Лекция 2. Устройства полносвязных сетей

О лекторе

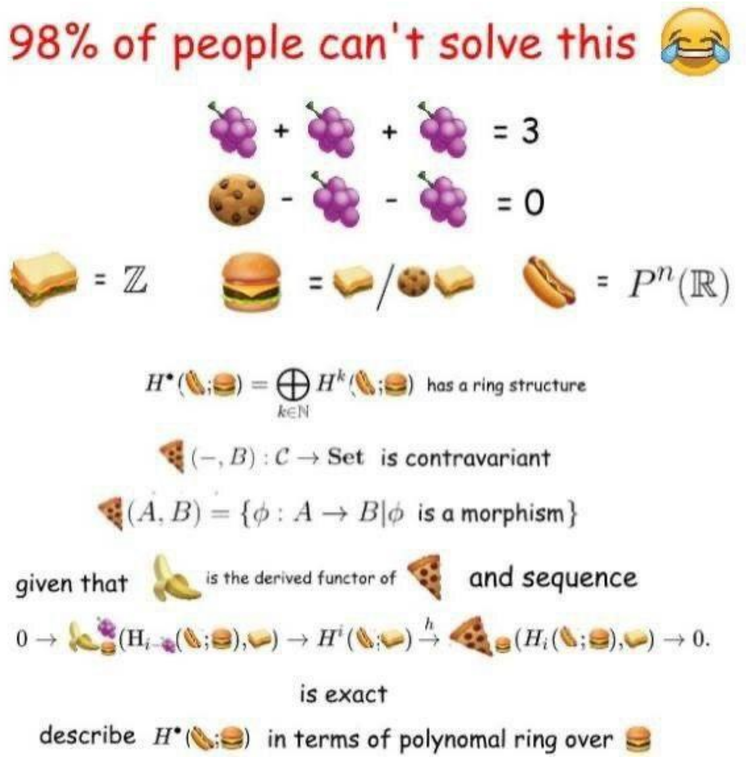
- Алеша
- ex Sber AI
- ex AIRI (New Materials Design)
- Tinkoff RL Research

github.com/zzmtsvv
t.me/zzmtsvv

План лекции

- От линейных моделей к нелинейным нейронным
- Связь с моделью нейрона 1940-ых
- Общий вид моделей прямого распространения
- Нотация
- Виды активаций
- Теоретические выкладки (немножко)
- Что происходит внутри нейронной сети
- Вопрос оптимизации

"Математики не будет"



Будет, но не Rocket Science

Высокоуровневый вид всех мл задач

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

From Linear to NonLinear Approximation (ANN)

$$y(\mathbf{x}, \mathbf{w}) = f\left(\sum_{j=1}^{M} w_j \phi_j(\mathbf{x})\right)$$

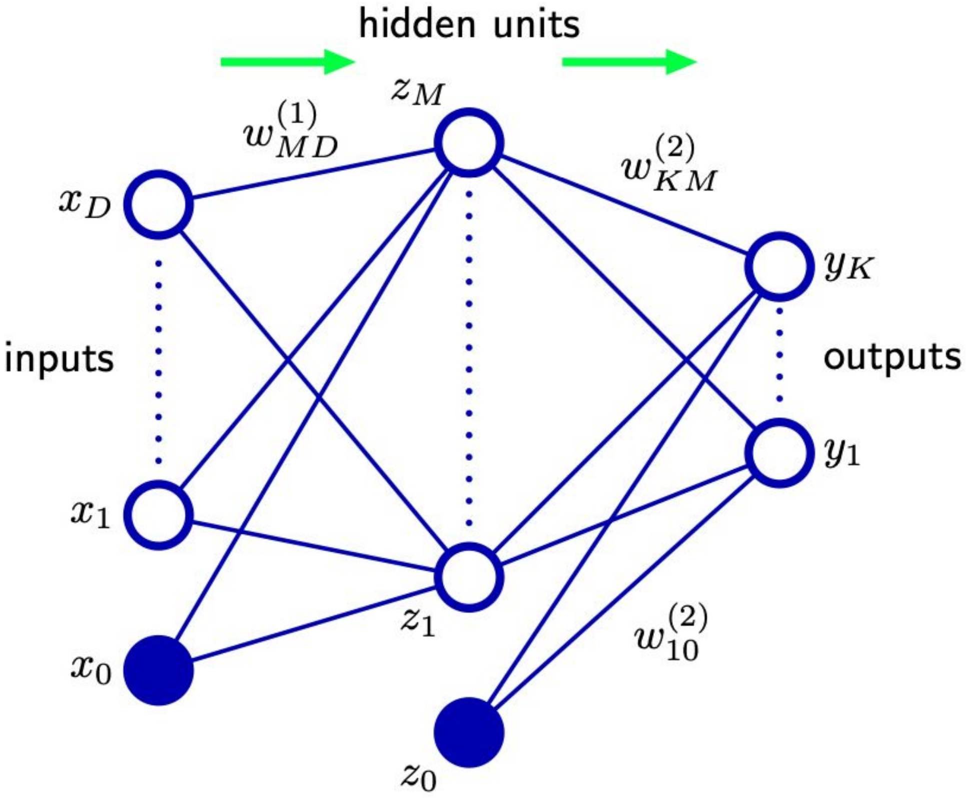
$$y(\mathbf{x}, \mathbf{w}) = f(\mathbf{w}^T \phi(\mathbf{x}))$$

$$\mathbf{x} \in \mathbb{R}^M$$

$f()$ - нелинейная функция в случае классификации, $f(x) = x$ в случае регрессии

$\phi_j(\mathbf{x})$ - "базисные" функции

Общий вид Feedforward Neural Network



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

$$z_j = h(a_j),$$

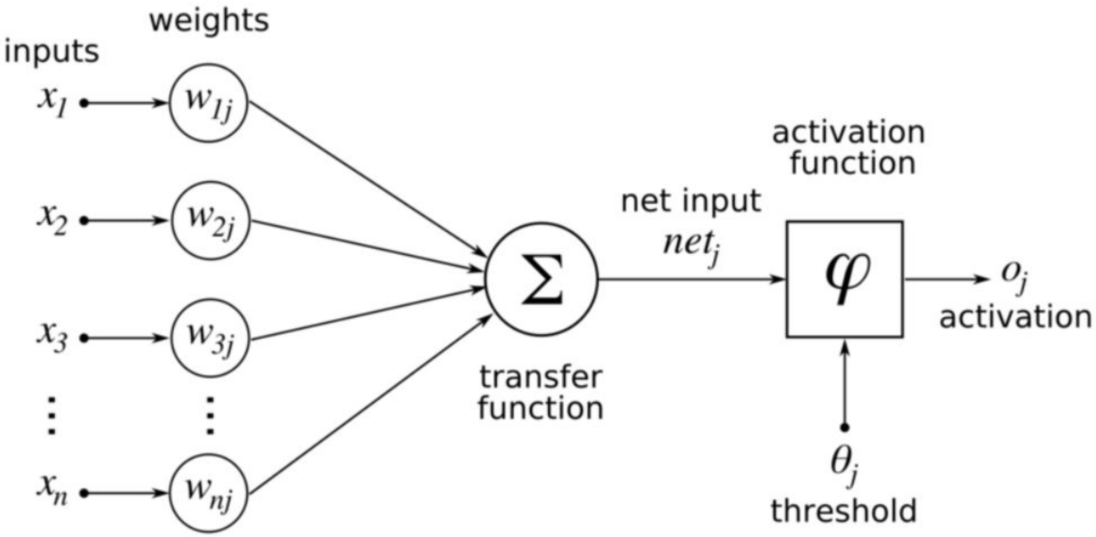
$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)}$$

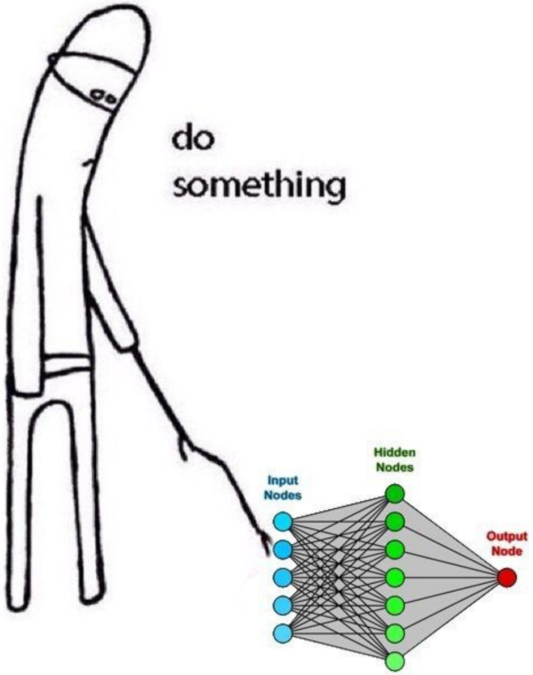
$$y_k = \sigma(a_k)$$

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

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

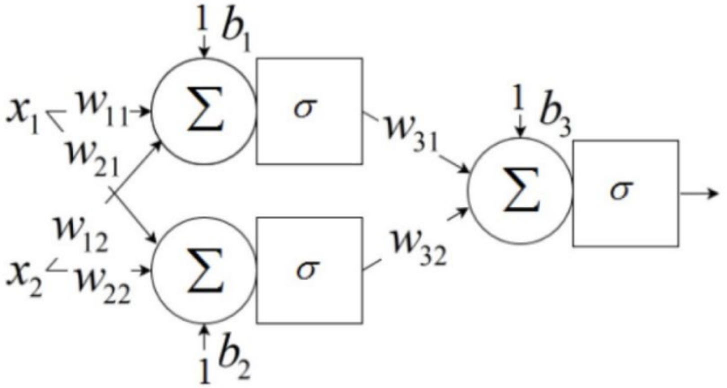
Связь с моделью нейрона 1940-ых





Переход к матричной форме

Двуслойная нейросеть



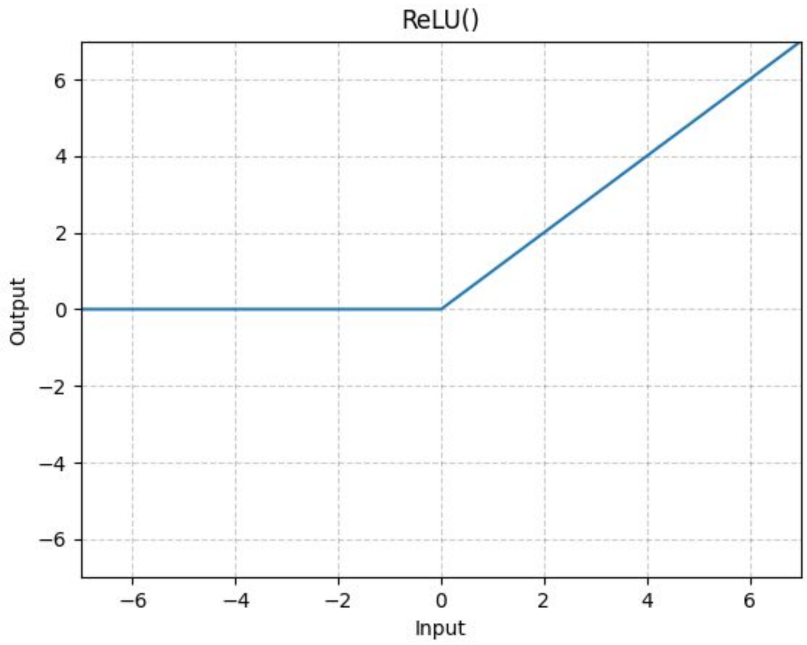
$$\sigma\left(\begin{bmatrix} w_{31} & w_{32} & b_3 \end{bmatrix}\begin{bmatrix} \sigma\left(\begin{bmatrix} w_{11} & w_{12} & b_1 \\ w_{21} & w_{22} & b_2 \end{bmatrix}\begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}\right) \\ 1 \end{bmatrix}\right)$$

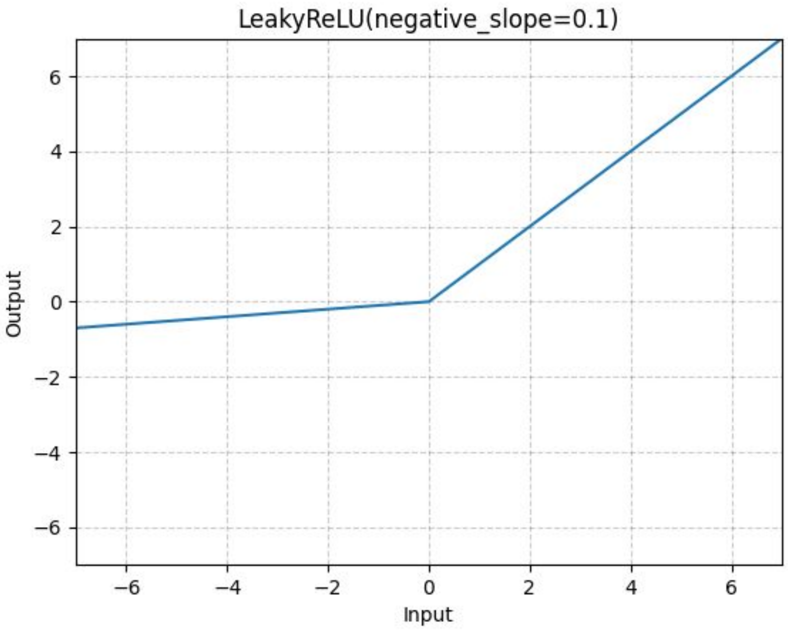
Classic Matrix Notation (by Goodfellow, 2014)

Require: Network depth, l
Require: $\boldsymbol{W}^{(i)}, i \in \{1, \dots, l\}$, the weight matrices of the model
Require: $\boldsymbol{b}^{(i)}, i \in \{1, \dots, l\}$, 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 $k = 1, \dots, l$ **do**
  $\boldsymbol{a}^{(k)} = \boldsymbol{b}^{(k)} + \boldsymbol{W}^{(k)} \boldsymbol{h}^{(k-1)}$
  $\boldsymbol{h}^{(k)} = f(\boldsymbol{a}^{(k)})$
end for
$\hat{\boldsymbol{y}} = \boldsymbol{h}^{(l)}$
$J = L(\hat{\boldsymbol{y}}, \boldsymbol{y}) + \lambda \Omega(\theta)$

Виды активаций

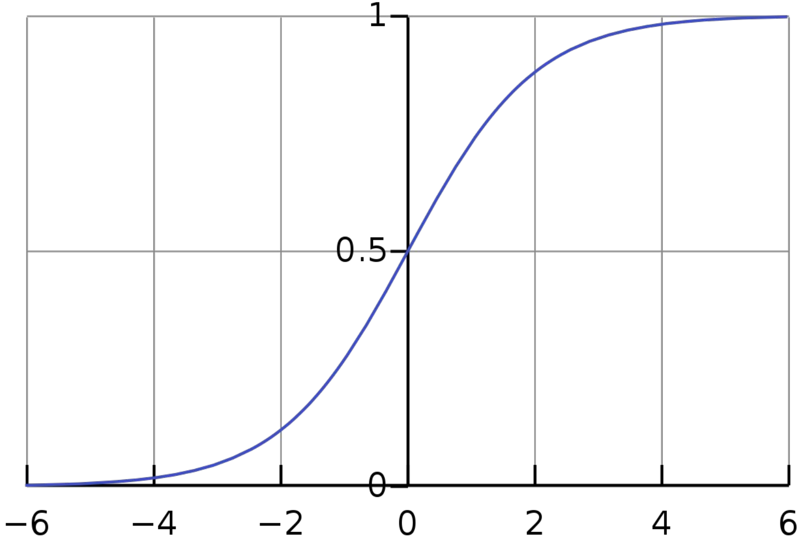


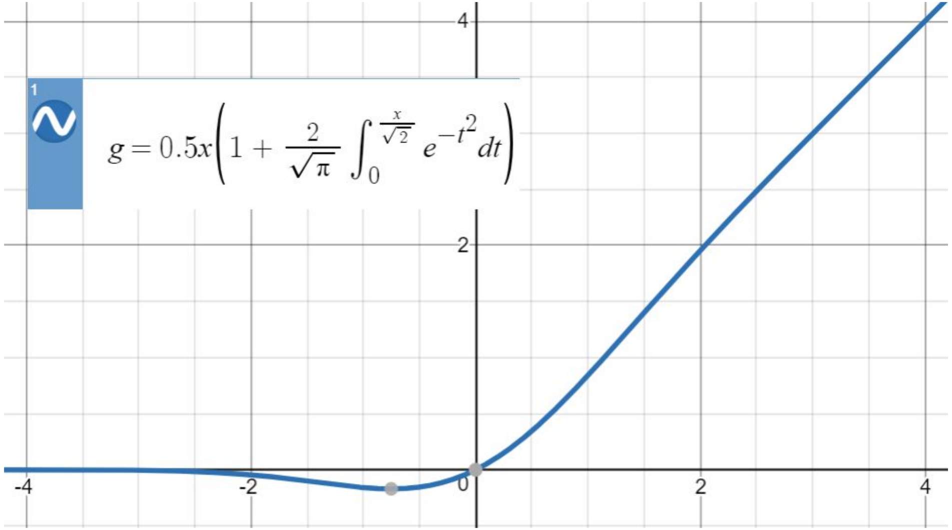


$\mathrm{ReLU}(x)=\max(0,x)$

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

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	$\log\left(1+e^x\right)$	$\frac{1}{1+e^{-x}}$
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	$\begin{cases}\alpha\left(e^x-1\right), & x<0\\ x, & x\geq 0\end{cases}$	$\begin{cases}f(x)+\alpha, & x<0\\ 1, & x\geq 0\end{cases}$

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 $\mathbf{w_1}, \mathbf{w_2}, \dots, \mathbf{w_N}, \alpha$ 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

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

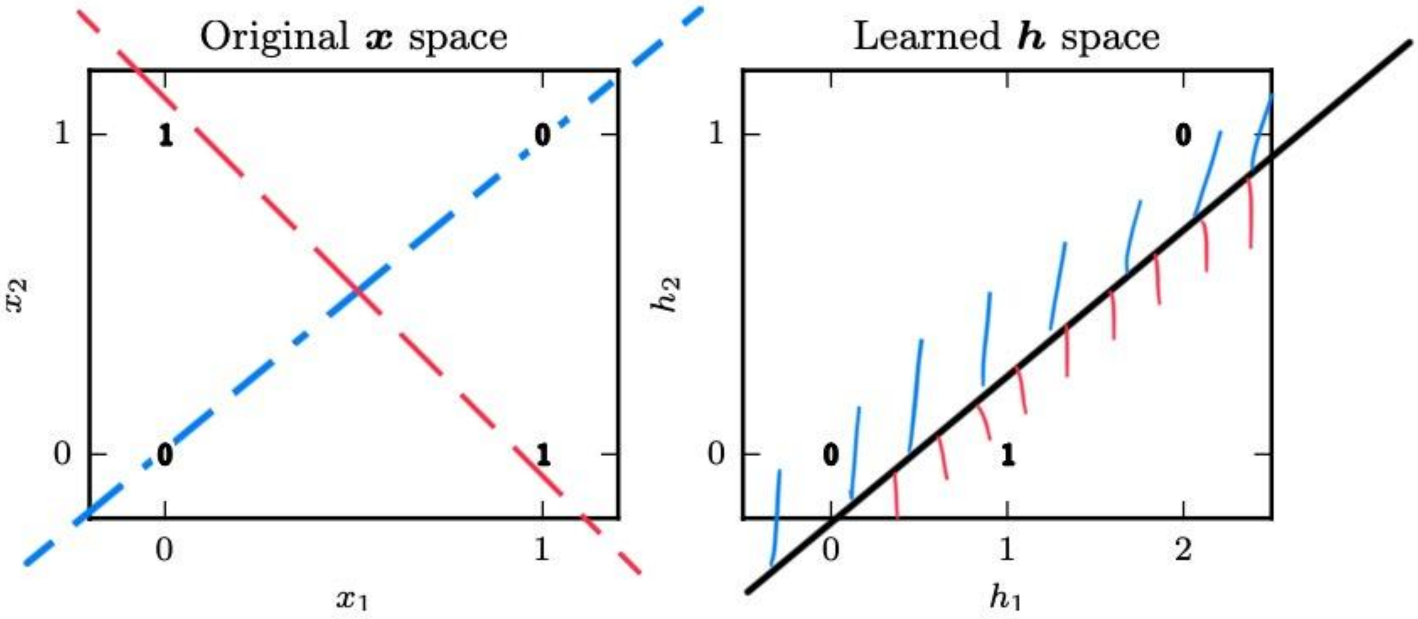
где

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

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

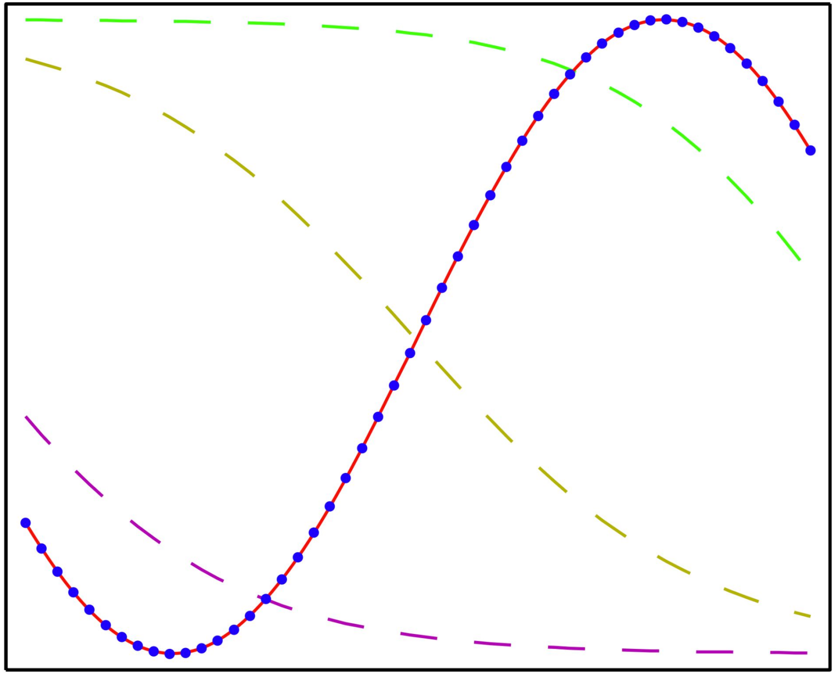
Cool, but impractical 🥴

XOR Problem



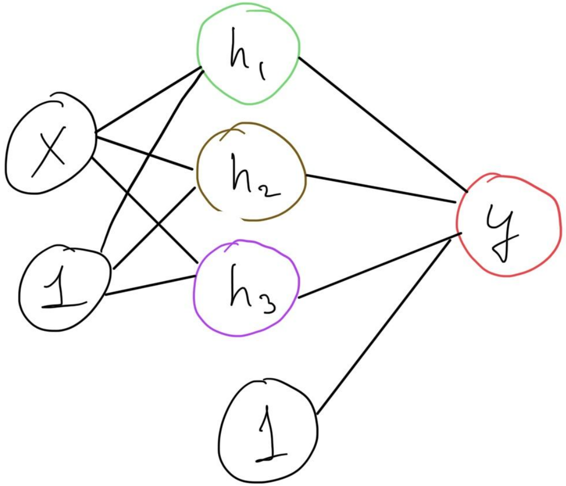
Original $\boldsymbol{x}$ space
x_1
x_2
0
1
0
1
1
0
0
1
Learned $\boldsymbol{h}$ space
h_1
h_2
0
1
2
0
1
0
0
1

More Toy Problems

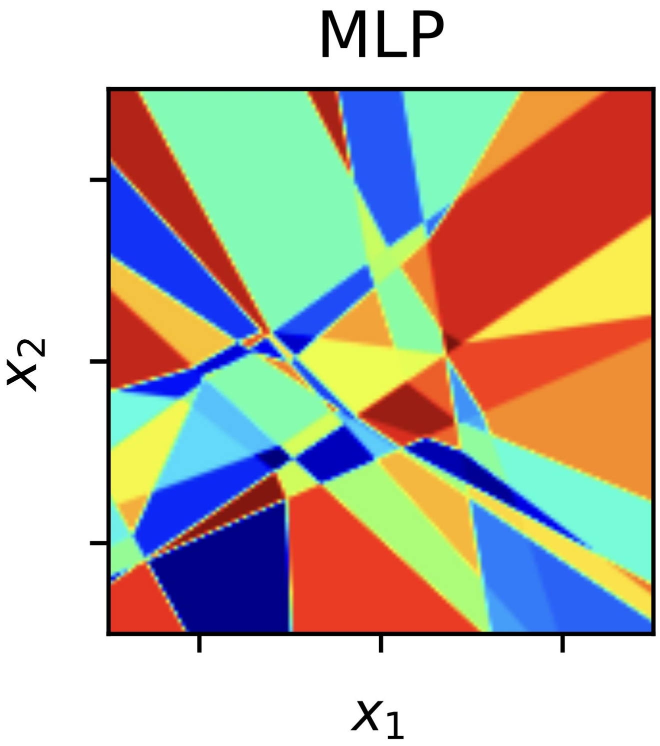


N = 50 точек

MLP: 3 скрытых нейрона с tanh активацией



ReLU MLP



Оказывается, перцептрон с активациями ReLU аппроксимирует кусочно-линейную функцию очень удобным образом!!

Проблема неединственности и численного диапазона

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c}, \boldsymbol{w}, b) = \boldsymbol{w}^\top \max\{0, \boldsymbol{W}^\top \boldsymbol{x} + \boldsymbol{c}\} + b.$$

$$=$$

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c}, \boldsymbol{w}, b) = \underset{10w^\top}{\cancel{\boldsymbol{w}}^\top} \max\{0, \underset{\frac{1}{10}W^\top}{\cancel{\boldsymbol{W}}^\top} \boldsymbol{x} + \underset{\frac{1}{10}c}{\cancel{\boldsymbol{c}}}\} + b.$$

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

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

Виды вычисления градиента

$$\frac{\partial f(\mathbf{x})}{\partial x_i} \approx \frac{f(\mathbf{x} + h\mathbf{e}_i) - f(\mathbf{x})}{h}$$

$$\frac{d}{dx}(f(x)g(x)) = \left(\frac{d}{dx}f(x)\right)g(x) + f(x)\left(\frac{d}{dx}g(x)\right)$$

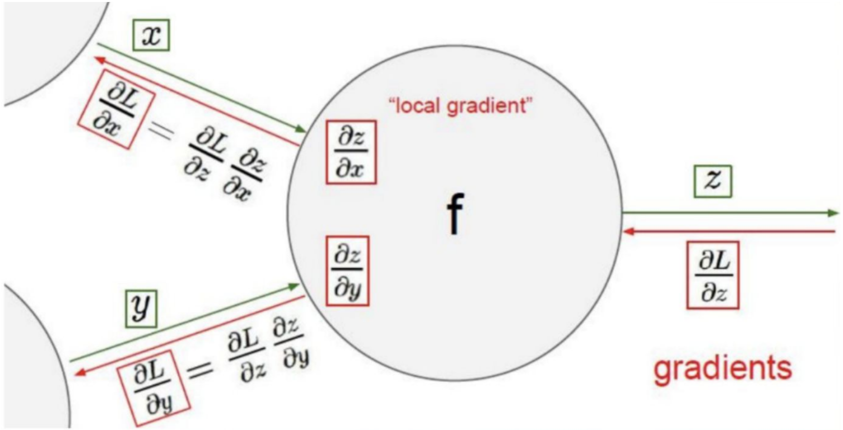
Численное дифференцирование

- вызов функции на каждое вычисление производной (покоординатно!)
- Two commandments of numerical analysis: “thou shalt not add small numbers to big numbers”, and “thou shalt not subtract numbers which are approximately equal”.

Символьное дифференцирование

- Аналитический вывод для данной функции
- При изменении входной функции приходится рассчитывать все заново

Autograd (Automatic Differentiation)



$$L = \sum_t \text{loss}(y_t - f(x_t, w)) \sim L(w, f_1(w), \ldots, f_k(w))$$

$$\nabla L(w, f_1(w), \ldots, f_k(w)) = \frac{\partial L}{\partial f_1}\nabla f_1(w) + \ldots + \frac{\partial L}{\partial f_k}\nabla f_k(w)$$

Autograd (Automatic Differentiation)

```python
class Add(Function):
    @staticmethod
    def forward(ctx: Ctx,
                a: tensor.Tensor,
                b: tensor.Tensor) -> tensor.Tensor:
        return tensor.Tensor(a.data + b.data)

    @staticmethod
    def backward(ctx, grad_in):
        return [grad_in, grad_in]
```

```python
class Mul(Function):
    @staticmethod
    def forward(ctx: Ctx,
                a: tensor.Tensor,
                b: tensor.Tensor) -> tensor.Tensor:
        ctx.save_for_backward(a, b)
        return tensor.Tensor(a.data * b.data)

    @staticmethod
    def backward(ctx: Ctx, grad_in):
        a, b = ctx.saved
        return [grad_in * b.data, grad_in * a.data]
```

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
- github.com/zzmtsvv/smaller_micrograd
- github.com/karpathy/micrograd