Оптимизация в машинном обучении

Сбертех, МФТИ

Содержание

- Решение задачи в МО через призму статистической теории решений
- Постановка задачи оптимизации для минимизации эмпирического риска
- Определение класса функций для оптимизации
- Методы первого порядка
- Методы второго порядка

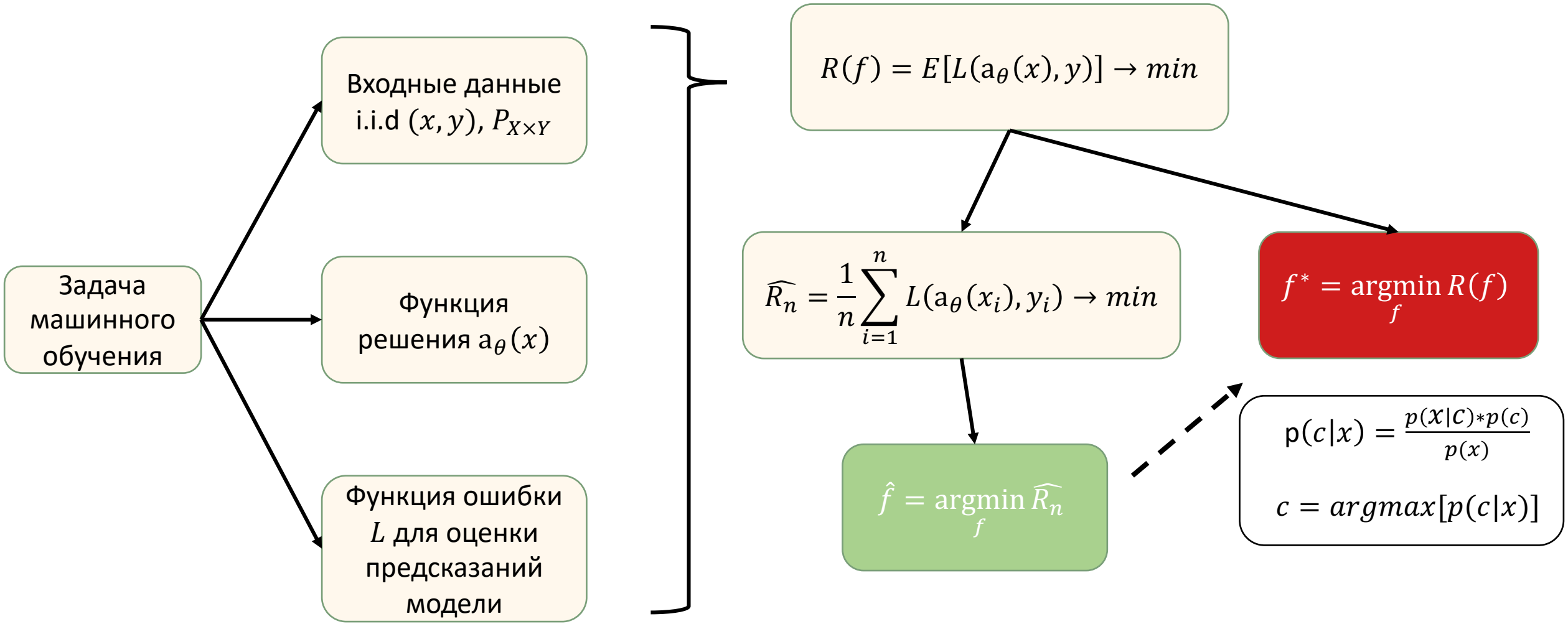
Формализация параметров

- $\boldsymbol{x} \in \boldsymbol{X}$ – вход модели
- $\boldsymbol{a_\theta}$: $\boldsymbol{X} -> \boldsymbol{A}$ – модель, которая данному входу ставит в соответствие некоторое значение из множества решений. Модель параметризована весами θ.
- $\boldsymbol{L}$: $\boldsymbol{A} \times \boldsymbol{Y}$ **- > R** – функция оценки, которая сравнивает решение модели с истинным значением для корректировки параметров (функция ошибки/loss function/objective)

Представим, что к ДС'у пришел стейкхолдер и попросил сделать модель.
Что должен сделать ДС?

1. Определить, что является входом модели
2. Определить, что является пространством решений для такой модели
3. Определить функцию оценки решений модели

Статистическая теория решений



Минимизация эмпирического риска

- **Риском** модели называется мат. ожидание ошибки модели $a_\theta(x)$: X -> A на объекте (x, y) выбранном случайно из $x \in X$.

$$\boldsymbol{R(a) = E[L(a_\theta(x), y))]}$$

- Знаем ли мы истинное распределение (x, y)? **Нет, но мы можем его оценить.**
- По закону больших чисел **–** среднее значение некоторой выборки $(x_1, x_2, \ldots x_n)$ сходится к математическому ожиданию

$$\lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n} x_i = E[x]$$

- Аналогично, мы можем оценить наш **риск** с помощью **эмпирического риска** на подвыборке

$$\hat{R}(x) = \frac{1}{n} \sum_{i=0}^{n} L(a_\theta(x_i), y_i)$$

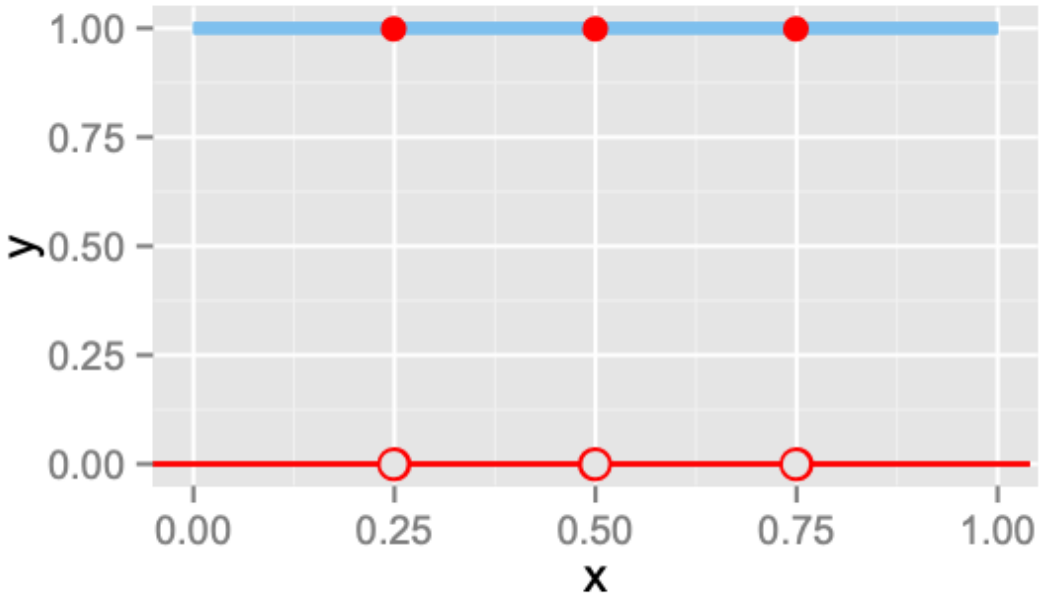
- Эмпирический риск модели машинного обучения по ЗБЧ будет аналогично стремится к мат. ожиданию риска

$$\lim_{n\to\infty} \hat{R}(x) = E[L(a_\theta(x), y)]$$

- Задача машинного обучения, найти такую модель $\hat{a}$ машинного обучения на множестве моделей A, так что ошибка модели минимальна на подвыборке

$$\hat{a} = \arg\min_{a\in A} \frac{1}{n} \sum_{i=0}^{n} L(a_\theta(x_i), y_i)$$

Разница между эмпирическим риском и риском



$$y = \begin{cases} if\ x \in [0.25, 0.50, 0.75] = 1 \\ else\ 0 \end{cases}$$

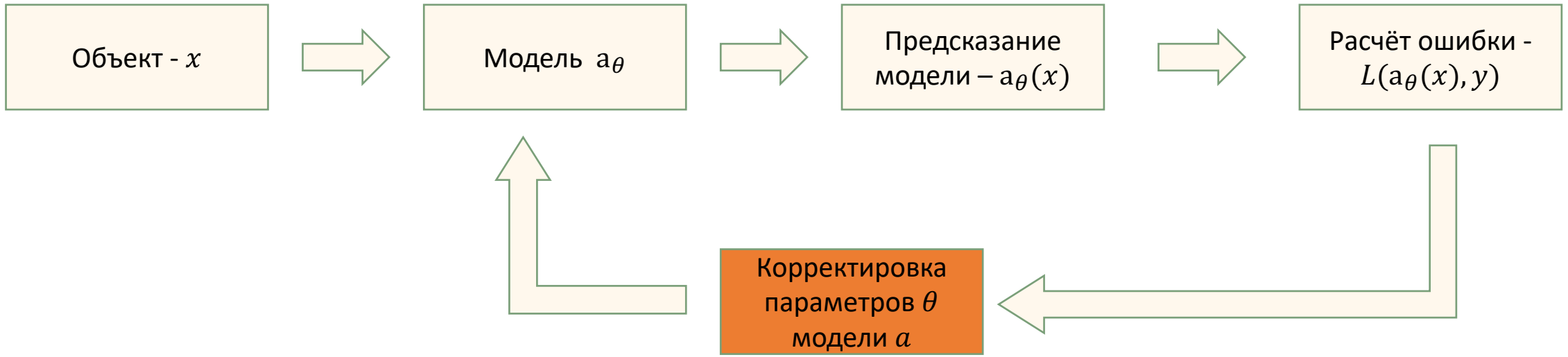
- Уменьшение эмпирического риска может вести к переобучению
- Можно выбрать более гладкий функционал, который приведет к большим значениям эмпирического риска, но уменьшит общий риск

Сведение задачи обучения к задаче оптимизации

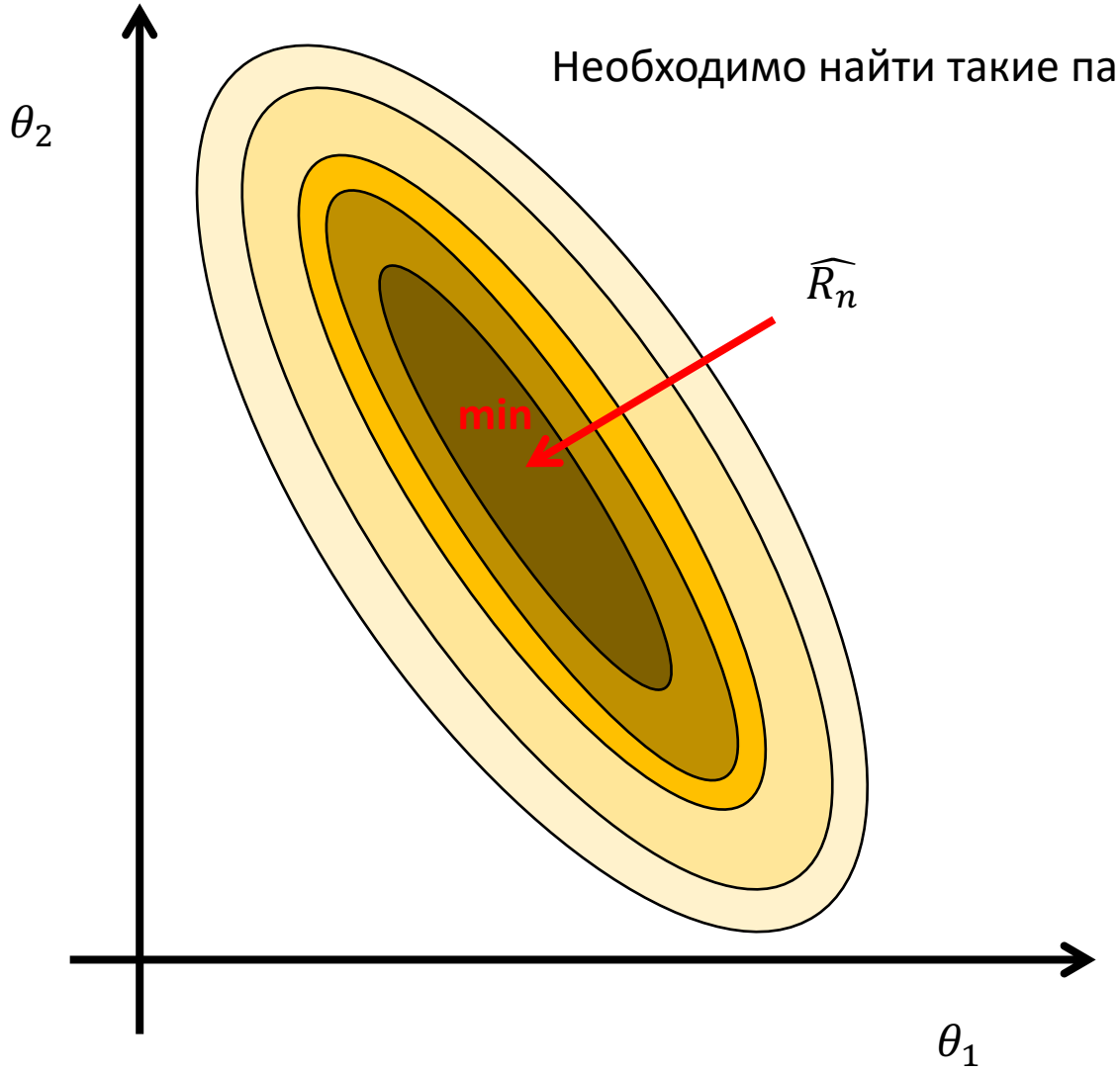
- Модель – параметрическое семейство функций $A = \{a_\theta(x) | \theta \in \Theta\}$, где a: $X \times \theta \to Y$- функция, Θ – множество допустимых параметров θ. Обозначается как a_θ
- Целевая переменная y. Обладает скрытой зависимостью с x. Задача – аппроксимировать y с помощью a_θ
- $L(a_\theta(x), y)$ – величина ошибки модели a_θ на объекте x относительно целевой переменной y
  - Функция ошибки на задаче классификации – $L(a_\theta(x), y) = [a_\theta(x) \neq y]$
  - Функция ошибки на задаче регрессии $-L(a_\theta(x), y) = (a_\theta(x) - y)^2$
- **Эмпирический риск** – усредненное значение функции $L(a_\theta(x), y)$на подвыборке X размера n

$$\widehat{R_n} = \frac{1}{n}\sum_{i=1}^{n} L(\mathrm{a}_\theta(x_i), y_i) \to min$$

- Задача оптимизации – $\arg\min_{a \in A} \frac{1}{n}\sum_{i=0}^{n} L(a_\theta(x_i), y_i)$



Оптимизация функции ошибки



Необходимо найти такие параметры θ при которых эмпирический риск минимален

Что надо знать?

- Класс оптимизируемых функций
- Свойства таких функций
- Методы оптимизации таких функций

Пример квадратичной функции потерь $L(a_\theta(x), y) = (a_\theta(x) - y)^2, \theta \in R^2$

Производная и выпуклость

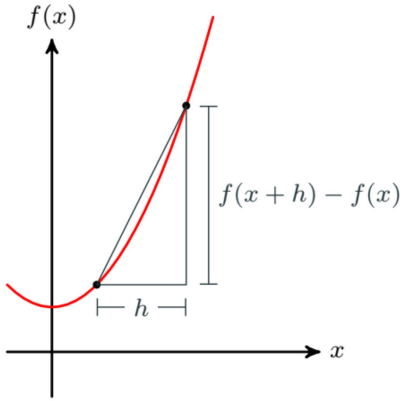
Производная

Пусть есть некоторая функция $f\colon X \to R$, где $X \in R^d$. Функция f дифференцируема если существует следующий предел:

$$\frac{d}{dx} f(x) = \lim_{h \to 0} \frac{\left(f(x_t + h) - f(x_t)\right)}{h}$$

Если функция f имеет производную в точке x_t, то в точке x_t можно приблизить линейной функцией f_l . В многомерном случае заменяем $f'(x)$ на $\nabla f(x)$**(градиент)**

$$f_l(x_t) = f(x_t) + \nabla f(x)(x - x_t)$$

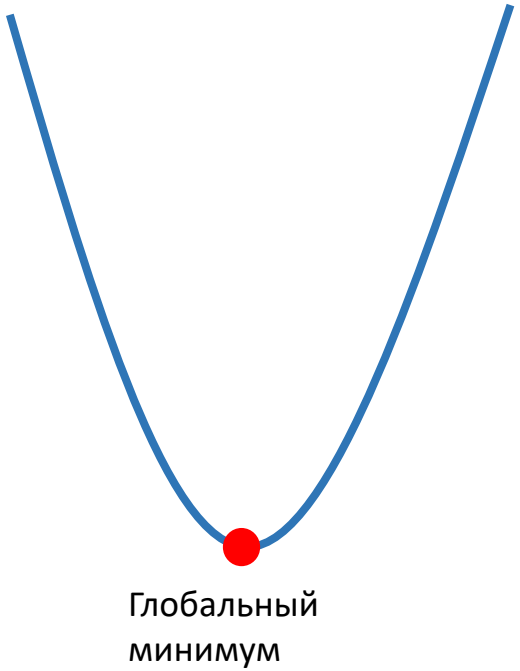


Производная – тангенс угла наклона касательной прямой

Стационарная точка – точка x_t в которой выполняется $\nabla f(x_t) = 0$.

Для поиска минимума мы будем брать производную(градиент) от функции ошибки L по параметрам θ модели a

Выпуклые функции



Выпуклая функция

Невыпуклая функция

- Глобальный минимум – единственный минимум функции, где производная равна 0.
- Локальный минимум – место, где производная равна 0. В выпуклых функциях – локальный минимум == глобальному минимуму.
- Локальный минимум/глобальный минимум –стационарные точки(седловые/критические).

Свойства выпуклых функций

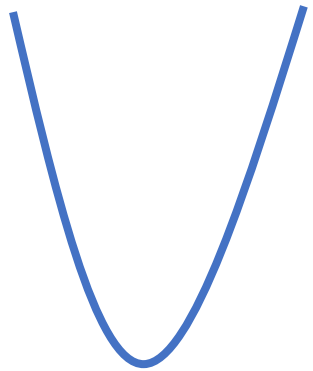
Функция определенная на множестве $X \in R^2$ называется **выпуклой** на X, если

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) * f(x_2)$$

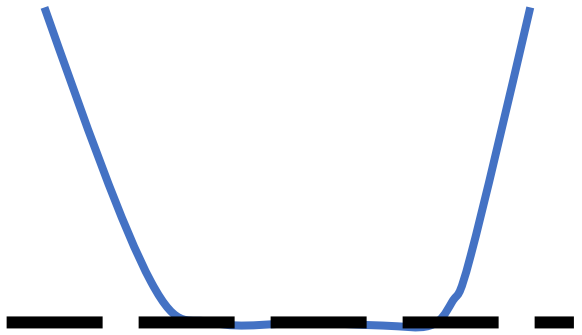
Функция определенная на множестве $X \in R^2$ называется **сильно выпуклой** на X, если

$$f(\lambda x_1 + (1-\lambda)x_2) < \lambda f(x_1) + (1-\lambda) * f(x_2)$$

У сильно выпуклой функции – 1 глобальный минимум.



Сильно выпуклая

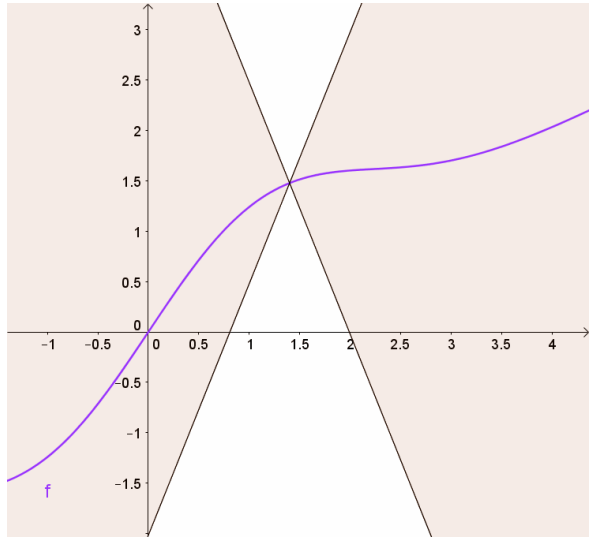


Выпуклая

Константа Липшица(L) – верхняя граница наклона прямой. Если у функции f нет константы Липшица, она может возрастать бесконечно быстро(у функции есть разрыв).

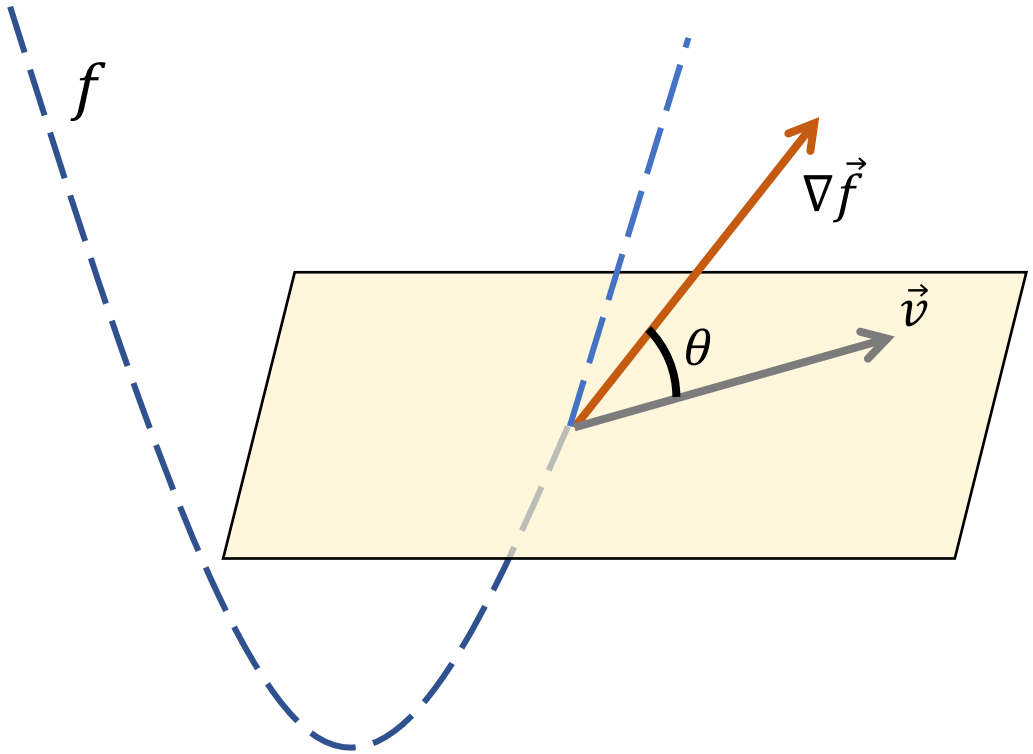
$$L = \sup \frac{f(\theta_1) - f(\theta_2)}{\theta_1 - \theta_2} = \sup \left| \frac{\mathrm{d}f}{\mathrm{d}\theta} \right|$$

Константа Липшица(L) – показатель того, что у функции нет разрывов и она дифференцируема по всей области определений. Константа Липшица – число.



Направление минимума

Градиент($grad\ f(x)$, $\nabla f(x)$) — вектор, показывающая направление наискорейшего возрастания некоторой величины, значение которой меняется от одной точки пространства к другой, по параметрам этой функции(в виде частных производных). Градиент f по параметрам θ обозначается как $\nabla_\theta f$.



[Производная по направлению]

$$\nabla f(x) = \left(\frac{df}{dx_1}\right) v_1 + \ \ldots + \left(\frac{df}{dx_n}\right) v_n$$

$$\cos\theta = \frac{\nabla\vec{f}\cdot\vec{v}}{\left|\nabla\vec{f}\right| * |\vec{v}|} \rightarrow \nabla\vec{f}\cdot\vec{v} = \left|\nabla\vec{f}\right||\vec{v}|\cos\theta$$

$\nabla\vec{f}\cdot\vec{v}$ - принимает наибольшее значение когда векторы сонаправлены ($\cos\theta$ = 1), $\vec{v}$ - единичный вектор

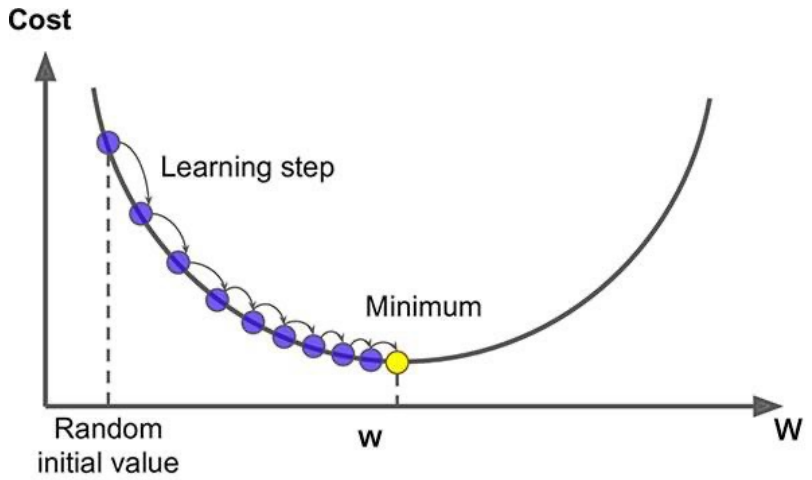
$\cos\theta = \frac{\nabla\vec{f}\cdot\vec{v}}{|\nabla\vec{f}|}$ - направление максимального роста функции

$-\frac{\nabla\vec{f}\cdot\vec{v}}{|\nabla\vec{f}|}$ - направление минимального роста функции

Градиентный спуск

Алгоритм градиентного спуска

$t \leftarrow 0$
$InitialValue\ \theta_t\ =\ 0$
$while\ \theta_{t+1}\ -\theta_t \geq\ \epsilon$:
$AverageGrad = \frac{1}{n}\sum_{i=0}^{n} \nabla_\theta L\ (a_\theta(x_i), y_i)$
$\theta_{t+1}\ \leftarrow\ \theta_t - \eta\ * AverageGrad$
$t \leftarrow t + 1$



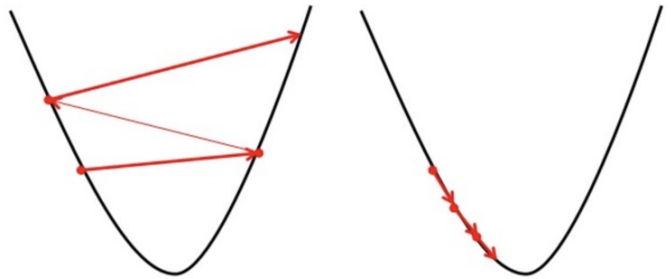
Алгоритм градиентного спуска

Шаг в градиенте – наискорейшее направление к минимуму функции:

$$\underset{(\theta\ -\theta_0) \leq \eta \nabla f(\theta)}{\operatorname{argmax}} f(\theta_t) +\ \nabla f(\theta) * (\theta\ -\theta_t) = \eta \nabla f(\theta)$$

η – шаг градиентного спуска – множитель градиента

1. Большой шаг позволяет быстрее найти оптимум, но может пропустить его
2. Маленький шаг вряд ли пропустит оптимум, но будет медленно сходиться
3. На практике шаг меняют на разных этапах обучения
4. Вычисление обратной константы Липшица 1/L для нашей функции ошибки L позволяет вычислить допустимый шаг η , так что при $\eta < \frac{1}{L}$ мы сможем найти локальный минимум(или стационарную точку) за фиксированное число шагов.



Разница между большим и маленьким шагом

Стохастический градиентный спуск(SGD)

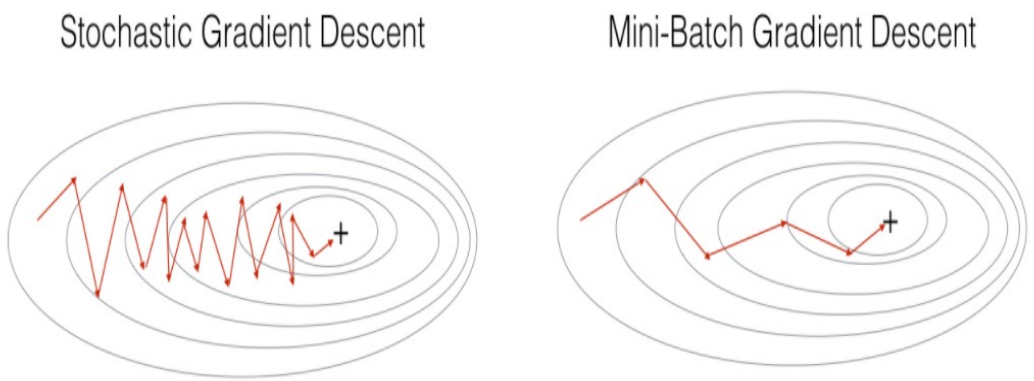
$$AverageGrad = \nabla\widehat{R_n}(\theta) = \frac{1}{n}\sum_{i=0}^{n} \nabla_\theta L\left(a_\theta(x_i), y_i\right)$$

1. Для того чтобы сделать 1 шаг в градиентном спуске нам нужно посчитать градиент в n точках – $O(n)$ для одного шага.
2. Можно взять часть данных исходных данных и посчитать градиент на них – риск на всех объектах равен риску на части объектах. Докажем, что мы можем аппроксимировать всю выборку подвыборкой:

$$E\left[\nabla\hat{R}_N(\theta)\right] = \sum_{i=0}^{N} P(i = X) * \nabla_\theta L\left(a_\theta(x_i), y_i\right) = \frac{1}{n}\sum_{i=0}^{N} \nabla_\theta L\left(a_\theta(x_i), y_i\right) = E[\nabla\hat{R}_N(\theta)]$$

$t \leftarrow 0$
$InitialValue\ \theta_t = 0$
$while\ \theta_{t+1} - \theta_t \geq \epsilon$:
  Выбор N семплов $- (x_i, y_i)_{i=1}^{N}$
  $AverageGrad = \frac{1}{N}\sum_{i=0}^{N} \nabla_\theta L\left(a_\theta(x_i), y_i\right)$
  $\theta_{t+1} \leftarrow \theta_t - \eta * AverageGrad$
  $t \leftarrow t + 1$

$N = 1$ – **стохастический градиентный спуск**
$1 < N < n$ – **mini-batch градиентный спуск**



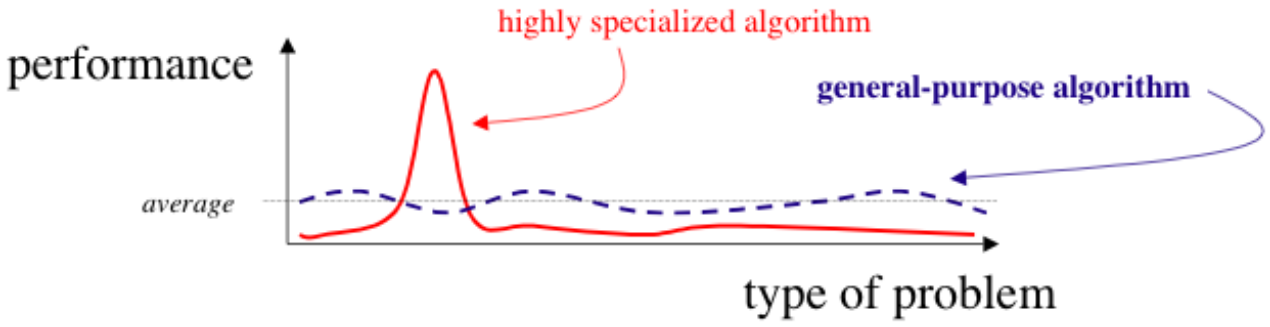
Все последующие методы наиболее часто используют SGD подход.

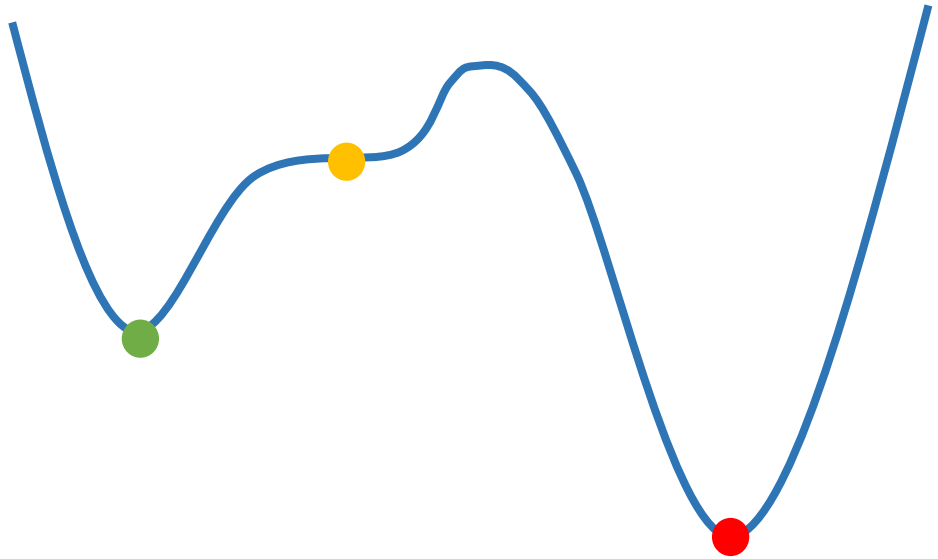
No free lunch

No free lunch theorem

- На общем пространстве задач оптимизации, не существует алгоритма, который работает лучше всех.

- Если на каком типе подзадач алгоритм работает лучше чем случайный поиск, значит на других типах задач он будет работать хуже чем случайный поиск.

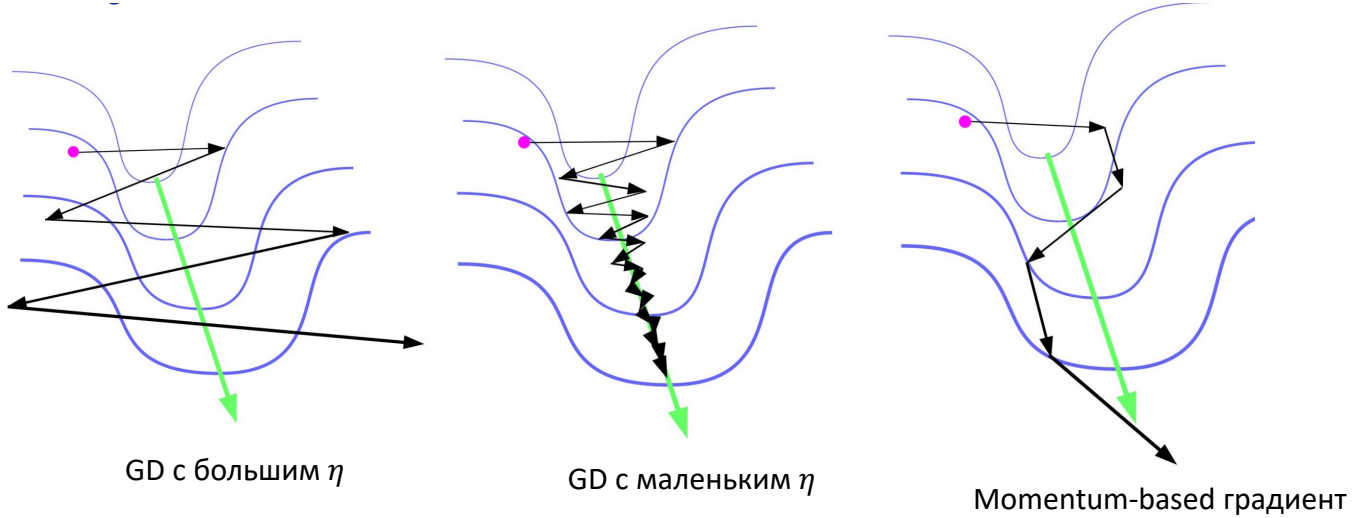




У нас нет универсального алгоритма оптимизации невыпуклых функций, но мы будем использовать алгоритм оптимизации выпуклых функций чтобы найти хоть какие-то стационарные точки.

Momentum-based подходы

Мотивация – градиентный спуск имеет тенденцию прыгать в разные стороны. Как его ускорить?



GD с большим η

GD с маленьким η

Momentum-based градиент

Ключевая идея - Давайте выбирать такое направление, которое среди всех направлений ведет нас ниже всего.

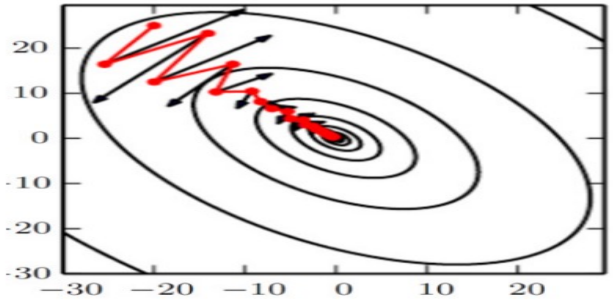
Алгоритм - Представим что наш вектор параметров это мяч, который спускается вниз по склону, искривление которого задано нашей функцией a_θ.

Мотивация momentum-based подхода

Momentum-based подходы

Проблема градиентного спуска – сильная осцилляция относительно правильного направления к минимуму.

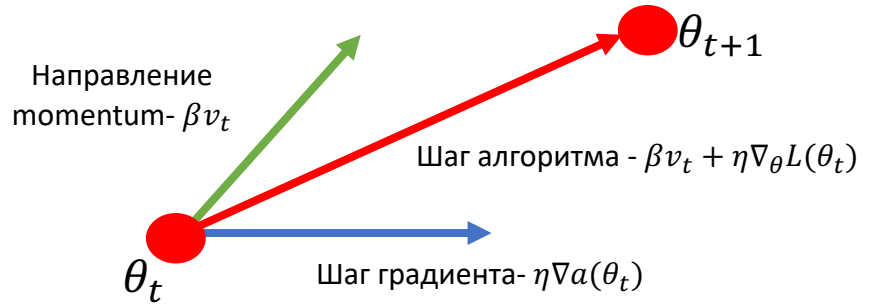
$$\theta_{t+1} \leftarrow \theta_t - \eta * \nabla_\theta L(\theta_t)$$



- **Momentum** – экспоненциальное скользящее по по $\frac{1}{1-\beta}$ последним итерациям(Поляк, 1964)
  - К градиенту добавляем взвешенное изменение веса за последние итерации(чем больше β -> тем больше последних итераций учитываем при изменении). По умолчанию, значение **0.9.**

$$v_{t+1} = \beta v_t + \eta \nabla_\theta L\,(\theta_t)$$

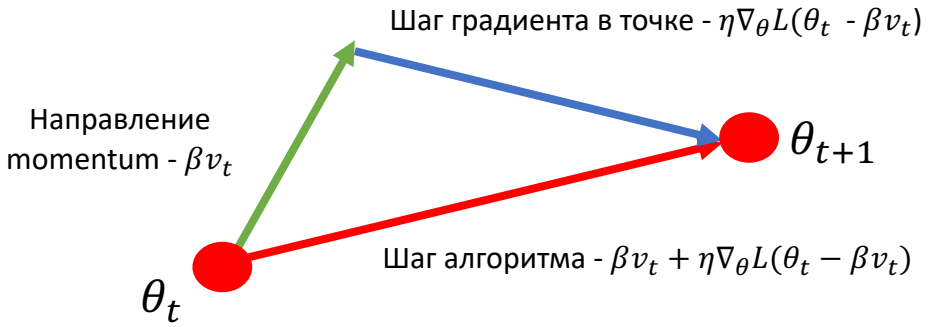
$$\theta_{t+1} = \theta_t - v_{t+1}$$



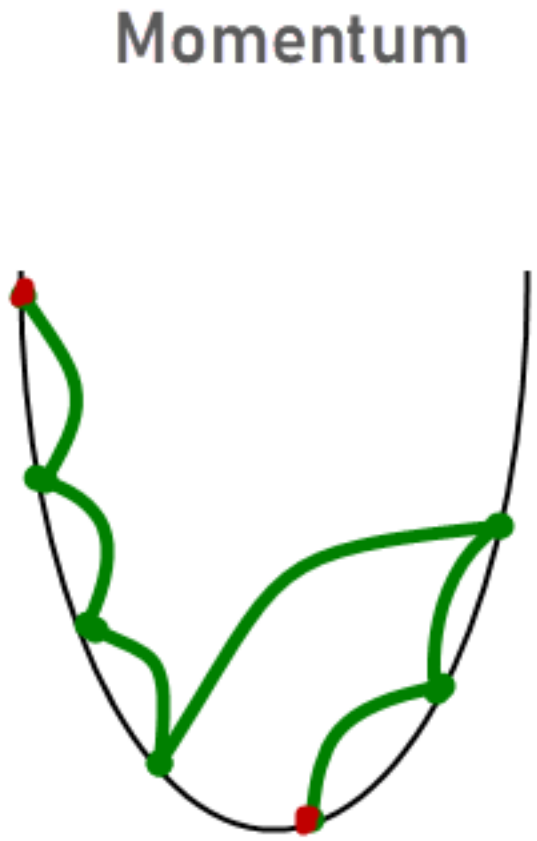
- **Nesterov's accelerated gradient(NAG)** – стохастический градиент с импульсом (Нестеров, 1983)
  - Вычисляем значение градиента в точке будущей точке momentum, вместо старого значения θ_t

$$v_{t+1} = \beta v_t + \eta \nabla_\theta L(\theta_t - \beta v_t)$$

$$\theta_{t+1} = \theta_t - v_{t+1}$$



Momentum-based подходы



Благодаря предварительной оценке в NAG, мы не накапливаем большой градиентный импульс и следуем ему, а делаем сначала разведку, а потом корректируем шаг.

Преимущества momentum-based подходов

- В случае если на некотором отрезке поверхность функции ошибки очень искривлена в рамках локального минимума (высокое значение L) – градиент примет большое значение, отклонившись от правильного пути сходимости. Это приведет к осцилляции сходимости.
  - С помощью добавления информации об изменениях градиента на предыдущих итерациях – мы можем исправить направление градиента .
  - Другими словами – если на каком то шаге у градиента высокое отклонение относительно предыдущих значений – история градиентов позволяет не сильно изменить направление поиска.
- По теории NAG имеет лучшую оценку скорости сходимости для выпуклых функций, а на практике работает быстрее.
  - Суть NAG – сделать более умный momentum, который двигается не просто по инерции, а с учетом градиента функции.
  - NAG ограничивает постоянно набирающий скорость momentum, позволяя не пропустить минимум.

Тезис Ilya Sutskever(часть 7.2) - http://www.cs.utoronto.ca/~ilya/pubs/ilya_sutskever_phd_thesis.pdf

Adagrad

В градиентных методах выбор шага очень сильно влияет на сходимость алгоритма. **Adagrad** позволяет автоматически подстраивать наш алгоритм под искривлённость нашей функции.

- Adagrad подбирает **большой** шаг для тех параметров, что обновляются редко и **маленький** для тех, что обновляются часто.
- Чем **больше** сумма квадратов градиентов, тем **меньше** шаг. Значит при большой дисперсии градиентов (меньшая искривлённость функции) – шаг уменьшается.

Algorithm 1 AdaGrad

Require: Cost function $J(\mathbf{w})$, initial $\mathbf{w}^{(0)}$, initial $\mathbf{G}^{(0)}$, constants η and ϵ.

1: Set $i \leftarrow 0$
2: **while** $\|\mathbf{w}^{(i)} - \mathbf{w}^{(i-1)}\| > \epsilon$ **do**
3: Compute gradient vector $\mathbf{g}^{(i)} = \nabla J(\mathbf{w}^{(i)})$
4: Accumulate $\mathbf{G}^{(i+1)} \leftarrow \mathbf{G}^{(i)} + \mathbf{g}^{(i)} \cdot \mathbf{g}^{(i)}$
5: Update parameters $\mathbf{w}^{(i+1)} \leftarrow \mathbf{w}^{(i)} - \frac{\eta}{\sqrt{\mathbf{G}^{(i+1)}+\epsilon}}\mathbf{g}^{(i)}$
6: Update $i \leftarrow i + 1$
7: **end while**
8: Return $\widehat{\mathbf{w}} \leftarrow \mathbf{w}^{(i-1)}$

Обычно вместо G_i используют $diag(G_i)$ - диагональную матрицу быстрее инвертировать и извлекать корень, а на качество не влияет.

Если начальные градиенты большие, то сразу будет маленький шаг, а это замедлит сходимость. Поэтому Adagrad все равно сильно зависит от глобального μ – он позволяет влиять на скорость сходимости.

RMSprop

Для решения проблемы **Adagrad** с затуханием шага используется **RMSProp**.

AdaGrad

$$g_0 = 0$$

$$g_{t+1} \leftarrow g_t + \nabla_\theta \mathcal{L}(\theta)^2$$

$$\theta_j \leftarrow \theta_j - \epsilon \frac{\nabla_\theta \mathcal{L}}{\sqrt{g_{t+1}} + 1e^{-5}}$$

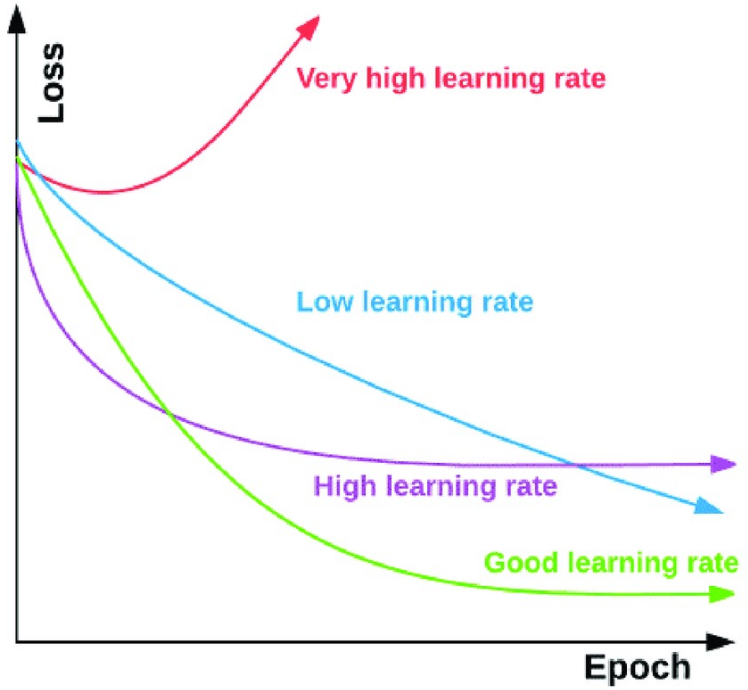
RMS Prop

$$g_0 = 0, \alpha \simeq 0.9$$

$$g_{t+1} \leftarrow \alpha \cdot g_t + (1-\alpha)\nabla_\theta \mathcal{L}(\theta)^2$$

$$\theta_j \leftarrow \theta_j - \epsilon \frac{\nabla_\theta \mathcal{L}}{\sqrt{g_{t+1}} + 1e^{-5}}$$

RMSProp устанавливает меньший вес текущим градиентам, и больший вес сумме предыдущих значений.



Влияние размера размера шага на ошибку функции ошибки.

RMSProp(или AdaGrad) стоит использовать если в обучении много редко встречающихся семплов. Например, матрицы эмбедингов контекстно-независимых моделях.

ADAM

Adam – использует адаптивный шаг, как RMSProp и историю градиентов как Momentum.

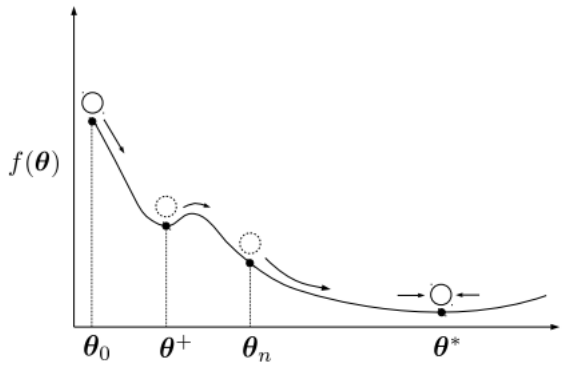
m –момент первого рода градиента (среднее)
v – нецентрированный момент второго рода градиента (дисперсия)

Если параметры шага моментов низкие(β_1, β_2 блики к 1) - m, v – стремятся к 0, поэтому авторы предложили обратное взвешивание моментов с изменением шага(bias-correction term):

$$m_t \leftarrow m_t/(1-\beta_1^t)$$
$$v_t \leftarrow v_t/(1-\beta_2^t)$$

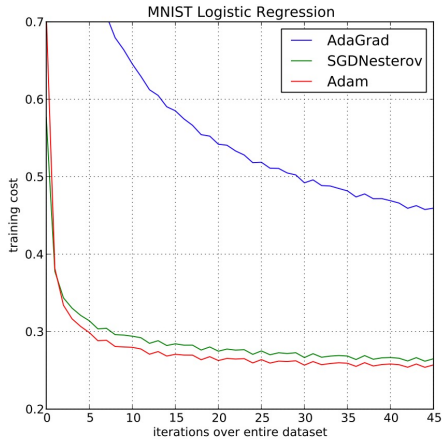
Значения гиперпараметров:
$\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$

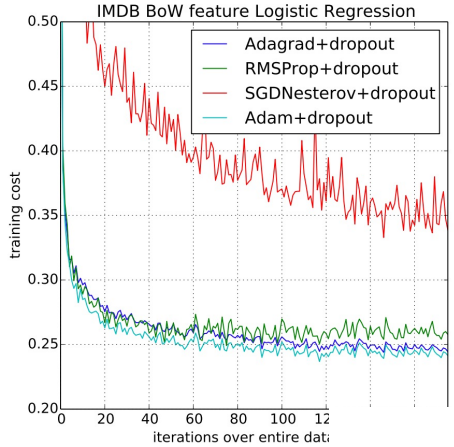
Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
$t \leftarrow 0$ (Initialize timestep)
while θ_t not converged **do**
$t \leftarrow t + 1$
$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
$\hat{m}_t \leftarrow m_t/(1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
$\hat{v}_t \leftarrow v_t/(1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t/(\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
end while
return θ_t (Resulting parameters)

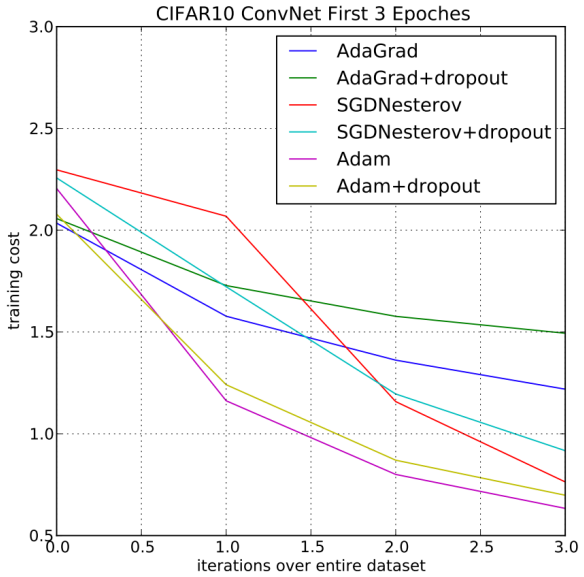


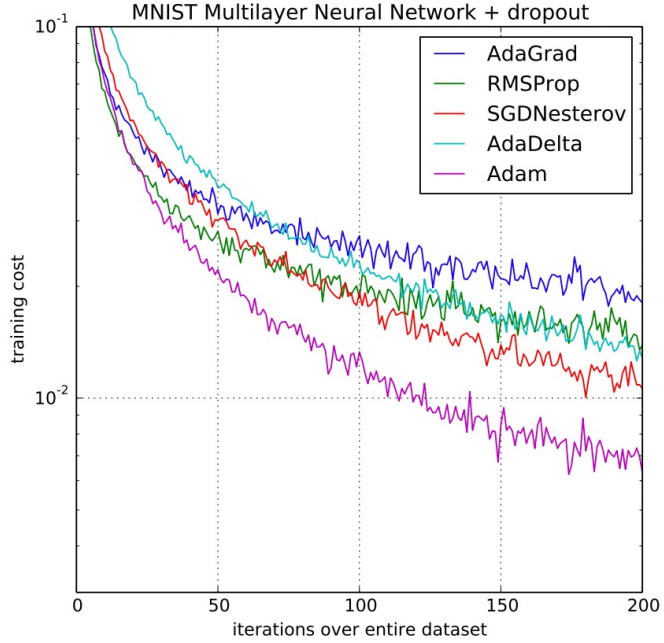
Поведение ADAM схоже с тяжелым мячом с учетом трения – он перескакивает локальный минимум θ^+, останавливаясь в плоском минимуме θ^*

ADAM

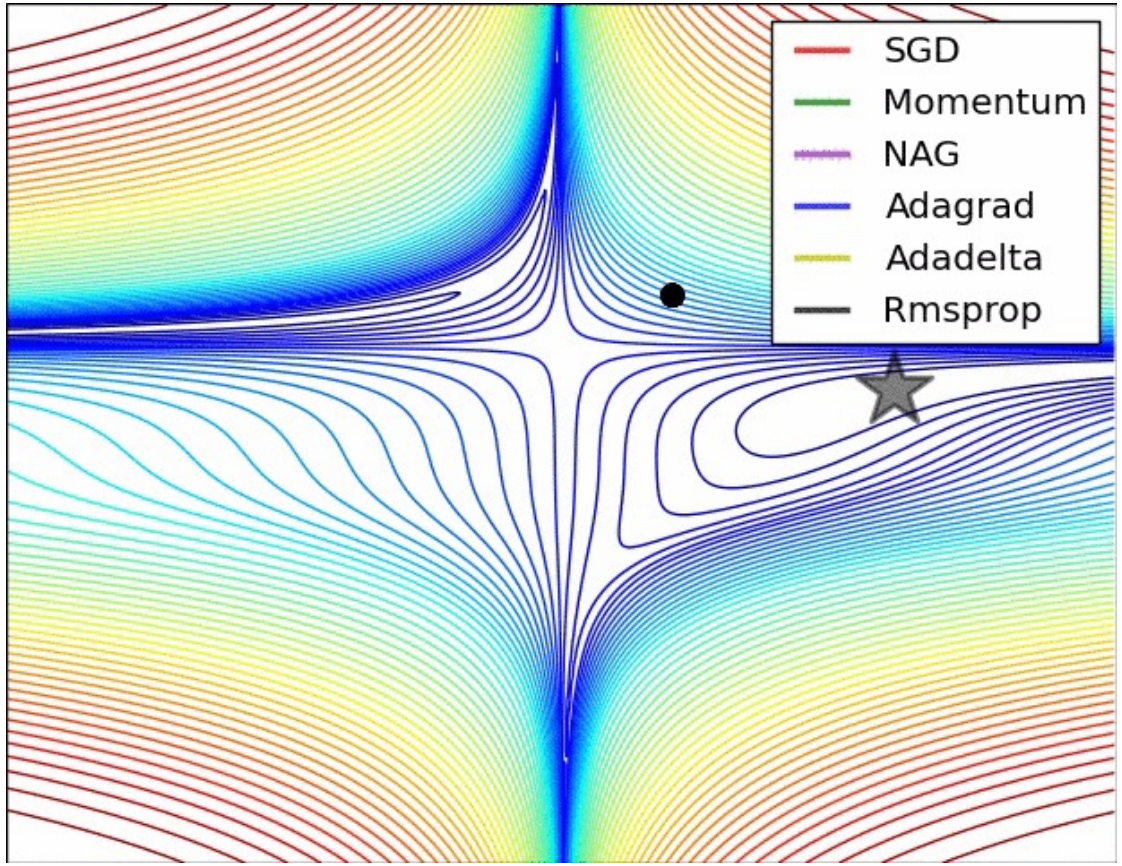

MNIST Logistic Regression
AdaGrad
SGDNesterov
Adam
training cost
iterations over entire dataset


IMDB BoW feature Logistic Regression
Adagrad+dropout
RMSProp+dropout
SGDNesterov+dropout
Adam+dropout
training cost
iterations over entire dat

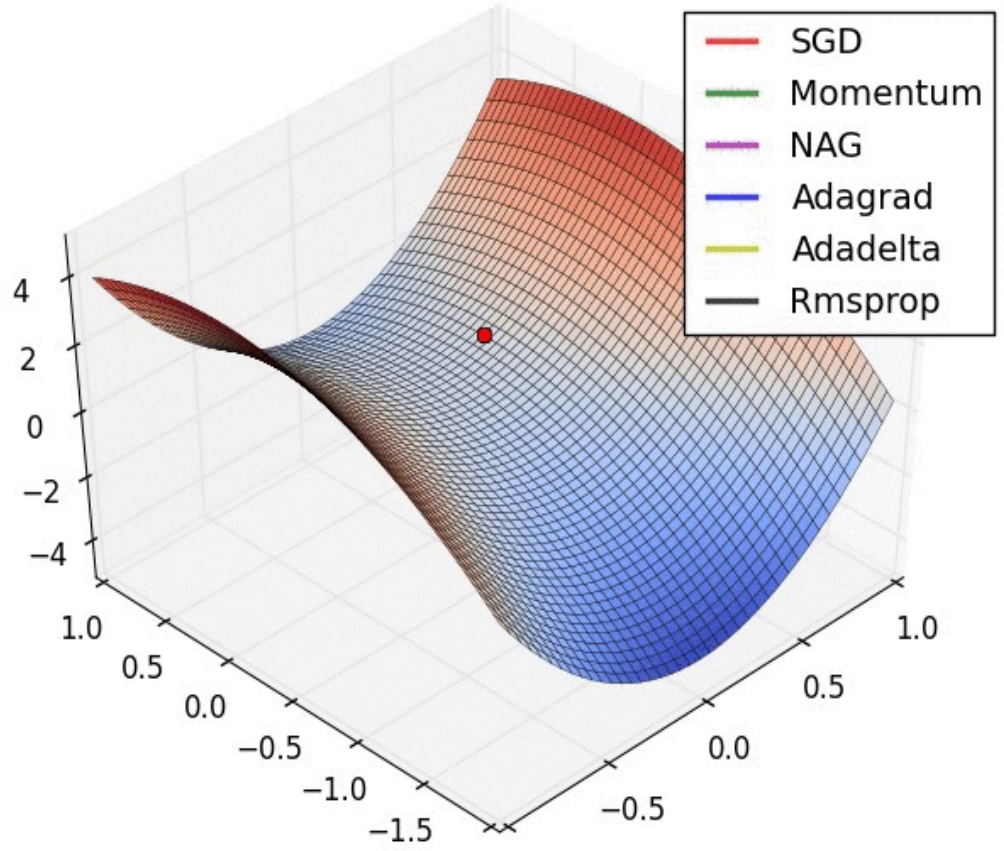

CIFAR10 ConvNet First 3 Epoches
AdaGrad
AdaGrad+dropout
SGDNesterov
SGDNesterov+dropout
Adam
Adam+dropout
training cost
iterations over entire dataset


MNIST Multilayer Neural Network + dropout
AdaGrad
RMSProp
SGDNesterov
AdaDelta
Adam
training cost
iterations over entire dataset

Анимация алгоритмов сходимости



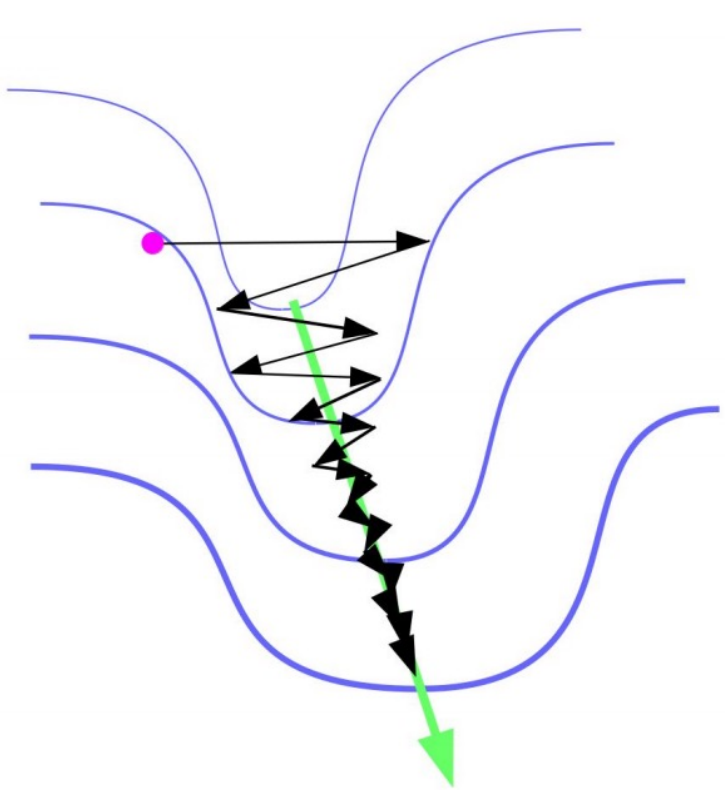
Заметно, что из-за инерциальной породы momentum-based методов – Momentum и NAG сначала уходят не туда
Адаптивные методы сначала выбирают координаты наименьшего спуска



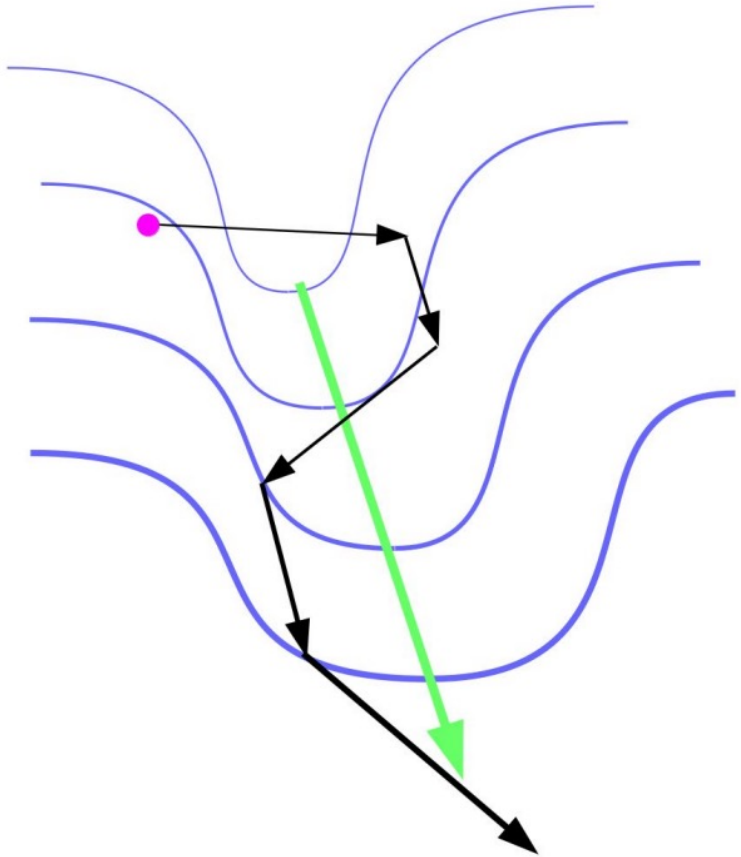
В случае симметричной искривленности пространства – SGD не может найти оптимальное направление спуска, а momentum – based после определенного набора итераций - находят

Методы второго порядка

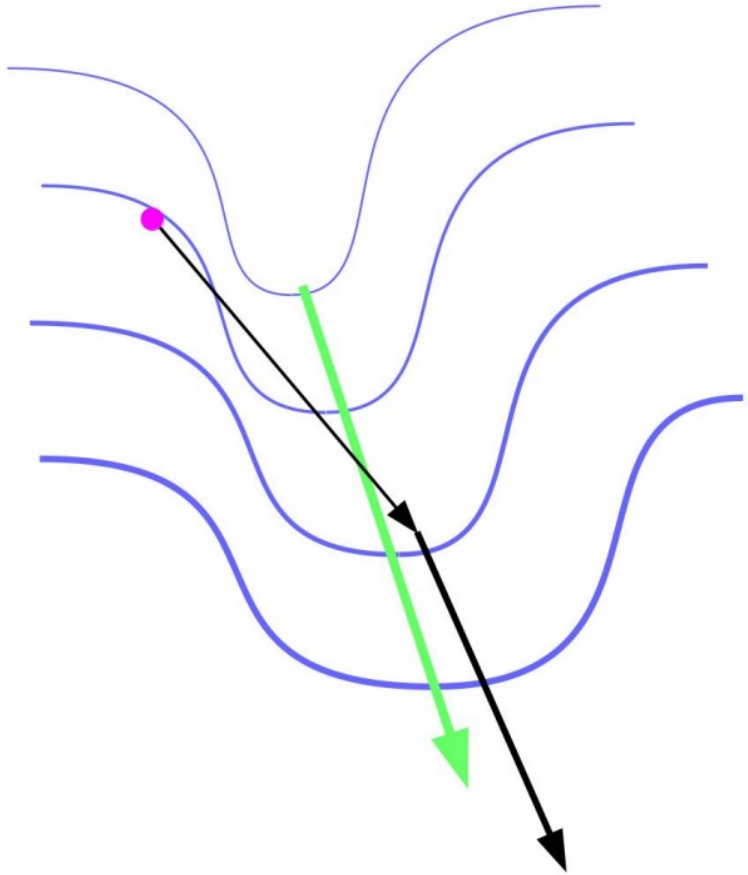
Возможно ли аппроксимировать функционал ошибки более эффективно?



Градиентный спуск



Градиентный спуск с моментом



Методы второго порядка

Методы второго порядка

Аппроксимация функции разложением может сходится за длительное время(зависит от искривлённости функции)

$$f(\theta_t) = f(\theta_t) + \nabla f(\theta)(\theta - \theta_t)$$

Наилучшая аппроксимация на x_0 достигается выбором правильного η

Возможно ли нам ускорить сходимость функции, если аппроксимировать функцию квадратичным функционалом?

$$f(\theta) = f(\theta_t) + \nabla f(\theta)(\theta - \theta_t) + \nabla^2 f(\theta) * (\theta - \theta_t)^2$$

$$\frac{df}{d(\theta - \theta_t)} = -\frac{\nabla f(\theta)}{\nabla^2 f(\theta)}$$

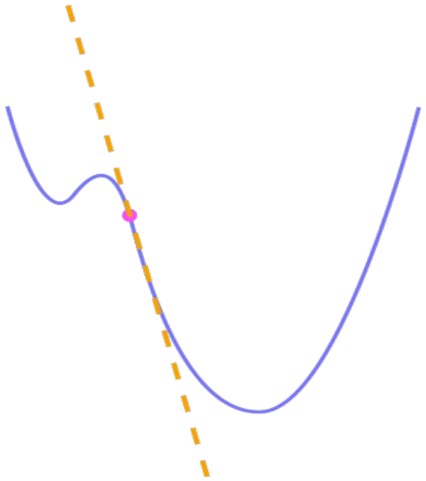
Минимум квадратичной аппроксимации находим с помощью второй производной

Тогда обновление значений функции

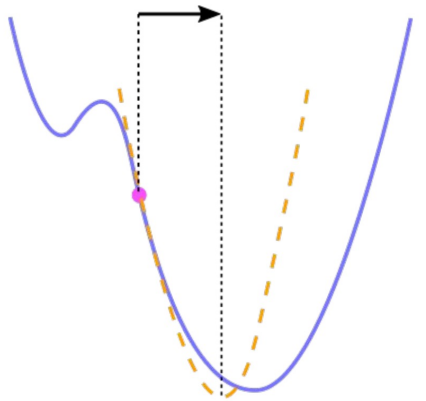
$$\theta_{t+1} = \theta_t - \frac{\nabla f(\theta_t)}{\nabla^2 f(\theta_t)} = \theta_t - \nabla f(\theta_t) \cdot \left(\nabla^2 f(\theta_t)\right)^{-1}$$

Почему стоит использовать? Если начальное приближение выбрано удачно, то сходимость метода на порядок выше методов первого порядка. Можно использовать для простых моделей или небольших сетов.

Почему всегда не использовать? Обратный гессиан вычислительно сложно считать и хранить. Для больших нейронных сетей невыгодно использовать.



Аппроксимация полиномом 1ого порядка



Аппроксимация полиномом второго порядка

Определения

Гессиан($\nabla^2 f(x)$) – квадратная матрица вторых производных функции.

- Гессиан симметричен - $\nabla^2 f(x) = [\nabla^2 f(x)]^{\mathrm{T}}$
- Сложность вычисления – $O(\frac{n^2}{2} * k)$, где n – количество независимых переменных, k– сложность вычисления функции

$$\begin{pmatrix} \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\boldsymbol{x})}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f(\boldsymbol{x})}{\partial x_n \partial x_n} \end{pmatrix}$$

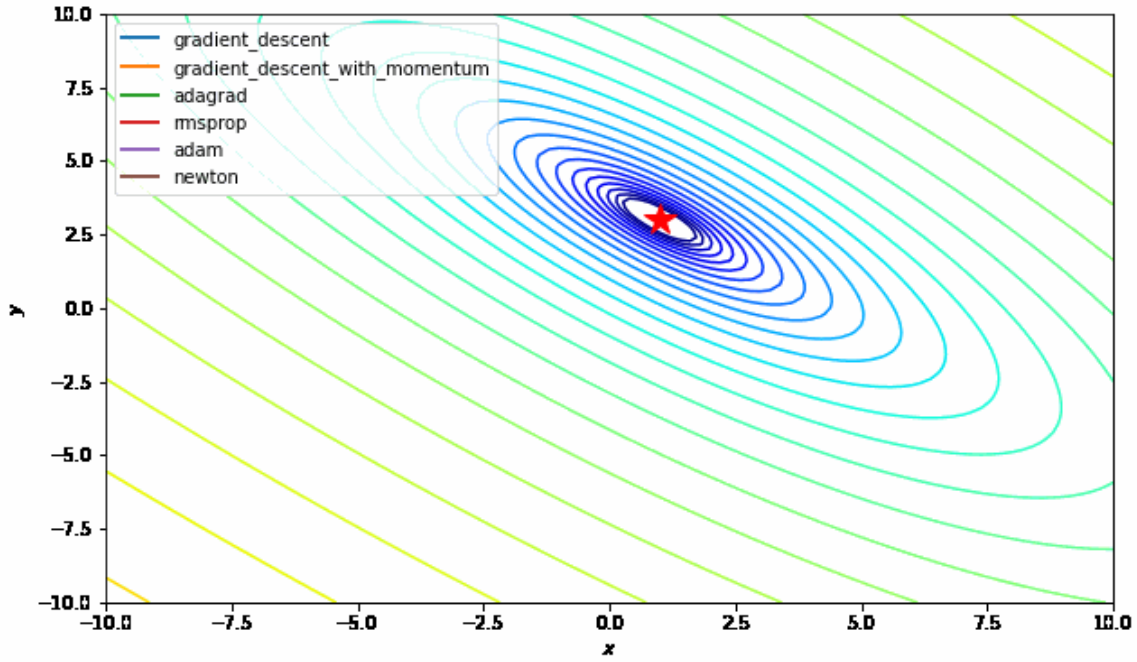
При условии вычисленной стационарной точки, гессиан позволяет определить тип этой стационарной точки – локальный минимум, локальный максимум или просто стационарная точка.

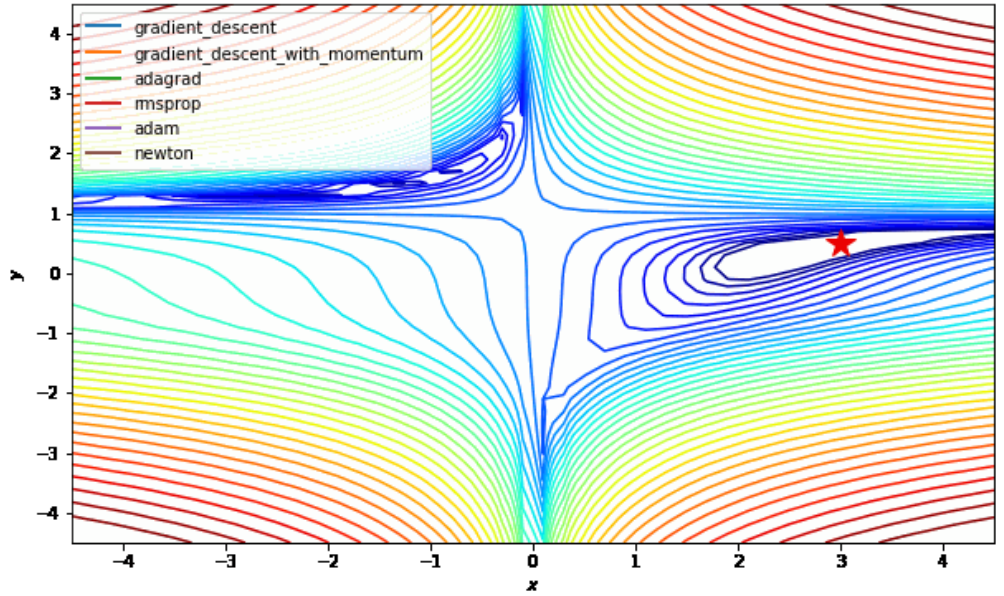
Условия максимума, минимума в гессиане:

Гессиан $\boldsymbol{\nabla^2 f(x)}$ и $x_0 \in R^d$ стационарная точка, тогда:

- Если $\boldsymbol{\nabla^2 f(x_0)}$ – положительно определен, x_0 - локальный минимум
- Если $\boldsymbol{\nabla^2 f(x_0)}$ – отрицательно определен, x_0 - локальный максимум
- Если $\boldsymbol{\nabla^2 f(x_0)}$ – не определён, о x_0 ничего нельзя утверждать.

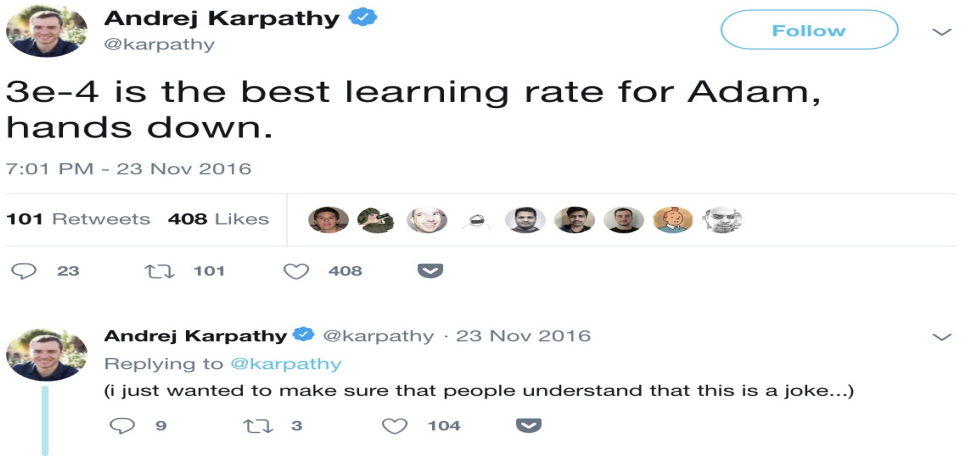
Анимация алгоритмов сходимости





Рекомендации по обучению

Выбор шага



Критерий останова

- Пока норма градиента весов не станет меньше ϵ
- На основании значений функции ошибки
- На основании значений бизнес метрики

Выбор размера батча

1. Выбор батча:
   - Большой батч – дольше считаем градиент, но точнее. Для маленьких датасетов лучше использовать полный сет.
   - Маленький батч – быстрее считаем градиент, но менее точно(1 <= 248)
2. Общие рекомендации - https://arxiv.org/abs/1206.5533 - статья от Йошуа Бенжио с общими рекомендациями к обучению.
3. Для маленьких сетов(10к)/моделей можно использовать методы второго порядка на полном батче.

Общие рекомендации по обучению

1) Всегда мешайте объекты перед обучением
2) Проверяйте ошибку на трейне и на тесте
3) Проверяйте что градиенты посчитаны корректно
4) Пробуйте разные η на небольших подмножествах выборки
5) В среднем adam работает лучше всех.