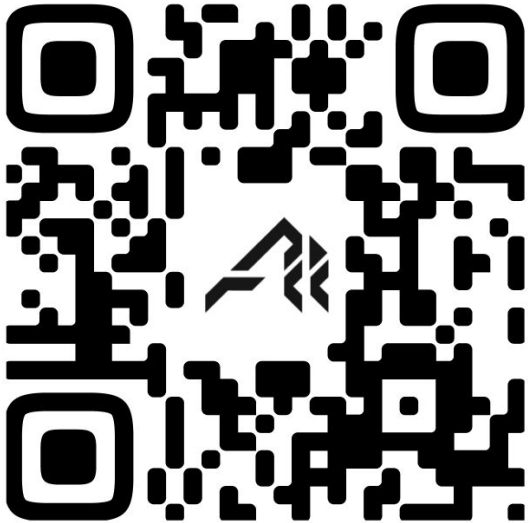
Лекция 1. Устройства полносвязных сетей

Ссылка!



>> ТЕЛЕГРАММ КАНАЛ

>> YOUTUBE КАНАЛ

О лекторах

- Данила Малинин
- Sber AI Lab

- Даниил Сергеев
- Точка

План лекции

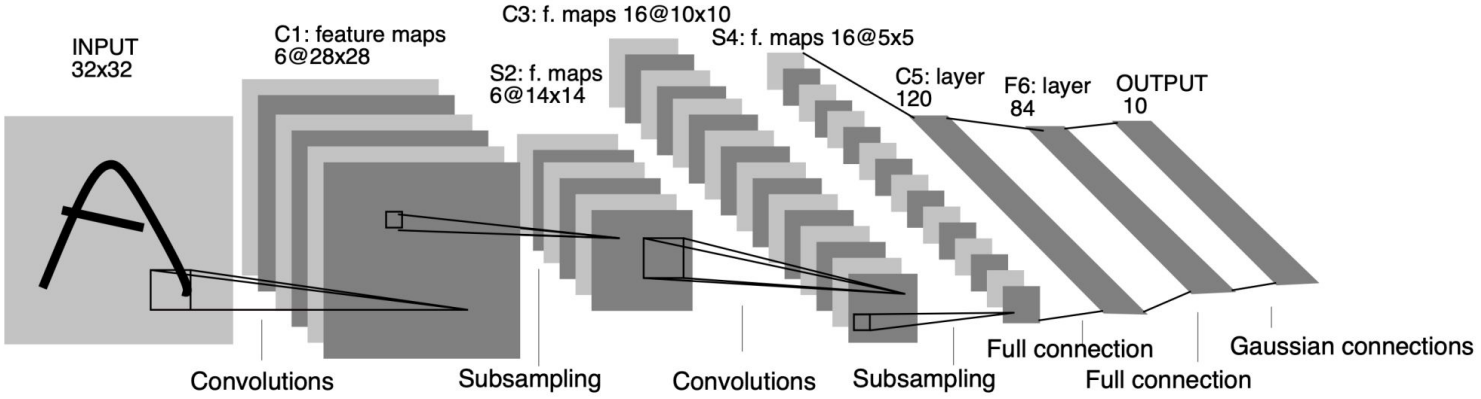
- Виды задач по DL
- Успехи в глубинном обучении
- Отличие ML от DL
- Задача с нелинейной закономерностью
- Связь с моделью нейрона 1940-ых
- От линейных моделей к нелинейным нейронным
- Общий вид Feedforward Neural Network
- Зачем нам нужна нелинейность
- Виды активаций
- Вопрос оптимизации
- Интересные факты про нейронные сети

Классическое компьютерное зрение

1. Считаем признаки (есть ли усы, какой формы уши, какой длины хвост, …)
2. Обучаем на них градиентный бустинг

• Посчитать признаки — целая история

Современное компьютерное зрение



http://vision.stanford.edu/cs598_spring07/papers/Lecun98.pdf

Классический NLP

1. Подсчитываем статистику, как часто то или иное слово встречается после данного
2. Генерируем следующее слово из этого распределения

```
"Manure, almond gelato and frozen pies, you are also had it was in one but it will post office buildings s
ucks). their chinese food. comfort food while they liked their lids ripped off. it an early morning of jon
still a spade so maybe too much. the same. but, at the baked rigatoni, and not in other options and it see
ms odd taste). our visit). i go to nfl kickoff arrived with $. that's about when you come down hoyt street
is actually higher than impressed with a regular theater! so at it, halfway through their pork and though
i've"
```

https://medium.com/analytics-vidhya/making-a-text-generator-using-markov-chains-e17a67225d10

Современный NLP

GPT-3 — нейронная сеть, обученная на огромном корпусе текстов

The article you are writing about is going to be based around this new technology, so you have been spending a lot of time playing around with it. You have also been using your own brain to test out the new models, which is something no one else in the world has done. As a result, you have become somewhat obsessed with it. You constantly think about how it can create such fantastic sentences and how it might be used to solve the world's problems.

Успехи в глубинном обучении

CV (Computer Vision – Компьютерное зрение):

- YOLOv12 - SOTA модель для задач компьютерного зрения в реальном времени
- Amazon Go - Магазины без касс, где система распознаёт, какие товары покупатель взял с полки.

GenAI (Generative AI – Генеративный ИИ):

- Kandinsky 3.1 - модель для создания высокохудожественных изображений в разных стилях и коротких видео
- flux-RealismLora - модель, предназначенную для создания фотореалистичных изображений на основе текстовых описаний

NLP (Natural Language Processing – Обработка естественного языка):

- Perplexity - модель для поиска информации в интернете через семантический поиск и llm
- GPT-4o-mini, DeepSeek-R1 - авторегрессионный трансформер для генерации ответа

Audio (Speech & Music AI – Работа с аудио):

- Whisper - мощная ASR, предназначенная для точного распознавания речи на множестве языков
- SUNO - современная модель для генерации песен

Высокоуровневый вид всех ML задач

- Работа с данными для обучения
- **Выбор модели**
- Выбор функции потерь
- **Выбор метода обучения**
- Проверка по тестовой выборке

Отличие ML от DL

Classic ML:

- Требуется **ручная инженерия признаков** (Feature Engineering). Аналитики и ML-инженеры подготавливают признаки, выбирают их, создают новые комбинации и отбирают важные характеристики.
- Может хорошо работать даже на **малых и средних** объемах данных.
- Относительно **легковесные алгоритмы**, могут работать на **CPU.**
- **Легче интерпретировать.**

Deep Learning:

- Самостоятельно **извлекает признаки** из данных, особенно из изображений, аудио и текста. Глубокие нейросети способны автоматически находить сложные паттерны.
- Требует **больших объемов данных.**
- Требует **мощных графических процессоров (GPU)** или **TPU**
- Часто является **"черным ящиком"**

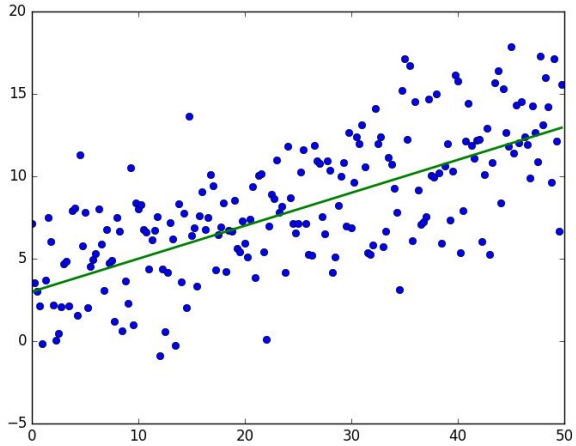
Линейная регрессия

Линейная регрессия — это статистическая модель, которая описывает зависимость между целевой переменной y и набором признаков x1, x2, ..., x_n с помощью линейного уравнения.

$$y = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

Где:

- y — предсказанное значение,
- $x_1, x_2, \ldots, x_n$ — признаки,
- w_0 — **свободный член (bias)**,
- $w_1, w_2, \ldots, w_n$ — **веса (коэффициенты)** модели.



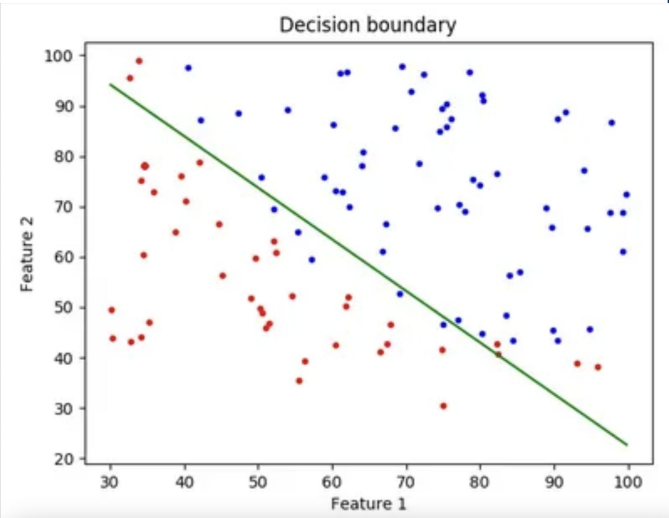
Логистическая регрессия

Линейная регрессия — это **метод машинного обучения** для классификации, который предсказывает вероятность принадлежности объекта к определенному классу.

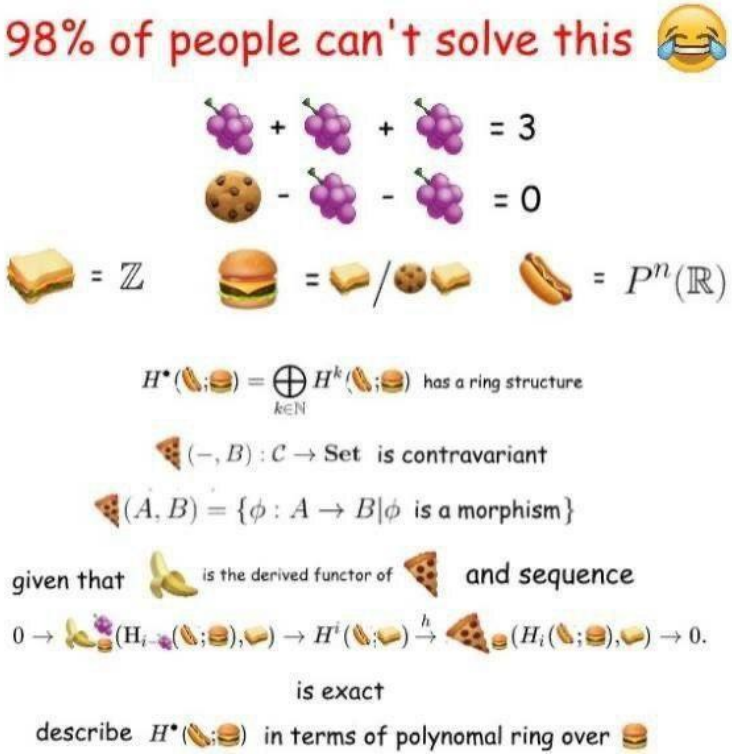
$$p(y = 1|x) = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n)}}$$

где:

- $p(y = 1|x)$ — вероятность принадлежности к классу 1,
- $x_1, x_2, \ldots, x_n$ — признаки,
- $w_0, w_1, \ldots, w_n$ — веса модели.

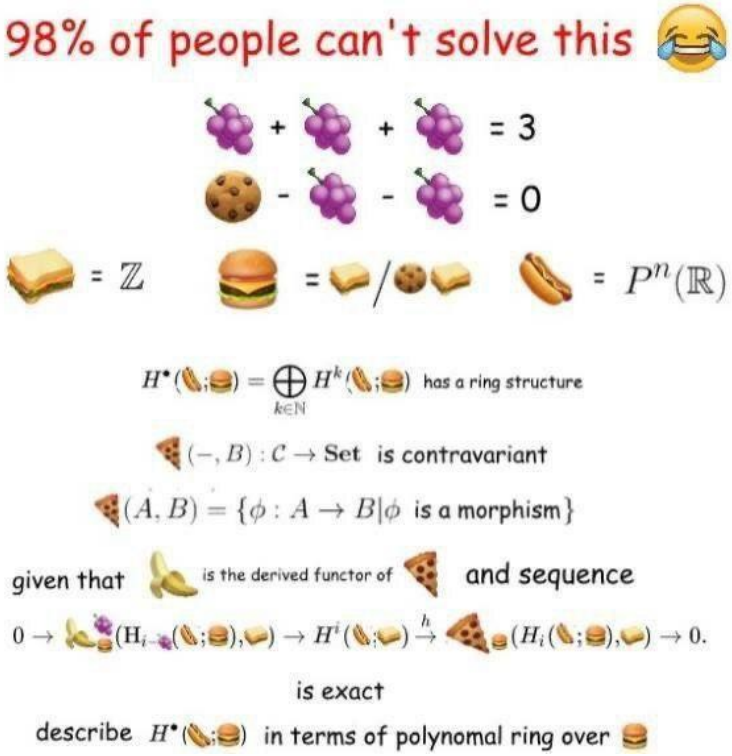


“Математики не будет”



Будет, но не Rocket Science

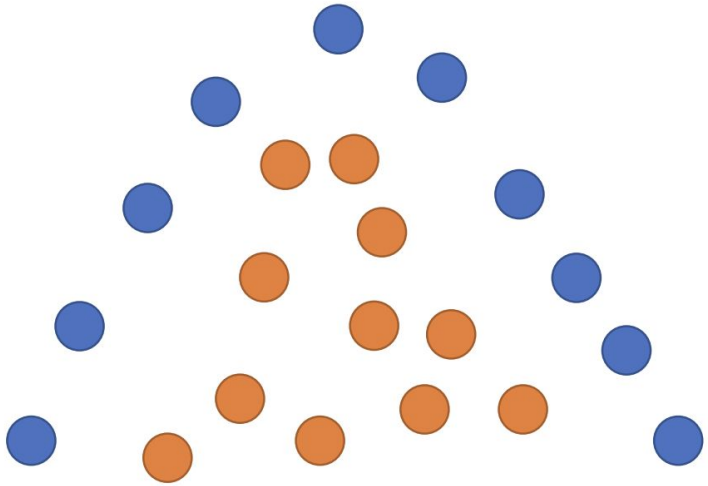
“Математики не будет”



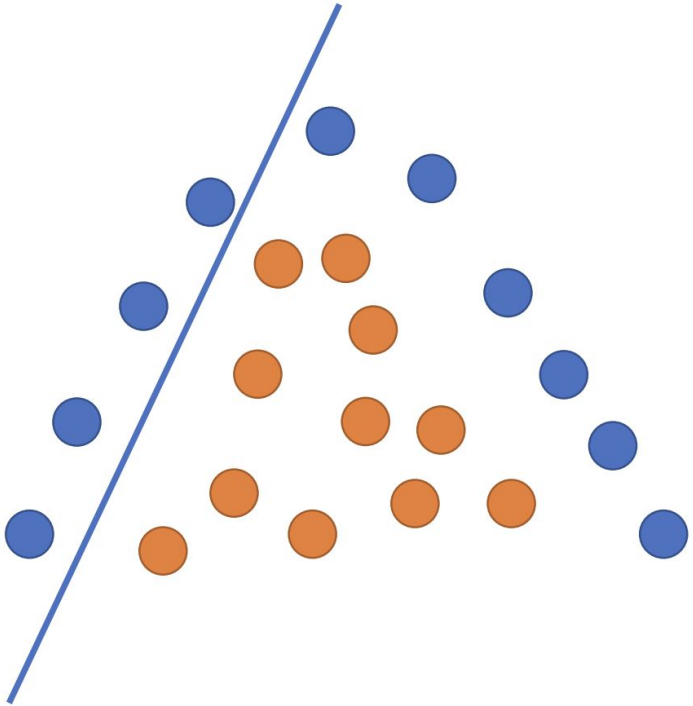
Будет, но не Rocket Science

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

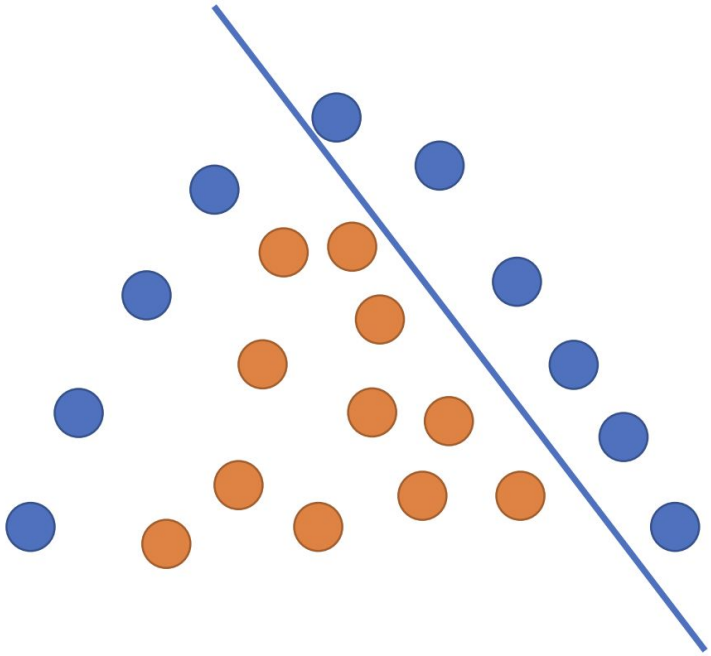
Задача с нелинейной закономерностью



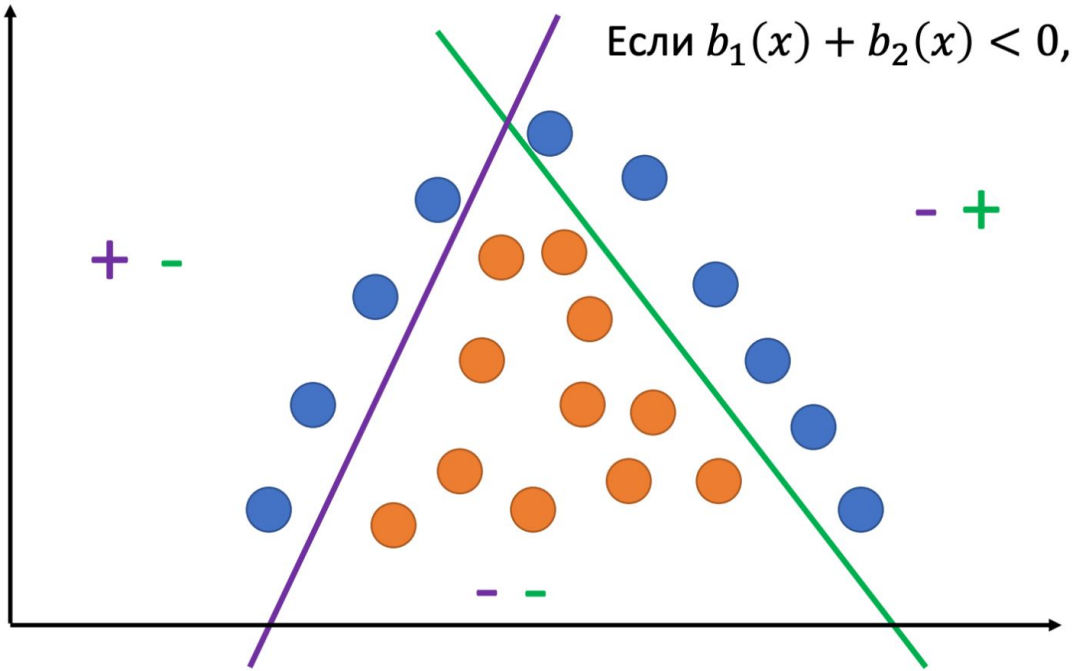
Задача с нелинейной закономерностью



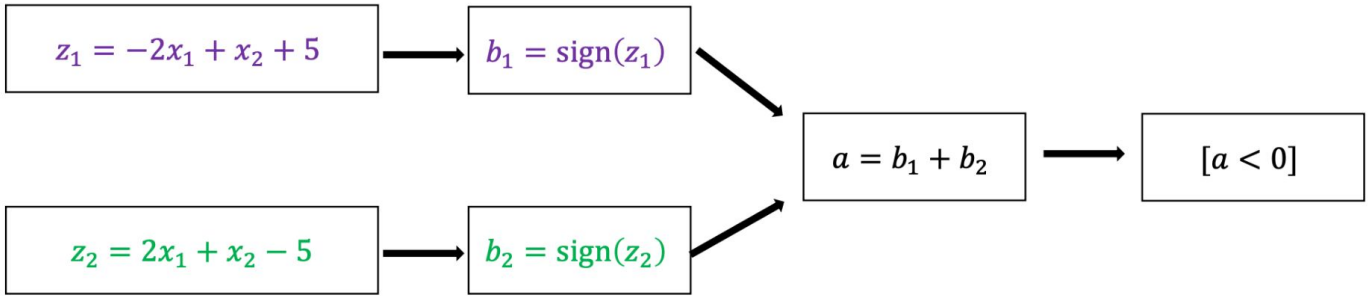
Задача с нелинейной закономерностью



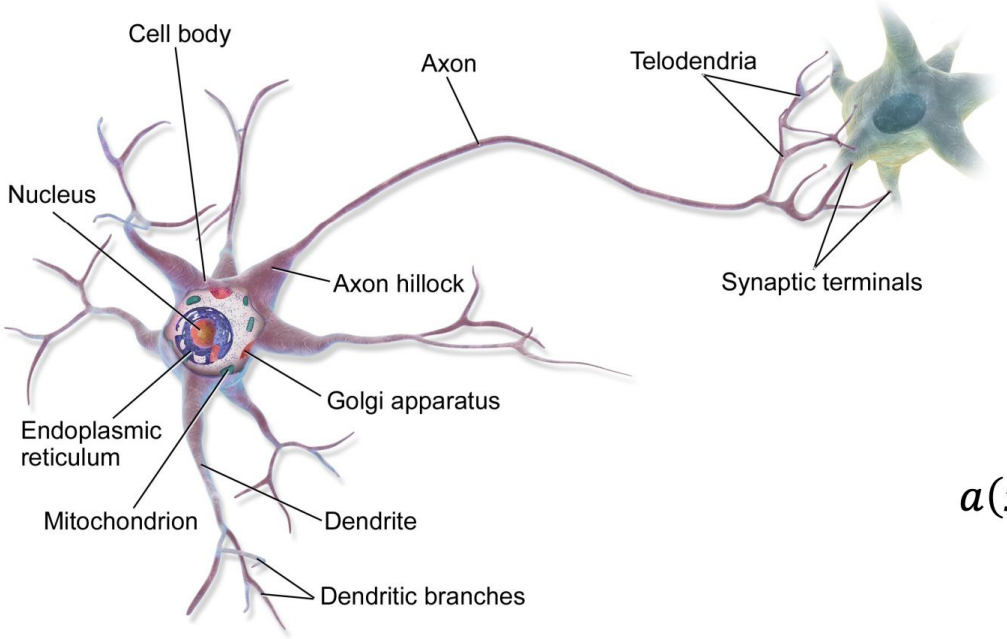
Задача с нелинейной закономерностью



Задача с нелинейной закономерностью

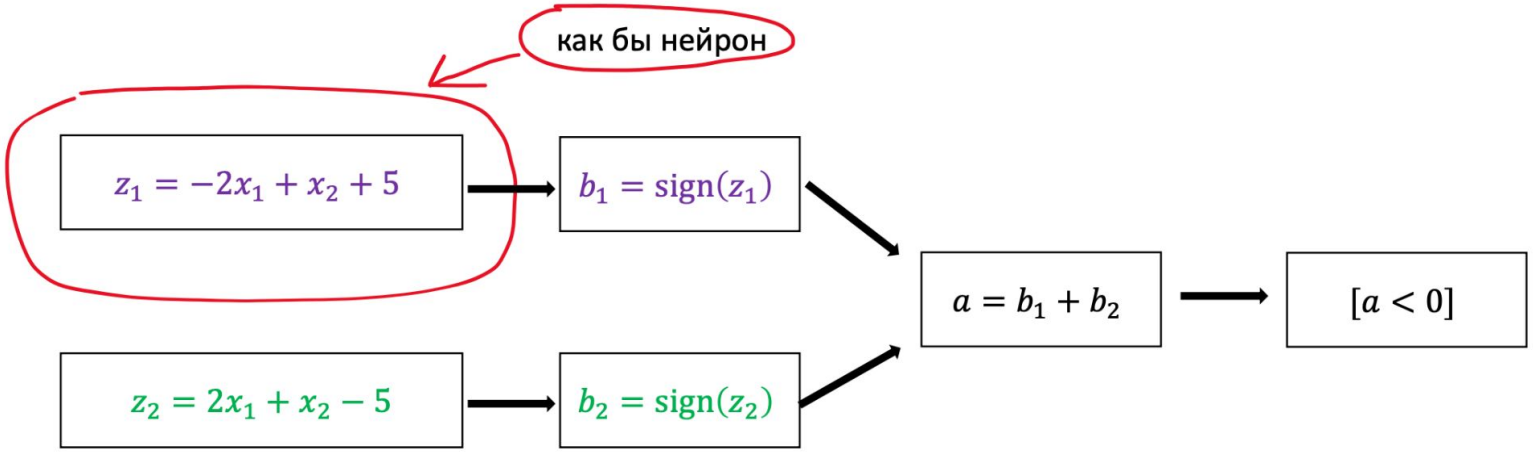


Нейрон в биологии

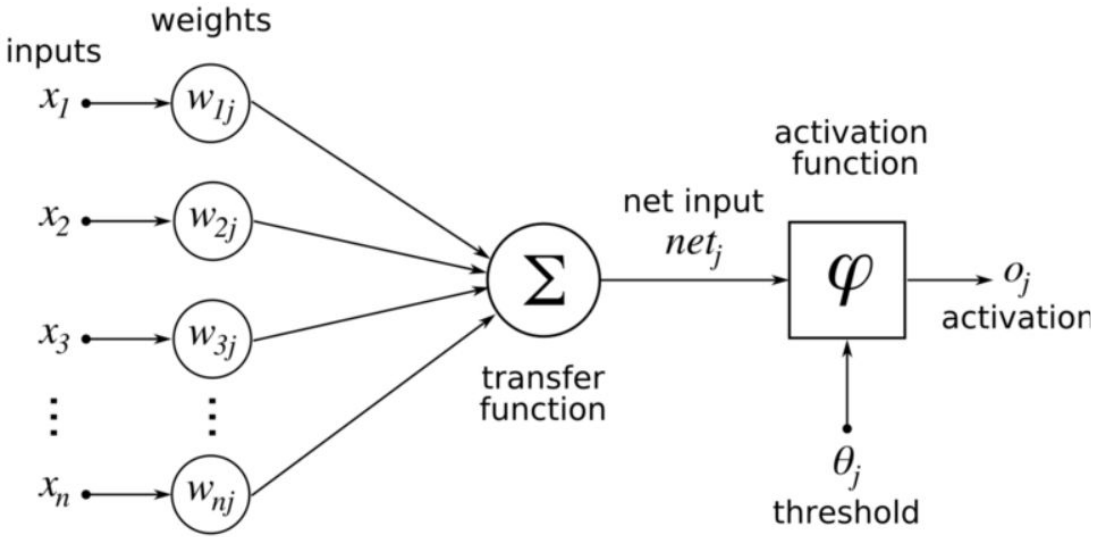


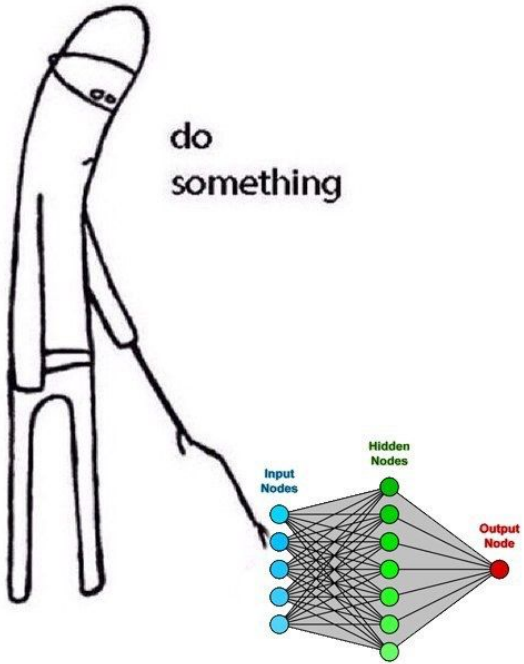
$$a(x) = \sum_{j=1}^{d} w_j x_j$$

Задача с нелинейной закономерностью



Связь с моделью нейрона 1940-х



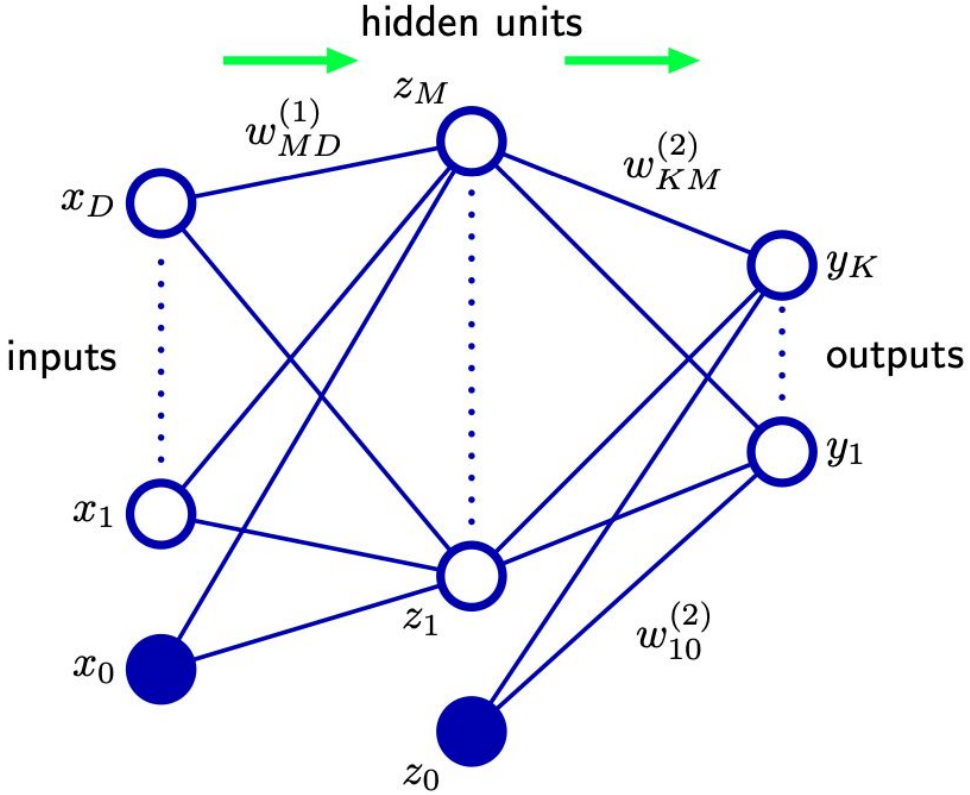


Разбор нейрона через текст

Require: w (weight vector)
Require: b (bias)
Require: x (input vector)
Require: f (activation function)

$a = w \cdot x + b$ // Linear transformation
$y = f(a)$ // Activation function application
Return y

Общий вид Feedforward Neural Network



$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)},$$

$$z_j = h(a_j),$$

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}$$

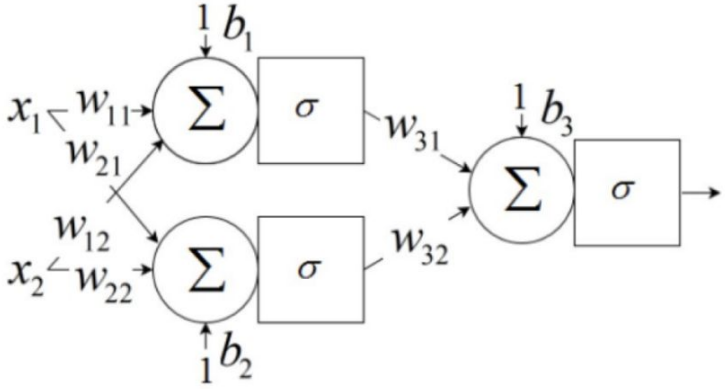
$$y_k = \sigma(a_k)$$

$$j = 1, 2, ..M$$

$$k = 1, 2, ..K$$

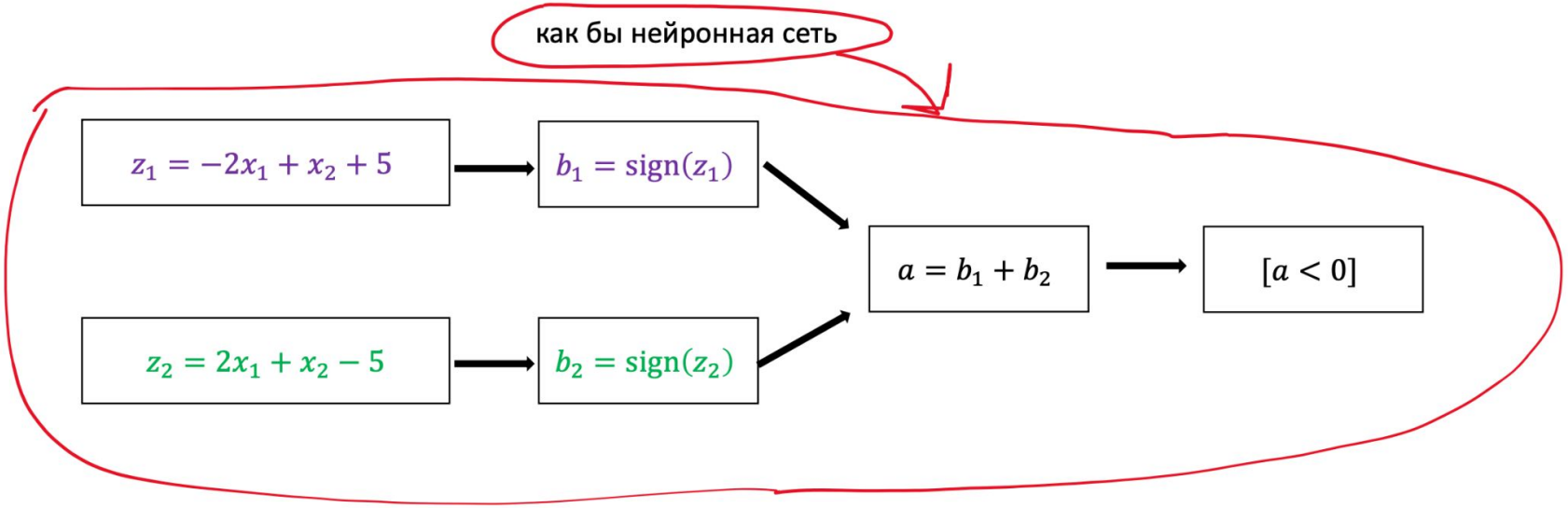
Переход к матричной форме

Двуслойная нейросеть



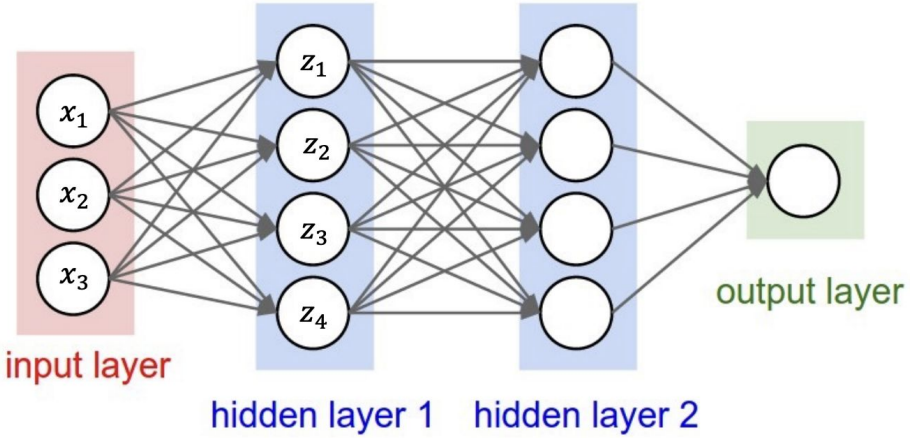
$$\sigma\left(\begin{bmatrix} w_{31} & w_{32} & b_3 \end{bmatrix}\left[\sigma\left(\begin{bmatrix} w_{11} & w_{12} & b_1 \\ w_{21} & w_{22} & b_2 \end{bmatrix}\begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}\right)\right]\right)$$

Задача с нелинейной закономерностью



Полносвязный слой (fully connected, FC)

$$z_j = \sum_{i=1}^{n} w_{ij} x_i + b_j$$



Classic Matrix Notation (by Goodfellow, 2014)

Require: Network depth, l
Require: $\boldsymbol{W}^{(i)}, i \in \{1, \ldots, l\}$, the weight matrices of the model
Require: $\boldsymbol{b}^{(i)}, i \in \{1, \ldots, l\}$, the bias parameters of the model
Require: $\boldsymbol{x}$, the input to process
Require: $\boldsymbol{y}$, the target output

$\boldsymbol{h}^{(0)} = \boldsymbol{x}$
for $k = 1, \ldots, l$ **do**
  $\boldsymbol{a}^{(k)} = \boldsymbol{b}^{(k)} + \boldsymbol{W}^{(k)} \boldsymbol{h}^{(k-1)}$
  $\boldsymbol{h}^{(k)} = f(\boldsymbol{a}^{(k)})$
end for
$\hat{\boldsymbol{y}} = \boldsymbol{h}^{(l)}$
$J = L(\hat{\boldsymbol{y}}, \boldsymbol{y}) + \lambda \Omega(\theta)$

Нелинейность

1. Рассмотрим два полносвязных слоя

$$s_k = \sum_{j=1}^{m} v_{kj} z_j + c_k = \sum_{j=1}^{m} v_{kj} \sum_{i=1}^{n} w_{ji} x_i + \sum_{j=1}^{m} v_{kj} b_j + c_k =$$

$$= \sum_{j=1}^{m} \left(\sum_{i=1}^{n} v_{kj} w_{ji} x_i + v_{kj} b_j + \frac{1}{m} c_k \right)$$

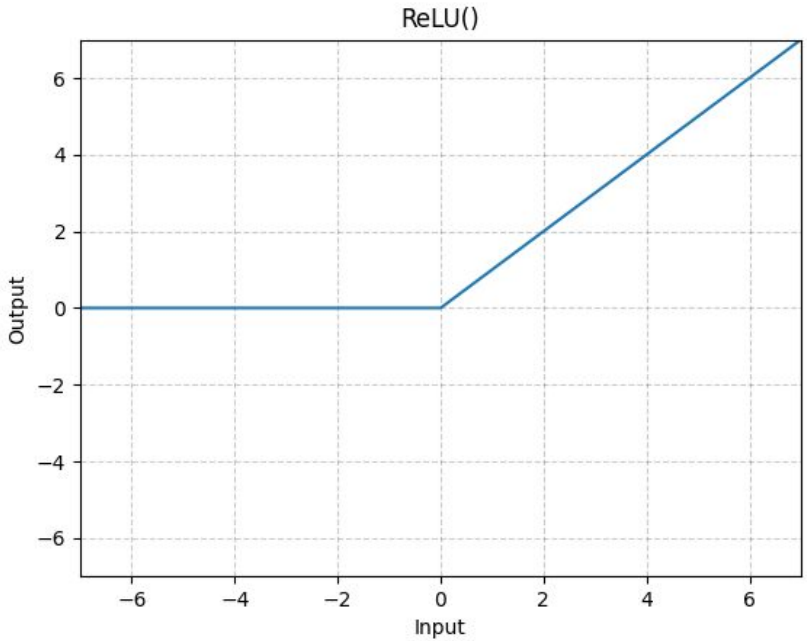
2. То есть это ничем не лучше одного полносвязного слоя

Нелинейность

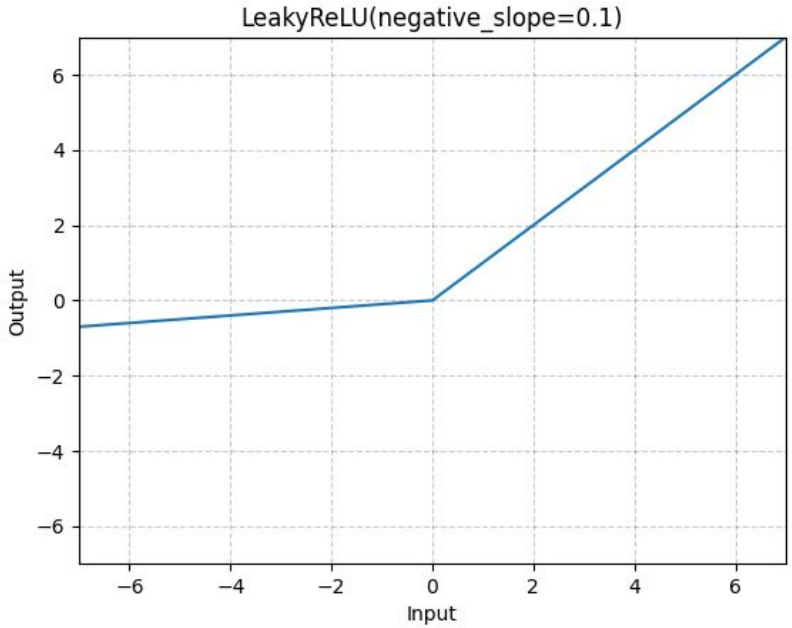
Нужно добавлять нелинейную функцию после полносвязного слоя

$$z_j = f\left(\sum_{i=1}^{n} w_{ji} x_i + b_j\right)$$

Виды активаций

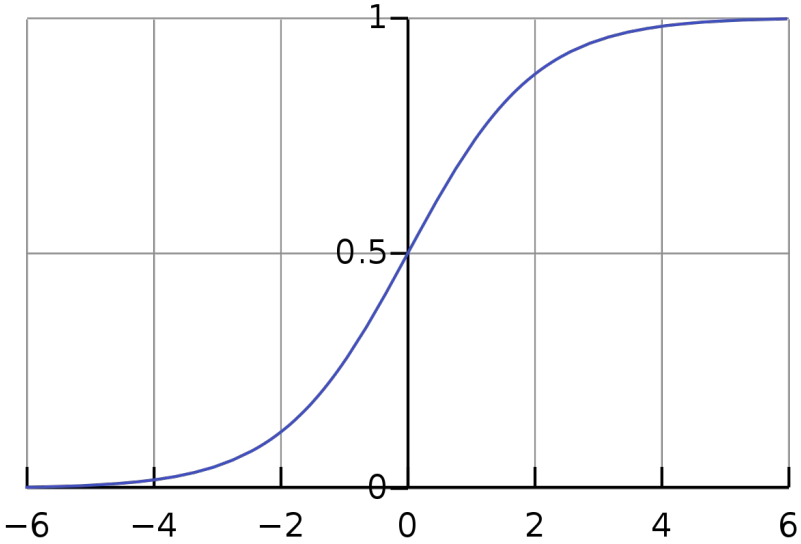


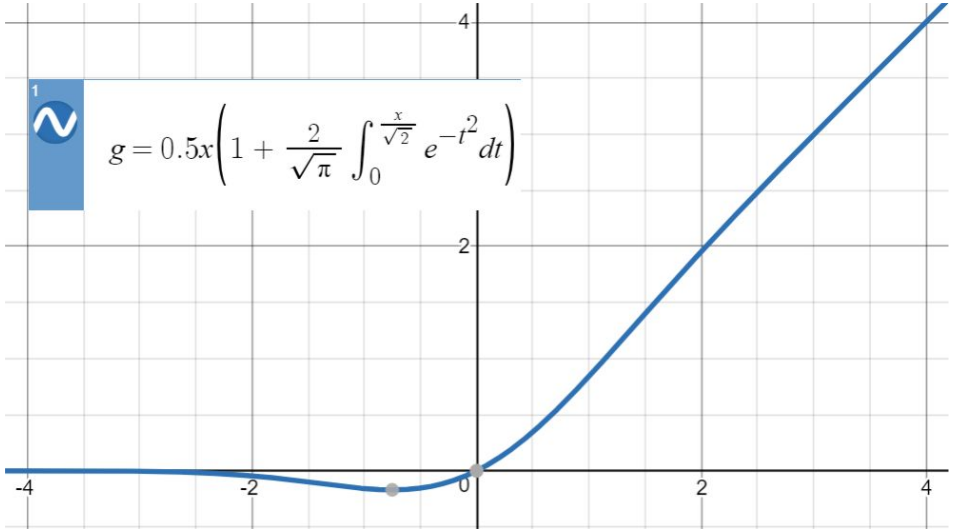
$\mathrm{ReLU}(x)=\max(0,x)$



$\mathrm{LeakyReLU}(x)=\max(0,x)+\mathrm{negative\_slope}*\min(0,x)$

Виды активаций





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

$$\mathrm{GELU}(x) = x * \Phi(x)$$

Виды активаций

И многие остальные из ~400 штук, упомянутых здесь
А часто используемые можно посмотреть в документации pytorch

Таблица 3.1. Различные функции активации: сводная таблица

Название функции	Формула $f(x)$	Производная $f'(x)$
Логистический сигмоид σ	$\frac{1}{1+e^{-x}}$	$f(x)\,(1-f(x))$
Гиперболический тангенс tanh	$\frac{e^x-e^{-x}}{e^x+e^{-x}}$	$1-f^2(x)$
SoftSign	$\frac{x}{1+\lvert x\rvert}$	$\frac{1}{(1+\lvert x\rvert)^2}$
Ступенька (функция Хевисайда)	$\begin{cases}0, & x<0\\ 1, & x\geq 0\end{cases}$	0
SoftPlus	$\log(1+e^x)$	$\frac{1}{1+e^{-x}}$
ReLU	$\begin{cases}0, & x<0\\ x, & x\geq 0\end{cases}$	$\begin{cases}0, & x<0\\ 1, & x\geq 0\end{cases}$
Leaky ReLU, Parameterized ReLU	$\begin{cases}ax, & x<0\\ x, & x\geq 0\end{cases}$	$\begin{cases}a, & x<0\\ 1, & x\geq 0\end{cases}$
ELU	$\begin{cases}\alpha\,(e^x-1), & x<0\\ x, & x\geq 0\end{cases}$	$\begin{cases}f(x)+\alpha, & x<0\\ 1, & x\geq 0\end{cases}$

Вопрос оптимизации

$$\min_w \frac{1}{m} \sum_{i=1}^{m} L\left(\hat{y}\left(x_i, w\right), y_i\right)$$

L - функция потерь (optimisation objective)

$\hat{y}$ - выход модели

w - веса модели

x_i - входные данные

y_i - истинные значения (targets)

$i = 1, ..m$

m - размер тренировочной выборки

Вопрос оптимизации

$$\min_w \frac{1}{m} \sum_{i=1}^{m} L\left(\hat{y}\left(x_i, w\right), y_i\right)$$

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

MSE = mean squared error
n = number of data points
Y_i = observed values
$\hat{Y}_i$ = predicted values

L - функция потерь (optimisation objective)

$\hat{y}$ - выход модели

w - веса модели

x_i - входные данные

y_i - истинные значения (targets)

$i = 1, ..m$

m - размер тренировочной выборки

Вопрос оптимизации

$$\min_{w} \frac{1}{m} \sum_{i=1}^{m} L\left(\hat{y}\left(x_i, w\right), y_i\right)$$

L - функция потерь (optimisation objective)

$\hat{y}$ - выход модели

w - веса модели

x_i - входные данные

y_i - истинные значения (targets)

$i = 1, ..m$

m - размер тренировочной выборки

$$H_p(q) = -\frac{1}{N} \sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Binary Cross-Entropy / Log Loss

Градиентный спуск, а как вычислять

$$w^{(t+1)} = w^{(t)} - \eta \nabla L$$

η - learning rate (step size, шаг обучения)

$\nabla_w L$ - градиент L относительно w

Градиентный спуск, а как вычислять

$$w^{(t+1)} = w^{(t)} - \eta \nabla L \Longleftrightarrow$$

$$w_1 = w_1 - \eta \frac{\partial f}{\partial w_1} \quad \dots \quad w_n = w_n - \eta \frac{\partial f}{\partial w_n}$$

η - learning rate (step size, шаг обучения)

$\nabla_w L$ - градиент L относительно w

Градиентный спуск, а как вычислять

$$y = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + \varepsilon$$

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

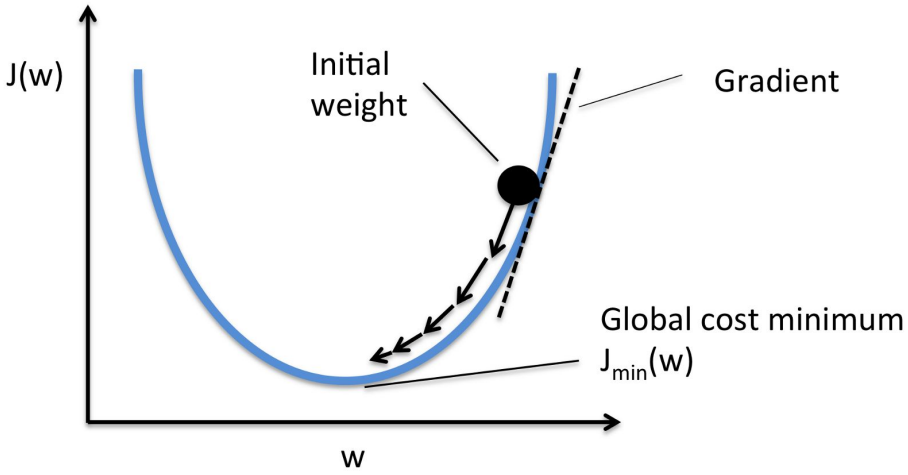
$$w_1 = w_1 - \eta\frac{\partial f}{\partial w_1} \quad \dots \quad w_n = w_n - \eta\frac{\partial f}{\partial w_n}$$

Градиентный спуск, а как вычислять

$$y = w_1 x_1 + \varepsilon$$

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

$$w_1 = w_1 - \eta \frac{\partial f}{\partial w_1}$$



Градиентный спуск, а как вычислять

$$y = wx + b$$

$$w = 1, \quad b = 0, \quad \eta = 0.01$$

$$(x = 3, y_{\text{true}} = 7)$$

$$\hat{y} = wx + b = 1 \cdot 3 + 0 = 3$$

$$L = (y_{\text{true}} - \hat{y})^2 = (7 - 3)^2 = 16$$

Градиентный спуск, а как вычислять

$$\frac{\partial L}{\partial w} = 2(wx + b - y_{\text{true}})x$$

$$\frac{\partial L}{\partial b} = 2(wx + b - y_{\text{true}})$$

$$\frac{\partial L}{\partial w} = 2(3 - 7) \cdot 3 = -24$$

$$\frac{\partial L}{\partial b} = 2(3 - 7) = -8$$

Градиентный спуск, а как вычислять

$$w = w - \eta \frac{\partial L}{\partial w} = 1 - 0.01 \cdot (-24) = 1 + 0.24 = 1.24$$

$$b = b - \eta \frac{\partial L}{\partial b} = 0 - 0.01 \cdot (-8) = 0 + 0.08 = 0.08$$

$$w = 1.24, \quad b = 0.08$$

Градиентный спуск, а как вычислять

$$y = wx + b$$

$$w = 1.24, \quad b = 0.08, \quad \eta = 0.01$$

$$(x = 3, y_{\text{true}} = 7)$$

$$\hat{y} = wx + b = 1.24 \cdot 3 + 0.08 = 3.8$$

$$L = (y_{\text{true}} - \hat{y})^2 = (7 - 3.8)^2 = 10.24$$

Теорема Цыбенко

Пусть φ любая непрерывная сигмоидная функция, например, $\varphi(\xi) = 1/(1+e^{-\xi})$. Тогда, если дана любая непрерывная функция действительных переменных f на $[0,1]^n$ (или любое другое компактное подмножество $\mathbb{R}^n$) и $\varepsilon > 0$, то существуют векторы $\mathbf{w_1}, \mathbf{w_2}, \ldots, \mathbf{w_N}, \alpha$ и $\boldsymbol{\theta}$ и параметризованная функция $G(\cdot, \mathbf{w}, \alpha, \theta) : [0,1]^n \to R$ такая, что для всех $\mathbf{x} \in [0,1]^n$ выполняется

$$\left|G(\mathbf{x}, \mathbf{w}, \alpha, \theta) - f(\mathbf{x})\right| < \varepsilon,$$

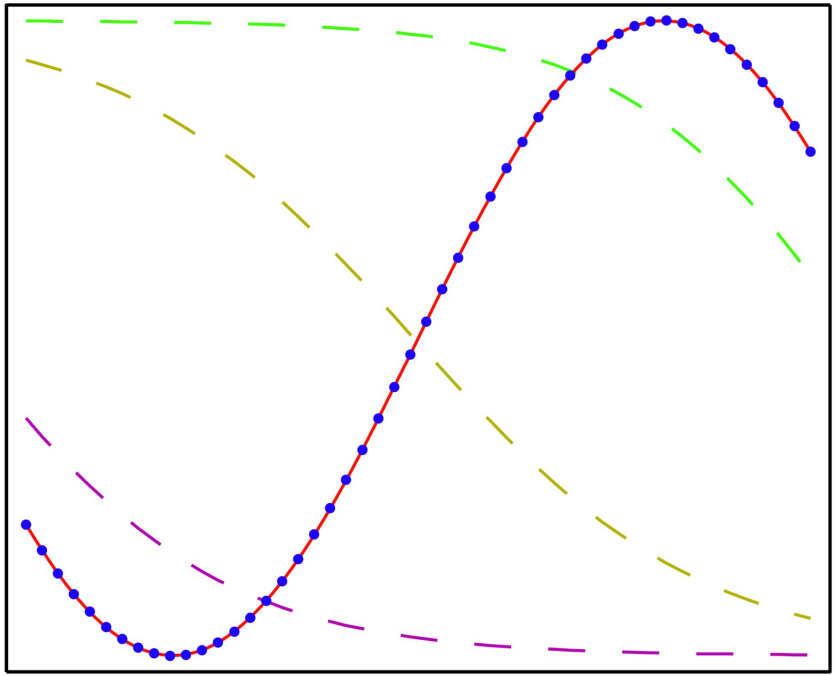
где

$$G(\mathbf{x}, \mathbf{w}, \alpha, \theta) = \sum_{i=1}^{N} \alpha_i \varphi(\mathbf{w}_i^T \mathbf{x} + \theta_i),$$

и $\mathbf{w}_i \in \mathbb{R}^n$, $\alpha_i, \theta_i \in \mathbb{R}$, $\mathbf{w} = (\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N)$, $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_N)$, и $\theta = (\theta_1, \theta_2, \ldots, \theta_N)$.

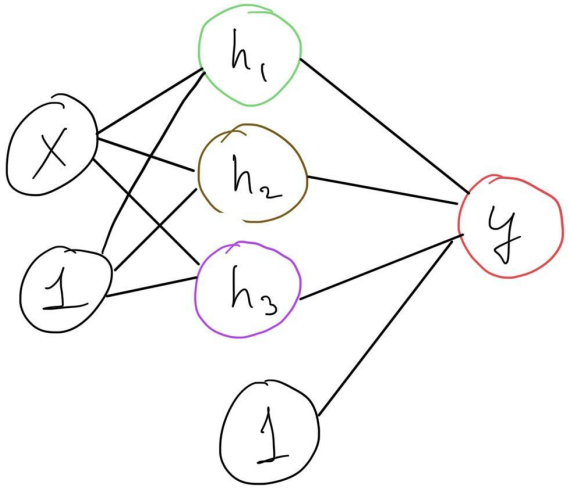
Cool, but impractical 🥴

More Toy Problems



N = 50 точек

MLP: 3 скрытых нейрона с tanh активацией



References

- Pattern recognition and machine learning (Bishop, 2006)
- Deep Learning (Goodfellow, 2014)
- Learning Theory from First Principles (Bach, 2023)
- Wikipedia
- Глубокое обучение (Николенко, 2018)
- Automatic Differentiation in Machine Learning: a Survey
- https://github.com/MALINAYAGODA
- https://github.com/DaniilSergeev17
- github.com/karpathy/micrograd