

Введение в глубокое обучение (deep learning)

Н.Ю. Золотых
ИТММ, ННГУ им. Н.И. Лобачевского
www.uic.unn.ru/~zny

Что такое машинное обучение (machine learning)?

Идея обучающихся машин (learning machines) принадлежит А. Тьюрингу

[*Turing A. Computing Machinery and Intelligence // Mind. 1950. V. 59. P. 433–460; перепечатано: Can the Machine Think? // World of Mathematics. Simon and Schuster, N.Y. 1956. V. 4. P. 2099–2123; рус. перев.: Тьюринг А. М. Может ли машина мыслить? // М.: Физматлит, 1960*]

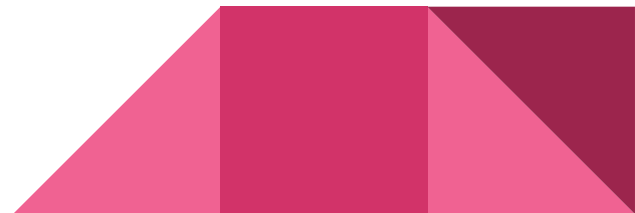


Искусственный интеллект (AI – artificial intelligence)

Искусственный интеллект – компьютер/программа, решающая интеллектуальные задачи, т.е. выполнение которых традиционно считалось прерогативой человека.

Сильный искусственный интеллект (strong AI) – компьютер/программа, способная решать все интеллектуальные задачи.

Слабый искусственный интеллект (weak AI) – компьютер/программа, способная решать конкретный класс интеллектуальных задач.



Очень краткая история ML (и AI)

- < 1950-е гг. Статистические методы
- 1950-е гг. Начало (шашки Самуэля, персептроны, логический вывод, ...)
- 1960-е гг. Байесовские методы
- 1970-е гг. “Зима” AI
- 1980-е гг. Backpropagation, сверточные сети и др. - “оттепель”
- 1990-е гг. Машина опорных векторов и др.
(смещение от дедуктивного обучения к индуктивному)
- 2000-е гг. Ансамбли деревьев решений, “ядерные” методы
- 2010-е гг. Глубокое обучение (вторая “весна” AI)



Машинное обучение сегодня

Причины “Второй весны AI”:

- Новые алгоритмы (deep learning - глубокое обучение)
- Мощные компьютеры
- Много данных (Big Data)

Достижения:

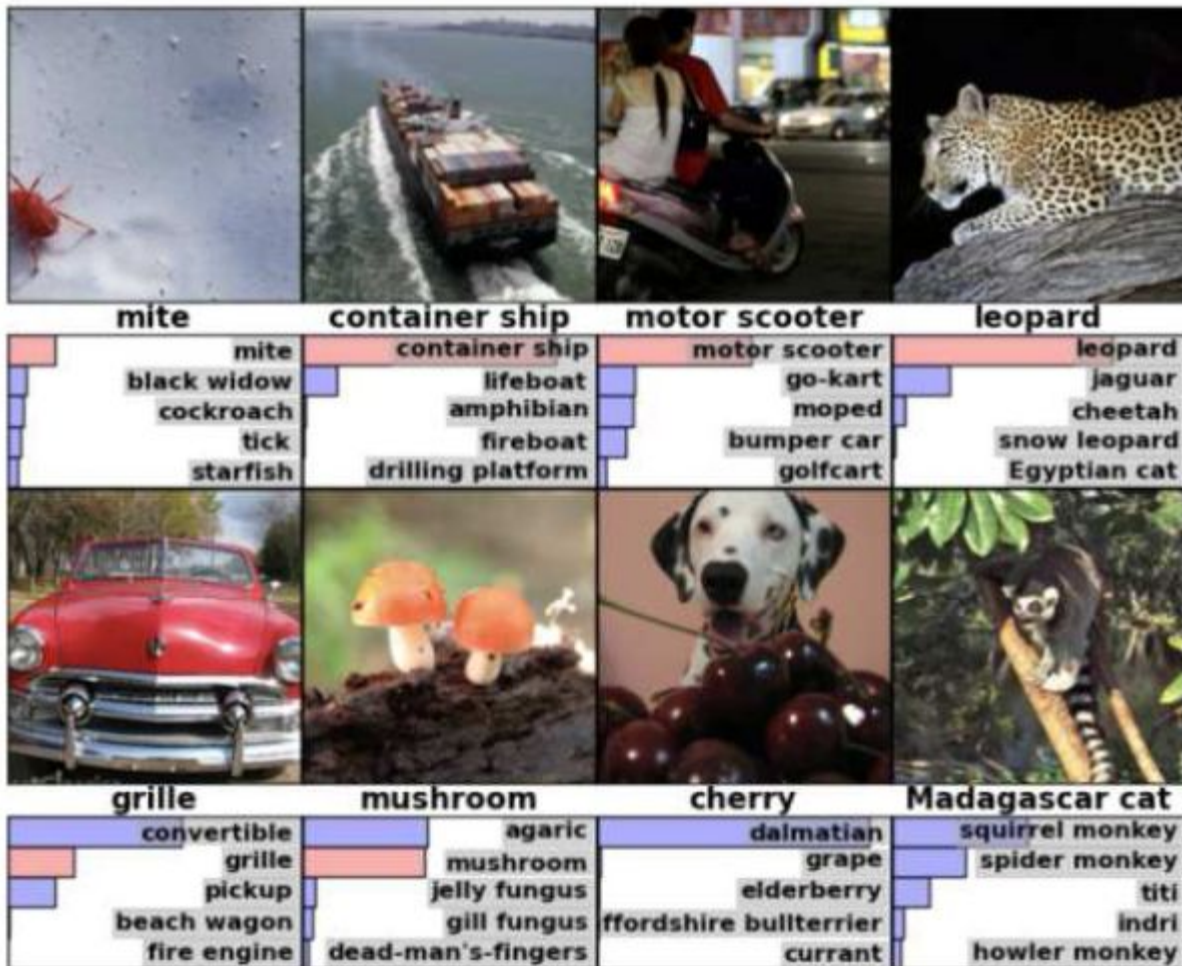
- ...
- Машинное обучение стало технологией



ImageNet

Прорыв 2012:
ImageNet ILSVRC-2012
(около 1 млн.
изображений, 1000
классов).

Ошибку удалось
понижить с 26% до 15%
(сейчас еще меньше) –
A.Krizhevsky,
I. Sutskever, G. E.Hinton



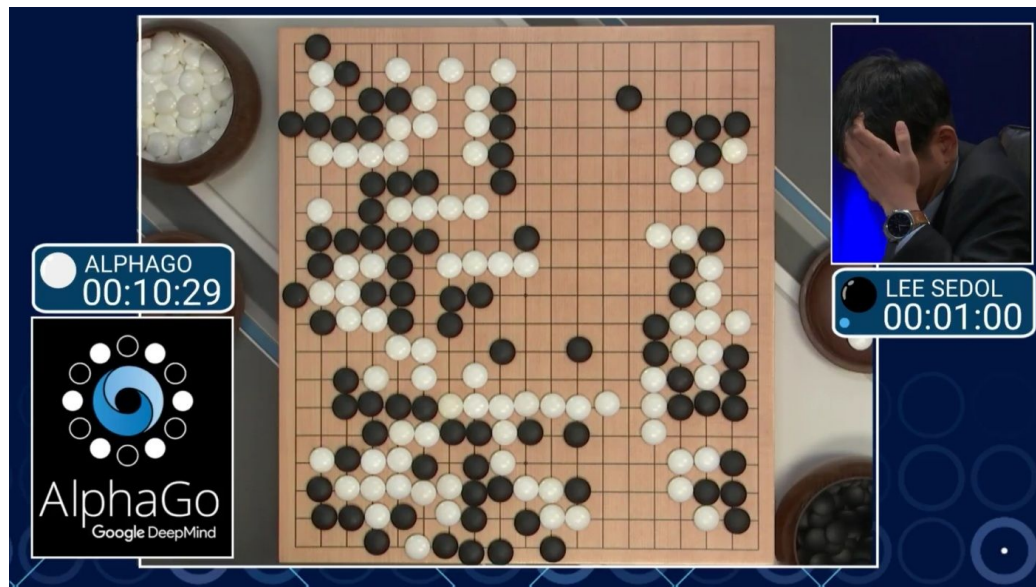
AlphaGo (Google DeepMind)

В 2015 г. – победа над чемпионом Европы Фань Хуэем

В 2016 г – победа над чемпионом мира Ли Седодем

Развитие:

- AlphaGo (использовалась база из 10000 партий + игры с собой)
- AlphaGo Zero (без априорных знаний)
- AlphaZero (Го, Сеги, шахматы, ...) 5000 TPU 280 Тфлоп каждый

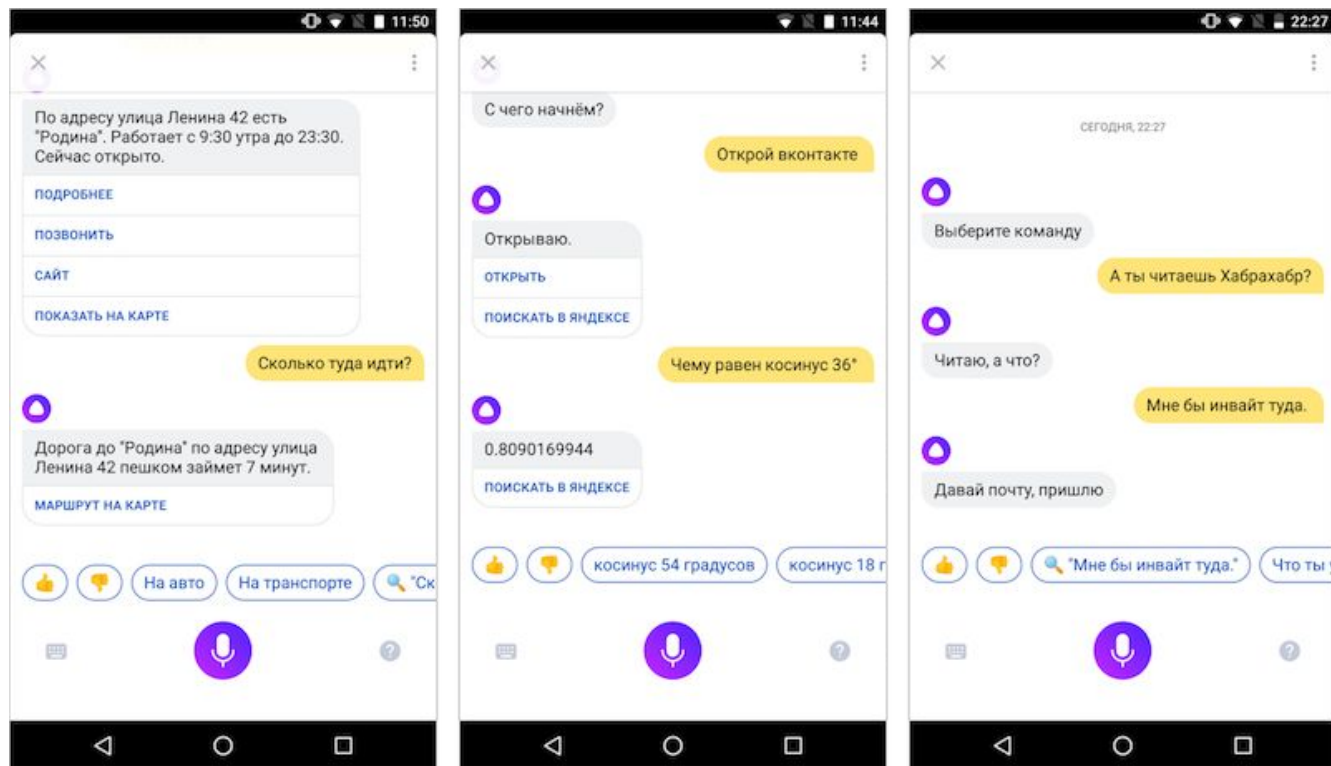


Boston Dynamics

BigDog,
CHEETAH,
LittleDog, RiSE,
PETMAN, Atlas,
Handle,
SpotMini, ...



Персональные помощники



Беспилотные автомобили



Компьютерный перевод

Google Переводчик



Текст

Документы

ИТАЛЬЯНСКИЙ (ОПРЕДЕЛЕН АВТОМАТИЧЕСКИ)

ИТАЛЬЯНСКИЙ

P



РУССКИЙ

АНГЛИЙСКИЙ

УКРАИНСКИЙ



Una volta si giocava a carte dalla guardia a cavallo Narumov. La lunga notte invernale passò insensibilmente; sedettero a cenare dopo le quattro del mattino. Chi era in vincita mangiavano con grande appetito gli altri stavano distratti davanti ai piatti vuoti. Ma champagne apparso, la conversazione si rianimò e tutti vi presero parte. (Traduzione di Silvio Polledro.)



Однажды карты разыграли конные стражи Нарумова. Долгая зимняя ночь прошла незаметно; они сели на ужин после четырех утра. Те, кто был в выигрыше, ели с большим аппетитом, остальные отвлекались перед пустыми тарелками. Но шампанское появилось, разговор ожил и все приняли участие. (Перевод Сильвио Полледро.)



Odnazhdy karty razygrali konnyye strazhi Narumova. Dolgaya zimnyaya noch' proshla nezametno; oni seli na uzhin posle chetyrekh utra. Te, kto byl v vyigryshe, yeli s bol'shim appetitom, osta'nyye otvlekalis' pered pustymi tarelkami. No shampanskoye poyavilos',

Развернуть



370/5000



Основные направления ML

- Обучение с учителем:
 - классификация
 - регрессия
 - восстановление временных рядов (например, seq2seq)
 - генеративные модели
 - ...
- Обучение без учителя
- Обучение с подкреплением
- ...

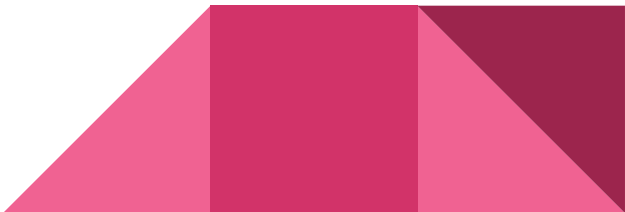


Обучение с учителем

- Множество X – объекты, наблюдения, примеры, ситуации, входы (samples) – пространство признаков
- Множество Y – ответы, отклики, «метки», выходы (responses)
- Имеется некоторая зависимость (детерминированная или вероятностная), позволяющая по $x \in X$ предсказать $y \in Y$.
- Зависимость известна только на объектах из обучающей выборки

Задача обучения с учителем: восстановить (аппроксимировать) зависимость, т. е. построить функцию (решающее правило) $f: X \rightarrow Y$, по новым объектам $x \in X$ предсказывающую $y \in Y$: $y = f(x)$

Важно: нужно уметь предсказывать y не только для объектов из обучающей выборки, но и для новых объектов!



- Медицинская диагностика
Симптомы → заболевание
- Фильтрация спама
Письмо → спам/не спам
- Рекомендательные системы
Прошлые покупки → рекомендация
- Компьютерное зрение
Изображение → что изображено
- Распознавание текста
Рукописный текст → текст в машинном коде
- NLP
Текст на русском языке → перевод на английский
- Распознавание речи
Аудиозапись речи → текст



Каким бывает x ?

Каждый объект x должен как-то кодироваться.

Самый распространенный способ:
 x представлен как вектор (набор) из d признаков

Признак может быть

- номинальным (категориальным)
- количественным (числовым):
 - вещественный
 - дискретный
- ...



Иногда x сложно (или неразумно) задать как вектор признаков (фиксированной длины).

Например, x – это временной ряд, дерево, ...



Каким бывает y ?

- номинальный – задача классификации
- количественный – задача регрессии
- временной ряд – задача предсказания временного ряда
- ...



Признаковые описания объектов обучающей выборки обычно записывают в таблицу:

$$(X \mid y) = \left(\begin{array}{cccccc|c} x_1^{(1)} & x_2^{(1)} & \dots & x_j^{(1)} & \dots & x_d^{(1)} & y^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_j^{(2)} & \dots & x_d^{(2)} & y^{(2)} \\ \dots & \dots & \dots & \dots & \dots & \dots & \vdots \\ x_1^{(i)} & x_2^{(i)} & \dots & x_j^{(i)} & \dots & x_d^{(i)} & y^{(i)} \\ \dots & \dots & \dots & \dots & \dots & \dots & \vdots \\ x_1^{(N)} & x_2^{(N)} & \dots & x_j^{(N)} & \dots & x_d^{(N)} & y^{(N)} \end{array} \right)$$

i -я строка соответствует i -му объекту в обучающей выборке
 j -й столбец – j -му признаку

Некоторые методы ML

- Метод наименьших квадратов
- Линейный и квадратичный дискриминантный анализ
- Логистическая регрессия
- Метод k ближайших соседей
- Наивный байесовский классификатор
- Машина опорных векторов (SVM)
- Деревья решений (C4.5, CART и др.)
- Ансамбли решающих функций (бустинг, баггинг и т. п.)
- Нейронные сети (включая глубокое обучение)
- ...





- “The Most Popular Python Data Science Platform”
- <https://www.anaconda.com/download/>

Anaconda - это дистрибутив Питона, включающий в себя Jupiter Notebook, некоторое количество нужных библиотек, spyder и др.



Необходимые библиотеки

- Numpy - векторы, матрица, линейная алгебра
- Scipy - другие численные методы
- Pandas - манипуляция таблицами (dataframe) и временными рядами (series)
- Scikit-Learn - методы машинного обучения



Google Colaboratory

Возможность все это запустить удаленно
<https://colab.research.google.com>



Пример 1


Имеются данные о 114 лицах с заболеванием щитовидной железы.

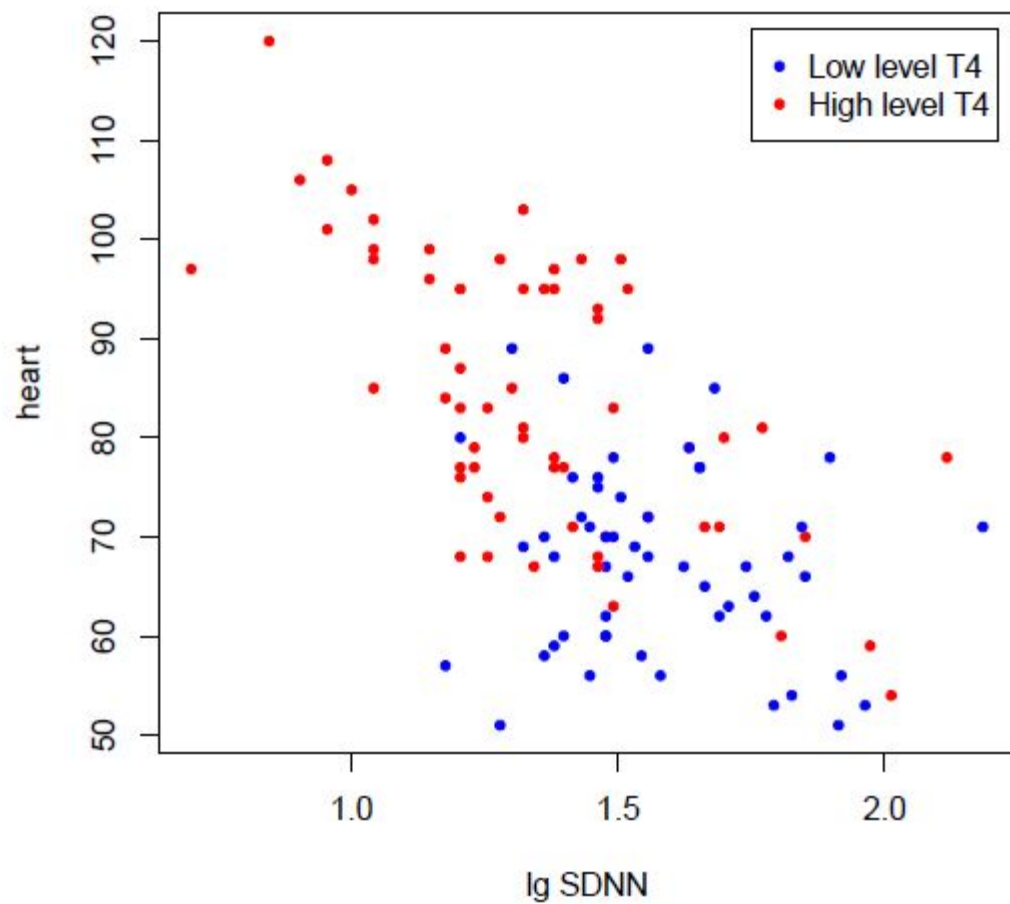
- У 61 — повышенный уровень свободного гормона T4,
- у 53 — уровень гормона в норме.

Для каждого пациента известны следующие показатели:

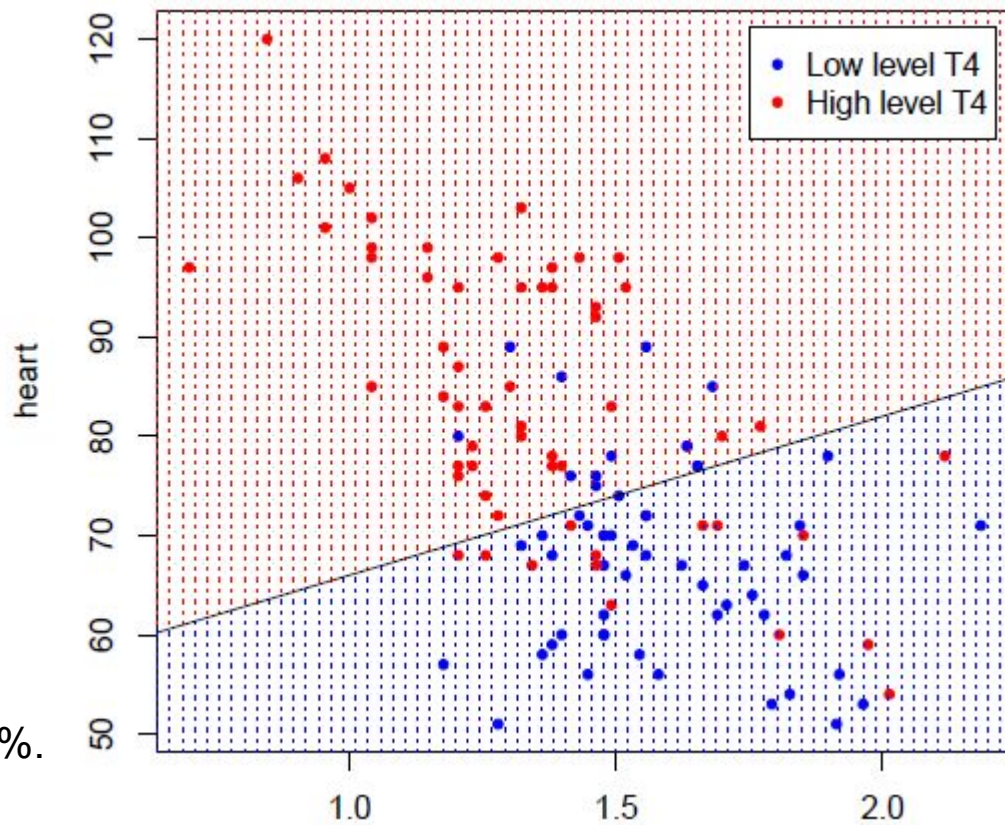
- $x_1 = \text{heart}$ — частота сердечных сокращений (пульс),
- $x_2 = \lg \text{SDNN}$ — логарифм стандартного отклонение длительности интервалов между синусовыми сокращениями сердца.

Можно ли научиться предсказывать уровень свободного T4 по heart и SDNN?

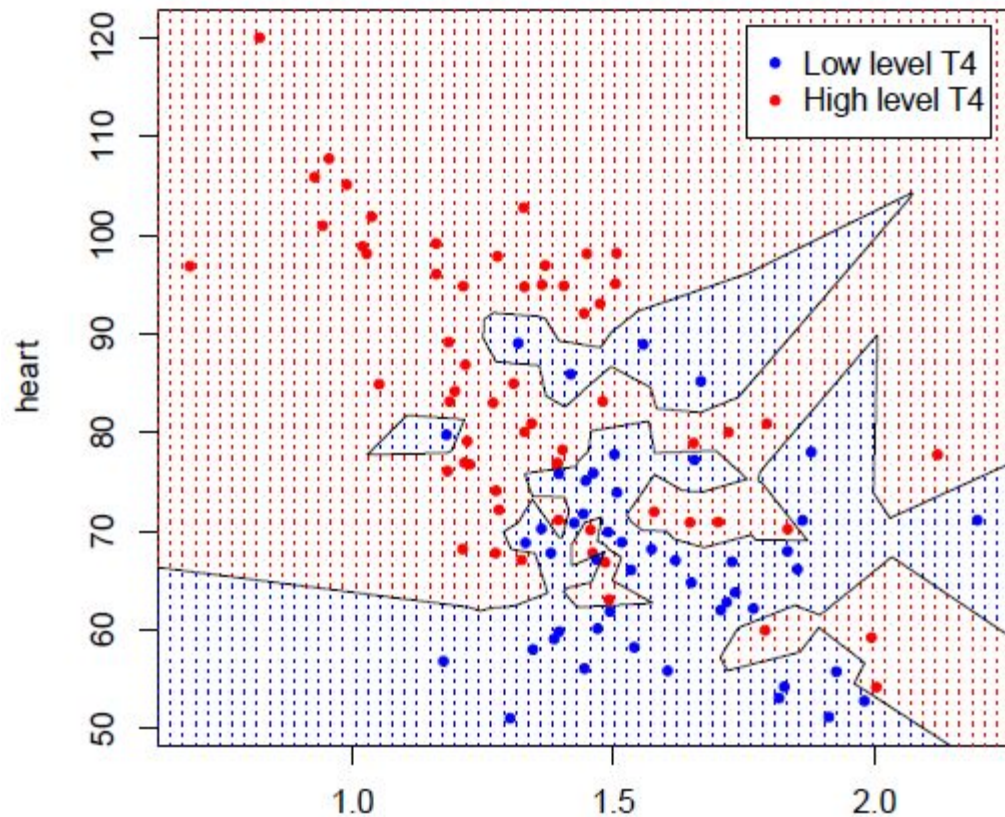




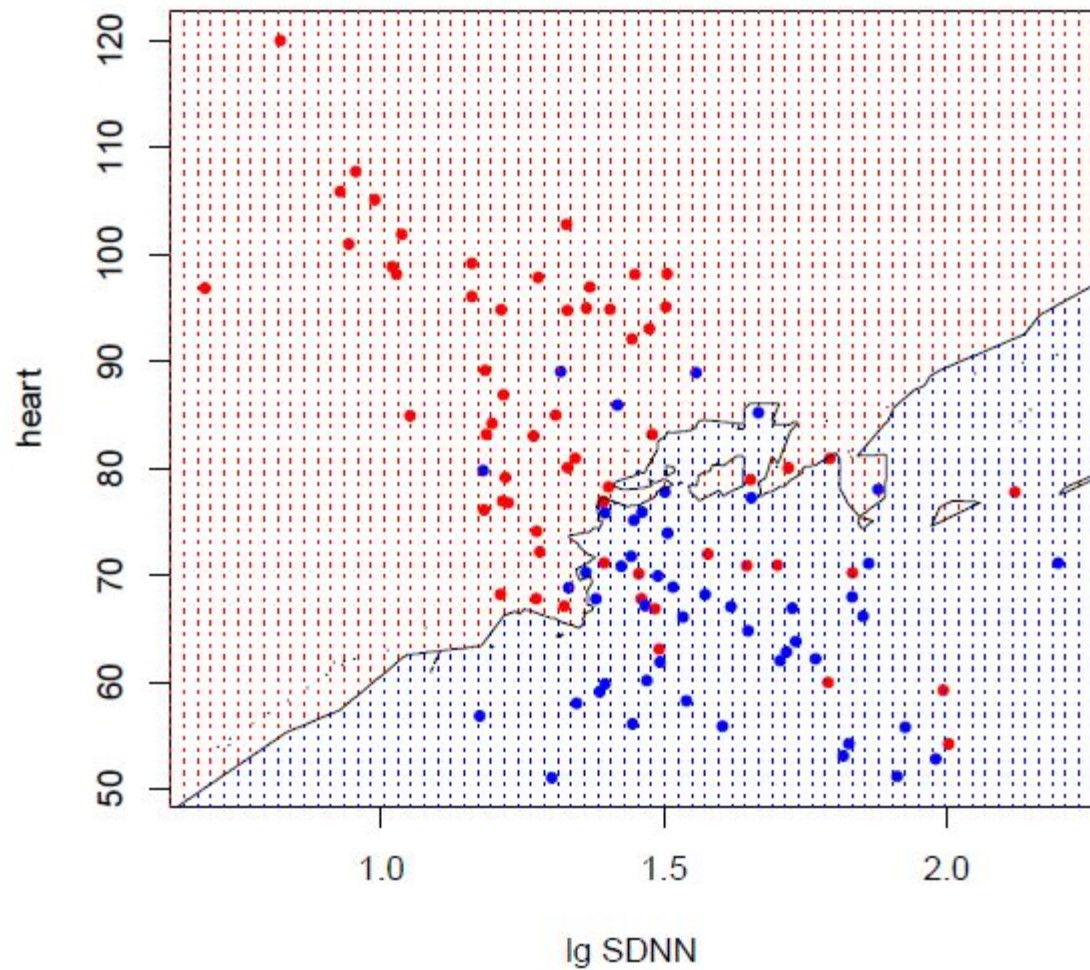
$16 \cdot \lg \text{SDNN} - \text{heart} + 50 = 0$
Ошибка на обучающей выборке 23%.
Можно ли ее сделать меньше?



Метод ближайшего соседа
(с масштабированием)
Ошибка на обучающей выборке 0%.



Метод 15 ближайших соседей



Переобучение и недообучение

Малая ошибка на обучающей выборке не означает, что мы хорошо классифицируем новые объекты.

Переобучение — решающее правило хорошо решает задачу на обучающей выборке, но плохо предсказывает ответ на новых данных.

Недообучение — решающее правило показывает плохие результаты и на обучающей выборке и на новых данных.



Обучающая и тестовая выборки


- Обучаем модель на тестовой выборке
- Тестируем — на тестовой



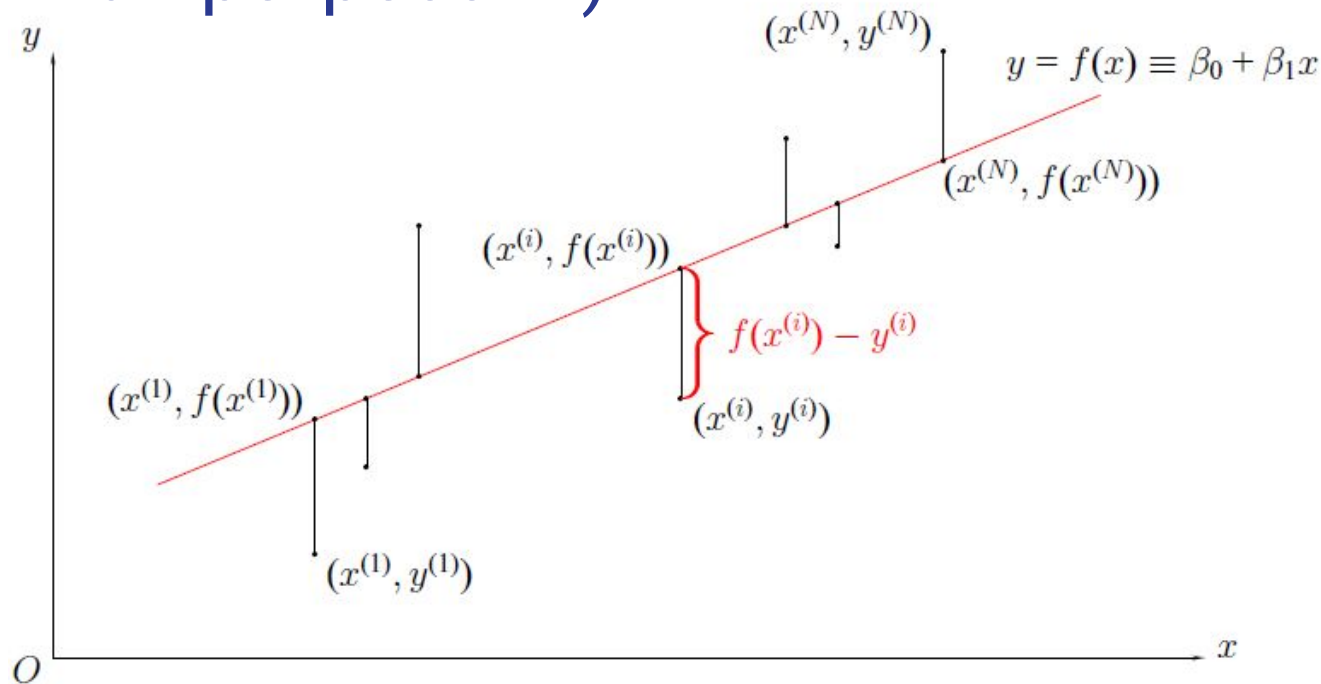
Пример 2

Имеются данные о стоимости 72379 квартир

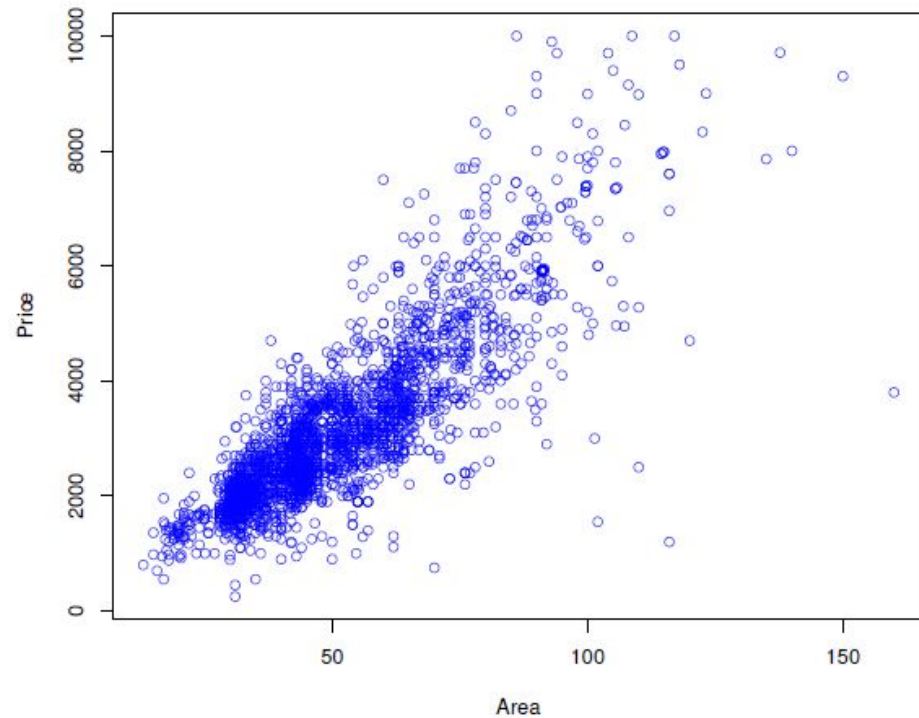
Требуется научиться предсказывать цену — задача восстановления *регрессии*

1. Date — дата
 2. Lat — широта (числовой)
 3. Lng — долгота (числовой)
 4. Housing — тип недвижимости (новостройка, вторичка)
 5. Floors — к-во этажей в доме (числовой)
 6. House — тип строения (кирпичный, панельный, блочный, монолитный, деревянный)
 7. Rooms — количество комнат (студия, 1, 2, . . .)
 8. Floor — № этажа
 9. Area — площадь (числовой)
 10. Price — цена (числовой)
- 

Метод наименьших квадратов (линейная регрессия)

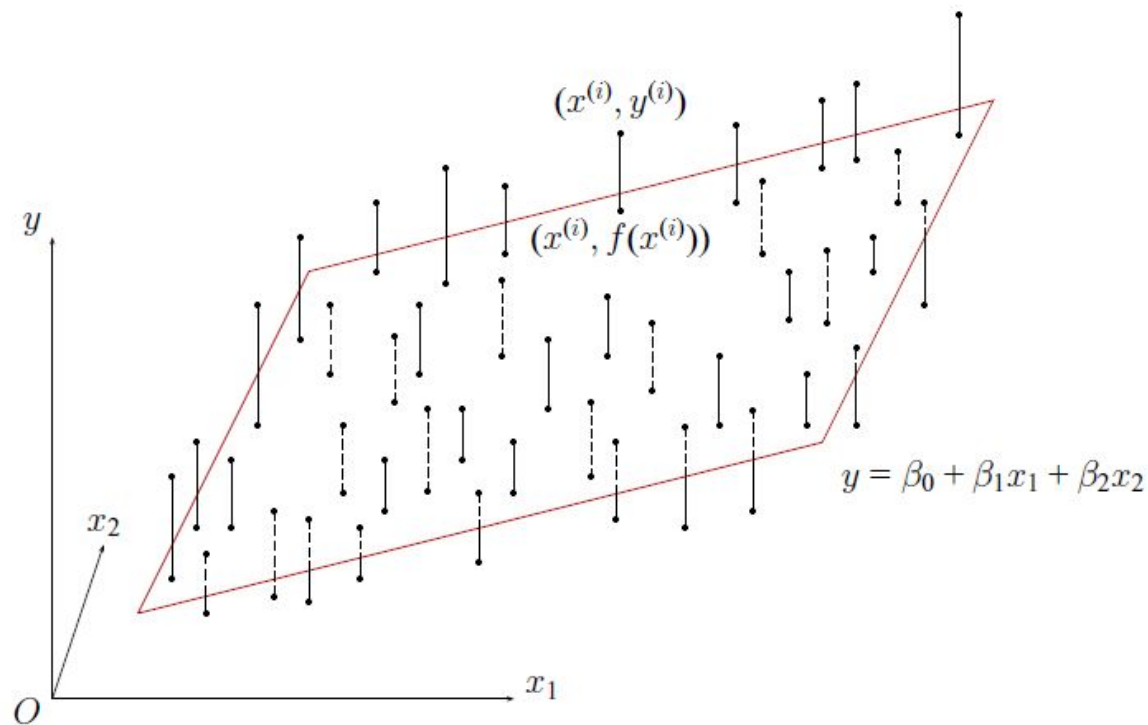


$$\text{RSS}(\beta_0, \beta_1) = \sum_{i=1}^N \left(f(x^{(i)}) - y^{(i)} \right)^2 = \sum_{i=1}^N \left(\beta_0 + \beta_1 x^{(i)} - y^{(i)} \right)^2 \rightarrow \min$$

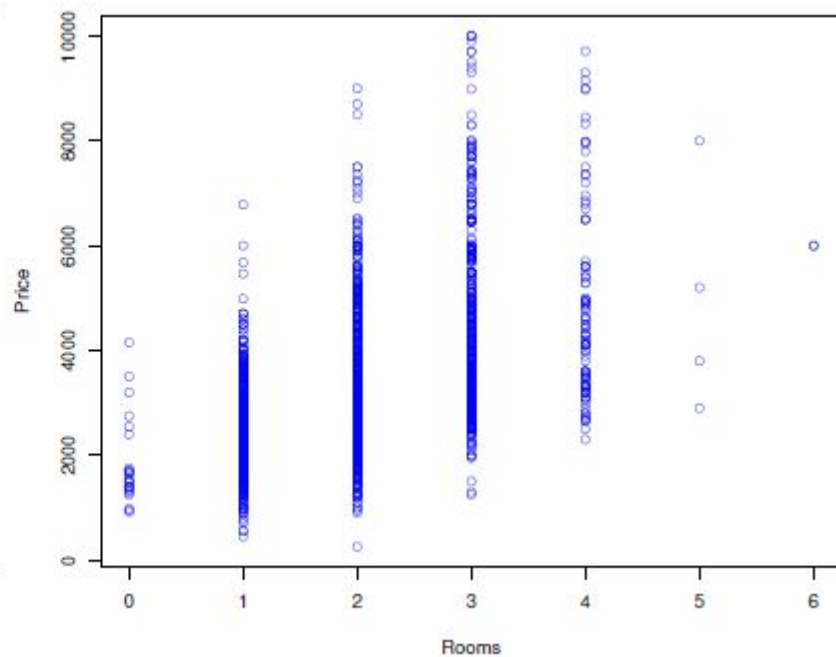
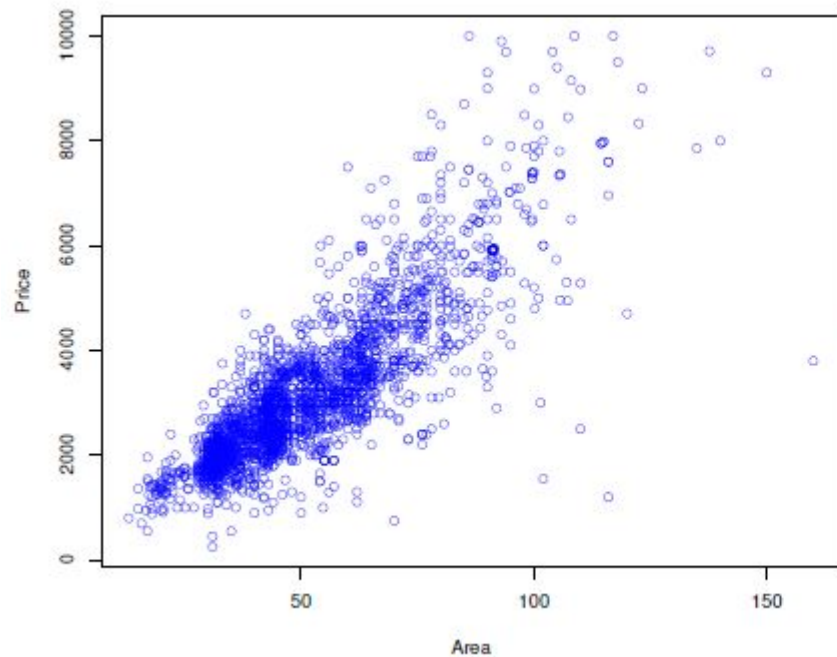


$$y = \beta_0 + \beta_1 x, \quad y \equiv \text{Price}, \quad x \equiv \text{Area}$$

$$\beta_0 = -119.53, \quad \beta_1 = 64.89$$



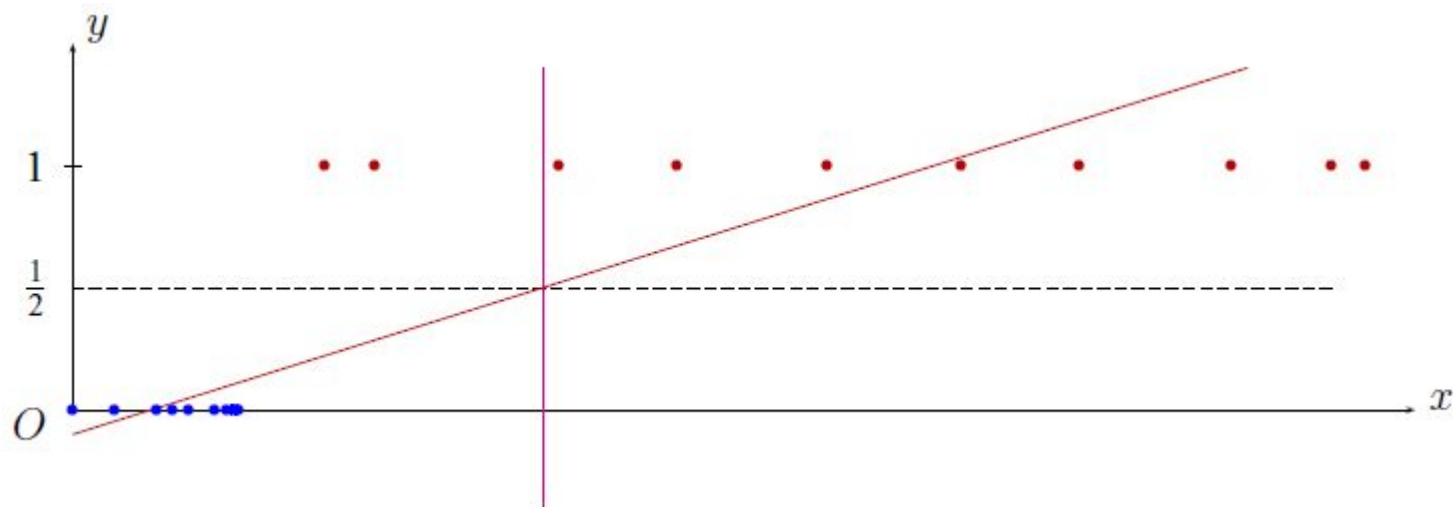
$$\sum_{i=1}^N \left(\beta_0 + \sum_{j=1}^d \beta_j x_j^{(i)} - y^{(i)} \right)^2 \rightarrow \min$$



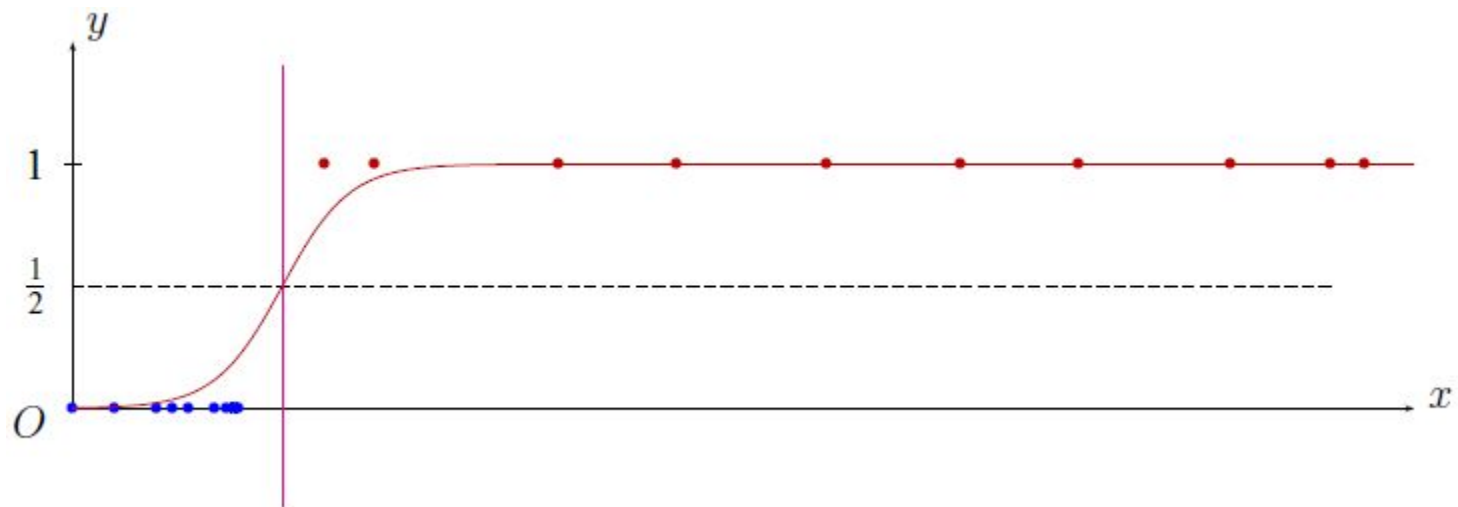
$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2, \quad y \equiv \text{Price}, \quad x_1 \equiv \text{Area}, \quad x_2 \equiv \text{Rooms}$$

$$\beta_0 = -79.32, \quad \beta_1 = 84.15, \quad \beta_2 = -542.04$$

Метод наименьших квадратов для задачи классификации (?)



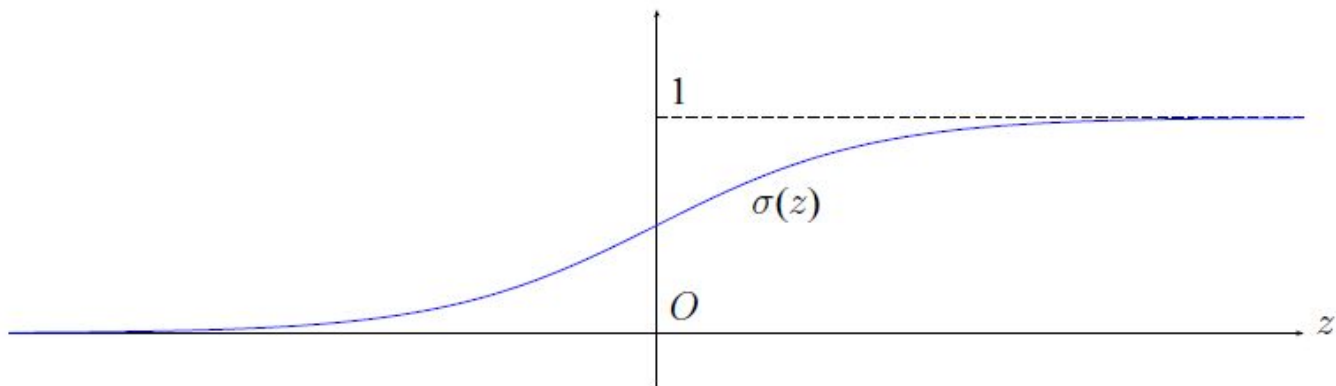
Хочется чего-то такого...



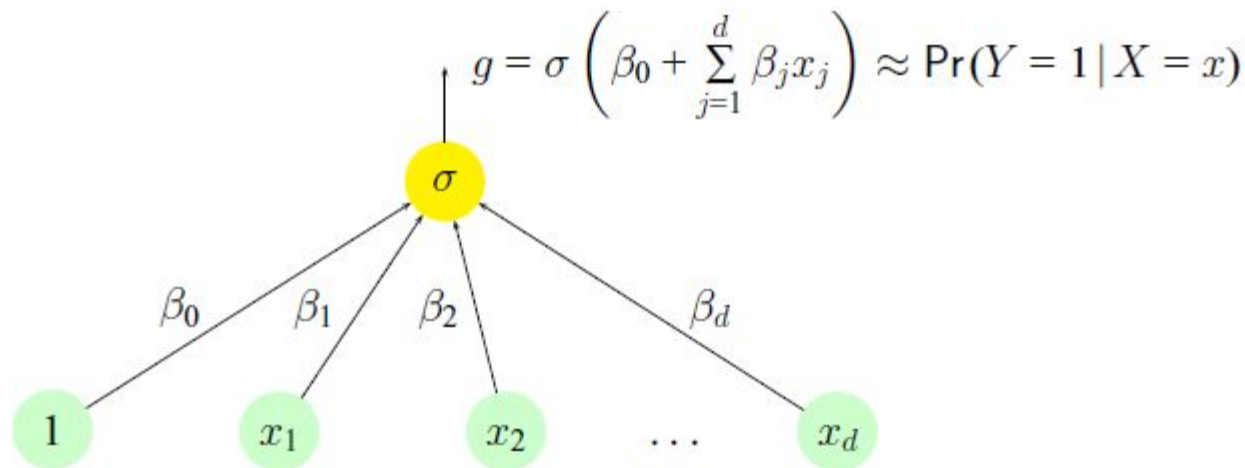
Логистическая регрессия

$$\Pr(Y = 1 | X = x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \dots + \beta_d x_d)}} = \sigma(\beta_0 + \beta^\top x),$$

где $\sigma(z) = \frac{1}{1 + e^{-z}}$ — логистическая функция (элементарный сигмоид)



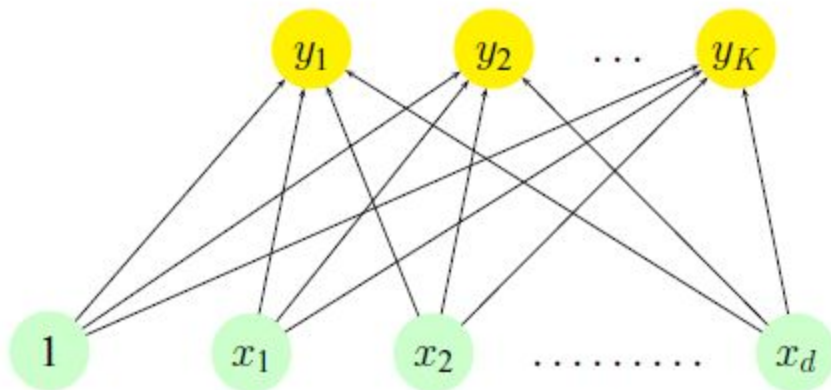
$$\Pr(Y = 1 | X = x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \dots + \beta_d x_d)}} = \sigma(\beta_0 + \beta^\top x), \quad \sigma(z) = \frac{1}{1 + e^{-z}}$$



Разделяющая поверхность - линейная (гиперплоскость):

$$\Pr(Y = 0 | x) = \Pr(Y = 1 | x) = \frac{1}{2} \quad \Leftrightarrow \quad \beta_0 + \beta_1 x_1 + \dots + \beta_d x_d = 0$$

Случай K классов:



$$y_k = \frac{\exp\left(\beta_{k0} + \sum_{j=1}^d \beta_{kj}x_j\right)}{\sum_{\ell=1}^K \exp\left(\beta_{\ell 0} + \sum_{j=1}^d \beta_{\ell j}x_j\right)} \approx \Pr(k|x)$$

$(k = 1, 2, \dots, K)$

(функция softmax)

Как обучать модель?

Минимизируем *кросс-энтропию* (*logloss-функцию*), вычисленную на обучающей выборке

2 класса:
$$L(g(x, \beta), y) = -y \ln g(x, \beta) - (1 - y) \ln(1 - g(x, \beta))$$

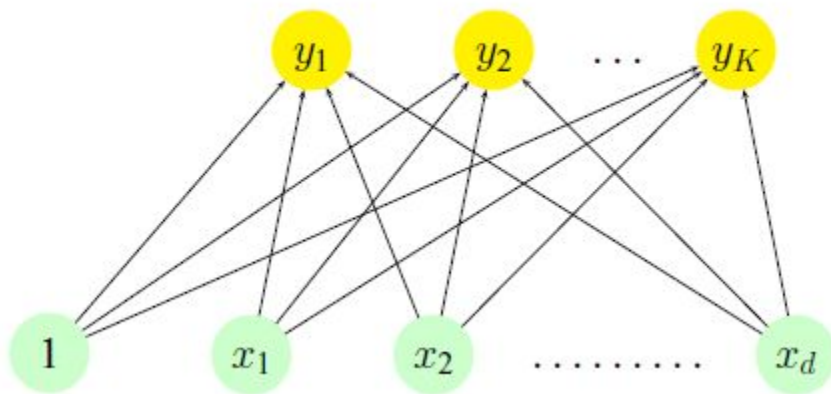
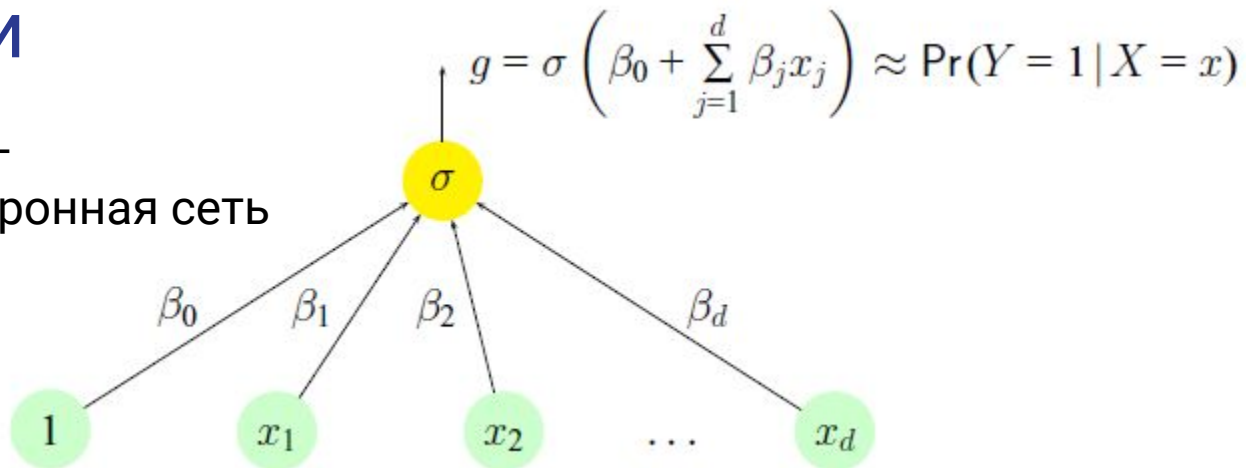
$$\hat{R}(\beta) = -\frac{1}{N} \sum_{i=1}^N \left(y^{(i)} \ln g(x^{(i)}, \beta) + (1 - y^{(i)}) \ln(1 - g(x^{(i)}, \beta)) \right)$$

К классов:
$$L(g(x, \beta), y) = \sum_{k=1}^K I(y = k) \ln g(x, \beta)$$

$$\hat{R}(\beta) = -\frac{1}{N} \sum_{i=1}^N \sum_{k=1}^K I(y^{(i)} = k) \ln g(x^{(i)}, \beta)$$

Нейронные сети

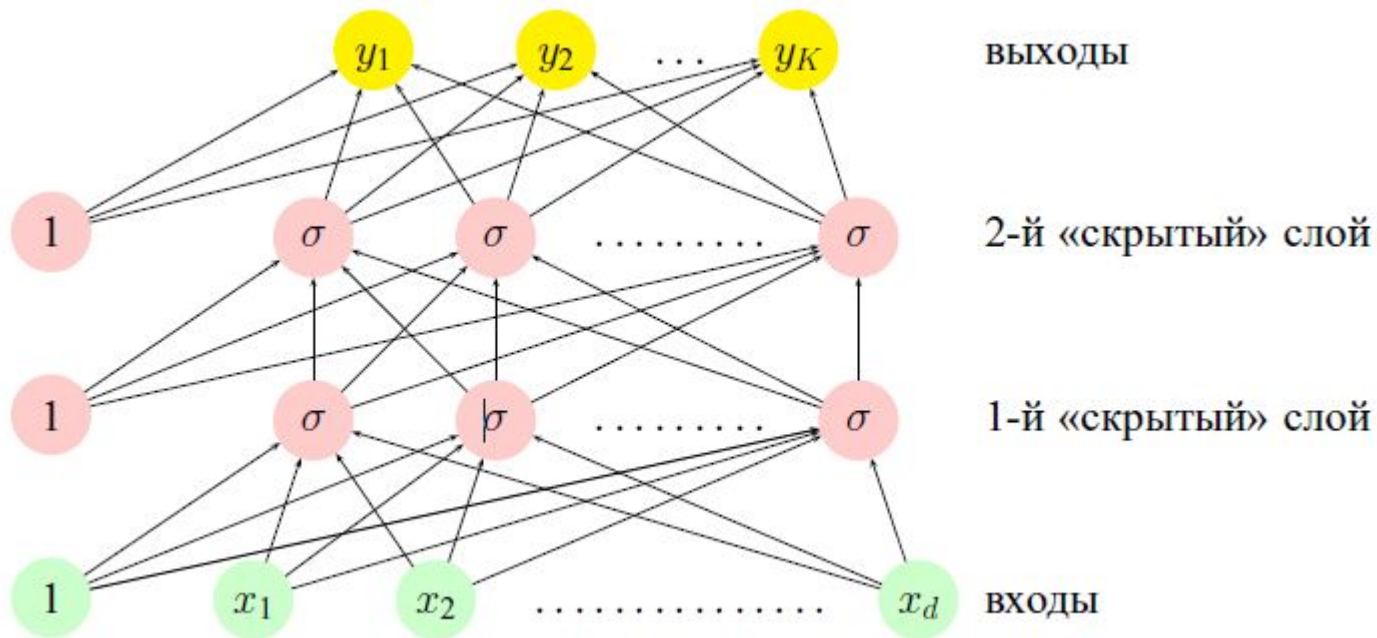
Логистическая регрессия -
это уже однослойная* нейронная сеть



$$y_k = \frac{\exp \left(\beta_{k0} + \sum_{j=1}^d \beta_{kj} x_j \right)}{\sum_{\ell=1}^K \exp \left(\beta_{\ell 0} + \sum_{j=1}^d \beta_{\ell j} x_j \right)} \approx \Pr(k | x)$$

$(k = 1, 2, \dots, K)$

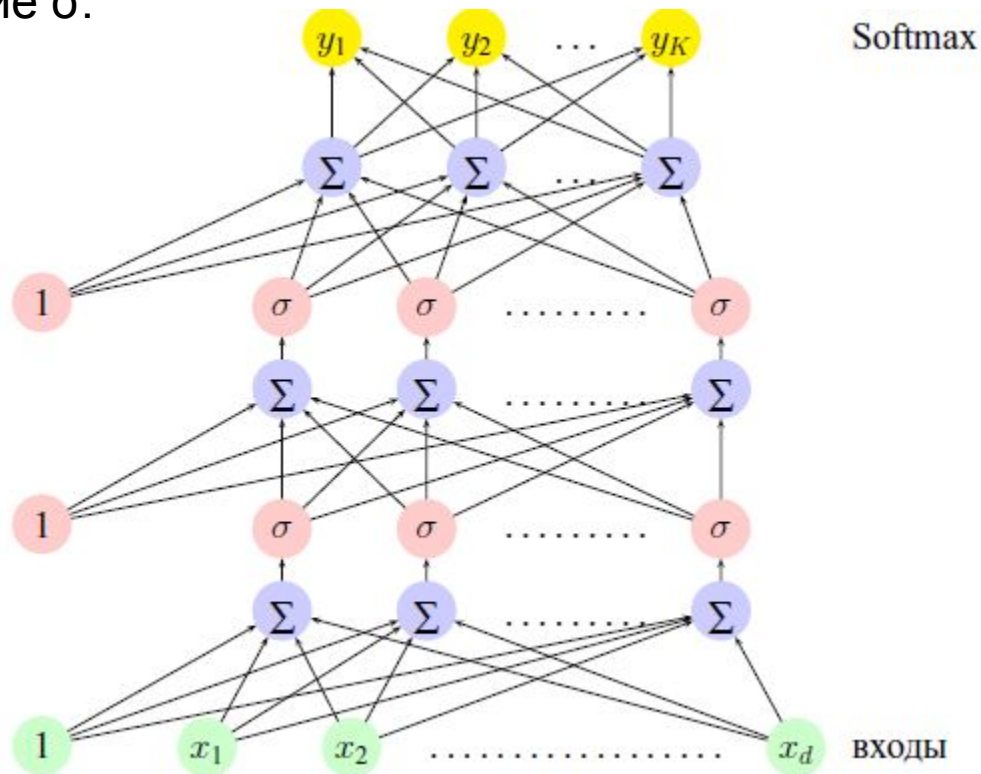
Из логистических функций можно составить суперпозицию (функция от функций от функций от ...)



Таким образом, выходы из каждого узла (нейрона) умножаются на соответствующие веса и складываются.

Далее к полученному результату z применяется функция $\sigma(z)$.

Иногда отдельно изображают суммирующие элементы и элементы, вычисляющие σ :



$$z_1 = \sigma(B_1 x), \quad z_2 = \sigma(B_2 z_1), \quad t = B_3 z_2, \quad g = \text{softmax}(t)$$

Кроме сигмоидальной используют и др. функции.

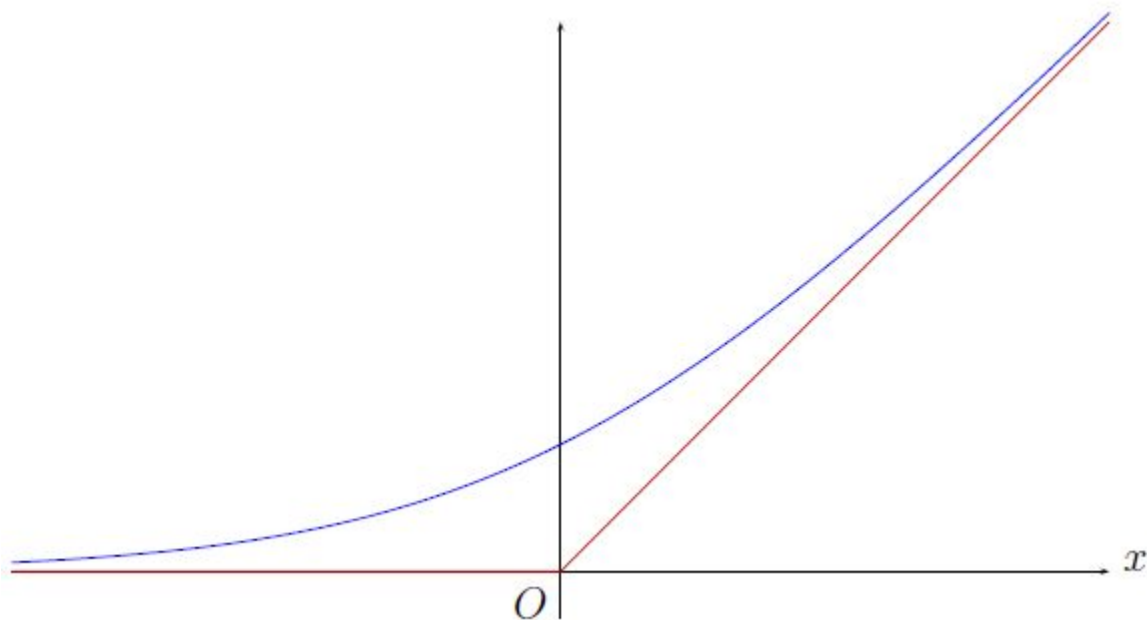
Сейчас наиболее популярна

положительная срезка линейной функции (linear rectifier):

$$g(x_1, x_2, \dots, x_q) = (\beta_0 + \beta_1 x_1 + \dots + \beta_q x_q)_+ \quad (x)_+ = \max \{0, x\}$$

или ее сглаженный вариант *softplus*:

$$g = \ln(1 + \exp(\beta_0 + \beta_1 x_1 + \dots + \beta_q x_q))$$



Обучение нейронной сети

- Штраф - сумма квадратов для задачи восстановления регрессии:

$$R(w) = \frac{1}{N} \sum_{i=1}^N \underbrace{\frac{1}{2} \left(y^{(i)} - f(x^{(i)}) \right)^2}_{R^{(i)}} \rightarrow \min$$

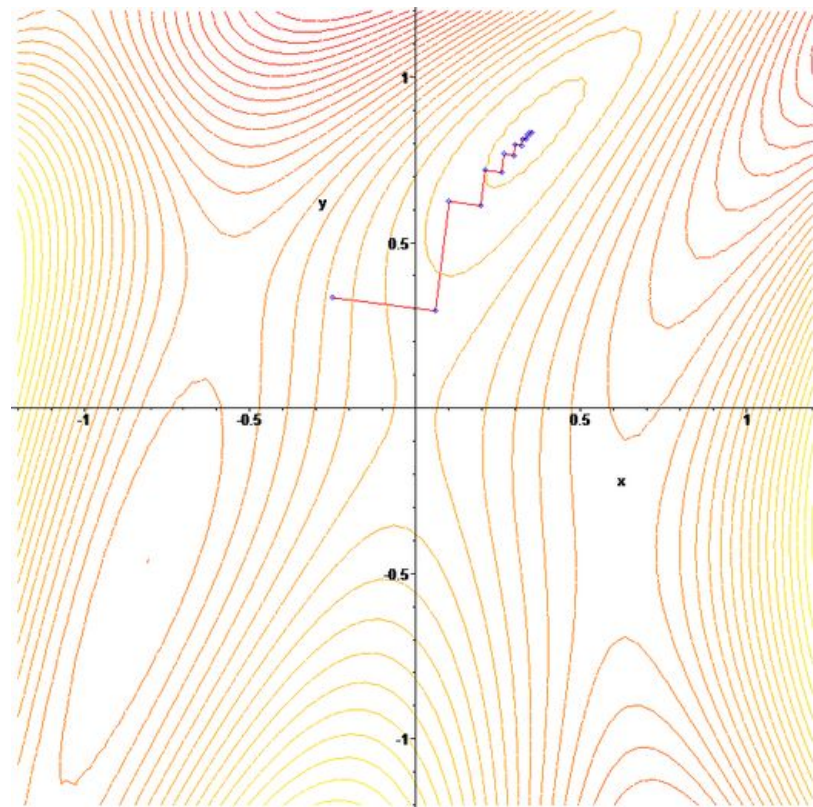
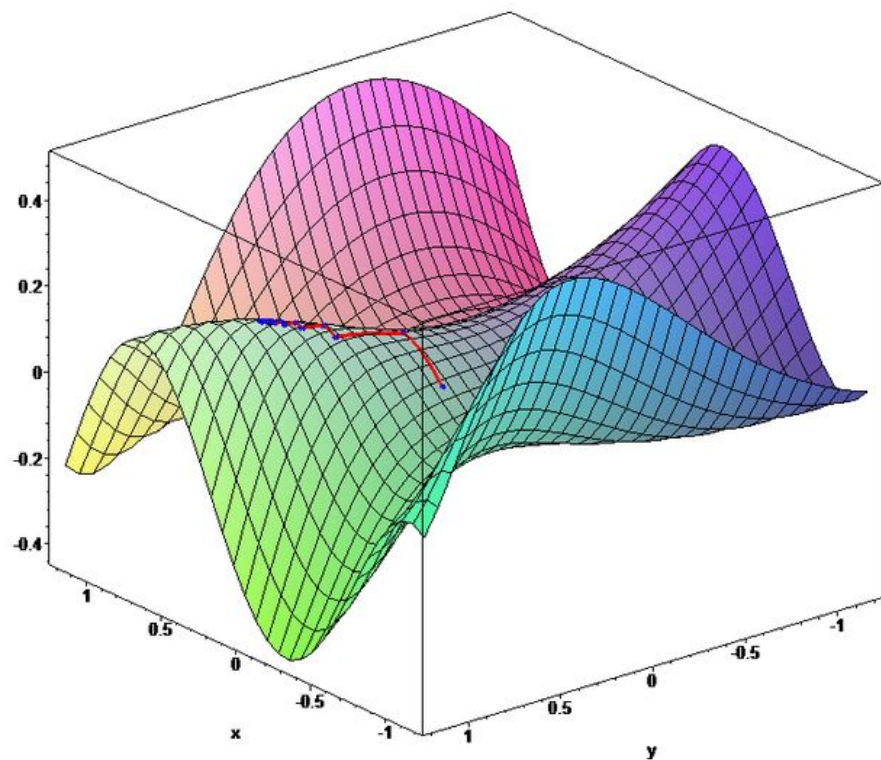
- Штраф - кросс-энтропия для задачи классификации:

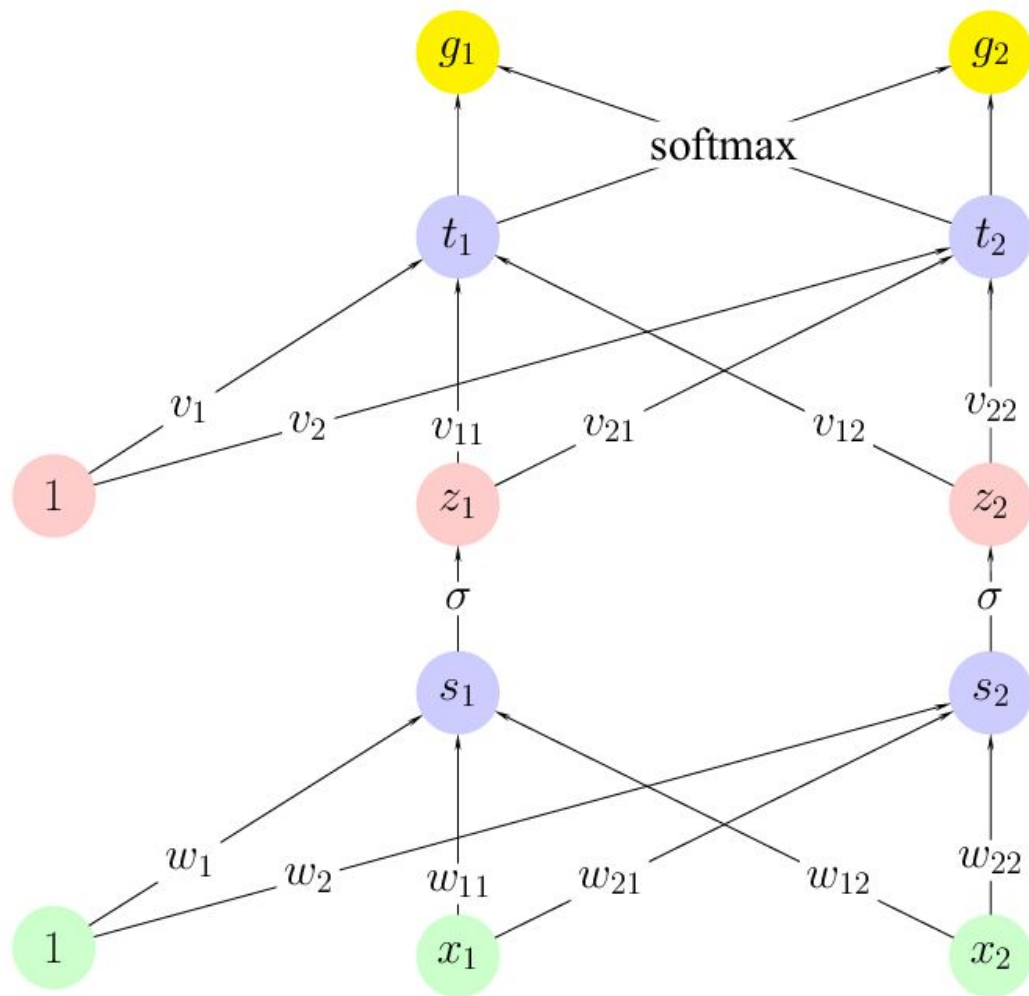
$$R(w) = -\frac{1}{N} \sum_{i=1}^N \underbrace{\sum_{k=1}^K I(y^{(i)} = k) \ln g_k(x^{(i)})}_{R^{(i)}} \rightarrow \min$$

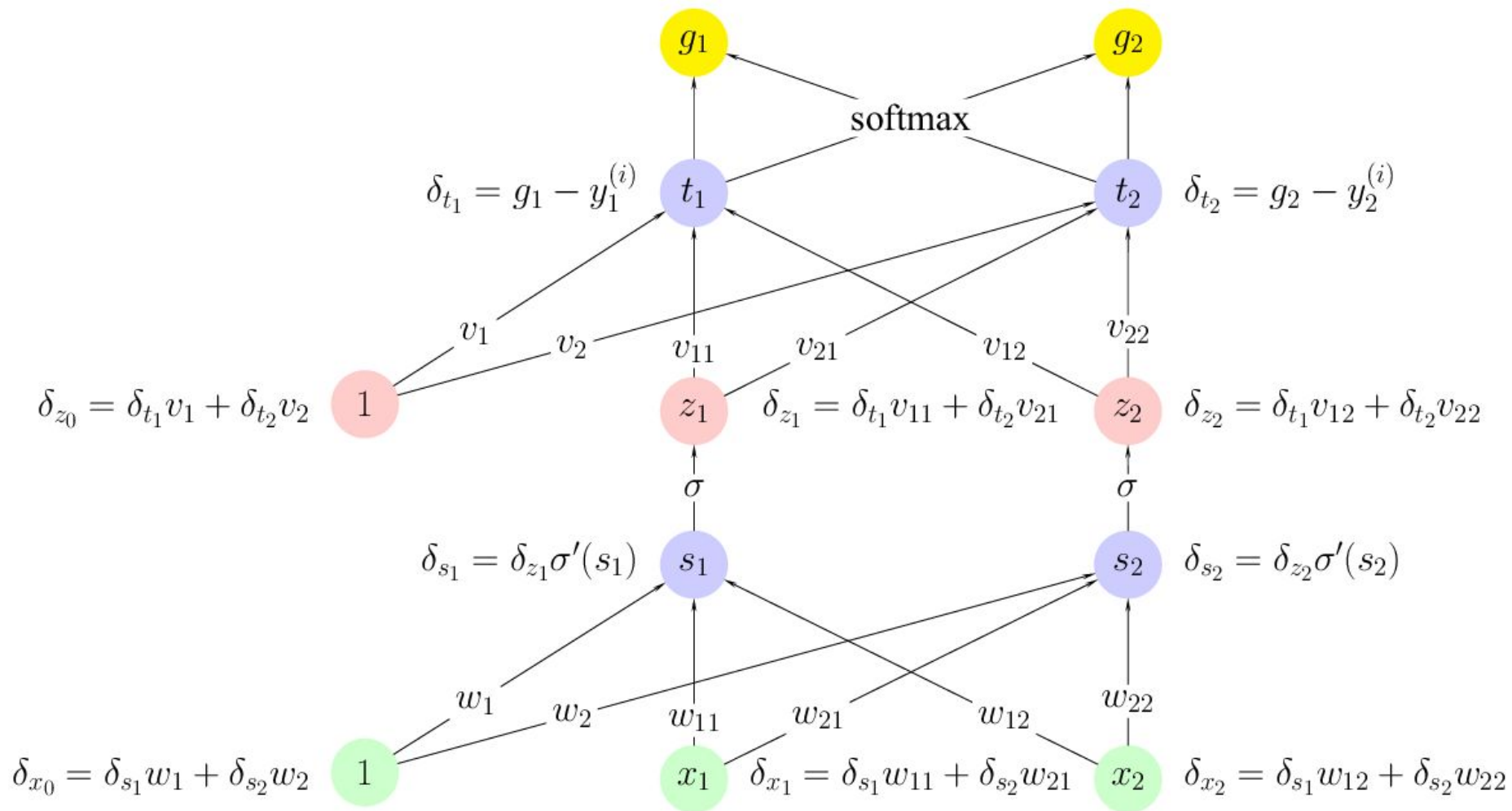
Для решения задачи минимизации используем алгоритм стохастического градиентного спуска

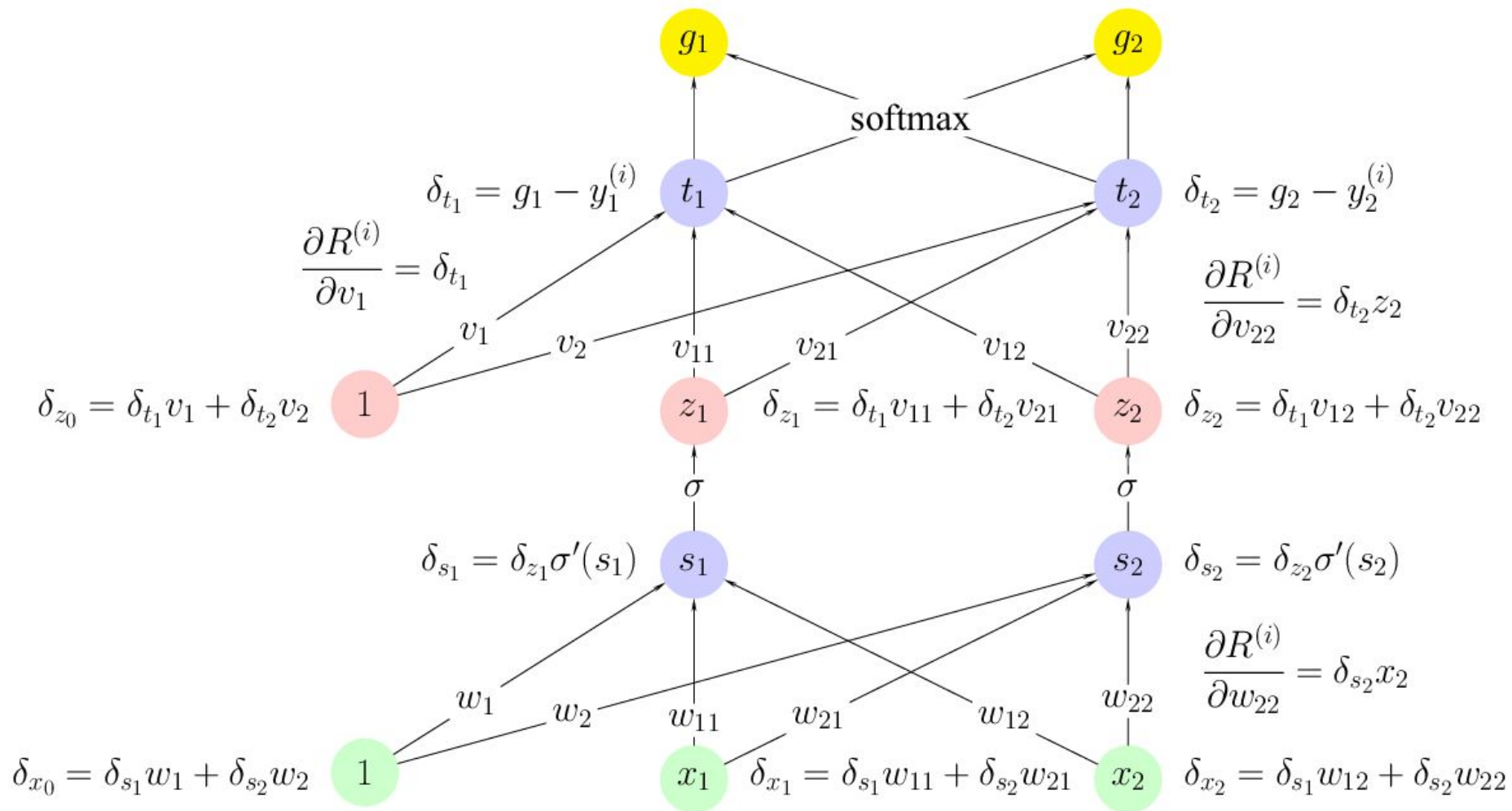
BackPropagation - это алгоритм вычисления компонент градиента $\partial R^{(i)} / \partial w$

Стохастический градиентный спуск









$$g = \text{softmax}(V\sigma(Wx + w) + v)$$

$$R^{(i)} = \text{logloss}(g) = \text{logloss}\left(\text{softmax}\left(\underbrace{V\sigma(\underbrace{Wx + w}_s)}_z + v\right)\right)$$

$$\delta_x = \frac{\partial R^{(i)}}{\partial x} = \underbrace{\underbrace{(g - y)}_{\delta_t} \cdot V \cdot \text{diag}(\sigma'(s))}_{\delta_z} \cdot W$$

$$\frac{\partial R^{(i)}}{\partial W} = \delta_s \cdot x, \quad \frac{\partial R^{(i)}}{\partial w} = \delta_s, \quad \frac{\partial R^{(i)}}{\partial V} = \delta_t \cdot z, \quad \frac{\partial R^{(i)}}{\partial v} = \delta_t$$

$$W \leftarrow W - \rho \frac{\partial R^{(i)}}{\partial W}, \quad w \leftarrow w - \rho \frac{\partial R^{(i)}}{\partial w}, \quad V \leftarrow V - \rho \frac{\partial R^{(i)}}{\partial V}, \quad v \leftarrow v - \rho \frac{\partial R^{(i)}}{\partial v}$$

$$g = \text{softmax}(V\sigma(Wx + w) + v)$$

$$R^{(i)} = \text{logloss}(g) = \text{logloss}\left(\text{softmax}\left(\underbrace{V\sigma(\underbrace{Wx + w}_s)}_z + v\right)\right)$$

$$\delta_x = \frac{\partial R^{(i)}}{\partial x} = \underbrace{(g - y)}_{\delta_t} \cdot \underbrace{V \cdot \text{diag}(\sigma'(s))}_{\delta_s} \cdot W$$

$$\frac{\partial R^{(i)}}{\partial W} = \delta_s \cdot x, \quad \frac{\partial R^{(i)}}{\partial w} = \delta_s, \quad \frac{\partial R^{(i)}}{\partial V} = \delta_t \cdot z, \quad \frac{\partial R^{(i)}}{\partial v} = \delta_t$$

$$W \leftarrow W - \rho \frac{\partial R^{(i)}}{\partial W}, \quad w \leftarrow w - \rho \frac{\partial R^{(i)}}{\partial w},$$

$$V \leftarrow V - \rho \frac{\partial R^{(i)}}{\partial V}, \quad v \leftarrow v - \rho \frac{\partial R^{(i)}}{\partial v}$$

```
def NeuralNetFit(X, Y, rho = 1, nepoch = 1000, m1 = 10):
    N, d = X.shape
    N, K = Y.shape

    W = np.random.randn(m1, d)
    V = np.random.randn(K, m1)
    w = np.random.randn(m1)
    v = np.random.randn(K)

    for epoch in range(nepoch):
        for i in range(N):
            x = X[i, :]
            y = Y[i, :]

            # Neural Net Forward
            s = W.dot(x) + w
            z = sigmoid(s)
            t = V.dot(z) + v
            g = softmax(t)

            # Neural Net Backward
            delta_t = g - y
            delta_z = delta_t.dot(V)
            delta_s = delta_z * sigmoiddiff(z)
            #delta_x = delta_s.dot(W)

            W = W - rho*np.outer(delta_s, x)
            V = V - rho*np.outer(delta_t, z)
            w = w - rho*delta_s
            v = v - rho*delta_t

    return W, w, V, v
```

Глубокое обучение

(Yann LeCun, Yoshua Bengio, Geoffrey Hinton и др.)

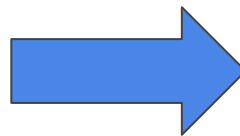
Глубокое обучение (Deep learning) — подход, основанный на моделировании высокоуровневых абстракций (новых признаков) с помощью последовательных нелинейных преобразований.

Более высокие уровни нейронной сети представляют абстракцию на базе предыдущих слоев.

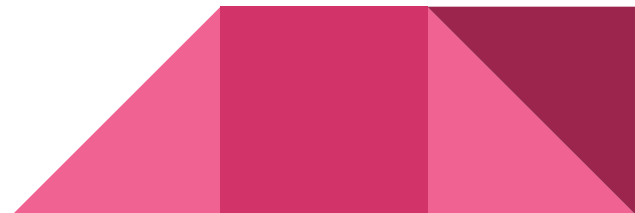


Глубокое обучение

- Больше данных
- Глубже модели
- Дольше обучение



Выше точность!



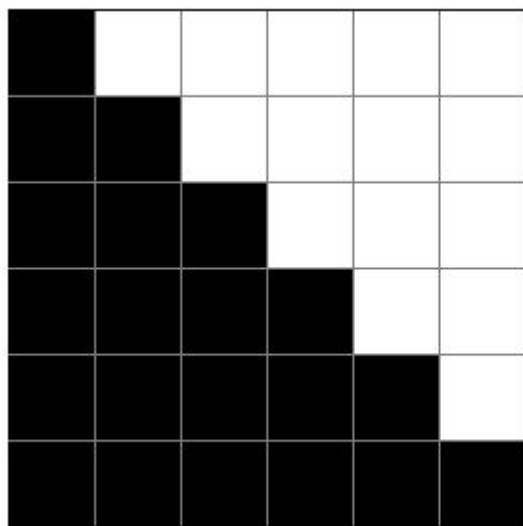
Сверточные сети

Линейный фильтр $I * K$ с ядром K :

$$(I * K)_{pq} = \sum_{i=1}^h \sum_{j=1}^w I_{p+i-1, q+j-1} K_{ij}$$

Например,

$$K = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$



0	1	0
1	-4	1
0	1	0

$$(I * K)_{pq} = \sum_{i=1}^h \sum_{j=1}^w I_{p+i-1, q+j-1} K_{ij}$$

$$K = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

0	255	255	255	255	255
0	0	255	255	255	255
0	0	0	255	255	255
0	0	0	0	255	255
0	0	0	0	0	255
0	0	0	0	0	0

0	1	0
1	-4	1
0	1	0

$$(I * K)_{pq} = \sum_{i=1}^h \sum_{j=1}^w I_{p+i-1, q+j-1} K_{ij}$$

$$K = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

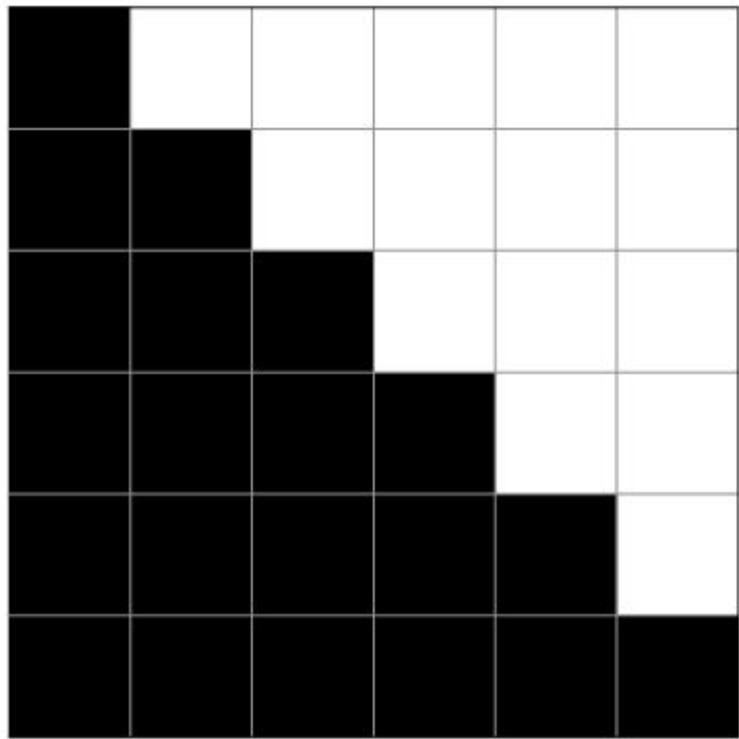
0	255	255	255	255	255
0	0	255	255	255	255
0	0	0	255	255	255
0	0	0	0	255	255
0	0	0	0	0	255
0	0	0	0	0	0

0	1	0
1	-4	1
0	1	0

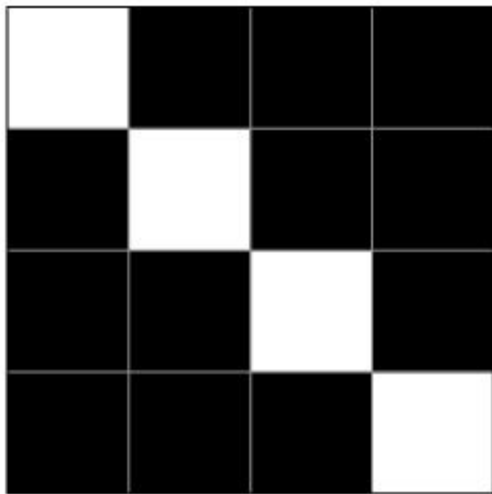
510	-510	0	0
0	510	-510	0
0	0	510	-510
0	0	0	510

$$(I * K)_{pq} = \sum_{i=1}^h \sum_{j=1}^w I_{p+i-1, q+j-1} K_{ij}$$

$$K = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$



0	1	0
1	-4	1
0	1	0





Линейный фильтр (свертка) $I * K$ с ядром K :

$$(I * K)_{pq} = \sum_{i=1}^h \sum_{j=1}^w I_{p+i-1, q+j-1} K_{ij}$$

$$K = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Сверточные сети

Основная идея сверточных сетей (сверточных слоев):

Параметры фильтров будем подбирать с помощью обучения

$$z_{pq} = \sigma \left(\beta_0 + \sum_{i=1}^h \sum_{j=1}^w \beta_{ij} x_{p+i-1, q+j-1} \right)$$

x_{ij} - узлы (нейроны) одного слоя (например, входного)

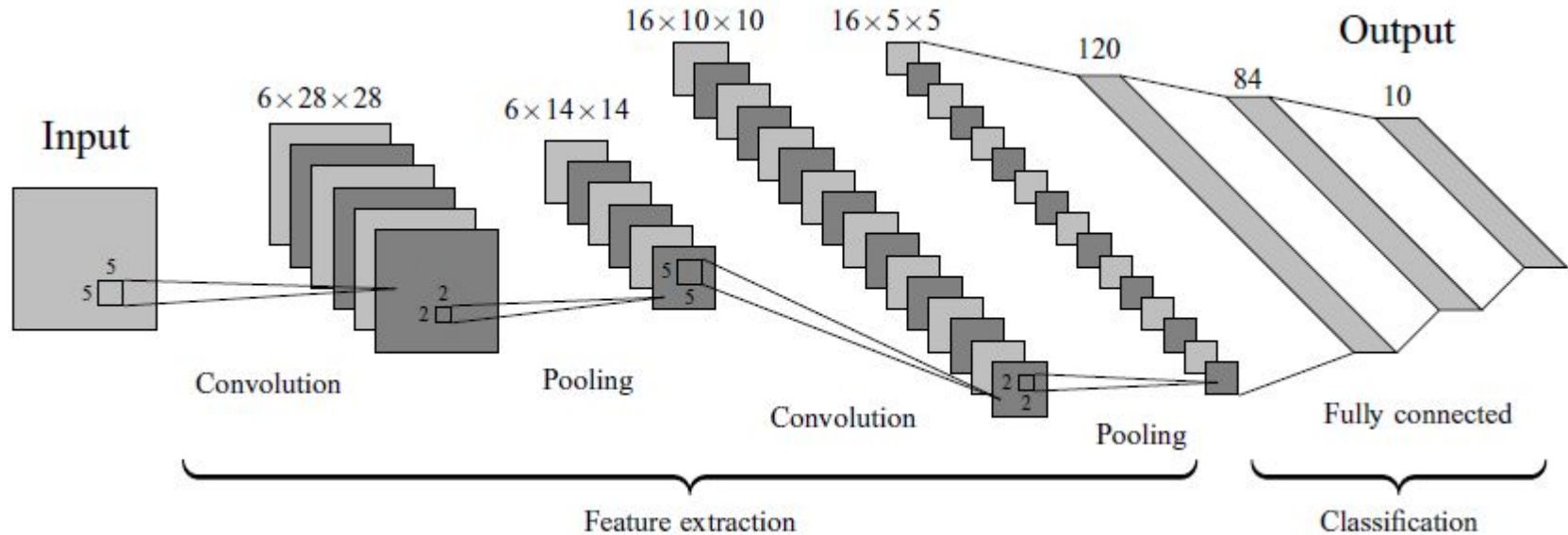
z_{pq} - узлы следующего слоя

Параметры фильтра - это теперь веса нейронной сети.

Отличия от полносвязной сети (полносвязного слоя):

- Нет соединения каждого узла одного слоя со всеми узлами следующего.
- Веса становятся *разделяемыми*.

LeNet-5 [Le Cun et al., 1998]

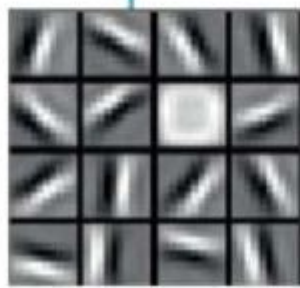


- Сверточные слои (convolutional layers)
- «Выборочные» слои, или слои объединения (subsampling/pooling layers)
- Полносвязные слои (fully connected layers)
- Регуляризация (weight decay, dropout, normalization)

Input Layer



Hidden Layer 1



edges

Hidden Layer 2



combinations of edges

Hidden Layer 3



object models



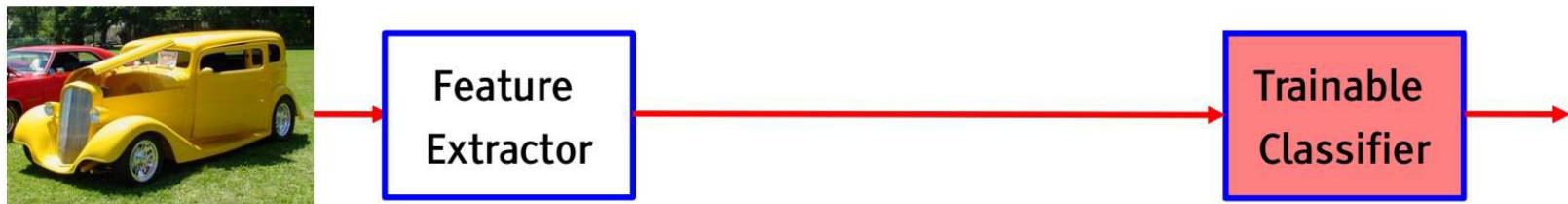
Traditional Pattern Recognition: Fixed/Handcrafted Feature Extractor



Feature
Extractor

Trainable
Classifier

Traditional Pattern Recognition: Fixed/Handcrafted Feature Extractor



Mainstream Modern Pattern Recognition: Unsupervised mid-level features



Deep Learning: Representations are hierarchical and trained



Другие примеры глубоких нейронных сетей

- *AlexNet* (Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton, 2012) — победитель ImageNet-2012 — 8 слоев, 61 млн. параметров
- *GoogLeNet* (Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, 2015) — 22 слоя, 7 млн. параметров
- Microsoft *ResNet* (Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, 2015) — 101 слой, 44.6 млн. параметров
- *DenseNet* (Gao Huang, Zhuang Liu, Laurens Van Der Maaten, Kilian Q. Weinberger, 2017) — 201 слой, 20 млн. параметров
- ...
- Playing Atari with Deep Reinforcement Learning (V. Mnih, 2013) — 10 слоев
- AlphaGo (D.Silver et al, Alphabet Inc.'s Google DeepMind, 2015) — 2 нейронных сети по 13 слоев

Библиотеки глубокого обучения

- TensorFlow
- Theano
- Keras
- Torch
- Caffe

