



Phystech@DataScience

Введение в нейронные сети

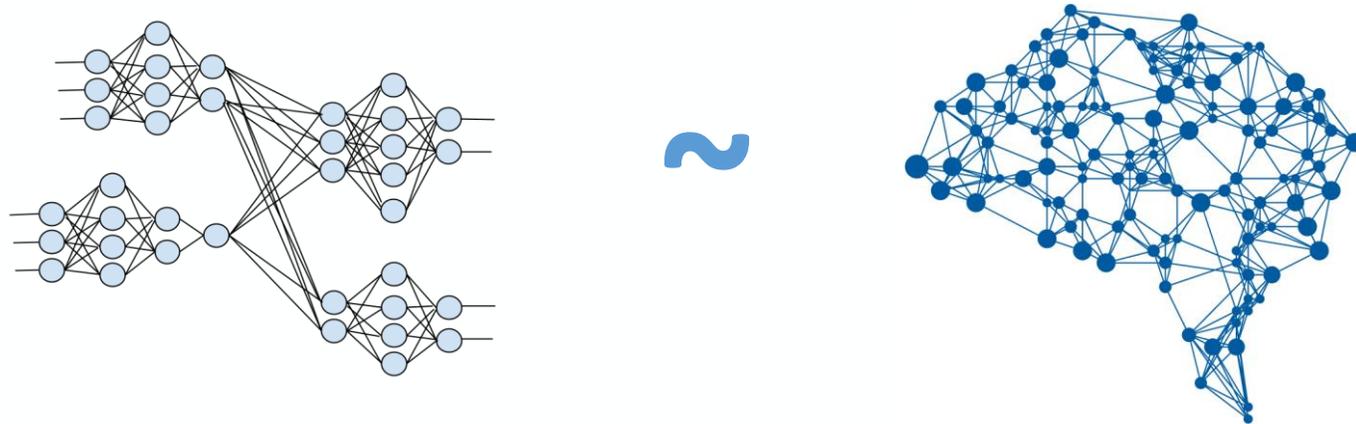
Лектор — Логинов Артём



Что такое ~~эти ваши~~ нейронные сети?

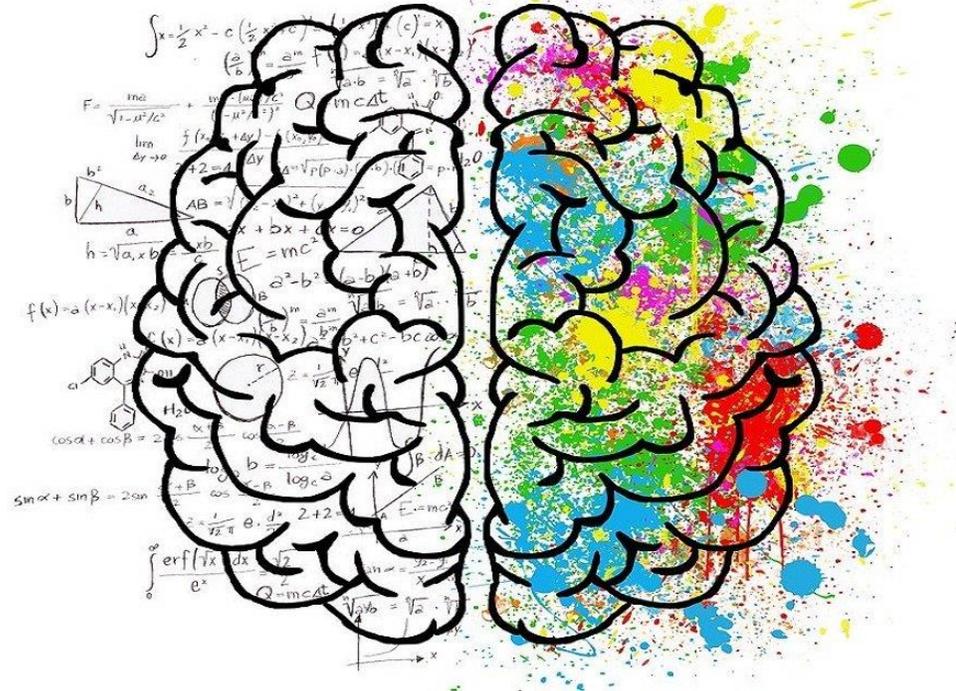
Нейронная сеть — математическая модель,
а также ее программное или аппаратное воплощение,
построенная по принципу организации и функционирования
биологических нейронных сетей — сетей нервных клеток живого организма.

Это понятие возникло при изучении процессов, протекающих в мозге.





Мозг



Мозг – это

- Система обработки информации
- Сложный, нелинейный, параллельный компьютер

Решает множество сложных задач из области распознавания образов, обработки сигналов и прочее



Примеры

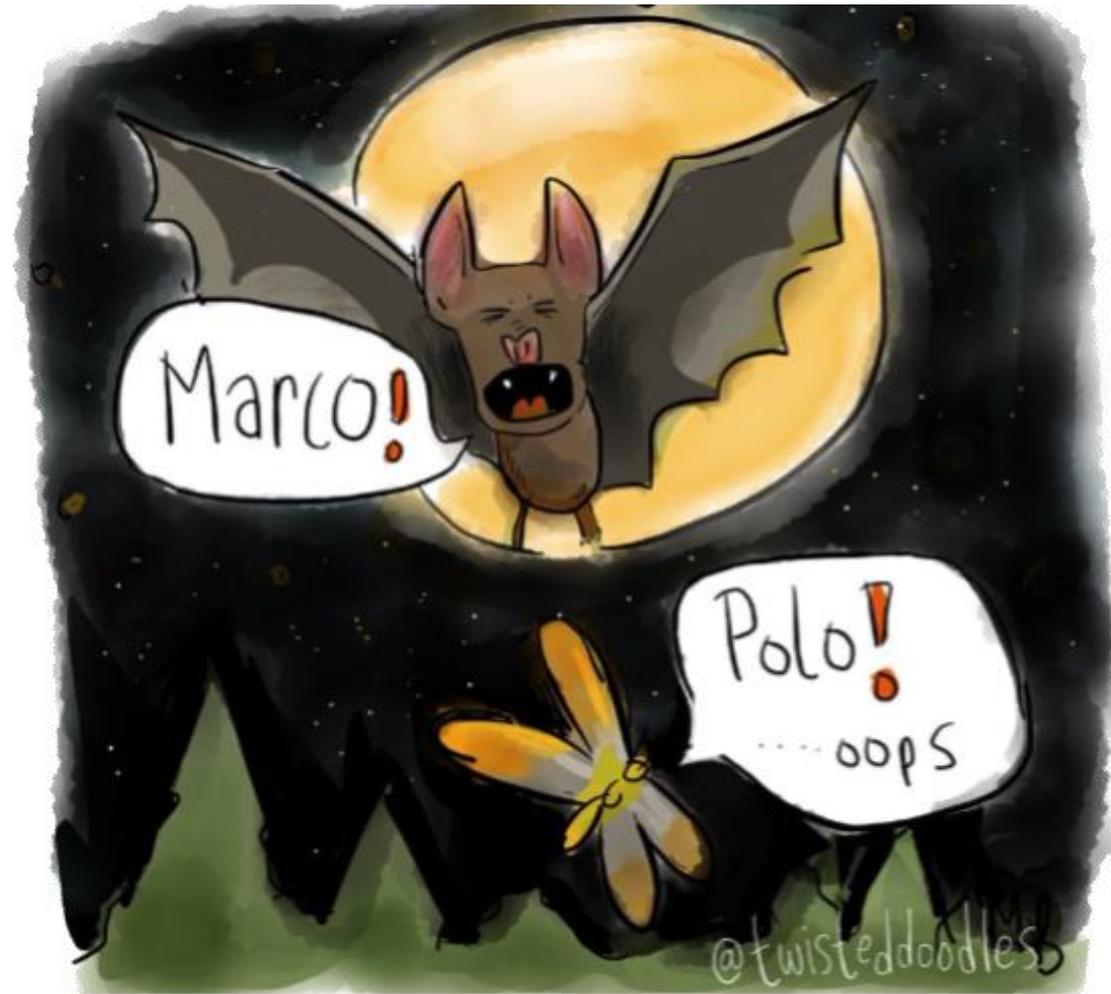


Задача распознавания
знакомого лица в толпе
займет **0.1-0.2 секунды**



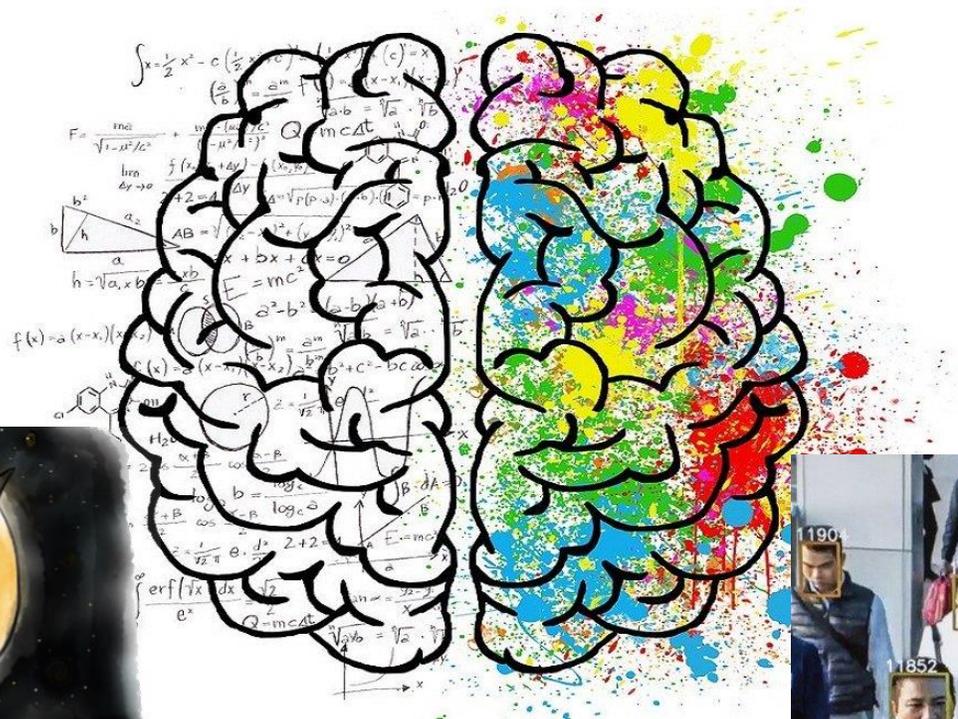
Примеры

- Эхо-локация:
летучая мышь получает информацию про положение, размер, скорость объекта и др..



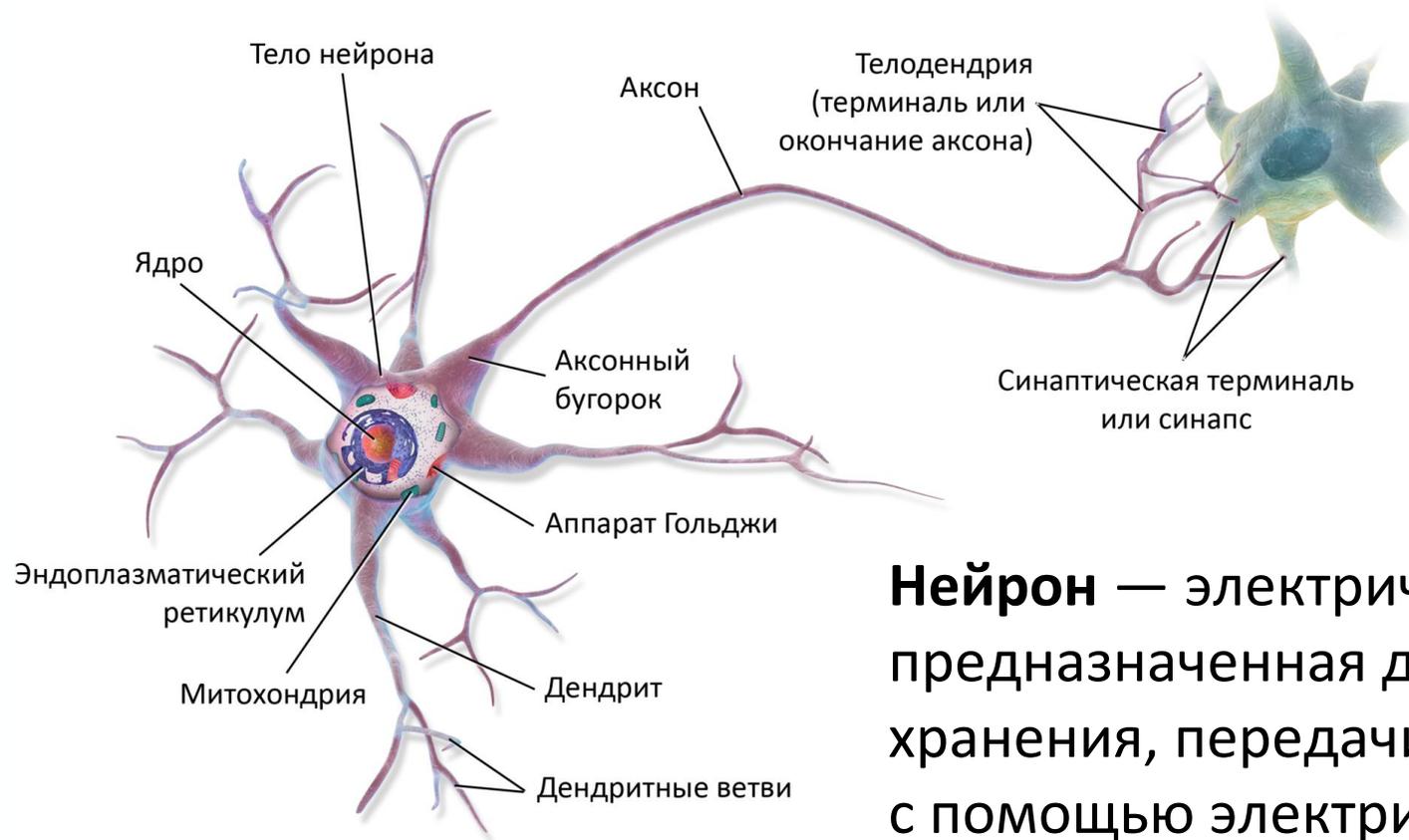


Почему мы справляемся со столь сложными задачами?





Биологический нейрон

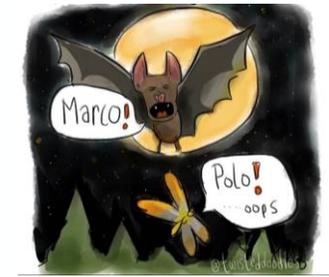


Структурно-функциональная единица нервной системы — это нейрон.

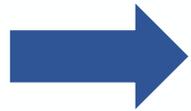
Нейрон — электрически возбудимая клетка, предназначенная для приёма, обработки, хранения, передачи и вывода информации с помощью электрических и химических сигналов.



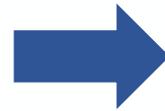
Почему мы справляемся со столь сложными задачами?



Опыт



Нейронная сеть



Способность решать сложные задачи

Обучение
/ training

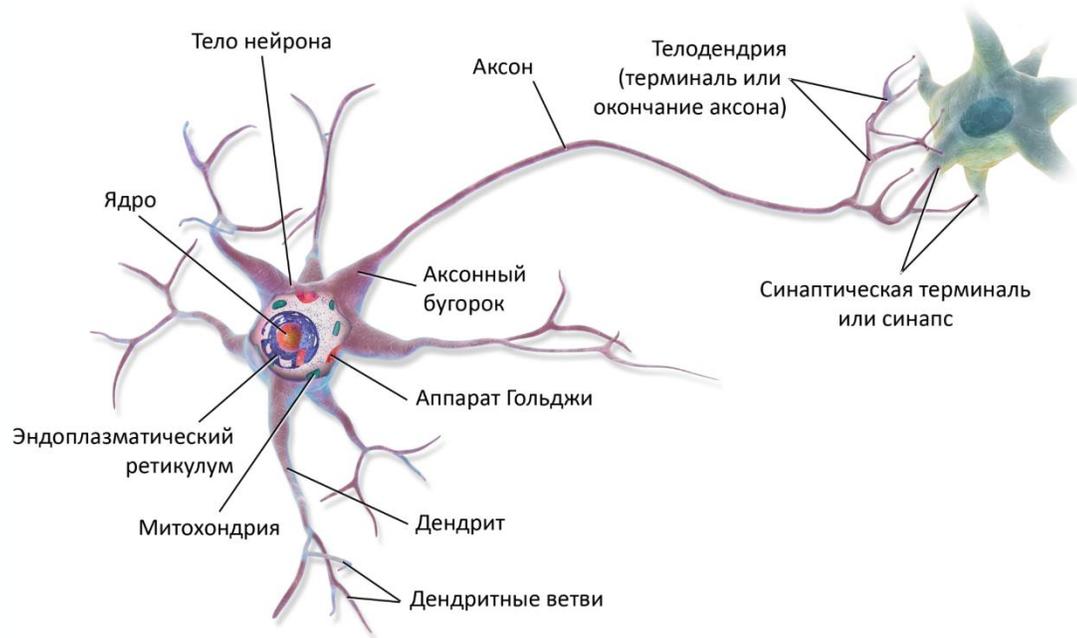
Применение
/ inference



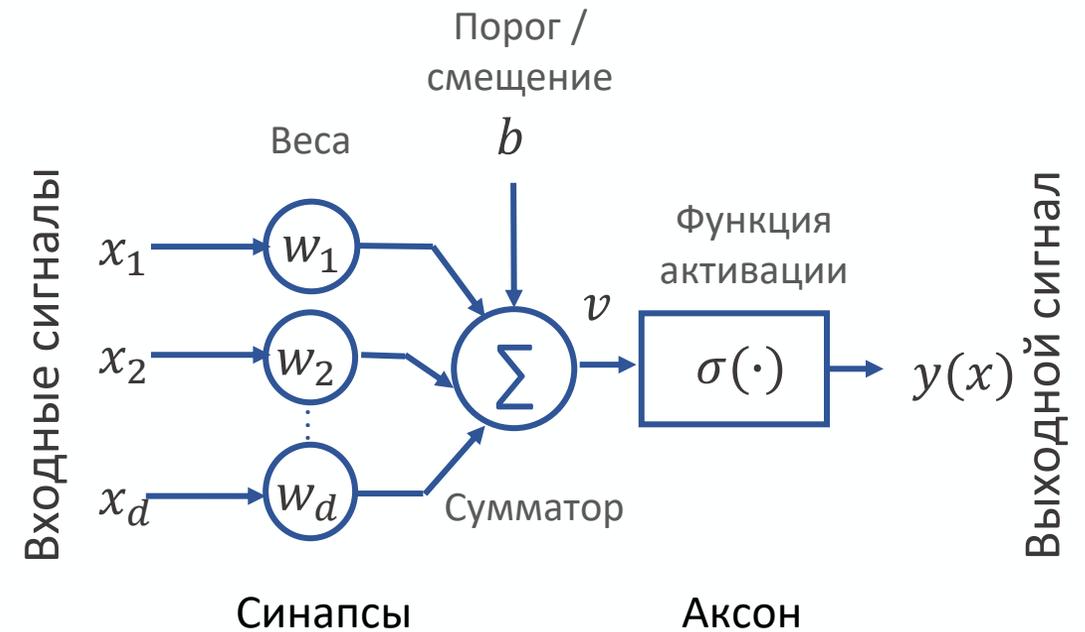


Бионика: нейронные сети

Биологический нейрон



Искусственный нейрон





Историческая справка

1943 г. - Мак-Каллок и Питц

- Модель нейрона как «*всё или ничего*»
- Принципиальная возможность выполнить любые вычисления сетью нейронов

1949 г. – Хебб

- Открытие принципа формирования нейронных связей, взаимодействия нейронов

1958 г. – Розенблатт

- Открытие «персептрона» - устройства, моделирующего процесс человеческого восприятия



Области применения

Распознавание лиц / Face detection

Цель — определить личность человека по снимку.

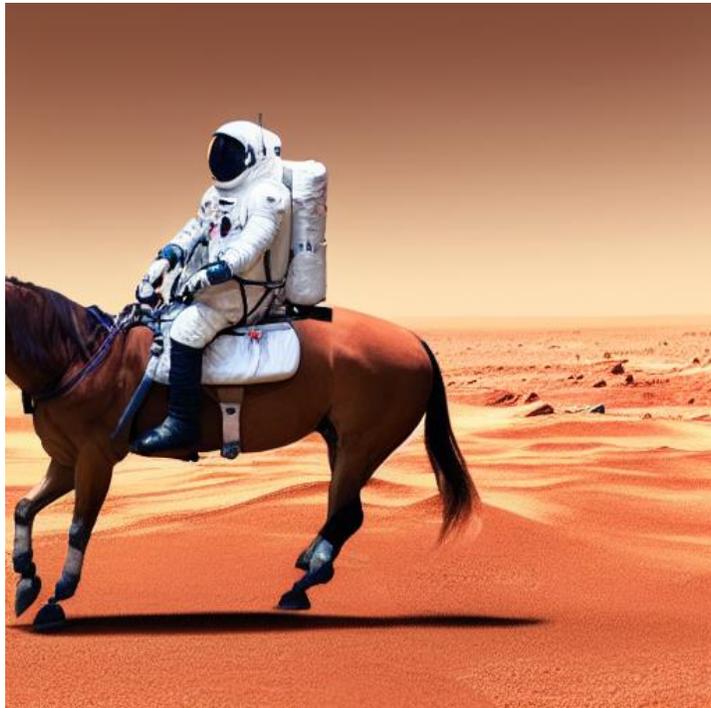




Области применения

Генерация изображений / Image generation

Цель — создание уникальных изображений.

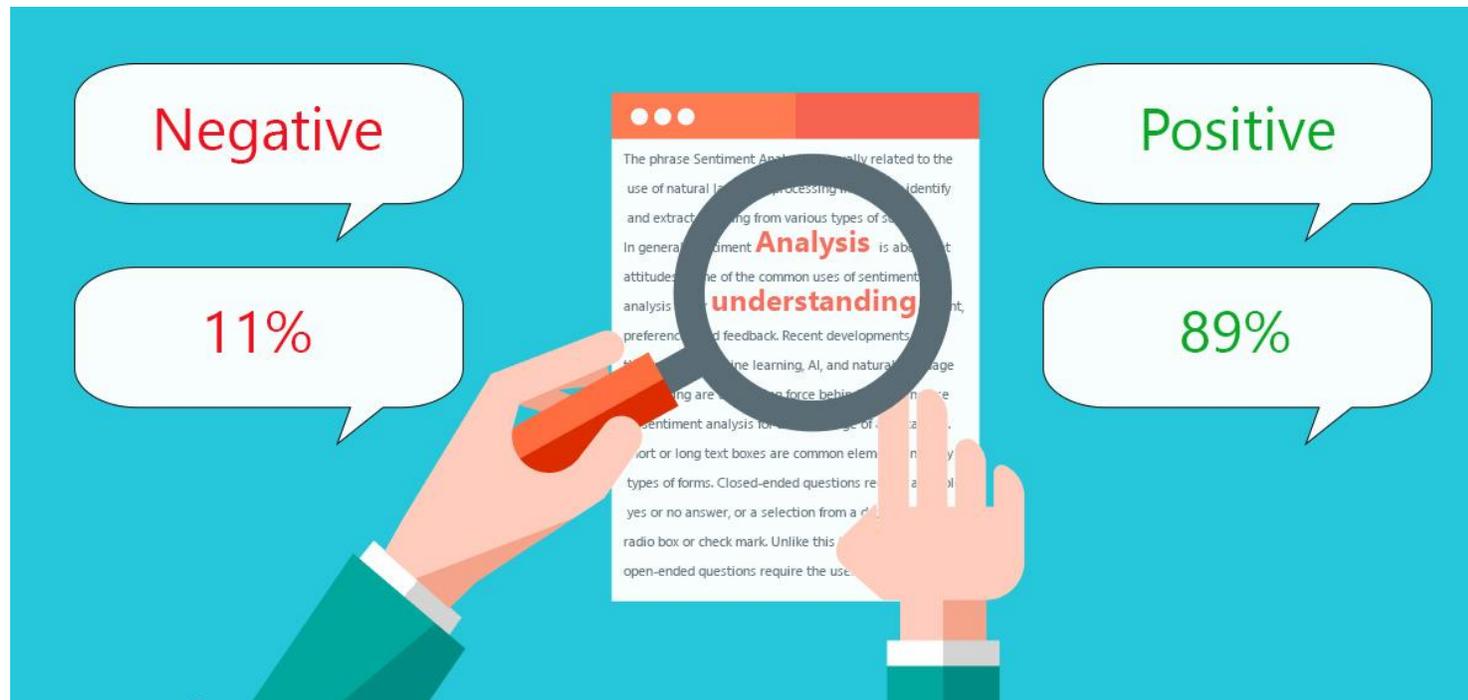




Области применения

Определение тональности текста / Sentiment analysis

Цель — по определить является текст ПОЗИТИВНЫМ или НЕГАТИВНЫМ.





Области применения

Машинный перевод / Machine Language translation

Цель — перевести текст на одном языке на другой язык.

The screenshot displays the Yandex Translate web interface. At the top, the text 'Яндекс Переводчик' is followed by navigation options: 'Текст', 'Сайты', 'Документы', and 'Картинки'. On the right, there is a 'Для бизнеса' button and a user profile icon. The main area shows a language selection bar with 'АНГЛИЙСКИЙ' selected on the left and 'РУССКИЙ' on the right, separated by a double-headed arrow. Below this, the input text 'neural networks are so cool!' is shown in the left pane, and the translated text 'нейронные сети - это так круто!' is shown in the right pane. At the bottom of the interface, there are icons for audio playback, a character count '28 / 10000', a keyboard icon, a speaker icon, and a row of utility icons including 'Перевести в Google', a bookmark icon, a share icon, and thumbs up/down icons.



Области применения

Генерация текста / text generation

Цель — сгенерировать связный текст по соответствующей тематике.

Proof. Omitted. □

Lemma 0.1. *Let \mathcal{C} be a set of the construction.*
Let \mathcal{C} be a gerber covering. Let \mathcal{F} be a quasi-coherent sheaves of \mathcal{O} -modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

Proof. This is an algebraic space with the composition of sheaves \mathcal{F} on $X_{\acute{e}tate}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where \mathcal{G} defines an isomorphism $\mathcal{F} \rightarrow \mathcal{F}$ of \mathcal{O} -modules. □

Lemma 0.2. *This is an integer Z is injective.*

Proof. See Spaces, Lemma ??.

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $U \subset X$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \rightarrow Y' \rightarrow Y \rightarrow Y' \times_X Y \rightarrow X.$$

be a morphism of algebraic spaces over S and Y .

Proof. Let X be a nonzero scheme of X . Let X be an algebraic space. Let \mathcal{F} be a quasi-coherent sheaf of \mathcal{O}_X -modules. The following are equivalent

- (1) \mathcal{F} is an algebraic space over S .
- (2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram

is a limit. Then \mathcal{G} is a finite type and assume S is a flat and \mathcal{F} and \mathcal{G} is a finite type f_* . This is of finite type diagrams, and

- the composition of \mathcal{G} is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

Proof. We have see that $X = \text{Spec}(R)$ and \mathcal{F} is a finite type representable by algebraic space. The property \mathcal{F} is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U . □

Proof. This is clear that \mathcal{G} is a finite presentation, see Lemmas ??.

A reduced above we conclude that U is an open covering of \mathcal{C} . The functor \mathcal{F} is a “field

$$\mathcal{O}_{X,x} \rightarrow \mathcal{F}_{\bar{x}}^{-1}(\mathcal{O}_{X_{\acute{e}tate}}) \rightarrow \mathcal{O}_{X_t}^{-1} \mathcal{O}_{X_\lambda}(\mathcal{O}_{X_t}^{\bar{v}})$$

is an isomorphism of covering of \mathcal{O}_{X_t} . If \mathcal{F} is the unique element of \mathcal{F} such that X is an isomorphism.

The property \mathcal{F} is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme \mathcal{O}_X -algebra with \mathcal{F} are opens of finite type over S .

If \mathcal{F} is a scheme theoretic image points. □

If \mathcal{F} is a finite direct sum \mathcal{O}_{X_λ} is a closed immersion, see Lemma ??.

This is a sequence of \mathcal{F} is a similar morphism.



Области применения

Текстовые боты / text bots

GPT-4

You will be given a name of an object (such as Car, Chair, Elephant) and a letter in the alphabet. Your goal is to first produce a 1-line description of how that object can be combined with the letter in an image (for example, for an elephant and the letter J, the trunk of the elephant can have a J shape, and for the letter A and a house, the house can have an A shape with the upper triangle of the A being the roof). Following the short description, please create SVG code to produce this (in the SVG use shapes like ellipses, triangles etc and polygons but try to defer from using quadratic curves).

Round 1: A pig and the letter H.

Description: The pig can be lying on its back with its four legs forming the two vertical bars of the H, and its curly tail forming the horizontal bar.



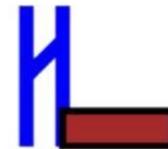
Round 2: A house and the letter U.

Description: A house with a U shape, where the two vertical lines of the U are the walls and the horizontal line is the floor. The roof is a triangle above the U.



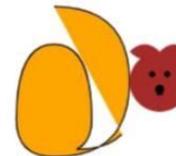
Round 3: A chair and the letter K.

Description: The back of the chair can have a K shape, with two vertical lines and a diagonal line connecting them. The seat of the chair can be a rectangle.



Round 4: A dog and the letter Q.

Description: The dog is sitting with its tail curled around its body, forming a Q shape. The Q is filled with orange and has a black outline.



Round 5: A car and the letter E.

Description: The car has an E shape on its front bumper, with the horizontal lines of the E being lights and the vertical line being the license plate.





Области применения

Распознавание речи / speech recognition

Цель — перевести человеческую речь в текст.

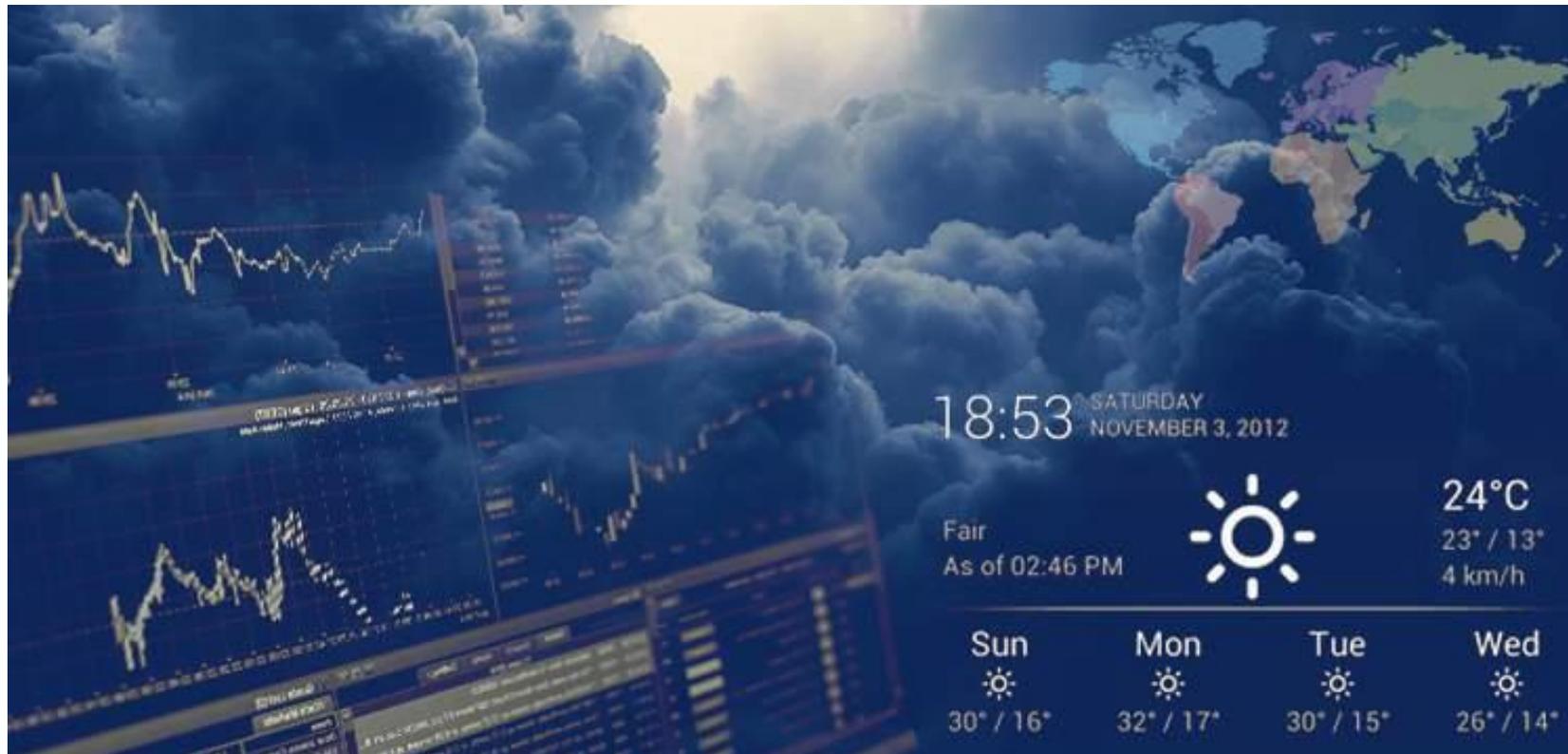




Области применения

Прогноз погоды / whether forecasting

Цель — предсказать температуру, осадки, скорость и направление ветра и тд.

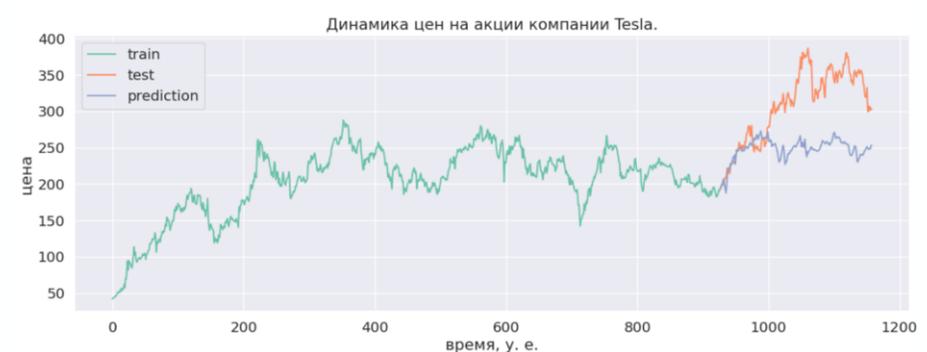
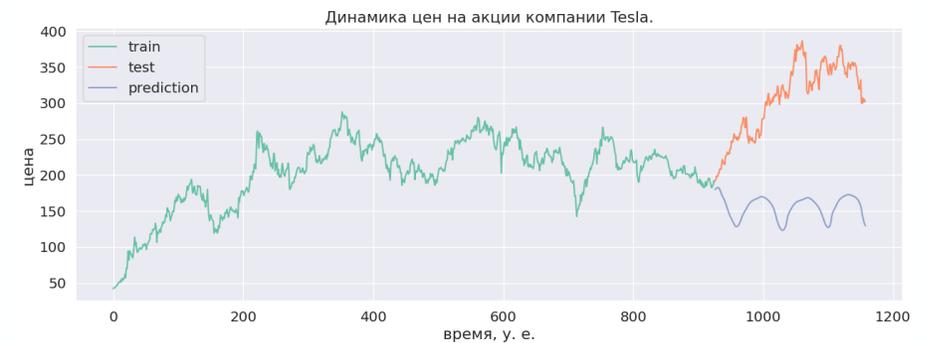
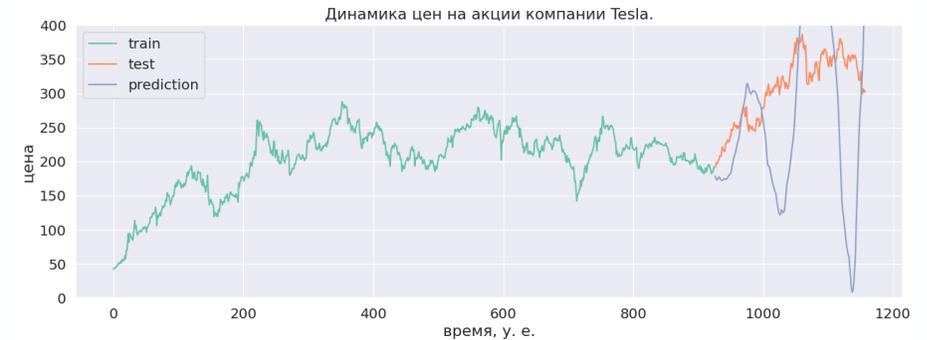
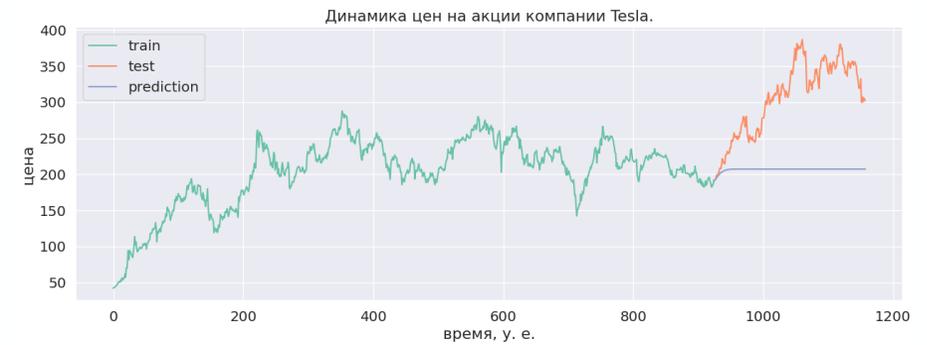




Области применения

Предсказание продаж / sales prediction

Цель — предсказать продажи товаров на некоторый период времени вперед.

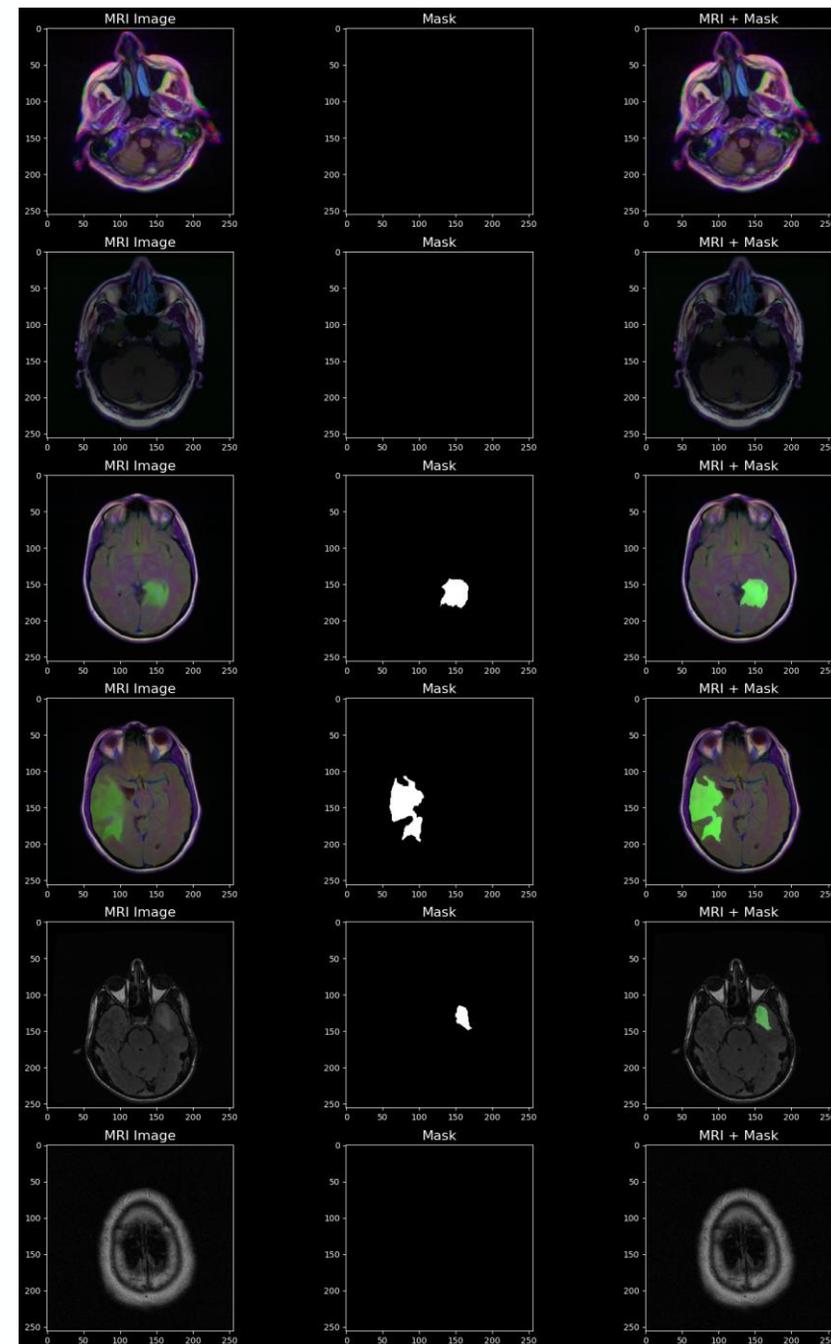
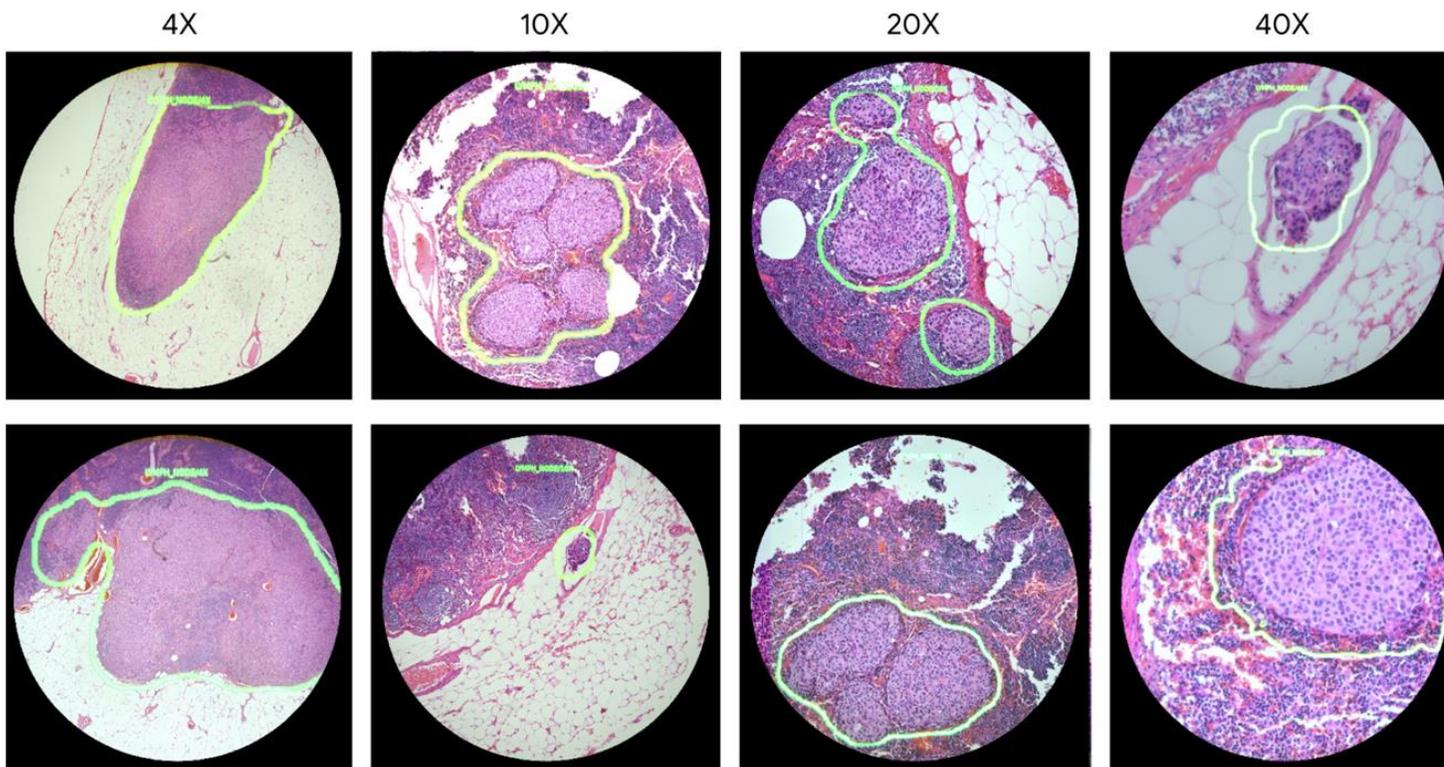




Области применения

Сегментация опухоли / Tumor segmentation

Цель — для каждого пикселя определить принадлежит ли он опухоли или нет.

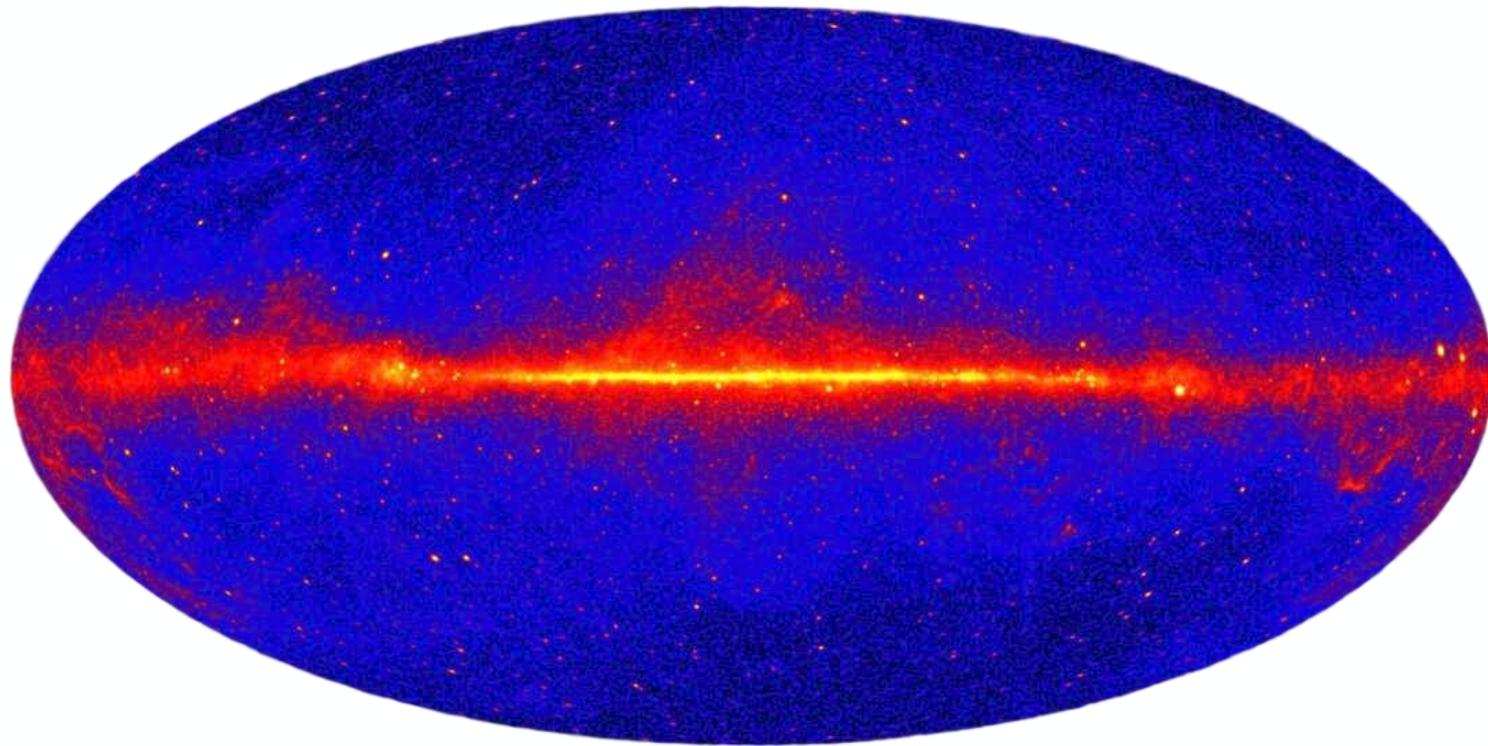




Области применения

Поиск точечных источников на изображениях телескопа Fermi

Цель — выдать координаты точечных источников.



Области применения

Работа с ДНК как с текстом

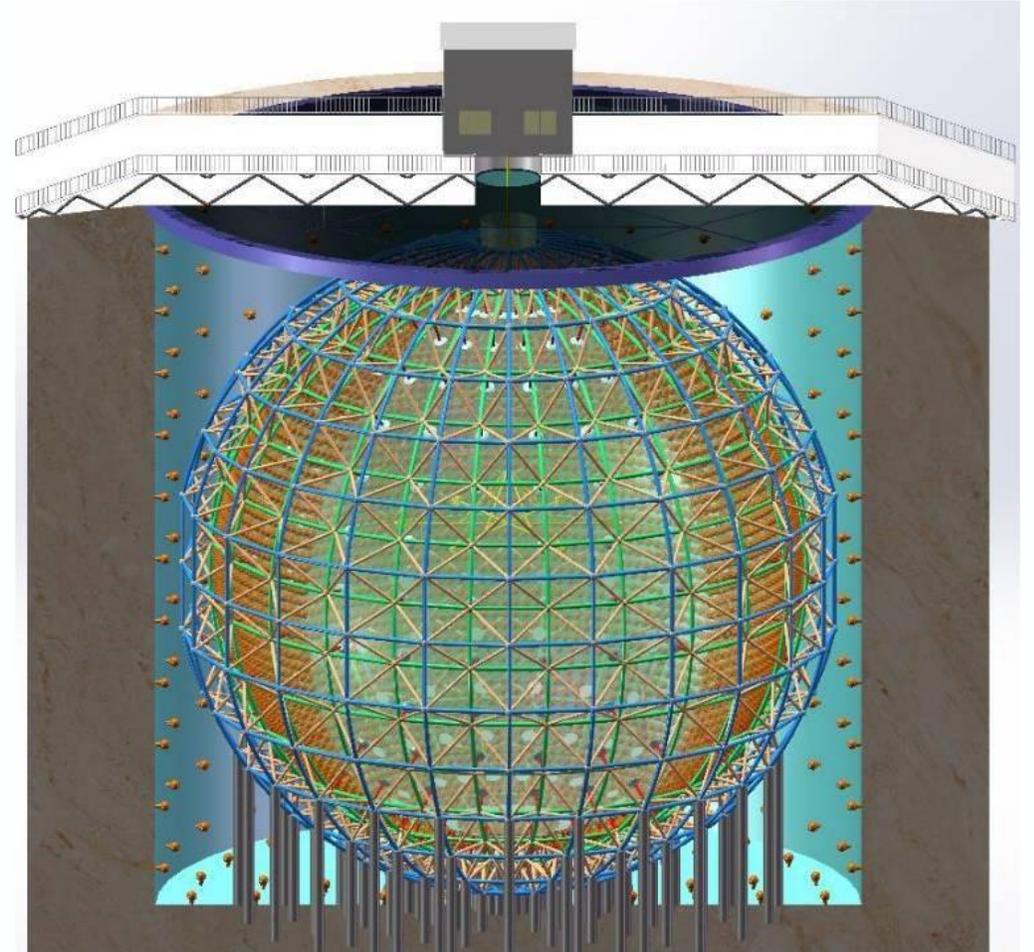
- Структурная аннотация
- Функциональная аннотация
- Предсказания сайтов связывания транскрипционного фактора



Области применения

Применение в физике высоких энергий:

- отделение событий, содержащих b -кварки на Большом Адронном Коллайдере;
- идентификация распадов τ -лептона;
- разделение кварк-глюонных струй;
- восстановление параметров нейтрино в эксперименте JUNO;
- аппроксимация сложных нелинейных зависимостей;
- и др.



Области применения

Предсказание структуры белка.

Цель: по последовательности ДНК
определить структуру белка.

Предсказание связывания белков/соединений.

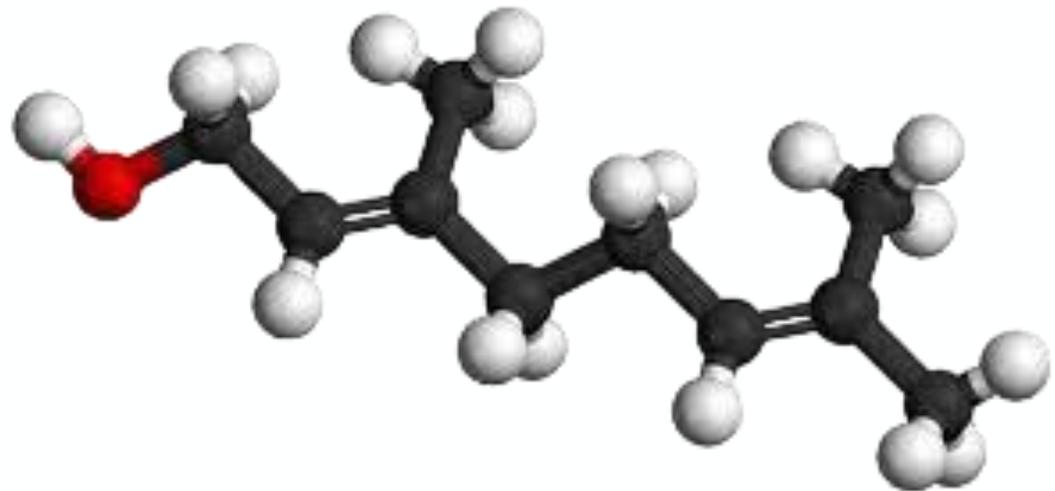
Применение: поиск новых лекарств.



Области применения

Предсказание свойств химических соединений.

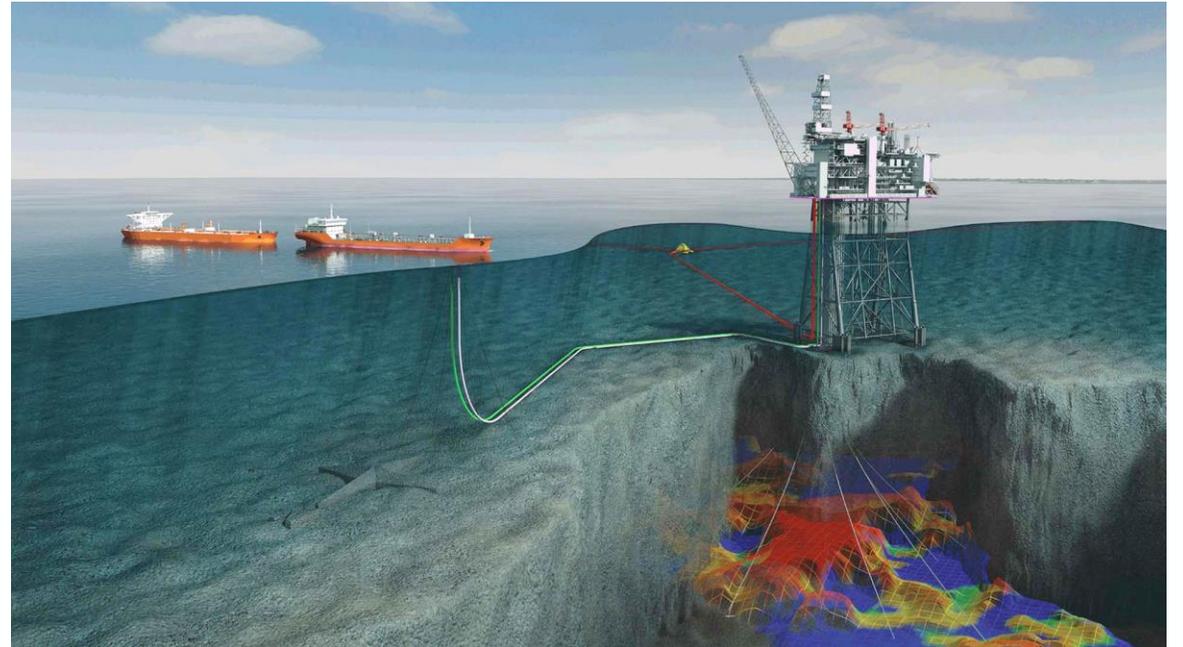
Цель: предсказать растворимость и др. свойства.



Области применения

Применение в нефтегазовой отрасли:

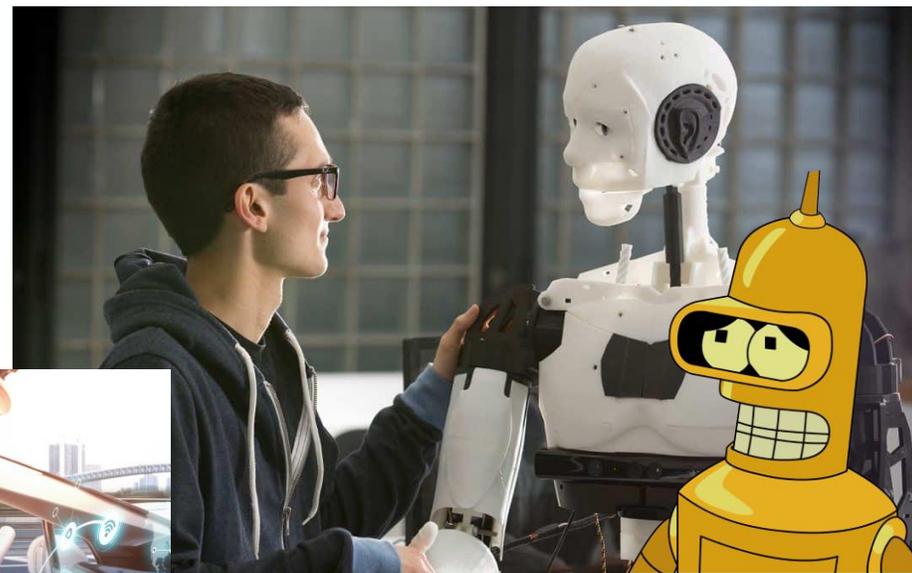
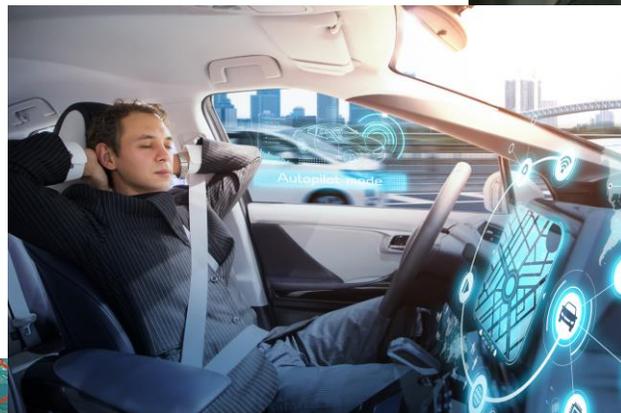
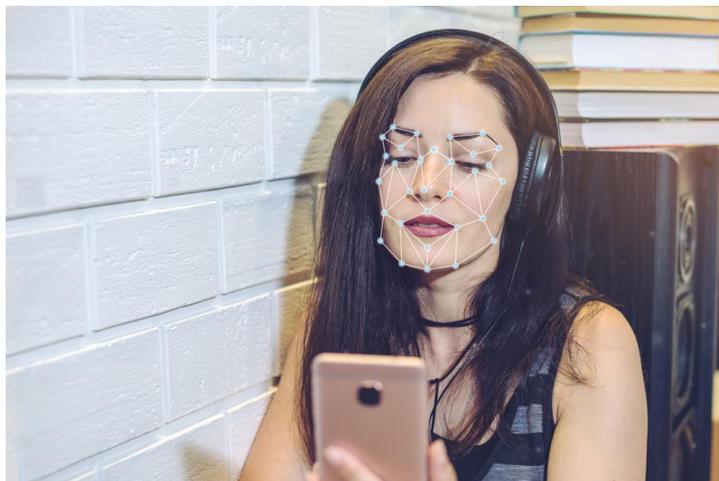
- геофизическое исследование и интерпретация данных;
- прогнозирование и оптимизация добычи;
- мониторинг и управление скважинами;
- автоматизация процессов бурения;
- и др.





Области применения

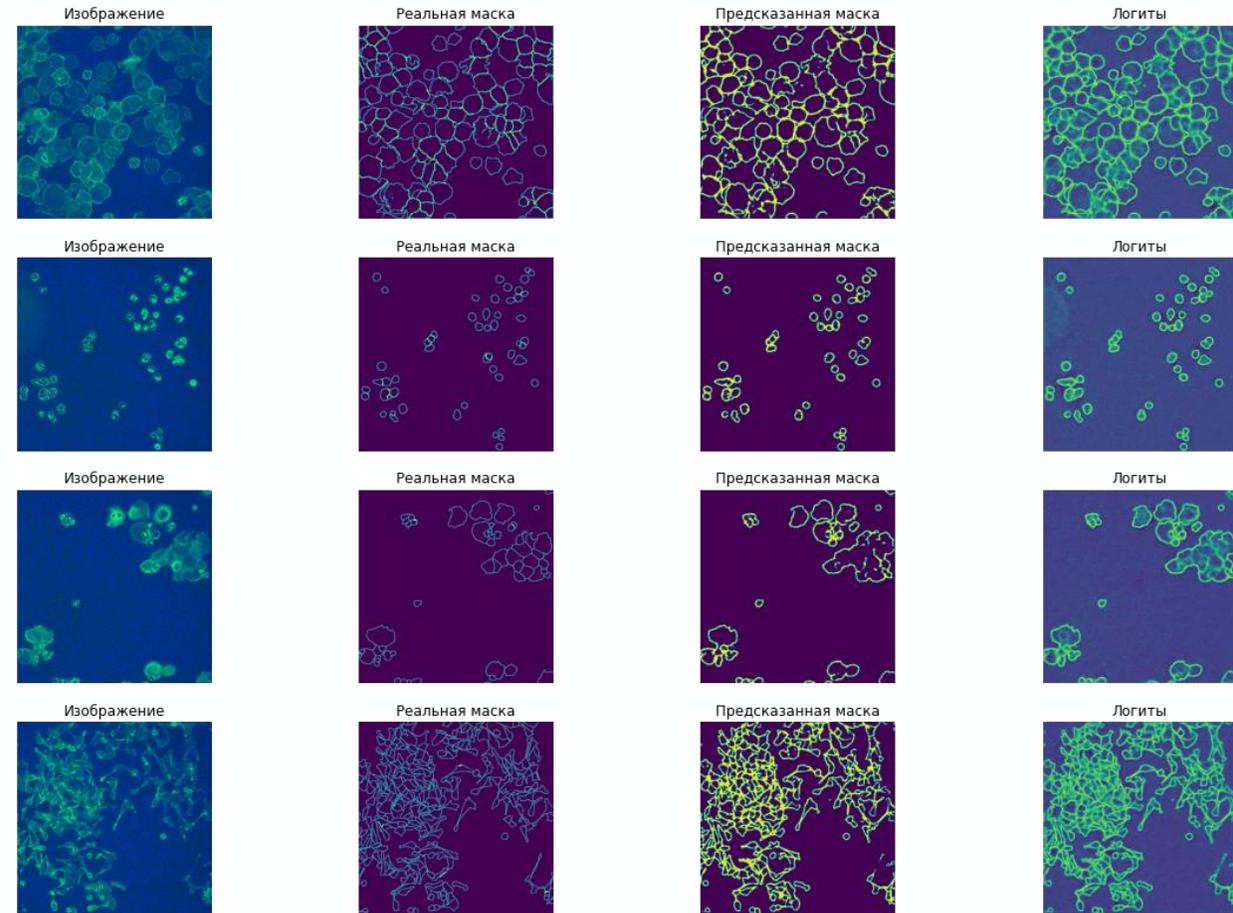
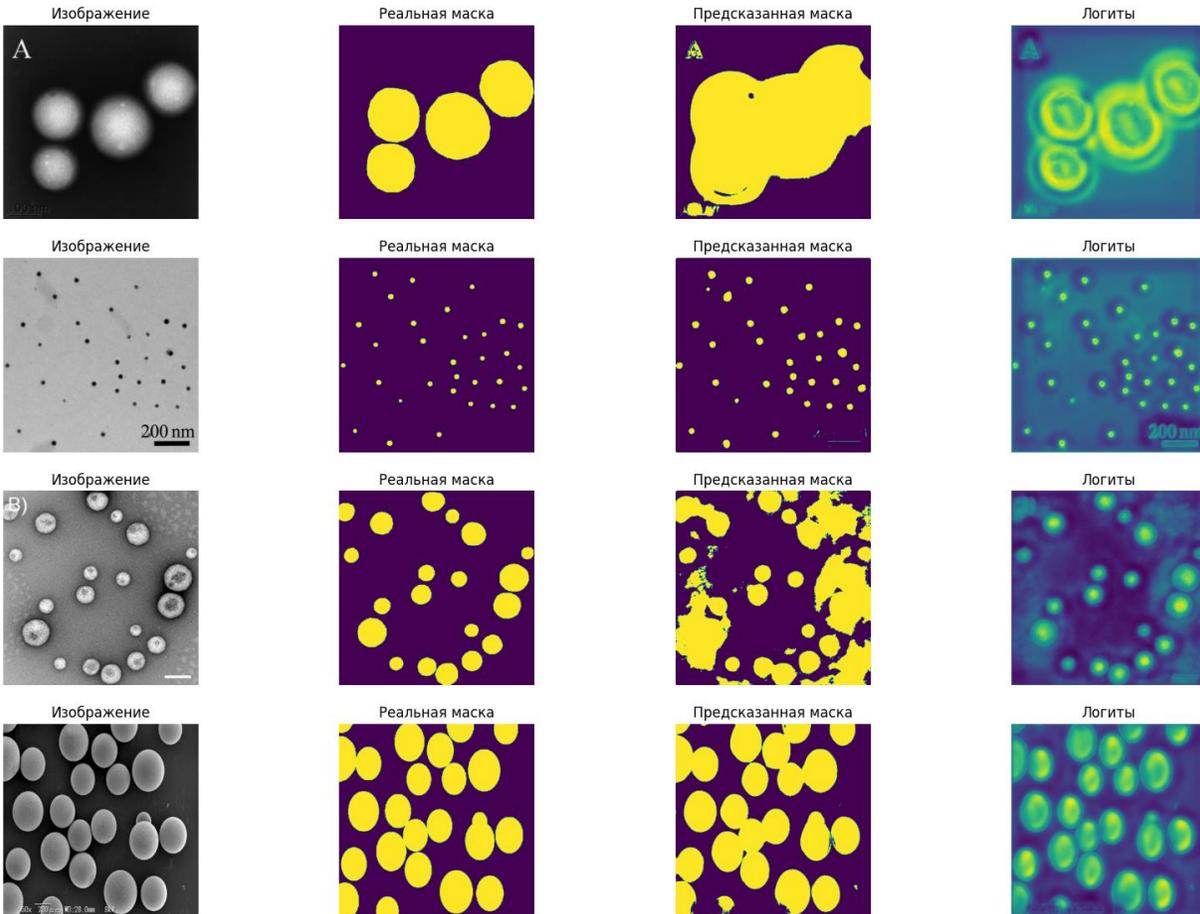
И многое другое...



Немного из осеннего курса

Выделение границ объектов на «снимках» электронного микроскопа

Выделение границ клеток на снимках микроскопа



Немного из осеннего курса...

```
def conv_bn_relu(in_channels, out_channels, kernel=3, stride=1, padding=1):
    net = nn.Sequential(
        nn.Conv2d(in_channels, out_channels,
                  kernel_size=kernel, stride=stride, padding=padding),
        nn.BatchNorm2d(num_features=out_channels),
        nn.ReLU(inplace=True)
    )
    return net

class Stacked2ConvsBlock(nn.Module):
    def __init__(self, in_channels, out_channels):
        super(Stacked2ConvsBlock, self).__init__()
        self.blocks = nn.Sequential(
            conv_bn_relu(in_channels, out_channels),
            conv_bn_relu(out_channels, out_channels)
        )

    def forward(self, net):
        net = self.blocks(net)
        return net

class UpSamplingBlock(nn.Module):
    def __init__(self, in_channels, out_channels):
        super(UpSamplingBlock, self).__init__()

        # Понижаем число каналов
        self.upsample = nn.ConvTranspose2d(
            in_channels, in_channels, kernel_size=2, stride=2)

        # Ставим с симметричным слоем из левой половины "U".
        # Число каналов входной карты при этом удваивается.
        self.convolve = Stacked2ConvsBlock(2 * in_channels, out_channels)

    def forward(self, left_net, right_net):
        right_net = self.upsample(right_net)
        net = torch.cat([left_net, right_net], dim=1)
        net = self.convolve(net)
        return net

class DownSamplingBlock(nn.Module):
    def __init__(self, in_channels, out_channels):
        super(DownSamplingBlock, self).__init__()
        self.blocks = nn.Sequential(
            nn.MaxPool2d(2, 2),
            Stacked2ConvsBlock(in_channels, out_channels)
        )

    def forward(self, net):
        return self.blocks(net)

def conv_bn_relu(in_channels, out_channels, kernel=3, stride=1, padding=1):
    net = nn.Sequential(
        nn.Conv2d(in_channels, out_channels,
                  kernel_size=kernel, stride=stride, padding=padding),
        nn.BatchNorm2d(num_features=out_channels),
        nn.ReLU(inplace=True)
    )
    return net
```

```
def forward(self, left_net, right_net):
    right_net = self.upsample(right_net)
    net = torch.cat([left_net, right_net], dim=1)
    net = self.convolve(net)
    return net

class DownSamplingBlock(nn.Module):
    def __init__(self, in_channels, out_channels):
        super(DownSamplingBlock, self).__init__()
        self.blocks = nn.Sequential(
            nn.MaxPool2d(2, 2),
            Stacked2ConvsBlock(in_channels, out_channels)
        )

    def forward(self, net):
        return self.blocks(net)

class Unet(nn.Module):
    def __init__(self):
        super(Unet, self).__init__()
        self.init_conv = Stacked2ConvsBlock(3, 64)

        self.downsample_1 = DownSamplingBlock(64, 128)
        self.downsample_2 = DownSamplingBlock(128, 256)
        self.downsample_3 = DownSamplingBlock(256, 512)
        self.downsample_4 = DownSamplingBlock(512, 1024)

        # В середине есть блок без пары с 1024 каналами
        # с ним конкатенировать некого, потому просто свернем его
        self.upconv = Stacked2ConvsBlock(1024, 512)

        # Подъем. Аналогично.
        self.upsample_1 = UpSamplingBlock(512, 256)
        self.upsample_2 = UpSamplingBlock(256, 128)
        self.upsample_3 = UpSamplingBlock(128, 64)

        # Чтобы учесть входной слой после самой первой свертки и согласовать размерности
        self.upsample_4 = UpSamplingBlock(64, 64)

        # Предсказание
        self.agg_conv = nn.Conv2d(64, 1, kernel_size=1)

    def forward(self, x):
        net0 = self.init_conv(x) # 3 --> 64

        net1 = self.downsample_1(net0) # 64 --> 128
        net2 = self.downsample_2(net1) # 128 --> 256
        net3 = self.downsample_3(net2) # 256 --> 512
        net = self.downsample_4(net3) # 512 --> 1024

        net = self.upconv(net) # 1024 --> 512

        net = self.upsample_1(net3, net) # 512 --> 256
        net = self.upsample_2(net2, net) # 256 --> 128
        net = self.upsample_3(net1, net) # 128 --> 64
        net = self.upsample_4(net0, net) # 64 --> 64

        net = self.agg_conv(net) # 64 --> 1

    return net
```

А вы нейронной сетью
рыбов ловите?

Нет, мы ей на рыбков
зарабатываем.

Красиво...

Проверка активности – ВОПРОС В БОТ!
Напишите 2 примера применения нейронок, не упомянутые на лекции. Желательно в науке.
Решение домашек ДИЯ не в счёт.



Модель нейрона

Обозначим

$x = (x_1, \dots, x_d)^T \in \mathbb{R}^d$ — один объект, где x_1, \dots, x_d — признаки;

$w = (w_1, \dots, w_d)^T \in \mathbb{R}^d$ — вектор весов;

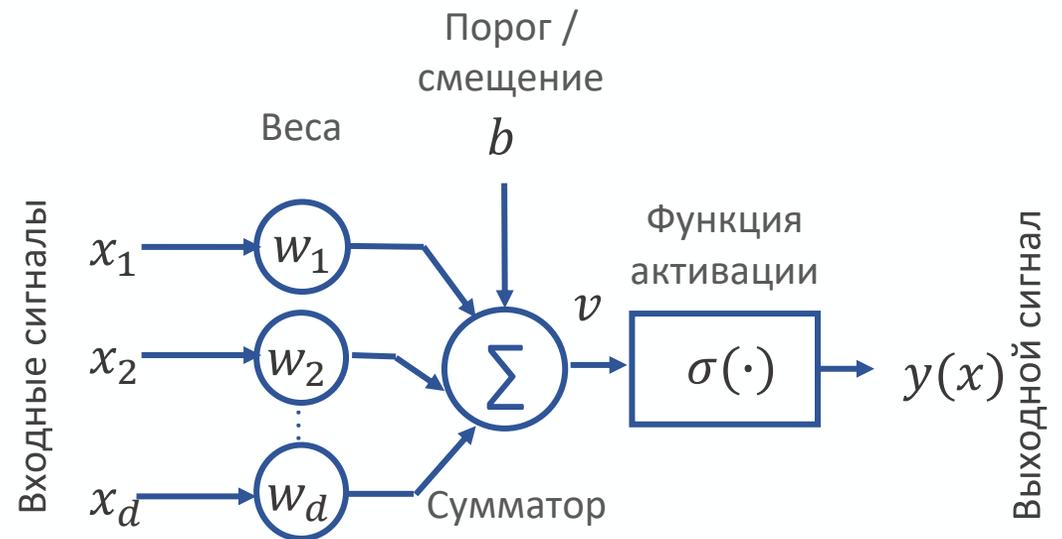
$b \in \mathbb{R}$ — смещение.

Выход нейрона —

$$y(x) = \sigma(\langle x, w \rangle + b) = \sigma\left(\sum_{j=1}^d w_j x_j + b\right)$$

где $\sigma(\cdot)$ — некоторая кус.-дифф. функция.

Назовем ее **функцией активации**.

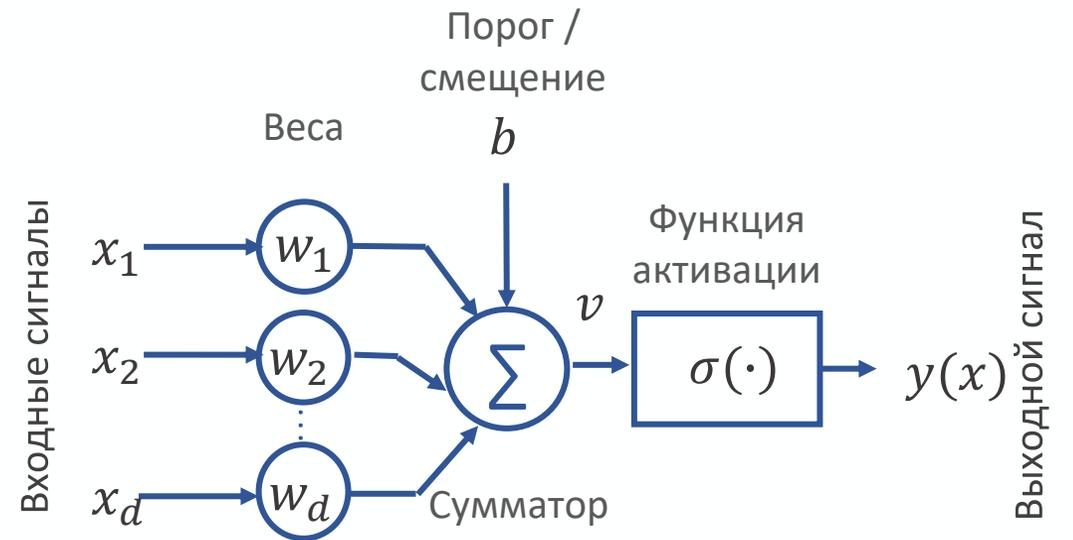




Что-то знакомое...

На что похожа эта формула?

$$y(x) = \sigma(\langle x, w \rangle + b) = \sigma\left(\sum_{j=1}^d w_j x_j + b\right)$$





Что-то знакомое...

На что похожа эта формула?

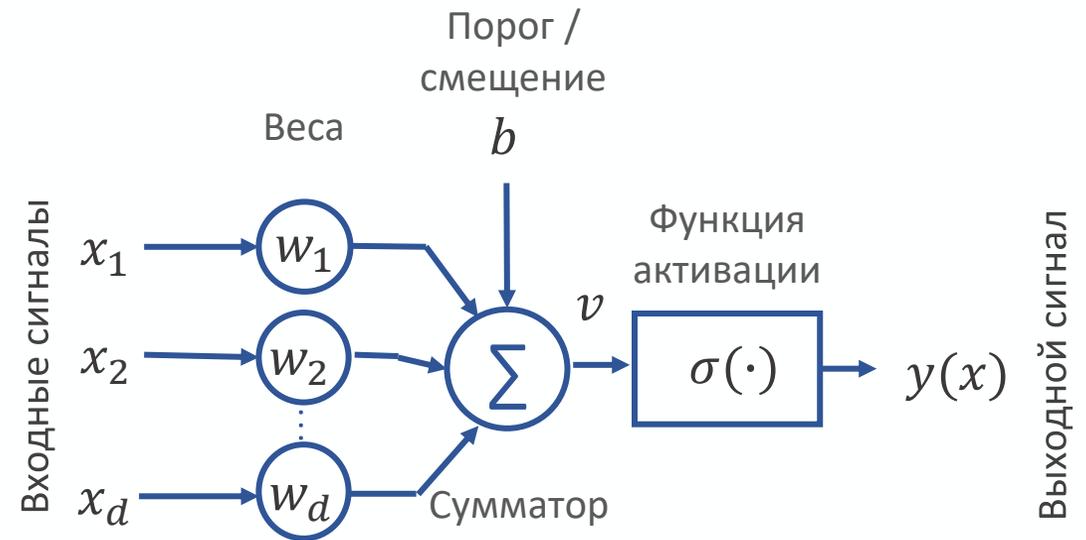
$$y(x) = \sigma(\langle x, w \rangle + b) = \sigma\left(\sum_{j=1}^d w_j x_j + b\right)$$

Линейная регрессия для 1 элемента:

$$y(x) = \langle x, w \rangle + b = \sigma(\langle x, w \rangle + b) = \sigma\left(\sum_{j=1}^d x_j w_j + b\right), \quad \text{где } \sigma(z) = z \text{ — линейная функция.}$$

Логистическая регрессия для 1 элемента:

$$y(x) = \sigma(\langle x, w \rangle + b) = \sigma\left(\sum_{j=1}^d x_j w_j + b\right), \quad \text{где } \sigma(z) = \frac{1}{1+e^{-z}} \text{ — логистическая сигмоида.}$$





Решение задачи регрессии

Линейная регрессия: $\hat{y} = Xw + b$, т.е. $\hat{y}_i = \sum_{j=1}^d x_{ij}w_j + b$,

где $X = (x_{ij})_{ij}$ — матрица входных данных, $i \in \{1, \dots, n\}$, $j \in \{1, \dots, d\}$

$\hat{y} = (\hat{y}_1, \dots, \hat{y}_d)^T$ — вектор предсказания,

$w = (w_1, \dots, w_d)^T$ — вектор весов, b — сдвиг.

Задачу можно решить аналитически. А можно с помощью **градиентного спуска**.

Зададим функцию, которую хотим минимизировать:

$$L = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2 \text{ — MSE (Mean Squared Error)}$$

Пусть $\theta = (w, b)$, тогда оптимизация будет следующей:

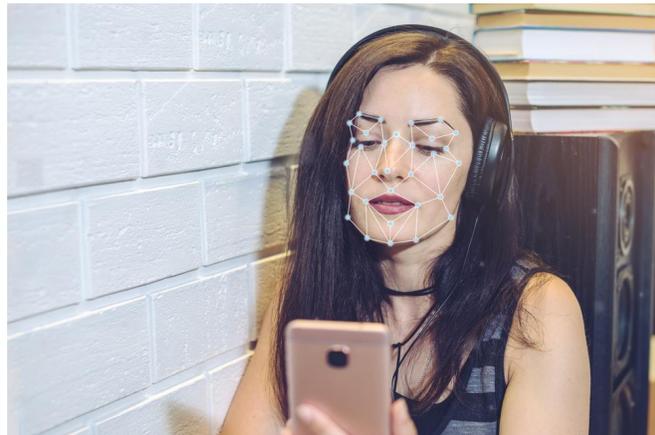
$$\theta_t = \theta_{t-1} - \eta \nabla L(\theta_t), \text{ где } \eta \text{ — скорость обучения.}$$



Более сложные задачи

- У линейной и логистической регрессий ограниченная область применения.
- Для того, чтобы решить нелинейную задачу, нужно делать сложные преобразования с признаками.
- Один нейрон не справится со сложными задачами... 😞

Вспомним, что в нервной системе очень много нейронов.
Значит, будем использовать **больше нейронов!**



Proof. Omitted. □

Lemma 0.1. Let C be a set of the construction. Let \mathcal{C} be a gerbe covering. Let \mathcal{F} be a quasi-coherent sheaves of \mathcal{O} -modules. We have to show that

$$\mathcal{O}_{\mathcal{C}_X} = \mathcal{O}_X(\mathcal{L})$$

Proof. This is an algebraic space with the composition of sheaves \mathcal{F} on $X_{\text{étale}}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{ \text{morph}_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F}) \}$$

where \mathcal{G} defines an isomorphism $\mathcal{F} \rightarrow \mathcal{F}$ of \mathcal{O} -modules.

Lemma 0.2. This is an integer Z is injective.

Proof. See Spaces, Lemma ??.

Lemma 0.3. Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $U \subset X$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b: X \rightarrow Y' \rightarrow Y \rightarrow Y' \times_X Y \rightarrow X,$$

be a morphism of algebraic spaces over S and Y .

Proof. Let X be a nonzero scheme of X . Let X be an algebraic space. Let \mathcal{F} be a quasi-coherent sheaf of \mathcal{O}_X -modules. The following are equivalent

- (1) \mathcal{F} is an algebraic space over S .
- (2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram

$$\begin{array}{ccc} S & \xrightarrow{\quad} & \\ \downarrow \zeta & \xrightarrow{\quad} & \mathcal{O}_X \\ \text{gerbe} & \uparrow & \downarrow \\ \mathcal{O}' & \xrightarrow{\quad} & \mathcal{O} \\ \downarrow \mathcal{O}' & \xrightarrow{\quad} & \mathcal{O} \\ \mathcal{O}' & \xrightarrow{\quad} & \mathcal{O} \end{array}$$

$\text{Spec}(K_0)$ $\text{Mor}_{\text{sets}} \text{d}(\mathcal{O}_{X_{\text{étale}}}, \mathcal{G})$ X

is a limit. Then \mathcal{G} is a finite type and assume S is a flat and \mathcal{F} and \mathcal{G} is a finite type \mathcal{F} . This is of finite type diagrams, and

- the composition of \mathcal{G} is a regular sequence,
- \mathcal{O}_X is a sheaf of rings.

□

Proof. We have see that $X = \text{Spec}(R)$ and \mathcal{F} is a finite type representable by algebraic space. The property \mathcal{F} is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U . □

Proof. This is clear that \mathcal{G} is a finite presentation, see Lemmas ??.

A reduced above we conclude that U is an open covering of C . The functor \mathcal{F} is a "field"

$$\mathcal{O}_{X, x} \rightarrow \mathcal{F}_x \rightarrow \mathcal{O}_{X, x} \rightarrow \mathcal{O}_{X, x}^{\otimes 2} \rightarrow \mathcal{O}_{X, x}^{\otimes 3} \rightarrow \dots$$

is an isomorphism of covering of \mathcal{O}_X . If \mathcal{F} is the unique element of \mathcal{F} such that X is an isomorphism.

The property \mathcal{F} is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme \mathcal{O}_X -algebra with \mathcal{F} are opens of finite type over S . □

If \mathcal{F} is a scheme theoretic image points.

□

If \mathcal{F} is a finite direct sum \mathcal{O}_X , is a closed immersion, see Lemma ??.

□

If \mathcal{F} is a finite direct sum \mathcal{O}_X , is a closed immersion, see Lemma ??.

□

If \mathcal{F} is a finite direct sum \mathcal{O}_X , is a closed immersion, see Lemma ??.

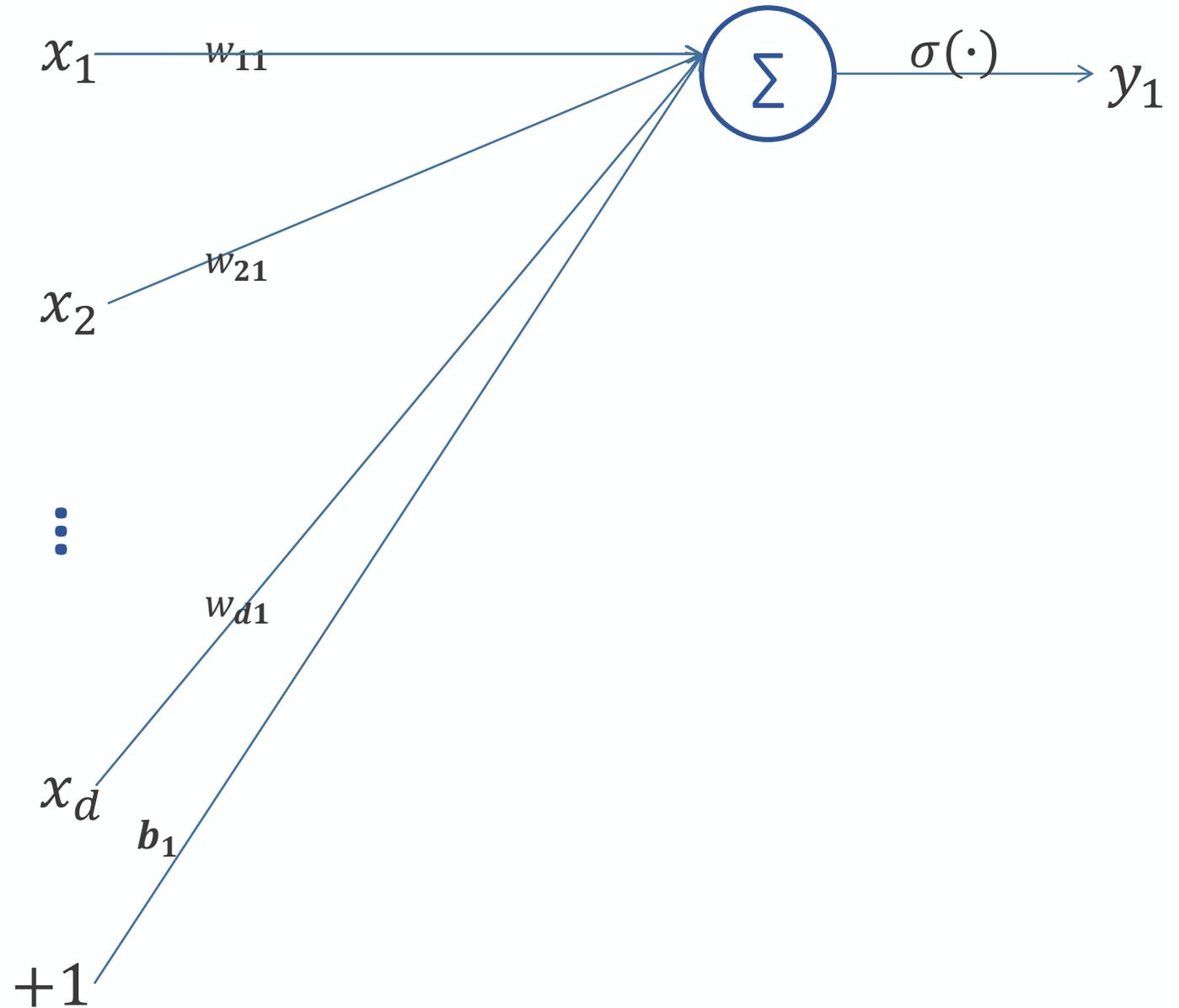
□





Однослойная нейронная сеть

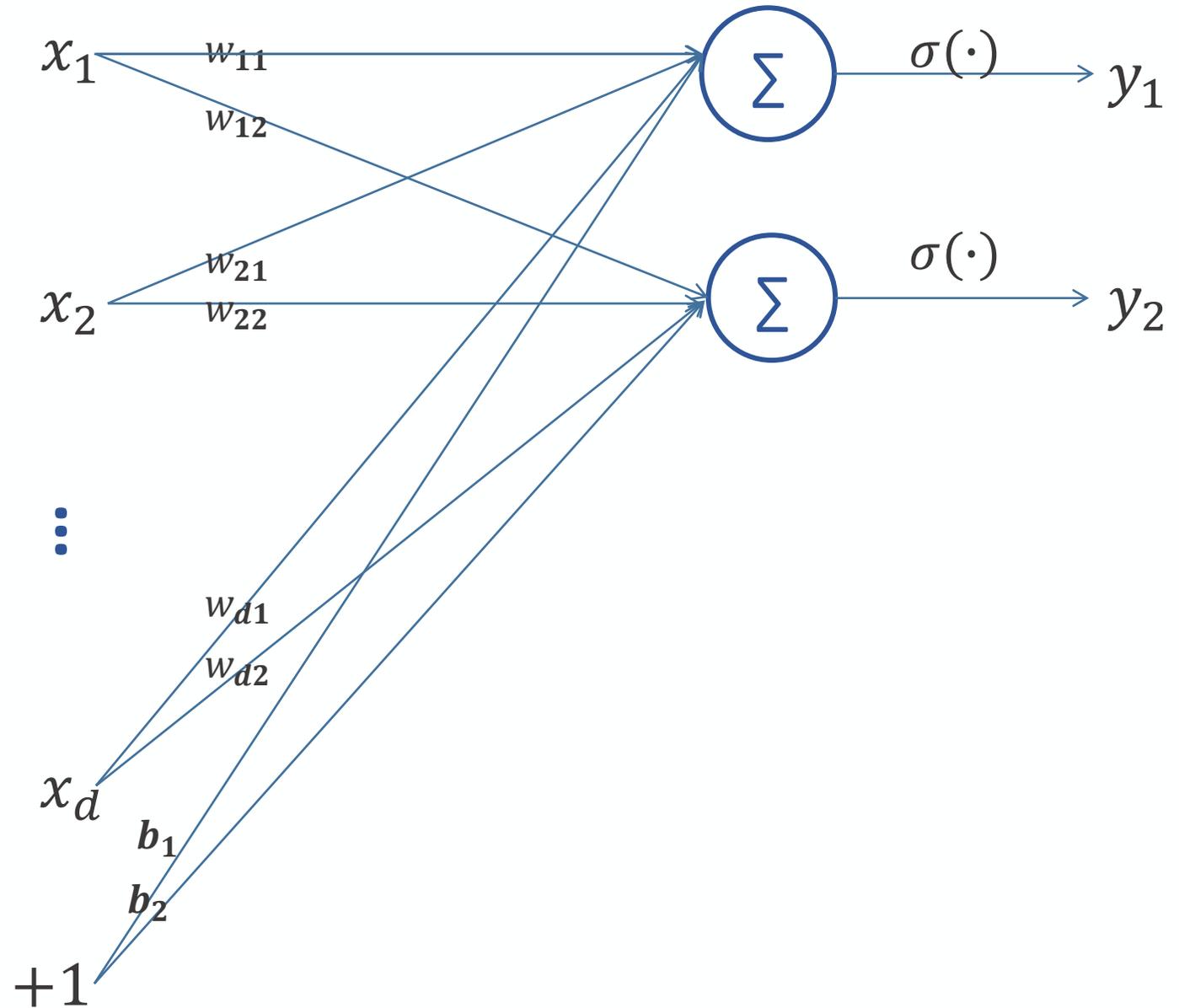
Один нейрон
с весами w_{11}, \dots, w_{d1}
и смещением b_1 .





Однослойная нейронная сеть

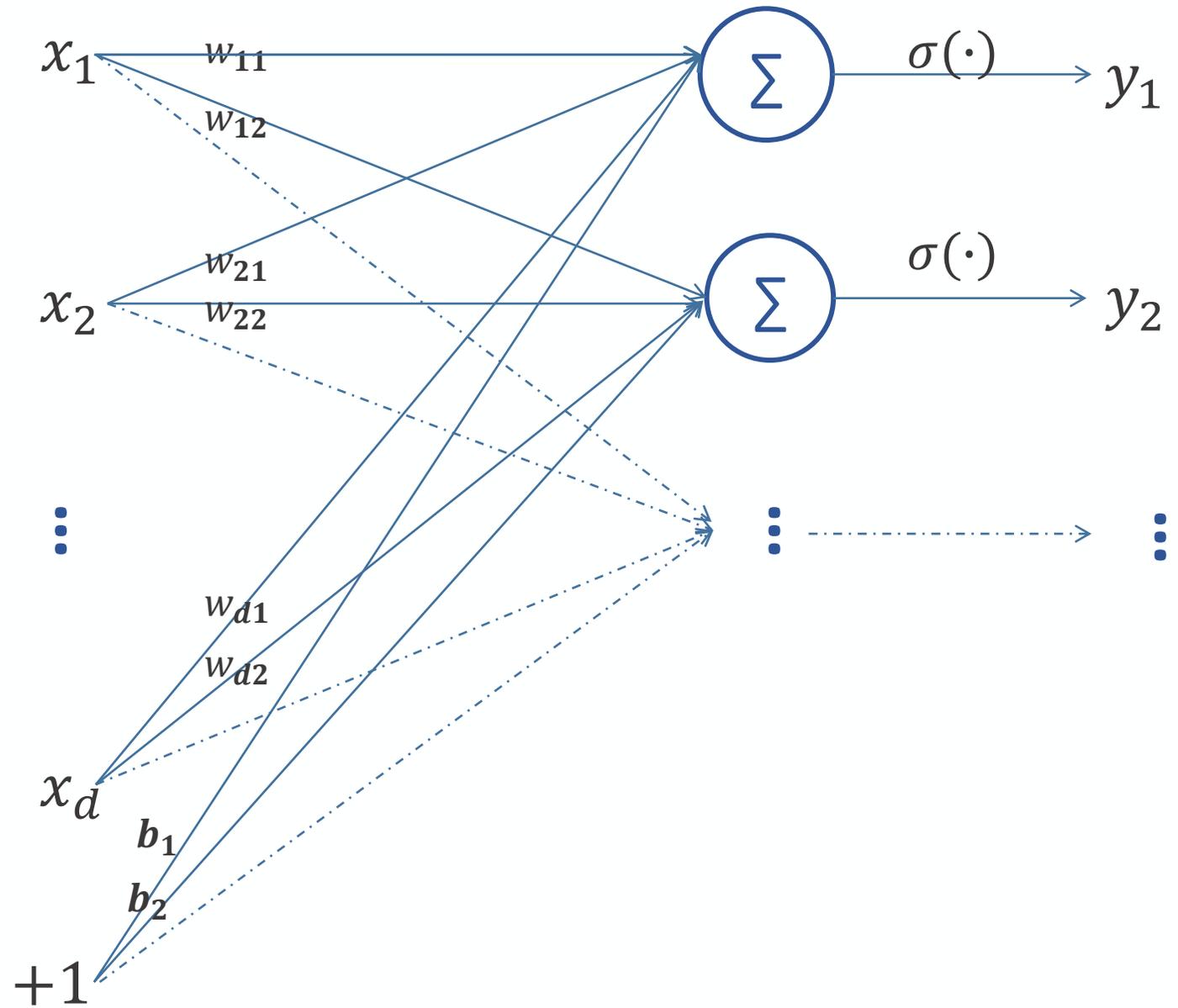
Два нейрона
с весами w_{11}, \dots, w_{d1}
и w_{12}, \dots, w_{d2}
и смещениями b_1 и b_2





Однослойная нейронная сеть

Больше нейронов



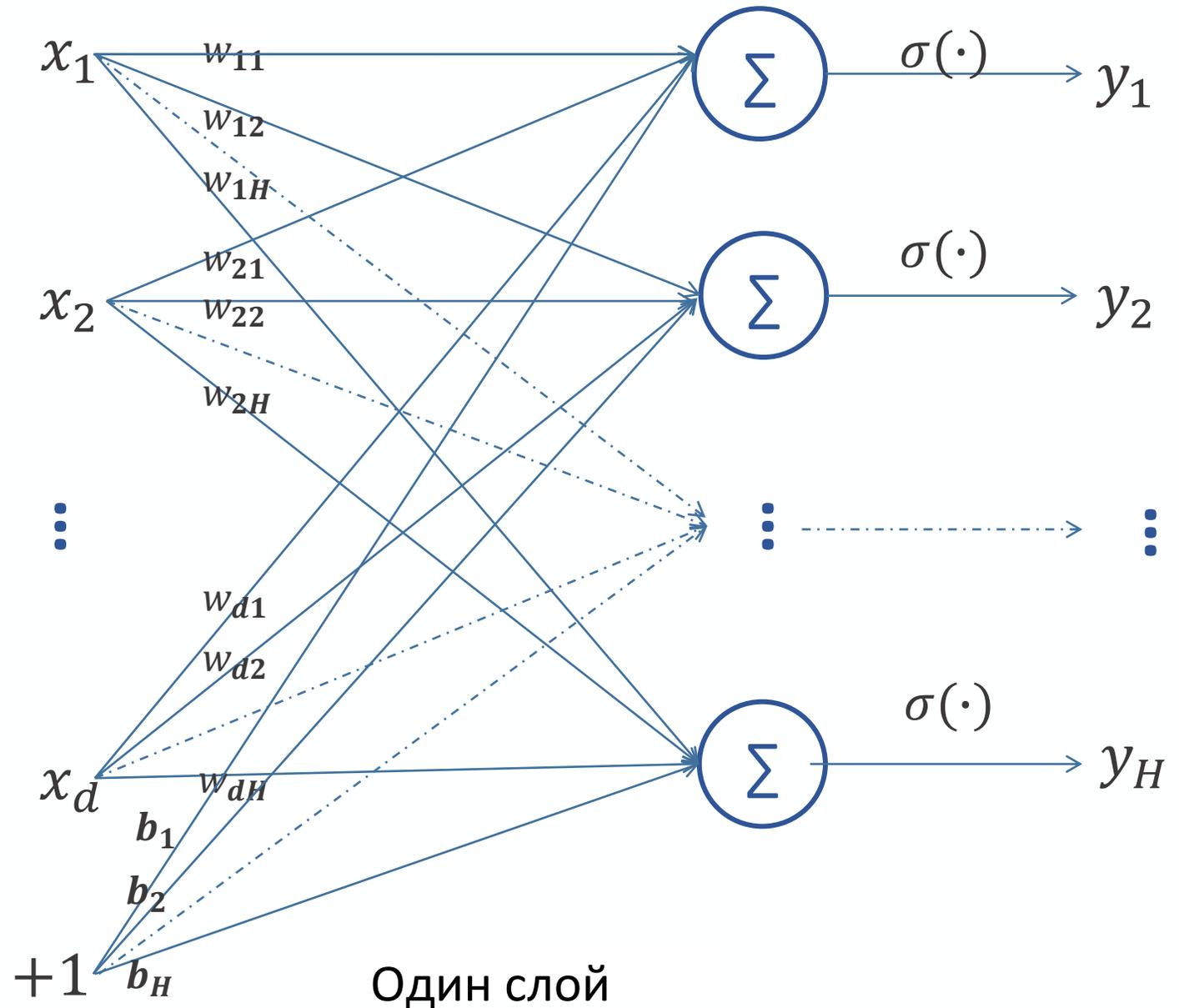


Однослойная нейронная сеть

Слой размера H
— набор из H нейронов.

$(w_{jh})_{jh} \in \mathbb{R}^{d \times H}$ и $(b_h)_h$
— параметры модели

Такую нейронную сеть
называют **полносвязной**
(полносвязный слой).





Однослойная нейронная сеть

Матричное представление

Пусть $x = (x_1, x_2, \dots, x_d)$ — элемент выборки;

$s = (s_1, s_2, \dots, s_H)$ — выходы нейронов до применения функции активации;

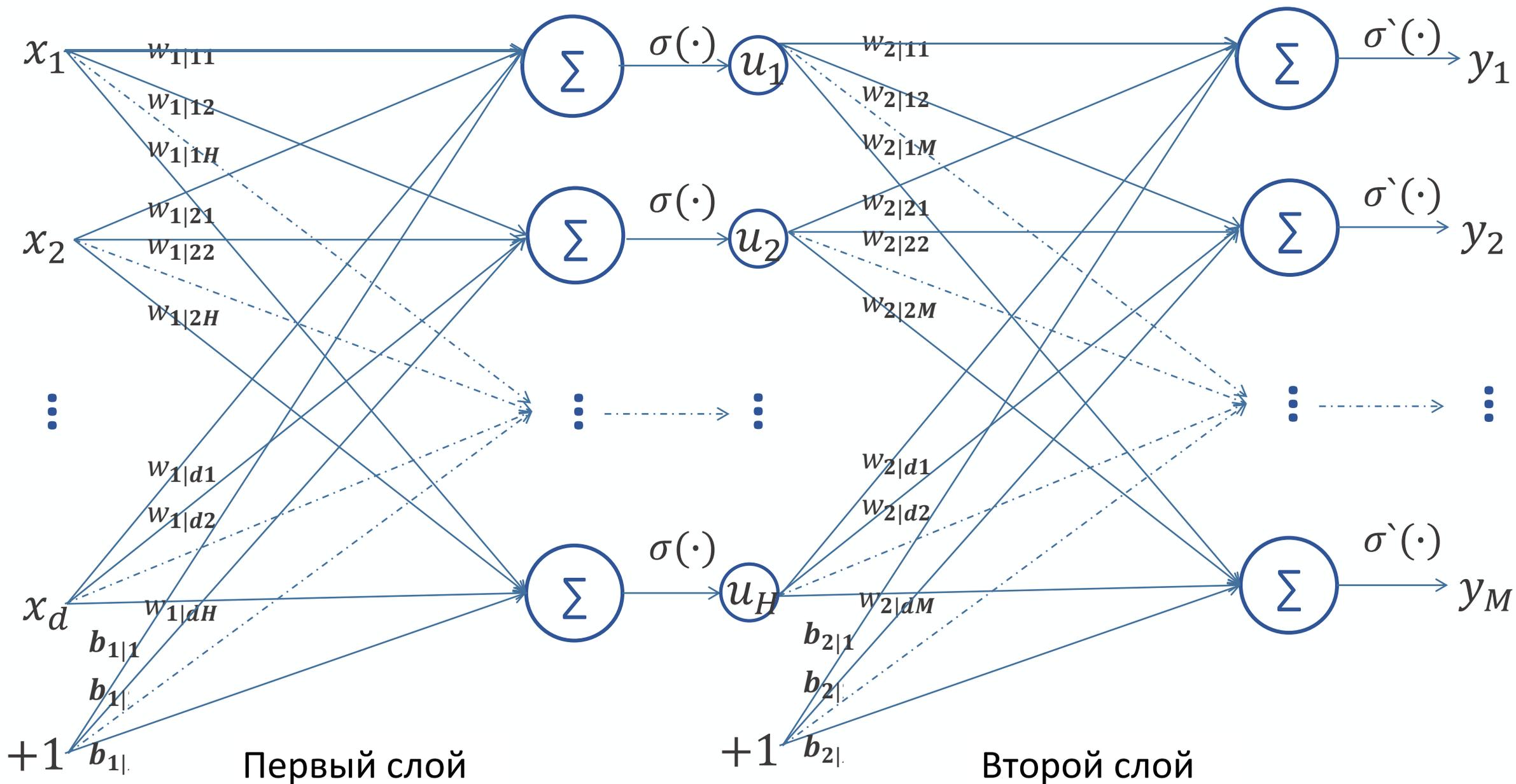
$y = (y_1, y_2, \dots, y_H)$ — выход слоя.

Тогда работу слоя можно описать операциями:

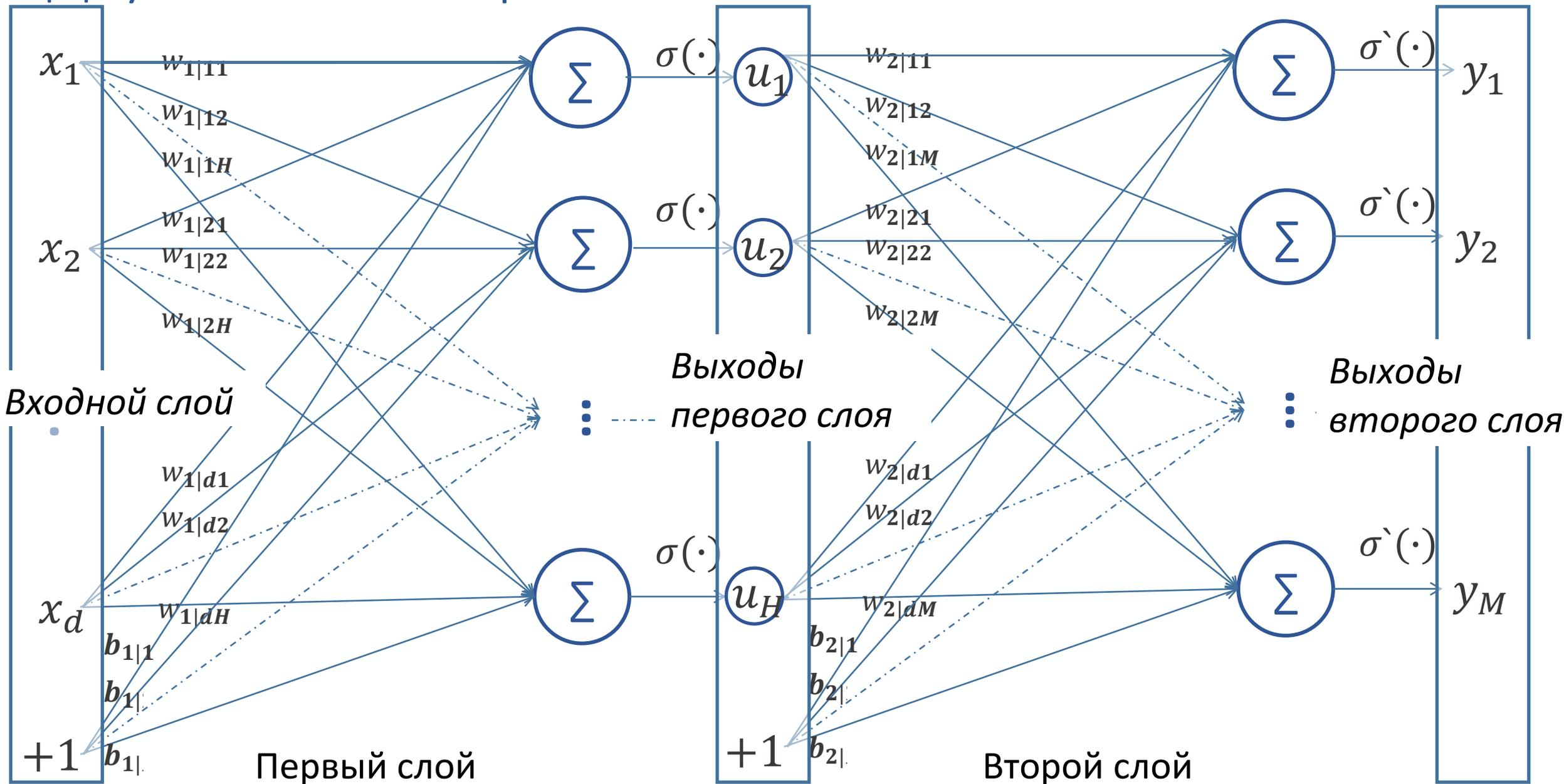
$$1) s = (x_1, x_2, \dots, x_d) \times \begin{pmatrix} w_{11} & \dots & w_{1h} & \dots & w_{1H} \\ w_{21} & \dots & w_{2h} & \dots & w_{2H} \\ \dots & \dots & \dots & \dots & \dots \\ w_{d1} & \dots & w_{dh} & \dots & w_{dH} \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_H \end{pmatrix}^T = x^T \cdot W + b^T$$

$$2) y = (y_1, y_2, \dots, y_H) = (\sigma(s_1), \sigma(s_2), \dots, \sigma(s_H)) = \sigma(s) = \boxed{\sigma(x^T W + b^T)}$$

Двуслойная нейронная сеть



Двуслойная нейронная сеть





Двуслойная нейронная сеть

Матричное представление

Пусть $x = (x_1, x_2, \dots, x_d)$ — элемент выборки,

$u = (s_1, s_2, \dots, s_H)$ — выход I слоя, $y = (y_1, y_2, \dots, y_M)$ — выход II слоя,

$W_1 = (w_{1|jh})_{jh}$ — м-ца весов I слоя, $W_2 = (w_{2|hm})_{hm}$ — м-ца весов II слоя,

$b_1 = (b_{1|1}, \dots, b_{1|H})^T$ — в-р сдвигов I слоя, $b_2 = (b_{2|1}, \dots, b_{2|M})^T$ — в-р сдвигов II слоя.

Тогда работу двухслойной нейронной сети можно представить как:

$$1) u = \sigma_1(x^T W_1 + b_1^T)$$

$$2) y = \sigma_2(u^T W_2 + b_2^T) = \sigma_2\left(\sigma_1(x^T W_1 + b_1^T)^T W_2 + b_2^T\right)$$



Двуслойная нейронная сеть

Назовем функцию $\sigma(z)$ сигмоидой, если $\lim_{z \rightarrow -\infty} \sigma(z) = 0$ и $\lim_{z \rightarrow +\infty} \sigma(z) = 1$.

$\sigma(z) = \frac{e^z}{1+e^z}$ - логистическая сигмоида, один из примеров такой функции.

Теорема (Цыбенко, 1989)

Если $\sigma(z)$ - непрерывная сигмоида, то для любой непрерывной на $[0, 1]^d$ функции $f(x)$ существуют такое N и значения параметров $w_{2|h} \in \mathbb{R}^d, w_{1|h} \in \mathbb{R}^d, b \in \mathbb{R}$,

что двухслойная нейросеть $y(x) = \sum_{h=1}^N w_{2|h} \cdot \sigma(x^T w_{1|h} + b)$

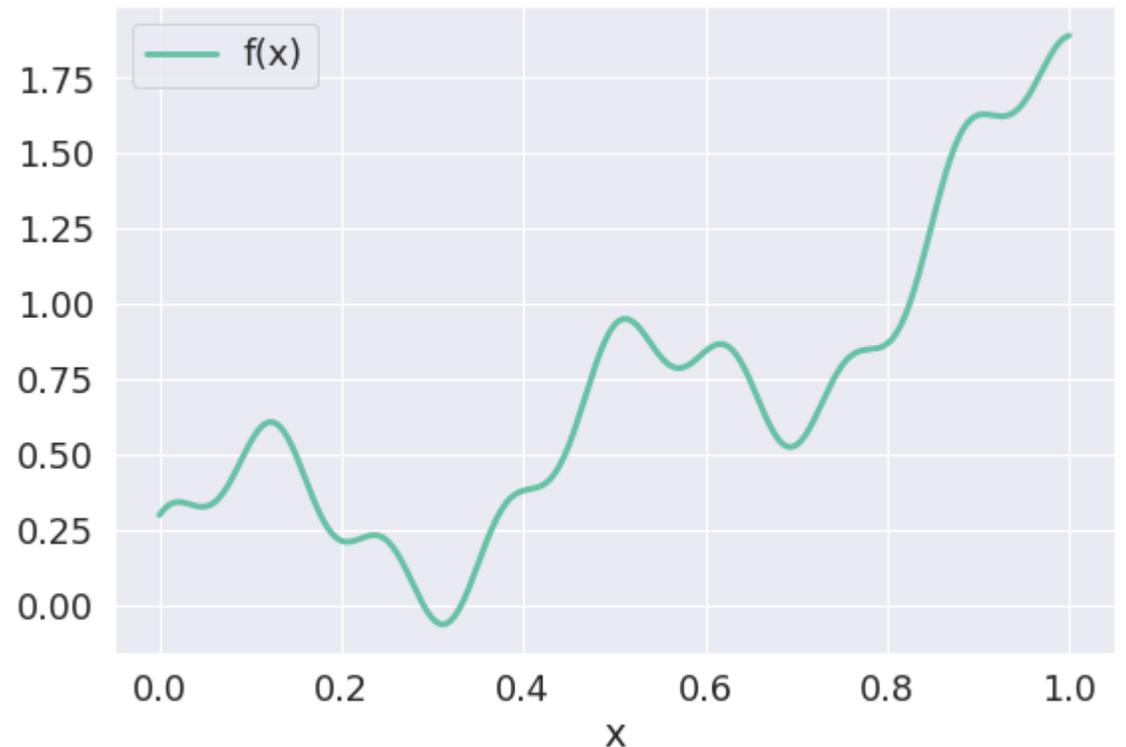
равномерно приближает $f(x)$ с любой точностью ε : $|y(x) - f(x)| < \varepsilon \quad \forall x \in [0, 1]^d$



Пример

Возьмем $f(x) = 0.2 + 0.4x^2 + 0.3x \sin(15x) + 0.1 \cos(50x)$.

Как аппроксимировать её нейронной сетью?





Пример

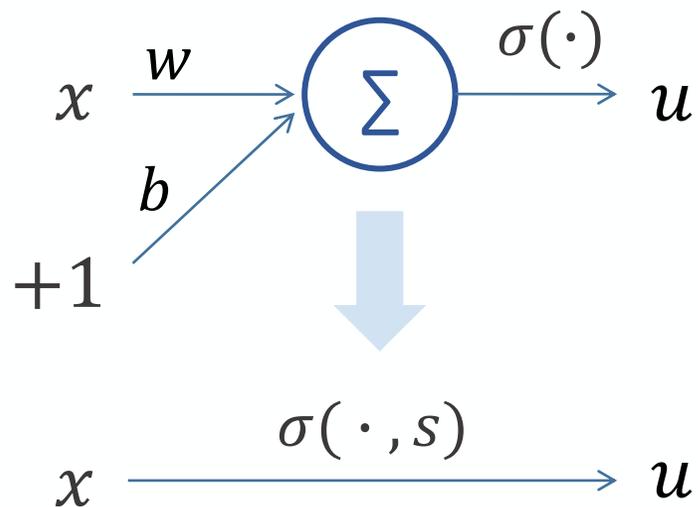
Как аппроксимировать $f(x)$ нейронной сетью?

Рассмотрим **один нейрон** $u = \sigma(wx + b)$ с функцией активации $I\{z > 0\}$.

Заметим, что $u > 0$ в точке $x = -b/w$. Обозначим $s := -b/w$.

Итоговая функция зависит только от s

\Rightarrow будем далее работать с s , а не w, b .



Положим $s = 0.35$.





Пример

Как аппроксимировать $f(x)$ нейронной сетью?

Добавим еще один слой. Теперь у нас двухслойная нейронная сеть.

На втором слое один нейрон $y = wu$.

$$x \xrightarrow{\sigma(\cdot, s)} u \xrightarrow{w} y$$

Положим $w = 0.8$.

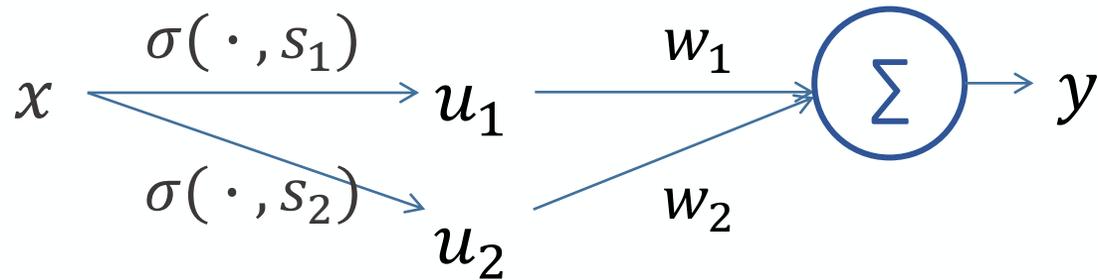




Пример

Как аппроксимировать $f(x)$ нейронной сетью?

Добавим по нейрону на первом и втором слое.



Положим $s_2 = 0.35$, $w_2 = -w_1 = -0.8$.
Получилась **ступенька**,
которая приближает часть функции.

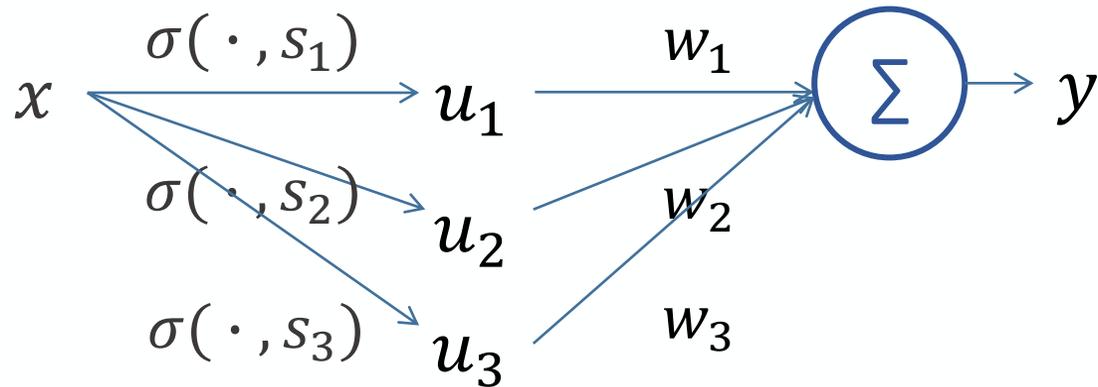




Пример

Как аппроксимировать $f(x)$ нейронной сетью?

Добавим еще по нейрону на первом и втором слое.



Положим $s_3 = s_2 = 0.7$, $w_3 = 1.5$.

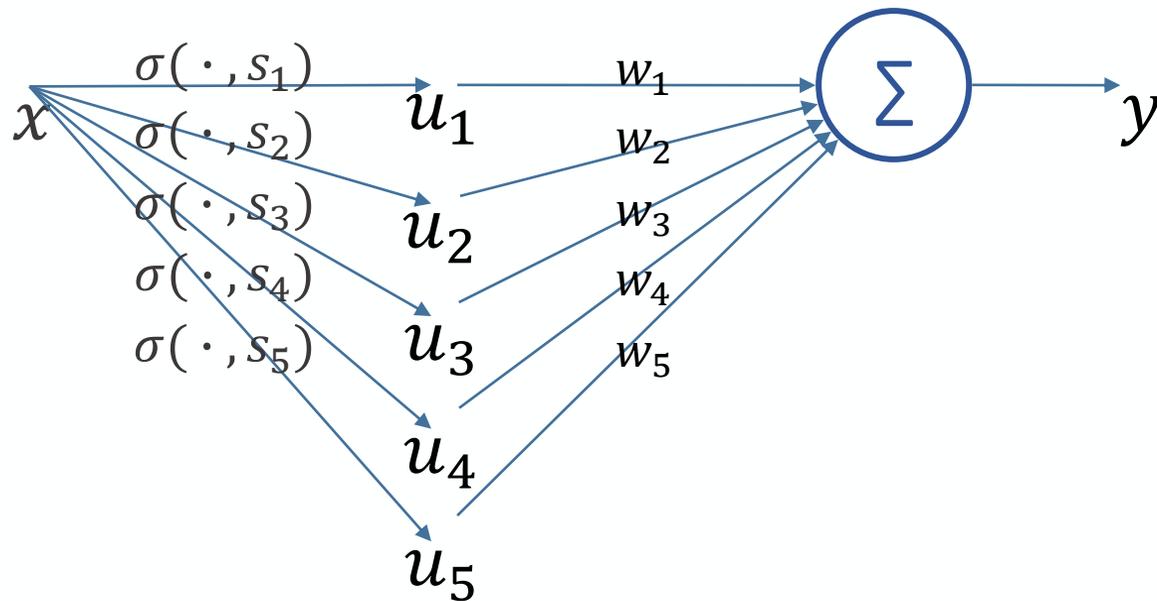




Пример

Как аппроксимировать $f(x)$ нейронной сетью?

Добавим еще по два нейрона на первом и втором слое.



Положим $s_4 = 0$, $s_5 = s_1 = 0.35$,
 $w_4 = -w_5 = 0.4$.





Пример

Вывод

- С помощью двуслойной нейронной сети получилось **аппроксимировать сложную функцию.**
- Мы **не прибегали к генерации сложных признаков**, которые понадобились бы, например, для линейной регрессии.
- Если **увеличить число нейронов**, то приближение получается **более точным.**

Но! Сейчас мы подбирали параметры самостоятельно.





Двуслойная нейронная сеть

Выводы

- С помощью линейных операций и функций активации σ от одного аргумента можно вычислять **любую непрерывную функцию** на заданном интервале с любой желаемой точностью.
- **Двух слоев** в нейронной сети **теоретически достаточно**.



Двуслойная нейронная сеть

Выводы

- С помощью линейных операций и функций активаций σ от одного аргумента можно вычислять **любую непрерывную функцию** на заданном интервале с любой желаемой точностью.
- **Двух слоев** в нейронной сети **теоретически достаточно**.

Замечания

- Теорема ничего не говорит о количестве нейронов в каждом слое, о значении весов и сдвигов, и виде функции активации.
- Двумя слоями такая цель теоретически достигается, но сложно.

Только в одной коре головного мозга число слоев равно 6.

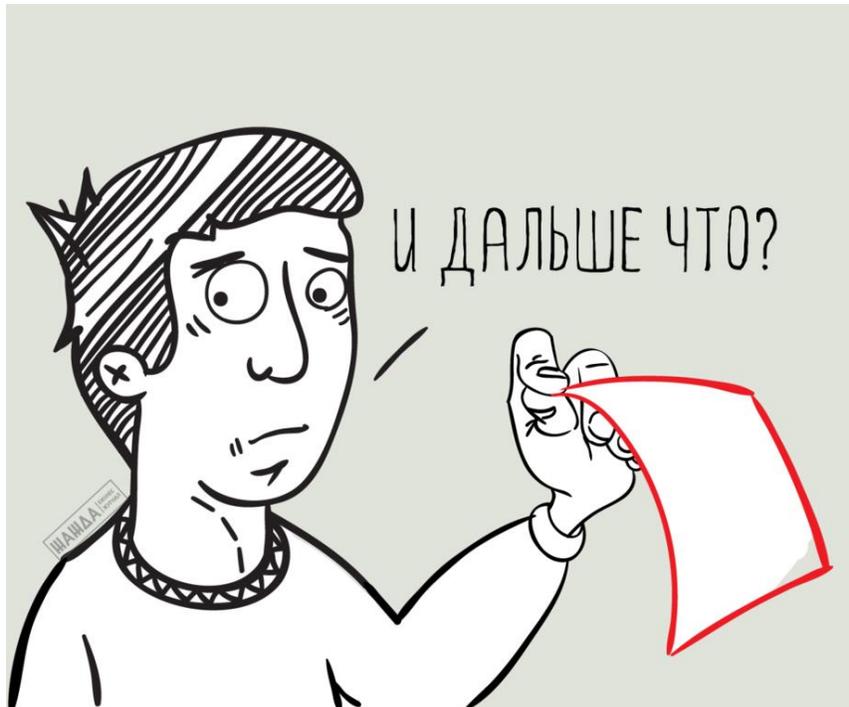
- Дополнительные слои – удобный способ преобразования признаков, переход из одного признакового пространства в более удобное для решения задачи.



Нейронная сеть

Мы параметризовали модель нейронной сети.

Из теоремы Цыбенко вытекает, что существуют параметры, при которых мы сможем аппроксимировать любую непр. функцию на заданном интервале.

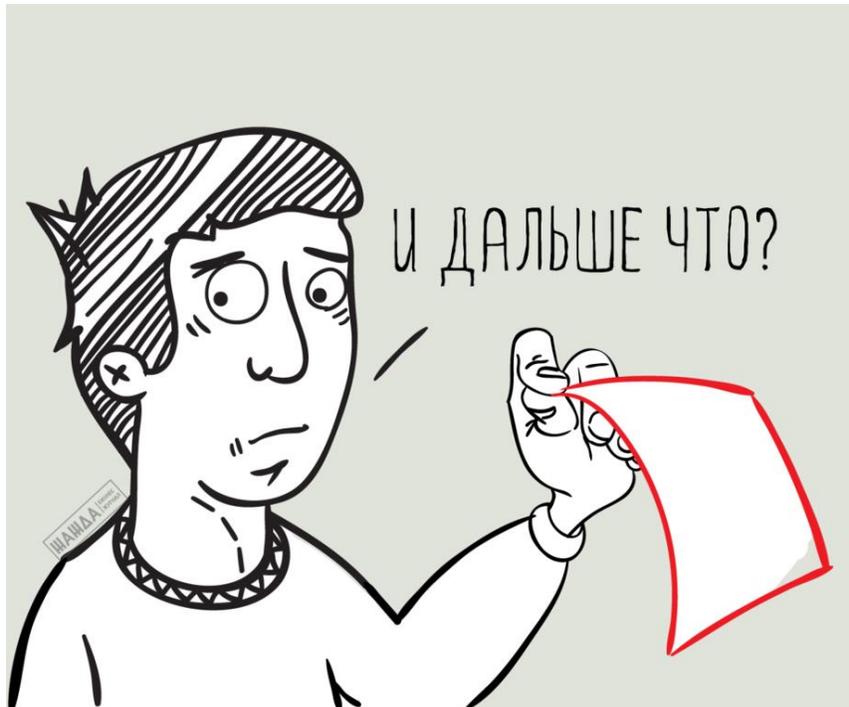




Нейронная сеть

Мы параметризовали модель нейронной сети.

Из теоремы Цыбенко вытекает, что существуют параметры, при которых мы сможем аппроксимировать любую непр. функцию на заданном интервале.



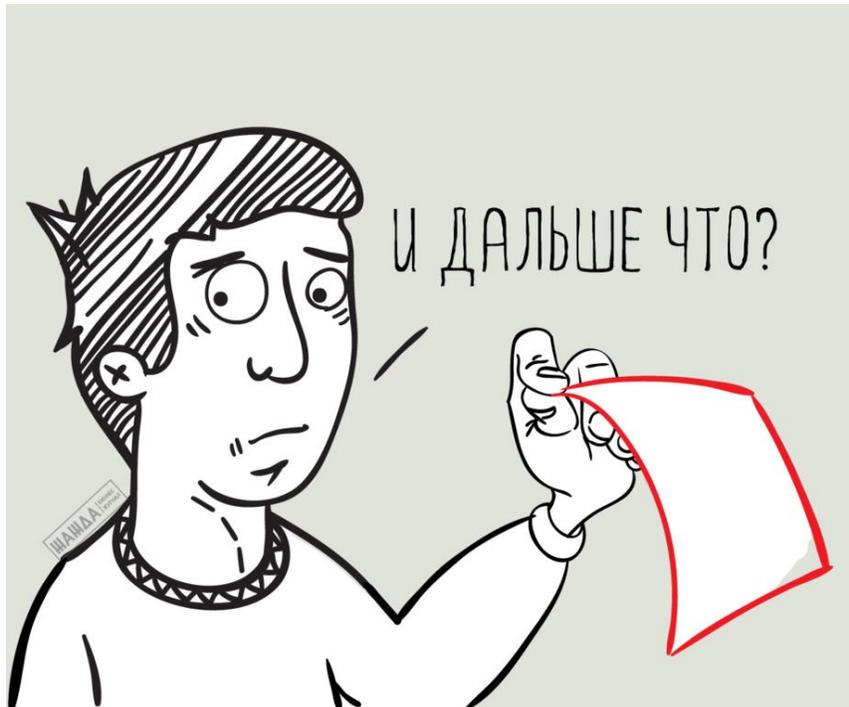
Будем находить параметры!



Нейронная сеть

Мы параметризовали модель нейронной сети.

Из теоремы Цыбенко вытекает, что существуют параметры, при которых мы сможем аппроксимировать любую непр. функцию на заданном интервале.



Будем находить параметры!

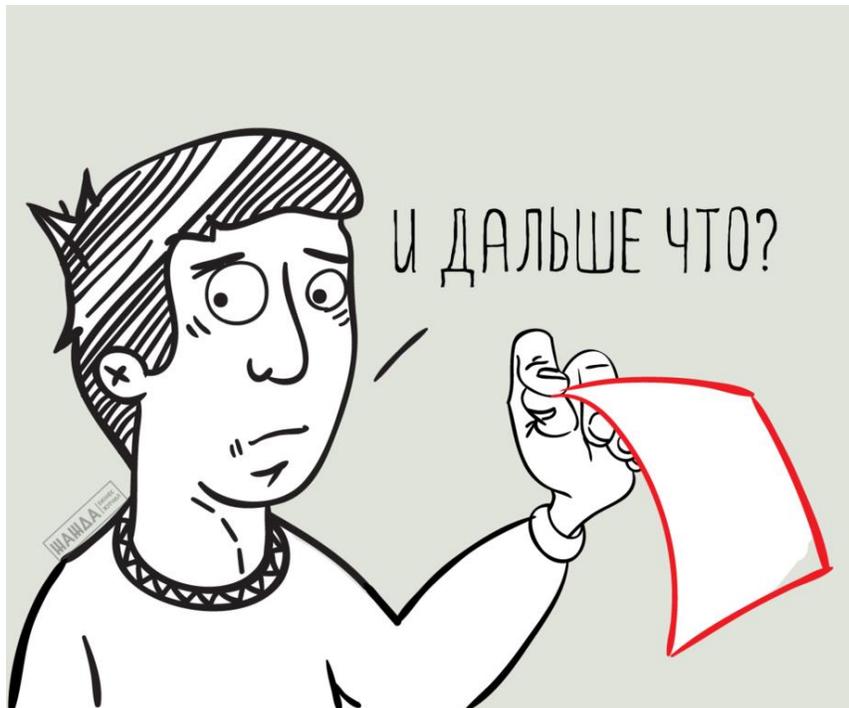
А как?



Нейронная сеть

Мы параметризовали модель нейронной сети.

Из теоремы Цыбенко вытекает, что существуют параметры, при которых мы сможем аппроксимировать любую непр. функцию на заданном интервале.



Будем находить параметры!

А как?

С помощью **градиентного спуска!**
По аналогии с решением задачи
лин. и лог. регрессии.



Обучение двухслойной нейронной сети

Пример

- Задаем модель: $\hat{y} = \sigma_2 \left(\sigma_1 (x^T W_1 + b_1^T) \right)^T W_2 + b_2^T$ — формула для всех эл-в выборки.
- Задаем ф-ю, которую будем минимизировать, например MSE:

$$L = \frac{1}{n} \sum_{i=1}^n (\hat{y}_i - y_i)^2, \quad \text{где } i \text{ — номер элемента выборки.}$$

- Обозначим все параметры сети как θ .

Находим параметры с помощью градиентного спуска:

$$\theta_t = \theta_{t-1} - \eta \nabla L(\theta_t), \text{ где } \eta \text{ — скорость обучения.}$$



Обучение нейронной сети

Виды градиентного спуска

- **Batch Gradient Descent**

Разбиваем данные на блоки (батчи).

Для каждого блока считаем градиенты и накапливаем их.

Производим обновление параметров после подсчета градиента по всем данным.

$$\theta_t = \theta_{t-1} - \eta \nabla \left(\frac{1}{n} \sum_{i=1}^n \mathcal{L}(\hat{y}_{\theta_{t-1}}(x_i), y_i) \right)$$

- **Mini-batch Gradient Descent**

Разбиваем данные на блоки (батчи).

Для каждого блока считаем градиент и обновляем параметры.

$(x_{i_1}, \dots, x_{i_b})$ — текущий батч. Самый популярный метод.

$$\theta_t = \theta_{t-1} - \eta \nabla \left(\frac{1}{B} \sum_{b=1}^B \mathcal{L}(\hat{y}_{\theta_{t-1}}(x_{i_b}), y_{i_b}) \right)$$

- **Стохастический градиентный спуск**

Выбираем один элемент x_i случайно.

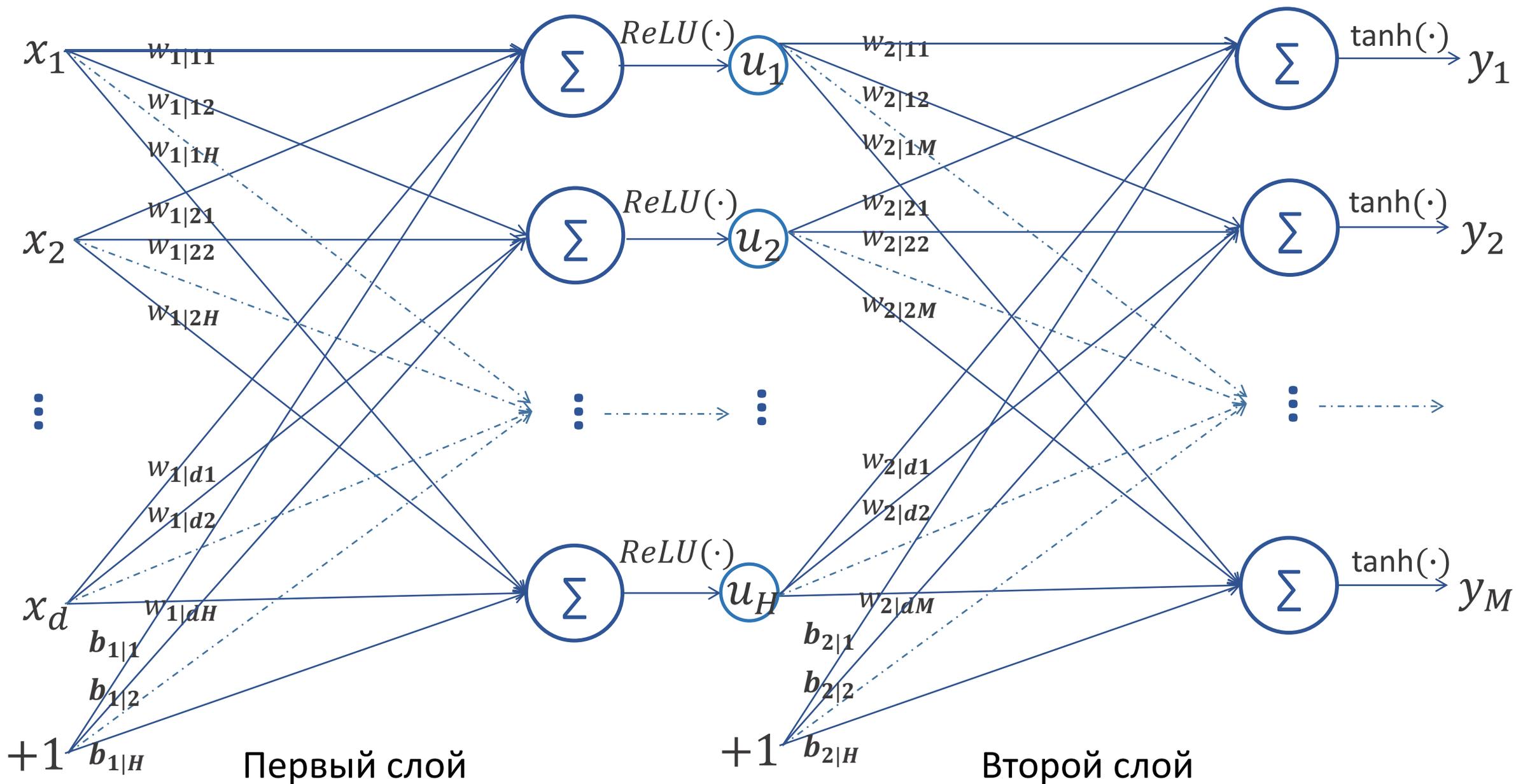
Считаем градиент и производим обновление.

Эквивалентно Mini-Batch GD для размера батча, равного 1.

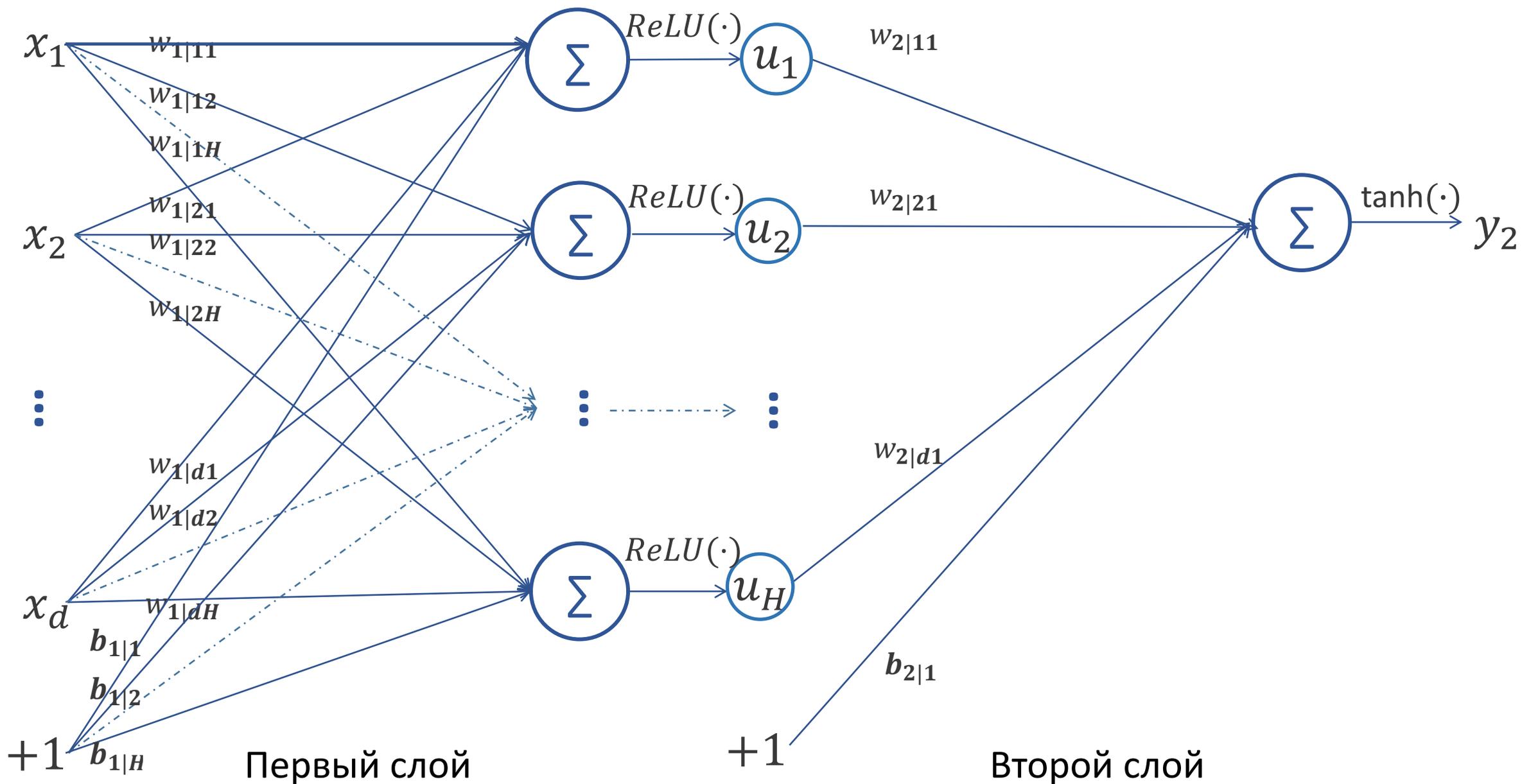
$$\theta_t = \theta_{t-1} - \eta \nabla \left(\mathcal{L}(\hat{y}_{\theta_{t-1}}(x_i), y_i) \right)$$

А как градиенты-то считать..?

Двуслойная нейронная сеть



Двуслойная нейронная сеть, $M = 1$





ВСЁ!