

Введение в машинное обучение (machine learning)

Н.Ю. Золотых

www.uic.unn.ru/~zny

4 июля 2019

1. Что такое *машинное обучение* (*machine learning*)?

1. Что такое машинное обучение (*machine learning*)?

Идея обучающихся машин (learning machines) принадлежит А. Тьюрингу
(*A. Turing* Computing Machinery and Intelligence // *Mind*. 1950. V. 59. P. 433–460;
перепечатно: 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)

1. Что такое машинное обучение (*machine learning*)?

Идея обучающихся машин (learning machines) принадлежит А. Тьюрингу (*A. Turing* Computing Machinery and Intelligence // *Mind*. 1950. V. 59. P. 433–460; перепечатно: 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)

Машинное обучение — процесс, в результате которого машина (компьютер) способна показывать поведение, которое в нее не было *явно* заложено (запрограммировано).

A.L. Samuel Some Studies in Machine Learning Using the Game of Checkers
// *IBM Journal*. July 1959. P. 210–229.

1. Что такое машинное обучение (*machine learning*)?

Идея обучающихся машин (learning machines) принадлежит А. Тьюрингу (А. *Turing* Computing Machinery and Intelligence // Mind. 1950. V. 59. P. 433–460; перепечатно: 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)

Машинное обучение — процесс, в результате которого машина (компьютер) способна показывать поведение, которое в нее не было *явно* заложено (запрограммировано).

A.L. Samuel Some Studies in Machine Learning Using the Game of Checkers
// IBM Journal. July 1959. P. 210–229.

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 Machine Learning. McGraw-Hill, 1997.

Alan Mathison Turing (1912–1954)

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.

	PAGE
I.—Computing Machinery and Intelligence: A. M. TURING	433
II.—Subject and Predicate: P. T. GEACH	461
III.—Frege's <i>Sinn und Bedeutung</i> : P. D. WIENPAHL	483
IV.—The Theory of Sovereignty Restated: W. J. REES	495
V.—A Note on Verification: F. C. COPLESTON	522
Notes	529
VI.—Discussions:—	
Ostensive Definition and Empirical Certainty: A. PAP	530
Pragmatic Paradoxes: P. ALEXANDER	536
The Causal Theory of Perception: J. WATLING	539
"Fallacies in Moral Philosophy." A Reply to Mr. Baier: S. HAMPSHIRE	541
The Existence of God: T. MCPHERSON	545
Berkeley's <i>Philosophical Commentaries</i> : A. A. LUCE	551
A Note on Aristotle. Categories 6a 15: M. WARNOCK	552
VII.—Critical Notice:—	
<i>Moral Obligation</i> : Essays and Lectures by H. A. Prichard: C. D. BROAD	555
VIII.—New Books	567

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

NEW YORK: THOMAS NELSON & SONS

Price Four Shillings and Sixpence. All Rights Reserved.
Yearly Subscribers will receive MIND post free from the Publishers on payment (in advance) of Sixteen Shillings.

Entered as Second Class Matter, October 1st, 1948, at the Post Office at New York, N.Y. under the Act of March 3rd, 1933, and July 2nd, 1946.

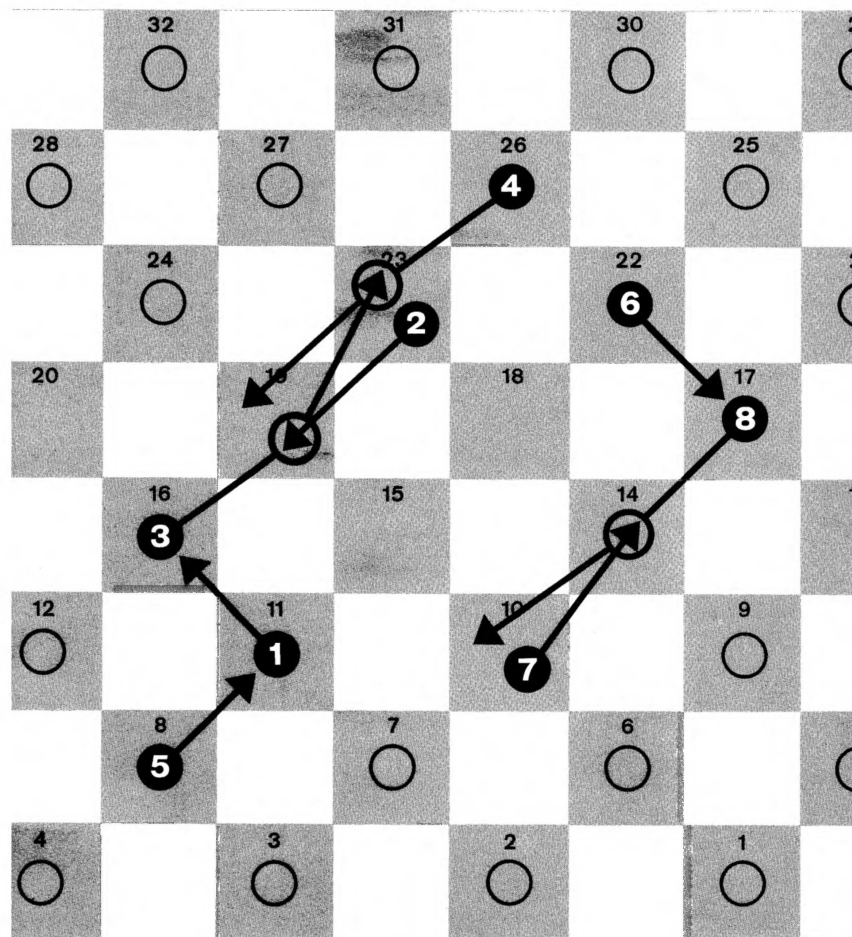
Printed in Great Britain



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.

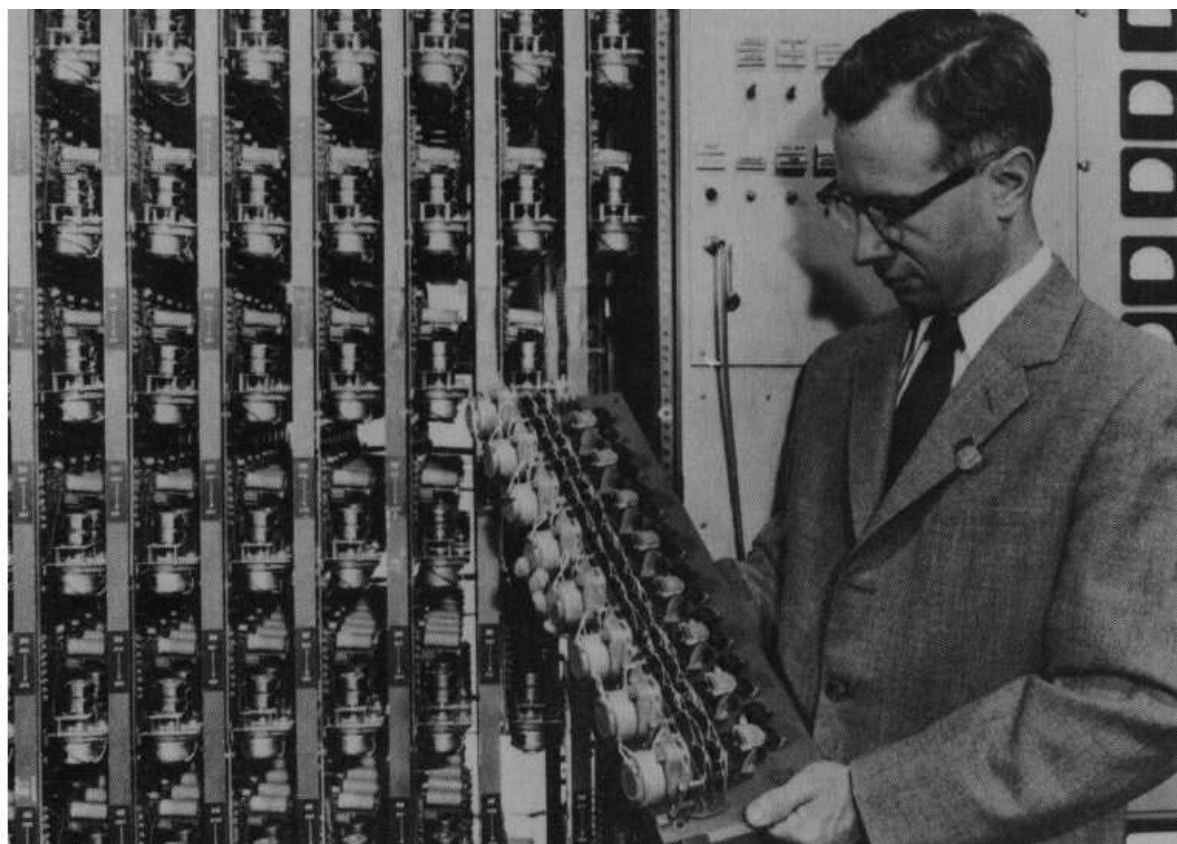
Some of these characteristics might well be enumerated. They are:

(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.



1.1. Machine Learning сегодня

- Нейронные сети — второе (или третье) рождение. Глубокое обучение (deep learning)
- Big Data

1.1. Machine Learning сегодня

- Нейронные сети — второе (или третье) рождение. Глубокое обучение (deep learning)
- Big Data

Некоторые последние достижения:

1.1. Machine Learning сегодня

- Нейронные сети — второе (или третье) рождение. Глубокое обучение (deep learning)
- Big Data

Некоторые последние достижения:

- Компьютерное зрение
Прорыв 2012: ImageNet ILSVRC-2012 (около 1 млн. изображений, 1000 классов).
Ошибку удалось понизить с 26% до 15% (сейчас меньше 2%) – ALEXNet –
A. Krizhevsky, I. Sutskever, G. E. Hinton
Для многих задач компьютерного зрения точность не хуже, чем у человека.

1.1. Machine Learning сегодня

- Нейронные сети — второе (или третье) рождение. Глубокое обучение (deep learning)
- Big Data

Некоторые последние достижения:

- Компьютерное зрение
Прорыв 2012: ImageNet ILSVRC-2012 (около 1 млн. изображений, 1000 классов).
Ошибку удалось понизить с 26% до 15% (сейчас меньше 2%) – ALEXNet –
A. Krizhevsky, I. Sutskever, G. E. Hinton
Для многих задач компьютерного зрения точность не хуже, чем у человека.
- Беспилотные автомобили

1.1. Machine Learning сегодня

- Нейронные сети — второе (или третье) рождение. Глубокое обучение (deep learning)
- Big Data

Некоторые последние достижения:

- Компьютерное зрение
Прорыв 2012: ImageNet ILSVRC-2012 (около 1 млн. изображений, 1000 классов).
Ошибку удалось понизить с 26% до 15% (сейчас меньше 2%) – ALEXNet –
A. Krizhevsky, I. Sutskever, G. E. Hinton
Для многих задач компьютерного зрения точность не хуже, чем у человека.
- Беспилотные автомобили
- AlphaGo, AlphaZero, ...

1.1. Machine Learning сегодня

- Нейронные сети — второе (или третье) рождение. Глубокое обучение (deep learning)
- Big Data

Некоторые последние достижения:

- Компьютерное зрение
Прорыв 2012: ImageNet ILSVRC-2012 (около 1 млн. изображений, 1000 классов).
Ошибку удалось понизить с 26% до 15% (сейчас меньше 2%) – ALEXNet –
A. Krizhevsky, I. Sutskever, G. E. Hinton
Для многих задач компьютерного зрения точность не хуже, чем у человека.
- Беспилотные автомобили
- AlphaGo, AlphaZero, ...
- Умные помощники


1.1. Machine Learning сегодня

- Нейронные сети — второе (или третье) рождение. Глубокое обучение (deep learning)
- Big Data

Некоторые последние достижения:

- Компьютерное зрение
Прорыв 2012: ImageNet ILSVRC-2012 (около 1 млн. изображений, 1000 классов).
Ошибку удалось понизить с 26% до 15% (сейчас меньше 2%) – ALEXNet –
A. Krizhevsky, I. Sutskever, G. E. Hinton
Для многих задач компьютерного зрения точность не хуже, чем у человека.
- Беспилотные автомобили
- AlphaGo, AlphaZero, ...
- Умные помощники
- ...

ImageNet

			
mite	container ship	motor scooter	leopard
<div> <div></div> <div>mite</div> <div>black widow</div> <div>cockroach</div> <div>tick</div> <div>starfish</div> </div>	<div> <div></div> <div>container ship</div> <div>lifeboat</div> <div>amphibian</div> <div>fireboat</div> <div>drilling platform</div> </div>	<div> <div></div> <div>motor scooter</div> <div>go-kart</div> <div>moped</div> <div>bumper car</div> <div>golfcart</div> </div>	<div> <div></div> <div>leopard</div> <div>jaguar</div> <div>cheetah</div> <div>snow leopard</div> <div>Egyptian cat</div> </div>
			
grille	mushroom	cherry	Madagascar cat
<div> <div></div> <div>convertible</div> <div>grille</div> <div>pickup</div> <div>beach wagon</div> <div>fire engine</div> </div>	<div> <div></div> <div>agaric</div> <div>mushroom</div> <div>jelly fungus</div> <div>gill fungus</div> <div>dead-man's-fingers</div> </div>	<div> <div></div> <div>dalmatian</div> <div>grape</div> <div>elderberry</div> <div>ffordshire bullterrier</div> <div>currant</div> </div>	<div> <div></div> <div>squirrel monkey</div> <div>spider monkey</div> <div>titi</div> <div>indri</div> <div>howler monkey</div> </div>

1.2. 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]

1.2. 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.

1.2. 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:

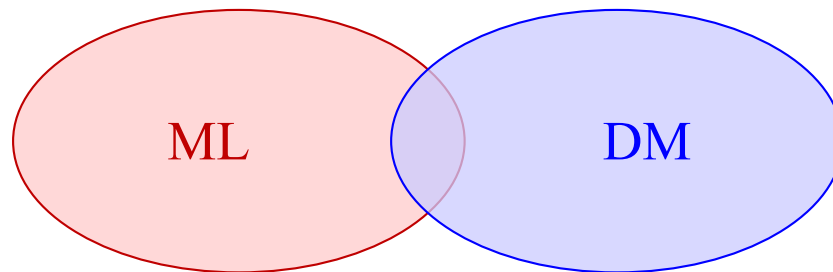
- ML — чтобы обучить машину;
- DM — чтобы обучить человека.

Поэтому

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

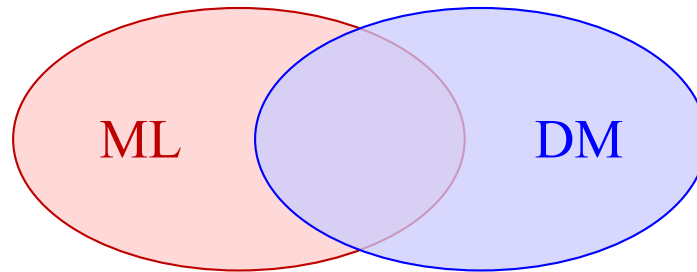
Machine Learning vs Data Mining

Содержательные
постановки задач

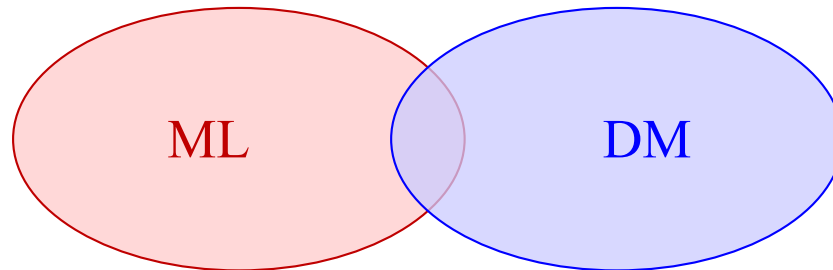


Machine Learning vs Data Mining

Математические
постановки задач

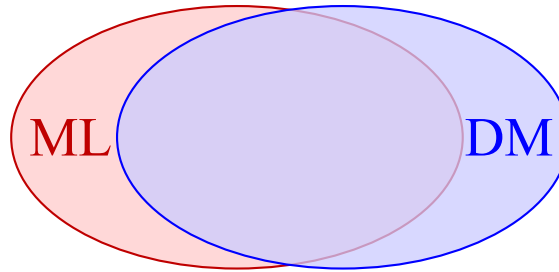


Содержательные
постановки задач

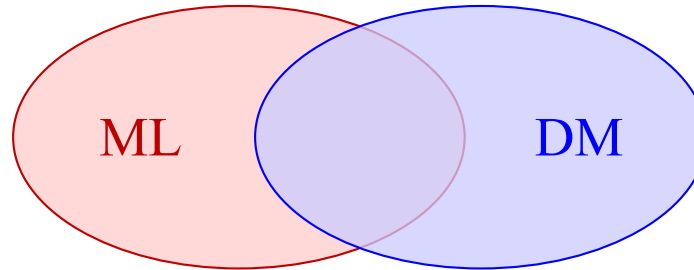


Machine Learning vs Data Mining

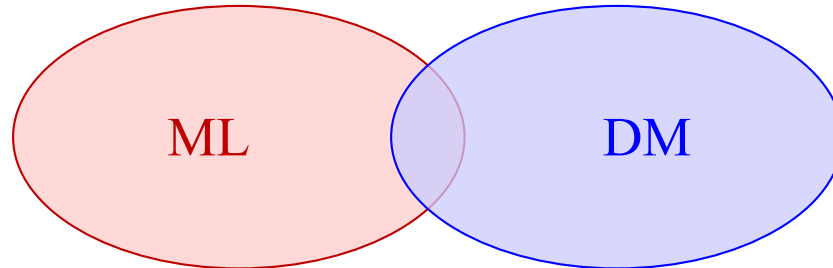
Методы



Математические
постановки задач



Содержательные
постановки задач



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»

1.3. Ресурсы

- * Wiki-портал <http://www.machinelearning.ru>
- ** *Воронцов К. В.* Машинное обучение (курс лекций)
см. <http://www.machinelearning.ru>,
видео-лекции http://shad.yandex.ru/lectures/machine_learning.xml
- Мой курс: <http://www.uic.unn.ru/~zny/ml> (презентации лекций, лабораторные работы, описание системы R, ссылки, ML для «чайников» и др.)
- *Орельен Ж.* Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow. Концепции, инструменты и техники для создания интеллектуальных систем. М.: Вильямс, 2018
- *Мюллер А., Гвидо С.* Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными. М.: Вильямс, 2017
- *Джеймс Г., Уиттон Д., Хасты Т., Тибширани Р.* Введение в статистическое обучение с примерами на языке R. М.: ДМК Пресс, 2016
- *Ng A.* Machine Learning Course (video, lecture notes, presentations, labs)
<http://ml-class.org>
- ** *Hastie T., Tibshirani R., Friedman J.* The elements of statistical learning: Data Mining, Inference, and Prediction. 2nd Edition. Springer, 2009 <http://www-stat.stanford.edu/tibs/ElemStatLearn/>
- ...

1.4. Software

- Python Scikit-Learn scikit-learn.org, Pandas pandas.pydata.org,
Jupyter Notebook jupyter.org и др.
(все есть, например, в дистрибутиве Anaconda www.anaconda.com)
- Google Colab colab.research.google.com
- Система для статистических вычислений R r-project.org
- MATLAB Statistics and Machine Learning Toolbox + Neural Network Toolbox
mathworks.com
- Intel DAAL: Intel Data Analytics Acceleration Library
software.intel.com/en-us/daal
- 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
- ...

Software. Глубокое обучение

- Keras
- TensorFlow
- Torch
- Caffe
- Theano
- ...

Соревнования и данные для экспериментов

- *Данные для экспериментов:* UCI Machine Learning Repository
archive.ics.uci.edu/ml/
- Kaggle: The Home of Data Science — платформа для соревнований, датасеты, туториалы www.kaggle.com
- ...

1.5. Классификация задач машинного обучения

1.5. Классификация задач машинного обучения

- Обучение с учителем (supervised learning):
 - классификация
 - восстановление регрессии
 - ...

1.5. Классификация задач машинного обучения

- Обучение с учителем (supervised learning):
 - классификация
 - восстановление регрессии
 - ...
- Обучение без учителя (unsupervised learning):
 - кластеризация
 - понижение размерности
 - ...

1.5. Классификация задач машинного обучения

- Обучение с учителем (supervised learning):
 - классификация
 - восстановление регрессии
 - ...
- Обучение без учителя (unsupervised learning):
 - кластеризация
 - понижение размерности
 - ...
- Обучение с подкреплением (reinforcement learning)
- Активное обучение
- ...

2. Обучение с учителем

Множество \mathcal{X} — объекты, наблюдения, примеры, ситуации, входы (samples)
— пространство признаков

2. Обучение с учителем

Множество \mathcal{X} — объекты, наблюдения, примеры, ситуации, входы (samples)
— пространство признаков

Множество \mathcal{Y} — ответы, отклики, «метки», выходы (responses)

2. Обучение с учителем

Множество \mathcal{X} — объекты, наблюдения, примеры, ситуации, входы (samples)
— пространство признаков

Множество \mathcal{Y} — ответы, отклики, «метки», выходы (responses)

Имеется некоторая зависимость (детерминированная или вероятностная), позволяющая по $x \in \mathcal{X}$ предсказать $y \in \mathcal{Y}$.

т. е. если зависимость детерминированная, существует функция $f^* : \mathcal{X} \rightarrow \mathcal{Y}$.

2. Обучение с учителем

Множество \mathcal{X} — *объекты, наблюдения, примеры, ситуации, входы (samples)*
— *пространство признаков*

Множество \mathcal{Y} — *ответы, отклики, «метки», выходы (responses)*

Имеется некоторая зависимость (детерминированная или вероятностная), позволяющая по $x \in \mathcal{X}$ предсказать $y \in \mathcal{Y}$.

т. е. если зависимость детерминированная, существует функция $f^* : \mathcal{X} \rightarrow \mathcal{Y}$.

Зависимость известна только на объектах из *обучающей выборки*:

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(N)}, y^{(N)})\}$$

2. Обучение с учителем

Множество \mathcal{X} — *объекты, наблюдения, примеры, ситуации, входы (samples)*
— *пространство признаков*

Множество \mathcal{Y} — *ответы, отклики, «метки», выходы (responses)*

Имеется некоторая зависимость (детерминированная или вероятностная), позволяющая по $x \in \mathcal{X}$ предсказать $y \in \mathcal{Y}$.

т. е. если зависимость детерминированная, существует функция $f^* : \mathcal{X} \rightarrow \mathcal{Y}$.

Зависимость известна только на объектах из *обучающей выборки*:

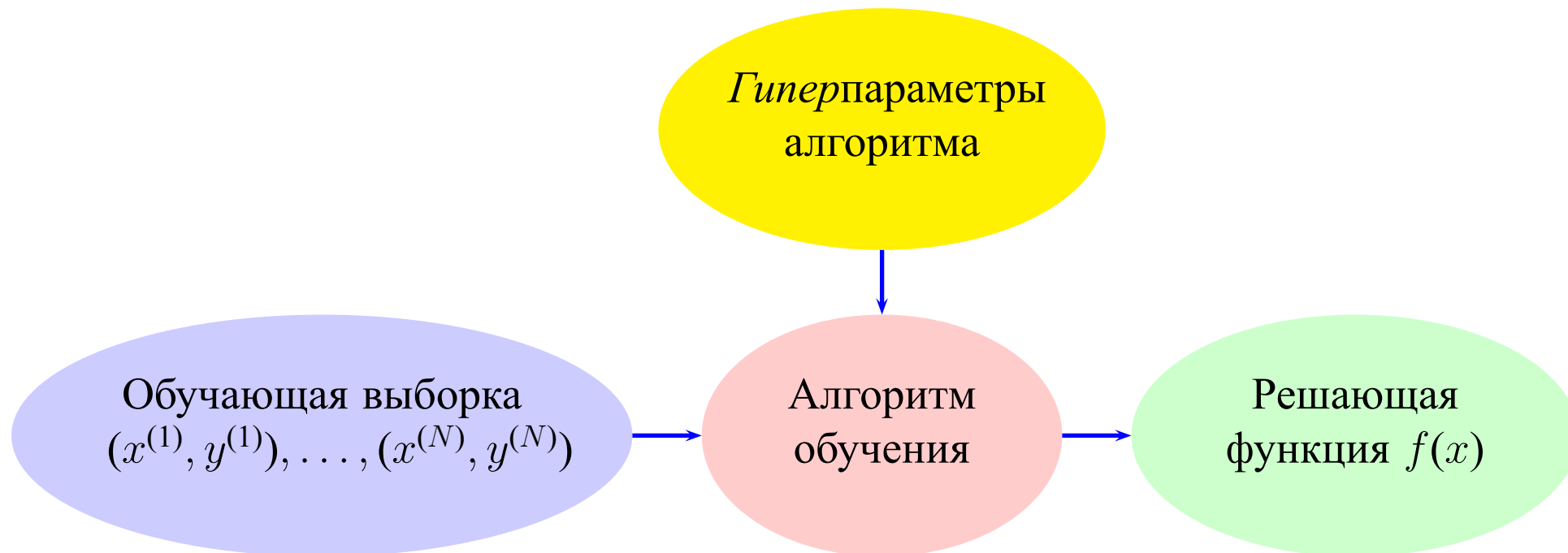
$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(N)}, y^{(N)})\}$$

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

$$y = f(x) \approx f^*(x).$$

- 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
- ...

2.1. Схема обучения с учителем



2.2. Признаковые описания

Вход:

$$x = (x_1, x_2, \dots, x_d) \in \mathcal{X} = Q_1 \times Q_2 \times \dots \times Q_d,$$

x_j — j -й признак (свойство, атрибут, предикативная переменная, *feature*) объекта x .

2.2. Признаковые описания

Вход:

$$x = (x_1, x_2, \dots, x_d) \in \mathcal{X} = Q_1 \times Q_2 \times \dots \times Q_d,$$

x_j — j -й признак (свойство, атрибут, предикативная переменная, *feature*) объекта x .

- Если Q_j конечно, то j -й признак — *номинальный* (категориальный или *фактор*).

2.2. Признаковые описания

Вход:

$$x = (x_1, x_2, \dots, x_d) \in \mathcal{X} = Q_1 \times Q_2 \times \dots \times Q_d,$$

x_j — j -й признак (свойство, атрибут, предикативная переменная, *feature*) объекта x .

- Если Q_j конечно, то j -й признак — *номинальный* (категориальный или *фактор*).

Например, $Q_j = \{\text{Alfa Romeo, Audi, BMW, } \dots, \text{Volkswagen}\}$

2.2. Признаковые описания

Вход:

$$x = (x_1, x_2, \dots, x_d) \in \mathcal{X} = Q_1 \times Q_2 \times \dots \times Q_d,$$

x_j — j -й признак (свойство, атрибут, предикативная переменная, *feature*) объекта x .

- Если Q_j конечно, то j -й признак — *номинальный* (категориальный или *фактор*).

Например, $Q_j = \{\text{Alfa Romeo, Audi, BMW, } \dots, \text{Volkswagen}\}$

Если $|Q_j| = 2$, то признак *бинарный* и можно считать, например, $Q_j = \{0, 1\}$.

2.2. Признаковые описания

Вход:

$$x = (x_1, x_2, \dots, x_d) \in \mathcal{X} = Q_1 \times Q_2 \times \dots \times Q_d,$$

x_j — j -й признак (свойство, атрибут, предикативная переменная, *feature*) объекта x .

- Если Q_j конечно, то j -й признак — *номинальный* (категориальный или *фактор*).
Например, $Q_j = \{\text{Alfa Romeo, Audi, BMW, } \dots, \text{Volkswagen}\}$
Если $|Q_j| = 2$, то признак *бинарный* и можно считать, например, $Q_j = \{0, 1\}$.
- Если Q_j конечно и упорядочено, то признак *порядковый*.

2.2. Признаковые описания

Вход:

$$x = (x_1, x_2, \dots, x_d) \in \mathcal{X} = Q_1 \times Q_2 \times \dots \times Q_d,$$

x_j — j -й признак (свойство, атрибут, предикативная переменная, *feature*) объекта x .

- Если Q_j конечно, то j -й признак — *номинальный* (категориальный или *фактор*).

Например, $Q_j = \{\text{Alfa Romeo, Audi, BMW, } \dots, \text{Volkswagen}\}$

Если $|Q_j| = 2$, то признак *бинарный* и можно считать, например, $Q_j = \{0, 1\}$.

- Если Q_j конечно и упорядочено, то признак *порядковый*.

Например, $Q_j = \{\text{Beginner, Elementary, Intermediate, Advanced, Proficiency}\}$

2.2. Признаковые описания

Вход:

$$x = (x_1, x_2, \dots, x_d) \in \mathcal{X} = Q_1 \times Q_2 \times \dots \times Q_d,$$

x_j — j -й признак (свойство, атрибут, предикативная переменная, *feature*) объекта x .

- Если Q_j конечно, то j -й признак — *номинальный* (категориальный или *фактор*).
Например, $Q_j = \{\text{Alfa Romeo, Audi, BMW, } \dots, \text{Volkswagen}\}$
Если $|Q_j| = 2$, то признак *бинарный* и можно считать, например, $Q_j = \{0, 1\}$.
- Если Q_j конечно и упорядочено, то признак *порядковый*.
Например, $Q_j = \{\text{Beginner, Elementary, Intermediate, Advanced, Proficiency}\}$
- Если $Q_j = \mathbf{R}$, то признак *количественный*.

2.2. Признаковые описания

Вход:

$$x = (x_1, x_2, \dots, x_d) \in \mathcal{X} = Q_1 \times Q_2 \times \dots \times Q_d,$$

x_j — j -й признак (свойство, атрибут, предикативная переменная, *feature*) объекта x .

- Если Q_j конечно, то j -й признак — *номинальный* (категориальный или *фактор*).
Например, $Q_j = \{\text{Alfa Romeo, Audi, BMW, } \dots, \text{Volkswagen}\}$
Если $|Q_j| = 2$, то признак *бинарный* и можно считать, например, $Q_j = \{0, 1\}$.
- Если Q_j конечно и упорядочено, то признак *порядковый*.
Например, $Q_j = \{\text{Beginner, Elementary, Intermediate, Advanced, Proficiency}\}$
- Если $Q_j = \mathbf{R}$, то признак *количественный*.

Выход: $y \in \mathcal{Y}$

- $\mathcal{Y} = \mathbf{R}$ — задача восстановления регрессии
- $\mathcal{Y} = \{1, 2, \dots, K\}$ — задача классификации. Номер класса $k \in \mathcal{Y}$

Признаковые описания объектов обучающей выборки обычно записывают в таблицу:

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

i -я строка этой таблицы соответствует i -му объекту обучающей выборки

j -й столбец — j -му признаку

Пример 1. Медицинская диагностика

Пример 1. Медицинская диагностика

Имеются данные о 114 лицах с заболеванием щитовидной железы.

У 61 — повышенный уровень свободного гормона Т4,

у 53 — уровень гормона в норме.

Пример 1. Медицинская диагностика

Имеются данные о 114 лицах с заболеванием щитовидной железы.

У 61 — повышенный уровень свободного гормона Т4,

у 53 — уровень гормона в норме.

Для каждого пациента известны следующие показатели:

Пример 1. Медицинская диагностика

Имеются данные о 114 лицах с заболеванием щитовидной железы.

У 61 — повышенный уровень свободного гормона Т4,

у 53 — уровень гормона в норме.

Для каждого пациента известны следующие показатели:

- $x_1 = \text{heart}$ — частота сердечных сокращений (пульс),

Пример 1. Медицинская диагностика

Имеются данные о 114 лицах с заболеванием щитовидной железы.

У 61 — повышенный уровень свободного гормона Т4,

у 53 — уровень гормона в норме.

Для каждого пациента известны следующие показатели:

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

Пример 1. Медицинская диагностика

Имеются данные о 114 лицах с заболеванием щитовидной железы.

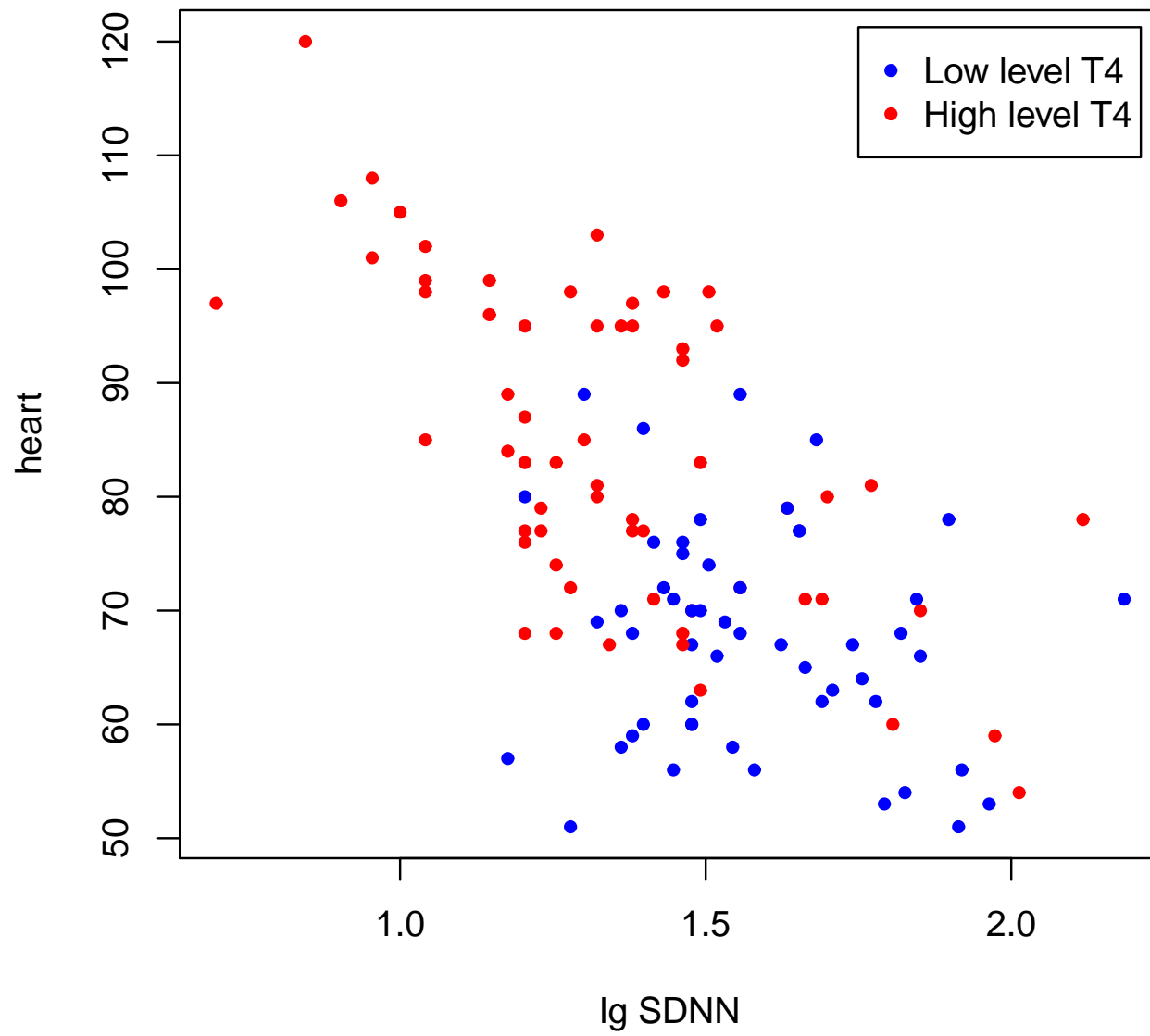
У 61 — повышенный уровень свободного гормона Т4,

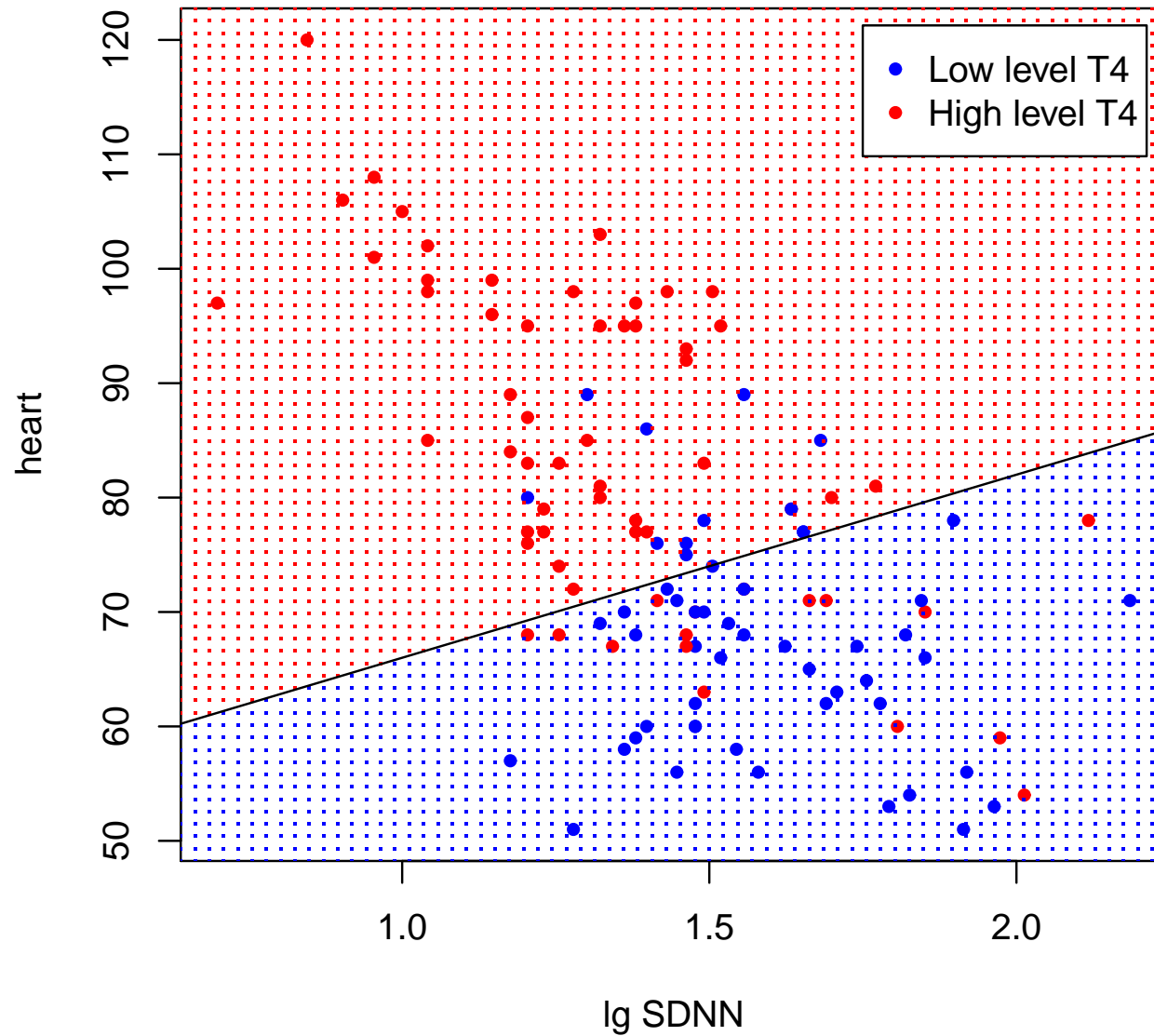
у 53 — уровень гормона в норме.

Для каждого пациента известны следующие показатели:

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

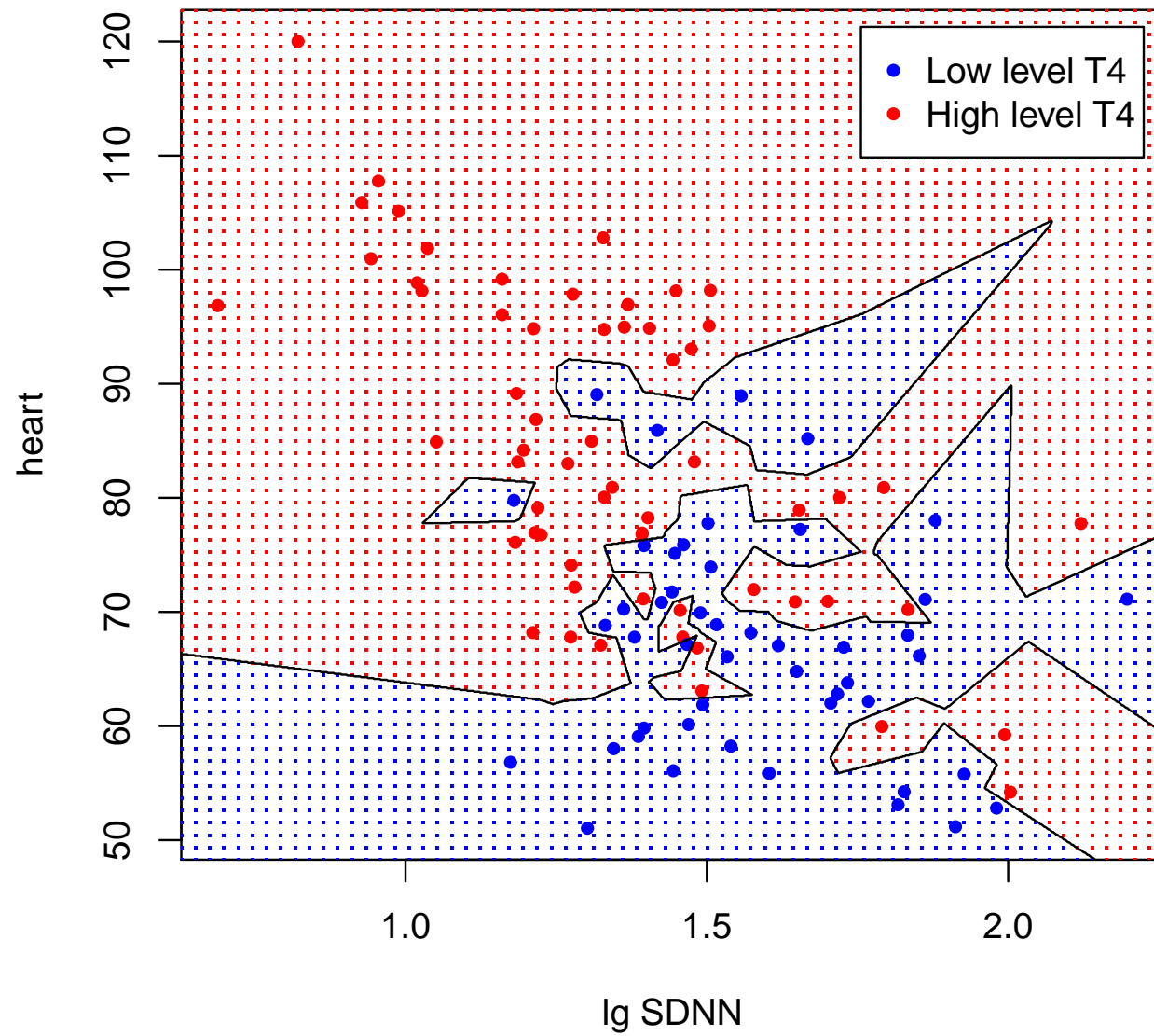
Можно ли научиться предсказывать (допуская небольшие ошибки) уровень свободного Т4 по heart и SDNN?





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

Ошибка на обучающей выборке — 23 %. Можно ли ее сделать меньше?



Метод ближайшего соседа (с масштабированием)

Ошибка на обучающей выборке — 0 %.

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.*

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.

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.

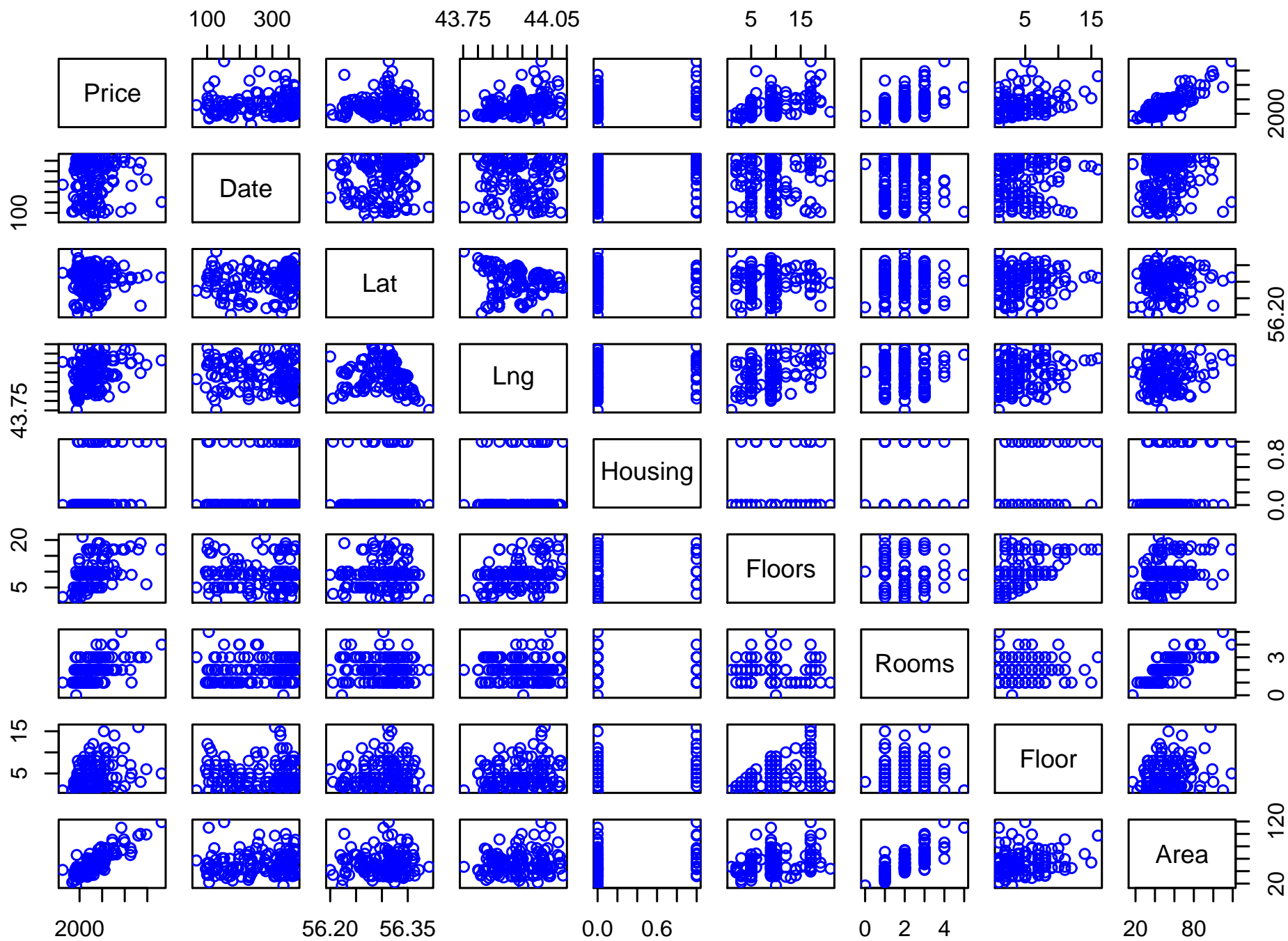
Переобучение — решающее правило хорошо решает задачу на обучающей выборке, но имеет плохую обобщающую способность.

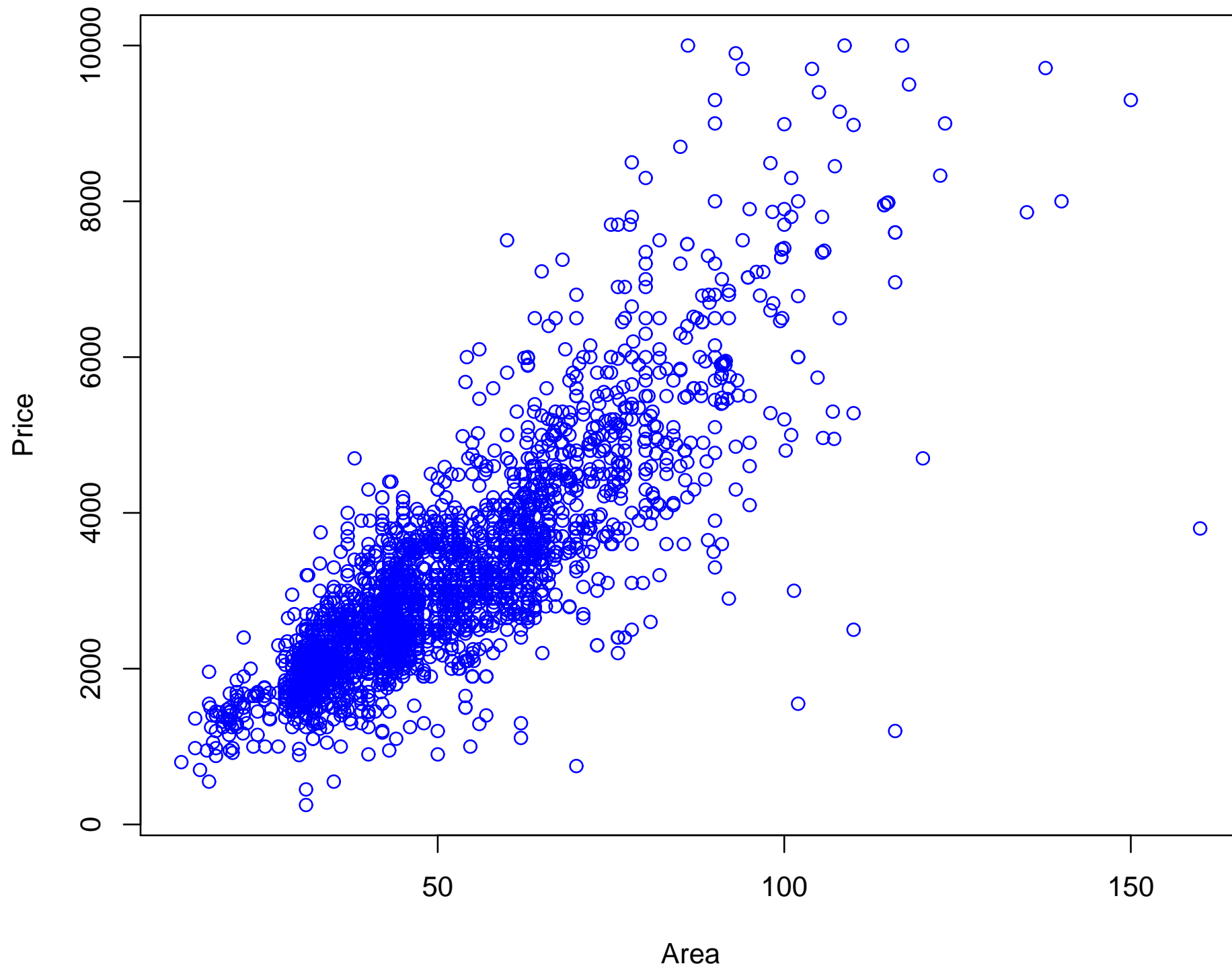
Пример 2. Оценка стоимости квартиры

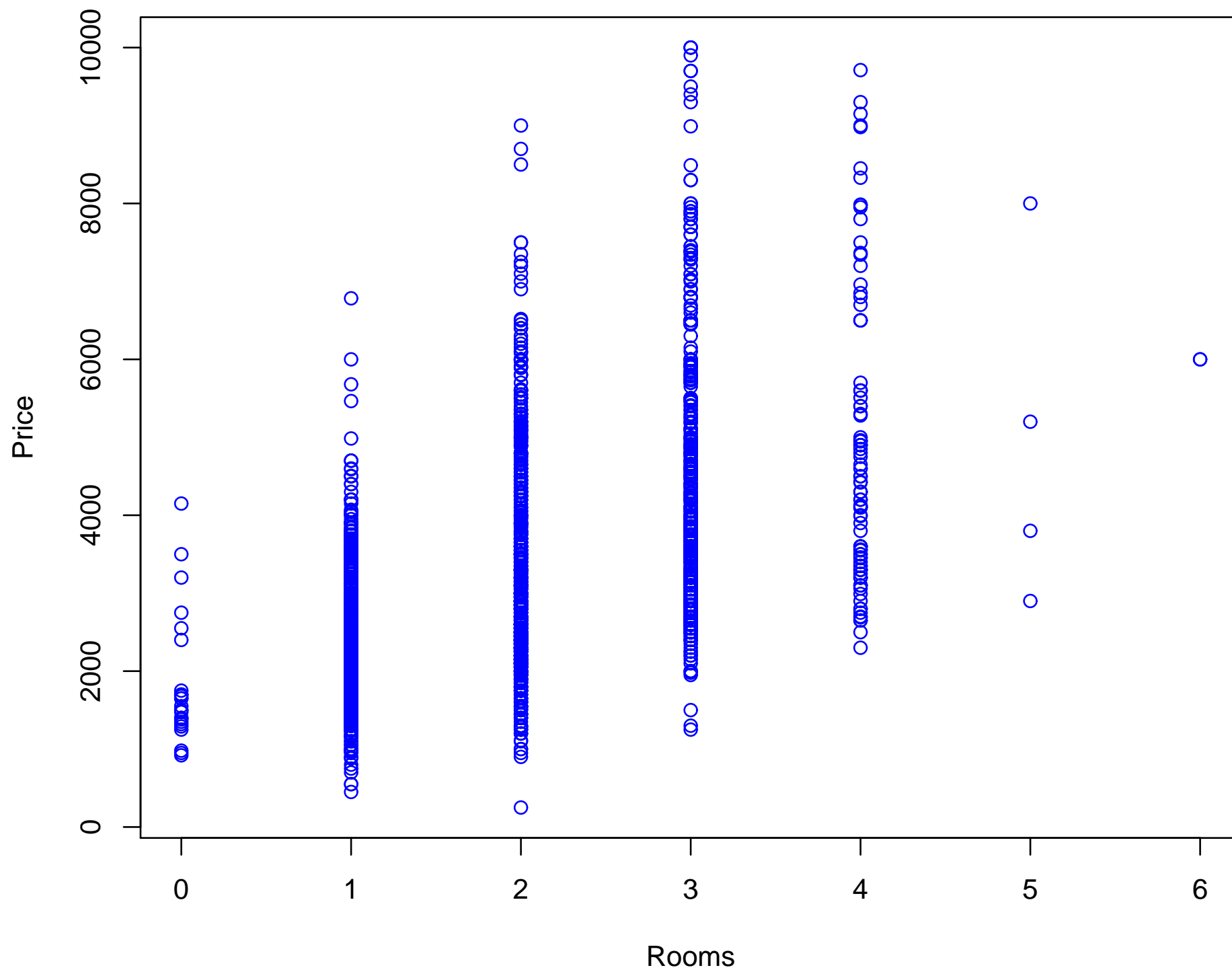
Пример 2. Оценка стоимости квартиры

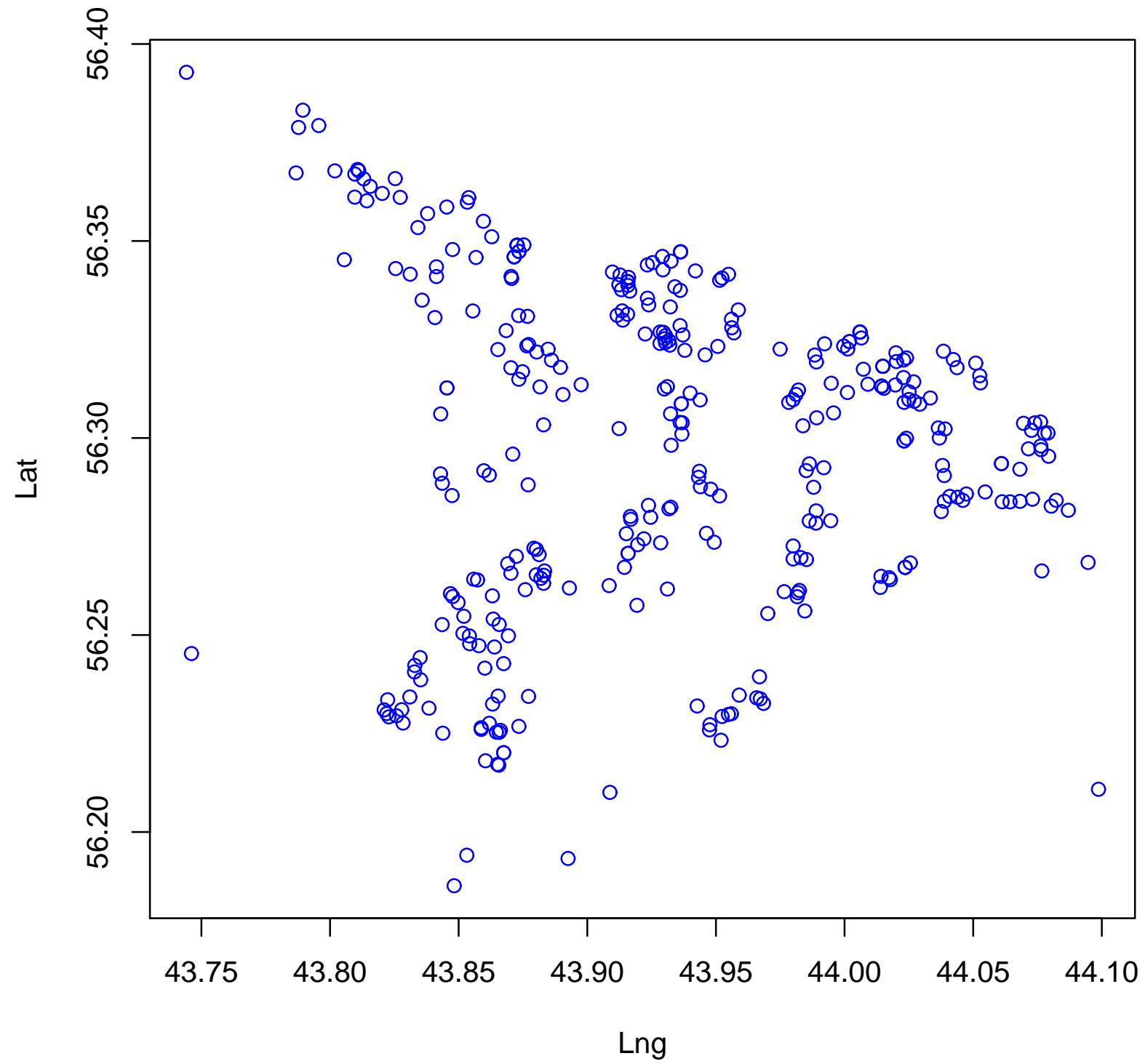
Price	цена квартиры (тыс. руб.)
Date	№ дня, в который квартира выставлена на продажу
Lat	географическая широта объекта недвижимости
Lng	географическая долгота объекта недвижимости
Housing	тип недвижимости (0 — вторичное жилье, 1 — новостройка)
Floors	количество этажей в доме
House	тип дома (Block — блочный, Brick — кирпичный, Monolithic — монолитный, Panel — панельный, Wooden — деревянный)
Rooms	количество комнат (0 — квартира-студия)
Floor	№ этажа
Area	площадь квартиры (м ²)

Выборка содержит 72379 записей.









3. Оценка качества решения

Алгоритм обучения старается подобрать такую функцию f , что $f(x^{(i)}) \approx y^{(i)}$ ($i = 1, 2, \dots, N$).

Близость $f(x^{(i)})$ и $y^{(i)}$ обычно определяется *функцией потерь (штрафа)* $L(f(x^{(i)}), y^{(i)})$

Например, для задачи восстановления регрессии:

$$L(y', y) = (y' - y)^2$$

Для задачи классификации:

$$L(y', y) = I(y' \neq y) \equiv \begin{cases} 0, & y' = y, \\ 1, & y' \neq y. \end{cases}$$

Минимизируем эмпирический риск (ошибку на обучающей выборке):

$$R(f) = \frac{1}{N} \sum_{i=1}^N L(f(x^{(i)}), y^{(i)}) \rightarrow \min$$

Но целью является создание решающей функции, делающей хорошие предсказания на новых данных. Чрезмерная минимизация ошибки на обучающей выборке может привести к плохим результатам на новых данных (*переобучение*).

3.1. Ошибка на тестовой выборке

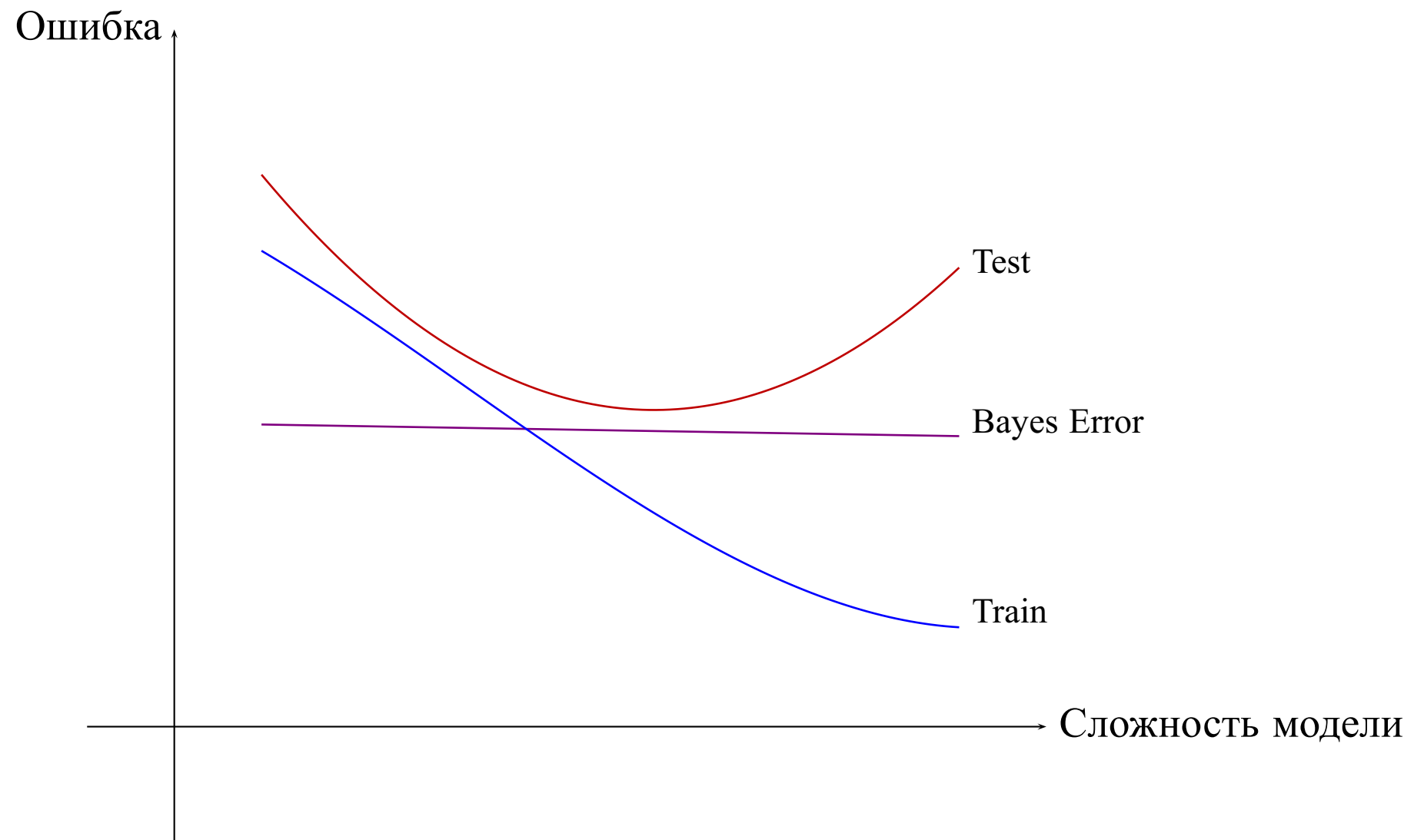
Если данных много, то можно выделить тестовую выборку, независимую от обучающей выборки:

$$(x_{\text{test}}^{(1)}, y_{\text{test}}^{(1)}), (x_{\text{test}}^{(2)}, y_{\text{test}}^{(2)}), \dots, (x_{\text{test}}^{(N_{\text{test}})}, y_{\text{test}}^{(N_{\text{test}})})$$

Несмещенная оценка качества построенной функции f :

$$\hat{R}(f) = \sum_{i=1}^N L \left(f(x_{\text{test}}^{(i)}), y_{\text{test}}^{(i)} \right)$$





3.2. Перекрестный контроль

Если данных мало, то можно использовать *перекрестный*, или *скользящий*, контроль (cross-validation)

Случайным образом разобьем исходную выборку на M непересекающихся примерно равных по размеру частей:



Случайным образом разобьем исходную выборку на M непересекающихся примерно равных по размеру частей:



Последовательно каждую из этих частей рассмотрим в качестве тестовой выборки, а объединение остальных частей — в качестве обучающей выборки:

Случайным образом разобьем исходную выборку на M непересекающихся примерно равных по размеру частей:



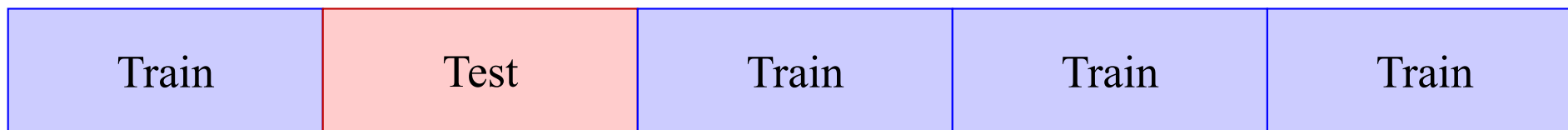
Последовательно каждую из этих частей рассмотрим в качестве тестовой выборки, а объединение остальных частей — в качестве обучающей выборки:



Случайным образом разобьем исходную выборку на M непересекающихся примерно равных по размеру частей:



Последовательно каждую из этих частей рассмотрим в качестве тестовой выборки, а объединение остальных частей — в качестве обучающей выборки:



Случайным образом разобьем исходную выборку на M непересекающихся примерно равных по размеру частей:



Последовательно каждую из этих частей рассмотрим в качестве тестовой выборки, а объединение остальных частей — в качестве обучающей выборки:



Случайным образом разобьем исходную выборку на M непересекающихся примерно равных по размеру частей:



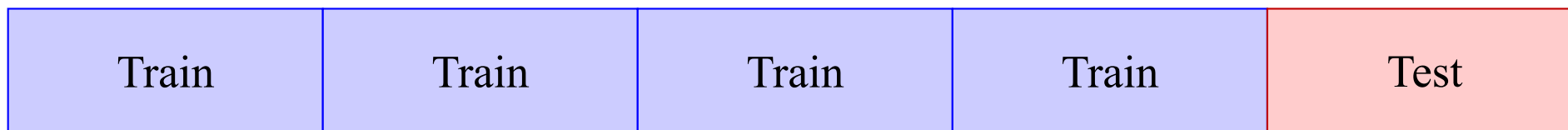
Последовательно каждую из этих частей рассмотрим в качестве тестовой выборки, а объединение остальных частей — в качестве обучающей выборки:



Случайным образом разобьем исходную выборку на M непересекающихся примерно равных по размеру частей:



Последовательно каждую из этих частей рассмотрим в качестве тестовой выборки, а объединение остальных частей — в качестве обучающей выборки:



Случайным образом разобьем исходную выборку на M непересекающихся примерно равных по размеру частей:



Последовательно каждую из этих частей рассмотрим в качестве тестовой выборки, а объединение остальных частей — в качестве обучающей выборки:



Таким образом построим M моделей и M оценок для ошибки предсказания:

$$f_1, f_2, \dots, f_M$$
$$\hat{R}_1, \hat{R}_2, \dots, \hat{R}_M.$$

Случайным образом разобьем исходную выборку на M непересекающихся примерно равных по размеру частей:



Последовательно каждую из этих частей рассмотрим в качестве тестовой выборки, а объединение остальных частей — в качестве обучающей выборки:



Таким образом построим M моделей и M оценок для ошибки предсказания:

$$f_1, f_2, \dots, f_M$$
$$\hat{R}_1, \hat{R}_2, \dots, \hat{R}_M.$$

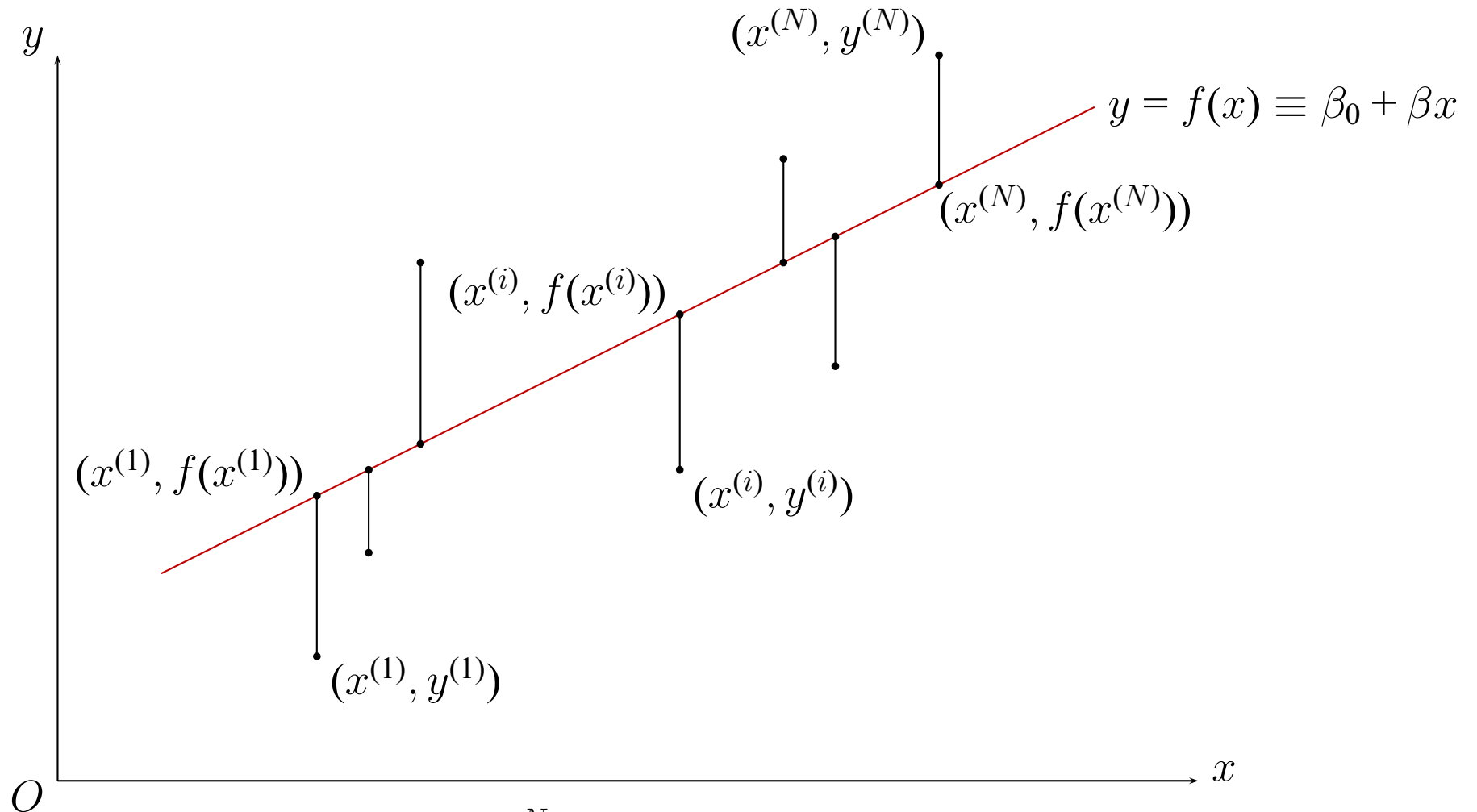
Функцию f построим по всем данным. В качестве оценки ее ошибки возьмем среднее:

$$\hat{R}(f) = \frac{1}{M} \sum_{m=1}^M \hat{R}_m.$$

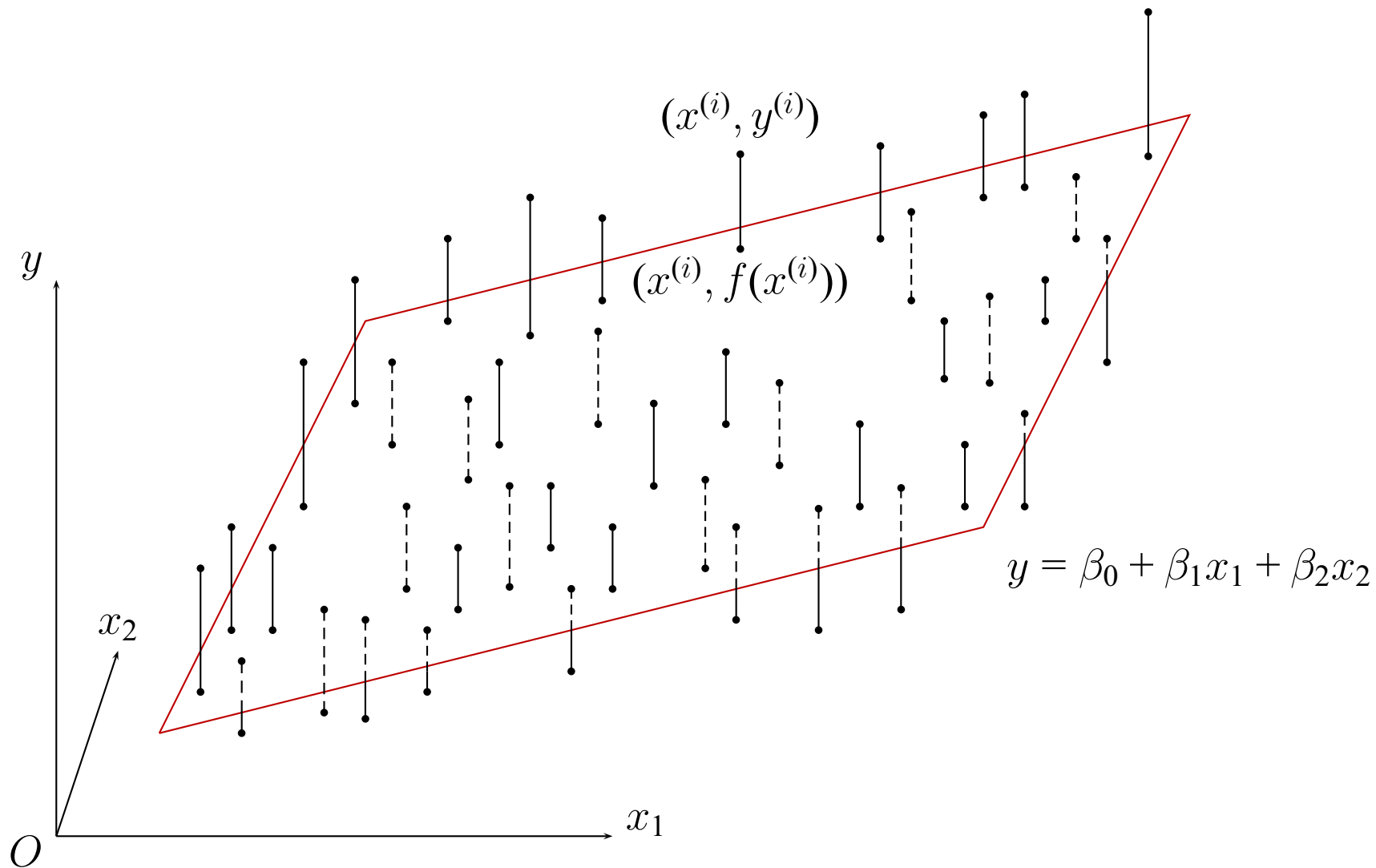
4. Некоторые методы обучения с учителем

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

4.1. Метод наименьших квадратов (линейная регрессия)

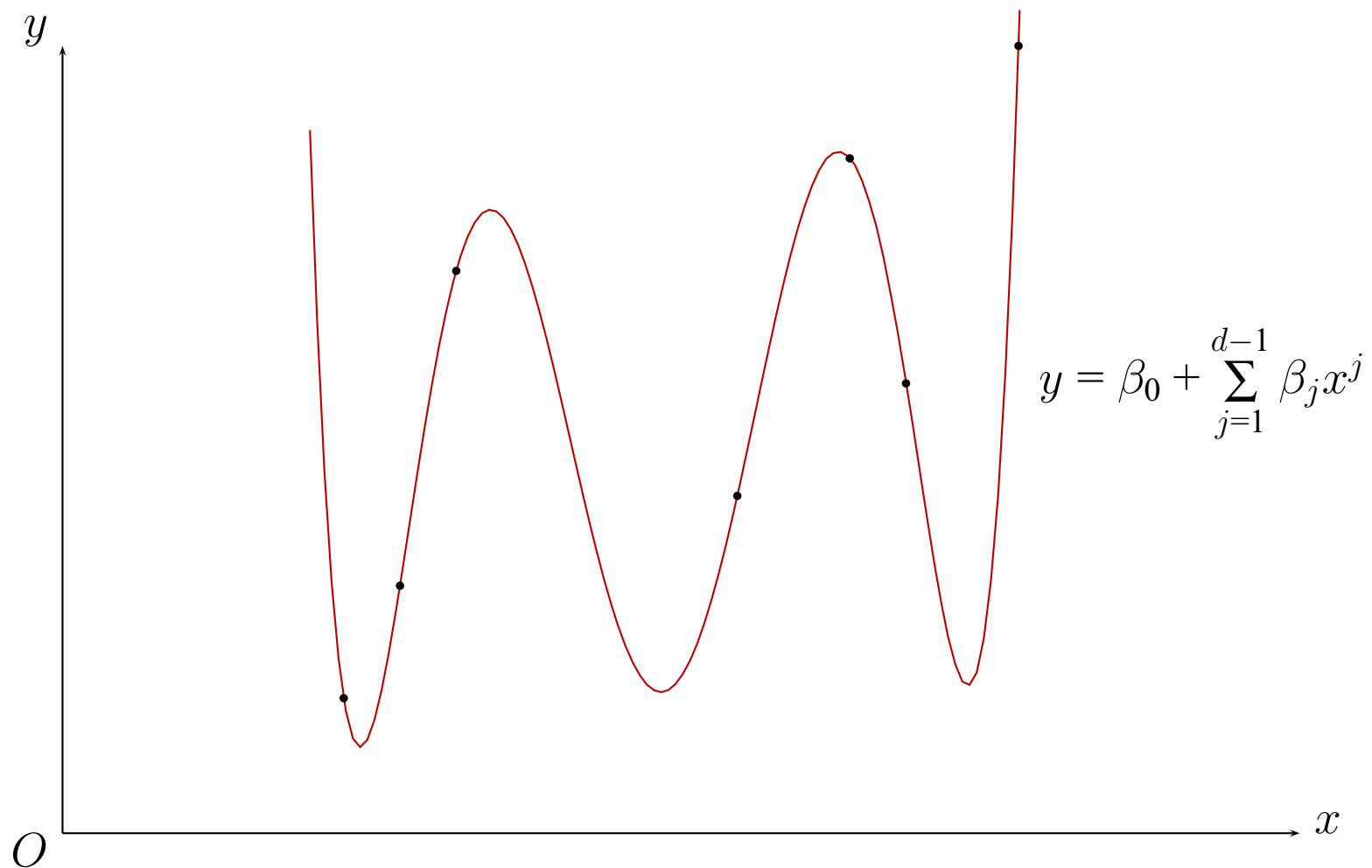


$$\hat{R}(f) = \frac{1}{N} \sum_{i=1}^N (\beta_0 + \beta x^{(i)} - y^{(i)})^2 \rightarrow \min_{\beta_0, \beta}$$



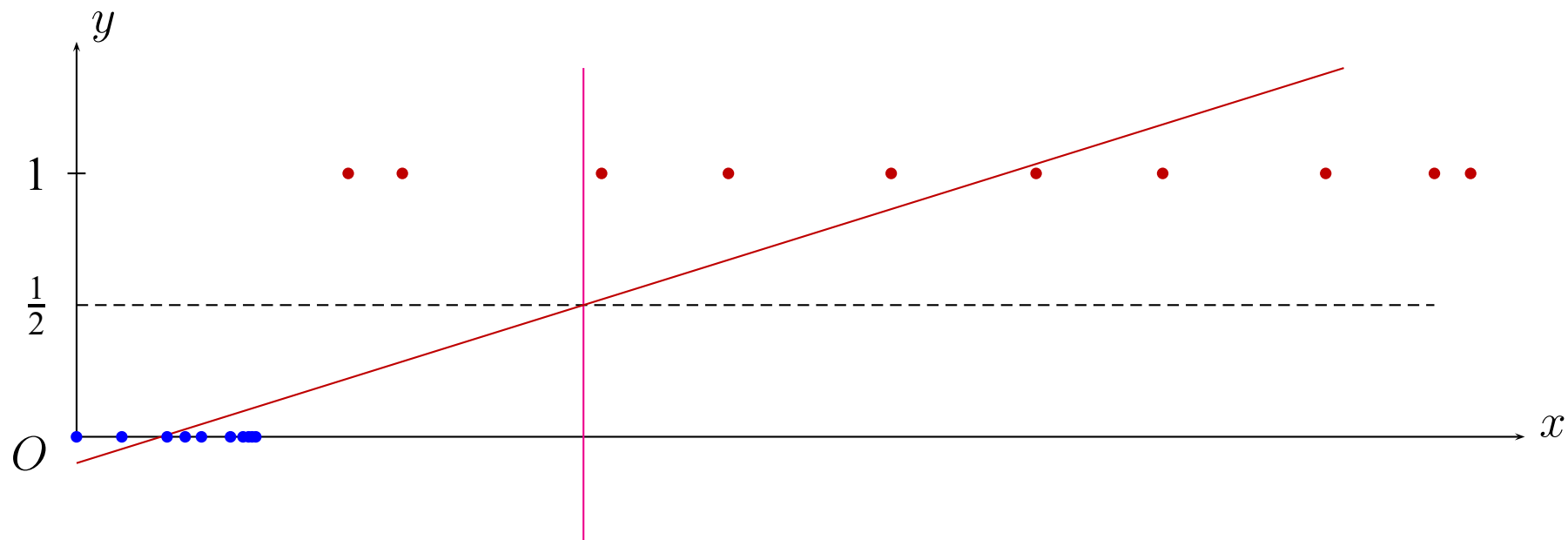
$$\hat{R}(f) = \frac{1}{N} \sum_{i=1}^N \left(\beta_0 + \sum_{j=1}^d \beta_j x_j^{(i)} - y^{(i)} \right)^2 \rightarrow \min$$

Переобучение

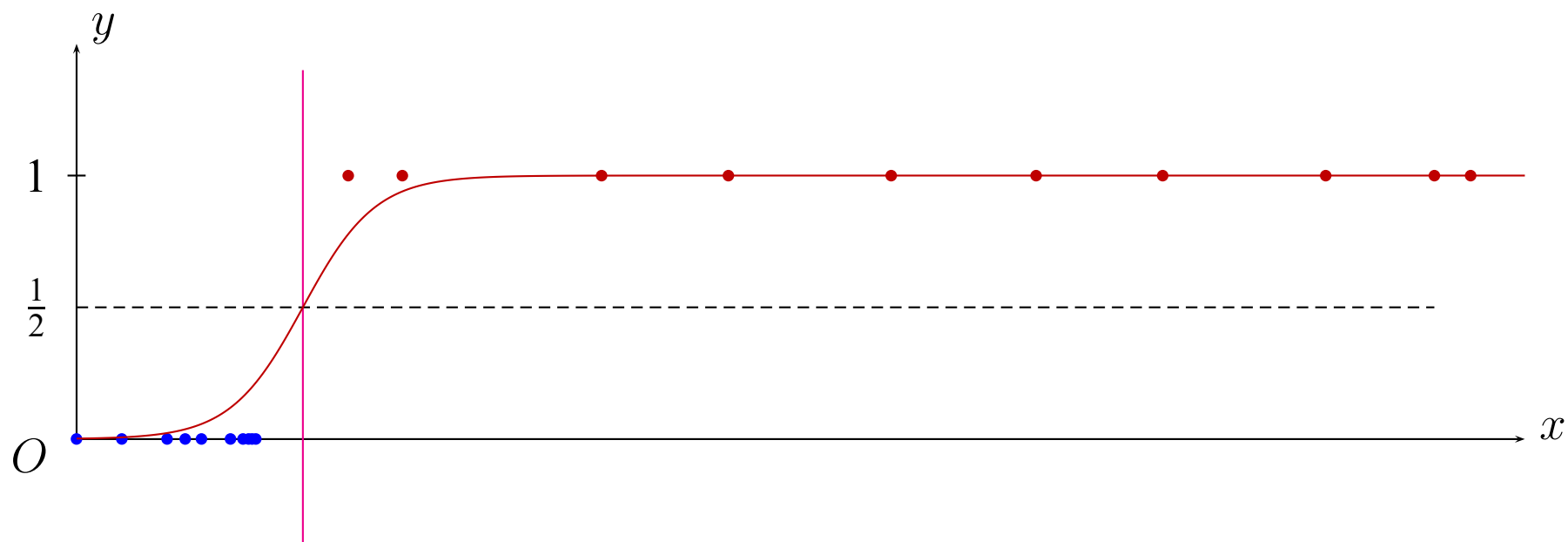
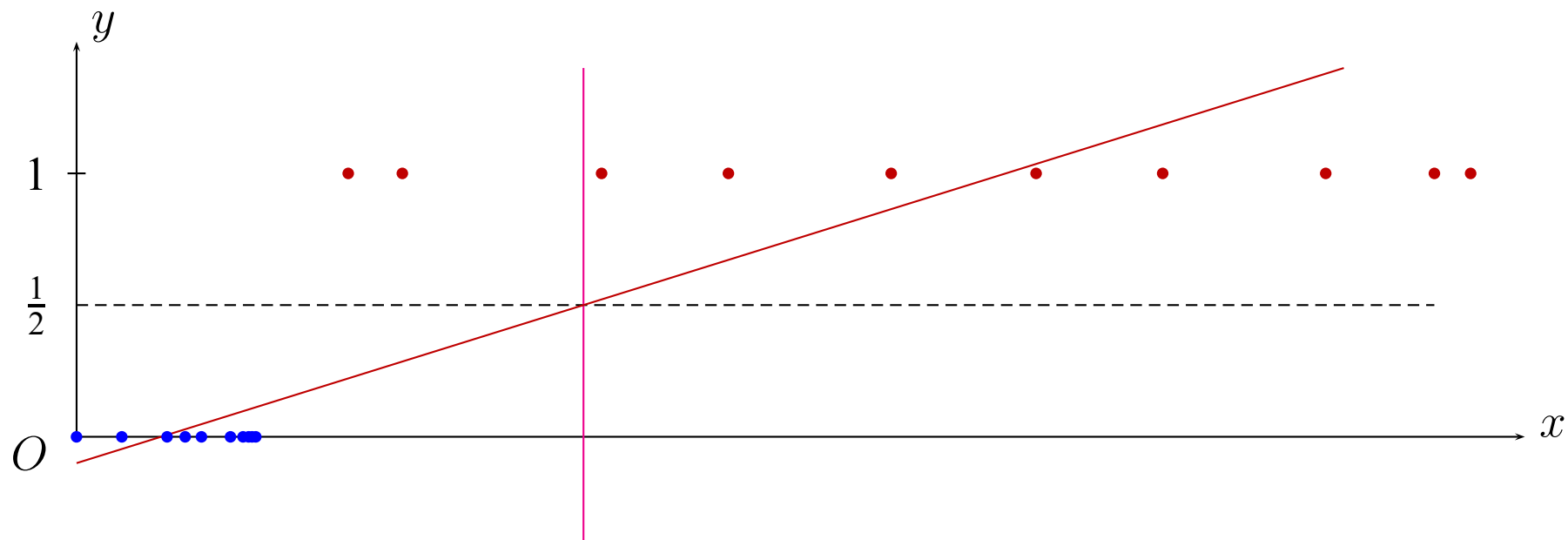


Метод наименьших квадратов для задачи классификации (?)

Метод наименьших квадратов для задачи классификации (?)



Метод наименьших квадратов для задачи классификации (?)



4.2. Логистическая регрессия

$$\mathcal{Y} = \{0, 1\}$$

4.2. Логистическая регрессия

$$\mathcal{Y} = \{0, 1\}$$

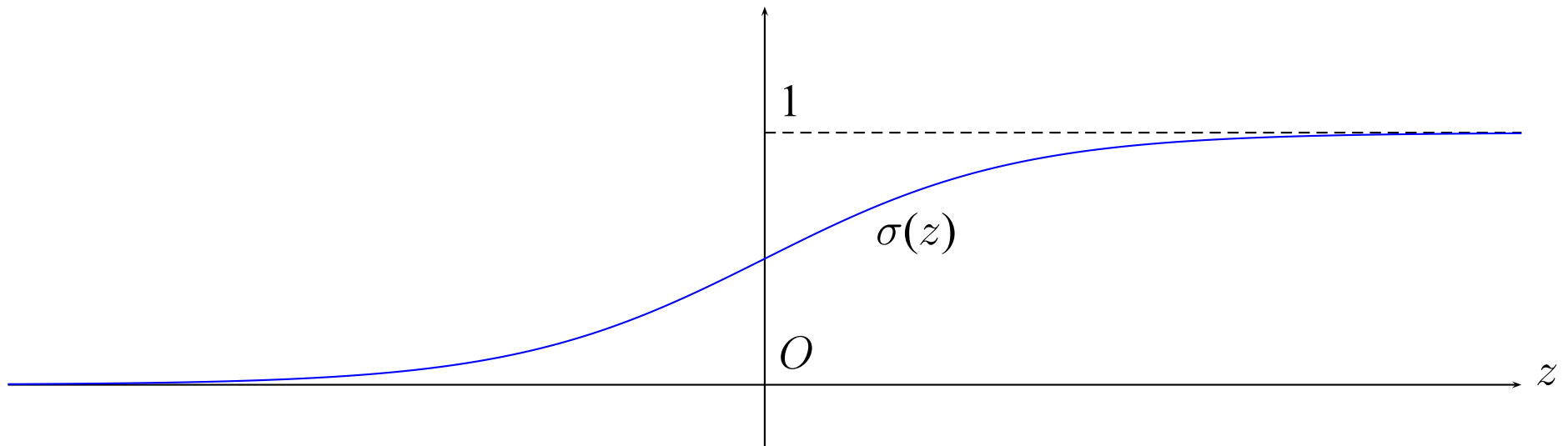
Пусть

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

где

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

— логистическая функция (элементарный сигмоид или логит-функция)



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

тогда

$$\Pr(Y = 0 | x) = 1 - \Pr(Y = 1 | x) = \frac{1}{1 + e^{\beta_0 + \beta_1 x_1 + \dots + \beta_d x_d}} = \sigma(-\beta_0 - \beta^\top x),$$

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

тогда

$$\Pr(Y = 0 | x) = 1 - \Pr(Y = 1 | x) = \frac{1}{1 + e^{\beta_0 + \beta_1 x_1 + \dots + \beta_d x_d}} = \sigma(-\beta_0 - \beta^\top x),$$

Разделяющая поверхность — линейная (гиперплоскость):

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

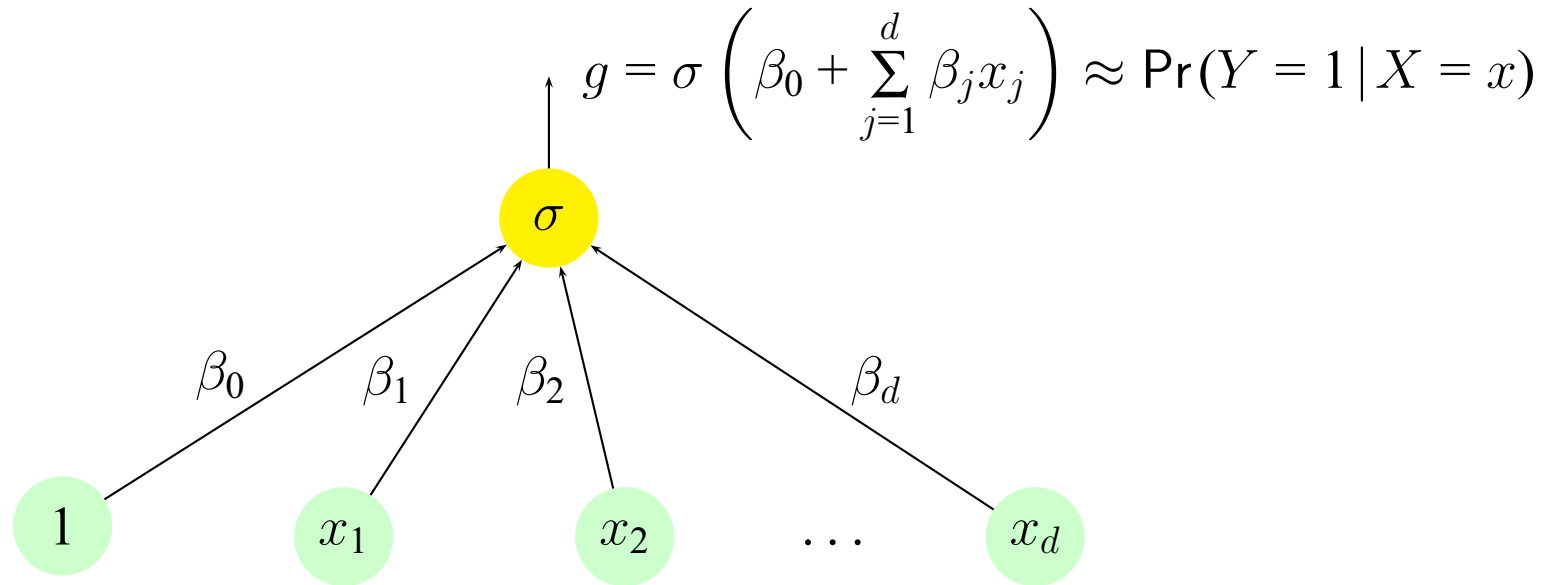
$$\Pr(Y = 1 | x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \dots + \beta_d x_d)}} = \sigma(\beta_0 + \beta^\top x),$$

тогда

$$\Pr(Y = 0 | x) = 1 - \Pr(Y = 1 | x) = \frac{1}{1 + e^{\beta_0 + \beta_1 x_1 + \dots + \beta_d x_d}} = \sigma(-\beta_0 - \beta^\top x),$$

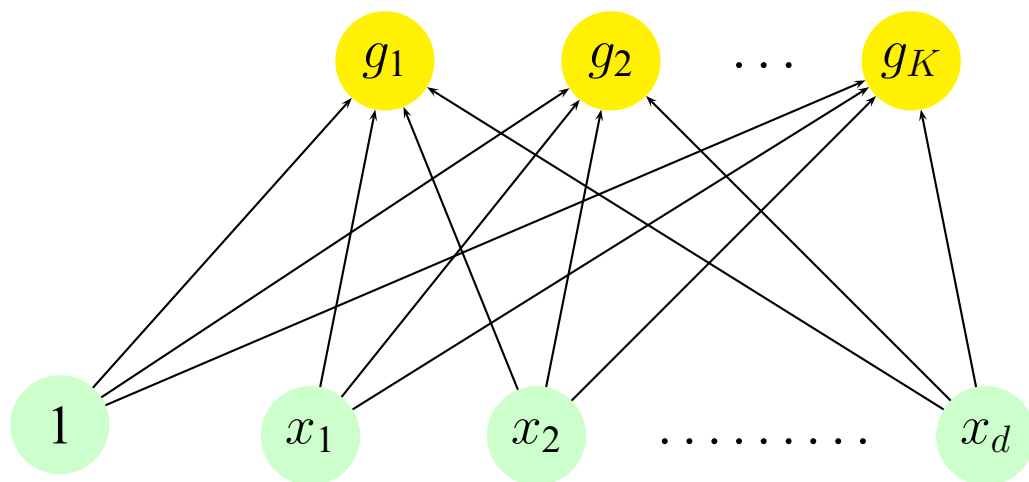
Разделяющая поверхность — линейная (гиперплоскость):

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



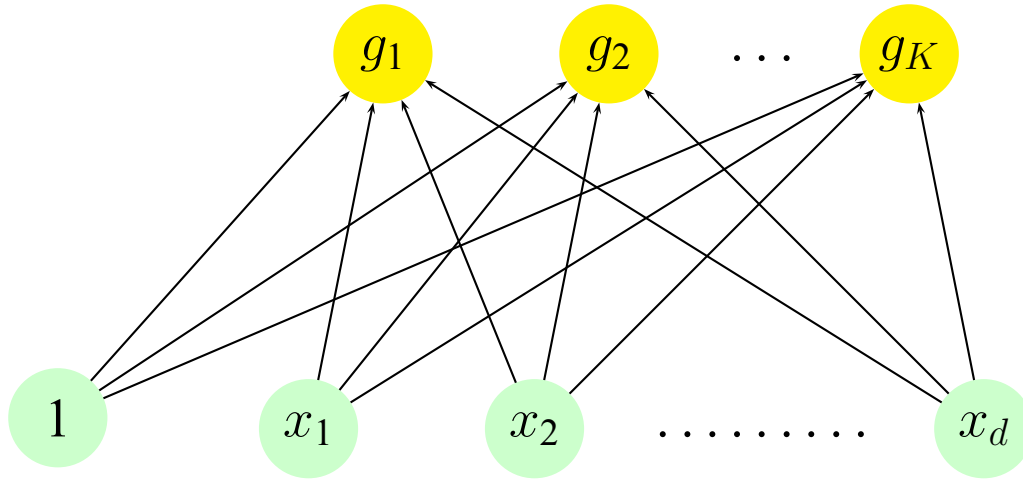
Случай K классов: $\mathcal{Y} = \{1, 2, \dots, K\}$. Функция softmax:

Случай K классов: $\mathcal{Y} = \{1, 2, \dots, K\}$. Функция softmax:



$$g_k(x) = \frac{\exp\left(\beta_{k0} + \sum_{j=1}^d \beta_{kj}x_j\right)}{\sum_{\ell=1}^K \exp\left(\beta_{\ell 0} + \sum_{j=1}^d \beta_{\ell j}x_j\right)} \approx \text{Pr}(k|x)$$
$$(k = 1, 2, \dots, K)$$

Случай K классов: $\mathcal{Y} = \{1, 2, \dots, K\}$. Функция softmax:



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

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

$$g_k(x) \geq 0 \quad (k = 1, 2, \dots, K), \quad \sum_{k=1}^K g_k(x) = 1.$$

Как обучить логистическую регрессию?

Как найти $\beta = (\beta_{10}, \beta_1, \beta_{20}, \beta_2, \dots, \beta_{K0}, \beta_K)$?

Как обучить логистическую регрессию?

Как найти $\beta = (\beta_{10}, \beta_1, \beta_{20}, \beta_2, \dots, \beta_{K0}, \beta_K)$?

Воспользуемся *методом максимального правдоподобия*.

Как обучить логистическую регрессию?

Как найти $\beta = (\beta_{10}, \beta_1, \beta_{20}, \beta_2, \dots, \beta_{K0}, \beta_K)$?

Воспользуемся *методом максимального правдоподобия*.

Правдоподобие – это вероятность, что наблюдаемые объекты принадлежат соответствующим классам:

$$\mathcal{L}(\beta) = \prod_{i=1}^N \Pr \{Y = y^{(i)} \mid X = x^{(i)}, \beta\} \rightarrow \max_{\beta}$$

Как обучить логистическую регрессию?

Как найти $\beta = (\beta_{10}, \beta_1, \beta_{20}, \beta_2, \dots, \beta_{K0}, \beta_K)$?

Воспользуемся *методом максимального правдоподобия*.

Правдоподобие – это вероятность, что наблюдаемые объекты принадлежат соответствующим классам:

$$\mathcal{L}(\beta) = \prod_{i=1}^N \Pr \{Y = y^{(i)} \mid X = x^{(i)}, \beta\} \rightarrow \max_{\beta}$$

Логарифмическая функция правдоподобия:

$$\ell(\beta) = \ln \mathcal{L}(\beta) = \sum_{i=1}^N \ln \Pr \{Y = y^{(i)} \mid X = x^{(i)}, \beta\} = \sum_{i=1}^N \sum_{k=1}^K I(y^{(i)} = k) \ln g_k(x^{(i)}, \beta) \rightarrow \max_{\beta}.$$

Как обучить логистическую регрессию?

Как найти $\beta = (\beta_{10}, \beta_1, \beta_{20}, \beta_2, \dots, \beta_{K0}, \beta_K)$?

Воспользуемся *методом максимального правдоподобия*.

Правдоподобие – это вероятность, что наблюдаемые объекты принадлежат соответствующим классам:

$$\mathcal{L}(\beta) = \prod_{i=1}^N \Pr \{Y = y^{(i)} | X = x^{(i)}, \beta\} \rightarrow \max_{\beta}$$

Логарифмическая функция правдоподобия:

$$\ell(\beta) = \ln \mathcal{L}(\beta) = \sum_{i=1}^N \ln \Pr \{Y = y^{(i)} | X = x^{(i)}, \beta\} = \sum_{i=1}^N \sum_{k=1}^K I(y^{(i)} = k) \ln g_k(x^{(i)}, \beta) \rightarrow \max_{\beta}.$$

Для минимизации используются численные методы (градиентный спуск, BFGS и др.)

Как обучить логистическую регрессию?

Как найти $\beta = (\beta_{10}, \beta_1, \beta_{20}, \beta_2, \dots, \beta_{K0}, \beta_K)$?

Воспользуемся *методом максимального правдоподобия*.

Правдоподобие – это вероятность, что наблюдаемые объекты принадлежат соответствующим классам:

$$\mathcal{L}(\beta) = \prod_{i=1}^N \Pr \{Y = y^{(i)} \mid X = x^{(i)}, \beta\} \rightarrow \max_{\beta}$$

Логарифмическая функция правдоподобия:

$$\ell(\beta) = \ln \mathcal{L}(\beta) = \sum_{i=1}^N \ln \Pr \{Y = y^{(i)} \mid X = x^{(i)}, \beta\} = \sum_{i=1}^N \sum_{k=1}^K I(y^{(i)} = k) \ln g_k(x^{(i)}, \beta) \rightarrow \max_{\beta}.$$

Для минимизации используются численные методы (градиентный спуск, BFGS и др.)

$-\ln \Pr \{Y = y^{(i)} \mid X = x^{(i)}, \beta\} = -\sum_{k=1}^K I(y^{(i)} = k) \ln g_k(x^{(i)}, \beta)$ называется *кросс-энтропией*.

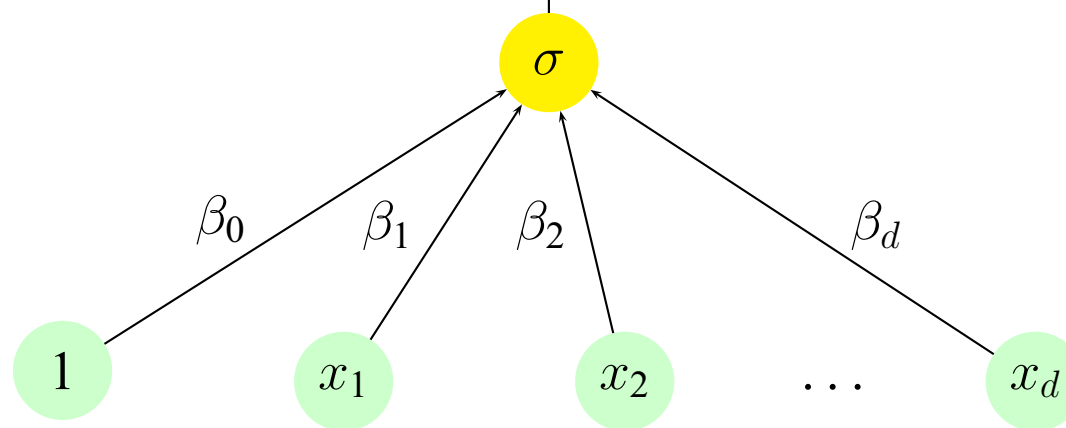
Таким образом, максимизация правдоподобия эквивалентна минимизации суммарной (или средней) кросс-энтропии.

4.3. Нейронные сети

4.3. Нейронные сети

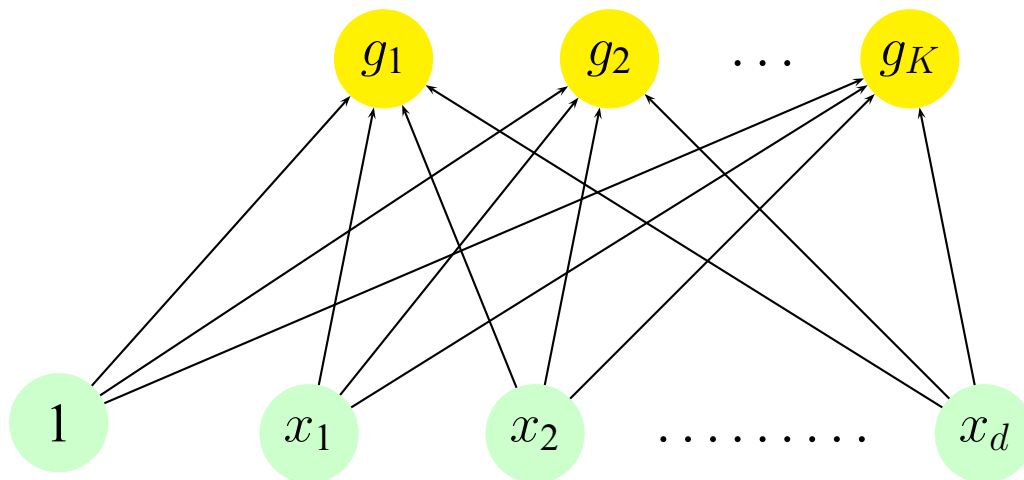
Логистическая регрессия — это однослойная* нейронная сеть.

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



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

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



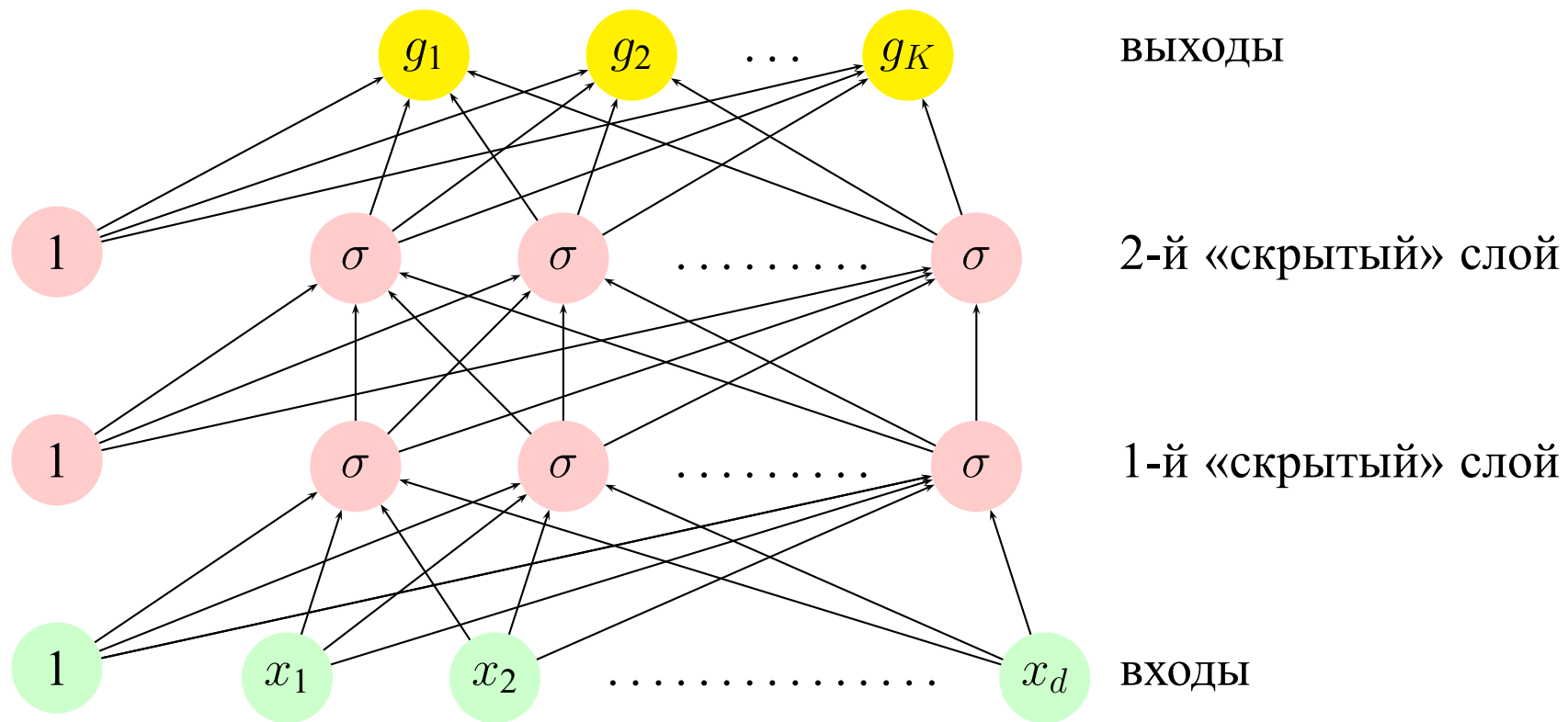
Нейронная сеть (или *искусственная нейронная сеть* — ориентированный граф, вершинам которого соответствуют функции (*функции активации*), а каждой входящей в вершину дуге (*синапсу*) — ее аргумент.

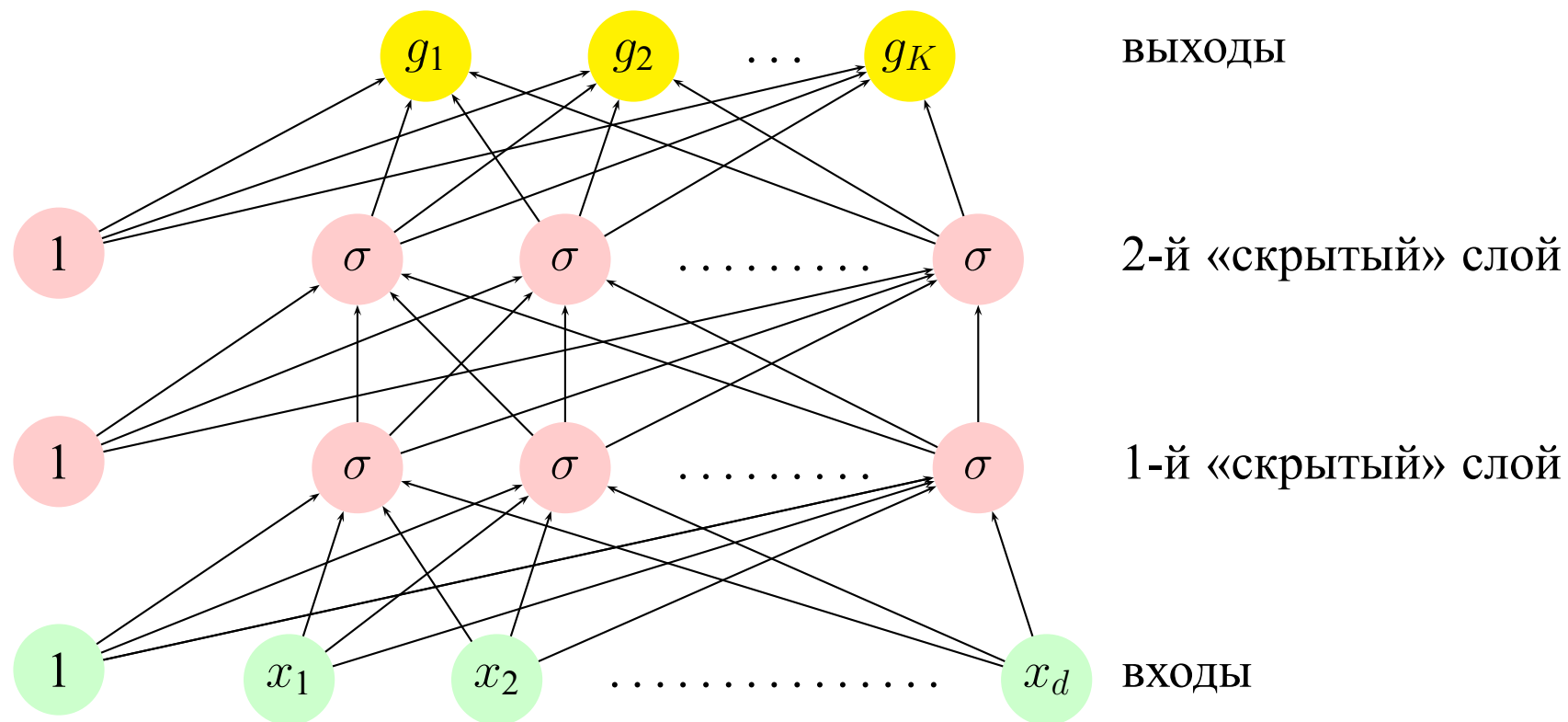
Вход нейрона — дендрит. Выход — аксон.

Нейронная сеть (или *искусственная нейронная сеть* — ориентированный граф, вершинам которого соответствуют функции (*функции активации*), а каждой входящей в вершину дуге (*синапсу*) — ее аргумент.

Вход нейрона — дендрит. Выход — аксон.

С математической точки зрения нейронная сеть графически задает суперпозицию функций активации.

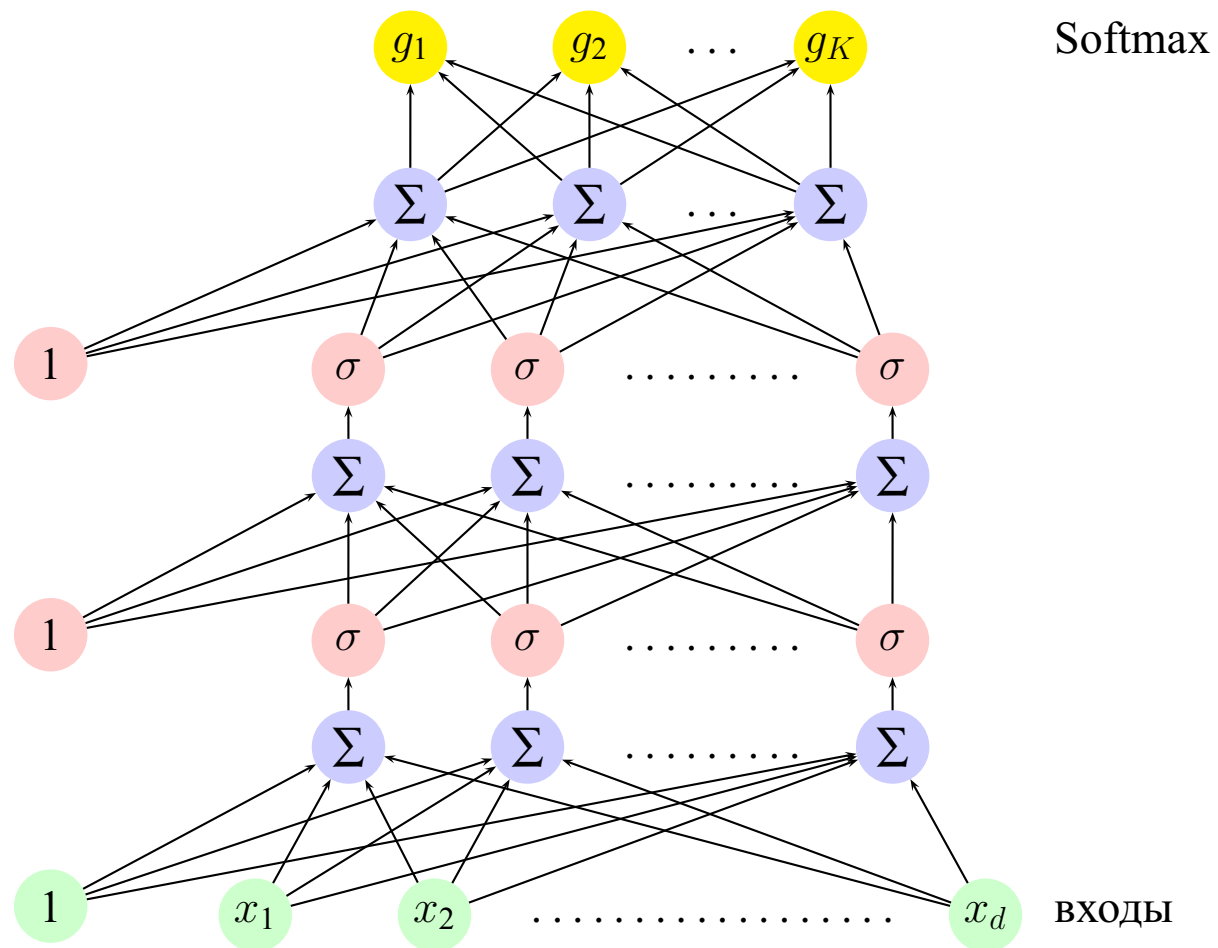




Таким образом, выходы из каждого узла (нейрона) умножаются на соответствующие веса и складываются.

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

Иногда отдельно изображают суммирующие элементы и элементы, вычисляющие σ :

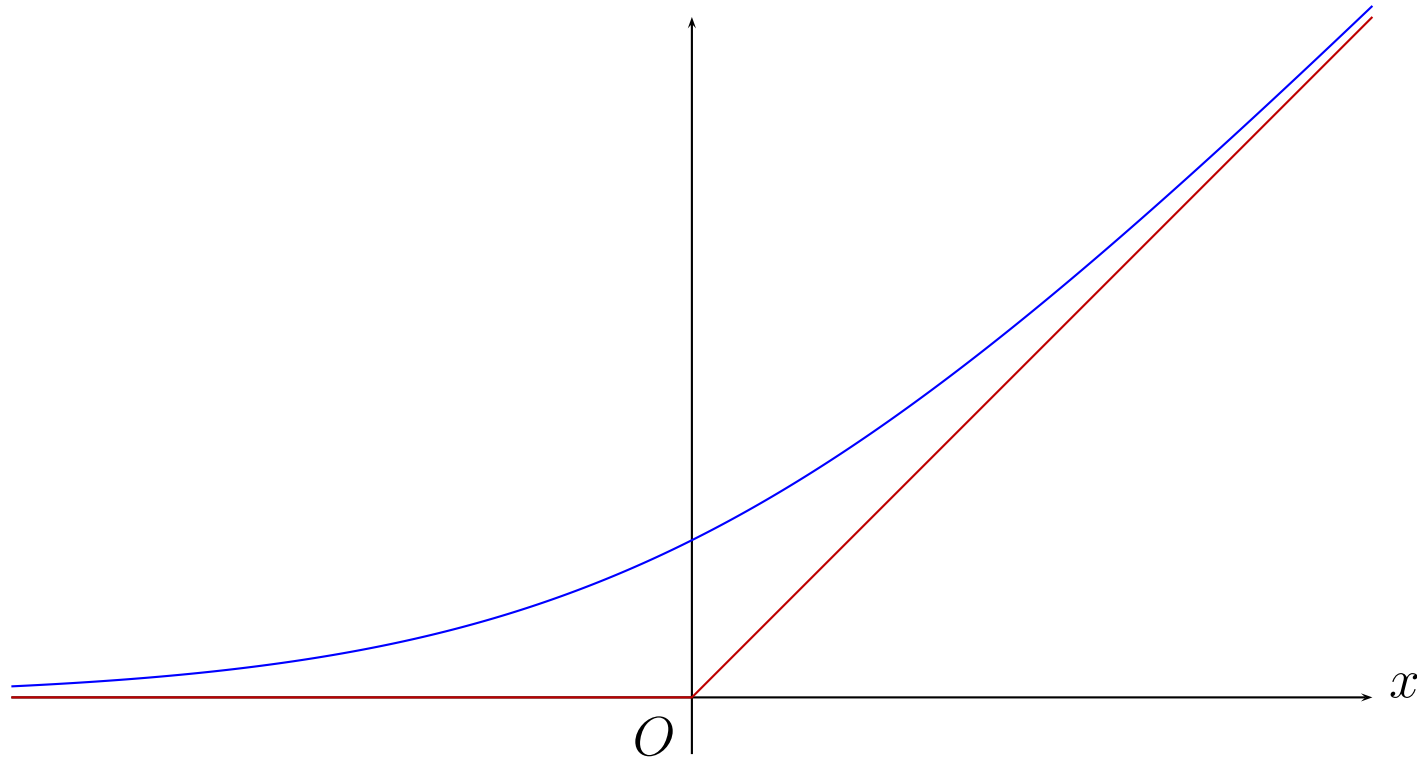


Кроме сигмоидальной используют и другие функции, например, *положительную срезку линейной функции* (RELU – linear rectifier):

$$g(x_1, x_2, \dots, x_q) = (\beta_0 + \beta_1 x_1 + \dots + \beta_q x_q)_+$$

где $(x)_+ = \max \{0, x\}$, или ее сглаженный вариант *softplus*:

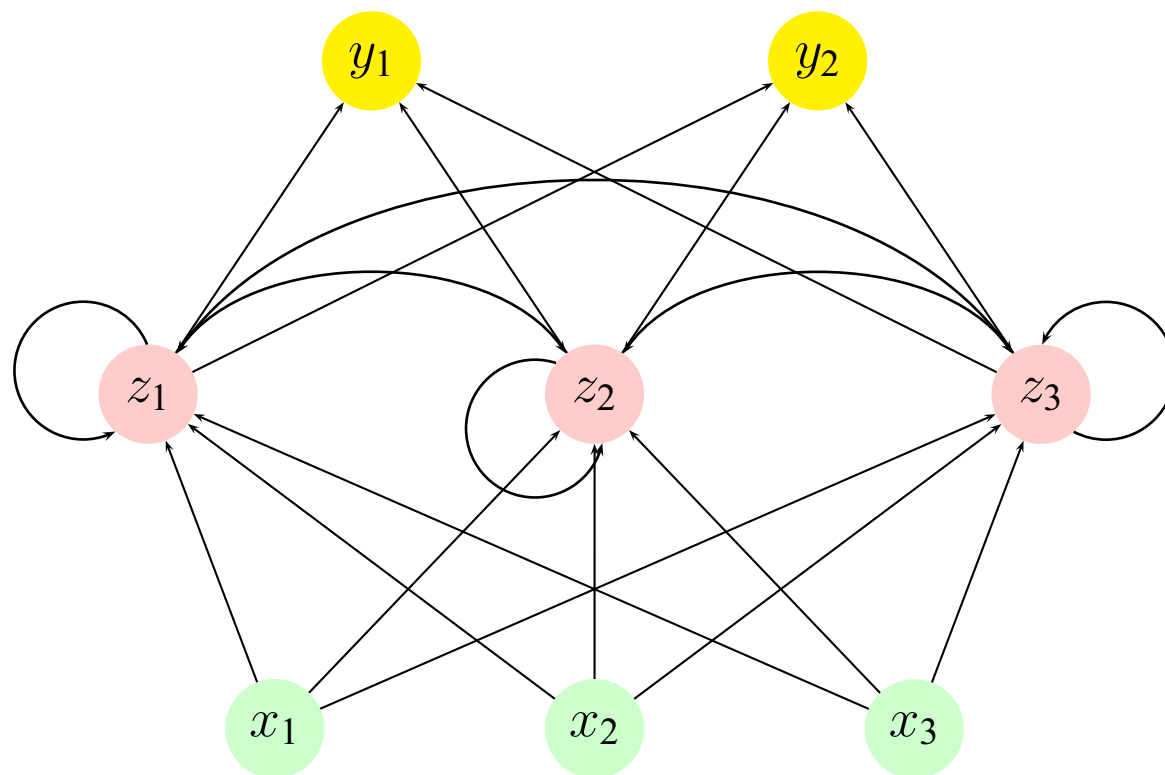
$$g = \ln(1 + \exp(\beta_0 + \beta_1 x_1 + \dots + \beta_q x_q))$$



Рекуррентные сети

В сети могут присутствовать орциклы.

Рекуррентные нейронные сети используют, например, для предсказания временных рядов.



Обучение нейронной сети (backpropagation)

Обучение нейронной сети — это подбор параметров (весов нейронной сети) для подгона сети под обучающую выборку.

Обучение нейронной сети (backpropagation)

Обучение нейронной сети — это подбор параметров (весов нейронной сети) для подгона сети под обучающую выборку.

В задачах классификации обычно минимизируют *кросс-энтропию* (с точностью до отрицательного множителя это логарифмическая функция правдоподобия!)

$$\hat{R}(\beta) = -\frac{1}{N} \sum_{i=1}^N \sum_{k=1}^K I(y_i = k) \ln g_k(x^{(i)}).$$

Обучение нейронной сети (backpropagation)

Обучение нейронной сети — это подбор параметров (весов нейронной сети) для подгона сети под обучающую выборку.

В задачах классификации обычно минимизируют *кросс-энтропию* (с точностью до отрицательного множителя это логарифмическая функция правдоподобия!)

$$\hat{R}(\beta) = -\frac{1}{N} \sum_{i=1}^N \sum_{k=1}^K I(y_i = k) \ln g_k(x^{(i)}).$$

Задача минимизации $\hat{R}(\beta)$ решается численными методами: например, методом градиентного спуска, квазиньютоновскими методами и т. п.

Нужно уметь вычислять компоненты градиента.

Обучение нейронной сети (backpropagation)

Обучение нейронной сети — это подбор параметров (весов нейронной сети) для подгона сети под обучающую выборку.

В задачах классификации обычно минимизируют *кросс-энтропию* (с точностью до отрицательного множителя это логарифмическая функция правдоподобия!)

$$\hat{R}(\beta) = -\frac{1}{N} \sum_{i=1}^N \sum_{k=1}^K I(y_i = k) \ln g_k(x^{(i)}).$$

Задача минимизации $\hat{R}(\beta)$ решается численными методами: например, методом градиентного спуска, квазиньютоновскими методами и т. п.

Нужно уметь вычислять компоненты градиента.

Алгоритм *обратного распространения ошибки* (*back propagation*) — это метод вычисления градиента.

Коротко 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 представляет абстракцию на базе предыдущих слоев.

4.4. Сверточные нейронные сети





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,

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



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,

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



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,

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

Основная идея сверточных нейронных сетей (сверточных слоев):

Параметры фильтра 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.

Основная идея сверточных нейронных сетей (сверточных слоев):

Параметры фильтра 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)

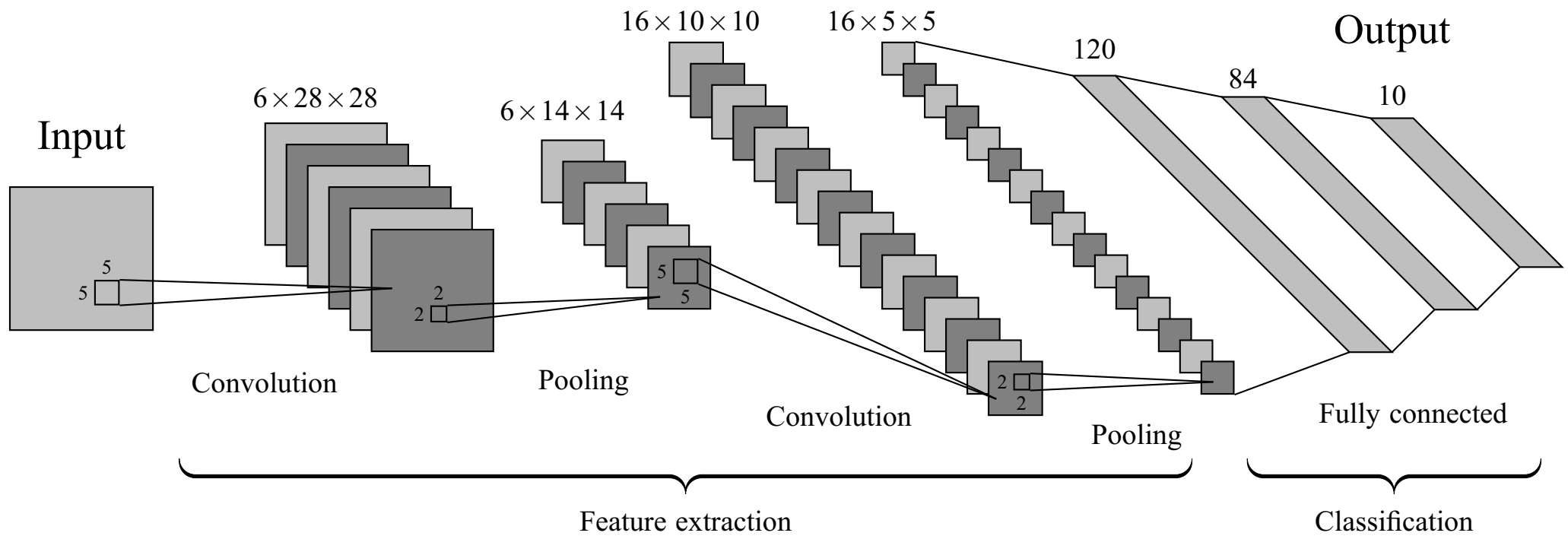
z_{pq} — узлы следующего слоя

Параметры фильтра — это теперь веса нейронной сети.

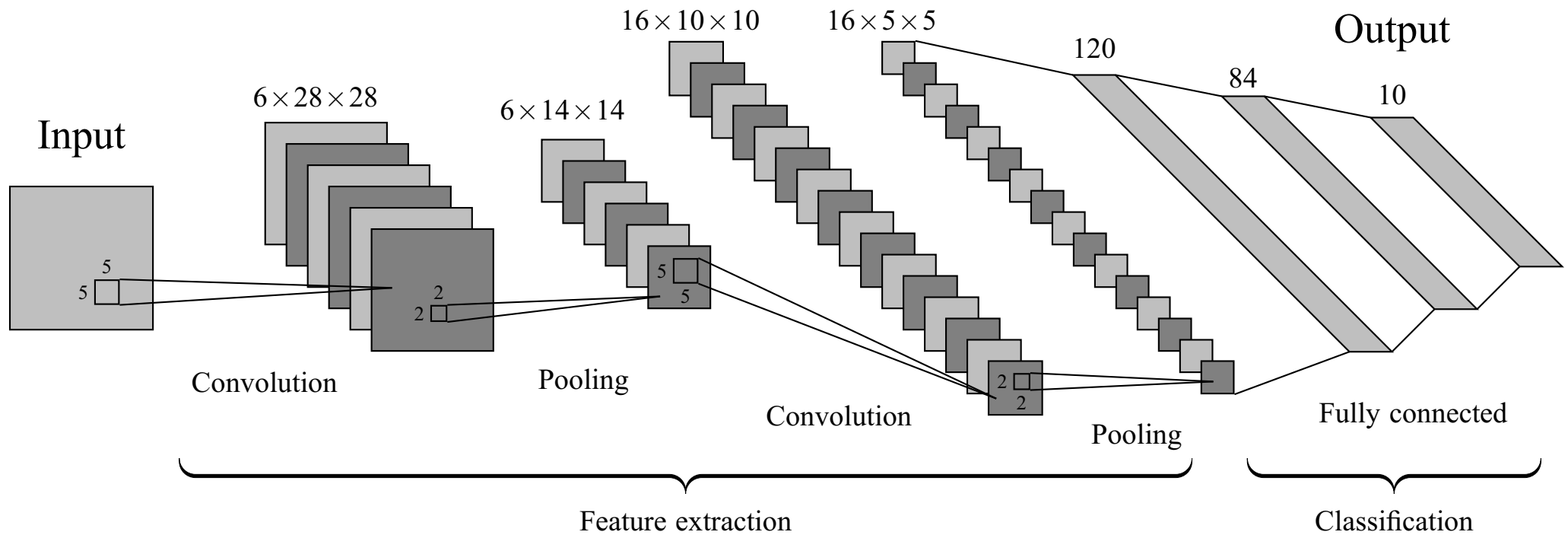
Отличия от полносвязной сети (полносвязного слоя):

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

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



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

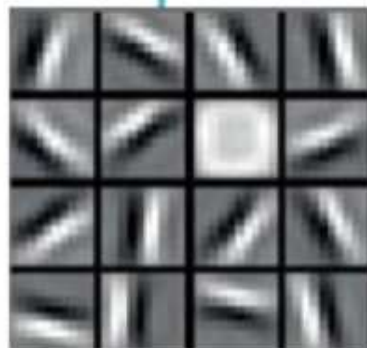


- *Сверточные слои (convolutional layers)*
- *«Выборочные» слои, или слои объединения (subsampling/pooling layers)*
- *Полносвязные слои (fully connected layers)*

Input Layer



Hidden Layer 1



edges

Hidden Layer 2



combinations of edges

Hidden Layer 3



object models

Другие примеры:

- AlexNet (Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton, 2012) — победитель ImageNet-2012 — 8 слоев
- 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
- Microsoft ResNet (2015) — 34 слоя
- GoogLeNet (2015) — 71 слой (хотя как считать)
- ...

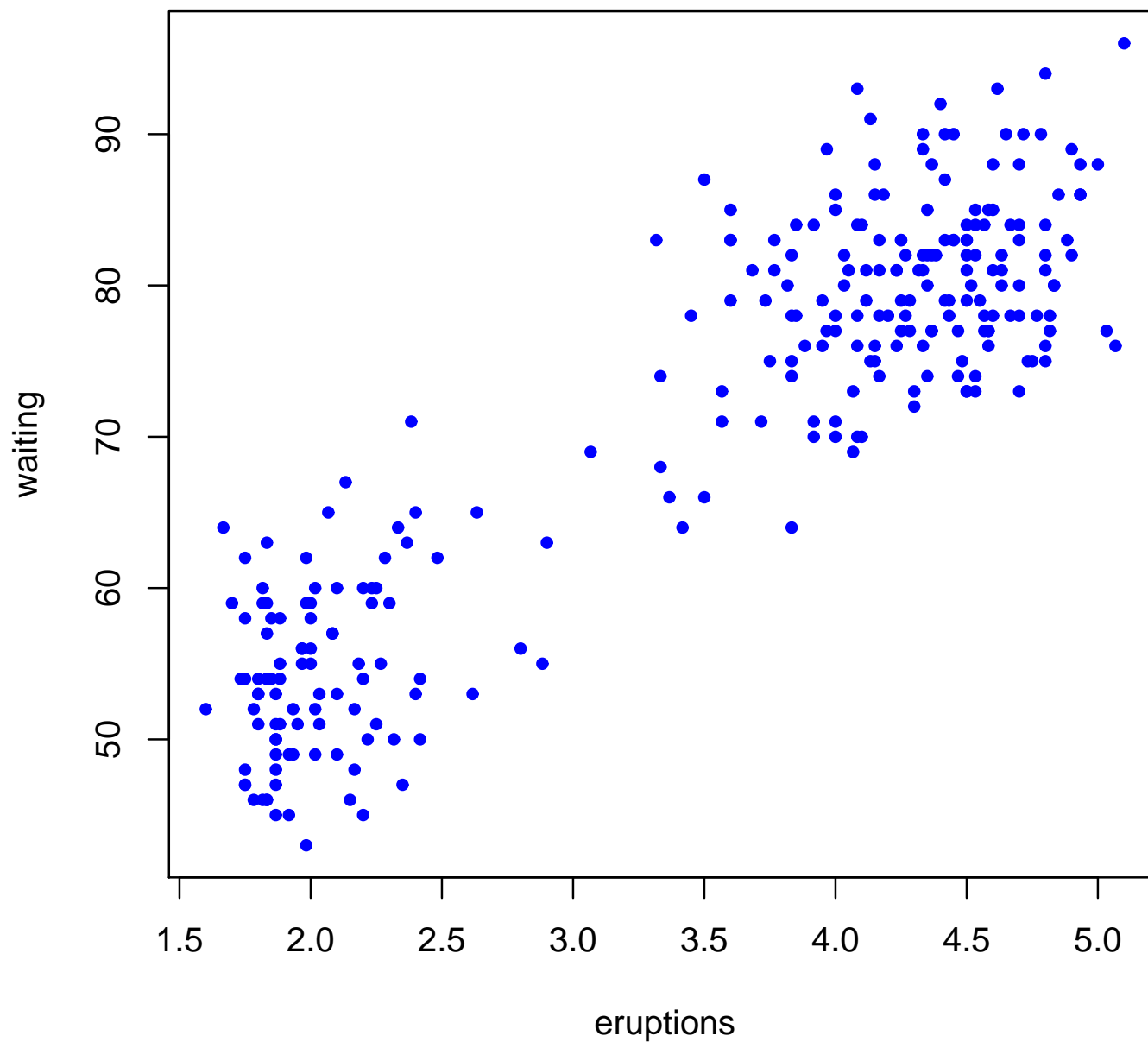
5. Обучение без учителя: кластеризация

5. Обучение без учителя: кластеризация

Пример 1. Извержения гейзера

Рассмотрим данные о времени между извержениями и длительностью извержения гейзера Old Faithful geyser in Yellowstone National Park, Wyoming, USA (*A. Azzalini, A.W. Bowman* A look at some data on the Old Faithful geyser // *Applied Statistics*. 1990, 39. P. 357--365.)

Диаграмма, представляющая данные о времени извержения и промежутках между извержениями гейзера.



Мы видим, что точки группируются в два кластера.

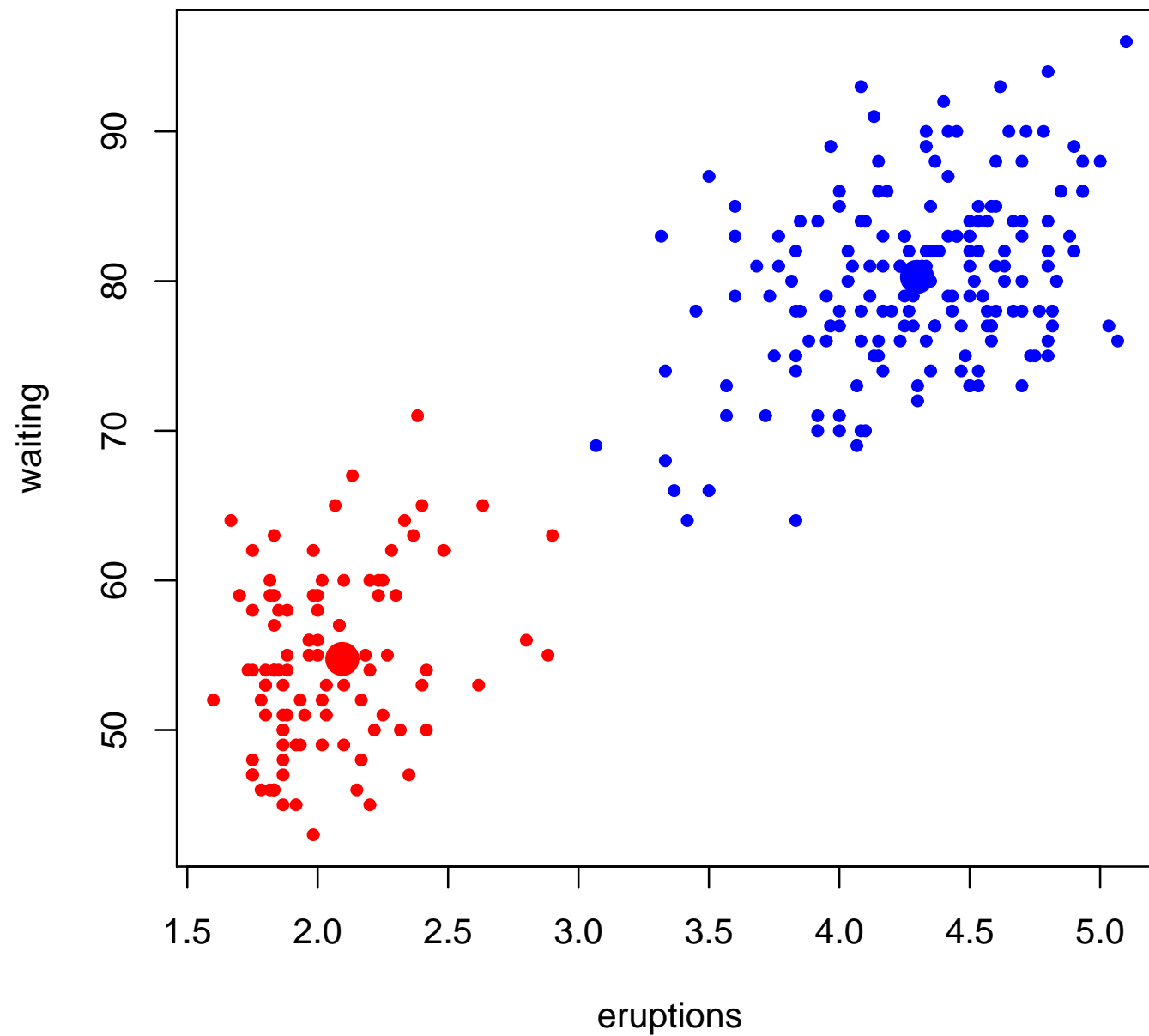
Мы видим, что точки группируются в два кластера.

В одном кластере находятся точки, соответствующие извержениям с малой длительностью и малым временем ожидания.

Мы видим, что точки группируются в два кластера.

В одном кластере находятся точки, соответствующие извержениям с малой длительностью и малым временем ожидания.

В другом — с большой длительностью и большим временем ожидания.



5.1. Метод центров тяжести

Рассмотрим *метод центров тяжести* (*K-means*).

5.1. Метод центров тяжести

Рассмотрим *метод центров тяжести* (*K-means*).

Предположим, что множество объектов уже разбито некоторым образом на K групп (кластеров) и значение $C(i)$ равно номеру группы, которой принадлежит i -й объект.

5.1. Метод центров тяжести

Рассмотрим *метод центров тяжести* (*K-means*).

Предположим, что множество объектов уже разбито некоторым образом на K групп (кластеров) и значение $C(i)$ равно номеру группы, которой принадлежит i -й объект.

Обозначим μ_k центр тяжести системы точек, принадлежащих k -му кластеру:

$$\mu_k = \frac{1}{|\{i : C(i) = k\}|} \sum_{C(i)=k} x^{(i)} \quad (k = 1, 2, \dots, K).$$

5.1. Метод центров тяжести

Рассмотрим *метод центров тяжести* (*K-means*).

Предположим, что множество объектов уже разбито некоторым образом на K групп (кластеров) и значение $C(i)$ равно номеру группы, которой принадлежит i -й объект.

Обозначим μ_k центр тяжести системы точек, принадлежащих k -му кластеру:

$$\mu_k = \frac{1}{|\{i : C(i) = k\}|} \sum_{C(i)=k} x^{(i)} \quad (k = 1, 2, \dots, K).$$

Будем минимизировать

$$\min_{C, \mu_k} \sum_{i=1}^N \|x^{(i)} - \mu_{C(i)}\|^2. \quad (1)$$

5.1. Метод центров тяжести

Рассмотрим *метод центров тяжести* (*K-means*).

Предположим, что множество объектов уже разбито некоторым образом на K групп (кластеров) и значение $C(i)$ равно номеру группы, которой принадлежит i -й объект.

Обозначим μ_k центр тяжести системы точек, принадлежащих k -му кластеру:

$$\mu_k = \frac{1}{|\{i : C(i) = k\}|} \sum_{C(i)=k} x^{(i)} \quad (k = 1, 2, \dots, K).$$

Будем минимизировать

$$\min_{C, \mu_k} \sum_{i=1}^N \|x^{(i)} - \mu_{C(i)}\|^2. \quad (1)$$

Для этого применим следующий итерационный алгоритм.

5.1. Метод центров тяжести

Рассмотрим *метод центров тяжести* (*K-means*).

Предположим, что множество объектов уже разбито некоторым образом на K групп (кластеров) и значение $C(i)$ равно номеру группы, которой принадлежит i -й объект.

Обозначим μ_k центр тяжести системы точек, принадлежащих k -му кластеру:

$$\mu_k = \frac{1}{|\{i : C(i) = k\}|} \sum_{C(i)=k} x^{(i)} \quad (k = 1, 2, \dots, K).$$

Будем минимизировать

$$\min_{C, \mu_k} \sum_{i=1}^N \|x^{(i)} - \mu_{C(i)}\|^2. \quad (1)$$

Для этого применим следующий итерационный алгоритм.

Если какое-то разбиение на кластеры уже известно, то найдем μ_k ($k = 1, 2, \dots, m$), а затем обновим $C(i)$ ($i = 1, 2, \dots, N$), приписав точке $x^{(i)}$ тот класс k ($k = 1, 2, \dots, K$), для которого расстояние от $x^{(i)}$ до μ_k минимально.

5.1. Метод центров тяжести

Рассмотрим *метод центров тяжести* (*K-means*).

Предположим, что множество объектов уже разбито некоторым образом на K групп (кластеров) и значение $C(i)$ равно номеру группы, которой принадлежит i -й объект.

Обозначим μ_k центр тяжести системы точек, принадлежащих k -му кластеру:

$$\mu_k = \frac{1}{|\{i : C(i) = k\}|} \sum_{C(i)=k} x^{(i)} \quad (k = 1, 2, \dots, K).$$

Будем минимизировать

$$\min_{C, \mu_k} \sum_{i=1}^N \|x^{(i)} - \mu_{C(i)}\|^2. \quad (1)$$

Для этого применим следующий итерационный алгоритм.

Если какое-то разбиение на кластеры уже известно, то найдем μ_k ($k = 1, 2, \dots, m$), а затем обновим $C(i)$ ($i = 1, 2, \dots, N$), приписав точке $x^{(i)}$ тот класс k ($k = 1, 2, \dots, K$), для которого расстояние от $x^{(i)}$ до μ_k минимально.

После этого повторим итерацию.

Получаем следующий алгоритм.

begin

Случайно инициировать μ_k ($k = 1, 2, \dots, K$)

repeat

Положить $C(i) \leftarrow \operatorname{argmin}_{1 \leq k \leq K} \|x^{(i)} - \mu_k\|^2 \quad (i = 1, 2, \dots, N)$

Найти $\mu_k \leftarrow \frac{1}{|\{i : C(i) = k\}|} \sum_{C(i)=k} x^{(i)} \quad (k = 1, 2, \dots, K)$

end

end

Получаем следующий алгоритм.

begin

Случайно инициировать μ_k ($k = 1, 2, \dots, K$)

repeat

Положить $C(i) \leftarrow \operatorname{argmin}_{1 \leq k \leq K} \|x^{(i)} - \mu_k\|^2 \quad (i = 1, 2, \dots, N)$

Найти $\mu_k \leftarrow \frac{1}{|\{i : C(i) = k\}|} \sum_{C(i)=k} x^{(i)} \quad (k = 1, 2, \dots, K)$

end

end

Итерации в алгоритме повторяются до тех пор, пока μ_k , $C(i)$ перестанут изменяться.

Получаем следующий алгоритм.

begin

Случайно инициировать μ_k ($k = 1, 2, \dots, K$)

repeat

Положить $C(i) \leftarrow \operatorname{argmin}_{1 \leq k \leq K} \|x^{(i)} - \mu_k\|^2 \quad (i = 1, 2, \dots, N)$

Найти $\mu_k \leftarrow \frac{1}{|\{i : C(i) = k\}|} \sum_{C(i)=k} x^{(i)} \quad (k = 1, 2, \dots, K)$

end

end

Итерации в алгоритме повторяются до тех пор, пока μ_k , $C(i)$ перестанут изменяться. Алгоритм может не сойтись к глобальному минимуму, но всегда найдет локальный минимум.

Получаем следующий алгоритм.

begin

Случайно инициировать μ_k ($k = 1, 2, \dots, K$)

repeat

Положить $C(i) \leftarrow \underset{1 \leq k \leq K}{\operatorname{argmin}} \|x^{(i)} - \mu_k\|^2 \quad (i = 1, 2, \dots, N)$

Найти $\mu_k \leftarrow \frac{1}{|\{i : C(i) = k\}|} \sum_{C(i)=k} x^{(i)} \quad (k = 1, 2, \dots, K)$

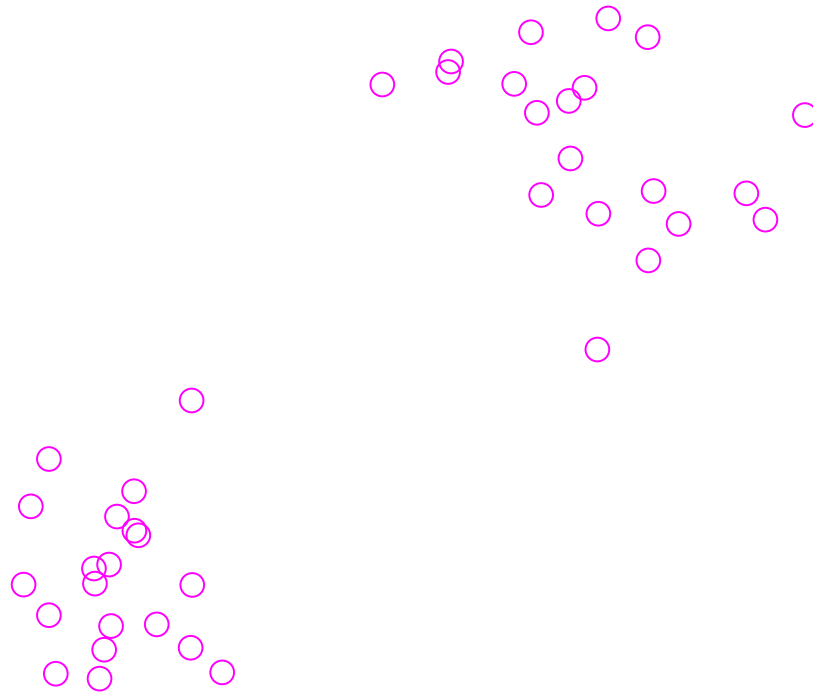
end

end

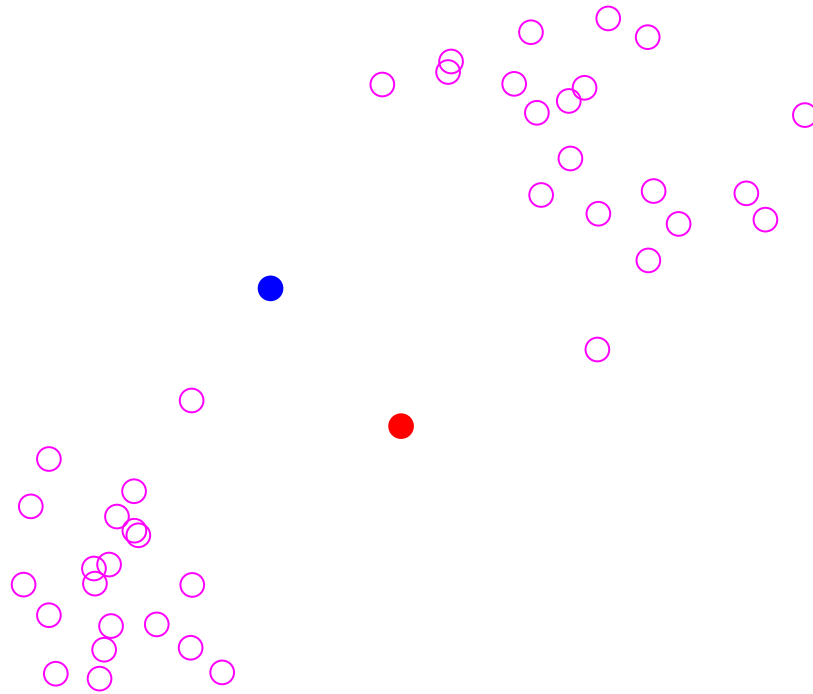
Итерации в алгоритме повторяются до тех пор, пока μ_k , $C(i)$ перестанут изменяться. Алгоритм может не сойтись к глобальному минимуму, но всегда найдет локальный минимум.

Можно начинать со случайного разбиения объектов на K кластеров, найти у них центры тяжести и продолжать итерации.

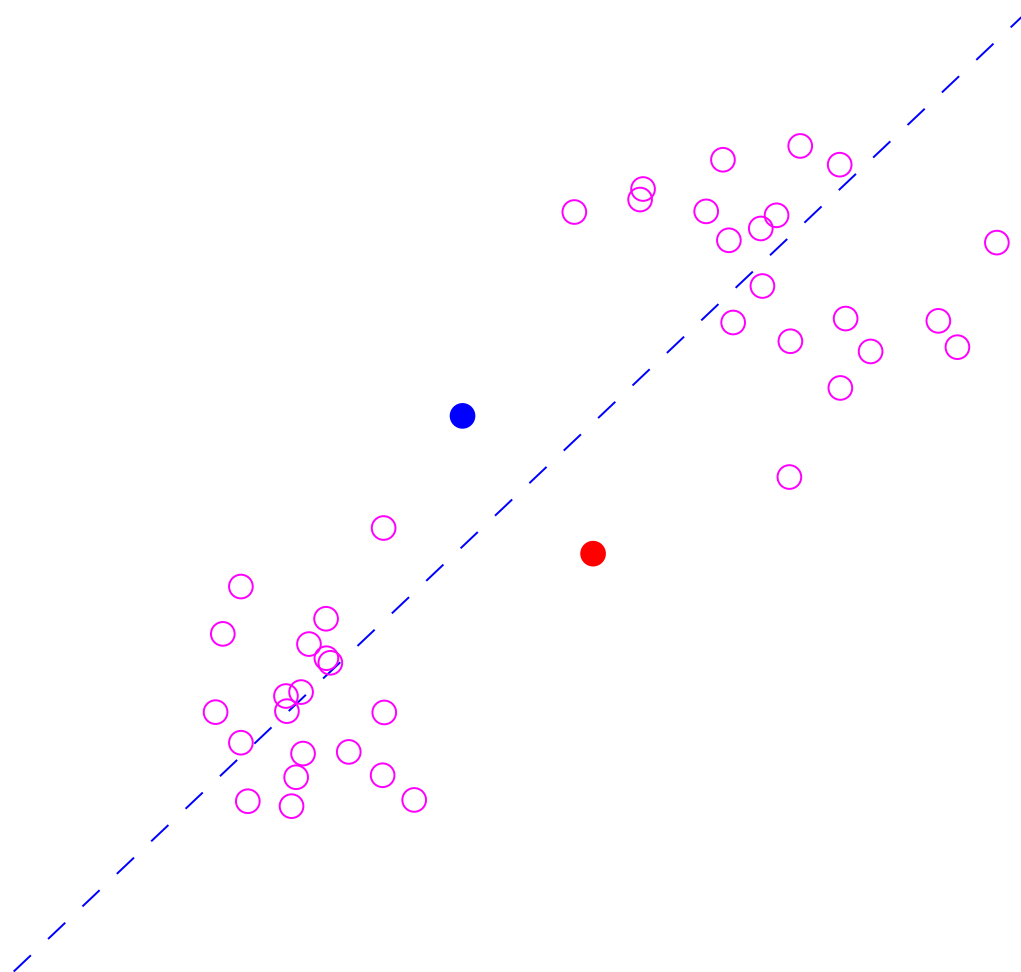
Иллюстрация, $K = 2$:



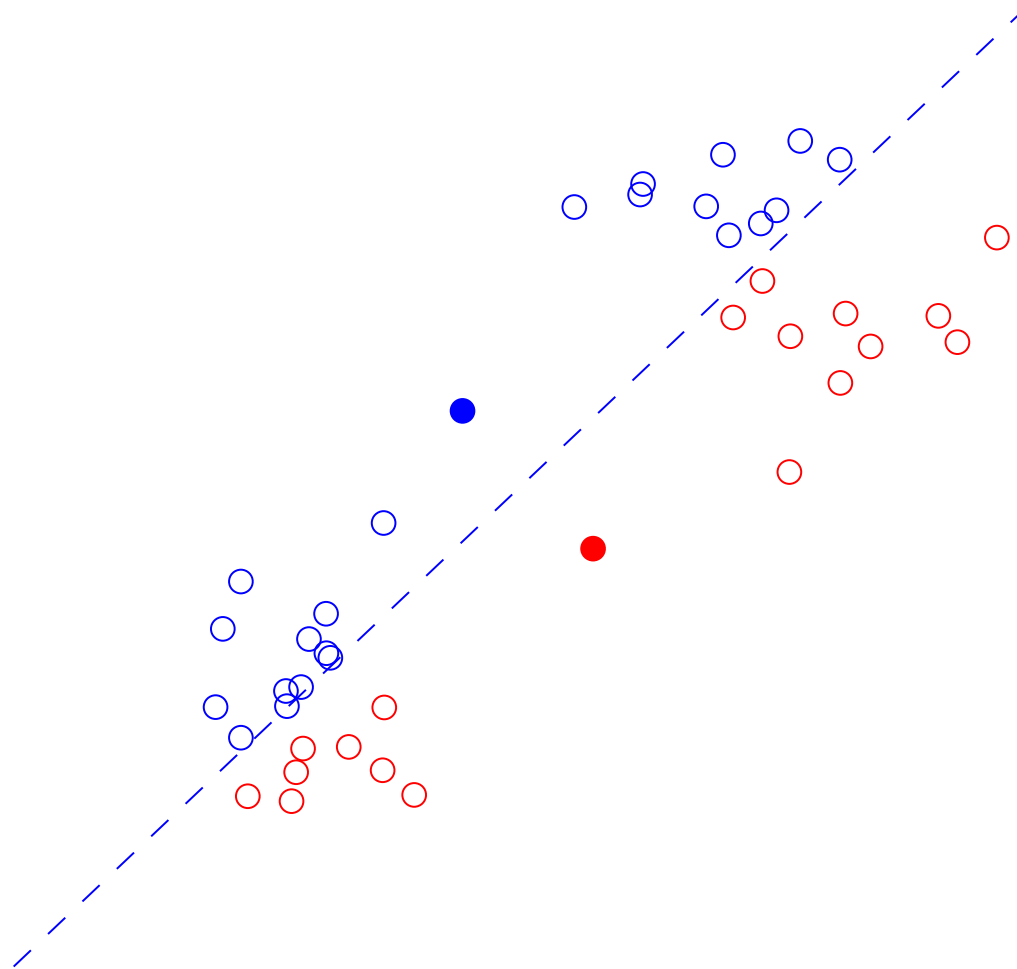
Случайно выбираем центры кластеров:



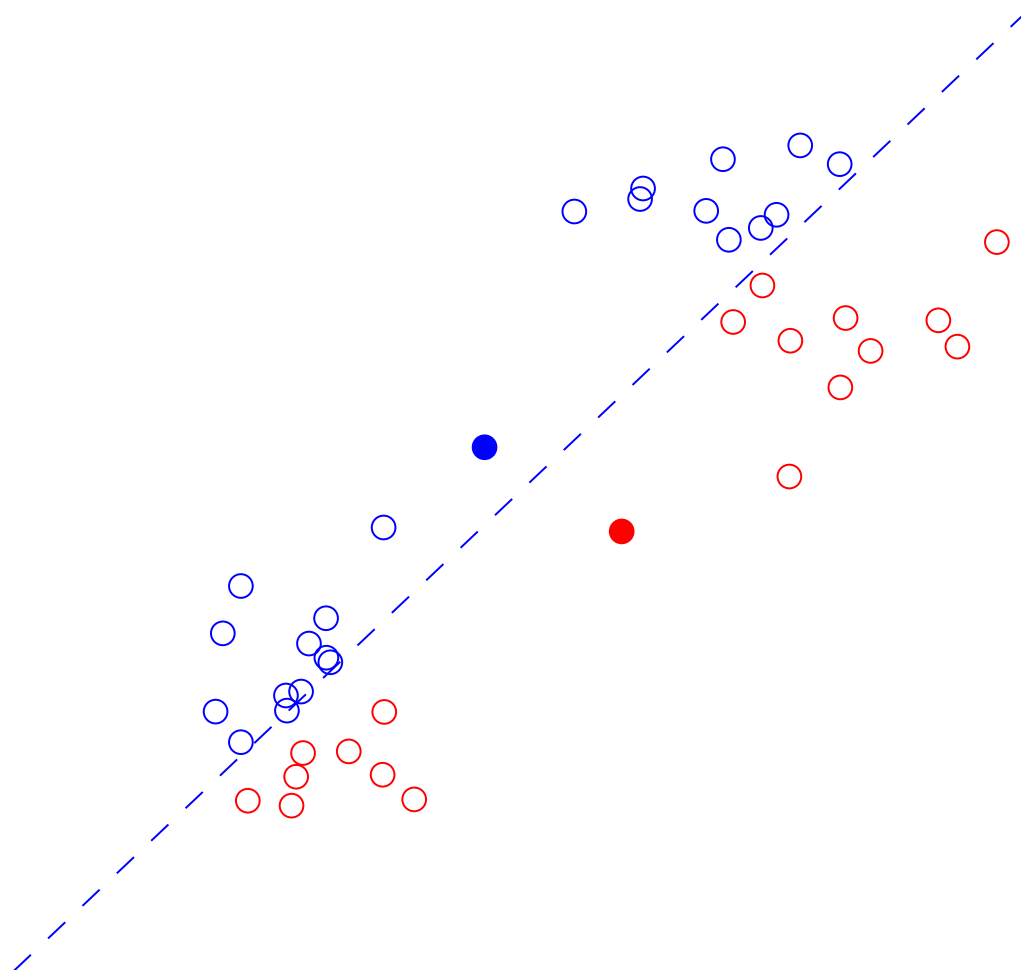
Находим ближайшие точки:



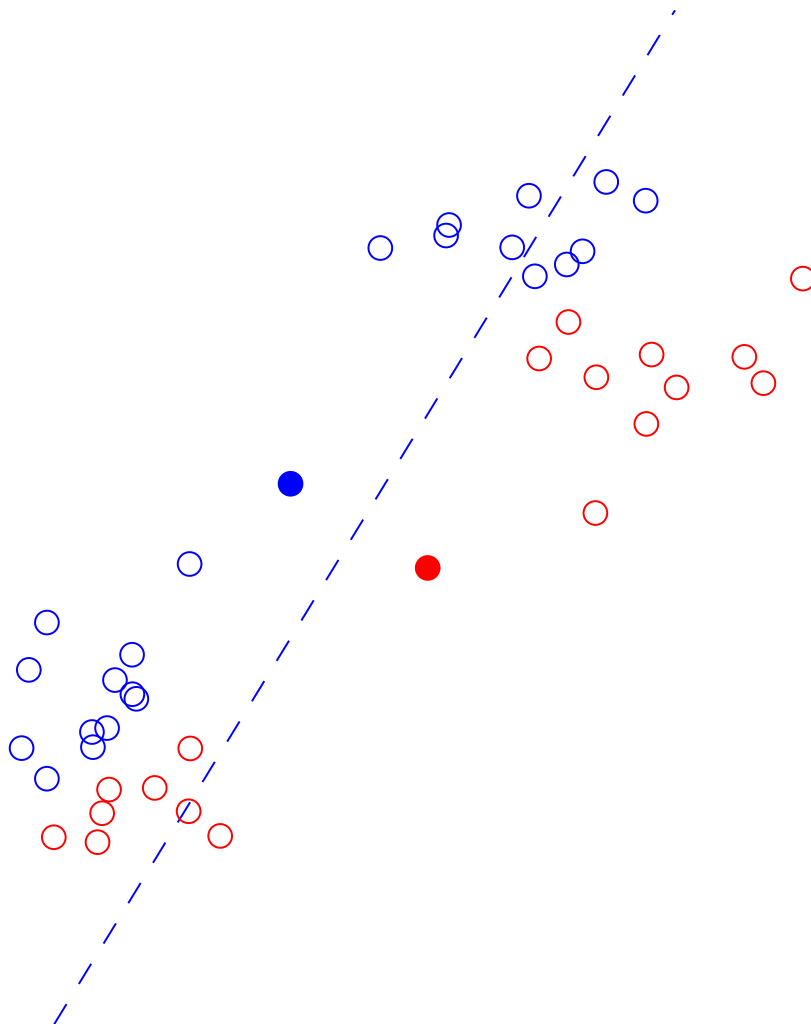
Находим ближайшие точки:



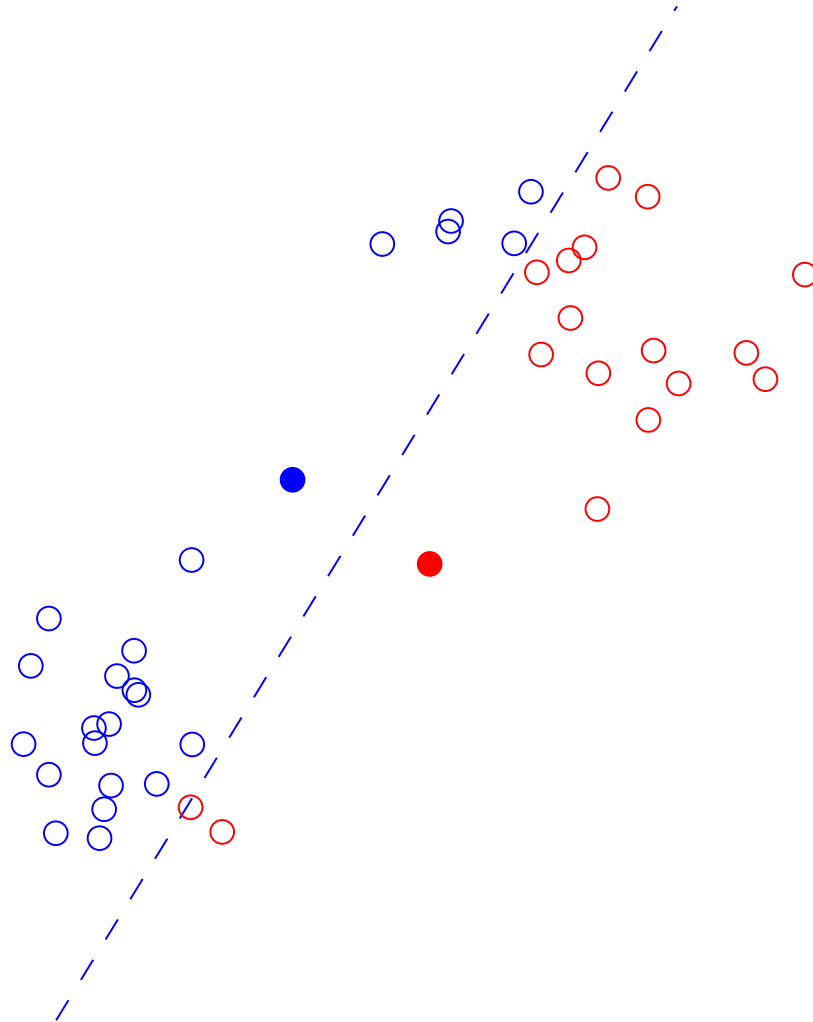
Вычисляем центры кластеров:



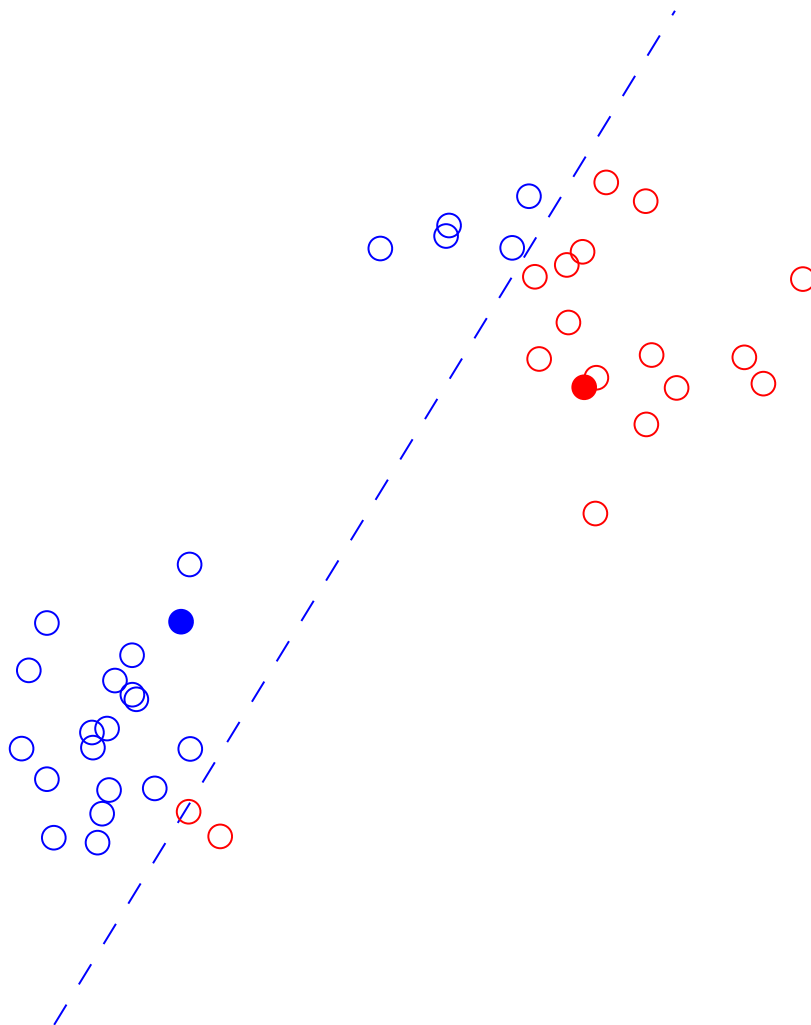
Находим ближайшие точки:



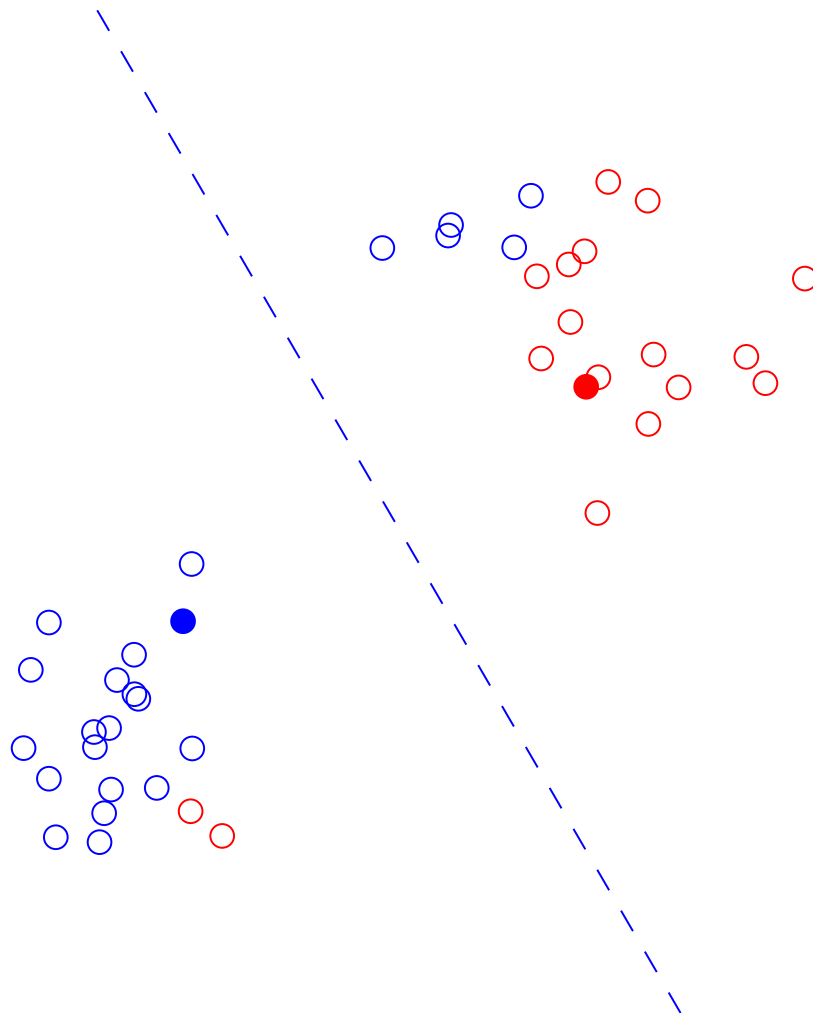
Находим ближайшие точки:



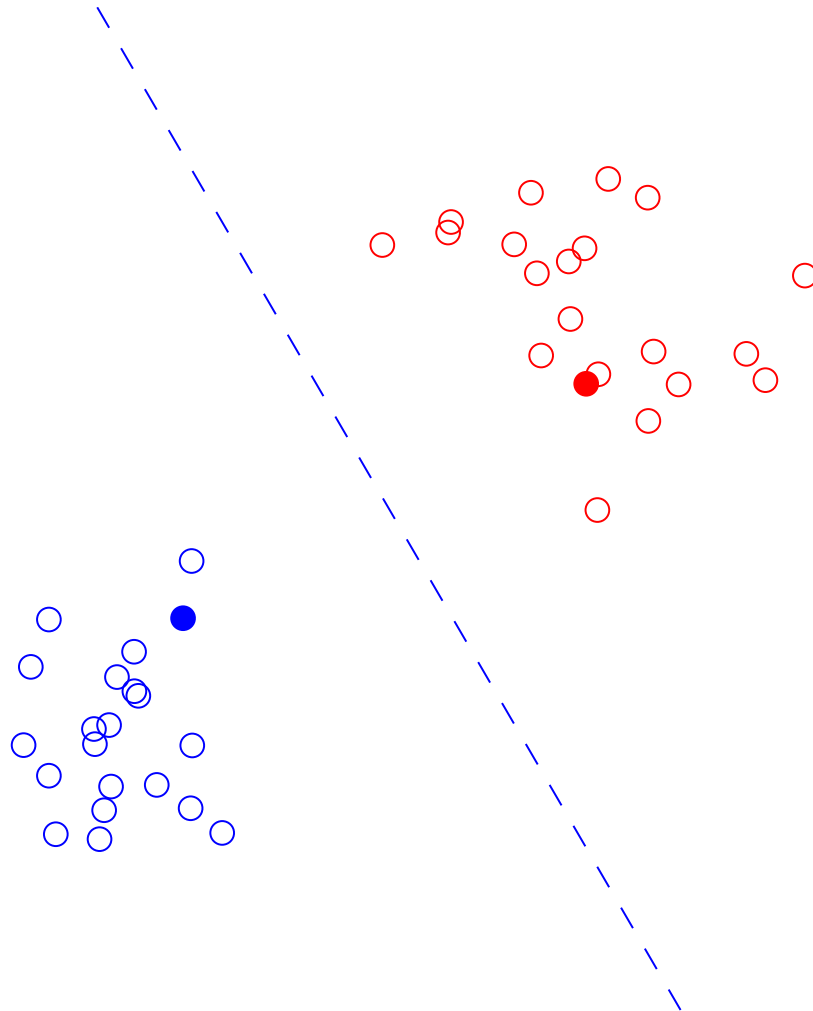
Вычисляем центры кластеров:



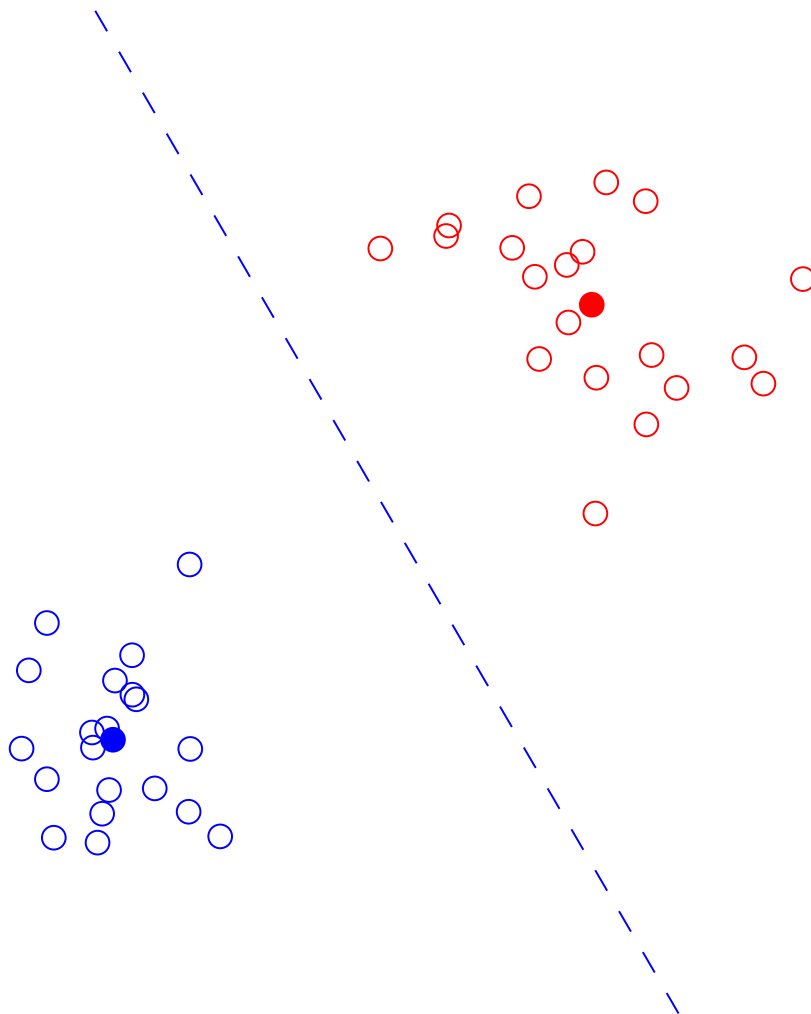
Находим ближайшие точки:



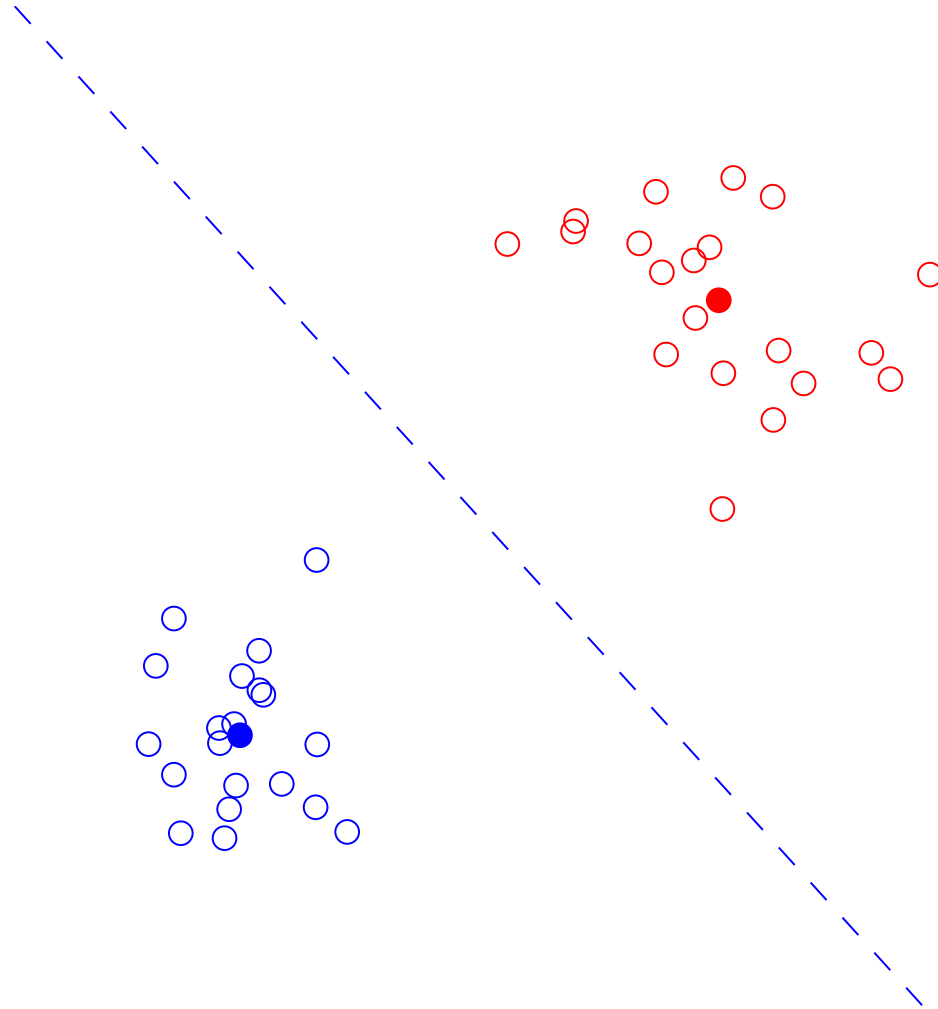
Находим ближайшие точки:



Вычисляем центры кластеров:



Находим ближайшие точки:



Пример 2. Анализ данных, полученных с биочипов

Пример 2. Анализ данных, полученных с биочипов

Биочип, или *микроэrray*, (biochip, microarray) — это миниатюрный прибор, измеряющий уровень экспрессии генов в имеющемся материале.

Пример 2. Анализ данных, полученных с биочипов

Биочип, или *микроэrray*, (biochip, microarray) — это миниатюрный прибор, измеряющий уровень экспрессии генов в имеющемся материале.

Экспрессия — это процесс перезаписи информации с гена на РНК, а затем на белок.

Пример 2. Анализ данных, полученных с биочипов

Биочип, или *микроэrray*, (biochip, microarray) — это миниатюрный прибор, измеряющий уровень экспрессии генов в имеющемся материале.

Экспрессия — это процесс перезаписи информации с гена на РНК, а затем на белок.

Количество и даже свойства получаемого белка зависят не только от гена, но также и от различных внешних факторов (например, от введенного лекарства).

Пример 2. Анализ данных, полученных с биочипов

Биочип, или *микроэrray*, (biochip, microarray) — это миниатюрный прибор, измеряющий уровень экспрессии генов в имеющемся материале.

Экспрессия — это процесс перезаписи информации с гена на РНК, а затем на белок.

Количество и даже свойства получаемого белка зависят не только от гена, но также и от различных внешних факторов (например, от введенного лекарства).

Таким образом, уровень экспрессии — это мера количества генерируемого белка (и скорости его генерирования).

Пример 2. Анализ данных, полученных с биочипов

Биочип, или *микроэrray*, (biochip, microarray) — это миниатюрный прибор, измеряющий уровень экспрессии генов в имеющемся материале.

Экспрессия — это процесс перезаписи информации с гена на РНК, а затем на белок.

Количество и даже свойства получаемого белка зависят не только от гена, но также и от различных внешних факторов (например, от введенного лекарства).

Таким образом, уровень экспрессии — это мера количества генерируемого белка (и скорости его генерирования).

На биочип кроме исследуемого материала помещается также «контрольный» генетический материал.

Пример 2. Анализ данных, полученных с биочипов

Биочип, или *микроэrray*, (biochip, microarray) — это миниатюрный прибор, измеряющий уровень экспрессии генов в имеющемся материале.

Экспрессия — это процесс перезаписи информации с гена на РНК, а затем на белок.

Количество и даже свойства получаемого белка зависят не только от гена, но также и от различных внешних факторов (например, от введенного лекарства).

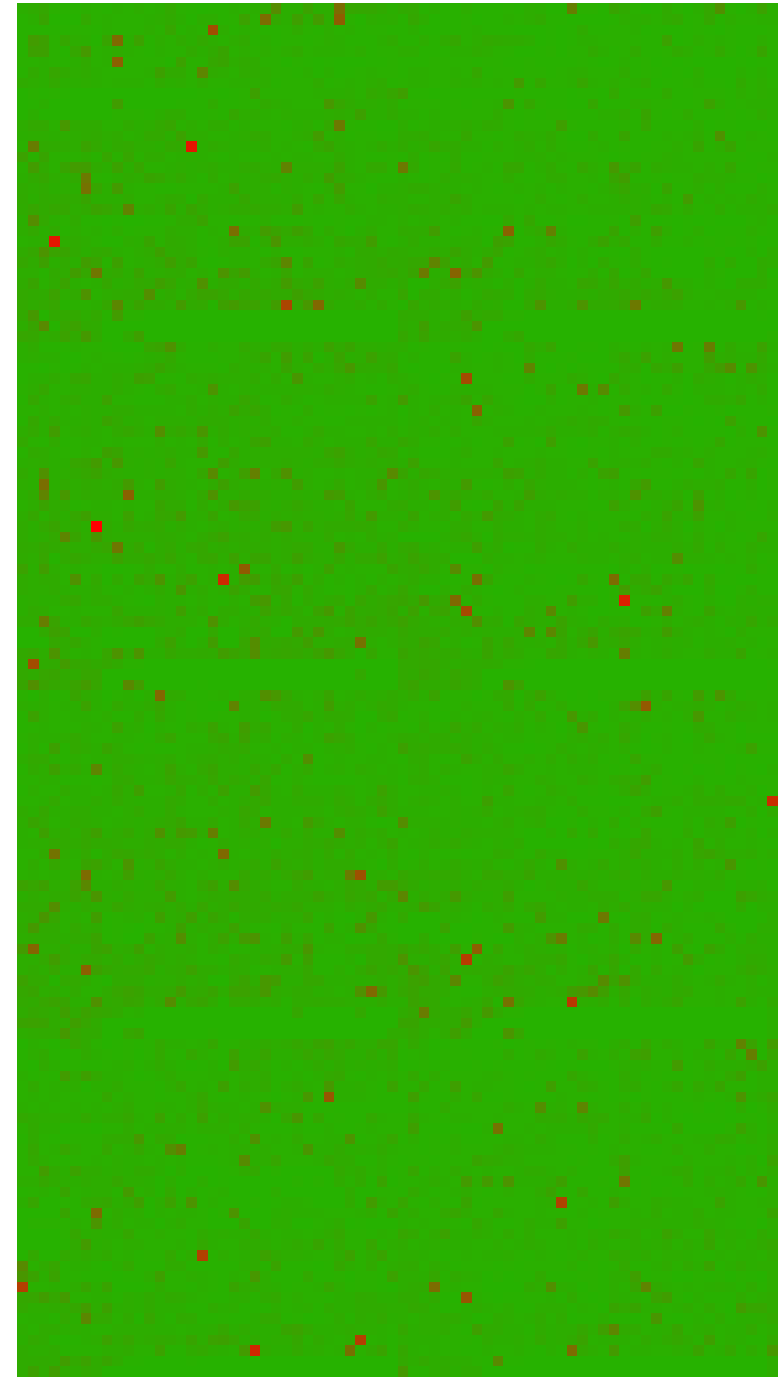
Таким образом, уровень экспрессии — это мера количества генерируемого белка (и скорости его генерирования).

На биочип кроме исследуемого материала помещается также «контрольный» генетический материал.

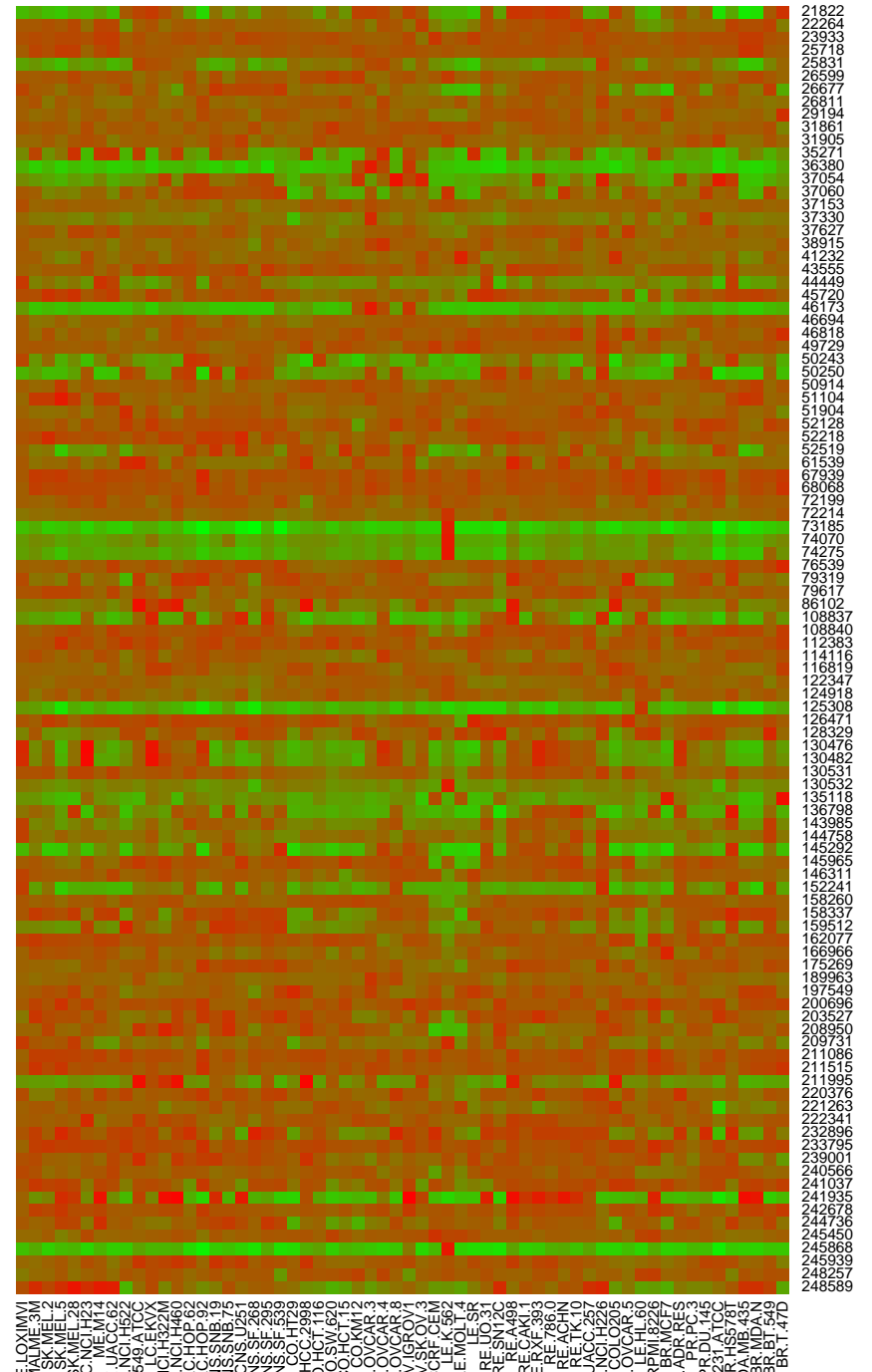
Положительные значения (красный цвет) — увеличение уровня экспрессии по сравнению с контрольным.

Отрицательные значения (зеленый цвет) — уменьшение.

Условное изображение биочипа. Каждая точка на рисунке соответствует определенному гену. Всего анализируется $132 \times 72 = 9504$ гена. Brown, V.M., Ossadtchi, A., Khan, A.H., Yee, S., Lacan, G., Melega, W.P., Cherry, S.R., Leahy, R.M., and Smith, D.J.; Multiplex three dimensional brain gene expression mapping in a mouse model of Parkinson's disease; *Genome Research* 12(6): 868-884 (2002).



Данные для 60 экспериментов с биочипом
<http://discover.nci.nih.gov/datasetsNature2000.jsp>
 Строки соответствуют генам, столбцы — экспериментам.
 Приведены только первые 100 строк (из общего числа 1375).
 Строки, содержащие отсутствующие значения, исключены.



Поставим следующие задачи:

Поставим следующие задачи:

- (a) Найти гены, показавшие высокую экспрессию, в заданных экспериментах.
т. е. найти наиболее красные клетки в заданных столбцах.

Поставим следующие задачи:

- (а) Найти гены, показавшие высокую экспрессию, в заданных экспериментах.
т. е. найти наиболее красные клетки в заданных столбцах.
- (б) Разбить гены на группы в зависимости от влияния на них экспериментов. Гены, реагирующие «почти одинаковым» образом в «большом» числе экспериментов, должны попасть в одну группу. Гены, реагирующие по-разному, должны находиться в разных группах.
т. е. разбить строки на группы (кластеры) «похожих» между собой строк

Поставим следующие задачи:

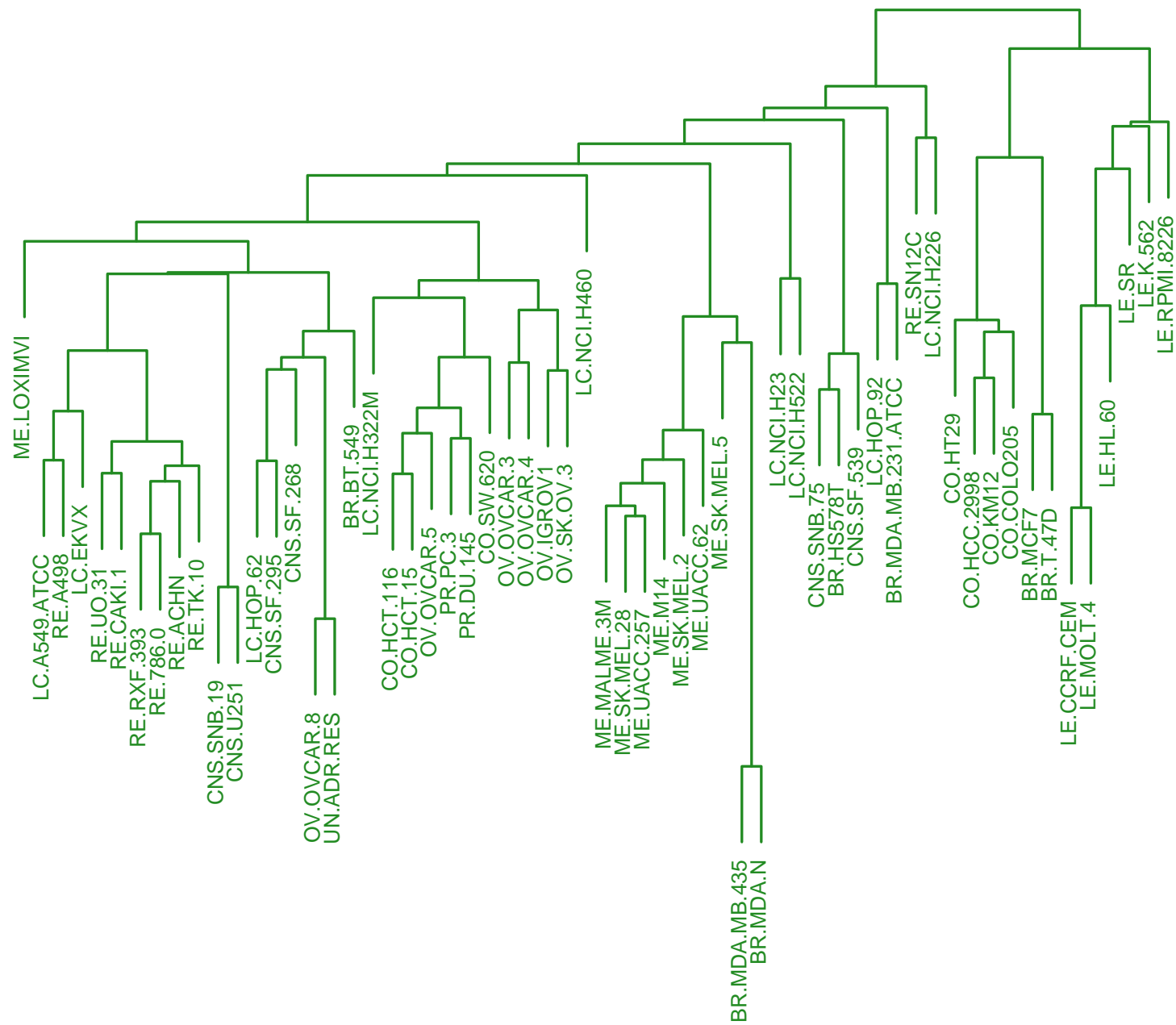
- (а) Найти гены, показавшие высокую экспрессию, в заданных экспериментах.
т. е. найти наиболее красные клетки в заданных столбцах.
- (б) Разбить гены на группы в зависимости от влияния на них экспериментов. Гены, реагирующие «почти одинаковым» образом в «большом» числе экспериментов, должны попасть в одну группу. Гены, реагирующие по-разному, должны находиться в разных группах.
т. е. разбить строки на группы (кластеры) «похожих» между собой строк
- (в) Разбить эксперименты на группы в зависимости от их влияния на гены. Эксперименты, в которых одинаковые гены реагировали «сходным» образом должны оказаться в одной группе. Эксперименты, в которых гены реагировали «различно», должны находиться в разных группах.
т. е. разбить столбцы на группы (кластеры) «похожих» между собой строк

Поставим следующие задачи:

- (а) Найти гены, показавшие высокую экспрессию, в заданных экспериментах.
т. е. найти наиболее красные клетки в заданных столбцах.
- (б) Разбить гены на группы в зависимости от влияния на них экспериментов. Гены, реагирующие «почти одинаковым» образом в «большом» числе экспериментов, должны попасть в одну группу. Гены, реагирующие по-разному, должны находиться в разных группах.
т. е. разбить строки на группы (кластеры) «похожих» между собой строк
- (в) Разбить эксперименты на группы в зависимости от их влияния на гены. Эксперименты, в которых одинаковые гены реагировали «сходным» образом должны оказаться в одной группе. Эксперименты, в которых гены реагировали «различно», должны находиться в разных группах.
т. е. разбить столбцы на группы (кластеры) «похожих» между собой строк

Задачи (б) и (в) — это задачи кластерного анализа.

Таксономия 60 клеток на основе анализа уровня экспрессии их генов (задача (в))



Пример 3. Списки Сводеша и таксономия языков

Пример 3. Списки Сводеша и таксономия языков

Список Сводеша (Morris Swadesh, 1909–1967) — список из слов *базового словаря* — *ядра* языка (термины родства, части тела, частые природные явления, животные и т.д.):

мать, отец, брат, сестра, человек, рука, нога, солнце, луна, . . .

Это самая старая лексика, она менее всего подвержена изменениям и заимствованиям.

Эти слова тоже могут заменяться другими, но с меньшей вероятностью.

Примеры:

око → глаз, чело → лоб, перст → палец, уста → рот, гад → змея, дитя → ребёнок,
пёс → собака, перси → грудь

Есть редакции из 100, 200, 207 понятий.

№	Русский	Английский	Немецкий	Итальянский	Французский	Чешский
1	я	I	ich	io	je	já
2	ты	you	du	tu	tu	ty
3	он	he	er	lui	il	on
4	мы	we	wir	noi	nous	my
5	вы	you	ihr	voi	vous	vy
6	они	they	sie	loro	ils	oni
7	этот	this	dieses	questo	ceci	tento
8	тот	that	jenes	quello	cela	tamten
9	здесь	here	hier	qui	ici	zde
10	там	there	dort	lá	lá	tam
11	кто	who	wer	chi	qui	kdo
12	что	what	was	che	quoi	co
13	где	where	wo	dove	où	kde
14	когда	when	wann	quando	quand	kdy
15	как	how	wie	come	comment	jak
16	не	not	nicht	non	ne. . . pas	ne
.....						
205	если	if	wenn	se	si	jestlize
206	потому что	because	weil	perché	parce que	protoze
207	имя	name	Name	nome	nom	jméno

Близость двух языков можно измерять по количеству родственных слов (*когнат*) — однокоренных слов, имеющих общее происхождение и близкое звучание.

Являются ли два слова родственными — определяют лингвисты (а не фоменки с задорновыми): родственны ли слова

год и *рік*, *цвeток* и *квітка*, *видеть* и *бачити*

или

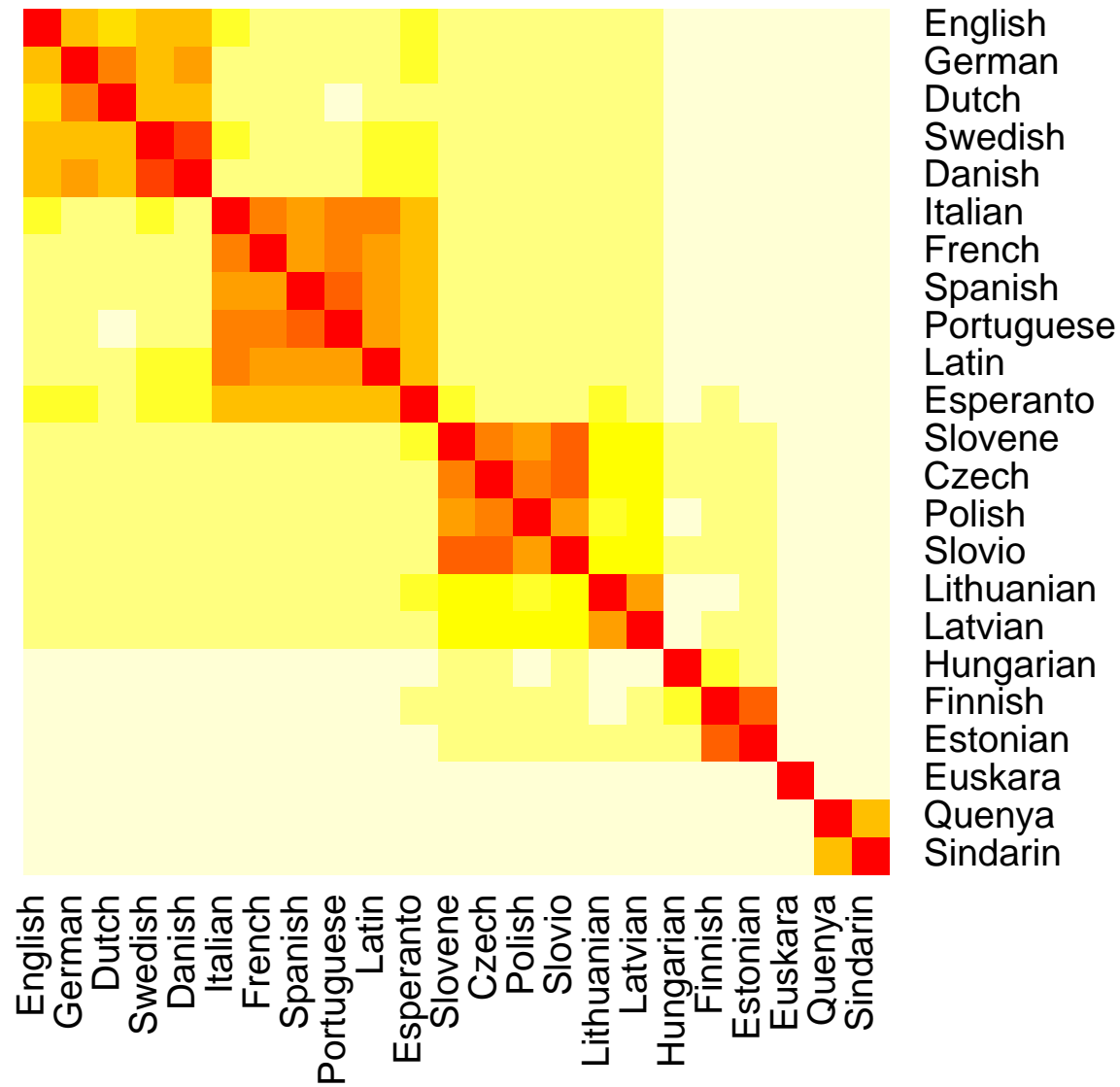
колесо и *wheel* и चक्र ?

Можно разбивать языки на группы близких друг другу языков — задача кластерного анализа.

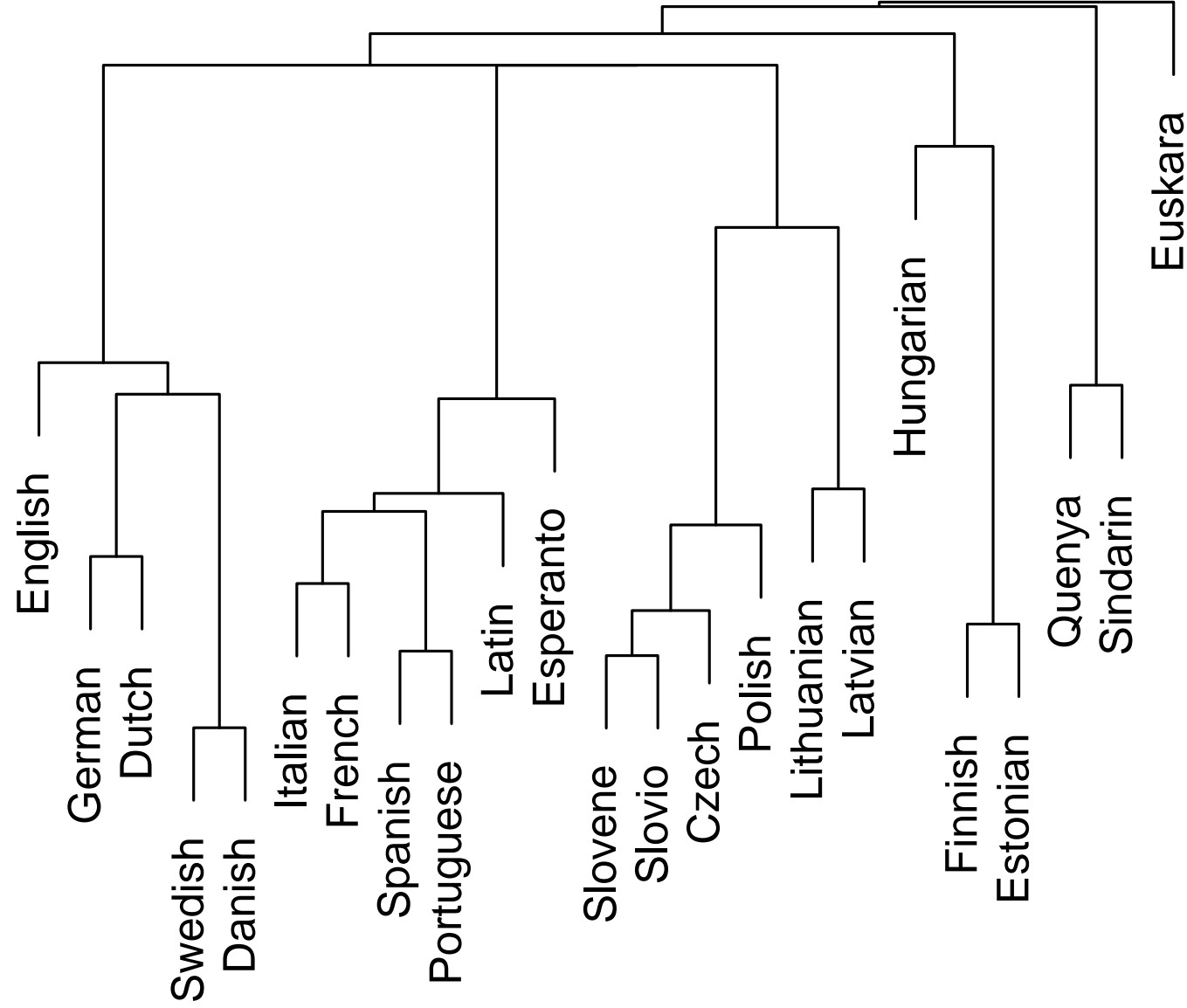
Более того, на основе анализа списка Сводеша для двух родственных языков можно приблизительно установить время их появления из единого пра-языка.

Считается, что в 100-словном списке сохраняется за тысячелетие около 86% слов, а в 200-словном в среднем 81% слов, соответственно. Отсюда «период полураспада» языкового «ядра» — для 100- и 200-словного списка равен соответственно 4.6 и 3.3 тыс. лет. (это один из методов *глоттохронологии*)

Матрица сходства между некоторыми языками, построенная на основе списков Сводеша.



Дерево иерархической кластеризации для 23 языков, построенное на основе списков Сводеша.



КОНЕЦ