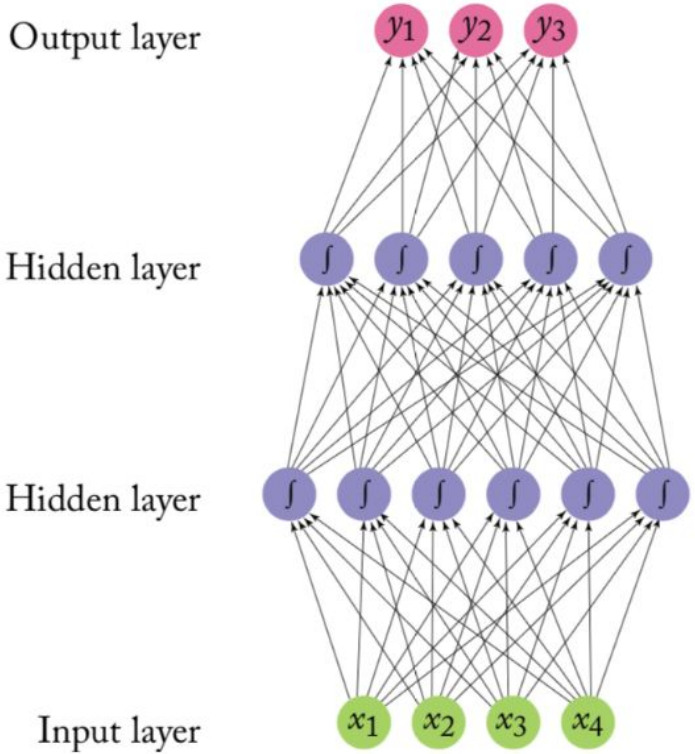
NNs and backprop

Маша Шеянова, masha.shejanova@gmail.com

Как устроена нейросеть

нейросеть in a nutshell



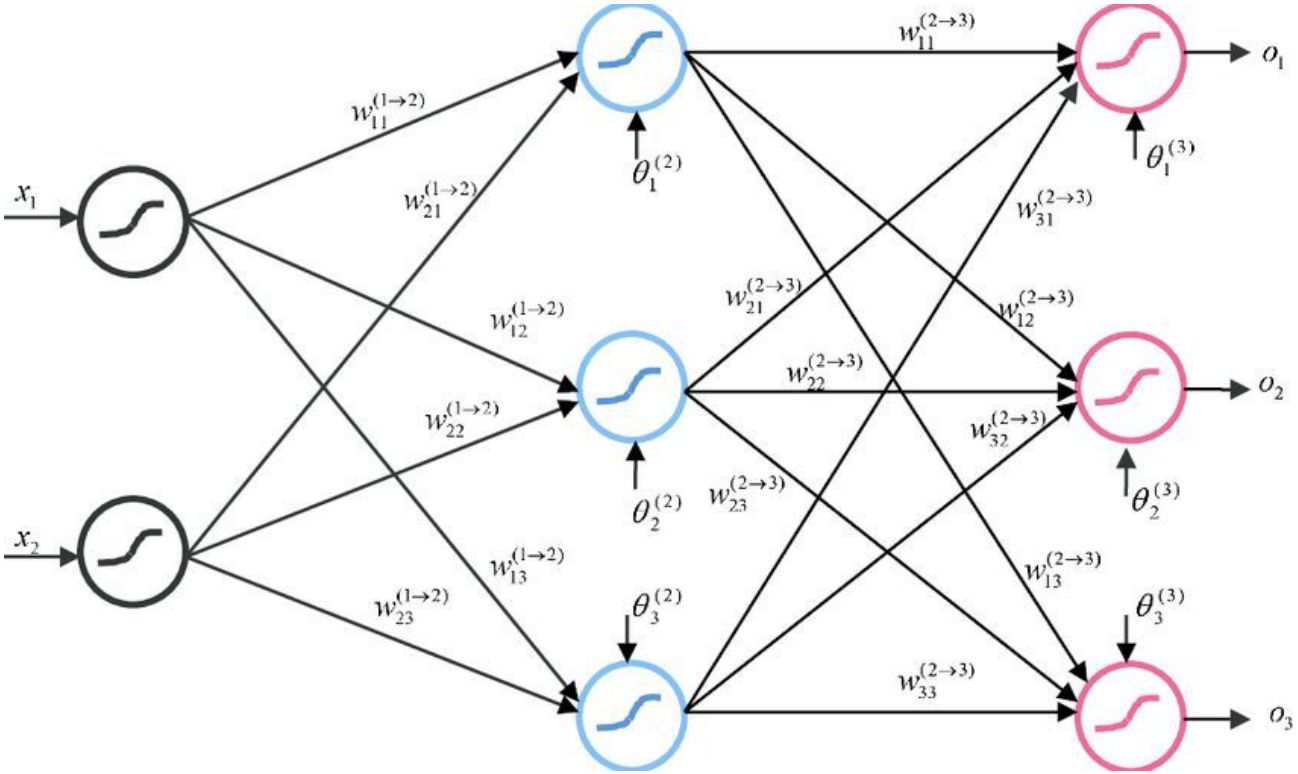
На входе — вектор признаков.

На каждой стрелочке — какие-то коэффициенты.

На выходе — вектор вероятностей того или иного класса.

“Нейрон” == один кружочек == функция от выдачи предыдущего слоя.

Нейросеть как функция

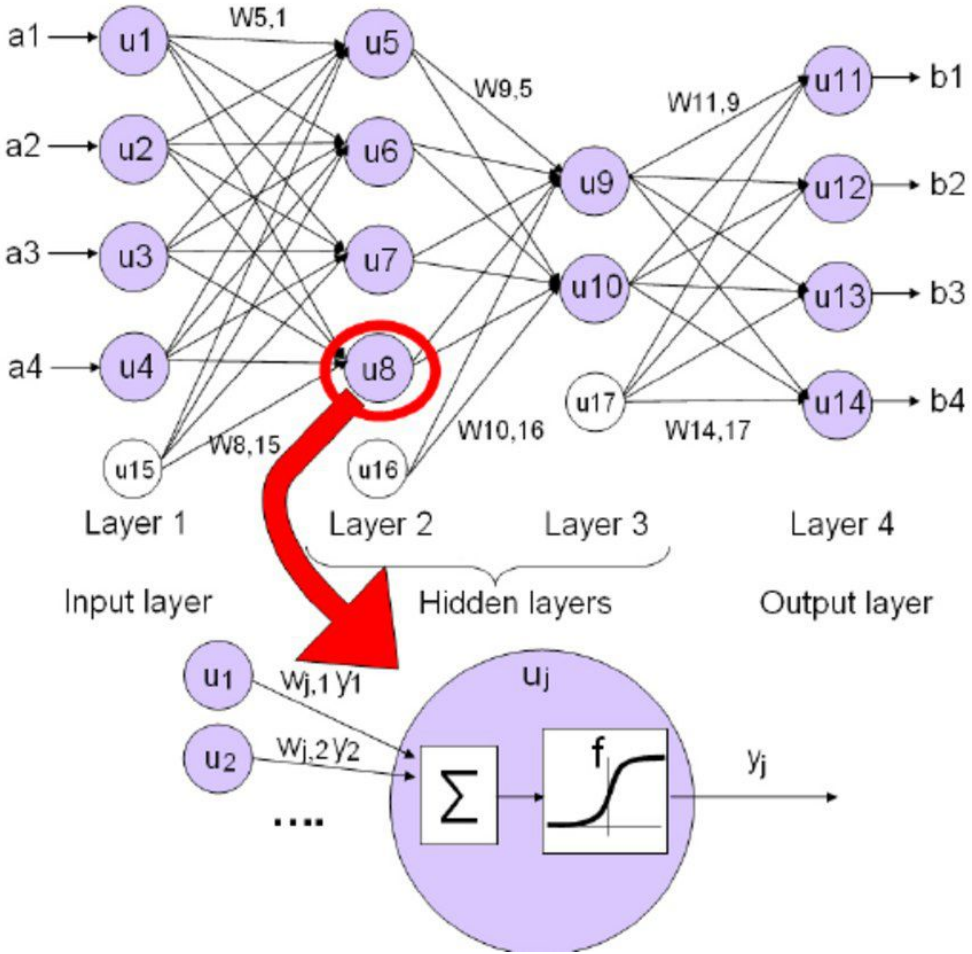


Один нейрон — функция от **вектора** параметров (w_{11}, w_{21}), умноженного на **вектор** объекта x (+ b).

*f(**wx** + b)*

Слой — функция от **матрицы** параметров * x + **вектор** b.

*f(**Wx** + **b**).*



Четырёхслойная нейросеть.

Универсальная теорема аппроксимации: любую функцию можно приблизить нейросетью.

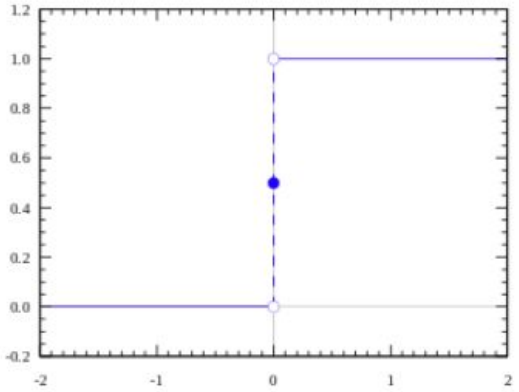
Функции активации

Почему “функция активации”?

… по аналогии с естественными нейросетями.

Справа — “step function”: нейрон активировался (1) или нет (0). (Источник картинки)

Но для artificial NN нужно что-то дифференцируемое.



Почти все картинки этого раздела взяты отсюда.

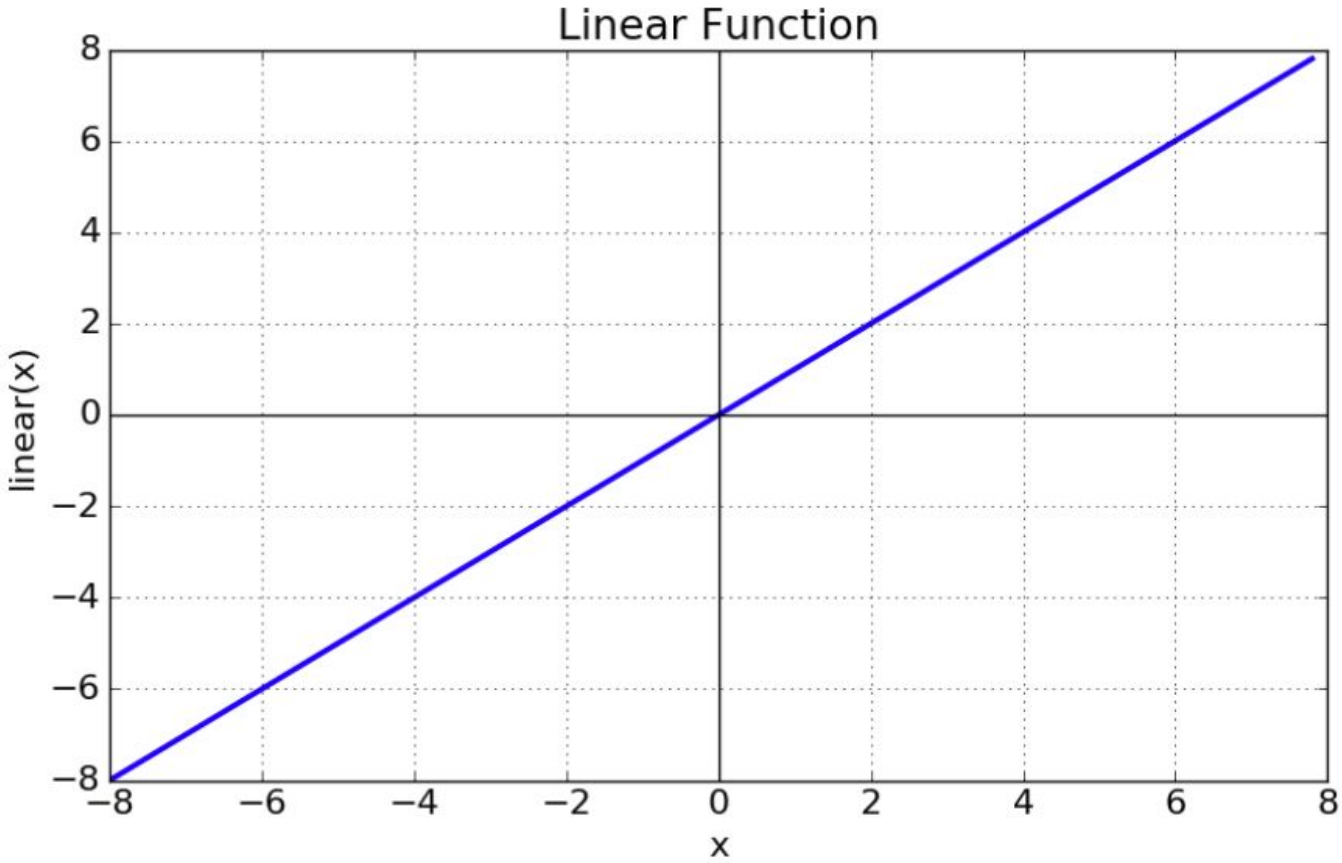
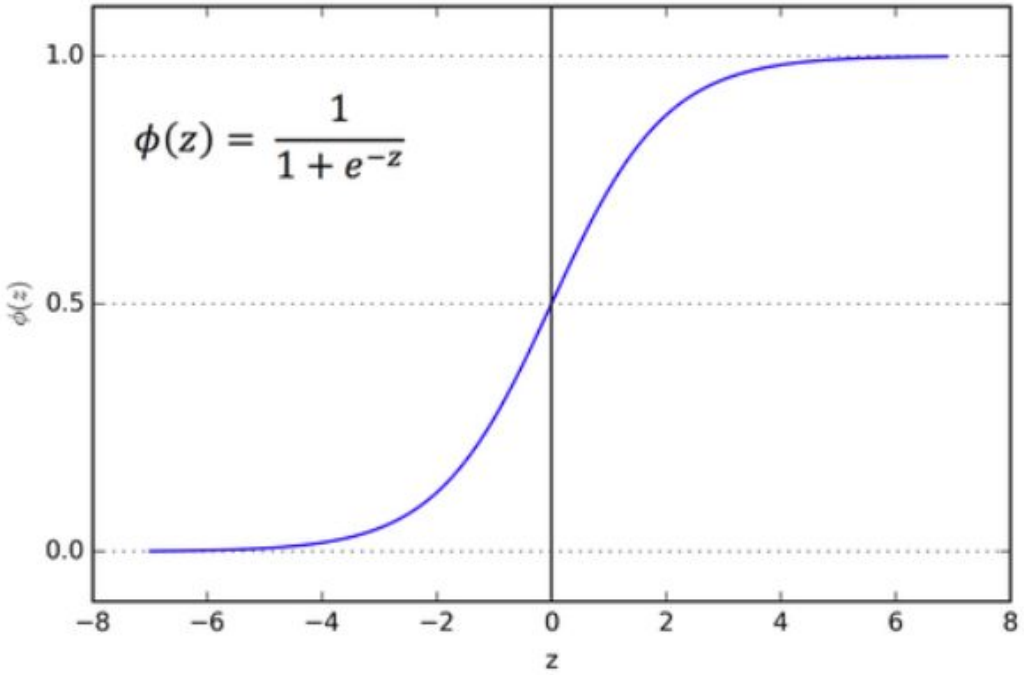


Fig: Linear Activation Function

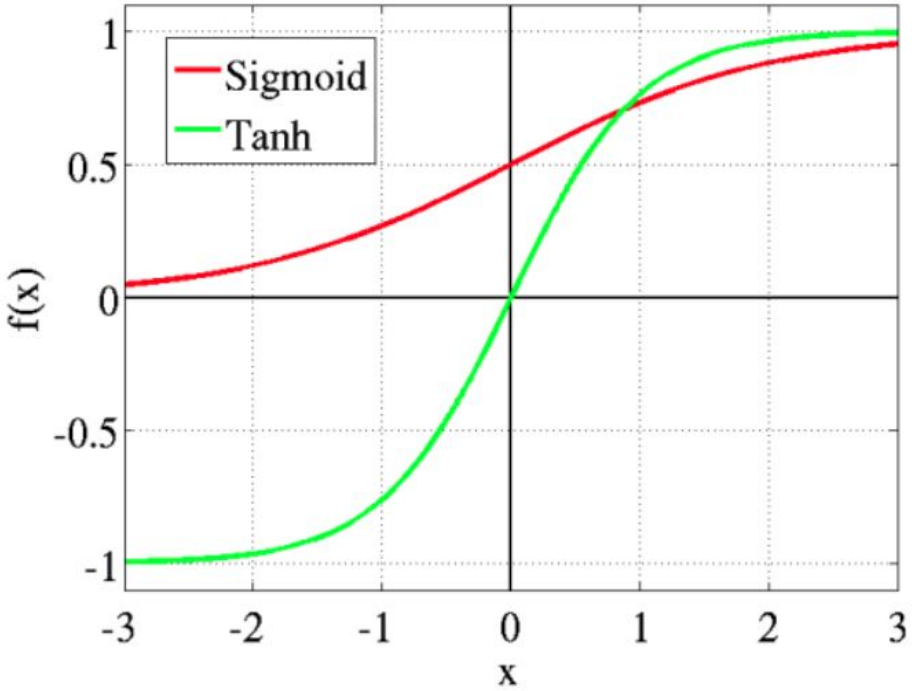
Sigmoid function



Это то же самое, что и логистическая регрессия.

Изменяется от 0 до 1 (и поэтому — хороший выбор для выдачи вероятностей).

Tanh or hyperbolic tangent Activation Function



Похожа на предыдущую, но изменяется от -1 до 1.

ReLU и Leaky ReLU

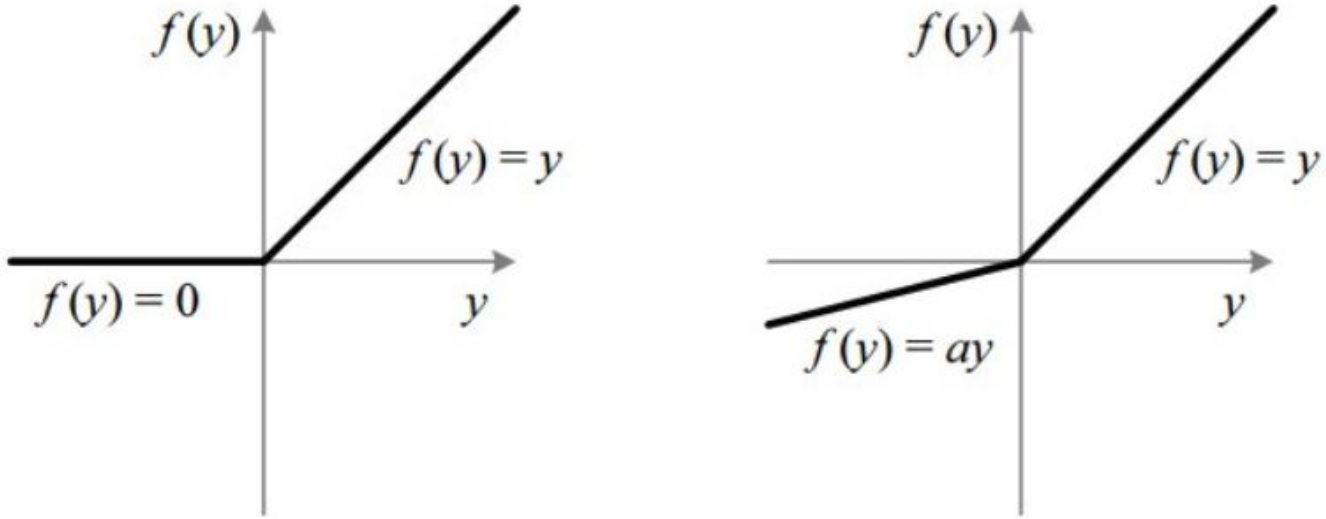


Fig : ReLU v/s Leaky ReLU

Softmax

Softmax

Формула: $$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Код (numpy):

```
logits = [2.0, 1.0, 0.1]

exps = [np.exp(i) for i in logits]

sum_of_exps = sum(exps)

softmax = [j/sum_of_exps for j in exps]
```

Loss-functions

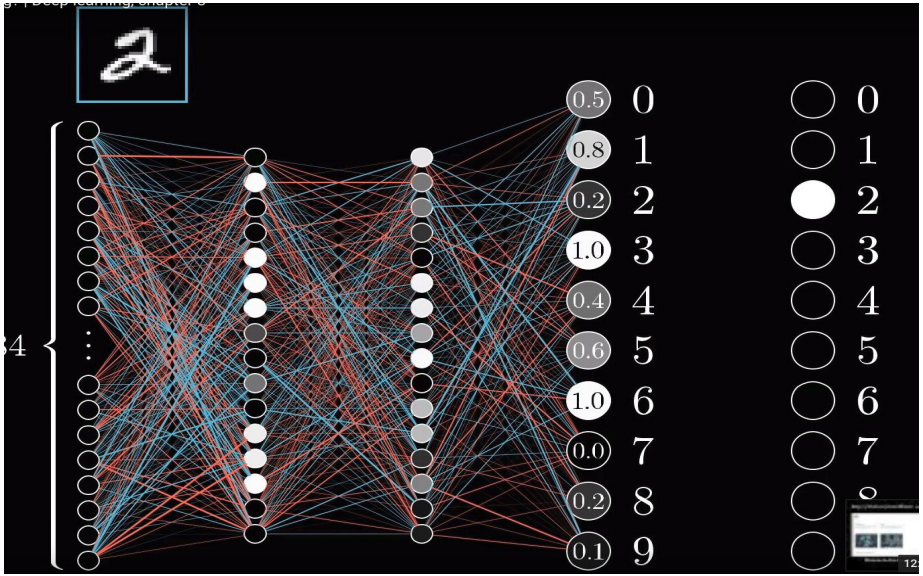
Mean squared error

Она может быть разной, **например** так:

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

MSE в виде кода:

```
L = ((y_true - y_pred) ** 2).mean()
```



Cross-entropy

В общем виде: $$H(p, q) = -\sum_{x \in \mathcal{X}} p(x) \log q(x)$$

В случае задачи классификации:

- x — объект данных
- p(x) — истинная вероятность класса
- q(x) — выдача нейросети про вероятность класса

Backpropagation

Градиентный спуск

Производная

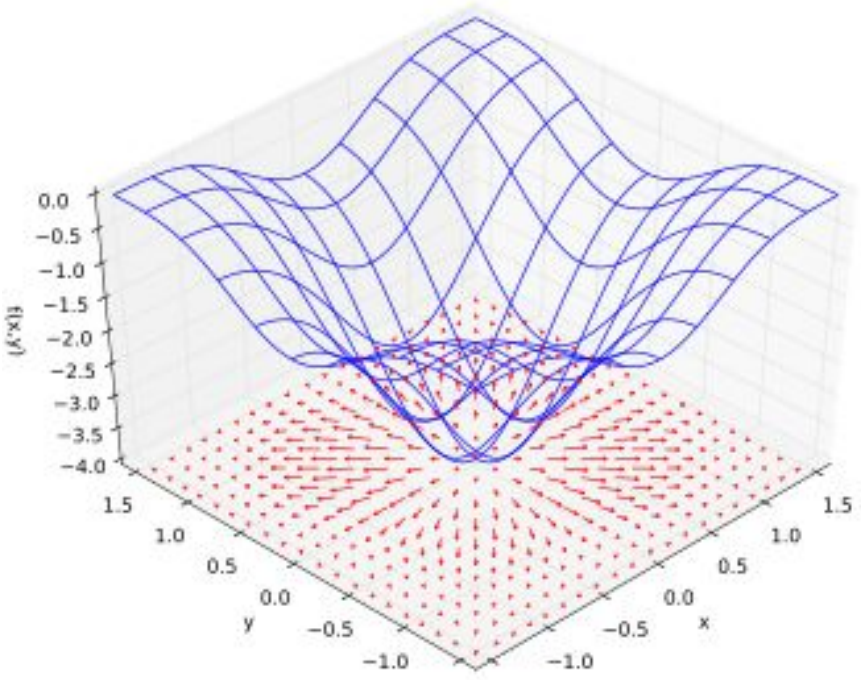
Производная — это мера, насколько быстро растёт функция.

$$f'(x_0) = \lim_{\Delta x \to 0} \frac{\Delta f}{\Delta x}$$

У функции от n переменных f(x1, x2, ... x_n) нет одной общей производной — зато есть n частные производные.

$$\frac{\partial f}{\partial x_k}(a_1, \cdots, a_n) = \lim_{\Delta x \to 0} \frac{f(a_1, \ldots, a_k + \Delta x, \ldots, a_n) - f(a_1, \ldots, a_k, \ldots, a_n)}{\Delta x}$$

Что такое градиент



Градиент — это вектор, элементы которого — значения всех возможных частных производных в конкретной точке.

Градиент соответствует вектору, указывающему направление наибольшего роста функции.

Идея

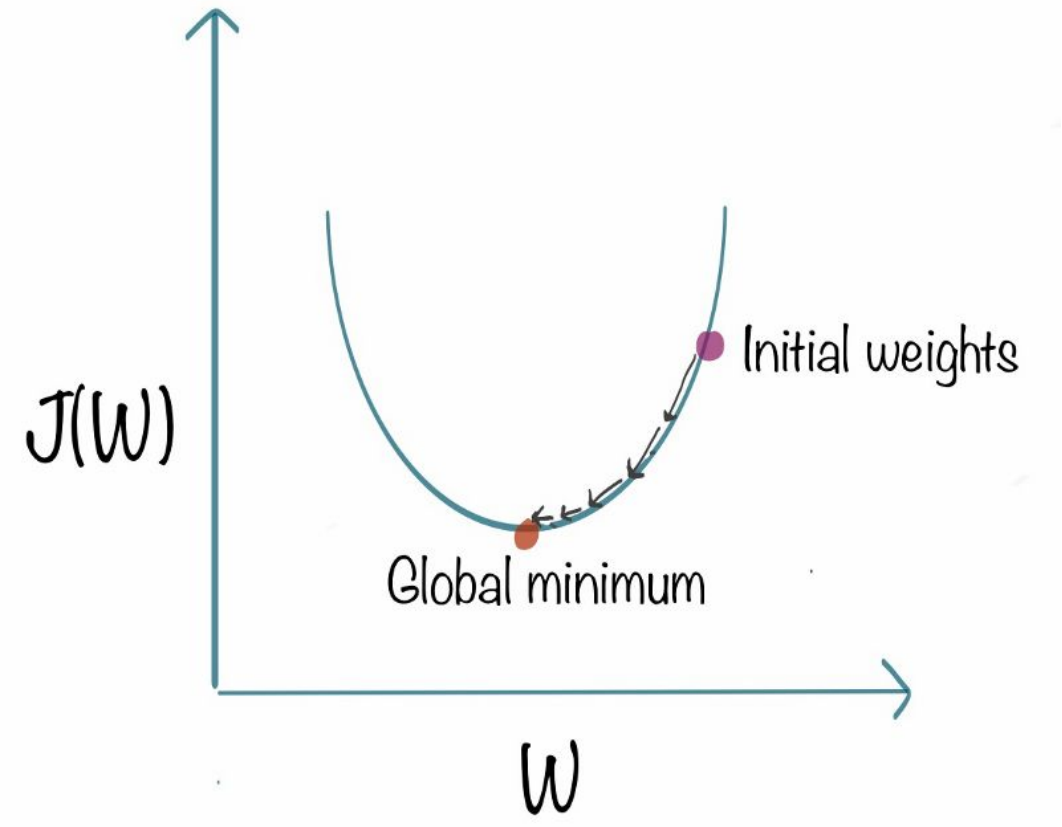
loss function = cost function = error function = функция потерь = J(W)

Её мы хотим минимизировать.

Теперь мы умеем находить, в каком направлении функция растёт быстрее всего. Но нам нужен минимум функции потерь, а не максимум!

Решение очевидно: найдём градиент и пойдём в обратную сторону.

С какой скоростью? Растёт быстро — с большой, медленно — с маленькой.



Источник картинки — очень понятно про то, как оно работает и какое бывает.

Шаги:

- подобрать случайные коэффициенты
- вычислить градиент функции потерь в этой точке
- обновить коэффициенты
- повторять, пока не сойдётся

Шаг градиентного спуска формулой

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n),\ n \geq 0.$$

x — вектор параметров (весов)

n — номер шага

гамма — learning rate

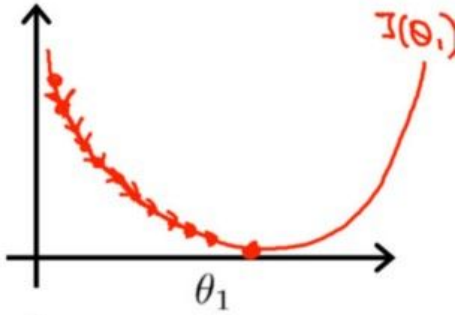
$F(x_n)$ — функция потерь, когда у нейросетки были параметры x_n (то есть обычно усреднённое значение на некотором количестве данных)

$\nabla F(x_n)$ — градиент (вектор частных производных) функции потерь в этой точке

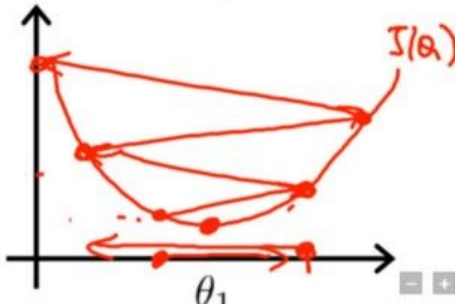
Learning rate

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

If α is too small, gradient descent can be slow.



If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.



Learning rate is a hyper-parameter that controls how much we are adjusting the weights of our network with respect the loss gradient. (отсюда)

Каким бывает градиентный спуск

- **Batch gradient descent**

  Считает градиент функции потерь с параметрами W сразу для всех обучающих данных. Работает жутко медленно.

- **Stochastic gradient descent (SGD)**

  Рандомно выбирает точку данных каждый раз

- **Mini-batch gradient descent**

  Выбираем кусочек выборки и по нему считаем

Что делать, если всё ещё ничего непонятно

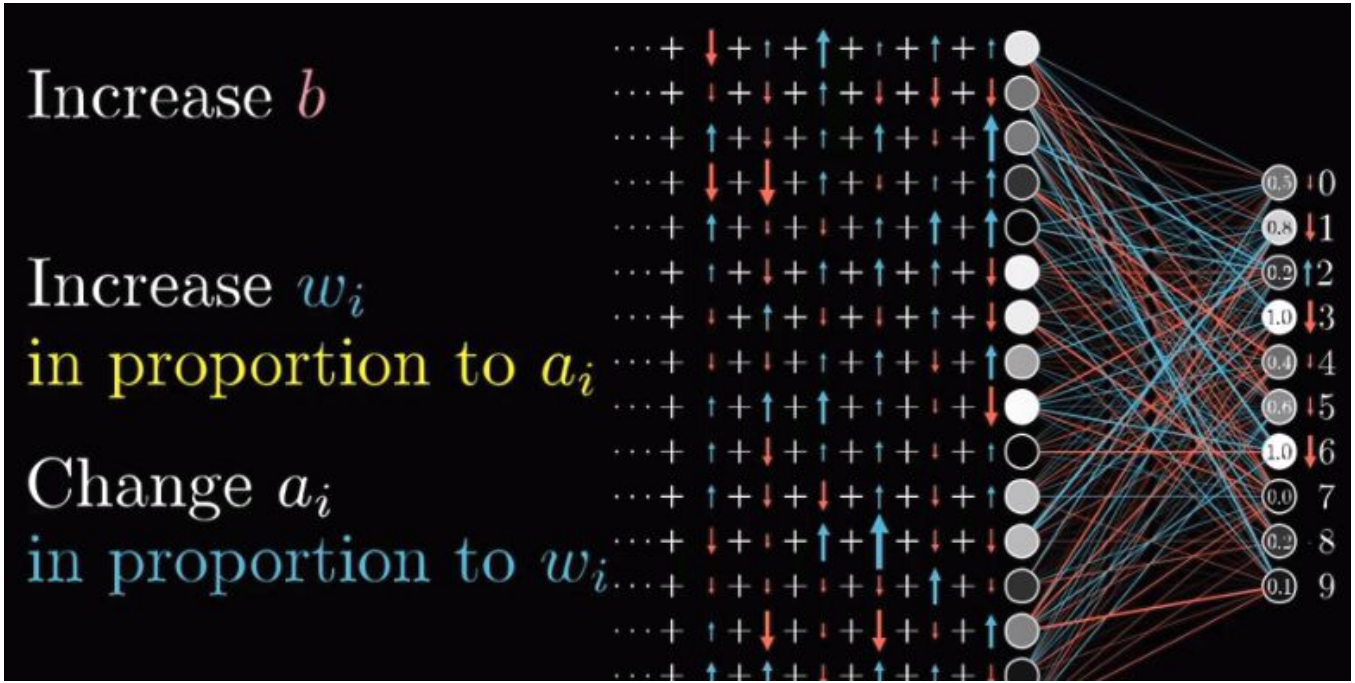
Непонимание градиентного спуска, в принципе, не помешает вам решать типичные задачи готовыми инструментами. Но может помешать улучшать модель и решать проблемы, если что-то пойдет не так.

Если всё ещё ничего непонятно, keep calm and:

- пройдите небольшой курс по multivariate calculus на khan academy
- посмотрите вот это видео про градиентный спуск
- прочитайте эту и эту статью
- если удастся сформулировать вопросы, спрашивайте :)

Интуиция за backpropagation

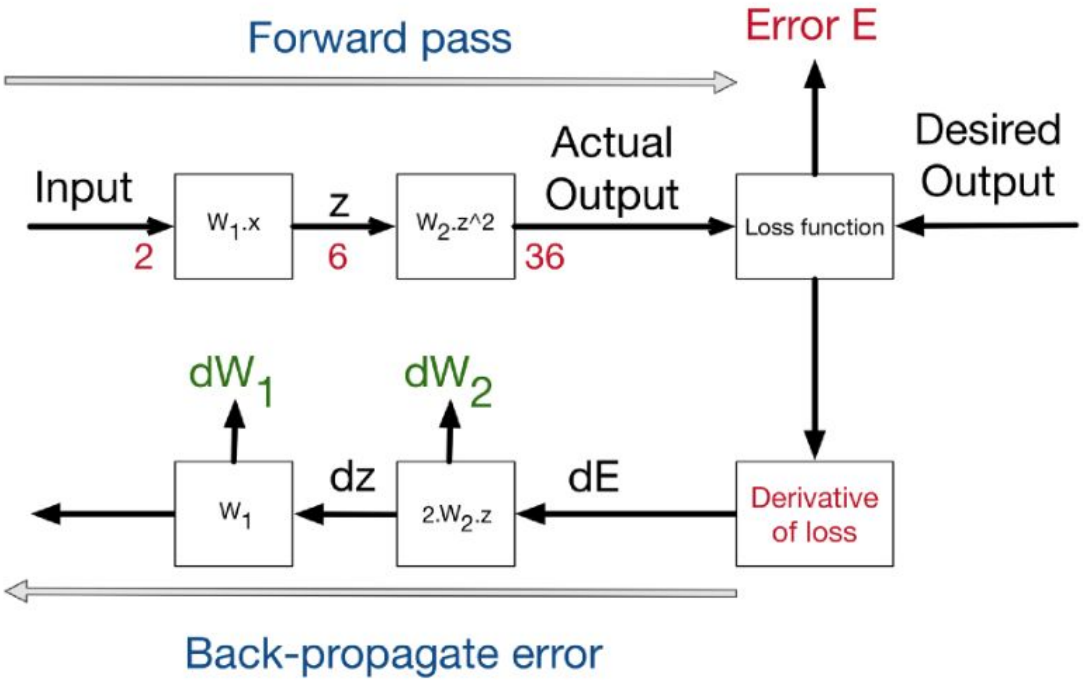
Суммирование ошибки



А дальше минимизируем функцию потерь.

Каждый объект говорит о том, как надо изменить веса, чтобы стало “правильнее”.

Обратное распространение



- слой за слоем применяем нейронку
- предсказываем класс объекта
- сравниваем с реальным и считаем ошибку
- слой за слоем изменяем параметры

Теория за backpropagation

Нейросеть — это тоже функция

x — входные данные (признаки); W — веса

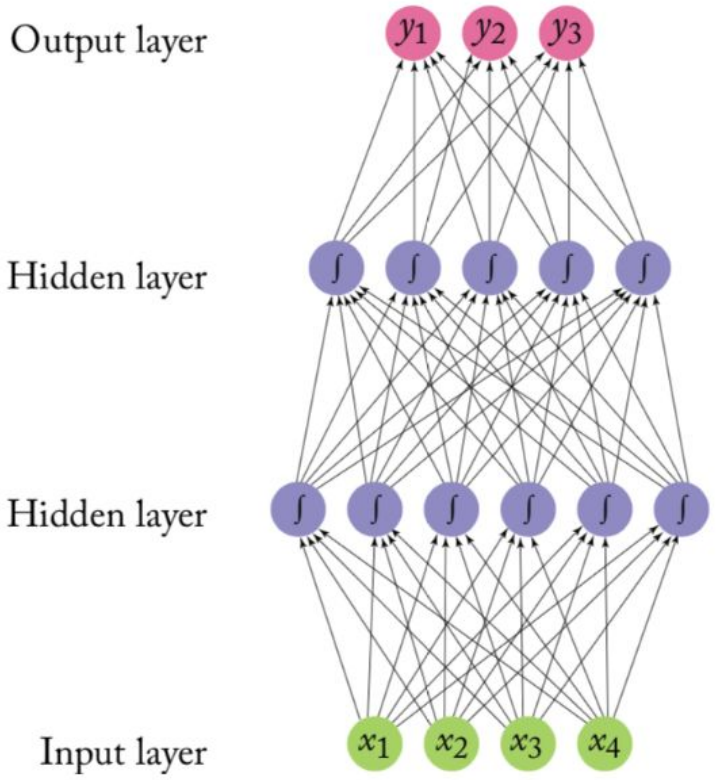
$h_1 = f_1(W_1 * x + b_1)$

$h_2 = f_2(W_2 * h_1 + b_2)$

$y\_pred = f_3(W_3 * h_2 + b_3)$

$y\_pred = f_3(W_3 * f_2(W_2 * h_1 + b_2) + b_3)$

$loss = ((\mathbf{y\_pred} - \mathbf{y\_true})^2).mean()$



Композиция функций

В математике:

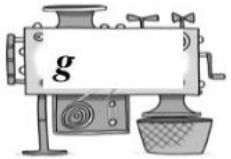
Substituting a function or it's value into **another** function.

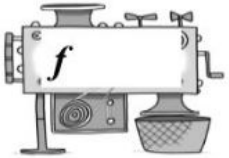
Second

$$f\left(g\left(x\right)\right)$$

First

(inside parentheses always first)





OR

$$f \circ g\left(x\right)$$

В программировании — то же самое!

Chain Rule

Это правило про то, как брать производную от композиции функций.

$$(f \circ g)' = (f' \circ g) \cdot g'.$$

This may equivalently be expressed in terms of the variable. Let $F = f \circ g$, or equivalently, $F(x) = f(g(x))$ for all x. Then one can also write

$$F'(x) = f'(g(x))g'(x).$$

Chain rule for NN

$$\frac{\partial E(\widetilde{y}(w))}{\partial w} = \frac{\partial E(\widetilde{y}(w))}{\partial \widetilde{y}(w)} \frac{\partial \widetilde{y}(w)}{\partial w}$$