрения сразу нескольких классов моделей в рамках базовых предположений, лежащих в основе их выбора. Сначала мы выбрали модель, которая предположительно должна была упростить описание сезонного поведения данных, но вместо этого обнаружили, что ни SARIMA, ни динамическая гармоническая регрессионная модель не являются достаточно хорошим выбором для полноценного описания системы.

Современные методы прогнозирования заболеваемости гриппом

Только что исследованные модели относительно простые, но достаточно эффективные для того, чтобы использоваться для составления кратко- и долгосрочных прогнозов заболеваемости гриппом в столичном регионе Парижа для коротких горизонтов. Эти модели являются хорошим началом как для ознакомления с рабочим набором данных, так и для выявления ограничений в зашумленном и сложном процессе, таком как распространение гриппа в целевом географическом регионе.

Но всегда интересно сравнить простые модели с передовыми решениями, которые уже используются государственными органами или только недавно разработаны учеными, стремящимися улучшить современное состояние отрасли. Далее вы узнаете о том, насколько хорошо в настоящее время подтверждаются прогнозы заболеваемости гриппом и какие новаторские подходы разработали исследователи за последние годы в попытках улучшить традиционные модели.

Исследования в прогнозировании распространения гриппа

Центр по контролю и профилактике заболеваний (CDC — Center for Diseases Control and Prevention) активно поощряет исследователей работать над методами

прогнозирования заболеваемости гриппом, в том числе спонсируя разработку инструментов пакета R, чтобы сделать их данные как можно более доступными. В течение более чем пяти лет CDC также спонсировал соревнования по прогнозированию заболеваемости гриппом, хотя только для данных сезона гриппа 2017 и 2018 годов. Фактически результаты прогнозирования заболеваемости гриппом были включены в официальные сообщения и бюллетени CDC. До последнего времени большинство таких соревнований выигрывала исследовательская группа Delphi из Университета Карнеги-Меллона. Для прогнозирования заболеваемости гриппом они используют три разных подхода.

- Эмпирический байесовский метод (`https://perma.cc/8EBA-GN2C`), в котором с помощью ряда манипуляций данные о заболеваемости гриппом в прошлом сезоне применяются для формирования априорных данных текущего сезона, основанных на общих "формах" временны*х* рядов в прошлом.
- Краудсорсинговая платформа (`https://perma.cc/XDE9-A9Y4`), на которой любой желающий может составить свой прогноз заболеваемости гриппом.
- Метод краткосрочного прогнозирования, в котором для создания локальных прогнозов заболеваемости гриппом используются данные, получаемые из *сочетания источников* (`https://perma.cc/NGZ8-TD39`), таких как счетчики доступа к Википедии и релевантные запросы Twitter.

И это набор подходов, с которыми работает всего одна исследовательская группа! В академических кругах наблюдается более широкое разнообразие рассматриваемых технологий.

- Использование глубоких сверточных сетей (CNN) для классификации изображений в сети Instagram и применение выходных данных этих сетей вместе с текстовыми показателями, полученными из сети Twitter, в качестве входных данных для различных моделей машинного обучения, включая деревья `XGBoost`. В одной из публикаций (`https://perma.cc/N39F-GSL5`) исследование проводилось в небольшом языковом сообществе (финской языковой группы), что позволило использовать основные социальные сети для получения прогнозов, имеющих исключительно региональный характер.
- Определение надежных пользователей в более крупных кластерах социальных сетей. В одной из статей (`https://perma.cc/25GR-MHRK`) основное внимание уделяется улучшению прогнозов заболеваемости гриппом путем нахождения наиболее подходящих и заслуживающих доверия пользователей на платформах социальных сетей.
- Получение доступа к электронным медицинским картам как полноценному и достоверному источнику данных в дополнение к общедоступным данным. В одной из статей (`https://perma.cc/Q8B7-5TC4`) показано, что чрезвычайно большой прирост точности прогнозирования в ряде

временны*х* периодов может быть достигнут за счет интеграции записей электронных медицинских карт в прогнозируемые входные потоки. К сожалению, это достаточно сложно организовать технически, и предполагается, что возможностями точного прогнозирования уровня заболеваемости гриппом будут обладать организации, предоставляющие обширные медицинские данные, а не только самые успешные исследовательские центры (хотя, конечно, иногда они могут выступать одним и тем же).

Как видим из разнообразия подходов к решению задачи, существует много способов получать точные прогнозы по заболеваемости гриппом, и в этом соревновании окончательный победитель пока не определен. Задача все еще остается областью активных исследований и разработок, даже несмотря на то, что некоторые из лучших стратегий уже приняты на вооружение правительственными агентствами и общественными организациями.

Наше обсуждение здесь — только вершина айсберга рассматриваемой проблематики. На заболеваемость гриппом влияют многочисленные биологические, социальные, медицинские и экономические факторы, и существует множество моделей, которые в меньшей степени ориентированы на анализ временны*х* рядов и в большей — акцентируются на других эпидемических аспектах. Временны*е* ряды представляют всего лишь один из многих инструментов выполнения этой сложной задачи.

Прогнозирование уровня глюкозы в крови

Еще одной областью исследований в здравоохранении с использованием технологий машинного обучения для временны*х* рядов является прогнозирование уровня глюкозы в крови отдельных пациентов. Больным диабетом приходится делать такие прогнозы постоянно, особенно при уровне заболевания, который предполагает инсулиновые инъекции, специально предназначенные для получения перед приемом пищи. В таких случаях больным диабетом нужно оценить, как потребляемая ими пища повлияет на уровень глюкозы в крови, и соответственно скорректировать дозу.

Аналогичным образом больные диабетом должны точно рассчитывать время приема пищи и лекарств, чтобы оптимизировать уровень глюкозы в крови, который лучше всего поддерживать в определенном диапазоне значений — не слишком высоком и не слишком низком. В дополнение к необходимости учитывать действия, влияющие на изменение уровня глюкозы в крови, такие как прием пищи и физические упражнения, диабетики должны принимать во внимание такие факторы, как время суток. Например, феномен "утренней зари" (`https://perma.cc/GE3B-MAKY`) заключается в повышении уровня глюкозы в крови, которое регистрируется у всех людей, но вызывает проблемы только у больных

диабетом. С другой стороны, для людей с сахарным диабетом 1-го типа снижение уровня глюкозы в крови во время сна может быть опасным для жизни событием, возникающим вследствие неправильно сделанного прогноза.

Далее мы рассмотрим небольшой набор данных: самостоятельно собранные значения непрерывного мониторинга уровня глюкозы (CGM — Continuous Glucose Monitor) одним из пациентов за несколько несмежных временных периодов. Эти данные были опубликованы самим больным диабетом в Интернете и изменены, чтобы сохранить его конфиденциальность.

Есть и другие источники наборов данных о протекании заболевания диабетом. Кроме крупных медицинских организаций и некоторых стартапов, располагающих большими объемами данных CGM, существует множество самостоятельно публикуемых наборов данных — люди все чаще справляются с лечением диабета собственными силами, — например на ресурсе Night Scout (`https://perma.cc/N42T-A35K`). Несколько открытых для исследовательских целей наборов данных CGM можно найти на сайте The OhioT1DM Dataset (`https://perma.cc/RXG2-CYEE`).

В этом разделе мы рассмотрим особенности реальных наборов данных и попытаемся сделать по ним прогноз.

Очистка и исследование данных

Данные хранятся в нескольких файлах, доступны для загрузки по такому адресу:

```
http://www.williamspublishing.com/Books/978-5-907365-04-9.html
```

Загрузим и объединим их в один файл `data.table`.

```
## R
> files <- list.files(full.names = TRUE)
> files <- grep("entries", files, value = TRUE)
> dt    <- data.table()
> for (f in files) {
>   dt <- rbindlist(list(dt, fread(f)))
> }
>
> ## Удаление столбцов, содержащих значения N/A
> dt <- dt[!is.na(sgv)]
```

Нам доступны строки с информацией о дате, но мы не располагаем столбцом с правильными временны*ми* метками, поэтому создадим его, основываясь на имеющейся информации о часовом поясе и дате.

```
## R
> dt[, timestamp := as.POSIXct(date)]
> ## Инструкция корректна, поскольку мой компьютер находится в
> ## часовом поясе EST, но у вас это условие может не выполняться
```

```
> ## Правильный способ
> dt[, timestamp := force_tz(as.POSIXct(date), "EST")]
>
> ## Хронологический порядок
> dt = dt[order(timestamp, decreasing = FALSE)]
```

Проверим результаты обработки данных.

```
## R
> head(dt[, .(date, sgv)])
                 date sgv
1: 2015-02-18 06:30:09 162
2: 2015-02-18 06:30:09 162
3: 2015-02-18 06:30:09 162
4: 2015-02-18 06:30:09 162
5: 2015-02-18 06:35:09 154
6: 2015-02-18 06:35:09 154
```

Наблюдается много повторяющихся записей данных, и их нужно очистить по двум причинам.

- Априори нет никаких причин на то, чтобы считать определенные точки данных привилегированными или задавать им больший вес, чем другим, но дублирование приведет к подобному эффекту.
- Собираясь сформировать объект, основанный на окнах временны*х* рядов, примите к сведению, что повторяющиеся записи будут бессмысленными при наличии продублированных временны*х* точек.

Давайте посмотрим, можно ли решить эту проблему, удалив все продублированные строки.

```
## R
> dt <- dt[!duplicated(dt)]
```

Однако не будем самонадеянными и удостоверимся, что предпринятый шаг позволил устранить проблему. Проверим, существуют ли неидентичные точки данных для одной и той же временно*й* точки.

```
## R
> nrow(dt)
[1] 24861
> length(unique(dt$timestamp))
[1] 23273
> ## Среди данных все еще наблюдаются дубликаты временных точек.
> ## Удалим их
```

Поскольку продублированные временны*е* точки все еще встречаются, исследуем данные, чтобы понять, как такое может быть. Определим временную метку с наиболее часто повторяющимися строками и исследуем их.

```
## R
> ## Единичный экземпляр определяется с помощью инструкции
> ## dt[, .N, timestamp][order(N)]
> ## Проведем поиск наиболее часто повторяющихся данных
> dt[date == "2015-03-10 06:27:19"]
device                date                dateString  sgv direction type
1: dexcom 2015-03-10 06:27:19 Tue Mar 10 06:27:18 EDT 2015  66      Flat sgv
2: dexcom 2015-03-10 06:27:19 Tue Mar 10 06:27:18 EDT 2015  70      Flat sgv
3: dexcom 2015-03-10 06:27:19 Tue Mar 10 06:27:18 EDT 2015  66      Flat sgv
4: dexcom 2015-03-10 06:27:19 Tue Mar 10 06:27:18 EDT 2015  70      Flat sgv
5: dexcom 2015-03-10 06:27:19 Tue Mar 10 06:27:18 EDT 2015  66      Flat sgv
6: dexcom 2015-03-10 06:27:19 Tue Mar 10 06:27:18 EDT 2015  70      Flat sgv
7: dexcom 2015-03-10 06:27:19 Tue Mar 10 06:27:18 EDT 2015  66      Flat sgv
8: dexcom 2015-03-10 06:27:19 Tue Mar 10 06:27:18 EDT 2015  70      Flat sgv
## Дополнительная проверка показывает, что данные не очень важные
```

Просматривая полученные значения, легко заметить, что они относятся к различным, но не сильно различающимся значениям уровня глюкозы в крови[10], поэтому можем смело удалить дубликаты временны*х* меток, даже если они не полностью совпадают.[11]

```
## R
> dt <- unique(dt, by=c("timestamp"))
```

Теперь, получив одни только правильные временны*е* метки и свои значения для каждой из них, повторно построим график исходных значений, чтобы оценить их разброс и поведении (рис. 13.9).

Увеличим масштаб начального периода временно*г*о ряда, т.е. за март 2015 года (рис. 13.10).

```
## R
> ## Более короткий временной период
> start.date <- as.Date("2015-01-01")
> stop.date <- as.Date("2015-04-01")
> dt[between(timestamp, start.date, stop.date),
>              plot(timestamp, sgv, cex = .5)]
```

[10] Допустимая погрешность приборов измерения уровня глюкозы в крови, как правило, составляет около 15 единиц измерения, принятых в США.

[11] Есть и другие варианты, кроме удаления. Изучите их самостоятельно.

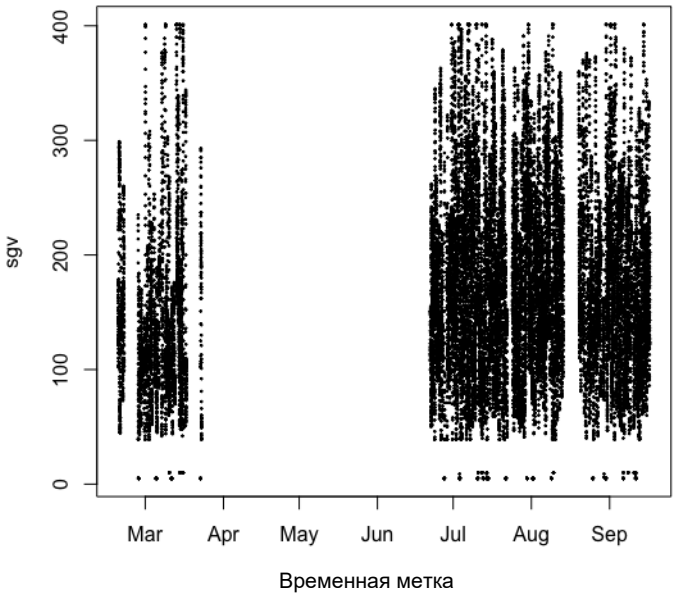


Рис. 13.9. *Исходный график временного ряда для всех доступных данных, построенный на полном временном диапазоне*

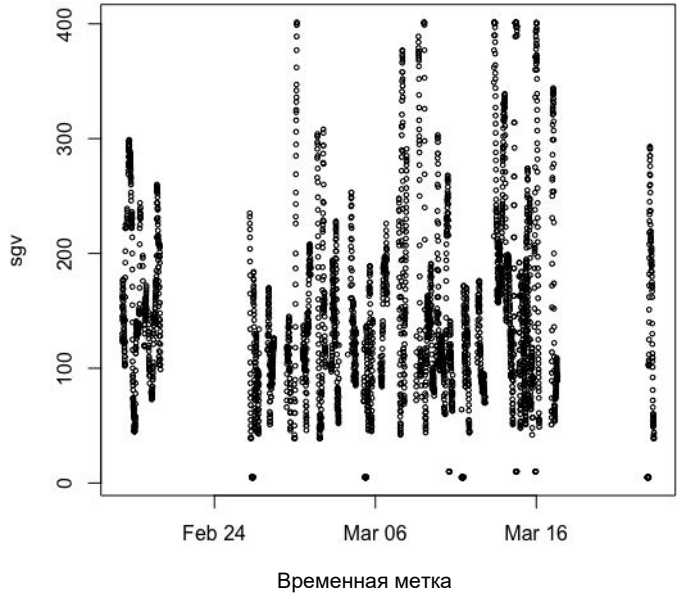


Рис. 13.10. *Детальный вид отдельного отрезка временного ряда более информативный, но все еще не дает полного представления о его динамике. Необходимо увеличить масштаб времени еще больше*

Даже график, построенный только по мартовским данным, не представляет полных сведений о поведении временного ряда. Присутствует ли в нем сезонная составляющая? Наблюдается ли дрейф? Какой их дневной шаблон? Ответов на эти вопросы нет, поэтому построим график для еще более короткого временного окна (рис. 13.11).

```
## R
> ## Изучим отдельный день для поиска в нем поведенческих шаблонов.
> ## При наличии достаточных данных желательно построить и изучить
> ## двумерные гистограммы для дневных данных
> par(mfrow = c(2, 1))
> start.date = as.Date("2015-07-04")
> stop.date = as.Date("2015-07-05")
> dt[between(timestamp, start.date, stop.date),
>                plot(timestamp, sgv, cex = .5)]
>
> start.date = as.Date("2015-07-05")
> stop.date = as.Date("2015-07-06")
> dt[between(timestamp, start.date, stop.date),
>                plot(timestamp, sgv, cex = .5)]
> ## Располагая данными по другим дням, можно было бы попробовать
> ## найти "дневные формы", но данных явно недостаточно
```

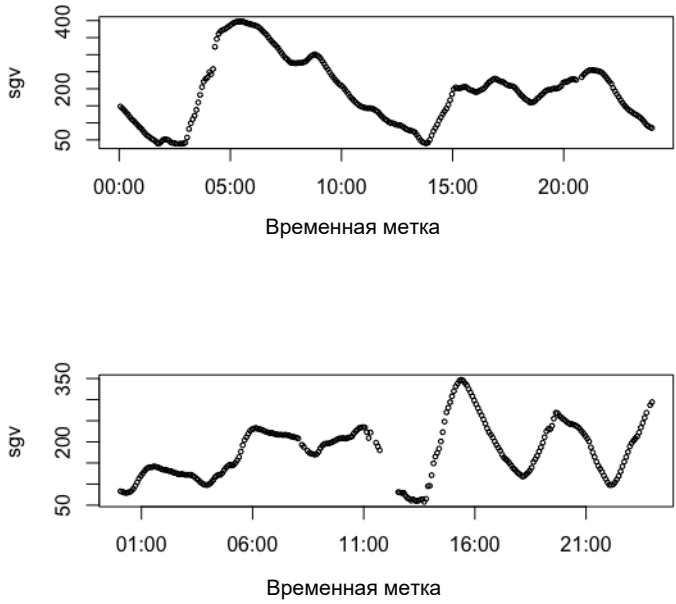


Рис. 13.11. *График данных для двух отдельных дней в июле. Наблюдается определенный суточный шаблон. Наконец-то временные данные представлены в форме, подлежащей хоть какому-то осмыслению*

Создавая все больше таких исследовательских графиков (оставляем эту задачу в качестве дополнительного упражнения), вы будете развивать интуитивное представление о масштабе данных, их качественной динамике и особенностях поведения. Перечислим другие доступные исследовательские методы.

- Двумерные гистограммы дневной длительности для поиска почасовых шаблонов и сопутствующих упражнений по кластеризации данных. Можно ли разделить дни на виды?
- Группировка статистики по часам дня или временам года для поиска систематических временны*х* отличий.
- Сглаживание данных для поиска долгосрочных тенденций и, в частности, для сравнения соотношений для разных дней, а не на протяжении одного и того же дня. Полезно разработать сверхдолгосрочные прогнозы для уровня глюкозы в крови, что можно сделать, если выйти за пределы более очевидных почасовых схем.

Генерация признаков

Вооружившись некоторыми базовыми знаниями, а также результатами наблюдений, полученных в ходе краткого исследования набора данных, можно приступить к генерированию признаков, которые могут оказаться полезными для прогнозирования уровня глюкозы в крови.

Начнем с понятия времени как такового. Например, у дня наблюдается упорядоченная временна*я* структура, которая предполагает, что точность предсказаний зависит от времени дня, для которых они делаются. Аналогично, просматривая временны*е* ряды со значениями уровня глюкозы в крови для разных месяцев, можно заметить, что для одних месяцев дисперсия больше, чем для других.[12] В приведенном ниже коде генерируется сразу несколько признаков, связанных с понятием времени (рис. 13.12).

```
## R
> ## Рассмотрим признаки для месяцев года
> ## Возможны сезонные эффекты
> dt[, month := strftime(timestamp, "%m")]
>
> dt[, local.hour := as.numeric(strftime(timestamp, "%H"))]
> ## Некоторые часы в течение суток представлены намного чаще других.
```

[12] Все такие наблюдения следует воспринимать с большой степенью скептицизма. Этот набор состоит только из данных, полученных от одного пользователя для частей одного года, и они не являются непрерывными. Это означает, что мы не можем делать выводы о сезонности, даже несмотря на большой соблазн так поступить. Тем не менее, чтобы показать обучаемость модели на ограниченных данных, которыми мы располагаем, рассмотрим указанные временны*е* компоненты.

```
> ## Пациент либо неправильно пользовался устройством, либо оно
> ## было запрограммировано на указанные часы, либо данные
> ## записывались только в указанное время. Необычно высокий пик
> ## в полночь обусловлен скорее особенностью поведения устройства,
> ## чем пациента (зачем выключать прибор в полночь?)
>
> hist(dt$local.hour)
```

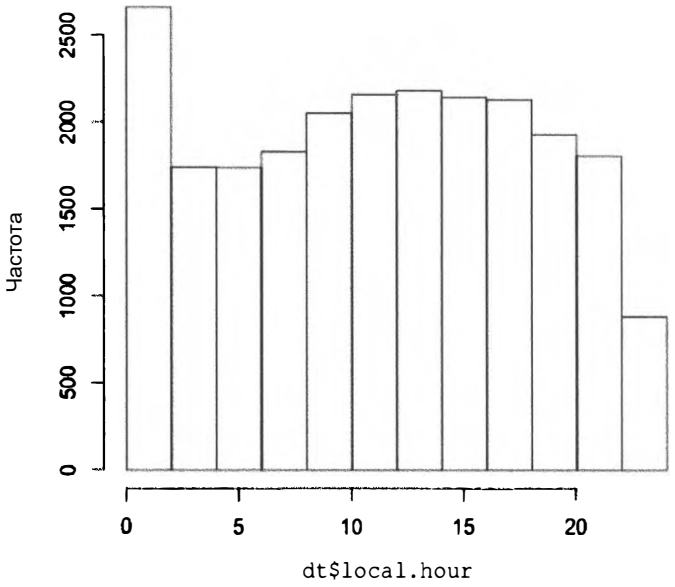


Рис. 13.12. *Почасовая гистограмма CGM показывает, что данные собирались и записывались не случайным образом. Высокая концентрация значений в окрестности полуночи свидетельствует о том, что в работе записывающего устройства или самого прибора наблюдается регулярность, которая никак не связана с привычками пациента (маловероятно, чтобы он использовал прибор в это время суток)*

Также обратим внимание на величину значений временного ряда, а не только на временные метки их регистрации. Например, можно видеть, что средние значения уровня глюкозы в крови подвержены изменению в зависимости от времени суток (рис. 13.13).

```
## R
> ## Значения уровня глюкозы в крови зависит от времени суток
> plot(dt[, mean(sgv), local.hour][order(local.hour)], type = 'l',
>       ylab = "Mean blood glucose per local hour",
>       xlab = "Local hour")
```

Значения уровня глюкозы в крови характеризуются другими интересными показателями. Столбец с названием `direction` в наборе данных включает информацию, предоставляемую CGM-прибором с помощью запатентованного программного обеспечения, — подпись направленного тренда. Этот признак можно использовать вместо того, чтобы пытаться искать собственные тренды. Попробуем разобраться в нем, например, для определения того, существуют ли тренды для разных часов.

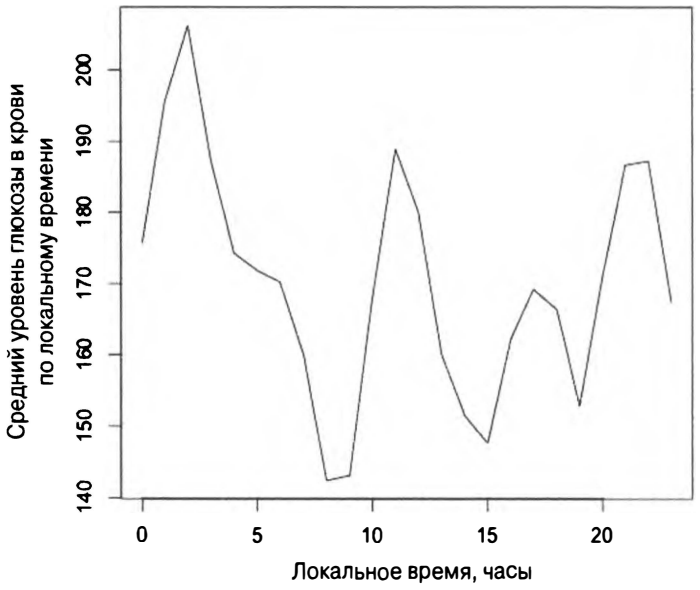


Рис. 13.13. *Средний уровень глюкозы в крови существенно изменяется в течение суток*

Вначале создадим функцию, которая возвращает *n*-ю самую популярную подпись для заданного вектора, и используем ее для нахождения самой популярной подписи для каждого часа (по местному времени) дня.

```
## R
> nth.pos = function(x, pos) {
>   names(sort(-table(x)))[pos]
>   ## Код заимствован у сообщества пользователей языка R
> }
> dt[, nth.pos(direction, 1), local.hour][order(local.hour)]
     local.hour   V1
 1:          0 Flat
 2:          1 Flat
 3:          2 Flat
 ...
21:         20 Flat
```

```
22:          21 Flat
23:          22 Flat
24:          23 Flat
     local.hour   V1
```

Наиболее популярная подпись направления — "Flat" для всех часов дня, что обнадеживает, потому что динамика системы была бы сомнительной, если бы тренд можно было точно предсказать по времени дня.

Тем не менее вторая по распространенности подпись направления демонстрирует зависимость от времени суток.

```
## R
> dt[, nth.pos(direction, 2), local.hour][order(local.hour)]
    local.hour             V1
1:          0    FortyFiveUp
2:          1    FortyFiveUp
3:          2    FortyFiveUp
4:          3  FortyFiveDown
5:          4  FortyFiveDown
6:          5    FortyFiveUp
7:          6  FortyFiveDown
8:          7  FortyFiveDown
9:          8  FortyFiveDown
10:         9    FortyFiveUp
11:        10    FortyFiveUp
12:        11  FortyFiveDown
13:        12  FortyFiveDown
14:        13  FortyFiveDown
15:        14  FortyFiveDown
16:        15  FortyFiveDown
17:        16    FortyFiveUp
18:        17    FortyFiveUp
19:        18  FortyFiveDown
20:        19    FortyFiveUp
21:        20    FortyFiveUp
22:        21    FortyFiveUp
23:        22    FortyFiveUp
24:        23  FortyFiveDown
    local.hour             V1
```

Интересно сопоставить эти подписи с графиками почасовых данных, которые рассматривались в предыдущем разделе. Совпадают ли результаты? (Ответ на этот вопрос будет упражнением для самостоятельного выполнения.)

Далее мы обращаемся к числовой подписи, которая может быть прогностической, а именно — показывающей дисперсию уровня глюкозы в крови в самом последнем временном окне. Для этого вычислим стандартное отклонение для небольшого окна за предыдущий период.

Нам придется учитывать несмежный характер данных. Нельзя вычислять стандартные отклонения для несмежных во времени данных, хотя они и записаны последовательно в объекте `data.table`. Вычислим разницу во времени между начальной и конечной точками целевого окна, чтобы убедиться в том, что они относятся к общему временному масштабу. На этом этапе нужно быть предельно осторожными. Ниже показано, как можно легко допустить ошибку.

```
## R
> ## СНАЧАЛА о том, как нельзя поступать
> ## Замечание: не вычисляйте бессмысленные временные разности!
> as.numeric(dt$timestamp[10] - dt$timestamp[1])
[1] 40
> as.numeric(dt$timestamp[1000] - dt$timestamp[1])
[1] 11.69274
> dt$timestamp[1000] - dt$timestamp[1]
Time difference of 11.69274 days
```

Если временные разности задаются в разных масштабных единицах, то результаты, возвращаемые предыдущим кодом, оказываются бессмысленными. Ниже проиллюстрирован правильный способ вычисления временных разностей, а также способ определения достоверности вычисления стандартного отклонения для заданной строки.

```
## R
> dt[, delta.t := as.numeric(difftime(timestamp, shift(timestamp, 6),
>                                     units = 'mins'))]
> dt[, valid.sd := !is.na(delta.t) & delta.t < 31]
> dt[, .N, valid.sd]
valid.sd     N
1: FALSE  1838
2: TRUE  21435
```

Запускаем вычисления, как только будут подписаны строки, для которых рассчитывается стандартное отклонение. Проходим по всем строкам, перезаписывая недопустимые величины средними значениями по столбцу и устраняя недостающие значения наиболее простым из возможных способом.

```
## R
> dt[, sd.window := 0]
> for (i in 7:nrow(dt)) {
>   dt[i, ]$sd.window = sd(dt[(i-6):i]$sgv)
> }
> ## Восполняем пропущенные данные для некорректного рассчитанного
> ## стандартного отклонения, заменяя средним по генеральной
> ## совокупности.
> ## Внимание: это УПРЕЖДЕНИЕ, но заранее оговоренное
> imputed.val = mean(dt[valid.sd == TRUE]$sd.window)
> dt[valid.sd == FALSE, sd.window := imputed.val]
```

Теперь нужно выделить столбец для значения, которое будет получено в процессе прогнозирования. Попробуем предсказать уровень глюкозы в крови на 30 минут вперед, чего вполне достаточно, чтобы заранее предупредить пациента об опасно высоком или низком значении. Целевое значение прогноза заносится в столбец с именем `target`. Также нужно создать еще один столбец, `pred.valid`, в котором будет указываться, являются ли данные, предшествующие времени прогноза, достаточно полными (регулярно отбираемыми в течение 30 минут с шагом 5 минут).

```
## R
> ## Теперь необходимо задать целевое значение для прогноза
> ## Кроме того, нужно проверить корректность вычисления
> ## стандартного отклонения по разреженным данным в одном
> ## и том же файле data.table. Прогноз строится на 30 минут вперед
> ## (короткие прогнозы легче рассчитать)
>
> ## Сдвиг на 6 точек, поскольку данные наблюдаются через 5 минут
> dt[, pred.delta.t := as.numeric(difftime(shift(timestamp, 6,
>                                                type = "lead"),
>                                          timestamp,
>                                          units = 'mins'))]
> dt[, pred.valid := !is.na(pred.delta.t) & pred.delta.t < 31]
>
> dt[, target := 0]
> for (i in 1:nrow(dt)) {
>   dt[i, ]$target = dt[i + 6]$sgv
> }
```

Проверим результаты проделанной работы, чтобы убедиться в оправданности возлагаемых надежд.

```
## R
> ## Проверка полученных результатов
> i = 300
> dt[i + (-12:10), .(timestamp, sgv, target, pred.valid)]
             timestamp sgv target pred.valid
1: 2015-02-19 16:15:05  146    158       TRUE
2: 2015-02-19 16:20:05  150    158       TRUE
3: 2015-02-19 16:25:05  154    151      FALSE
4: 2015-02-19 16:30:05  157    146      FALSE
5: 2015-02-19 16:35:05  160    144      FALSE
6: 2015-02-19 16:40:05  161    143      FALSE
7: 2015-02-19 16:45:05  158    144      FALSE
8: 2015-02-19 16:50:05  158    145      FALSE
9: 2015-02-19 17:00:05  151    149       TRUE
10: 2015-02-19 17:05:05 146    153       TRUE
11: 2015-02-19 17:10:05 144    154       TRUE
12: 2015-02-19 17:15:05 143    155       TRUE
13: 2015-02-19 17:20:05 144    157       TRUE
14: 2015-02-19 17:25:05 145    158       TRUE
15: 2015-02-19 17:30:05 149    159       TRUE
```

```
16: 2015-02-19 17:35:05 153     161      TRUE
17: 2015-02-19 17:40:05 154     164      TRUE
18: 2015-02-19 17:45:05 155     166      TRUE
19: 2015-02-19 17:50:05 157     168      TRUE
20: 2015-02-19 17:55:05 158     170      TRUE
21: 2015-02-19 18:00:04 159     172      TRUE
22: 2015-02-19 18:05:04 161     153      FALSE
23: 2015-02-19 18:10:04 164     149      FALSE
                    timestamp sgv target pred.valid
```

Внимательно ознакомьтесь с результатами. Что-то в них должно заставить вас усомниться: не слишком ли строго мы оценили достоверность данных при вычислении стандартного отклонения уровня глюкозы для последнего окна? В качестве самостоятельного упражнения попробуйте определить, как можно учесть это замечание, и задумайтесь о том, как переназначить подписи `pred.valid` так, чтобы сделать их более полными.

Теперь, определившись с набором применяемых признаков и целевым значением, которое можно использовать для обучения любой модели, применяемой для составления прогнозов, нужно закончить с генерированием признаков. Давайте упростим отдельные, ранее заданные, признаки времени, чтобы понизить сложность модели. Например, вместо того, чтобы представлять часы локального времени 23 двоичными числами (по одному на каждый час дня минус один), уменьшим количество категорий "час", как показано ниже.

```
## R
> ## Разделим сутки на четверти, а не на 24 часа.
> ## Введем понятие 'типичного' дня
> dt[, day.q.1 := between(local.hour, 5, 10.99)]
> dt[, day.q.2 := between(local.hour, 11, 16.99)]
> dt[, day.q.3 := between(local.hour, 17, 22.99)]
> dt[, day.q.4 := !day.q.1 & !day.q.2 & !day.q.3]
```

Подобным образом упростим данные месяцев, сократив количество их категорий.

```
## R
> ## Введем подписи "зима"/"не зима" вместо подписей месяцев
> ## Решение частично основано на временном распределении данных
> dt[, is.winter := as.numeric(month) < 4]
```

Наконец, для использования столбца `direction` необходимо обработать его значение в коде так, как это делалось для разных времен дня. Кроме того, в следующем коде упорядочиваются отдельные неодинаково обозначенные признаки (`"NOT COMPUTABLE"` наряду с `"NOT_COMPUTABLE"`).

```
## R
> ## Описываем признак направления и приводим в порядок данные
> dt[direction == "NOT COMPUTABLE", direction := "NOT_COMPUTABLE"]
```

```
> dir.names = character()
> for (nm in unique(dt$direction)) {
>    new.col = paste0("dir_", nm)
>    dir.names = c(dir.names, new.col)
>    dt[, eval(parse(text = paste0(new.col, " :=
                                    (direction == '", nm, "')")))]
> }
```

Теперь мы располагаем релевантными признаками прямого отображения и другими упрощенными признаками, которые можно включать в модель.

Обучение модели

Наконец, мы добрались до самой интересной части анализа временных рядов — прогнозирования.

Сколько времени длится моделирование

Как было показано раннее на примере двух моделей, используемых в отраслевых приложениях, реальные данные очень запутанные, и при каждой их очистке нужно соблюдать осторожность, учитывая требования предметной области и руководствуясь здравым смыслом. Здесь нет общего шаблона, тем не менее старайтесь действовать предусмотрительно и последовательно. Приступайте к обучению модели только тогда, когда удостоверитесь в том, что не вводите в нее мусор!

Наша первая задача — создание наборов данных для обучения и тестирования.[13]

```
## R
> ## Настройка обучающих данных и данных тестирования
> ## Тестирование нельзя выполнять на данных конца периода тестирования,
> ## поскольку данные предположительно носят сезонный характер,
> ## а значит, оно должно проводиться на данных конца обоих "сезонов"
> winter.data      <- dt[is.winter == TRUE]
> train.row.cutoff <- round(nrow(winter.data) * .9)
> train.winter     <- winter.data[1:train.row.cutoff]
> test.winter      <- winter.data[(train.row.cutoff + 1):
                                  nrow(winter.data)]
>
```

[13] На практике при работе с большими наборами данных необходимо предусмотреть набор проверочных тестов, представленный специальным поднабором обучающих данных, похожих на тестовые, но непосредственно тестирование на которых не выполняется. Подобно тому, как тестовые данные обычно представляются последними временными значениями (для предотвращения упреждения, т.е. утечки информации из будущего в прошлое), набор данных для проверки выбирается в конце периода обучения. Подумайте также о том, как правильно подбирать гиперпараметры модели.

```
> spring.data      <- dt[is.winter == FALSE]
> train.row.cutoff <- round(nrow(spring.data) * .9)
> train.spring     <- spring.data[1:train.row.cutoff]
> test.spring      <- spring.data[(train.row.cutoff + 1):
                                  nrow(spring.data)]
>
> train.data       <- rbindlist(list(train.winter, train.spring))
> test.data        <- rbindlist(list(test.winter, test.spring))
>
> ## Включаются только столбцы с категориальными значениями:
> ## valid.sd, day.q.1, day.q.2, day.q.3, is.winter
> ## а также все столбцы с префиксом 'dir_'
> col.names <- c(dir.names, "sgv", "sd.window", "valid.sd",
> "day.q.1", "day.q.2", "day.q.3", "is.winter")
>
> train.X <- train.data[, col.names, with = FALSE]
> train.Y <- train.data$target
>
> test.X <- test.data[, col.names, with = FALSE]
> test.Y <- test.data$target
```

Ориентируясь на самые совершенные технологии, используем в качестве модели деревья с градиентным бустингом XGBoost. В последних публикациях показано, что в некоторых сценариях они соответствуют современным показателям прогнозирования уровня глюкозы в крови и даже превышают их.[14]

```
## R
> model <- xgboost(data = as.matrix(train.X), label = train.Y,
>                  max.depth = 2,
>                  eta       = 1,
>                  nthread   = 2,
>                  nrounds   = 250,
>                  objective = "reg:linear")
> y.pred <- predict(model, as.matrix(test.X))
```

Теперь можно переходить к изучению результатов прогнозирования (рис. 13.14). Рассмотрим прогноз, составленный для отдельного дня (рис. 13.15).

```
## R
> ## Прогноз на определенный день
> test.data[date < as.Date("2015-03-17")]
> par(mfrow = c(1, 1))
> i <- 1
> j <- 102
> ylim <- range(test.Y[i:j], y.pred[i:j])
> plot(test.data$timestamp[i:j], test.Y[i:j], ylim = ylim)
> points(test.data$timestamp[i:j], y.pred[i:j], cex = .5, col = 2)
```

[14] Здесь гиперпараметры не настраиваются (требуют отдельного набора данных для проверки), но такую настройку нужно выполнять, работая в XGBoost и придерживаясь общего конвейера моделирования.

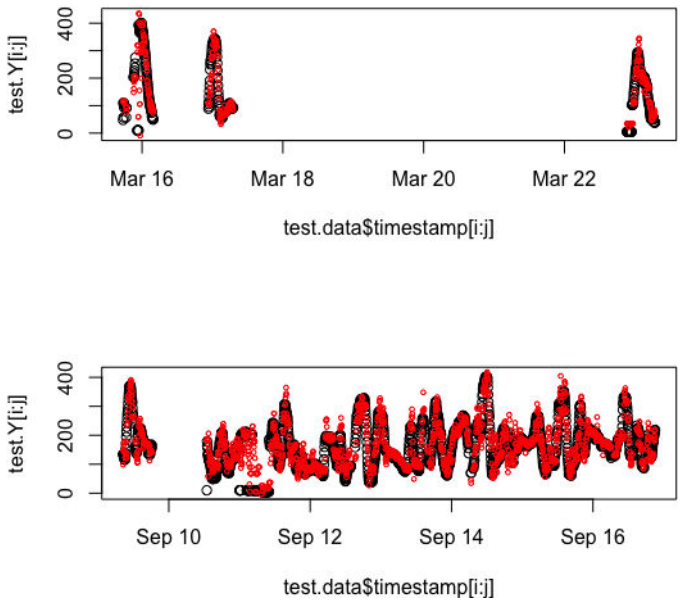


Рис. 13.14. *В крупном масштабе прогнозы выглядят точными, но трудно сказать, будут ли они таковыми при увеличении масштаба. Хочется верить, что точность прогноза не ухудшится*

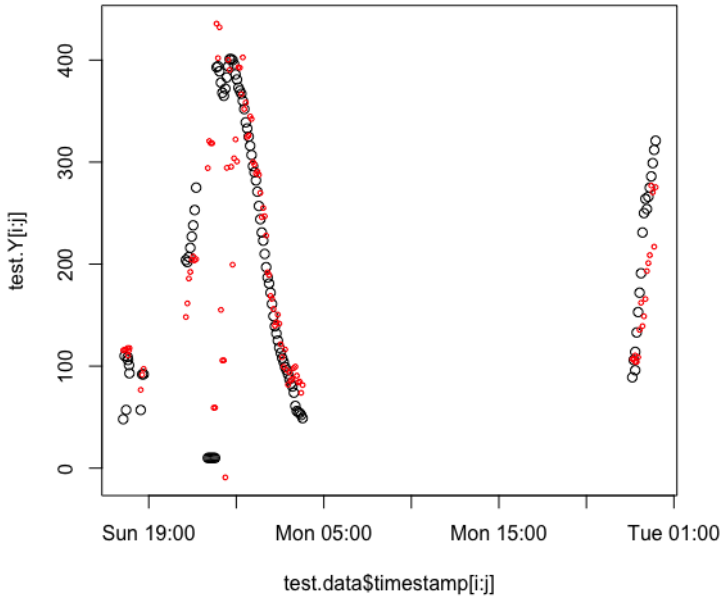


Рис. 13.15. *Почасовой анализ прогноза для отдельного дня позволяет получить более точное представление о его точности*

Прогнозы выглядят вполне правдоподобными, но на графике показано, что в его исходных данных просматриваются определенные несоответствия. Точки данных для полуночи с воскресенья на понедельник представлены нереально малыми значениями, особенно с учетом показателей соседних точек. Скорее всего, это объясняется неисправностью устройства, а не резким, кратковременным падением уровня глюкозы в крови пользователя. Рассмотрите возможность дополнительной очистки данных или включения дополнительных подписей, указывающих на "выпадение" определенных точек.[15] Предположительно это свидетельствует о том, что модель неверно прогнозирует данные для низкого уровне глюкозы в крови. Нам нужно, чтобы модель предоставляла прогноз фактических значений уровня глюкозы в крови, а не показаний неисправности устройства.

Перейдем к рассмотрению целей нашего алгоритма. Если нужно просто показать, что он выполнил достаточно хорошую работу по прогнозированию уровня глюкозы в крови на полчаса вперед, не учитывая то, чем занимается пациент в это время (принимает пищу или тренируется), то можно смело утверждать, что "миссия выполнена". Это впечатляющий результат сам по себе, свидетельствующий о том, что можно делать правдоподобные прогнозы, не располагая такими сопутствующими данными, как сведения о физической деятельности пациента.

Однако мы стремимся к основной цели — составлению прогноза, призванного предупредить пациентов об опасности, возникающей, когда уровень глюкозы в крови становится слишком низким или высоким. Давайте сосредоточимся на таких точках данных при сопоставлении прогнозируемых и измеренных значений.

Важно отметить, что высокий и низкий уровни глюкозы в крови рассматриваются отдельно. При их совместной визуализации можно наблюдать сильную корреляцию между высокими и низкими значениями, описываемыми неестественным гантелеобразным распределением (рис. 13.16).

```
## R
> par(mfrow = c(2, 1))
> high.idx = which(test.Y > 300)
> plot(test.Y[high.idx], y.pred[high.idx], xlim = c(200, 400),
>                                         ylim = c(200, 400))
> cor(test.Y[high.idx], y.pred[high.idx])
[1] 0.3304997
>
```

[15] На практике при работе с большими наборами данных необходимо предусмотреть набор проверочных тестов, представленный специальным поднабором обучающих данных, похожих на тестовые, но непосредственно тестирование на которых не выполняется. Подобно тому, как тестовые данные обычно представляются последними временными значениями (для предотвращения упреждения, т.е. утечки информации из будущего в прошлое), набор данных для проверки выбирается в конце периода обучения. Подумайте также о том, как правильно подбирать гиперпараметры модели.

```
> low.idx = which((test.Y < 60 & test.Y > 10))
> plot(test.Y[low.idx], y.pred[low.idx], xlim = c(0, 200),
>                                        ylim = c(0, 200))
> cor(test.Y[low.idx], y.pred[low.idx])
[1] 0.08747175
```

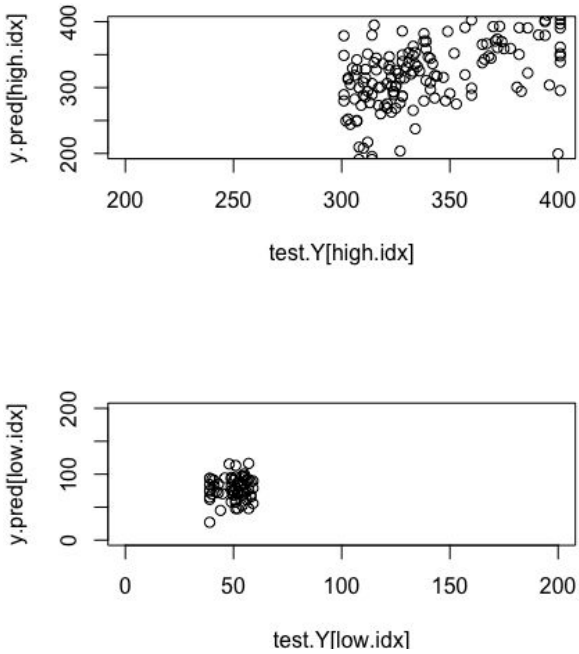


Рис. 13.16. *Диаграмма прогнозируемых и реальных значений. В верхней области построения отображаются высокие, а в нижней — низкие значения уровня глюкозы в крови*

Приведенные графики дают представление о точности модели в более приземленном виде. Особое беспокойство вызывает то, что модель не очень хорошо справляется с предсказанием низких значений уровня глюкозы в крови. Это может быть вызвано тем, что опасно низкие уровни глюкозы в крови регистрируются крайне редко, поэтому у модели не так уж и много возможностей обучения им. Нужно обязательно рассмотреть возможность увеличения объема данных или проведения повторной выборки, чтобы сделать такие события более заметными для модели. Также попробуйте изменить функцию потерь для увеличения веса примеров, уже присутствующих в данных.

Здесь мы столкнулись с ситуацией, когда прогнозы исходной модели оказываются достаточно правдоподобными, но не соответствуют первоначальной (наиболее важной) цели. В данном случае хорошо предсказываются общие тенденции в изменении уровня глюкозы в крови, но нам предстоит выполнить работу

по более точному прогнозированию высокого и низкого уровней глюкозы в крови, угрожающих жизни пациента.

В данном случае, как и в случае прогнозирования заболеваемости гриппом, мы увидели, что моделирование временных рядов требует здравого и трезвого подхода. Нельзя беспринципно очищать данные или генерировать признаки, не задумываясь об их предназначении в контексте использования данных. Точно так же, чем больше нам известно об условиях, в которых собирались данные временных рядов, например время суток регистрации уровня глюкозы в крови пациентов, тем лучше мы сможем очистить данные и подготовить признаки и точнее оценить правдоподобность прогнозных значений.

Дополнительные источники

Vasileios Lampos et al., "Advances in Nowcasting Influenza-Like Illness Rates Using Search Query Logs," Scientific Reports 5, no. 12760 (2015), `https://perma.cc/NQ6B-RUXF`

Статья в Nature Communications 2015 года, предоставленная для широкого доступа, включает обзор современных методологий прогнозирования заболеваемости гриппом. Содержит исторический материал, в котором рассмотрена революционная роль больших данных и социальных сетей, работающих в реальном времени, а также данных, полученных из Интернета, в развитии технологий краткосрочного и долгосрочного прогнозирования. Авторы статьи описывают как традиционную линейную, так и инновационную нелинейную модели, сравнивая различные подходы, чтобы показать важность последней в прогнозировании таких сложных процессов, как сезонное распространение инфекционных заболеваний.

David Farrow, "Modeling the Past, Present, and Future of Influenza," doctoral thesis, Computational Biology Department, School of Computer Science, Carnegie Mellon University, 2016, `https://perma.cc/96CZ-5SX2`

В этой диссертации подробно рассматриваются некоторые теоретические и практические аспекты совместного использования социальных сетей в различной географической и временной привязанности для получения одного из наиболее важных прогнозов по распространению гриппа группы Delphi. Она содержит чрезвычайно подробное описание концепций прогнозирования заболеваемости гриппом, подтверждаемых специалистами по вирусологии, динамикой численности населения и многими научными экспериментами, призванными показать, как обработка данных влияет на прогнозирование.

Rob J. Hyndman, "Dynamic Regression," lecture notes, n.d., `https://perma.cc/5TPY-PYZS`

Лекционные заметки, в которых рассматриваются практические примеры использования методов динамической регрессии для совмещения традиционных статистических моделей прогнозирования с рядом альтернативных моделей описания сезонности в случаях, когда модель SARIMA оказывается несостоятельной ввиду того, что периодическая составляющая процесса слишком сложна или характеризуется слишком большими периодами для рассматриваемого объема данных или производительности вычислительного оборудования.

ГЛАВА 14

Финансовые приложения

Финансовые рынки считаются праматерью всех временны*х* рядов. Оплачивая доступ к данным о торговых сделках на высокотехнологичных биржевых площадках, будьте готовы обрабатывать терабайтные потоки данных, на что может уйти несколько дней даже с учетом преимуществ современного компьютерного оборудования и технологий распараллеливания вычислений.

Высокочастотные трейдеры — новые и печально известные члены финансового сообщества, обменивающиеся информацией и знаниями, полученными в результате анализа временны*х* рядов, на микросекундном уровне. Несмотря ни на что, традиционные финансовые организации, ориентирующиеся на работу с долгосрочными временны*ми* рядами, собранными в течение нескольких часов, дней или даже месяцев, продолжают добиваться успеха в биржевых торгах, демонстрируя, что анализ временны*х* рядов финансовых данных можно успешно проводить множеством способов и во временны*х* масштабах, охватывающих несколько порядков — от миллисекундного до годичного.

Естественно параллельными называются задачи обработки данных, в которых результаты обработки одного сегмента данных никоим образом не зависят от значений другого сегмента данных. В таких случаях задачи анализа данных легко выполнять параллельно, а не последовательно, используя очевидные преимущества многоядерных или многомашинных вычислений.

Рассмотрим, например, задачу вычисления среднесуточной доходности определенной акции, регистрируемой раз в минуту. Каждый день можно рассматривать отдельно, а несколько дней — параллельно. Напротив, вычисление экспоненциально взвешенного скользящего среднего значения дневной волатильности не является естественно параллельным процессом, поскольку значение для определенного дня зависит от значений за предыдущие дни. Иногда задачи, которые не являются естественно параллельными, все же могут выполняться частично параллельно, что зависит от их условий.

Рассмотрим классический пример анализа временны*х* рядов для прогнозирования доходности фондового индекса S&P 500 для следующего биржевого дня.

Получение и изучение финансовых данных

Получить финансовые данные может быть чрезвычайно сложно, особенно для строго заданного инструмента или временного разрешения. В таких случаях данные приходится покупать. Однако исторические данные о ценах на акции широко предоставляются различными службами.

- Yahoo Finance. Хотя компания Yahoo прекратила обслуживание программного интерфейса публикации финансовых данных,[1] ежедневные исторические биржевые данные все еще доступны для загрузки (`https://perma.cc/RQ6D-U4JX`).
- Новые игроки на финансовом рынке, такие как AlphaVantage (`https://www.alphavantage.co/`) и Quandl (`https://www.quandl.com/`), предоставляют как исторические, так и потоковые данные о ценах на акции.

Мы ограничимся анализом общедоступных ежедневных данных о стоимости биржевого индекса S&P 500, предоставляемых компанией Yahoo. Будем использовать данные о котировках индекса S&P 500 за период с 1990 по 2019 год. Следующий код позволяет ознакомиться со столбцами загруженного набора данных и строит график стоимости индекса на конец биржевого дня, с которого начинает анализ данных (рис. 14.1).

```
## Python
>>> df = pd.read_csv("sp500.csv")
>>> df.head()
>>> df.tail()
>>> df.index = df.Date
>>> df.Close.plot()
```

Можно видеть, что значения в начале и в конце периода дат, охватываемого файлом CSV, заметно различаются. Изменение значений становится более очевидным при построении полного временного ряда цены на момент закрытия (рис. 14.2) по сравнению с выборками из фрейма данных.

На рис. 14.2 показано, что временной ряд не является стационарным. Также на графике представлены разные "режимы" ценообразования. Исследуя причины возникновения таких режимов, четко просматриваемых на графике, финансовые аналитики стремятся разработать модели, описывающие временное смещение цен на акции. Похоже на то, что поведение данных описывается

[1] Кстати, вследствие этого в сообществах R и Python многие программы утратили работоспособность.

несколькими разными режимами, хотя у нас нет четкого понимания, в какие временные моменты заканчиваются одни и начинаются другие режимы.[2]

```
In [9]: df.head()
```

Out[9]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	1990-02-13	330.079987	331.609985	327.920013	331.019989	331.019989	144490000
1	1990-02-14	331.019989	333.200012	330.640015	332.010010	332.010010	138530000
2	1990-02-15	332.010010	335.209991	331.609985	334.890015	334.890015	174620000
3	1990-02-16	334.890015	335.640015	332.420013	332.720001	332.720001	166840000
4	1990-02-20	332.720001	332.720001	326.260010	327.989990	327.989990	147300000

```
In [75]: df.tail()
```

Out[75]:

	Date	Open	High	Low	Close	Adj Close	Volume
7301	2019-02-06	2735.050049	2738.080078	2724.149902	2731.610107	2731.610107	3472690000
7302	2019-02-07	2717.530029	2719.320068	2687.260010	2706.050049	2706.050049	4099490000
7303	2019-02-08	2692.360107	2708.070068	2681.830078	2707.879883	2707.879883	3622330000
7304	2019-02-11	2712.399902	2718.050049	2703.790039	2709.800049	2709.800049	3361970000
7305	2019-02-12	2722.610107	2748.189941	2722.610107	2744.729980	2744.729980	3827770000

Рис. 14.1. *Начальный и конечный фрагменты файла CSV с исходными данными. Обратите внимание, насколько сильно изменились значения с 1990 по 2019 год: такая ситуация не вызовет удивления только у финансовых специалистов*

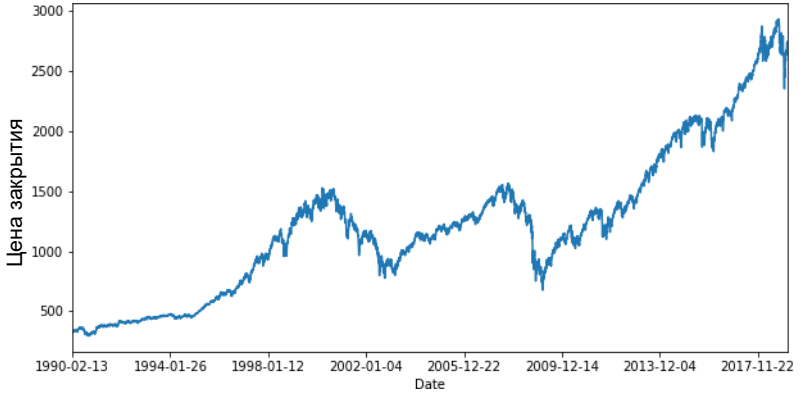


Рис. 14.2. *Дневные цены индекса S&P 500 на момент закрытия торгов не образуют стационарный временной ряд*

[2]Обратите внимание, что индекс S&P 500 довольно сложный, потому что рассчитывается через стоимость совокупности большого количества акций с периодически корректируемыми весами и полностью проприетарным коэффициентом, который также используется для деления средневзвешенных акций. По этой причине глубокое знание предметной области и понимание того, как различные действия, предпринимаемые компаниями, могут повлиять на цены акций и весовые коэффициенты в индексе S&P 500, крайне важны для понимания долгосрочного поведения котировок, которое продемонстрировано на рис. 14.2.

Вариативность положения точек и режимов на графике предполагает, что исходный набор данных целесообразно разделить на несколько частичных наборов, каждый из которых будет моделироваться отдельно. Однако нам нужно сохранить полностью все доступные данные, поскольку дневные данные не дают много точек даже за несколько десятилетий. Следует хорошо подумать, будет ли оправданным хранение всех данных в задачах составления прогноза всего на один день вперед.

Нужно выяснить, поможет ли нормализация сопоставить данные из разных периодов времени. Давайте изучим разнообразие масштабированных недельных цен на момент закрытия торгов за три разных десятилетия, содержащихся во временном ряду (рис. 14.3).

```
## Python
>>> ## Выбираем три недели (с понедельника по пятницу) из разных лет
>>> ## Масштабирование цен закрытия для каждого дня по недельной
>>> ## средней цене закрытия
>>> ## 1990
>>> vals = df['1990-05-07':'1990-05-11'].Close.values
>>> mean_val = np.mean(vals)
>>> plt.plot([1, 2, 3, 4, 5], vals/mean_val)
>>> plt.xticks([1, 2, 3, 4, 5],
>>>     labels = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'])
>>>
>>> ## 2000
>>> vals = df['2000-05-08':'2000-05-12'].Close.values
>>> mean_val = np.mean(vals)
>>> plt.plot([1, 2, 3, 4, 5], vals/mean_val)
>>> plt.xticks([1, 2, 3, 4, 5],
>>>    labels = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'])
>>>
>>> ## 2018
>>> vals = df['2018-05-07':'2018-05-11'].Close.values
>>> mean_val = np.mean(vals)
>>> plt.plot([1, 2, 3, 4, 5], vals/mean_val)
>>> plt.xticks([1, 2, 3, 4, 5],
>>>    labels = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday'])
```

Графики отображают цены на момент закрытия за три недели для трех разных десятилетий, каждое из которых масштабировано по отношению к среднему недельному значению, как реализовано в предыдущем коде. Относительные (процентные) изменения дневной цены в течение недели кажутся примерно одинаковыми для всех анализируемых десятилетий.

Графики содержат обнадеживающие сведения. Хотя средние значения и дисперсия цены на момент закрытия существенно изменяются во времени, после нормализации данных к среднему значению для заданного десятилетия графики демонстрируют одинаковое поведение показателей в разные периоды времени.

Учитывая этот факт, попытаемся найти способ сделать данные за весь период времени достаточно однородными, чтобы их можно было использовать

для полноценного обучения модели. Попробуем найти способ преобразования, который сохраняет финансовую значимость данных и в то же время позволяет сопоставлять их за весь анализируемый период времени.

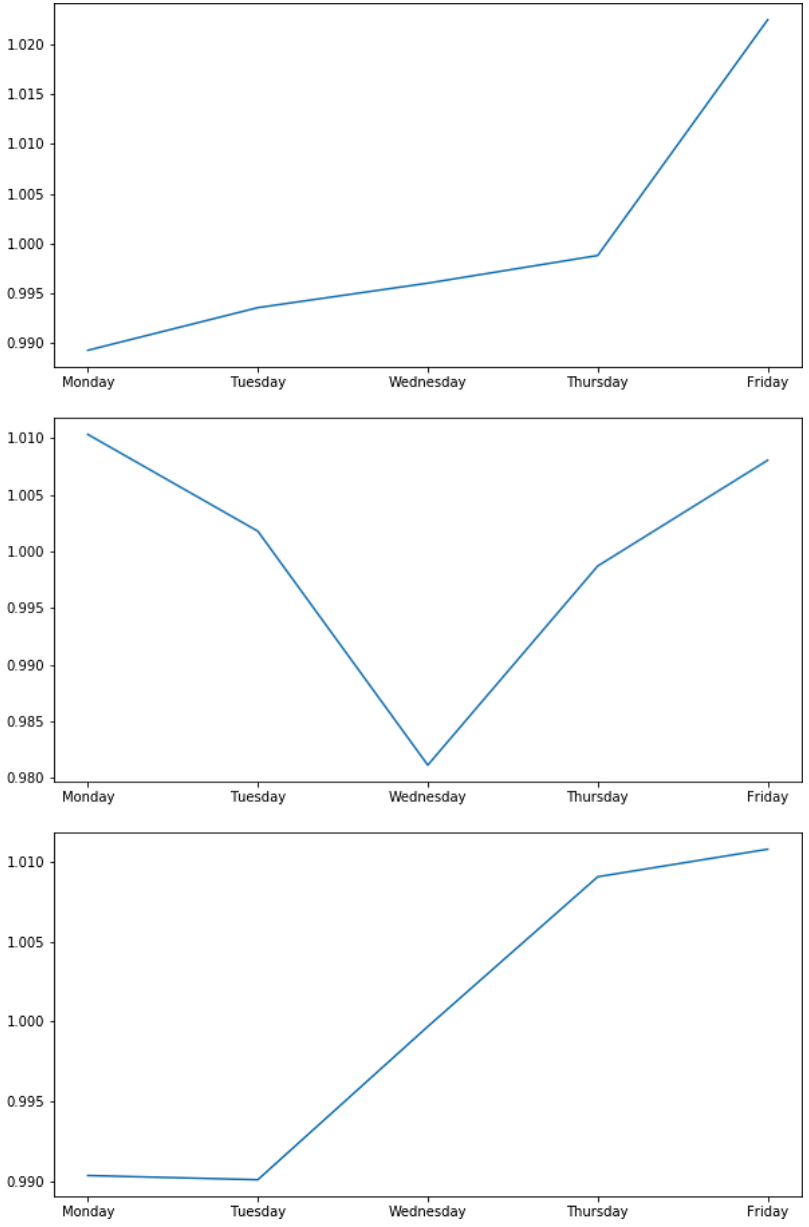


Рис. 14.3. *Масштабированные дневные цены на момент закрытия индекса S&P 500 в мае 1990, 2000 и 2018 годов*

Вычислим ежедневную доходность, т.е. изменение цены с начала по конец каждого торгового дня (рис. 14.4).

```
## Python
>>> df['Return'] = df.Close - df.Open
>>> df.Return.plot()
```

Как показано на рис. 14.4, одного этого недостаточно, чтобы сделать данные сопоставимыми. Также нужно найти способ нормализовать данные без упреждения, чтобы значения, подаваемые на вход и выход модели, были более однородными в течение интересующего периода времени. О том, как это можно сделать, речь пойдет в следующем разделе.

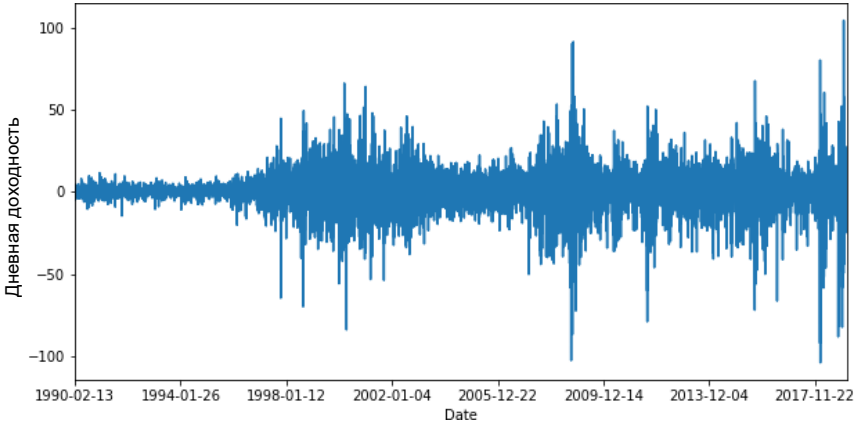


Рис. 14.4. *Дневная доходность во времени имеет почти нулевое математическое ожидание, но ее дисперсия в разные периоды времени заметно изменяется. Такое поведение стало мотиватором разработки моделей, подобных GARCH, кратко описанной в главе 6*

Финансовые рынки как случайное блуждание

Случайное блуждание — хорошая стартовая модель для описания финансовых временных рядов. Действительно, цены на акции часто приводятся в качестве наглядного примера естественного случайного блуждания. При описании наших данных с его помощью следует помнить о двух важных моментах.

Во-первых, многие смены режимов могут быть просто следствием естественных процессов модели случайного блуждания. Если не верите, рассмотрите некоторые из следующих примеров, которые были реализованы на языке R (рис. 14.5).

```
## R
> ## Графики строятся для каждого из начальных значений
> ## set.seed(1)
> ## set.seed(100)
> ## set.seed(30)
```

```
> N <- 10000
> x <- cumsum(sample(c(-1, 1), N, TRUE))
> plot(x)
```

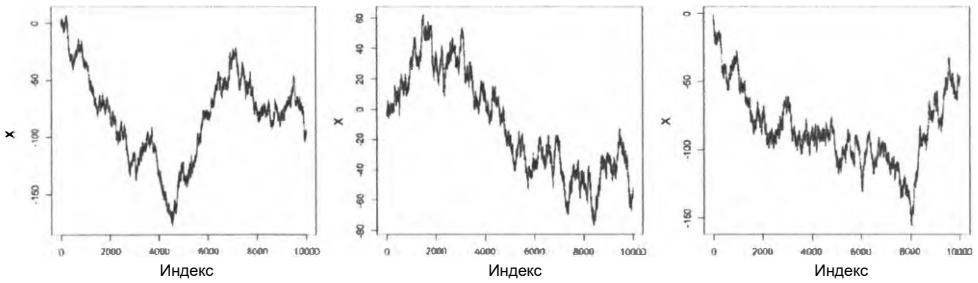


Рис. 14.5. *Результаты случайного блуждания с разными начальными значениями*

Во-вторых, поскольку случайное блуждание описывает данные котировок естественным образом, нашей базовой моделью, с которой сравниваются другие, более сложные, модели, будет та, которая позволяет спрогнозировать цену на завтра по сегодняшней цене, исходя из понимания фондового рынка как случайного блуждания. В подобном случае ожидается очень сильная корреляция.

```
## R
> cor(x, shift(x), use = "complete.obs")
[1] 0.9996601
```

С другой стороны, при таком построении разностных временных рядов, при котором значение на каждом временном шаге представляет изменение значения временного ряда от одного временного шага к другому, корреляция не проявлялась бы; она предопределяется общим трендом, а не фактической прогнозирующей способностью простой модели.

```
## R
> cor(diff(x), shift(diff(x)), use = "complete.obs")
[1] -0.005288012
```

Таким образом, в финансовом анализе, независимо от того, создается модель с высокой или низкой корреляцией, многое зависит от типа данных, подлежащих моделированию. При моделировании доходности (что мы и будем делать), а не цен на акции (распространенная ошибка даже в отрасли) корреляции в данных будут заметно меньшими, а сами модели при решении реальных задач с большей вероятностью будут скорее прогностическими, чем многообещающими, но не имеющими практической ценности. Именно поэтому в следующем примере мы будем рассматривать модель описания доходности, а не цен на акции.

Предварительная обработка финансовых данных для глубокого обучения

Предварительная обработка данных будет выполняться в три этапа.

1. Формирование новых, экономически значимых величин из исходных данных.
2. Вычисление экспоненциально взвешенного скользящего среднего и дисперсии целевых величин для масштабирования данных без упреждения.
3. Упаковка полученных результатов в формат, подходящий для целевой модели глубокого обучения, для обеспечения целостности данных.

Анализ финансовых временных рядов — отдельная научная дисциплина

Анализ финансовых временны*х* рядов — это отдельная научная дисциплина. Тысячи ученых прилагают максимум усилий, чтобы понять механизм функционирования финансовых рынков, как с целью максимизации прибыли, так и с точки зрения оптимального регулирования. Существует множество статистических моделей, разработанных для описания отдельных сложных финансовых механизмов, с которыми вам уже доводилось встречаться, например GARCH. Если вы планируете применять методы статистического и машинного обучения к моделированию финансовых временны*х* рядов, то обязательно познакомьтесь с историей финансовой математики и основными классами наиболее распространенных моделей.

Добавление новых величин к исходным данным

В предыдущем разделе мы уже вычисляли ежедневную доходность. Другой количественный показатель, который можно вычислить по исходным данным, — это ежедневная волатильность, которая представляет собой разницу между самой высокой и самой низкой ценами, зарегистрированными в течение торгового дня. Она легко рассчитывается по необработанным данным (рис. 14.6).

```
## Python
>>> df['DailyVolatility'] = df.High - df.Low
>>> df.DailyVolatility.plot()
```

Как ежедневная доходность, так и ежедневная волатильность представлены нестационарными временны*ми* рядами. Это еще раз подтверждает, что их нужно соответствующим образом масштабировать перед дальнейшей обработкой, чтобы исключить упреждение.

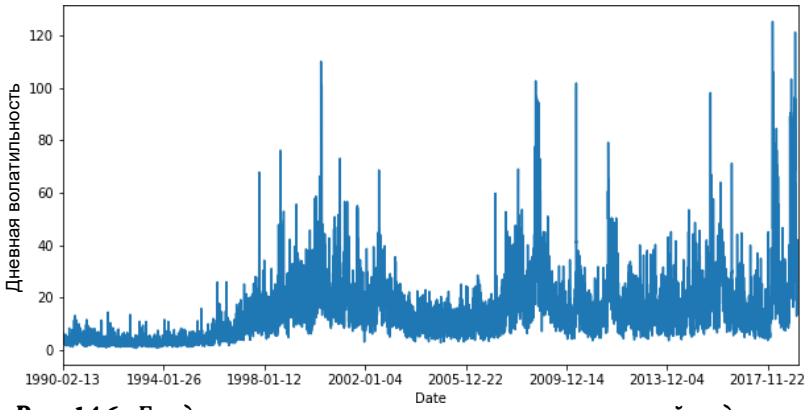


Рис. 14.6. *Ежедневная волатильность — это временной ряд положительных значений (по определению), характеризующийся сильно изменяющейся дисперсией в разных точках временного ряда S&P 500*

Масштабирование целевых величин без упреждения

Нашей целью будет прогнозирование дневной доходности на один день вперед. Перечислим основные интересующие нас величины.

- Предыдущая дневная доходность.
- Предыдущая дневная волатильность.
- Предыдущий дневной объем торгов.

Для масштабирования этих величин будем вычитать из каждой из них экспоненциально взвешенное скользящее среднее, а затем делить полученную разность на экспоненциально взвешенное стандартное отклонение. Наше предыдущее исследование еженедельных данных показало, что при соответствующей предварительной обработке многие целевые показатели будут формировать стационарные временные ряды.

Сначала вычислим экспоненциально взвешенное скользящее среднее каждого столбца в фрейме данных и построим график экспоненциально взвешенного скользящего среднего (AWMA) дневной волатильности (рис. 14.7). Сравните его с графиком на рис. 14.6. Он имеет намного более гладкий характер из-за усреднения. Обратите внимание, что форма графика описывается параметром, который можно рассматривать как гиперпараметр целевой модели даже на этапе предварительной обработки данных: период полураспада экспоненциального сглаживания. Поведение модели, безусловно, сильно зависит от значения этого параметра.

```
## Python
>>> ewdf = df.ewm(halflife = 10).mean()
>>> ewdf.DailyVolatility.plot()
```

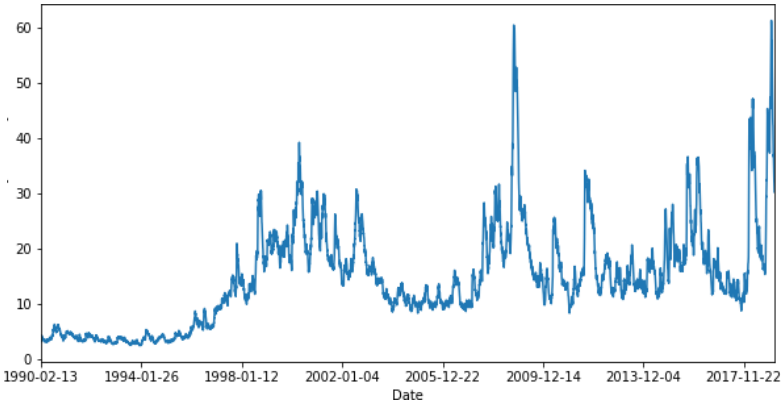


Рис. 14.7. *График экспоненциально взвешенного скользящего среднего ежедневной волатильности имеет более гладкий характер, чем график исходных данных, но все еще не представлен стационарным временным рядом*

Теперь, используя это значение, а также экспоненциально взвешенную скользящую дисперсию, вычисленную здесь же, можно переходить к масштабированию интересующих значений, чтобы получить ряды, которые демонстрируют более согласованное временное поведение (рис. 14.8).

```
## Python
>>> ## Вычисление экспоненциально взвешенной скользящей дисперсии
>>> vewdf = df.ewm(halflife = 10).var()
>>>
>>> ## Масштабирование по среднему и нормализация
>>> scaled = df.DailyVolatility - ewdf.DailyVolatility
>>> scaled = scaled / vewdf.DailyVolatility**0.5
>>> scaled.plot()
```

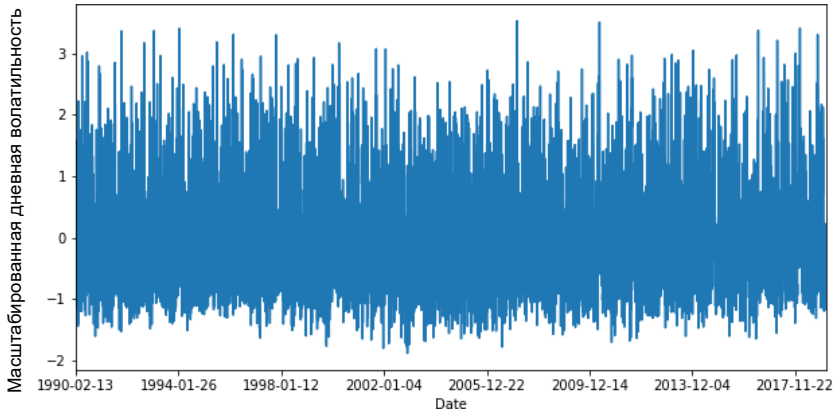


Рис. 14.8. *Нормализация данных с помощью экспоненциально взвешенного среднего значения и дисперсии позволяет получить более однородные временные ряды по сравнению со значениями, полученными за период с 1990 по 2019 год*

Преобразуем все три целевых входных ряда в их масштабированные версии, воспользовавшись таким кодом.

```
## Python
>>> df['ScaledVolatility'] = ((df.DailyVolatility -
>>>                            ewdf.DailyVolatility)
>>>                            /vewdf.DailyVolatility**0.5)
>>> df['ScaledReturn'] = ((df.Return - ewdf.Return)
>>>                            / vewdf.Return**0.5)
>>> df['ScaledVolume'] = ((df.Volume - ewdf.Volume)
>>>                            / vewdf.Volume**0.5)
```

Наконец, отбросим значения NA, полученные при экспоненциальном сглаживании.[3]

```
## Python
>>> df = df.dropna()
```

Необязательно использовать дневные данные

Надеюсь, вы заметили, что обработка данных проводилась для значений одного дня. Как указывалось в главах 2 и 3, увеличивая и уменьшая частоту выборки данных, можно изменить размер временной шкалы анализа. Более того, в нашей модели можно использовать входные данные из разных временных шкал. Например, в качестве разных входных данных можно использовать экспоненциально взвешенные скользящие средние и дисперсии для различных временных периодов. Кроме того, можно было бы вычислить интересующие нас показатели, такие как еженедельная доходность или ежемесячная волатильность, в разных временных масштабах. Все эти возможности нужно обязательно изучать при построении модели по данным временных рядов.

Желательно, чтобы временные шкалы исходных данных и прогнозируемых величин имели одинаковую глубину исследования. Чем дальше вы хотите заглянуть в будущее, тем больше старых данных нужно учитывать и тем более длинный временной период использовать для сглаживания ряда при распознавании долгосрочных трендов. Только так можно получить достоверный долгосрочный прогноз.

Форматирование данных для нейронной сети

В настоящее время целевые данные находятся в фрейме данных Pandas, а запланированные входные данные хранятся вместе со многими необработанными входными данными, которые не планируется использовать. Кроме того, перед

[3] Вместо того чтобы отбрасывать их, можно приравнять экспоненциально сглаженные значения всего столбца к единственному значению, известному в текущее время.

передачей нейронной сети исходные данные нужно преобразовать в формат TNC, который, как вы помните, представляется как *время* × *количество выборок* × *каналы*. Следовательно, потребуется выполнить специальную предварительную обработку даже масштабированных данных, которые были получены ранее.

Сначала разбиваем данные на наборы для обучения и тестирования.[4]

```
## Python
>>> ## Разделение данных на наборы для обучения и тестирования
>>> train_df = df[:7000]
>>> test_df = df[7000:]
>>>
>>> ## Создание переменных конвейера из обучающих данных;
>>> ## выбираем только целевые значения из больших фреймов данных
>>> horizon = 10
>>> X = train_df[:(7000 - horizon)][["ScaledVolatility", "ScaledReturn",
                                     "ScaledVolume"]].values
>>> Y = train_df[horizon:]["ScaledReturn"].values
```

Обратите внимание на проблему зависимости результата прогнозирования от способа задания величины `Y`. Задумайтесь об этом прежде, чем переходить к последующему материалу.

Выхолащивание задачи прогнозирования

То, что мы сделали с нашими данными, выхолостило задачу прогнозирования. Мы берем будущее значение и "разбавляем" его информацией из прошлого. Хотя такой перенос прошлого в будущее, казалось бы, представляет собой противоположность упреждению, это связанные проблемы. Информация может рассматриваться в последовательности, отличной от задаваемой временной осью, что, в свою очередь, приводит к неправильному пониманию точности модели. Нужно помнить о том, каким преобразованиям подвергались данные на этапе предварительной обработки, и принимать это в расчет при оценке модели.

Проблема прогнозирования сглаженных значений широко освещена в экономической и финансовой литературе и часто остается неучтенной. Хотя она и не является критической, иногда авторы преувеличивают полученные результаты, утверждая, что им удалось превзойти эталонный тест, в то время, как на самом занимались решением более простой задачи — прогнозирования сглаженного, а не измеренного значения.

[4] Не забывайте о необходимости выделения набора данных для проверки, чтобы избежать утечки информации из набора для тестирования. Здесь он не создается, чтобы сохранить код предельно простым.

Проблема с этой настройкой в том, что значение Y — это масштабированная, а не фактическая доходность. Ее использование облегчает обучение, обеспечивая попадание значений в соответствующий диапазон, но при этом величина Y, прогнозированием которой мы занимаемся, — это не целевая доходность, а доходность, скорректированная с помощью скользящего среднего. На самом деле прогнозированию подлежит не абсолютная доходность, а показатель отличия значения доходности от экспоненциально взвешенной скользящей средней.

В таком подходе нет ничего предосудительного, но он предусматривает некоторое упрощение истинной задачи прогнозирования. Если результаты обучения превосходят реальную точность модели, то это отчасти объясняется тем, что обучение ведется с оглядкой на гибридную задачу, а действительная ценность модели будет зависеть только от результатов истинного прогнозирования.

Сосредоточимся на данных обучения и переведем величину X в формат, поддерживаемый целевой архитектурой нейронных сетей, а именно — TNC. Применим для этого ряд инструментов пакета NumPy.

Изначально переменная X является двумерной, что следует из фрейма данных Pandas. Мы хотим добавить в нее третье измерение, ось 1 (тем самым второе измерение смещает второе измерение на ось 2; оси нумеруются с 0).

```
## Python
>>> X = np.expand_dims(X, axis = 1)
```

Наша временная ось имеет номер 0, потому что фрейм данных уже отсортирован по времени. Последняя ось с номером 2 рассматривается как ось канала, поскольку каждый из входов занимает один столбец этого измерения.

Обратимся к модели, которая способна наблюдать 10 временных шагов, — 10 дней в прошлое. Таким образом, нам нужно обрезать ось 0 так, чтобы сделать ее длину равной 10. Отрежем вдоль оси 0 каждые 10 строк и переформируем результирующий список подматриц таким образом, чтобы количество выборок (т.е. длина результирующего списка) стало измерением второй оси.[5]

```
## Python
>>> X = np.split(X, X.shape[0]/10, axis = 0)
>>> X = np.concatenate(X, axis = 1)
>>> X.shape
(10, 699, 3)
```

Итак, у нас есть временной ряд длиной 10 с тремя параллельными входами для данных в формате TNC. Для них у нас есть 699 примеров. Размер пакета будет определять длину эпохи, где эпоха — это один цикл в наших данных.

Данных для обучения не так уж и много, учитывая небольшое количество примеров. Как мы перешли от данных за 30 лет к такому небольшому количеству

[5] Для пользователей языка R: помните, что в Python отсчет начинается с 0, поэтому вторая ось — это ось 1, а не 2.

значений? Ответ заключается в том, что в текущий момент каждая точка данных включена только в один временной ряд выборки. Однако каждая точка данных может находиться в 10 разных временных рядах, занимая разные позиции в каждом из них.

Это может быть неочевидно, поэтому давайте рассмотрим простой пример. Рассмотрим следующий временной ряд.

1, 3, 5, 11, 3, 2, 22, 11, 5, 7, 9

Предположим, что с помощью этого временного ряда нужно обучить нейронную сеть, выбирая временное окно с длиной 3. Если придерживаться только что выполненной обработки данных, то можно получить такие примеры временных рядов.

- 1, 3, 5
- 11, 3, 2
- 22, 11, 5
- 7, 9, _

Тем не менее нет никаких оснований выбирать те или иные начало и конец каждого временного ряда. Окна произвольны. Например, допустимыми будут считаться также следующие временные ряды, вырезанные из исходного временного ряда.

- 3, 5, 11
- 2, 22, 11
- 5, 7, 9

Таким образом, если бы нам потребовалось больше данных, то можно было бы генерировать выборки временных рядов, просто перемещая окно по всему набору данных. Это привело бы к большему количеству отдельных выборок временных рядов по сравнению с методом разделения данных на непересекающиеся выборки. Имейте это в виду при подготовке собственных наборов данных. Ниже вы увидите, что метод скользящего окна будет использован для предварительной обработки данных.

Построение и обучение RNN

Как было обозначено во введении, финансовые временные ряды трудны для моделирования и понимания. Даже несмотря на то, что финансовая отрасль продолжает играть главенствующую роль в западной экономике, эксперты сходятся во мнении, что делать прогнозы в ней очень сложно. Следовательно, нам

нужен метод прогнозирования, который хорошо подходит для сложных систем с потенциально нелинейной динамикой, а именно — нейронная сеть с глубоким обучением. Однако из-за недостатка данных остановимся на простой архитектуре рекуррентных нейронных сетей (LSTM) и режиме обучения, описываемом следующими параметрами.

```
## Python
>>> ## Параметры архитектуры
>>> NUM_HIDDEN = 4
>>> NUM_LAYERS = 2
>>>
>>> ## Параметры форматирования данных
>>> BATCH_SIZE = 64
>>> WINDOW_SIZE = 20
>>>
>>> ## Параметры обучения
>>> LEARNING_RATE = 1e-2
>>> EPOCHS = 30
```

В отличие от главы 10, здесь будет использован пакет `TensorFlow`, а не `MXNet`, — вы получаете возможность познакомиться с еще одной широко используемой средой глубокого обучения. В `TensorFlow` определяются переменные для всех величин, с которыми работает сеть, даже с изменяющимися значениями, представляющими входные данные. Входные данные задаются с помощью метода `placeholder()`, определяющего форму графа.

```
## Python
>>> Xinp = tf.placeholder(dtype = tf.float32,
>>> shape = [WINDOW_SIZE, None, 3])
>>> Yinp = tf.placeholder(dtype = tf.float32, shape = [None])
```

Построим сеть и реализуем этап вычисления и оптимизации потерь.

```
## Python
>>> with tf.variable_scope("scope1", reuse=tf.AUTO_REUSE):
>>>     cells = [tf.nn.rnn_cell.LSTMCell(num_units=NUM_HIDDEN)
>>>                                 for n in range(NUM_LAYERS)]
>>>     stacked_rnn_cell = tf.nn.rnn_cell.MultiRNNCell(cells)
>>>     rnn_output, states = tf.nn.dynamic_rnn(stacked_rnn_cell,
>>>                                            Xinp,
>>>                                            dtype=tf.float32)
>>>     W = tf.get_variable("W_fc", [NUM_HIDDEN, 1],
>>>                         initializer =
>>>                         tf.random_uniform_initializer(-.2, .2))
>>>
>>>     ## Смещения нет, потому что ожидаемое среднее равно нулю
>>>     output = tf.squeeze(tf.matmul(rnn_output[-1, :, :], W))
>>>
>>>     loss = tf.nn.l2_loss(output - Yinp)
```

```
>>>     opt = tf.train.GradientDescentOptimizer(LEARNING_RATE)
>>>     train_step = opt.minimize(loss)
```

Здесь применяется довольно сложный способ ввода данных из-за того, что каждая точка данных должна присутствовать в нескольких временных рядах в зависимости от заданного смещения. Здесь нам приходится иметь дело с проблемой форматирования данных, подробно описанной в главе 10.

```
## Python
>>> ## Для каждой эпохи
>>> y_hat_dict = {}
>>> Y_dict = {}
>>>
>>> in_sample_Y_dict = {}
>>> in_sample_y_hat_dict = {}
>>>
>>> for ep in range(EPOCHS):
>>>    epoch_training_loss = 0.0
>>>    for i in range(WINDOW_SIZE):
>>>    X = train_df[:(7000 - WINDOW_SIZE)][["ScaledVolatility",
>>>                                         "ScaledReturn",
>>>                                         "ScaledVolume"]].values
>>>    Y = train_df[WINDOW_SIZE:]["ScaledReturn"].values
>>>
>>>    ## Обеспечение делимости на размер окна
>>>    num_to_unpack = math.floor(X.shape[0] / WINDOW_SIZE)
>>>    start_idx = X.shape[0] - num_to_unpack * WINDOW_SIZE
>>>    X = X[start_idx:]
>>>    Y = Y[start_idx:]
>>>
>>>    X = X[i:-(WINDOW_SIZE-i)]
>>>    Y = Y[i:-(WINDOW_SIZE-i)]
>>>
>>>    X = np.expand_dims(X, axis = 1)
>>>    X = np.split(X, X.shape[0]/WINDOW_SIZE, axis = 0)
>>>    X = np.concatenate(X, axis = 1)
>>>    Y = Y[::WINDOW_SIZE]
>>>    ## Обучение
>>>    ## Разделение данных на пакеты и запуск сеанса
>>>     for j in range(math.ceil(Y.shape[0] / BATCH_SIZE)):
>>>        ll = BATCH_SIZE * j
>>>        ul = BATCH_SIZE * (j + 1)
>>>
>>>        if ul > X.shape[1]:
>>>           ul = X.shape[1] - 1
>>>           ll = X.shape[1]- BATCH_SIZE
>>>
>>>        training_loss, _, y_hat = sess.run([loss, train_step,
>>>                                    output],
```

```
>>>                                     feed_dict = {
>>>                                          Xinp: X[:, ll:ul, :],
>>>                                            Yinp: Y[ll:ul]
>>>                                     })
>>>         epoch_training_loss += training_loss
>>>
>>>         in_sample_Y_dict[ep] = Y[ll:ul]
>>>         ## Последняя часть данных
>>>         in_sample_y_hat_dict[ep] = y_hat
>>>
>>>     ## Тестирование
>>>     X = test_df[:(test_df.shape[0] - WINDOW_SIZE)]
>>>                     [["ScaledVolatility", "ScaledReturn",
>>>                       "ScaledVolume"]].values
>>>     Y = test_df[WINDOW_SIZE:]["ScaledReturn"].values
>>>     num_to_unpack = math.floor(X.shape[0] / WINDOW_SIZE)
>>>     start_idx = X.shape[0] - num_to_unpack * WINDOW_SIZE
>>>     ## Лучше отбросить начало, чем конец периода обучения
>>>     X = X[start_idx:]
>>>     Y = Y[start_idx:]
>>>
>>>     X = np.expand_dims(X, axis = 1)
>>>     X = np.split(X, X.shape[0]/WINDOW_SIZE, axis = 0)
>>>     X = np.concatenate(X, axis = 1)
>>>     Y = Y[::WINDOW_SIZE]
>>>     testing_loss, y_hat = sess.run([loss, output],
>>>                               feed_dict = { Xinp: X, Yinp: Y })
>>>     ## Не лучшее решение — проверку проводить отдельно
>>>     ## от тестирования
>>>
>>>     print("Epoch: %d Training loss: %0.2f
>>>           Testing loss %0.2f:" %
>>>           (ep, epoch_training_loss, testing_loss))
>>>     Y_dict[ep] = Y
>>>     y_hat_dict[ep] = y_hat
```

Рассмотрим результаты обучения и тестирования.

```
Epoch: 0 Training loss: 2670.27 Testing loss 526.937:
Epoch: 1 Training loss: 2669.72 Testing loss 526.908:
Epoch: 2 Training loss: 2669.53 Testing loss 526.889:
Epoch: 3 Training loss: 2669.42 Testing loss 526.874:
Epoch: 4 Training loss: 2669.34 Testing loss 526.862:
Epoch: 5 Training loss: 2669.27 Testing loss 526.853:
Epoch: 6 Training loss: 2669.21 Testing loss 526.845:
Epoch: 7 Training loss: 2669.15 Testing loss 526.839:
Epoch: 8 Training loss: 2669.09 Testing loss 526.834:
Epoch: 9 Training loss: 2669.03 Testing loss 526.829:
Epoch: 10 Training loss: 2668.97 Testing loss 526.824:
Epoch: 11 Training loss: 2668.92 Testing loss 526.819:
```

```
Epoch: 12 Training loss: 2668.86 Testing loss 526.814:
Epoch: 13 Training loss: 2668.80 Testing loss 526.808:
Epoch: 14 Training loss: 2668.73 Testing loss 526.802:
Epoch: 15 Training loss: 2668.66 Testing loss 526.797:
Epoch: 16 Training loss: 2668.58 Testing loss 526.792:
Epoch: 17 Training loss: 2668.49 Testing loss 526.788:
Epoch: 18 Training loss: 2668.39 Testing loss 526.786:
Epoch: 19 Training loss: 2668.28 Testing loss 526.784:
Epoch: 20 Training loss: 2668.17 Testing loss 526.783:
Epoch: 21 Training loss: 2668.04 Testing loss 526.781:
Epoch: 22 Training loss: 2667.91 Testing loss 526.778:
Epoch: 23 Training loss: 2667.77 Testing loss 526.773:
Epoch: 24 Training loss: 2667.62 Testing loss 526.768:
Epoch: 25 Training loss: 2667.47 Testing loss 526.762:
Epoch: 26 Training loss: 2667.31 Testing loss 526.755:
Epoch: 27 Training loss: 2667.15 Testing loss 526.748:
Epoch: 28 Training loss: 2666.98 Testing loss 526.741:
Epoch: 29 Training loss: 2666.80 Testing loss 526.734:
```

Выбранная оценка ошибки не дает представления о том, насколько хорошо целевые общие данные соответствуют результатам, поэтому их лучше визуализировать. Рассчитаем как вневыборочную (очень важно), так и внутривыборочную (менее важно) точность, как показано на рис. 14.9.

```
## Python
>>> plt.plot(test_y_dict[MAX_EPOCH])
>>> plt.plot(test_y_hat_dict[MAX_EPOCH], 'r--')
>>> plt.show()
```

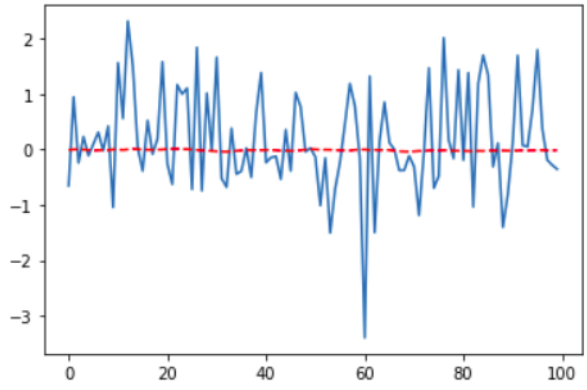


Рис. 14.9. *Фактические значения доходности за период тестирования (сплошная кривая) и прогноз, построенный нейронной сетью (пунктирная линия). Величина ошибки настолько велика, что модель нельзя считать приемлемой*

Легко видеть, что прогнозируемые значения доходности не совпадают с фактическими значениями. Определим корреляцию Пирсона следующим образом.

```
## Python
>>> pearsonr(test_y_dict[MAX_EPOCH], test_y_hat_dict[MAX_EPOCH])
(0.035957868817734l9, 0.20105107068949668)
```

Если вам не приходилось работать с финансовыми временны*ми* рядами, то полученные числа могут показаться удручающими. Как ни странно, в финансовой отрасли наша модель может найти достойное применение, даже несмотря на график и поведение *p*-значения. В финансах положительная корреляция является хорошим признаком, который можно постепенно улучшать. На самом деле немногие настолько сильные корреляции могут наблюдаться во многих исследовательских проектах.

Чтобы получить лучшее представление о том, следуют ли прогнозы в одном и том же направлении, нужно масштабировать предсказанную доходность на порядок и снова построить график (рис. 14.10).

```
## Python
>>> plt.plot(test_y_dict[MAX_EPOCH][:100])
>>> plt.plot(test_y_hat_dict[MAX_EPOCH][:100] * 50, 'r--')
>>> plt.show()
```

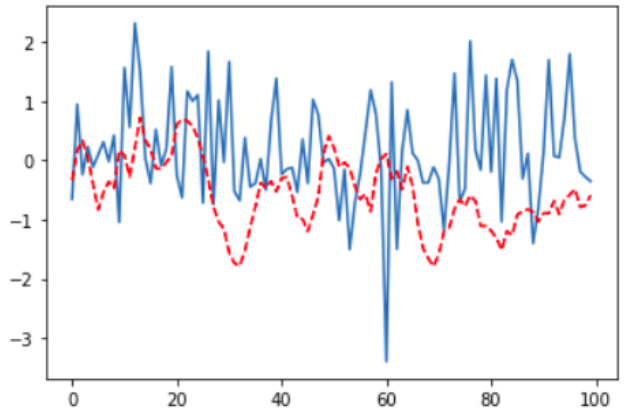


Рис. 14.10. *Более наглядное сравнение фактических значений доходности за период тестирования (сплошная кривая) и спрогнозированных нейронной сетью величин (пунктирная линия). При столь низкой корреляции вероятность распознавания ложных шаблонов чрезвычайно высока. Здесь количественные показатели оказываются полезнее визуальной оценки зашумленных финансовых данных*

Если изучали обзорные статьи об использовании глубокого обучения для анализа финансовых временных рядов, то вас ожидает сильное разочарование. Например, скорее всего, вам знакомы свидетельства того, что применение простой многослойной сети LSTM к дневным данным о ценах на акции позволяет получить прогнозы, практически идентичные фактическим данным фондового рынка — даже вне выборки. Существуют две важные причины, по которым такие результаты могут только выглядеть хорошо, но не являться таковыми на самом деле.

- Предварительная обработка кода, заключающаяся в его масштабировании с помощью специализированного готового решения, такого как `sklearn.preprocessing.MinMaxScaler`. Это не идеальный вариант, поскольку он предопределяет упреждение, вызванное масштабированием на данных за весь временной период.
- Прогнозирование цены, а не доходности. Это гораздо более простая задача — для начала отличным прогнозом цены в день $T + 1$ является цена в день T. Таким образом, легко построить модель, которая, по-видимому, достаточно хорошо прогнозирует цену и создает впечатляющие графики. К сожалению, такие модели не могут быть использованы для достижения успеха в биржевых торгах.

Мы попытались рассмотреть более реалистичный пример, показывающий, что задача намного сложнее, а графики не такие уж и хорошие, как кажется.

Вопросы производительности

В рассмотренном примере использована стандартная сеть LSTM `tf.nn.rnn_cell.LSTMCell` — не в последнюю очередь потому, что это универсальный рабочий инструмент. Кроме того, она хорошо подходит под наши цели, поскольку выполнение используемого кода не требует задействования графического процессора, и вы можете запускать его даже на стандартном ноутбуке.

Если ваш компьютер оснащен мощным графическим процессором, то обратитесь к программному интерфейсу `cuDNN` компании NVIDIA, который представляет надстройку для ее аппаратного обеспечения, поддерживающую работу с двумя основными вариантами стандартной сети RNN: LSTM и GRU. Если у вас стандартный графический процессор компании NVIDIA, то, скорее всего, обучение модели будет выполняться намного быстрее, чем в стандартной реализации. Предложенное решение доступно в фреймворке `TensorFlow`, а также в некоторых других популярных пакетах глубокого обучения. Принципы его

построения схожи с применяемыми нами, за исключением использования другой архитектуры RNN.

Этот совет может потерять свою актуальность после выпуска компанией Google процессоров TPU и других специализированных аппаратных решений для глубокого обучения.

Излишне говорить, что полный анализ производительности модели не завершен. Мы выяснили, как правильно строить модели для прогнозирования, на что нужно обращать внимание и как сделать так, чтобы модель глубокого обучения оправдала излишнюю сложность по сравнению с линейной моделью.

Существует множество способов повысить точность модели, которые следует рассматривать как советы по модификации кода ее реализации.

- Добавить больше входных данных из необработанных данных путем создания дополнительных признаков на основе таких входных данных. Мы использовали не все столбцы необработанных входных данных, и существуют другие способы представления целевых величин. Рассмотрите категориальные переменные, описываемые вопросом "Совпадают ли максимальные или минимальные дневные цены с моментами открытия или закрытия торгов?" (Для проверки нужно использовать несколько бинарных условий.)
- Интегрировать параллельные временны*е* ряды для других акций. Это предоставит дополнительную информацию и данные для обучения.
- Использовать данные в разных временны*х* масштабах. В одном широко цитируемом документе, посвященном такому подходу, описывается специально разработанная архитектура ClockworkRNN (`https://perma.cc/9C62-7GFK`).
- Дополнить входные данные, используя существующие примеры временны*х* рядов и добавляя флуктуации. Это поможет расширить набор данных.
- Позволить сетевой архитектуре расти при увеличении объема входных данных. Более сложная архитектура не всегда равнозначна высокой точности прогнозирования, но оказывается целесообразной, когда текущая сеть работает на пределе.
- Попробовать обучать данные в хронологическом порядке, а не методом циклического перебора несколько раз за эпоху. Иногда это оказывается весьма действенным (зависит от набора данных). Учитывая изменение поведения временны*х* рядов во времени, лучше закончить обучение на последних данных, чтобы отобразить измененное поведение в весах.

- Рассмотреть различные функции потерь. Мы использовали норму L_2, которая имеет тенденцию штрафовать большие различия намного сильнее, чем малые. Однако, учитывая предметную область, успех можно оценивать по-разному. Возможно, нужно всего лишь предсказать знак ежедневной доходности и не беспокоиться о его величине. В этом случае рассмотрите возможность применения целевых направлений в качестве категориальных переменных: положительное, отрицательное, ноль. В случае использования категориальных данных обычно рассматривается кросс-энтропийная мера потерь. Однако, учитывая, что предложенные данные ранжируются (ноль ближе к отрицательному значению, чем положительное значение к отрицательному) и их нельзя считать полностью категориальными, рассмотрите вариант применения пользовательской функции потерь, чтобы отобразить этот факт.
- Подумать о создании множества простых нейронных сетей, а не одной. Поддерживайте маленькой каждую отдельную сеть. Ансамбли оказываются полезными для данных с низким отношением "сигнал/шум", в частности для финансовых данных.
- Определить, почему шкала прогнозов сильно отличается от шкалы фактических значений. Начните с оценки того, является ли используемая функция потерь проблематичной, учитывая сильное преобладание дневной доходности с нулевым значением.

Как видите, существует множество способов повысить точность сети или расширить ее возможности. То, как это правильно сделать, зависит от набора данных. Не пренебрегайте визуализацией полученных сетью результатов, изучайте предметную область и предельно точно очерчивайте целевые задачи (в данном случае, скорее всего, максимизация доходности). В противном случае вы легко потеряетесь в невероятном разнообразии доступных вариантов.

Дополнительные источники

Joumana Ghosn and Yoshua Bengio, "Multi-Task Learning for Stock Selection," Cambridge: MIT Press, 1996, `https://perma.cc/GR7A-5PQ5`

Документ 1997 года, представляющий ранний пример применения нейронной сети к проблеме финансовых рынков. Автор использовал структуру, которая сегодня представляется достаточно простой сетью с минимальным объемом данных, и, как ни странно, обнаружил, что ее можно обучить на удивление удачному выбору биржевых позиций. Интересно то, что это один из самых ранних примеров многозадачного обучения.

Lawrence Takeuchi and Yu-Ying Lee, "Applying Deep Learning to Enhance Momentum Trading Strategies in Stocks," 2013, `https://perma.cc/GJZ5-4V6Z`

В этой статье авторы рассматривают созданную ими нейронную сеть с позиции "импульсного обучения" — традиционного способа количественного прогнозирования финансовых рынков до появления машинного обучения. В ней описаны принципы принятия решения по обучению и оценке точности модели.

"Is anyone making money by using deep learning in trading?" (Можно ли заработать деньги в биржевой торговле с помощью глубокого обучения?) Quora, `https://perma.cc/ Z8C9-V8FX`

В ответах на этот вопрос представлены различные мнения об успешности применения методов глубокого обучения в финансовых приложениях. Как показано в некоторых ответах, любое решение, позволяющее зарабатывать на финансовых рынках, в значительной степени защищено соглашениями о неразглашении и стимулом к получению прибыли, что не позволяет хотя бы приблизительно оценить его действительную точность. Ответы также указывают на существование широкого спектра потенциально успешных финансовых приложений — прогнозирование прибыли является лишь одной из многих отраслевых задач!

ГЛАВА 15

Временны*е* ряды в государственном управлении

Анализ временны*х* рядов представляет актуальную и важную задачу в государственном управлении по ряду причин. Во-первых, правительства как крупных, так и небольших государств выступают одними из наибольших мировых держателей важных данных, которые могут выступать источниками временны*х* рядов, включая отчеты о занятости населения США, данные о температуре океана (т.е. данные о глобальном потеплении) и статистику по совершенным преступлениям. Во-вторых, правительствами стран по определению предоставляется большое количество услуг, на которые существует общественный запрос, и поэтому им приходится выступать достаточно искусными прогнозистами, чтобы оптимизировать работу соответствующих государственных служб. Таким образом, в сфере государственного управления проявляются все без исключения аспекты обработки данных временны*х* рядов: хранение, очистка, изучение и прогнозирование.

Как было рассказано в главе 2 при извлечении временны*х* рядов из источников данных, находящихся в ведомстве правительственных организаций, они могут выглядеть как прошедшие некоторую реструктуризацию. Как правило, большинство правительственных наборов данных является результатом непрерывного сбора информации, а не накопления за отдельный, пусть и большой, временной период. Тем не менее правительственные наборы данных неудобны для обработки методами анализа временны*х* рядов по целому ряду причин.

- Непоследовательный учет (ввиду организационных ограничений или политических изменений).
- Непрозрачные или запутанные методы обработки данных.
- Огромные наборы данных с относительно низким содержанием целевой информации.

Тем не менее нам будет довольно интересно рассмотреть правительственные наборы данных как с концептуальной, так и с практической точек зрения. В этой главе мы сконцентрируем свое внимание на наборах данных, в которых собраны сведения обо всех жалобах, поступающих от жителей Нью-Йорка на горячую

телефонную линию (по номеру 311) с 2010 года по настоящее время[1] (`https://perma.cc/BXF6-BZ4X`). Поскольку набор данных постоянно обновляется, приведенные в книге данные наблюдений будут отличаться от загружаемых вами — несомненно, вы будете располагать большим объемом целевой информации, чем автор при подготовке материала главы. Тем не менее вы должны получить похожие результаты. В этой главе мы рассмотрим несколько важных тем.

- Знакомство с важными источниками правительственных данных, в том числе с теми, которые будут применяться нами для прогнозирования.
- Работа с очень большими файлами, содержащими текстовые данные.
- Потоковый статистический анализ больших наборов данных и другие возможности по анализу данных, выполняемому без сохранения информации в памяти.

Получение правительственных данных

Временны*е* ряды, извлекаемые из правительственных наборов данных, представляются кошмаром с точки зрения целостности информации. Такие наборы данных, хотя и снабжаются временны*ми* метками, чаще всего собираются для обычного информирования заинтересованных в них организаций и отдельных граждан, и не служат определенной цели, предполагающей дальнейшее изучение накопленной информации с помощью методов анализа временны*х* рядов. Обычно в них напрочь отсутствуют сведения об условных обозначениях. Также очень трудно получить подтверждение о том, что регистрация и сбор данных выполнялись с использованием согласованных инструментов.[2]

Тем не менее, если вы готовы к новым вызовам или хотите быть первым, кому удастся выявить интересные временны*е* зависимости в поведении людей, касающиеся государственного управления, то вам несказанно повезло жить в эпоху демократического общества и открытой информации. За последние годы правительства многих стран прилагали больше усилий на всех уровнях управления, чтобы сделать данные собираемых ими временны*х* рядов общедоступными для своих граждан. Ниже приведено всего лишь несколько примеров источников данных, из которых можно почерпнуть открытые правительственные данные, представленные в формате временны*х* рядов.

[1] Имеется в виду время завершения работы над текущей главой. — *Примеч. ред.*

[2] Обратите внимание, что данные востребованных временны*х* рядов, такие как отчет о трудовых вакансиях в США, очень хорошо подготовлены, тщательно очищены и отформатированы. Тем не менее такие данные хорошо проанализированы и предоставляют мало возможностей по созданию новых приложений временны*х* рядов потенциальному исследователю или предпринимателю.

- Помесячные данные о работе больниц (`https://perma.cc/4TR3-84WA`), предоставляемые Национальной службой здравоохранения Соединенного Королевства. Этот набор данных, на удивление, поддерживает извлечение временны*х* рядов. Перейдя к разделу “MAR timeseries”, вы сможете ознакомиться с используемыми соглашениями и проследить за временны*ми* изменениями регистрируемых показателей.
- Общедоступный портал данных Ямайки, заслуживший доверие и высокие оценки исследователей наборов временны*х* данных, в том числе предоставляющий детальные сведения о лихорадке Чикунгунья в 2014 году (`https://perma.cc/4RCP-VMY6`) и соответствующий отчет с наглядной анимацией (визуализацией данных временны*х* рядов) эпидемической кривой заболеваемости.
- Сингапурский портал общедоступных данных (`https://perma.cc/N9W4-ZDM8`), включающий обширные наборы информации, поведение которых продемонстрировано непосредственно на главной странице в виде доступных графиков временны*х* рядов (рис. 15.1).

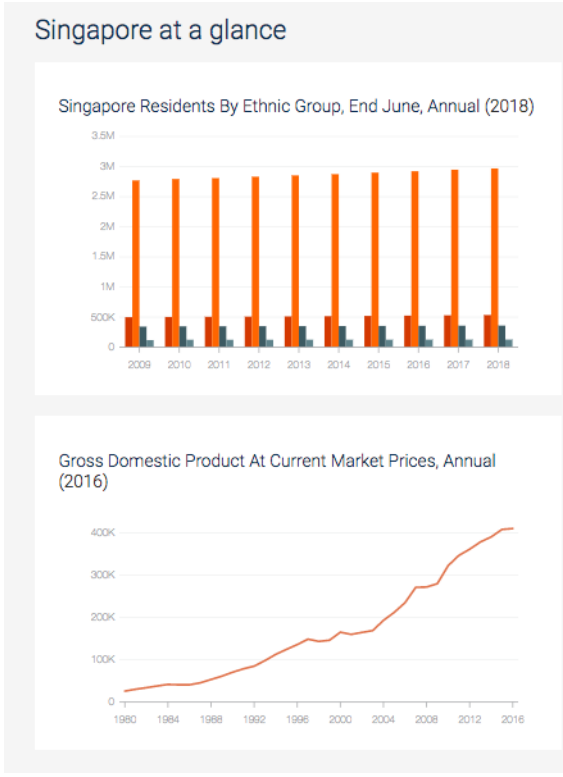


Рис. 15.1. *Два из четырех графиков на главной странице сингапурского портала общедоступной информации (весна 2019 г.), построенных по данным временны*х* рядов, на которых приведены важные сведения из жизни страны*

Как вы могли заметить, приведенные выше примеры представлены для англоязычных стран мира, но ими разнообразие источников открытых правительственных данных не исчерпывается. Например, мэрия Парижа (`https://perma.cc/7V8Z-JZ4T`), правительство Сербии (`https://perma.cc/U3SQ-WF3C`) и Группа Африканского банка развития (`https://perma.cc/7L6X-5B9F`) поддерживают собственные веб-сайты, на которых публикуют большие объемы общедоступных данных.[3]

В примерах из этой главы используется информация с портала общедоступных данных, публикуемых мэрией Нью-Йорка. Выбор не случаен: город очень большой и в нем постоянно что-то происходит; кроме того, по случайному стечению обстоятельств в нем живу я. В следующем разделе мы займемся анализом набора данных, собираемых службой горячей линии 311.

Изучение данных большого временно*го* ряда

Если данные достаточно велики, то вы не сможете разместить их все в памяти. То, насколько большой объем данных поместится в памяти, прежде всего, зависит от используемого вычислительного оборудования.[4] Как бы там ни было, вам нужно научиться проходить по данным, одномоментно обрабатывая их по отдельным фрагментам. Всем, кто знаком с методами глубокого обучения, конечно же, приходилось так поступать, особенно при обработке изображений. В алгоритмах глубокого обучения активно применяются итераторы Python, обеспечивающие быстрый проход по набору данных, которые хранятся в множестве файлов, организованных в отдельные папки.[5]

Объем информации, представленной в формате CSV, скачанного мною набора данных с горячей линии 311, превышал 3 гигабайта. Я никак не могла открыть его на своем компьютере, поэтому моей первой идеей было использование стандартных опций операционной системы Unix, таких как `head`.

К сожалению, то, что было выведено на экран, выглядело настолько громоздким, что не подлежало обработке с помощью командного интерфейса Unix, по крайней мере для человека, не являющегося опытным специалистом по инструментам командной строки Linux.

```
## Командная строка Linux
$ head 311.csv
```

[3] Конечно, для полноценного доступа к таким данным понадобится хорошее знание языка, на котором собирались и публиковались данные.

[4] Если проблема нехватки оперативной памяти решается одним только постоянным масштабированием данных, то вы делаете что-то неправильно.

[5] Чтобы вдохновиться, прочитайте документацию к `TensorFlown` по работе с наборами данных и связанными классами.

```
Unique Key,Created Date,Closed Date,Agency,Agency Name,Complaint Type,Descripto
27863591,04/17/2014 12:00:00 AM,04/28/2014 12:00:00 AM,DOHMH,Department of Heal
27863592,04/17/2014 12:00:00 AM,04/22/2014 12:00:00 AM,DOHMH,Department of Heal
27863595,04/17/2014 10:23:00 AM,0417/2014 12:00:00 PM,DSNY,Queens East 12,Derel
27863602,04/17/2014 05:01:00 PM,04/17/2014 05:01:00 PM,DSNY,BCC - Queens East,D
27863603,04/17/2014 12:00:00 AM,04/23/2014 12:00:00 AM,HPD,Department of Housin
27863604,04/17/2014 12:00:00 AM,04/22/2014 12:00:00 AM,HPD,Department of Housin
27863605,04/17/2014 12:00:00 AM,04/21/2014 12:00:00 AM,HPD,Department of Housin
```

Хотя представленная информация выглядит запутанно, при детальном изучении в ней можно рассмотреть записи временных меток, а также некоторые другие ценные сведения, например географические координаты. Очевидно, что данные очень обширны, и нам нужно научиться обрабатывать их таким образом, чтобы извлекать только нужные столбцы.[6]

Даже начинающим пользователям Linux известны простые инструменты командной строки, которые могут пригодиться при изучении набора данных. Чтобы получить представление о рассматриваемом временном масштабе данных, а именно количестве точек данных, достаточно узнать число строк в файле CSV. Такая задача выполняется с помощью следующей простой команды.

```
## Командная строка Linux

$ wc -l 311.csv
19811967 311.csv
```

Как видите, начиная с 2010 года жители Нью-Йорка подали на горячую линию 311 около 20 миллионов жалоб. Что составляет более двух жалоб на каждого жителя мегаполиса.

Вооружившись этим знанием, воспользуемся пакетом `data.table` языка R — в частности его функцией `fread()` для частичного чтения файлов (в документации (`https://perma.cc/ZHN9-5HD3`) такая возможность рассматривается в разделе с описанием параметров `nrows` и `skip`[7]). Пакет `data.table` показывает необычайно высокую эффективность при обработке больших наборов данных, и мы можем использовать его для получения начальной информации о наших данных, выполнив следующий код.

---

[6] Для эффективного управления файлами формата CSV из интерфейса командной строки или оболочки запуска сценариев опытные пользователи Unix применяют команду `awk`. Подобные инструменты отлично подходят для работы с большими данными, поскольку хорошо отлажены и характеризуются высокой производительностью, в отличие от многих стандартных инструментов анализа данных, представленных в языках R и Python. Если у вас возникают трудности с использованием привычных инструментов обработки данных, то не поленитесь изучить отдельные инструменты командной строки Unix, особенно в случае работы с большими объемами данных.

[7] Обратите внимание, что в библиотеке `Pandas` языка Python предусмотрены аналогичные по функциональным возможностям инструменты (`https://perma.cc/68EE-2ZZ9`).

```
## R
> df = fread("311.csv", skip = 0, nrows = 10)
> colnames(df)
[1] "Unique Key"                          "Created Date"
[3] "Closed Date"                         "Agency"
[5] "Agency Name"                         "Complaint Type"
[7] "Descriptor"                          "Location Type"
[9] "Incident Zip"                        "Incident Address"
[11] "Street Name"                        "Cross Street 1"
[13] "Cross Street 2"                     "Intersection Street 1"
[15] "Intersection Street 2"              "Address Type"
[17] "City"                               "Landmark"
[19] "Facility Type"                      "Status"
[21] "Due Date"                           "Resolution Description"
[23] "Resolution Action Updated Date"     "Community Board"
[25] "BBL" "Borough"
[27] "X Coordinate (State Plane)"         "Y Coordinate (State Plane)"
[29] "Open Data Channel Type"             "Park Facility Name"
[31] "Park Borough"                       "Vehicle Type"
[33] "Taxi Company Borough"               "Taxi Pick Up Location"
[35] "Bridge Highway Name"                "Bridge Highway Direction"
[37] "Road Ramp"                          "Bridge Highway Segment"
[39] "Latitude"                           "Longitude"
[41] "Location"
```

Ознакомившись с информацией в первых 10 строках, можно узнать названия столбцов данных. Несмотря на перечисленные ранее сильные стороны технологии NoSQL, применимых к временны*м* рядам, в большом наборе данных имена столбцов лучше знать заранее — перед детальным его исследованием. Конечно, в NoSQL существуют определенные обходные пути, но в большинстве случаев они требуют дополнительных усилий со стороны пользователей и не предполагают выполнения всех необходимых действий автоматически.

Названия некоторых столбцов сами по себе включают полезную информацию.

```
"Created Date"
"Closed Date"
"Due Date"
"Resolution Action Updated Date"
```

Скорее всего, до преобразования они имеют символьный тип данных, но как только будут приведены к типу POSIXct, можно будет увидеть, как в них выражается промежуток времени между датами.

```
## R
> df$CreatedDate = df[, CreatedDate := as.POSIXct(CreatedDate,
> format = "%m/%d/%Y %I:%M:%S %p")
```

В строке задания форматирования нужно использовать флаг `%I` для указания часа, поскольку они представляются в формате 01–12, и флаг `%p`, так как временные метки должны содержать обозначение "AM/PM".

Чтобы получить представление о том, как распределены даты, в частности, моментов подачи и закрытия жалоб, загрузим больше строк данных и изучим распределение того, что называется *временем жизни* жалобы (т.е. интервал между временем подачи и закрытия жалобы).

```
## R
> summary(as.numeric(df$ClosedDate - df$CreatedDate,
> units = "days"))
   Min.  1st Qu.  Median    Mean  3rd Qu.     Max.  NA's
-75.958    1.000   4.631  13.128   12.994  469.737   113
```

Как видите, распределение этой величины характеризуется широким разбросом значений. Неожиданно то, что некоторые жалобы рассматривались больше года и поэтому имеют отрицательное и даже большое отрицательное время жизни. Если бы отрицательное время составляло около –365 дней (год), можно было бы предположить, что это вызвано проблемами с вводом данных, но для таких чисел, как, например, –75 дней, это кажется маловероятным. В любом случае с такими значениями нам только предстоит разобраться.

В данных наблюдается и другая проблема, которая проявляется при изучении диапазона дат подачи жалоб.

```
## R
> range(df$CreatedDate)
[1] "2014-03-13 12:56:24 EDT" "2019-02-06 23:35:00 EST"
```

Учитывая размер CSV-файла и тот факт, что он должен постоянно обновляться, просто удивительно, что первые строки не относятся к 2010 году, в котором отмечают самые ранние записи данных. Ожидается, что файл CSV подвергается регулярному дополнению. Еще более удивителен тот факт, что даты за 2019 год указаны в первых строках, а даты за 2014 и 2019 годы — в первых 10 000 строках. Это говорит о том, что нельзя сразу определить порядок данных в файле. Давайте визуализируем построчное распределение дат, представив его в виде графика зависимости индекса строки от даты (рис. 15.2).

Избежать такой проблемы с датировкой записей не удастся. Попытавшись понять, как поведение данных изменяется со временем, придется исследовать неупорядоченные данные. Здесь у нас есть несколько вариантов.

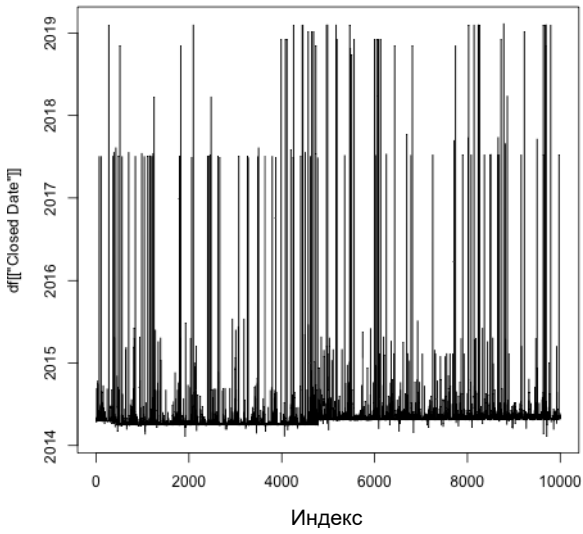


Рис. 15.2. *Несмотря на то что в первых 10 тысячах строк набора большинство дат относятся к осени 2014 года, довольно часто среди них встречаются "перескоки" к 2019 году*

Обновление и агрегирование проходом по данным

Один из вариантов заключается в повышении частоты выборки в процессе прохода по данным для получения сжатых временны*х* рядов с агрегированной статистикой. Разрешение и количество агрегаций можно выбрать в самом начале анализа, а вычислять их — в ходе дальнейшего прохода по данным. После этого нужно будет отсортировать полученные результаты. Попробуем создать словарь/список всех дат с 2010 года по настоящее время, заполняемый соответствующими датами каждой строки. В подобном случае такой список/словарь будет содержать относительно небольшое количество записей, и впоследствии его можно будет отсортировать по датам относительно легко.

Преимущество предложенного подхода заключается в относительно простом коде реализации, в котором также будут проводиться очистка, исследование и анализ данных. А его недостаток выражается в том, что необходимые сведения все равно теряются в запутанности неотсортированных данных, и если нас интересует определенный период времени, то нам придется прочитать весь неотсортированный файл, чтобы отыскать все целевые записи.

В главе 2 рассматривались примеры решения задач повышения частоты выборки, поэтому здесь я оставляю ее в качестве домашнего упражнения.

Сортировка данных

Другим возможным вариантом является сортировка данных. Это сложная задача, если учитывать большой размер файла и неупорядоченность дат, которые наблюдаются в исходном наборе. Даты не упорядочены даже для данных за 2014 год, что более чем странно. По этой причине у нас нет доверия к каким бы то ни было структурным проявлениям в отдельных фрагментах данных, и мы будем рассматривать их все как кучу данных, собранных в случайном порядке.

Сортировка полного файла потребует чрезвычайно интенсивного использования памяти, и есть целые две причины, по которым ее все же стоит выполнить. Во-первых, сортировка проводится только единожды с последующим сохранением результатов для последующего использования. Во-вторых, в процессе сортировки сохраняется исходный уровень детализация данных, что позволит в дальнейшем предельно внимательно изучать данные для выбранных периодов времени, чтобы разобраться в существующих закономерностях.

Существуют несколько вариантов сортировки.

- В командную строку Linux встроен специальный инструмент сортировки данных.
- Большинство процессоров баз данных включают собственные команды сортировки данных — всегда можно перенести записи в базу данных и решить задачу ее силами.[8]
- Можно предложить и реализовать свой алгоритм сортировки. Его код не должен требовать выделения для решения задачи огромного количества памяти. Однако нам вряд ли удастся написать решение, хотя бы столь же эффективное, как готовые функции сортировки.

Остановим свой выбор на командной строке Linux. Хотя обучение может занять некоторое время, вы приобретете новый навык, а заодно — и отсортированный большой файл.

Сначала создадим небольшой тестовый файл.

```
## Командная строка Linux
$ head -n 1000 311.csv | tail -n 999 > test.csv
```

Обратите внимание, что в командную строку введены команды `head` (распечатать начало) и `tail` (распечатать конец). Команда `head` обрабатывает первую строку файла, в которой указаны имена столбцов. Если включить ее в сортировку вместе с общими значениями, то имена столбцов в верхней заголовочной строке файла сохранены не будут, поэтому предварительно вырежем ее из набора.

[8] Нам нужно определиться с базой данных, которая обладает всеми необходимыми функциональными возможностями, — далеко не все они, и особенно не все базы данных временны*х* рядов, создаются в предположении, что данные поступают не в хронологическом порядке.

В операционной системе Linux встроенная команда сортировки запускается следующим образом.

```
## Командная строка Linux
$ sort --field-separator=',' --key=2,3 test.csv > testsorted.csv
```

В нашем случае сначала определяется разделитель полей, а затем указывается способ сортировки файла — по второму и третьему столбцам, т.е. по датам создания и закрытия (расположение которых известно из предварительного анализа файла). Результаты передаются в новый файл, поскольку сохранять их в исходном месте будет далеко не самой удачной идеей.

Теперь можно приступать к изучению отсортированного файла с помощью инструментов языка R, но легко обнаружить, что в этом случае он, к сожалению, не возвращается. Чтобы понять, почему это происходит, вернемся к моменту применения команды `head` к файлу CSV. Его сортировка проводилась по столбцу даты, который, тем не менее, обрабатывается как строка, а не как дата. Поэтому текущий формат даты начинается с указания месяца, и сортировка будет выполняться по месяцам, а не по общим времени и дате, которые наблюдаются при просмотре "отсортированного" файла CSV, полученного с помощью предыдущей команды.

```
  1: 02/02/2019 10:00:27 AM 02/06/2019 07:44:37 AM
  2: 03/27/2014 07:38:15 AM 04/01/2014 12:00:00 AM
  3: 03/27/2014 11:07:31 AM 03/28/2014 01:09:00 PM
  4: 03/28/2014 06:35:13 AM 03/28/2014 10:47:00 PM
  5: 03/28/2014 08:31:38 AM 03/28/2014 10:37:00 AM
---
995: 07/03/2017 12:40:04 PM 07/03/2017 03:40:02 PM
996: 07/04/2017 01:30:35 AM 07/04/2017 02:50:22 AM
997: 09/03/2014 03:32:57 PM 09/04/2014 04:30:30 PM
998: 09/05/2014 11:17:53 AM 09/08/2014 03:37:25 PM
999: 11/06/2018 07:15:28 PM 11/06/2018 08:17:52 PM
```

Очевидно, сортировка имеет смысл только для строк, но не для дат. Фактически это указывает на преимущество формата дат ISO, при использовании которого сортировка строки все равно будет выполнена правильно, в отличие от предыдущего формата. Это пример распространенной проблемы в анализе извлекаемых временны*х* рядов: используемый в данных формат временны*х* меток часто не самый подходящий для анализа.

Вновь обратимся к программному интерфейсу Open Data NYC, чтобы увидеть, можно ли обойти такое форматирование (рис. 15.3).

Взглянув на веб-страницу с данными, представленными в табличном формате, можно заметить, что они совпадают с записями в исходном CSV-файле, не отсортированными во временно*м* порядке. Однако на сайте предусмотрена команда сортировки — используем ее и дождемся обновления представления набора данных, на что уходит определенное время, предположительно

затрачиваемое на сохранение обновленного представления. Это было бы отличным решением, если бы таким образом можно было привести в должный вид загруженный файл CSV.

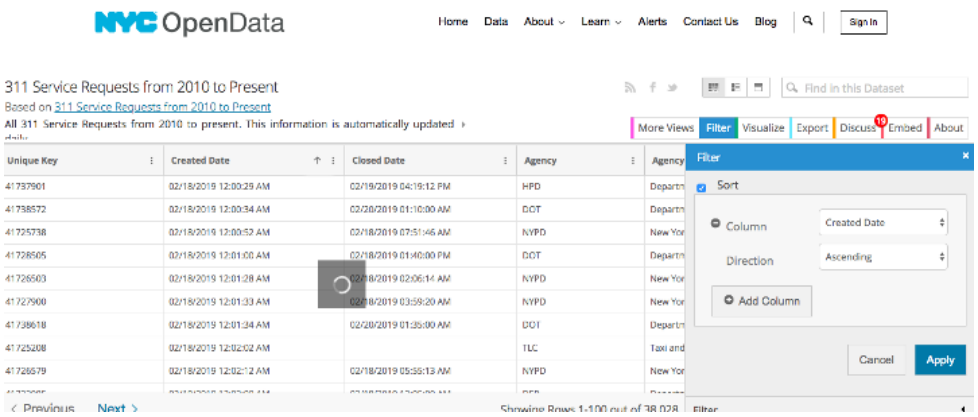


Рис. 15.3. *Портал Open Data NYC предлагает выполнять сортировку с помощью специальной встроенной команды, которая доступна для любого столбца большого набора данных. Это очень хорошо проработанный ресурс, обладающий достаточной производительностью для сортировки больших наборов данных*

В настоящее время можно рассмотреть ряд других решений, например проверить, позволяет ли программный интерфейс получить более удобочитаемые даты или применить более эффективные способы сортировки, чем предлагаются веб-интерфейсом Open Data NYC. Кроме того, можно обратиться к другим инструментам командной строки Linux, таким как `awk`, для разнесения отдельных компонентов временной метки в разные столбцы, или добавить их в общий реорганизованный столбец с ISO-форматированием.

Вместо этого будем придерживаться более простого подхода, чтобы понять, будут ли используемые нами инструменты обрабатывать наш CSV-файл при считывании только определенных столбцов. Первый вопрос, на который хочется получить ответ при исследовании набора данных, — как изменяется временная разность между датами поступления жалоб на горячую линию 311 и их закрытия. В данном случае предполагается, что нам понадобится всего два столбца: `CreatedDate` и `ClosedDate`. Попробуем определить, можно ли считать данные этих двух столбцов, составляющих всего лишь небольшую часть набора данных, с точки зрения как общего количества столбцов, так и количества символов в них (отдельные столбцы очень длинные) на моем простом ноутбуке. (Более “ленивый” способ решения проблемы заключается в финансово затратном обновлении аппаратного обеспечения.)

Итак, попробуем прочитать строки данных, но наш последующий анализ будет выполняться для полного набора данных, а не его первых 1000 строк.

```
## R
> ## Считывание определенных столбцов
> df = fread("311.tsv", select = c("Created Date", "Closed Date"))
>
> ## Придерживаемся стандартной для пакета data.table парадигмы
> ## именования столбцов
> setnames(df, gsub(" ", "", colnames(df)))
>
> ## Исключение строк с пустыми полями дат
> df = df[nchar(CreatedDate) > 1 & nchar(ClosedDate) > 1]
>
> ## Преобразование строковых данных столбцов в формат POSIXct
> fmt.str = "%m/%d/%Y %I:%M:%S %p"
> df[, CreatedDate := as.POSIXct(CreatedDate, format = fmt.str)]
> df[, ClosedDate := as.POSIXct(ClosedDate, format = fmt.str)]
>
> ## Упорядочивание столбца CreatedDate в хронологическом порядке
> setorder(df, CreatedDate)
>
> ## Вычисление количество дней между подачей и закрытием жалоб
> ## на горячую линию 311
> df[, LagTime := as.numeric(difftime(ClosedDate, CreatedDate,
>                                     units = "days"))]
```

Этот код выполнен на самом обычном ноутбуке, приобретенном в 2015 году, и вы сможете сделать это на любом домашнем или рабочем компьютере. Те из вас, кто плохо знаком с большими данными, будут удивлены тем, что 19 миллионов строк можно обработать со столь высокой скоростью, но это происходит довольно часто. На самом деле нам понадобится всего лишь небольшой фрагмент данных для решения текущего вопроса представления временного ряда.

В столбце `LagTime` встречались совершенно неправильные числа — десятки тысяч дней или даже отрицательные значения. Они были отсеяны — мы также установили ограничение на данные, обусловленное случайным распределением выборки.

```
## R
> summary(df$LagTime)
    Min.  1st Qu.  Median  Mean   3rd Qu.      Max.    NA's
-42943.4      0.1     2.0   2.0       7.9  368961.1  609835
> nrow(df[LagTime < 0]) / nrow(df)
[1] 0.01362189

> nrow(df[LagTime > 1000]) / nrow(df)
[1] 0.0009169934
```

```
> df = df[LagTime < 1000]
> df = df[LagTime > 0]

> df.new = df[seq(1, nrow(df), 2), ]
> write.csv(df.new[order(ClosedDate)], "abridged.df.csv")
```

Мы отбрасываем данные с отрицательным временем задержки, так как нам не хватает документации или знаний в предметной области, чтобы понять, что они означают. Также отвергаются данные, которые, по нашему мнению, имеют нереалистичные или бесполезные значения, — исключению подлежат данные, время задержки до закрытия жалобы на горячую линию 311 которых составляло более 1000 дней.[9]

Проявляйте осторожность при отбраковке данных. В этом упражнении было отвергнуто около 1,3% данных с задержками рассмотрения жалоб, которые не имели смысла либо потому, что они требовали объяснений, которые невозможно получить, либо потому, что они представлялись отрицательными значениями, подразумевающими ошибку при вводе данных или даже техническую проблему, выходящую за рамки нашего понимания.

Учитывая, что рассматриваемый вопрос относится к общему распределению данных, маловероятно, что такая небольшая часть отброшенных данных повлияет на исходное распределение оставшихся записей. Тем не менее в реальном мире желательно исследовать все данные, чтобы понять возможные последствия такой операции. Это замечание относится не только к временны*м* рядам, оно касается общей культуры работы с данными.

Теперь, когда мы можем одновременно хранить в памяти все данные, представляющие интерес, можно приступить к анализу временно*г*о ряда. Однако интерес представляет вопрос, изменилось ли и как именно распределение времени задержки со временем. Для ответа на него можно использовать скользящее или разворачивающееся окно в данных временно*г*о ряда, но это связано с заметными вычислительными затратами — нам придется многократно выполнять большое количество связанных расчетов, перемещая окно по диапазону данных. Кроме того, исследование временно*г*о ряда нужно выполнять в потоковом режиме, поскольку проект предполагает постоянное обновление данных по мере их

[9]Хотя 1000 дней могут показаться невероятно долгим сроком, на самом деле я лично подал несколько жалоб на горячую линию 311, которые не были рассмотрены в течение стольких дней. Я годами ждал, когда коммунальные службы пересадят дерево перед моим домом после того, как оно засохло — каждый раз, когда я звонил на горячую линию 311, мне говорили, что работают над этим вопросом.

поступления. Следовательно лучше вообще отказаться от бесконечного повторного сохранения файла исходных данных размером в несколько гигабайтов.

Потоковый статистический анализ временны*х* рядов

Далее мы обратимся к довольно простому онлайн-инструменту оценки квантилей, известному как *алгоритм P-squre* (`https://perma.cc/G8LA-7738`), с одной небольшой модификацией, позволяющей ему учитывать время. Исходный алгоритм предполагал существование устойчивого распределения, из которого рассчитывались квантили, но нам нужно выполнить эту задачу для распределений, которые изменяются со временем. Как и в случае экспоненциально взвешенного скользящего среднего, добавляем в модель осведомленность о времени, приписывая более ранним наблюдениям меньший вес, — будем вводить коэффициент уменьшения веса предыдущих измерений каждый раз, когда становится доступным новое измерение (рис. 15.4).

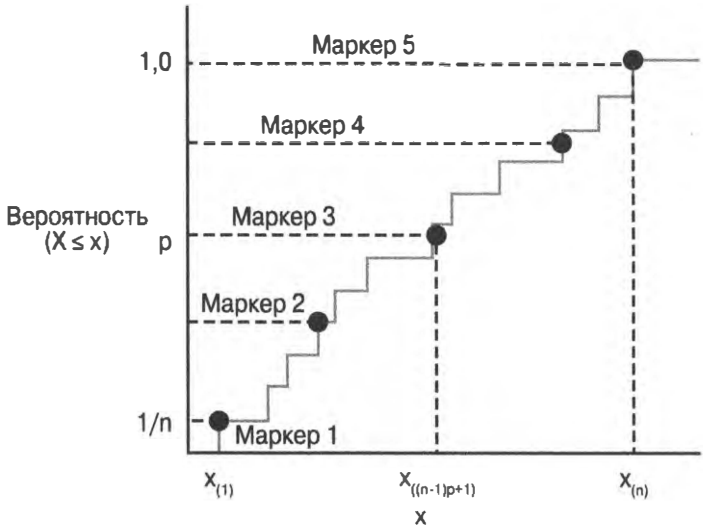


Рис. 15.4. *Схема вычисления квантилей по алгоритму P-squre. Маркерами обозначаются предполагаемые значения квантилей, а также совокупное количество точек данных, меньших или равных каждому из этих квантилей*

Алгоритм работает на небольшом количестве входных записей, что облегчает его реализацию на более выраженном объектно-ориентированном языке программирования, поэтому мы переключимся с R на Python. Однако обратите внимание, что далее по-прежнему используются данные, предварительно обработанные на R, поскольку пакет `data.table` обладает несравнимо большей производительностью при работе с большими данными, чем инструменты языка Python.

Реализуемый далее вариант алгоритма P-square отвечает за построение гистограммы возвращаемых значений. В нем создается объект `PQuantile`, определяется необходимое количество интервалов для значений гистограммы, позиций таких интервалов и текущих результатов наблюдений.

```
## Python
>>> ## Импорт пакетов и лямбда-функции
>>> import bisect
>>> import math
>>>
>>> sign = lambda x: (1, -1)[x < 0]
>>>
>>> ## Начало определения класса
>>> class PQuantile:
>>> def __init__(self, b, discount_factor):
>>> ## Инициализация
>>> self.num_obs = 0 ## очевидно
>>> ## Количество на квантиль
>>> self.n = [i for i in range(self.b+1)]
>>> self.q = [] ## оценка квантиля
>>>
>>> ## b — количество квантилей,
>>> ## включая 0- и 100-й
>>> ## (минимальный и максимальный) квантили
>>> self.b = b
>>> ## Коэффициент понижения предыдущих оценок
>>> ## при поступлении новых данных
>>> self.discount_factor = discount_factor
```

Существует всего два настраиваемых параметра: количество равномерно распределенных квантилей для оценки и коэффициент дисконтирования для старых наблюдений.

К другим членам класса относятся промежуточный итог количества наблюдений (зависящий от дисконтирования времени по настраиваемому коэффициенту дисконтирования времени), оценочных квантилей и текущего подсчета наблюдений, меньших или равных данному значению квантиля.

Здесь применяется только одна открытая функция, роль которой заключается в принятии следующего наблюдения. При поступлении нового наблюдения производятся действия, которые зависят от того, насколько глубоко мы находимся внутри временного ряда. Для первых значений `self.b` входные данные принимаются полностью и обязательно включаются в оценку квантиля. Массив `self.q` отсортировывается так, что его значения представляют квантили.

Так, например, представьте, что количество предполагаемых значений квантилей задается как `ab = 5`, а входные данные представлены последовательностью 2, 8, 1, 4, 3. В результате массив `self.q` будет равен `[1, 2, 3, 4, 8].self.n`,

а число значений, меньшее или равное каждому из квантилей, будет равно [1, 2, 3, 4, 5], т.е. значению, в которое оно было инициализировано в методе __init__().

```
## Python
>>>     def next_obs(self, x):
>>>         if self.num_obs < (self.b + 1):
>>>           self.q.append(x)
>>>           self.q.sort()
>>>           self.num_obs = self.num_obs + 1
>>>     else:
>>>           self.next_obs2(x)
>>>           self.next_obs = self.next_obs2
```

Ситуация становится интереснее по мере увеличения количества значений self.b. На этом этапе в коде принимаются решения о том, как объединять значения для оценки квантилей, не сохраняя все точки данных, предусмотренные для повторного анализа. В алгоритме P-squre эта задача решается с помощью специального объекта, который в нашем случае получил название self.next_obs2.

```
## Python
>>>     def next_obs2(self, x):
>>>         ## Уменьшение количества наблюдений
>>>         if self.num_obs > self.b * 10:
>>>            corrected_obs = max(self.discount_factor * self.num_obs,
>>>                                self.b)
>>>            self.num_obs = corrected_obs + 1
>>>            self.n = [math.ceil(nn * self.discount_factor)
>>>            for nn in self.n]
>>>
>>>            for i in range(len(self.n) - 1):
>>>                if self.n[i + 1] - self.n[i] == 0:
>>>                   self.n[i+1] = self.n[i + 1] + 1
>>>                elif self.n[i + 1] < self.n[1]:
>>>                   ## На практике это условие почти не выполняется
>>>                   self.n[i + 1] = self.n[i] - self.n[1 + 1] + 1
>>>     else:
>>>           self.num_obs = self.num_obs + 1
>>>
>>>     k = bisect.bisect_left(self.q, x)
>>>     if k is 0:
>>>          self.q[0] = x
>>>     elif k is self.b+1 :
>>>          self.q[-1] = x
>>>          k = self.b
>>>     if k is not 0:
>>>          k = k - 1
>>>
```

```
>>>      self.n[(k+1):(self.b+1)] = [self.n[i] + 1
>>>                                  for i in range((k+1),
>>>                                                 (self.b+1))]
>>>      for i in range(1, self.b):
>>>          np = (i)*(self.num_obs - 1 )/(self.b)
>>>          d = np - self.n[i]
>>>          if (d >= 1 and (self.n[i+1] - self.n[i]) > 1):
>>>              self._update_val(i, d)
>>>          elif (d <= -1 and (self.n[i-1] - self.n[i]) < -1):
>>              self._update_val(i, d)
```

В идеале значение *i*-го квантиля должно соответствовать такому равномерному распределению, чтобы точно равняться *i*/*b* × *Общее количество наблюдений, меньших данного значения*. Если это не так, маркер смещается на одну позицию влево или вправо, и соответствующее значение квантиля изменяется с помощью формулы, полученной из предположения о локальной параболической форме гистограммы. Эта формула диктует критерии, указанные ранее для определения величины переменной *d*, указывающей на необходимость корректировки определенного значения квантиля и проведения уточняющих расчетов.

Если значение подлежит уточнению, то необходимо принять еще одно решение — о способе корректировки: параболическая или линейная, как показано в следующем коде. Для получения более подробной информации ознакомьтесь с оригинальной статьей (`https://perma.cc/G8LA-7738`). Материал статьи отличается доступными математическими формулировками, а также предельно точными инструкциями по реализации метода с последующей проверкой полученных результатов.

```
## Python
>>>     ## Общее обновление
>>>     ## При изменении позиции квантиля
>>>     ## изменяются и self.q, и self.n
>>>     def _update_val(self, i, d):
>>>        d = sign(d)
>>>        qp = self._adjust_parabolic(i, d)
>>>        if self.q[i] < qp < self.q[i+1]:
>>>            self.q[i] = qp
>>>        else:
>>>            self.q[i] = self._adjust_linear(i, d)
>>>        self.n[i] = self.n[i] + d
>>>
>>>     ## Основной метод обновления
>>>     def _adjust_parabolic(self, i, d):
>>>        new_val = self.q[i]
>>>        m1 = d/(self.n[i+1] - self.n[i-1])
>>>        s1 = (self.n[i] - self.n[i-1] + d) *
>>>                 (self.q[i+1] - self.q[i]) /
>>>                 (self.n[i+1] - self.n[i])
```

```
>>>       s2 = (self.n[i+1] - self.n[i] - d) *
>>>
>>>    ## Линейное уточнение, если условия параболического
>>>    ## уточнения не выполняются
>>>    def _adjust_linear(self, i, d):
>>>       new_val = self.q[i]
>>>       new_val = new_val + d * (self.q[i + d] - self.q[i]) /
>>>                               (self.n[i+d] - self.n[i])
>>>       return new_val
```

Для более простого изучения ниже приводится весь код класса.

```
## Python
>>> class PQuantile:
>>>    ## Инициализация
>>>    def __init__(self, b, discount_factor):
>>>       self.num_obs = 0
>>>       self.b = b
>>>       self.discount_factor = discount_factor
>>>       self.n = [i for i in range(self.b+1)]
>>>       self.q = []
>>>
>>>    ## Ввод данных
>>>    def next_obs(self, x):
>>>        if self.num_obs < (self.b + 1):
>>>          self.q.append(x)
>>>          self.q.sort()
>>>          self.num_obs = self.num_obs + 1
>>>        else:
>>>          self.next_obs2(x)
>>>          self.next_obs = self.next_obs2
>>>
>>>    def next_obs2(self, x):
>>>        ## Уменьшение количества наблюдений
>>>        if self.num_obs > self.b * 10:
>>>           corrected_obs = max(self.discount_factor
>>>                                  * self.num_obs,
>>>                               self.b)
>>>           self.num_obs = corrected_obs + 1
>>>           self.n = [math.ceil(nn * self.discount_factor)
>>>                                        for nn in self.n]
>>>
>>>           for i in range(len(self.n) - 1):
>>>               if self.n[i + 1] - self.n[i] == 0:
>>>                  self.n[i+1] = self.n[i + 1] + 1
>>>               elif self.n[i + 1] < self.n[1]:
>>>                   ## На практике это условие почти не выполняется
>>>                   self.n[i + 1] = self.n[i] - self.n[1 + 1] + 1
>>>    else:
>>>        self.num_obs = self.num_obs + 1
```

```
>>>
>>>    k = bisect.bisect_left(self.q, x)
>>>    if k is 0:
>>>         self.q[0] = x
>>>    elif k is self.b+1 :
>>>         self.q[-1] = x
>>>         k = self.b
>>>    if k is not 0:
>>>         k = k - 1
>>>
>>>    self.n[(k+1):(self.b+1)] = [self.n[i] + 1
>>>                                for i in range((k+1),
>>>                                               (self.b+1))]
>>>    for i in range(1, self.b):
>>>        np = (i)*(self.num_obs - 1)/(self.b)
>>>        d = np - self.n[i]
>>>        if (d >= 1 and (self.n[i+1] - self.n[i]) > 1):
>>>           self._update_val(i, d)
>>>        elif (d <= -1 and (self.n[i-1] - self.n[i]) < -1):
>>>             self._update_val(i, d)
>>>
>>>    ## Уточнение гистограммы
>>>     def _update_val(self, i, d):
>>>     d = sign(d)
>>>     qp = self._adjust_parabolic(i, d)
>>>     if self.q[i] < qp < self.q[i+1]:
>>>         self.q[i] = qp
>>>     else:
>>>         self.q[i] = self._adjust_linear(i, d)
>>>     self.n[i] = self.n[i] + d
>>>
>>>   def _adjust_parabolic(self, i, d):
>>>      new_val = self.q[i]
>>>      m1 = d/(self.n[i+1] - self.n[i-1])
>>>      s1 = (self.n[i] - self.n[i-1] + d) *
>>>               (self.q[i+1] - self.q[i]) /
>>>               (self.n[i+1] - self.n[i])
>>>      s2 = (self.n[i+1] - self.n[i] - d) *
>>>               (self.q[i] - self.q[i-1]) /
>>>               (self.n[i] - self.n[i-1])
>>>      new_val = new_val + m1 * (s1 + s2)
>>>      return new_val
>>>
>>>    def _adjust_linear(self, i, d):
>>>       new_val = self.q[i]
>>>       new_val = new_val + d * (self.q[i + d] - self.q[i]) /
>>>                                (self.n[i+d] - self.n[i])
>>>       return new_val
```

Теперь, получив в свое распоряжение метод, учитывающий временную изменчивость распределения, нужно убедиться в его работоспособности, проверив на тестовом наборе. Сначала попытаемся выбрать точки данных из одного распределения, а затем скачкообразно перейдем к другому распределению. В каждом случае будем выполнять выборку для 40-го процентиля, хотя, основываясь на исходной конфигурации из 10 точек, гистограмма предполагает отображение сведений для 0-, 10-, 20-… 90-, 100-го процентиля. Они позволяют подробно описать поведение изменяющегося во времени распределения. В следующем примере тестирования метода мы сосредоточимся на рассмотрении только 40-го процентиля (`qt.q [4]`). График его изменения показан на рис. 15.5.

```
## Python
>>> qt = PQuantile(10, 0.3)
>>> qt_ests = []
>>>
>>> for _ in range(100):
>>>    b.next_obs(uniform())
>>>    if len(b.q) > 10:
>>>       qt_ests.append(qt.q[4])
>>> for _ in range(100):
>>>     b.next_obs(uniform(low = 0.9))
>>>     qt_ests.append(qt.q[4])
>>>
>>> plt.plot(qt_ests)
```

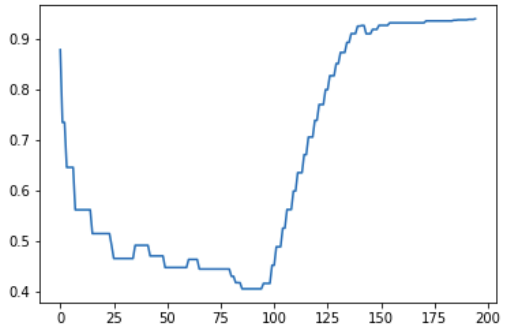


Рис. 15.5. *При сильном дисконтировании предыдущих измерений (используя меньший коэффициент дисконтирования) можно легко убедиться в заметном изменении исходного распределения*

В случае большего коэффициента дисконтирования, напротив, наблюдается более медленная реакция на изменяющийся квантиль (рис. 15.6).

```
## Python
>>> qt = PQuantile(10, 0.8)
>>> qt_ests = []
>>>
```

```
>>> for _ in range(100):
>>>     b.next_obs(uniform())
>>>     if len(b.q) > 10:
>>>         qt_ests.append(qt.q[4])
>>>     for _ in range(100):
>>>         b.next_obs(uniform(low = 0.9))
>>>         qt_ests.append(qt.q[4])
>>>
>>> plt.plot(qt_ests)
```

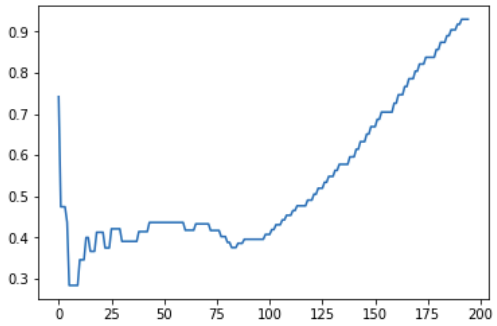


Рис. 15.6. *При дисконтировании предыдущих измерений (используя больший коэффициент дисконтирования) изменения в исходном распределении проявляются медленнее*

Теперь используем полученный скользящий квантиль к подмножеству анализируемых данных (рис. 15.7). Мы не будем подвергать обработке полный набор данных не из-за высокой вычислительной сложности такой операции, а из-за того, что мой рабочий ноутбук слишком медленно отображает рассчитываемые квантили!

```
## Python
>>> import numpy
>>> nrows = 1000000
>>> qt_est1 = np.zeros(nrows)
>>> qt_est2 = np.zeros(nrows)
>>> qt_est3 = np.zeros(nrows)
>>> qt_est4 = np.zeros(nrows)
>>> qt_est5 = np.zeros(nrows)
>>> qt_est6 = np.zeros(nrows)
>>> qt_est7 = np.zeros(nrows)
>>> qt_est8 = np.zeros(nrows)
>>> qt_est9 = np.zeros(nrows)
>>> for idx, val in enumerate(df.LagTime[:nrows]):
>>>     qt.next_obs(val)
>>>     if len(qt.q) > 10:
>>>         qt_est1[idx] = qt.q[1]
>>>         qt_est2[idx] = qt.q[2]
```

```
>>>        qt_est3[idx] = qt.q[3]
>>>        qt_est4[idx] = qt.q[4]
>>>        qt_est5[idx] = qt.q[5]
>>>        qt_est6[idx] = qt.q[6]
>>>        qt_est7[idx] = qt.q[7]
>>>        qt_est8[idx] = qt.q[8]
>>>        qt_est9[idx] = qt.q[9]
>>>
>>> plot(qt_est9, color = 'red')
>>> plt.plot(qt_est7, color = 'pink')
>>> plt.plot(qt_est5, color = 'blue')
>>> plt.plot(qt_est3, color = 'gray')
>>> plt.plot(qt_est2, color = 'orange'
```

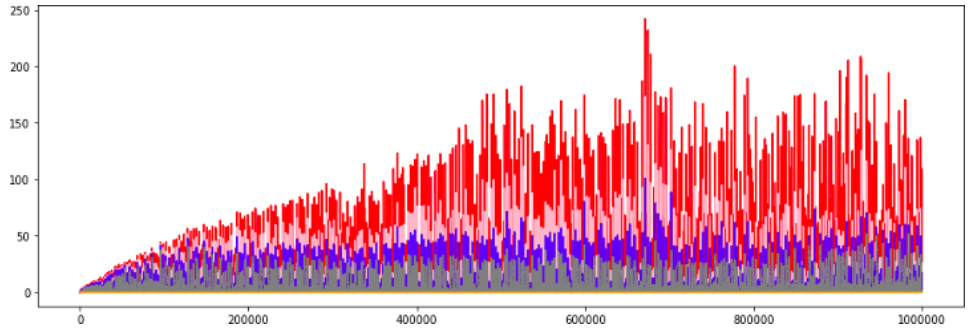


Рис. 15.7. *Временные изменения значений 90-, 70-, 50-, 30- и 20-го процентилей для первых 100 000 строк набора данных при сортировке по дате закрытия жалоб*

На рис. 15.7 проиллюстрировано изменение значений 90-, 70-, 50-, 30- и 20-го процентилей во времени для первых 100 000 строк набора, отсортированного по дате закрытия жалоб. Поскольку сортировка выполнялась по дате закрытия, вполне вероятно, что многие быстро рассмотренные жалобы были учтены заранее, что объясняет гораздо меньшие оценки квантилей в первой части набора данных.[10]

Так изменилось ли распределение? Визуальная оценка показывает, что да, изменилось. Один из подтверждающих факторов — левое цензурирование, указывающее на способ сортировки и отбора данных. Упорядочивание данных по столбцу `ClosedData` в сочетании с тем фактом, что этот набор данных, по-видимому, не имеет бесконечного горизонта событий (жалобы, поданные до определенной даты, попросту не попали в обработку), делает разброс в начале более коротким. Иначе говоря, очевидно, что изменение во времени — это всего

[10] 20-й процентиль настолько мал, что его невозможно увидеть в масштабе других квантилей, представленных на графике. Он выглядит как сплошная горизонтальная линия у основания других распределений, но вы могли бы увидеть его лучше, перейдя к логарифмическому масштабу или построив в отдельных координатах.

лишь следствие неполноты входных данных (а потому и базового набора данных) и выбора неправильного метода сортировки.

С другой стороны, на графике наблюдаются явные признаки того, что распределения изменяются с течением времени. В кривых оценок квантилей просматриваются пики и впадины, указывая на необходимость изучения таких кривых на предмет периодических изменений, поскольку рост и спад значений квантиля в определенные периоды времени вполне могут обусловливаться внешними факторами, например, организационного характера (закрытие жалоб преимущественно в конце месяца или ввиду циклического финансирования, позволяющего привлечь больше сотрудников для рассмотрения жалоб только в определенное время года).

Учитывая предварительные результаты, лучшим вариантом дальнейших действий будет определение ключевых, значимых дат (дат, в которых наблюдаются пики или периодическое поведение) и попытка сопоставить их с любыми институциональными фактами о рабочих ритмах службы 311, которые только удастся установить. Неплохо бы также провести моделирование данных, чтобы понять, как левое цензурирование сопоставляется с ранними оценками значений квантилей в различных сценариях. Таким образом можно будет получить лучшее качественное и количественное понимание неизвестных ранее поведенческих аспектов системы, и эта информация может стать ключевой при обосновании предположений о временном изменении распределения разрешения времени и в случае их справедливости помочь определить причины таких изменений.

Допустим, мы предприняли все необходимые действия, и что же дальше? Нам нужно будет найти методологию сравнения распределений по разностям подобий, для которых были доступны только квантили распределения, а не все точки выборки. Один из способов достичь этого — выполнить моделирование всего процесса по методу бустрэпинга. Это позволит полностью сформулировать и проконтролировать предположения, которые должны включаться в модель по результатам моделирования. Фактически многие статистические подходы к проведению таких сравнений также основаны на методах бутстрэпа.

Левое, правое и интервальное цензурирование

Для анализа временны*х* рядов и выполнения связанных задач, таких как анализ выживаемости, нужно точно знать, какое влияние на получаемый результат будут оказывать способы выбора, сортировки и представления данных. В нашем случае ранние периоды времени, как представляется, характеризуются более узким распределением, чем наблюдается в поздние периоды времени. Вероятнее всего, это связано с сортировкой данных по дате закрытия, представляющего собой *левое цензурирование*, указывающее, что событие произошло до начала исследования. В данном случае интересующее нас событие — это даты подачи и за-

крытия жалоб, а весь набор ограничен датой начала сбора данных. Противоположная концепция *правого цензурирования* означает, что интересующее событие произойдет после завершения наблюдений. Наконец, *интервальное цензурирование* описывает ситуации, в которых известно, что неточные записи о событиях, которые обязательно повлияют на результат проводимого анализа, наблюдаются в строго заданном временном интервале.

Нерассмотренные вопросы

Результаты визуализации порождают новые вопросы. Один из них связан с возможностью циклического, или сезонного, поведения. Изучение графика дает основания полагать, что во всех квантилях наблюдаются периодические всплески. Эти проявления требует дальнейшего изучения, и для его проведения есть несколько возможностей.

- Попытаться описать кривые квантилей гармоническими колебаниями (синусоидами и косинусоидами) и посмотреть, носят ли изменения периодичный характер.
- Смоделировать поведение квантиля как процесса ARIMA или SARIMA и провести поиск сезонных зависимостей. Это потребует предварительной обработки данных, например исследования кривых ACF и PACF, для получения которых нужно прибегать к моделированию временных рядов.
- Запросить в бюро службы 311 дополнительную информацию и попытаться узнать, обусловливается ли периодичность организационной структурой и/или операционными процедурами.

В дополнение к периодическому поведению на графике просматривается ярко выраженный скачок в оценочных значениях квантилей для индекса 70 000. Предположение о том, что скачок совершают сразу все квантили, кажется маловероятным, и, скорее всего, он сформирован выбросами одного или нескольких квантилей. Существует несколько подходов к исследованию этого явления.

- Вернуться к необработанным данным за указанный период времени и посмотреть, какие признаки может объяснить наблюдаемый скачок. Наблюдался ли всплеск в количестве поданных жалоб? Или в количестве жалоб определенного типа, решение которых отнимает больше времени?
- В качестве альтернативы можно вернуться к необработанным данным, чтобы вычислить приблизительную дату скачка в квантилях и сопоставить их с записями новостных лент, воспользовавшись помощью кого-то, кто может указать правильное направление поиска. Наиболее ценной будет помощь сотрудников агентства или служащего, имеющего непосредственное

отношение к управлению городом. Кроме того, дата рассматриваемого явления может соответствовать некому массовому мероприятию или чрезвычайному событию, наблюдаемому в Нью-Йорке, например прохождению через город урагана Сэнди в 2012 году.

Дальнейшее улучшение

Всегда можно разработать алгоритм, более чувствительный к временным изменениям в данных. Модификация алгоритма P-squre отвергает предыдущие наблюдения, но предполагает, что все они распределены равномерно. Это проявляется в том факте, что входные данные следующего наблюдения не содержат временные метки, и к ним всегда применяется один и тот же коэффициент дисконтирования. Можно также рассмотреть более гибкий алгоритм, который предполагает использование временных меток, описывая изменения в новой информации через проявление старой информации так, что дисконтирование будет зависеть от временного интервала, прошедшего с момента последнего обновления. Такой алгоритм также будет характеризоваться высокой точностью для нашего набора данных. Его реализацию оставим в качестве домашнего задания, которое сводится к изменению всего нескольких строк кода. *Подсказка*: коэффициент дисконтирования должен стать функцией времени.

Дополнительные источники

Ted Dunning and Otmar Ertal, "Computing Extremely Accurate Quantiles Using t-Digests," research paper, 2019, `https://perma.cc/Z2A6-H76H`

В последнее время алгоритм f-digest, предназначенный для чрезвычайно эффективного и гибкого вычисления квантилей в потоковых данных, все чаще применяется в качестве основного средства оценки потоковых временных рядов, основанного на квантильном анализе, даже для случаев нестационарных распределений. Его реализации доступны на нескольких языках, включая Python и высокопроизводительные варианты C++ и Go. Предложенный подход особенно ценен тем, что в нем не нужно заранее определяться с целевыми квантилями, — вместо этого распределение моделируется как набор кластеров, из которых можно выбрать для анализа любой интересующий вас квантиль.

Dana Draghicescu, Serge Guillas, and Wei Biao Wu, "Quantile Curve Estimation and Visualization for Nonstationary Time Series," Journal of Computational and Graphical Statistics 18, no. 1 (2009): 1–20, `https://perma.cc/Z7T5-PSCB`

В статье описано несколько непараметрических методов моделирования нестационарных распределений, применяемых при проведении квантильной

оценки временны*х* рядов. В ней рассматриваются основанные на реальных данных примеры, базовые принципы моделирования и нестандартные типы распределений (такие, как негауссовы распределения). В ней также приводится пример реализации (что необычно для статьи в академическом журнале по статистике), хотя и с платным доступом.

Andras A. Benczur, Levente Kocsis, and Robert Palovics, "Online Machine Learning in Big Data Streams," research paper, 2018, `https://perma.cc/9TTY-VQL3`

В этой статье обсуждаются технические аспекты реализации общих задач машинного обучения, связанных с обработкой данных временны*х* рядов. Особенный интерес представляет обсуждение принципов потоковой обработки данных, передаваемых в алгоритмы машинного обучения, и технические советы по распараллеливанию потоковых вычислений.

Sanjay Dasgupta, "Online and Streaming Algorithms for Clustering," lecture notes, Computer Science and Engineering, University of California San Diego, Spring 2008, `https://perma.cc/V3XL-GPK2`

Хотя эти лекционные заметки касаются не только временны*х* рядов, в них приведен полный обзор методов неконтролируемой кластеризации потоковых данных. Приведенных сведений достаточно, чтобы, основываясь на них, построить собственные решения для приложений, ориентированных на обработку временны*х* рядов.

Ruofeng Wen et al., "A Multi-Horizon Quantile Recurrent Forecaster," research paper, November 2017, `https://perma.cc/22AE-N7F3`

В этой научной работе Amazon представлен пример использования информации о квантилях в исходных данных для эффективного обучения рекуррентных нейронных сетей при прогнозировании временны*х* рядов. Исследователи указывают на высокую обучаемость нейронной сети, которая способна проводить вероятностную, а не точечную оценку данных. Статья является хорошей иллюстрацией другого потенциального варианта использования информации о квантилях, недостаточно используемого ресурса для анализа временны*х* рядов.

ГЛАВА 16

Пакеты для анализа временных рядов

За последние несколько лет некоторые IT-гиганты выпустили ряд пакетов для анализа временных рядов с сопутствующей документацией. В ней широко освещаются возможности предлагаемых решений по обработке огромных объемов данных временных рядов, собираемых благодаря привлечению широкой клиентской базы, внедрению сложной процедуры регистрации, использованию современных аналитических инструментов и получению многочисленных запросов на прогнозирование самых разных процессов.

В этой главе мы обсудим некоторые из основных исследований и разработок, проводимых с привлечением постоянно увеличивающихся в размере наборов временных рядов, а именно: прогнозирование в масштабе и обнаружение аномалий.

Прогнозирование в масштабе

Для многих крупных технологических компаний потребность в обработке временных рядов возникла естественным образом и вскоре стала одной из важнейших проблем, требующих оперативного решения. Некоторые из таких компаний отреагировали на возникший запрос разработкой интеллектуальных автоматических пакетов, специально предназначенных для анализа временных рядов и "прогнозирования в масштабе", — прогнозы стали востребованными в самых разных предметных областях. Вот как были описаны обстоятельства, вынудившие руководство компании Google принять решение о разработке системы автоматического прогнозирования, двумя из ее непосредственных создателей в блоге 2017 года (`https://perma.cc/6M7J-MWDY`) (форматирование сохранено).

> Запрос на прогнозирование временных рядов в рядах специалистов Google стремительно рос вместе с компанией в первом десятилетии ее существования. *Острая потребность в таких инструментах в деловых и инженерных кругах вызвала разработку множества подходов к прогнозированию, в большей части полагающихся на непосредственное участие специалистов по анализу данных.*

Количество и разнообразие подходов, а в некоторых случаях и их несоответствие, указывают на необходимость унификации, автоматизации и совершенствования методологий прогнозирования, а также распространения полученных результатов с помощью инструментов, которые могут быть развернуты компанией повсеместно. Иначе говоря, необходимо разработать методы и инструменты, которые облегчили бы построение точных прогнозов для крупномасштабных временны*х* рядов, собранных компанией Google.

Существует настолько много релевантных данных и разнообразных задач прогнозирования, что крайне дорого и организационно сложно нанимать "армии" аналитиков, чтобы заниматься всеми ими по-отдельности. Вместо этого компании стали придерживаться философии "разумной достаточности". Иначе говоря, достаточно хороший прогноз гораздо лучше, чем отсутствие прогноза в ожидании идеального эксперта по временны*м* рядам со знанием предметной области. Далее мы более подробно обсудим две системы автоматического прогнозирования от компаний Google и Facebook.

Промышленное решение по прогнозированию Google

Компания Google опубликовала информацию (`https://perma.cc/N3AU-5VWK`) о своем инструменте автоматического прогнозирования, который появился благодаря усилиям нескольких ученых из подразделения разработки поисковой инфраструктуры компании. Задача состояла в том, чтобы выработать единый подход к созданию автоматических инструментов прогнозирования, используемых всей организацией. Поскольку процессы были автоматизированы, это означало, что результаты должны быть безопасными — не должны уходить слишком далеко от основного тренда, но при этом должны представлять некоторую оценку неопределенности в прогнозе. Более того, поскольку команда искала универсальный инструмент, методология должна предусматривать решение распространенных проблем, которые встречаются в наборах данных временны*х* рядов, предопределенных человеческим фактором, таких как сезонность, недостающие данные, праздники в рабочем календаре и т.п.

Решение, предложенное компанией Google, состоит из трех интересных шагов, которые были рассмотрены нами в предыдущих главах.

1. Автоматизированная и всеобщая очистка и сглаживание данных.
2. Временна*я* агрегация и географическое/тематическое разделение данных.
3. Объединение прогнозов и генерация оценок неопределенности, основанных на моделировании.

Далее каждый шаг рассматривается более подробно в порядке следования.

Автоматизированная и всеобщая очистка и сглаживание данных

В упомянутом ранее неофициальном извещении блога Google Data Science два руководителя проекта, выполняемого исключительно силами компании, настаивали на том, что очистка и сглаживание данных позволяют решить ряд важных проблем.

Несовершенство данных

- Отсутствующие данные.
- Наличие выбросов.
- Необоснованное изменение уровня (например, из-за запуска продукта или внезапных, но постоянных изменений в поведении)
- Преобразование данных.

Несовершенные данные — это реалии жизни. Отсутствие данных может произойти из-за технических сбоев. Выбросы могут обусловливаться сходными причинами или представляться действительными значениями, которые не стоит включать в прогноз, поскольку они вряд ли будут повторяться. Изменения уровня (*изменения режима*) могут происходить по множеству причин, в частности из-за резкого изменения базового поведения (эволюционирующий мир), предмета измерения (эволюционирующий продукт) или способа записи (эволюционирующая регистрация). Наконец, данные могут описываться распределениями, для которых не выполняется условие нормальности или стационарности, выдвигаемое во многих моделях анализа временны*х* рядов. Подход компании Google позволяет устранить каждую из проблем несовершенства данных по очереди, придерживаясь автоматизированных методологий их выявления и исправления.

Календарные эффекты

- Годовая сезонность.
- Недельная сезонность (влияние дня недели).
- Выходные и праздники.

Обработка эффектов, связанных с календарем, была особенно сложной задачей для такой организации, как Google, предоставляющей услуги и работающей с пользователями по всему миру. Годовая сезонность сильно различается в разных частях мира, особенно на противоположных полушариях с их различными погодными условиями, а также культурными традициями народов, придерживающихся собственных календарей. Как отмечалось в сообщении компании, иногда один и тот же праздник может наступать более одного раза за один (григорианский) календарный год. Аналогично сезоны, в которые определенная группа людей совершает определенные действия, сильно смещаются в рамках

григорианского календарного года, что часто происходит, например, в исламском мире, где периодичность праздников сложно отслеживать по одному только григорианскому календарю.

Временная агрегация и географическое/тематическое разделение данных

Команда Google обнаружила, что недельные данные хорошо подходят для построения большинства прогнозов, поэтому после предварительной очистки на предыдущем этапе такие данные часто использовались как недельные приращения в дальнейшем прогнозировании. В этом и заключается временная агрегация.

Тем временем команда также обнаружила, что иногда полезно разделить данные — согласно географическим или тематическим категориям (например, по типу устройств), а иногда даже соответственно комбинации этих факторов (например, по географическому региону и типу устройства). Группа обнаружила, что в таких случаях более эффективно составлять прогнозы для подверженных разделению частичных временных рядов, которые впоследствии будут согласовываться для построения общего прогноза, если конечный интерес представляли как частичные, так и глобальный ряд. В главе 6 обсуждались различные способы аппроксимации иерархических временных рядов. Данный метод отражает методологию в направлении снизу вверх.

Объединение прогнозов и генерация оценок неопределенности, основанных на моделировании

Компания Google использует ансамблевые подходы, комбинируя ряд различных прогностических моделей для получения окончательного прогноза. Это оправдано по ряду причин. В Google считают, что тем самым воспроизводится прогноз, основанный на "мудрости экспертов", в которой собраны преимущества многих хорошо функционирующих и обоснованных прогностических моделей (таких, как экспоненциальное сглаживание, ARIMA и др.). Кроме того, ансамбль прогностических моделей генерирует распределение прогнозов, позволяя определить, являются ли какие-либо из них настолько необычными, чтобы считаться необоснованными. Наконец, ансамбль предоставляет способ количественной оценки неопределенности прогноза, что в компании Google также делают, моделируя прямое распространение ошибок во времени.

В конечном итоге, как признает команда специалистов в области анализа данных, подход Google выигрывает от крупных масштабов и методов с высокой степенью параллелизации, получающих преимущество за счет доступа к почти безграничным вычислительным ресурсам компании. Кроме того, выигрыш достигается за счет решения параллельных задач (таких, как моделирование ошибки, многократно распространяющейся во времени) и автоматизированных

процессов с возможностью получения данных от специалистов по анализу, хотя в этом нет большой необходимости. Таким образом, разумные прогнозы генерируются в большом масштабе и для различных наборов данных.

Хотя ваши проекты, скорее всего, не потребуют столь сложных подходов и не будут нуждаться в высоком уровне автоматизации, демонстрируемом компанией Google, вы сможете, даже работая в одиночку, внедрить в свой рабочий процесс многие идеи, заимствованные из их моделей.

- Создание конвейера для предварительной очистки данных и получения базовой версии набора временны*х* рядов перед моделированием.
- Создание ансамбля обоснованных моделей прогнозирования.

Пакет `Prophet` с открытым исходным кодом компании Facebook

Компания Facebook выпустила свой пакет для автоматизированного прогнозирования временны*х* рядов `Prophet` примерно в то же время, когда компания Google опубликовала информацию о своем передовом решении. В своем блоге, посвященном проекту Prophet (`https://perma.cc/V6NC-PZYJ`), компания Facebook освещала те же самые проблемы, над которыми работали в Google.

- Сезонность поведения людей и нерегулярность праздников.
- Необоснованное изменение уровня.
- Отсутствие данных и наличие выбросов.

Кроме того, пакет компании Facebook обладал следующими возможностями.

- Способность обрабатывать наборы данных с различными уровнями детализации, например собранные на поминутной или почасовой основе в дополнение к данным с дневной выборкой
- Тенденции, указывающие на нелинейный рост, например о достижении точки насыщения

Компания Facebook отметила, что результаты, полученные с помощью ее пакета, часто так же хороши, как и оценки аналитиков. Как и сотрудники Google, специалисты Facebook показали, что при разработке своего конвейера для аппроксимации временны*х* рядов очень многие задачи выполнялись в режиме распараллеленных вычислений. В Facebook утверждают, что пакет `Prophet` превратился в настолько надежное средство прогнозирования, что получаемые с его помощью прогнозы начали использоваться не только во внутренних, но и во многих сторонних продуктах. Кроме того, компания Facebook заявила, что

разработала рабочую схему "аналитик в цикле", что позволяет контролировать и корректировать автоматизированный процесс, в конечном итоге приводя к продукту, который может дополнить или полностью устранить человеческий фактор в зависимости от количества ресурсов, выделенных для решения задачи.

Подход Facebook заметно отличается от принятого в Google — он основан на трех простых компонентах.

- Годовые и недельные сезонные зависимости.
- Пользовательские списки праздников.
- Кусочно-линейная или логистическая кривая тренда.

Эти компоненты используются для формирования аддитивной регрессионной модели. Это непараметрическая регрессионная модель, которая не делает никаких предположений о форме базовой регрессионной функции и не требует, чтобы она была линейной. Модель является более гибкой и интерпретируемой, чем линейная регрессия, но характеризуется более высоким значением дисперсии (вспомните о балансе между смещением и дисперсией в машинном обучении и статистике) и большим количеством проблем при переобучении. Эта модель оказывается успешной при решении задач более общего, чем предполагают в Facebook, характера: сложные нелинейные поведения моделируются автоматически, что позволяет избежать необоснованного ограничения методологий.

Пакет `Prophet` обладает рядом преимуществ, среди которых такие.

- Простой программный интерфейс.
- Открытый исходный код в стадии активной разработки.
- Полные и эквивалентные программные интерфейсы на языках Python и R, которые помогают эффективно работать многоязычным командам по анализу данных

Пакет `Prophet` прост в использовании. В конце этого раздела приведен фрагмент кода, взятого из руководства по быстрому знакомству с пакетом (`https://perma.cc/9TLC-FFRM`), чтобы вы могли увидеть минимальный код, необходимый для разворачивания пакета для автоматического прогнозирования. Здесь пакет `Prophet` используется совместно с пакетом `pageviews` языка R, обеспечивающим простое извлечение данных временны*х* рядов о просмотренных страницах Википедии. Вначале данные временны*х* рядов нужно загрузить.

```
## R
> library(pageviews)
> df_wiki = article_pageviews(project = "en.wikipedia",
>                             article = "Facebook",
>                             start = as.Date('2015-11-01'),
>                             end = as.Date("2018-11-02"),
```

```
>                                user_type = c("user"),
>                                platform = c("mobile-web"))
> colnames(df_wiki)
[1] "project" "language" "article" "access" "agent" "granularity"
[7] "date"    "views"
```

Получив данные с дневным временны*м* разрешением за несколько лет, можно попытаться построить на их основании прогноз с помощью пакета `Prophet`, выполнив всего несколько простых действий (рис. 16.1).

```
## R
> ## Выбор целевых данных
> ## и присвоение имен соответствующим столбцам
> df = df_wiki[, c("date", "views")]
> colnames(df) = c("ds", "y")

> ## Логарифмическое преобразование данных для масштабирования,
> ## чтобы нивелировать большие суточные скачки
> df$y = log(df$y)

> ## Создание фрейма будущих данных, включающего будущие даты,
> ## для которых строится прогноз
> ## Прогнозы строятся вперед на 365 дней
> m = prophet(df)
> future <- make_future_dataframe(m, periods = 365)
> tail(future)
            ds
1458 2019-10-28
1459 2019-10-29
1460 2019-10-30
1461 2019-10-31
1462 2019-11-01
1463 2019-11-02

> ## Генерирование прогноза для заданного диапазона дат.
> forecast <- predict(m, future)
> tail(forecast[c('ds', 'yhat', 'yhat_lower', 'yhat_upper')])
            ds     yhat yhat_lower yhat_upper
1458 2019-10-28 9.119005   8.318483   9.959014
1459 2019-10-29 9.090555   8.283542   9.982579
1460 2019-10-30 9.064916   8.251723   9.908362
1461 2019-10-31 9.066713   8.254401   9.923814
1462 2019-11-01 9.015019   8.166530   9.883218
1463 2019-11-02 9.008619   8.195123   9.862962

> ## Получив прогноз целевых величин, построим их график,
> ## чтобы оценить работу пакета
> plot(df$ds, df$y, col = 1, type = 'l', xlim = range(forecast$ds),
> points(forecast$ds, forecast$yhat, type = 'l', col = 2)
```

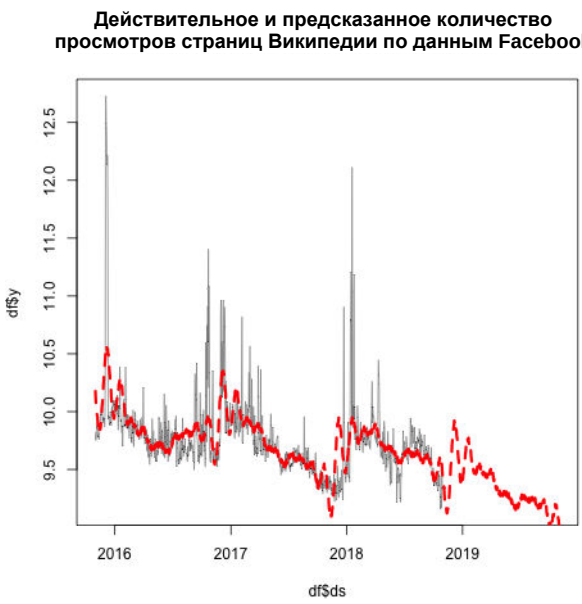


Рис. 16.1. *График частоты посещения Википедии (тонкая сплошная кривая) и прогнозы, полученные с помощью пакета* `Prophet` *для этих данных (тонкая пунктирная кривая)*

Пакет `Prophet` также предлагает возможность построения графиков отдельных компонентов (тренда и сезонных изменений) прогноза (рис. 16.2).

```
## R
> prophet_plot_components(m, forecast)
```

Ограничьте пакет `Prophet` только дневными данными. Как утверждают разработчики, пакет `Prophet` исходно создавался для обработки дневных данных и на них показывает наилучшие результаты. К сожалению, такая узкая специализация предполагает, что одни и те же методы могут демонстрировать разные уровни надежности для данных, полученных в разное время. По этой же причине вам следует относиться к инструментам пакета и связанным методам с п редельной осторожностью при обработке данных с частотой дискретизации, отличной от дневной.

Со временем появятся другие, более доступные, автоматизированные пакеты с открытым исходным кодом для работы с данными временны*х* рядов, а также решения, до определенного времени считающиеся “темной лошадкой”. Как бы там ни было, они послужат хорошей отправной точкой для всех, кто плохо ориентируется в технологиях прогнозирования, но хочет научиться строить правдоподобные прогнозы. Тем не менее не стоит рассчитывать, что в ближайшем будущем такие пакеты позволят добиться точных результатов в прогнозировании

любых временных рядов в произвольной предметной области. Как только вы сможете включать знания предметной области и организационных требований в модель временных рядов, сразу же получите отличный результат, но поскольку универсальные пакеты прогнозирования не очень хорошо справляются с этой задачей, оптимальное решение остается за человеком.

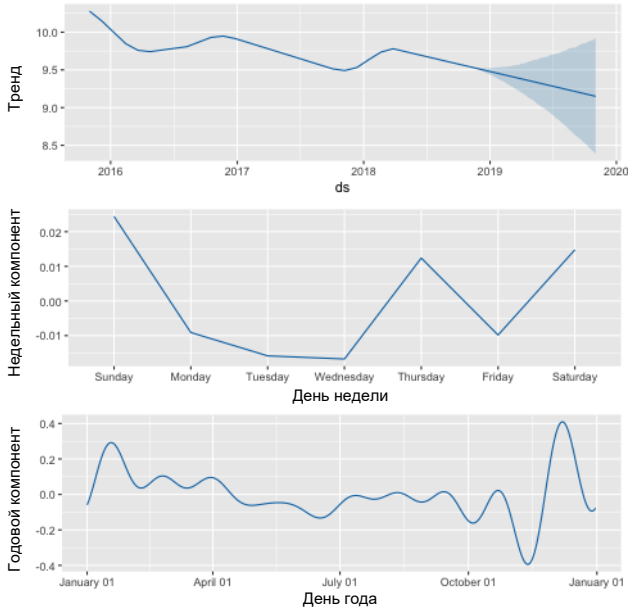


Рис. 16.2. *Прогноз разложен на тренд, недельную и годовую составляющие — он представляет сумму этих компонентов. Обратите внимание на то, что разные составляющие представлены по-разному. Данные тренда имеют линейную форму, а годовые данные образуют извилистую кривую из-за приближения по методу Фурье (детали — в официальном блоге Facebook)*

Обнаружение аномалий

Обнаружение аномалий — это еще одна область, в которой технологические компании прилагают значительные усилия и делятся своими наработками в виде открытого кода. Задача обнаружения аномалий во временных рядах важна для изучения по нескольким причинам.

- Удаление выбросов оказывается полезным при настройке моделей, которые недостаточно к ним устойчивы.
- Обнаружение выбросов выполняется в рамках построения модели, специально предназначенной для прогнозирования величины таких выбросов по времени их появления.

Перейдем к изучению открытого решения по обнаружению аномалий, принятого в Twitter.

Пакет обнаружения аномалий с открытым исходным кодом компании Twitter

Пакет с открытым исходным кодом, который был разработан компанией Twitter четыре года назад, предназначенный для обнаружения аномалий и получивший название `AnomalyDetection`[1], представляет надежный высокопроизводительный инструмент. В нем для выявления выбросов применяется метод Seasonal Hybrid ESD (Extreme Studentized Deviant), основанный на более сложной модели, чем в методе Generalized ESD. При этом обобщенный критерий ESD базируется на статистическом критерии Граббса (`https://perma.cc/MKR5-UR3V`), который используется для проверки гипотезы о существовании в наборе данных одного выброса. В обобщенном критерии ESD критерий Граббса применяется неоднократно: сначала — к наиболее экстремальным выбросам, а затем — последовательно к выбросам все меньшей величины, корректируя критические значения для учета результатов нескольких последовательных проверок. Комбинированный сезонный критерий ESD (Seasonal Hybrid ESD) основывается на обобщенном критерии ESD и применяется для учета сезонности в поведении данных, определяемой с помощью разложения временны*х* рядов.

Пример использования этого пакета можно наблюдать в следующем коде, написанном на языке R. Сначала загрузим выборки данных, любезно предоставленные Twitter.

```
## R
> library(AnomalyDetection)
> data(raw_data)
> head(raw_data)
timestamp count
1 1980-09-25 14:01:00 182.478
2 1980-09-25 14:02:00 176.231
3 1980-09-25 14:03:00 183.917
4 1980-09-25 14:04:00 177.798
5 1980-09-25 14:05:00 165.469
6 1980-09-25 14:06:00 181.878
```

Теперь используем функцию автоматического обнаружения аномалий, разработанную компанией Twitter и включающую два набора параметров.

1. Поиск большей части аномалий выполняется в диапазоне положительных или отрицательных смещений.

[1] Дополнительная информация представлена в репозитории проекта на GitHub (`https://perma.cc/RV8V-PZXU`) и в блоге Twitter (`https://perma.cc/6GPY-8VVT`).

2. Поиск меньшей части аномалий выполняется в диапазоне только положительных смещений.

Ниже продемонстрированы некоторые варианты использования такой функции.

```
## R
> ## Обнаружение большого числа аномалий в любом направлении
> general_anoms = AnomalyDetectionTs(raw_data, max_anoms=0.05,
>                         direction='both')
>
> ## Обнаружение меньшего числа аномалий только в положительном направлении
> high_anoms = AnomalyDetectionTs(raw_data, max_anoms=0.01,
>                         direction='pos')
```

Результаты для обоих случаев показаны на рис. 16.3.

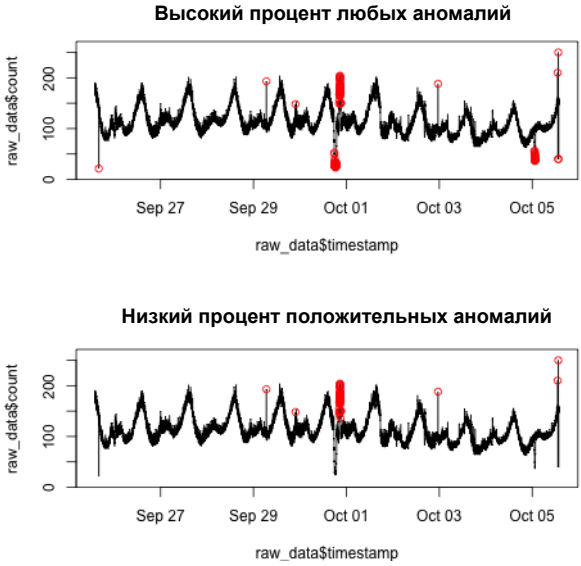


Рис. 16.3. *Аномалии, выявленные функцией* `AnomalyDetectionTs()`*, разработанной компанией Twitter, при очень слабых (вверху) и более строгих ограничениях (внизу)*

Мы видим, что большое количество аномалий наблюдается только в отдельной части временного ряда, поэтому давайте обрежем и масштабируем график до этой его части, чтобы лучше понять поведение данных (рис. 16.4).

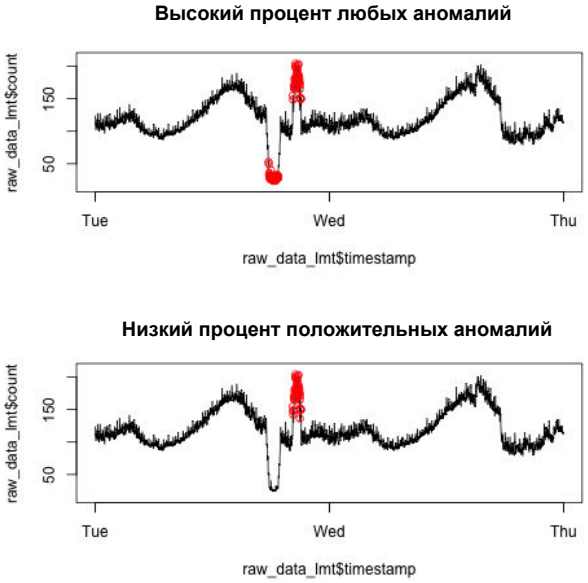


Рис. 16.4. *Те же выявленные аномалии — теперь основное внимание уделяется аномалиям, относящимся к одному кластеру и выявленным в один день*

Чтобы узнать, почему указанные точки рассматриваются как аномальные, нужно изучить их более внимательно. Необходимость в изучении только положительных аномалий лучше всего объяснить на примере разработки инфраструктуры сайта с высоким трафиком и интенсивным использованием данных. В то же время как отрицательные аномалии представляют интерес только с академической точки зрения, положительные всплески указывают на моменты времени, в которых инфраструктура не справляется с обработкой чрезмерно больших потоков данных. Заинтересованность в обнаружении пиков возникает по целому ряду причин.

- Если полученные значения оказываются фиктивными, то их нужно отсеять, чтобы исключить из списка критериев, по которым определяется пороговая нагрузка на инфраструктуру. Аномально большие числа мотивируют к приобретению более производительного вычислительного оборудования, а аномально низкие — нет. В процессе обнаружения аномалий вы дополнительно очистите исходные данные, однажды подвергнувшиеся предварительной обработке.
- При использовании низкопроизводительного компьютерного оборудования вам придется его обновить, чтобы учесть потребности, предопределяемые выявленными аномалиями. Если аномалии можно снабдить подписями, то это будет первым шагом к разметке данных и дальнейшему

их прогнозированию. Однако по определению аномалии плохо подлежат предсказанию, поэтому бо́льшая часть потраченных на это занятие усилий и времени не принесет желаемых результатов!

Пакет для автоматического обнаружения аномалий, разработанный Twitter, подлежит детальной параметризации (`https://perma.cc/BR4K-R8GL`). Это хороший инструмент, который можно использовать после исследования наборов временны́х рядов как с целью последующей очистки, так и для моделирования данных.[2]

Другие пакеты анализа временны́х рядов

В этой главе мы в основном рассмотрели популярные пакеты, разработанные крупнейшими технологическими компаниями, а также большие наборы данных и связанные прогнозы, на которых основывается деятельность таких компаний. Однако IT-гиганты далеки от того, чтобы быть основными поставщиками программного обеспечения для анализа временны́х рядов. Существует невероятно большая экосистема пакетов, предназначенных для работы с временны́ми рядами и призванных решать следующие задачи.

- Хранение временны́х рядов и построение сопутствующей инфраструктуры.
- Создание наборов временны́х рядов.
- Обнаружение точек излома.
- Прогнозирование.
- Частотный анализ данных.[3]
- Нелинейные временны́е ряды.
- Автоматизированное прогнозирование временны́х рядов.

Список можно продолжать до бесконечности. На рынке есть пакеты буквально для всего — десятки или даже сотни пакетов для обработки данных временны́х рядов. Профессор Роб Хиндман (Rob Hyndman) ведет учет списка пакетов с открытым исходным кодом для языка R на официальной веб-странице репозитория CRAN (`https://perma.cc/HWY6-W2VU`). Изучите его, чтобы получить

[2]Стоит отметить, что одновременно с пакетом `AnomalyDetection` компания Twitter разработала пакет для идентификации изменений уровня `BreakoutDetection`, который служит для выявления временны́х точек, в которых наблюдается сдвиг уровня. Этот пакет так же доступен и прост в использовании, хотя и не получил такого широкого признания, как `AnomalyDetection`. Существует множество альтернативных пакетов для обнаружения точек излома, которые протестированы и отлажены более тщательно.

[3]Частотный анализ данных в этой книге упоминается лишь кратко, но он остается важной областью анализа временны́х рядов и широко используется в некоторых научных дисциплинах, таких как физика и климатология.

представление о пакетах, которые соответствуют вашим аналитическим потребностям в конкретном проекте, а также в общеобразовательных целях — для знакомства с доступными инструментами анализа временных рядов и их роли в жизни всех нас. Как и в книге, на этой его странице приведены обзор и описание задач, связанных с вопросами анализа временных рядов. Для языка Python такой же полный и консолидированный список пакетов для работы с временными рядами не представлен, хотя более сокращенный перечень популярных решений можно найти на странице Макса Криста (Max Christ) (`https://perma.cc/GEQ3-Q54X`).

Дополнительные источники

StatsNewbie123, "Is it Possible to Automate Time Series Forecasting?" post on Cross Validated, December 6, 2019, `https://perma.cc/E3C4-RL4L`

> В этой недавней публикации на сайте StackExchange поднимается вопрос о возможности проведения автоматизированного прогнозирования данных абсолютного любого временного ряда. В ней приводятся две важные и подробно аргументированные точки зрения на этот вопрос, включая описание пакетов автоматизированного прогнозирования, а также обсуждение всех проблем решения такой задачи.

CausalImpact, Google's open source package for causal inference, `https://perma.cc/Y72Z-2SFD`

> Этот пакет компании Google с открытым исходным кодом построен поверх другого решения компании Google — `bsts` — пакета Bayesian Structural Time Series, который использован в главе 7. Пакет CausalImpact обращается к пакету `bsts` сначала для обучения моделей, а затем — для построения специальных контрольных примеров для оценки причинно-следственной связи и величины эффекта в данных временных рядах. Репозиторий пакета на GitHub содержит ссылки на соответствующие исследовательские работы и полезный видеоблог одного из создателей пакета. Также стоит ознакомиться с соответствующей научной статьей (`https://perma.cc/Q8K9-ZP7N`).

Murray Stokely, Farzan Rohani, and Eric Tassone, "Large-Scale Parallel Statistical Forecasting Computations in R," in JSM Proceedings, Section on Physical and Engineering Sciences (Alexandria, VA: American Statistical Association, 2011), `https://perma.cc/25D2-RVVA`

> В документе приведено подробное и очень качественное описание принципов написания пакетов компании Google на языке R, предназначенных для внутреннего использования при прогнозировании временных рядов, с высоким количеством параллельных вычислений и операций масштабирования

данных. Он позволяет не только прояснить важные аспекты прогнозирования данных, но и лучше понять принципы построения конвейеров обработки данных временны*х* рядов в крупных организациях с большими и различными типами временны*х* рядов.

Danilo Poccia, "Amazon Forecast: Time Series Forecasting Made Easy," AWS News Blog, November 28, 2018, `https://perma.cc/Y2PE-EUDV`

В обзоре, приведенном в этой главе, мы не уделили внимания еще одному популярному инструменту автоматизированного прогнозирования временны*х* рядов — службе прогнозирования Amazon Forecast. Она не является решением с открытым исходным кодом, хотя и получила много лестных отзывов. В ней предлагается проводить анализ временны*х* рядов с помощью моделей, разработанных компанией Amazon, в которых учитывается ее громадный опыт в ведении розничной торговли. Служба предназначена, в первую очередь, для составления бизнес-прогнозов для сторонних компаний. Хотя она распространяется на платной основе, ее протестировать ее можно совершенно бесплатно — ограниченная подписка предлагает довольно щедрый набор услуг. Amazon Forecast отличается высокой точностью и удобством в использовании и послужит хорошей альтернативой для тех организаций, которые находятся в поиске надежной модели построения прогнозов для больших объемов данных. В пакете Amazon применяется удачная комбинация методов глубокого обучения и традиционных статистических моделей, подобных простой модели LSTNET, рассмотренной в главе 10, в сочетании с моделью глубокого обучения с авторегрессионным компонентом. Также не забудьте познакомиться с фирменной архитектурой нейросетей Amazon, применяемой для прогнозирования, — DeepAR (`https://perma.cc/DNF9-LJKC`).

ГЛАВА 17

Прогнозы о прогнозировании

Есть много хороших цитат о безнадежности предсказания будущего, и все же я не могу не завершить эту книгу некоторыми мыслями о том, что нас ждет дальше.

Прогнозирование как услуга

Поскольку в прогнозировании временных рядов меньше специалистов-практиков, чем в других областях науки о данных, в их среде наблюдается стремление вывести анализ и прогнозирование временных рядов на уровень услуги, которая может быть легко упакована и развернута предельно эффективным способом. Например, как отмечалось в главе 16, Amazon недавно запустила службу прогнозирования временных рядов, и это не единственная компания, которая в этом преуспела. Модель компании Amazon выглядит преднамеренно универсальной и представляет прогнозирование всего лишь как отдельный шаг в конвейере обработки данных (рис. 17.1).

Рис. 17.1. *Пример конвейера в виде службы для прогнозирования временных рядов. Такие службы предлагаются как небольшими стартапами, так и крупными компаниями, среди которых выделяется Amazon, развернувшая набор инструментов, предназначенных для автоматического прогнозирования временных рядов и масштабирования на основе машинного обучения и статистических моделей*

Эти попытки моделирования как услуги ориентированы на создание достаточно хорошей общей модели, которая сможет работать в различных предметных областях без предоставления ужасающе неточных прогнозов. Большинство разработчиков описывают свои модели как полученные в результате комбинирования моделей глубокого обучения с традиционными статистическими моделями. Однако, поскольку служба в конечном итоге является "черным ящиком", трудно понять, что именно вызывает ошибки прогнозирования, и совершенно невозможно улучшить их с помощью методов ретроспективных исследований. Это означает, что прогнозы имеют достаточно высокий уровень точности, для которого, тем не менее, задается некое потолочное значение.

Эта услуга может быть востребована компаниями, заинтересованными в частом прогнозировании, но не располагающими достаточным количеством ресурсов, чтобы выполнить его собственными силами. Однако компаниям, накопившим значительные объемы исторических данных, в которых могут быть обнаружены общие эвристические закономерности, лучше обратиться к специалистам по анализу данных — их решение с большей вероятностью превзойдет автоматизированные алгоритмы, особенно учитывая хорошее знание ими предметной области.

Обратите внимание, что достаточное большое количество продуктов, предлагаемых как службы прогнозирования, снабжены хорошими инструментами визуализации целевых прогнозов и конвейерными утилитами, позволяющими легко пересматривать прогнозы, изменять частоту прогнозирования и т.д. Даже если ваша организация в конечном итоге решится на разработку собственного средства прогнозирования, всегда полезно сопоставить его с отраслевым стандартом.

Глубокое обучение расширяет возможности

За последние несколько лет многие крупнейшие технологические компании обнародовали частичные сведения о принципах построения прогнозов для своих наиболее важных услуг. Далее речь пойдет не о необходимости составления параллельных прогнозов большого количества показателей, оказывающих влияние на работу компании, а скорее о ключевых проблемах этого процесса. В случаях, когда качество прогноза имеет первостепенное значение, компании часто делают акцент на использовании глубокого обучения с вероятностным компонентом.

Например, компания Uber описала в своем блоге (`https://perma.cc/3W54-BK8C`) работоспособную службу прогнозирования спроса на поездки на автомобилях, а компания Amazon разработала рекуррентную сеть с авторегрессией (`https://perma.cc/UL77-BY3T`), вдохновленную статистическим подходом к анализу данных, для прогнозирования спроса на продукцию. Чем лучше в прогнозировании будут учитываться статистические методологии, такие как внедрение априорных знаний о предметной области и количественная оценка

неопределенности, тем меньше будет причин дополнять модели глубокого обучения статистическими моделями.

Однако создание правильно интерпретируемых моделей глубокого обучения — для понимания того, насколько неточным или экстремально завышенным может быть прогноз — остается трудной задачей, поэтому маловероятно, что традиционные статистические модели с глубоким теоретическим обоснованием и высокой степенью ясности будут отложены в сторону. При составлении прогнозов в критически важных предметных областях, в которых здоровье и человеческая жизнь могут подвергаться опасности, мы будем продолжать полагаться на то, что работало в течение многих десятилетий, до тех пор, пока не будут разработаны более прозрачные и проверяемые методы машинного обучения для прогнозирования.

Повышение важности машинного обучения

Складывается вполне обоснованное впечатление, что выверенные статистические методы все реже и реже используются для моделирования данных и составления прогнозов. Не отчаивайтесь: математическая статистика продолжает бурно развиваться и отвечать на интересные вопросы. И все же в реальном мире победителями выходят методы машинного обучения и ориентированные на результат статистические методы, а не причудливые теории и решения в замкнутой форме или доказательства сходимости, — особенно при построении прогнозов с низкими уровнем ответственности.

Для практикующих специалистов это хороший знак. Если вы оставили все трудности позади, не желая ничего доказывать, то вам не придется нести груз надлежащих доказательств. С другой стороны, это тревожная тенденция, поскольку технологии все глубже проникают во все фундаментальные аспекты жизни. Я не имею ничего против того, что на веб-сайте розничной компании будут использоваться технологии машинного обучения для угадывания моих будущих действий как покупателя. Но мне искренне хотелось бы, чтобы прогнозы для данных временны*х* рядов, которые моделируют состояние моего здоровья или успеваемость детей, были выверенными и статистически обоснованными, потому что предвзятая модель может нанести реальный вред людям, на которых оказывают непосредственное влияние результаты прогнозирования.

На данный момент лидеры в разработке методов анализа временны*х* рядов для промышленных приложений работают исключительно в области задач с низким уровнем ответственности. В таких задачах, как прогнозирование доходов от рекламной кампании или разворачивания продукта в социальных сетях, не критично, если прогнозы не подтверждаются полностью. Поскольку в список предметных областей моделирования входят и такие критически важные сферы, как

уход за людьми и производство продуктов питания, будем надеяться, что статистические методы будут продолжать играть главенствующую роль при составлении прогнозов в таких областях.

Повышение роли комбинированных моделей — статистического и машинного обучения

Ряд свидетельств указывает на преимущества комбинирования технологий машинного обучения и статистических методологий[1] перед даже самыми лучшими методами прогнозирования. Растущее признание такого подхода и использования ансамблевых методов прогнозирования — феномен, с которым мы сталкивались на протяжении всей книги.

Примером чрезвычайно надежного теста с множеством реальных наборов данных является недавнее соревнование M4 (`https://perma.cc/68AC-BKN7`) по анализу временны*х* рядов, которое заключалось в измерении точности прогнозирования наборов данных, состоящих из 100 000 временны*х* точек (`https://perma.cc/76BQ-SZW9`), кратко описанное в главах 2 и 9. Победитель в этом соревновании достиг успеха за счет объединения компонентов статистической модели и нейронной сети. Подобным образом участник, занявший второе место, использовал как алгоритмы машинного обучения, так и статистический подход, — в данном случае применяя ансамбль статистических моделей, а затем — дерево градиентного бустинга (`XGBoost`), чтобы выбрать относительные веса каждой модели в ансамбле. В этом примере мы видим два отличных способа комбинирования методов машинного обучения и статистического подхода: либо в виде отдельных моделей, собранных вместе (как в случае победителя), либо используя один из методов для определения метапараметров другого метода (как в случае второго призера). Всеобъемлющее и простое для изучения резюме результатов конкурса (`https://perma.cc/T8WW-6MDN`) было впоследствии опубликовано в журнале *International Journal of Forecasting*.

По мере того как такие комбинации набирают силу, мы становимся свидетелями исследовательского интереса к решению круга задач, наиболее подходящих для описания с помощью комбинированных статистических и машинных моделей обучения, а также передовых методов настройки точности таких моделей и выбора соответствующих архитектур. Тут ожидается получить такую же эволюцию, как в других сложных архитектурах, подобных нейронным сетям, благодаря

[1] Выражаю особую благодарность техническому рецензенту Робу Хиндману, который предложил (наряду с другими ценными советами по написанию книги) включить эту тему в материал книги.

которым возникают стандартные парадигмы проектирования с известными преимуществами, недостатками и методами обучения.

Больше прогнозов в повседневной жизни

Более ориентированные на потребителя компании, производящие такие целевые решения, как мобильные приложения для мониторинга состояния здоровья и поддержания здорового образа жизни, развернули (`https://perma.cc/QXT9-4B8T`) или собираются развернуть (`https://perma.cc/M8W7-EDCE`) персонализированные службы прогнозирования. По мере все большего осознания, насколько большие объемы персональных данных обрабатываются приложениями, пользователи стремятся использовать их для получения индивидуальных прогнозов в таких сферах интереса, как здоровый образ жизни и фитнес. Точно так же людям интересны прогнозы обо всем на свете: от будущих цен на недвижимость (`https://perma.cc/R5WR-T7XP`) (сложно предсказать) до даты прилета мигрирующих видов птиц (`https://perma.cc/5LTM-WRPB`).

Все больше программных продуктов в явном виде полагаются на прогнозы, будь то эзотерические темы или отдельные показатели. Это означает, что все чаще конвейеры прогнозирования будут интегрироваться в решения, в которых им раньше было не место, — в такие, как мобильные приложения и веб-сайты общей направленности, а не представляющие специфические отрасли и услуги. Чем чаще это будет происходить, тем быстрее терминология из области анализа и прогнозирования временны*х* рядов проникнет в нашу речь и станет неотъемлемой частью повседневной жизни. Хочется верить, что люди тоже будут стремиться получить образование, достаточное для изучения временны*х* рядов, чтобы понимать возможности и ограничения методов прогнозирования, не слишком полагаясь на возвращаемые ими результаты.

Предметный указатель

Л

М

Н

О

П

O'REILLY®

Практический анализ временных рядов

Анализ временных рядов становится все более и более актуальным разделом науки о данных в связи с широким распространением Интернета вещей, переходом здравоохранения на исключительно цифровой учет данных и ростом умных городов. Непрерывный мониторинг и сбор самых разных данных становятся повседневной действительностью и предопределяют постоянно растущую потребность в эффективных инструментах анализа временных рядов, основанных как на статистических методах, так и на методах машинного обучения.

В этом практическом руководстве описаны современные технологии анализа данных временных рядов и приведены примеры их практического использования в самых разных предметных областях. Оно призвано помочь в решении наиболее распространенных задач исследования и обработки временных рядов с помощью традиционных статистических методов и наиболее популярных моделей машинного обучения. В своей книге Эйлин Нильсен рассматривает самые распространенные и доступные инструменты анализа временных рядов, включенные в программные пакеты языков R и Python, которые могут применяться специалистами по работе с данными и разработчиками программного обеспечения для написания собственных эффективных решений.

Основные темы книги:

- **Поиск и извлечение временных рядов**
- **Глубокое исследование временных рядов**
- **Хранение временных данных**
- **Моделирование данных временных рядов**
- **Генерирование и отбор признаков для временных рядов**
- **Классификация и прогнозирование временных рядов с помощью методов машинного и глубокого обучения**
- **Оценка ошибок прогнозирования**
- **Оценка точности и производительности моделей**

"Незаурядное издание! Рано или поздно любому специалисту по анализу данных приходится работать с временными рядами или с подобными им технологическими данными. В этой книге вы найдете детальное описание методологий машинного обучения и обработки временных данных, сопровождаемое великолепными примерами их практической реализации."

Андреас В. Кемпа-Лир,
старший преподаватель, факультет инженерных наук, Оклендский университет

Эйлин Нильсен — разработчик программного обеспечения и специалист по анализу данных из Нью-Йорка. Она занимается обработкой данных временных рядов в самых разных предметных областях и научных дисциплинах — здравоохранении, политических кампаниях, научно-исследовательской деятельности и биржевой торговле. За свою карьеру она разработала несколько алгоритмов прогнозирования, основанных на нейронных сетях.

ISBN 978-5-907365-04-9

Категория: машинное обучение/математическая статистика/прогнозирование

ДИАЛЕКТИКА

http://www.williamspublishing.com
http://www.oreilly.com