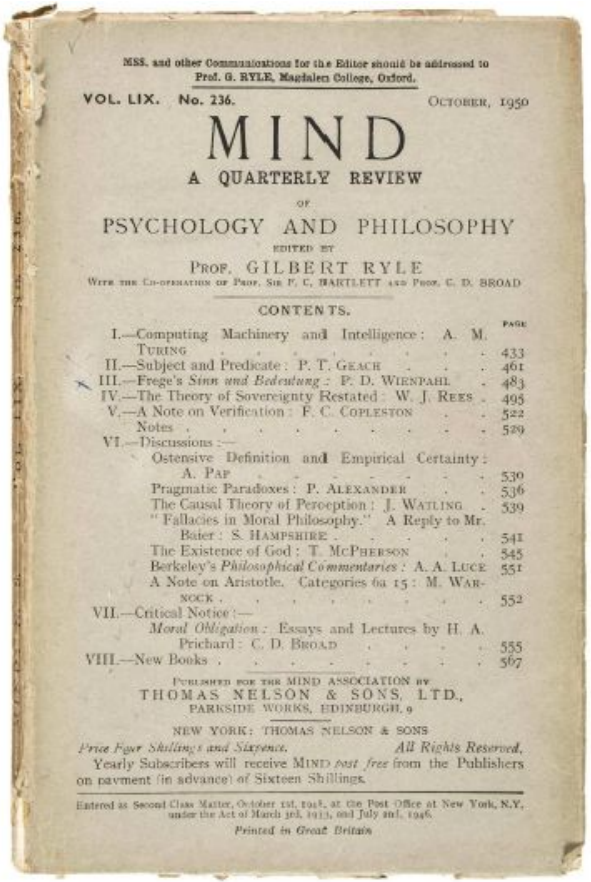
Введение
в машинное обучение

Николай 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. Тюрингу

[A. Turing 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.: А. М. Тьюринг Может ли машина мыслить? // М.: Физматлит, 1960]



MSS. and other Communications for the Editor should be addressed to
Prof. G. RYLE, Magdalen College, Oxford.

VOL. LIX. No. 236. OCTOBER, 1950

MIND

A QUARTERLY REVIEW

OF

PSYCHOLOGY AND PHILOSOPHY

EDITED BY

PROF. GILBERT RYLE

WITH THE CO-OPERATION OF PROF. SIR F. C. BARTLETT AND PROF. C. D. BROAD

CONTENTS.

PUBLISHED FOR THE MIND ASSOCIATION BY
THOMAS NELSON & SONS, LTD.,
PARKSIDE WORKS, EDINBURGH, 9

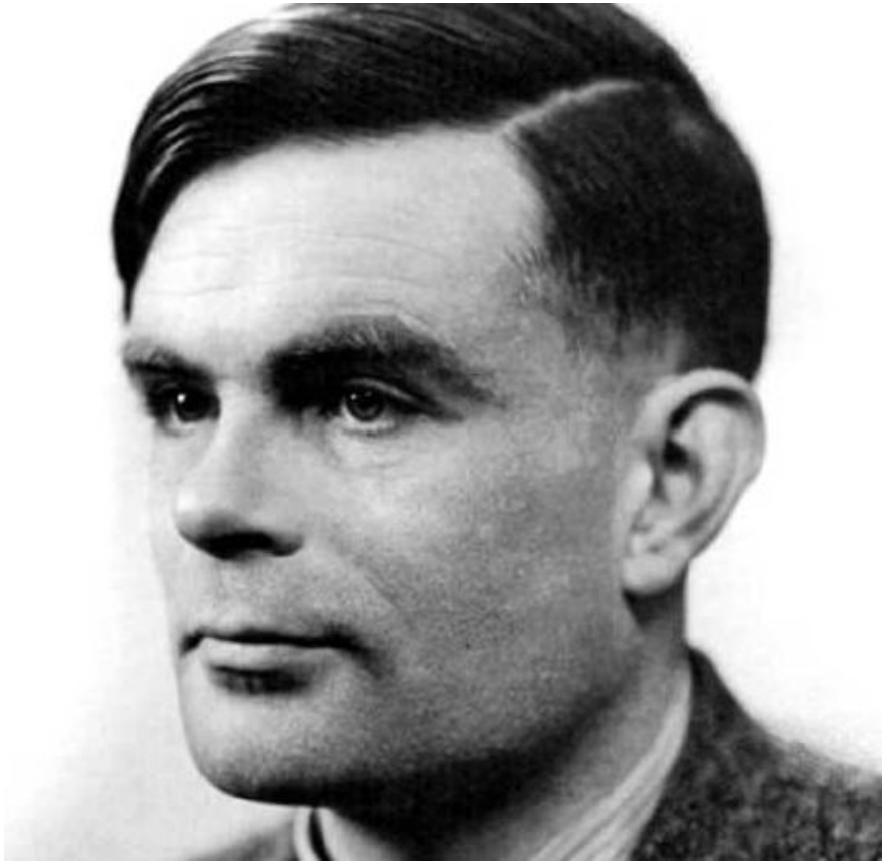
NEW YORK: THOMAS NELSON & SONS

Price Four Shillings and Sixpence.

Yearly Subscribers will receive MIND *post free* from the Publishers on payment (in advance) of Sixteen Shillings.

Entered as Second Class Matter, October 1st, 1918, at the Post Office at New York, N.Y., under the Act of March 3rd, 1913, and July 2nd, 1946.

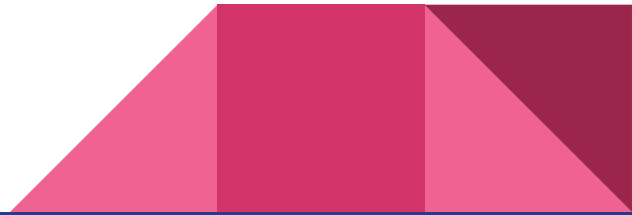
Printed in Great Britain



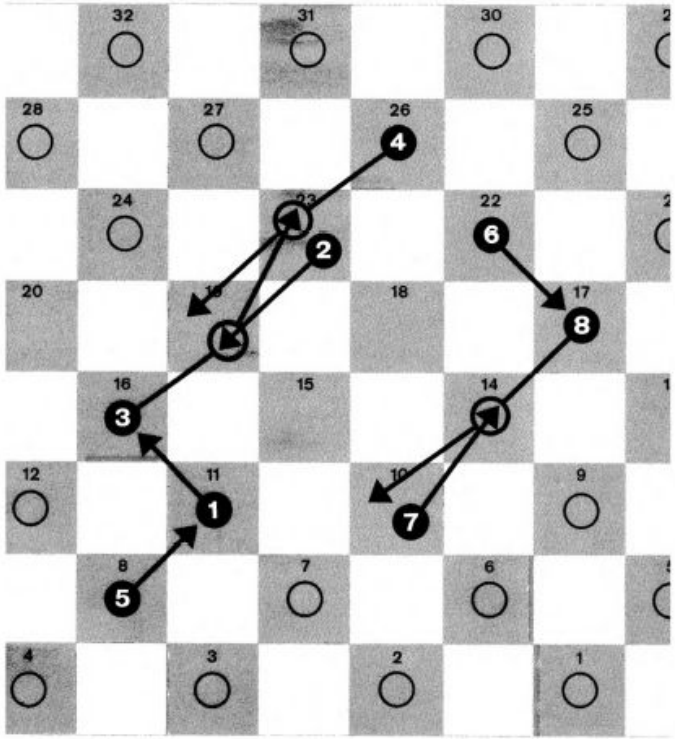
Alan Mathison Turing (1912–1954)

Машинное обучение — процесс, в результате которого машина (компьютер) способна показывать поведение, которое в нее не было явно заложено (запрограммировано). [A.L. Samuel, 1959]

Говорят, что компьютерная программа *обучается* на основе опыта *E* по отношению к некоторому классу задач *T* и меры качества *P*, если качество решения задач из *T*, измеренное на основе *P*, улучшается с приобретением опыта *E*. [T.M.Mitchell, 1997]



Arthur Lee Samuel (1901–1990)
Исследования по машинному обучению – примерно с 1949 г.



Eight-move opening utilizing generalization learning. (See Appendix B, Game G-43.)

Some Studies in Machine Learning Using the Game of Checkers

Abstract: Two machine-learning procedures have been investigated in some detail using the game of checkers. Enough work has been done to verify the fact that a computer can be programmed so that it will learn to play a better game of checkers than can be played by the person who wrote the program. Furthermore, it can learn to do this in a remarkably short period of time (8 or 10 hours of machine-playing time) when given only the rules of the game, a sense of direction, and a redundant and incomplete list of parameters which are thought to have something to do with the game, but whose correct signs and relative weights are unknown and unspecified. The principles of machine learning verified by these experiments are, of course, applicable to many other situations.

Introduction

The studies reported here have been concerned with the programming of a digital computer to behave in a way which, if done by human beings or animals, would be described as involving the process of learning. While this is not the place to dwell on the importance of machine-learning procedures, or to discourse on the philosophical aspects,[1] there is obviously a very large amount of work, now done by people, which is quite trivial in its demands on the intellect but does, nevertheless, involve some learning. We have at our command computers with adequate data-handling ability and with sufficient computational speed to make use of machine-learning techniques, but our knowledge of the basic principles of these techniques is still rudimentary. Lacking such knowledge, it is necessary to specify methods of problem solution in minute and exact detail, a time-consuming and costly procedure. Programming computers to learn from experience should eventually eliminate the need for much of this detailed programming effort.

- *General methods of approach*

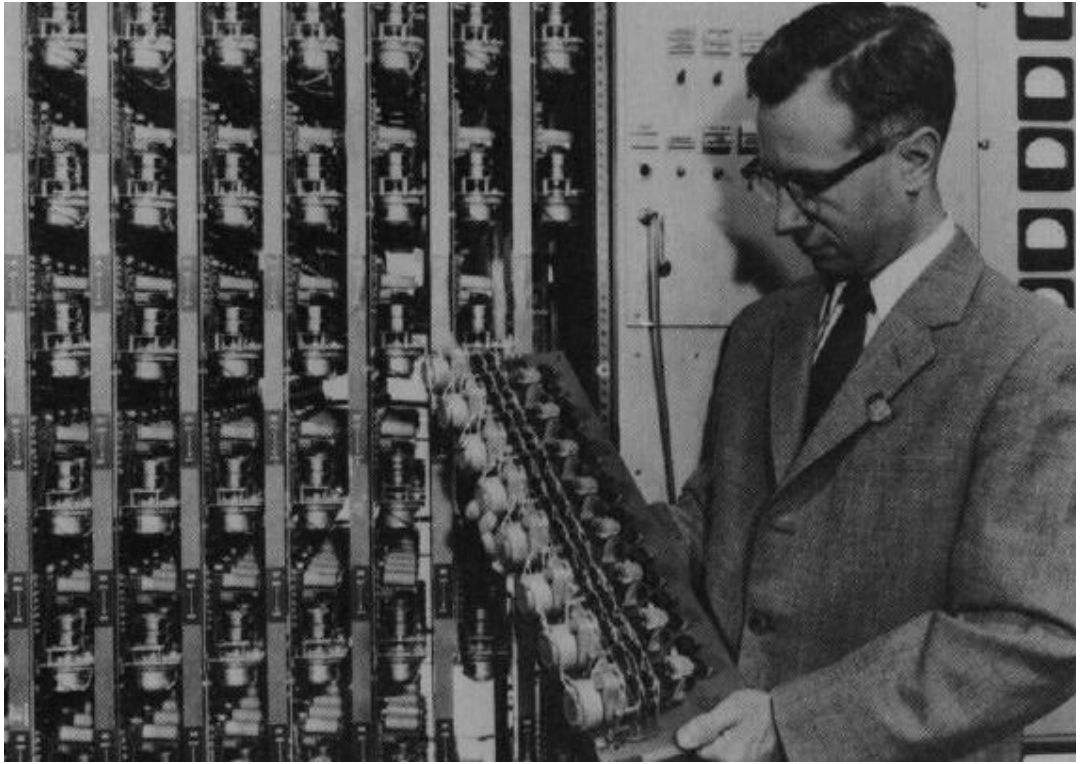
At the outset it might be well to distinguish sharply between two general approaches to the problem of machine learning. One method, which might be called the *Neural-Net Approach*, deals with the possibility of inducing learned behavior into a randomly connected switching net (or its simulation on a digital computer) as a result of a reward-and-punishment routine. A second, and much more efficient approach, is to produce the equivalent of a highly organized network which has been designed to learn only certain specific things. The first method should lead to the development of general-purpose learning machines. A comparison between the size of the switching nets that can be reasonably constructed or simulated at the present time and the size of the neural nets used by animals, suggests that we have a long way to go before we obtain practical devices.[2] The second procedure requires reprogramming for each new application, but it is capable of realization at the present time. The experiments to be described here were based on this second approach.

- *Choice of problem*

For some years the writer has devoted his spare time to the subject of machine learning and has concentrated on the development of learning procedures as applied to games.[3] A game provides a convenient vehicle for such study as contrasted with a problem taken from life, since many of the complications of detail are removed. Checkers, rather than chess,[4-7] was chosen because the simplicity of its rules permits greater emphasis to be placed on learning techniques. Regardless of the relative merits of the two games as intellectual pastimes, it is fair to state that checkers contains all of the basic characteristics of an intellectual activity in which heuristic procedures and learning processes can play a major role and in which these processes can be evaluated.

Some of these characteristics might well be enumerated. They are:

(1) The activity must not be deterministic in the practical sense. There exists no known algorithm which will guarantee a win or a draw in checkers, and the complete



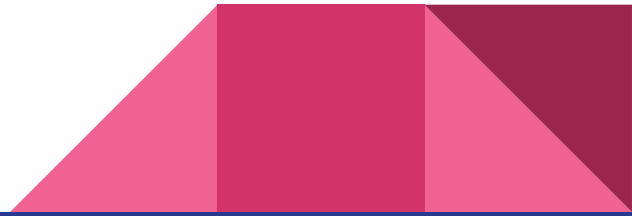
Frank Rosenblatt (1928–1971)
Программная реализация персептрона – 1957 г.
Первый нейронный компьютер – MARK 1 – 1958 г.

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)

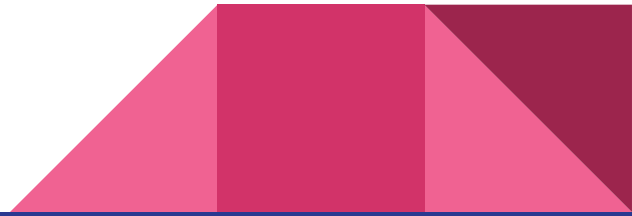
Машинное обучение сегодня

Причины “Второй весны AI”:

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

Достижения:

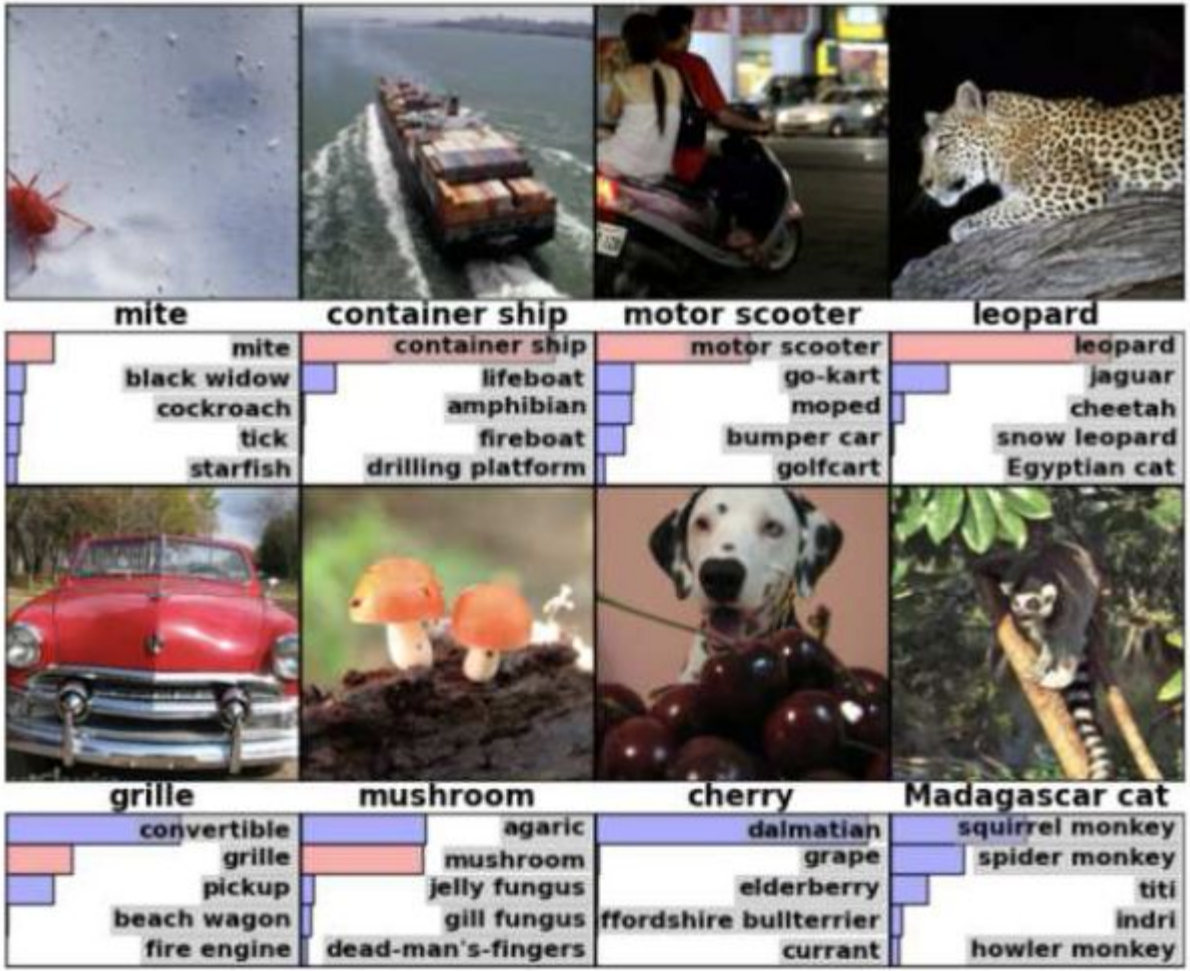
- …
- Машинное обучение стало технологией



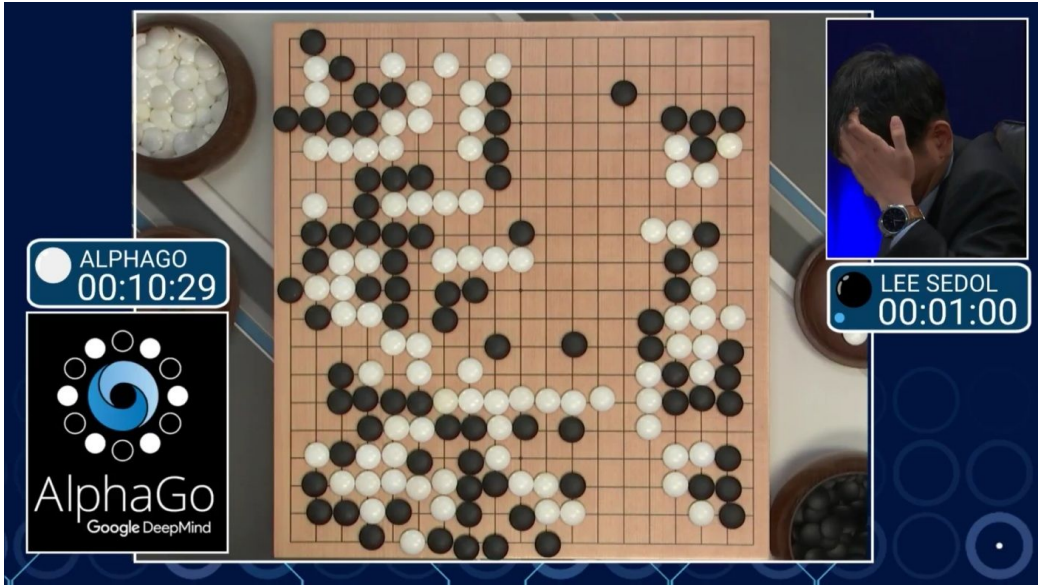
ImageNet

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

Ошибку удалось понизить 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 Седолем

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

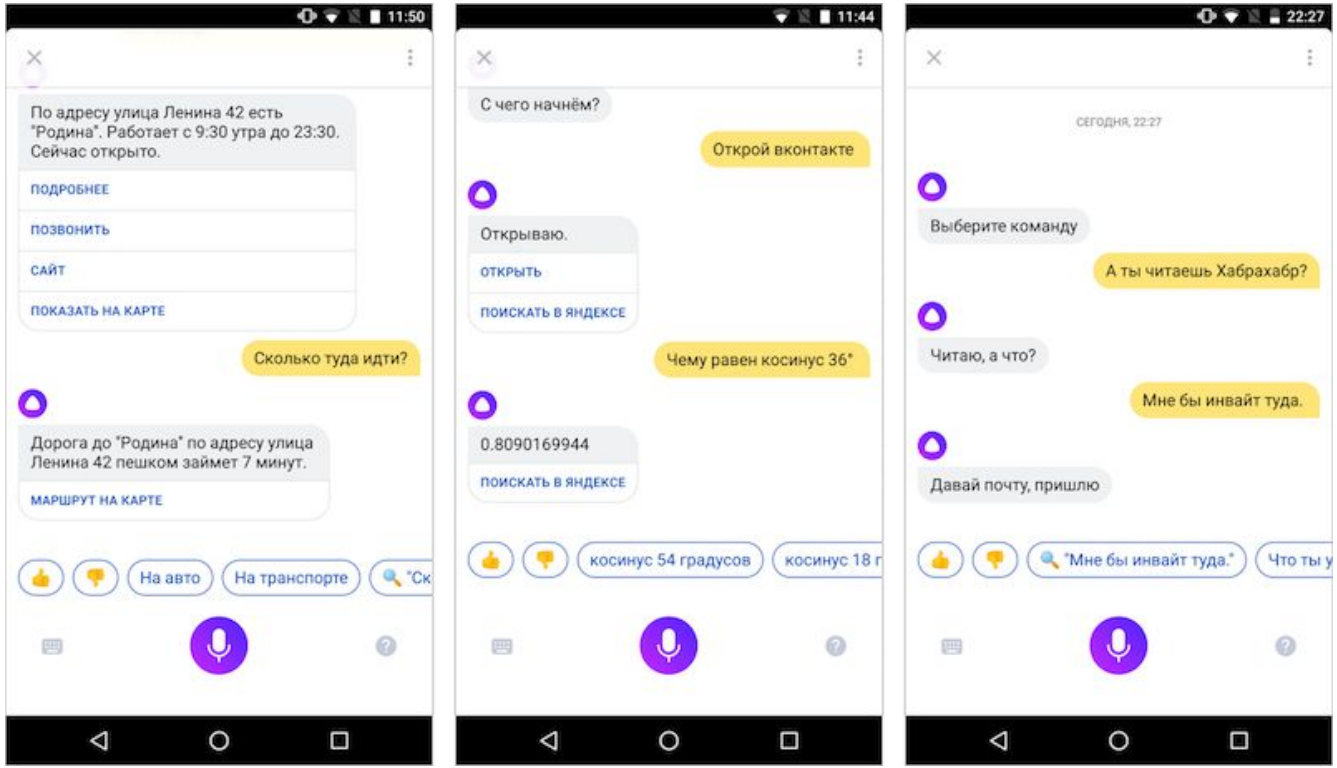
BigDog,
CHEETAH,
LittleDog, RiSE,
PETMAN, Atlas,
Handle,
SpotMini, ...

IBM Watson

Вопросно-ответная система на естественном языке

- 2011 - победа в Jeopardy
- Watson Healph
- Когнитивный компьютер для бизнеса

Yandex Алиса



Некоторые последние достижения

- Беспилотные автомобили (Google и др.)
- AlphaGo, AlphaGo Zero, AlphaZero
- Синтез речи (Google WaveNet, Tacotron-2, Baidu DeepSpeach3, ...)
- Вопросно-ответные системы на естественном языке (IBM Watson)
- Умные персональные помощники и чат-боты (Ok Google, Apple Siri, Yandex Алиса, ...)
- Генеративные модели (перенос стиля, генерация речи, музыки, стихов, молекул, ...)
- Автоматический перевод (Google Translate, ...)
- ...

Machine Learning *vs* Data Mining

Data Mining (добыча данных, интеллектуальный анализ данных, глубинный анализ данных) — совокупность методов обнаружения в данных ранее неизвестных, нетривиальных, практически полезных и доступных интерпретации знаний, необходимых для принятия решений в различных сферах человеческой деятельности. [Г.Пятецкий-Шапиро, 1989]

Итак, и ML, и DM извлекают закономерности («знания») из данных, но (немного) с разными целями:

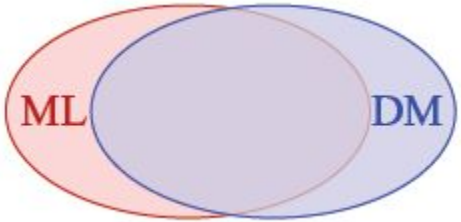
- ML — чтобы обучить машину;
- DM — чтобы обучить человека.

Поэтому

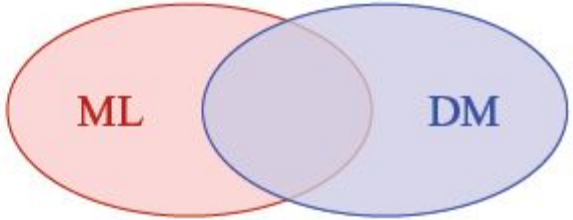
- в ML минимизируют ошибку;
- в DM важна интерпретируемость результата.

Machine Learning *vs* Data Mining

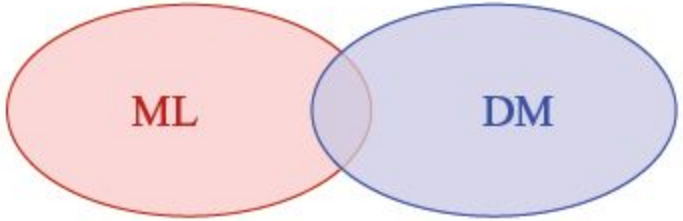
Методы



Математические постановки задач



Содержательные постановки задач



Machine Learning *vs* Data Mining

Есть немного другая точка зрения на вопрос, чем ML отличается от DM:

- Data Mining имеет дело с содержательными задачами, а Machine Learning —с математической теорией,

отсюда немного странные термины, например,

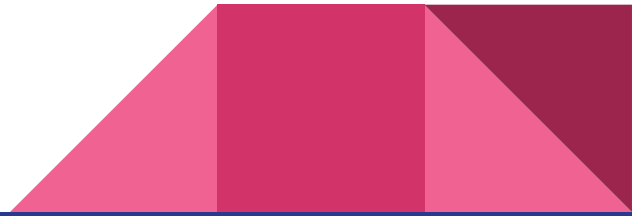
- «алгоритмы машинного обучения в анализе данных»

Software

- Python Scikit-Learn scikit-learn.org и Pandas pandas.pydata.org
- Система для статистических вычислений R r-project.org
- MATLAB Statistics and Machine Learning Toolbox + Neural Network Toolbox mathworks.com
- Библиотека алгоритмов для анализа данных Weka (Java) www.cs.waikato.ac.nz/~ml/weka
- Пакет для решения задач машинного обучения и анализа данных Orange orange.biolab.si
- ...

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) $f : X \to Y$, 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
- Компьютерная лингвистика
  Предложение на русском языке → Дерево синтаксического разбора
- Машинный перевод
  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: как вектор (набор) признаков фиксированной длины *d*

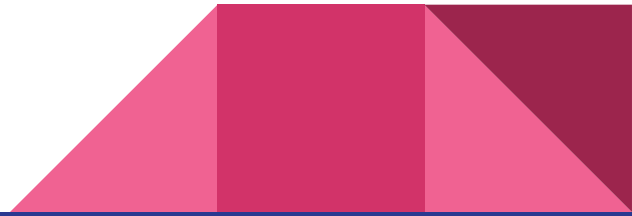
Признак может быть

- номинальным (принимает конечное число значений)
- количественным (принимает вещественные значения)
- ...

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:

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

i-я строка соответствует *i*-му объекту в обучающей выборке

j-й столбец – *j*-му признаку

Пример 1

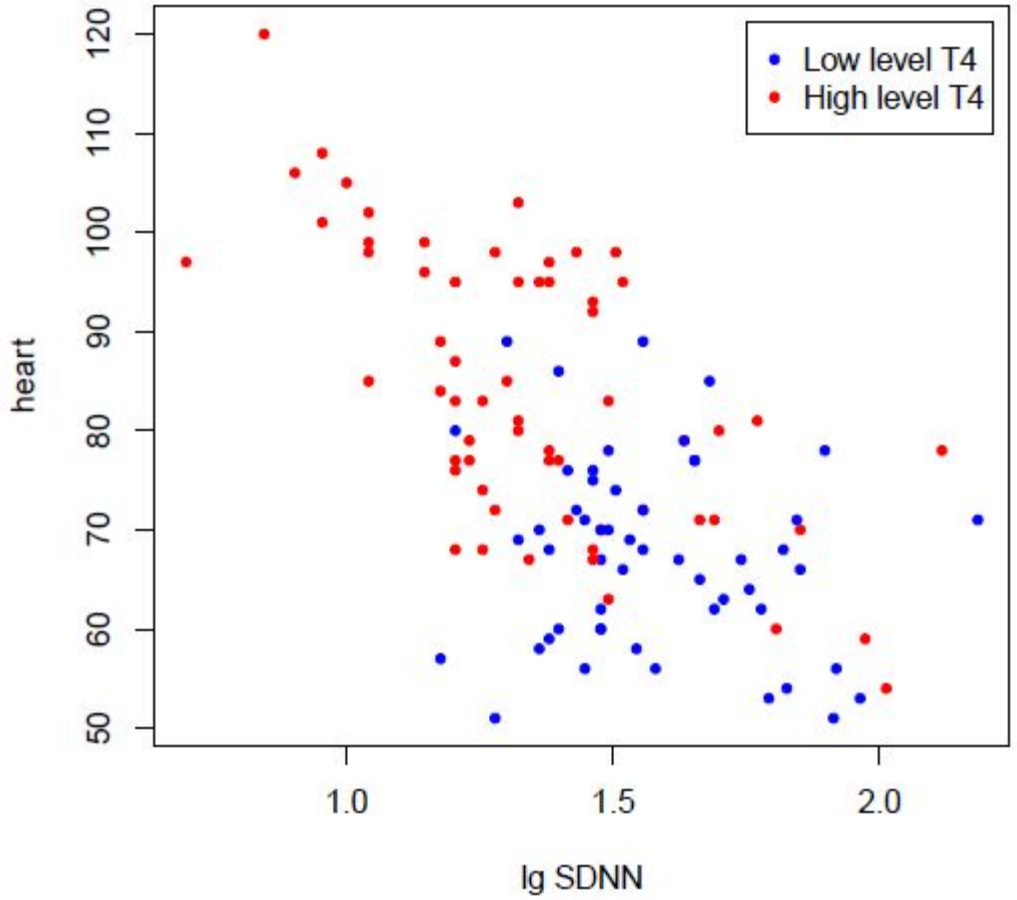
Имеются данные о 114 лицах с заболеванием щитовидной железы.

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

Для каждого пациента известны следующие показатели:

- x1 = heart — частота сердечных сокращений (пульс),
- x2 = SDNN — стандартное отклонение длительности интервалов между синусовыми сокращениями сердца.

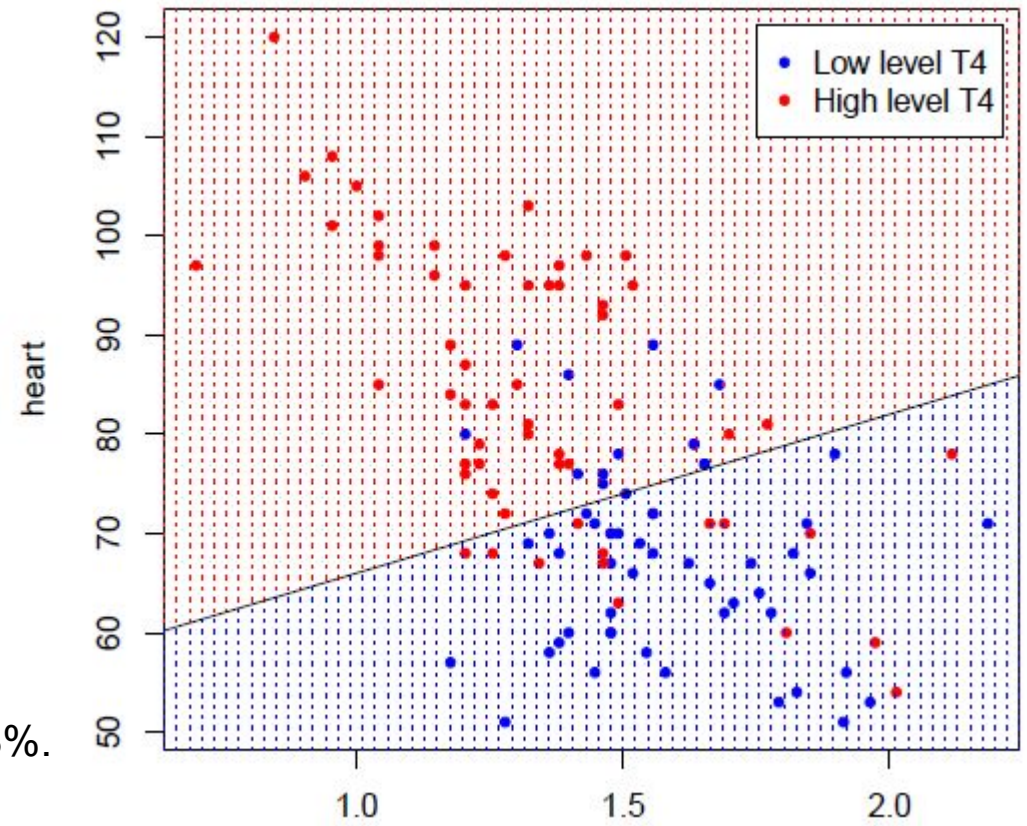
Можно ли научиться предсказывать уровень свободного T4 по heart и SDNN?


Low level T4
High level T4
heart
50
60
70
80
90
100
110
120
1.0
1.5
2.0
lg SDNN

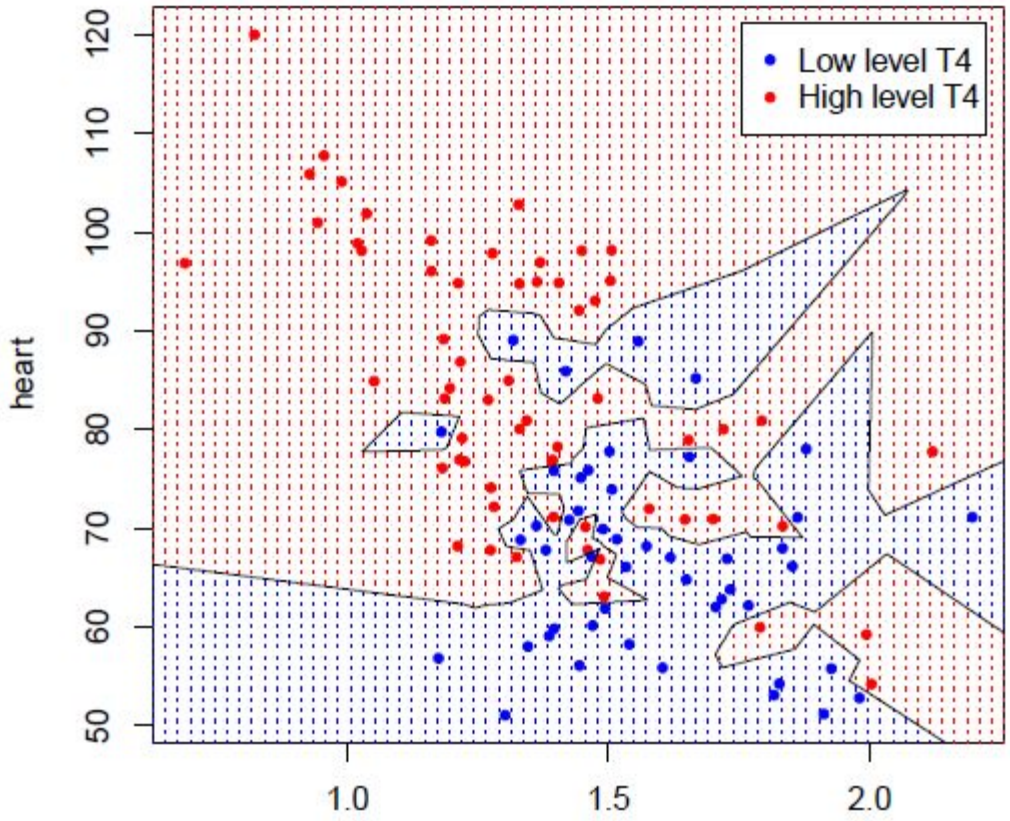
$16 \cdot \lg \mathrm{SDNN} - \mathrm{heart} + 50 = 0$

Ошибка на обучающей выборке 23%.

Можно ли ее сделать меньше?



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



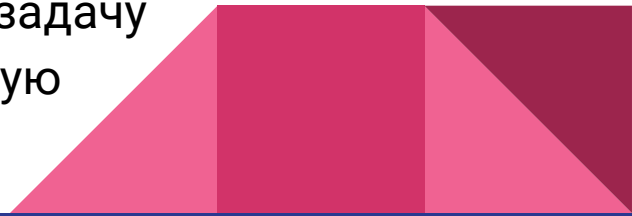
Переобучение

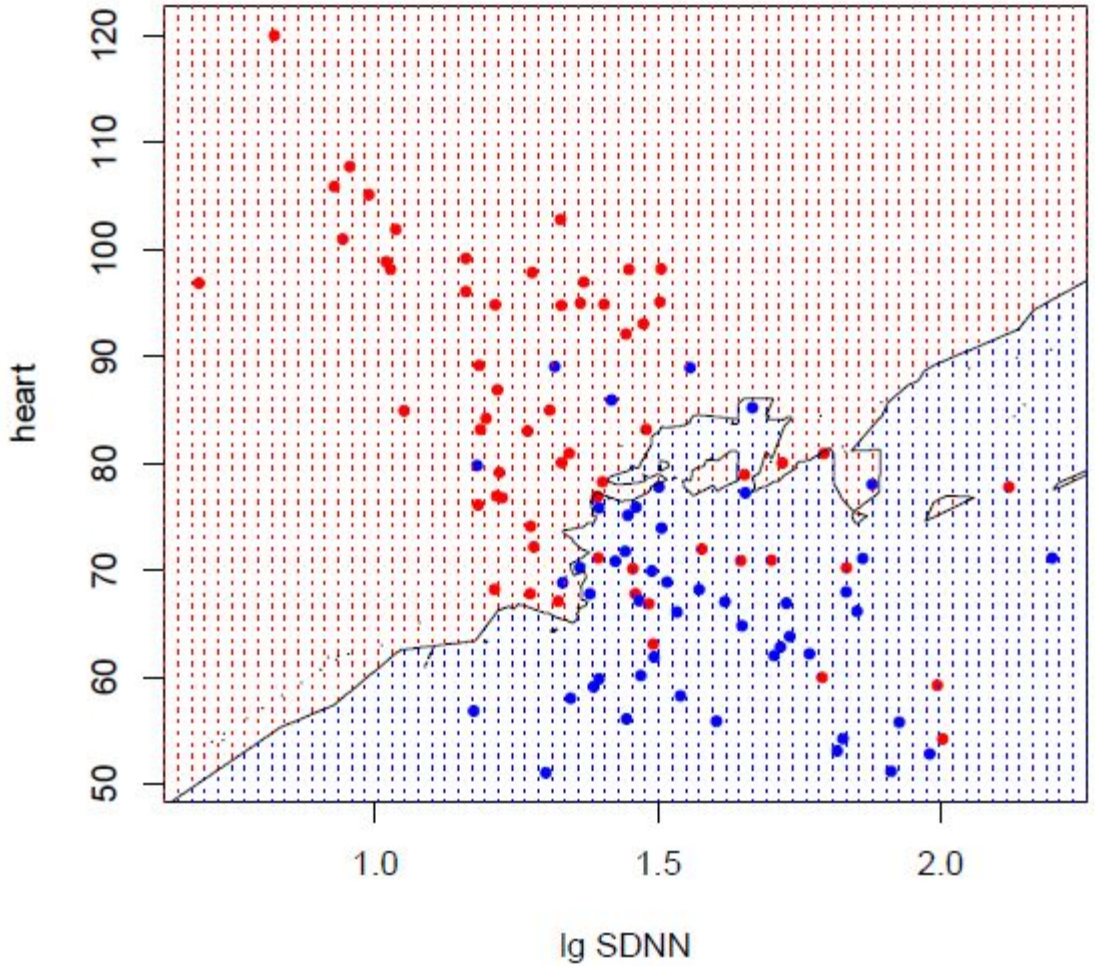
Малая ошибка на обучающей выборке не означает, что мы хорошо классифицируем новые объекты.

Итак, малая ошибка на данных, по которым построено решающее правило, не гарантирует, что ошибка на новых объектах также будет малой.

Обобщающая способность (*качество*) решающего правила — это способность решающего правила правильно предсказывать выход для новых объектов, не вошедших в обучающую выборку.

Переобучение — решающее правило хорошо решает задачу на обучающей выборке, но имеет плохую обобщающую способность.





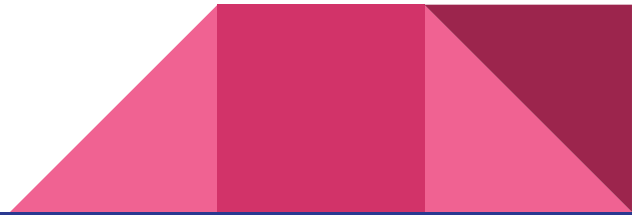
Метод 15 ближайших соседей

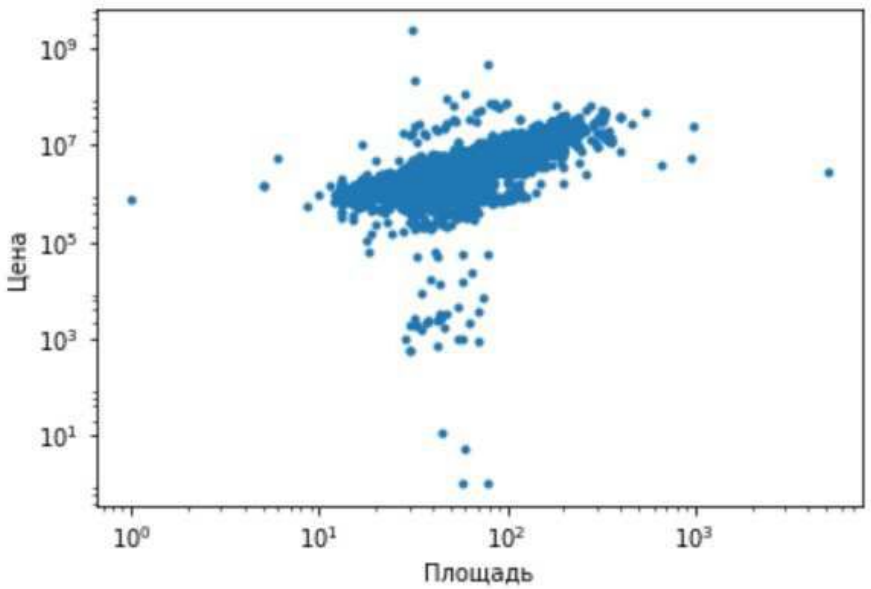
Пример 2

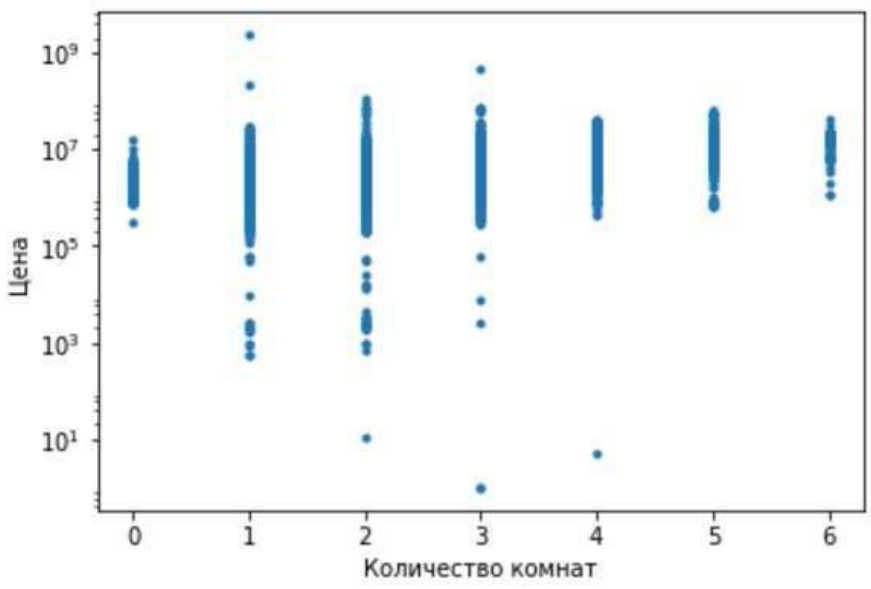
Имеются данные о стоимости 72379 квартир

Требуется научиться предсказывать цену — задача восстановления *регрессии*

1. Дата
2. Широта (числовой)
3. Долгота (числовой)
4. Вид объекта (новостройка, вторичка)
5. Этажей в доме (числовой)
6. Тип дома (кирпичный, панельный, блочный, монолитный, деревянный)
7. Количество комнат (студия, 1, 2, . . .)
8. Площадь (числовой)
9. Цена (числовой)

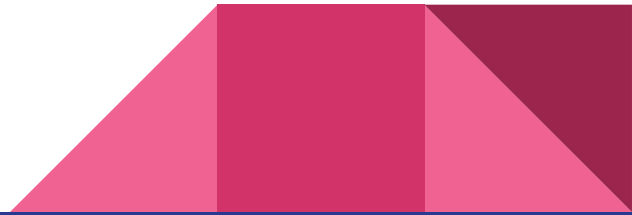



Цена
10^9
10^7
10^5
10^3
10^1
10^0
10^1
10^2
10^3
Площадь

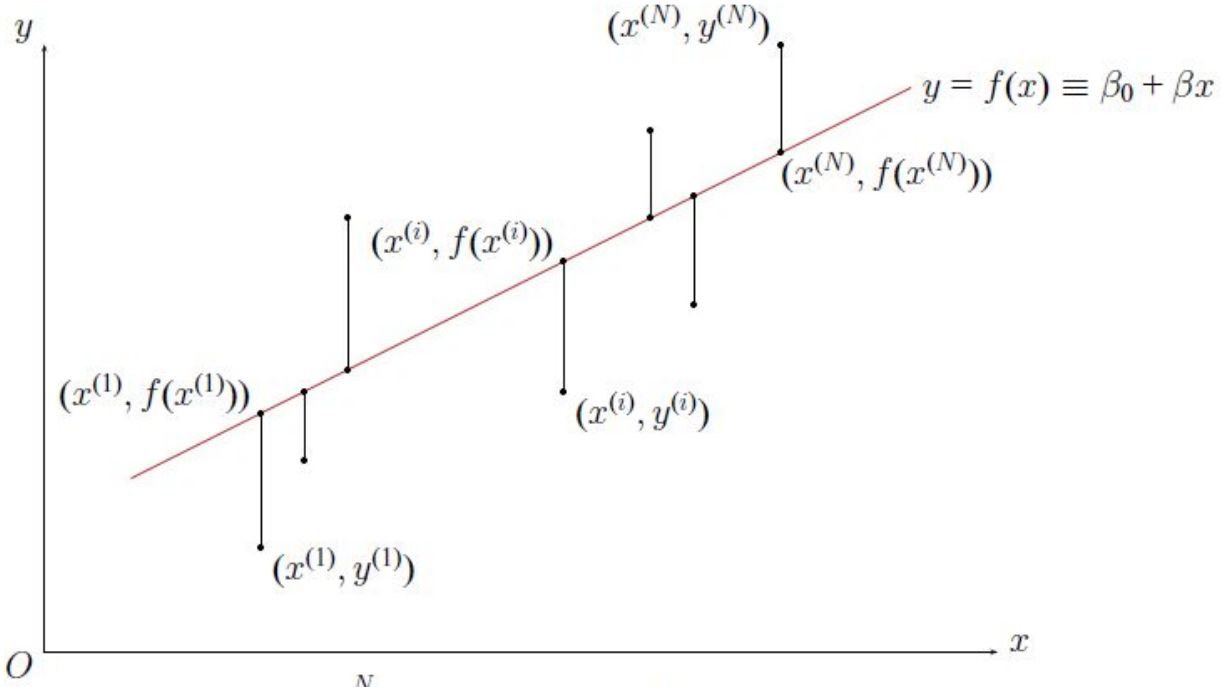

Цена
10^9
10^7
10^5
10^3
10^1
0
1
2
3
4
5
6
Количество комнат

Некоторые методы ML

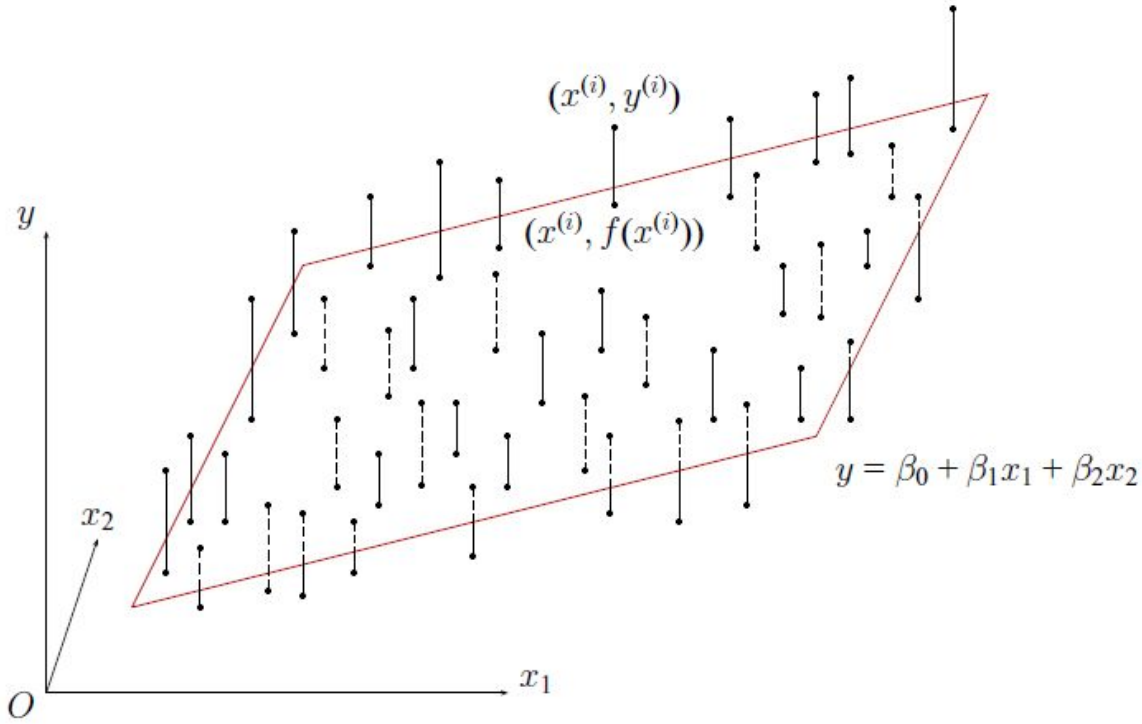
- Метод наименьших квадратов
- Линейный и квадратичный дискриминантный анализ
- Логистическая регрессия
- Метод *k* ближайших соседей
- Наивный байесовский классификатор
- Машина опорных векторов (SVM)
- Деревья решений (C4.5, CART и др.)
- Ансамбли решающих функций (бустинг, баггинг и т. п.)
- Нейронные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



Метод наименьших квадратов (линейная регрессия)

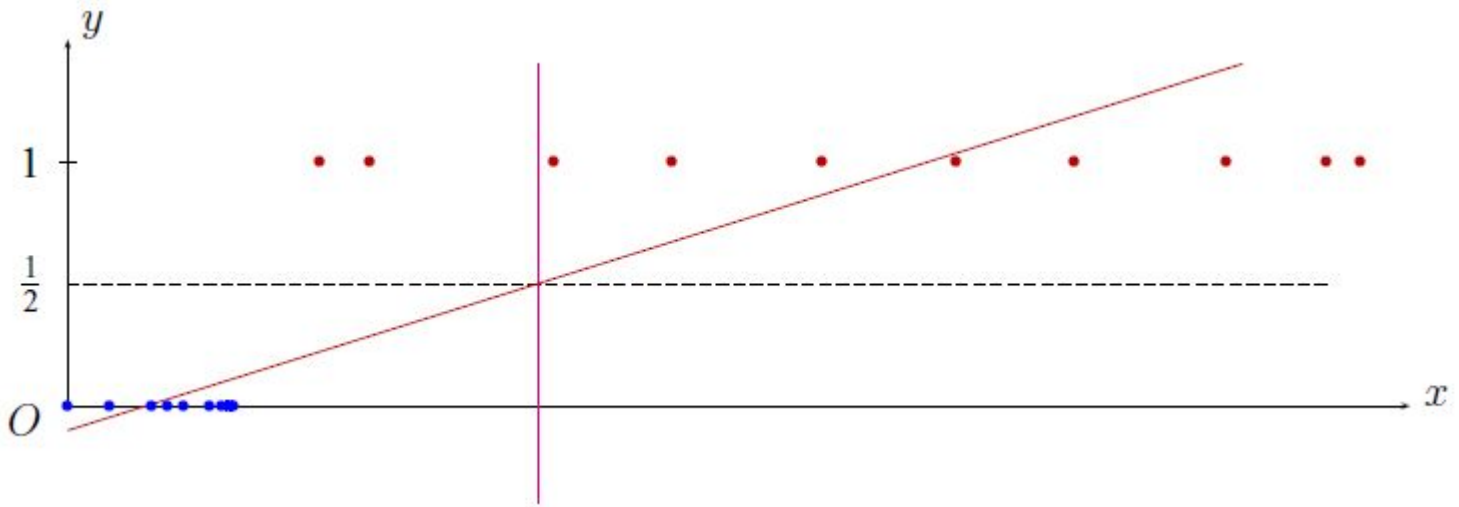


$$\sum_{i=1}^{N} \left(\beta_0 + \beta x^{(i)} - y^{(i)}\right)^2 \to \min_{\beta_0, \beta}$$

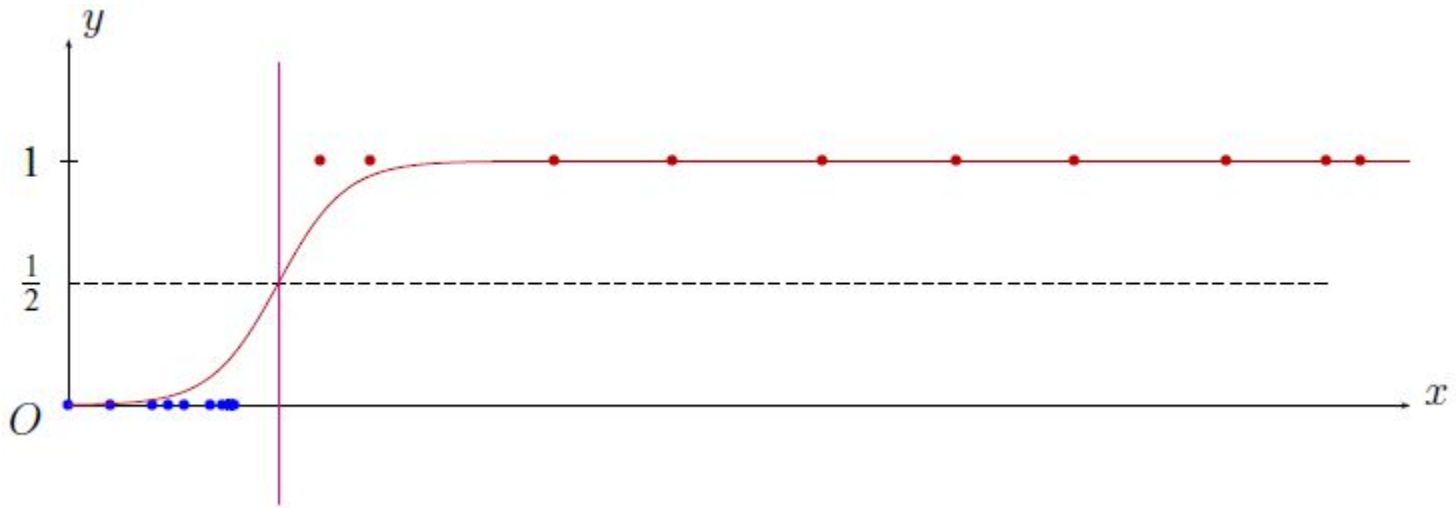


$$\sum_{i=1}^{N} \left(\beta_0 + \sum_{j=1}^{d} \beta_j x_j^{(i)} - y^{(i)} \right)^2 \to \min$$

Метод наименьших квадратов для задачи классификации (?)



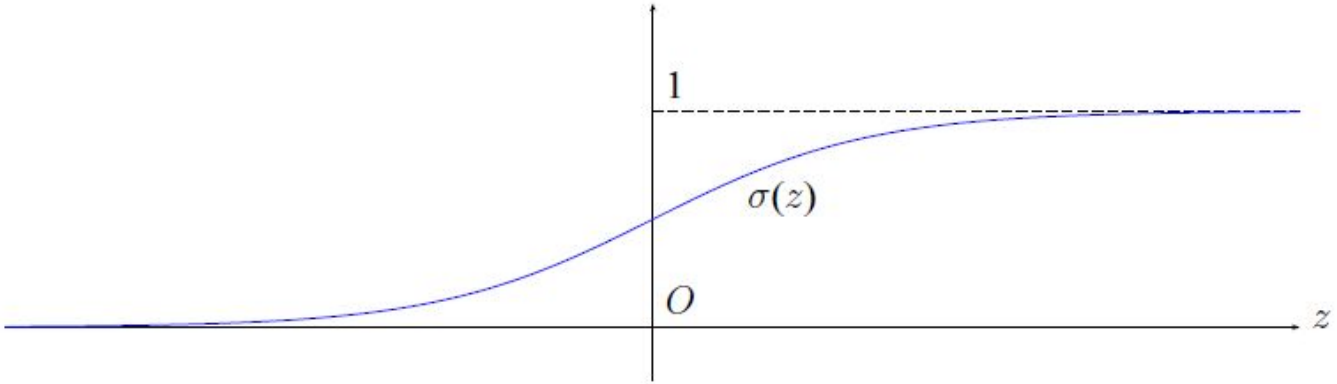
Метод наименьших квадратов для задачи классификации (?)



Логистическая регрессия

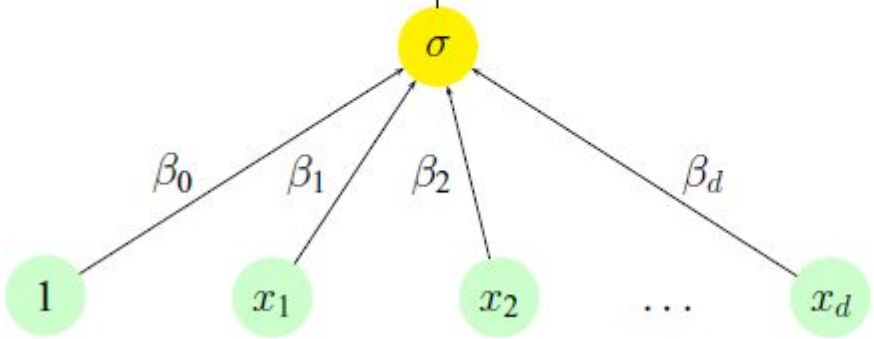
$$\Pr(Y = 1 \mid X = x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \ldots + \beta_d x_d)}} = \sigma(\beta_0 + \beta^\top x),$$

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

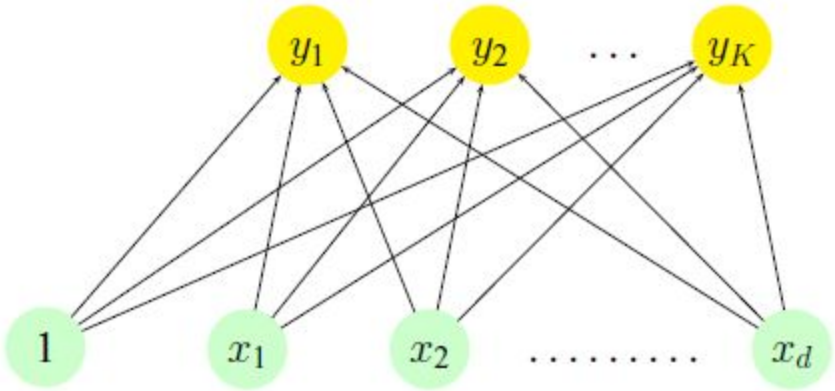


$$\Pr(Y = 1 \mid X = x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \ldots + \beta_d x_d)}} = \sigma(\beta_0 + \beta^\top x), \qquad \sigma(z) = \frac{1}{1 + e^{-z}}$$

$$g = \sigma\left(\beta_0 + \sum_{j=1}^{d} \beta_j x_j\right) \approx \Pr(Y = 1 \mid X = x)$$



Случай *K* классов:



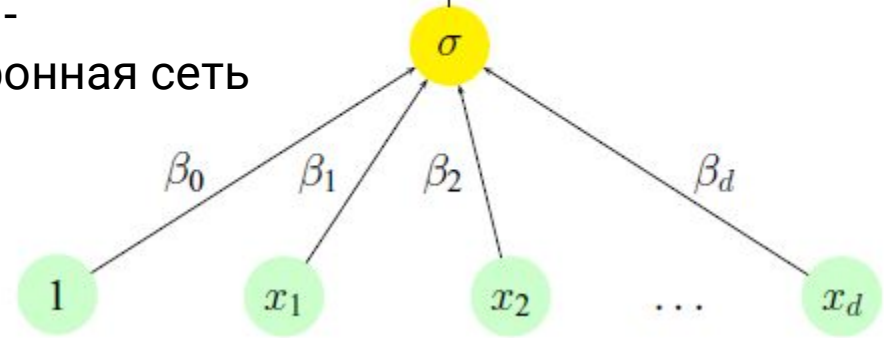
$$y_k = \frac{\exp\left(\beta_{k0} + \sum_{j=1}^{d} \beta_{kj} x_j\right)}{\sum_{\ell=1}^{K} \exp\left(\beta_{\ell 0} + \sum_{j=1}^{d} \beta_{\ell j} x_j\right)} \approx \Pr(k \mid x)$$

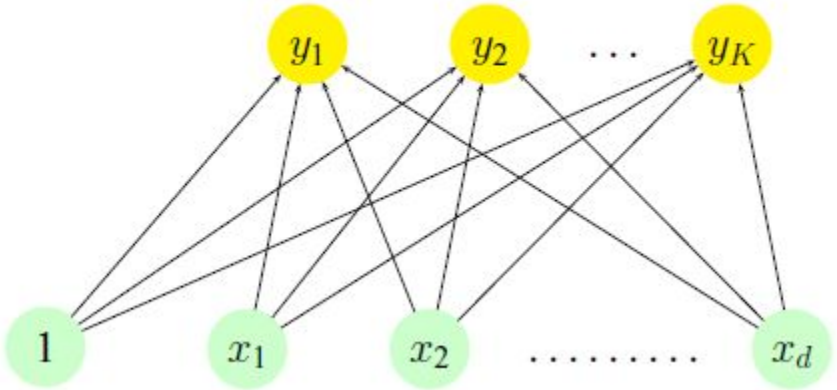
$$(k = 1, 2, \ldots, K)$$

Нейронные сети

Логистическая регрессия -
это уже двуслойная* нейронная сеть

$$g = \sigma\left(\beta_0 + \sum_{j=1}^{d} \beta_j x_j\right) \approx \Pr(Y = 1 \mid X = x)$$

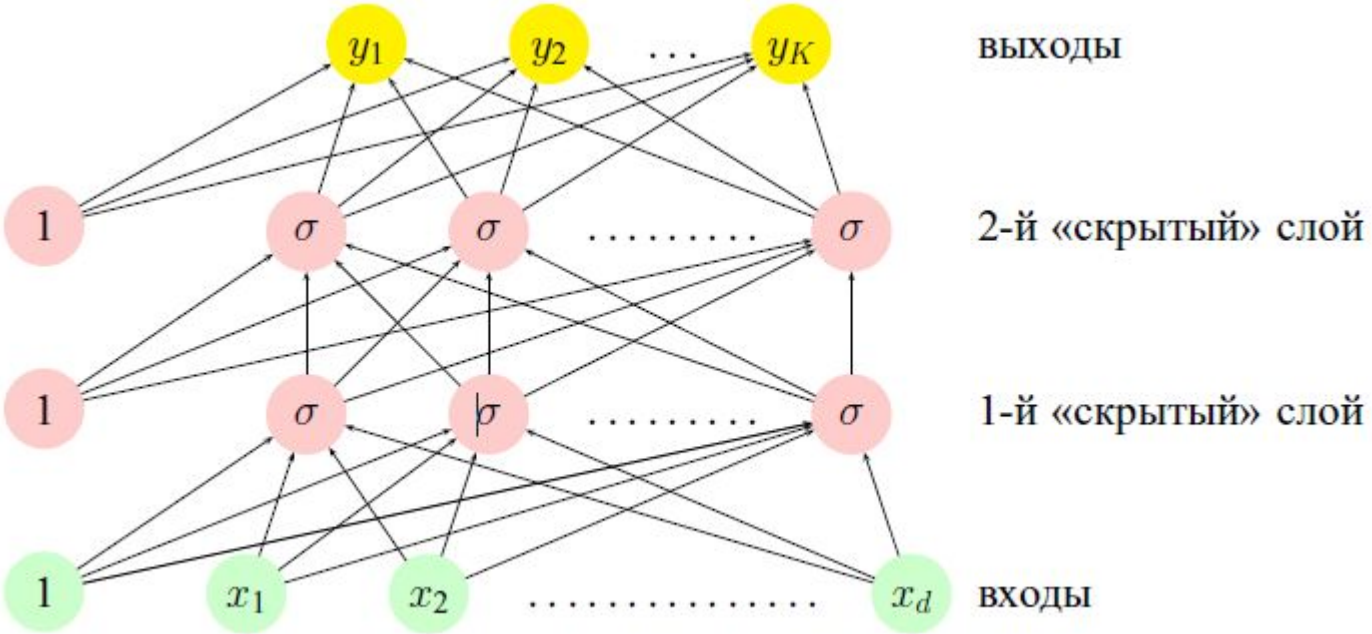




$$y_k = \frac{\exp\left(\beta_{k0} + \sum_{j=1}^{d} \beta_{kj} x_j\right)}{\sum_{\ell=1}^{K} \exp\left(\beta_{\ell 0} + \sum_{j=1}^{d} \beta_{\ell j} x_j\right)} \approx \Pr(k \mid x)$$

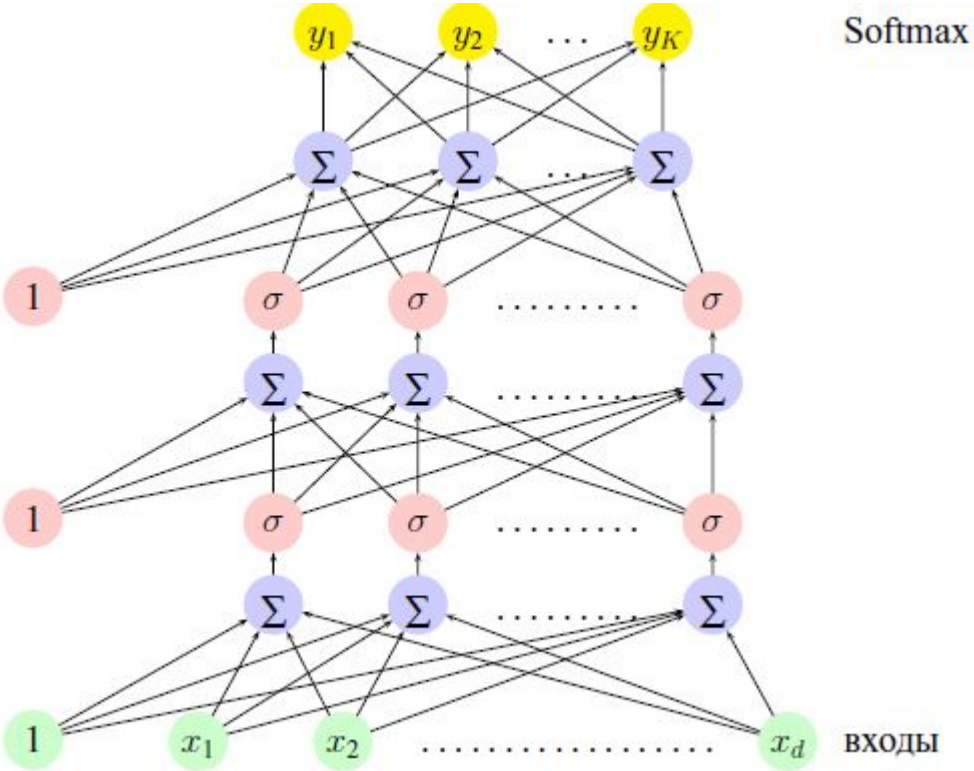
$$(k = 1, 2, \ldots, K)$$

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 σ(*z*).

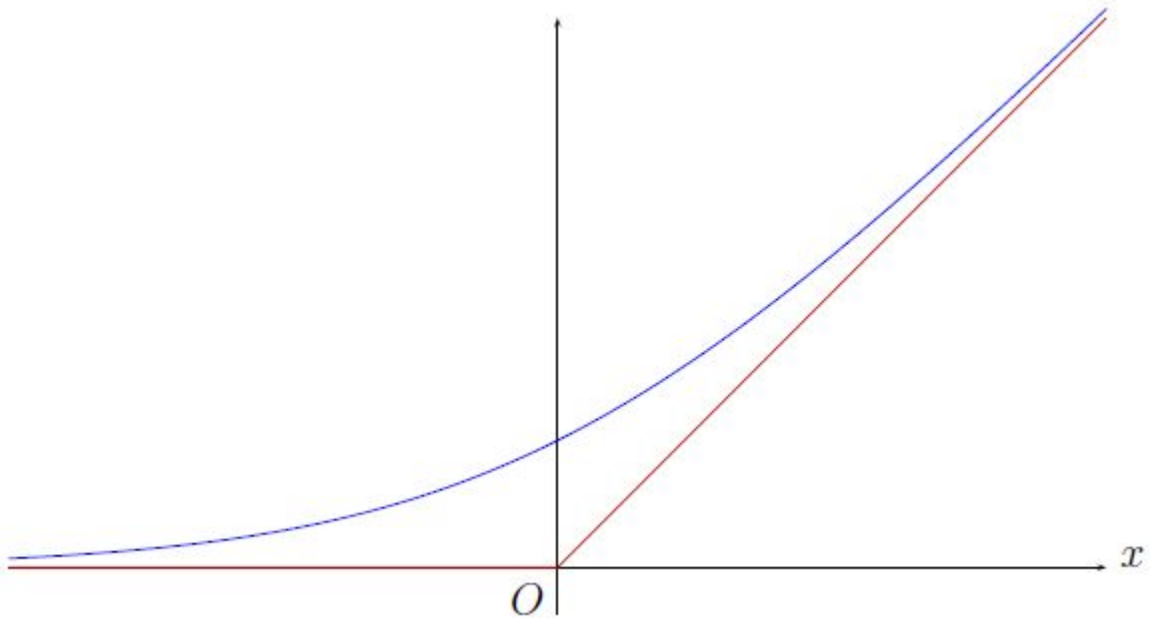
Иногда отдельно изображают суммирующие элементы и элементы, вычисляющие σ:



Кроме сигмоидальной используют и др/ функции, например, *положительную срезку линейной функции* (linear rectifier):

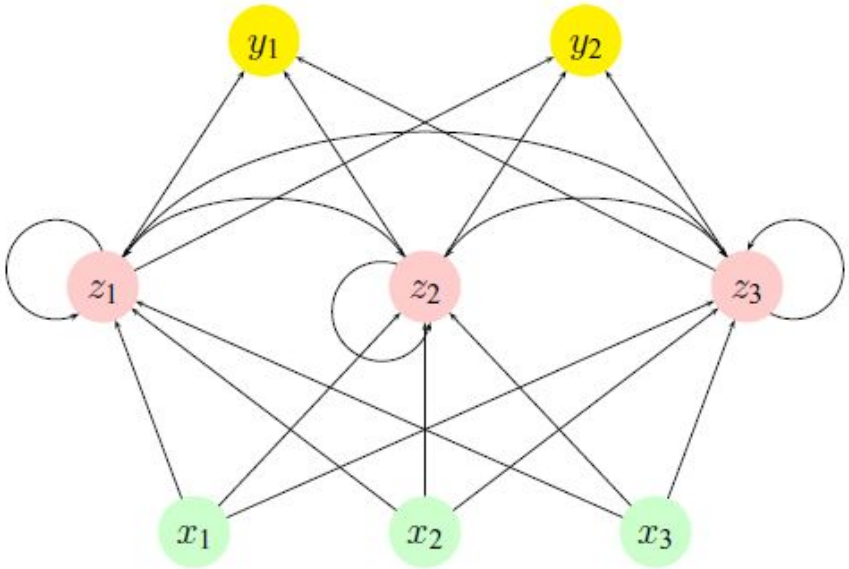
$$g(x_1, x_2, \ldots, x_q) = (\beta_0 + \beta_1 x_1 + \ldots + \beta_q x_q)_+ \qquad (x)_+ = \max\{0, x\}$$

или ее сглаженный вариант *softplus*: $g = \ln\left(1 + \exp(\beta_0 + \beta_1 x_1 + \ldots + \beta_q x_q)\right)$



Рекуррентные сети

В рекуррентных сетях присутствуют орциклы.
Рекуррентные нейронные сети используют, например, для предсказания временных рядов.



Кратко о глубоком обучении

(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.

Некоторые подходы в глубоком обучении

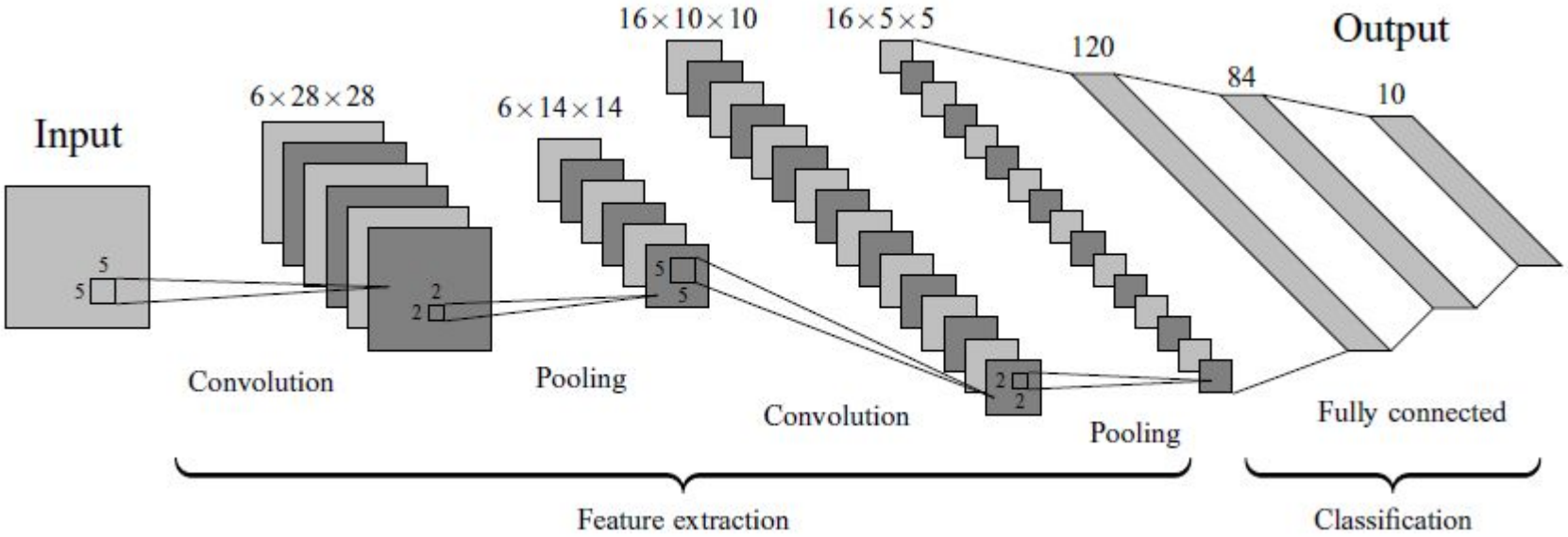
- Сверточные нейронные сети
- Автокодировщики (autoencoders) и стеки автокодировщиков
- Ограниченная машина Больцмана и глубокие сети доверия (deep belief networks)

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

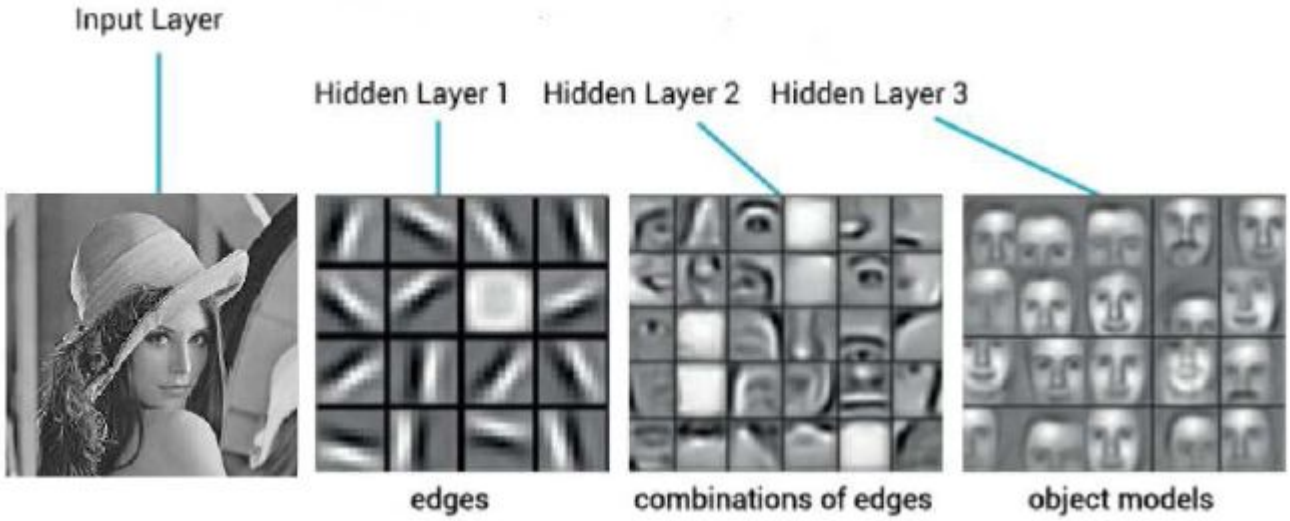
$$(I * K)_{pq} = \sum_{i=1}^{h} \sum_{j=1}^{w} I_{p+i-1,\, q+j-1} K_{ij}$$

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

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



- Сверточные слои (convolutional layers)
- «Выборочные» слои, или слои объединения (subsampling/pooling layers)
- Полносвязные слои (fully connected layers)


Input Layer
Hidden Layer 1
Hidden Layer 2
Hidden Layer 3
edges
combinations of edges
object models