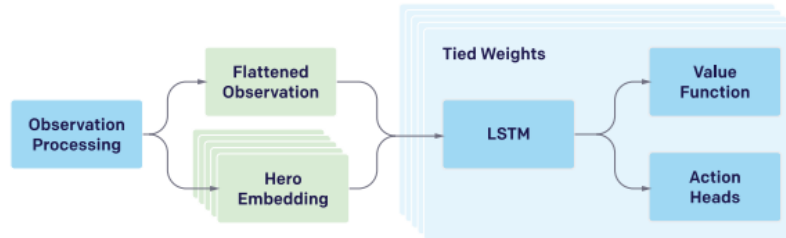
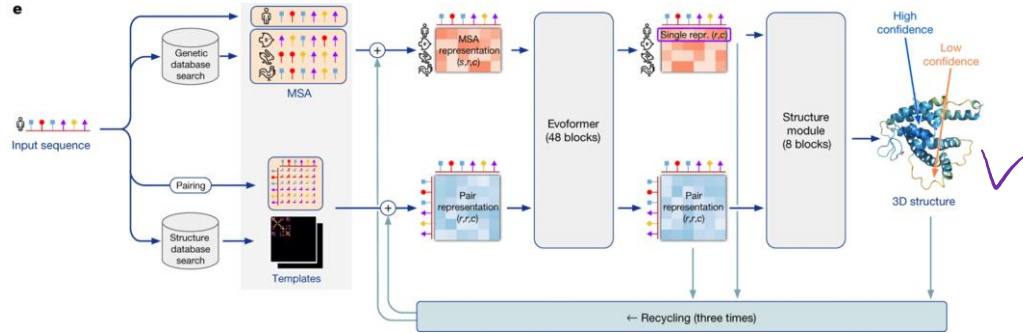
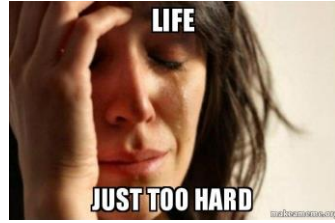
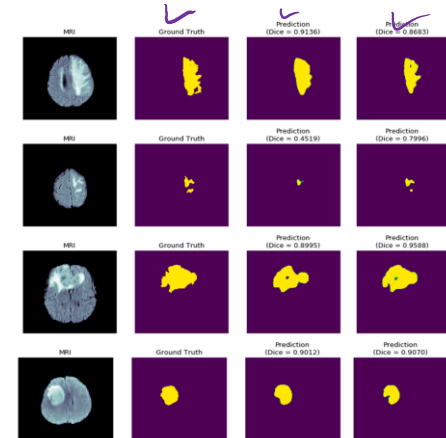
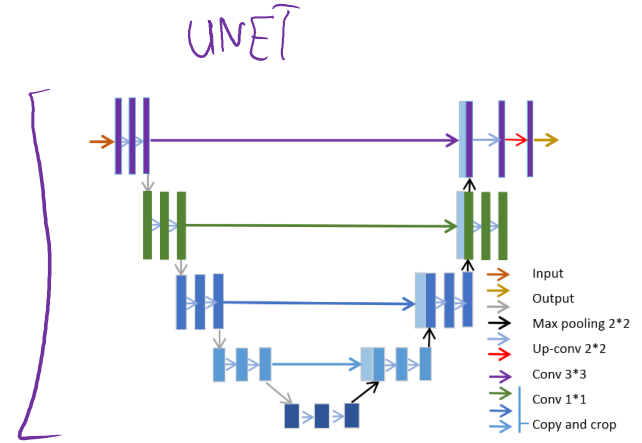


—

Начало.



- OPENAI  
BOB5



### Inventing magic spells



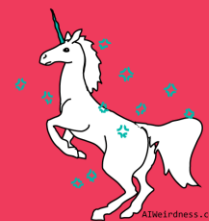
I trained a neural network twice on [Dungeons and Dragons spells](#), and once on [spells from Harry Potter](#). See if you can figure out which list is which.

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



With all my  
sparklepants!



Hugs for your  
Valentine,  
from the  
inside!



Supermacroo-  
textacularly  
big thanks  
for you!



Hacks, kisses  
and nuzzle  
nuzzles



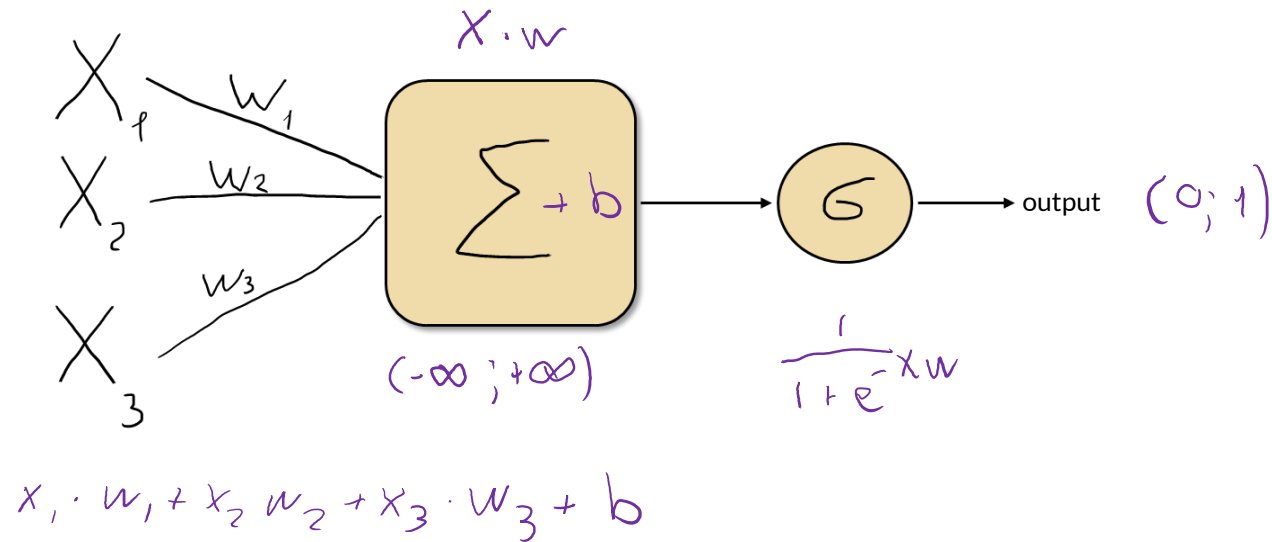
Valentine:  
How cuddly!



You're the  
snail's  
poise!



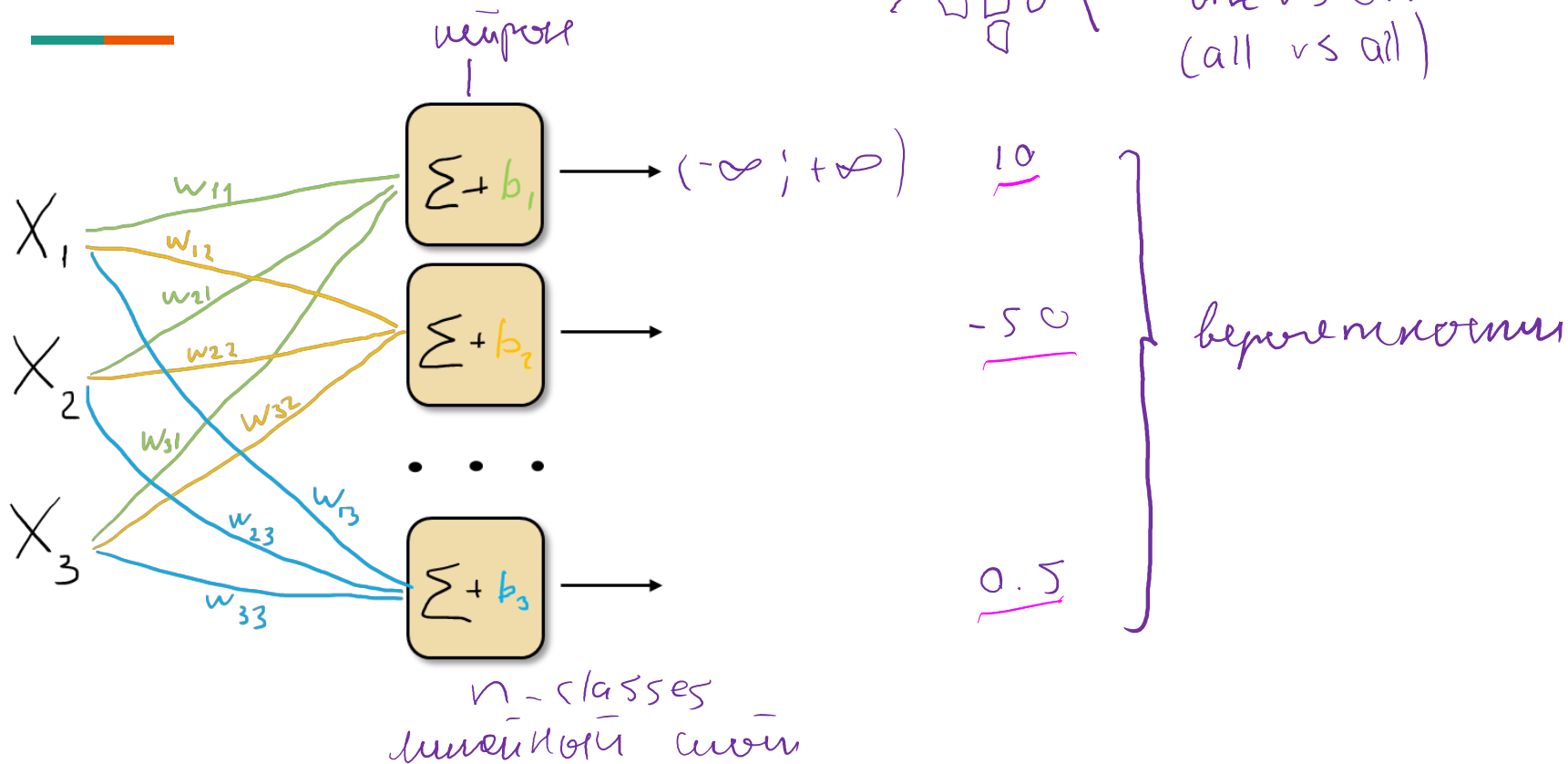
## Вспомним логистическую регрессию



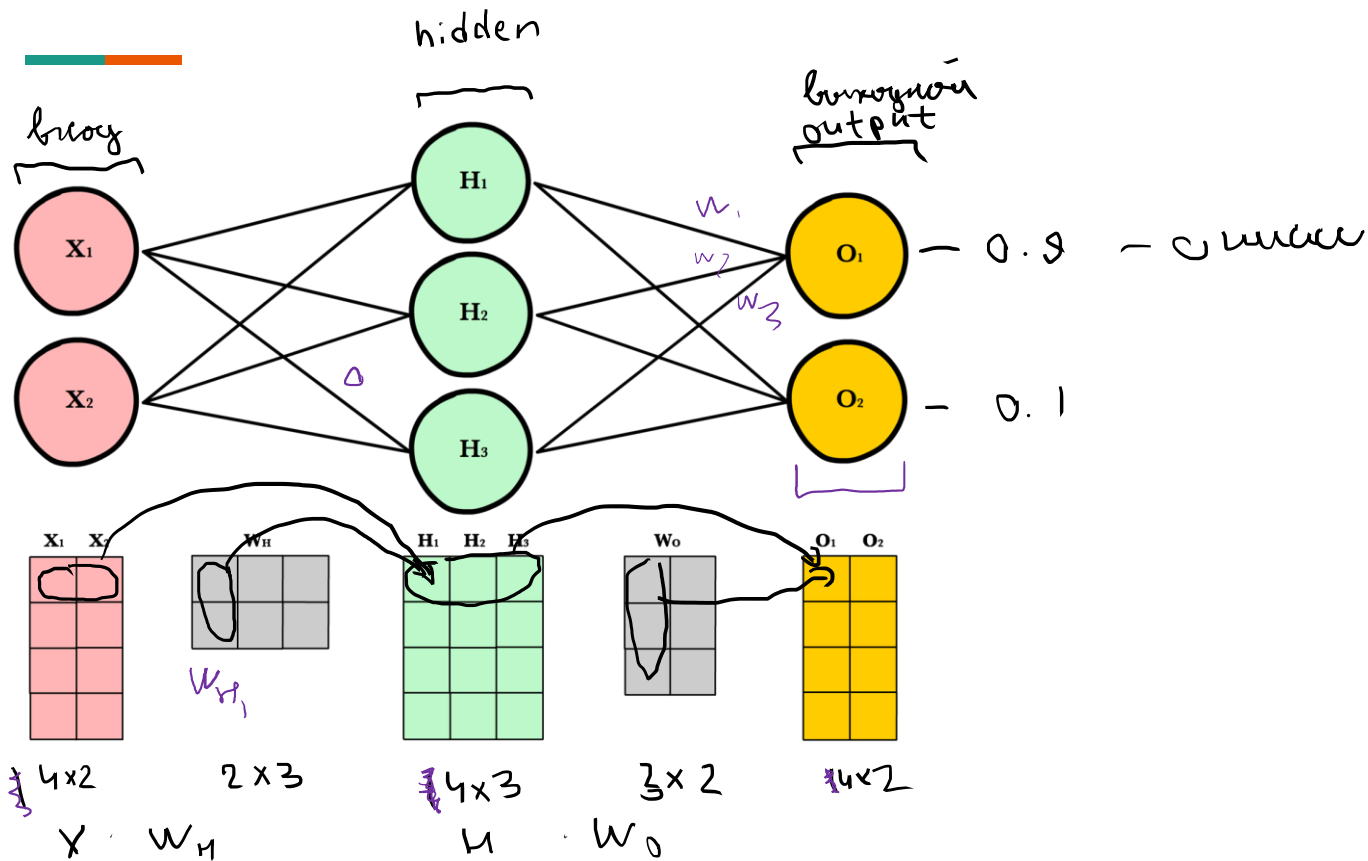
А если классов больше, чем 2...



one vs all  
one vs one  
(all vs all)



Ещё разок Summing w.  $w_{11}$



# Бинарная классификация



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

100 , 110  
 ↓        ↓  
 ~1       ~1

X

0.5	1	0	1
-----	---	---	---

0.5	0
1	0.5

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

logits

W

↙ 4x2 ↘    число связей

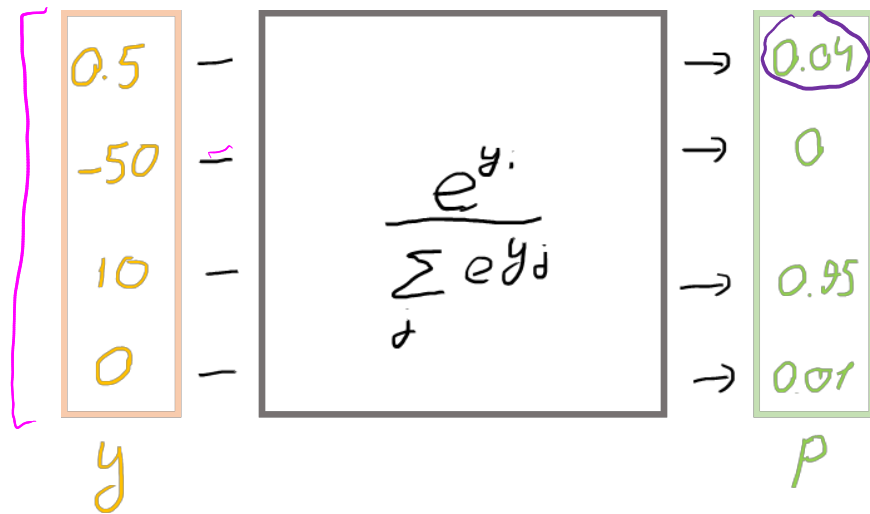
↗    ↘

4 пружн.

# Softmax



logits



$\arg \max(\text{logits}) \rightarrow 2$


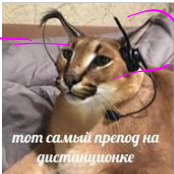
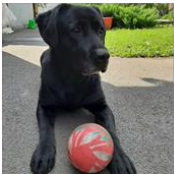
$$\frac{e^{0.5}}{\sum_i e^{\text{logits}_i}}$$

$$(x_1 \cdot w_1 + x_2 \cdot w_2 + b) = \text{logit}$$



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

Принцип максимального правдоподобия. Maximum likelihood

✓ samples	features	labels	predictions
1 	$x_1, x_2, \dots, x_n$	0	$\begin{bmatrix} 0.9 & 0.09 & 0.01 \\ 0.1 & 0.9 & 0 \end{bmatrix}$
2 	$x_1, x_2, \dots, x_n$	1	$\begin{bmatrix} 0.01 & 0.99 & 0 \end{bmatrix}$
3 	$x_1, x_2, \dots, x_n$	2	$\begin{bmatrix} \dots & 0 \end{bmatrix}$

likelihood    pred st    true label

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

$$\downarrow \ln$$

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

$$\downarrow -1$$

$$\frac{1}{n} - \sum_s \ln p(c = gt_s | x_s) \rightarrow \min$$

минимум

Cross entropy



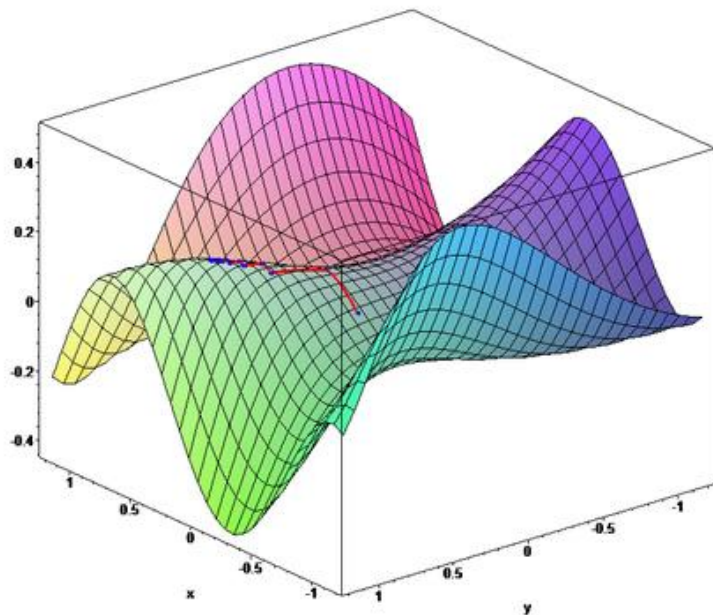
# А как учиться?



Лекции в универе похожи на просмотр Даши путешественницы: препод задаёт вопросы и несколько секунд пялится на аудиторию, а потом сам же отвечает.



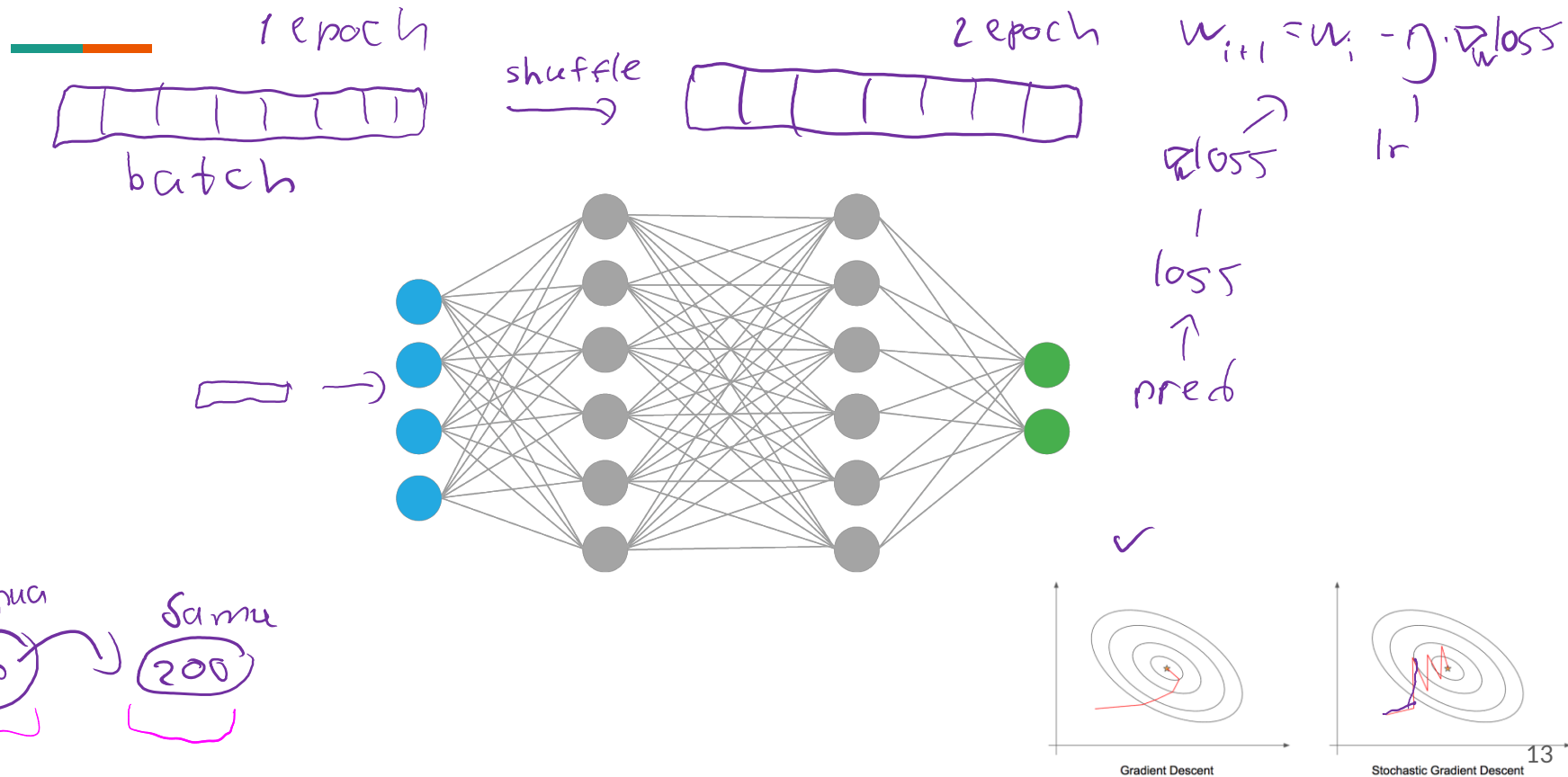
# А как учиться? Градиентный спуск



RAM = 8 GB  
VRAM = 4 GB

float32 - 4 bytes  
1000 000 - 4 MB  
1000 0000 - 4 GB

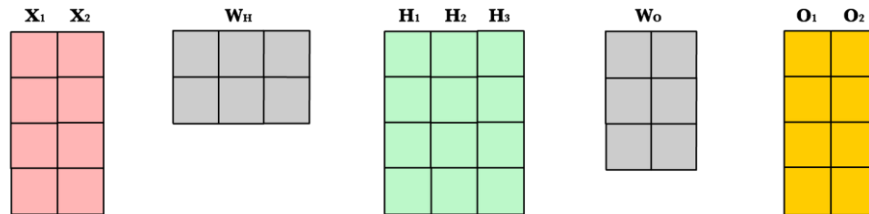
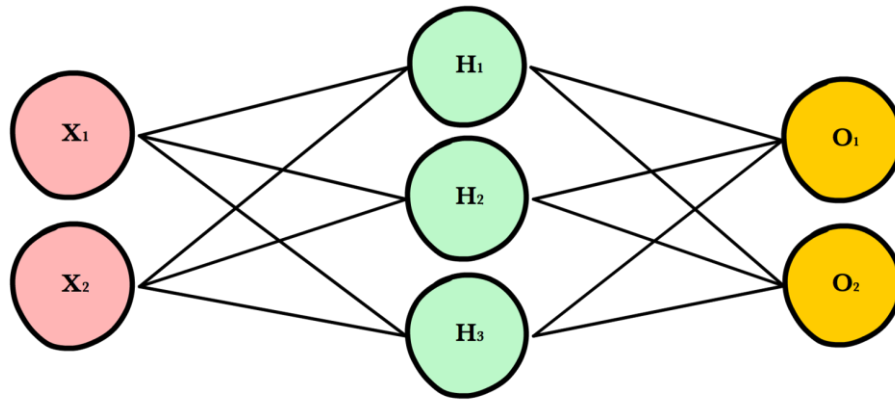
# Стохастический градиентный спуск. Stochastic Gradient Descent (SGD)



---

# Отдых

Будет ли работать просто так?



$$X \cdot (W_H \cdot W_O)$$

$4 \times 2$     $2 \times 3$     $3 \times 2$

$$W_H \cdot W_O = W_{new}$$

$2 \times 2$

# 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:

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

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

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

## Дифференцируемая:

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

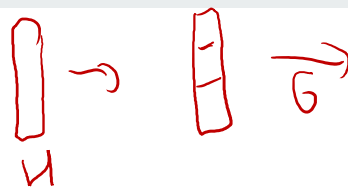
# Функции активации. Сигмоида

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

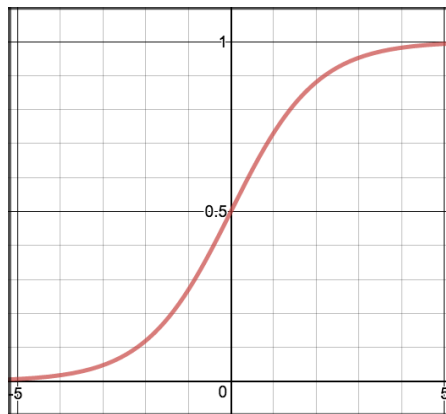
$$\sigma'(x) = \frac{e^x \cdot (e^x+1) - e^x \cdot e^x}{(e^x+1)^2} =$$

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

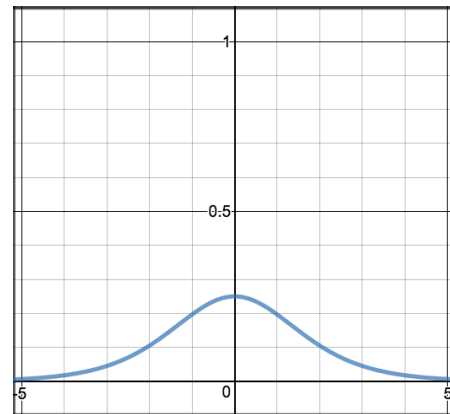
$$= \frac{e^x}{e^x+1} \cdot \frac{1}{e^x+1} = \sigma(x) \cdot (1-\sigma(x))$$



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



$\sigma$




$\sigma'$

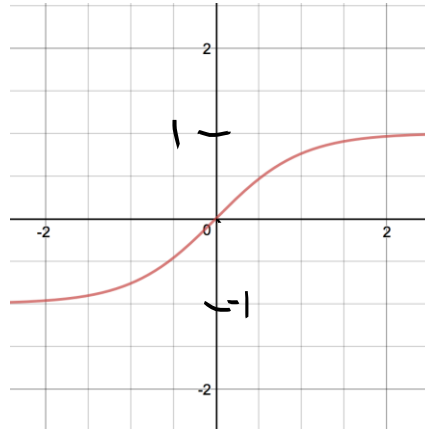
## Недостатки:

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

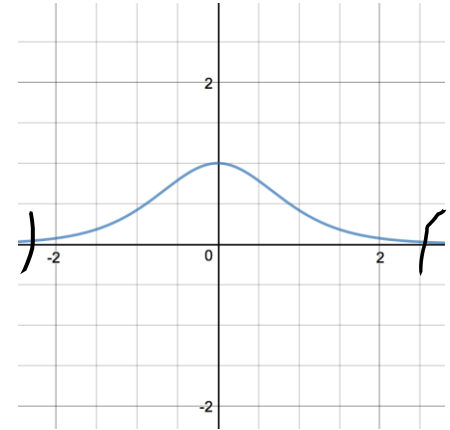


## Функции активации. Гиперболический тангенс


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



$\tanh(x)$



$\tanh'(x)$

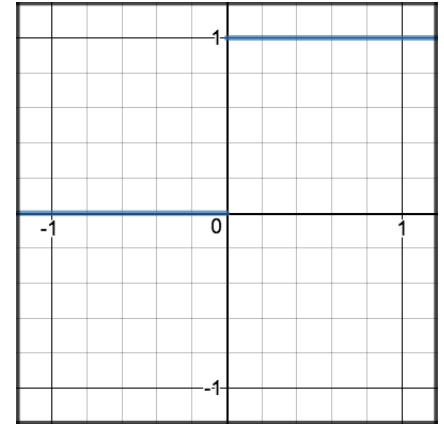
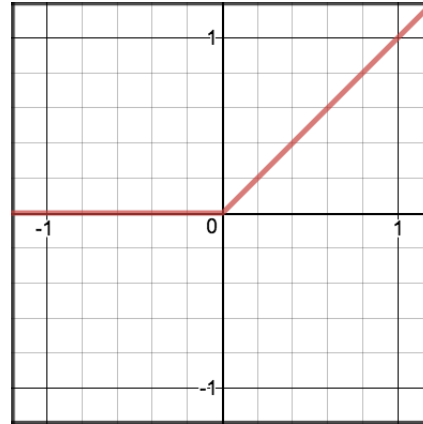
### Недостатки:

1. Снова затухание градиентов

## Функции активации. ReLU (rectified linear unit)



$$\text{relu}(x) = \max(0, x)$$



### Недостатки:

1. Может “умереть” в области слева от 0

# Функции активации. Другие функции активации



Identity	Sigmoid	TanH	ArcTan
ReLU	Leaky ReLU	Randomized ReLU	Parametric ReLU
Binary	Exponential Linear Unit	Soft Sign	Inverse Square Root Unit (ISRU)
Inverse Square Root Linear	Square Non-Linearity	Bipolar ReLU	Soft Plus

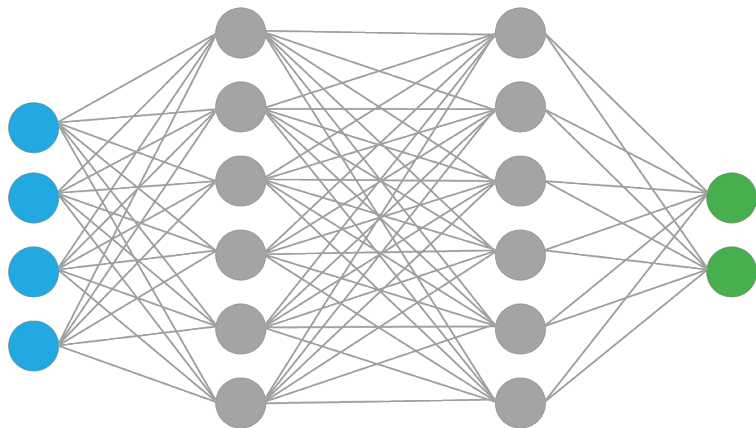
# Тренировка



forward pass



$X \rightarrow$

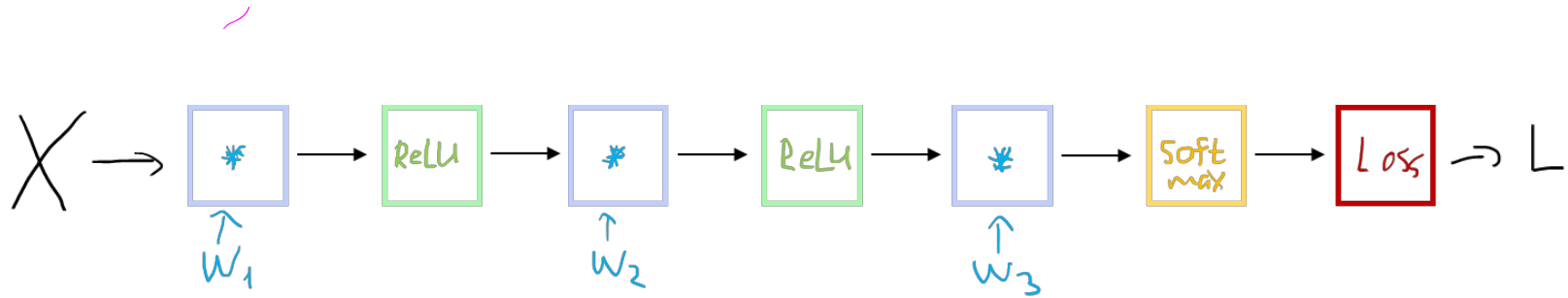


$\rightarrow$  outputs  $\rightarrow$  loss



backward pass

# Граф вычислений

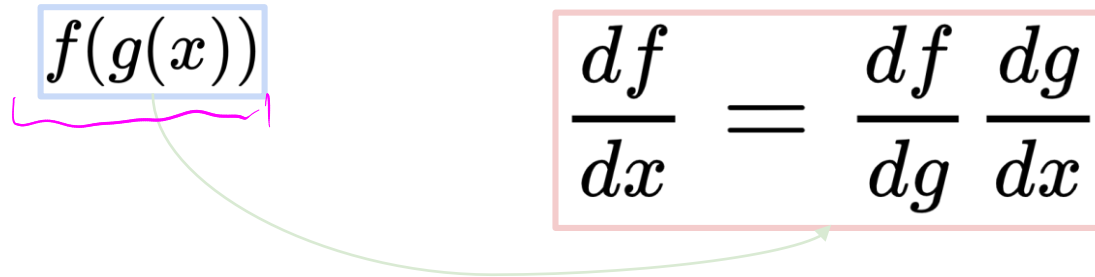


$$(\text{ReLU}(Xw_1) \cdot w_2) \dots$$

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

# 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).


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

# Тренируемся



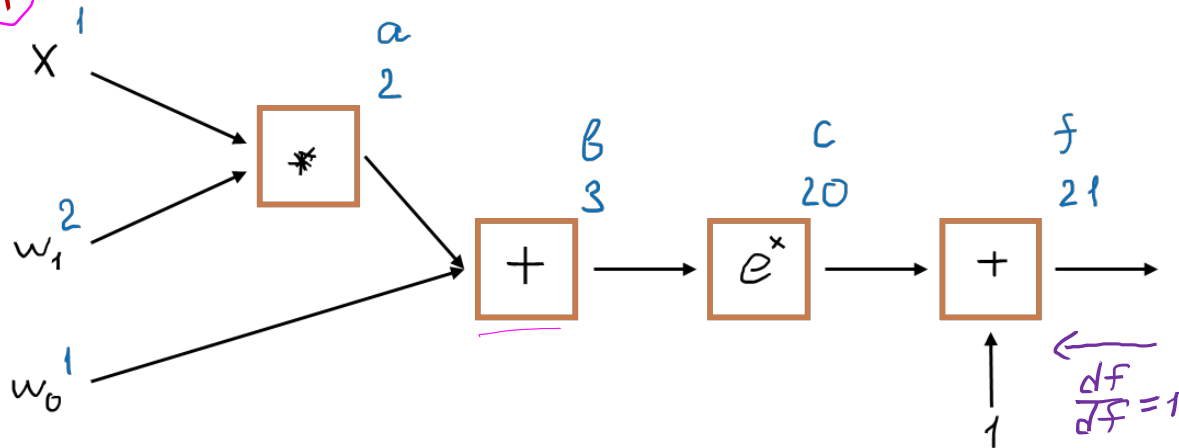
$$f(g(x))$$

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

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

$$f = c + 1$$

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



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

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

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

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

Тренируемся

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

$$f(g(x))$$

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

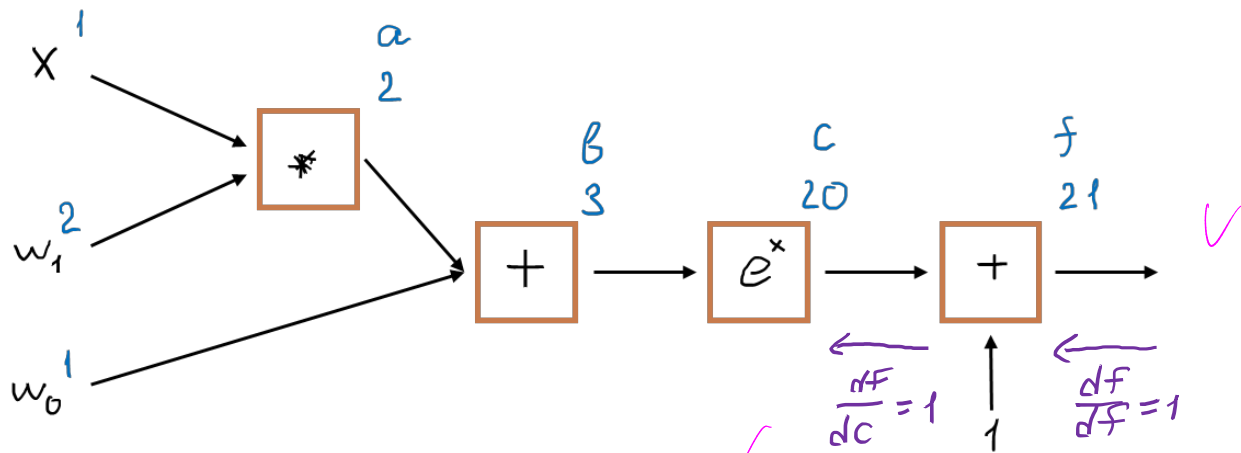
$$f(x, w_0, w_1) = 1 + e^{w_0 + w_1 x}$$

$$c = e^b$$

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

" " " "

1 20





# Тренируемся



$$f(g(x))$$

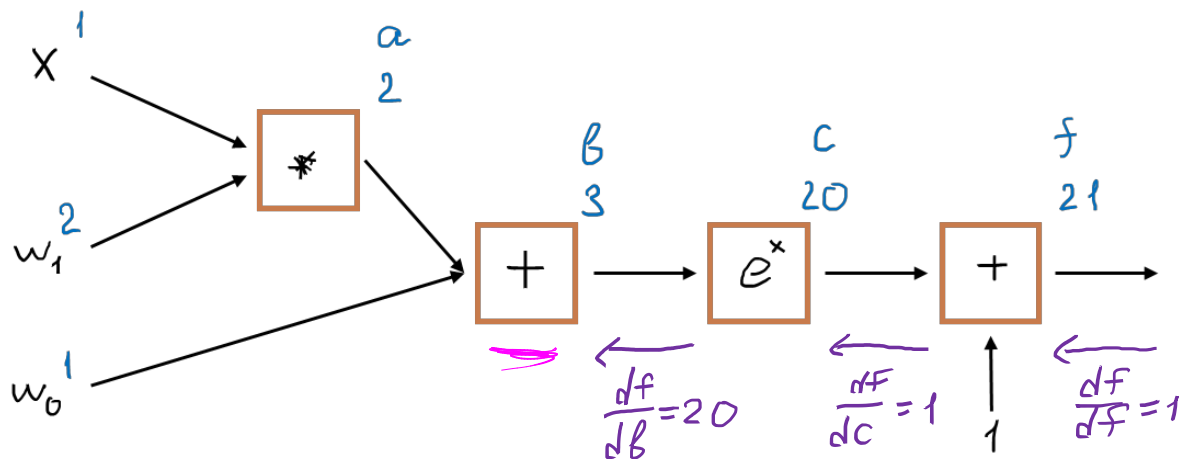
$$\frac{df}{dx} = \frac{df}{dg} \frac{dg}{dx}$$

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

$$b = a + w_0$$

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

$\begin{matrix} \text{"} & \text{"} \\ 20 & 1 \end{matrix}$



# Тренируемся

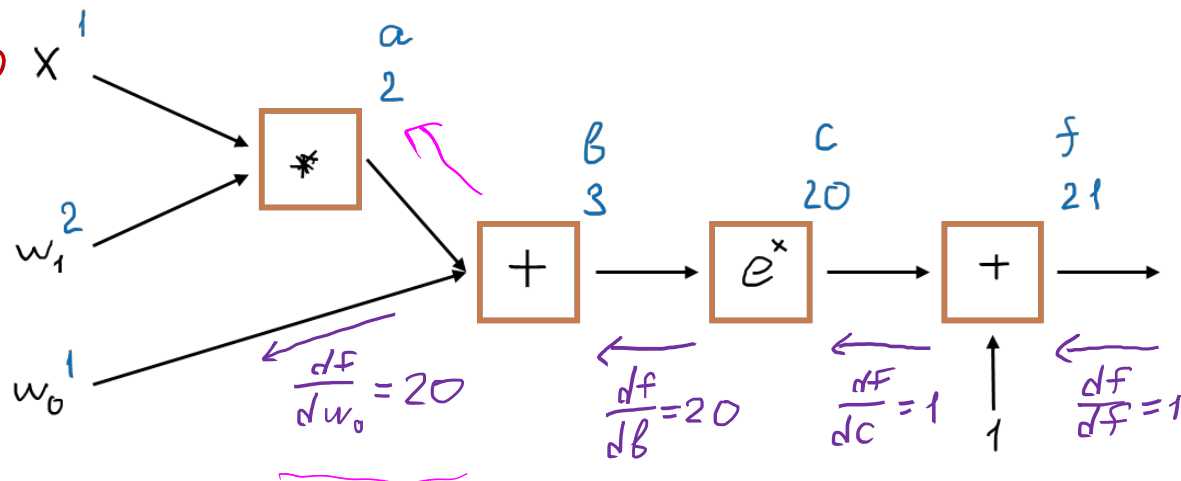
$$f(g(x))$$

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

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

$$b = a + w_0$$

$$\frac{df}{da} = \frac{df}{db} \cdot \frac{db}{da} = 20 \times 1 = 20$$



# Тренируемся

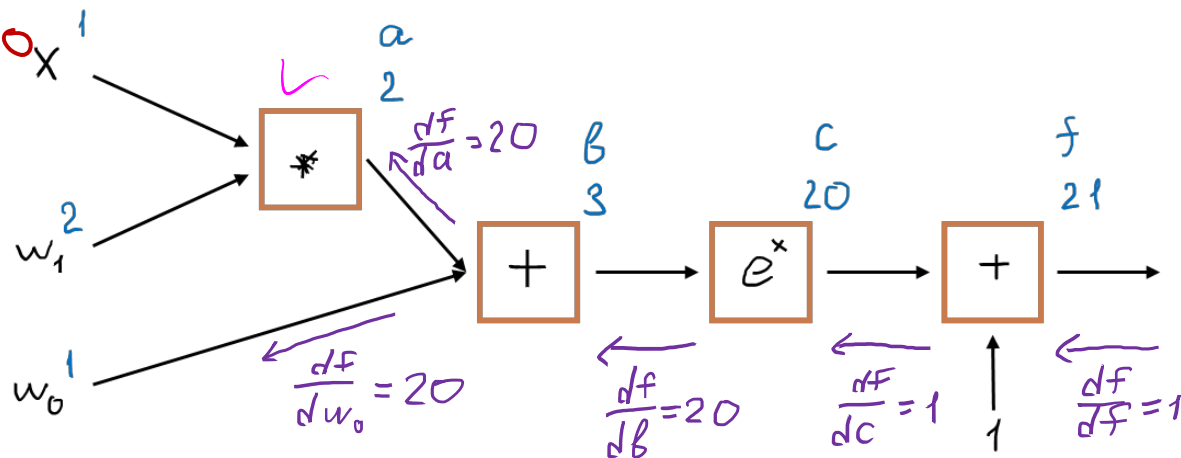
$$f(g(x))$$

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

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

$$a = w_1 \cdot x$$

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



# Тренируемся

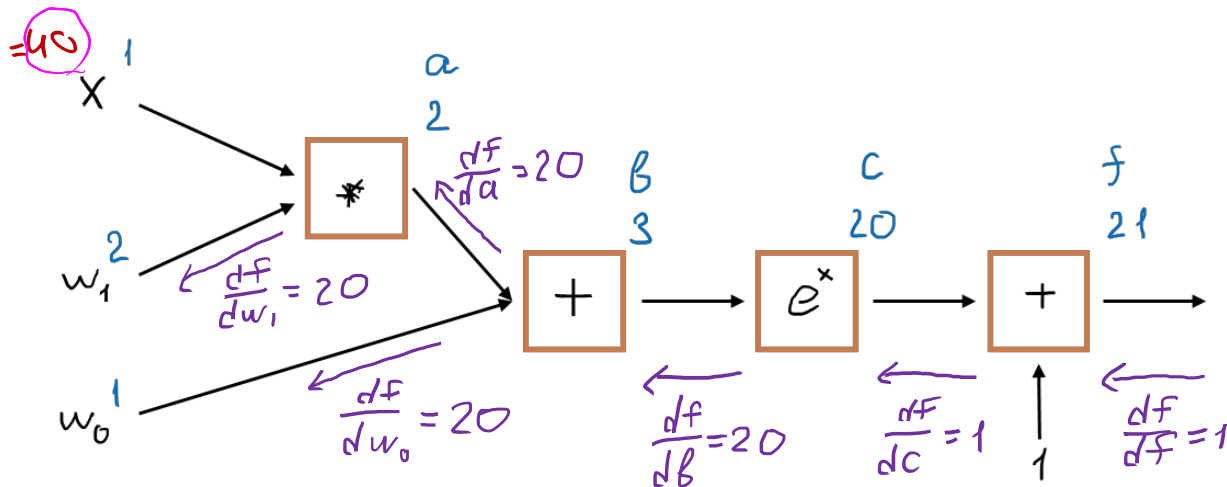
$$f(g(x))$$

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

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

$$a = w_1 \cdot x$$

$$\frac{df}{dx} = \frac{df}{da} \cdot \frac{da}{dx} = 20 \cdot 2 = 40$$

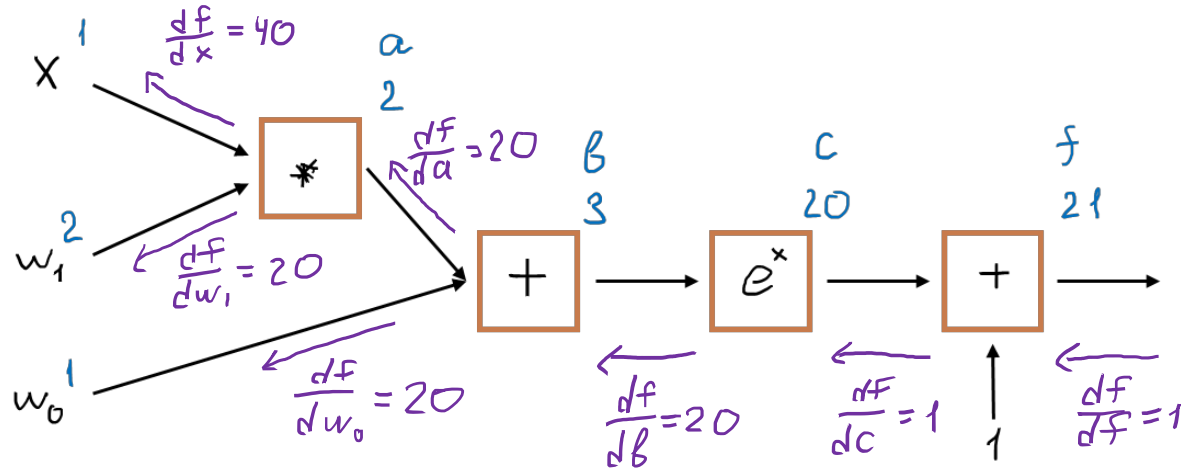


# Тренируемся

$$f(g(x))$$

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

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



# Тренируемся



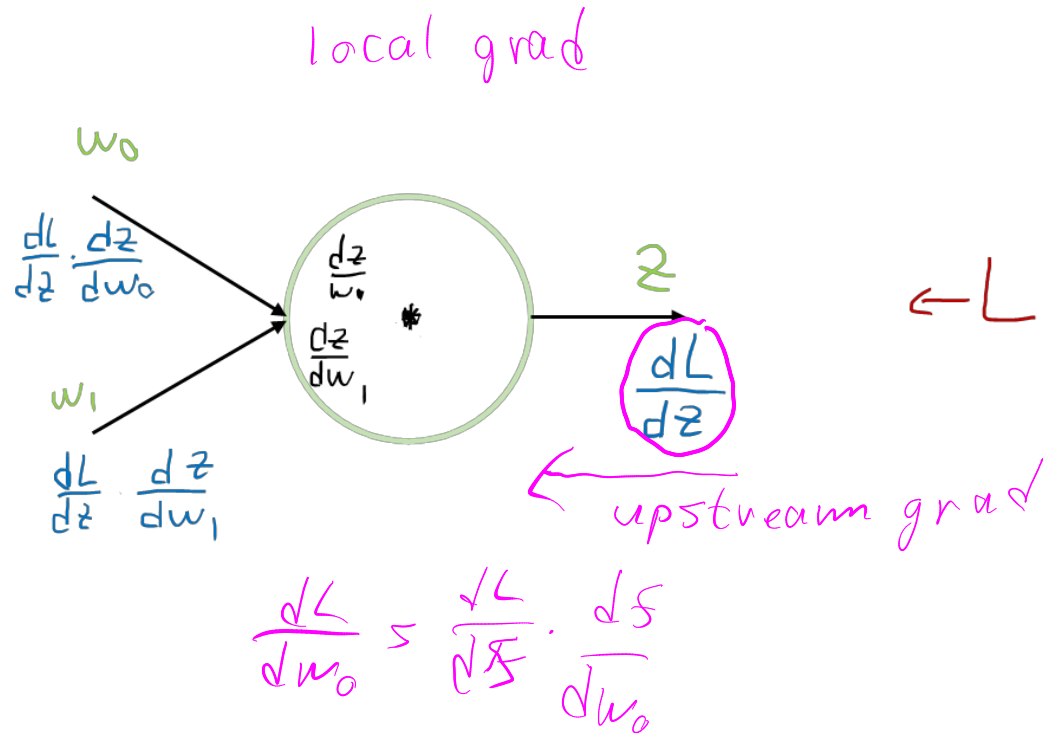
$$f(g(x))$$

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

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

А зачем мы все это считали?

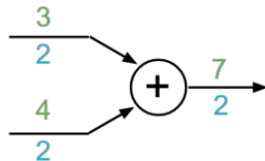
# Общая схема вычисления градиента



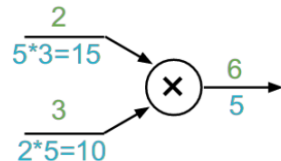
# Уточнения



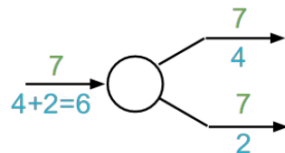
**add** gate: gradient distributor



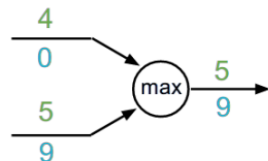
**mul** gate: "swap multiplier"



**copy** gate: gradient adder



**max** gate: gradient router

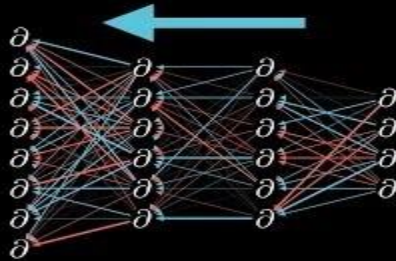




Если ничего не понятно...



## Backpropagation calculus



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

