Матанализ 2
Производные функций нескольких переменных

Денис Волк
Senior Data Scientist @ KPMG

- Математик, кандидат наук, МГУ
- Специальность: динамические системы и случайные процессы
- Работал в университетах Триеста, Рима, Стокгольма
- Автор 14 научных статей в международных журналах

Денис Волк

Senior Data Scientist @ KPMG

Функция нескольких переменных

Функция — это некоторое соответствие **x → f(x)**, причём для каждого **x** задано единственное значение **f(x)**.

Теперь **x** это не число, а вектор! $x = (x_1, \ldots, x_n)$

А **f(x)** по-прежнему число.

D(f) — область определения функции
E(f) — область значений функции

$$E(f) = f(D(f))$$

Теперь **D(f)**— подмножество $\mathbb{R}^n$, а **E(f)** — подмножество $\mathbb{R}$.

Частная производная

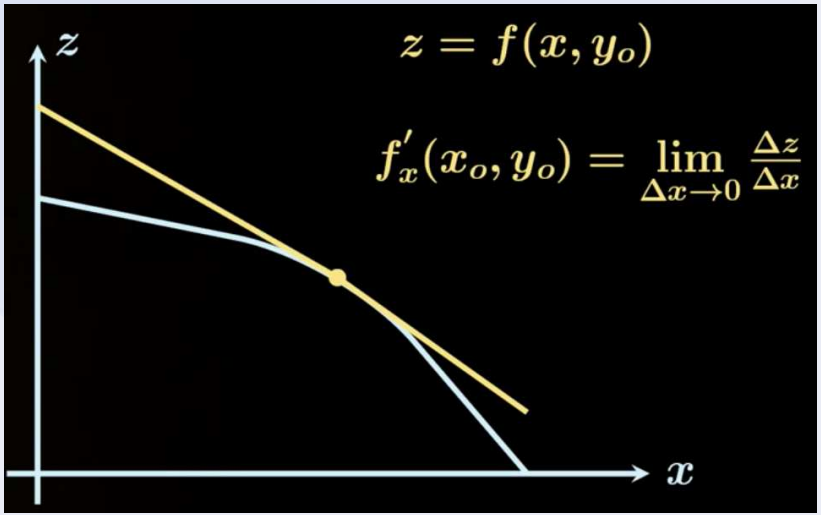
Частная производная функции **f(x, y)** по **x** определяется как производная по **x**, взятая в смысле функции одной переменной, при условии постоянства оставшейся переменной **y**.

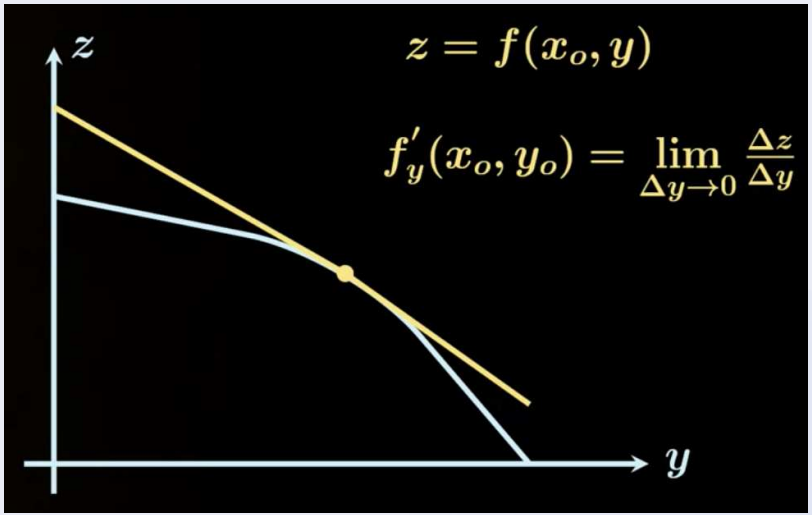
$$f'_x(x,y) = \lim_{h\to 0} \frac{f(x+h,y) - f(x,y)}{h}, \qquad f'_y(x,y) = \lim_{h\to 0} \frac{f(x,y+h) - f(x,y)}{h}.$$

Обозначение:

$f'_x \quad f'_y$ или $\frac{\partial f}{\partial x} \quad \frac{\partial f}{\partial y}$ или иногда $\frac{\Delta f}{\Delta x} \quad \frac{\Delta f}{\Delta y}$

Производные функций нескольких переменных





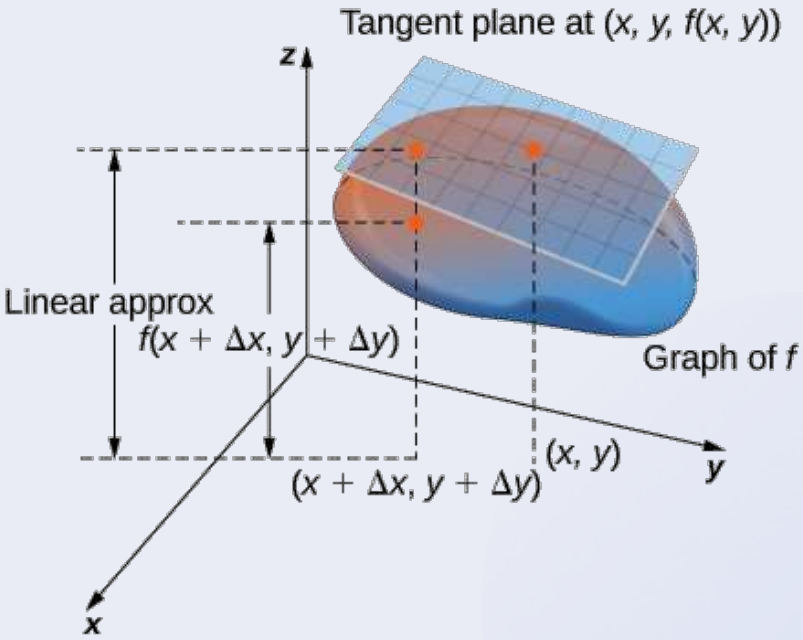
https://www.coursera.org/learn/mathematics-and-python

Касательная плоскость

$$f(x_0 + \Delta x, y_0 + \Delta y) \approx f(x_0, y_0) + f'_x(x_0, y_0)\Delta x + f'_y(x_0, y_0)\Delta y$$

График **z = f(x, y)** — некоторая поверхность в трехмерном пространстве.

Если в некоторой точке **(x_0, y_0)** функция дифференцируема как функция многих переменных, то правая часть — касательная плоскость к графику в этой точке.



https://openstax.org/books/calculus-volume-3/pages/4-4-tangent-planes-and-linear-approximations

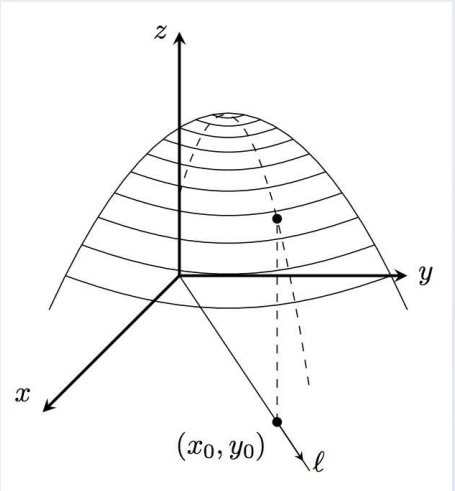
Производная по направлению

Пусть $f(\vec{x}) = f(x_1, x_2, \ldots, x_n)$ — функция n переменных, $\vec{\ell} \in \mathbb{R}^n$, $|\vec{\ell}| = 1$, тогда частной производной в точке x_0 по направлению $\vec{\ell}$ называется

$$\frac{\partial f}{\partial \vec{\ell}}(\vec{x}_0) = \lim_{t \to 0} \frac{f(\vec{x}_0 + t \cdot \vec{\ell}) - f(\vec{x}_0)}{t}.$$

Если функция дифференцируема, то производная по любому направлению существует.

Производная по направлению показывает, насколько быстро функция изменяется при движении вдоль заданного направления.



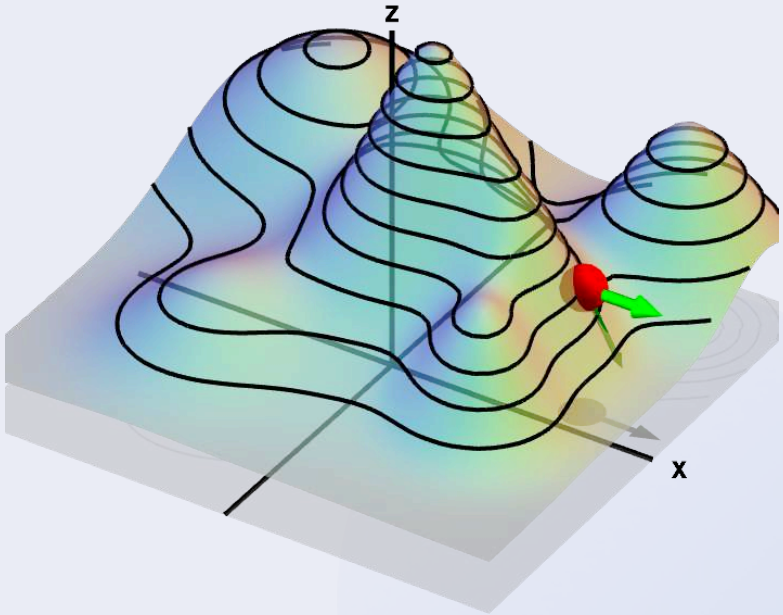
Градиент и линии уровня функции

Если $f(x_1, \ldots, x_n)$ — функция **n** переменных $x_1, \ldots, x_n$, то **n**-мерный вектор из частных производных, взятых в одной и той же точке **x**

$$\operatorname{grad} f = \nabla f = \left(\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n} \right)$$

называется **градиентом** функции.

Линией уровня функции называется множество точек, в которых функция принимает одно и то же фиксированное значение. Оказывается, что градиент перпендикулярен линии уровня.



https://mathinsight.org/directional_derivative_gradient_introduction

Связь градиента и производной по направлению

Если функция дифференцируема в (x0,y0), то в окрестности её можно приблизить линейно:

$$\Delta f = f(x_0 + \Delta x, y_0 + \Delta y) - f(x_0, y_0) \approx f'_x(x_0, y_0)\Delta x + f'_y(x_0, y_0)\Delta y = \left\langle \nabla f(x_0, y_0), \left(\Delta x, \Delta y\right) \right\rangle$$

Пусть $\vec{\ell} \in \mathbb{R}^2$, $|\vec{\ell}| = 1$, тогда приращения можно задать вдоль вектора $\vec{\ell}$:

$$\Delta x = t \cdot \ell_x, \qquad \Delta y = t \cdot \ell_y.$$

Подставим в первое выражение:

$$\Delta f \approx \left\langle \nabla f(x_0, y_0), \begin{pmatrix} t \cdot \ell_x \\ t \cdot \ell_y \end{pmatrix} \right\rangle = t \cdot \left\langle \nabla f(x_0, y_0), \vec{\ell} \right\rangle$$

Связь градиента и производной по направлению

Производная по направлению дифференцируемой функции **f** равна проекции градиента функции на это направление, или иначе, скалярному произведению градиента на единичный вектор направления **l**:

$$\frac{\partial f}{\partial \bar{l}} = (\nabla f, \bar{l})$$

Следовательно, направление, вдоль которого производная по направлению **максимальна**, есть направление градиента функции в данной точке: скалярное произведение максимально, когда векторы сонаправлены.

Градиент в задачах оптимизации

Задача оптимизации — найти экстремум функции, например, минимум:

$$f(x_1, \ldots, x_n) \to \min$$

Часто встречается в приложениях.

Любой метод машинного обучения при обучении ищет оптимальные параметры, для которых ошибка на обучающих данных минимальна — задача оптимизации!

В случае функций нескольких переменных необходимое условие экстремума:
градиент равен нулю
(т.е. все частные производные равны нулю).

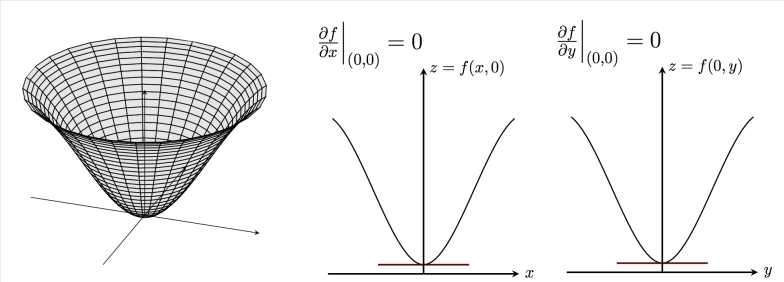


Рис. 2: Функция двух переменных достигает минимума в начале координат.

Градиентный спуск

Задачу минимизации функции **n** переменных редко можно решить аналитически. В таком случае используется численная оптимизация. Наиболее простым методом является **градиентный спуск**.

Идея: идти в направлении наискорейшего спуска, а это направление задаётся антиградиентом $-\nabla f$

Итерационный метод:

1) начинаем с некоторого начального положения: $x^{(0)}$
2) делаем шаг в направлении антиградиента: $x^{(1)} = x^{(0)} - \gamma \cdot \nabla f(x^{(0)})$
3) повторяем $x^{(i+1)} = x^{(i)} - \gamma \cdot \nabla f(x^{(i)})$
4) пока не выполнено условие остановки:
    1) градиент стал почти нулевым
    2) уменьшение значений функции почти перестало происходить

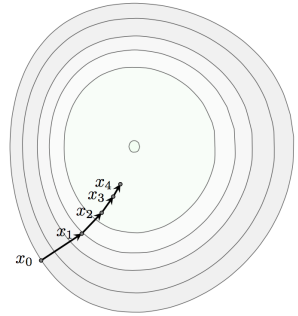


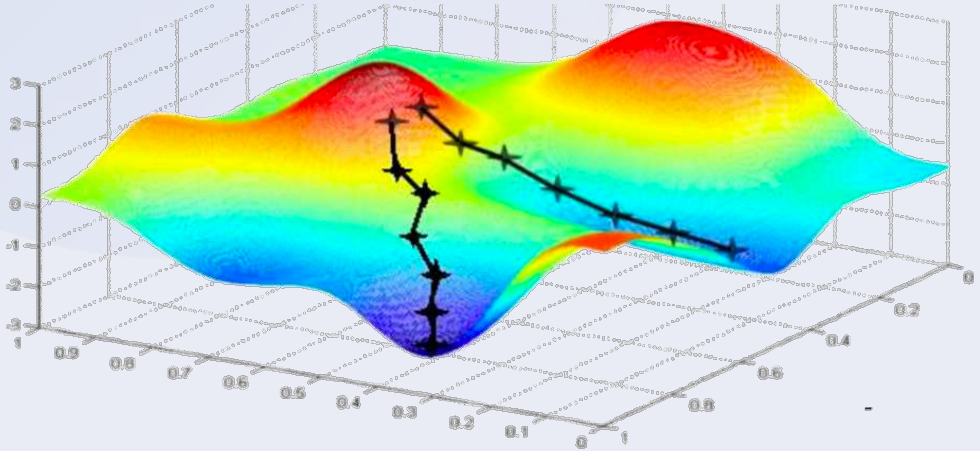
Рис. 3: Градиентный спуск

Локальные и глобальные экстремумы

Нет гарантии, что спустимся в **глобальный** минимум.

Более того, спуск может “застрять” на “плато”.

Можно попробовать начинать из разных точек, и выбрать из результатов самый минимальный минимум



Хорошо распараллеливается

https://shashank-ojha.github.io/ParallelGradientDescent/

Применение: обучение нейросетей

Нейросеть — сложная функция **F**: $X = (x_1, \ldots, x_n) \xrightarrow{F} y$

Зависит от большого числа параметров (м.б. миллионов): $F(X) = F_{a_1,\ldots,a_N}(X)$

Функция потерь: $\mathrm{Loss}(X, y)$ (Например, расстояние^2 между **F(X)** и истинным **y**)

Обучение: найти параметры $\mathbf{a_j}$, минимизирующие ошибки на обучающем множестве:

$$L = \sum_i \mathrm{Loss}(X^{[i]}, y^{[i]}) \to \min$$

(i — номер примера в обучающем множестве)

Решение: градиент **L** по параметрам и спуск $\nabla L = \left(\frac{\partial L}{\partial a_1}, \ldots, \frac{\partial L}{\partial a_N} \right)$

Авто вычисление градиента

autograd, tensorflow, pytorch, theano и др.

```
import tensorflow as tf
```

```
x = tf.ones((2, 2))

with tf.GradientTape() as t:
  t.watch(x)
  y = tf.reduce_sum(x)
  z = tf.multiply(y, y)

# Derivative of z with respect to the original input tensor x
dz_dx = t.gradient(z, x)
```

https://www.tensorflow.org/tutorials/customization/autodiff

Стохастический градиентный спуск (SGD)

$$\nabla L = \nabla \sum_i \text{Loss}\left(X^{[i]}, y^{[i]}\right) = \left(\sum_i \frac{\partial \text{Loss}\left(X^{[i]}, y^{[i]}\right)}{\partial a_1}, \ldots, \sum_i \frac{\partial \text{Loss}\left(X^{[i]}, y^{[i]}\right)}{\partial a_N} \right)$$

Упрощение: на каждом шаге вычислять не все суммы, а только их **i**-ые слагаемые.

Несколько “эпох”, в каждой заново перетасовываем обучающий набор.

В каждой следующей эпохе уменьшаем learning rate **γ**: $x^{(i+1)} = x^{(i)} - \gamma \cdot \nabla f(x^{(i)})$

Если функция потерь **L** выпукла (и правильно уменьшаем **γ**), то SGD почти наверное сходится к глобальному минимуму **L**. Иначе — к локальному.

Развития идеи: momentum, mini-batches, адаптивный learning rate, …

Summary

Узнали:

1. Функции нескольких переменных
2. Частные производные
3. Линеаризация функции
4. Градиент
5. Связь с задачами оптимизации
6. Градиентный спуск в машинном обучении
7. Стохастический градиентный спуск