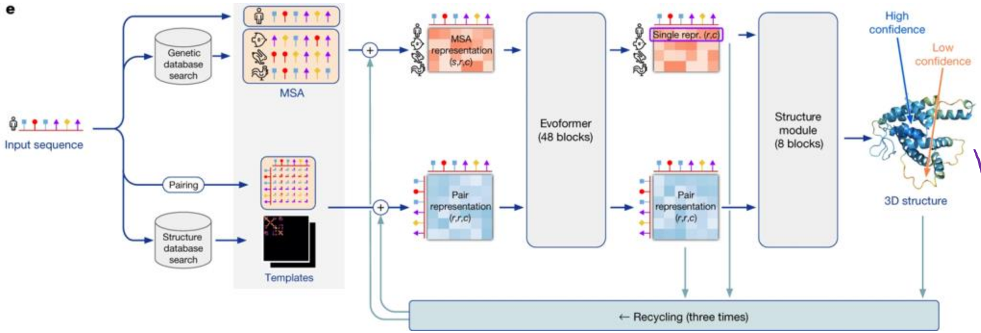
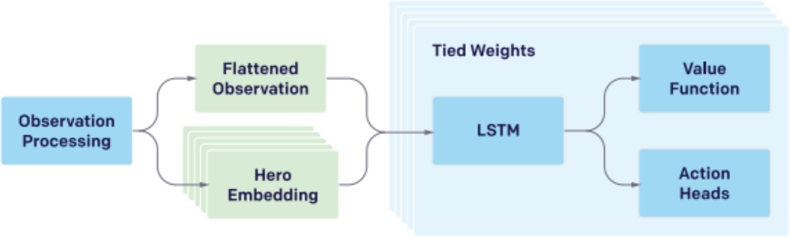
Машинное обучение

Лекция 7. Нейронные сети. Начало

(13.03.2023)

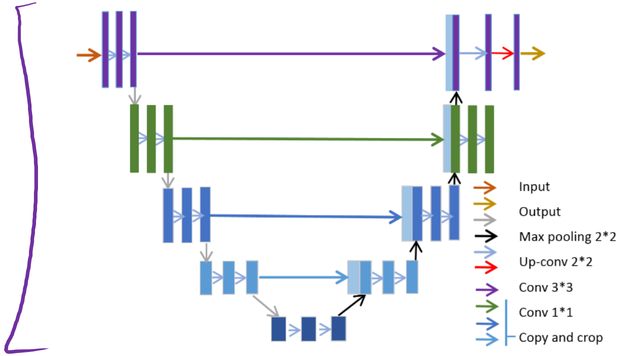
Начало.

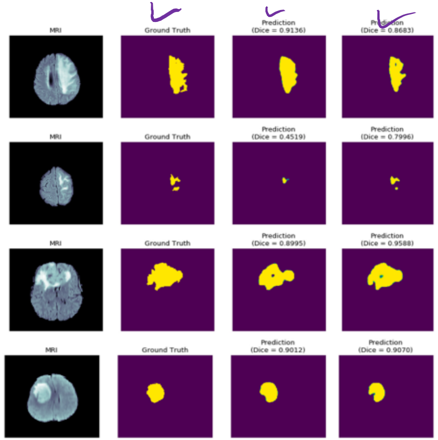




- openAI bots

UNET



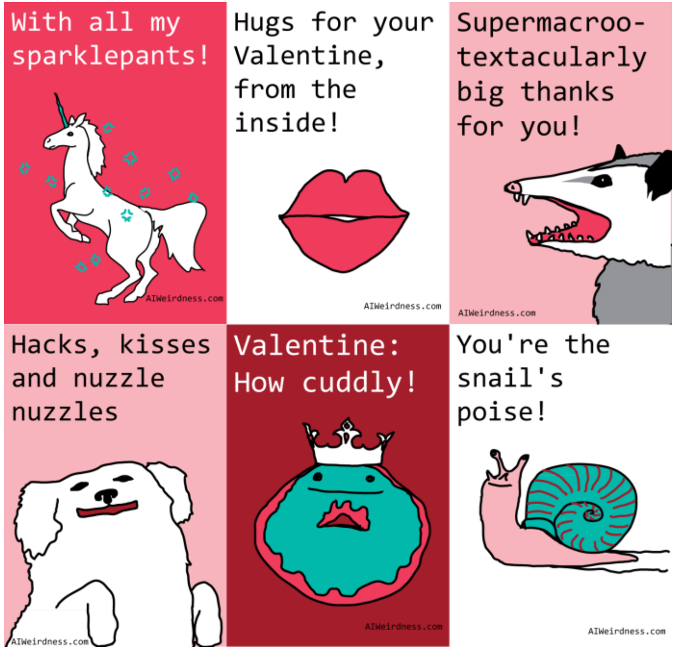


Inventing magic spells

I trained a neural network twice on Dungeons and Dragons spells, and once on spells from Harry Potter. See if you can figure out which list is which.

> Chorus of the dave
> Song of the doom goom
> Barking Sphere
> Gland Growth
> Hold Mouse

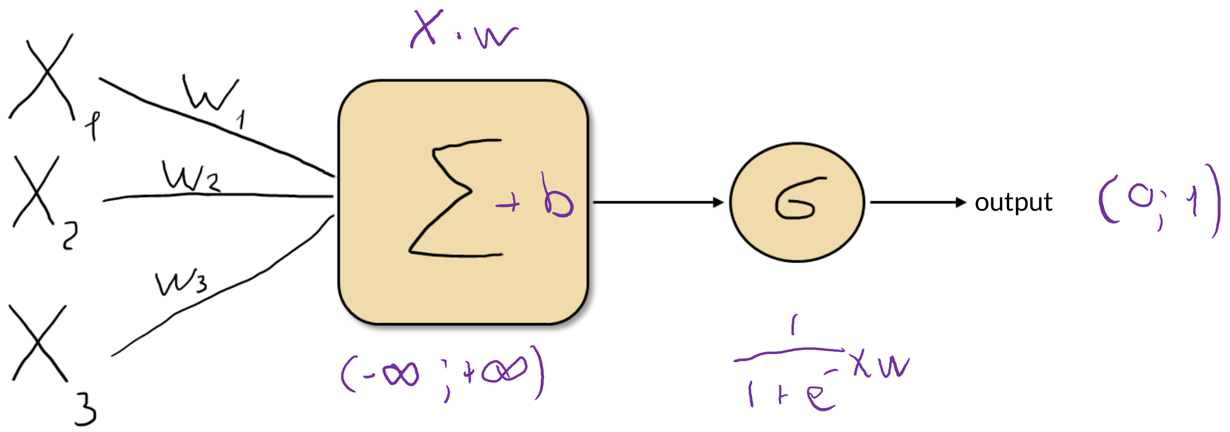
> Hurder-gerping Charm
> Regrowing hair to curse of the Bogies
> Brechaim hedbivicus Doobers Spell
> Fubbledory Charm
> Squggly-wing fart



ссылка

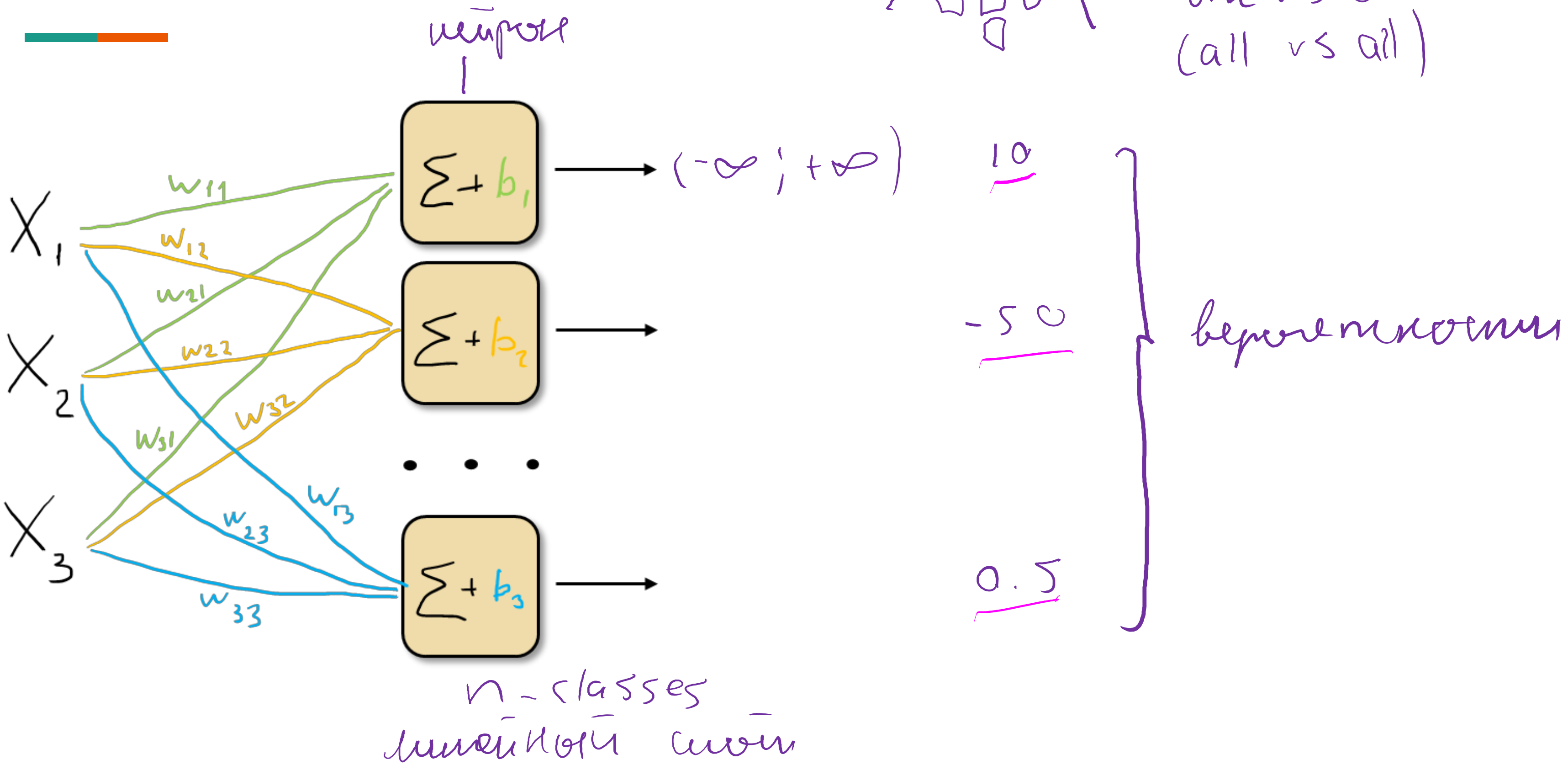
ссылка

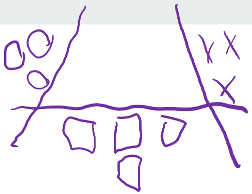
Вспомним логистическую регрессию



$$x_1 \cdot w_1 + x_2 w_2 + x_3 \cdot w_3 + b$$

А если классов больше, чем 2...



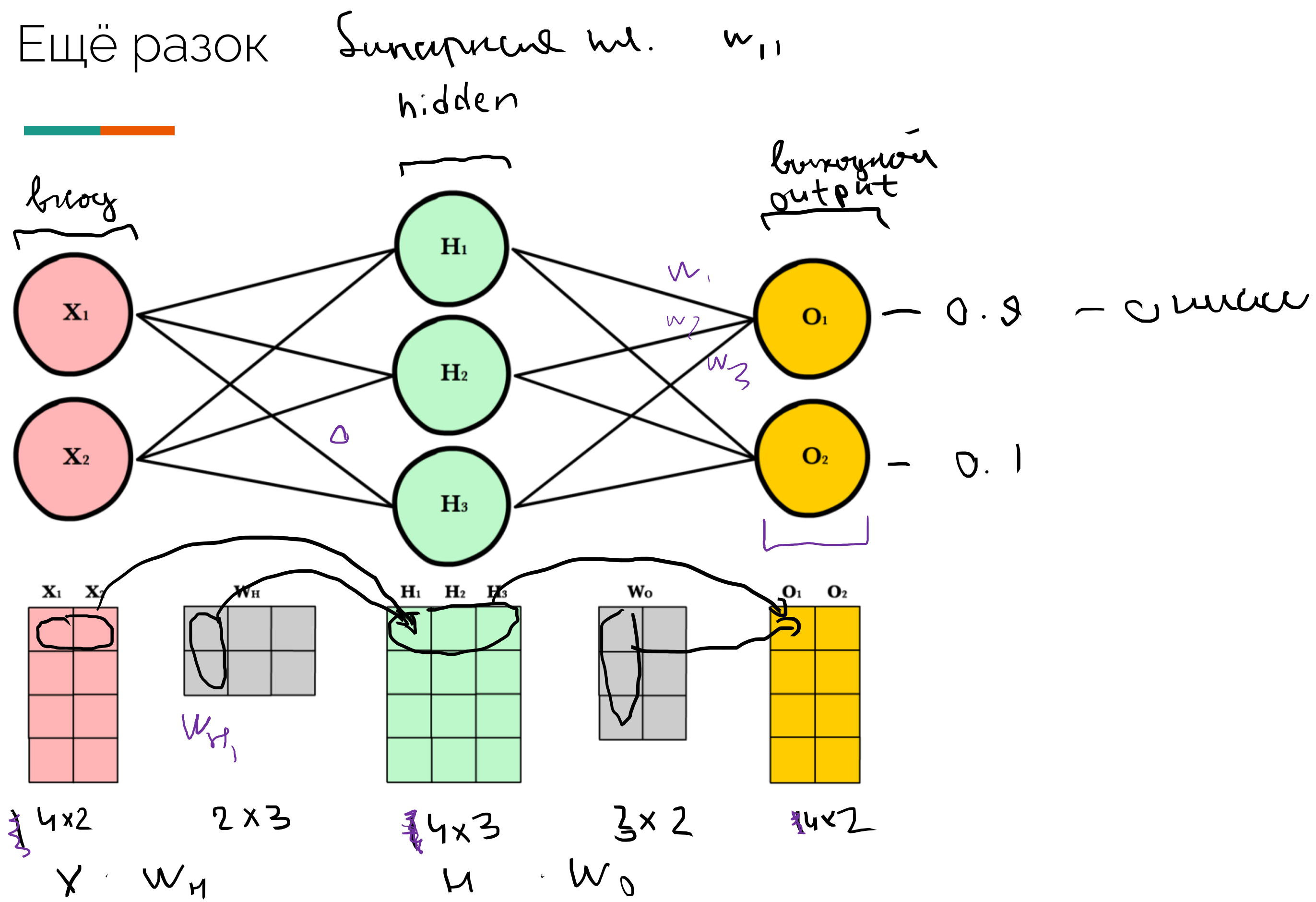


one vs all

one vs one

(all vs all)

Ещё разок



hidden
output
X1
X2
H1
H2
H3
O1
O2
– 0.9
– 0.1
X1 X2
WH
H1 H2 H3
Wo
O1 O2
4x2
2x3
4x3
3x2
4x2
X · WH
H · Wo

Бинарная классификация

$$X \cdot W + b = \text{logits}$$

100 , 110

$\downarrow$ $\quad$ $\downarrow$

~ 1 $\quad$ ~ 1

X

$$\begin{bmatrix} 0.5 & 1 & 0 & 1 \end{bmatrix}$$

$$\cdot \underset{W}{\begin{bmatrix} 0 & -2 \\ 1 & 0 \\ 0.5 & -1 \\ -1 & 3 \end{bmatrix}} + \underbrace{\begin{bmatrix} 0 \\ 0.5 \end{bmatrix}^T}_{b} = \underset{\text{logits}}{\begin{bmatrix} 0 \\ 3.5 \end{bmatrix}^T}$$

1

4 x 2 $\quad$ число классов

4 призн.

Softmax

logits

$$y = \begin{bmatrix} 0.5 \\ -50 \\ 10 \\ 0 \end{bmatrix} \rightarrow \frac{e^{y_i}}{\sum_j e^{y_j}} \rightarrow p = \begin{bmatrix} 0.04 \\ 0 \\ 0.95 \\ 0.01 \end{bmatrix}$$

$$\frac{e^{0.5}}{\sum_i e^{logits_i}}$$

$\text{argmax}(logits) \rightarrow 2$

$(x_1 \cdot w_1 \cdot x_2 \cdot w_2 + b) - (-\infty; \infty)$

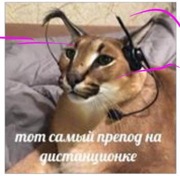
logit

$-\frac{1}{N}\sum_i y_i \log p_i + (1-y_i)\log(1-p_i)$

Принцип максимального правдоподобия. Maximum likelihood

samples | features | labels | predictions

1  $x_1\ x_2 \dots x_n$ | 0 | 0.9 0.09 0.01
0.1 0.9 0

2 

... | 1 | 0.01 0.99 0

3  . . . | 2 | . . .

likelihood, pred cl, true label

$$\prod_s p(c = gt_s|x_s) \to \max$$

ln

$$\frac{1}{N}\sum_s \ln p(c = gt_s|x_s) \to \max$$

$\cdot(-1)$

$$\frac{1}{N}\cdot -\sum_s \ln p(c = gt_s|x_s) \to \min$$

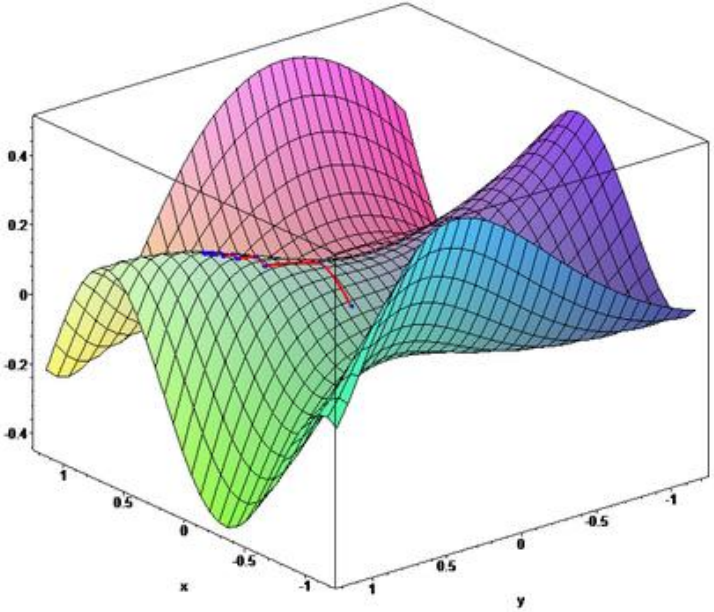
N — число объектов

Cross entropy

А как учиться?

А как учиться? Градиентный спуск

RAM = 8 ГБ
vRAM = 4 ГБ



float32 – 4 байта
1 000 000 – 4 МБ
1 000 000 000 – 4 ГБ

Стохастический градиентный спуск. Stochastic Gradient Descent (SGD)

1 epoch

batch

shuffle

2 epoch

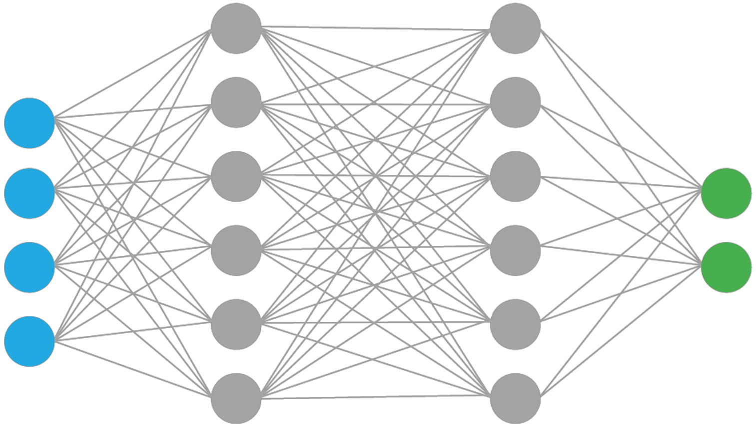
$w_{i+1} = w_i - \eta \cdot \nabla_w loss$

$\nabla loss$

lr

loss

pred

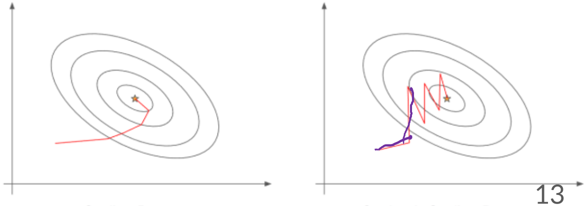


выборка

10000

батч

200

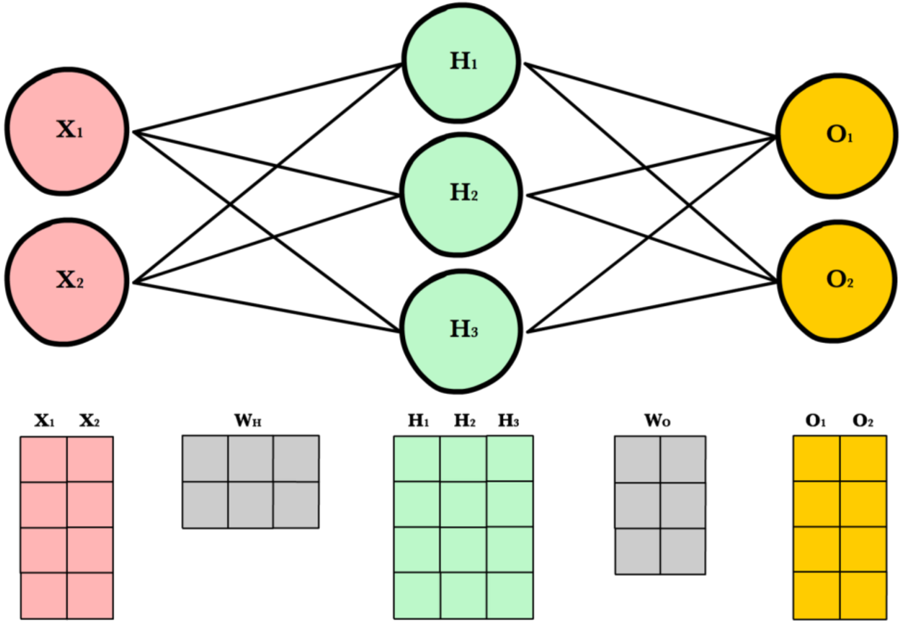


Gradient Descent

Stochastic Gradient Descent

Отдых

Будет ли работать просто так?



$$\underset{4\times 2}{X} \cdot \left(\underset{2\times 3}{W_H} \cdot \underset{3\times 2}{W_O}\right)$$

$$W_H \cdot W_O = \underset{2\times 2}{W_{new}}$$

Функции активации

Хотим, чтобы активация была:

Нелинейная:
Функция активации необходима для введения нелинейности в нейронные сети. Если функция активации не применяется, выходной сигнал становится простой линейной функцией. Неактивированная нейронная сеть будет действовать как линейная регрессия с ограниченной способностью к обучению:

$$\hat{y} = NN(X, W_1, \ldots, W_n) = X \cdot W_1 \cdot \ldots \cdot W_n = X \cdot W$$

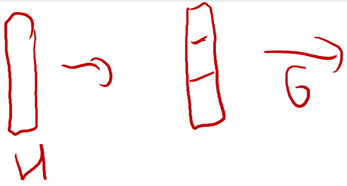
Только нелинейные функции активации позволяют нейронным сетям решать задачи аппроксимации нелинейных функций:

$$\hat{y} = NN(X, W_1, \ldots, W_n) = \sigma(\ldots \sigma(X \cdot W_1) \ldots \cdot W_n) \neq X \cdot W$$

Дифференцируемая:
Функции активации должны быть дифференцируемые, то есть от них можно взять производную

ссылка

Функции активации. Сигмоида



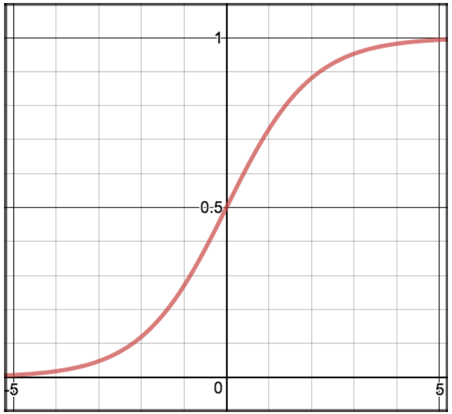
$$1-\sigma(x) = \frac{e^x+1}{e^x+1} - \frac{e^x}{e^x+1} = \frac{1}{e^x+1}$$

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

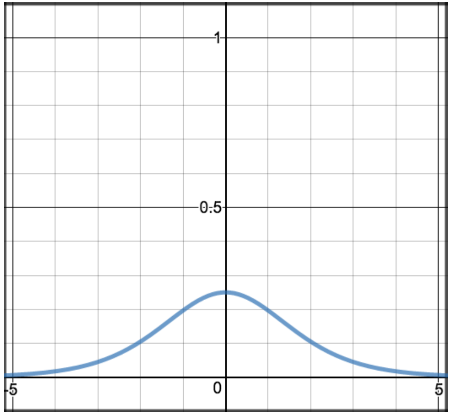
$$\sigma'(x) = \frac{e^x\cdot(e^x+1) - e^x\cdot e^x}{(e^x+1)^2} =$$

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

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



σ



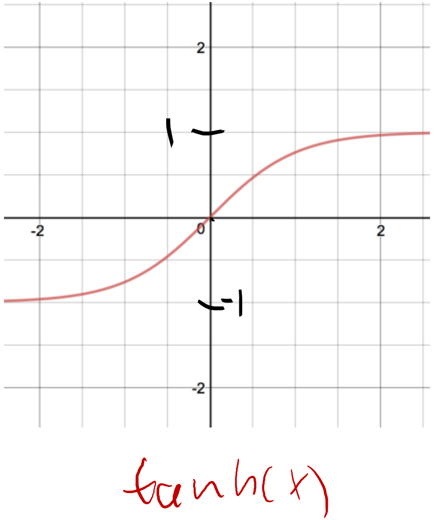
σ'

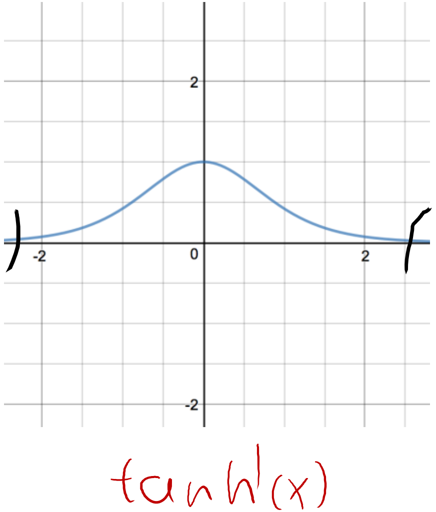
Недостатки:

1. Насыщение сигмоиды приводит к затуханию градиентов
2. Выход сигмоиды не центрирован относительно нуля

Функции активации. Гиперболический тангенс

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



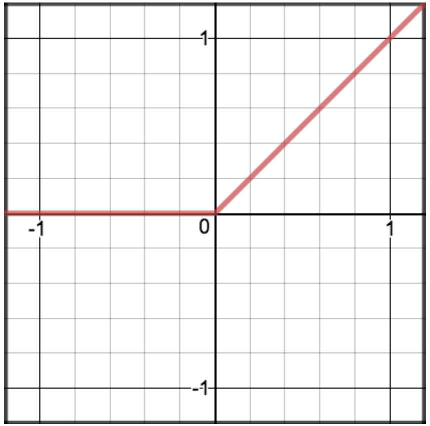


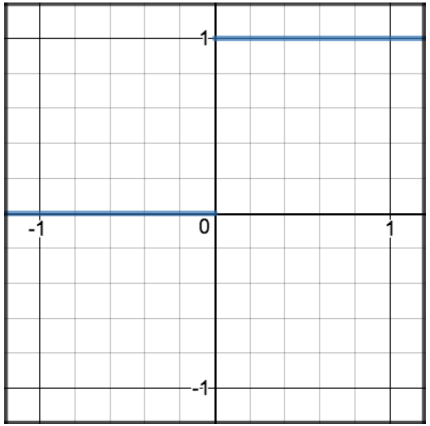
Недостатки:

1. Снова затухание градиентов

Функции активации. ReLU (rectified linear unit)

$$relu(x) = max(0, x)$$

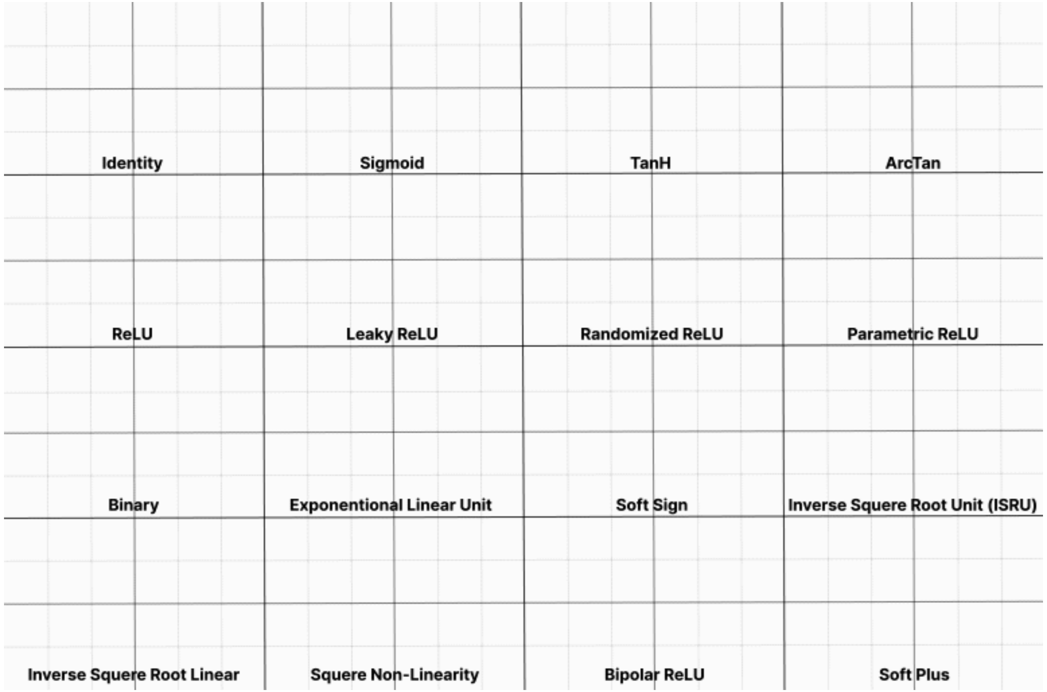




Недостатки:

1. Может “умереть” в области слева от 0

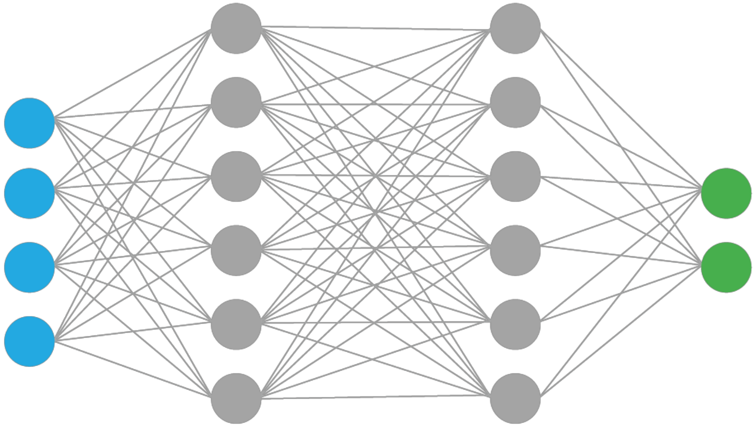
Функции активации. Другие функции активации



Тренировка

forward pass

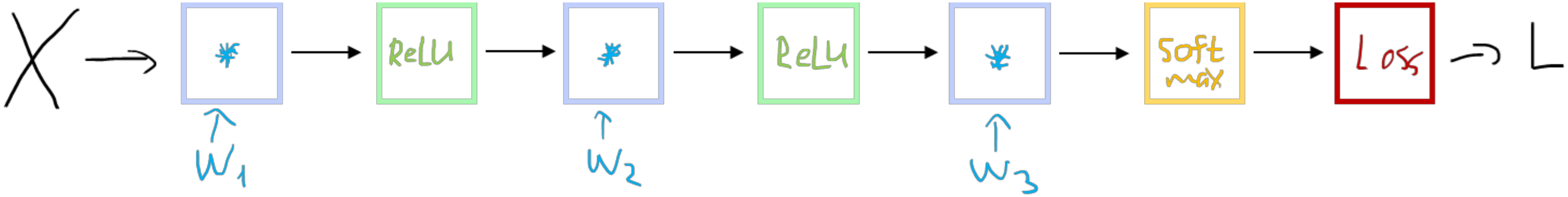
X →



→ outputs → loss

backward pass

Граф вычислений



$(Relu(Xw_1)\cdot w_2)\ldots$

$m(f(g(x)))$

Алгоритм обратного распространения ошибки. Backpropagation

Алгоритм обратного распространения ошибки позволяет находить градиенты для любого графа вычислений, если функция которую он описывает дифференцируема (каждый из узлов дифференцируемый). В его основе лежит правило взятия производной сложной функции (chain rule).

$$\boxed{f(g(x))}$$

$$\boxed{\frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx}}$$

ссылка

Тренируемся

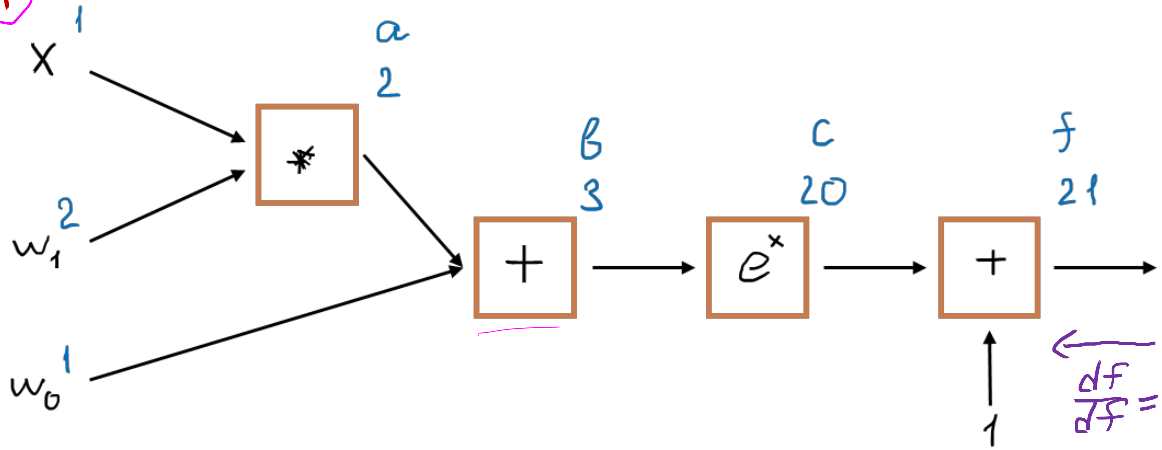
$f(g(x))$

$$\frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx}$$

$$f(x, w_0, w_1) = 1 + e^{w_0 + w_1 x}$$

$f = c + 1$

$\frac{df}{dc} = \frac{df}{df} \cdot \frac{df}{dc} = 1$



$\frac{df}{df} = 1$

$\frac{df}{dx}$ - ?

$\frac{df}{dw_0}$ - ?

$\frac{df}{dw_1}$ - ?

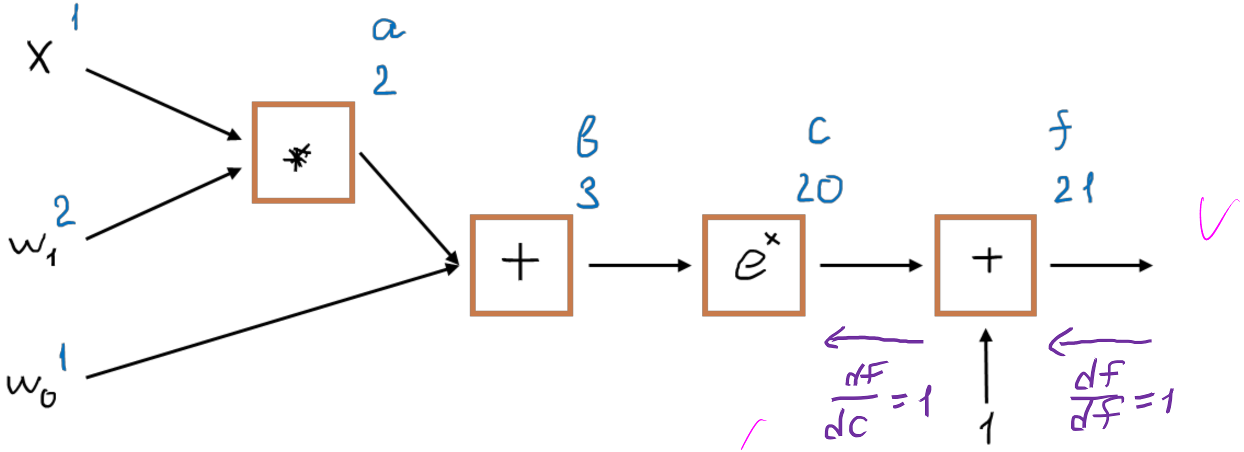
Тренируемся

$(e^x)' = e^x$

$\boxed{f(g(x))}$ $\boxed{\frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx}}$ $\boxed{f(x, w_0, w_1) = 1 + e^{w_0 + w_1 x}}$

$c = e^b$

$\frac{df}{db} = \frac{df}{dc} \cdot \frac{dc}{db}$, $\frac{df}{dc} = 1$, $\frac{dc}{db} = 20$



Тренируемся

$$f(g(x))$$

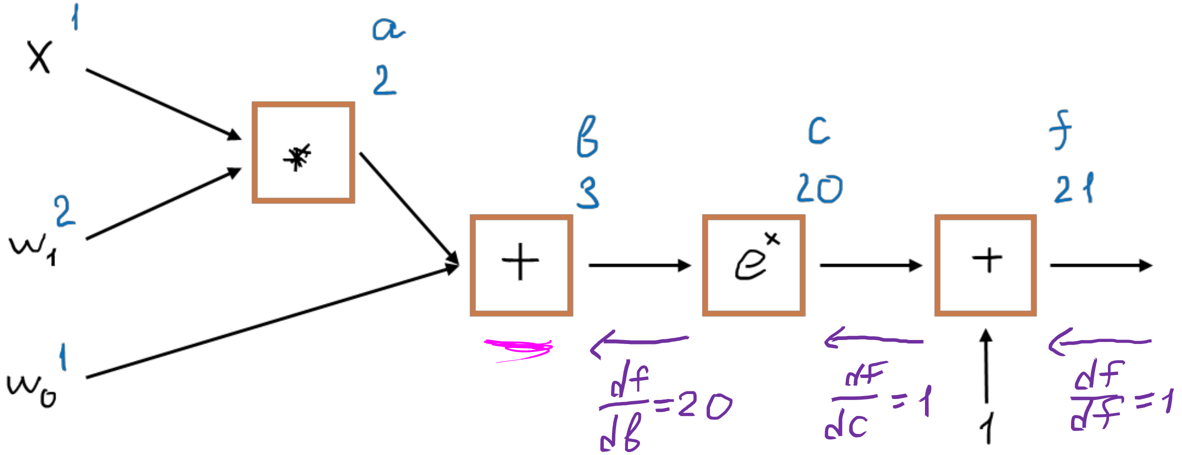
$$\frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx}$$

$$f(x, w_0, w_1) = 1 + e^{w_0 + w_1 x}$$

$b = a + w_0$

$$\frac{df}{dw_0} = \frac{df}{db} \cdot \frac{db}{dw_0}$$

$\frac{df}{db} = 20$, $\frac{db}{dw_0} = 1$



Тренируемся

$$f(g(x))$$

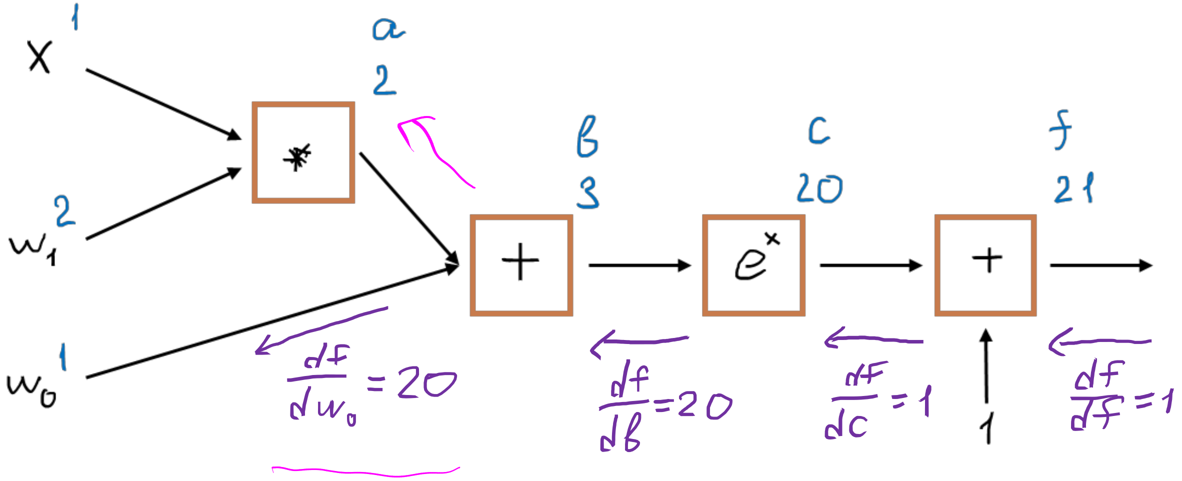
$$\frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx}$$

$$f(x, w_0, w_1) = 1 + e^{w_0 + w_1 x}$$

$b = a + w_0$

$\frac{df}{da} = \frac{df}{db} \cdot \frac{db}{da} = 20$

$\frac{df}{db} = 20$, $\frac{db}{da} = 1$



Тренируемся

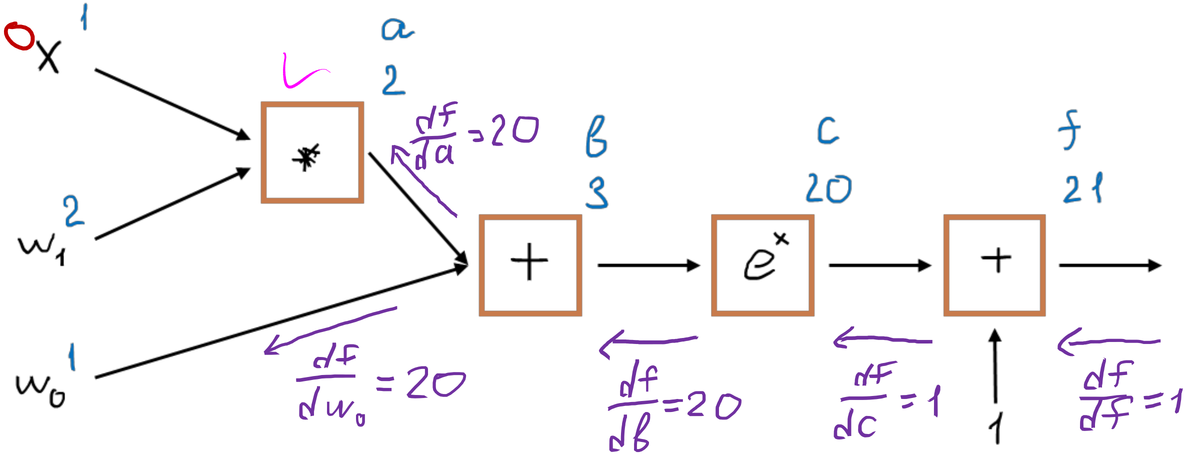
$f(g(x))$

$$\frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx}$$

$$f(x, w_0, w_1) = 1 + e^{w_0 + w_1 x}$$

$a = w_1 \cdot x$

$$\frac{df}{dw_1} = \underset{20}{\underset{\parallel}{\frac{df}{da}}} \cdot \underset{1}{\underset{\parallel}{\frac{da}{dw_1}}} - 20$$



Тренируемся

$$f(g(x))$$

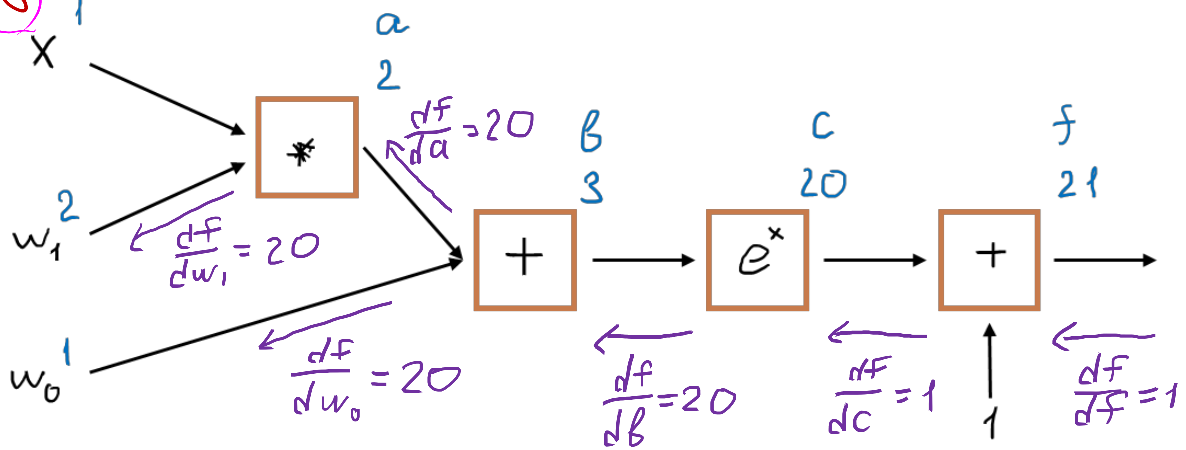
$$\frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx}$$

$$f(x, w_0, w_1) = 1 + e^{w_0 + w_1 x}$$

$a = w_1 \cdot x$

$\frac{df}{dx} = \frac{df}{da} \cdot \frac{da}{dx} = 40$

$\frac{df}{da} = 20$, $\frac{da}{dx} = 2$

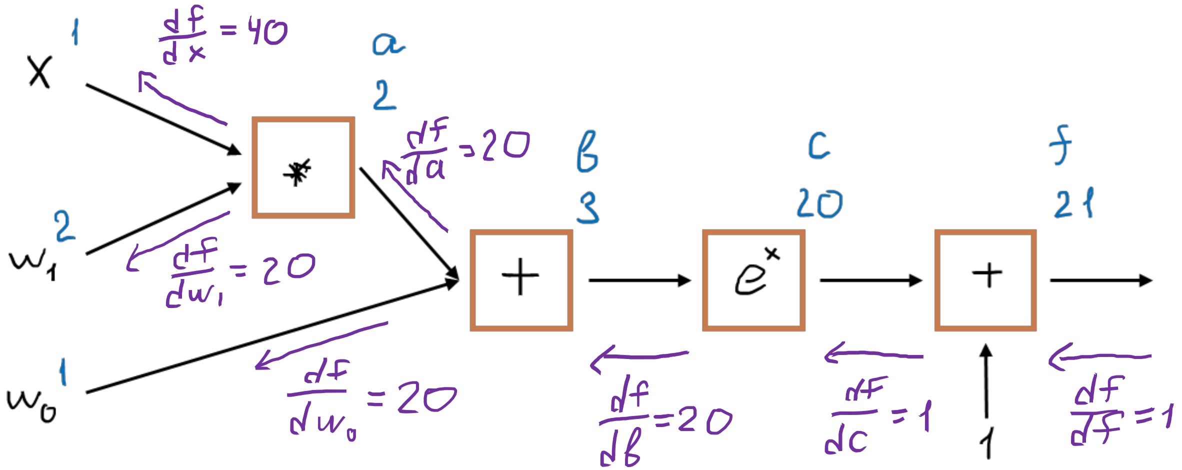


Тренируемся

$f(g(x))$

$\frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx}$

$f(x, w_0, w_1) = 1 + e^{w_0 + w_1 x}$



Тренируемся

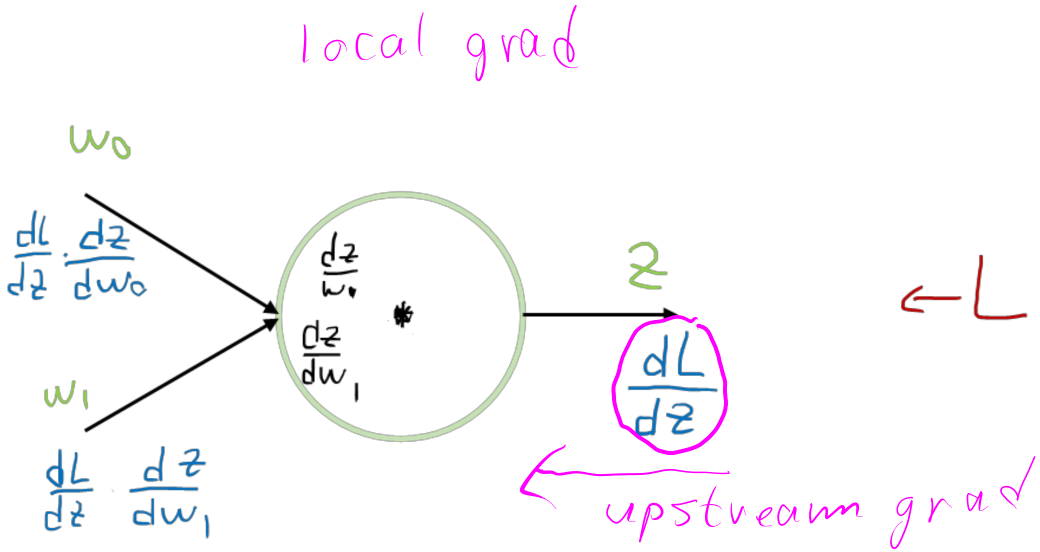
$$f(g(x))$$

$$\frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx}$$

$$f(x, w_0, w_1) = 1 + e^{w_0 + w_1 x}$$

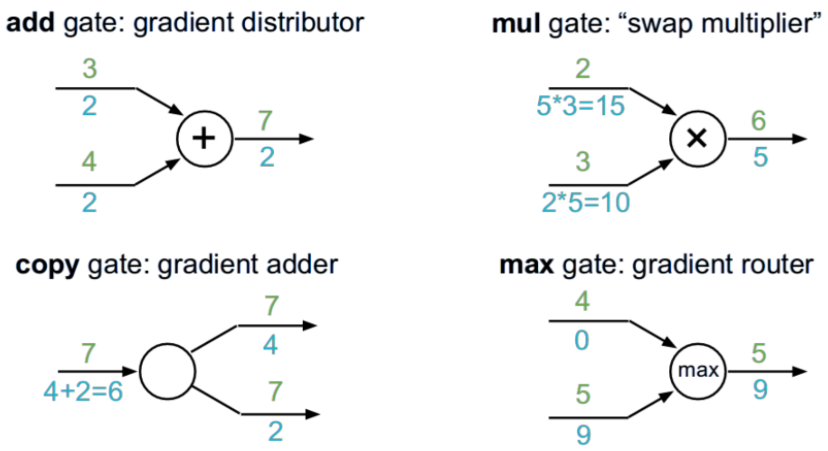
А зачем мы все это считали?

Общая схема вычисления градиента



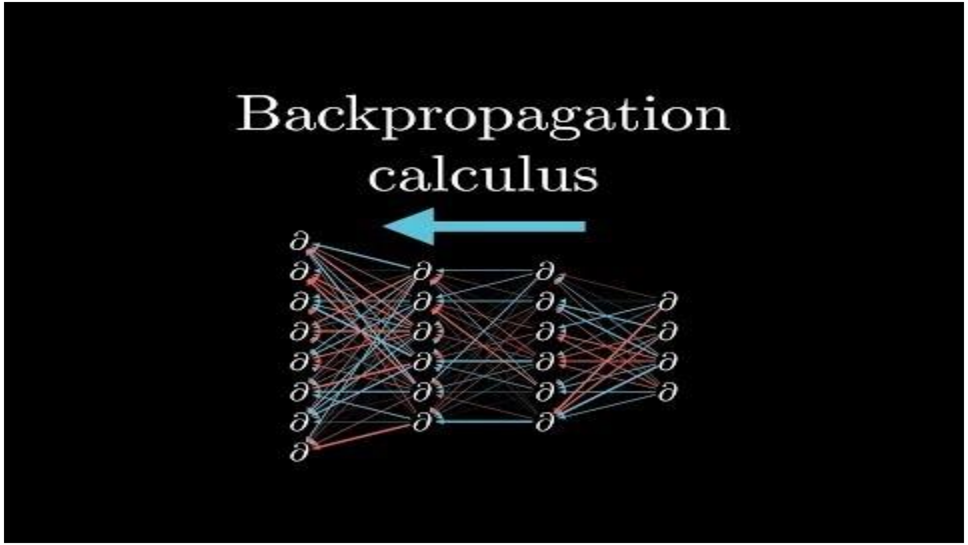
$$\frac{dL}{dw_0} = \frac{dL}{df} \cdot \frac{df}{dw_0}$$

Уточнения



add gate: gradient distributor
3
2
4
2
+
7
2
mul gate: “swap multiplier”
2
5*3=15
3
2*5=10
×
6
5
copy gate: gradient adder
7
4+2=6
7
4
7
2
max gate: gradient router
4
0
5
9
max
5
9

Если ничего не понятно…




знаешь почему ты
так сильно устаёшь к концу дня?
потому что ты весь день был
замечательным котёночком,
а это тяжелый труд