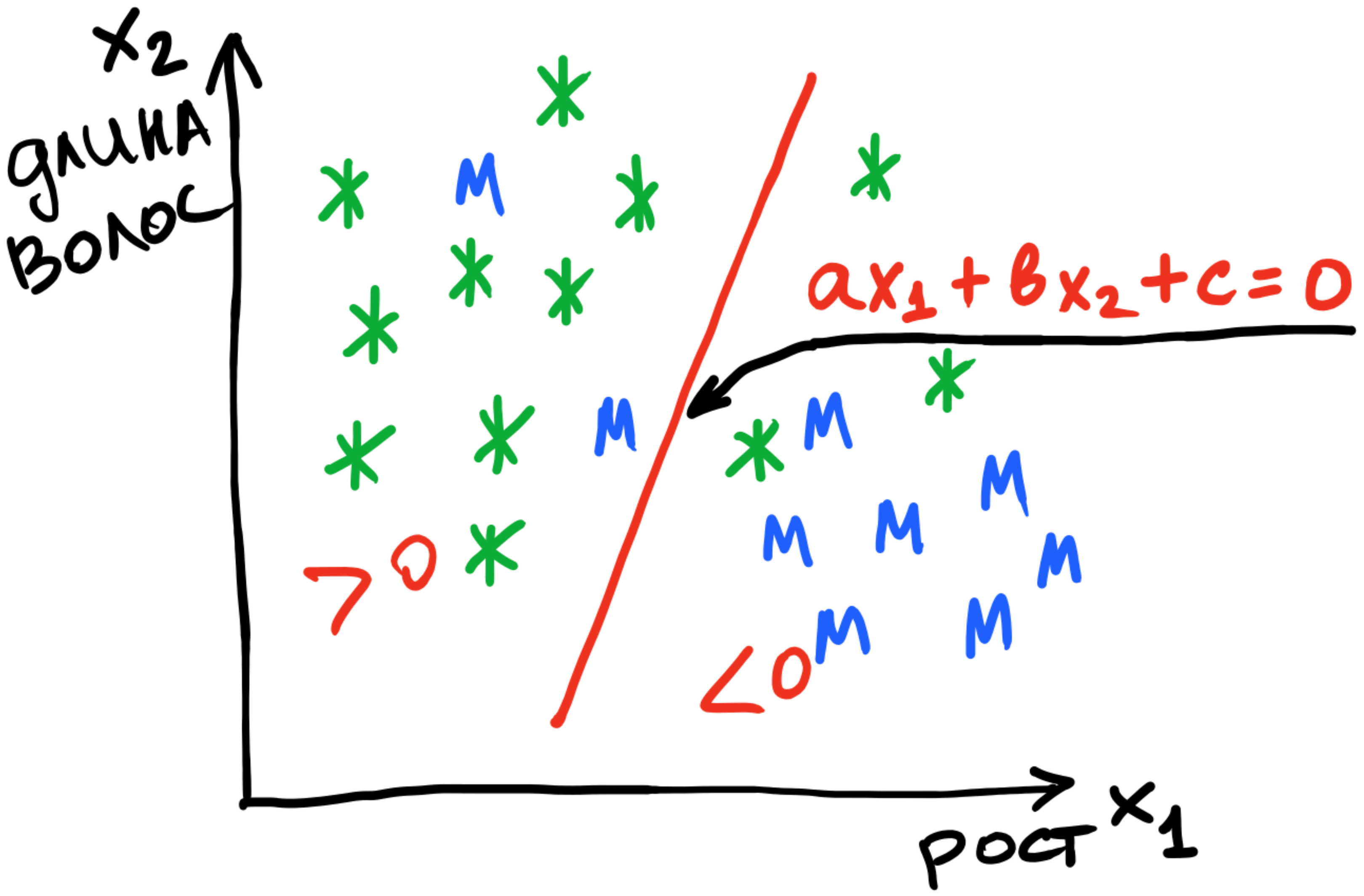
MML minor #4

Нейронные сети

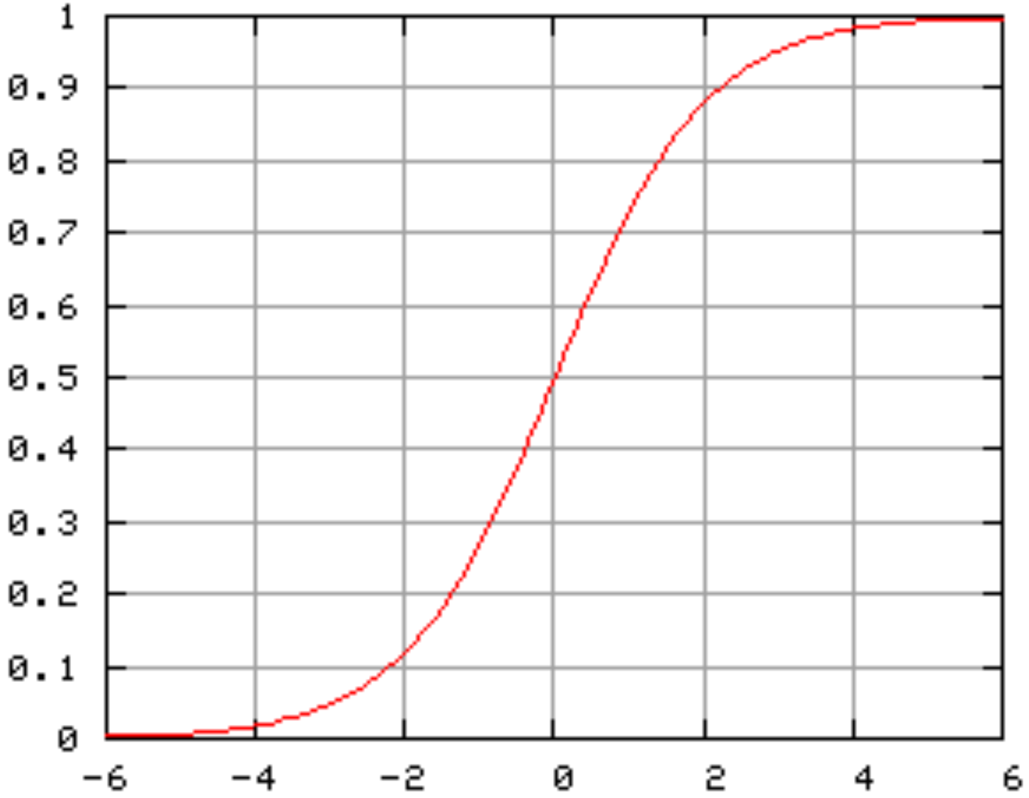
Линейная классификация



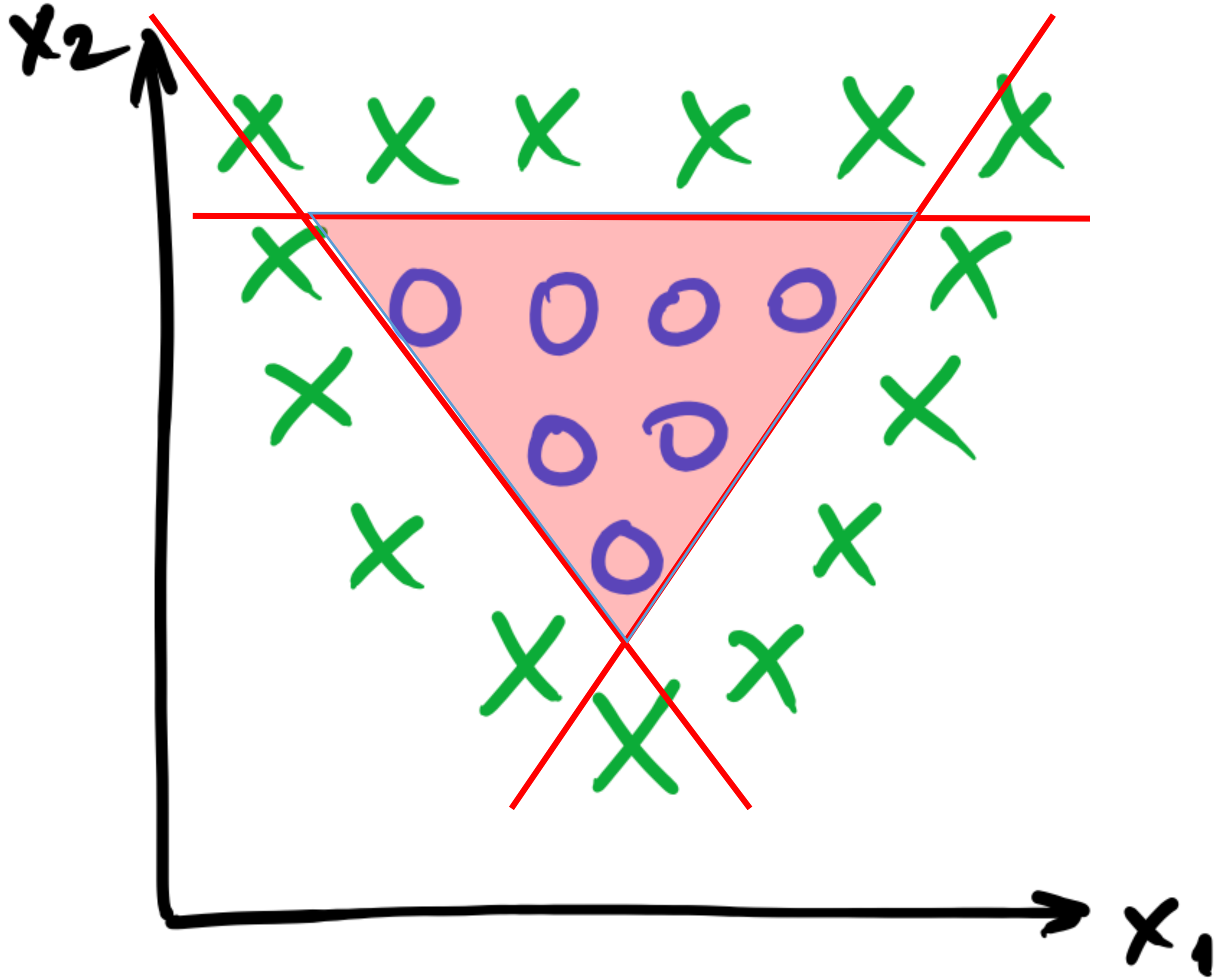
Напоминание: логистическая регрессия

Вероятность положительного класса: $y(x) = \sigma(w^T x) = \frac{1}{1+e^{-(w^T x)}}$

Логистическая функция $\sigma(x)$ взамен знака возвращает уверенность.



Если строить несколько линейных моделей?



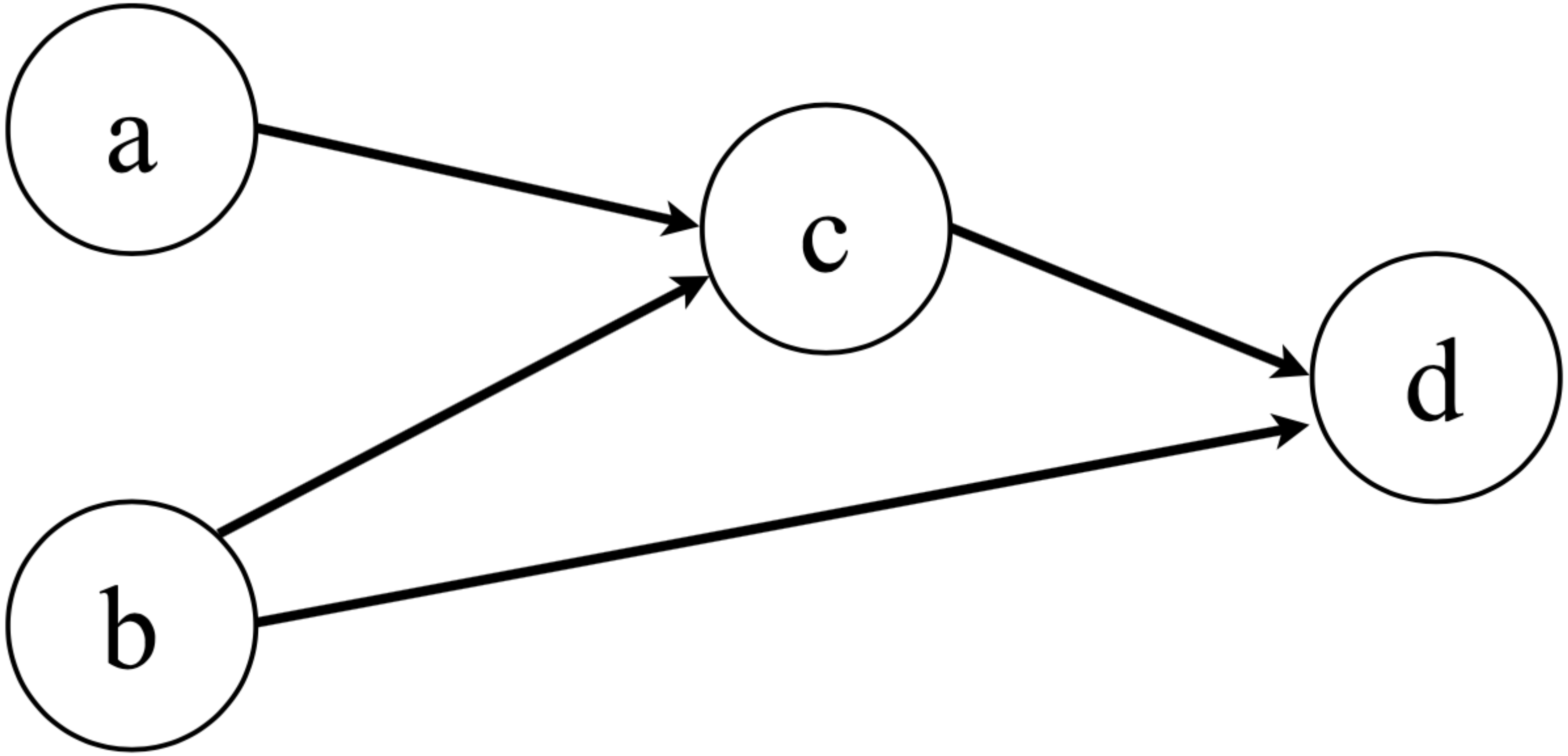
Построим три линейные модели, отвечающие за разделение в разных областях

На их предсказаниях построим финальную линейную модель

Осталось понять как…

Композиции в виде графов

$$d(c(a, b), b)$$



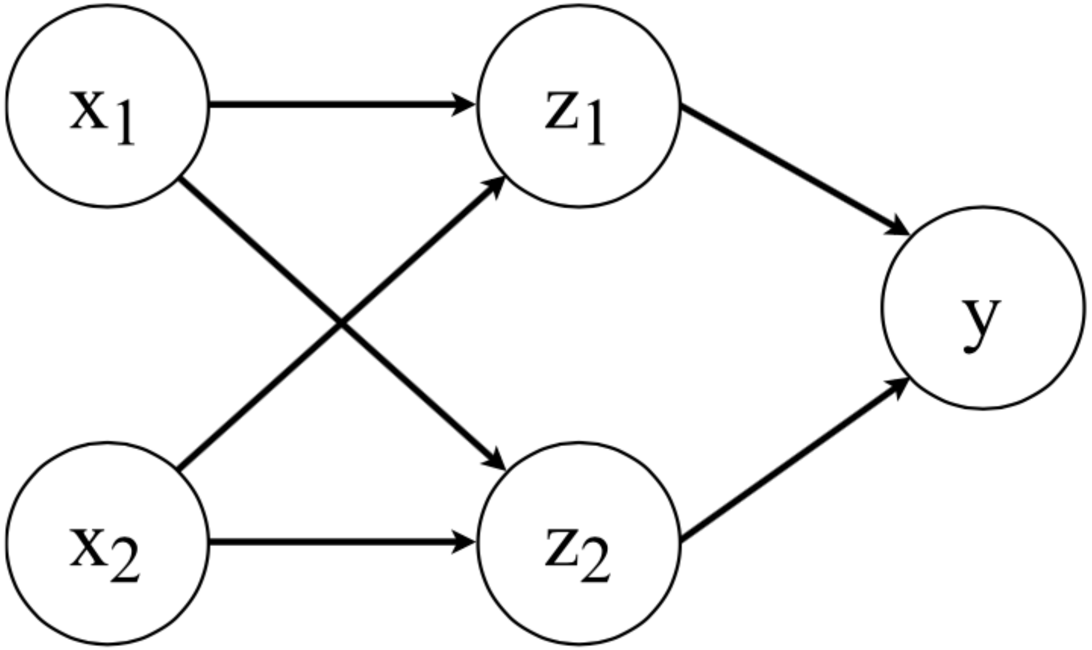
Композиции линейных функций

Идея: ставить в узел линейную функцию.

$$f(x_1, \ldots, x_k) = \sum_{i=1}^{k} w_i x_i + w_0$$

(в каждом узле свой набор w_i)

Композиции линейных функций



$$z_1 = w_{11}x_1 + w_{12}x_2$$

$$z_2 = w_{21}x_1 + w_{22}x_2$$

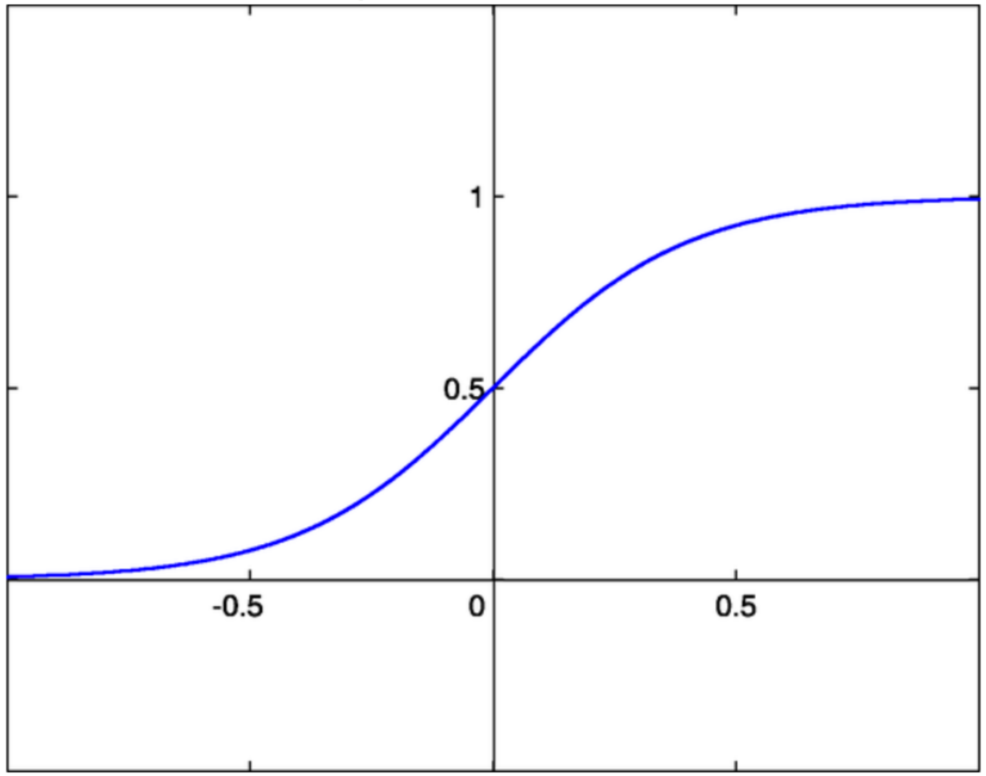
$$y = w_{31}z_1 + w_{32}z_2$$

Проблема: сохраняется линейность

$$y = (w_{31}w_{11} + w_{32}w_{21})x_1 + (w_{31}w_{12} \quad + w_{32}w_{22})x_2$$

Нужны нелинейности!

$h : \mathbb{R} \to \mathbb{R}$ — некоторая нелинейная функция



Пример: сигмоида

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

В каждом узле:

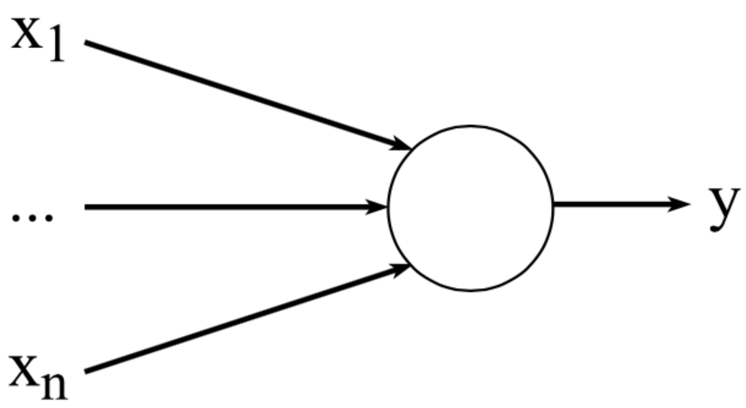
$$f(x_1, \ldots, x_k) = h\left(\sum_{i=1}^{k} w_i x_i + w_0\right)$$

Нейрон

$$y = h\left(\sum_{i=1}^{k} w_i x_i + w_0\right)$$

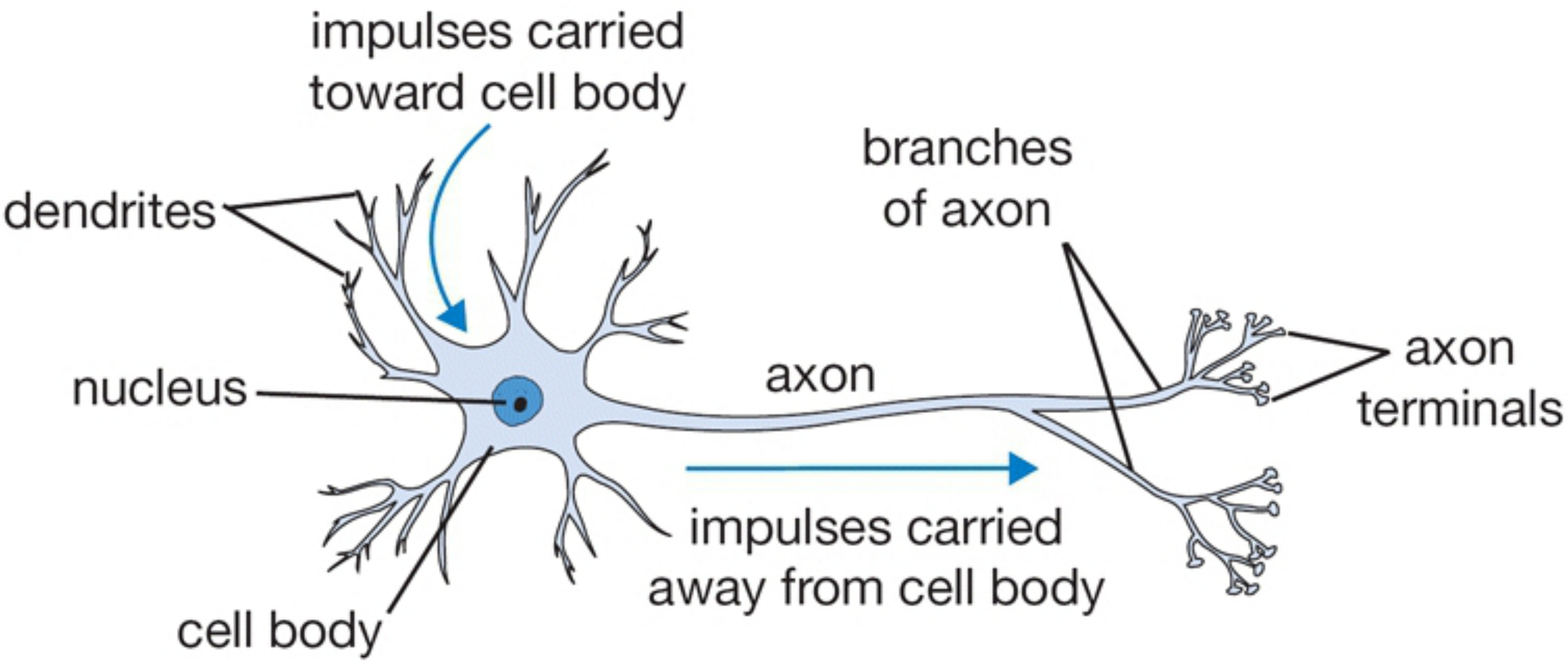
$w_0, w_1, \ldots, w_k$ — параметры нейрона

$h(\cdot)$ — гиперпараметр нейрона



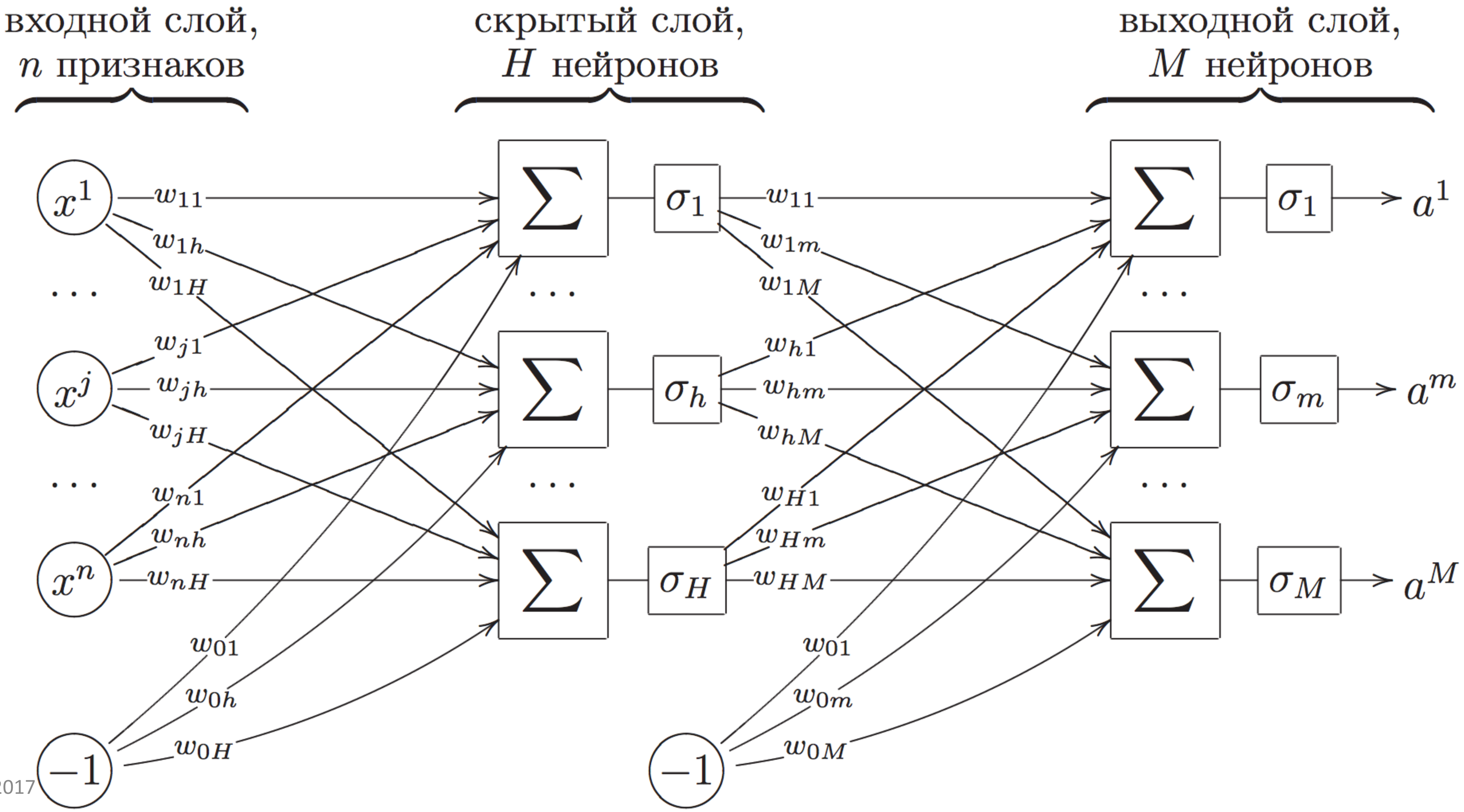
Нейронная сеть = сеть (композиция) нейронов

Почему такое название



Упрощенное устройство нейрона человека

Как выглядит нейронная сеть



Обучение нейронной сети

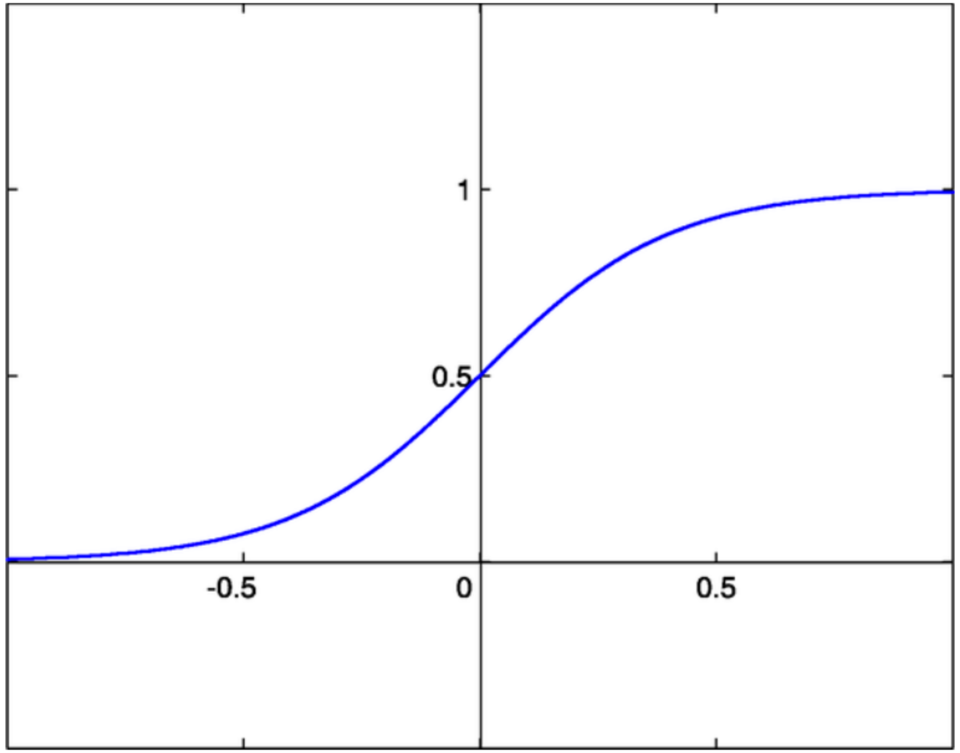
- $\{x_i, y_i\}_{i=1}^{N}$ — обучающая выборка
- $a(x; w)$ — модель
  - x — входной объект
  - w — параметры (веса входов нейронов)
  - архитектура сети — гиперпараметр
- $L(y, p)$ — функция потерь

$$Q(w) = \sum_{i=1}^{N} L(y_i, a(x_i; w)) \to \min_{w}$$

Оптимизация градиентным спуском: $\frac{\partial Q}{\partial w}$

Производная сигмоиды

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



$$\sigma'(z) = \sigma(z)(1 - \sigma(z))$$

Пример: один нейрон

$$a(x; w) = \sigma\left(\sum_{i=0}^{d} w_i x^i\right) = \sigma(w^T x)$$

$$L(y, p) = (y - p)^2$$

$$Q(w) = \sum_{i=1}^{N} L(y_i, a(x_i; w)) \qquad \frac{\partial Q(w)}{\partial w} = \sum_{i=1}^{N} \frac{\partial \big(y_i - \sigma(w^T x_i)\big)^2}{\partial w} = ?$$

Производная композиции функций:

$$\frac{\partial f(g(x))}{\partial x} = \frac{\partial f}{\partial g}\Big(g(x)\Big)\frac{\partial g}{\partial x}(x)$$

Производная сложной функции (chain rule)

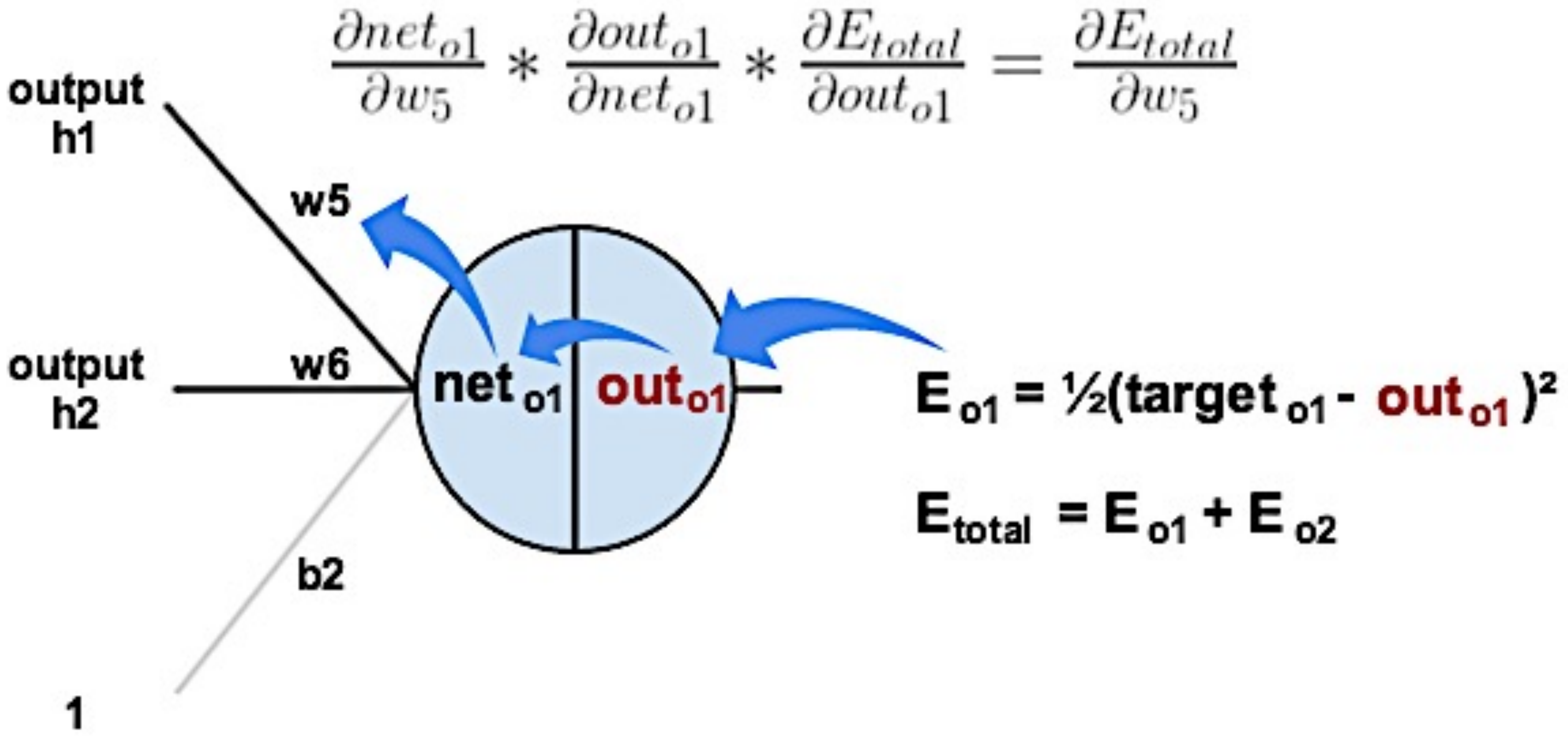
$$\frac{\partial Q(w)}{\partial w} = \sum_{i=1}^{N} \frac{\partial \big(y_i - \sigma(w^T x_i)\big)^2}{\partial w} =$$

$$= \sum_{i=1}^{N} \boxed{-2(y_i - \sigma(w^T x_i))} \frac{\partial \sigma(w^T x_i)}{\partial w} =$$

$$= \sum_{i=1}^{N} \boxed{-2(y_i - \sigma(w^T x_i))} \boxed{\sigma(w^T x_i)(1 - \sigma(w^T x_i))} \frac{\partial (w^T x_i)}{\partial w} =$$

$$= \sum_{i=1}^{N} \boxed{-2(y_i - \sigma(w^T x_i))} \boxed{\sigma(w^T x_i)(1 - \sigma(w^T x_i))} x_i$$

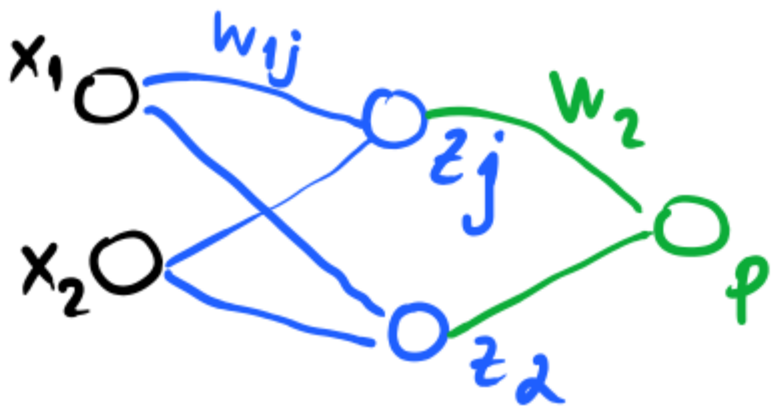
То же самое визуально



Если добавить скрытый слой

$$z_j(x) = \sigma(w_{1,j}^T x), \qquad p(z) = \sigma(w_2^T z)$$

$$L(y, p) = (y - p)^2$$



$$\frac{\partial Q}{\partial w_2} = \sum_{i=1}^{N} \boxed{\frac{\partial L}{\partial p}} \frac{\partial p}{\partial w_2}$$

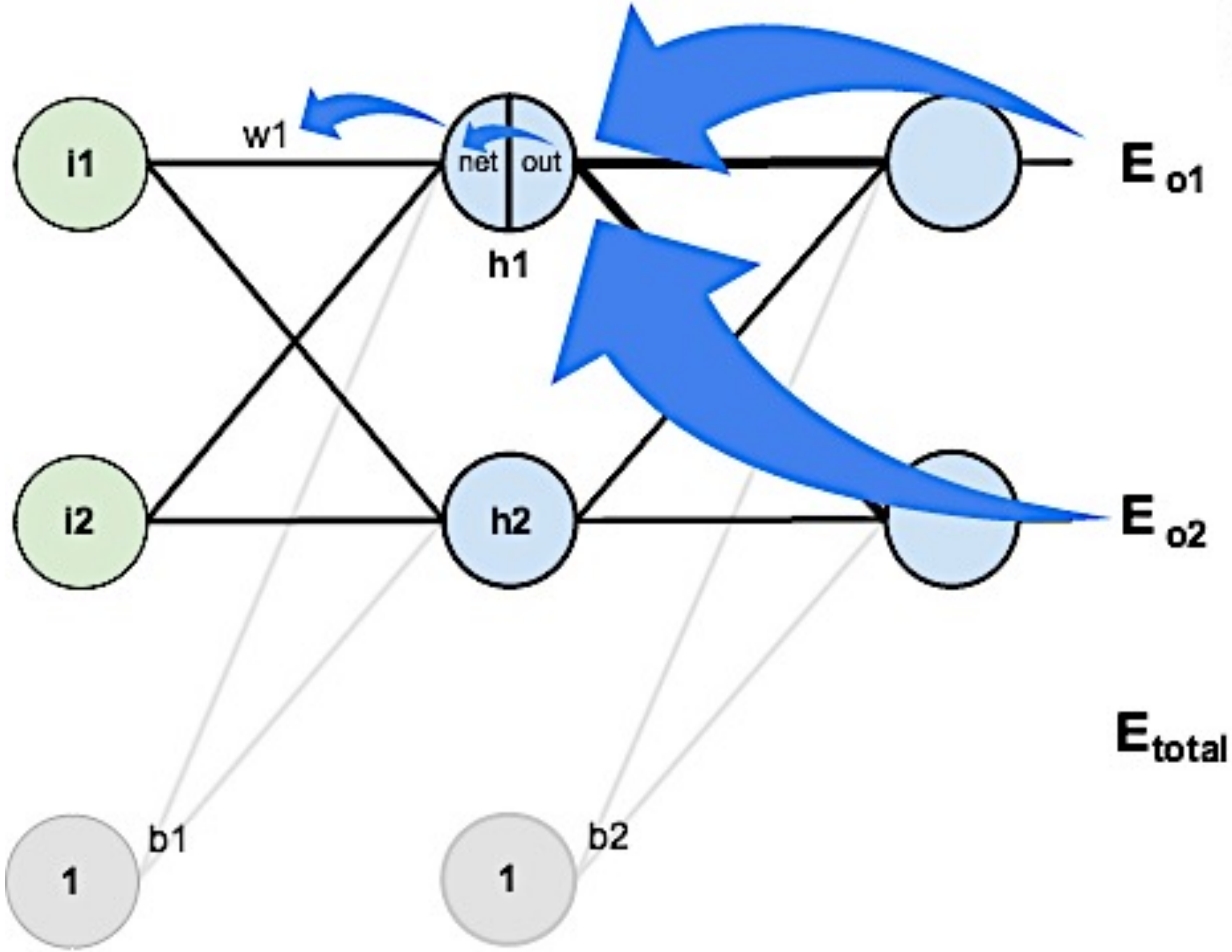
Это уже умеем считать

$$\frac{\partial Q}{\partial w_{1,j}} = \sum_{i=1}^{N} \boxed{\frac{\partial L}{\partial p}} \frac{\partial p}{\partial w_{1,j}} = \sum_{i=1}^{N} \boxed{\frac{\partial L}{\partial p}} \sum_{s=1}^{J} \frac{\partial p}{\partial z_s} \frac{\partial z_s}{\partial w_{1,j}} =$$

Только одно не ноль

$$= \sum_{i=1}^{N} \boxed{\frac{\partial L}{\partial p}} \frac{\partial p}{\partial z_j} \frac{\partial z_j}{\partial w_{1,j}}$$

Похожий пример визуально

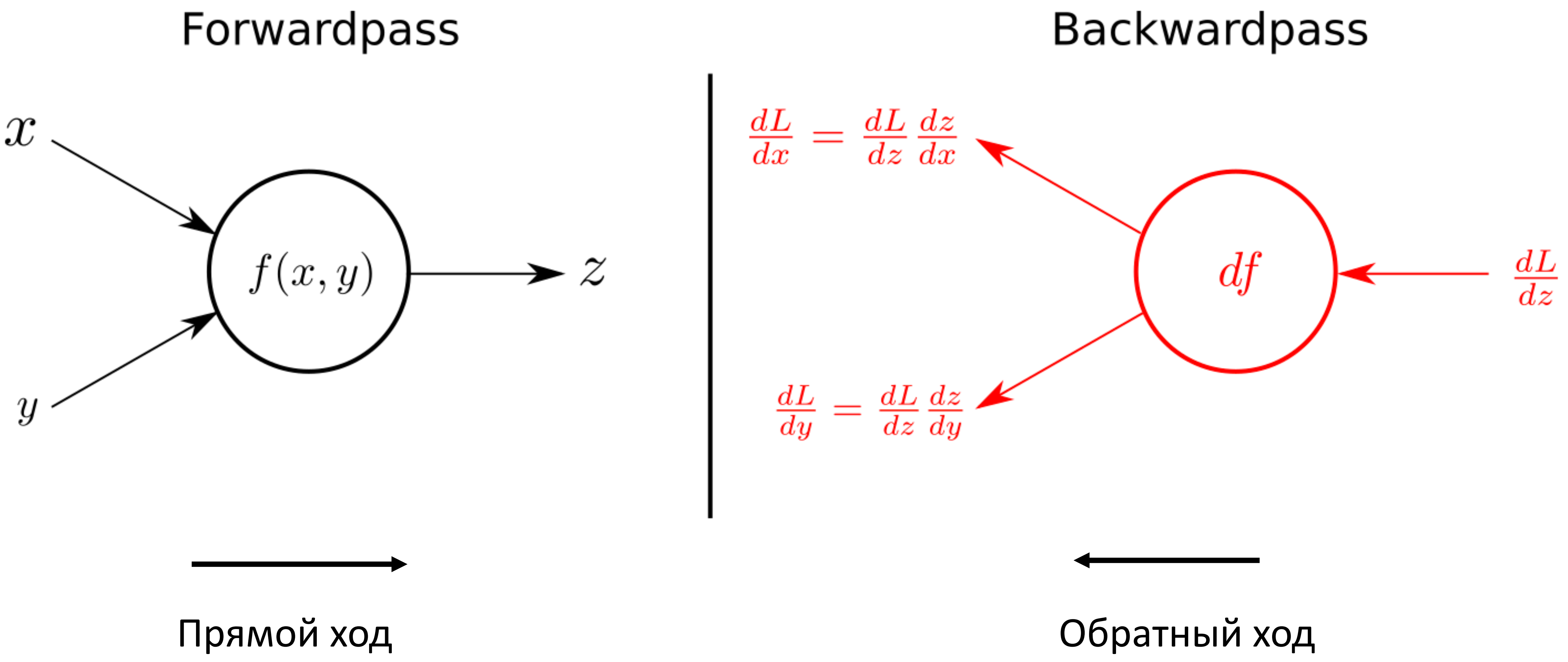


$$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial out_{h1}} * \frac{\partial out_{h1}}{\partial net_{h1}} * \frac{\partial net_{h1}}{\partial w_1}$$

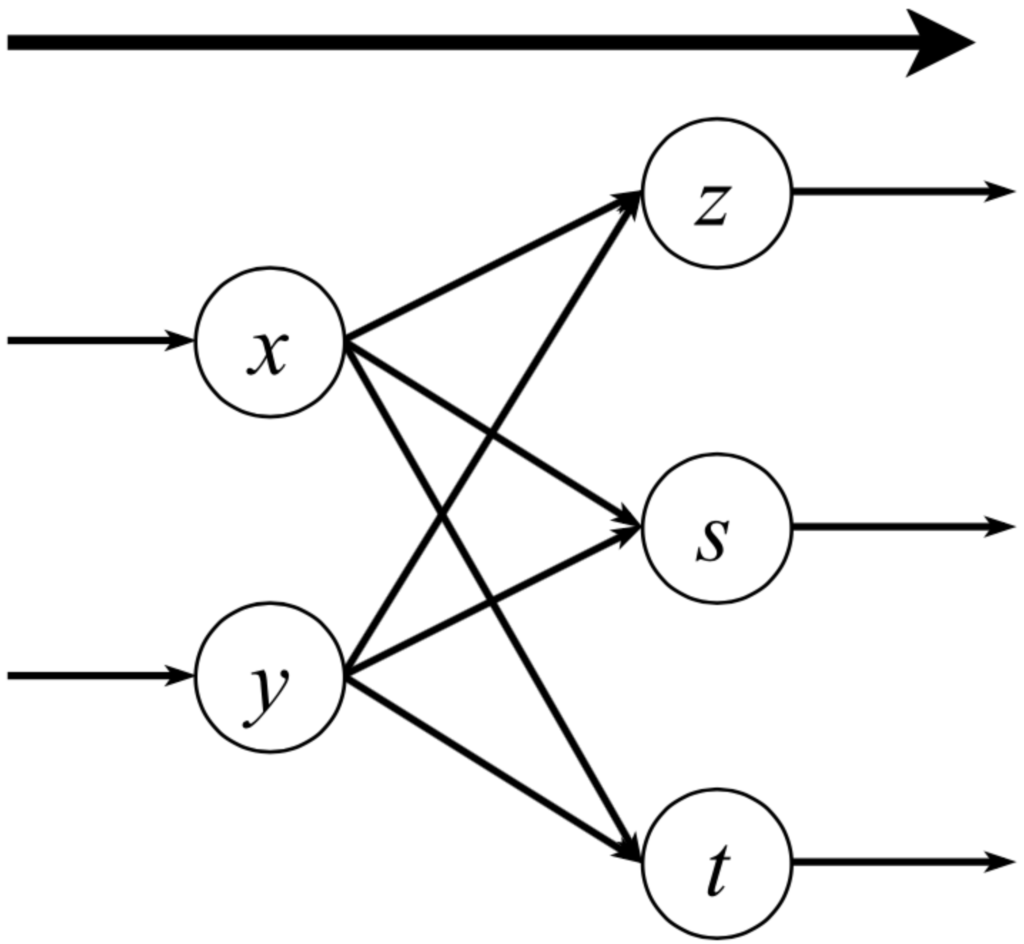
$$\frac{\partial E_{total}}{\partial out_{h1}} = \frac{\partial E_{o1}}{\partial out_{h1}} + \frac{\partial E_{o2}}{\partial out_{h1}}$$

$$E_{total} = E_{o1} + E_{o2}$$

Обратное распространение ошибки (back-prop)

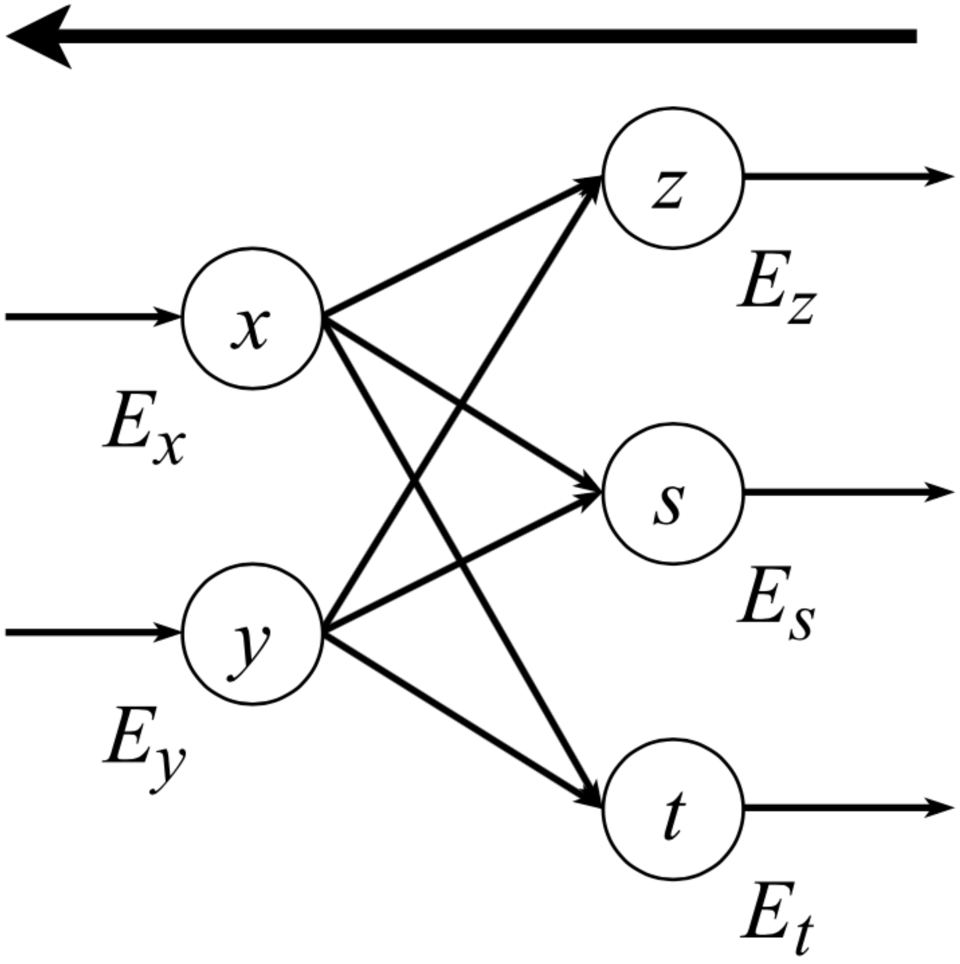


Прямой ход



$$z = h(w_{z,x}x + w_{z,y}y + w_{z,0})$$

Обратный ход



$$E_\alpha = \frac{\partial Q}{\partial \alpha}$$

$$\alpha = x, y, z, s, t, ..$$

$$\frac{\partial Q}{\partial w_\alpha} = E_\alpha \frac{\partial \alpha}{\partial w_\alpha}$$

$$E_x = \frac{\partial Q}{\partial x} = \frac{\partial Q}{\partial z}\frac{\partial z}{\partial x} + \frac{\partial Q}{\partial s}\frac{\partial s}{\partial x} + \frac{\partial Q}{\partial t}\frac{\partial t}{\partial x} = E_z\frac{\partial z}{\partial x} + E_s\frac{\partial s}{\partial x} + E_t\frac{\partial t}{\partial x}$$

Обратное распространение ошибки (back-prop)

- $a(x; w)$ — сеть

  (композиция дифференцируемых функций)

- $L(y, p)$ — функция потерь

  (дифференцируемая функция)

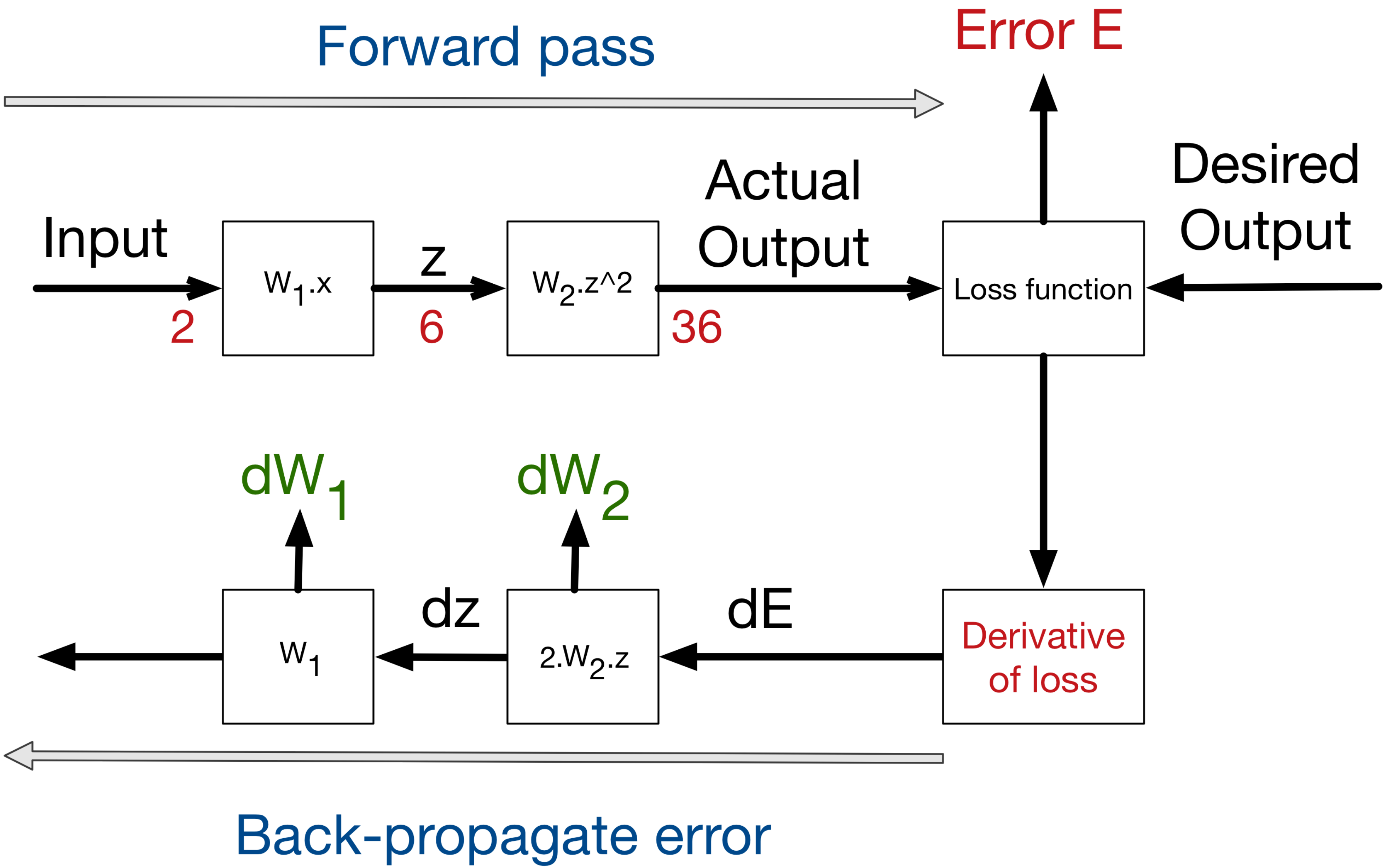
- $Q(w) = \sum_{i=1}^{N} L(y_i, a(x_i; w))$ — потери на выборке

  (композиция дифференцируемых функций)

$$\frac{\partial Q(\alpha_1, \alpha_2, \ldots)}{\partial w_j} = \sum_s \frac{\partial Q}{\alpha_s} \frac{\partial \alpha_s(\beta_1, \beta_2, \ldots)}{\partial w_j} =$$

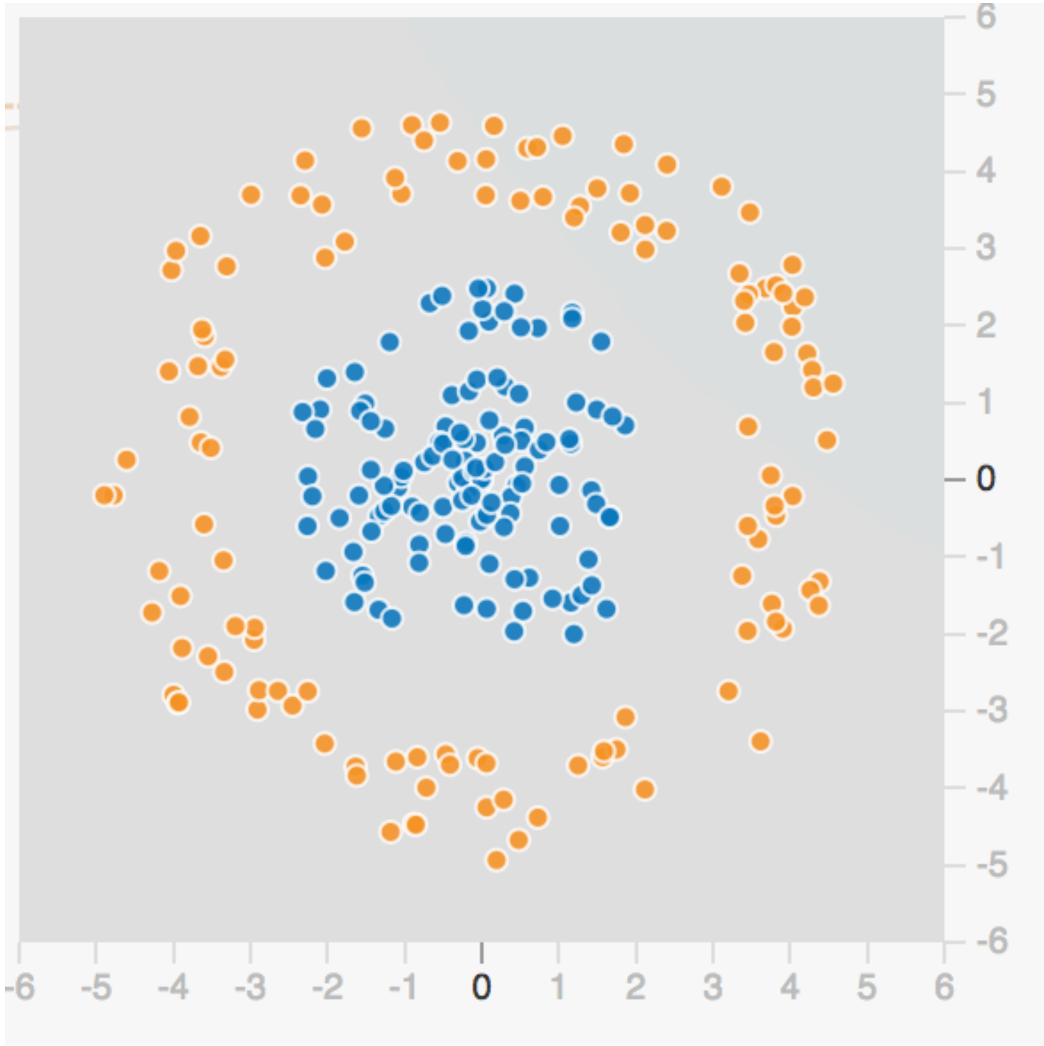
$$= \sum_s \frac{\partial Q}{\alpha_s} \sum_t \frac{\partial \alpha_s}{\partial \beta_t} \frac{\partial \beta_t(\gamma_1, \gamma_2, \ldots)}{\partial w_j} = \ldots$$

Обратные шаги это тоже граф

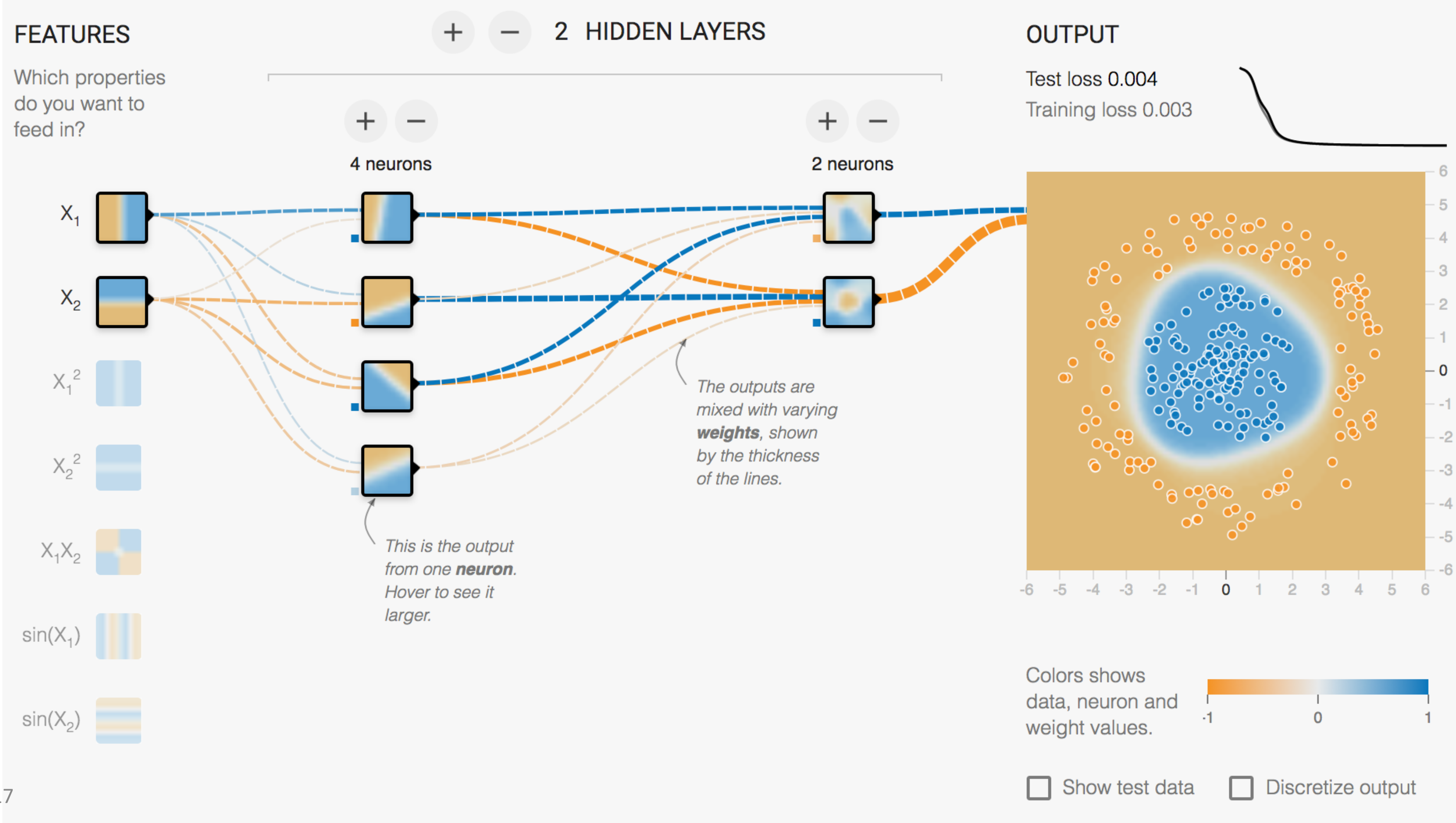


Демо: нейросети в TensorFlow Playground

- http://playground.tensorflow.org



Демо: нейросети в TensorFlow Playground



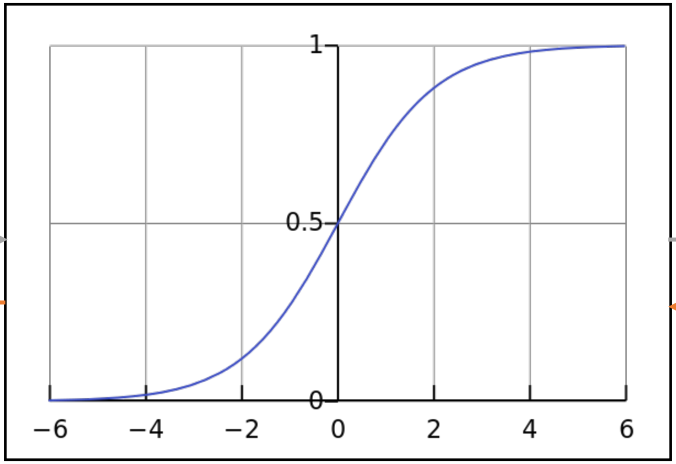
Резюме

- **Плюсы:**
  - Универсальные аппроксиматоры (приближают сложные функции)
  - Сложные композиции простых функций (легко дифференцировать)

- **Минусы:**
  - Архитектуру надо подбирать руками
  - Сильное переобучение (нужна регуляризация)
  - Проблемы с затухающими или взрывающимися градиентами

Sigmoid активация

x



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

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial \sigma}\frac{\partial \sigma}{\partial x}$$

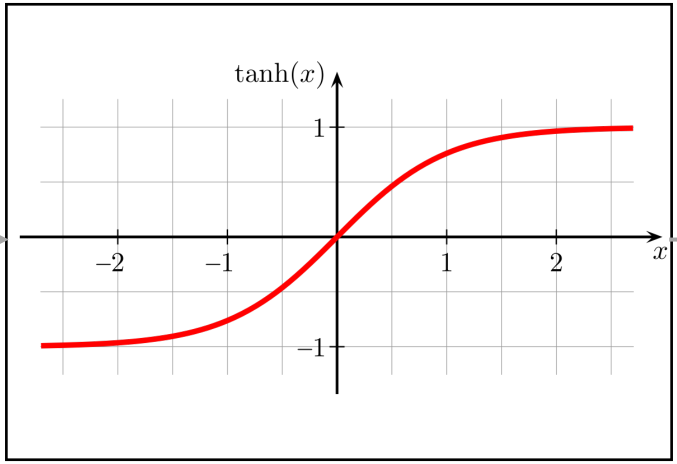
$$\frac{\partial L}{\partial \sigma}$$

$$\frac{\partial \sigma}{\partial x} = \sigma(x)(1 - \sigma(x))$$

- Нейроны с сигмоидой могут насыщаться и приводить к **угасающим градиентам**.
- Не центрированы в нуле.
- e^x дорого вычислять.

Tanh активация

x

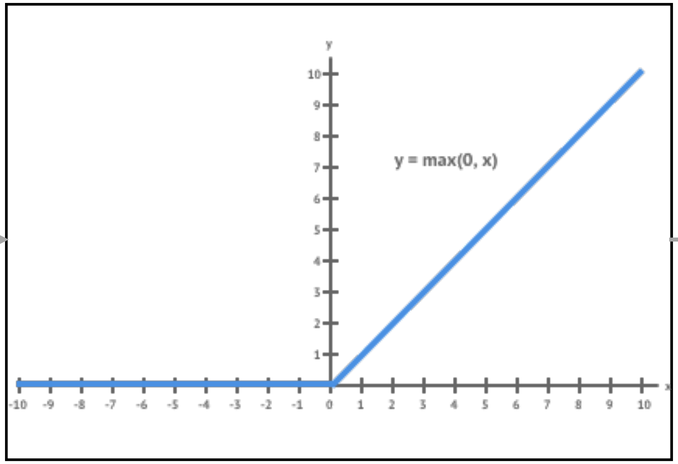


$$\tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

- Центрирован в нуле.
- Но все еще как сигмоида.

ReLU активация

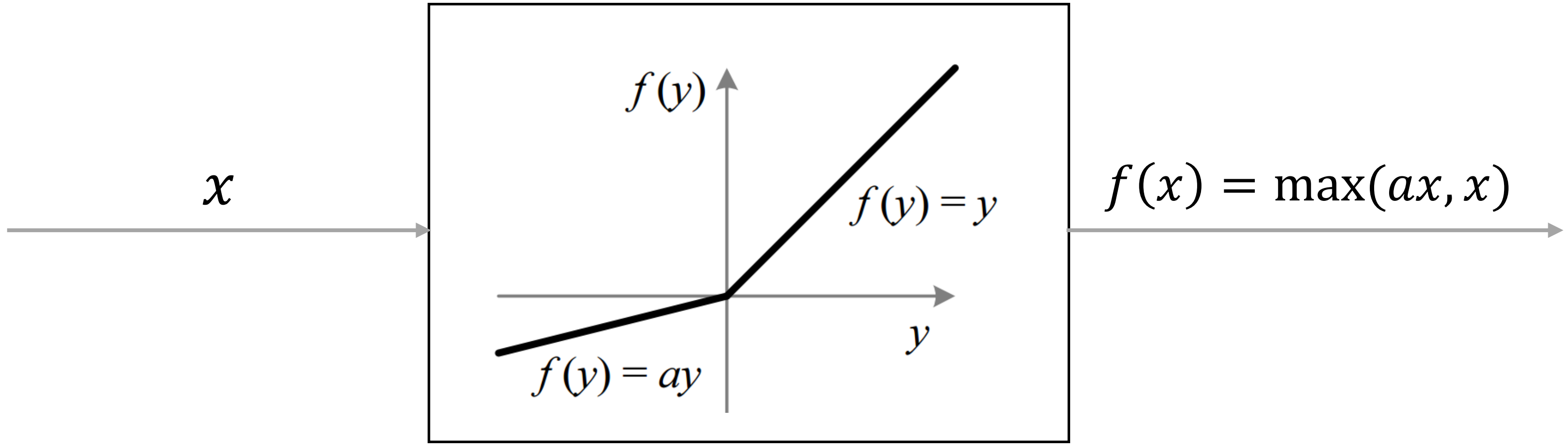
x



$f(x) = \max(0, x)$

- Быстро считается.
- Градиенты не угасают при $x > 0$.
- На практике ускоряет сходимость!
- Не центрирован в нуле.
- Могут умереть: если не было активации - не будет обновления!

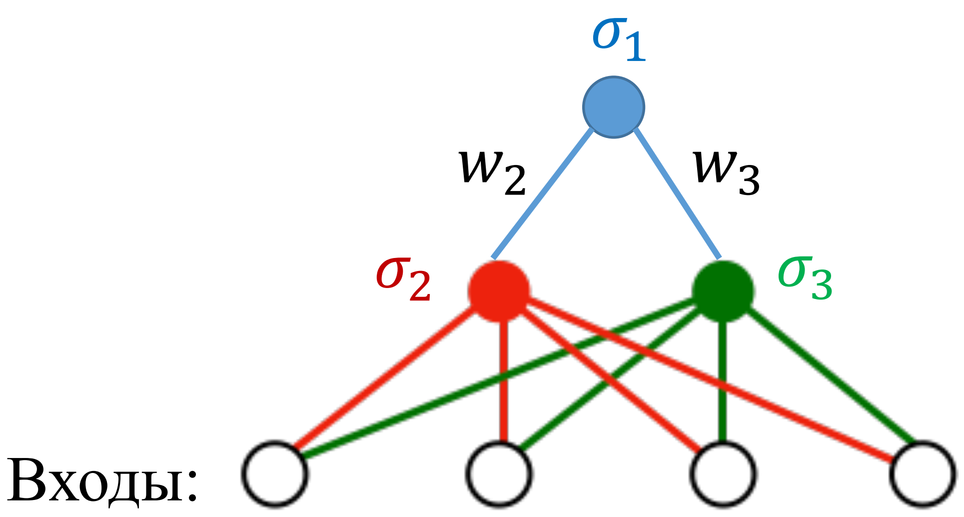
Leaky ReLU активация



- Всегда будут обновления!
- $a \neq 1$

Инициализация весов

Давайте начнем с нулей?



$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \sigma_1}\sigma_1(1-\sigma_1)\sigma_2$$

$$\frac{\partial L}{\partial w_3} = \frac{\partial L}{\partial \sigma_1}\sigma_1(1-\sigma_1)\sigma_3$$

σ_2 и σ_3 обновляются одинаково!

- Нужно сломать симметрию!
- Может случайным шумом?
- Но насколько большим? $0.03 \cdot \mathcal{N}(0,1)$?

Weights initializations

- Linear models work best when inputs are normalized.
- Neuron is a linear combination of inputs + activation.
- Neuron output will be used by consecutive layers.

Weights initializations

- Let's look at the neuron output before activation: $\sum_{i=1}^{n} x_i w_i$.
- If $E(x_i) = E(w_i) = 0$ and we generate weights independently from inputs, then $E(\sum_{i=1}^{n} x_i w_i) = 0$.
- But variance can grow with consecutive layers.
- Empirically this hurts convergence for deep networks!

Weights initializations

- Let's look at the variance of $\sum_{i=1}^{n} x_i w_i$:

Weights initializations

- Let's look at the variance of $\sum_{i=1}^{n} x_i w_i$:

$$Var(\sum_{i=1}^{n} x_i w_i) =$$

i.i.d. w_i and mostly uncorrelated x_i

$$= \sum_{i=1}^{n} Var(x_i w_i) =$$

Weights initializations

- Let's look at the variance of $\sum_{i=1}^{n} x_i w_i$:

$$Var\left(\sum_{i=1}^{n} x_i w_i\right) =$$

i.i.d. w_i and mostly uncorrelated x_i

$$= \sum_{i=1}^{n} Var(x_i w_i) =$$

independent factors w_i and x_i

$$= \sum_{i=1}^{n} \begin{pmatrix} [E(x_i)]^2 Var(w_i) \\ +[E(w_i)]^2 Var(x_i) \\ +Var(x_i) Var(w_i) \end{pmatrix} =$$

Weights initializations

- Let's look at the variance of $\sum_{i=1}^{n} x_i w_i$:

$$Var(\sum_{i=1}^{n} x_i w_i) =$$

i.i.d. w_i and mostly uncorrelated x_i

$$= \sum_{i=1}^{n} Var(x_i w_i) =$$

independent factors w_i and x_i

$$= \sum_{i=1}^{n} \begin{pmatrix} [E(x_i)]^2 Var(w_i) \\ +[E(w_i)]^2 Var(x_i) \\ +Var(x_i) Var(w_i) \end{pmatrix} =$$

w_i and x_i have 0 mean

$$= \sum_{i=1}^{n} Var(x_i) Var(w_i) = Var(x)[n\, Var(w)]$$

Weights initializations

- Let's look at the variance of $\sum_{i=1}^{n} x_i w_i$:

$$Var(\sum_{i=1}^{n} x_i w_i) =$$ i.i.d. w_i and mostly uncorrelated x_i

$$= \sum_{i=1}^{n} Var(x_i w_i) =$$ independent factors w_i and x_i

$$= \sum_{i=1}^{n} \begin{pmatrix} [E(x_i)]^2 Var(w_i) \\ +[E(w_i)]^2 Var(x_i) \\ +Var(x_i) Var(w_i) \end{pmatrix} =$$ w_i and x_i have 0 mean

$$= \sum_{i=1}^{n} Var(x_i) Var(w_i) = Var(x) [n\, Var(w)]$$

We want this to be 1

Weights initializations

- Let's use the fact that $Var(aw) = a^2 Var(w)$.

- For $[n\, Var(aw)]$ to be 1
  we need to multiply $\mathcal{N}(0,1)$ weights ($Var(w) = 1$)
  by $a = 1/\sqrt{n}$.

- Xavier initialization (Glorot et al.)
  multiplies weights by $\sqrt{2}/\sqrt{n_{in} + n_{out}}$.

- Initialization for ReLU neurons (He et al.)
  uses multiplication by $\sqrt{2}/\sqrt{n_{in}}$.

Batch normalization

- We know how to initialize our network to constrain variance.
- But what if it grows during backpropagation?
- Batch normalization controls mean and variance of outputs **before activations**.

Batch normalization

- Let's normalize h_i – neuron output before activation:

$$h_i = \gamma_i \boxed{\frac{h_i - \mu_i}{\sqrt{\sigma_i^2}}} + \beta_i$$

0 mean, unit variance

Batch normalization

- Let's normalize h_i – neuron output before activation:

$$h_i = \gamma_i \boxed{\frac{h_i - \mu_i}{\sqrt{\sigma_i^2}}} + \beta_i$$

0 mean, unit variance

- Where do μ_i and σ_i^2 come from? We can estimate them having a **current training batch**!

Batch normalization

- Let's normalize h_i – neuron output before activation:

$$h_i = \gamma_i \boxed{\frac{h_i - \mu_i}{\sqrt{\sigma_i^2}}} + \beta_i$$

0 mean, unit variance

- Where do μ_i and σ_i^2 come from? We can estimate them having a **current training batch**!
- During testing we will use an exponential moving average over train batches:

$$0 < \alpha < 1 \qquad \begin{aligned} \mu_i &= \alpha \cdot \mathbf{mean_{batch}} + (1 - \alpha) \cdot \mu_i \\ \sigma_i^2 &= \alpha \cdot \mathbf{variance_{batch}} + (1 - \alpha) \cdot \sigma_i^2 \end{aligned}$$

Batch normalization

- Let's normalize h_i – neuron output before activation:

$$h_i = \gamma_i \boxed{\frac{h_i - \mu_i}{\sqrt{\sigma_i^2}}} + \beta_i$$

0 mean, unit variance

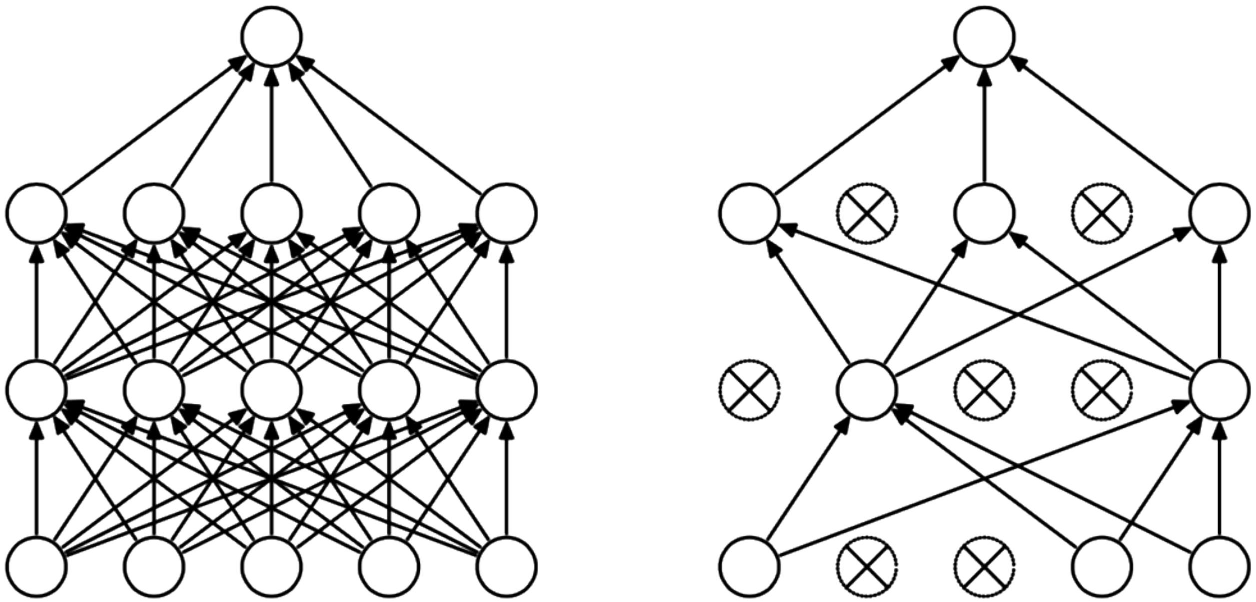
- Where do μ_i and σ_i^2 come from? We can estimate them having a **current training batch**!
- During testing we will use an exponential moving average over train batches:

$$0 < \alpha < 1 \qquad \begin{aligned} \mu_i &= \alpha \cdot \mathbf{mean_{batch}} + (1 - \alpha) \cdot \mu_i \\ \sigma_i^2 &= \alpha \cdot \mathbf{variance_{batch}} + (1 - \alpha) \cdot \sigma_i^2 \end{aligned}$$

- What about γ_i and β_i? Normalization is a differentiable operation and we can apply **backpropagation**!

Dropout

- Regularization technique to reduce overfitting.
- We keep neurons active (non-zero) with probability p.
- This way we sample the network during training and change only a subset of its parameters on every iteration.



(a) Standard Neural Net

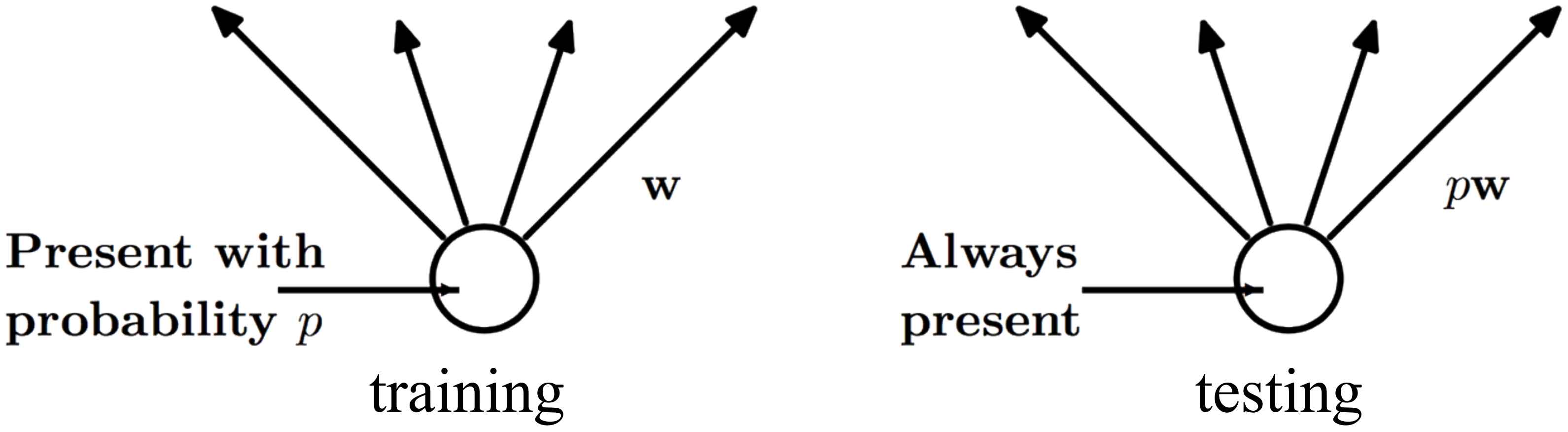
(b) After applying dropout.

http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf

Dropout

- During testing all neurons are present but their outputs are multiplied by p to maintain the scale of inputs:

Expected input weight: $p \cdot w + (1 - p) \cdot 0$



http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf

- The authors of dropout say it's similar to having an ensemble of exponentially large number of smaller networks.

Ссылки

- https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/
- http://www.machinelearning.ru/wiki/images/c/c2/Voron-ML-NeuralNets-slides.pdf