Современные методы анализа данных и машинного обучения

Тема 5. Лекция 6

Математика для машинного обучения. Математический анализ

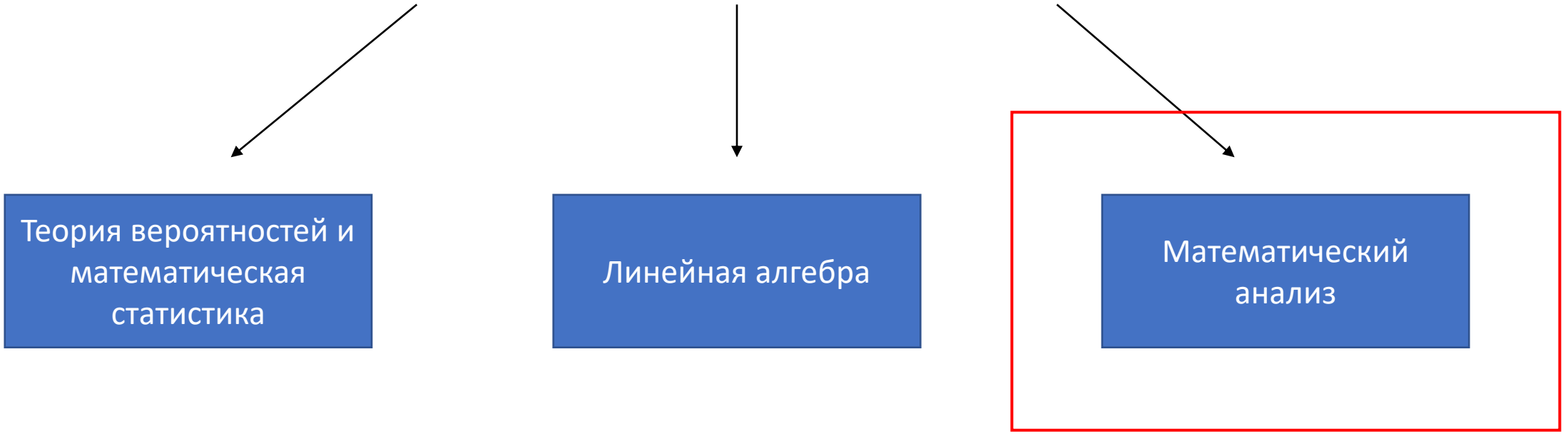
Юрий Саночкин

ysanochkin@hse.ru

НИУ ВШЭ, 2024

Математика для анализа данных

Математика для анализа данных



Сегодня, как мы и договаривались, поговорим об этом!

Математический анализ. Мотивация

Математический анализ. Мотивация

- Пойдем по уже знакомому нам паттерну и обсудим для начала, почему математический анализ в принципе так важен в рамках задач машинного обучения.
- Как вы сами можете ответить на этот вопрос: про что вообще этот раздел и в чем, на ваш взгляд, заключается его значимость?

Математический анализ. Мотивация

- Дифференциальная оптимизация – пламенный мотор машинного обучения!

Математический анализ. Мотивация

- Дифференциальная оптимизация – пламенный мотор машинного обучения!
- Не слишком ли громко сказано?

Математический анализ. Мотивация

- Дифференциальная оптимизация – пламенный мотор машинного обучения!
- Не слишком ли громко сказано?
- Вовсе нет.
- Будем продвигаться шаг за шагом в математическом анализе и постепенно поймём, почему в действительности всё так абсолютно и есть.

Математический анализ. Мотивация

- Давайте вспомним: в чем заключается основная задача классического машинного обучения?

Математический анализ. Мотивация

- Давайте вспомним: в чем заключается основная задача классического машинного обучения?
- Основная задача — как можно более точно восстановить скрытую (неизвестную) зависимость в данных.

Математический анализ. Мотивация

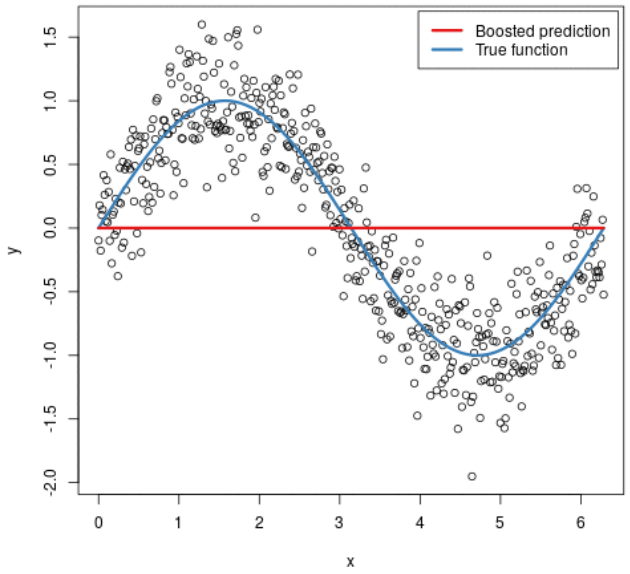
- Давайте вспомним: в чем заключается основная задача классического машинного обучения?
- Основная задача — как можно более точно восстановить скрытую (неизвестную) зависимость в данных.
- Зависимость эта может быть функциональной; иметь стохастическую природу; представлять собой кластерную структуру, порождающую распределение данных; или же быть чем-либо ещё. В сущности, это не так здесь и важно.
- Самое главное, что машинное обучение — это в любом случае про определение и детектирование подобной зависимости.

Математический анализ. Мотивация

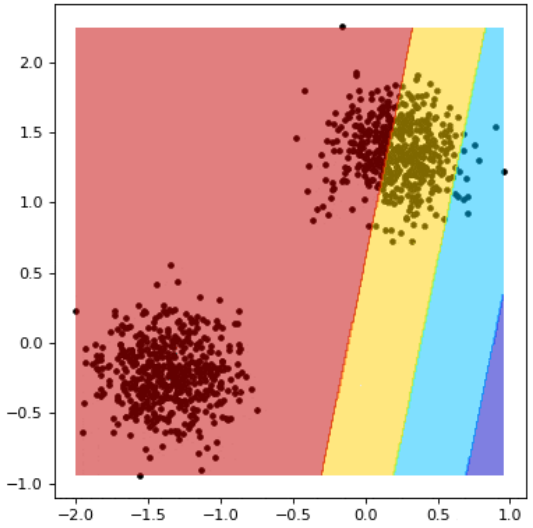
- Поэтому еще раз концептуально вспомним, как это всё работает:

Математический анализ. Мотивация

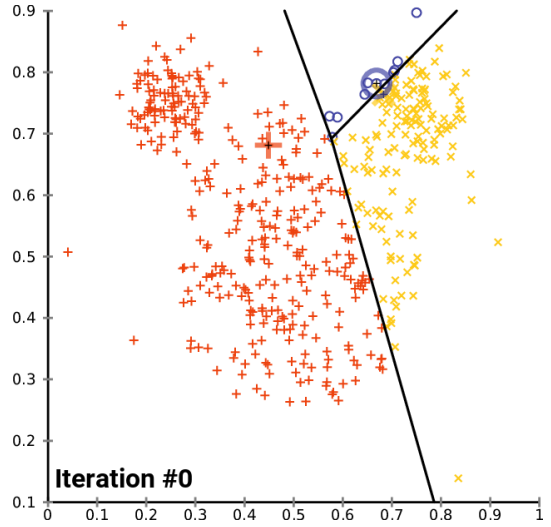
- Поэтому еще раз концептуально вспомним, как это всё работает:



Решение задачи регрессии при помощи градиентного бустинга



Решение задачи классификации при помощи метода опорных векторов



Решение задачи кластеризации при помощи метода KMeans

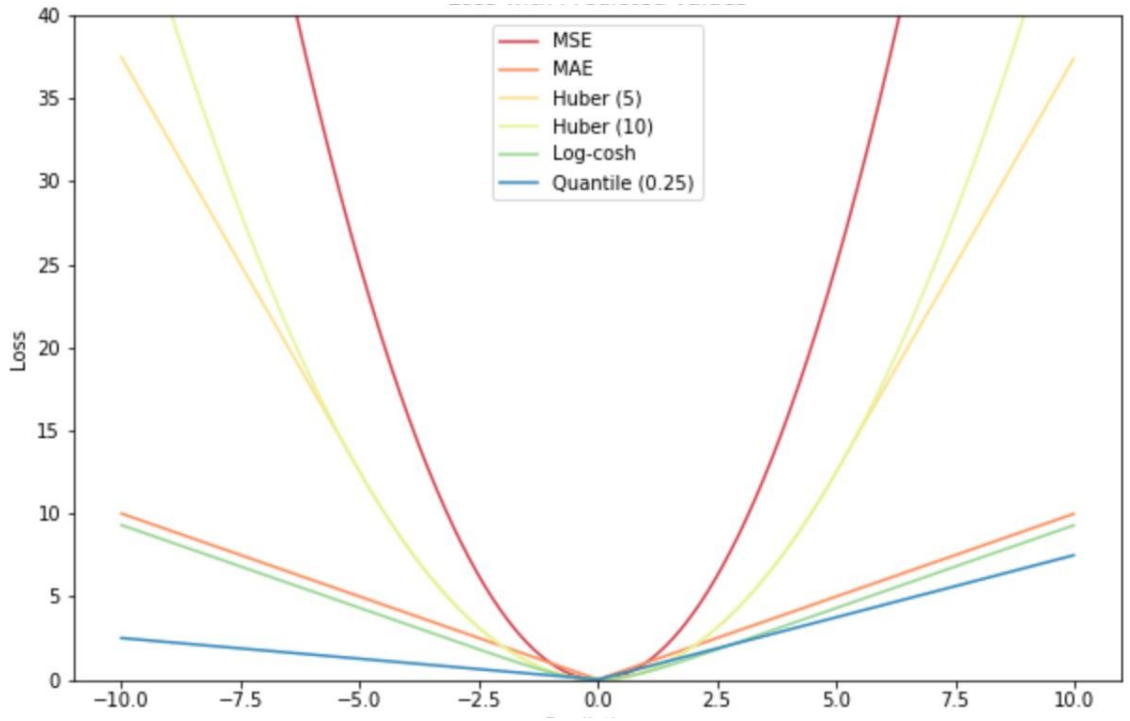
Математический анализ. Мотивация

- Но и проблема, которая сразу же появляется тут, очень понятна: а как вообще определить, хорошо алгоритм справляется со своей задачей или же как-то не очень?
- Да и вообще в целом — что такое хорошо, а что такое плохо в контексте ML? Как объяснить это компьютеру?

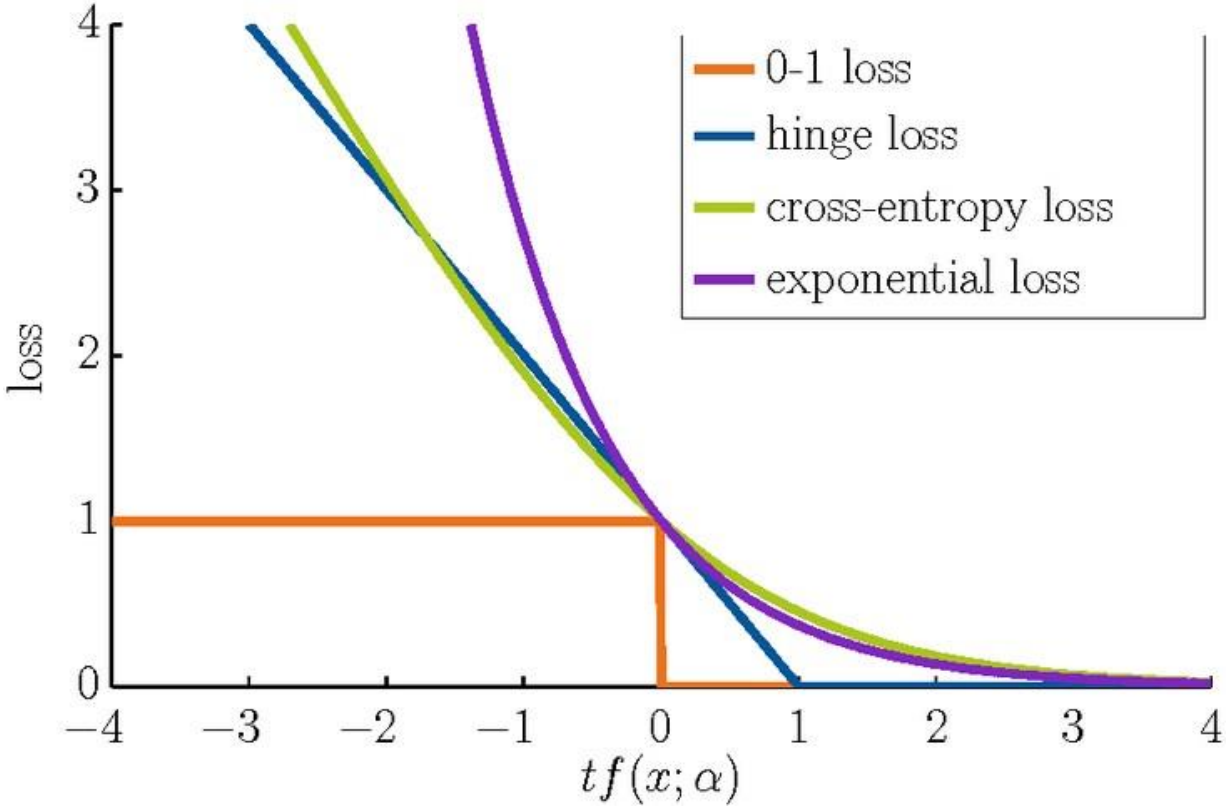
Математический анализ. Мотивация

- Но и проблема, которая сразу же появляется тут, очень понятна: а как вообще определить, хорошо алгоритм справляется со своей задачей или же как-то не очень?
- Да и вообще в целом — что такое хорошо, а что такое плохо в контексте ML? Как объяснить это компьютеру?
- Оказывается, при обучении наш алгоритм практически всегда старается минимизировать некую функцию, — т.н. функцию ошибки, — причем делает он это на тех данных, на которых собственно и обучается!

Функция ошибки



Типичные функции ошибки в задаче регрессии



Типичные функции ошибки в задаче бинарной классификации

Функция ошибки

- Не углубляясь пока супер детально в различные разновидности функций ошибок, для нас сейчас важно понять главное: о какой бы задаче машинного обучения ни шла речь, практически любой алгоритм, её решающий, будет явным или неявным образом минимизировать некоторый функционал.

Функция ошибки

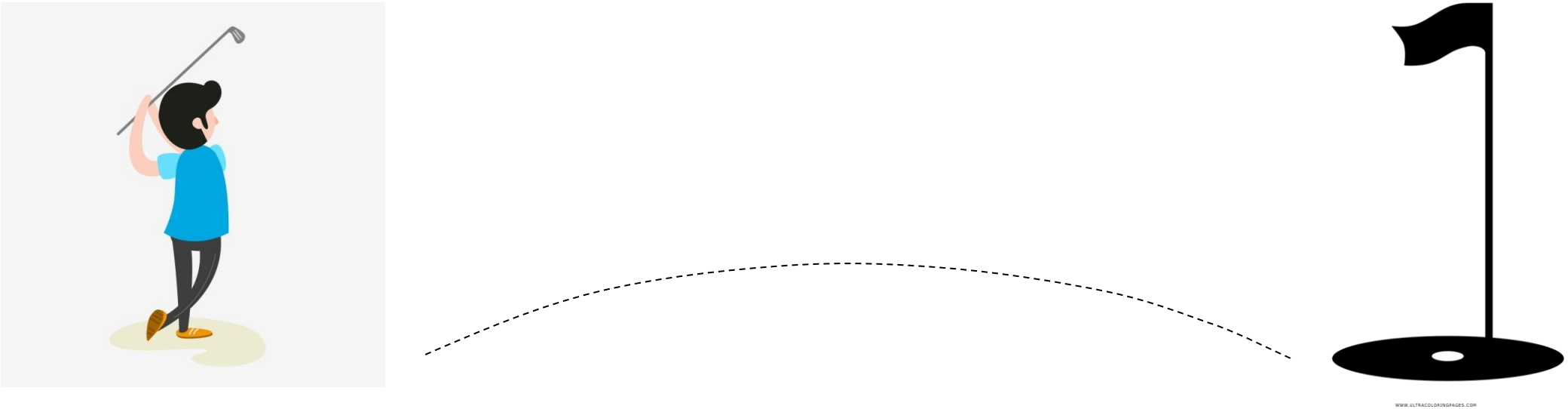
- Не углубляясь пока супер детально в различные разновидности функций ошибок, для нас сейчас важно понять главное: о какой бы задаче машинного обучения ни шла речь, практически любой алгоритм, её решающий, будет явным или неявным образом минимизировать некоторый функционал.
- В контексте ML минимизация этого функционала происходит посредством подбора параметров алгоритма. Причем параметров этих может быть сколько угодно: как один, два или пара десятков; так и миллионы и даже миллиарды (как в случае нейросетей в духе GPT и тому подобного).

Функция ошибки

- Наглядный пример оптимизации:

Функция ошибки

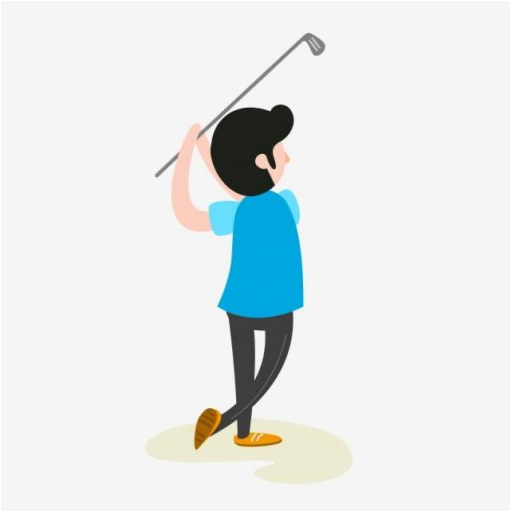
- Наглядный пример оптимизации:

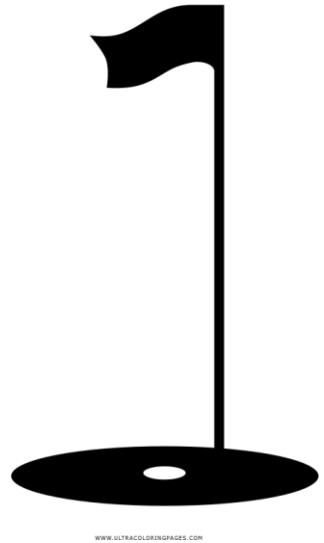


Функция ошибки

- Наглядный пример оптимизации:

Игрок — ML-алгоритм





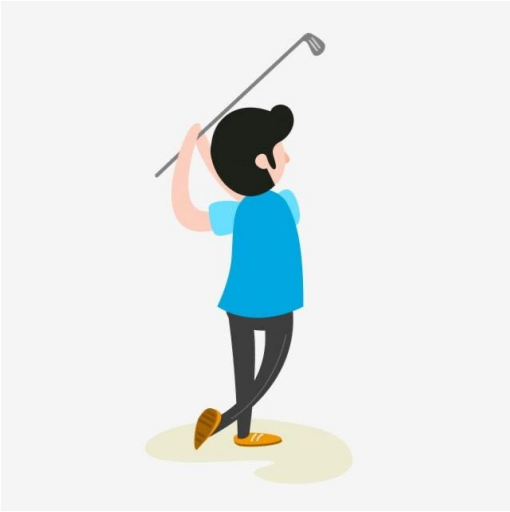
Оптимизируемый параметр — сила удара

Функция ошибки — то, на сколько игрок промахнулся по лунке

Функция ошибки

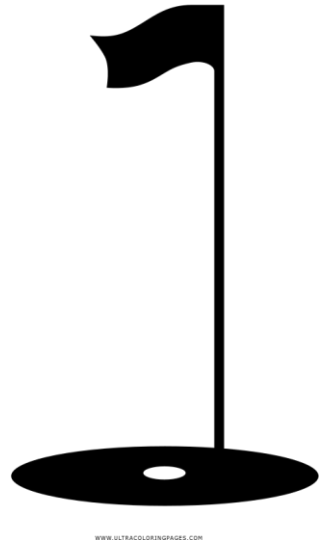
- Наглядный пример оптимизации:

Игрок — ML-алгоритм



Оптимизируемый параметр — сила удара

Алгоритм оптимизации — это мы!



Функция ошибки — то, на сколько игрок промахнулся по лунке

Функция ошибки

- Общая схема обучения ML-алгоритмов:

Функция ошибки

- Общая схема обучения ML-алгоритмов:
- Пока ошибка уменьшается
  - Сделать предсказания на обучающей выборке

Функция ошибки

- Общая схема обучения ML-алгоритмов:
- Пока ошибка уменьшается
  - Сделать предсказания на обучающей выборке
  - Посчитать функцию ошибки

Функция ошибки

- Общая схема обучения ML-алгоритмов:
- Пока ошибка уменьшается
  - Сделать предсказания на обучающей выборке
  - Посчитать функцию ошибки
  - Обновить параметры модели так, чтобы минимизировать функцию ошибки

Функция ошибки

- Ну, хорошо. Допустим, как посчитать функцию ошибки и понять, меняется ли она, — ясно.

Функция ошибки

- Ну, хорошо. Допустим, как посчитать функцию ошибки и понять, меняется ли она, — ясно.
- А как изменить параметры таким образом, чтобы алгоритм ошибался меньше?

Функция ошибки

- Ну, хорошо. Допустим, как посчитать функцию ошибки и понять, меняется ли она, — ясно.
- А как изменить параметры таким образом, чтобы алгоритм ошибался меньше?
  - Проблема. Параметров очень много.

Функция ошибки

- Ну, хорошо. Допустим, как посчитать функцию ошибки и понять, меняется ли она, — ясно.
- А как изменить параметры таким образом, чтобы алгоритм ошибался меньше?
  - Проблема. Параметров очень много.
  - Проблема. Данных тоже много.

Функция ошибки

- Ну, хорошо. Допустим, как посчитать функцию ошибки и понять, меняется ли она, — ясно.
- А как изменить параметры таким образом, чтобы алгоритм ошибался меньше?
  - Проблема. Параметров очень много.
  - Проблема. Данных тоже много.
  - Проблема. Параметры нужно обновлять не только правильно, но и желательно быстро.

Математический анализ. Мотивация

- И вот оказывается, ответы на эти (а также многие другие) вопросы кроются в разделе математики под названием математический анализ; а если конкретнее — подразделе математического анализа под названием “дифференциальное исчисление” (скалярных функций многих переменных).
- Данная трехсотлетняя наука изначально создавалась под нужды физиков; тем не менее, она не теряет своей актуальности и по сей день, ведь без неё какое-либо машинное обучение было бы попросту невозможно.

Математический анализ. Мотивация

- Именно поэтому на сегодняшней лекции мы вначале вспомним ключевые понятия дифференциального исчисления.
- А затем увидим, как из них естественным образом возникают т.н. методы первого порядка для решения задач безусловной непрерывной оптимизации.

Математический анализ. Мотивация

- Именно поэтому на сегодняшней лекции мы вначале вспомним ключевые понятия дифференциального исчисления.
- А затем увидим, как из них естественным образом возникают т.н. методы первого порядка для решения задач безусловной непрерывной оптимизации.
- Ну что, вы готовы?
- Поехали! :)

Основные понятия

Дифференциальное исчисление

- Дифференциальное исчисление — раздел математического анализа.

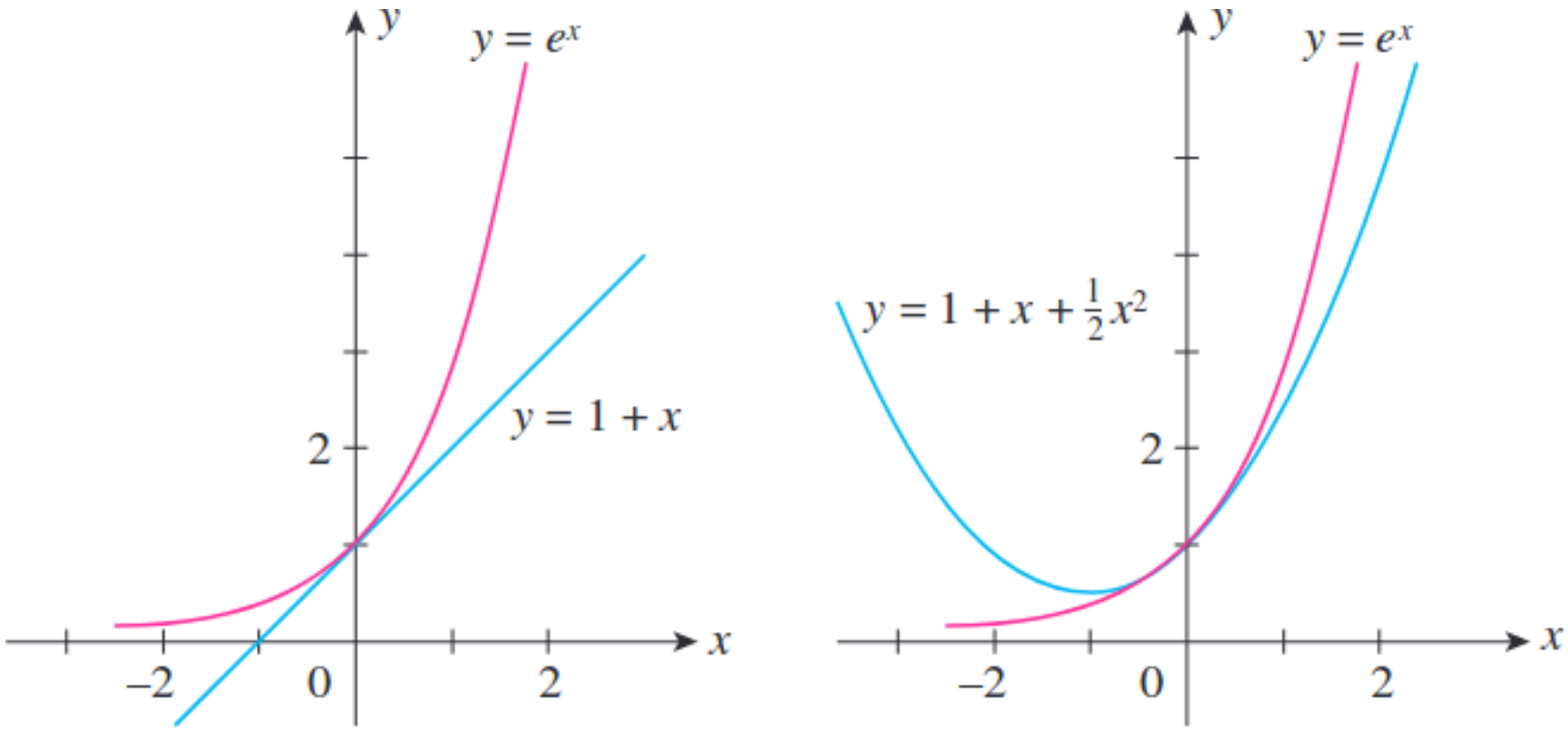
Дифференциальное исчисление

- Дифференциальное исчисление — раздел математического анализа.
- Изначальные задачи этой науки:
  - Анализ поведения функций в малой окрестности точки;

Дифференциальное исчисление

- Дифференциальное исчисление — раздел математического анализа.
- Изначальные задачи этой науки:
  - Анализ поведения функций в малой окрестности точки;
  - Локальное приближение сложных функций простыми и понятными математическими объектами в духе многочленов или тригонометрических сумм.

Дифференциальное исчисление



Приближение первого и второго порядка
для функции $y = e^x$ в окрестности точки 0

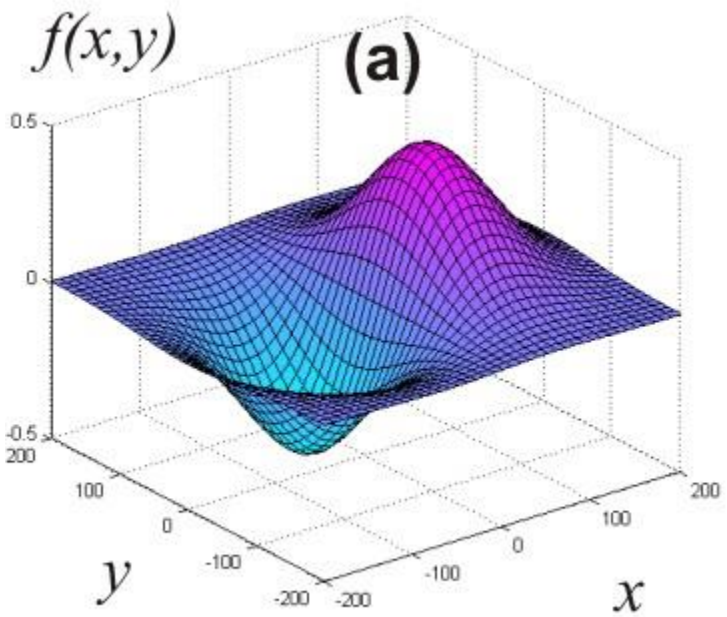
Дифференциальное исчисление

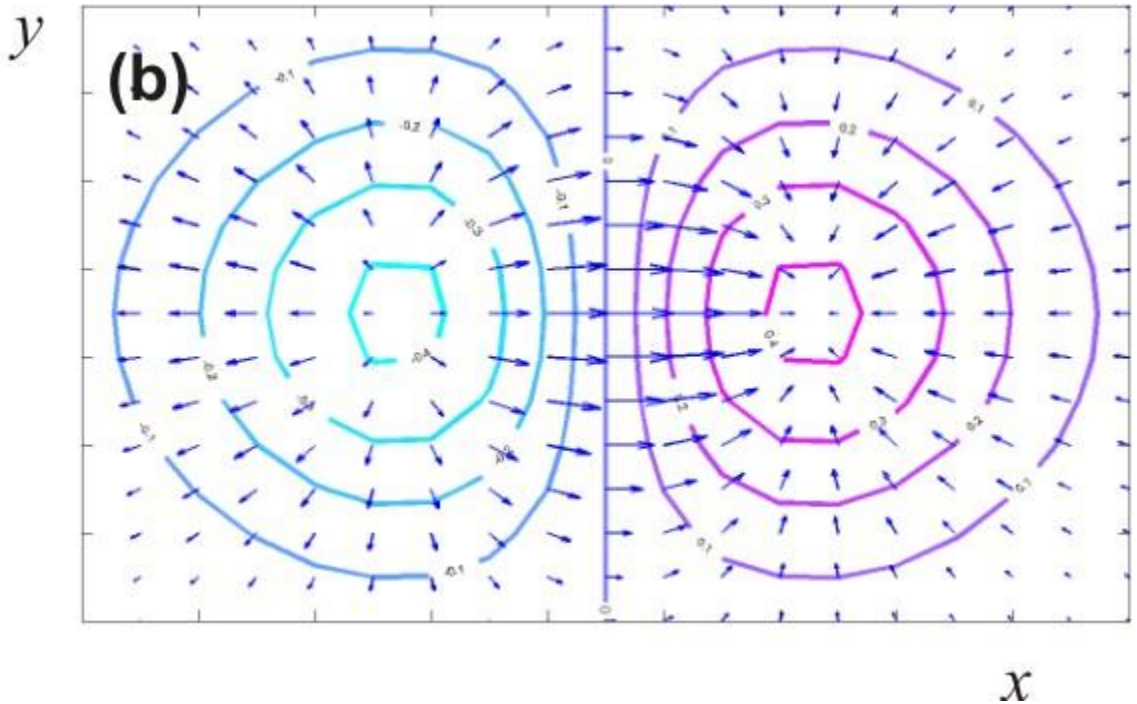
- Для большей части методов машинного обучения достаточно понимать, что такое первая производная.
- Знание первой производной в точке позволяет строить линейное приближение функции в малой окрестности этой точки.

Дифференциальное исчисление

- Для большей части методов машинного обучения достаточно понимать, что такое первая производная.
- Знание первой производной в точке позволяет строить линейное приближение функции в малой окрестности этой точки.
- Оказывается, этого достаточно для того, чтобы понять, в каком направлении значение функции возрастает (или убывает) быстрее всего.

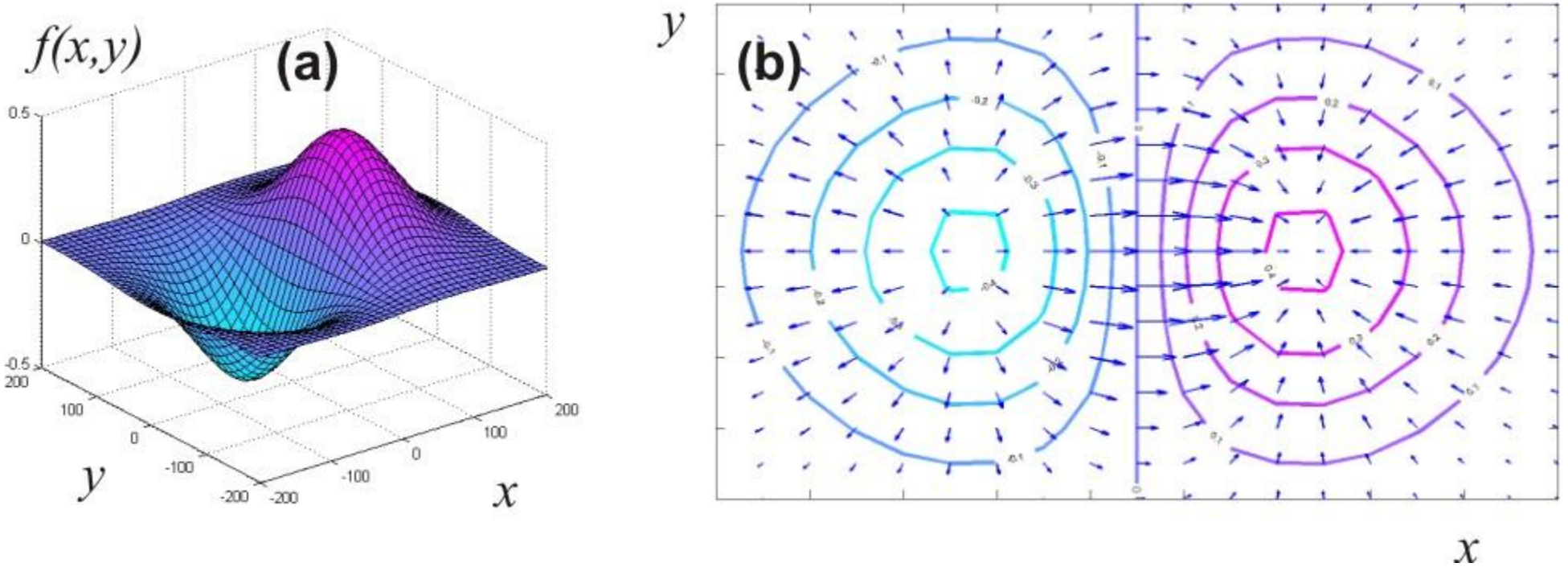
Первая производная





- На уровне интуиции, первая производная позволяет построить “карту высот” вашей функции.

Первая производная



- В каждой точке вы будете знать, в каком направлении от неё находится наиболее крутой подъём или спуск.

Первая производная

- Итак, зафиксируем цель: вспомнить и полноценно повторить понятие первой производной, а также то, как она связана с поиском локального оптимума (минимума или максимума) для достаточно “гладких” функций.

Первая производная

- Итак, зафиксируем цель: вспомнить и полноценно повторить понятие первой производной, а также то, как она связана с поиском локального оптимума (минимума или максимума) для достаточно “гладких” функций.
- Понятия математического анализа находятся друг с другом в строгой связке и иерархии.
- А потому, чтобы перейти к предметному разговору про производную, нам сперва нужно вспомнить еще несколько важных, предваряющих её понятий.

Первая производная

- Итак, зафиксируем цель: вспомнить и полноценно повторить понятие первой производной, а также то, как она связана с поиском локального оптимума (минимума или максимума) для достаточно “гладких” функций.
- Понятия математического анализа находятся друг с другом в строгой связке и иерархии.
- А потому, чтобы перейти к предметному разговору про производную, нам сперва нужно вспомнить еще несколько важных, предваряющих её понятий.
- Сделаем это!

Предел функции в точке

- Ну что же, и первым таким определением будет определение предела функции в точке.
- Помните ли вы, что это такое?

Предел функции в точке

- Ну что же, и первым таким определением будет определение предела функции в точке.
- Помните ли вы, что это такое?
- Своими словами:
- Если у функции f существует предел y_0 в точке x_0, то, по какой бы траектории в области определения функции вы ни подходили к точке x_0, значения функции в точках траектории будут становиться всё ближе и ближе к y_0.

Предел функции в точке

Непрерывность функции в точке

- Следующее понятие, которое нам понадобится, — это понятие непрерывности функции в точке. Данное понятие тесно связано с понятием предела (формальное определение так вообще почти такое же).
- Кто нам тут с ним поможет?

Непрерывность функции в точке

- Следующее понятие, которое нам понадобится, — это понятие непрерывности функции в точке. Данное понятие тесно связано с понятием предела (формальное определение так вообще почти такое же).
- Кто нам тут с ним поможет?
- Своими словами:
- Если функция f непрерывна в x_0, то, независимо от того, по какой траектории мы подходим к точке x_0, мы получим в пределе значение $f(x_0)$

Непрерывность функции в точке

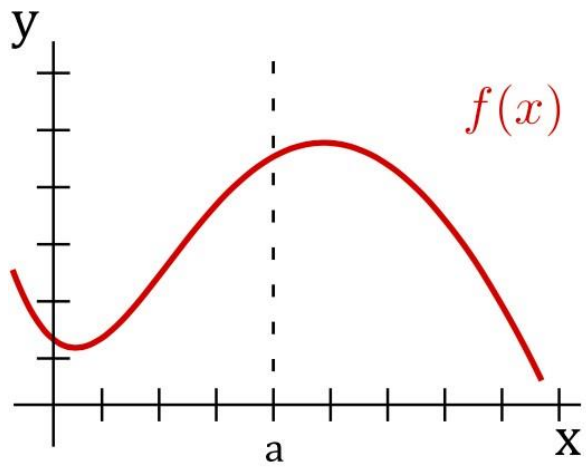
- Следующее понятие, которое нам понадобится, — это понятие непрерывности функции в точке. Данное понятие тесно связано с понятием предела (формальное определение так вообще почти такое же).
- Кто нам тут с ним поможет?
- Своими словами:
- Если функция f непрерывна в x_0, то, независимо от того, по какой траектории мы подходим к точке x_0, мы получим в пределе значение $f(x_0)$
  - То есть, по сути, в определении предела просто заменяем y_0 на $f(x_0)$.

Непрерывность функции в точке

- Большинство функций, с которыми мы будем иметь дело в этом курсе, непрерывны.
- Тем не менее, это далеко не “бесплатное” свойство.
- Оно достаточно часто нарушается, но при этом почти везде требуется.
- Обязательно обращайте на это внимание в своих рассуждениях.

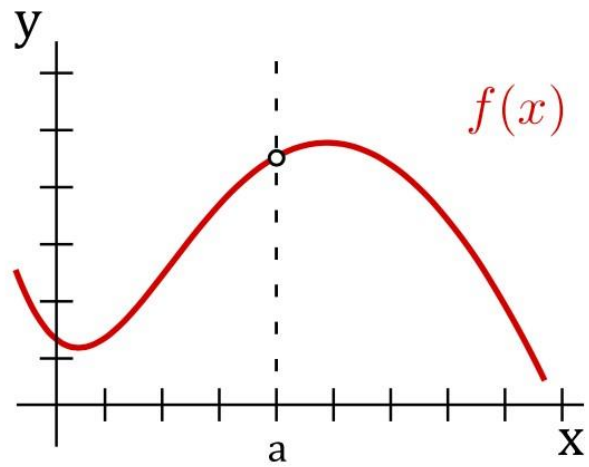
Непрерывность функции в точке

- Большинство функций, с которыми мы будем иметь дело в этом курсе, непрерывны.
- Тем не менее, это далеко не “бесплатное” свойство.
- Оно достаточно часто нарушается, но при этом почти везде требуется.
- Обязательно обращайте на это внимание в своих рассуждениях.
- Кстати, как вообще можно классифицировать случаи, когда функция не является непрерывной?



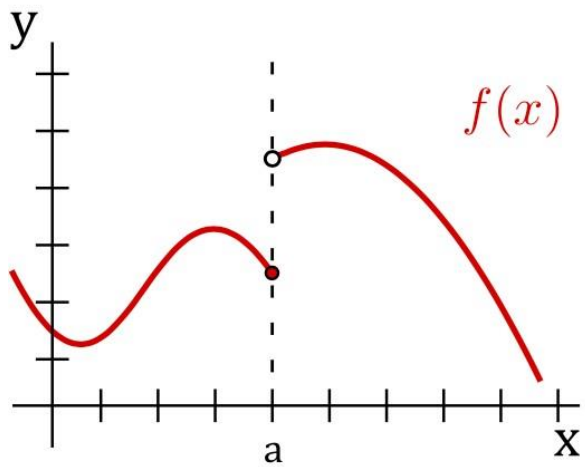
continuous at x = a

$\left(\lim_{x \to a} f(x) = f(a) \right)$



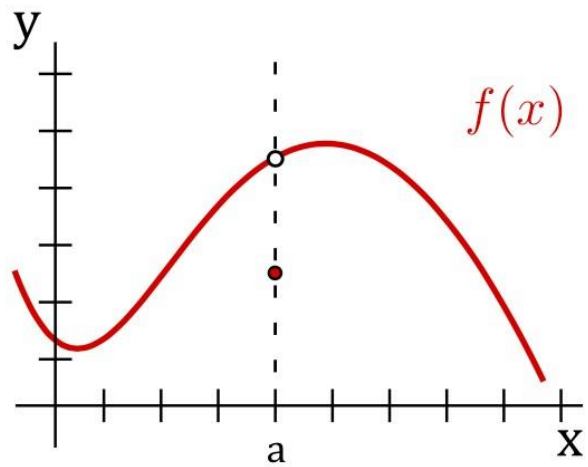
f(a) not defined

(i) fails to hold



$\lim_{x \to a} f(x)$ does not exist

(ii) fails to hold



$\lim_{x \to a} f(x) \neq f(a)$

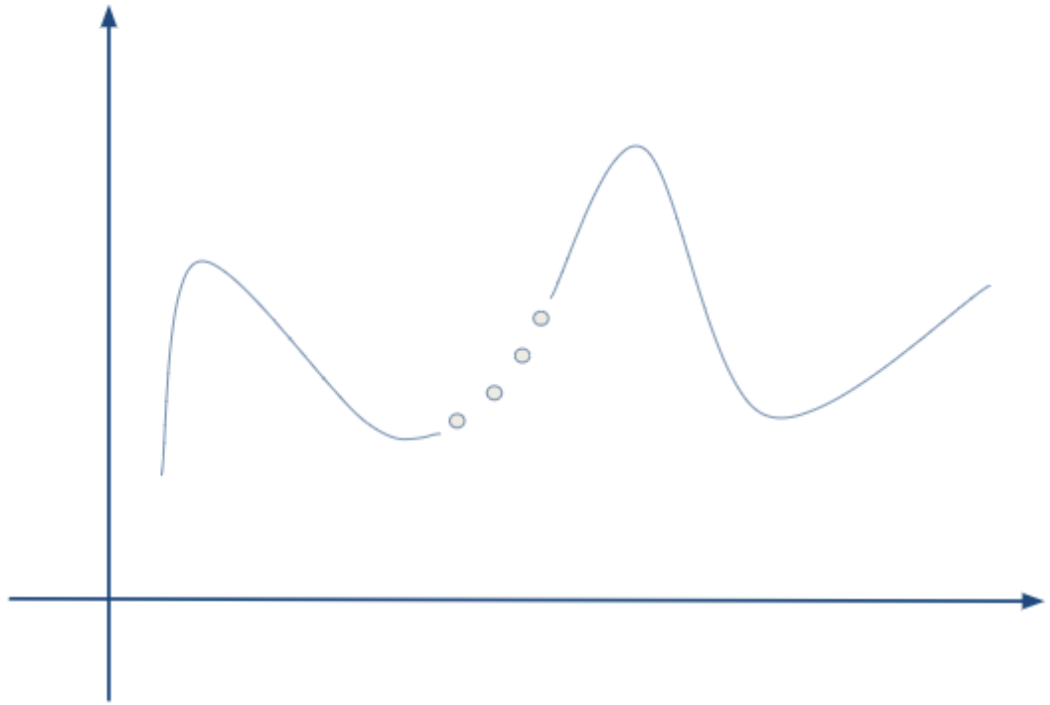
(iii) fails to hold

Непрерывность функции на множестве

- Что насчёт непрерывности функции на множестве?

Непрерывность функции на множестве

- Что насчёт непрерывности функции на множестве?
- Говорят, что функция непрерывна на множестве, если она непрерывна в каждой точке этого множества (в качестве множества может быть простой отрезок на оси).



Дифференцируемость

- А что такое дифференцируемость?

Дифференцируемость

- А что такое дифференцируемость?
- Если функция в окрестности точки достаточно хорошо приближается линейной зависимостью от параметров, то говорят, что она дифференцируема в этой точке.
- Говорят, что функция дифференцируема на множестве, если она дифференцируема в каждой внутренней точке этого множества.

Дифференцируемость и непрерывность

- Непрерывные функции обладают огромным количеством приятных свойств, на обсуждение которых у нас, увы, не хватит времени в рамках курса.
- Для нас важно, что непрерывность — необходимое условие дифференцируемости (ради которой мы все тут и затеяли): если функция дифференцируема на множестве, то она обязательно непрерывна на нём.

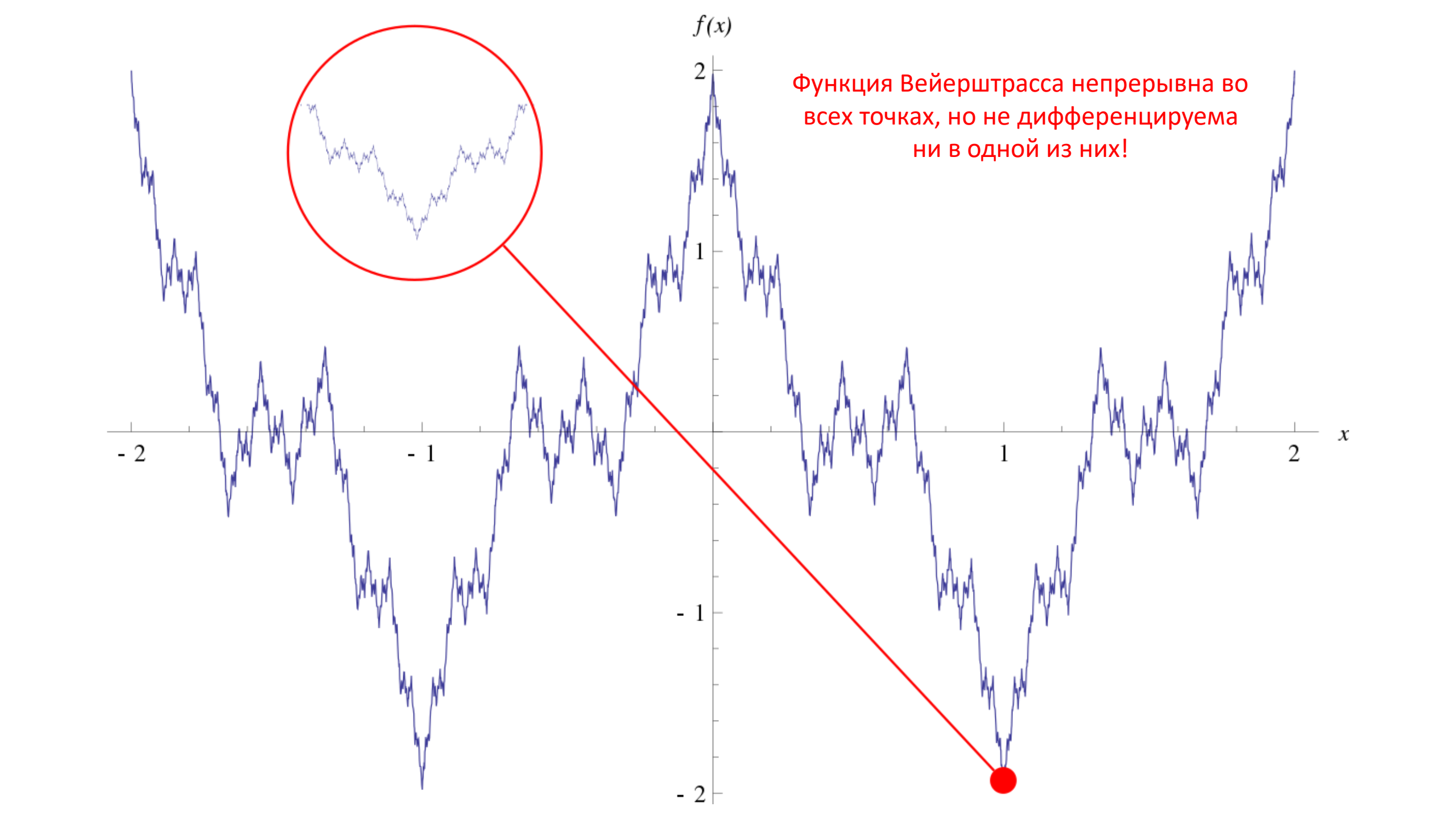
Дифференцируемость и непрерывность

- Непрерывные функции обладают огромным количеством приятных свойств, на обсуждение которых у нас, увы, не хватит времени в рамках курса.
- Для нас важно, что непрерывность — необходимое условие дифференцируемости (ради которой мы все тут и затеяли): если функция дифференцируема на множестве, то она обязательно непрерывна на нём.
- Обратное, вообще говоря, неверно.

Дифференцируемость и непрерывность

- Непрерывные функции обладают огромным количеством приятных свойств, на обсуждение которых у нас, увы, не хватит времени в рамках курса.
- Для нас важно, что непрерывность — необходимое условие дифференцируемости (ради которой мы все тут и затеяли): если функция дифференцируема на множестве, то она обязательно непрерывна на нём.
- Обратное, вообще говоря, неверно.

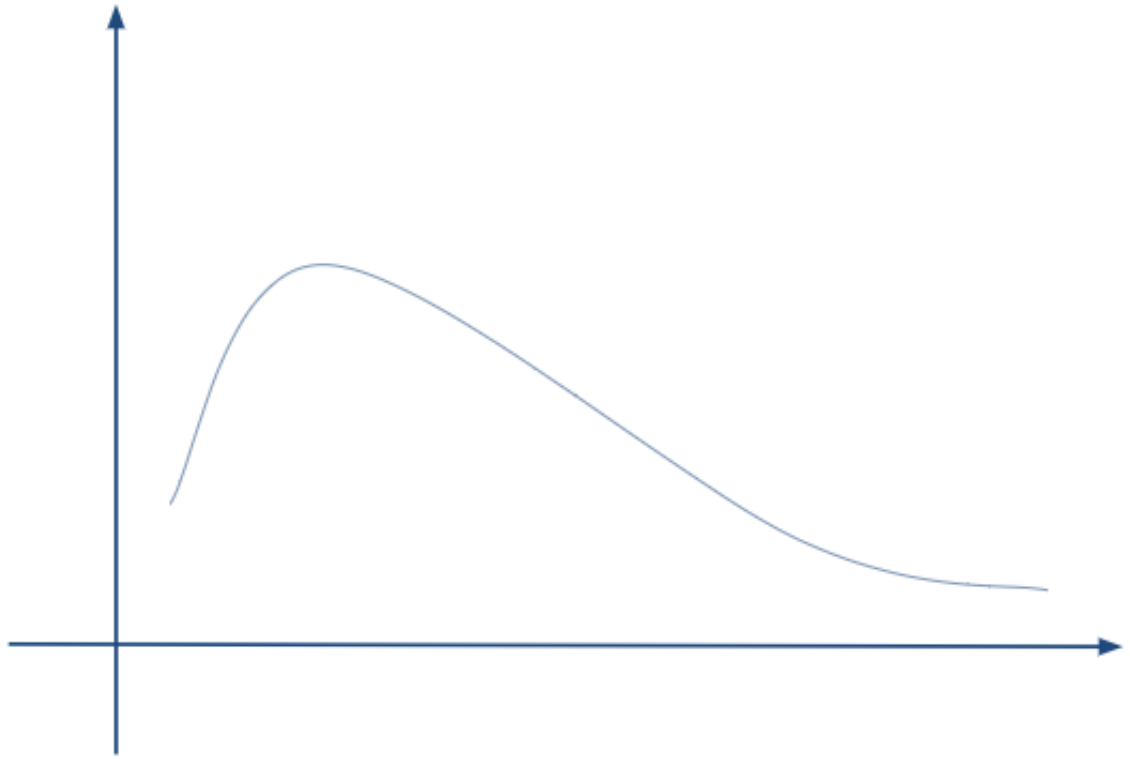
Может быть, вы даже можете привести контрпример?



Функция Вейерштрасса непрерывна во всех точках, но не дифференцируема ни в одной из них!

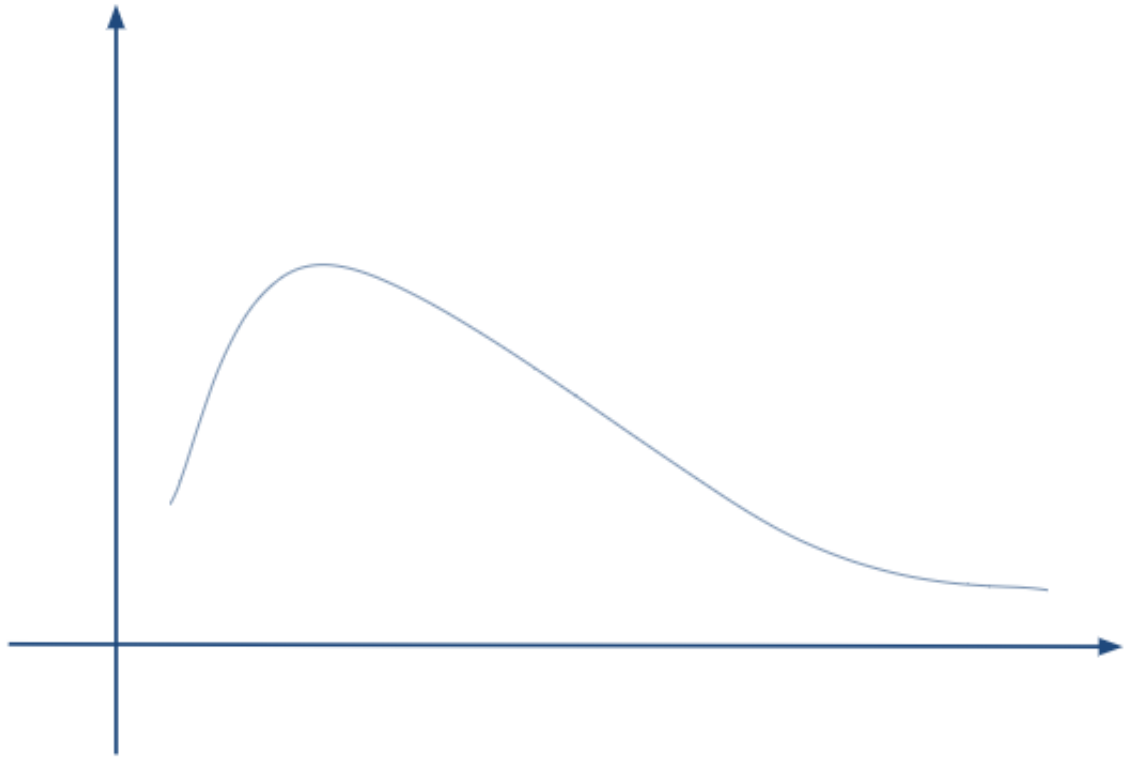
Первая производная

- Но вернёмся теперь к производной!



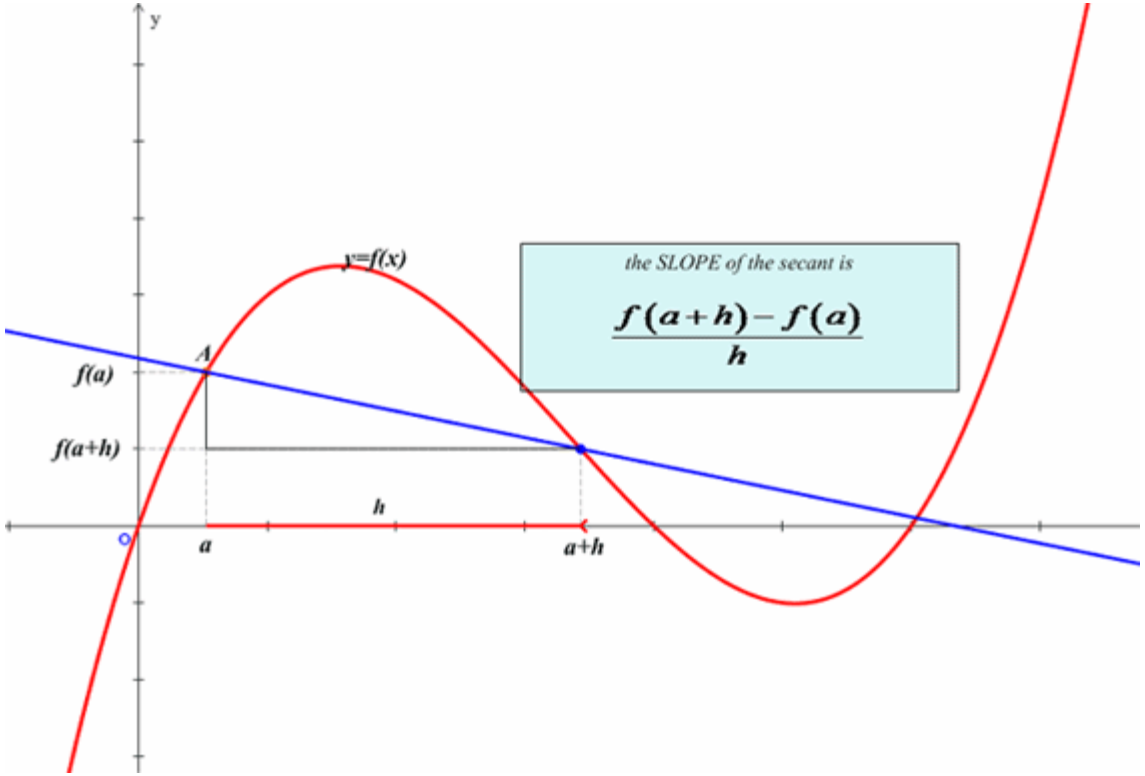
Первая производная

- Но вернёмся теперь к производной!
- Удобно представлять себе производную как мгновенную скорость материальной точки, траектория которой задана функцией $f(x)$



Первая производная

- Производная в точке однозначным образом задаёт касательную к функции в окрестности этой точки и численно равна тангенсу угла ее наклона.



Первая производная

- Для вычисления производных существует набор формальных правил. Более того, есть пакеты автоматического дифференцирования в духе sympy, torch.autograd и jax, которые сделают всё за вас.

Первая производная

- Для вычисления производных существует набор формальных правил. Более того, есть пакеты автоматического дифференцирования в духе sympy, torch.autograd и jax, которые сделают всё за вас.
- Однако крайне важно понимать, что перед тем, как подставлять значения аргументов в формулу производной, необходимо убедиться, что функция является дифференцируемой в рассматриваемой точке!

Первая производная

- Из связи производной с уравнением касательной в точке очевидно следует, что:
- Если функция дифференцируема на интервале, то можно найти промежутки её возрастания, убывания и постоянства.

Первая производная

- Из связи производной с уравнением касательной в точке очевидно следует, что:
- Если функция дифференцируема на интервале, то можно найти промежутки её возрастания, убывания и постоянства.
  - Если тангенс угла наклона касательной положительный, то функция возрастает.
  - Если он отрицательный, то убывает.
  - Если равен нулю, то функция постоянна в окрестности точки.

Первая производная

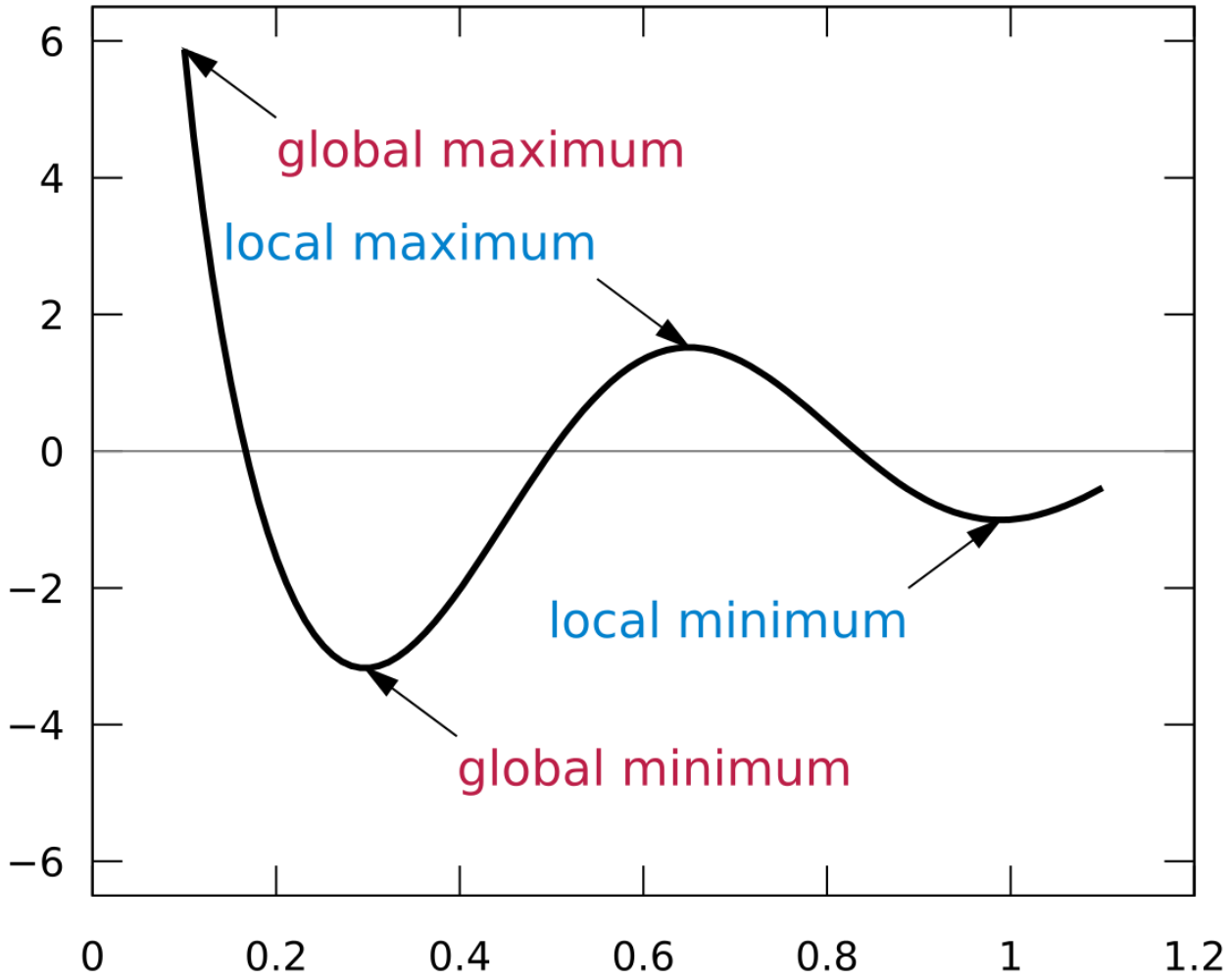
- Этот прекрасный факт позволяет определить необходимое условие наличия локального экстремума в терминах первой производной!

Первая производная

- Этот прекрасный факт позволяет определить необходимое условие наличия локального экстремума в терминах первой производной!

Пусть для функции $f: X \to \mathbb{R}$ точка x является точкой локального экстремума, тогда если функция непрерывно-дифференцируема в окрестности этой точки, то её производная в этой точке равна нулю: $df(x) = 0$

Точки экстремума в одномерном случае



Точки экстремума в одномерном случае

- Соответственно, первый шаг аналитического поиска локальных оптимумов дифференцируемой функции заключается в нахождении корней производной функции — так называемых стационарных точек.

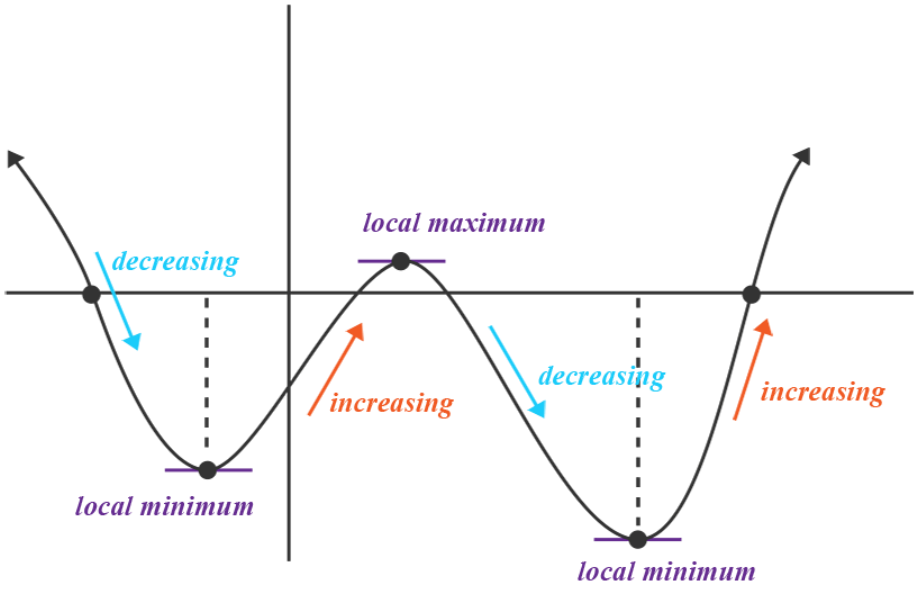
Точки экстремума в одномерном случае

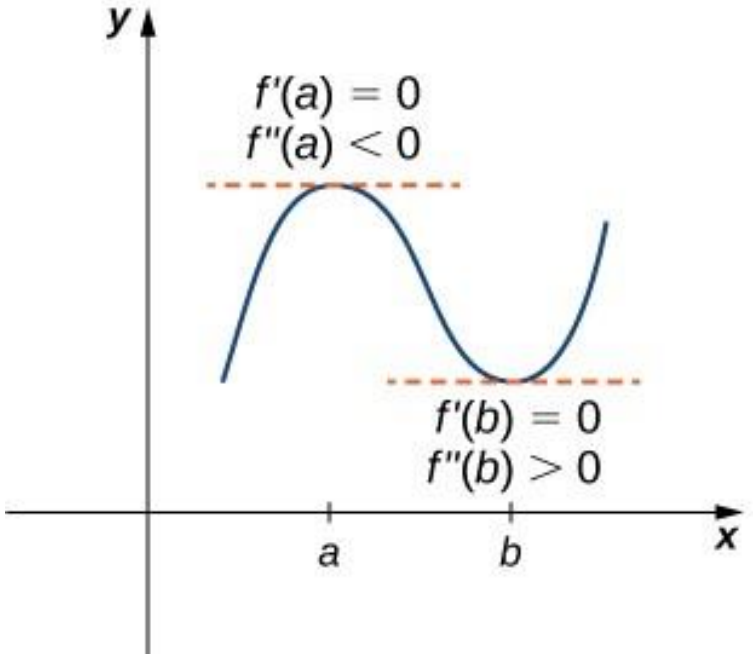
- Соответственно, первый шаг аналитического поиска локальных оптимумов дифференцируемой функции заключается в нахождении корней производной функции — так называемых стационарных точек.

- Замечание: Вторая и далее производные определяются рекурсивно: (n+1)-я производная функции это производная её n-й производной.

Точки экстремума в одномерном случае

- Если функция выпукла вверх в окрестности стационарной точки, то она имеет в этой точке локальный максимум.
- Если выпукла вниз, то — локальный минимум. Иначе оптимума нет!





Выпуклость

- В то же время выпуклость — важное понятие само по себе.
- Давайте обсудим его отдельно.

Выпуклость

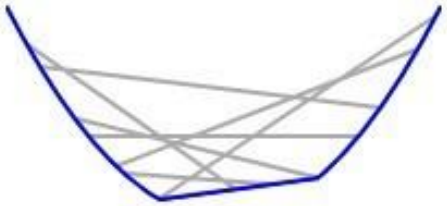
- В то же время выпуклость — важное понятие само по себе.
- Давайте обсудим его отдельно.
- Определение?

Выпуклость

- В то же время выпуклость — важное понятие само по себе.
- Давайте обсудим его отдельно.
- Определение?
- Рассмотрим функцию f и секущую, проходящую через точки $\big(x_1, f(x_1)\big), \big(x_2, f(x_2)\big), x_1 < x_2$.
- Если график f на промежутке $(x1, x2)$ лежит строго под секущей, то функция f на этом промежутке выпукла вниз, а если строго над секущей — выпукла вверх.

Выпуклость

A concave function:
no line segment joining
two points on the graph
lies above the graph
at any point



A convex function:
no line segment joining
two points on the graph
lies below the graph
at any point

A function that is neither
concave nor convex:
the line segment shown lies
above the graph at some
points and below it at others

- Рассмотрим функцию f и секущую, проходящую через точки $\big(x_1, f(x_1)\big), \big(x_2, f(x_2)\big), x_1 < x_2$.
- Если график f на промежутке $(x1, x2)$ лежит строго под секущей, то функция f на этом промежутке выпукла вниз, а если строго над секущей — выпукла вверх.

Дифференциальное исчисление

- Окей, мы успешно повторили с вами основные понятия математического анализа и даже уже кое-что вспомнили о производных и математической оптимизации, что не может не радовать!
- Но в чем же заключается главная проблема — или, можно сказать, главная недосказанность — в рамках той картины, которую мы с вами на текущий момент рассматриваем в дифференциальном исчислении?

Дифференциальное исчисление

- Окей, мы успешно повторили с вами основные понятия математического анализа и даже уже кое-что вспомнили о производных и математической оптимизации, что не может не радовать!
- Но в чем же заключается главная проблема — или, можно сказать, главная недосказанность — в рамках той картины, которую мы с вами на текущий момент рассматриваем в дифференциальном исчислении?
- Дадим тут короткий ответ —

Дифференциальное исчисление

- Окей, мы успешно повторили с вами основные понятия математического анализа и даже уже кое-что вспомнили о производных и математической оптимизации, что не может не радовать!
- Но в чем же заключается главная проблема — или, можно сказать, главная недосказанность — в рамках той картины, которую мы с вами на текущий момент рассматриваем в дифференциальном исчислении?
- Дадим тут короткий ответ — одномерность

Многомерная оптимизация

Многомерная оптимизация

- Несмотря на наши предыдущие рассуждения про функции одной переменной, в реальности вам почти никогда не придётся иметь дело с такими функциями.
- Задача, которую мы хотим решить, — найти оптимальные параметры алгоритма машинного обучения, — требует от нас определить производную функции многих переменных.

Многомерная оптимизация

- Сразу заметим, что наше интуитивное представление о производной как о мгновенной скорости даёт сбой, если функция зависит хотя бы от двух аргументов.

Многомерная оптимизация

- Сразу заметим, что наше интуитивное представление о производной как о мгновенной скорости даёт сбой, если функция зависит хотя бы от двух аргументов.
- Приведем пример. Представьте, что вы стоите на холме и хотите скатиться с него на санках.

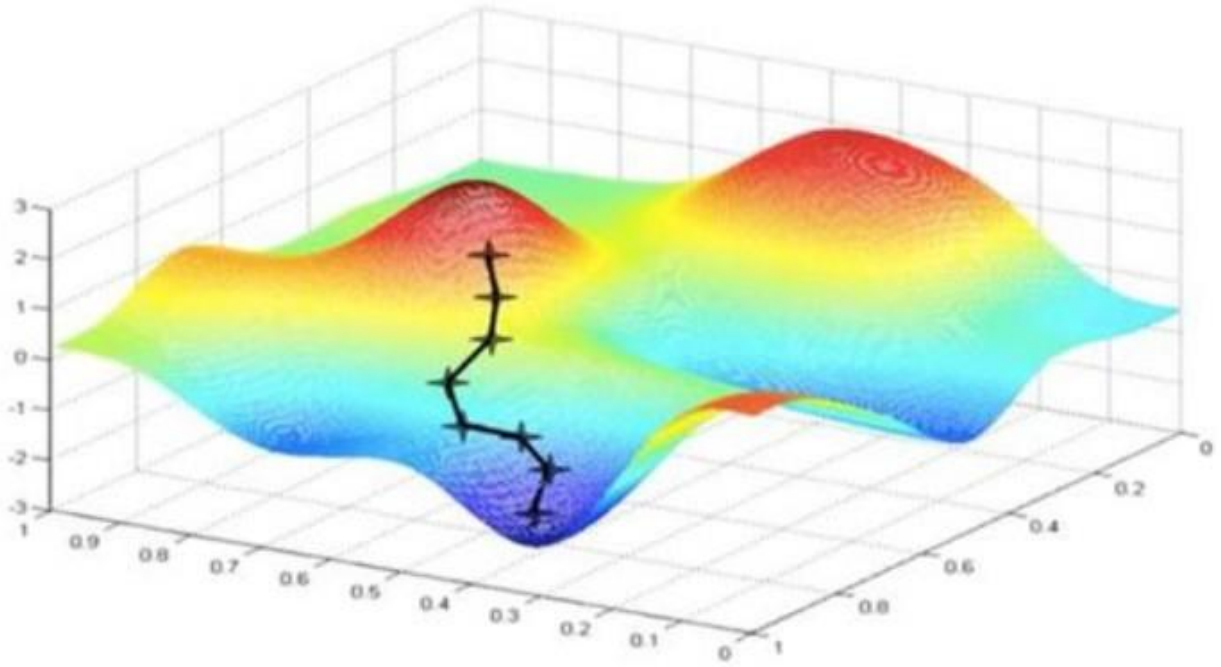
Многомерная оптимизация

- Сразу заметим, что наше интуитивное представление о производной как о мгновенной скорости даёт сбой, если функция зависит хотя бы от двух аргументов.
- Приведем пример. Представьте, что вы стоите на холме и хотите скатиться с него на санках.
- Оглянитесь вокруг: скорость вашего движения будет зависеть от выбранного направления.

Многомерная оптимизация

- Сразу заметим, что наше интуитивное представление о производной как о мгновенной скорости даёт сбой, если функция зависит хотя бы от двух аргументов.
- Приведем пример. Представьте, что вы стоите на холме и хотите скатиться с него на санках.
- Оглянитесь вокруг: скорость вашего движения будет зависеть от выбранного направления.
- Более того, вы можете скатиться в одну и ту же точку не двумя способами, как раньше (слева и справа), а по бесчисленному количеству траекторий!

Многомерная оптимизация



Теперь мгновенная скорость зависит не только от точки,
но и от направления движения.

Производная по направлению

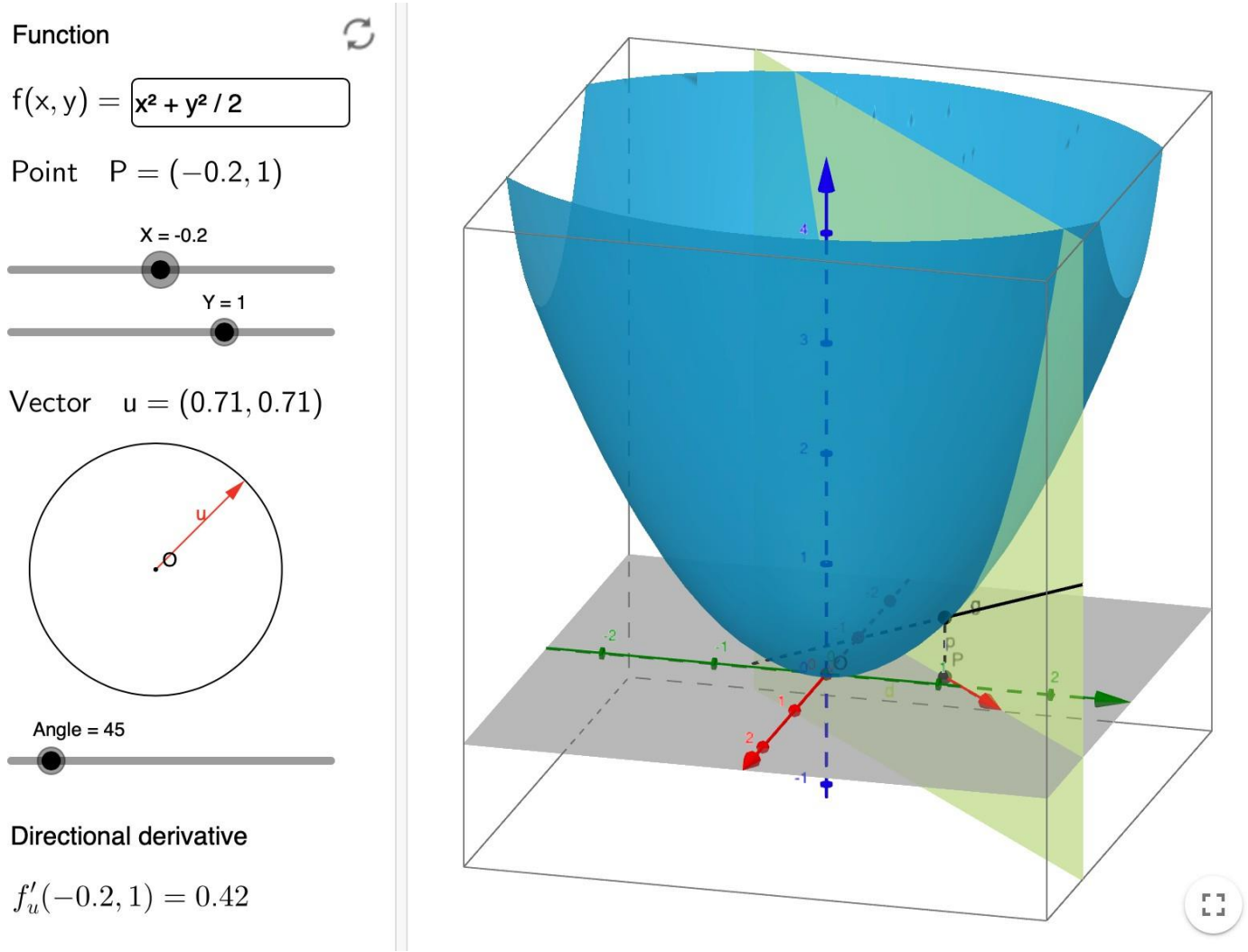
- Данное наблюдение естественным образом мотивирует понятие производной по направлению.
- Что же это такое?

Производная по направлению

- Данное наблюдение естественным образом мотивирует понятие производной по направлению.
- Что же это такое?
- Можно взять проекцию функции многих переменных на направление и получить привычную функцию одной переменной! Результат взятия производной от такой функции и будет являться производной по направлению.

Производная по направлению

“Потрогать” это определение можно на GeoGebra



Частная производная

- Что же такое частная производная?

Частная производная

- Что же такое частная производная?
- Производные по направлению каждой из главных осей называются частными производными.

Частная производная

- Что же такое частная производная?
- Производные по направлению каждой из главных осей называются частными производными.
- Частные производные существенно проще вычислять, т.к. значения остальных переменных можно считать фиксированными и, как следствие, “игнорировать” при расчете.

Частная производная

- Что же такое частная производная?
- Производные по направлению каждой из главных осей называются частными производными.
- Частные производные существенно проще вычислять, т.к. значения остальных переменных можно считать фиксированными и, как следствие, “игнорировать” при расчете.
- Давайте немного потренируемся и найдем частные производные, например, для функции:

$$f(x, y) = x^2 + 2xy - y$$

Градиент

- Итак, ну а теперь важнейшее понятие для математики машинного обучения — градиент! Что такое градиент?

Градиент

- Итак, ну а теперь важнейшее понятие для математики машинного обучения — градиент! Что такое градиент?
- Ковектор, составленный из частных производных, называется градиентом.

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right)$$

Градиент

- Итак, ну а теперь важнейшее понятие для математики машинного обучения — градиент! Что такое градиент?
- Ковектор, составленный из частных производных, называется градиентом.

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right)$$

- Как производные по направлению связаны с градиентом?

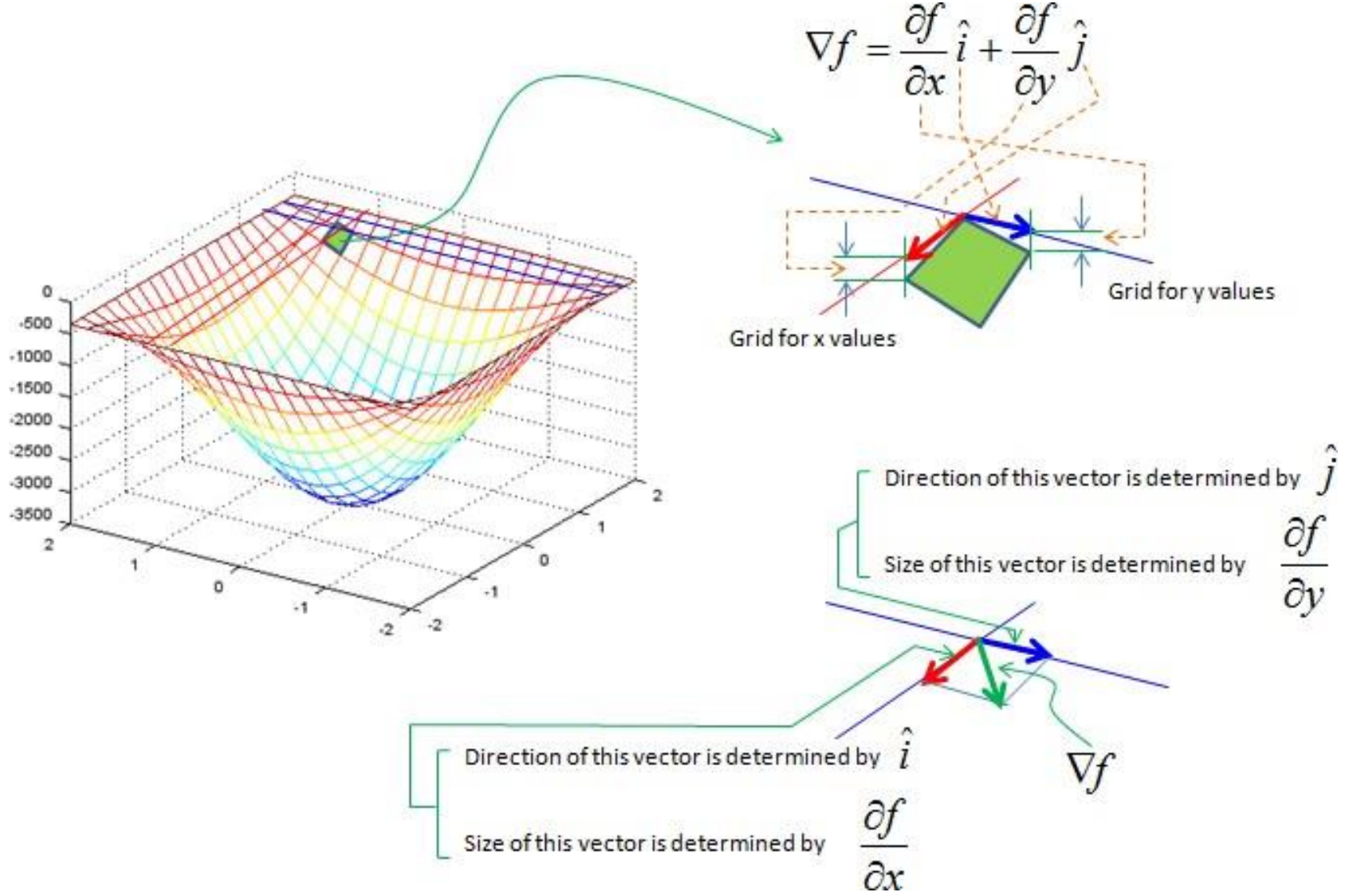
Градиент

- Итак, ну а теперь важнейшее понятие для математики машинного обучения — градиент! Что такое градиент?
- Ковектор, составленный из частных производных, называется градиентом.

$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right)$$

- Как производные по направлению связаны с градиентом?
- Производную по любому направлению можно вычислить через скалярное произведение градиента с направляющим вектором данного направления.

Градиент


$\nabla f = \frac{\partial f}{\partial x}\hat{i} + \frac{\partial f}{\partial y}\hat{j}$
Grid for y values
Grid for x values
Direction of this vector is determined by $\hat{j}$
Size of this vector is determined by $\frac{\partial f}{\partial y}$
Direction of this vector is determined by $\hat{i}$
Size of this vector is determined by $\frac{\partial f}{\partial x}$
∇f

Многомерная оптимизация

- Мы опустим обсуждения того, при каких условиях функция многих переменных будет дифференцируема, — нам интересно рассмотреть, что мы получаем, если функция является дифференцируемой.

Многомерная оптимизация

- Мы опустим обсуждения того, при каких условиях функция многих переменных будет дифференцируема, — нам интересно рассмотреть, что мы получаем, если функция является дифференцируемой.
- Если вы знаете, что функция дифференцируема в точке, то направление градиента в этой точке даёт вам очень важную информацию!
  - А для большинства стандартных функций в ML вы это знаете!

Многомерная оптимизация

- Мы опустим обсуждения того, при каких условиях функция многих переменных будет дифференцируема, — нам интересно рассмотреть, что мы получаем, если функция является дифференцируемой.
- Если вы знаете, что функция дифференцируема в точке, то направление градиента в этой точке даёт вам очень важную информацию!
  - А для большинства стандартных функций в ML вы это знаете!
- Что за информацию?

Многомерная оптимизация

Градиент скалярного поля (скалярной функции многих переменных) ориентирован вдоль направления наискорейшего возрастания функции в окрестности этой точки!

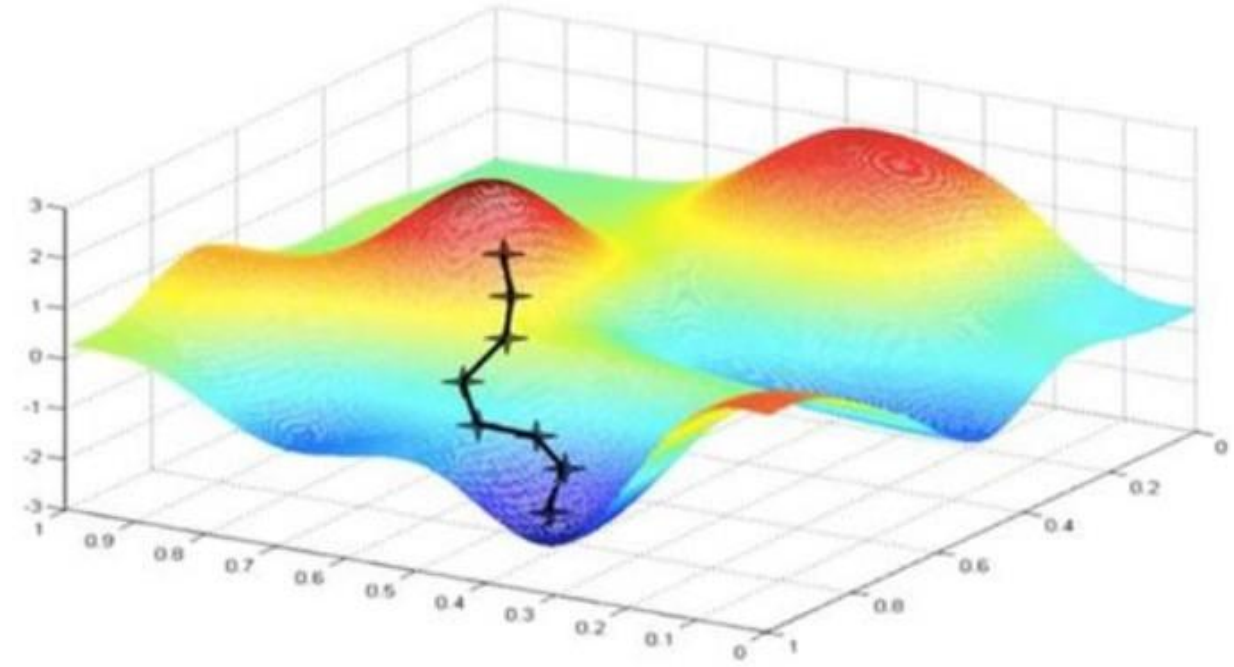
Многомерная оптимизация

Градиент скалярного поля (скалярной функции многих переменных) ориентирован вдоль направления наискорейшего возрастания функции в окрестности этой точки!

- Ура! Если функция дифференцируема, то мы в каждой точке знаем направление её наискорейшего возрастания (или же убывания — достаточно умножить градиент на -1 и получить антиградиент).

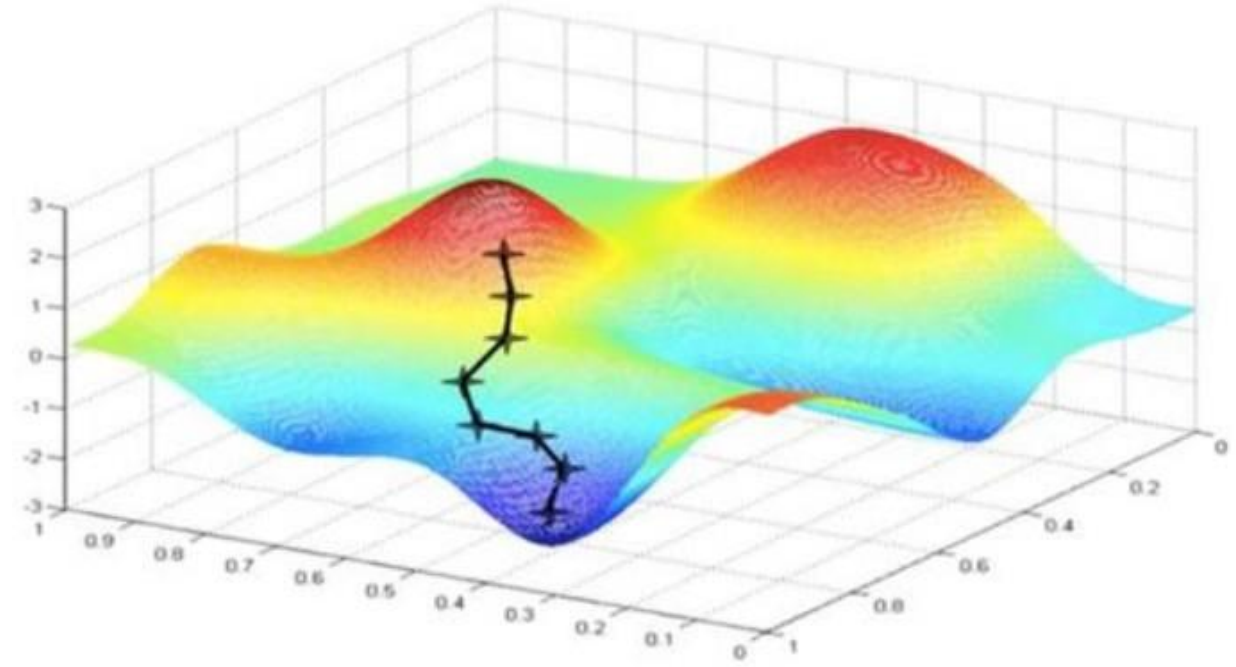
Многомерная оптимизация

- Вспомним наш пример с горкой и санками.



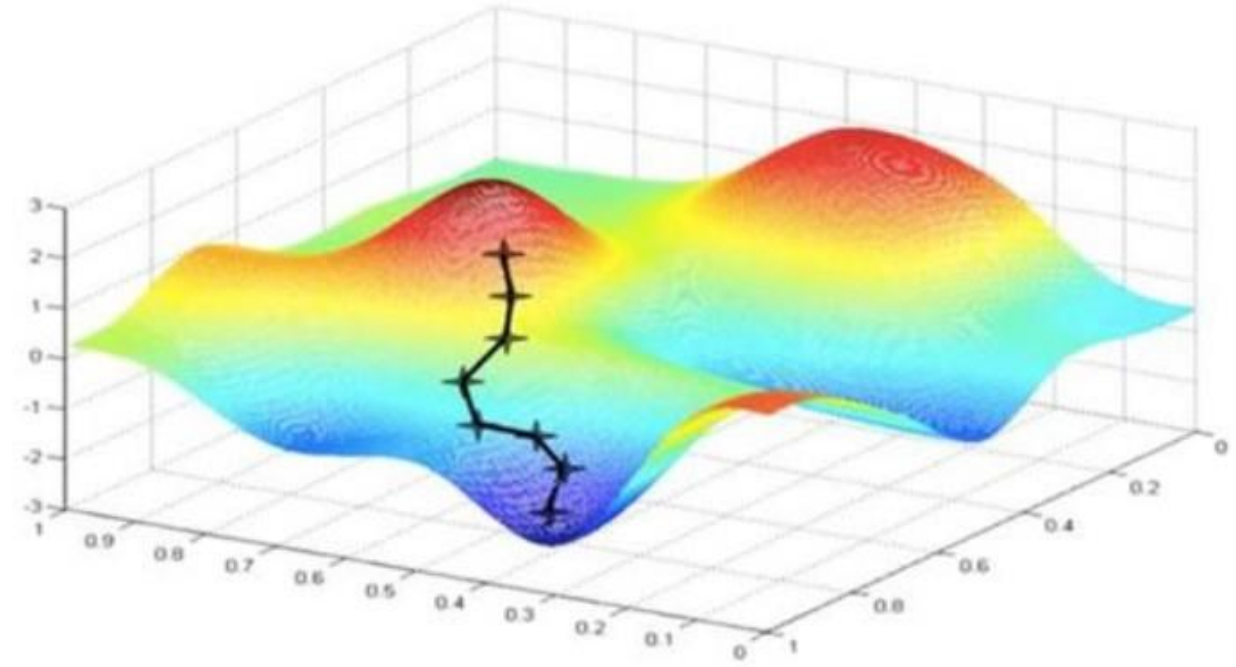
Многомерная оптимизация

- Вспомним наш пример с горкой и санками.
- Вы интуитивно понимаете, что быстрее всего спуститесь вниз, если будете ехать по самым крутым склонам.



Многомерная оптимизация

- Выражаясь языком математики, вам нужно всё время двигаться по антиградиенту.



Многомерная оптимизация

- Поздравляю! Мы только что изобрели метод градиентного спуска! :)
- Его модификация — стохастический градиентный спуск — один из главных столпов нейросетевого ML.

Многомерная оптимизация

- Поздравляю! Мы только что изобрели метод градиентного спуска! :)
- Его модификация — стохастический градиентный спуск — один из главных столпов нейросетевого ML.
- Разумеется, это далеко не единственный алгоритм оптимизации, который вам стоит знать.
- Тем не менее, самые ходовые алгоритмы — так называемые методы первого порядка — представляют собой модификации обычного градиентного спуска.

Градиентный спуск

Градиентный спуск

- Идейно градиентный спуск мы с вами, по сути, уже вывели.
- Осталось записать его формулами.

Градиентный спуск

- Идейно градиентный спуск мы с вами, по сути, уже вывели.
- Осталось записать его формулами.

$$w^{(n+1)} \leftarrow w^{(n)} - \alpha(n) \frac{\nabla \mathcal{L}(w^{(n)}; X, Y)}{\left\| \nabla \mathcal{L}(w^{(n)}; X, Y) \right\|}$$

Градиентный спуск

- Идейно градиентный спуск мы с вами, по сути, уже вывели.
- Осталось записать его формулами.

$$w^{(n+1)} \leftarrow w^{(n)} - \alpha(n) \frac{\nabla \mathcal{L}(w^{(n)}; X, Y)}{\left\| \nabla \mathcal{L}(w^{(n)}; X, Y) \right\|}$$

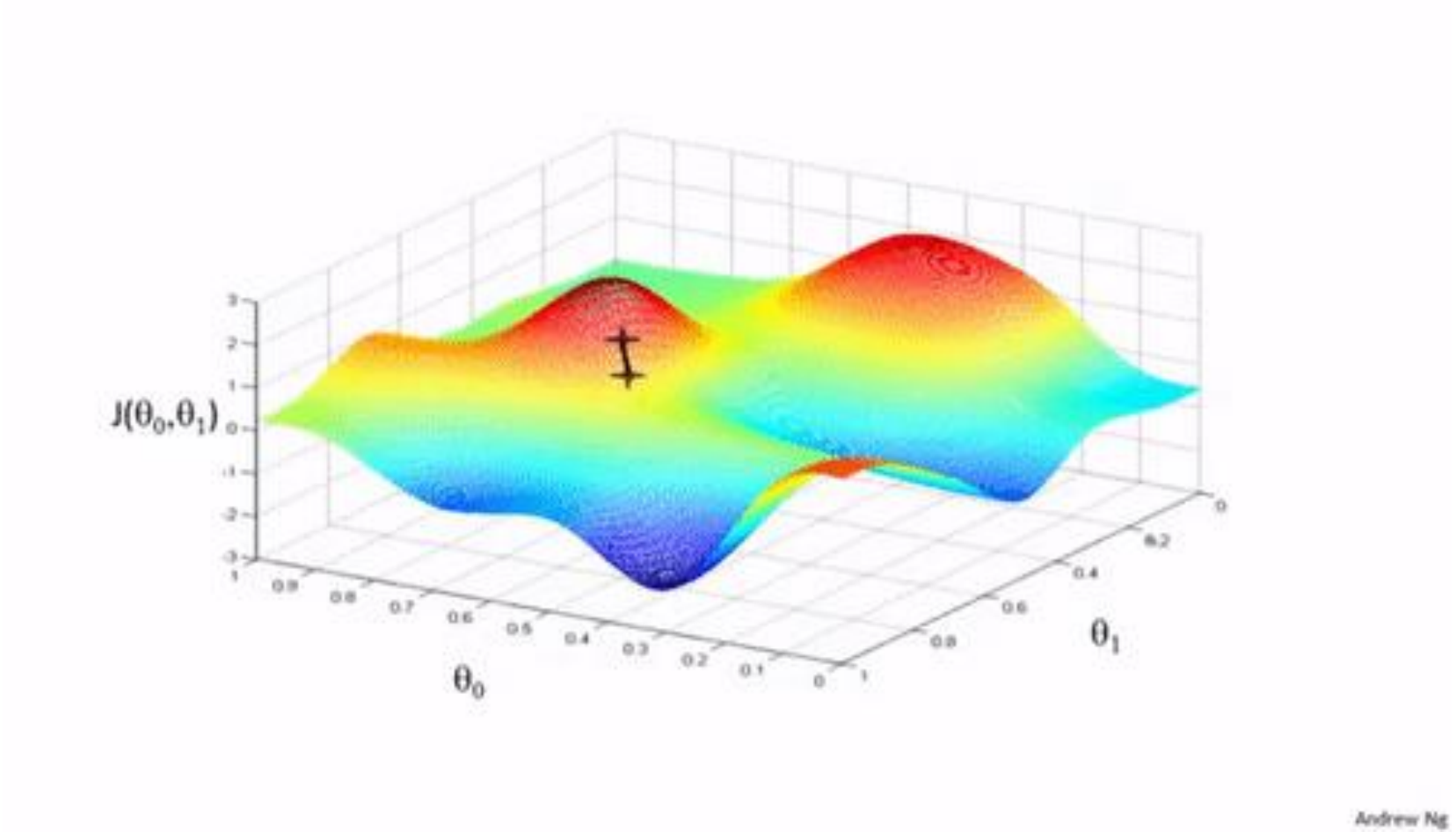
- Страшно? :)

Градиентный спуск

$$w^{(n+1)} \leftarrow w^{(n)} - \alpha(n) \frac{\nabla \mathcal{L}(w^{(n)}; X, Y)}{\left\| \nabla \mathcal{L}(w^{(n)}; X, Y) \right\|}$$

- где
  - $w^{(n)}$ — параметры алгоритма на шаге n;
  - $\frac{\nabla \mathcal{L}(w^{(n)}; X, Y)}{\left\| \nabla \mathcal{L}(w^{(n)}; X, Y) \right\|}$ — направление движения, заданное градиентом функции ошибок при текущем значении параметров модели, на обучающей выборке X с референсными ответами Y (если известны);
  - $\alpha(n)$ — learning rate, длина градиентного шага.

Градиентный спуск



Визуализация градиентного спуска для
функции двух вещественных переменных

Градиентный спуск

- У градиентного спуска есть довольно очевидные недостатки.
- Как вы думаете, какие?

Градиентный спуск

- У градиентного спуска есть довольно очевидные недостатки.
- Как вы думаете, какие?
  1. На каждом шаге приходится вычислять функцию ошибки сразу по всей обучающей выборке (она может быть очень большой, миллионы объектов);

Градиентный спуск

- У градиентного спуска есть довольно очевидные недостатки.
- Как вы думаете, какие?
  1. На каждом шаге приходится вычислять функцию ошибки сразу по всей обучающей выборке (она может быть очень большой, миллионы объектов);
  2. Нужно подбирать длину шага такой, чтобы и не топтаться на месте, и не перепрыгнуть локальный оптимум;

Градиентный спуск

- У градиентного спуска есть довольно очевидные недостатки.
- Как вы думаете, какие?
  1. На каждом шаге приходится вычислять функцию ошибки сразу по всей обучающей выборке (она может быть очень большой, миллионы объектов);
  2. Нужно подбирать длину шага такой, чтобы и не топтаться на месте, и не перепрыгнуть локальный оптимум;
  3. Нет гарантий, что мы найдём глобальный минимум.

Градиентный спуск

- У градиентного спуска есть довольно очевидные недостатки.
- Как вы думаете, какие?
  1. На каждом шаге приходится вычислять функцию ошибки сразу по всей обучающей выборке (она может быть очень большой, миллионы объектов);
  2. Нужно подбирать длину шага такой, чтобы и не топтаться на месте, и не перепрыгнуть локальный оптимум;
  3. Нет гарантий, что мы найдём глобальный минимум.

- Недостатки 1 и 2 устранимы, 3 — в общем случае, нет, но во многих важных для нас случаях — очень даже да!

Градиентный спуск

- Только что, поговорив про проблемы, мы выяснили, что очень хочется не считать полный градиент функции ошибок на каждом шаге. Но как это возможно?

Градиентный спуск

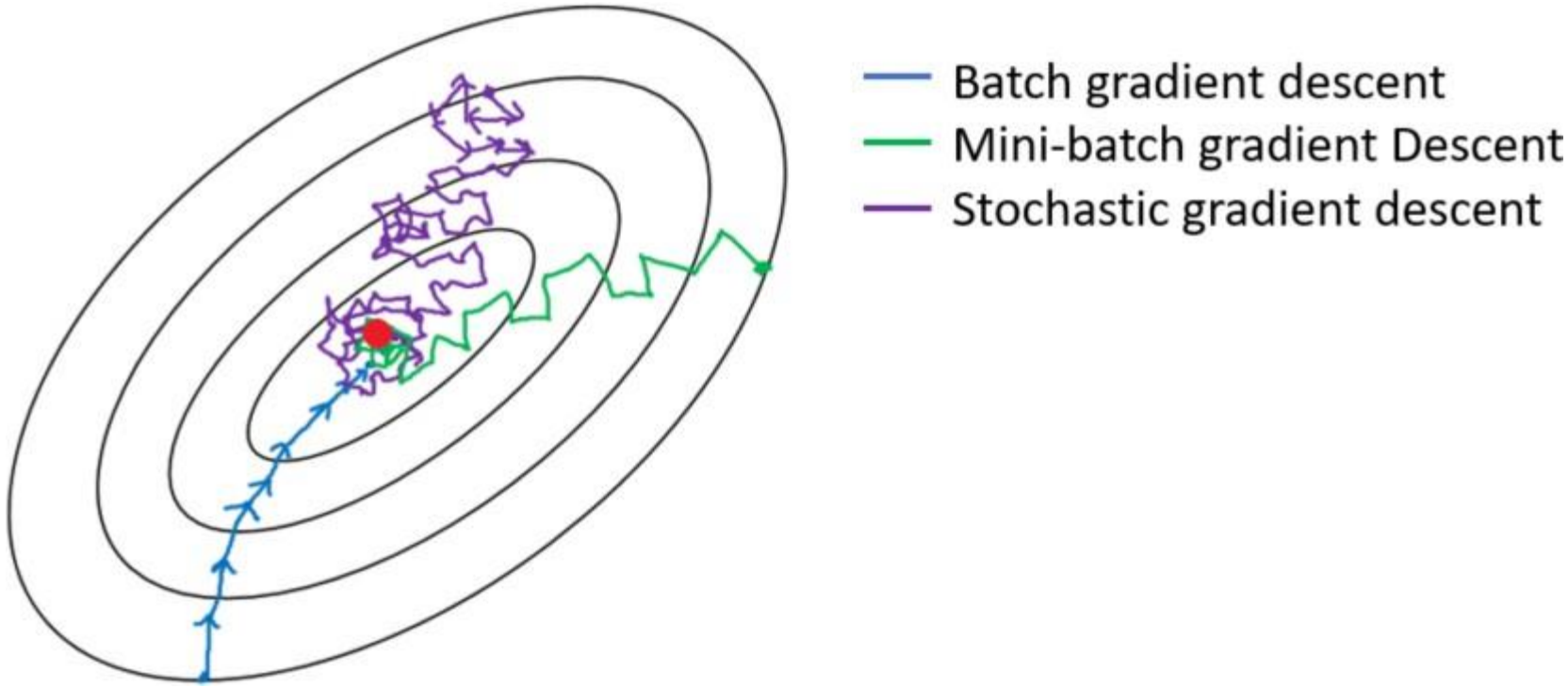
- Только что, поговорив про проблемы, мы выяснили, что очень хочется не считать полный градиент функции ошибок на каждом шаге. Но как это возможно?
- Можно считать градиент по случайному подмножеству объектов — так называемому минибатчу.
- Этот алгоритм представляет собой так называемый MiniBatch GradientDescent.
- Если грамотно подобрать размер минибатча, то эта модификация сойдётся гораздо быстрее оригинального градиентного спуска, что, на самом деле, очень приятно!

Градиентный спуск

- А еще бывает стохастический градиентный спуск…

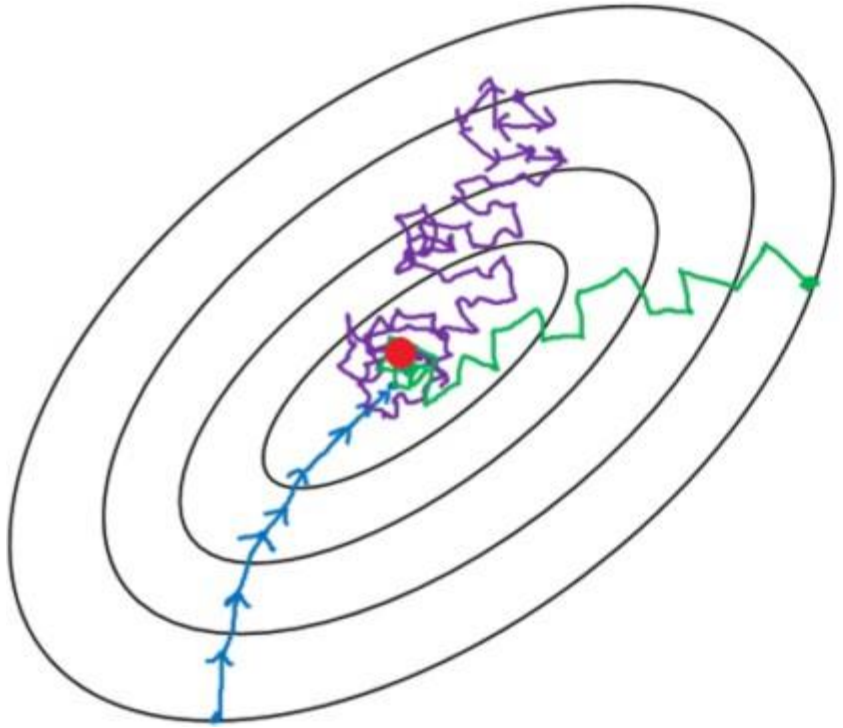
Градиентный спуск

- А еще бывает стохастический градиентный спуск…



Градиентный спуск

- А еще бывает стохастический градиентный спуск…



Batch gradient descent
Mini-batch gradient Descent
Stochastic gradient descent

Batch-, minibatch- и просто stochastic gradient descent отличаются друг от друга размером случайной подвыборки (условно 500 против 50 против 1).

Увеличение размера батча повышает стабильность обучения, но замедляет обновление весов.

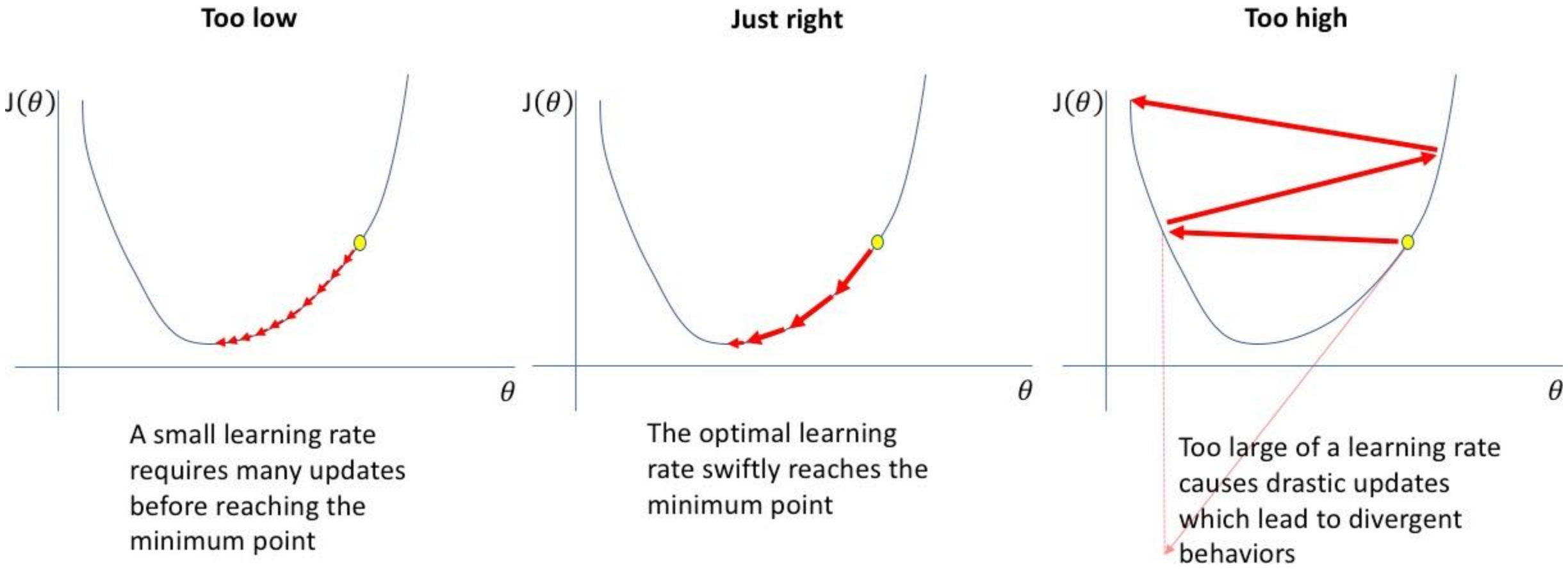
Градиентный спуск

- Также мы только что говорили и о другой проблеме: как подобрать длину шага?

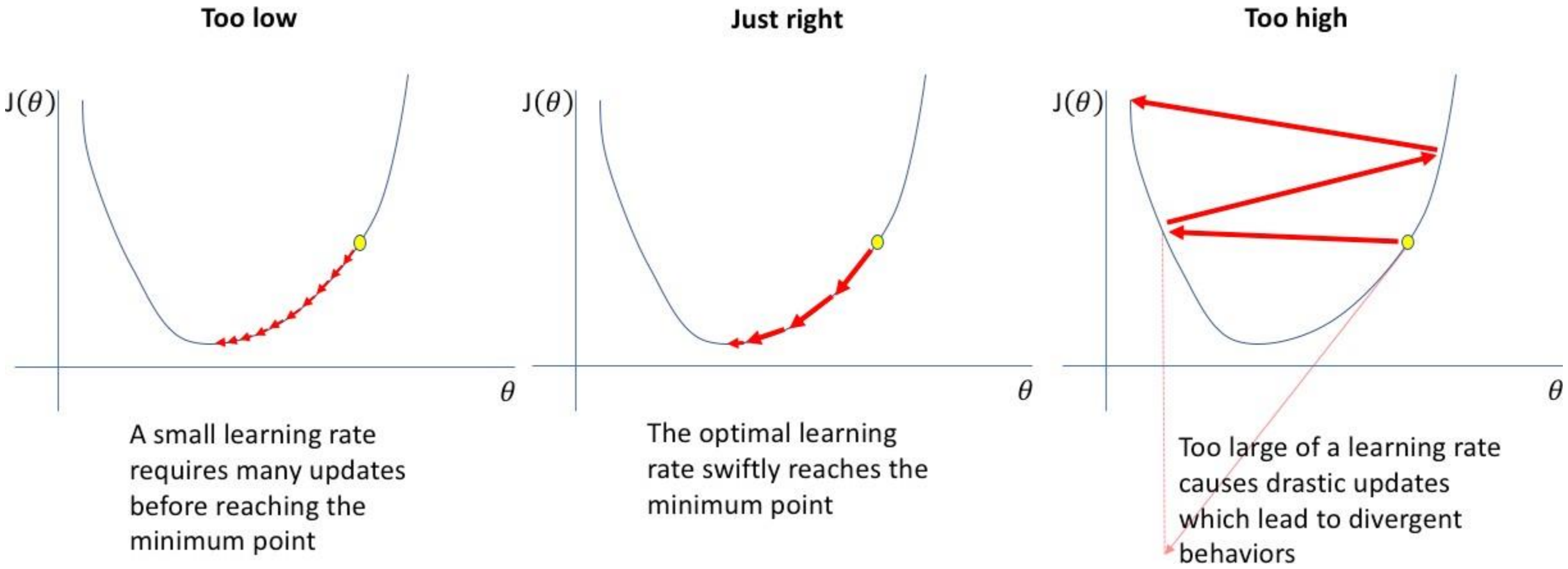
Градиентный спуск

- Также мы только что говорили и о другой проблеме: как подобрать длину шага?
- В начале обучения длина шага должна быть большой, чтобы градиентный спуск быстрее сошёлся к окрестности локального оптимума.
- Ближе к концу обучения длина шага должна быть маленькой, чтобы оптимизатор не выпрыгнул из потенциальной ямы.

Градиентный спуск



Градиентный спуск



Если длину шага подбирать неправильно, то градиентный спуск не будет сходиться даже в простейших случаях.

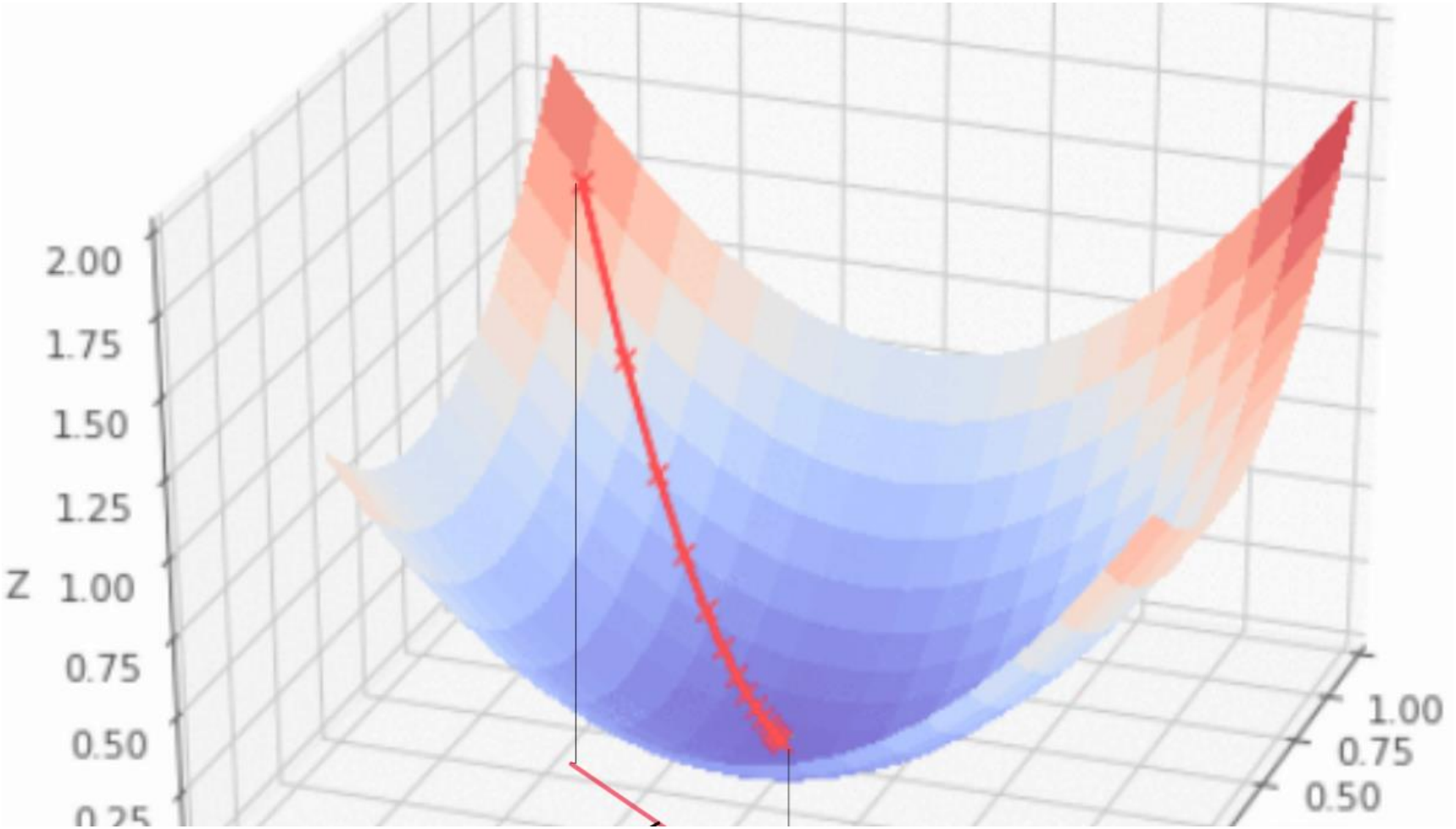
Градиентный спуск

- Наконец, третья рассматриваемая нами проблема — это про то, сходится градиентный спуск или не сходится.
- Как же понять, когда наш алгоритм всё-таки сможет найти глобальный оптимум?

Градиентный спуск

- Наконец, третья рассматриваемая нами проблема — это про то, сходится градиентный спуск или не сходится.
- Как же понять, когда наш алгоритм всё-таки сможет найти глобальный оптимум?
- Он сможет это сделать тогда, когда оптимизируемый функционал — ограниченная выпуклая функция, а оптимизация происходит на выпуклом множестве. У таких функций существует единственный оптимум!
- Многие важные для ML функции ошибок в духе MSE, MAE и так далее, выпуклые. А оптимизация обычно безусловная, т.е. происходит на всём $\mathbb{R}^n$, а это выпуклое множество.

Градиентный спуск



В задаче выпуклой оптимизации градиентный спуск с правильно подобранной длиной шага обязательно сойдётся к глобальному оптимуму!

Градиентный спуск

- А что же делать, если задача невыпуклая?

Градиентный спуск

- А что же делать, если задача невыпуклая?
- В таком случае можно запустить N градиентных спусков параллельно и выбрать лучший результат.
- Также можно периодически делать рестарты: сбрасывать learning rate на промежуточном шаге, чтобы искусственным образом увеличить длину шага.

Градиентный спуск

- Так или иначе, независимо от тех проблем, которые мы уже обсудили и частично решили, у нас имеется как минимум еще одна существенная проблема.
- Градиентный спуск всё равно медленный. Даже если он стохастический!

Градиентный спуск

- Так или иначе, независимо от тех проблем, которые мы уже обсудили и частично решили, у нас имеется как минимум еще одна существенная проблема.
- Градиентный спуск всё равно медленный. Даже если он стохастический!
- По смыслу — он не учитывает динамику системы: как меняются градиенты, какие направления важные, какие нет, и т.д.
- Другими словами алгоритм никак не принимает во внимание прошлые шаги. Если научить его этому, можно получить существенно более эффективные методы.

Градиентный спуск

- Так или иначе, независимо от тех проблем, которые мы уже обсудили и частично решили, у нас имеется как минимум еще одна существенная проблема.
- Градиентный спуск всё равно медленный. Даже если он стохастический!
- По смыслу — он не учитывает динамику системы: как меняются градиенты, какие направления важные, какие нет, и т.д.
- Другими словами алгоритм никак не принимает во внимание прошлые шаги. Если научить его этому, можно получить существенно более эффективные методы.

Это будет предметом нашего изучения на будущих занятиях :)