Факультет математики и компьютерных наук СПбГУ

Машинное обучение

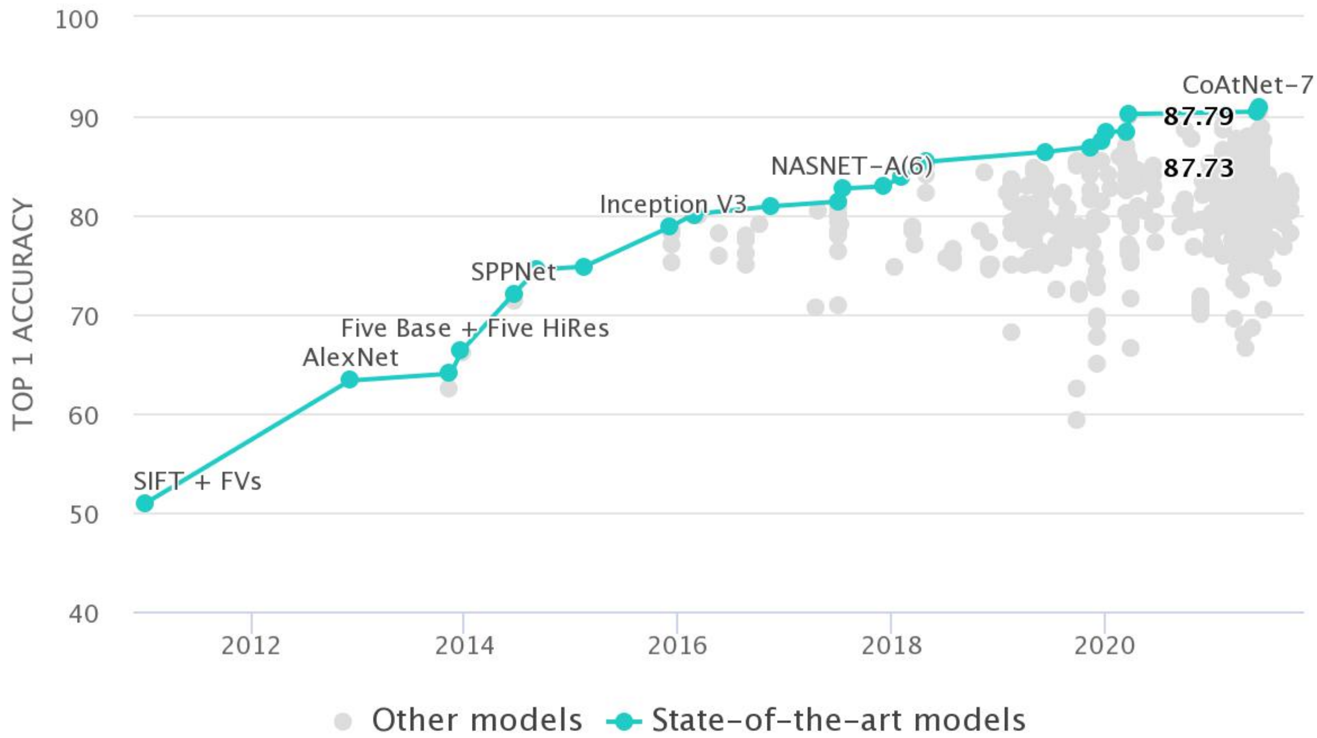
Лекция 10. Введение в нейронные сети

19 ноября 2021

Пятиминутка

1. Приведите несколько примеров композиций моделей
2. Опишите алгоритм AdaBoost
3. Назовите популярные реализации градиентного бустинга и их отличительные особенности

Задачи, решаемые нейросетями



[https://paperswithcode.com/sota/image-classification-on-imagenet (https://paperswithcode.com/sota/image-classification-on-imagenet)](https://paperswithcode.com/sota/image-classification-on-imagenet)

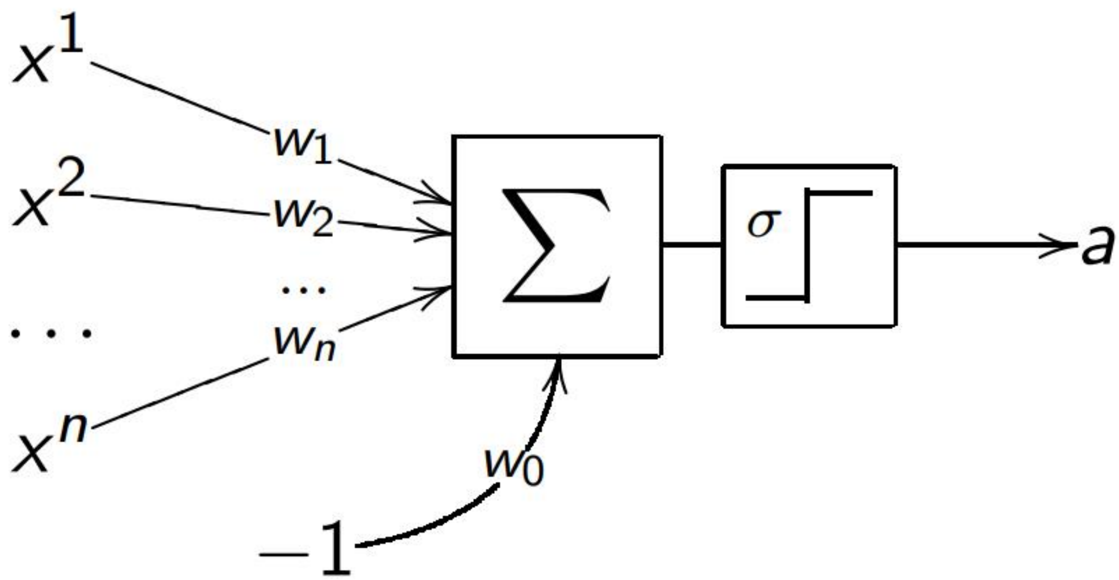
Линейная модель (напоминание)

$f_j : X \to \mathbb{R}$ — числовые признаки

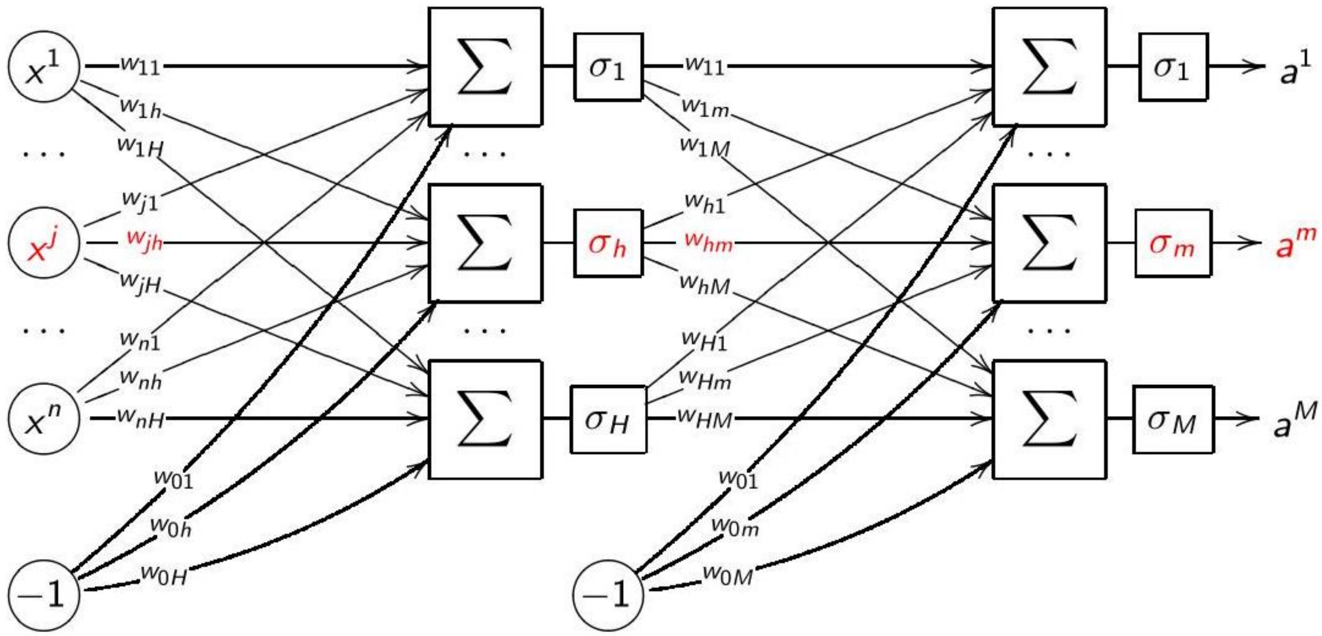
$$a(x, w) = \sigma(\langle w, x \rangle) = \sigma\left(\sum_{j=1}^{n} w_j f_j(x) - w_0\right),$$

где $w_0, w_1, \ldots, w_n \in \mathbb{R}$ — веса признаков

$\sigma(z)$ — функция активации, например, $\operatorname{sign}(z)$, $\frac{1}{1+e^{-z}}$, $(z)_+$



Нейронная сеть как комбинация линейных моделей



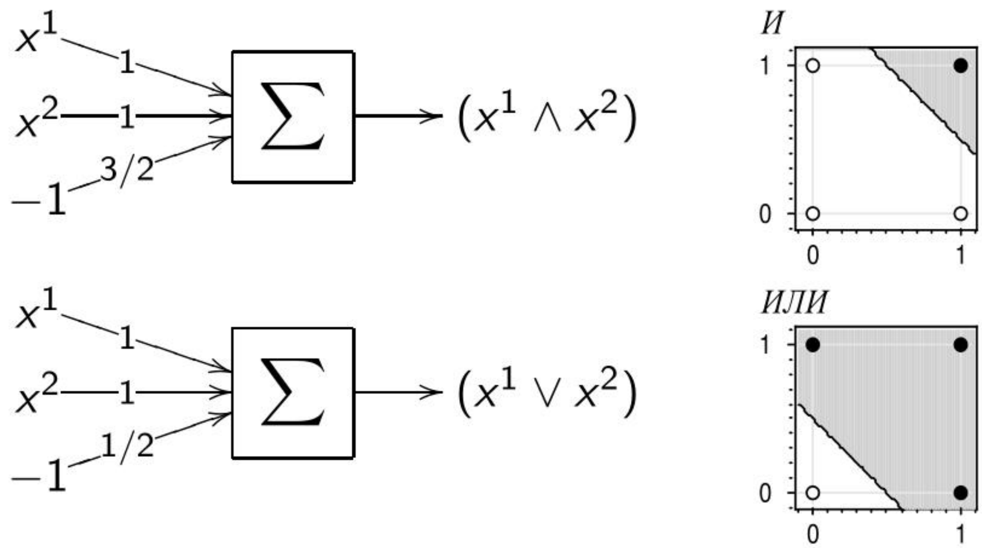
Нейронная реализация логических функций

Функции И, ИЛИ, НЕ от бинарных переменных x^1 и x^2:

$$x^1 \wedge x^2 = [x^1 + x^2 - \tfrac{3}{2} > 0]$$

$$x^1 \vee x^2 = [x^1 + x^2 - \tfrac{1}{2} > 0]$$

$$\neg x^1 = [-x^1 + \tfrac{1}{2} > 0]$$



Логическая функция XOR (исключающее ИЛИ)

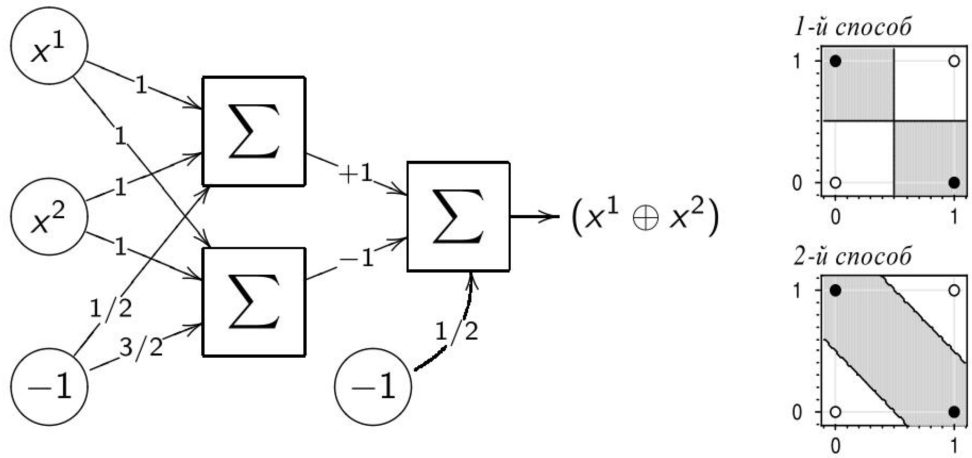
Функция $x^1 \bigoplus x^2 = [x^1 \neq x^2]$

не реализуема одним нейроном. Два способа реализации:

- Добавлением нелинейного признака:

  $x^1 \bigoplus x^2 = [x^1 + x^2 - 2x^1x^2 - \frac{1}{2} > 0]$
- Сетью (двухслойной суперпозицией) функций И, ИЛИ, НЕ:

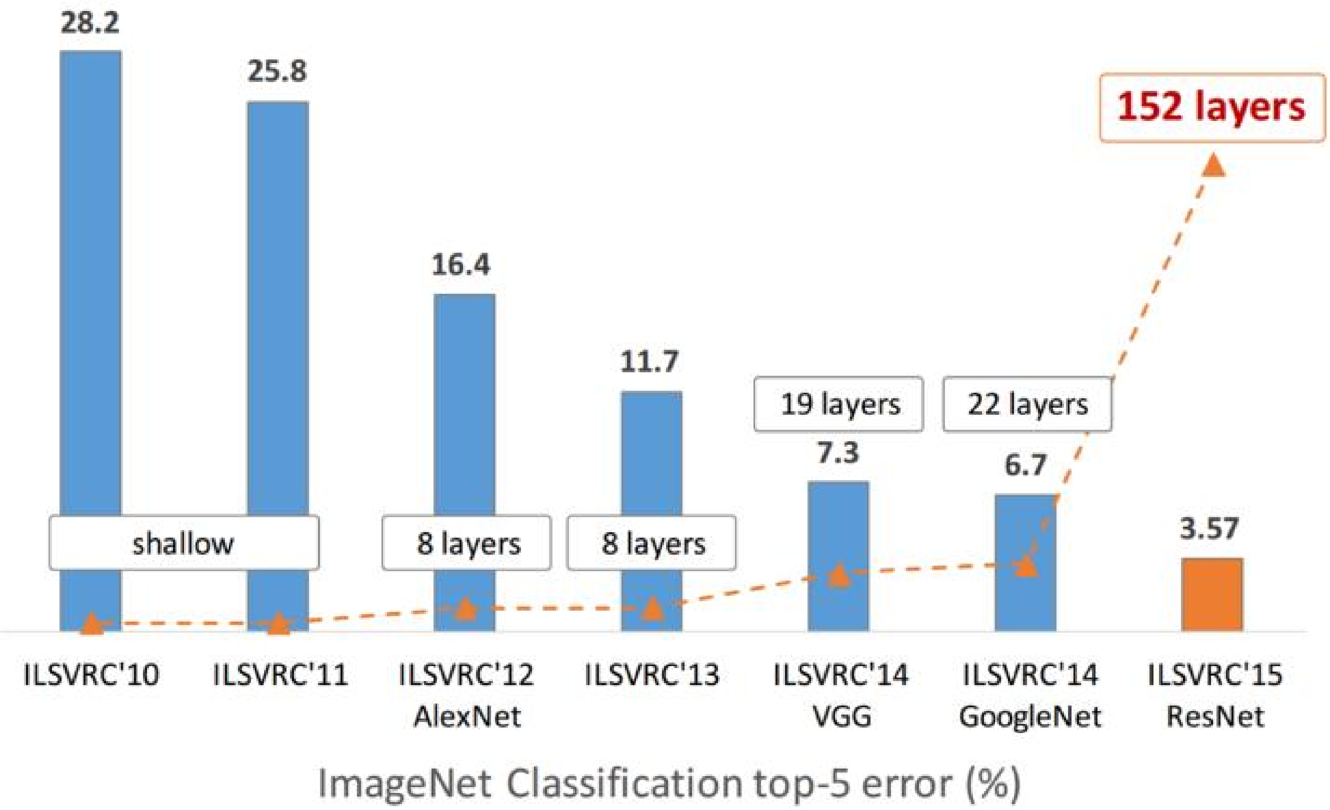
  $x^1 \bigoplus x^2 = [(x^1 \vee x^2) - (x^1 \wedge x^2) - \frac{1}{2} > 0]$.



Выразительная способность нейронной сети

- Двухслойная сеть в $\{0, 1\}^n$ позволяет реализовать произвольную булеву функцию
- Двухслойная сеть в $\mathbb{R}^n$ позволяет отделить произвольный выпуклый многогранник
- Трёхслойная сеть в $\mathbb{R}^n$ позволяет отделить произвольную многогранную область (может быть не выпуклой и не связной)
- С помощью линейных операций и одной нелинейной функции активации σ можно приблизить любую непрерывную функцию с любой желаемой точностью
- Для некоторых специальных классов глубоких нейронных сетей доказано, что они обладают экспоненциально большей выразительной силой, чем неглубокие сети. [V. Khrulkov, A. Novikov, I. Oseledets. Expressive power of recurrent neural networks, Feb 2018, ICLR 2018 (https://arxiv.org/pdf/1711.00811.pdf)](https://arxiv.org/pdf/1711.00811.pdf)

Развитие свёрточных сетей (или краткая история ImageNet)



Выразительная способность нейронной сети

Функция $\sigma(z)$ — сигмоида, если $\lim\limits_{z\to-\infty}\sigma(z)=0$ и $\lim\limits_{z\to+\infty}\sigma(z)=1$

Теорема Цыбенко (опирается на теорему Колмогорова о представимости многомерных функций)

Если $\sigma(z)$ — непрерывная сигмоида, то для любой непрерывной на $[0,1]^n$ функции $f(x)$ существуют такие значения параметров $w_h\in\mathbb{R}^n,\ w_0\in\mathbb{R},\ \alpha_h\in\mathbb{R}$, что однослойная сеть

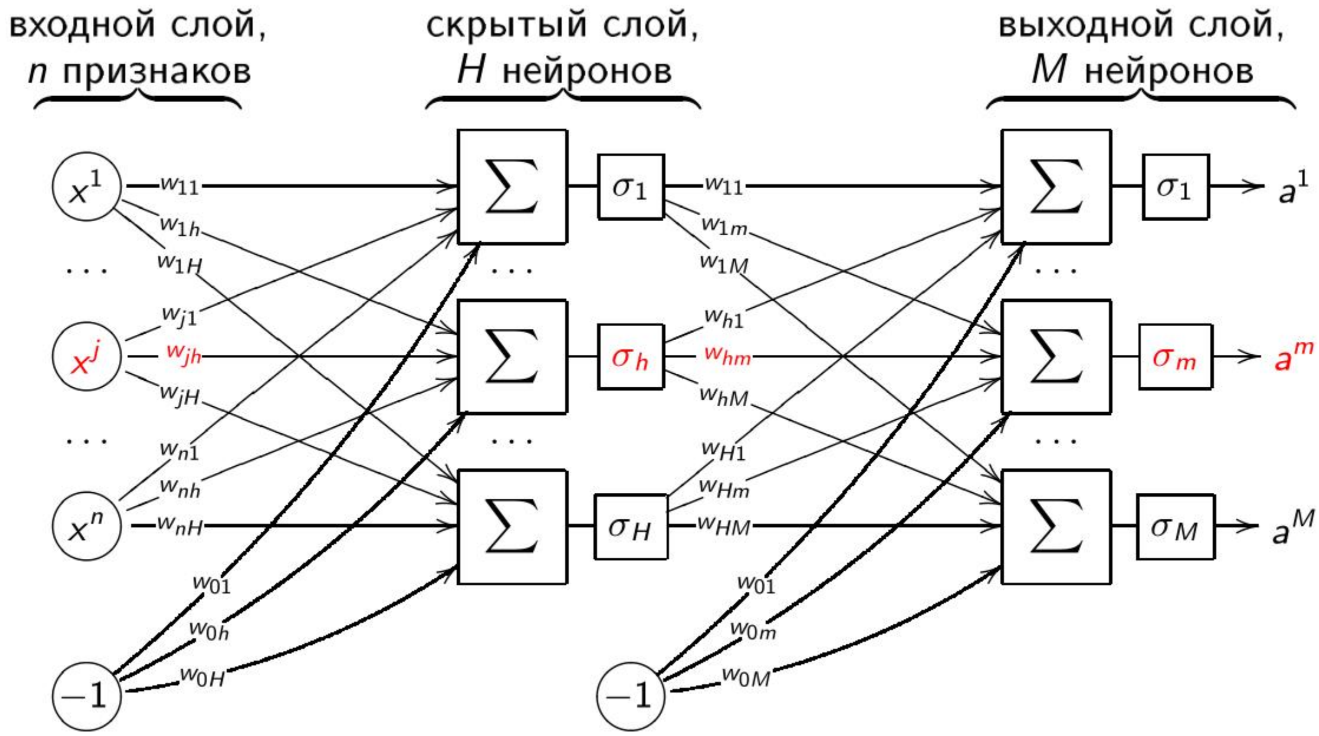
$$a(x)=\sum_{h=1}^{H}\alpha_h\sigma(\langle x,w_h\rangle-w_0)$$

равномерно приближает $f(x)$ с любой точностью ε:

$|a(x)-f(x)|<\varepsilon$, для всех $x\in[0,1]^n$

G. Cybenko. Approximation by Superpositions of a Sigmoidal Function. Mathematics of Control, Signals, and Systems (MCSS) 2 (4): 303--314 (Dec 1, 1989)

Двухслойная нейронная сеть с M-мерным выходом

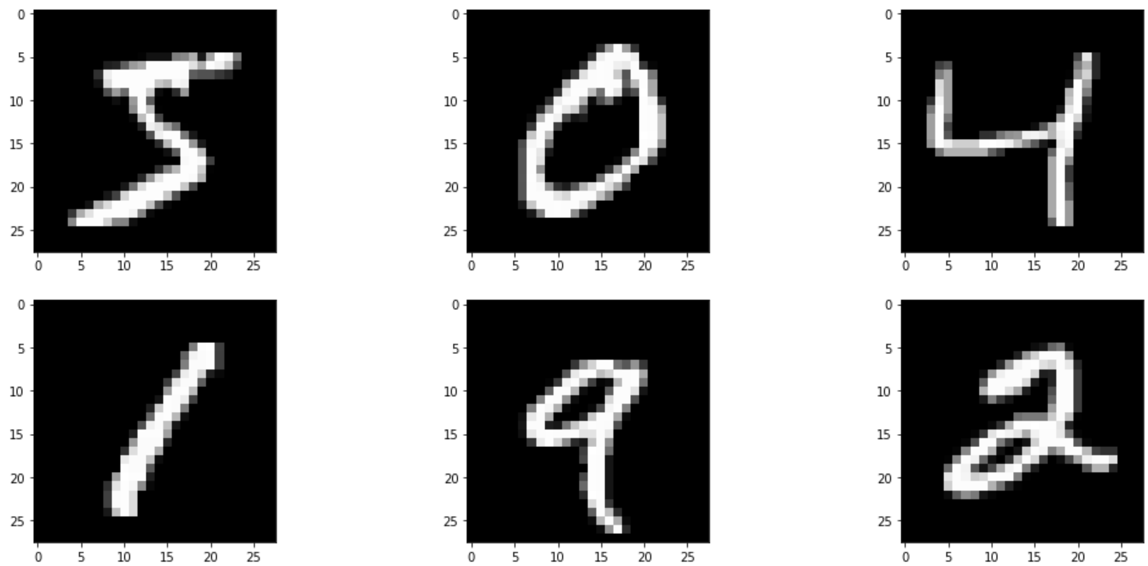


Вектор параметров модели $w \equiv (w_{jh}, w_{hm}) \in \mathbb{R}^{Hn+H+MH+M}$

> **Вопрос 1:** Что лучше — увеличивать число слоев (глубину) или количество нейронов в слое (ширину)?

```
In [1]:
```

In [2]:



In [4]:

Out[4]: (60000, 28, 28)

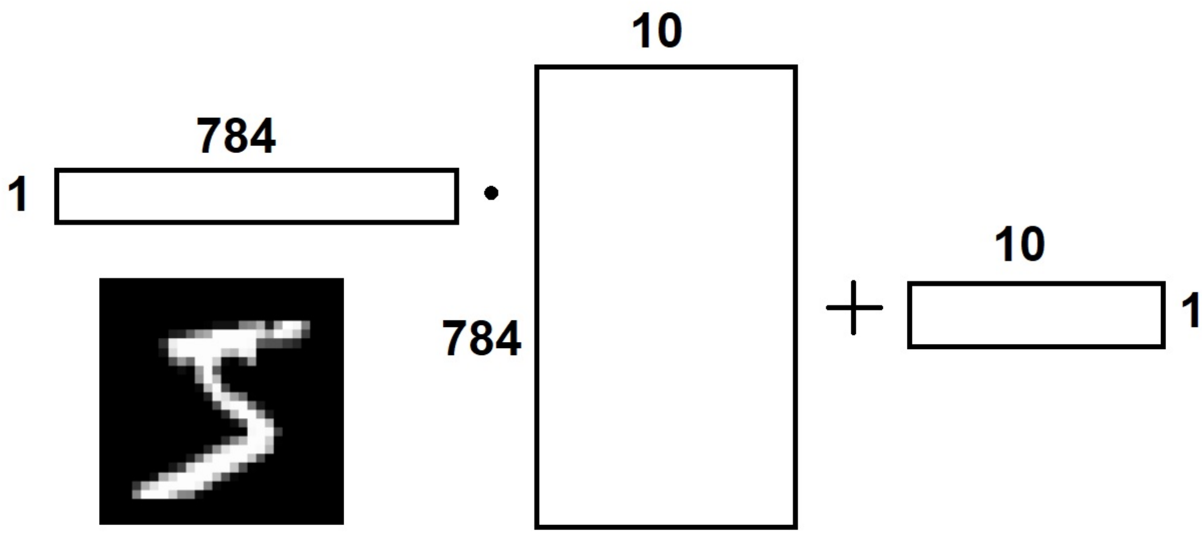
In [5]:

Out[5]:
```
array([  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   3,
        18,  18,  18, 126, 136, 175,  26, 166, 255, 247, 127,   0,   0,
         0,   0], dtype=uint8)
```

Линейный классификатор

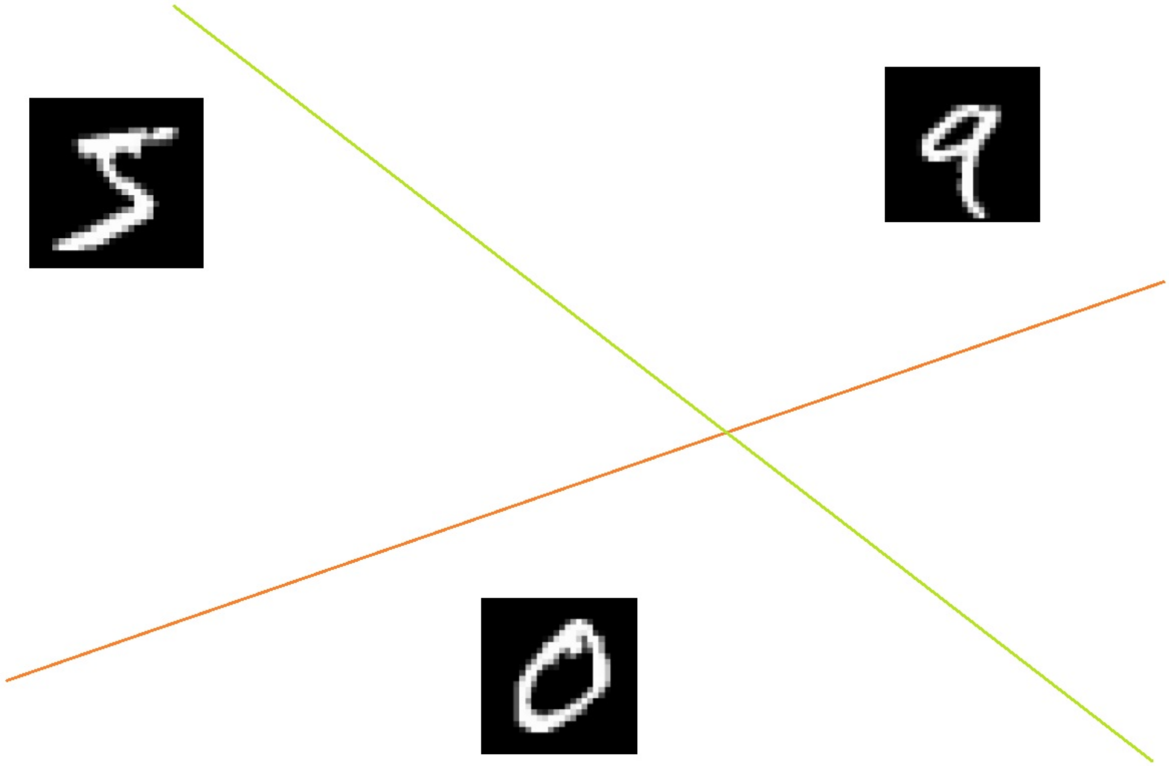
Предсказание

$$y_{pred} = x \cdot W + b$$



$$x \quad \cdot \quad W \quad + \quad b$$

Десять разделяющих плоскостей



- В нашем примере пространство 784-мерное ($\mathbb{R}^{784}$)

Вопрос 2: Как найти лучшие параметры: матрицу весов W и смещение b?

Если бы $\mathrm{y\_true}_i \in \mathbb{R}$ (то есть задача линейной регресии), то для минимизации суммы квадратов разностей (метод наименьших квадратов) ответ вычисляется **аналитически** формулой:

$$\hat{W} = (X^T X)^{-1} X^T \mathrm{y\_true}$$

В общем случае решается **численно** минимизацией функции потерь. Чаще всего градиентным спуском (gradient descent).

[Distill.pub momentum (https://distill.pub/2017/momentum/)](https://distill.pub/2017/momentum/)

Softmax — для классификации

Переводим наши ответы линейной модели в вероятности классов:

$$p(c=0|x) = \frac{e^{y_0}}{e^{y_0} + e^{y_1} + \dots + e^{y_n}} = \frac{e^{y_0}}{\sum\limits_i e^{y_i}}$$

$$p(c=1|x) = \frac{e^{y_1}}{e^{y_0} + e^{y_1} + \dots + e^{y_n}} = \frac{e^{y_1}}{\sum\limits_i e^{y_i}}$$

$$\dots$$

Принцип максимального правдоподобия (напоминание)

$$\arg\max_w P(Y|w, X)P(w) = \arg\max_w \prod_{i=1}^{\ell} P(y_i|w, x_i)P(w) =$$

$$\arg\max_w \sum_{i=1}^{\ell} \log P(y_i|w, x_i) + \log P(w)$$

Минимизация функции потерь

$$L(w) = \sum_{i=1}^{\ell} \mathcal{L}(y_i, x_i, w) = -\ln P(y_i|w, x_i) \to \min_w$$

- это cross-entropy loss для случая $y_i \in \{0, 1\}$
- в нашем случае

$$L(W, b) = -\sum_j \ln \frac{e^{(x_j W + b)_{y_j}}}{\sum_i e^{(x_j W + b)_i}}$$

- минимум функции находим стохастическим градиентным спуском

$$W^{k+1} = W^k - \eta \frac{\partial L}{\partial W}$$
$$b^{k+1} = b^k - \eta \frac{\partial L}{\partial b}$$

Вопрос 3: Почему такая функция потерь — это cross-entropy loss?

Обучение по мини-подвыборкам (mini-batch)

Снижаем разброс градиента.

Вход: выборка X^ℓ, темп обучения η, темп забывания λ

Выход: вектор весов $w \equiv (w_{jh}, w_{hm})$

1. инициализировать веса
2. инициализировать оценку функционала

$$Q = \frac{1}{\ell}\sum_{i=1}^{\ell}\mathcal{L}_i(w)$$

3. **повторять**
   A. выбрать M объектов x_i из X^ℓ случайным образом
   B. вычислить потерю: $\varepsilon = \frac{1}{M}\sum_{i=1}^{M}\mathcal{L}_i(w)$
   C. сделать градиентный шаг: $w = w - \eta\frac{1}{M}\sum_{i=1}^{M}\nabla\mathcal{L}_i(w)$
   D. оценить функционал: $Q = \lambda\varepsilon + (1-\lambda)Q$
4. **пока** значение Q и/или веса w не сойдутся

Однослойная нейросеть для классификации

Для построения нейронной сети на Python продолжим работать с библиотекой keras. Это в свою очередь высокоуровневая надстройка над tensorflow. Большим ее преимуществом является интерфейс, совместимый с sklearn.

In [4]:

Для того, чтобы градиенты были более стабильными, поделим входные данные на 255 (чтобы они были из диапазона [0,1]).

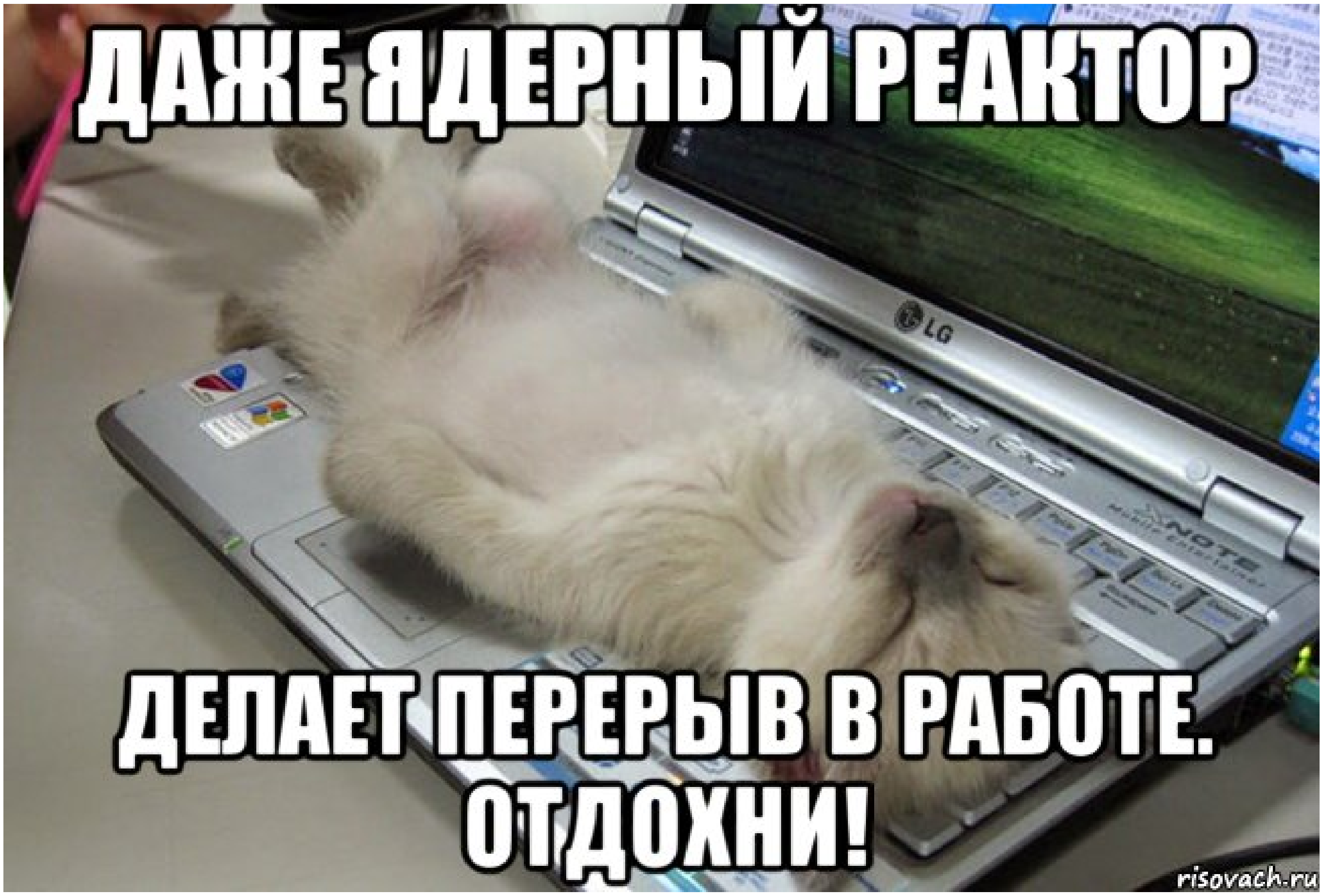
И запустим обучение!

In [5]:

```
Epoch 1/20
1875/1875 [==============================] - 3s 1ms/step - loss: 0.7832 -
accuracy: 0.8151
Epoch 2/20
1875/1875 [==============================] - 3s 2ms/step - loss: 0.4565 -
accuracy: 0.8803
Epoch 3/20
1875/1875 [==============================] - 2s 1ms/step - loss: 0.4037 -
accuracy: 0.8911
Epoch 4/20
1875/1875 [==============================] - 2s 1ms/step - loss: 0.3771 -
accuracy: 0.8969
Epoch 5/20
1875/1875 [==============================] - 2s 1ms/step - loss: 0.3604 -
accuracy: 0.9003
Epoch 6/20
1875/1875 [==============================] - 2s 1ms/step - loss: 0.3486 -
accuracy: 0.9035
Epoch 7/20
1875/1875 [==============================] - 3s 1ms/step - loss: 0.3396 -
accuracy: 0.9057
Epoch 8/20
1875/1875 [==============================] - 2s 1ms/step - loss: 0.3323 -
accuracy: 0.9075
Epoch 9/20
1875/1875 [==============================] - 2s 1ms/step - loss: 0.3265 -
accuracy: 0.9095
Epoch 10/20
1875/1875 [==============================] - 3s 1ms/step - loss: 0.3216 -
accuracy: 0.9105
Epoch 11/20
1875/1875 [==============================] - 2s 1ms/step - loss: 0.3174 -
accuracy: 0.9120
Epoch 12/20
1875/1875 [==============================] - 2s 1ms/step - loss: 0.3137 -
accuracy: 0.9129
Epoch 13/20
1875/1875 [==============================] - 2s 1ms/step - loss: 0.3105 -
accuracy: 0.9141
Epoch 14/20
1875/1875 [==============================] - 2s 1ms/step - loss: 0.3077 -
accuracy: 0.9147
Epoch 15/20
1875/1875 [==============================] - 3s 1ms/step - loss: 0.3050 -
accuracy: 0.9151
Epoch 16/20
1875/1875 [==============================] - 3s 2ms/step - loss: 0.3027 -
accuracy: 0.9161
Epoch 17/20
1875/1875 [==============================] - 3s 2ms/step - loss: 0.3006 -
accuracy: 0.9167
Epoch 18/20
1875/1875 [==============================] - 2s 1ms/step - loss: 0.2986 -
accuracy: 0.9173
Epoch 19/20
1875/1875 [==============================] - 3s 2ms/step - loss: 0.2969 -
accuracy: 0.9172
```

```
Epoch 20/20
1875/1875 [==============================] - 2s 1ms/step - loss: 0.2951 -
accuracy: 0.9179
```

Out[5]: `<keras.callbacks.History at 0x1b841f99648>`



Посмотрим, какое качество получилось

In [6]:

Out[6]: `array([7, 2, 1, ..., 4, 5, 6], dtype=int64)`

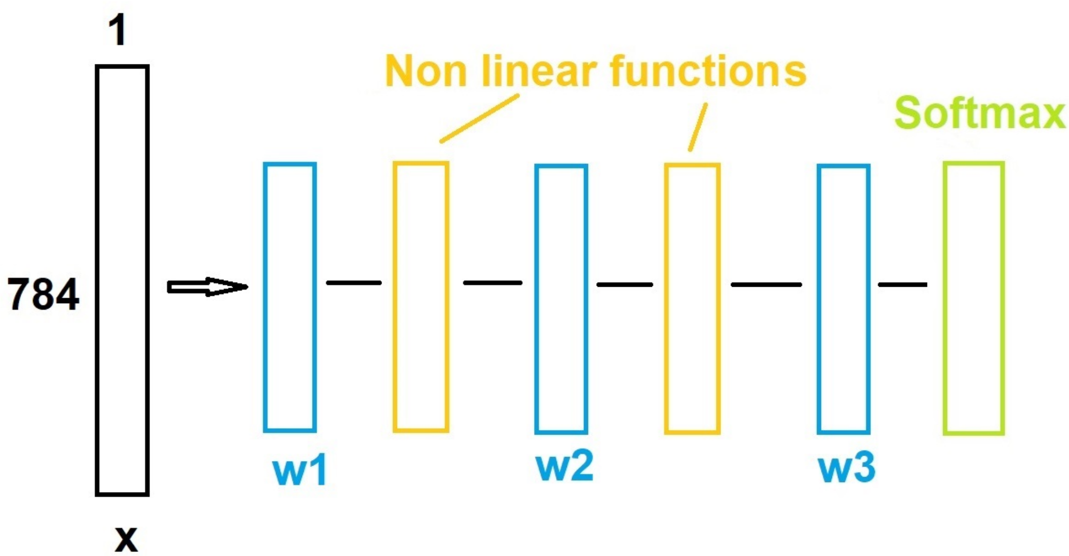
In [7]:

Out[7]: `0.9201`

In [8]:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
flatten (Flatten)            (None, 784)               0         
_________________________________________________________________
dense (Dense)                (None, 10)                7850      
=================================================================
Total params: 7,850
Trainable params: 7,850
Non-trainable params: 0
_________________________________________________________________
```

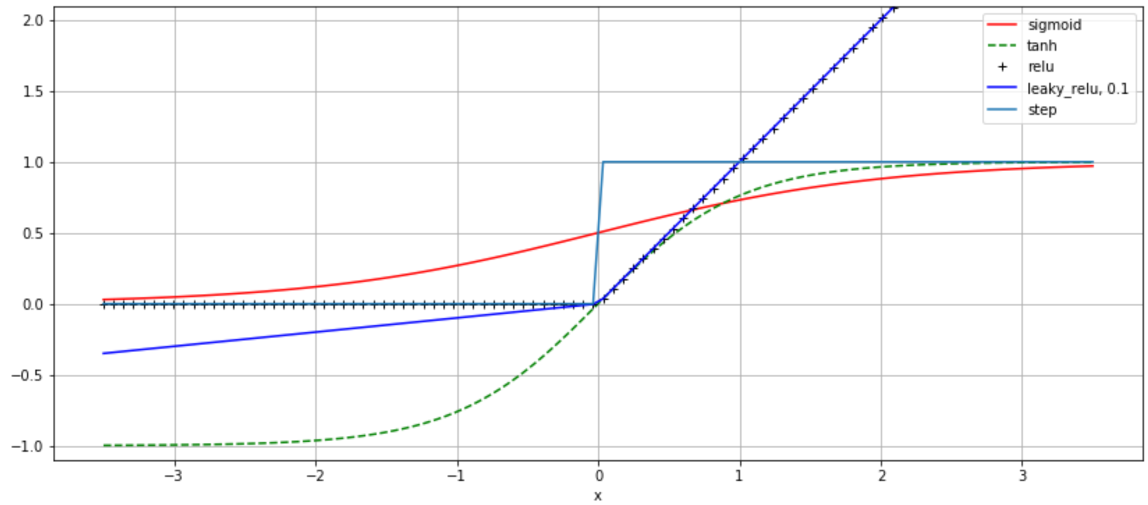
Нейронная сеть



Нелинейные функции: ReLU, PReLU (LeakyReLU)

$$\mathrm{ReLU}(y) = \max(0, y),$$
$$\mathrm{PReLU}(y) = \max(0, y) + \alpha \min(0, y)$$

In [1]:

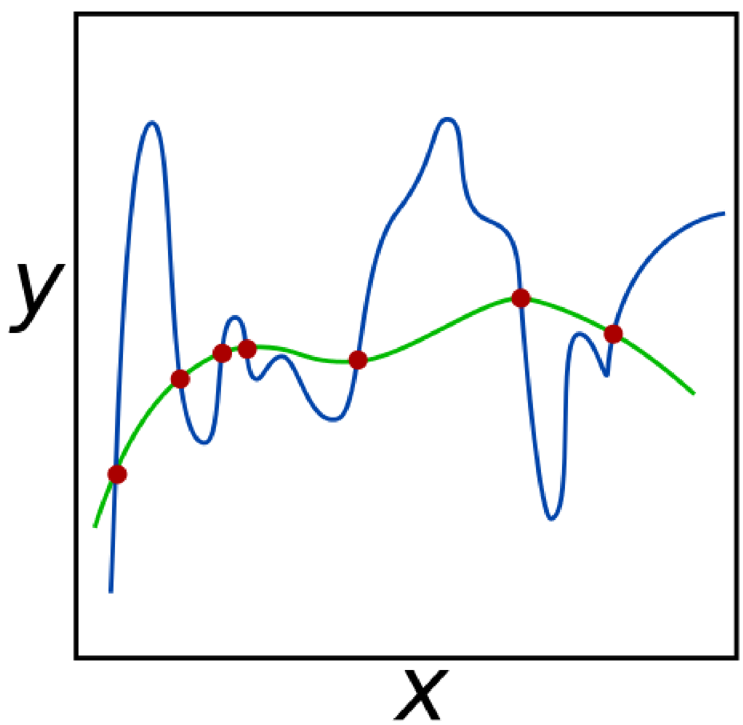


Регуляризация

$$L(W,b) = -\sum_j \ln \frac{e^{(x_j W + b)_{y_j}}}{\sum\limits_i e^{(x_j W + b)_i}} + \lambda R(W,b)$$

$$R(W,b) = \|W\|_2^2 + \|b\|_2^2$$

$$\|b\|_2^2 = b_0^2 + \cdots + b_k^2$$

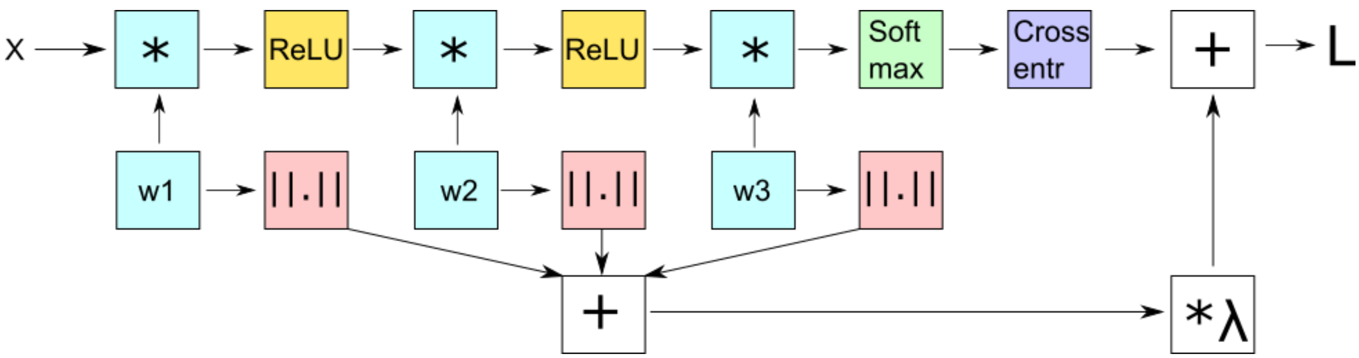


Метод обратного распространения ошибки

Граф вычислений

Внесём b в W

$$L(W) = -\sum_j \ln p(c = y_j | x_j) + \lambda R(W)$$

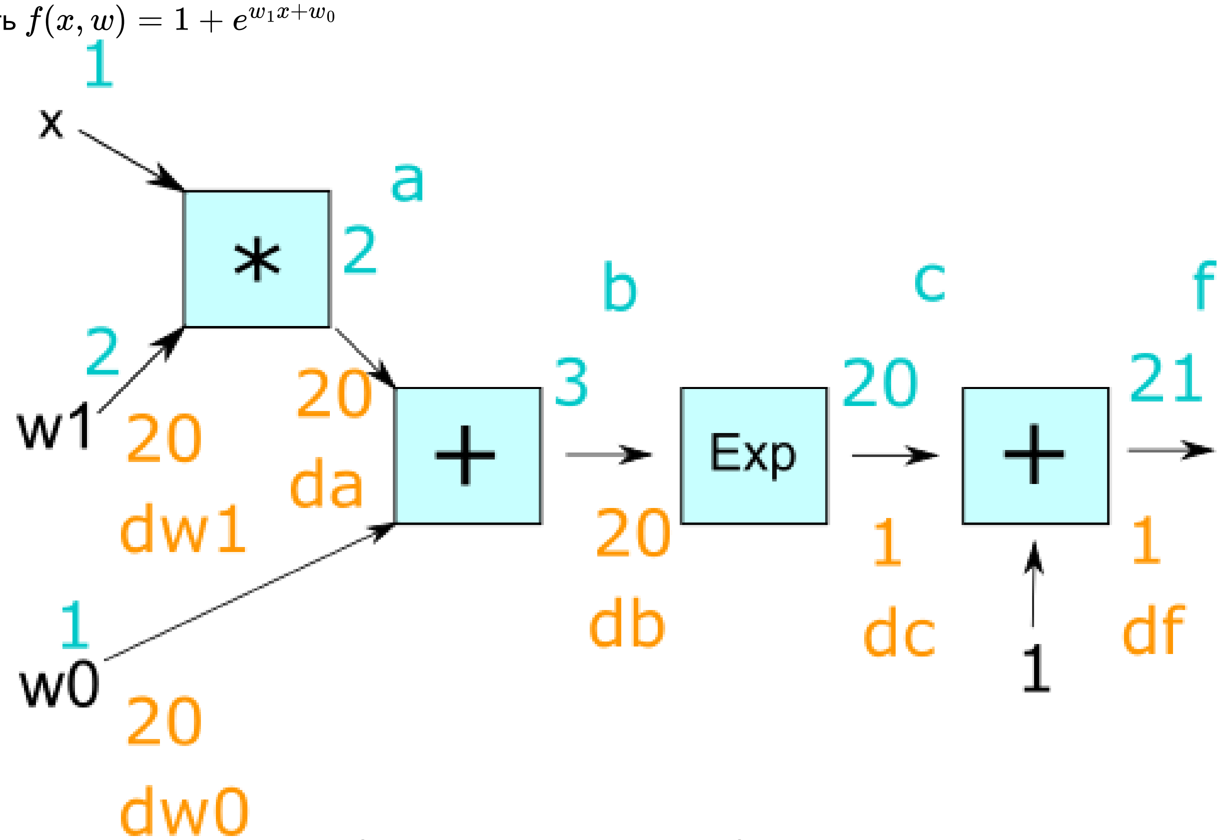


Хотим найти градиенты функции потерь L по всем входам графа вычислений.

Простой пример

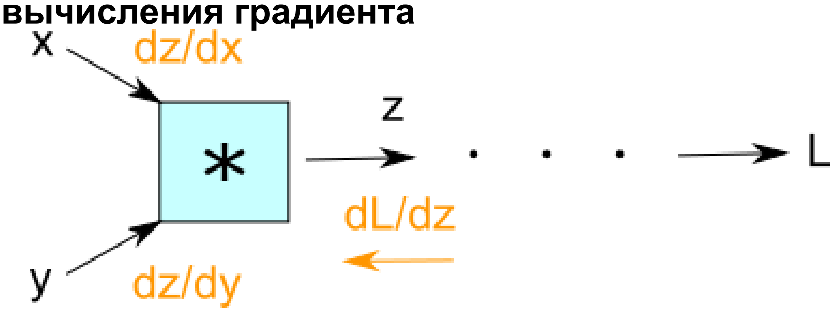
Производная сложной функции $f(g(x)) \quad \to \frac{df}{dx} = \frac{df}{dg}\frac{dg}{dx}$

Пусть $f(x, w) = 1 + e^{w_1 x + w_0}$



$$\frac{\partial f}{\partial f} = 1,\ f = c + 1,\ dc = \frac{\partial f}{\partial c} = 1, \dots$$

Общая схема вычисления градиента



Метод обратного распространения ошибки (backpropagation)

В итоге мы смогли вычислить все градиенты простыми операциями обратным проходом по графу

- не выписывали всю производную целиком аналитически
- на каждом шаге дифференцировали простую функцию
- за один проход по графу вычислений
- возможно распараллеливание

В случае двуслойной нейросети

Выходные значения сети $a^m(x_i), m = 1 \dots M$ на объекте x_i:

$$a^m(x_i) = \sigma_m\left(\sum_{h=0}^{H} w_{hm} u^h(x_i)\right)$$

$$u^h(x_i) = \sigma_h\left(\sum_{j=0}^{J} w_{jh} f_j(x_i)\right)$$

Пусть для определенности

$$\mathcal{L}_i(w) = \frac{1}{2}\sum_{m=1}^{M}(a^m(x_i) - y_i^m)^2$$

Промежуточная задача: частные производные

$$\frac{\partial \mathcal{L}_i(w)}{\partial a^m}, \quad \frac{\partial \mathcal{L}_i(w)}{\partial u^h}$$

Быстрое дифференцирование. Вспомогательные градиенты

Промежуточная задача: частные производные

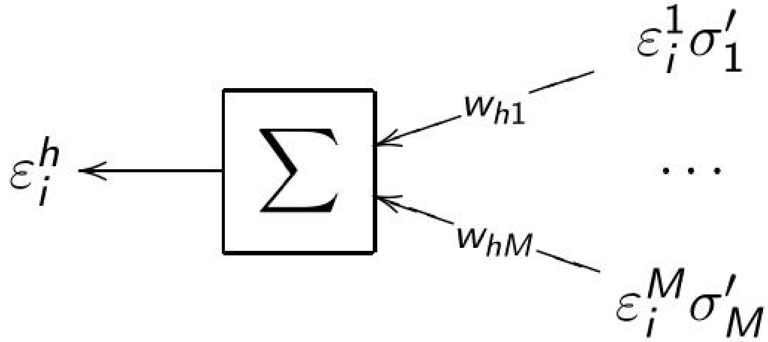
$\frac{\partial \mathcal{L}_i(w)}{\partial a^m} = a^m(x_i) - y_i^m = \varepsilon_i^m$

— это ошибка на выходном слое (для квадратичных потерь);

$\frac{\partial \mathcal{L}_i(w)}{\partial u^h} = \sum_{m=1}^{M} (a^m(x_i) - y_i^m)\sigma'_m w_{hm} = \sum_{m=1}^{M} \varepsilon_i^m \sigma'_m w_{hm} = \varepsilon_i^h$

— назовём это *ошибкой на скрытом слое*.

Получается, что ε_i^h вычисляется по ε_i^m, если запустить сеть «задом наперёд»:



Быстрое вычисление градиента

Теперь, имея частные производные $\mathcal{L}_i(w)$ по a^m и u^h, легко выписать градиент $\mathcal{L}_i(w)$ по весам w:

$\frac{\partial \mathcal{L}_i(w)}{\partial w_{hm}} = \frac{\partial \mathcal{L}_i(w)}{\partial a^m} \frac{\partial a^m}{\partial w_{hm}} = \varepsilon_i^m \sigma'_m u^h(x_i),$
$m = 1, \dots, M, h = 0, \dots, H$

$\frac{\partial \mathcal{L}_i(w)}{\partial w_{jh}} = \frac{\partial \mathcal{L}_i(w)}{\partial u^h} \frac{\partial u^h}{\partial w_{jh}} = \varepsilon_i^h \sigma'_h f_j(x_i),$
$h = 1, \dots, H, j = 0, \dots, n$

Алгоритм обратного распространения ошибки (BackPropagation)

1. инициализировать веса w_{jh}, w_{hm}
2. **повторять**
   A. выбрать объект x_i из X^ℓ (например, случайно)
   B. прямой ход

   $$u_i^h = \sigma_h\left(\sum_{j=0}^{J} w_{jh} x_i^j\right), h = 1, \ldots, H$$

   $$a_i^m = \sigma_m\left(\sum_{h=0}^{H} w_{hm} u_i^h\right), \varepsilon_i^m = a_i^m - y_i^m, m = 1, \ldots, M$$

   $$\mathcal{L}_i = \sum_{m=1}^{M} (\varepsilon_i^m)^2$$

   C. обратный ход

   $$\varepsilon_i^h = \sum_{m=1}^{M} \varepsilon_i^m \sigma'_m w_{hm}, h = 1 \ldots H$$

   D. градиентный шаг

   $$w_{hm} = w_{hm} - \eta \varepsilon_i^m \sigma'_m u_i^h, h = 0, \ldots, H, m = 1 \ldots M$$

   $$w_{jh} = w_{jh} - \eta \varepsilon_i^h \sigma'_h x_i^j, j = 0, \ldots, n, h = 1 \ldots H$$

   E. $Q = (1 - \lambda)Q + \lambda \mathcal{L}_i$
3. **пока** Q не стабилизируется

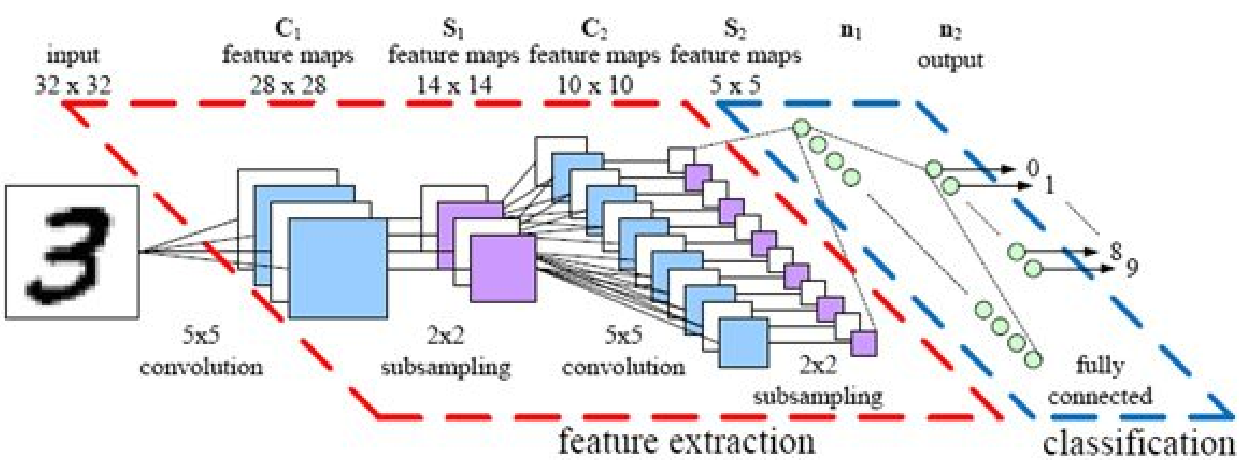
Резюме

- Нейрон = линейная классификация или регрессия
- Нейронная сеть = суперпозиция нейронов с нелинейной функцией активации
- BackPropagation = быстрое дифференцирование суперпозиций. Позволяет обучать сети практически любой конфигурации
- Методы улучшения сходимости и качества:
  - обучение по мини-подвыборкам (mini-batch)
  - различные функции активации
  - регуляризация
- Не было на этой лекции — будет во второй части курса:
  - различные алгоритмы оптимизации: adam, RMSProp
  - dropout
  - выбор начального приближения

Что ещё можно посмотреть?

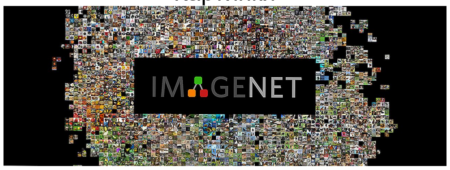
- Третья лекция курса «Deep Learning на пальцах» от Семена Козлова: https://www.youtube.com/watch?v=kWTC1NvL894 (https://www.youtube.com/watch?v=kWTC1NvL894)
- 3blue1brown: https://www.youtube.com/watch?v=Ilg3gGewQ5U (https://www.youtube.com/watch?v=Ilg3gGewQ5U)
- В курсе Стенфорда: http://cs231n.github.io/optimization-2/ (http://cs231n.github.io/optimization-2/)

Сверточная сеть — в следующем семестре



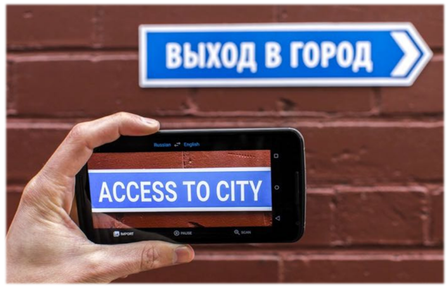
Source (https://www.pyimagesearch.com/wp-content/uploads/2014/06/cnn_architecture.jpg)

Картинки



25% → 3.5% ошибок против 5% у людей

Текст

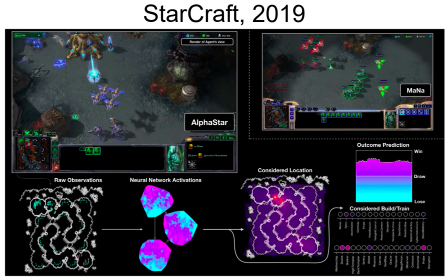


Голос

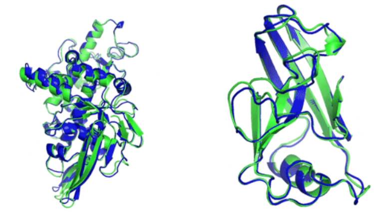
Siri Google Assistant Hey Cortana amazon alexa

Го, 2016

StarCraft, 2019



Структура белка, 2020



T1037 / 6vr4
90.7 GDT
(RNA polymerase domain)

T1049 / 6y4f
93.3 GDT
(adhesin tip)

Experimental result
Computational prediction