Lecture 4. Into to Neural Networks

Michael Vasilkovsky

Who am I?

MIPT
MOSCOW INSTITUTE
OF PHYSICS AND TECHNOLOGY

Yandex

ШКОЛА АНАЛИЗА ДАННЫХ

Dbrain

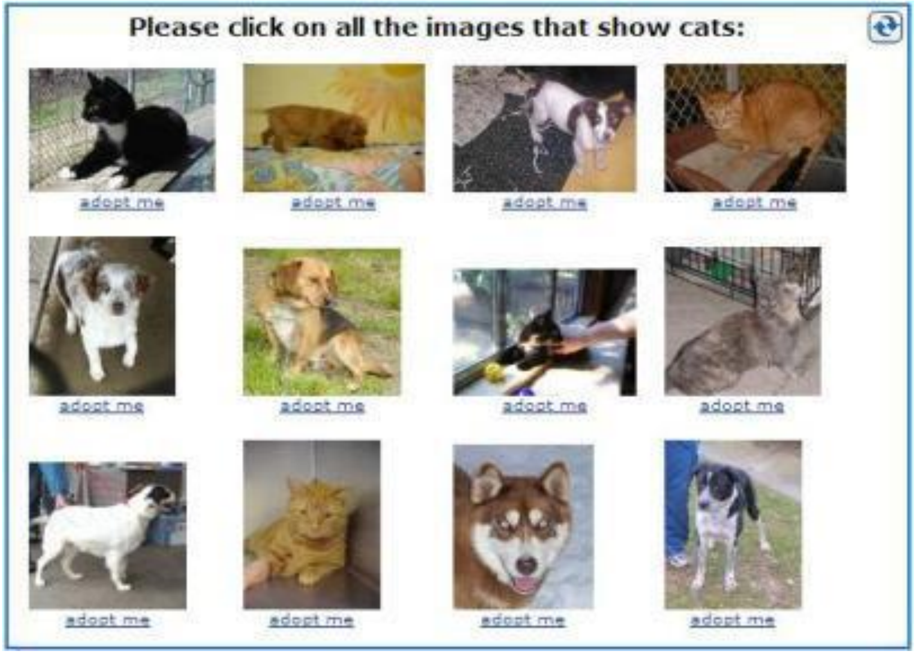
Skoltech
Skolkovo Institute of Science and Technology

Lecture plan

- Examples of NN usage
- Biological motivation
- Mathematical modelling
- Fully connected NNs
- Activation functions
- Loss functions and backprop
- Gradient descent
- Optimization
- Weight initialization

Once upon a time...

2006



Computer vision = 60%

$0.6^{12} = 0.00217$

2014

Completed • Swag • 215 teams

Dogs vs. Cats

Wed 25 Sep 2013 – Sat 1 Feb 2014 (8 months ago)

Dashboard

Private Leaderboard - Dogs vs. Cats

This competition has completed. This leaderboard reflects the final standings.

See someone

#	Δ1w	Team Name * in the money	Score	Entries	Last Submission UTC (Best – Las
1	—	Pierre Sermanet *	0.98914	5	Sat, 01 Feb 2014 21:43:19 (-
2	↑26	orchid *	0.98309	17	Sat, 01 Feb 2014 23:52:30
3	—	Owen	0.98171	15	Sat, 01 Feb 2014 17:04:40 (-
4	new	Paul Covington	0.98171	3	Sat, 01 Feb 2014 23:05:20
5	↓3	Maxim Milakov	0.98137	24	Sat, 01 Feb 2014 18:20:58

$$0.989^{12} = 0.875$$

2014

Microsoft Research

Search Mi

Our research   Connections   Careers   About us

All   Downloads   Events   Groups   News   People   Projects   Publications

ASIRRA

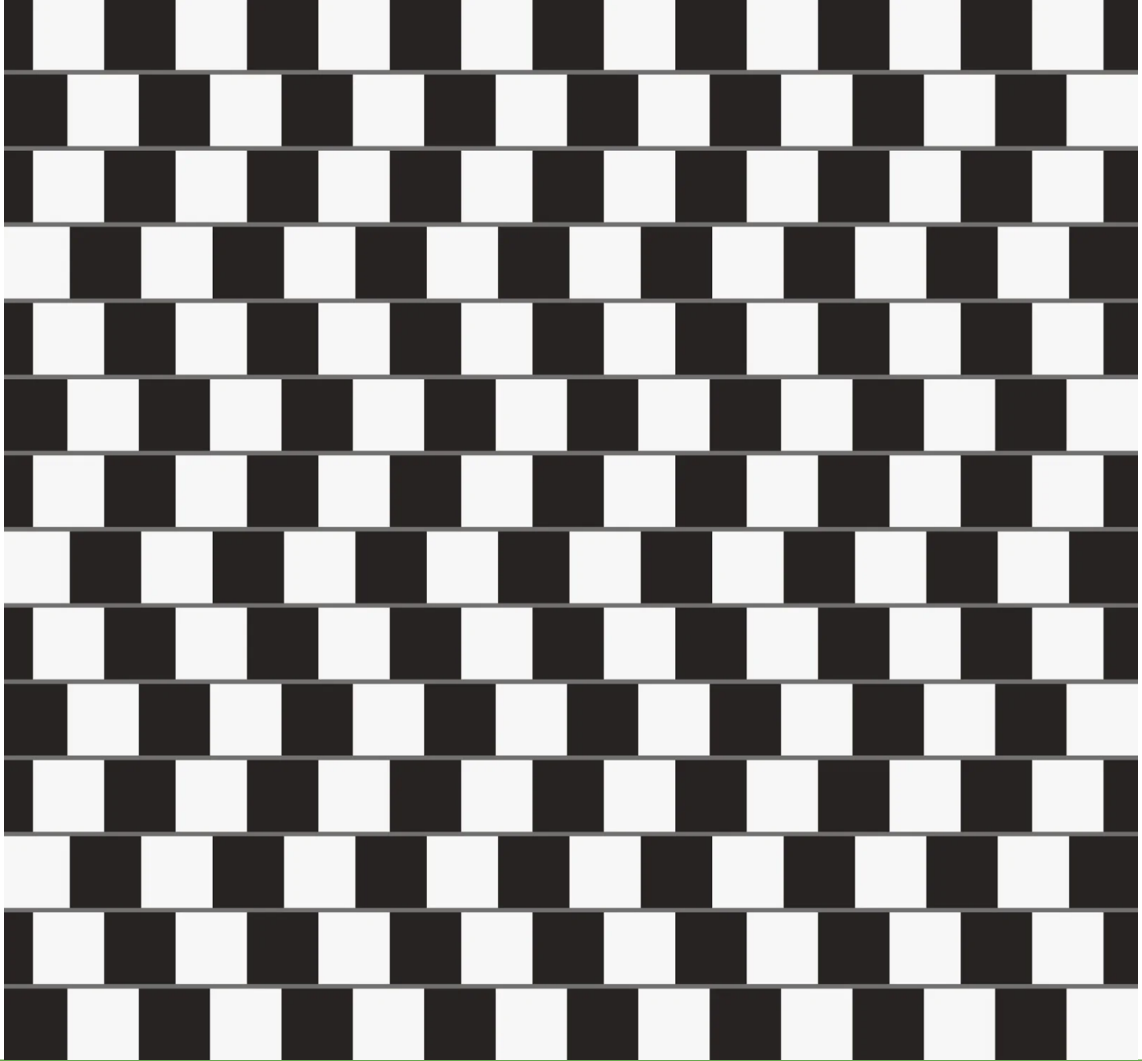
After 8 years of operation, Asirra is shutting down effective October 1, 2014. Thank you to all of our users!

Problem statement

What we see

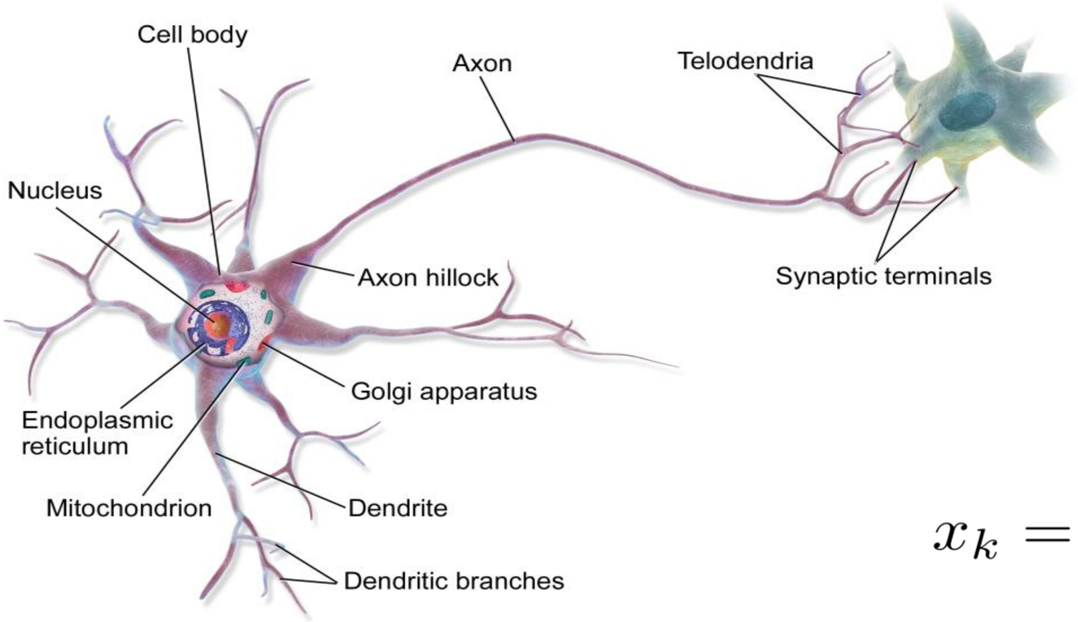
0	3	2	5	4	7	6	9	8
3	0	1	2	3	4	5	6	7
2	1	0	3	2	5	4	7	6
5	2	3	0	1	2	3	4	5
4	3	2	1	0	3	2	5	4
7	4	5	2	3	0	1	2	3
6	5	4	3	2	1	0	3	2
9	6	7	4	5	2	3	0	1
8	7	6	5	4	3	2	1	0

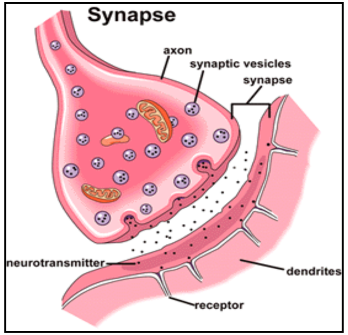
What a computer sees



Enough motivation!

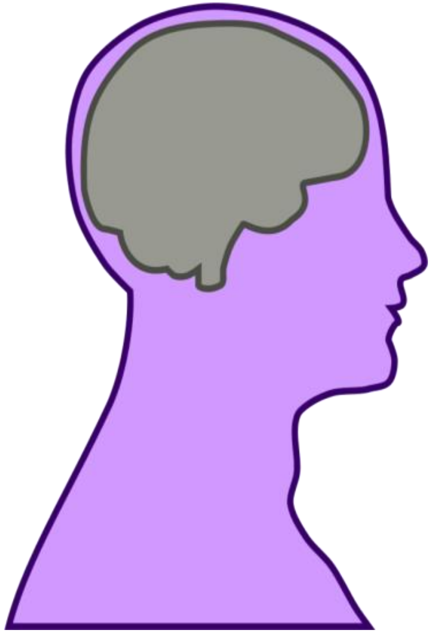
Neuron model





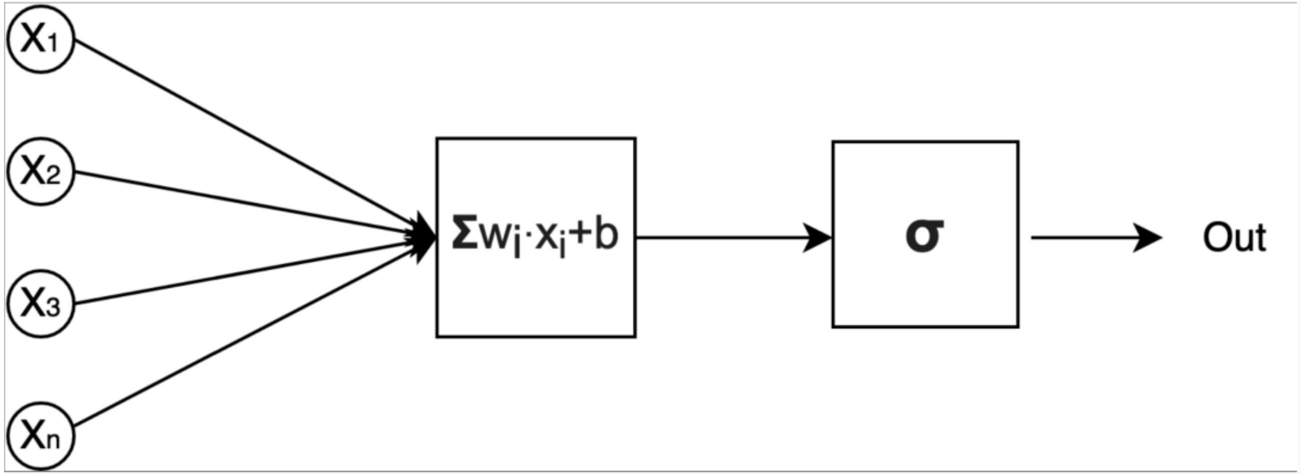
$$x_k = H\left(\sum_i w_{ik} x_i - \tau\right)$$

Our brain



- 100 billion neurons
- average neuron is connected to 1000-10000 other neurons
- 100 trillion synapses
- 10-25% is in visual cortex

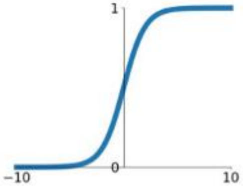
Mathematical model



Activation functions

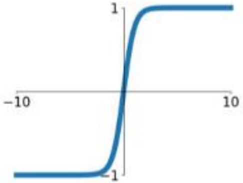
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



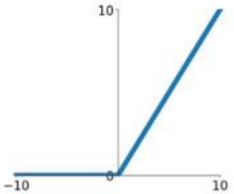
tanh

$\tanh(x)$



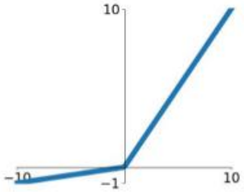
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

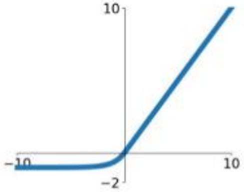


Maxout

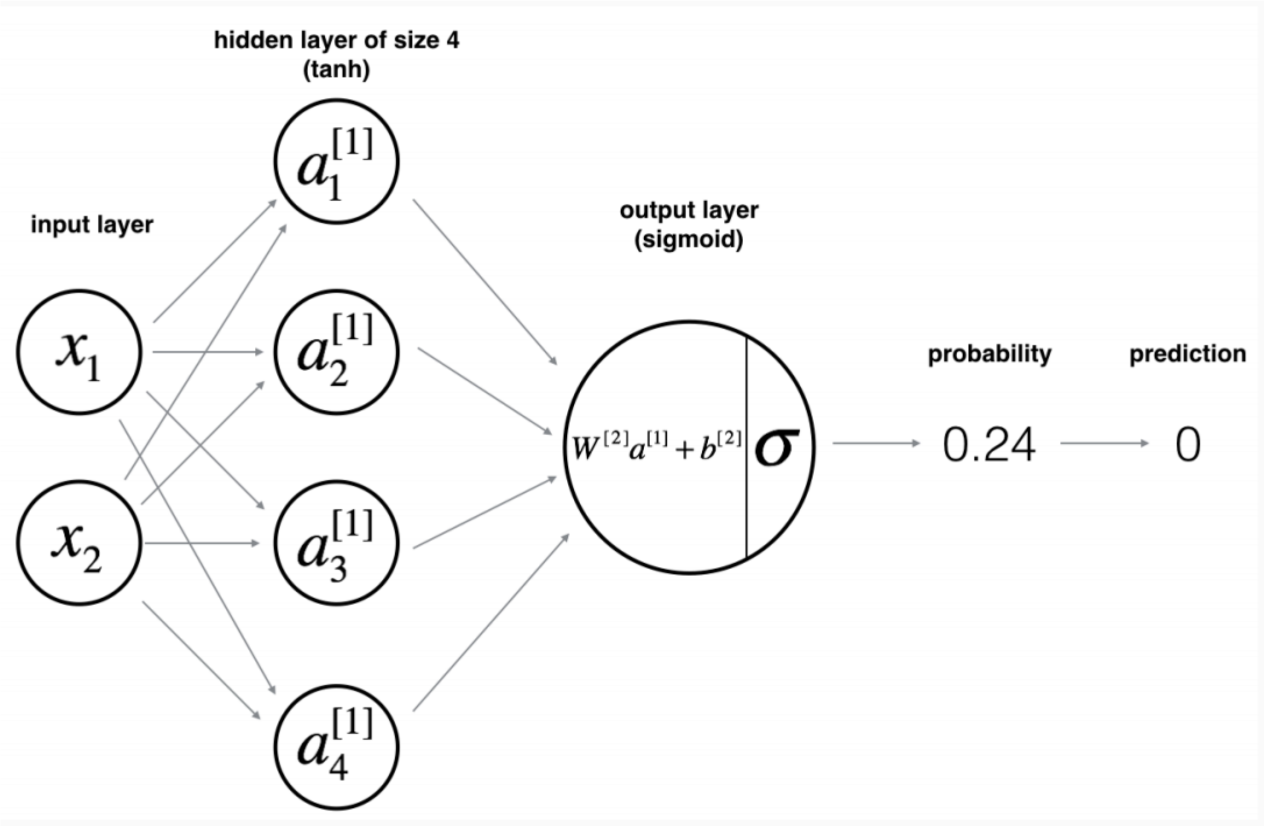
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Multilayer perceptron



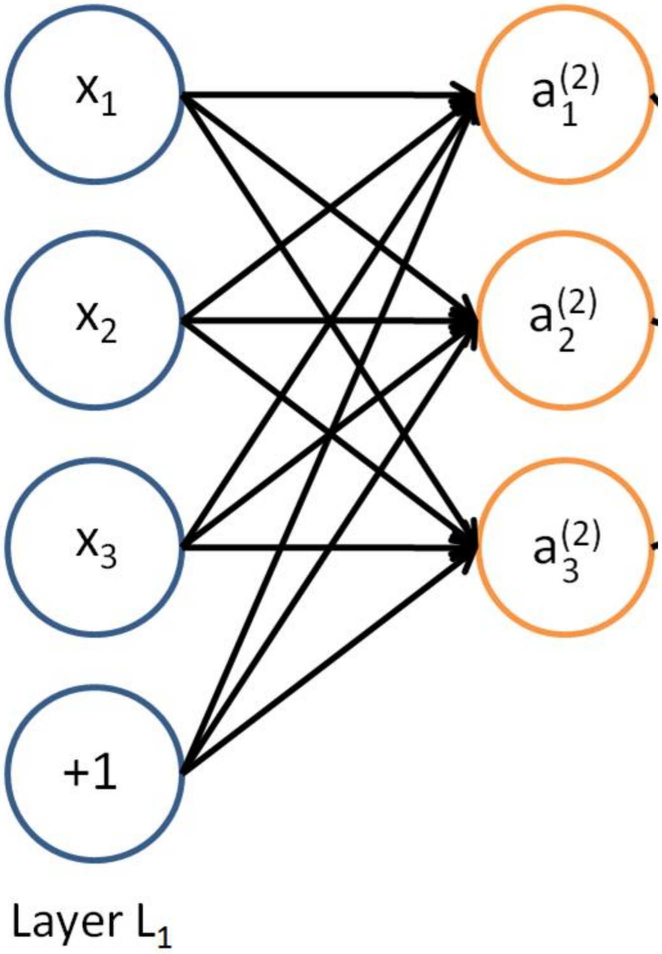
$$a = L(x), \quad x \in \mathcal{R}^m, a \in \mathcal{R}^n$$

$$a_i = \sigma\Big(\sum_j w_{ij} x_j + b_i\Big)$$

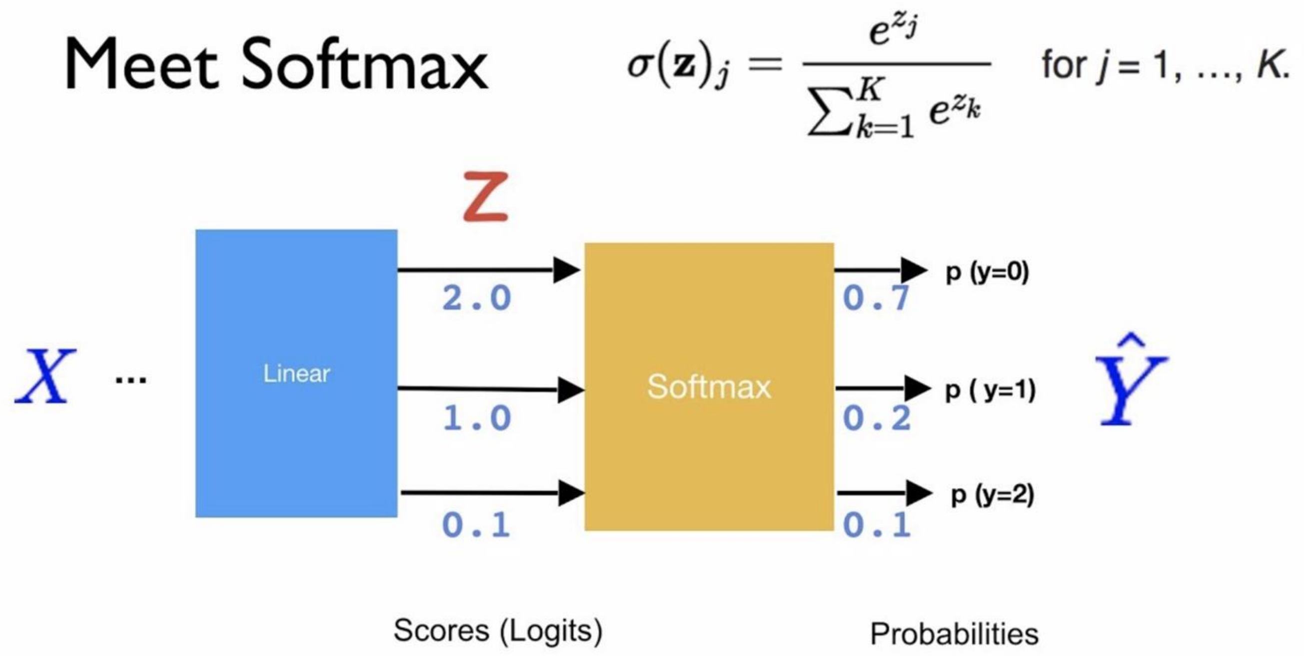
$$a = \sigma\Big(Wx + b\Big), \quad b \in \mathcal{R}^n$$

Hint, don’t look there!

$$\begin{bmatrix} 3 & 2 & 0 \\ 0 & 4 & 1 \\ 2 & 0 & 1 \end{bmatrix} \underset{x}{\begin{bmatrix} 4 \\ 3 \\ 1 \end{bmatrix}} = \underset{y}{\begin{bmatrix} 18 \\ 13 \\ 9 \end{bmatrix}}$$



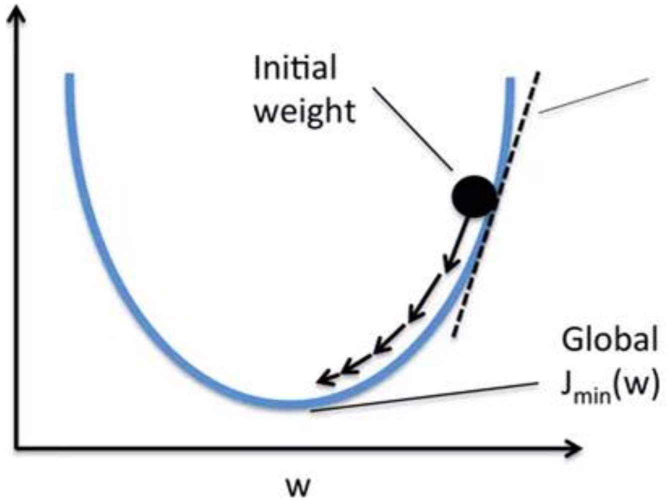
Multiclass case



Recap: Gradient decent

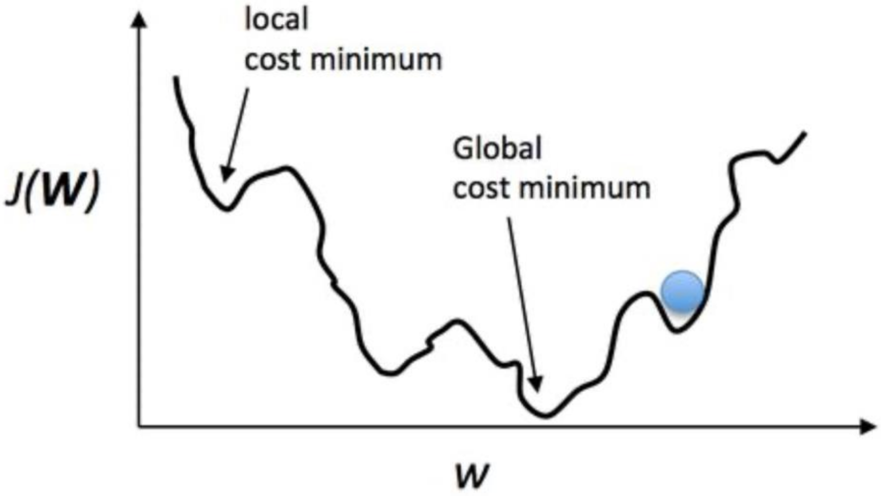
- We want to minimize $f(x)$
- Initialize x_0
- For each x_n do...
- Compute gradient (derivative) $f'(x_n)$
- Make a step:

$$x_{n+1} = x_n - \alpha f'(x_n)$$



- Just change the notation and minimize the loss:

$$w_n = w_{n-1} - \alpha \frac{\partial \mathcal{L}}{\partial w}(X, y, w_{n-1})$$


local
cost minimum
Global
cost minimum
J(W)
W

Backprop

$$x \to f \to y, \; y = f(x, w)$$

$$\frac{\partial L}{\partial y} \textbf{ is known}$$

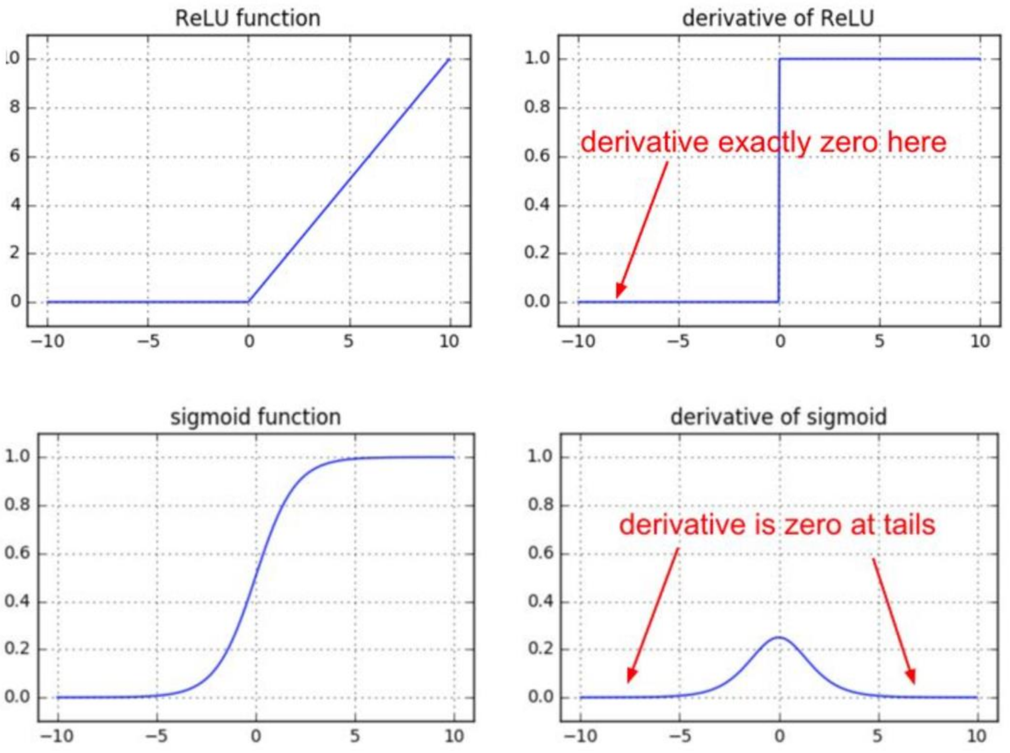
$$\frac{\partial L}{\partial x} = ?, \quad \frac{\partial L}{\partial w} = ?$$

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial x} = \frac{\partial L}{\partial y}\frac{\partial f}{\partial x}$$

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial w} = \frac{\partial L}{\partial y}\frac{\partial f}{\partial w}$$

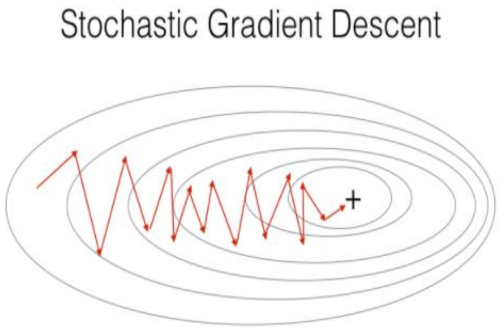
$$y = Wx \Rightarrow \frac{\partial L}{\partial x} = W^T \frac{\partial L}{\partial y}$$

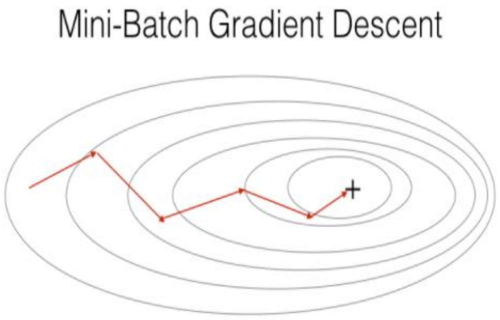
Vanishing gradients



Variations of stochastic gradient descent (SGD)

- Compute gradient using only a subsample of a smaller size (called batch size)
- It accelerates convergence because the algorithm usually converges in approximately the same number of steps
- Use always this method, never compute gradient by the entire sample

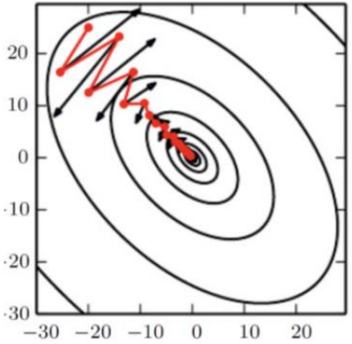




Optimizers

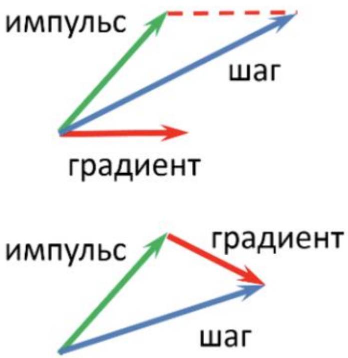
Momentum — экспоненциальное скользящее среднее градиента по $\approx \frac{1}{1-\gamma}$ последним итерациям [Б.Т.Поляк, 1964]:

$$v := \gamma v + \eta \mathscr{L}_i'(w)$$
$$w := w - v$$



NAG (Nesterov's accelerated gradient) — стохастический градиент с импульсом Нестерова [1983]:

$$v := \gamma v + \eta \mathscr{L}_i'(w - \gamma v)$$
$$w := w - v$$



Optimizers

RMSProp (running mean square) — адаптация скорости изменения весов, скользящим средним по $\approx \frac{1}{1-\alpha}$ итерациям:

$$G := \alpha G + (1-\alpha)\mathscr{L}_i'(w) \odot \mathscr{L}_i'(w)$$

$$w := w - \eta \mathscr{L}_i'(w) \oslash \left(\sqrt{G} + \varepsilon\right)$$

где $\odot$ и $\oslash$ — покоординатное умножение и деление векторов.

AdaDelta (adaptive learning rate) — двойная нормировка приращений весов, после которой можно брать $\eta = 1$:

$$G := \alpha G + (1-\alpha)\mathscr{L}_i'(w) \odot \mathscr{L}_i'(w)$$

$$\delta := \mathscr{L}_i'(w) \odot \frac{\sqrt{\Delta}+\varepsilon}{\sqrt{G}+\varepsilon}$$

$$\Delta := \alpha \Delta + (1-\alpha)\delta \odot \delta$$

$$w := w - \eta \delta$$

Optimizers

Adam (adaptive momentum) = импульс + RMSProp:

$$v := \gamma v + (1-\gamma)\mathscr{L}_i'(w) \qquad \hat{v} := v(1-\gamma^k)^{-1}$$

$$G := \alpha G + (1-\alpha)\mathscr{L}_i'(w) \odot \mathscr{L}_i'(w) \qquad \hat{G} := G(1-\alpha^k)^{-1}$$

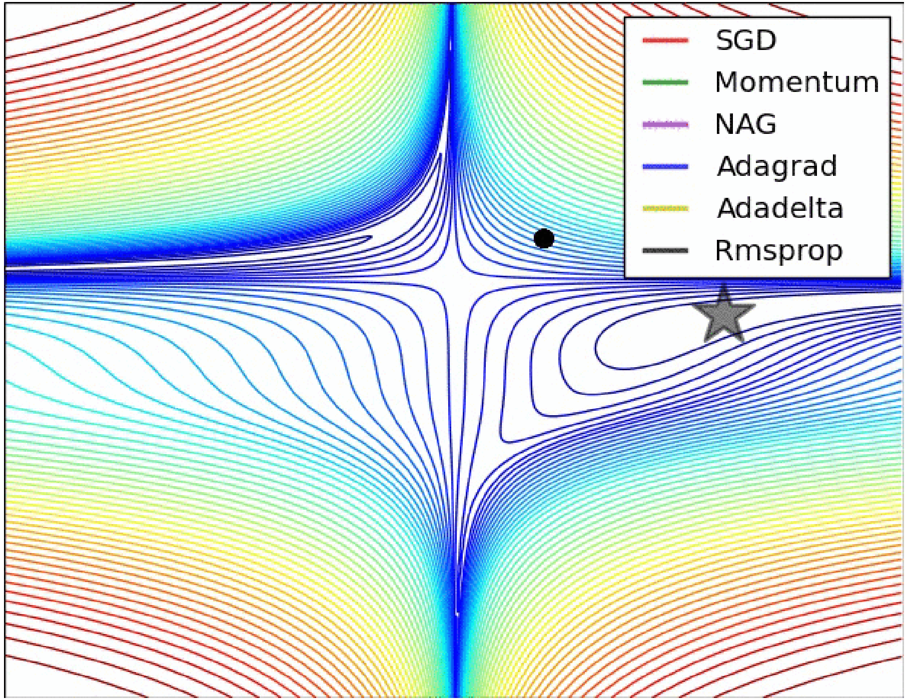
$$w := w - \eta \hat{v} \oslash \left(\sqrt{\hat{G}} + \varepsilon\right)$$

Калибровка $\hat{v}$, $\hat{G}$ увеличивает v, G на первых итерациях,
где k — номер итерации; $\gamma = 0.9$, $\alpha = 0.999$, $\varepsilon = 10^{-8}$

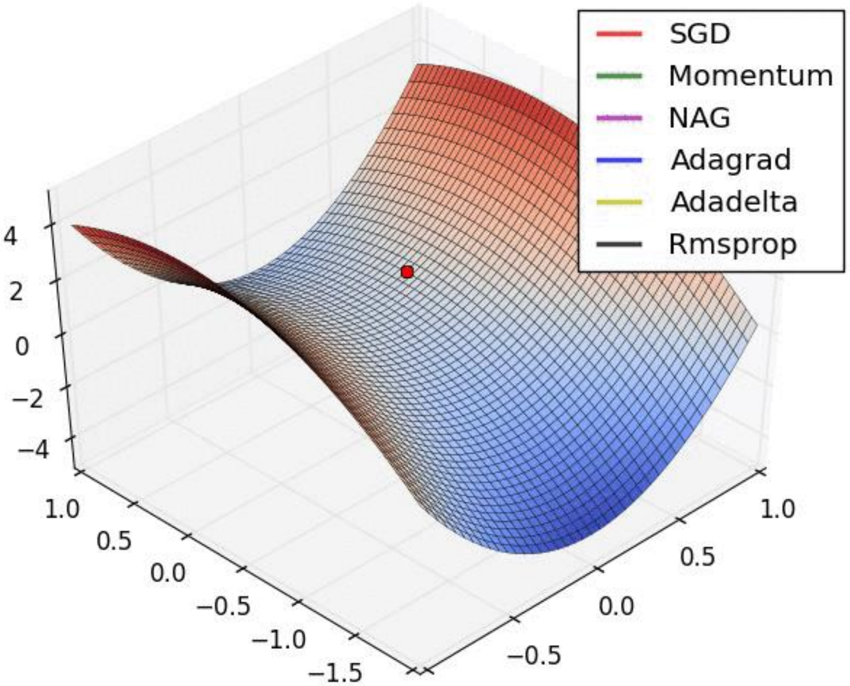
Nadam (Nesterov-accelerated adaptive momentum):
те же формулы для v, $\hat{v}$, G, $\hat{G}$,

$$w := w - \eta\left(\gamma\hat{v} + \tfrac{1-\gamma}{1-\gamma^k}\mathscr{L}_i'(w)\right) \oslash \left(\sqrt{\hat{G}} + \varepsilon\right)$$

Optimizers



SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop



SGD
Momentum
NAG
Adagrad
Adadelta
Rmsprop
4
2
0
−2
−4
1.0
0.5
0.0
−0.5
−1.0
−1.5
1.0
0.5
0.0
−0.5