

# **МАШИННОЕ ОБУЧЕНИЕ И АНАЛИЗ ДАННЫХ**

**(Machine Learning and Data Mining)**

Н. Ю. Золотых

<http://www.uic.unn.ru/~zny/ml>



*Глава 1*

## **Постановки и примеры задач**

## 1.1. Что такое машинное обучение?

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, New York. 1956. V. 4. P. 2099–2123; рус. перев.: *A. M. 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 мере качества  $P$ , если качество решения задач из  $T$ , измеренное на основе  $P$ , улучшается с приобретением опыта  $E$ .

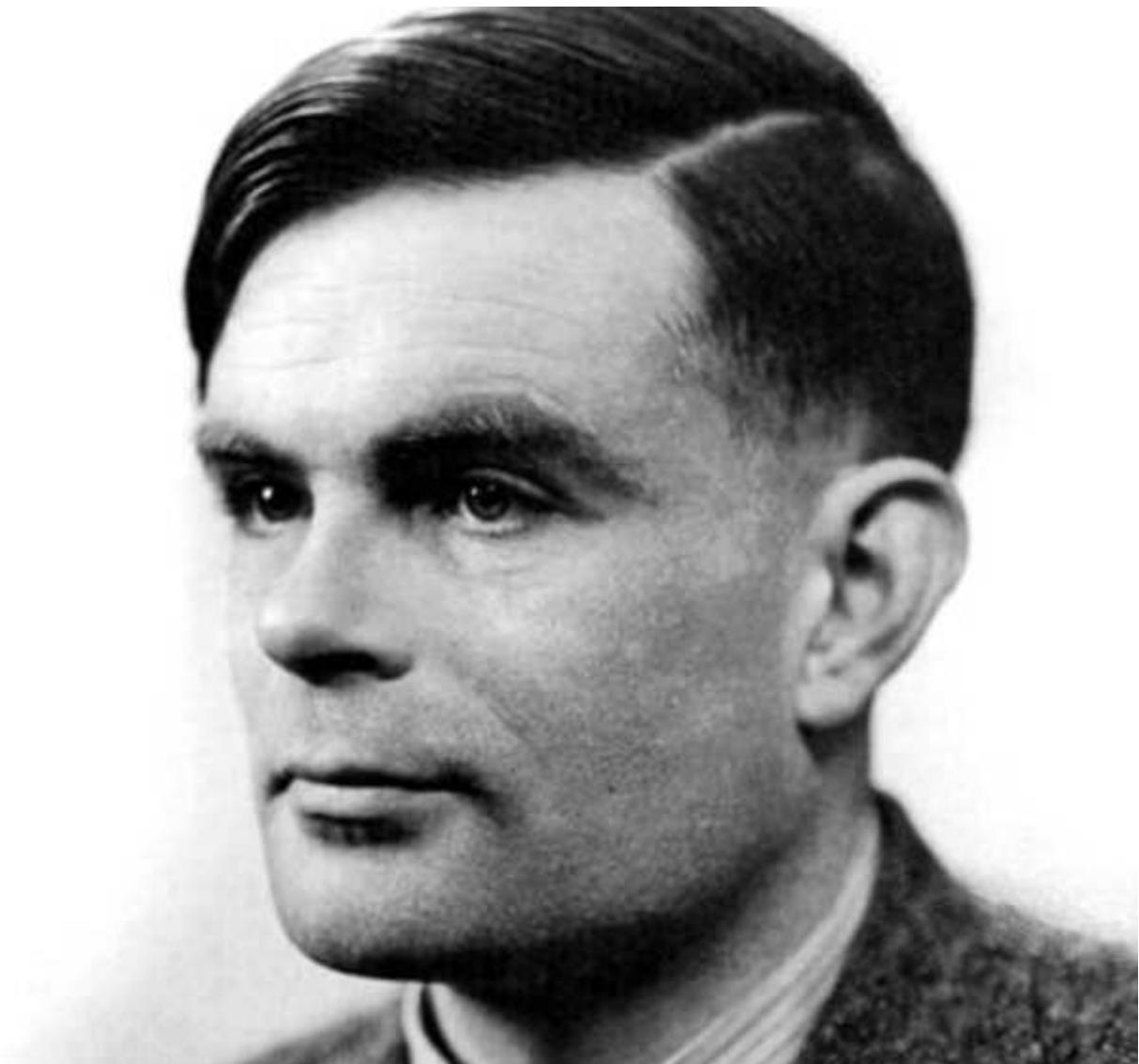
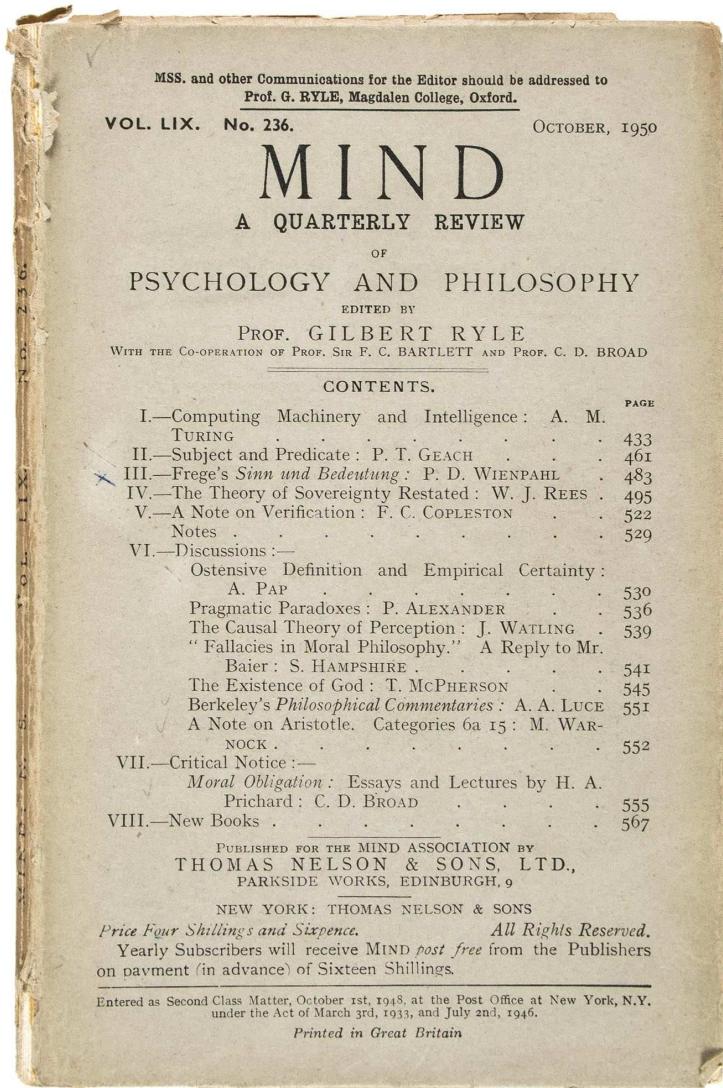
*T.M. Mitchell Machine Learning*. McGraw-Hill, 1997.

- На практике фаза обучения может предшествовать фазе работы алгоритма (например, детектирование лиц на снимке) — batch learning
- или обучение может проходить в процессе функционирования алгоритма (например, определение почтового спама) — online learning.

Например,

- Программа распознавания рукописных символов, после предъявления ей серии таких символов с правильными ответами, начинает распознавать точнее.
- Программа игры в шахматы после серии проведенных игр начинает играть лучше.
- Распознавание спама после обучения на примерах происходит точнее.

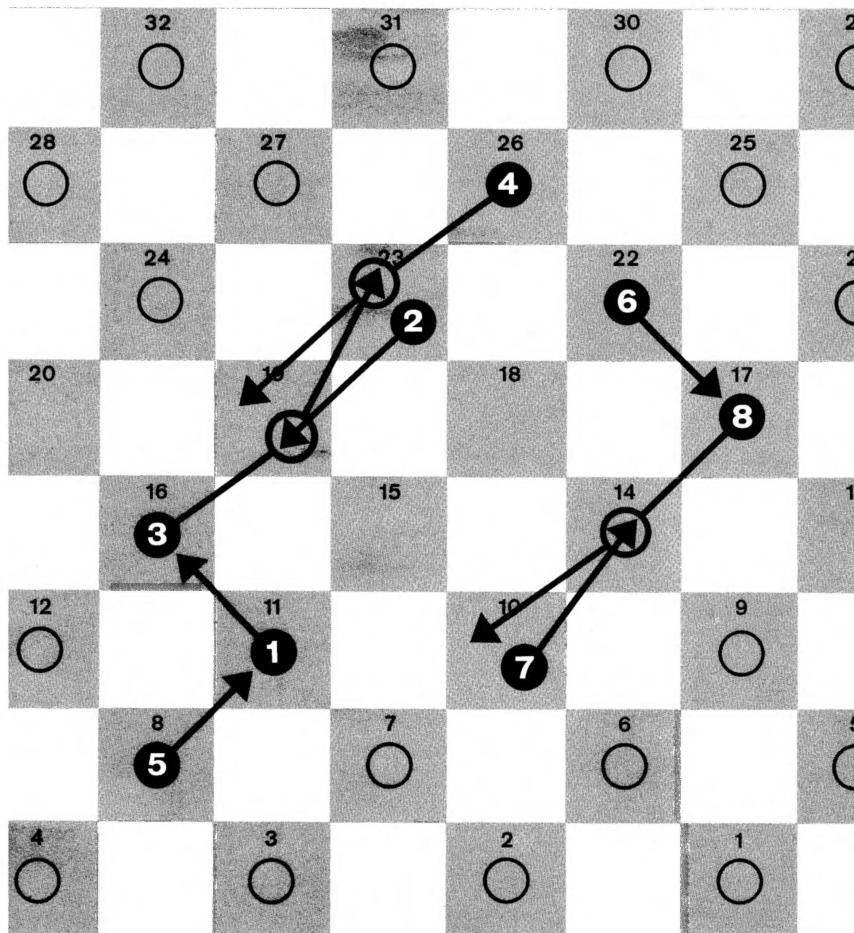
# Alan Mathison Turing (1912–1954)



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,<sup>1</sup> 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.<sup>2</sup> 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.<sup>3</sup> 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,<sup>4-7</sup> 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.

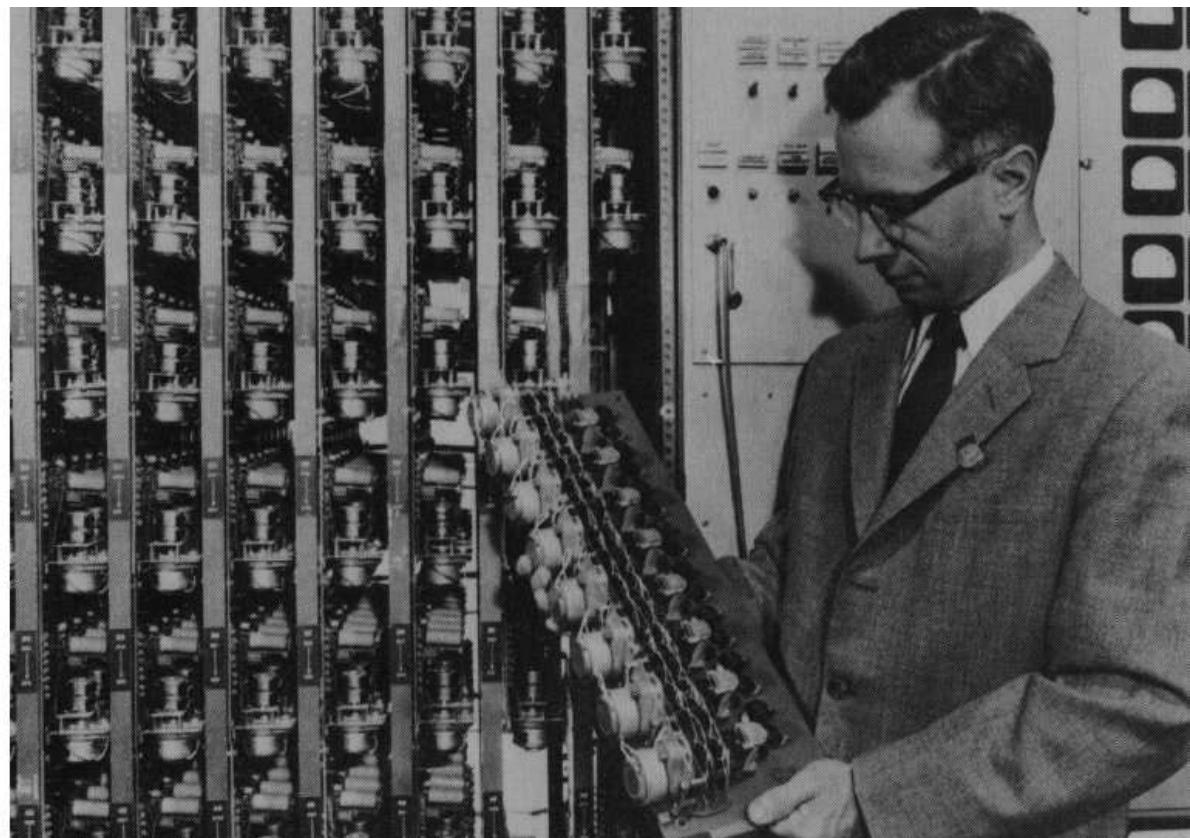
Some of these characteristics might well be enumerated. They are:

- (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 персептрана – 1957 г.

Первый нейронный компьютер – MARK 1 – 1958 г.



## *Н. Н. Носов «Незнайка в Солнечном городе», 1958*

Быстро приблизившись, автомобиль остановился у тротуара, и дверцы его открылись.

– Где же водитель? – с недоумением спросил Незнайка, заметив, что водителя за рулем не было.

– А водителя и не нужно, – ответил Кубик. – Это автоматическая кнопочная машина. Вместо водителя здесь, как видите, расположены кнопки с названиями улиц и остановок. Вы нажимаете нужную кнопку, и машина сама везет вас куда надо.

Все сели в машину. Кубик сказал:

– Вот смотрите, я нажимаю кнопку, где написано: «Архитектурная улица», и . . .

Он нажал одну из кнопок на щитке приборов, и . . . машина тронулась с места.

– Стойте, что вы делаете? – закричал Пестренький, хватая Кубика за руку. – А вдруг машина наедет на кого-нибудь?

– Машина не может ни на кого наехать, потому что в ней имеется ультразвуковое локаторное устройство, при помощи которого предотвращается возможность какого бы то ни было наезда или столкновения, – сказал Кубик. – Обратите внимание на два больших рупора, которые установлены впереди. Один рупор все время посыпает вперед ультразвуковые сигналы. Как только впереди появляется какое-нибудь препятствие, ультразвуковые сигналы начинают отражаться, то есть как бы отскакивать от него обратно, и попадают во второй рупор. Здесь ультразвуковая энергия преобразуется в электрическую. Электрическая же энергия включает тормоз или механизм поворота. Если препятствие небольшое, машина его обьедет, так как включится механизм поворота; если большое – остановится, потому что включится тормоз. Такие же рупоры имеются у машины сзади и по бокам, для того чтобы ультразвуковые сигналы могли посыпаться во все стороны. . .

- А какие это ультразвуковые сигналы? – спросил Незнайка.
- Это... как бы вам сказать... такие очень тоненькие звуки, что мы с вами их даже слышать не можем, но они все-таки обладают энергией, как и те звуки, которые мы слышим.

В это время машина подъехала к перекрестку и остановилась у светофора.

- В машине также имеется оптическое устройство, которое включает тормоз при красном светофоре, – сказал Кубик.

Автомобиль действительно неподвижно стоял перед светофором до тех пор, пока не погас красный свет и не включился зеленый.

- Что ж, в этом ничего удивительного нет, – сказал Пестренький. – Удивительно только, откуда машина знает, куда надо ехать.

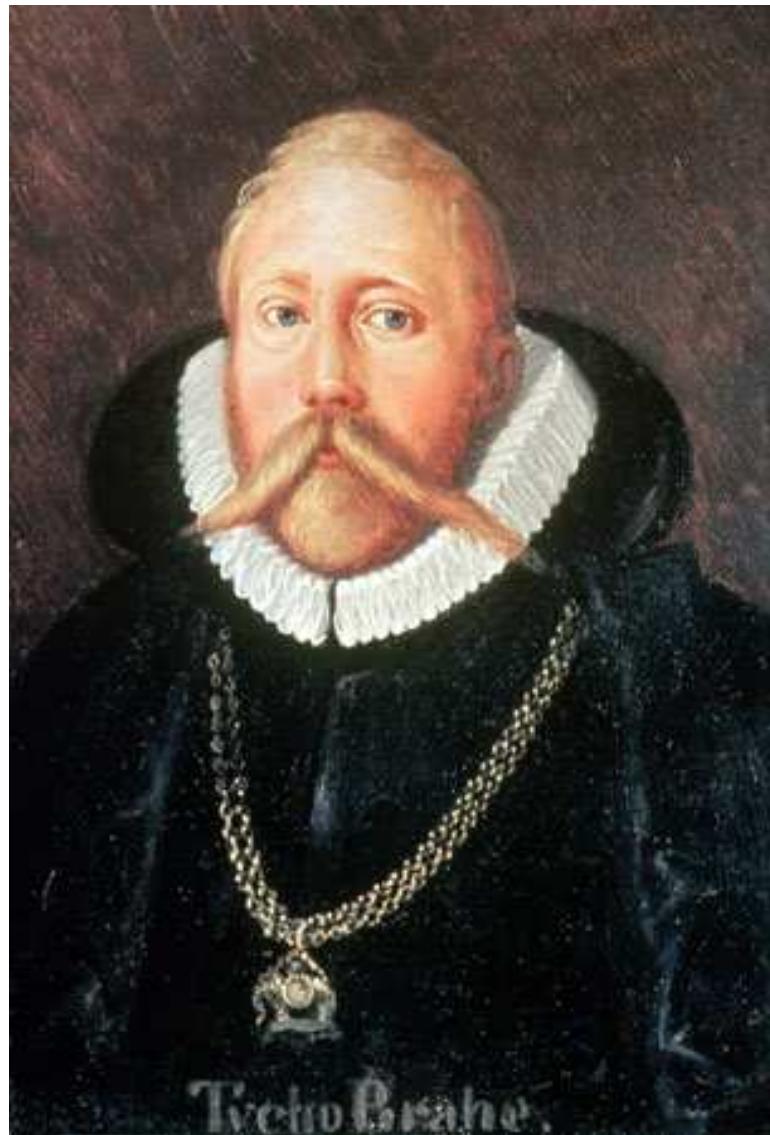
– Машина, безусловно, ничего знать не может, – ответил Кубик. – Но все же она отвезет вас куда надо, после того как вы нажмете кнопку, потому что в механизме имеется так называемое электронное запоминающее устройство. Запоминающим это устройство называется потому, что машина как бы запоминает маршруты, по которым ездит. **Каждый новый автомобиль, оборудованный этим устройством, первое время ездит с водителем и проходит как бы курс обучения.** Начиная такие учебные поездки, водитель обычно нажимает кнопку с названием какой-нибудь улицы, после чего ведет машину на эту улицу, потом нажимает кнопку с названием другой улицы и ведет машину на другую улицу. Рулевое управление автомобиля связано с электронным запоминающим устройством, поэтому когда в следующий раз нажимают кнопку, то электронное устройство само направляет автомобиль по заданному маршруту, и машина может ехать совсем без водителя.

- Ну, если так, то действительно ничего удивительного нет, – сказал Пестренький. – Вот если бы никакого устройства не было, а машина сама везла нас куда надо – это было бы удивительно.

## **1.2. Что такое (интеллектуальный) анализ данных (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]



Тихо Браге (1546–1601) и Иоганн Кеплер (1571–1630)

## 1.2.1. Анализ данных (data mining) vs машинное обучение (machine learning)

*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.

## 1.2.2. Смежные и близкие области

- Pattern Recognition (распознавание образов)

$$\text{Pattern Recognition} \subset \text{Machine Learning}$$

Хотя иногда считают, что

$$\text{Pattern Recognition} \approx \text{Machine Learning}$$

- Data Mining (интеллектуальный анализ данных)

Две точки зрения:

- Data Mining и Machine Learning имеют дело с различными *содержательными* постановками задач, но методы у них в целом совпадают.

$$\text{Data Mining} \cap \text{Machine Learning} \neq \emptyset$$

- Data Mining имеет дело с содержательными задачами, а Machine Learning – с математической теорией, отсюда немного странные термины, например, «алгоритмы машинного обучения в анализе данных»

- Artificial Intelligence (искусственный интеллект)

$$\text{Machine Learning} \subset \text{Artificial Intelligence}$$

### **1.2.3. Сфера приложения**

- Компьютерное зрение (computer vision)
- Распознавание речи (speech recognition)
- Компьютерная лингвистика и обработка естественных языков (natural language processing)
- Медицинская диагностика
- Биоинформатика
- Техническая диагностика
- Финансовые приложения
- Рубрикация, аннотирование и упрощение текстов
- Информационный поиск
- Интеллектуальные игры
- ...

## **1.2.4. Аппарат**

Теория вероятностей и математическая статистика

Линейная алгебра

Методы оптимизации

Численные методы

Математический анализ

Дискретная математика

и др.

## **1.2.5. Дедуктивное и индуктивное обучения**

У людей обучение происходит в результате ознакомления с правилами, теориями, инструкциями и т. п. и/или на основе опыта (собственного или чужого).

По аналогичному принципу можно выделить различные способы обучения и в искусственных системах:

- *Дедуктивное, или аналитическое, обучение* (экспертные системы).

Имеются знания, сформулированные экспертом и как-то формализованные.

Программа выводит из этих правил конкретные факты и новые правила.

- *Индуктивное обучение* ( $\approx$  статистическое обучение).

На основе эмпирических данных программа строит общее правило.

Эмпирические данные могут быть получены самой программой в предыдущие сеансы ее работы или просто предъявлены ей.

(Определение Митчелла относится только к такому обучению)

- *Комбинированное обучение.*

В курсе рассматривается только *индуктивное обучение* (обучение «на опыте»).

## 1.2.6. Классификация задач индуктивного обучения

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

В курсе рассматривается *обучение с учителем* и *обучение без учителя*.

## 1.3. Обучение с учителем

Множество  $\mathcal{X}$  – *объекты, примеры, входы* (samples)

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

Имеется некоторая зависимость (детерминированная или вероятностная), позволяющая по  $x \in \mathcal{X}$  предсказать (или оценить вероятность появления)  $y \in \mathcal{Y}$ .  
(в частности, если зависимость детерминированная, то существует  $f^* : \mathcal{X} \rightarrow \mathcal{Y}$ )

Зависимость известна только на объектах из *обучающей выборки*:

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

Пара  $(x^{(i)}, y^{(i)}) \in \mathcal{X} \times \mathcal{Y}$  – *прецедент*.

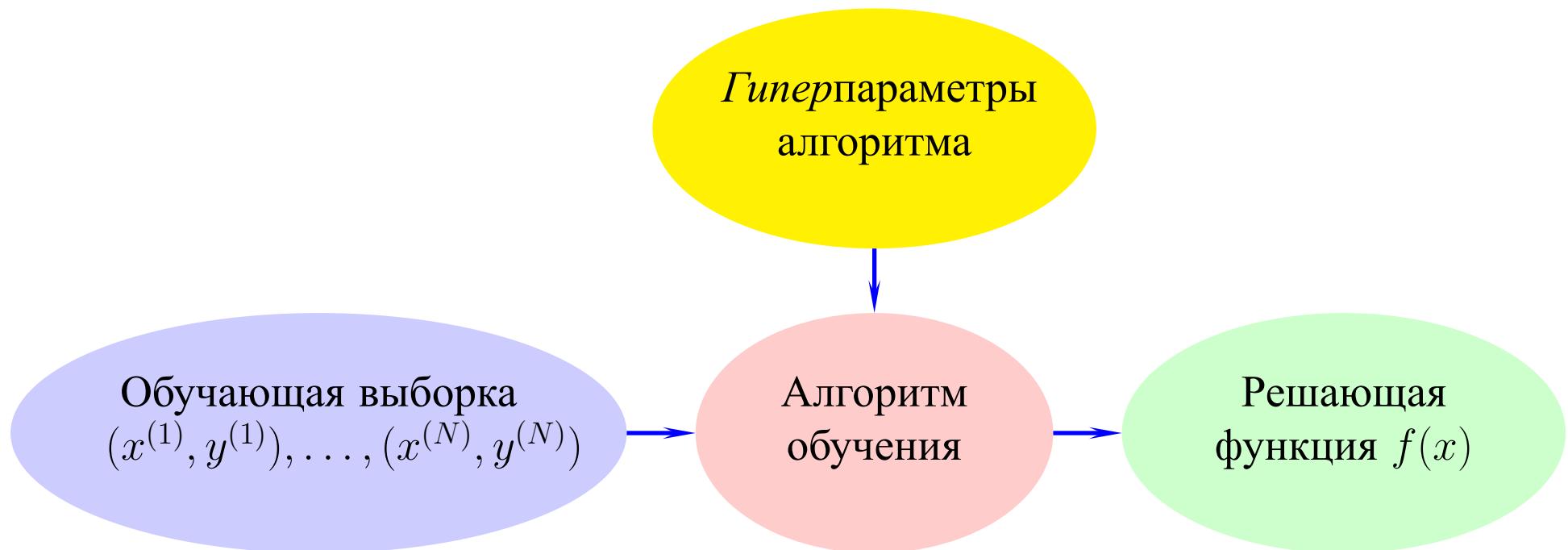
Задача *обучения с учителем*: восстановить зависимость, т. е. найти функцию  $f : \mathcal{X} \rightarrow \mathcal{Y}$ , предсказывающую по  $x \in \mathcal{X}$  ответы  $y \in \mathcal{Y}$ .

т. е. для детерминированной зависимости найти  $f(x) \approx f^*(x)$ ,  $x \in \mathcal{X}$ .

Можем требовать не нахождения  $f(x)$ , а оценки вероятности  $\Pr(y|x)$

- 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
- ...

### 1.3.1. Схема обучения с учителем



### 1.3.2. 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,  $Q_j = \{\text{Alfa Romeo, Audi, BMW, \dots, Volkswagen}\}$   
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 (аппроксимации)
- $\mathcal{Y} = \{1, 2, \dots, K\}$  — задача классификации. Номер класса  $k \in \mathcal{Y}$

Компоненты  $x_j$  вектора  $x$  так же называют *входами* или *предикатными*  
*(объясняющими)* *переменными*.

В мат. статистике  $x_j$  называют «независимыми» переменными, а  $y$  — «зависимой».

Значения признаков объектов из обучающей выборке и соответствующие ответы обычно записывают в матрицы:

$$\mathbf{X} = \begin{pmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_d^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_d^{(2)} \\ \dots & \dots & \dots & \dots \\ x_1^{(N)} & x_2^{(N)} & \dots & x_d^{(N)} \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{pmatrix}$$

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

### 1.3.3. Классификация задач обучения с учителем

В зависимости от множества  $\mathcal{Y}$  выделяют разные типы задачи обучения.

- $\mathcal{Y}$  конечно, например,  $\mathcal{Y} = \{1, 2, \dots, K\}$ , — *задача классификации* (или *задача распознавания образов*):  
 $\mathcal{X}$  разбивается на  $K$  классов

$$\mathcal{X}_k = \{x \in \mathcal{X} : f^*(x) = k\} \quad (k = 1, 2, \dots, K).$$

По  $x$  требуется предсказать, какому классу он принадлежит.

- $\mathcal{Y} = \mathbf{R}$  — *задача восстановления регрессии*.

Требуется найти функцию  $f$  из определенного класса, которая аппроксимирует неизвестную зависимость.

Ситуация, когда  $y$  — вектор, сводится к несколькими задачам со скалярным (атомарным) выходом.

$y$  может быть чем-то более хитрым, например, графом, деревом, цепочкой символов (нефиксированной длины) — *структурное машинное обучение* (structured learning)

#### **1.3.4. Примеры практических задач**

##### **Медицинская диагностика**

По набору определенных характеристик пациента (симптомов), таких как температура тела, артериальное давление, содержание гемоглобина в крови и т. п., требуется определить, какое у больного заболевание (и болен ли он вообще).

Объектами являются пациенты, их признаком описанием — набор характеристик, а выходом — номер класса.

Могут встречаться признаки разных типов:

- бинарные (пол, наличие головной боли),
- номинальные (боль может быть тупой, режущей, колющей и т. п.),
- порядковые (состояние больного может быть удовлетворительным, средней тяжести, тяжелым, крайне тяжелым),
- количественные (температура тела, пульс, давление).

Имеются данные о 114 лицах с заболеванием щитовидной железы.

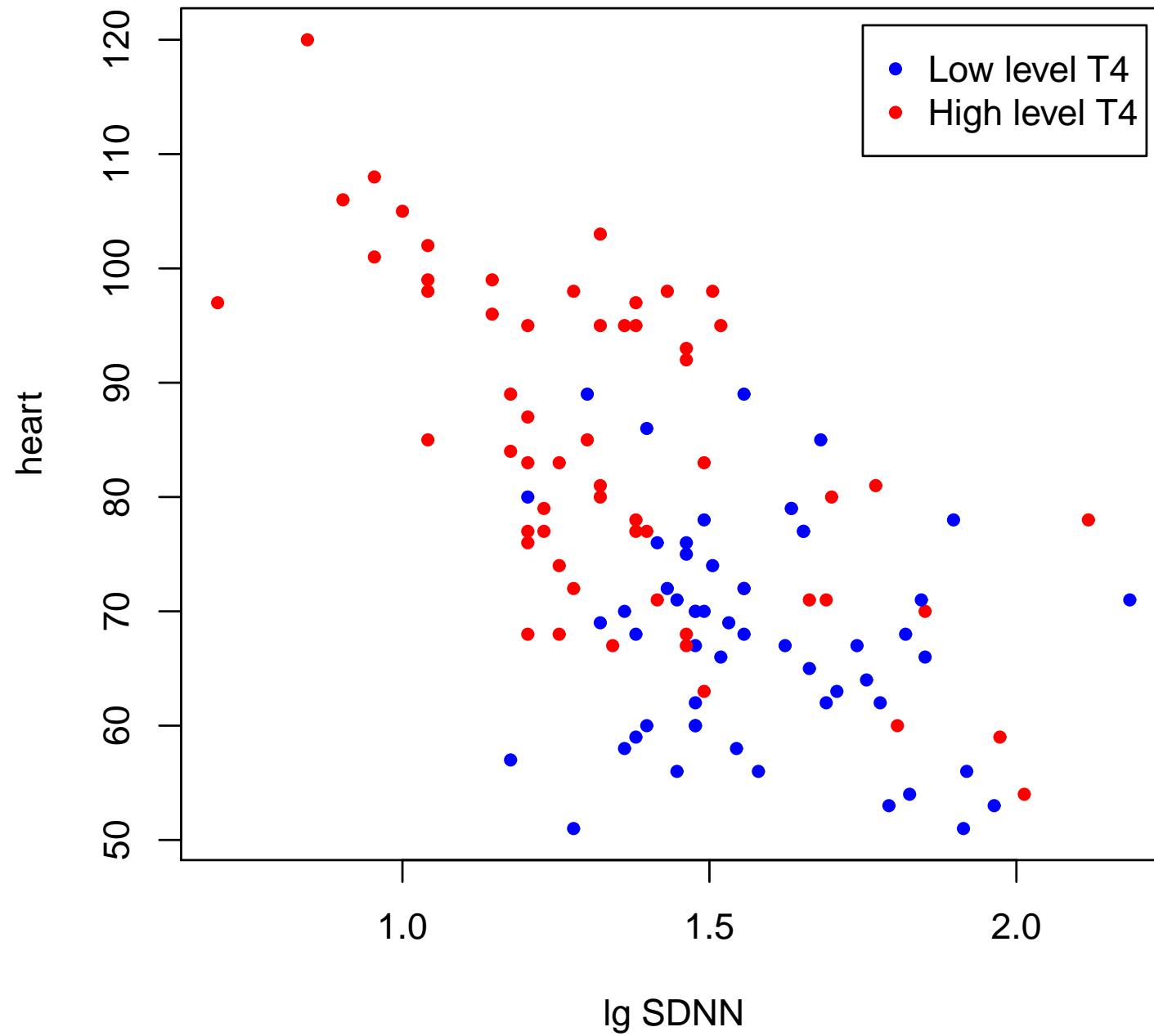
У 61 — повышенный уровень свободного гормона T4,

у 53 — уровень гормона в норме.

Признаки:

- heart — частота сердечных сокращений (пульс),
- SDNN — стандартное отклонение длительности RR-интервалов.

Можно ли научиться предсказывать (допуская небольшие ошибки) уровень свободного T4 по heart и SDNN у новых пациентов?



## **Пример** pima

Имеется информация о 768 пациентках

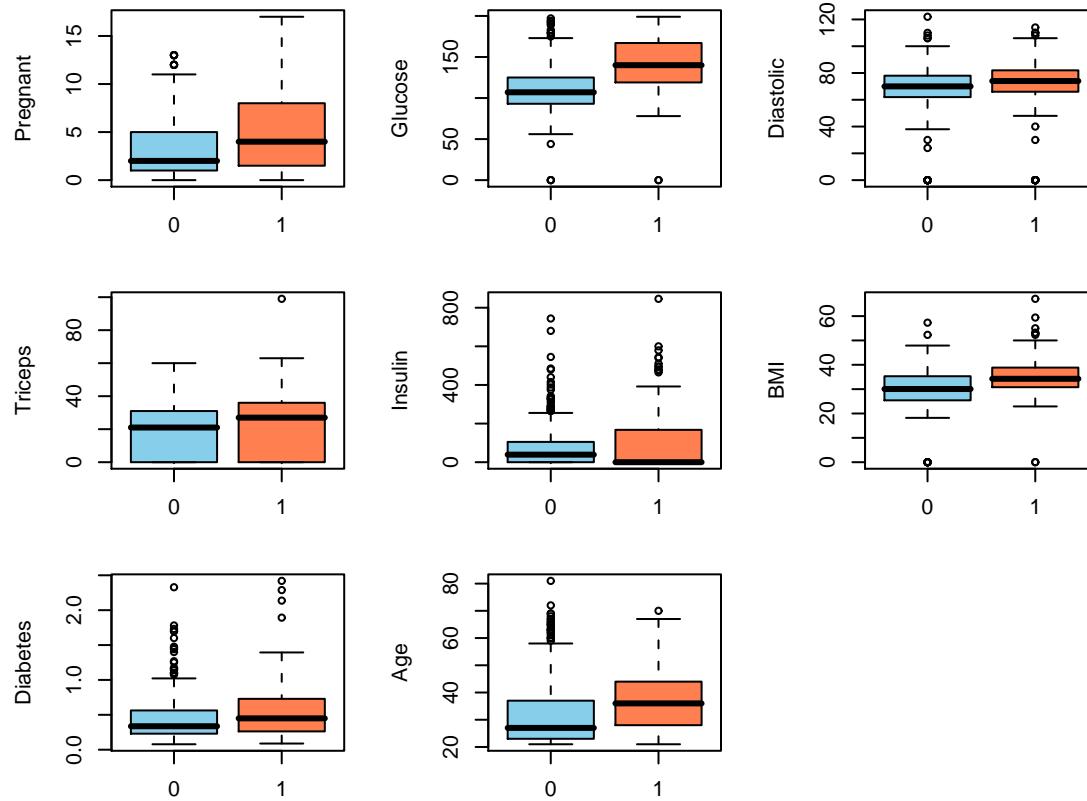
Все признаки количественные:

1. количество беременностей,
2. содержание глюкозы в крови,
3. кровяное давление,
4. толщина складки кожи на трицепсе,
5. 2-х часовой сывороточный инсулин,
6. индекс массы тела (масса/рост<sup>2</sup>),
7. диабетическая родословная функция,
8. возраст.

$$\mathcal{Y} = \{\text{есть диабет, нет диабета}\}$$

Имеем 768 точек в 8-мерном пространстве.

О расположении точек можно судить по 8 бокс-диаграммам («ящикам с усами»)

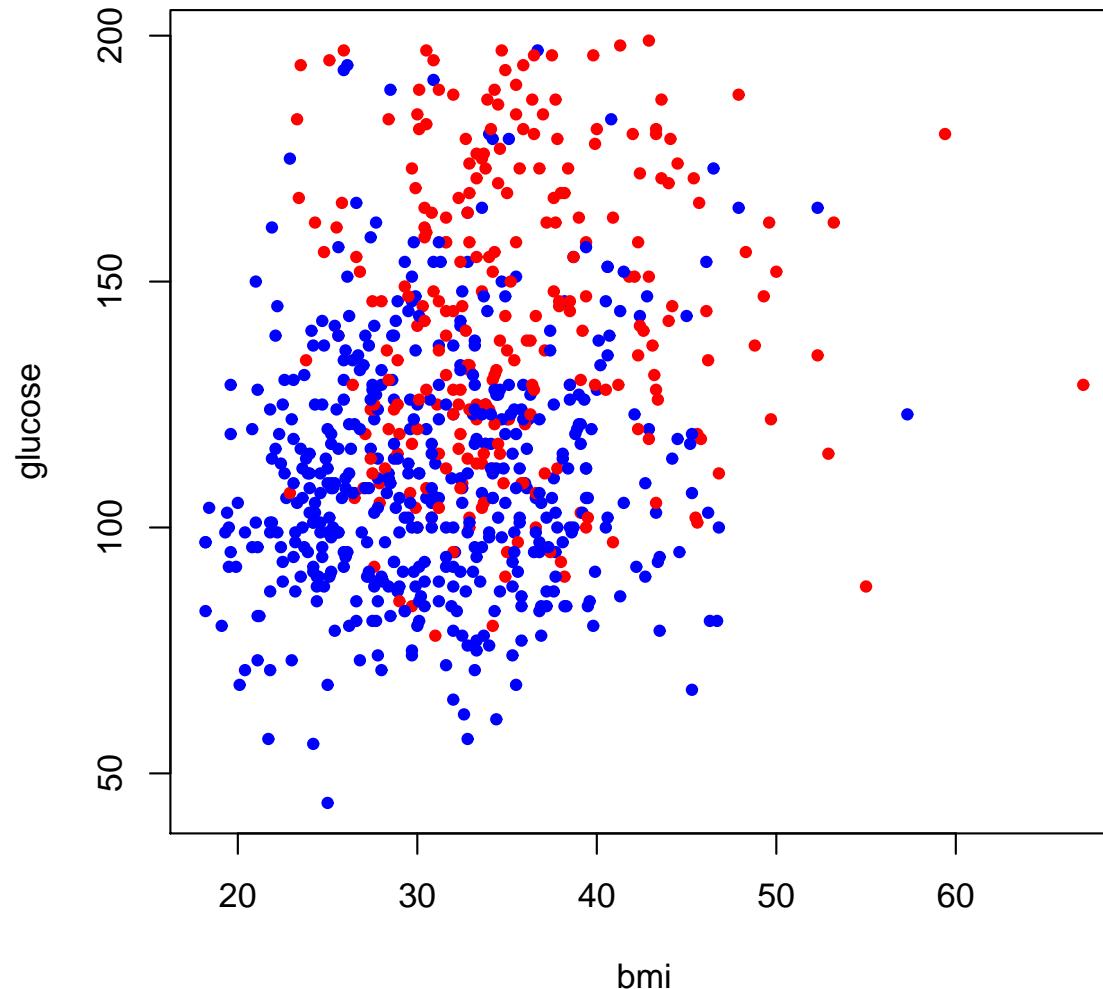


Границы ящика — первый и третий квартили ( $Q_1$ ,  $Q_3$ ), линия в середине соответствует медиане ( $Q_2$ ).

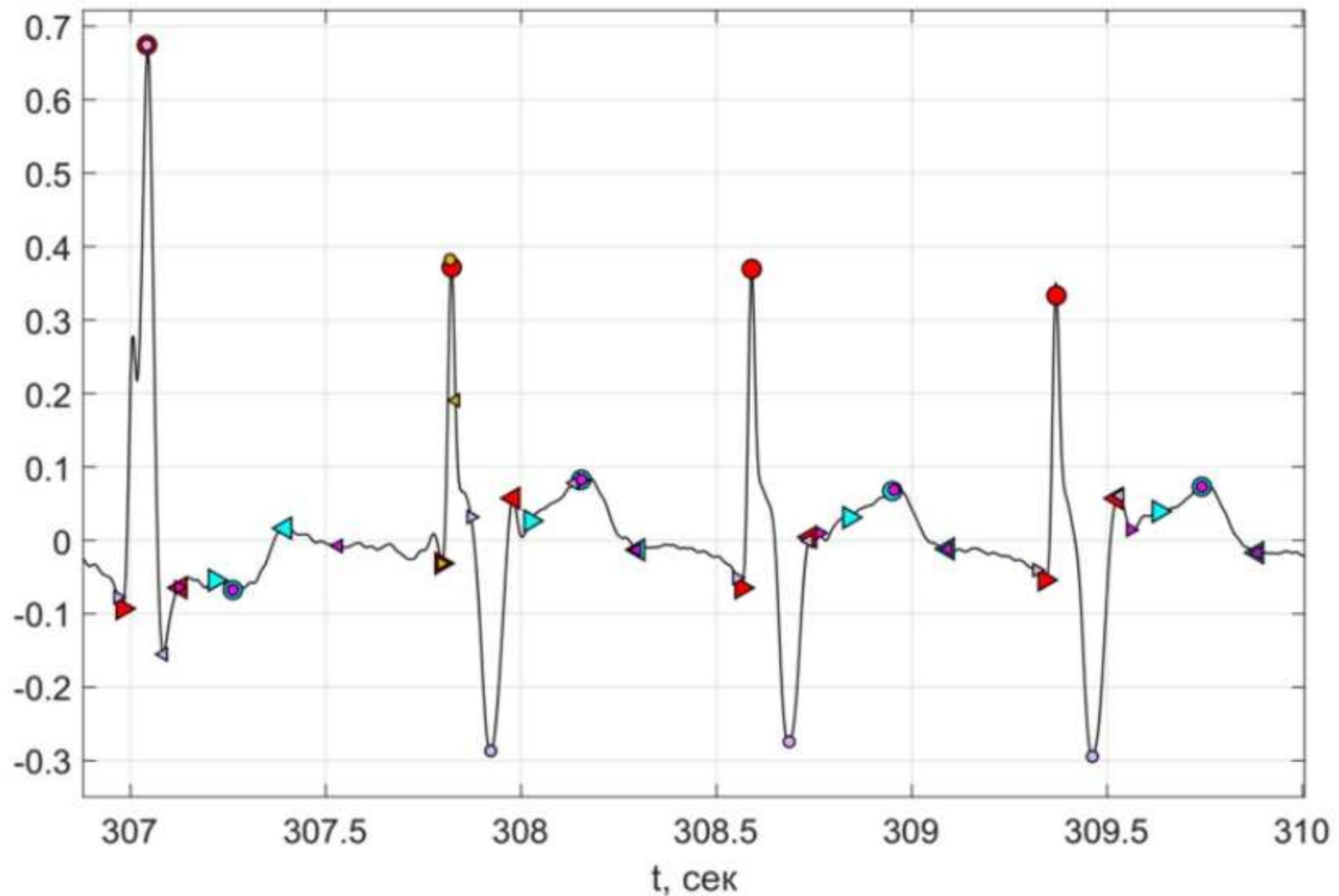
Усы:  $Q_1 - k(Q_3 - Q_1)$ ;  $Q_3 + k(Q_3 - Q_1)$ ;  $k = 1.5$ .

Наблюдения, выходящие за «усы», изображаются точками (выбросы).

Проекции точек на двумерную координатную плоскость переменных  $bmi$   
(масса/рост<sup>2</sup>), glucose (содержание глюкозы).



## Автоматическое определение заболевания по ЭКГ



В задачах медицинской диагностики может ставиться другая цель:  
определить оптимальный курс лечения (это может быть как задача классификации, так и задача восстановления регрессии),  
оценить оптимальную дозу лекарства (задача восстановления регрессии),  
спрогнозировать время протекания болезни (задача восстановления регрессии) и т. п.

## Распознавание изображений

Например, распознавание рукописного символа (цифры) по его изображению.

Данные optdigit <http://www.ics.uci.edu/~mlearn/MLRepository.html> содержат 1934 размеченных черно-белых изображений цифр  $32 \times 32$ .

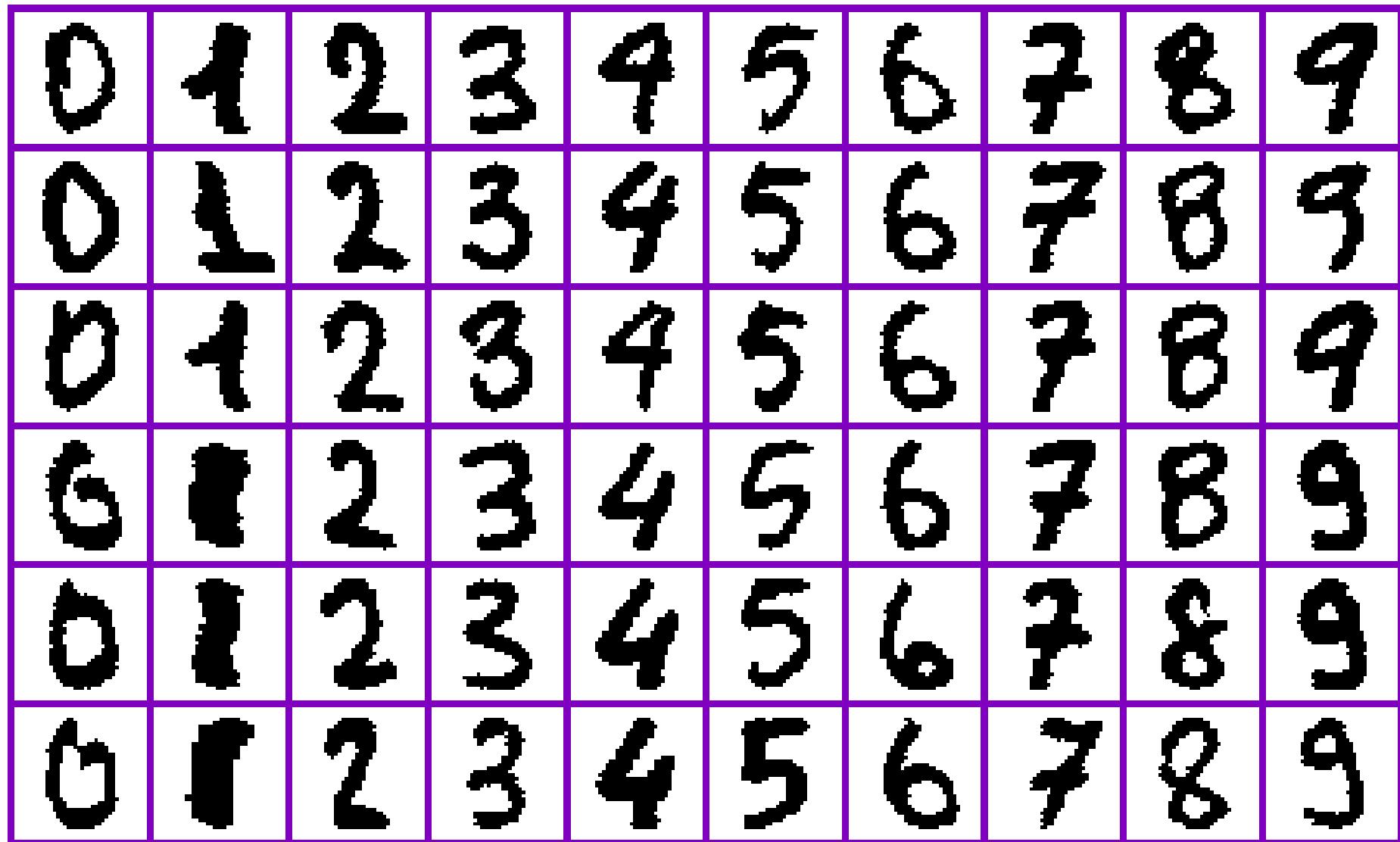
1 — пиксель черный, 0 — пиксель белый.

Признаковое описание — бинарный вектор  $x$  длины  $32^2 = 1024$ .

$$\mathcal{X} = \{0, 1\}^{1024}.$$

$$\mathcal{Y} = \{0, 1, 2, \dots, 9\}$$

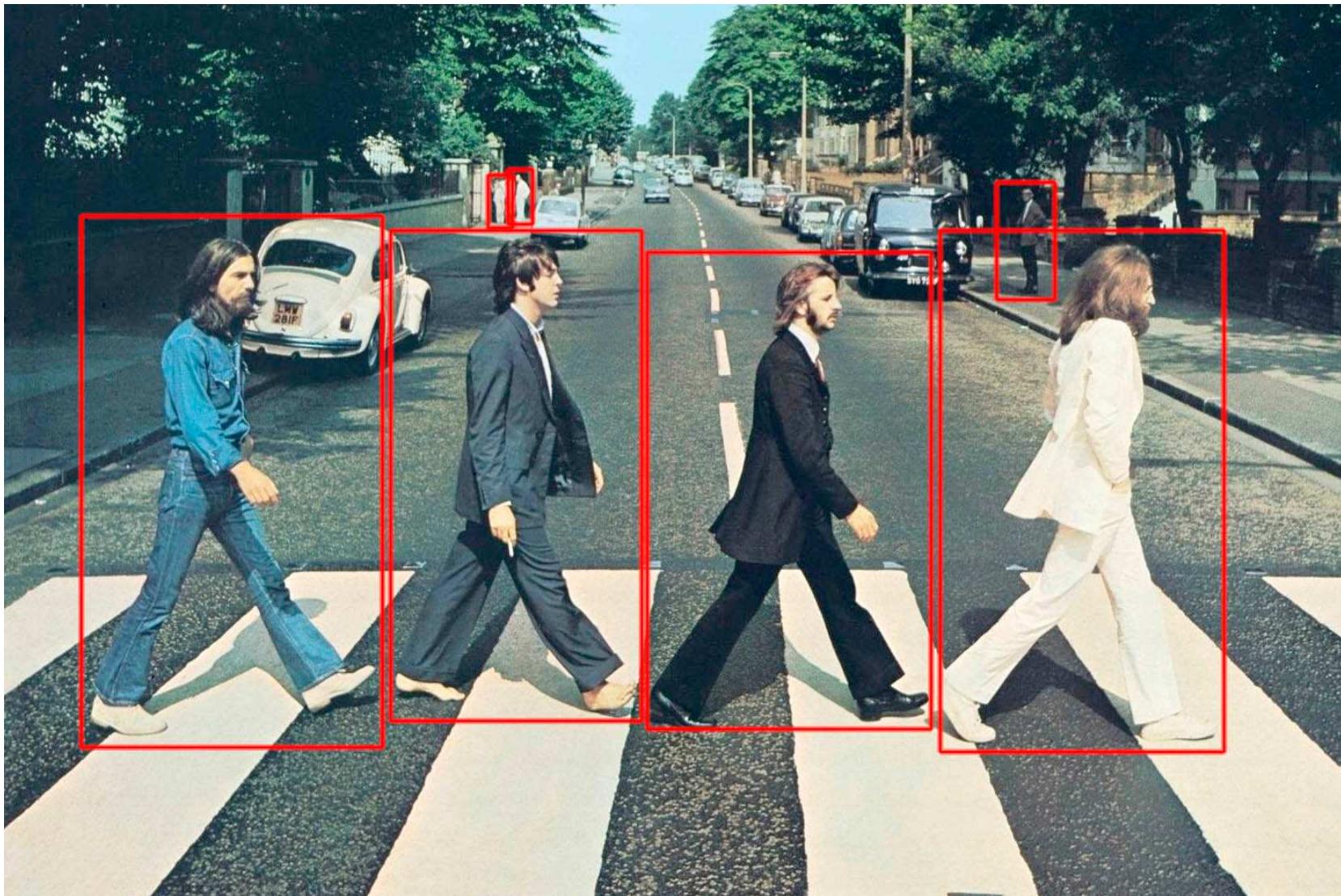
Некоторые объекты из обучающей выборки



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

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

## Детектирование объектов



*Замечание.* В настоящее время в компьютерном зрении используются сложные признаковые описания и изобретательные методы по их детектированию и описанию (SIFT, SURF, GLOH, HOG и др.), правда метод глубокого обучения позволил синтез признаков сделать во многих задачах автоматическим.

- *Отбор признаков* (features selection) — выбор значимых (информационных) признаков
- *Синтез признаков* (features extraction) — генерация новых признаков как функций от исходных («сырых») признаков

Обе задачи решают

- вручную (специалисты в своих предметных областях),
- существуют автоматические методы отбора/синтеза признаков.

## Распознавание спама

«Bag of words» — модель представления текста как набора (мультимножества, bag — сумки) слов, входящих в него.

При этом, как правило, не делают различий между разными формами одного и того же слова. Для этого слова приводят к начальной (канонической) форме (*лемме*) или находят основу слова (*stem*).

Процесс нахождения основы слова называется *стеммингом* (stemming).

Данные из SMS Spam Corpus <http://www.esi.uem.es/jmgomez/smsspamcorpus>

Almeida, T.A., Gomez Hidalgo, J.M., Yamakami, A. Contributions to the study of SMS Spam Filtering: New Collection and Results. Proceedings of the 2011 ACM Symposium on Document Engineering (ACM DOCENG'11), Mountain View, CA, USA, 2011.

A word cloud visualization where the size of each word indicates its frequency or importance in the SMS spam corpus. The most prominent words are 'call', 'get', and 'now'. Other significant words include 'like', 'will', 'make', 'good', 'need', 'txt', 'send', 'well', 'ill', 'cant', 'stop', 'think', 'work', 'new', 'take', 'back', 'repli', 'dont', 'meet', 'sorri', 'e', 'home', 'ask', 'today', 'phone', 'free', 'one', 'lor', 'later', 'know', 'day', 'just', 'see', 'tell', 'time', 'say', 'text', 'mobil', 'got', 'come', 'want', 'love', 'can', and 'hope'. The words are rendered in a variety of pink and purple hues.

guarante  
voucher **now**  
**just** show send repli get  
latest receiv everi Servic  
todaywin draw **free** cash per  
offer prize **week** day award  
tone **text** urgent rington custom  
nokia min stop willwon chat  
call new **claim** contact  
collect

can wait sorri good  
think hope dear  
well say night feel  
miss later today **will** hey  
ill take meet wat work  
love one back **now** lor  
day just much see like  
**call** dont home ask  
make cant send happy  
way tell **want** need

## Цена на недвижимость

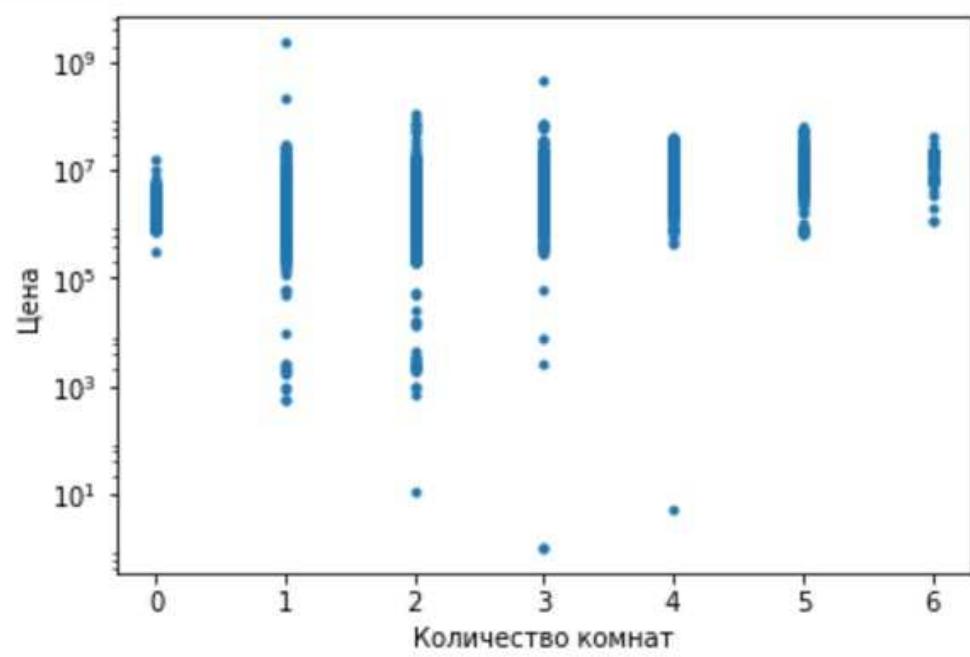
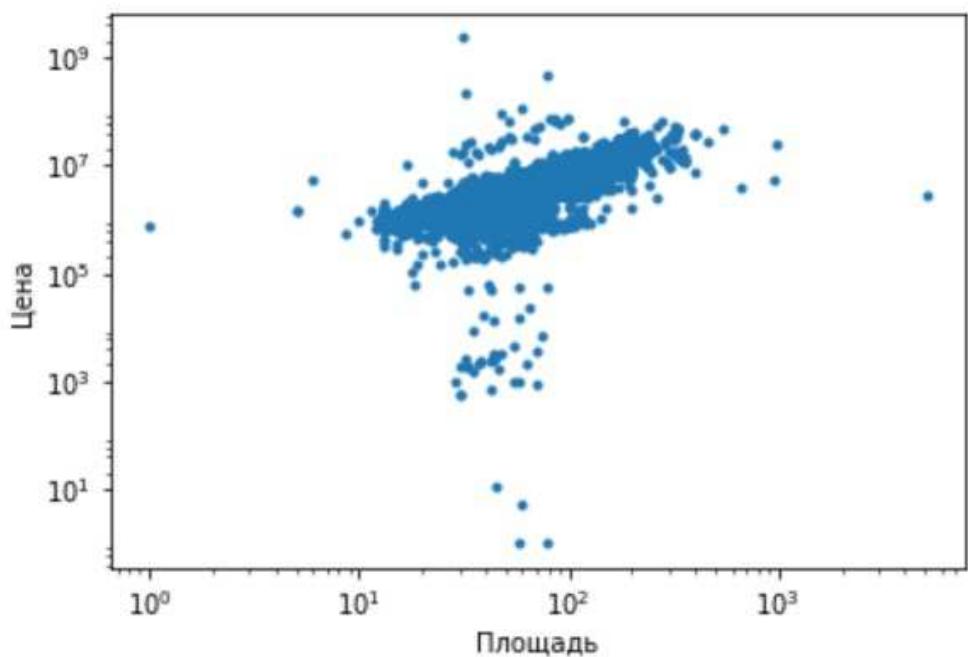
Имеются данные о стоимости квартир.

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)

Выборка содержит 72379 записей.

## Диаграммы рассеяния

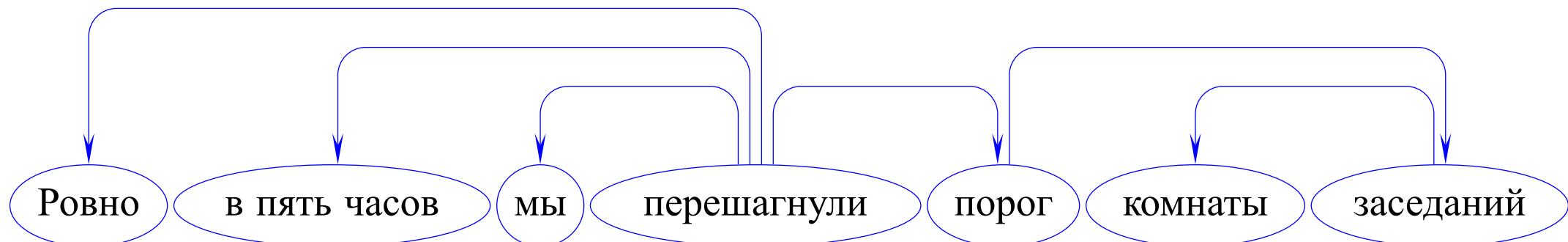


## Синтаксический разбор предложений (синтаксический анализ)

На вход подается предложение. Необходимо провести его синтаксический анализ, т. е. определить члены предложения и построить дерево синтаксической зависимости.

Ровно в пять часов мы перешагнули порог комнаты заседаний

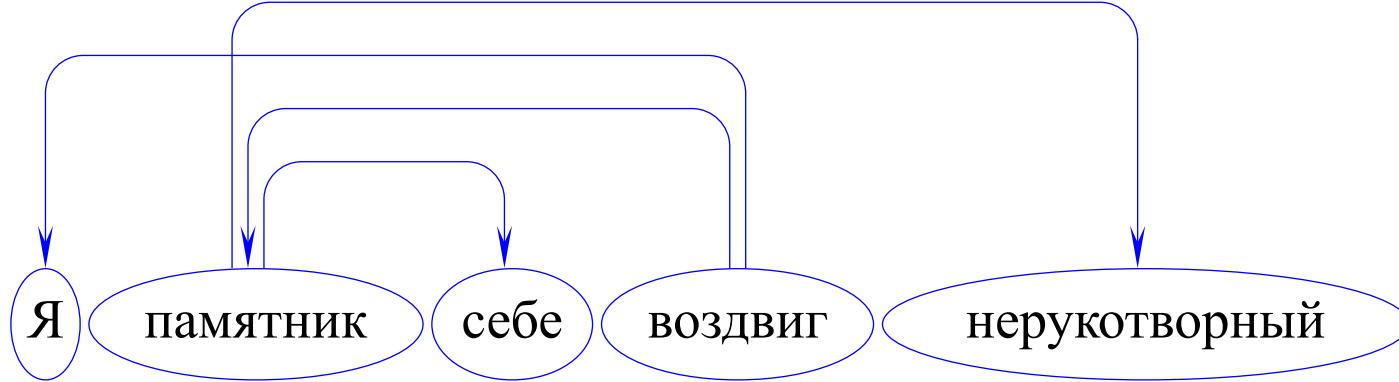
(А. и Б. Стругацкие)



Существуют алгоритмы, основанные на правилах, но можно применить и методы машинного обучения (структурное обучение)

Дерево синтаксического разбора обычно *проективное*.

Исключения встречаются в разговорной и поэтической речи.



1. Сегментация предложений
2. Лексический анализ (токенизация) — определение слов
3. Морфологический анализ — определение грамматической категории для каждого слова
4. Синтаксический анализ

Задачу синтаксического разбора можно поставить как задачу построения проективного дерева минимального веса, где веса подбираются на основе обучения по большой базе разобранных (вручную) предложений.

Для русского языка — «Национальный корпус русского языка» (<http://www.ruscorpora.ru>).

## 1.4. Обучение без учителя

*Обучение с учителем* можно рассматривать как игру двух лиц: ученика, который должен восстановить зависимость, и учителя, который для объектов из обучающей выборки указывает ученику соответствующий им выход.

Иногда можно считать, что объекты из обучающей выборки предъявляются средой, а иногда — их выбирает сам учитель, в некоторых случаях их выбирает ученик (*активное обучение*).

В случае *обучения без учителя* нет учителя и «обучающая выборка» состоит только из объектов.

Ученик, имея только список объектов  $x^{(1)}, x^{(2)}, \dots, x^{(N)}$ , должен определить, как объекты связаны друг с другом.

Например, разбить объекты на группы (*кластеры*), так, чтобы в одном кластере оказались «похожие» друг на друга объекты, а в разных кластерах — «мало похожие».

## Цели обучения без учителя

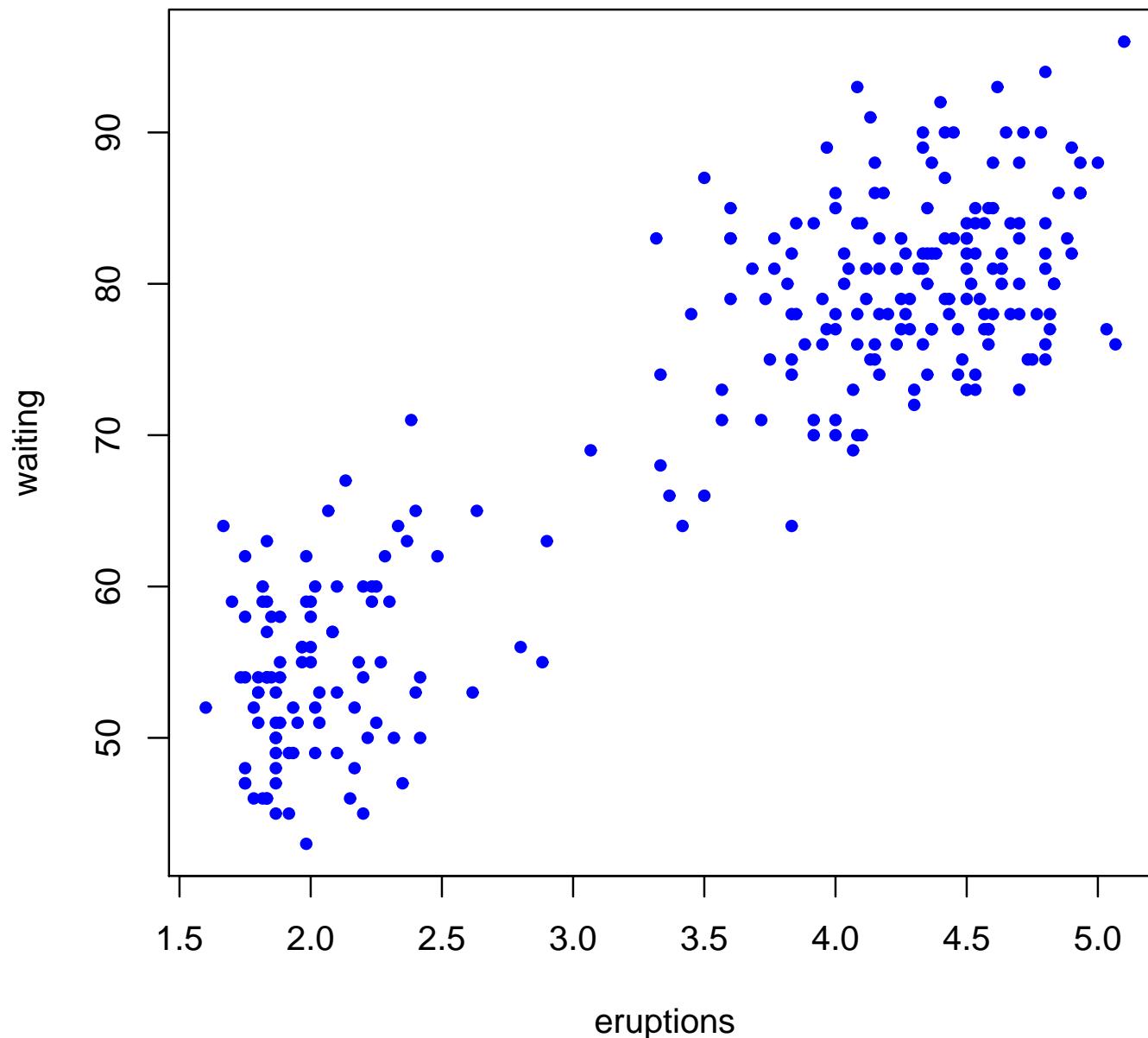
- в *Data Mining*:  
выявлять структуру в данных для лучшего их понимания;
- в *Machine Learning*:  
как предварительный этап при решении задачи обучения с учителем (например, сокращение размерности (PCA и др.) или решаем задачу кластеризации, а потом в каждом кластере — свою задачу классификации и т. п.).

## **1.4.1. Примеры практических задач**

### **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.*)

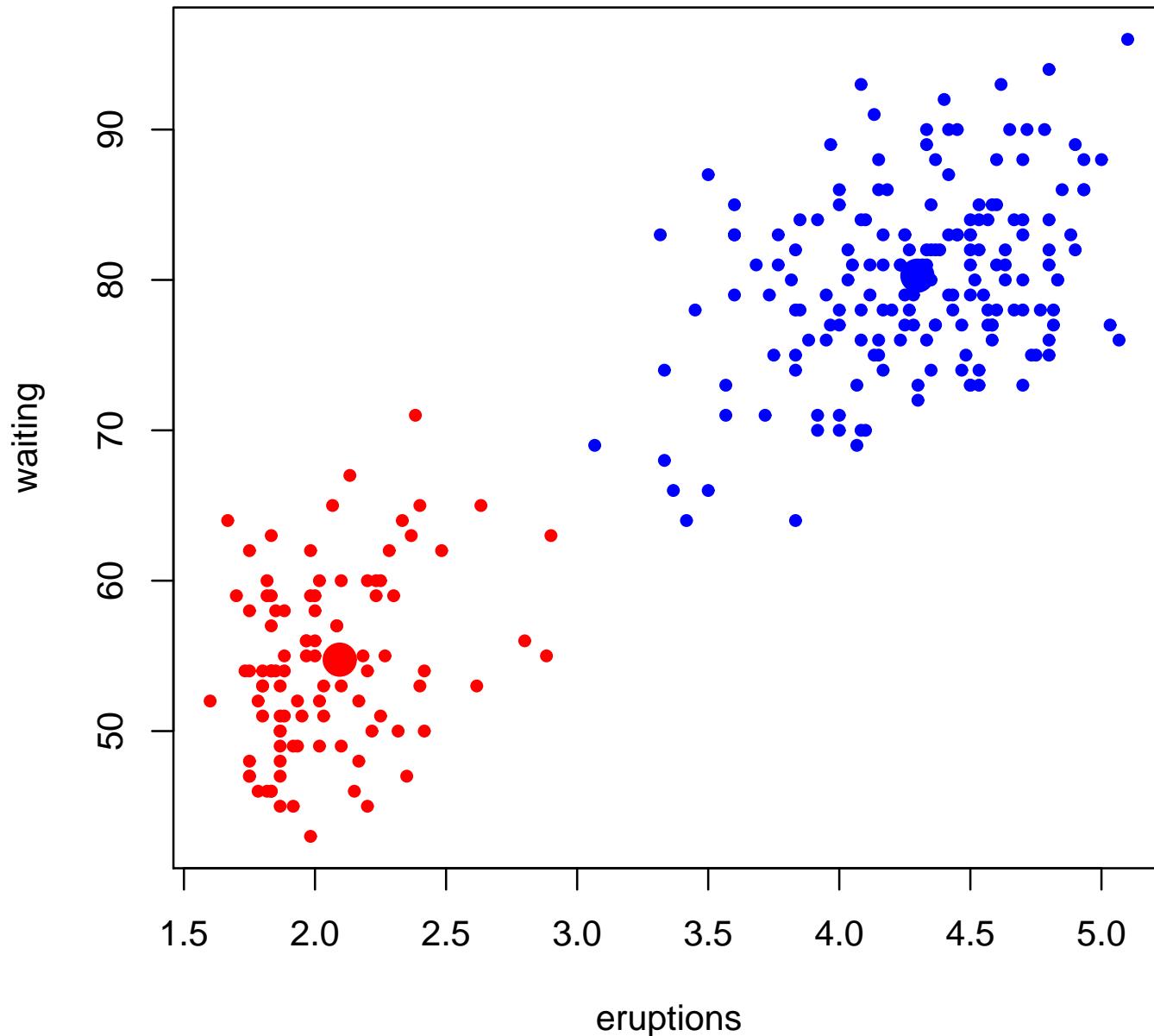
Диаграмма, представляющая данные о времени извержения и промежутках между извержениями гейзера.



Мы видим, что точки группируются в два кластера.

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

В другом — с большой длительностью и большим временем ожидания.



## **Анализ данных, полученных с биочипов**

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

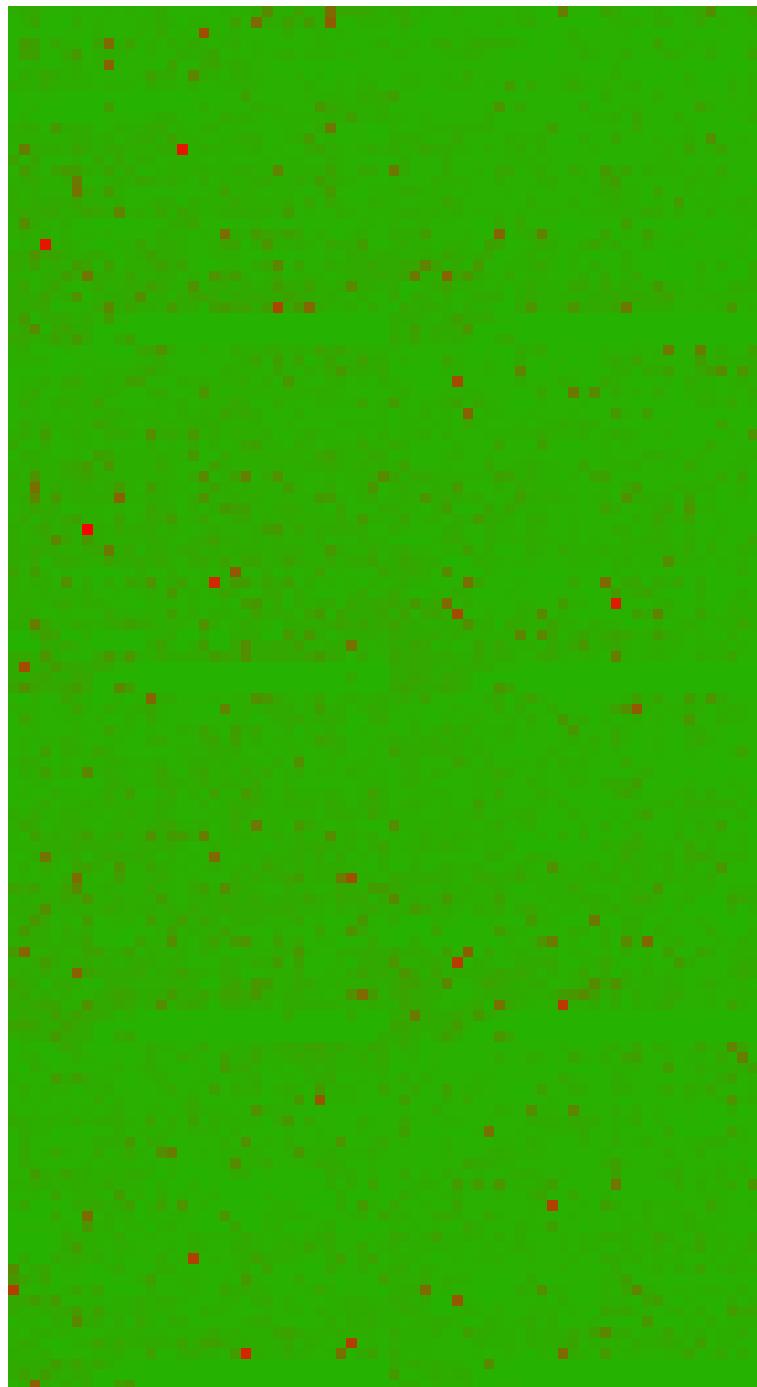
*Экспрессия — это процесс перезаписи информации с гена на РНК, а затем на белок. Количество и даже свойства получаемого белка зависят не только от гена, но также и от различных внешних факторов (например, от введенного лекарства, или от того, болеет клетка или нет).*

Таким образом, уровень экспрессии — это мера количества генерируемого белка. 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).



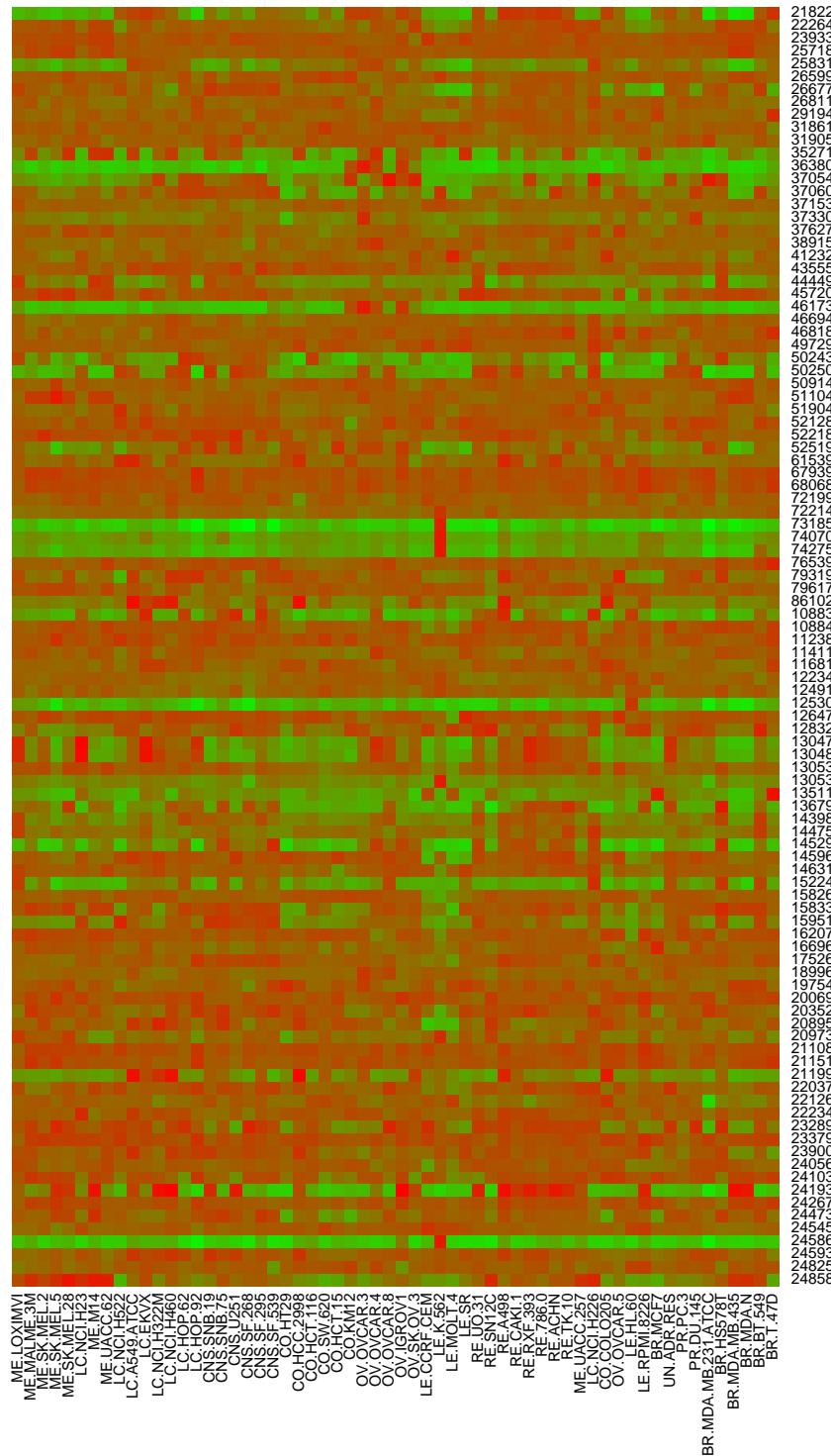
Пусть было проведено несколько экспериментов, в которых на биочип размещался контрольный и другие генетические материалы (например, с больными клетками). Информацию, полученную в результате проведения такой серии экспериментов можно представить в виде числовой матрицы, в которой строки соответствуют разным генам, а столбцы — разным экспериментам (разным клеткам).

Данные для 60 экспериментов с биочипом. «Genomics Bioinformatics Group»  
<http://discover.nci.nih.gov/datasetsNature2000.jsp>

Строки — гены,  
столбцы — эксперименты (различный материал с больными клетками).

Приведены только первые 100 строк (из общего числа 1375).

Строки, содержащие отсутствующие значения, исключены.



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.

## Генезис языков

*Список Сводешиа* (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 *rік*, *цветок* и *квітка*, *видеть* и *бачити*

или

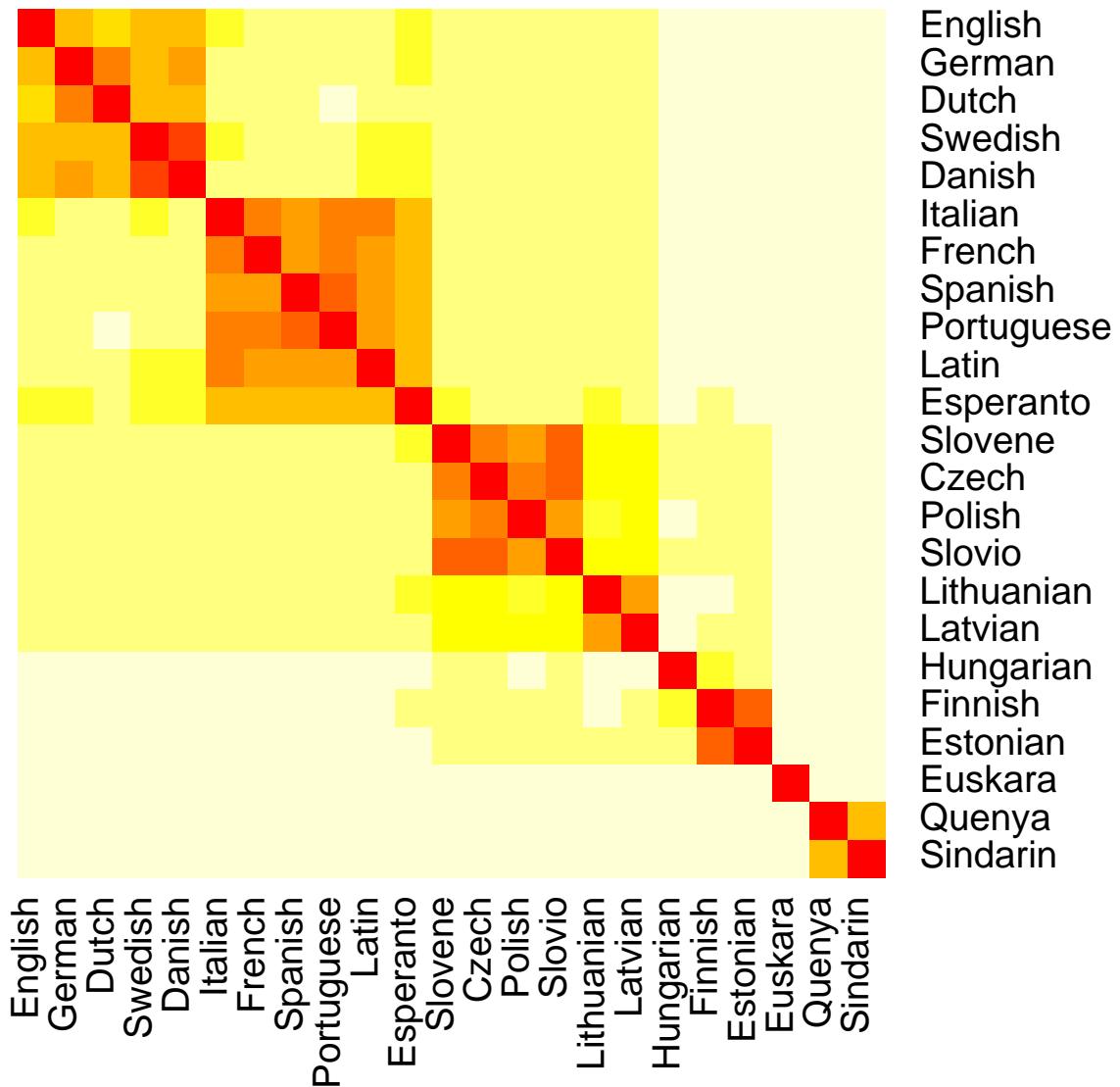
*колесо* и *wheel* и *ček* ?

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

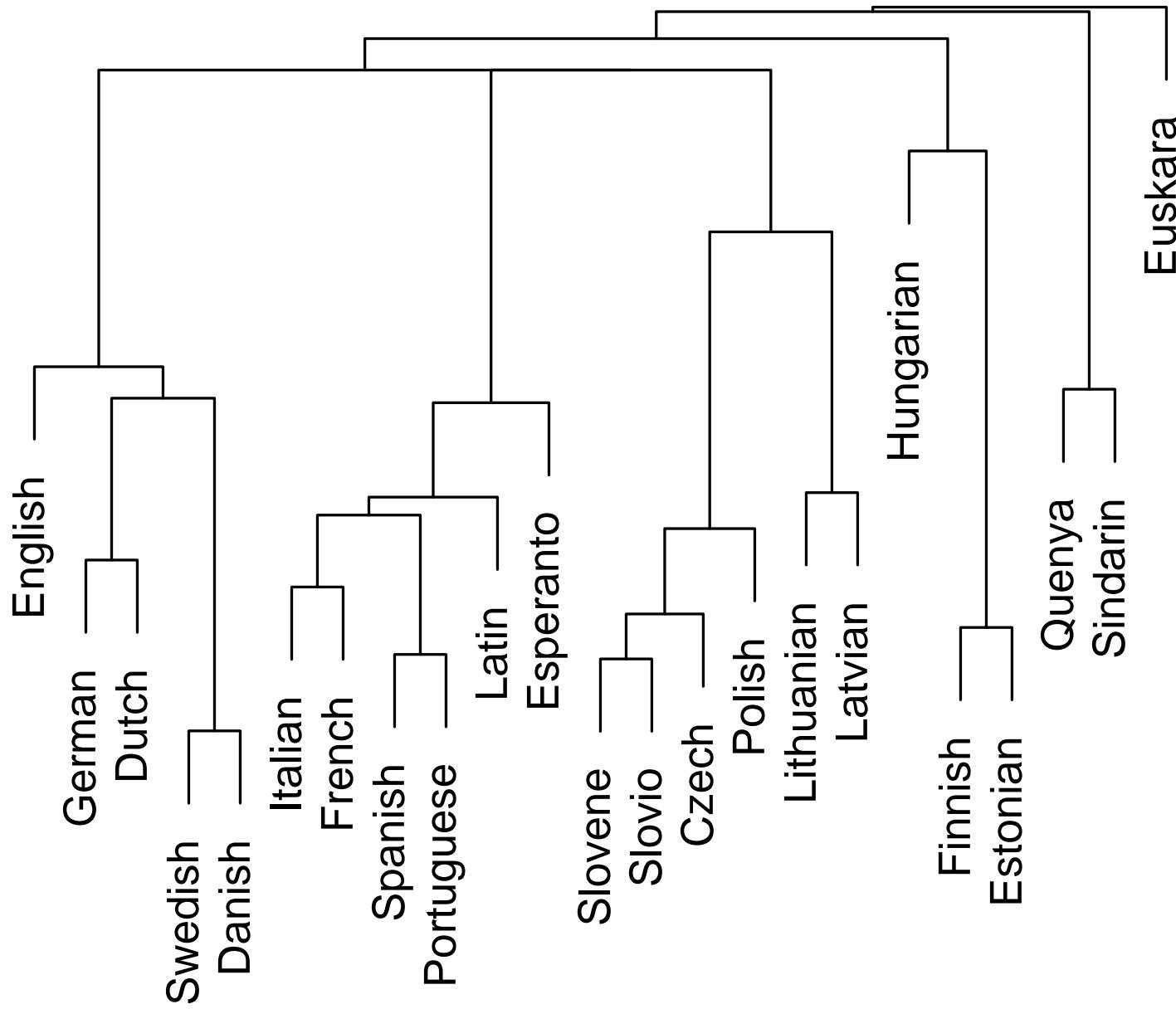
Более того, на основе анализа списка Сводеша для двух родственных языков можно приблизительно установить время их появления из единого пра-языка.

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.



## 1.5. 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).  
Ошибка удалось понизить с 26% до 15% (сейчас еще меньше) – A. Krizhevsky,  
I. Sutskever, G. E. Hinton
- Беспилотные автомобили (Google и др.)
- AlphaGo
- Prisma
- Умные помощники (Google, Amazon)
- Автономные роботы (Boston Dynamics и др.)
-

## 1.6. О курсе

- Различные алгоритмы и подходы к решению задач машинного обучения:
  - Линейная регрессия
  - Метод ближайших соседей
  - Машина опорных векторов
  - Нейронные сети
  - Деревья решений
  - Бустинг (AdaBoost, GBT, Random Forest) и баггинг
  - Обучение без учителя, кластеризация
- Элементы теории (Вапника–Червоненкиса) о качестве обучения

## 1.6.1. Ресурсы

- Wiki-портал <http://www.machinelearning.ru>
- Мой курс: <http://www.uic.unn.ru/~zny/ml>  
(презентации лекций, лабораторные работы, R и Python, полезные ссылки, ML для «чайников» и др.)
- *Воронцов К.В.* Машинаное обучение (курс лекций)  
см. <http://www.machinelearning.ru>,  
видео-лекции [http://shad.yandex.ru/lectures/machine\\_learning.xml](http://shad.yandex.ru/lectures/machine_learning.xml)
- *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/>
- *James G., Witten D., Hastie T., Tibshirani R.* An Introduction to Statistical Learning with Applications in R. Springer, 2013. Рус. пер.: *Джеймс Г., Уиттон Д., Хасти Т., Тибширани Р.* Введение в статистическое обучение с примерами на языке R. ДМК Пресс, 2016.
- *Flach P.* Machine Learning: The Art and Science of Algorithms That Make Sense of Data. Cambridge University Press, 2012. Рус. пер.: *Флах П.* Машинаное обучение. Наука и искусство построения алгоритмов, которые извлекают знания из данных. ДМК Пресс, 2016.

## 1.6.2. Software

- Библиотека Scikit-learn (Python) <http://scikit-learn.org/>
- Система для статистических вычислений R <http://www.r-project.org/>
- Statistics and Machine Learning + Neural Network Toolbox (Matlab)  
<https://www.mathworks.com/>
- Библиотека машинного зрения OpenCV (C, C++, интерфейс для Python) (раздел ML) <http://opencv.org/>
- Intel DAAL: Intel Data Analytics Acceleration Library  
<http://software.intel.com/en-us/daal/>
- Библиотека алгоритмов для анализа данных Weka (Java)  
<http://www.cs.waikato.ac.nz/~ml/weka/>
- Пакет для решения задач машинного обучения и анализа данных Orange  
<http://orange.biolab.si/>
- ...
- *Данные для экспериментов:* UCI Machine Learning Repository  
<http://archive.ics.uci.edu/ml/>

## Software. Глубокое обучение

№	<i>Пакет/библиотека</i>	<i>Интерфейс</i>	<i>ОС</i>	FCNN	CNN	RNN	AE	RBM
1	TensorFlow	Python, Java, Go	Linux, Windows, Mac OS, Android	+	+	+	+	+
2	Theano	Python	Linux, Windows, Mac OS	+	+	+	+	+
3	Keras	Python, R	Linux, Vagrant	+	+	+	+	+
4	Torch	Lua, C	Linux, iOS, Android	+	+	+	+	+
5	Caffe	C++, Python, Matlab	Linux, Windows, OS X	+	+	+	+	-
6	MXNet	C++, Python, R, Scala, Julia, Perl, MATLAB, JavaScript	Linux, Windows, Mac OS	+	+	+	+	+
7	Matlab Neural Networks Toolbox	Matlab	Linux, Windows, OS X	+	+	+	+	-
8	Wolfram Mathematica	Java, C++	Linux, Windows, OS X	+	+	-	+	+

Еще данные и идеи задач:

- <https://www.kaggle.com> Данные, использованные в соревнованиях по машинному обучению. Много задач по разной тематике.
- <https://mlbootcamp.ru> Соревнования по анализу данных от Mail.ru.
- <http://cs229.stanford.edu/projects2013.html> Отчеты студентов Стэнфорда (можно искать идеи задач, ссылки на данные).
- <http://archive.ics.uci.edu/ml> Опять же много наборов данных для различных прикладных задач. Имеет смысл смотреть данные, которые были добавлены недавно (задачи интересней, данных больше).
- <http://www.image-net.org>, <http://pascallin.ecs.soton.ac.uk/challenges/VOC> Данные соревнований по классификации изображений и детектированию объектов на них.
- [http://www.vision.caltech.edu/Image\\_Datasets/CaltechPedestrians](http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians) Данные для детектирования пешеходов на изображениях/видео. В разделе “Related Datasets” есть ссылки на другие подобные базы.
- <https://pslcdatashop.web.cmu.edu> Данные для анализа образовательных программ.
- <http://www.edutainme.ru/post/bolshie-dannye-v-obrazovanii>
- <http://nlp.stanford.edu/sentiment> Данные для анализа тональности текста (sentiment analysis).
- <http://www.kdnuggets.com/datasets>

Сайт курса (еще раз): <http://www.uic.unn.ru/~zny/ml>

(презентации лекций, лабораторные работы, R и Python, полезные ссылки, ML для «чайников» и др.)

## **Домашнее задание (к следующей лекции)**

1. Вспомнить теорию вероятностей (зависимые и независимые события, формулы сложения и умножения вероятностей, формулу Байеса, формулу полной вероятности, случайные величины, интегральная функция распределения, плотность вероятности, математическое ожидание, дисперсия, нормальная случайная величина).
2. Установить нужный софт

На следующей лекции будет турориал: использование Scikit-learn.

Желающие могут принести ноутбуки с установленными пакетами.

Я использую Anaconda 4.4.0 (<http://continuum.io/downloads>), которая содержит следующие версии библиотек (помимо всего прочего):

- python 3.5.3
- numpy 1.11.3
- scipy 0.18.1
- matplotlib 2.0.0
- scikit-learn 0.18.1
- pandas 0.19.2



*Глава 2*

## **Вероятностная постановка задачи и некоторые методы**



## Некоторые обозначения

$d$  число входных признаков

$N$  длина обучающей выборки

$\mathcal{X}$  множество объектов

$\mathcal{Y}$  множество ответов (выходов)

$x^{(1)}, x^{(2)}, \dots, x^{(N)}$  объекты обучающей выборки,  $x^{(i)} \in \mathcal{X}$  ( $i = 1, 2, \dots, N$ )

$y^{(1)}, y^{(2)}, \dots, y^{(N)}$  выходы для объектов из обучающей выборки,  $y^{(i)} \in \mathcal{Y}$

$K$  количество классов (в задачах классификации)

$\Pr A$  вероятность события  $A$

$\Pr(A|B)$  вероятность события  $A$  при условии, что наступило событие  $B$

$P_X(x)$  интегральная функция распределения:  $P_X(x) = \Pr\{X \leq x\}$

$p_X(x)$  плотность вероятности непрерывной случайной величины  $X$

$P(y|x)$  условная интегральная функция распределения

$p(y|x)$  условная плотность вероятности

$E X$  математическое ожидание случайной величины  $X$

$D X$  или  $\text{Var } X$  дисперсия случайной величины  $X$

$\sigma X$  среднее квадратическое отклонение:  $\sigma X = \sqrt{D X}$

## 2.1. Вероятностная постановка задачи

$\mathcal{X} = \mathbf{R}^d$  — множество объектов (входов) (точнее: множество их описаний)

$\mathcal{Y} = \mathbf{R}$  — множество ответов (выходов)

Будем рассматривать пары  $(x, y)$  как реализации  $(d + 1)$ -мерной случайной величины  $(X, Y)$ , заданной на вероятностном пространстве

$$(\mathcal{X} \times \mathcal{Y}, \mathbf{A}, \Pr), \quad X \in \mathbf{R}^d, Y \in \mathbf{R}.$$

$j$ -й признак — бинарный, номинальный или порядковый  $\Leftrightarrow X_j$  — дискретная с. в.

$j$ -й признак — количественный  $\Leftrightarrow X_j$  — непрерывная с. в.

Интегральный закон распределения  $P_{X,Y}(x, y)$  не известен, однако известна *обучающая выборка*

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

где  $(x^{(i)}, y^{(i)})$  являются независимыми реализациями случайной величины  $(X, Y)$ .

Требуется найти функцию  $f : \mathcal{X} \rightarrow \mathcal{Y}$ , которая по  $x$  предсказывает  $y$ ,  $f \in \mathcal{F}$

$f$  называется *решающей функцией*, *решающим правилом* или *моделью*, а также *классификатором* (в случае задачи классификации).

Построение  $f$  называют *обучением*, *настройкой модели* и т. п.

## Пример

Имеются данные о 114 лицах с заболеванием щитовидной железы.

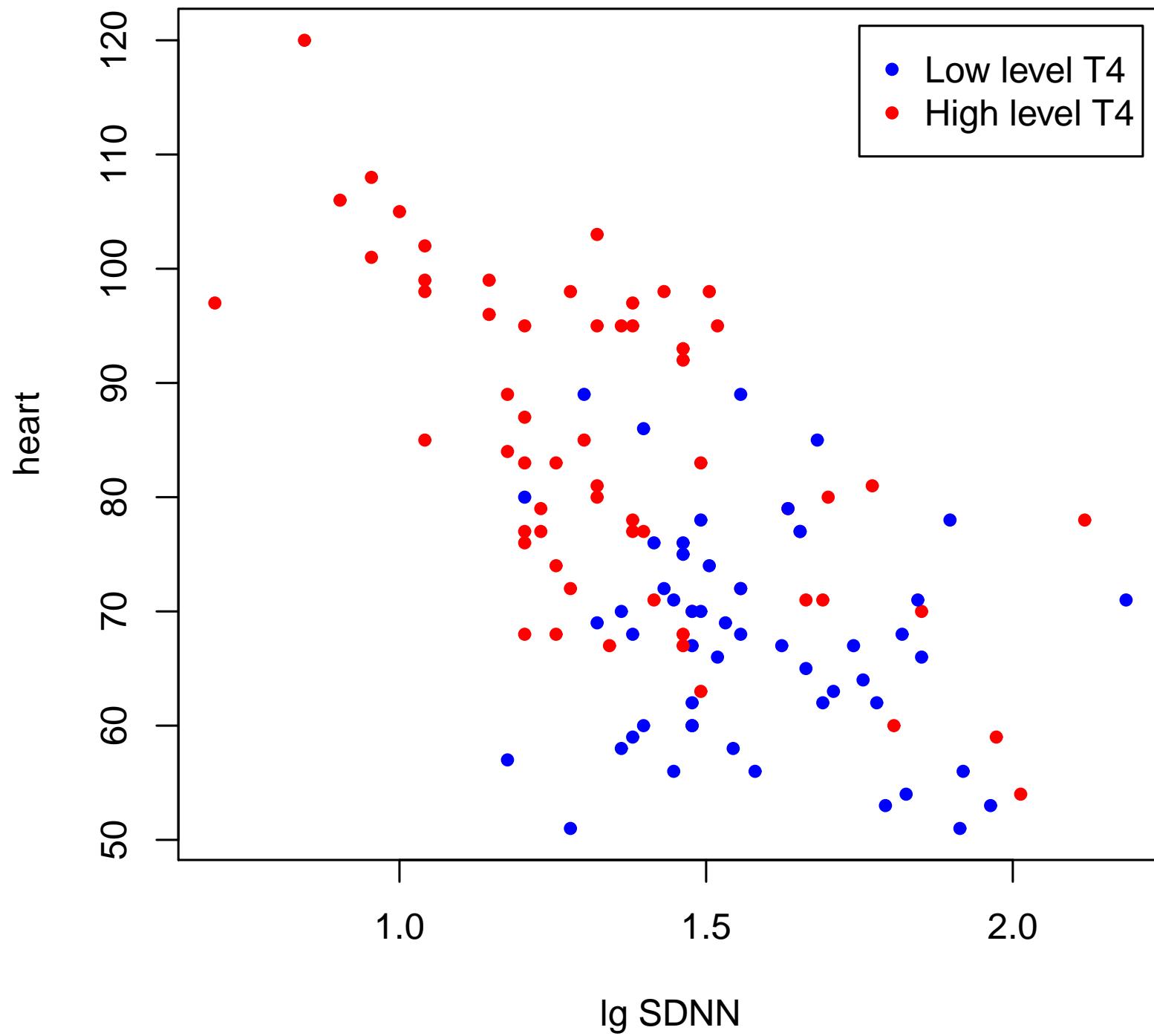
У 61 — повышенный уровень свободного гормона T4,

у 53 — уровень гормона в норме.

Для каждого пациента известны следующие показатели:

- heart — частота сердечных сокращений (пульс),
- SDNN — стандартное отклонение длительности интервалов между синусовыми сокращениями RR.

Можно ли научиться предсказывать (допуская небольшие ошибки) уровень свободного T4 по heart и SDNN?



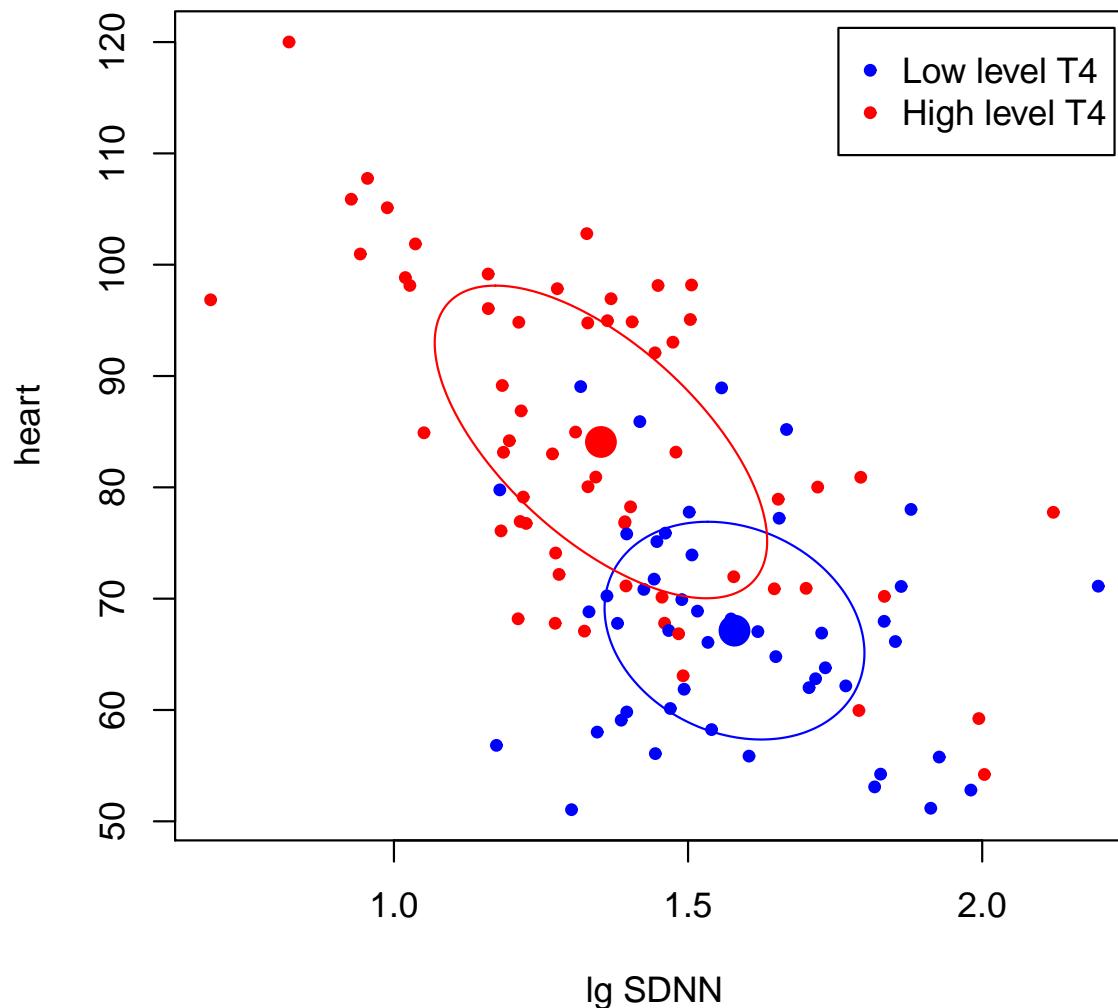
Многомерный тест Шапиро-Уилка проверки нормальности распределения

Гипотеза  $H_0$ : « $X$  распределено нормально». Пусть  $\alpha = 0.05$

Для синих точек (низкий уровень)  $W = 0.9809$ , p-value = 0.5527  $\Rightarrow$  принимаем

Для красных точек (высокий уровень)  $W = 0.9542$ , p-value = 0.02306  $\Rightarrow$  отвергаем

Для всей совокупности:  $W = 0.9784$ , p-value = 0.06239  $\Rightarrow$  принимаем



Пусть дана функция потерь (штраф)  $L(y', y) = L(f(x), y)$ .

$x$  — вход,  $y$  — соответствующий выход,  $y' = f(x)$  — предсказанное значение

Например, в задачах восстановления регрессии:

- квадратичная ошибка (квадратичная функция потерь):  $L(y', y) = (y' - y)^2$ ;
- абсолютная ошибка:  $L(y', y) = |y' - y|$ ;
- относительная ошибка:  $L(y', y) = \frac{|y' - y|}{|y|}$ .

В задачах классификации:

- симметричная 0–1 функция штрафа (индикатор ошибки):

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

- В общем случае функция потерь полностью описывается  $K \times K$  матрицей  $L = (\ell_{y'y})$ , где  $\ell_{y'y} = L(y', y)$ .

Пусть, например, в задаче медицинской диагностики  $\mathcal{Y} = \{0, 1\}$ , где  $y = 0$  — пациент здоров,  $y = 1$  — пациент болен.

$$L(1, 1) = L(0, 0) = 0$$

- ошибка 1-го рода — ложная тревога — false positive error

$$L(1, 0) = 1 — болезнь определена у здорового пациента$$

- ошибка 2-го рода — ложный пропуск — false negative error

$$L(0, 1) = 10 — болезнь не определена у больного пациента (!)$$

Аналогично: автоматическое определение почтового спама, техническая диагностика, обнаружение комп. вирусов и т. д.

## Мат. ожидание функции потерь

$$R(f) = \mathbb{E} L(f(X), Y) = \int_{\mathcal{X} \times \mathcal{Y}} L(f(x), y) p(x, y) dx dy$$

— средний риск, средняя ошибка или ожидаемая ошибка предсказания.

(Если  $L(y', y) = I(y' \neq y)$ , то  $R(f)$  — вероятность ошибки)

$R(f)$  характеризует качество, или обобщающую способность, функции  $f$ .

Чем меньше  $R(f)$ , тем качество лучше.

Разумный подход: в качестве  $f$  взять функцию из заданного класса  $\mathcal{F}$ , минимизирующую средний риск  $R(f)$  (*принцип минимизации среднего риска*)

НО: Закон  $P(x, y)$  не известен и поэтому мы не можем точно вычислить  $R(f)$ .

- Можем восстановить  $P(x, y)$  по выборке («классический» подход)
- Можем аппроксимировать  $R(f)$ , а затем минимизировать
- ...

(Кроме того, даже если  $P(x, y)$  знаем (аппроксимировали), то возникает задача поиска минимума на множестве функций  $\mathcal{F}$  — задача вариационного исчисления.)

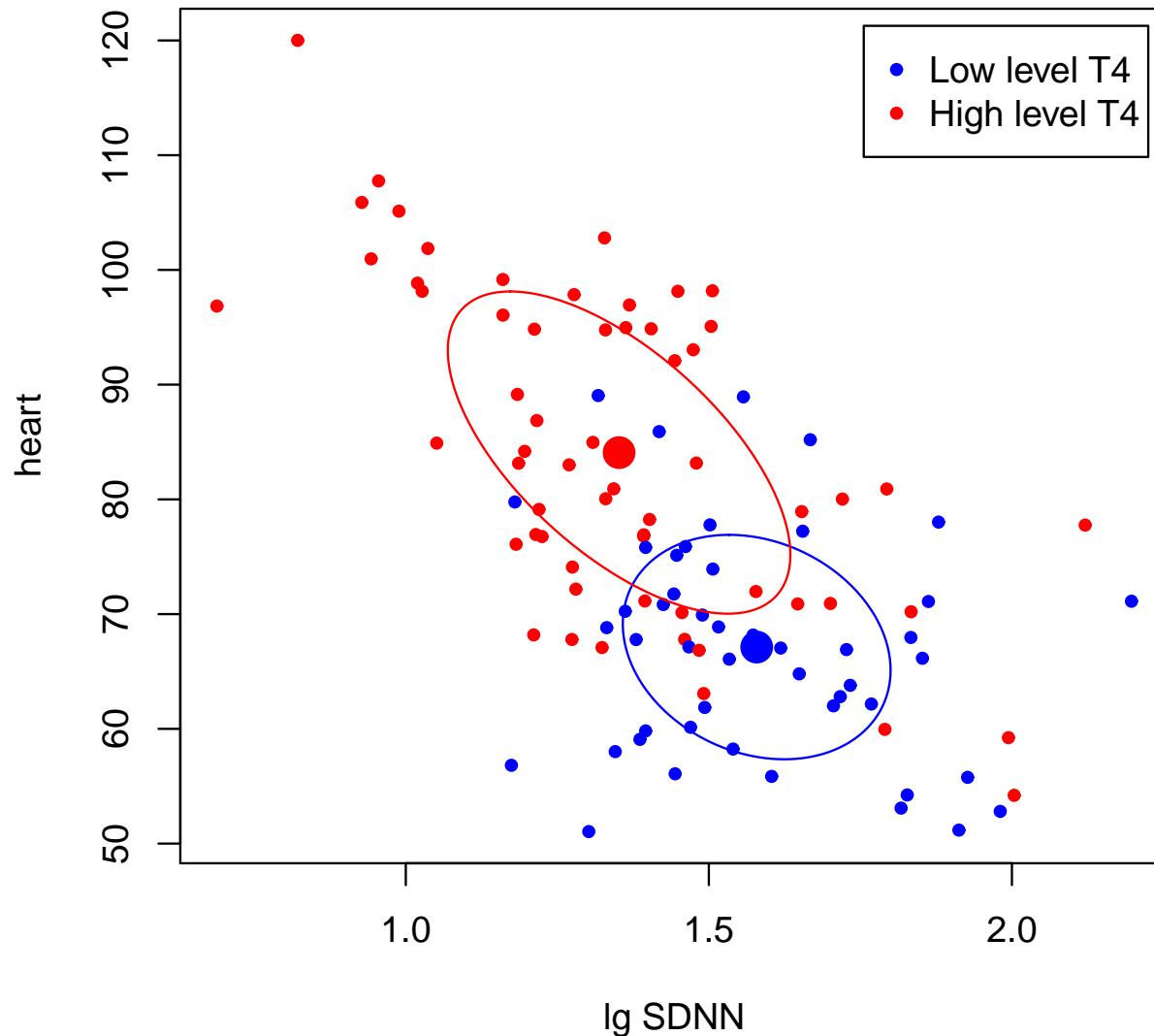
## 2.1.1. Восстановление функции распределения вероятности $P(X, Y)$

Будем минимизировать средний риск при  $f \in \mathcal{F}$

$$R(f) = \int_{\mathcal{X} \times \mathcal{Y}} L(f(x), y) p(x, y) dx dy. \quad (*)$$

- 1) по имеющейся выборке  $\{(x^{(1)}, y^{(1)}), \dots, (x^{(N)}, y^{(N)})\}$  решается задача восстановления функции распределения  $P(x, y)$ .
- 2) восстановленная функция  $\hat{P}(x, y)$  подставляется в  $(*)$  вместо  $P(x, y)$  и решается задача минимизации (точнее: вариационного исчисления, так как надо минимизировать на множестве функций  $\mathcal{F}$ ).
  - В качестве  $\hat{P}(x, y)$  можно взять выборочную функцию распределения. Согласно теореме Гливенко с ростом  $N$  эмпирическая функция распределения равномерно приближается к истинной. Нужна очень большая выборка!
  - Параметрические методы (например, дискриминантный анализ). Должны много знать о распределении. Выборка должна быть большой.
  - Непараметрические методы (например, парзеновские окна).

Многомерный тест Шапиро-Уилка проверки нормальности распределения  
Для синих точек (низкий уровень)  $W = 0.9809$ , p-value = 0.5527  
Для красных точек (высокий уровень)  $W = 0.9542$ , p-value = 0.02306  
Для всей совокупности:  $W = 0.9784$ , p-value = 0.06239



У метода восстановления функции распределения имеются большие недостатки. Известно, что задача восстановления функции распределения является некорректно поставленной, поэтому гарантировать успех в описанном подходе можно, только если об этой функции известно достаточно много и выборка большая.

Более того, задача восстановления функции распределения является центральной задачей в математической статистике и нецелесообразно сводить рассматриваемую частную задачу (минимизации среднего риска) к более общей.

Тем не менее, есть популярные методы, реализующие данный принцип (например, дискриминантный анализ)

## 2.1.2. Аппроксимация среднего риска $R(f)$

Элементы обучающей выборки  $\{(x^{(1)}, y^{(1)}), \dots, (x^{(N)}, y^{(N)})\}$  распределены случайно и независимо, каждый согласно закону распределения  $P(X, Y)$ , поэтому

$$R(f) \approx \hat{R}(f) = \hat{R}(f, x^{(1)}, y^{(1)}, \dots, x^{(N)}, y^{(N)}) = \frac{1}{N} \sum_{i=1}^N L(f(x^{(i)}), y^{(i)}),$$

$\hat{R}(f)$  — эмпирический риск (эмпирическая ошибка).

(Если  $L(y', y) = I(y' \neq y)$ , то  $\hat{R}(f)$  — доля ошибок на обучающей выборке)

*Принцип минимизации эмпирического риска:* минимизируем  $\hat{R}(f)$  на множестве  $\mathcal{F}$ .

В курсе будет рассмотрено много практических методов, реализующих этот принцип.

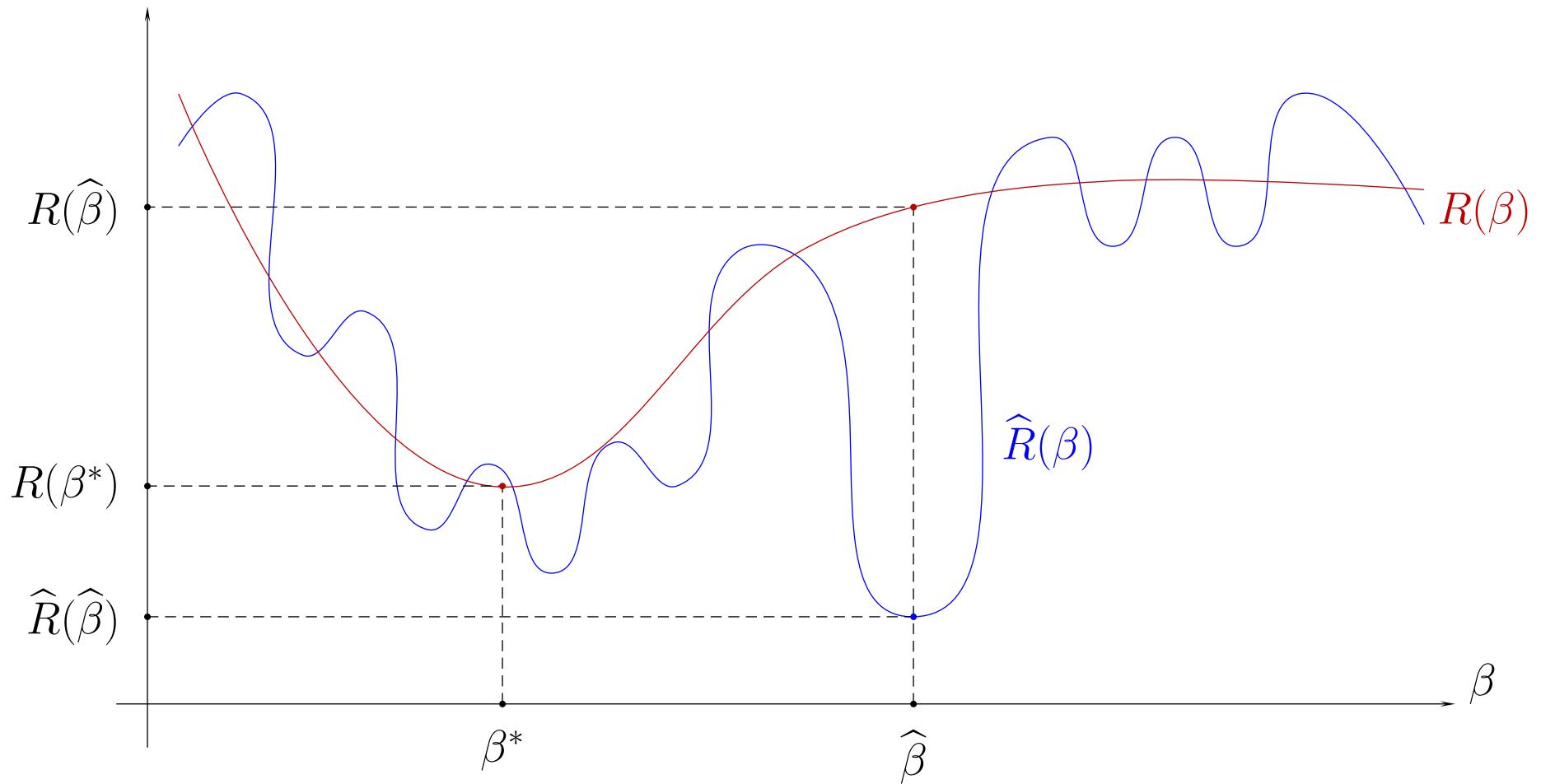
Недостатки: занижение величины риска и возможность переобучения.

*Переобучение:* ошибка на обучающей выборке много меньше  $R(f)$  (и  $\hat{R}_{\text{test}}(f)$ )

$$\hat{R}(f) \equiv \hat{R}_{\text{train}}(f) \equiv \frac{1}{N} \sum_{i=1}^N L(f(x^{(i)}), y^{(i)}) \ll R(f) \approx \hat{R}_{\text{test}}(f) \equiv \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} L(f(x_{\text{test}}^{(i)}), y_{\text{test}}^{(i)})$$

«Причина» переобучения при реализации метода минимизации эмпирического риска:

$$\mathcal{F} = \{f(\cdot, \beta) : \beta \in B\}$$



$$\hat{R}(\hat{\beta}) \ll R(\hat{\beta}) \quad \text{и} \quad R(\beta^*) \ll R(\hat{\beta})$$

## 2.2. Регрессионная функция

Рассматриваем задачу восстановления регрессии.

$$R(f) = \int_{\mathcal{X} \times \mathcal{Y}} L(f(x), y) p(x, y) dx dy = \int_{\mathcal{X}} \int_{\mathcal{Y}} L(f(x), y) p(y|x) dy p(x) dx,$$

т. е.

$$R(f) = \int_{\mathcal{X}} \mathsf{E} \left( L(f(x), Y) | x \right) p(x) dx$$

Рассмотрим задачу восстановления регрессии с квадратичной функцией потерь:

$$R(f) = \int_{\mathcal{X}} \int_{\mathcal{Y}} \left( y - f(x) \right)^2 dP(y|x) p(x) dx = \int_{\mathcal{X}} \mathbb{E} \left( (Y - f(x))^2 | x \right) p(x) dx.$$

Очевидно, минимизировать  $R(f)$  можно поточечно:

$$f^*(x) = \operatorname{argmin}_c \mathbb{E} \left( (Y - c)^2 | x \right), \quad (1)$$

откуда

$$f^*(x) = \mathbb{E}(Y|x). \quad (2)$$

Это так называемая *регрессионная функция*.

Итак, в случае квадратичной функции потерь наилучшим предсказанием  $y$  в ответ на вход  $x$  является условное среднее (регрессионная функция).

$R(f^*)$  назовем *байесовой ошибкой*, или *байесовым риском*, или *неустранимой ошибкой*.

**Упражнение 2.1** Доказать, что из (1) следует (2), при этом  $R(f^*) = \mathbb{E}_X \mathbb{D}_Y(Y|X)$ .

**Упражнение 2.2** Доказать, что если  $L(y', y) = |y' - y|$ , то минимум среднему риску доставляет условная медиана  $f^*(x) = \operatorname{median}(Y|x)$ . Чему равна при этом  $R(f^*)$ ?

## 2.2.1. Метод ближайшего соседа

Возникает задача аппроксимации условного среднего  $E(Y|x)$  по имеющейся выборке.

1) Заменим  $f^*(x)$  выборочным средним:

$$f(x) = \frac{1}{|I(x)|} \sum_{i \in I(x)} y^{(i)}, \quad \text{где} \quad I(x) = \left\{ i : x^{(i)} = x \right\},$$

Как правило, такое решение к успеху не приводит, так как обычно  $x$  встречается в обучающей выборке не более одного раза.

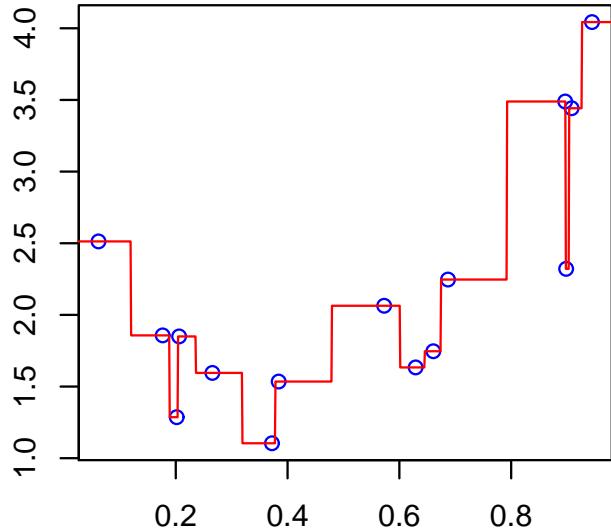
2) В методе  $k$  ближайших соседей ( $k$ NN –  $k$  nearest neighbours) вместо выборочного среднего берут

$$f(x) = \frac{1}{k} \sum_{x^{(i)} \in N_k(x)} y^{(i)},$$

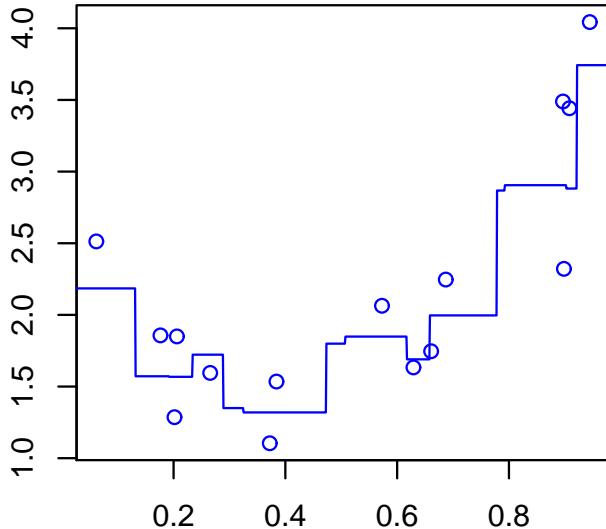
где через  $N_k(x)$  обозначено множество из  $k$  точек обучающей выборки, ближайших (например, по евклидову расстоянию) к  $x$ .

Частным случаем является метод (одного) ближайшего соседа, в котором  $f(x) = y^{(i)}$ , где  $x^{(i)}$  – ближайшая к  $x$  точка из обучающей выборки.

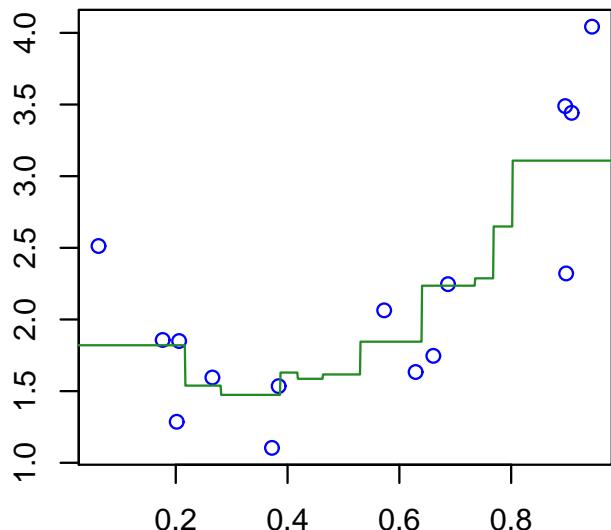
$$y = 8x^2 - 6.4x + 2.5 + N(0, 0.4)$$



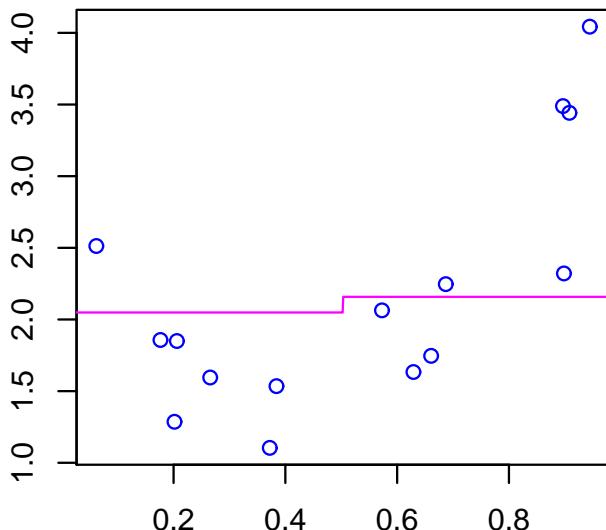
$k = 1$



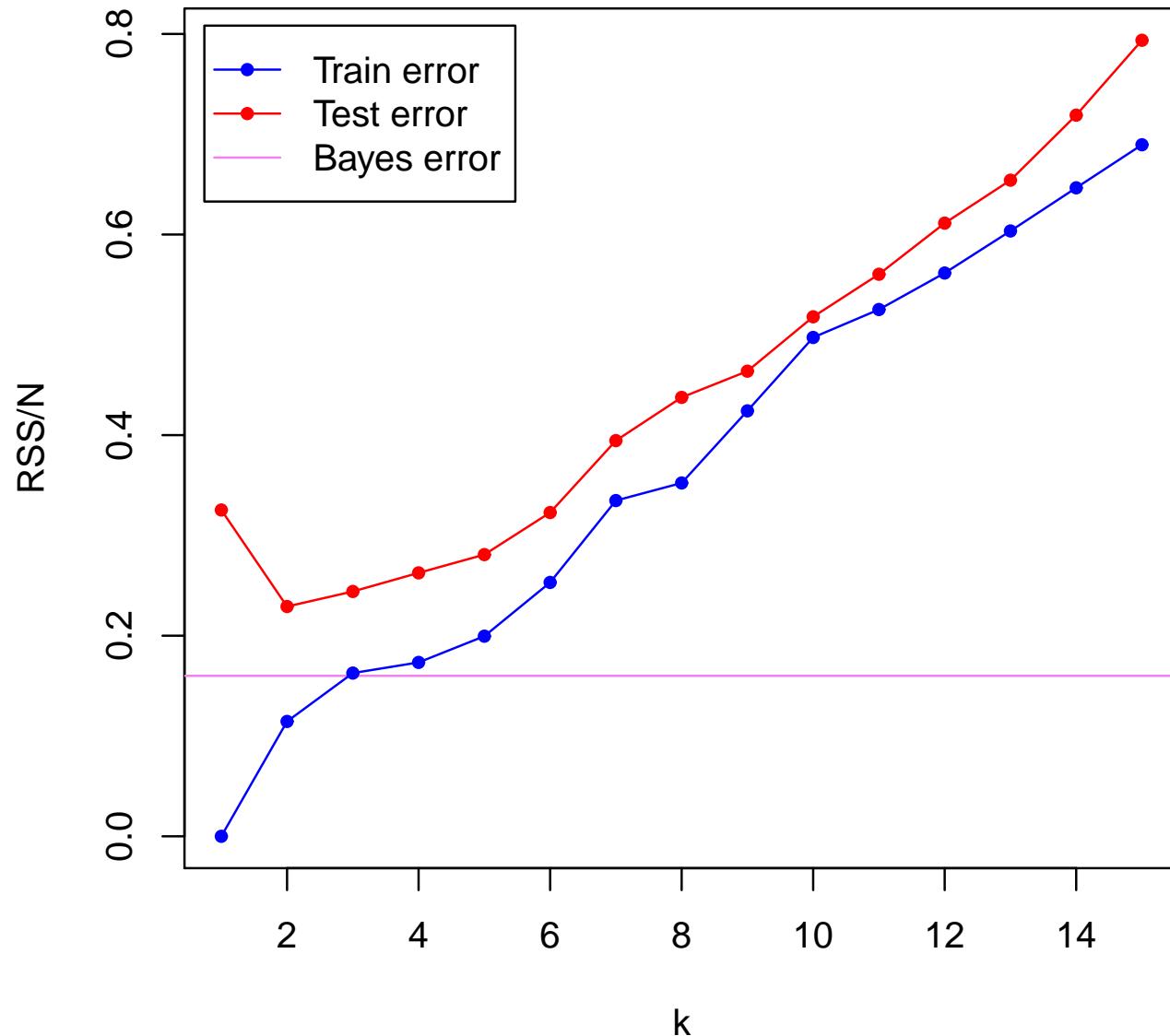
$k = 2$



$k = 5$

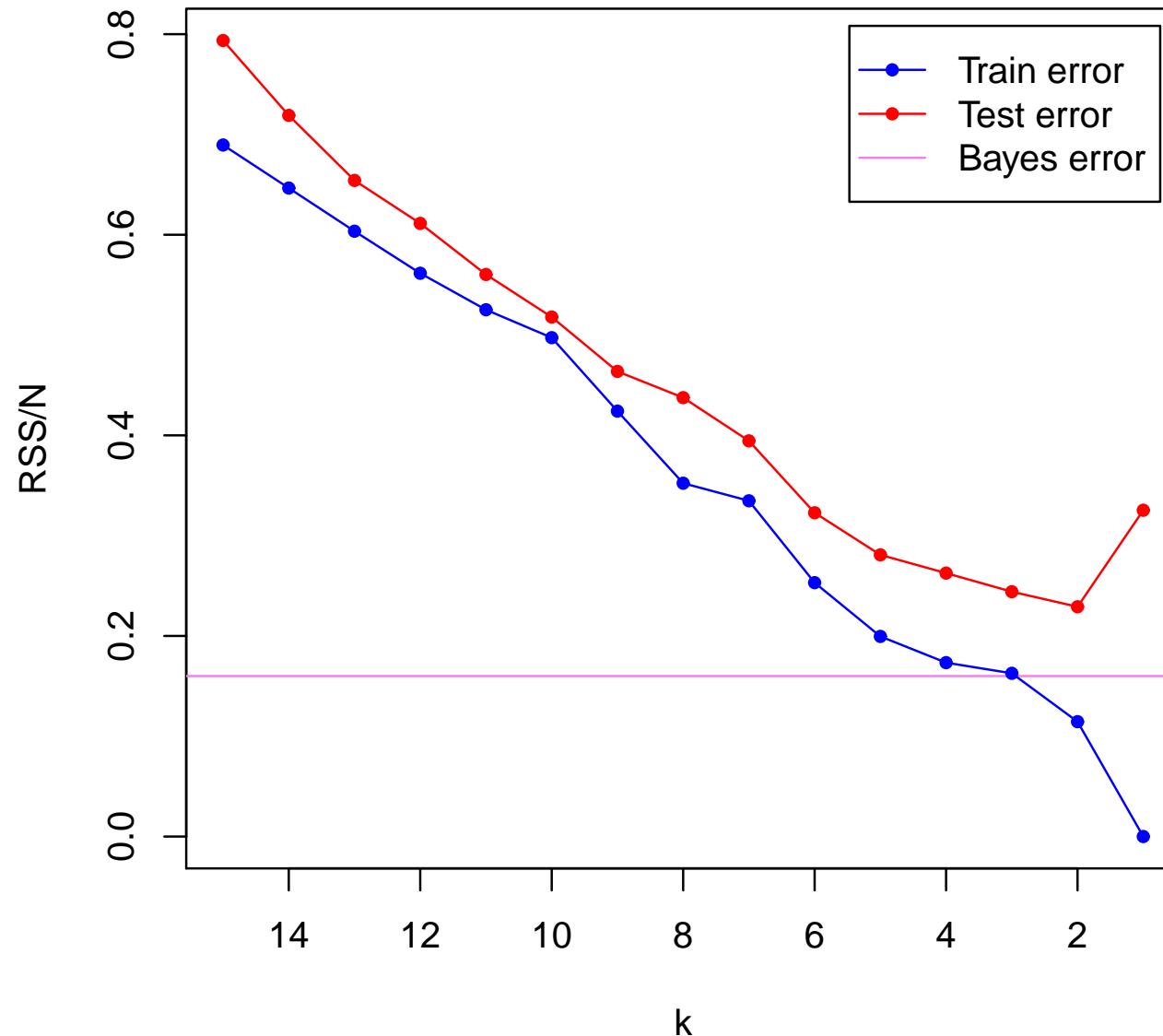


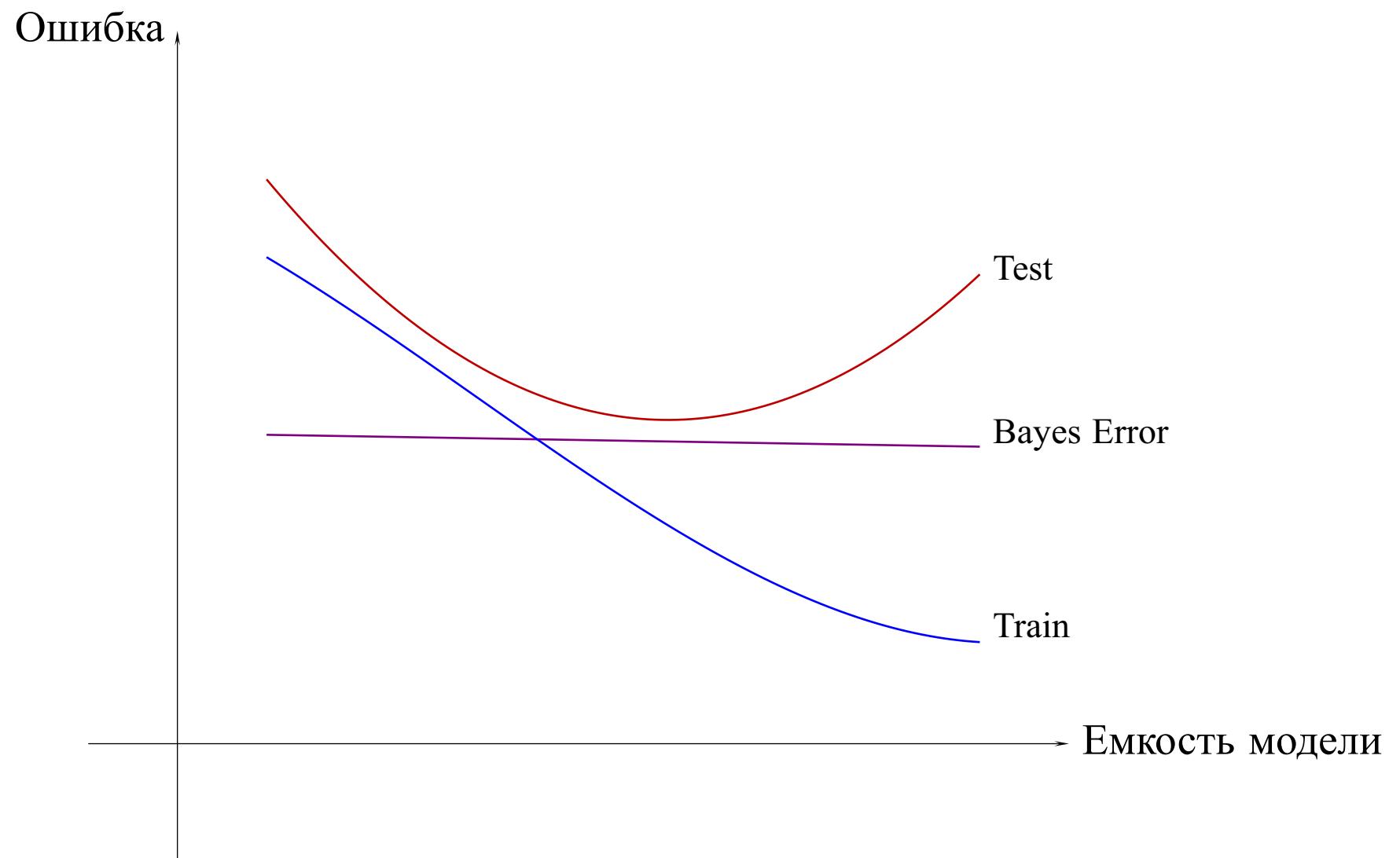
$k = 14$



$$R(f^*) = \mathbb{E}_X D_Y(Y \mid X) = \sigma^2 = 0.16$$

С увеличением  $k$  «емкость» («сложность») модели падает,  
поэтому развернем горизонтальную ось в обратном направлении  
(движение по ней вправо соответствует росту «емкости» модели)





Итак, метод ближайших соседей похож на метод восстановления функции распределения вероятности, только теперь аппроксимируется не плотность вероятности, а условное среднее.

## 2.3. Байесов классификатор

Рассмотрим задачу классификации.  $\mathcal{Y} = \{1, 2, \dots, K\}$ .

Минимизируем средний риск

$$R(f) = \int_{\mathcal{X}} \left( \sum_{y=1}^K L(f(x), y) \cdot \Pr(y|x) \right) p(x) dx. \quad (**)$$

Пусть функция потерь определяется формулой (симметричный 0–1 штраф)

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

Подынтегральная функция в  $(**)$  есть вероятность ошибки (при заданном  $x$ ),

$$R(f) = \int_{\mathcal{X}} \left( 1 - \Pr(Y = f(x) | x) \right) p(x) dx,$$

откуда находим  $f^*(x) = \operatorname{argmin} R(f)$ :

$$f^*(x) = \operatorname{argmin}_{y \in \mathcal{Y}} (1 - \Pr(y|x)),$$

$$f^*(x) = \operatorname{argmax}_{y \in \mathcal{Y}} \Pr(y|x) = \operatorname{argmax}_{y \in \mathcal{Y}} \frac{p(x|y) \Pr(y)}{p(x)} = \operatorname{argmax}_{y \in \mathcal{Y}} p(x|y) \Pr(y). \quad (+)$$

Функция  $f^*(x)$  называется *байесовым классификатором*.

Средний риск  $R(f^*)$  байесова классификатора называется *байесовой ошибкой*, или *байесовым риском*, или *неустранимой ошибкой*.

Байесов классификатор играет в задаче классификации роль, аналогичную той, которую играет регрессионная функция в задаче восстановления регрессии.

$\Pr(y)$  — *априорная вероятность* появления объекта из класса  $y$ .

$\Pr(y|x)$  — *апостериорная вероятность* появления объекта из класса  $y$ .

Правило (+) называется *принципом максимума апостериорной вероятности*.

Если классы равновероятны, т. е.  $\Pr(y) = 1/K$ , то

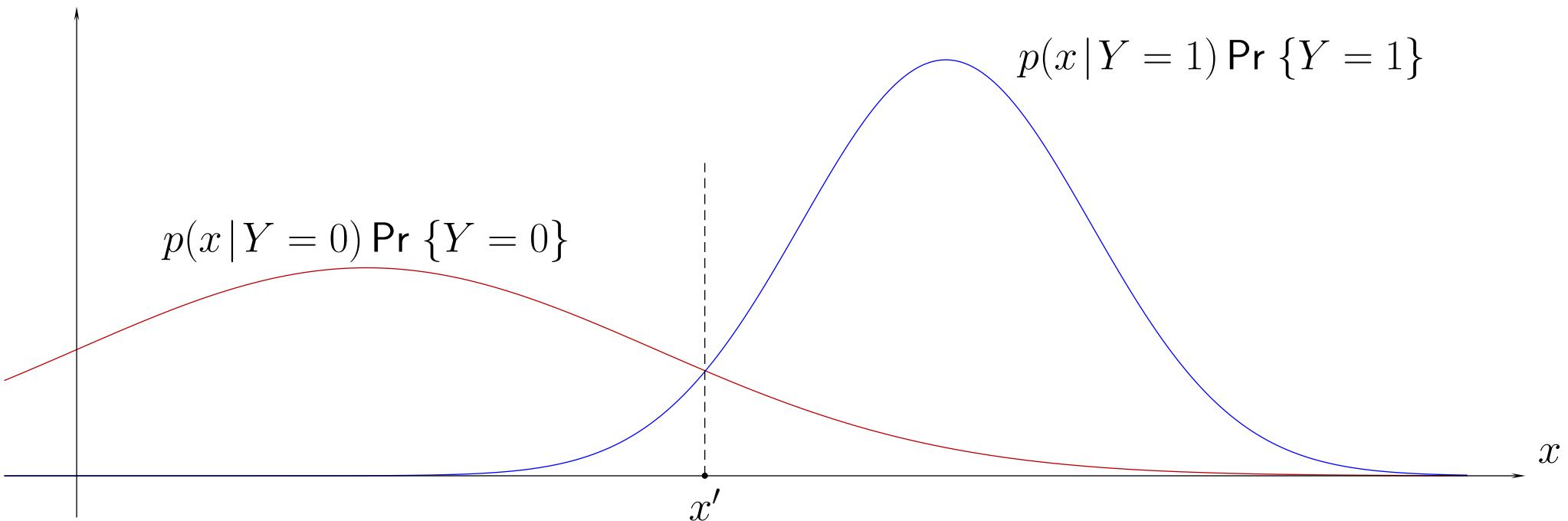
$$\Pr(y|x) = \frac{p(x|y) \Pr(y)}{p(x)} = \frac{p(x|y)}{K p(x)}$$

$$f^*(x) = \operatorname{argmax}_y p(x|y). \quad (++)$$

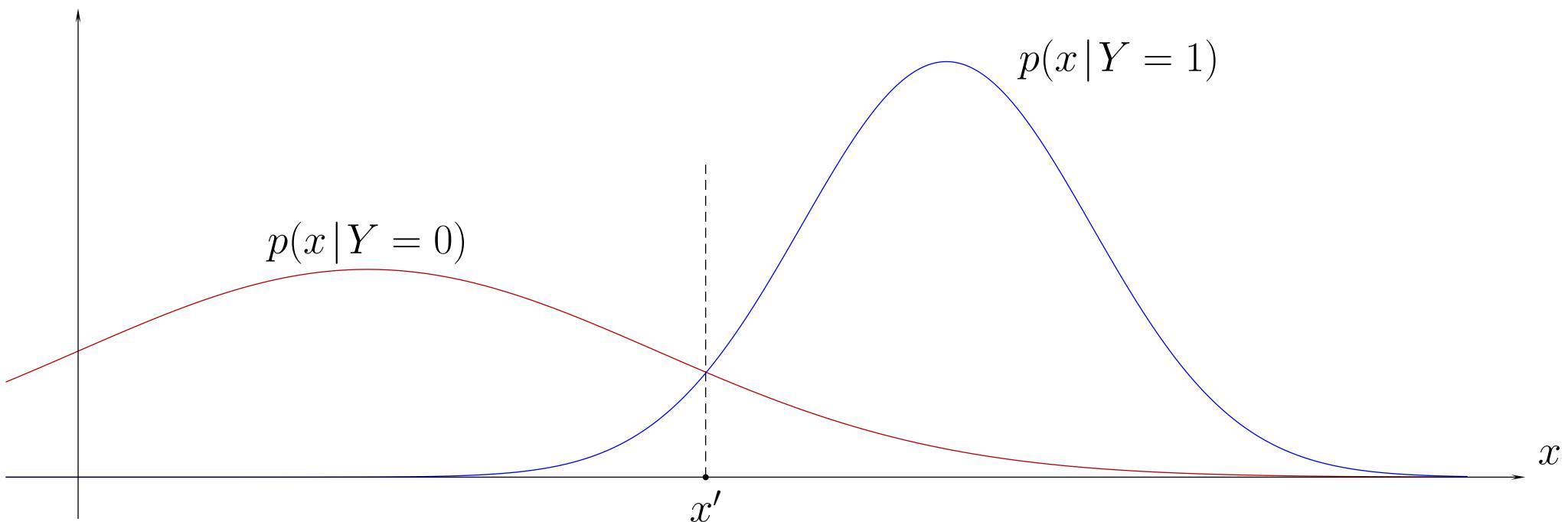
Плотность  $p(x|y)$  — *правдоподобие (likelihood)*.

Правило (++) — *метод максимального правдоподобия (maximum-likelihood method)*.

Принцип максимума апостериорной вероятности. При  $x < x'$  полагаем  $f(x) = 0$ , иначе  $f(x) = 1$ .



Принцип максимального правдоподобия. При  $x < x'$  имеем  $f(x) = 0$ , иначе  $f(x) = 1$ .



*Чтобы построить байесов классификатор, мы должны знать (или оценить)  $\Pr(y|x)$*

## **Пример**

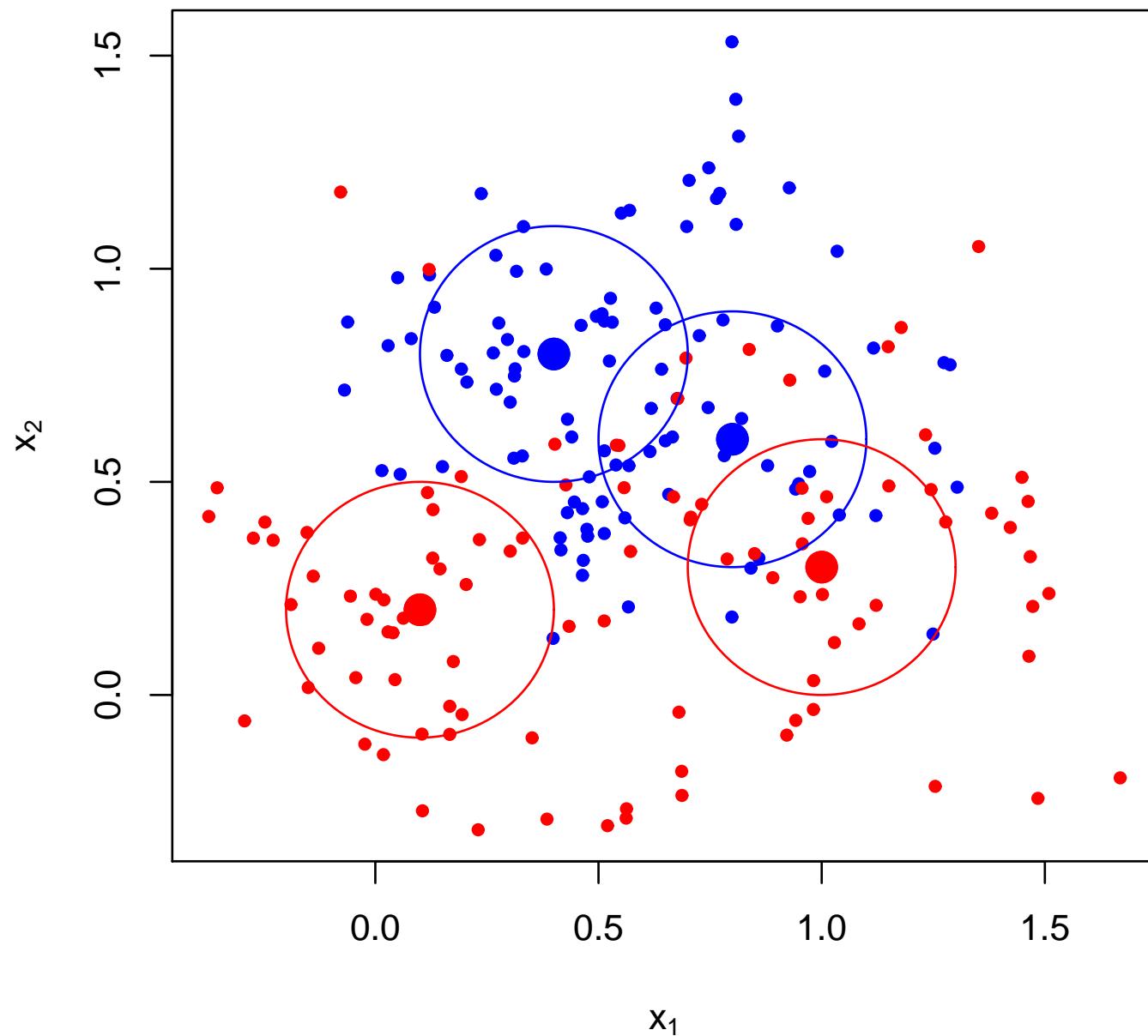
Рассмотрим обучающую выборку для задачи классификации на два класса.

Каждый класс — 100 прецедентов.

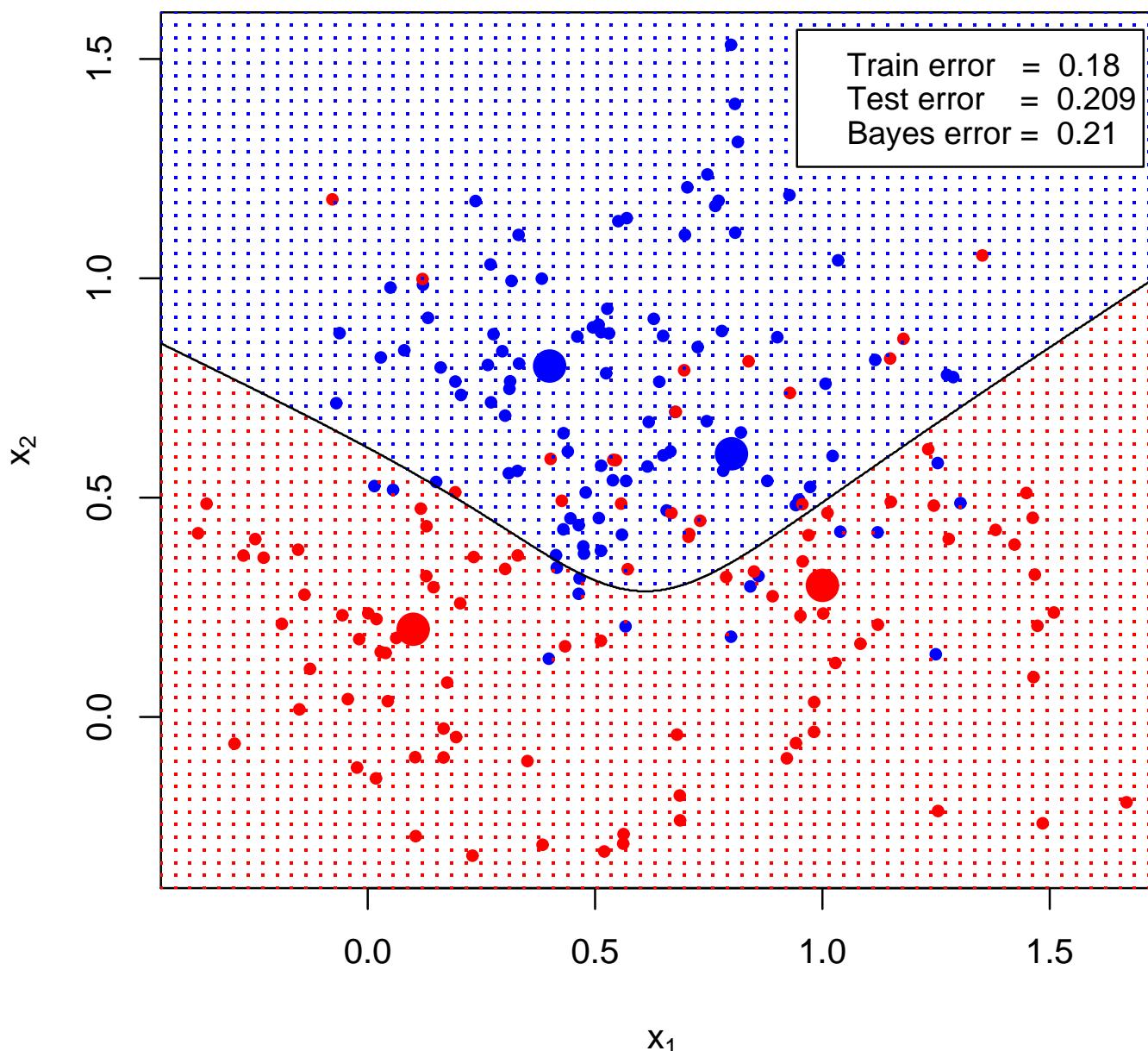
Распределение в каждом классе — смесь (взвешенная сумма) 2-х нормальных распределений (гауссианов).

$$\mu_1 = (0.4, 0.8), \mu_2 = (0.8, 0.6), \mu_3 = (1.0, 0.3), \mu_4 = (0.1, 0.2).$$

Матрица ковариации  $\sigma^2\mathbf{I}$ , где  $\sigma = 0.3$ .



Распределение известно, поэтому байесову ошибку можем найти точно.



Таким образом, байесов классификатор — это оптимальный классификатор.

Предполагается, что условные вероятности  $\Pr(y|x)$  известны.

Как это можно использовать на практике?

Будем аппроксимировать  $\Pr(y|x)$

- 1) Метод ближайших соседей (для задачи классификации)
- 2) Восстановление условной плотности вероятности

### 2.3.1. Метод ближайших соседей для задачи классификации

Будем, как и в задаче восстановления регрессии, для аппроксимации  $\Pr(y|x)$  использовать  $k$  ближайших (по некоторому, например, евклидову расстоянию) объектов из обучающей выборки. Получаем метод  $k$  ближайших соседей для задачи классификации.

Пусть  $N_k(x)$  — множество из  $k$  ближайших к  $x$  (по евклидову расстоянию) точек из обучающей выборки,

$I_k(x, y)$  — множество тех точек  $x^{(i)}$  из  $N_k(x)$ , для которых  $y^{(i)} = y$ .

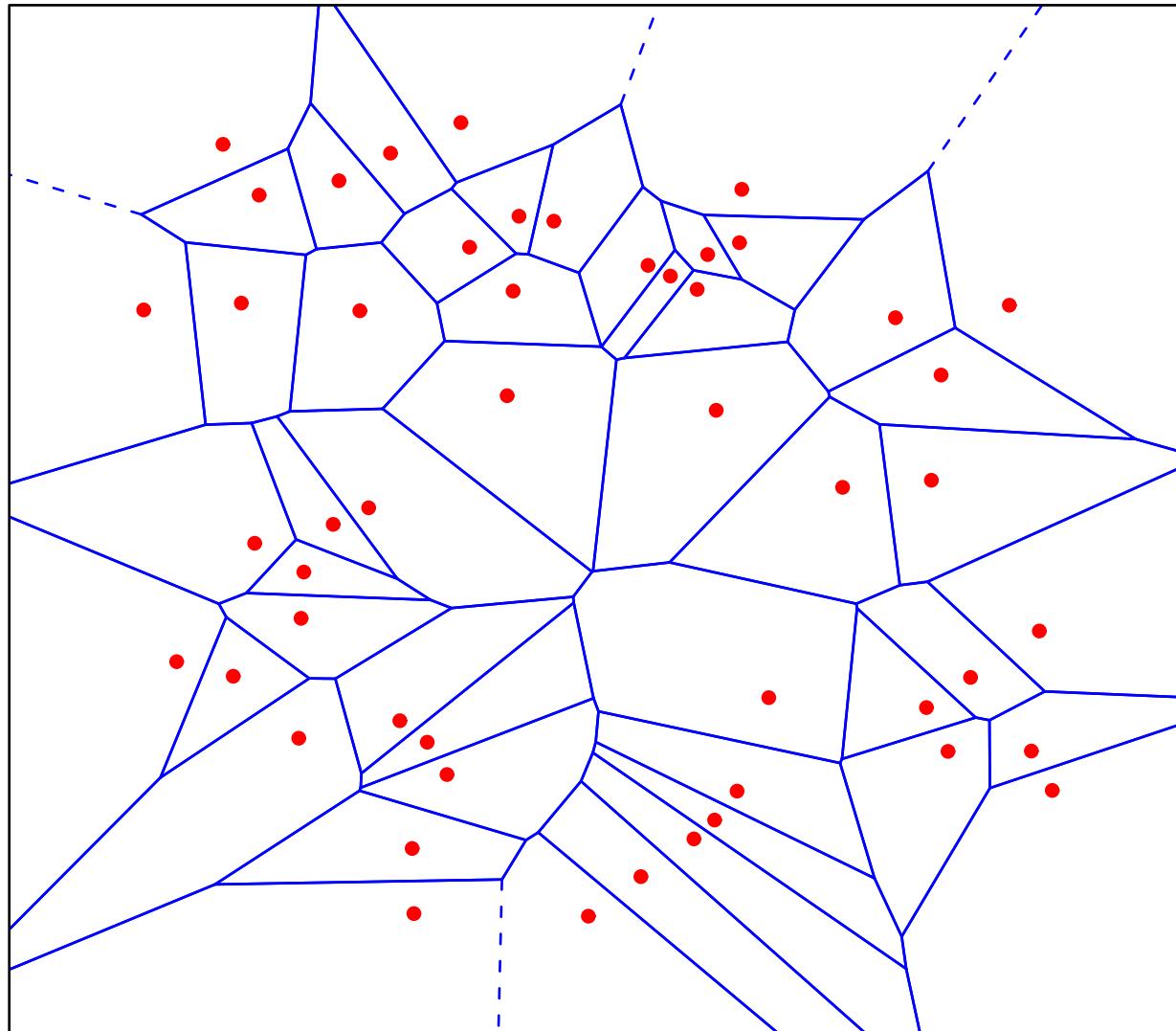
Согласно методу  $k$  ближайших соседей ( $k$ NN —  $k$  nearest neighbours) в качестве  $f(x)$  берем результат голосования по всем точкам из  $I_k(x, y)$ :

$$f(x) = \operatorname{argmax}_y |I_k(x, y)|,$$

Частным случаем является метод (одного) ближайшего соседа, в котором  $f(x) = y^{(i)}$ , где  $x^{(i)}$  — ближайший к  $x$  объект из обучающей выборки.

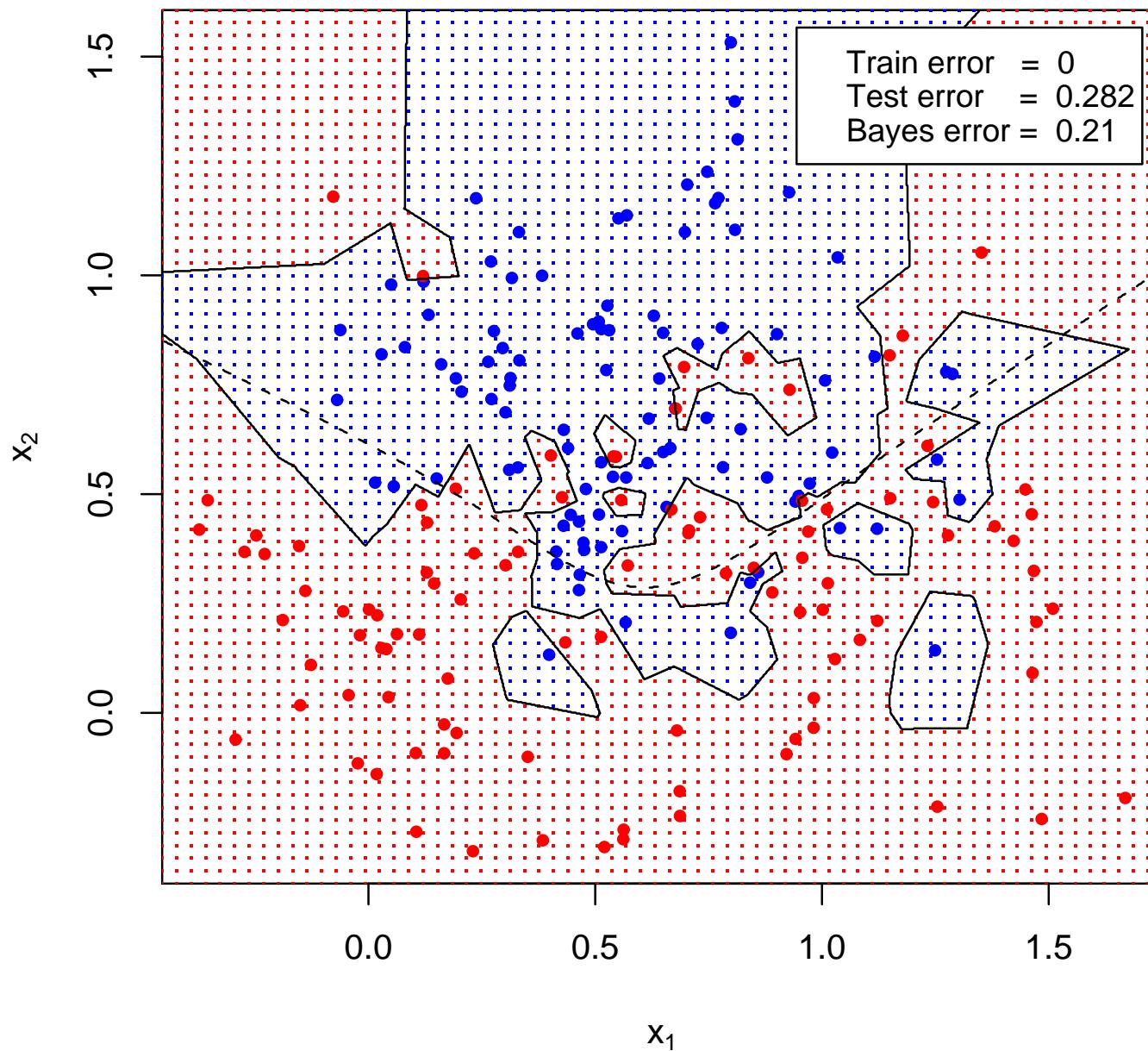
В этом случае  $D_y$  представляют собой области Вороного

Диаграмма Вороного для набора из 50 точек. Штриховыми линиями отмечены неограниченные участки границы

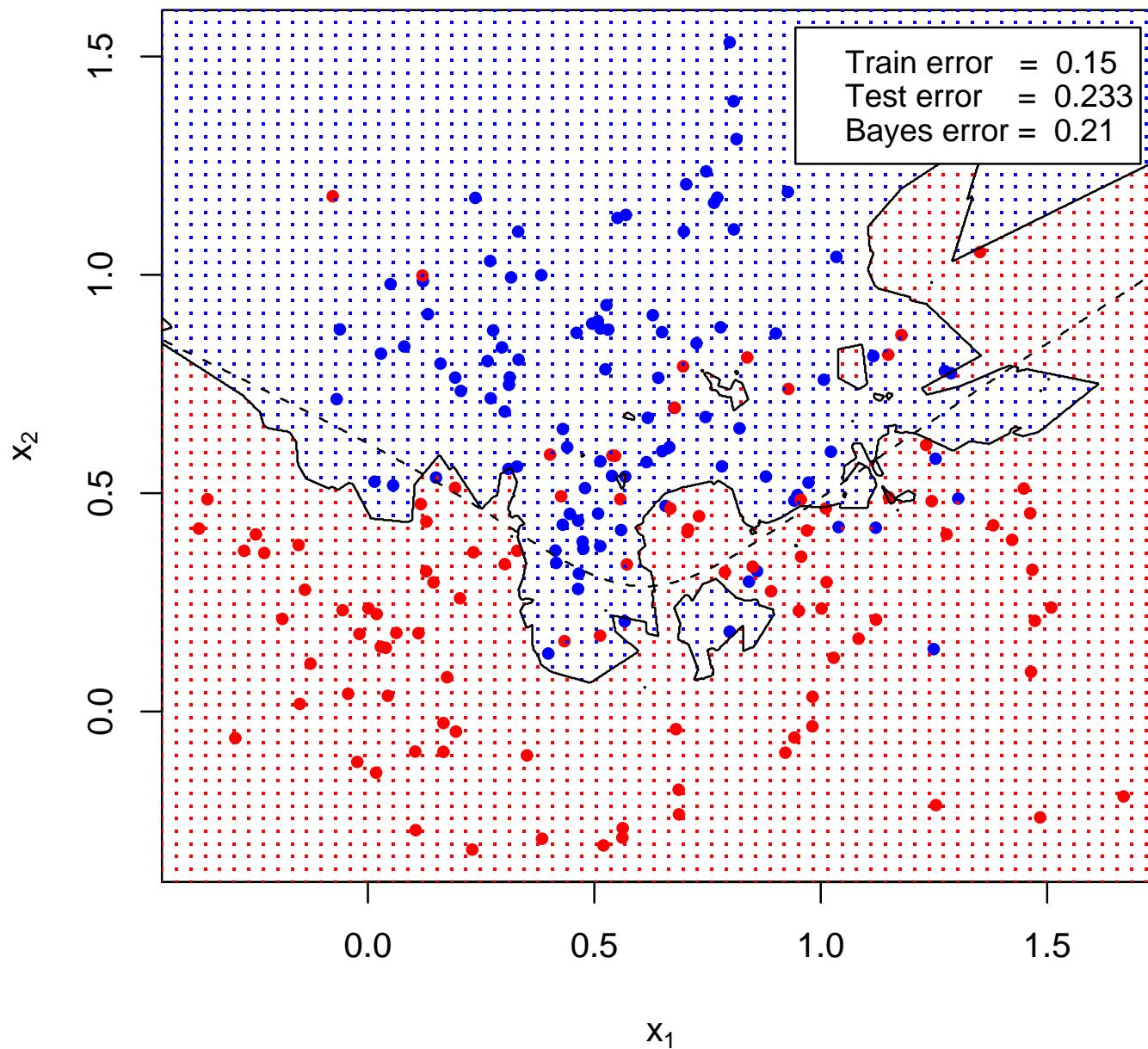


Все сильно зависит от масштаба!

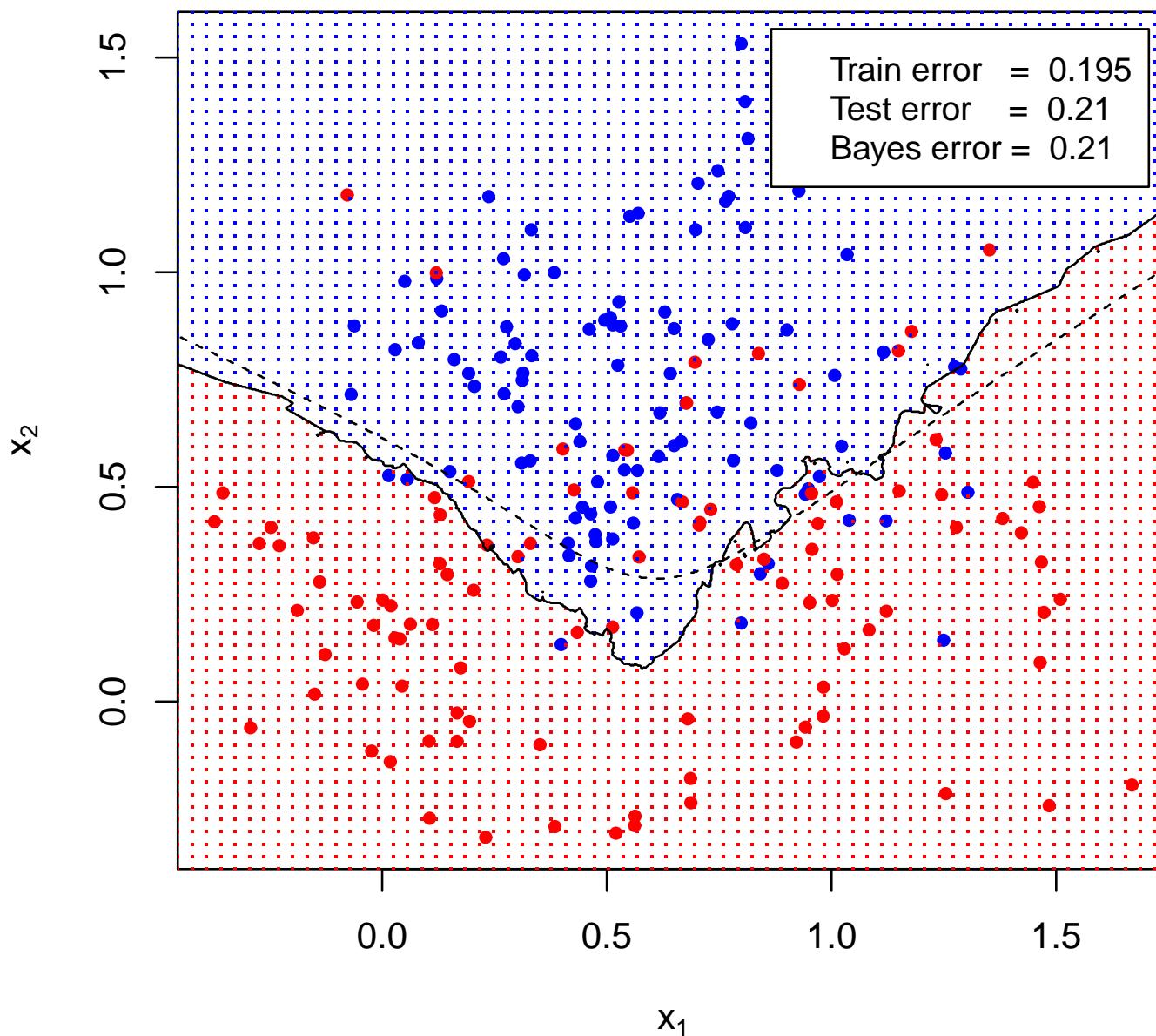
## Метод одного ближайшего соседа



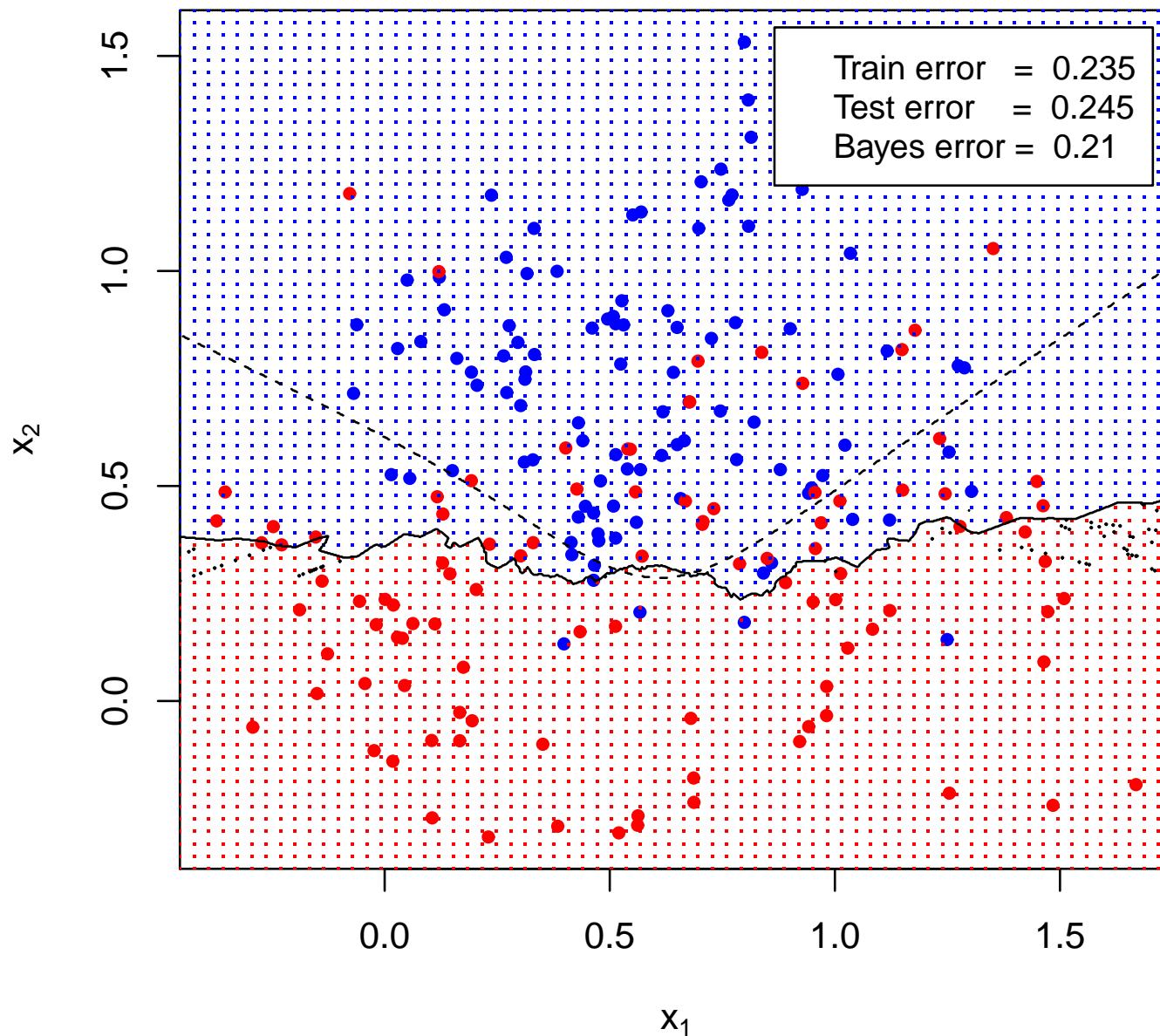
## Метод 5 ближайших соседей



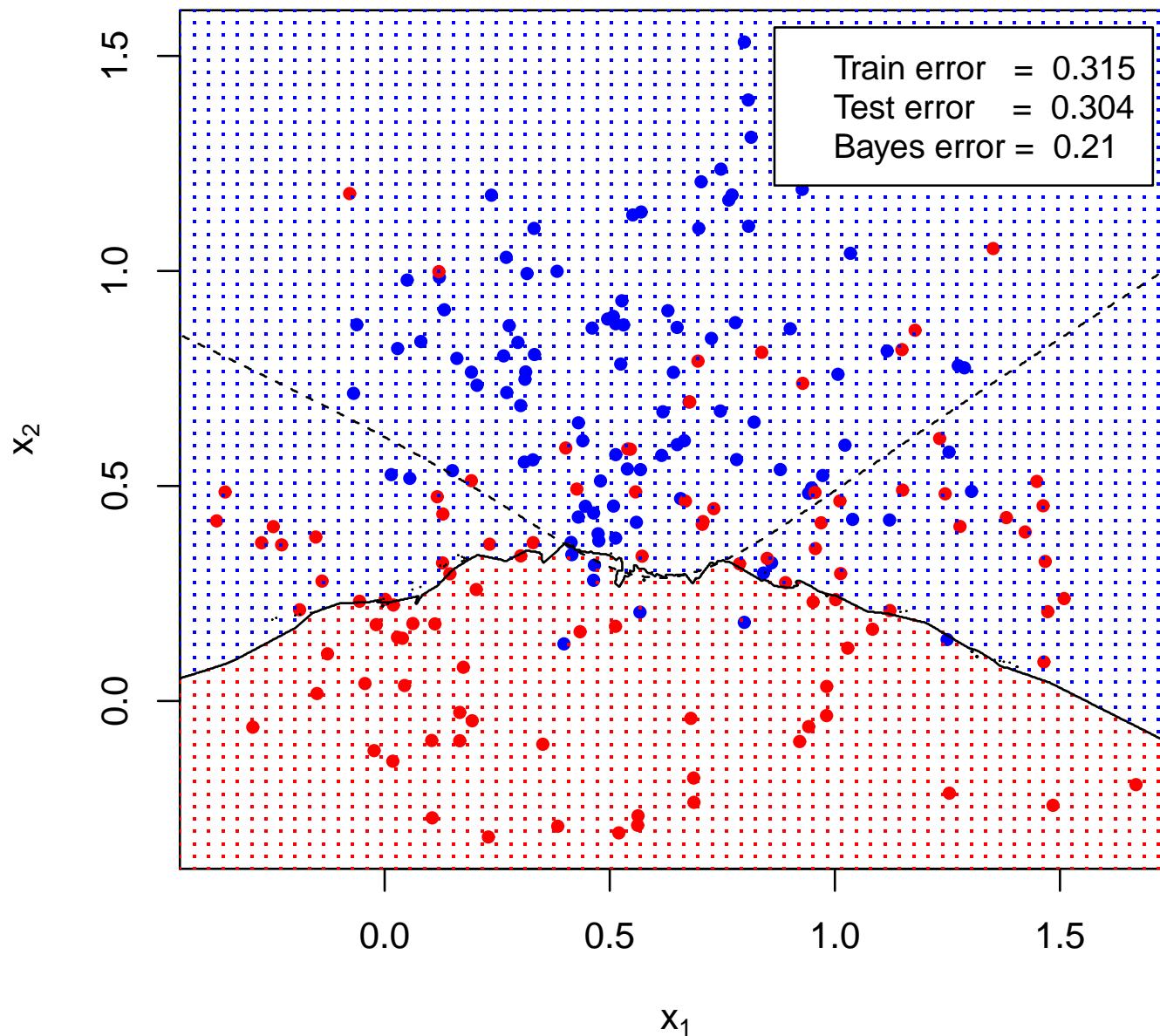
## Метод 31 ближайшего соседа

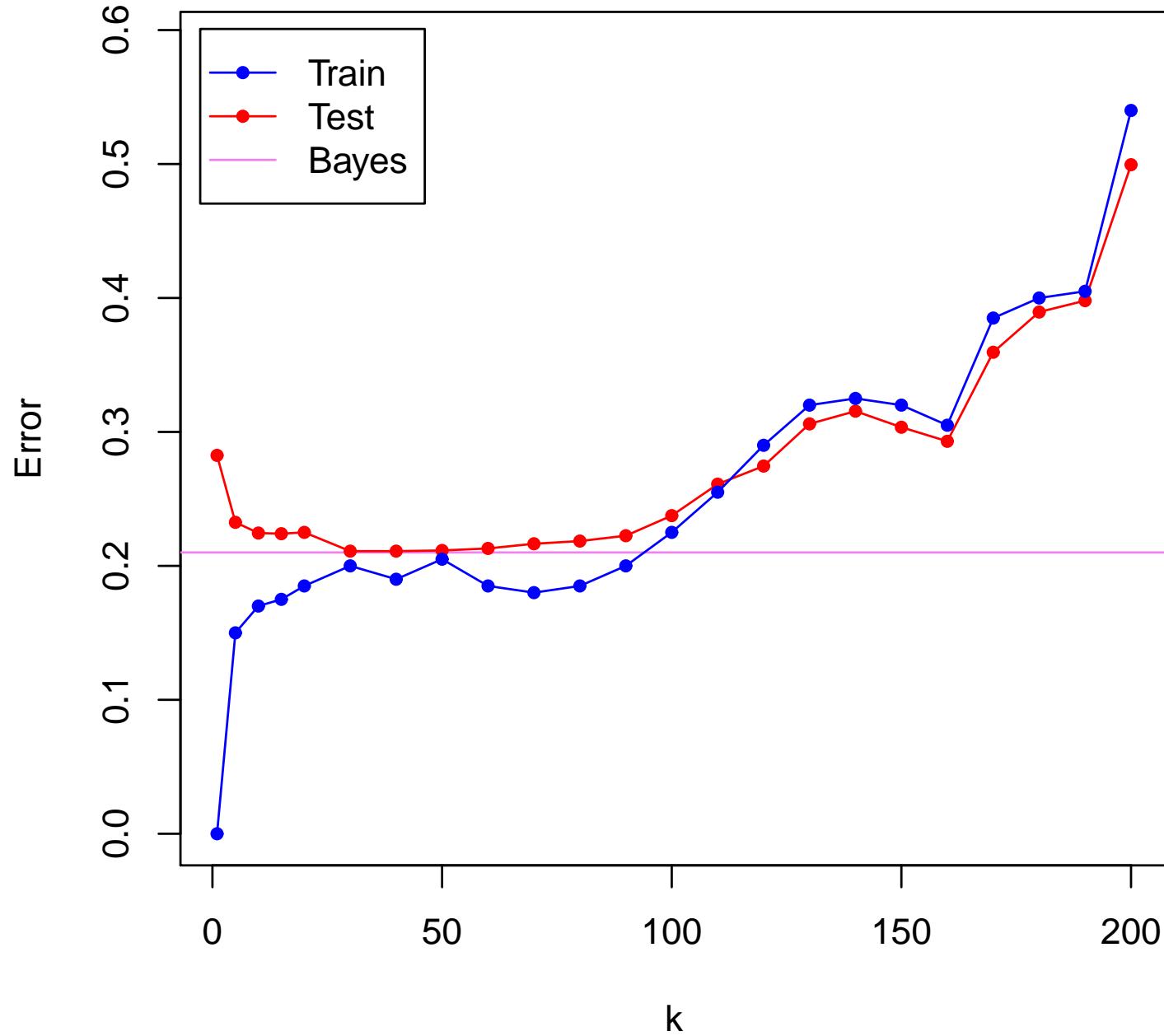


# 101 ближайший соседа

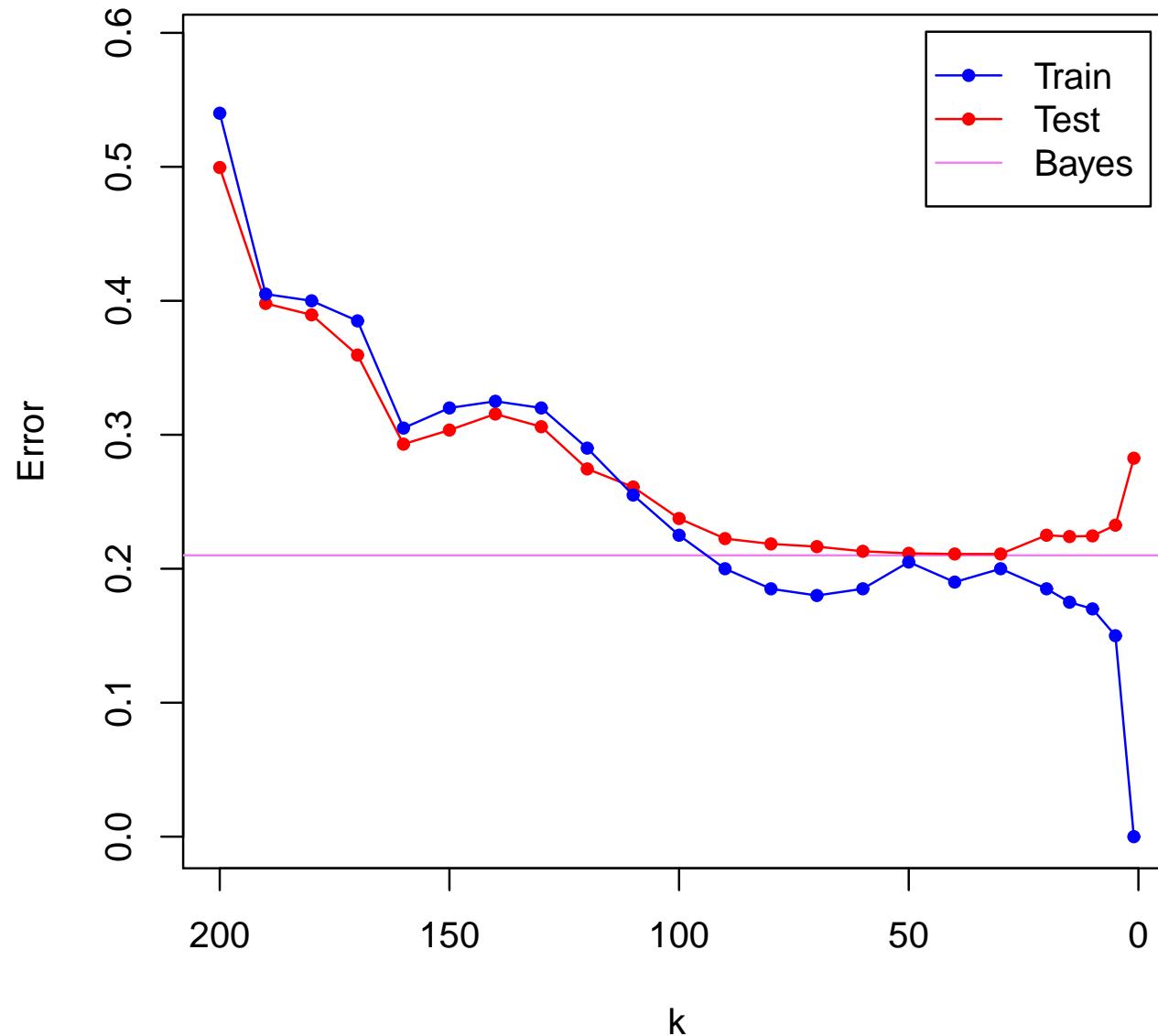


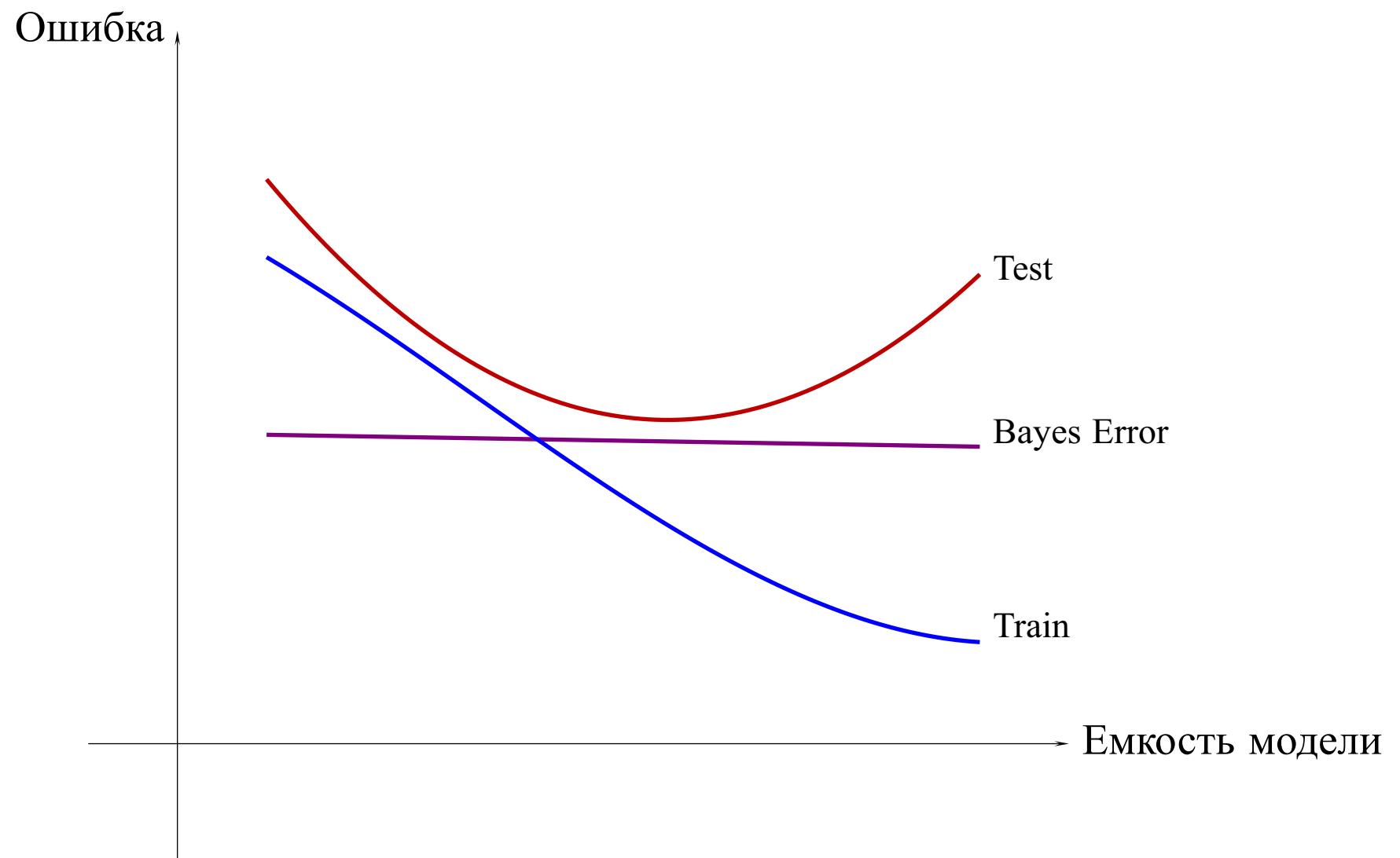
# 151 ближайший соседа



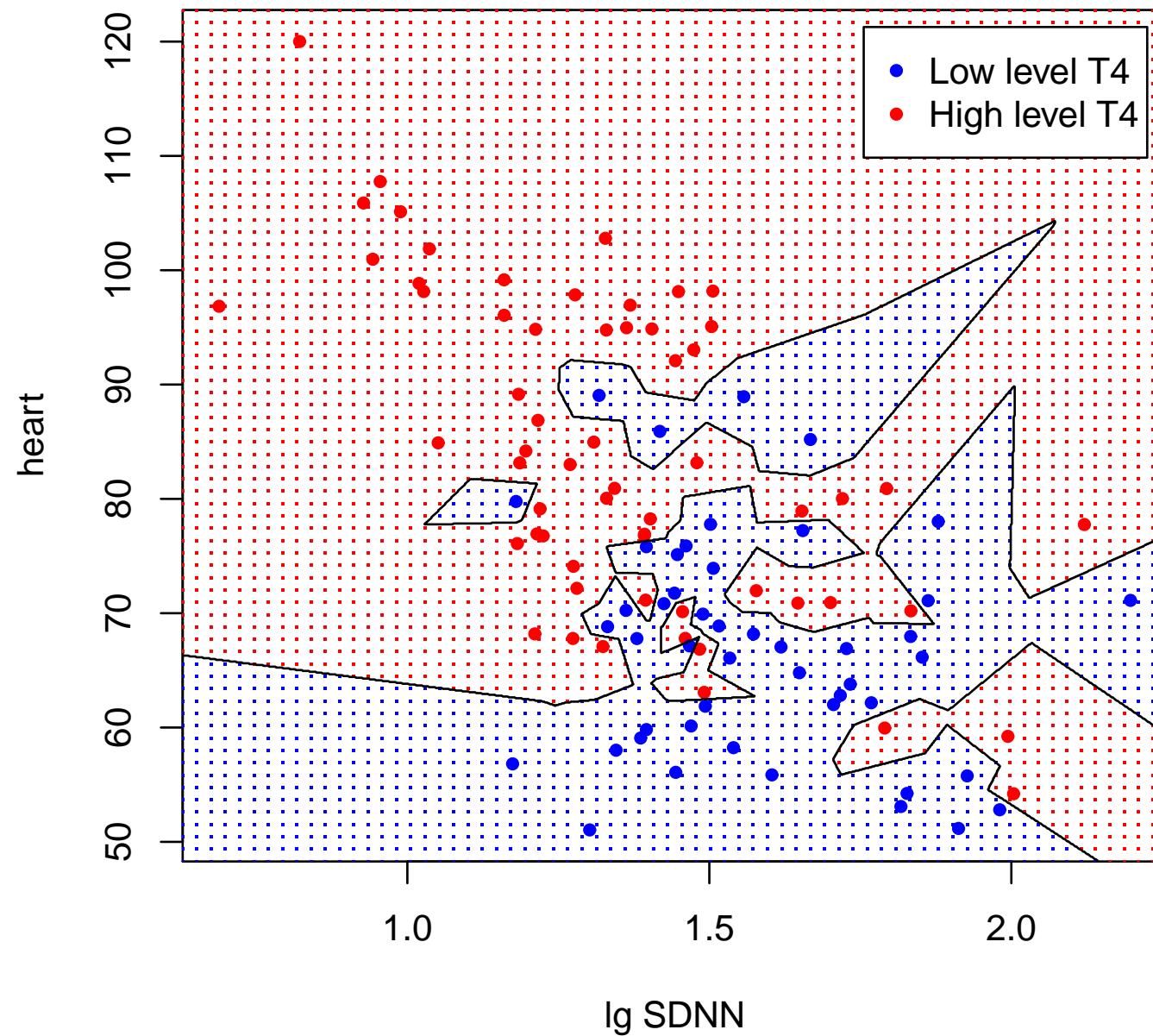


С увеличением  $k$  «емкость» («сложность») модели падает,  
поэтому развернем горизонтальную ось в обратном направлении  
(движение по ней вправо соответствует росту «емкости» модели)

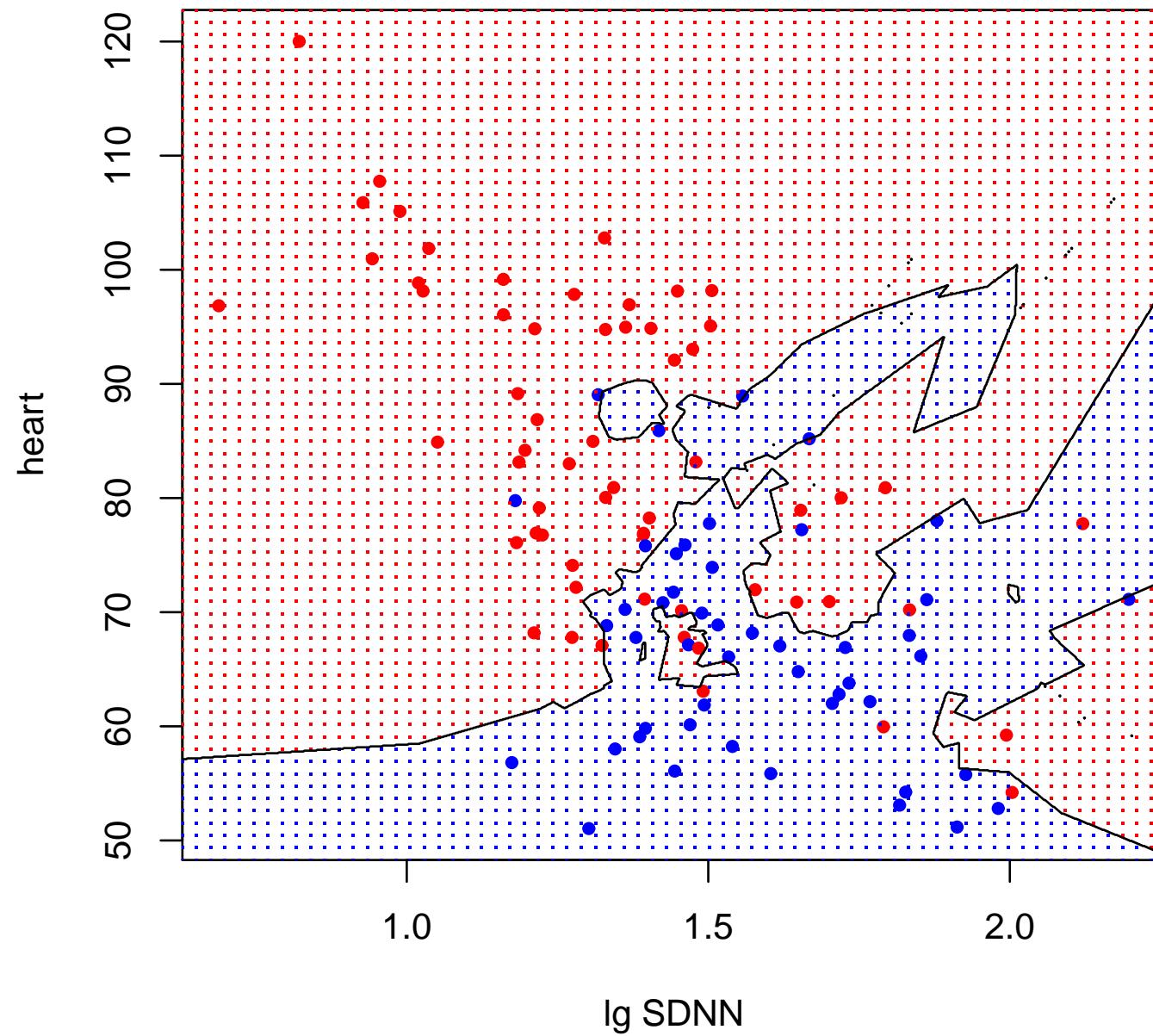




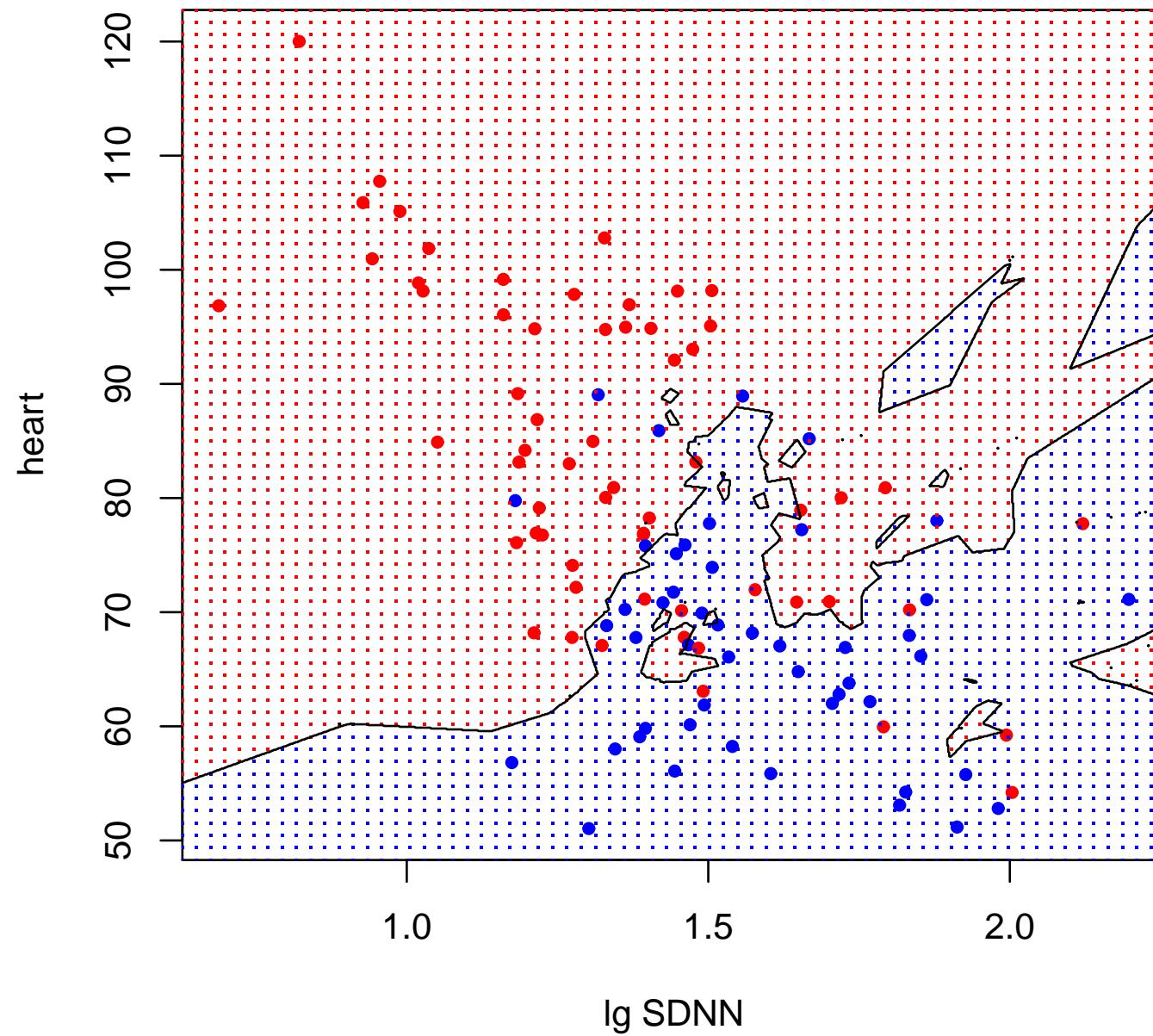
Задача медицинской диагностики. Метод 1 ближайшего соседа. 10-CV ошибка 0.30



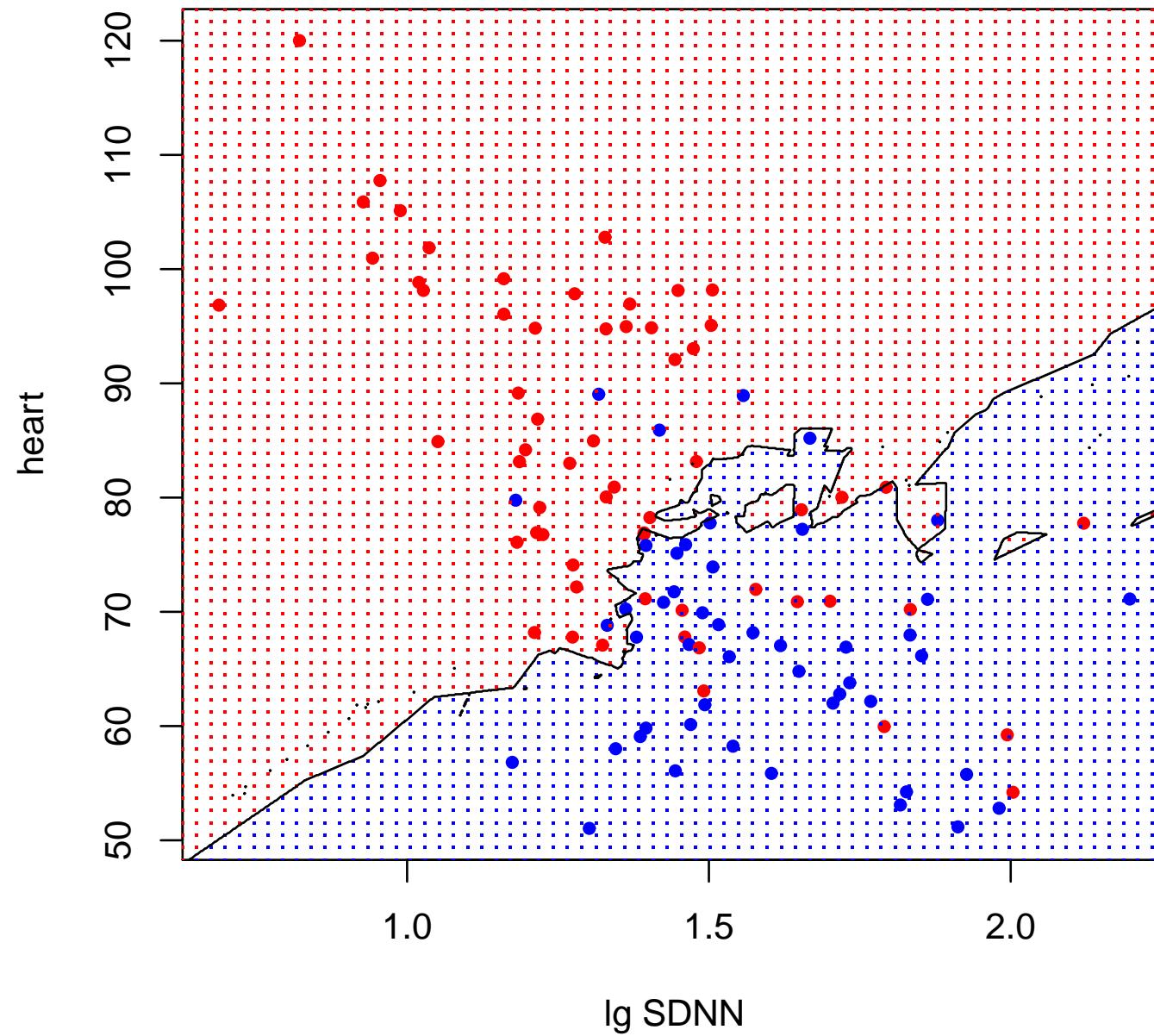
Метод 3 ближайших соседей 10-CV ошибка 0.26



Метод 5 ближайших соседей 10-CV ошибка 0.27



Метод 15 ближайших соседей 10-CV ошибка 0.25



## 2.3.2. Теорема об оценке риска

$$L(f(x), y) = \begin{cases} 0, & \text{если } f(x) = y, \\ 1, & \text{если } f(x) \neq y. \end{cases}$$

При достаточно большом объеме обучающей выборки средний риск классификатора одного ближайшего соседа не более чем в 2 раза превосходит байесов риск, а именно, справедлива теорема:

**Теорема 2.3 (Cover, Hart, 1967)** Пусть  $R^*$  – оптимальное (байесовское) значение среднего риска для некоторой задачи классификации на  $K$  классов. Тогда с ростом размера выборки  $N$  ожидаемый риск  $R$  для метода одного ближайшего соседа сходится к  $R^o$ , такому, что

$$R^* \leq R^o \leq R^* \cdot \left(2 - \frac{K}{K-1}R^*\right) \leq 2R^*.$$

ДОКАЗАТЕЛЬСТВО. (набросок)

$X'$  – объект из обучающей выборки, ближайший к  $X$ , а  $Y'$  – соответствующий выход

$k^* = \operatorname{argmax} \Pr(k|x)$  – байесово решение (самый популярный класс) в точке  $x$

$r^*(x) = 1 - \Pr(k^*|x)$  – предельный условный байесовский риск

$r(x)$  – предельный условный средний риск классификатора ближайшего соседа:

$$\begin{aligned}
r(x) &= \lim_{N \rightarrow \infty} \Pr(Y \neq Y'|x) = \lim_{N \rightarrow \infty} \int \Pr(Y \neq Y'|x, x') p(x'|x) dx' = \\
&= \lim_{N \rightarrow \infty} \int_{\mathcal{X}} \left( 1 - \sum_{k=1}^K \Pr(k|x) \Pr(k|x') \right) p(x'|x) dx' = \\
&= \int_{\mathcal{X}} \left( 1 - \sum_{k=1}^K \Pr(k|x) \Pr(k|x') \right) \delta(x' - x) dx' = 1 - \sum_{k=1}^K \Pr^2(k|x) = \\
&= 1 - \Pr^2(k^*|x) - \sum_{k \neq k^*} \Pr^2(k|x) \leq 1 - \Pr^2(k^*|x) - \frac{1}{K-1} \left( \sum_{k \neq k^*} \Pr(k|x) \right)^2 = \\
&= 1 - \Pr^2(k^*|x) - \frac{1}{K-1} \left( 1 - \Pr(k^*|x) \right)^2 = 1 - \left( 1 - r^*(x) \right)^2 - \frac{1}{K-1} r^*(x) = 2r^*(x) - \frac{K}{K-1} r^*(x)^2.
\end{aligned}$$

Умножая обе части неравенства на  $p(x)$  и интегрируя по  $x$ , получаем требуемое. ■

**Замечание 2.4** Верхние и нижние оценки в теореме являются точными. В частности, верхняя оценка

$$R^o \leq R^* \cdot \left( 2 - \frac{K}{K-1} R^* \right)$$

достигается в случае, когда плотности вероятности  $p(x|y)$  не зависят от  $y$ .

В этом случае  $\Pr\{y|x\} = \Pr\{y\}$  и  $r^*(x)$  не зависят от  $x$ .

**Замечание 2.5** К сожалению, сходимость  $R$  к  $R^o$  может быть очень медленной и сложно определить, насколько  $R$  близка к  $R^o$ .

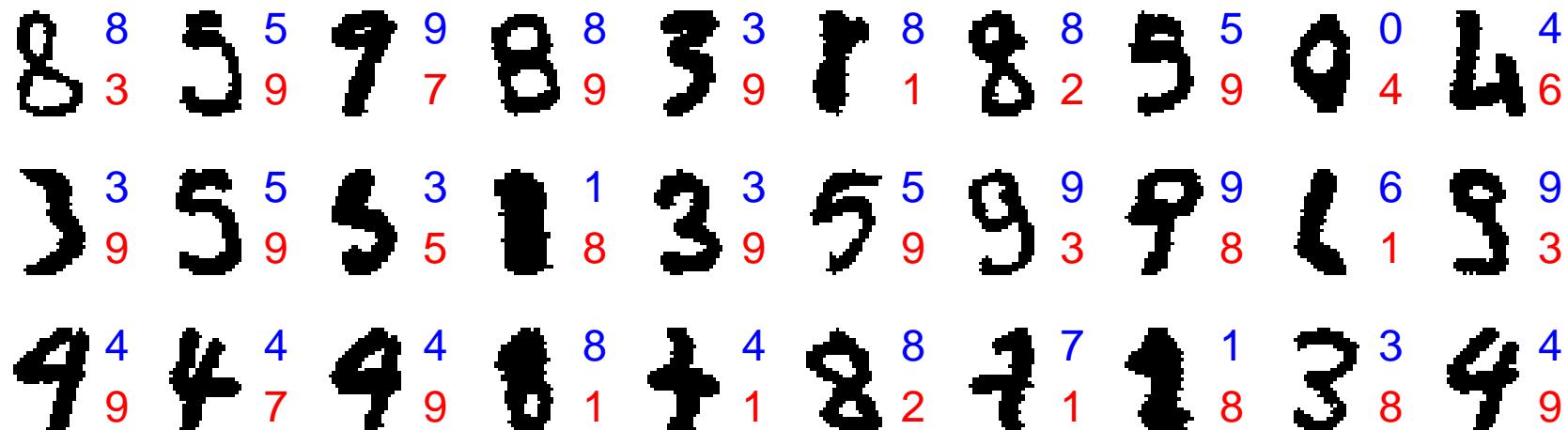
**Замечание 2.6** Результаты аналогичны для метода  $k$  ближайших соседей.

## Задача распознавания рукописных цифр

Выборка размера 1934 была случайным образом разбита на две группы: обучающую и тестовую — по 967 объектов в каждой.

$k$	<i>Ошибка</i>	
	<i>на обучающей выборке</i>	<i>на тестовой выборке</i>
1	0	0.031
2	0.017	0.046
3	0.014	0.031
5	0.026	0.033
10	0.034	0.038
15	0.042	0.046
20	0.051	0.050

Все случаи неправильной классификации цифр из тестовой выборки в случае  $k = 1$ .  
Красная цифра — ответ классификатора, синяя — верный ответ.



Как правило, метод ближайшего соседа имеет проблемы при большой размерности (если признаки количественные).

**Упражнение 2.7** Пусть  $N$  точек распределены случайно равномерно в единичной  $d$ -мерной гиперсфере. Доказать, что медианное расстояние от центра сферы до ближайшей точки равно

$$\rho(d, N) = \sqrt[d]{1 - \sqrt[N]{1/2}}.$$

Доказать, что предел  $\rho(d, N)$  при  $d \rightarrow \infty$ ,  $N = O(d)$  равен 1.

Например,  $\rho(100, 1000) = 0.93$ , т. е. в половине всех случаев *все* точки лежат на расстоянии  $\leq 0.07$  от границы.

## 2.4. Плюсы и минусы метода $k$ NN

### *Плюсы*

- Простой метод
- Для ряда задач показывает неплохие результаты
- Достаточно устойчив к выбросам (при подходящем выборе  $k$ )
- Работает как с числовыми, так и номинальными признаками
- Быстрый (если использовать специальные структуры данных:  $kd$ -деревья и т.д.)

### *Минусы*

- Сколько соседей брать?
- Какую метрику использовать?
- Необходимо хранить всю выборку
- Подвержен «проклятию размерности»

## 2.5. Экспериментальная оценка качества обучения и выбор модели

### 2.5.1. Оценка качества обучения

model assessment (model evaluation)

*Алгоритм обучения* по обучающей выборке  $D \in \mathcal{D}$  строит решающую функцию  $f \in \mathcal{F}$

Мы можем вычислить эмпирический риск  $\widehat{R}(f)$ , но нужно уметь оценивать  $R(f)$

Как правило,  $\widehat{R}(f) < R(f)$

Как оценить  $R(f)$ ?

Случайно разделим все имеющиеся данные

- на обучающую (train) выборку  $(x_{\text{train}}^{(1)}, y_{\text{train}}^{(1)}), (x_{\text{train}}^{(2)}, y_{\text{train}}^{(2)}), \dots, (x_{\text{train}}^{(N_{\text{train}})}, y_{\text{train}}^{(N_{\text{train}})})$ ,
- на тестовую (test) выборку  $(x_{\text{test}}^{(1)}, y_{\text{test}}^{(1)}), (x_{\text{test}}^{(2)}, y_{\text{test}}^{(2)}), \dots, (x_{\text{test}}^{(N_{\text{train}})}, y_{\text{test}}^{(N_{\text{test}})})$ .



Обучающая выборка используется для построения моделей  $f \in \mathcal{F}$ .

Тестовая — для оценки среднего риска  $R(f)$  решающей функции  $f$ .

$$\widehat{R}(f) = \widehat{R}_{\text{train}}(f) = \frac{1}{N_{\text{train}}} \sum_{i=1}^{N_{\text{train}}} L(f(x_{\text{train}}^{(i)}), y_{\text{train}}^{(i)})$$

$$\widehat{R}_{\text{test}}(f) = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} L(f(x_{\text{test}}^{(i)}), y_{\text{test}}^{(i)})$$

$$\widehat{R}_{\text{train}}(f) < R(f) \approx \widehat{R}_{\text{test}}(f)$$

## Метод перекрестного контроля. 1-й вариант

Разбиение на обучающую и тестовую выборки можно проделать  $L$  раз (каждый раз случайно).

В результате получим  $L$  решающих функций:  $f_1, f_2, \dots, f_L$

и  $L$  значений эмпирического риска:  $\widehat{R}_1^{\text{test}}(f_1), \widehat{R}_2^{\text{test}}(f_2), \dots, \widehat{R}_L^{\text{test}}(f_L)$

Окончательную модель  $f$  построим *на основе всех данных*.

Оцениваем риск:

$$R(f) \approx \overline{R} = \frac{1}{L} \sum_{\ell=1}^L \widehat{R}_\ell^{\text{test}}(f_\ell).$$

Очевидно, что оценка несмещенная.

Построение интервальных оценок для оценки риска в этом методе — пока нерешенная задача.

## Метод перекрестного контроля по $M$ блокам ( $M$ -fold CV)

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:



В результате получим  $M$  решающих функций:  $f_1, f_2, \dots, f_M$   
и  $M$  значений эмпирического риска:  $\widehat{R}_1^{\text{cv}}(f_1), \widehat{R}_2^{\text{cv}}(f_2), \dots, \widehat{R}_M^{\text{cv}}(f_M)$

Окончательную решающую функцию  $f$  построим на основе *всех* данных.

$$R(f) \approx \widehat{R}^{\text{cv}} = \frac{1}{M} \sum_{m=1}^M \widehat{R}_m^{\text{cv}}(f_m)$$

Всю процедуру перекрестного контроля можно повторить  $L$  раз, каждый раз разбивая случайно выборку на  $M$  частей.

Тогда оценка риска будут вычисляться на основе  $M \cdot L$  оценок эмпирического риска

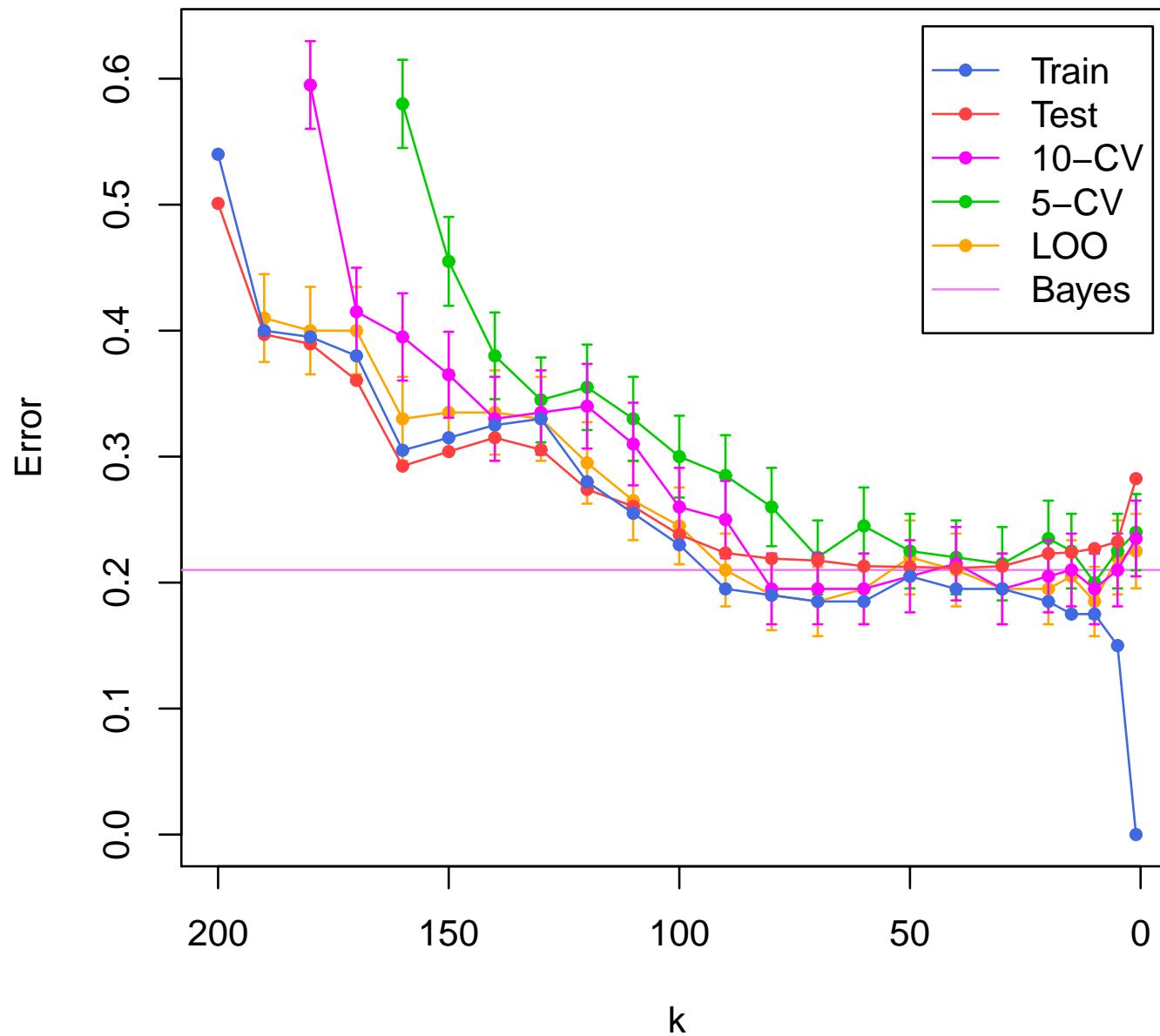
### **Какое $M$ выбирать?**

Часто используемые значения  $M$ :  $M = 5$  или  $M = 10$ .

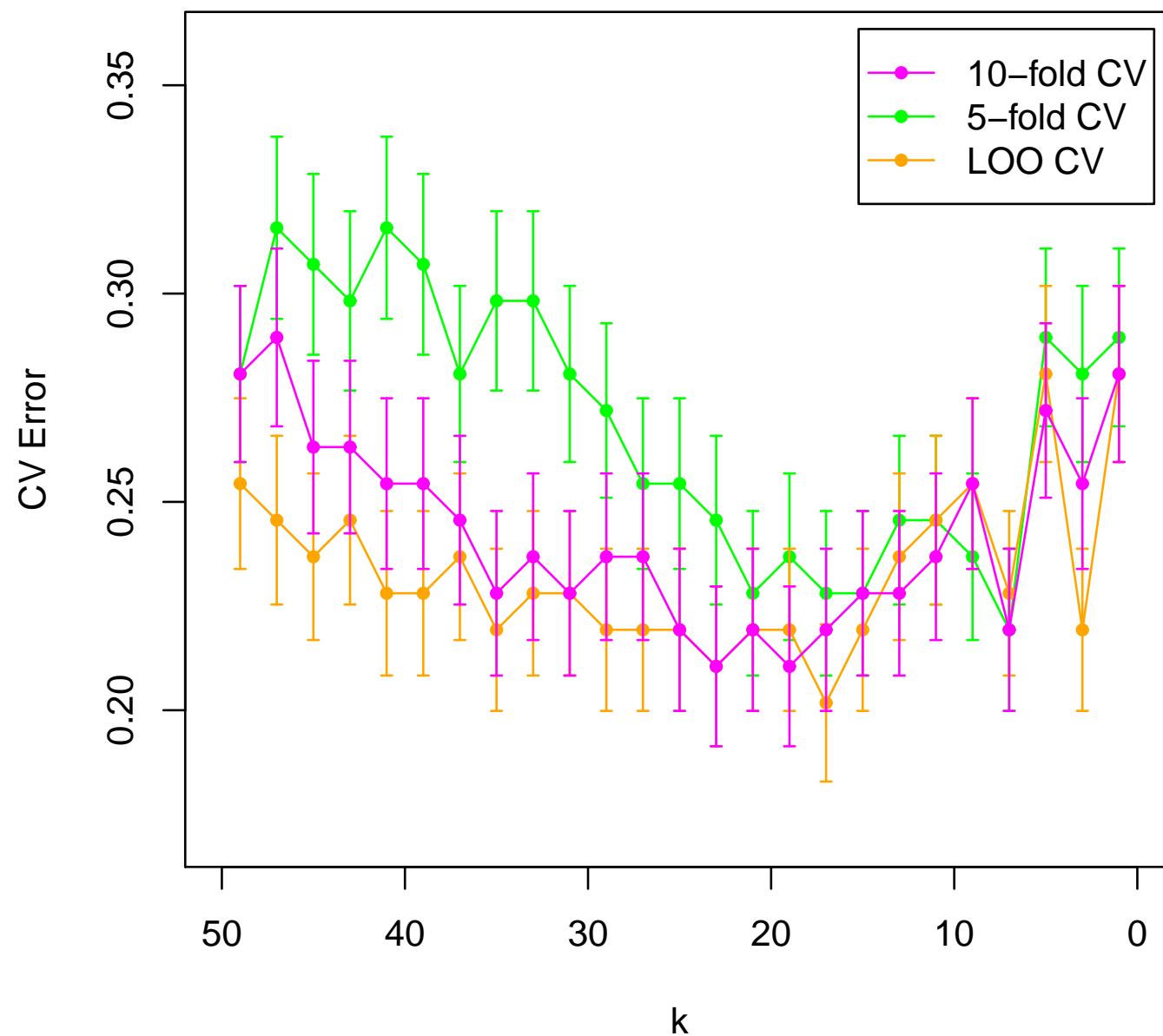
$M = N$  — метод перекрестного контроля с одним отделяемым элементом (*leave-one-out cross-validation*, LOO).

LOO — самый точный, но требует много времени.

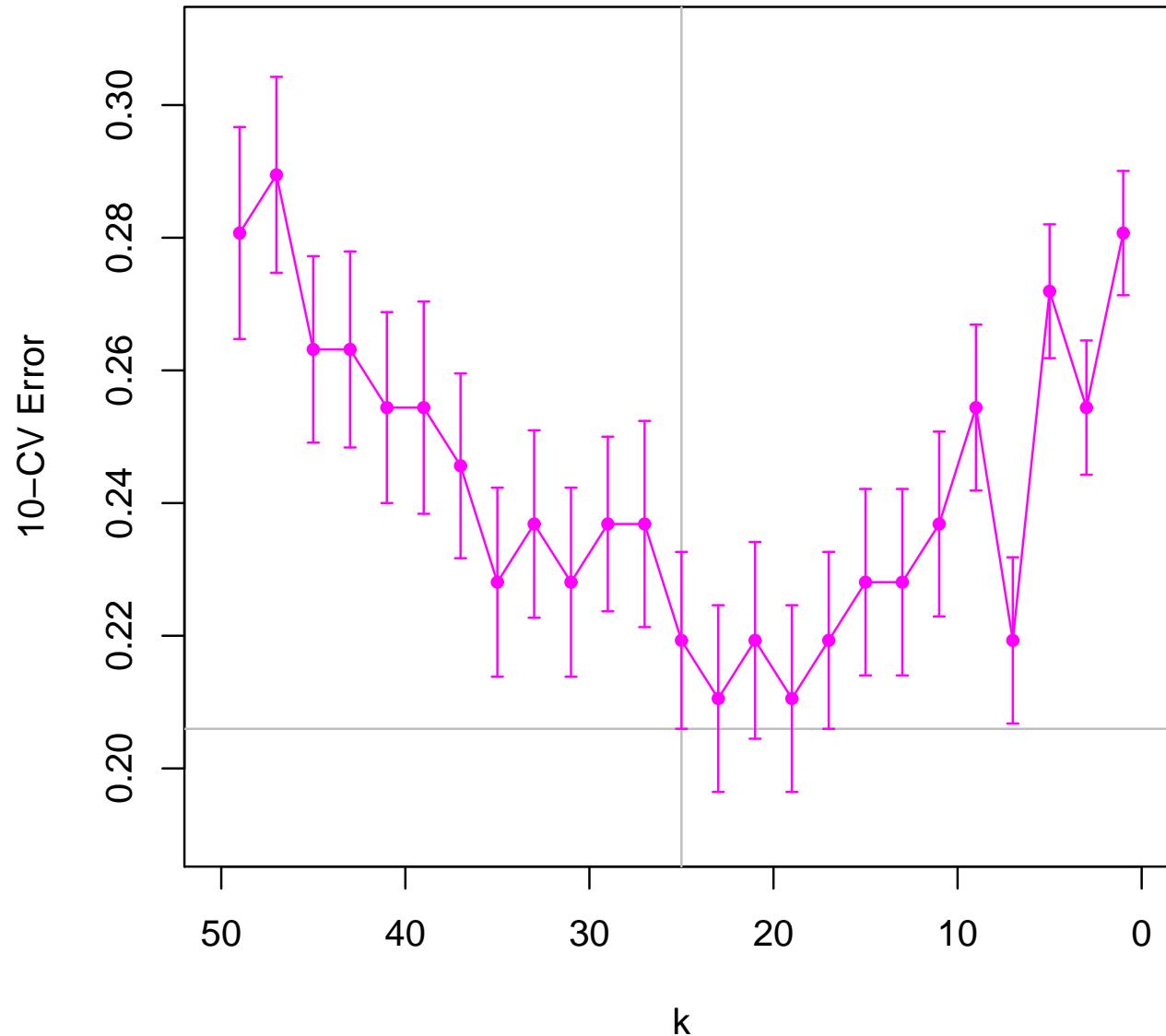
Перекрестный контроль для задачи классификации методом  $k$  ближайших соседей.



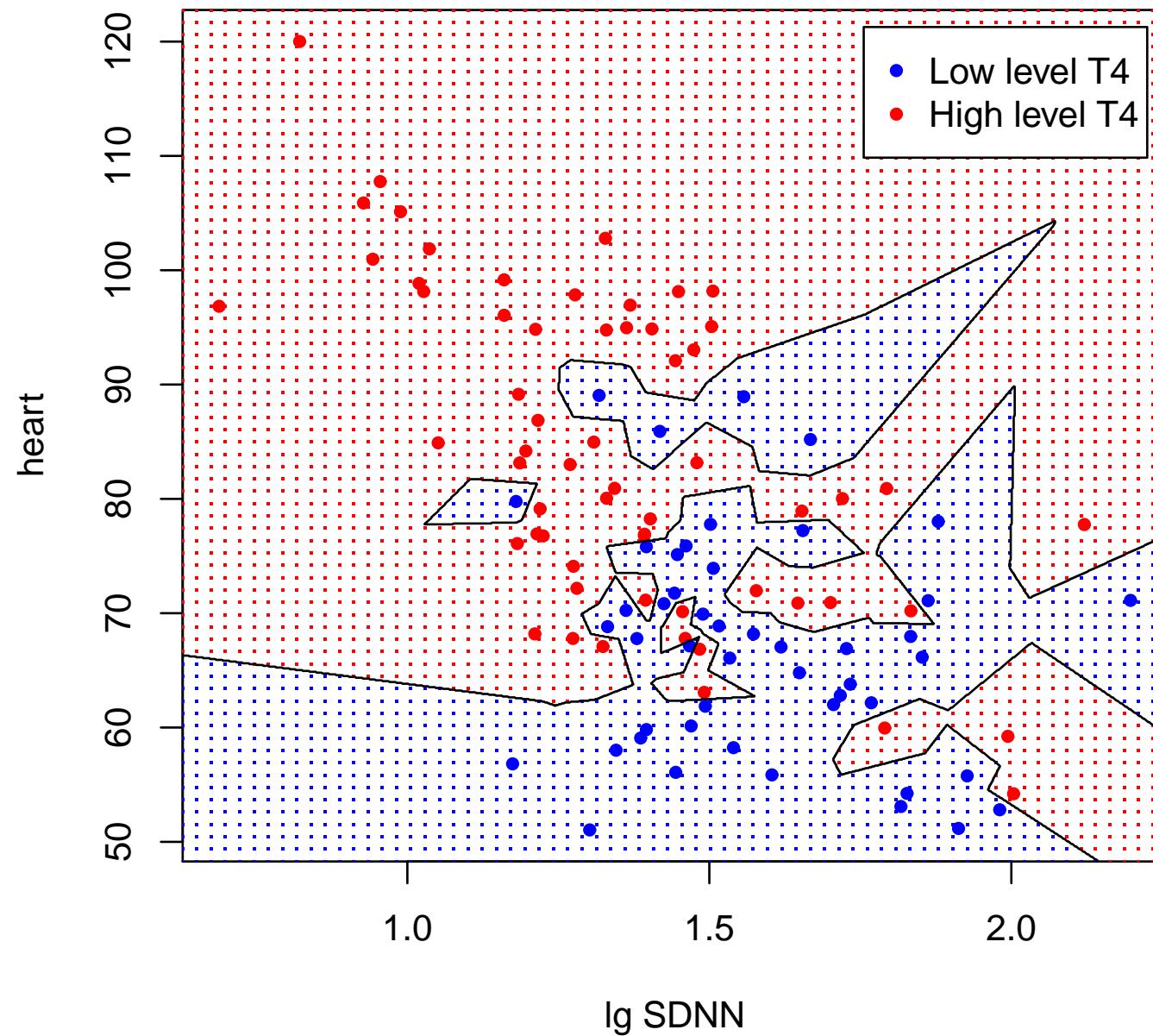
## Медицинская диагностика $0.5 \cdot se$



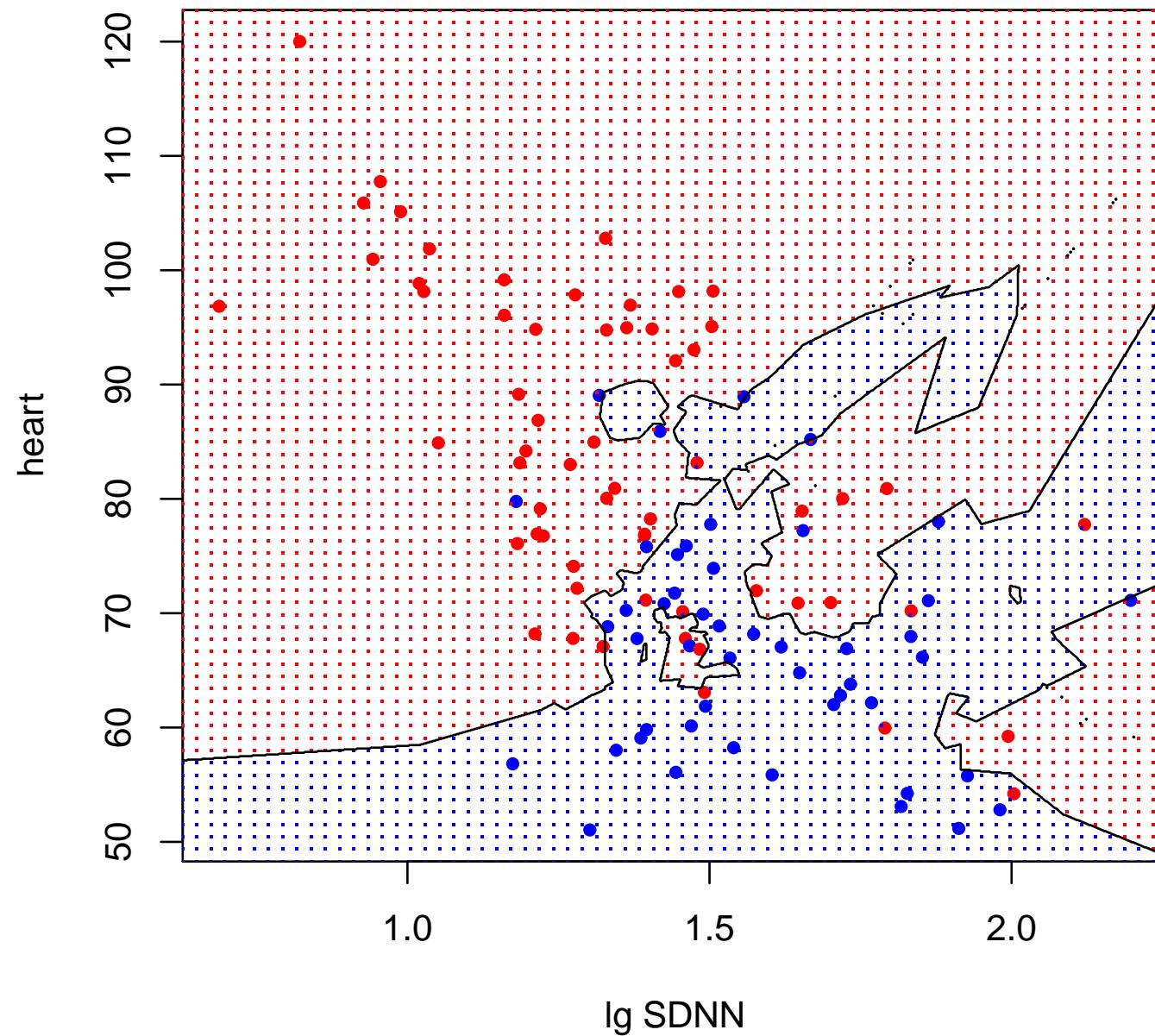
Правило одного se  $kk[i.\min] = 19$   $err[i.\min] = 0.2105263$   $se[i.\min] = 0.03835154$   
 $kk[i.\text{opt}] = 37$   $err[i.\text{opt}] = 0.245614$   $se[i.\text{opt}] = 0.04049339$



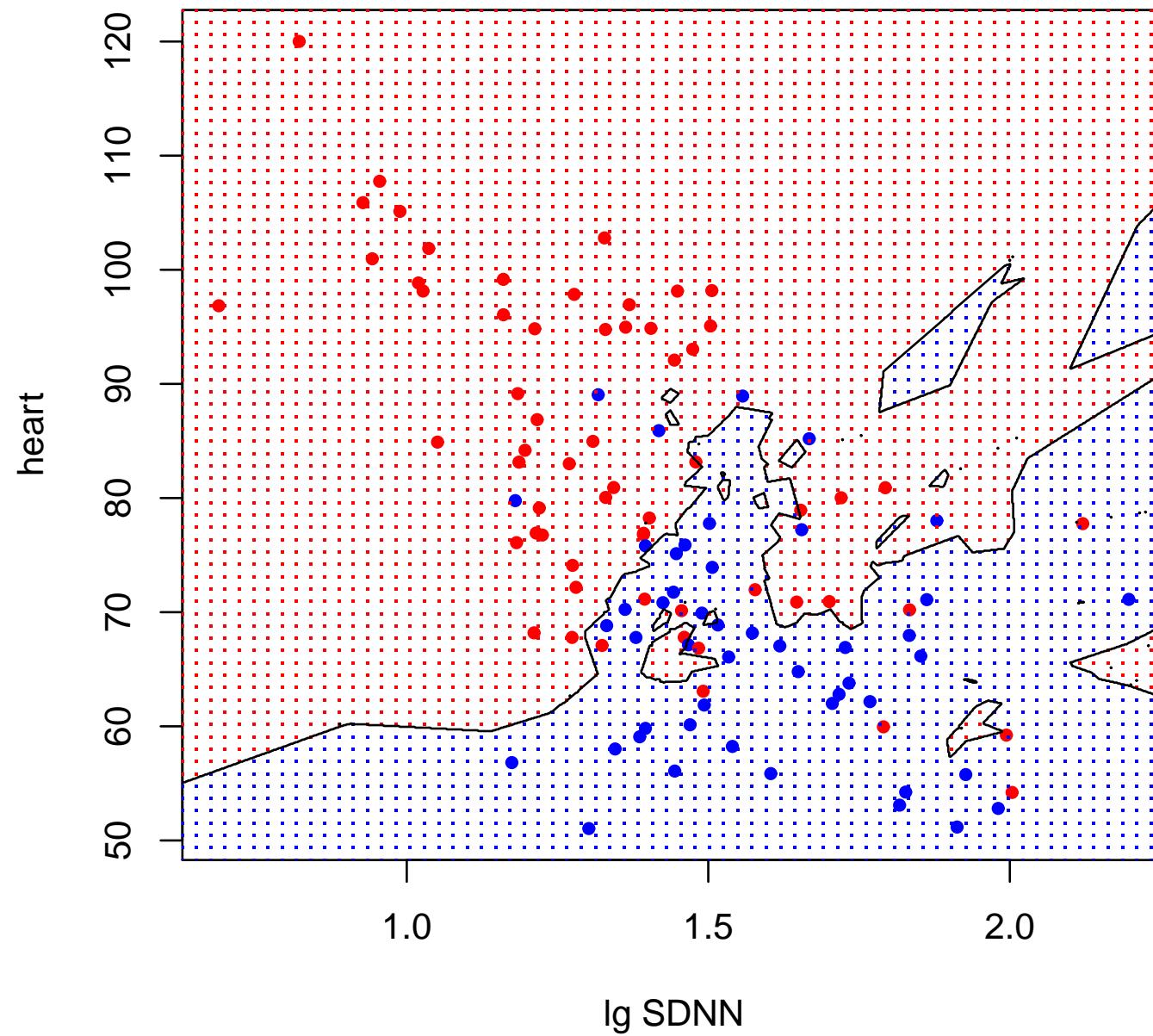
Задача медицинской диагностики. Метод 1 ближайшего соседа. 10-CV ошибка 0.30



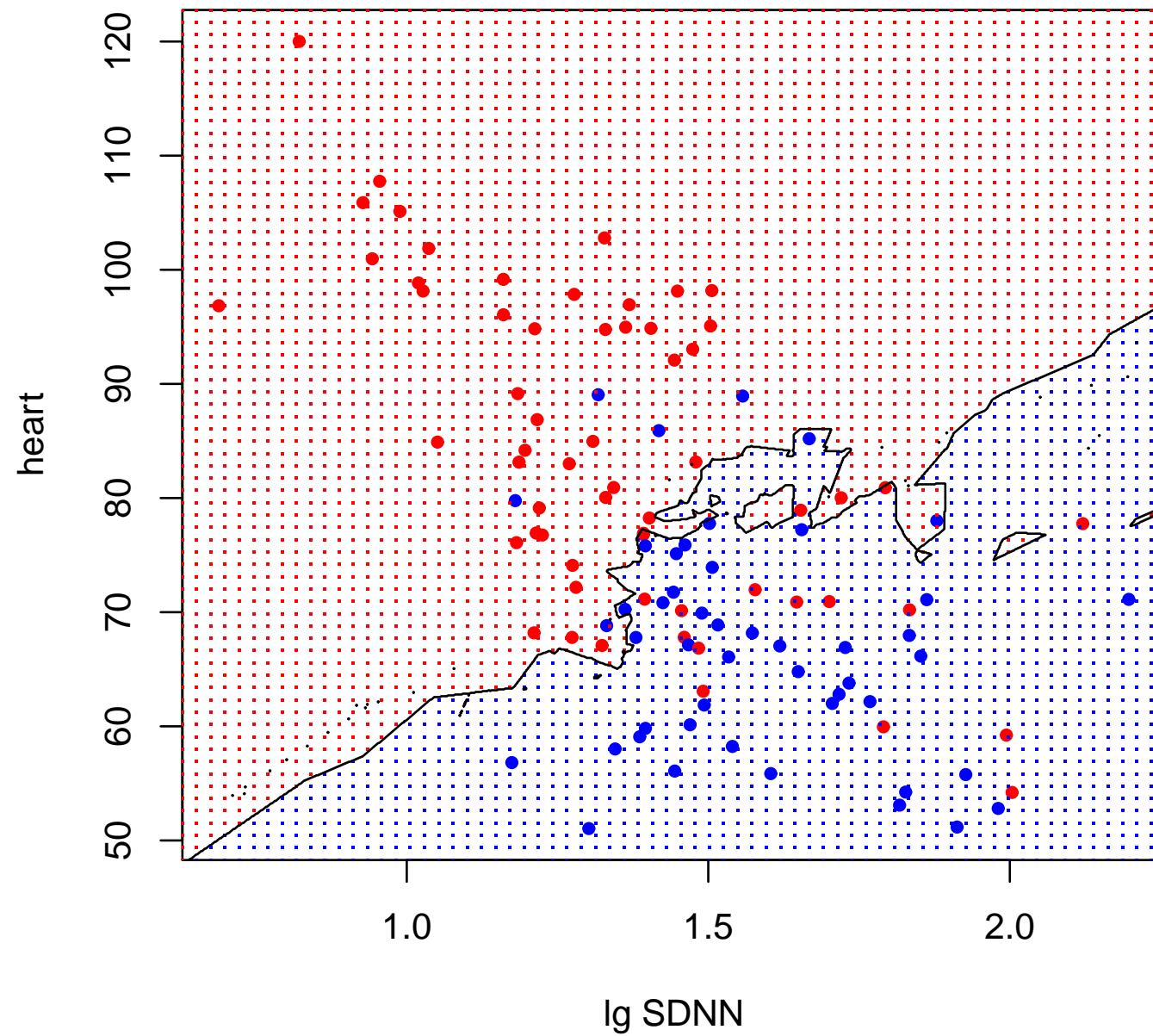
Метод 3 ближайших соседей 10-CV ошибка 0.26



Метод 5 ближайших соседей 10-CV ошибка 0.27



Метод 15 ближайших соседей 10-CV ошибка 0.25



## Какое значение $M$ использовать?

Метод скользящего контроля характеризует не конкретную решающую функцию  $f$ , а весь метод обучения.

При  $M = N$  (LOO) обучающие выборки похожи друг на друга, следовательно, решающие функции  $f_m$ , по-видимому, будут похожи на функцию  $f$ , построенную по всей выборке, поэтому LOO обычно дает лучшие оценки среднего риска.

Однако LOO требует много времени.

При небольших значениях  $M$  ( $M = 5, 10$ ) оценка скользящего контроля  $\widehat{R}^{\text{cv}}(f_i)$  может сильнее отличаться от среднего риска  $R(f)$  функции, построенной по всей обучающей выборке, так как

- 1) обучающие выборки сильнее отличаются друг от друга;
- 2) обучающие выборки могут оказаться слишком маленькими, чтобы построить хорошую решающую функцию.

## 2.5.2. Выбор модели

(model selection)

*Алгоритм обучения* по обучающей выборке  $D \in \mathcal{D}$  строит решающую функцию  $f \in \mathcal{F}$

Пусть  $\alpha$  — гиперпараметр (м.б. вектором) алгоритма (метода) обучения.

Для одной и той же обучающей выборки  $D$ , но разных  $\alpha$  будем получать разные решающие функции  $f(x, \alpha)$ .

Какую из них выбрать?

Хотелось бы, конечно, найти  $f^* = f(\cdot, \alpha^*)$ , где

$$\alpha^* = \operatorname{argmin}_{\alpha} R(f(\cdot, \alpha))$$

Но приходится довольствоваться  $f_o = f(\cdot, \alpha_o)$ , где

$$\alpha_o = \operatorname{argmin}_{\alpha} \widehat{R}^{\text{test}}(f(\cdot, \alpha)) \quad \text{или} \quad \alpha_o = \operatorname{argmin}_{\alpha} \widehat{R}^{\text{cv}}(f(\cdot, \alpha)).$$

Теперь тестовая выборка стала обучающей! Поэтому, как правило,

$$\widehat{R}^{\text{test}}(f(\cdot, \alpha_o)) < R(f_o)$$

Если данных достаточно, то их делят

- на обучающую (train) выборку,
- на проверочную (validation) выборку,
- на тестовую (test) выборку.

Train	Validation	Test
-------	------------	------

Обучающая выборка используется для построения моделей  $f(\cdot, \alpha) \in \mathcal{F}$  для разных  $\alpha$ .

Проверочная — для оценки оценки среднего риска каждой из построенной модели и выбора наилучшей модели в  $\mathcal{F}$ .

Тестовая — для оценки ошибки предсказания выбранной модели.

Для выбора наилучшей модели можно использовать перекрестную проверку.

Validation	Train	Train	Train	Train	Test
------------	-------	-------	-------	-------	------

Для выбора наилучшей модели можно использовать перекрестную проверку.

Train	Validation	Train	Train	Train	Test
-------	------------	-------	-------	-------	------

Для выбора наилучшей модели можно использовать перекрестную проверку.

Train	Train	Validation	Train	Train	Test
-------	-------	------------	-------	-------	------

Для выбора наилучшей модели можно использовать перекрестную проверку.

Train	Train	Train	Validation	Train	Test
-------	-------	-------	------------	-------	------

Для выбора наилучшей модели можно использовать перекрестную проверку.

Train	Train	Train	Train	Validation	Test
-------	-------	-------	-------	------------	------

Для выбора наилучшей модели можно использовать перекрестную проверку.

Cross-validation

Test

Для каждого  $\alpha$  запускаем процедуру обучения — перекрестного контроля.

Выбираем  $f_o = f(\cdot, \alpha_o)$ , где

$$\alpha_o = \operatorname{argmin}_{\alpha} \hat{R}^{\text{cv}}(f(\cdot, \alpha)).$$

Оцениваем риск на тестовой выборке.

Для построения окончательной модели используем найденный гиперпараметр  $\alpha_o$  и *всю* выборку

Train for final model

## 2.6. Метод главных компонент

Метод главных компонент (Principal Component Analysis, PCA) — один из основных, «классических», методов понижения размерности.

*Sylvester J.J.* On the reduction of a bilinear quantic of the  $n$ th order to the form of a sum of  $n$  products by a double orthogonal substitution. Messenger of Mathematics, 19, 42–46 (1889)

*Pearson C.* On lines and planes of closest fit to systems of points in space. Phil. Mag., Series B., 2 (11), 559–572 (1901)

Джеймс Джозеф Сильвестр (1814–1897) — английский математик. Несколько лет работал в США. Основал «American Journal of Mathematics».

Карл Пирсон (1857–1936) — английский математик, статистик и биолог. Основатель математической статистики.

$$x^{(1)}, x^{(2)}, \dots, x^{(N)} \in \mathbf{R}^d.$$

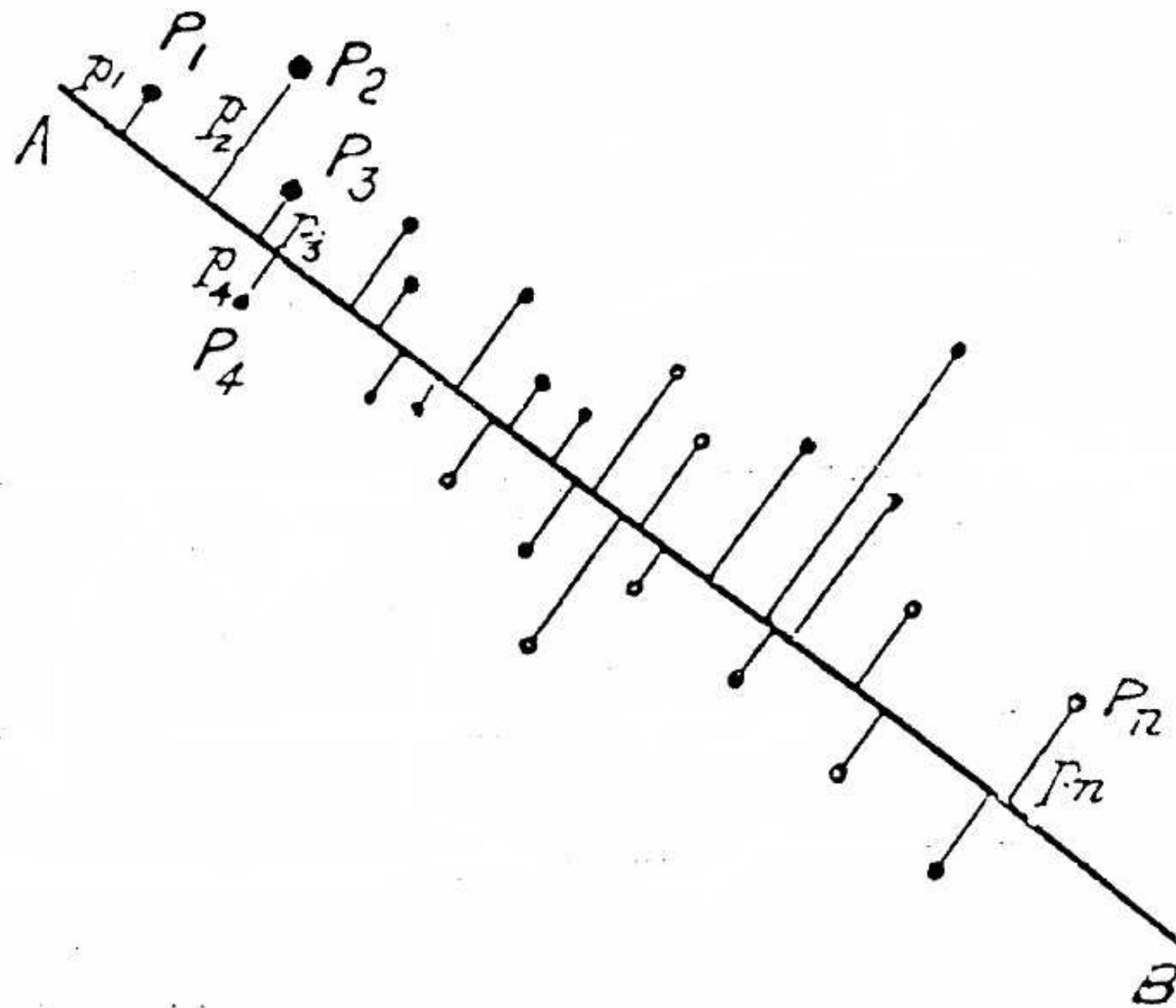
Рассмотрим следующие задачи:

- 1) Аппроксимировать данные линейными многообразиями меньшей размерности: найти линейное многообразие заданной размерности  $k < d$ , сумма квадратов расстояний до которого минимальна.
- 2) Найти подпространство заданной размерности, в ортогональной проекции на которое разброс (рассеивание) (выборочная дисперсия для  $k = 1$ ) максимальен.
- 3) Найти подпространство заданной размерности, в ортогональной проекции на которое среднеквадратичное расстояние между каждой парой точек максимально.

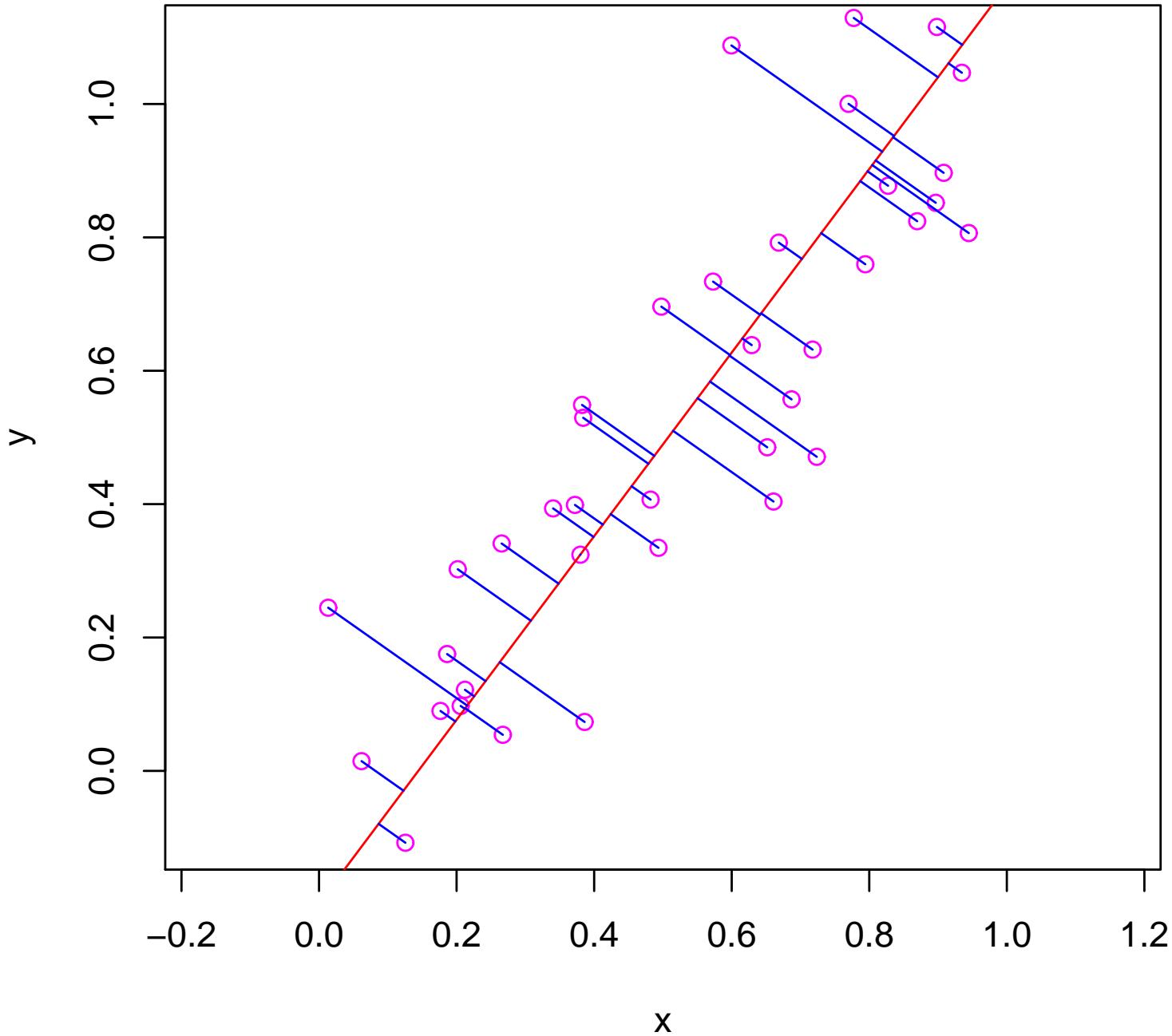
Все эти задачи имеют одно и то же решение — PCA.

Многочисленные *обобщения* (в том числе нелинейные): хорошо приспособленный базис (Ю. И. Неймарк), метод главных кривых и многообразий, поиск наилучшей проекции (Projection Pursuit), самоорганизующиеся карты Кохонена и т. д.

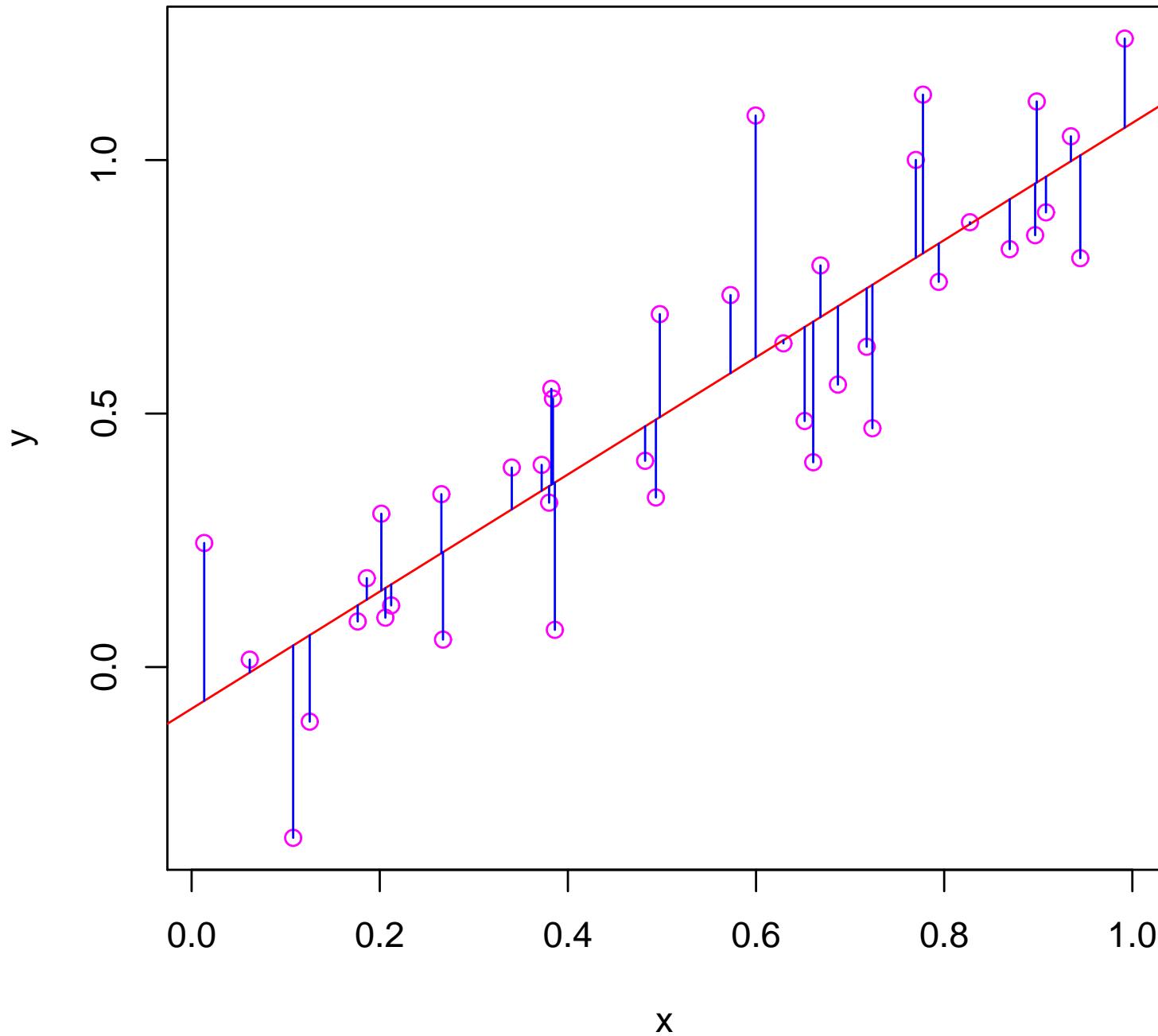
Рисунок из оригинальной статьи Пирсона



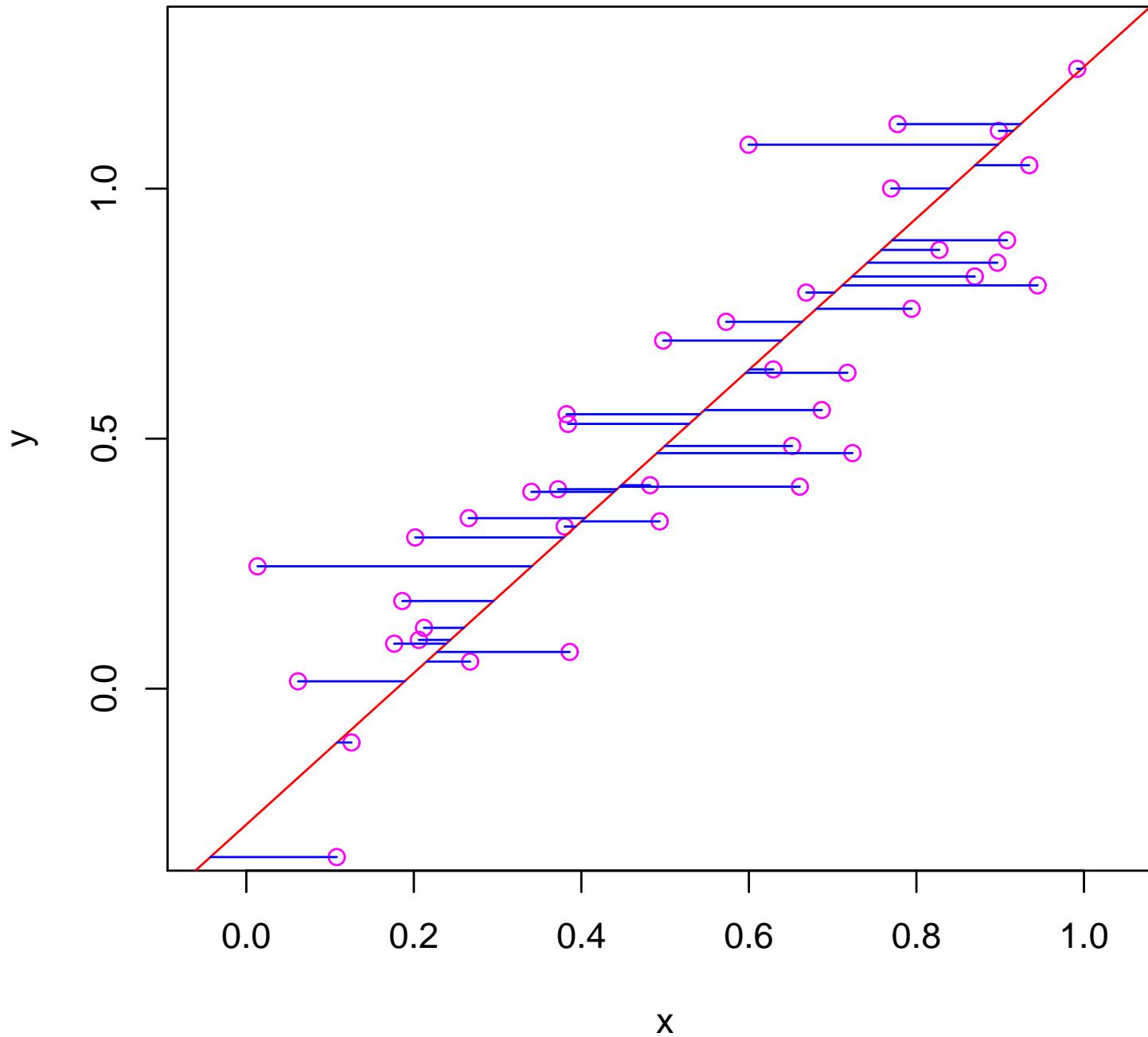
PCA и метод наименьших квадратов — это разные вещи!



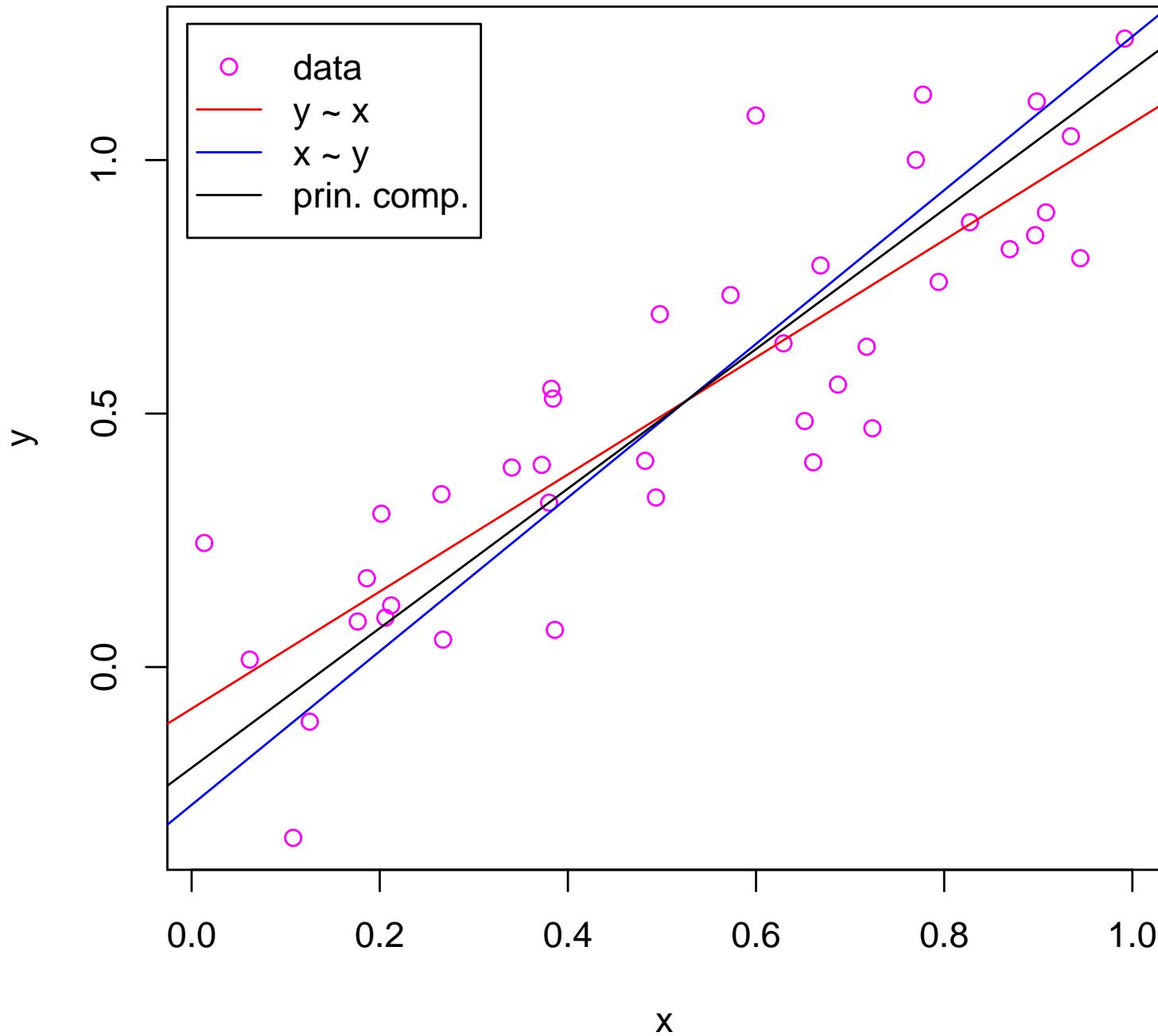
PCA и метод наименьших квадратов — это разные вещи!



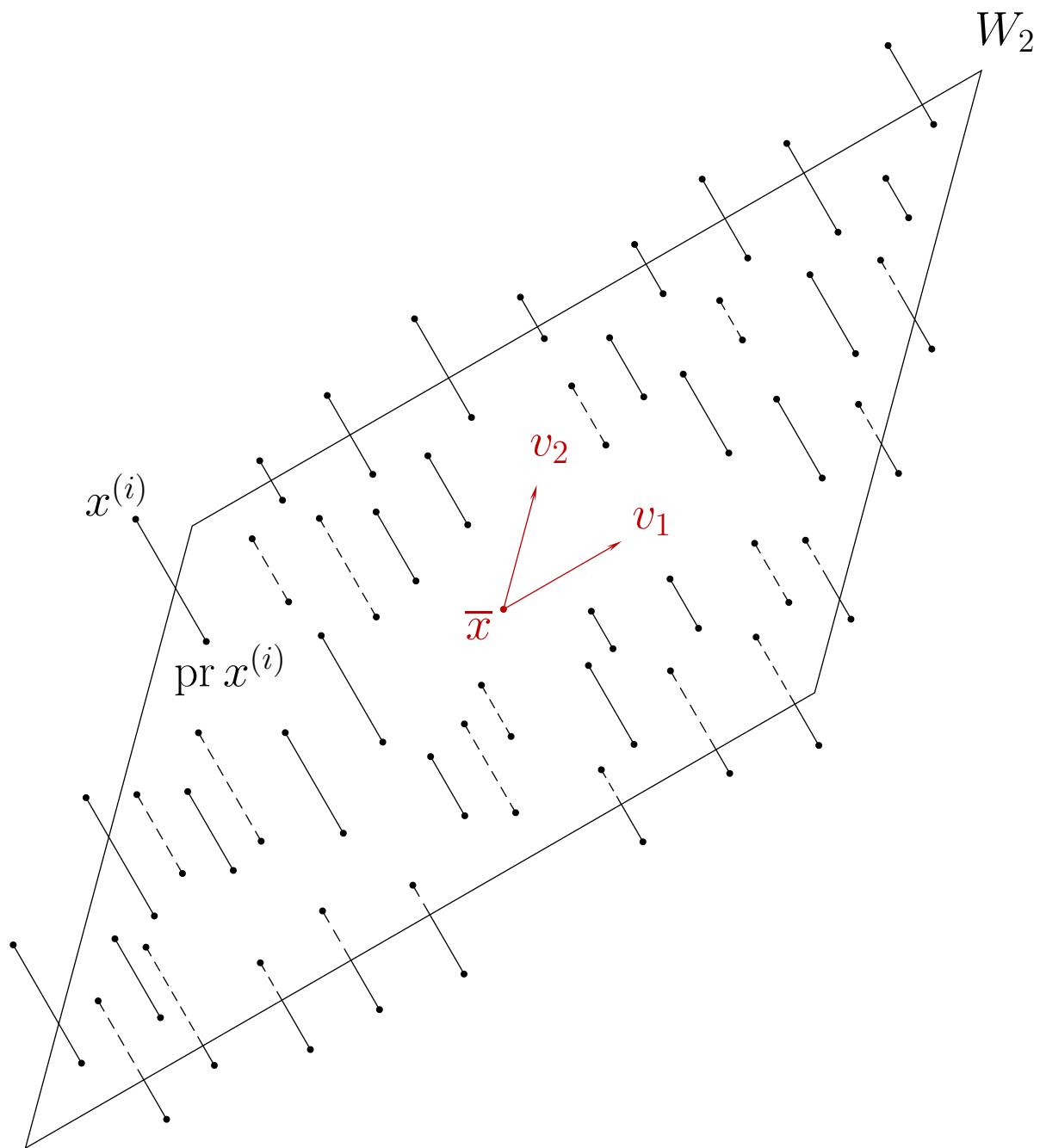
PCA и метод наименьших квадратов — это разные вещи!



PCA и метод наименьших квадратов — это разные вещи!







**1-я задача.** Ищем  $k$ -мерное линейное многообразие

$$W_k = v_0 + L(v_1, \dots, v_k), \quad v_j \in \mathbf{R}^d, \quad \|v_j\| = 1, \quad v_j \perp v_{j'} \quad (j \neq j')$$

такое, что

$$\sum_{i=1}^N \text{dist}^2(x^{(i)}, W_k) = \sum_{i=1}^N \|\text{ort}_{W_k} x^{(i)}\|^2 = \sum_{i=1}^N \left\| x^{(i)} - v_0 - \sum_{j=1}^k \langle x^{(i)} - v_0, v_j \rangle v_j \right\|^2 \rightarrow \min$$

Оказывается, что  $W_0 \subset W_1 \subset \dots \subset W_k$ ,  $v_0 = \bar{x} = \frac{1}{N} \sum_{i=1}^N x^{(i)}$

и векторы  $v_j$  (*векторы главных компонент*) можно находить итерационно!

**Упражнение 2.8** Доказать, что можно положить  $v_0 = \bar{x}$ . Описать множество всех возможных  $v_0$ , доставляющих минимум сумме квадратов расстояний до искомого многообразия.

Таким образом, в этой задаче можно считать, что данные центрированы:

$$x^{(i)} \leftarrow x^{(i)} - \bar{x} \quad (i = 1, 2, \dots, N).$$

**2-я задача** (максимизировать разброс в ортогональной проекции).

Пусть данные уже центрированы:  $x^{(i)} \leftarrow x^{(i)} - \bar{x}$  ( $i = 1, 2, \dots, N$ )

(очевидно, что дисперсия в ортогональной проекции от этого не изменится)

Ищем подпространство

$$W_k = L(v_1, v_2, \dots, v_k), \quad v_j \in \mathbf{R}^d, \quad \|v_j\| = 1, \quad v_j \perp v_{j'} \quad (j \neq j'),$$

такое, что

$$\frac{1}{N} \sum_{i=1}^N \left\| \text{pr}_{W_k} x^{(i)} \right\|^2 = \frac{1}{N} \sum_{i=1}^N \left\| \sum_{j=1}^k \langle x^{(i)}, v_j \rangle v_j \right\|^2 = \sum_{i=1}^N \sum_{j=1}^k \langle x^{(i)}, v_j \rangle^2 \rightarrow \max$$

Эта задача эквивалентна предыдущей, так как (по теореме Пифагора)

$$\| \text{pr}_{W_k} x^{(i)} \|^2 = \| x^{(i)} \|^2 - \| \text{ort}_{W_k} x^{(i)} \|^2 \Leftrightarrow \left\| \sum_{j=1}^k \langle x^{(i)}, v_j \rangle v_j \right\|^2 = \| x^{(i)} \|^2 - \left\| x^{(i)} - \langle x^{(i)}, v_j \rangle v_j \right\|^2$$

и первое слагаемое не зависит от  $v_j$ .

**3-я задача** (максимизировать в ортогональной проекции среднеквадратичное расстояние между каждой парой точек).

Пусть данные уже центрированы:  $x^{(i)} \leftarrow x^{(i)} - \bar{x}$  ( $i = 1, 2, \dots, N$ ).

Ищем подпространство

$$W_k = L(v_1, v_2, \dots, v_k), \quad v_j \in \mathbf{R}^d, \quad \|v_j\| = 1, \quad v_j \perp v_{j'} \quad (j \neq j'),$$

такое, что

$$\frac{1}{N(N-1)} \sum_{i \neq i'} \left\| \text{pr}_{W_j} x^{(i)} - \text{pr}_{W_j} x^{(i')} \right\|^2 \rightarrow \max$$

Эта задача эквивалентна 2-й задаче (а, следовательно, и 1-й), так как для любых векторов  $x^{(1)}, x^{(2)}, \dots, x^{(N)}$  в  $\mathbf{R}^d$

$$\frac{1}{N(N-1)} \sum_{i \neq i'} \left\| x^{(i)} - x^{(i')} \right\|^2 = \frac{2N^2}{N(N-1)} \left( \frac{1}{N} \sum_{i=1}^N \|x^{(i)}\|^2 - \left\| \frac{1}{N} \sum_{i=1}^N x^{(i)} \right\|^2 \right)$$

(справа в больших скобках стоит выборочная дисперсия).

$k = 1$ :

$$\sum_{i=1}^N \left\langle x^{(i)}, v_1 \right\rangle^2 = \|\mathbf{X}v_1\|^2 = v_1^\top \mathbf{X}^\top \mathbf{X} v_1 \rightarrow \max_{\|v_1\|=1}$$

Функция Лагранжа:

$$\mathcal{L} = v_1^\top \mathbf{X}^\top \mathbf{X} v_1 - \lambda_1 (\|v_1\|^2 - 1)$$

$$\frac{\partial \mathcal{L}}{\partial v_1} = 2\mathbf{X}^\top \mathbf{X} v_1 - 2\lambda_1 v_1 = 0 \quad \Leftrightarrow \quad \mathbf{X}^\top \mathbf{X} v_1 = \sigma_1 v_1,$$

т. е.  $v_1$  — собственный вектор матрицы  $\mathbf{X}^\top \mathbf{X}$ , относящийся к максимальному с. ч.  $\lambda_1$ .

$k = 2$ :

$$\sum_{i=1}^N \left\langle x^{(i)}, v_2 \right\rangle^2 = \|\mathbf{X}v_2\|^2 = v_2^\top \mathbf{X}^\top \mathbf{X} v_2 \rightarrow \max_{\substack{\|v_2\|=1 \\ v_1 \perp v_2}}$$

Функция Лагранжа:

$$\mathcal{L} = v_2^\top \mathbf{X}^\top \mathbf{X} v_2 - \lambda_2 (\|v_2\|^2 - 1) - \mu_{12} v_1^\top v_2 \quad \Rightarrow \quad \frac{\partial \mathcal{L}}{\partial v_2} = 2\mathbf{X}^\top \mathbf{X} v_2 - 2\sigma_2 v_2 - \mu_{12} v_1 = 0,$$

умножая на  $v_1^\top$  слева:

$$2v_1^\top \mathbf{X}^\top \mathbf{X} v_2 - 2\sigma_2 v_1^\top v_2 - \mu_{12} v_1^\top v_1 = 0 \quad \Rightarrow \quad 2\sigma_1 v_1^\top v_2 - 2\sigma_2 v_1^\top v_2 - \mu_{12} v_1^\top v_1 = 0 \quad \Rightarrow \quad \mu_{12} = 0,$$

откуда  $\mathbf{X}^\top \mathbf{X} v_2 = \sigma_2 v_2$  т. е.  $v_2$  — с. в. матрицы  $\mathbf{X}^\top \mathbf{X}$  для следующего с. ч.  $\sigma_2$ .

$$v_1 = \underset{\|v\|=1}{\operatorname{argmax}} \|\mathbf{X}v\|^2 \quad (1) \quad v_2 = \underset{\substack{\|v\|=1 \\ v \perp v_1}}{\operatorname{argmax}} \|\mathbf{X}v\|^2 \quad (2) \quad \dots \quad v_k = \underset{\substack{\|v\|=1 \\ v \perp v_1, \dots, v \perp v_{k-1}}}{\operatorname{argmax}} \|\mathbf{X}v\|^2 \quad (k)$$

Докажем, что жадный алгоритм работает, т. е.

$$\|\mathbf{X}v_1\|^2 + \dots + \|\mathbf{X}v_k\|^2 = \max_{\substack{\|w_i\|=1 \\ w_i \perp w_j}} (\|\mathbf{X}w_1\|^2 + \dots + \|\mathbf{X}w_k\|^2)$$

$k = 1$ : из (1)

$k = 2$ :

Пусть

$$w_2 \perp v_1, \quad W_2 = L(w_1, w_2) \quad (*)$$

$$\begin{aligned} \|\mathbf{X}v_1\|^2 &\geq \|\mathbf{X}w_1\|^2 \text{ из (1)} \\ \|\mathbf{X}v_2\|^2 &\geq \|\mathbf{X}w_2\|^2 \text{ из (2) и (*)} \end{aligned} \Rightarrow \|\mathbf{X}v_1\|^2 + \|\mathbf{X}v_2\|^2 \geq \|\mathbf{X}w_1\|^2 + \|\mathbf{X}w_2\|^2$$

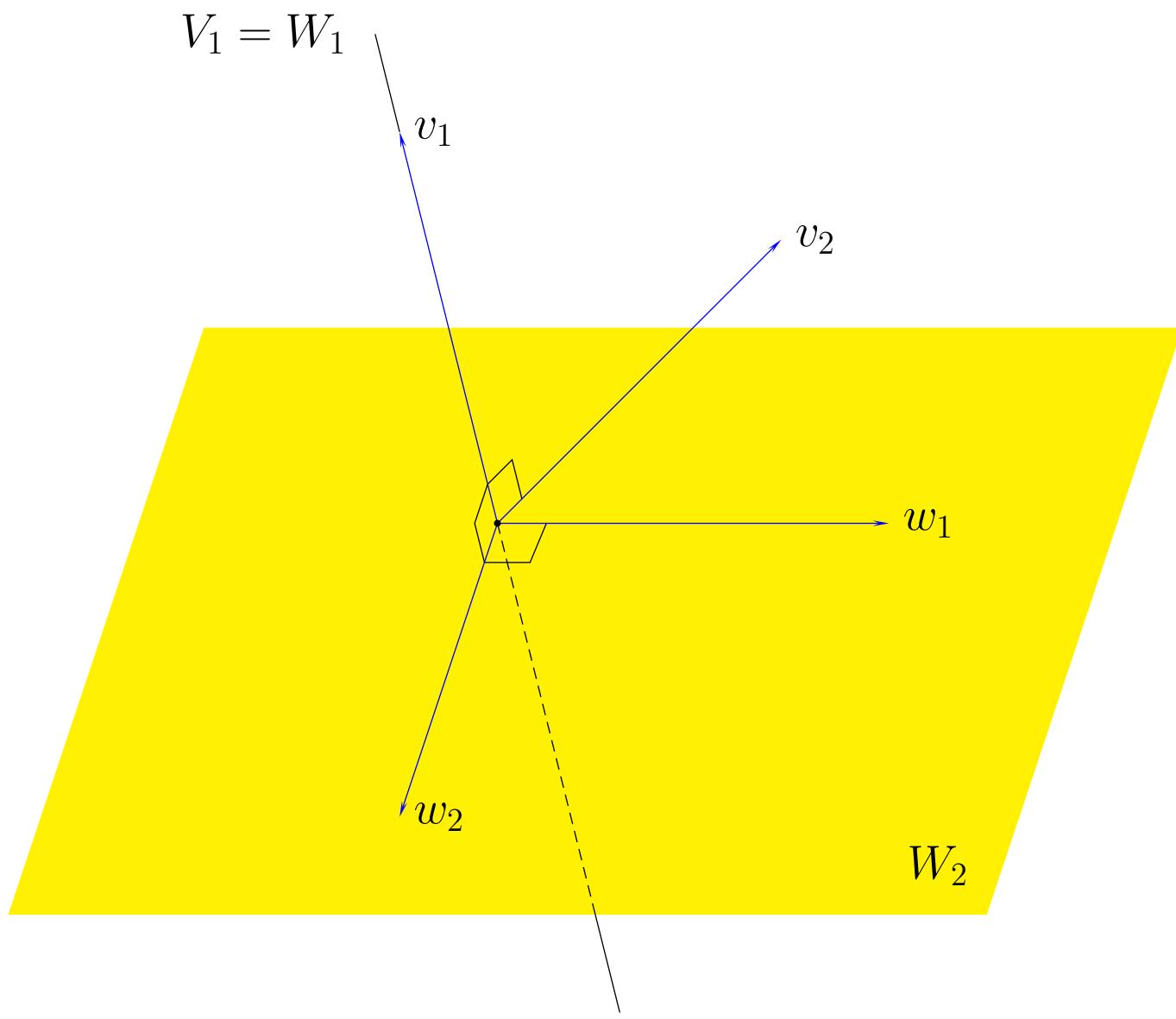
$k - 1 \rightarrow k$ :

Пусть

$$w_k \perp v_1, v_2, \dots, v_{k-1}, \quad W_k = L(w_1, \dots, w_k) \quad (**) \Rightarrow$$

$$\begin{aligned} \|\mathbf{X}v_1\|^2 + \dots + \|\mathbf{X}v_{k-1}\|^2 &\geq \|\mathbf{X}w_1\|^2 + \dots + \|\mathbf{X}w_{k-1}\|^2 \text{ по инд. предположению} \\ \|\mathbf{X}v_k\|^2 &\geq \|\mathbf{X}w_k\|^2 \text{ из (k) и (**)} \end{aligned}$$

$$\|\mathbf{X}v_1\|^2 + \dots + \|\mathbf{X}v_k\|^2 \geq \|\mathbf{X}w_1\|^2 + \dots + \|\mathbf{X}w_k\|^2$$



**4-я задача.** Пусть  $X$  —  $d$ -мерная случайная величина.

Требуется найти такой ортонормированный базис  $v_1, v_2, \dots, v_d$ , в котором ковариация между различными координатами равна нулю.

После преобразования к этому базису

$$\text{Cov}(X_j, X_{j'}) = 0 \quad (j \neq j'),$$

где  $\text{Cov}(X_j, X_{j'}) = \mathbb{E}((X_j - \mathbb{E} X_j)(X_{j'} - \mathbb{E} X_{j'}))$  — коэффициент ковариации.

Т. е. матрица  $\text{Cov } X$  должна стать диагональной.

Искомый базис — это базис из собственных векторов матрицы  $\text{Cov } X$ .

Первые три задачи являются аппроксимацией к 4-й задаче.

Дисперсия по направлению  $w$ ,  $\|w\| = 1$ , равна  $w^\top \text{Cov } X w$

Ее максимизация тоже приводит к той же задаче.

Ковариационная матрица ( $X$  — столбец):

$$\text{Cov } X = \mathbb{E} \left( (X - \mathbb{E} X) \cdot (X - \mathbb{E} X)^\top \right).$$

Эмпирическая (или выборочная) ковариационная матрица ( $x^{(i)}$  — столбец):

$$\widehat{\text{Cov}} X = \frac{1}{N-1} \sum_{i=1}^N (x^{(i)} - \bar{x})(x^{(i)} - \bar{x})^\top.$$

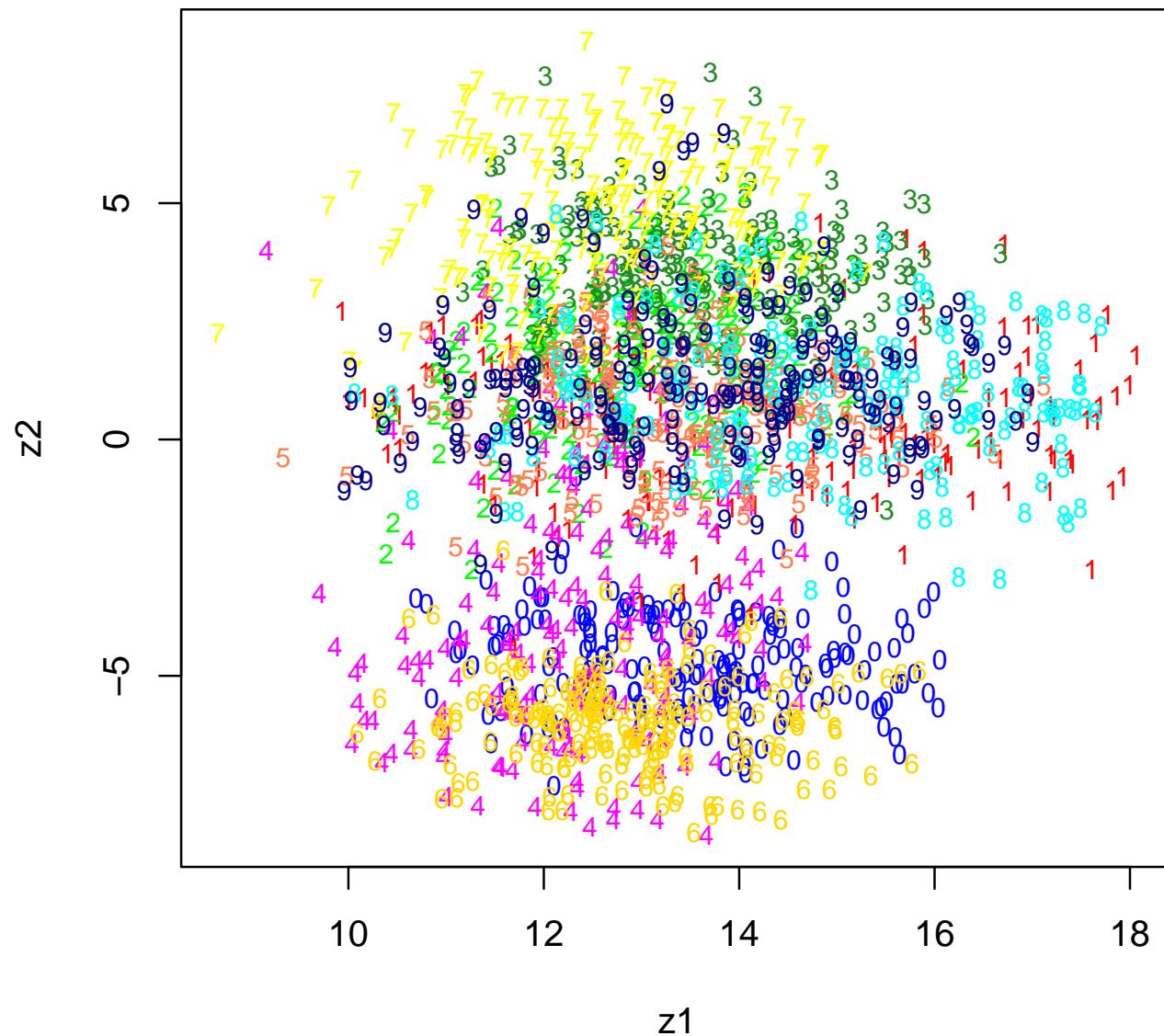
Если данные центрированы, то

$$\widehat{\text{Cov}} X = \frac{1}{N-1} \mathbf{X}^\top \mathbf{X}.$$

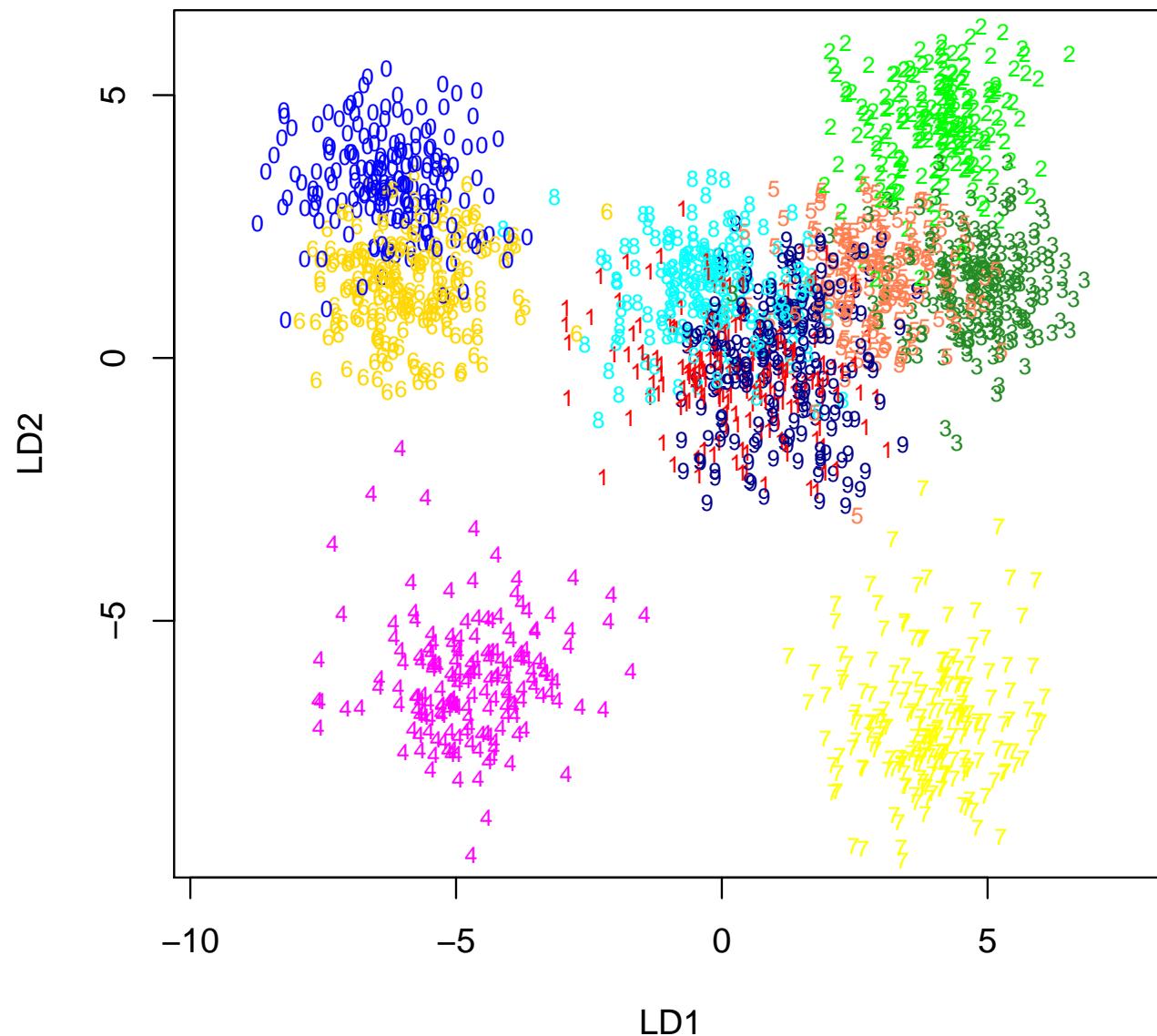
Векторы главных компонент  $v_1, v_2, \dots, v_d$  — это ортонормированный набор собственных векторов матрицы  $\widehat{\text{Cov}} X$ , расположенных в порядке убывания собственных значений

$$\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_d \geq 0.$$

Рукописные цифры.  $d = 1024$ ,  $k = 2$



## Рукописные цифры. LDA



Phoneme — распознавание голоса.

(Andreas Buja, Werner Stuetzle and Martin Maechler)

4509 log-периодограмм длины  $d = 256$  (все признаки количественные).

[a:] 695

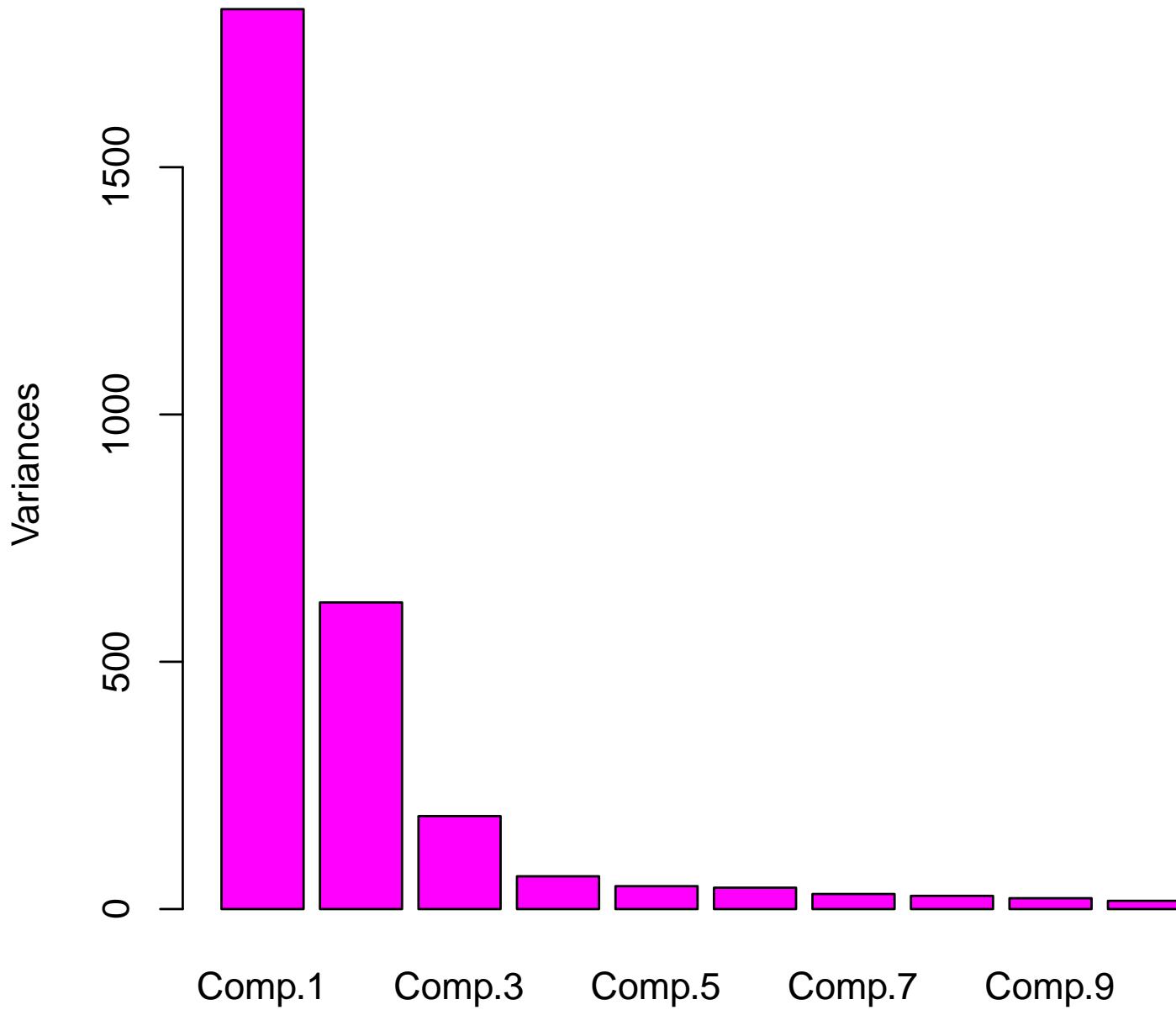
[ɔ] 1022

[i:] 1163

[d] 757

[ʃ] 873

# **phoneme.pca**



## Какое количество главных компонент выбрать?

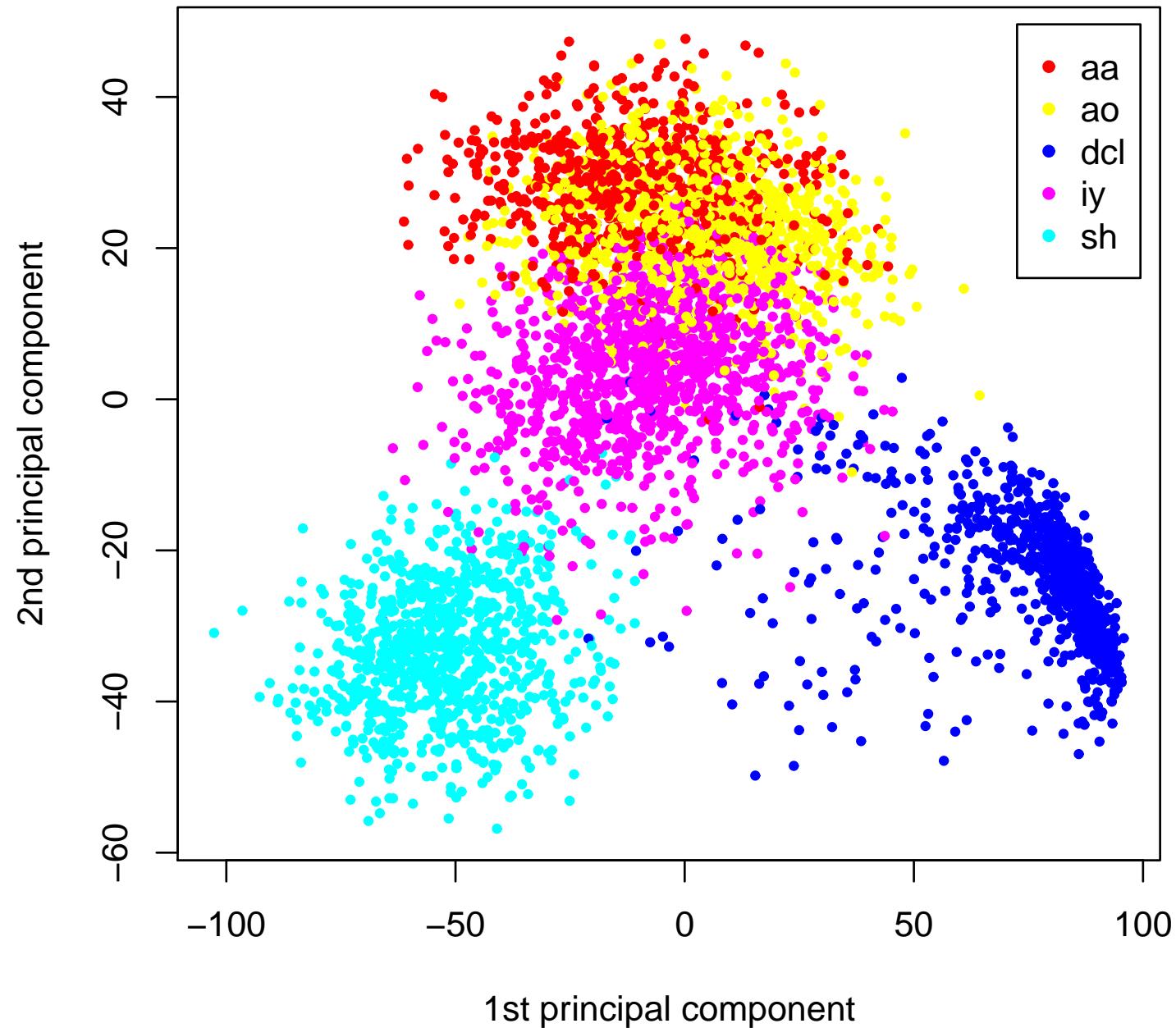
Существует много (эвристических) подходов.

Например:

Выбираем нужное количество  $s$  главных компонент так, чтобы объясненная дисперсия не меньше некоторого заданного уровня  $\alpha$  (например,  $\alpha = 0.95$ ):

$$\frac{\sigma_1^2 + \sigma_2^2 + \cdots + \sigma_s^2}{\sigma_1^2 + \sigma_2^2 + \cdots + \sigma_s^2 + \cdots + \sigma_d^2} \geq \alpha$$

Заметим, что  $\sigma_1^2 + \sigma_2^2 + \cdots + \sigma_s^2 + \cdots + \sigma_d^2 = \text{tr} \widehat{\text{Cov}} X$



Все данные были случайно поделены на обучающую (4/5) и тестовую (1/5) выборки  
Тестовая ошибка:

<i>Метод</i>	<i>Все признаки</i>	<i>15 главных компонент</i>
SVM	0.076	0.074
Random Forest	0.081	0.083
AdaBoost	0.083	0.088
Bagging	0.099	0.124

## 2.6.1. Eigen Faces

Sirovich, Kirby (1987) — для задачи реконструкции лица

Matthew Turk and Alex Pentland (1991) — для задачи распознавания

Пример.

Ariel Sharon	77
Colin Powell	236
Donald Rumsfeld	121
George W. Bush	530
Gerhard Schroeder	109
Hugo Chavez	71
Tony Blair	144

$$N = 1288, h = 50, w = 37, d = 50 \times 37 = 1850$$

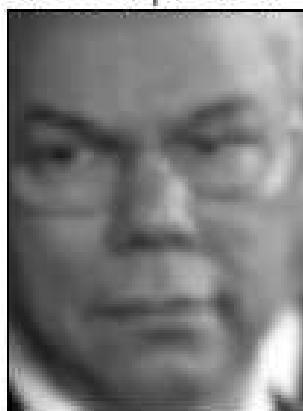
При использовании  $s = 100$  главных компонент + машина опорных векторов с ядром RBF получаем ошибку 14%.

При  $s > 300$  ошибка только возрастает.

Hugo Chavez



Powell | Powell



Rumsfeld | Rumsfeld



Bush | Bush



Bush | Bush



Bush | Bush



Rumsfeld | Rumsfeld



Sharon | Rumsfeld



Chavez | Chavez



Rumsfeld | Rumsfeld



Bush | Rumsfeld



Bush | Bush



Powell | Bush

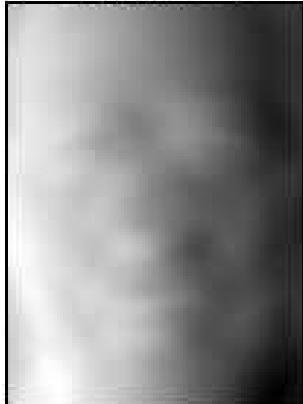




eigenface 0



eigenface 1



eigenface 2



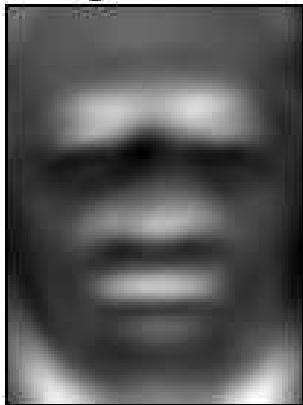
eigenface 3



eigenface 4



eigenface 5



eigenface 6



eigenface 7



eigenface 8



eigenface 9



eigenface 10



eigenface 11



## 2.7. Сингулярное разложение

- это самый «правильный» способ вычислить главные компоненты (нужно еще для многих других вещей)

*Сингулярное разложение* матрицы  $\mathbf{X}$  — это

$$\underbrace{\mathbf{X}}_{N \times d} = \underbrace{\mathbf{U}}_{N \times N} \cdot \underbrace{\Sigma}_{N \times d} \cdot \underbrace{\mathbf{V}^T}_{d \times d}$$

Матрицы  $\mathbf{U}$  и  $\mathbf{V}$  — ортогональные ( $\mathbf{U}^{-1} = \mathbf{U}^T$ ,  $\mathbf{V}^{-1} = \mathbf{V}^T$ )

Матрица  $\Sigma$  — диагональная с неотрицательными диагональными элементами (*сингулярными числами*), записанными в порядке убывания

В матрице  $\mathbf{U}$  по столбцам — *левые сингулярные векторы*

В матрице  $\mathbf{V}$  по столбцам — *правые сингулярные векторы* (это и есть главные компоненты!)

Пусть  $d < N$ , тогда лучше использовать *неполное сингулярное разложение*:

$$\underbrace{\mathbf{X}}_{N \times d} = \underbrace{\mathbf{U}}_{N \times d} \cdot \underbrace{\Sigma}_{d \times d} \cdot \underbrace{\mathbf{V}^T}_{d \times d}$$

Если  $d > N$ , то наоборот:

$$\underbrace{\mathbf{X}}_{N \times d} = \underbrace{\mathbf{U}}_{N \times N} \cdot \underbrace{\Sigma}_{N \times N} \cdot \underbrace{\mathbf{V}^T}_{N \times d}$$

Пусть данные в  $\mathbf{X}$  центрированы, тогда

$$C = \frac{1}{N-1} \mathbf{X}^T \mathbf{X} = \frac{1}{N-1} (\mathbf{U} \Sigma \mathbf{V}^T)^T (\mathbf{U} \Sigma \mathbf{V}^T) = \frac{1}{N-1} \mathbf{V} \Sigma^2 \mathbf{V}^T = \mathbf{V} \left( \frac{1}{N-1} \Sigma^2 \right) \mathbf{V}^T$$

Итак,  $\mathbf{V}$  составлена из главных компонент (правых сингулярных векторов).

С точностью до перестановки столбцов  $\mathbf{V} = Q$ .

$\frac{1}{N-1} \Sigma^2$  составлена из дисперсий по главным компонентам.

С точностью до перестановки диагональных элементов  $\frac{1}{N-1} \Sigma^2 = D$ .

Пусть используется  $k \leq d$  главных компонент.

$\mathbf{U}_k$  — первые  $k$  столбцов матрицы  $\mathbf{U}$

$\Sigma_k$  — первые  $k$  столбцов и  $k$  строк матрицы  $\Sigma$

$\mathbf{V}_k$  — первые  $k$  столбцов матрицы  $\mathbf{V}$

$$\underbrace{\mathbf{X}}_{N \times d} \approx \underbrace{\mathbf{U}_k}_{N \times k} \cdot \underbrace{\Sigma_k}_{k \times k} \cdot \underbrace{\mathbf{V}_k^\top}_{k \times d}$$

$\mathbf{V}_k$  — векторы главных компонент (по столбцам), или «матрица нагрузок» (loadings)

$\mathbf{X}\mathbf{V}_k = \mathbf{U}_k\Sigma_k$  — проекции векторов на г.к., или «матрица счетов» (scores)

**Упражнение 2.9** Доказать, что  $\mathbf{X}\mathbf{V}_k = \mathbf{U}_k\Sigma_k$

$\mathbf{X}(\Sigma_k)^{-1}\sqrt{N} = \mathbf{U}_k\sqrt{N}$  — нормировка на единичную дисперсию, или «матрица  $z$ -счетов» ( $z$ -scores) — whitening

$\mathbf{X} - \mathbf{U}_k\Sigma_k\mathbf{V}_k^\top$  — матрица ошибок

## 2.8. Напоминание: Проверка статистических гипотез

Пусть функция распределения  $P^*(x)$  случайной величины  $X$  не известна.

Гипотеза  $H_0: P^*(x) = P(x)$ .

Альтернативная гипотеза  $H_1$ .

Ошибка 1-го рода: Гипотеза  $H_0$  отвергается (и принимается  $H_1$ ), хотя  $H_0$  верна.

Ошибка 2-го рода: Гипотеза  $H_0$  принимается, хотя она не верна.

Аналогично для параметрических гипотез.

Вначале выбирается  $\alpha$  — вероятностный уровень значимости критерия, равный вероятности того, что произойдет ошибка 1-го рода (например, 0.1, 0.05, 0.01)

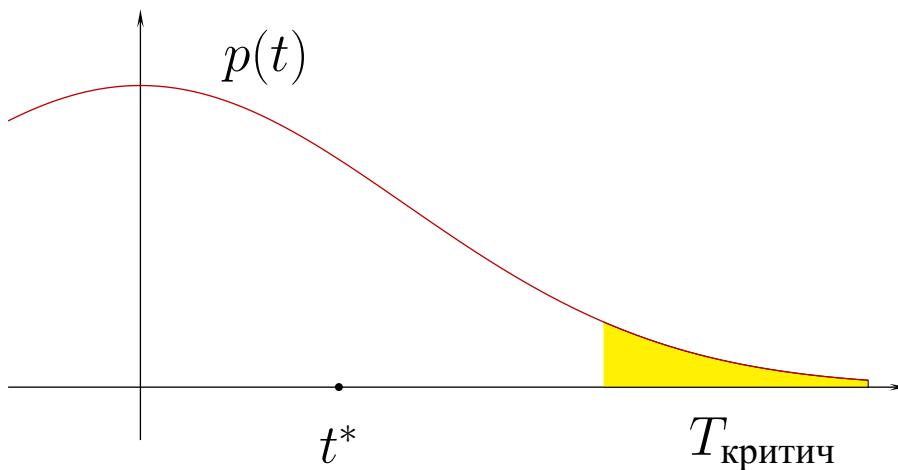
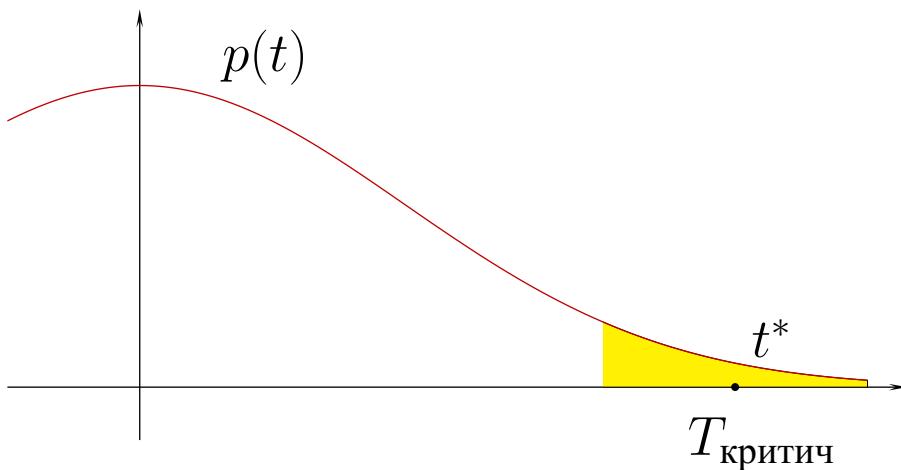
Вычисляется реализация  $t^*$  статистики  $t(X)$ , характеризующей отклонение эмпирических данных от теоретических (соответствующих гипотезе  $H_0$ ).

Пусть  $T_{\text{критич}} = T_{\text{критич}}(H_0, \alpha)$  — критическая область — подмножество значений статистики  $t(X)$ , такое, что

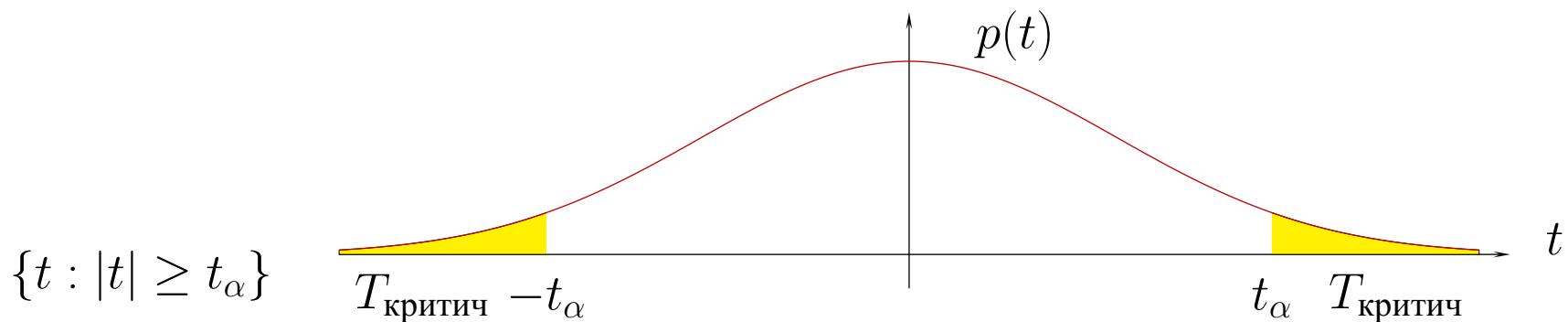
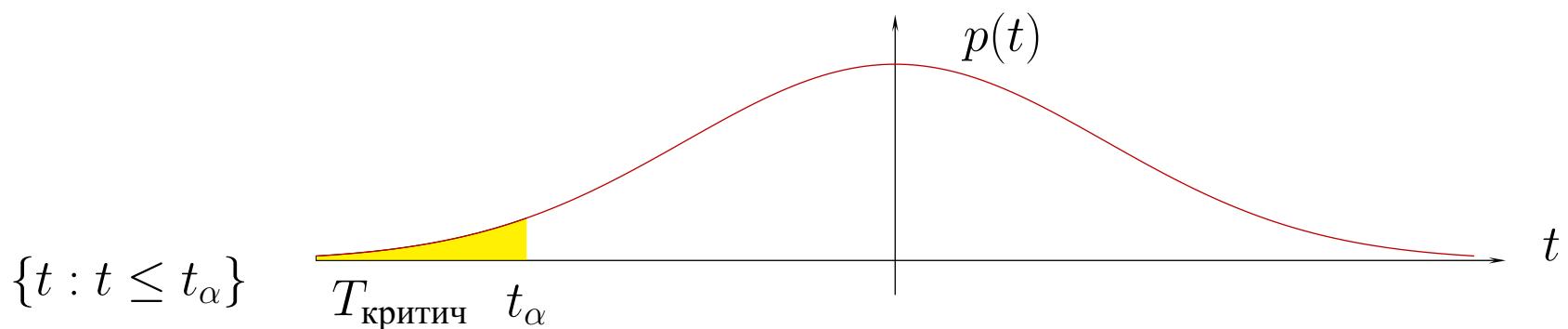
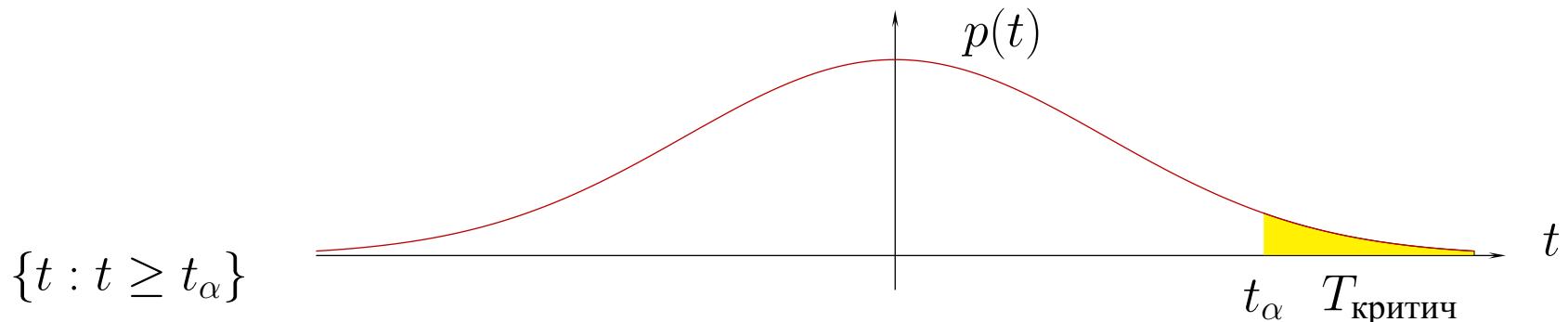
$$\Pr(t \in T_{\text{критич}} \mid H_0) = \alpha.$$

Если  $t^* \in T_{\text{критич}}$ , то гипотеза  $H_0$  отвергается.

Если  $t^* \notin T_{\text{критич}}$ , то можно считать, что эмпирические данные согласуются с гипотезой  $H_0$ , и гипотеза  $H_0$  принимается.



Часто в качестве  $T_{\text{критич}}(H_0, \alpha)$  выбираются области вида



Площадь желтой области на каждом графике равна вероятностному уровню значимости  $\alpha$ .

## 2.8.1. p-value

p-value принадлежит интервалу  $[0, 1]$  и вычисляется по выборочному значению  $t^*$  статистики  $t(X)$  (и зависит также от критерия и гипотезы  $H_0$ , но не зависит от  $\alpha$ ).

p-value — это минимальное значение уровня значимости  $\alpha$ , при котором данное значение  $t^*$  статистики  $t(X)$  принадлежит критической области  $T_{\text{критич}}(H_0, \alpha)$ :

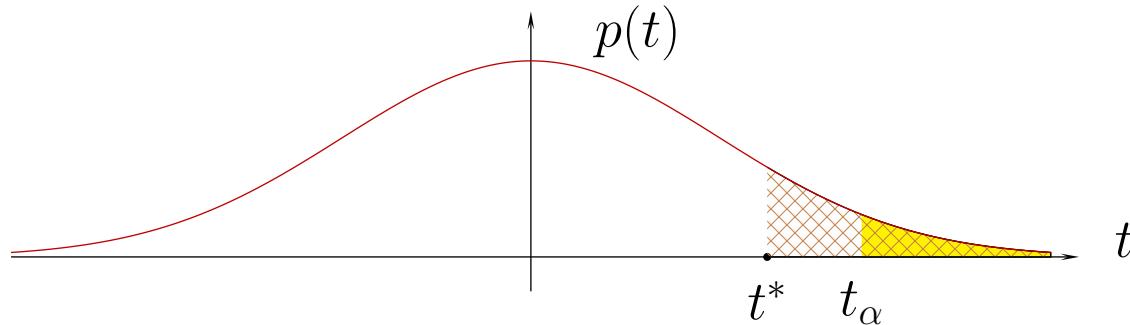
$$\text{p-value}(t^*, H_0) = \inf \{\alpha : t^* \in T_{\text{критич}}(H_0, \alpha)\}$$

Из этого определение вытекает следующее основное свойство p-value.

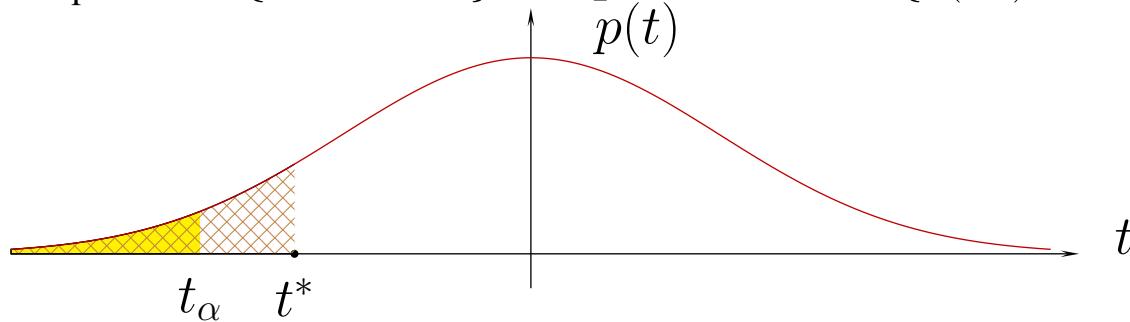
Если p-value меньше  $\alpha$ , то гипотеза  $H_0$  отвергается.

Иначе гипотезу  $H_0$  отвергнуть нельзя.

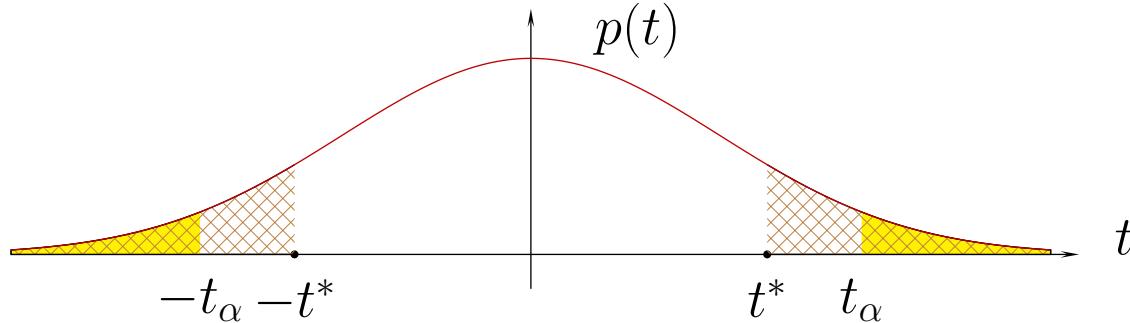
Если  $T_{\text{критич}} = \{t : t \geq t_\alpha\}$ , то p-value =  $\Pr \{t(X) \geq t^*\}$



Если  $T_{\text{критич}} = \{t : t \leq t_\alpha\}$ , то p-value =  $\Pr \{t(X) \leq t^*\}$



Если  $T_{\text{критич}} = \{t : |t| \geq t_\alpha\}$ , то p-value =  $\Pr \{|t(X)| \geq |t^*|\}$



p-value = площади заштрихованной фигуры,  $\alpha$  = площади желтой фигуры

*Глава 3*

## **Метод наименьших квадратов**

# Agenda

- Регрессионная функция
  - Метод наименьших квадратов
  - Метод максимального правдоподобия
- Линейная регрессия
- Оценка коэффициентов по выборке
- Переобучение
- Сокращение числа параметров и «усадка» коэффициентов
  - Выбор подмножества параметров
  - Гребневая регрессия
  - Лассо
  - Метод главных компонент
  - Частичные наименьшие квадраты

### **3.1. Регрессия**

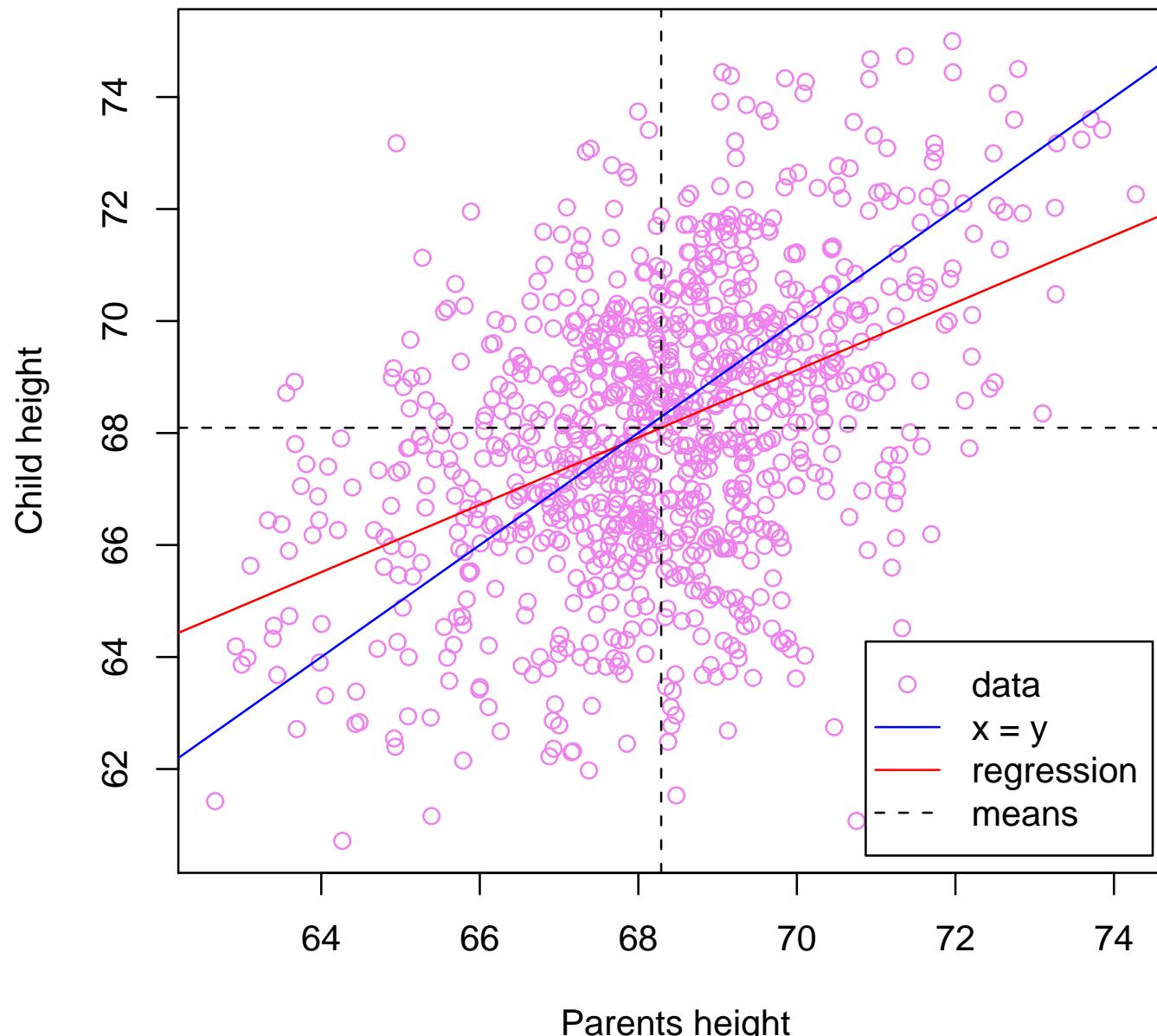
Гаусс (1794–1795), Лежандр (1805–1806) — метод наименьших квадратов  
(Исследование траектории астероида Церера)

Фрэнсис Гальтона (1822–1911)

«Регрессия к середине в наследовании роста» (1885 г.)

«Где это возможно, считайте». Ф.Гальтон

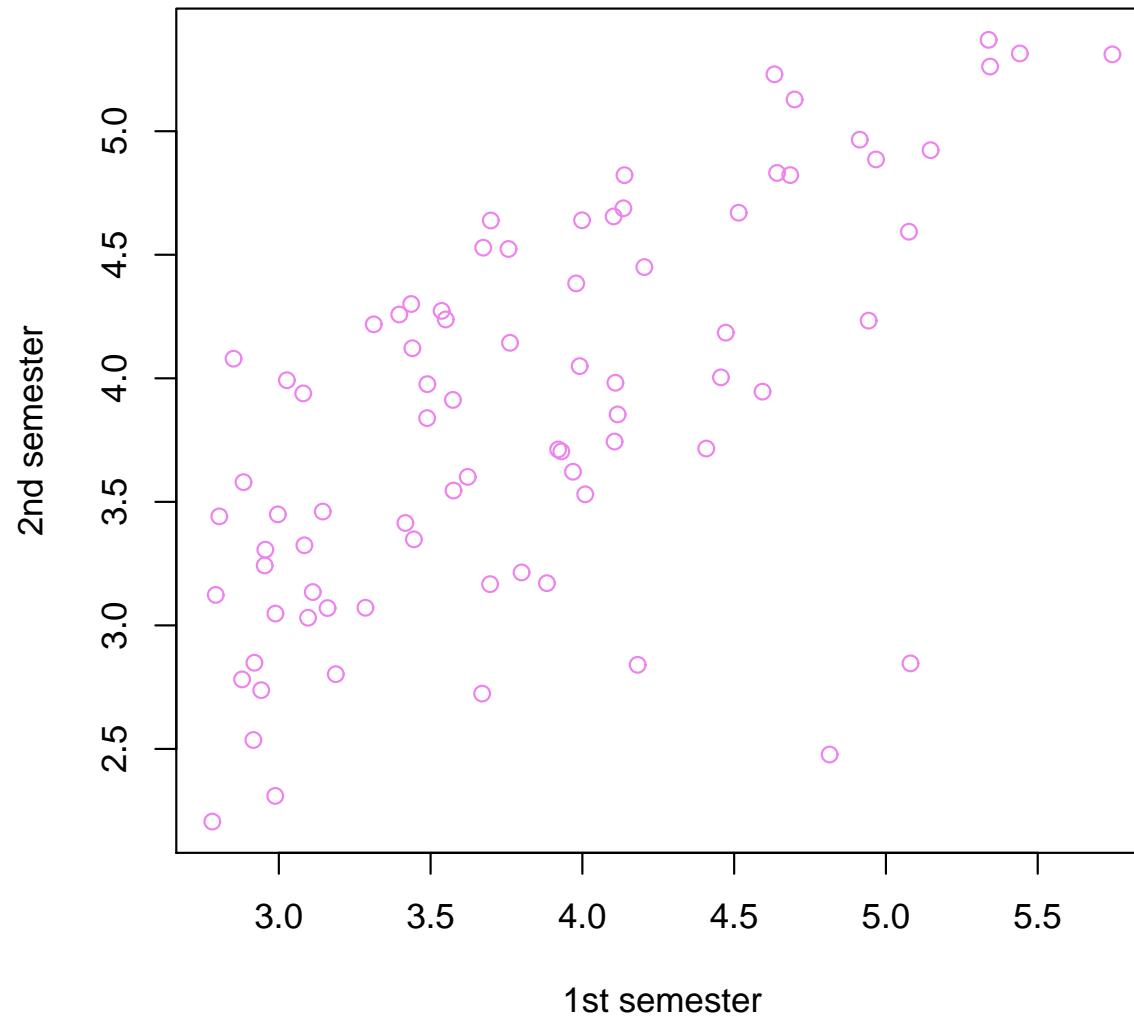
Зависимость роста ребенка от роста родителей в исследовании Ф. Гальтона



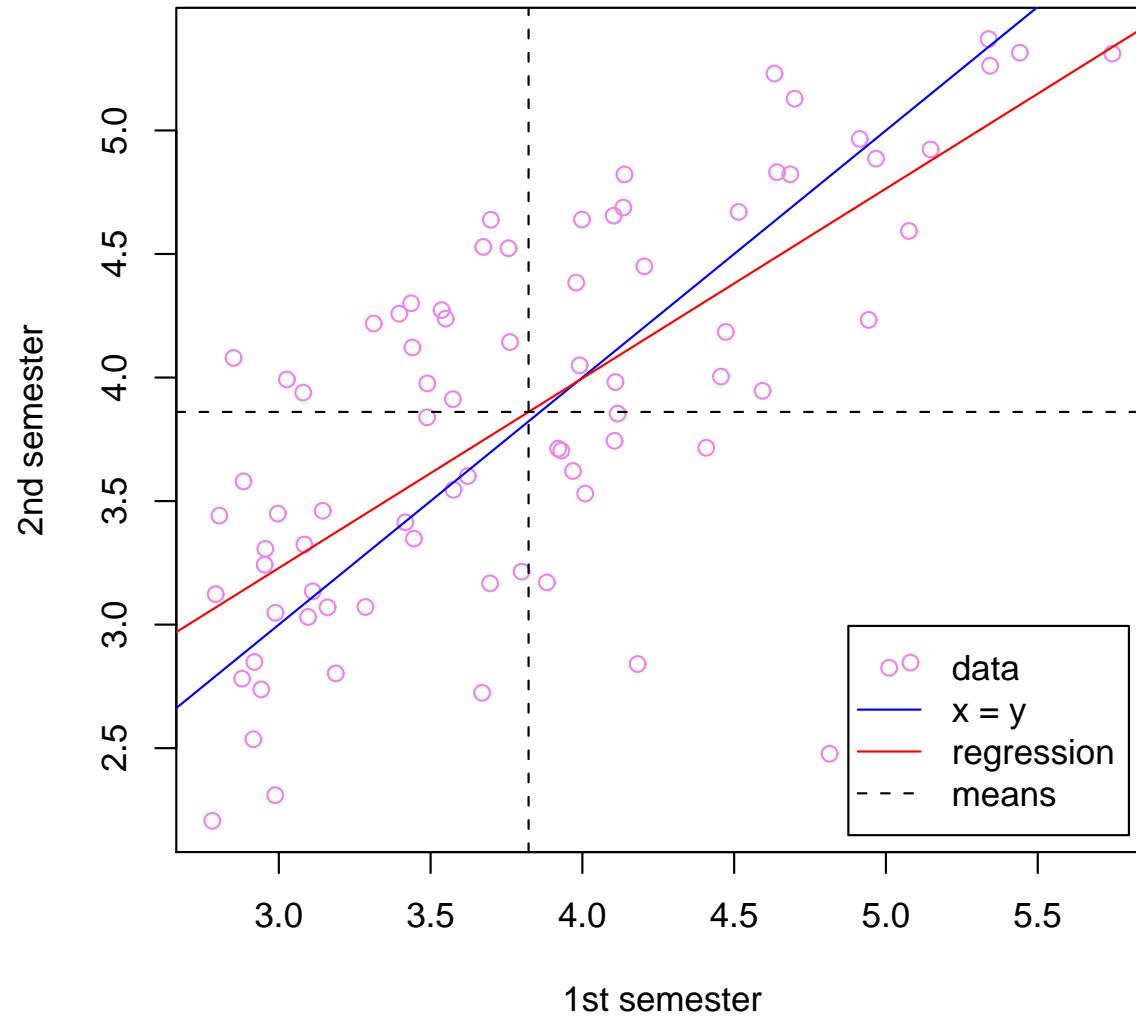
928 наблюдений  $ch = 0.65\text{par} + 24 = 68.2 + 0.65 \times (\text{par} - 68.2)$

$x$  = средняя оценка по мат. анализу и алгебре в 1-м семестре

$y$  = средняя оценка по мат. анализу, алгебре и программированию во 2-м семестре



79 студентов

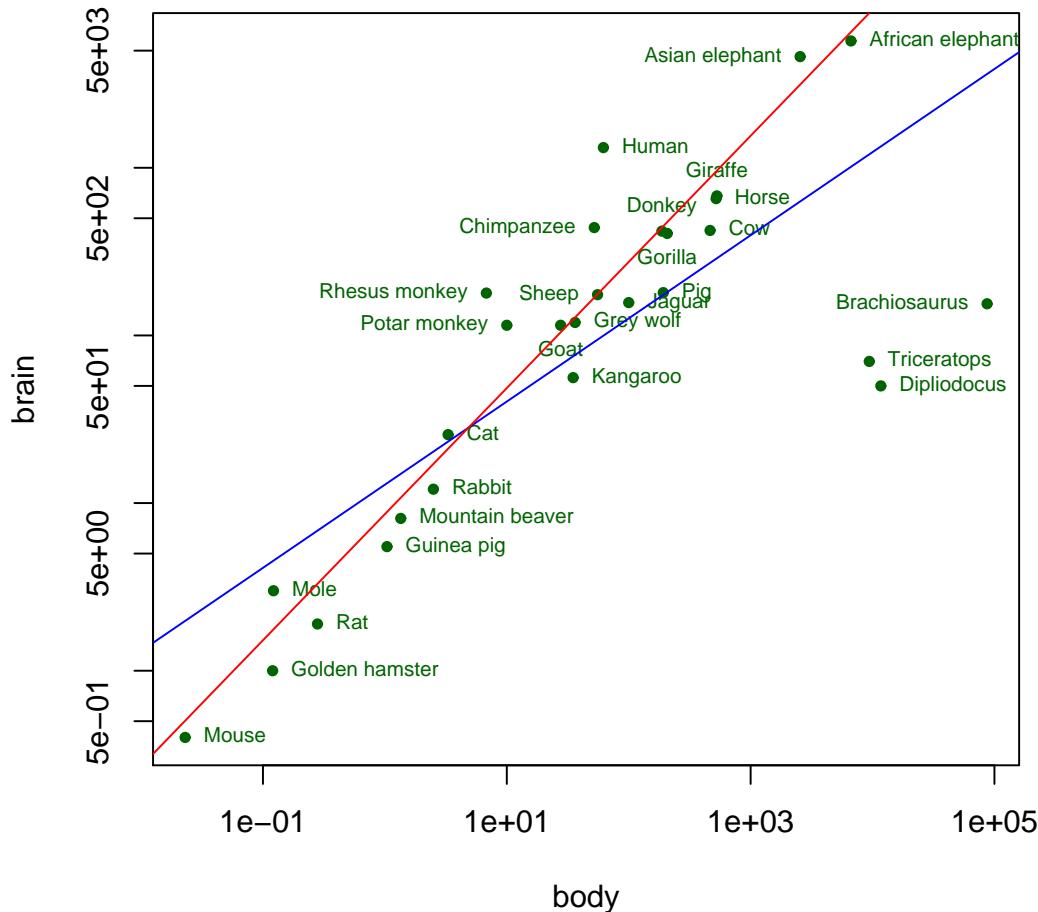


$$\text{sem2} = 0.93 + 0.77 \times \text{sem1} \approx 3.86 + 0.77 \times (\text{sem1} - 3.82)$$

3.82 — средняя оценка по всем студентам в 1-м семестре

3.86 — средняя оценка по всем студентам во 2-м семестре

## Зависимость между массой тела и массой мозга животного



$$\lg \text{brain} = \beta_0 + \beta_1 \lg \text{body}$$

$$\beta_0 = 0.94, \beta_1 = 0.75, \text{brain} = 8.6 \times (\text{body})^{3/4}$$

Найдите на картинке выбросы (outlayers).

## Обучающая выборка

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

$$x^{(i)} \in \mathcal{X}, \quad y^{(i)} \in \mathcal{Y} = \mathbf{R} \quad (i = 1, 2, \dots, N)$$

$$f^*(x^{(i)}) \approx y^{(i)} \quad (i = 1, 2, \dots, N)$$

Будем искать функцию  $f^*(x)$  в виде (*линейная регрессионная модель*):

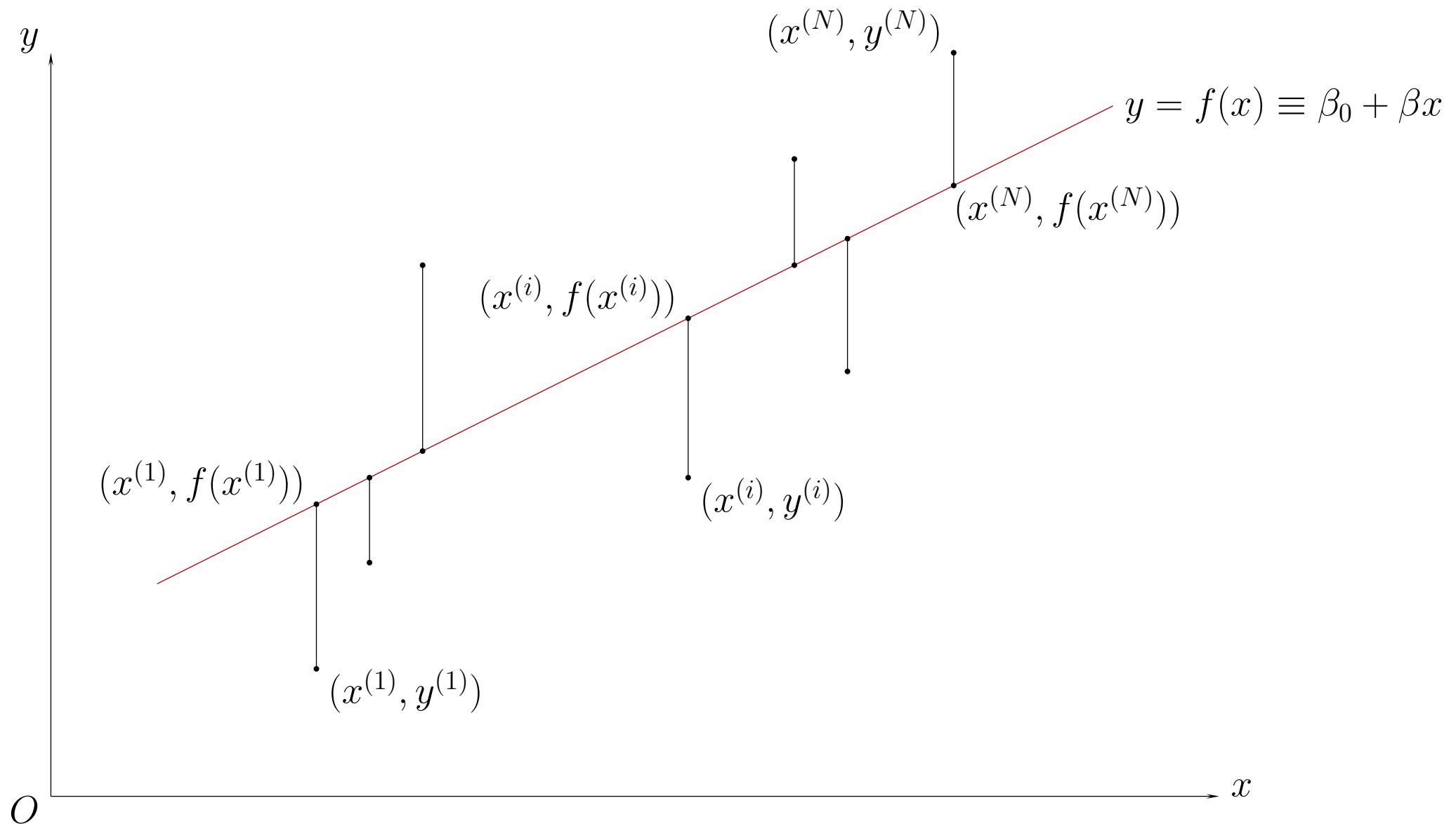
$$f(x) = f(x, \beta) = \beta_0 + \sum_{j=1}^d \beta_j x_j$$

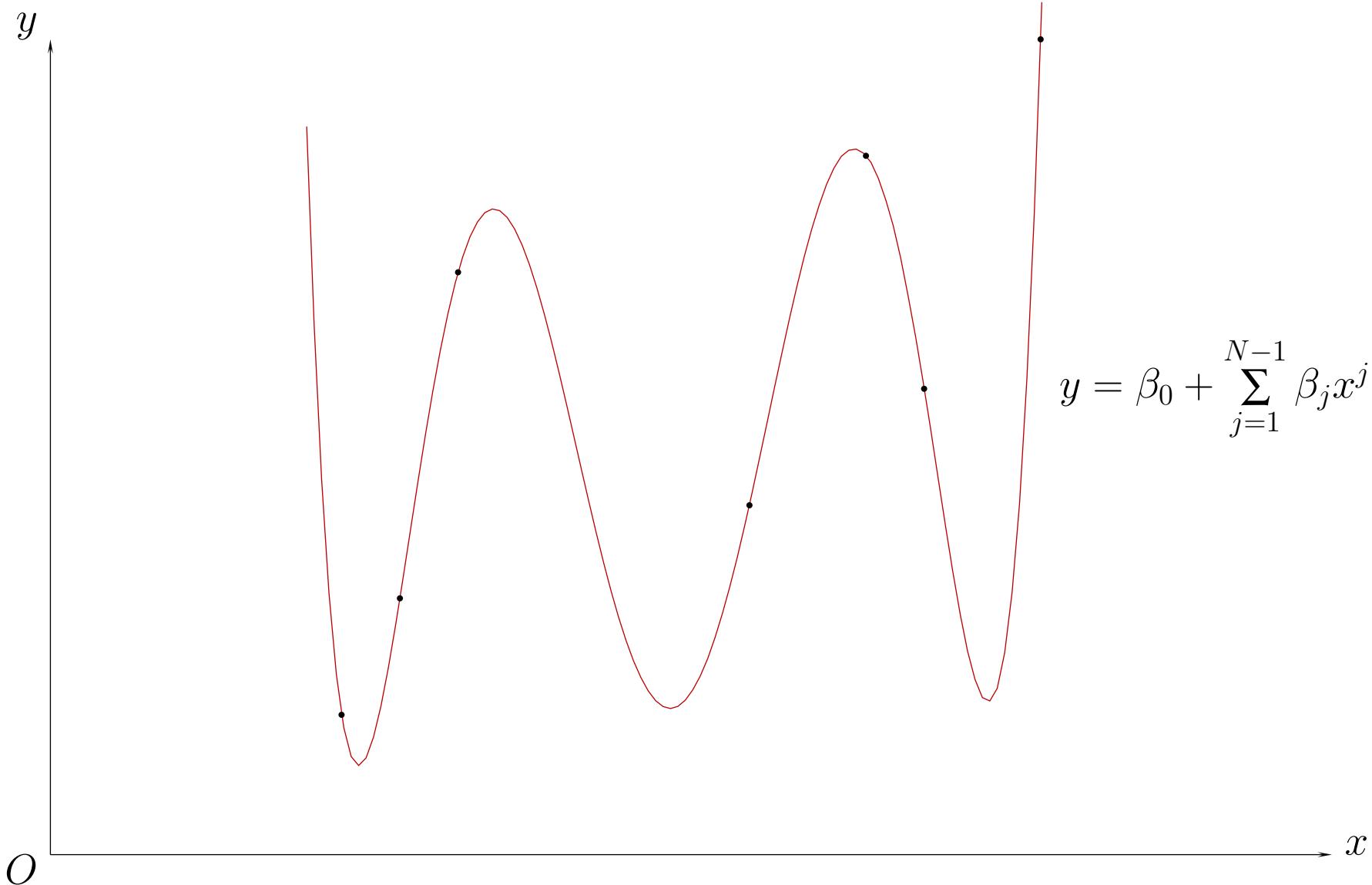
*Метод наименьших квадратов* — один из методов нахождения неизвестных параметров  $\beta_0, \beta_1, \dots, \beta_d$ .

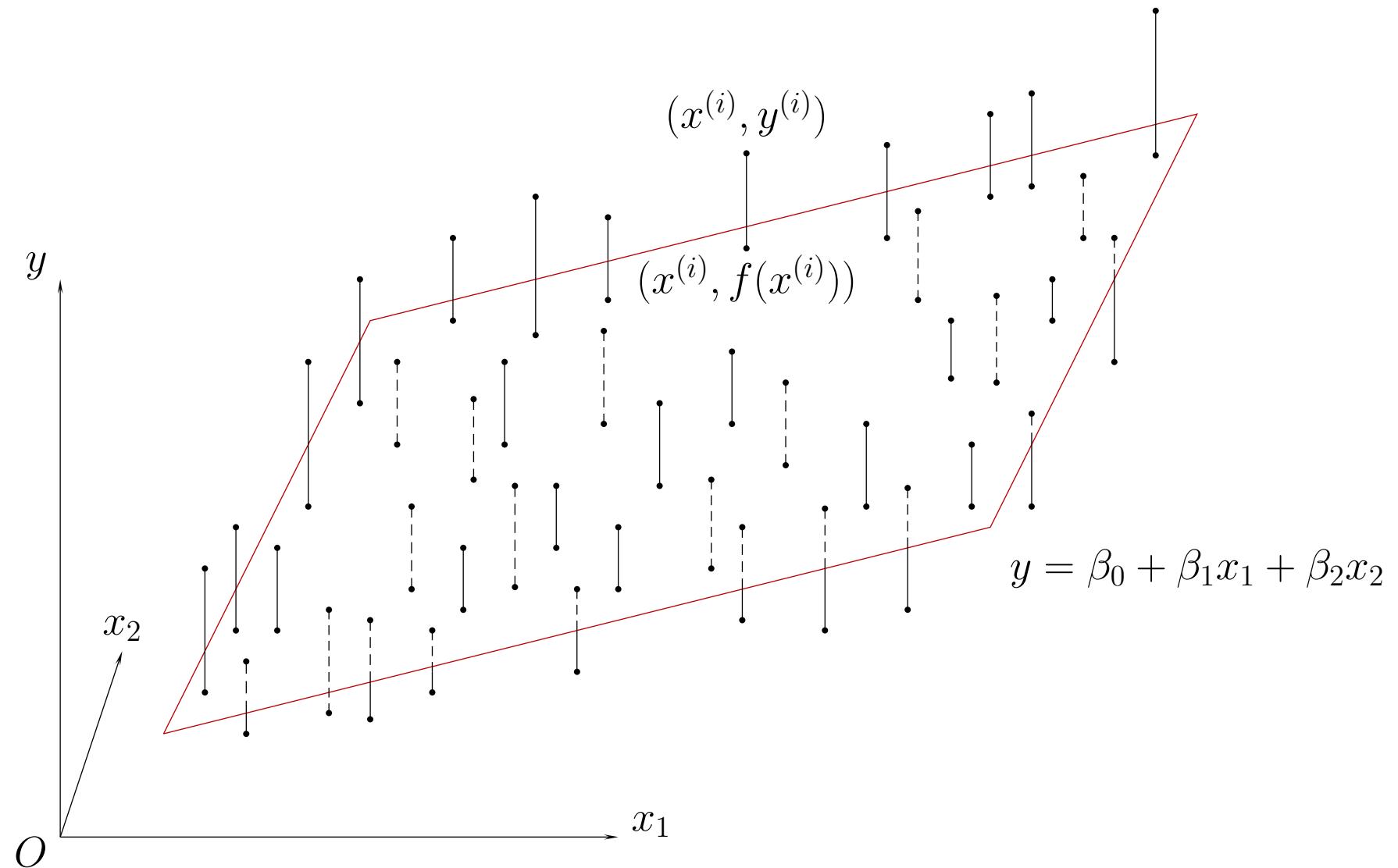
Ищем набор параметров  $\hat{\beta}$ , доставляющих минимум *сумме квадратов невязок*, или *остаточной сумме квадратов (residual sum of squares)*

$$\text{RSS}(\beta) = \sum_{i=1}^N \left( y^{(i)} - f(x^{(i)}, \beta) \right)^2.$$

Заметим, что  $\frac{1}{N} \text{RSS}(\beta)$  есть эмпирический риск для квадратичной функции потерь.







Функция  $f^*(x)$  может быть функцией более общего вида:

$$f(x) = f(x, \beta) = \sum_{j=1}^q \beta_j h_j(x),$$

где  $\beta_j$  — неизвестные параметры, а  $h_j(x)$  — заданные функции

Заметим, что параметры  $\beta_j$  входят *линейным* образом.

Но неизвестные параметры могут входить и *нелинейным* образом:

Например,

$$y = \beta_1 e^{\lambda_1 x} + \beta_2 e^{\lambda_2 x}.$$

$\beta_1, \beta_2$  — линейно,  $\lambda_1, \lambda_2$  — нелинейно.

Если  $\lambda_1, \lambda_2$  известны, то получаем линейную задачу (наименьших квадратов)

Если  $\lambda_1, \lambda_2$  не известны — нелинейную задачу (наименьших квадратов)

Задача аппроксимации сплайнами тоже сводится к линейной задаче наименьших квадратов.

## Почему метод наименьших квадратов?

Выведем (при некоторых дополнительных предположениях) метод наименьших квадратов из *принципа максимального правдоподобия*.

$Y$  — с. в. с плотностью вероятности  $p(y, \theta)$ , где  $\theta$  — вектор параметров.

$N$  копий непрерывной случайной величины  $Y$ :  $Y^{(1)}, Y^{(2)}, \dots, Y^{(N)}$

( $N$  независимых одинаково распределенных с.в. — испытания Бернулли)

$N$  реализаций этих величин:  $y^{(1)}, y^{(2)}, \dots, y^{(N)}$

Плотность вероятности  $N$ -мерной с.в.  $(Y^{(1)}, Y^{(2)}, \dots, Y^{(N)})$ :

$$L(\theta) = p(y^{(1)}, y^{(2)}, \dots, y^{(N)}, \theta) = p(y^{(1)}, \theta) \cdot p(y^{(2)}, \theta) \cdot \dots \cdot p(y^{(N)}, \theta)$$

$L(\theta)$  — функция правдоподобия

Логарифмическая функция правдоподобия:

$$\ell(\theta) = \ln L(\theta) = \sum_{i=1}^N \ln p(y^{(i)}, \theta).$$

*Принцип максимального правдоподобия* предполагает, что наиболее разумные значения неизвестных параметров  $\theta$  доставляют максимум функции  $L(\theta)$  (и  $\ell(\theta)$ ).

(Если  $Y$  — дискретная, то вместо  $p(y^{(i)}, \theta)$  нужно рассмотреть  $\Pr \{Y = y^{(i)}\}$ )

Модель с аддитивной случайной ошибкой:

$$y = f^*(x) + E,$$

где  $E$  – случайная величина (ошибка), не зависящая от  $x$ , и  $\mathbb{E} E = 0$ .

$f^*(x) = \mathbb{E}(Y | X = x)$  – регрессионная функция.

Дополнительно предположим, что  $E \sim N(0, \sigma^2) \Leftrightarrow$

$$p(y | x, \beta) = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{1}{2} \frac{(y - f(x, \beta))^2}{\sigma^2}}$$

Тогда

$$\ell(\beta) = \sum_{i=1}^N \ln p(y^{(i)} | x, \beta) = -\frac{N}{2} \ln 2\pi - N \ln \sigma - \frac{1}{2\sigma^2} \sum_{i=1}^N (y^{(i)} - f(x^{(i)}, \beta))^2$$

В ней только последний член содержит вектор параметров  $\beta$ .

С точностью до множителя этот член равен  $\text{RSS}(\beta)$

Итак, при сделанных предположениях метод наименьших квадратов эквивалентен принципу максимального правдоподобия

Как найти минимум функции  $\text{RSS}(\beta)$ ?

Пусть

$$\mathbf{X} = \begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \dots & x_d^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \dots & x_d^{(2)} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_1^{(N)} & x_2^{(N)} & \dots & x_d^{(N)} \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(N)} \end{pmatrix}, \quad \beta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_d \end{pmatrix}$$

Тогда

$$\text{RSS}(\beta) = \|\mathbf{y} - \mathbf{X}\beta\|^2 = (\mathbf{y} - \mathbf{X}\beta)^\top(\mathbf{y} - \mathbf{X}\beta) \rightarrow \min$$

Можно рассмотреть систему уравнений (относительно  $\beta$ )

$$\mathbf{X}\beta = \mathbf{y}$$

$\hat{\beta}$  называется *псевдорешением* этой системы (оно минимизирует норму невязки).

$\text{RSS}(\beta)$  — квадратичная функция от  $d + 1$  неизвестных (параметров)  $\beta_0, \beta_1, \dots, \beta_d$ .

Дифференцируя, находим:

$$\frac{\partial \text{RSS}}{\partial \beta} = -2\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\beta), \quad \frac{\partial^2 \text{RSS}}{\partial \beta \partial \beta^\top} = 2\mathbf{X}^\top \mathbf{X}.$$

Обозначим  $\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_d$  столбцы матрицы  $\mathbf{X}$ .

Если  $\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_d$  линейно независимы, то матрица  $\mathbf{X}^\top \mathbf{X}$  невырождена и положительно определена, поэтому минимум функции  $\text{RSS}(\beta)$  достигается, когда первая производная по  $\beta$  обращается в ноль:

$$\mathbf{X}^\top (\mathbf{y} - \mathbf{X}\beta) = 0 \quad \Leftrightarrow \quad \mathbf{X}^\top \mathbf{X}\beta = \mathbf{X}^\top \mathbf{y}.$$

Это *нормальная система уравнений*, или *система нормальных уравнений*.

Единственным решением является вектор

$$\hat{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

Итак, *псевдорешением* системы  $\mathbf{X}\beta = \mathbf{y}$  является *решение* системы  $\mathbf{X}^\top \mathbf{X}\beta = \mathbf{X}^\top \mathbf{y}$ .

Матрица  $\mathbf{X}^+ = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$  называется *псевдообратной* (Мура–Пенроуза) к  $\mathbf{X}$ .

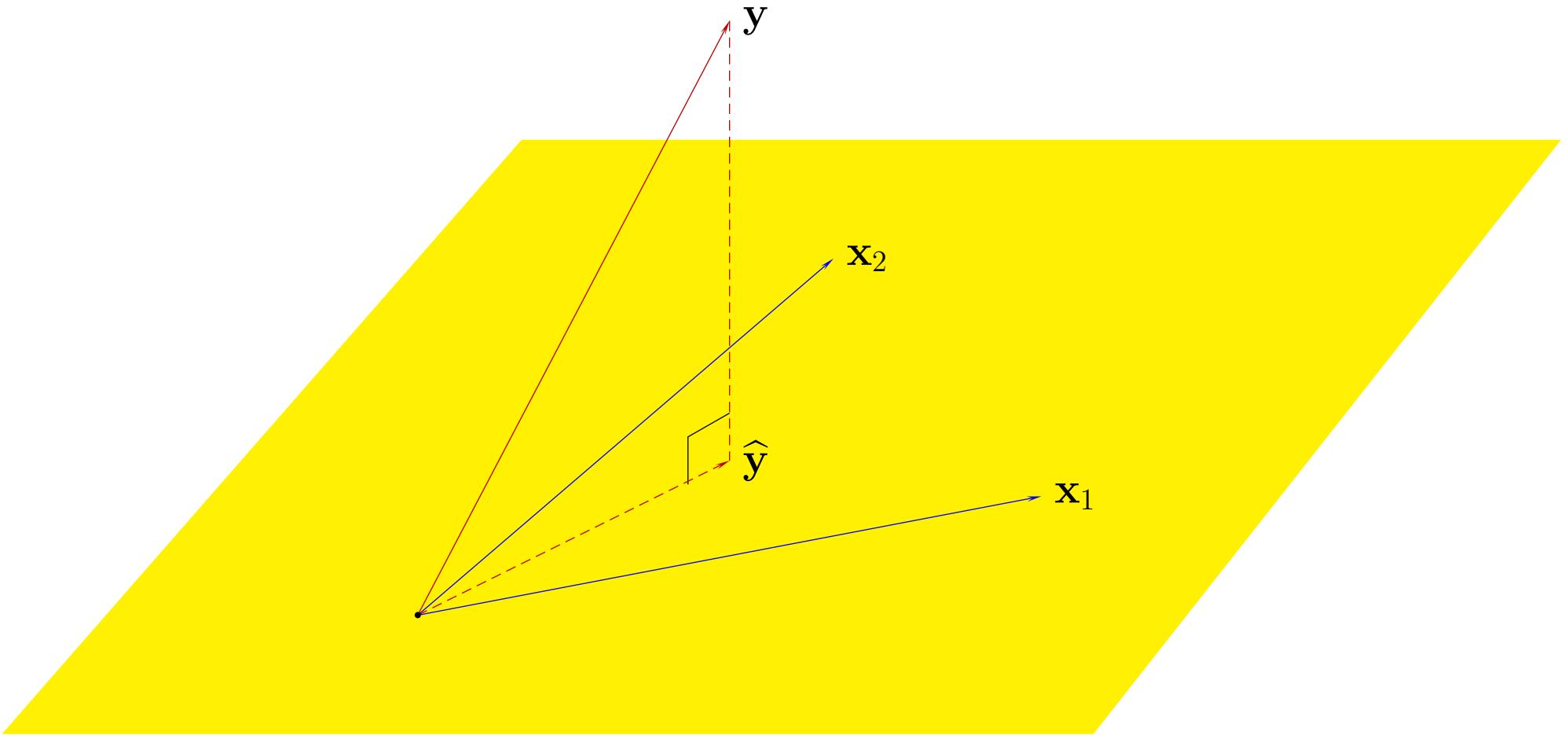
Входным значениям  $x_1, x_2, \dots, x_N$  будет соответствовать вектор выходных переменных

$$\hat{\mathbf{y}} = (\hat{y}_1, \hat{y}_2, \dots, \hat{y}_d)^\top = \mathbf{X}\hat{\beta} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}.$$

Пусть  $\mathbf{H} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top$ , тогда получаем  $\hat{\mathbf{y}} = \mathbf{H}\mathbf{y}$

$\hat{\mathbf{y}}$  есть ортогональная проекция  $\mathbf{y}$  на п/п-во, натянутое на  $\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_d$

$\mathbf{H}$  называется *матрицей проектирования*



Если столбцы матрицы  $\mathbf{X}$  линейно зависимы, то  $\widehat{\beta}$ , на котором достигается минимум функции  $\text{RSS}(\beta)$ , не единственен, однако, по-прежнему,  $\widehat{y}$  является ортогональной проекцией вектора  $y$  на линейную оболочку векторов  $\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_d$ .

### 3.2. Проверка значимости и доверительные интервалы для коэффициентов (регрессионный анализ)

$\mathbf{y}, E, \hat{\beta}$  случайны;  $\mathbf{X}, \beta$  детерминированы;  $\mathbb{E} E = 0, \text{Cov } E = \sigma^2 \mathbf{I}$ .

$$\mathbf{y} = \mathbf{X}\beta + E, \quad \hat{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top (\mathbf{X}\beta + E).$$

$$\hat{\beta} - \beta = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top (\mathbf{X}\beta + E) - \beta = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top E$$

Несмешенность:

$$\mathbb{E}(\hat{\beta} - \beta) = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbb{E} E = 0 \quad \Rightarrow \quad \mathbb{E} \hat{\beta} = \beta.$$

Матрица ковариации:

$$\begin{aligned} \text{Cov} \hat{\beta} &= \mathbb{E}((\hat{\beta} - \beta)(\hat{\beta} - \beta)^\top) = \mathbb{E}\left(((\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top E)((\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top E)^\top\right) \\ &= (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \cdot \mathbb{E}(EE^\top) \cdot \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \cdot \sigma^2 \mathbf{I} \cdot \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \\ &= \sigma^2 (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} = \sigma^2 (\mathbf{X}^\top \mathbf{X})^{-1} \end{aligned}$$

Несмешенной оценкой для  $\sigma^2$  является *остаточная дисперсия*  $\hat{\sigma}^2 = \text{RSS}/(N - d - 1)$ :

$$\mathbb{E} \hat{\sigma}^2 = \mathbb{E}(\text{RSS}/(N - d - 1)) = \mathbb{E}(\mathbf{y}^\top (\mathbf{I} - \mathbf{H})\mathbf{y}/(N - d - 1)) = \sigma^2.$$

Дополнительные предположения:

Пусть ошибки  $E^{(i)}$  распределены по нормальному закону:

$$E \sim N(0, \sigma \mathbf{I}).$$

В этом случае из некоррелированности случайных величин  $E^{(i)}$  следует их независимость.

Теперь можно показать (faciat, qui potest), что

$$\hat{\beta} \sim N(\beta, (\mathbf{X}^\top \mathbf{X})^{-1} \sigma^2) \quad \text{и} \quad (N - d - 1) \hat{\sigma}^2 \sim \sigma^2 \chi_{N-d-1}^2.$$

Эти свойства будем использовать для построения статистических тестов и доверительных интервалов для  $\beta_j$ .

## **Проверка значимости модели.**

Гипотеза  $H_0: \beta_1 = \dots = \beta_d = 0$

Альтернативная гипотеза  $H_1:$  найдется  $j$ , для которого  $\beta_j \neq 0$

Если гипотеза  $H_0$  верна, то решением задачи наименьших квадратов будет

$$\beta_0 = \bar{y} = \frac{1}{N} \sum_{i=1}^N y^{(i)}.$$

В этом случае остаточная сумма квадратов (называемая в данном случае *полной суммой квадратов относительно среднего*) равна

$$\text{TSS} = \sum_{i=1}^N (y^{(i)} - \beta_0)^2 = \sum_{i=1}^d (y^{(i)} - \bar{y})^2$$

Если принимается гипотеза  $H_1$ , то говорят, что модель *статистически значима*.

Можно показать, что

$$\sum_{i=1}^d (y^{(i)} - \bar{y})^2 = \sum_{i=1}^d (y^{(i)} - \hat{y}^{(i)})^2 + \sum_{i=1}^d (\hat{y} - \bar{y})^2$$

TSS                    RSS                    ESS

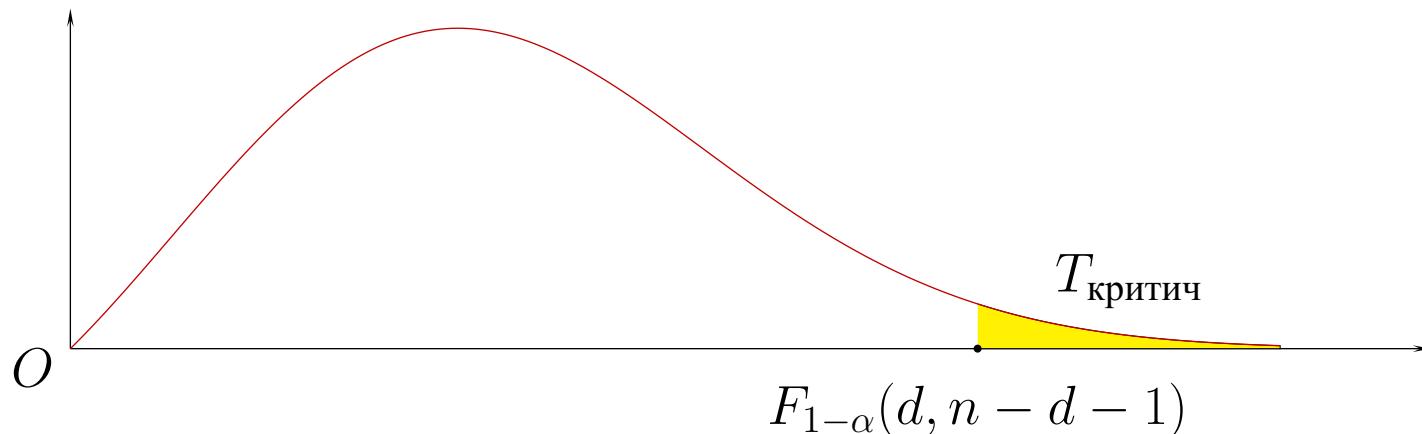
TSS =  $\sum_{i=1}^N (y^{(i)} - \bar{y})^2$  — полная сумма квадратов (total sum of squares)

ESS =  $\sum_{i=1}^N (\hat{y}^{(i)} - \bar{y})^2$  — сумма квадратов, обусловленная регрессией (explained s. of sq.)

Если гипотеза  $H_0$  верна, то

$$\hat{F} = \frac{\text{ESS}}{d\sigma^2} \Bigg/ \frac{\text{RSS}}{(n-d-1)\sigma^2} \sim F(d, n-d-1)$$

Если модель значима, то ESS большая и критическая область справа:



## **Коэффициент детерминации** (взгляд с другой стороны)

*Коэффициент детерминации, или коэффициент регрессии Пирсона*

$$R^2 = 1 - \frac{\text{RSS}}{\text{TSS}} = \frac{\text{ESS}}{\text{TSS}}. \quad (\star)$$

— доля объясняемого регрессией разброса относительно среднего.

$$0 \leq R^2 \leq 1.$$

Если  $R^2$  близок к 1, то RSS намного больше TSS.

Очевидно, что  $R = \sqrt{R^2}$  по модулю равно выборочной корреляции между  $y^{(i)}$  и  $\hat{y}^{(i)}$ .

Если  $d = 1$ , то  $R$  равен выборочной корреляции между  $x^{(i)}$  и  $y^{(i)}$ .

Если гипотеза  $H_0$  верна, то

$$\hat{F} = \frac{R^2/d}{(1-R^2)/(N-d-1)} \sim F(d, n-d-1).$$

Недостаток  $R^2$ : при добавлении в модель новых признаков  $R^2$  не изменяется. (Если  $d$  велико и близко к  $N$ , то может быть, что  $\text{RSS} \approx 0$  и  $R^2 \approx 1$ .) В этом случае используется скорректированный («подправленный») коэффициент

$$R_a^2 = R^2 - \frac{1-R^2}{N-d-1}.$$

**Упражнение 3.1** Доказать, что  $TSS = RSS + ESS$ . Это можно доказать непосредственное, а можно предварительно доказать, что  $\mathbf{y} - \hat{\mathbf{y}} \perp \hat{\mathbf{y}} - \bar{\mathbf{y}}$ , где  $\bar{\mathbf{y}}$  – вектор, составленный из  $\bar{y}$ , и воспользоваться этим.

**Упражнение 3.2** Разности

$$e^{(i)} = y^{(i)} - \hat{y}^{(i)} = y^{(i)} - \sum_{j=1}^d \hat{\beta}_j x_j^{(i)}$$

называются *остатками*. Доказать, что

$$\sum_{i=1}^N e^{(i)} = 0.$$

**Упражнение 3.3** Доказать, что

$$\bar{y} = \beta_0 + \sum_{j=1}^d \hat{\beta}_j \bar{x}_j.$$

**Упражнение 3.4** Доказать, что

$$\sum_{i=1}^N y^{(i)} = \sum_{i=1}^N \hat{y}^{(i)}.$$

## **Проверка значимости одного коэффициента.**

Гипотеза  $H_0: \beta_j = 0$  ( $j$  фиксировано):

использование переменной  $x_j$  не улучшает предсказание по сравнению с предсказанием, полученным на основе только остальных  $d - 1$  переменных.

Для проверки этой гипотезы (против гипотезы  $\beta_j \neq 0$ ) рассмотрим *стандартный коэффициент (z-score)*

$$t_j = \frac{\widehat{\beta}_j}{\text{se } \beta_j}, \quad (\star)$$

где

$$\text{se } \beta_j = \widehat{\sigma} \sqrt{v_j}$$

— *стандартная ошибка коэффициента  $\beta_j$* , а  $v_j$  —  $j$ -й диаг. элемент матрицы  $(\mathbf{X}^\top \mathbf{X})^{-1}$ .

В предположении  $\beta_j = 0$  коэффициент  $t_j$  имеет  $t$ -распределение Стьюдента  $t_{N-d-1}$ .

Если  $|t_j|$  «велико», то гипотезу  $H_0$  следует отбросить.

Если гипотеза  $H_0$  отбрасывается, то говорят, что коэффициент  $\widehat{\beta}_j$  *статистически значим*.

Можно проверить гипотезу  $\beta_j = \beta'_j$  (относительно односторонней или двусторонней альтернативы), где  $\beta'_j$  — некоторое заданное значение.

Статистика критерия имеет в этом случае вид

$$t'_j = \frac{\hat{\beta}_j - \beta'_j}{\text{se } \beta_j}.$$

Коэффициент  $t'_j$  имеет распределение  $t_{N-d-1}$ .

Проверка гипотезы зависит от вида альтернативной гипотезы и происходит обычным образом.

**Проверка значимости группы коэффициентов.** Гипотеза о равенстве нулю группы коэффициентов (против гипотезы, что по крайней мере один из коэффициентов не равен нулю): переменные этой группы не улучшают предсказание по отношению к предсказанию, полученному без этих переменных.

Будем использовать статистику

$$F = \frac{(\text{RSS}_2 - \text{RSS}_1)/(d_1 - d_2)}{\text{RSS}_1 / (N - d_1 - 1)},$$

где  $\text{RSS}_1$  — остаточная сумма квадратов «большой» модели с  $d_1 + 1$  параметрами, а  $\text{RSS}_2$  — остаточная сумма квадратов «вложенной» модели с  $d_2 + 1$  параметрами, («вложенная» модель получается из «большой» обнулением  $d_1 - d_2$  параметров).

В предположении, что  $E$  имеет нормальное распределение, статистика  $F$  имеет  $F(d_1 - d_2, N - d_1 - 1)$  распределение Фишера.

Если отбрасывается один коэффициент, то  $F$  равен квадрату стандартного коэффициента  $t_j$  из  $(\star)$ .

**Доверительные интервалы.** Для  $\beta_j$  доверительным интервалом является

$$(\hat{\beta}_j - z^{(1-\alpha)} \hat{\sigma} \sqrt{v_j}, \quad \hat{\beta}_j + z^{(1-\alpha)} \hat{\sigma} \sqrt{v_j}),$$

где  $z^{(1-\alpha)}$  есть  $(1 - \alpha)$ -процентиль для нормального распределения:

$$z^{(1-0.1)} = 1.645,$$

$$z^{(1-0.05)} = 1.96,$$

$$z^{(1-0.01)} = 2.58, \quad \text{и т. д.}$$

( $v_j$  —  $j$ -й диаг. элемент в  $(\mathbf{X}^\top \mathbf{X})^{-1}$ ,  $\text{se } \hat{\beta}_j = \hat{\sigma} \sqrt{v_j}$  — стандартная ошибка для  $\hat{\beta}_j$ ).

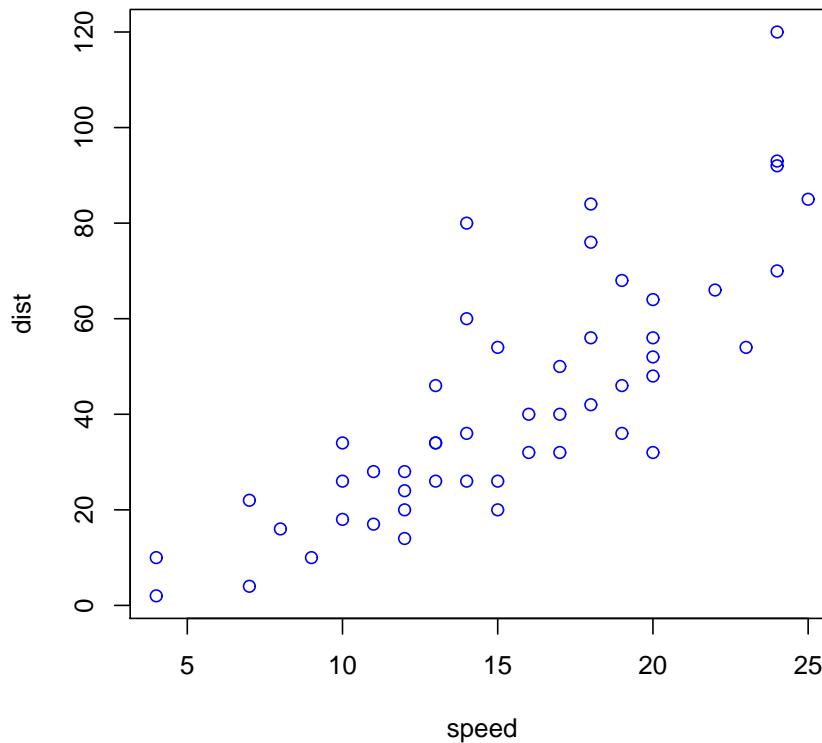
Таким образом, интервал  $\hat{\beta} \pm 2 \cdot \text{se } \hat{\beta}$  соответствует мере доверия около 95%.

### 3.2.1. Пример

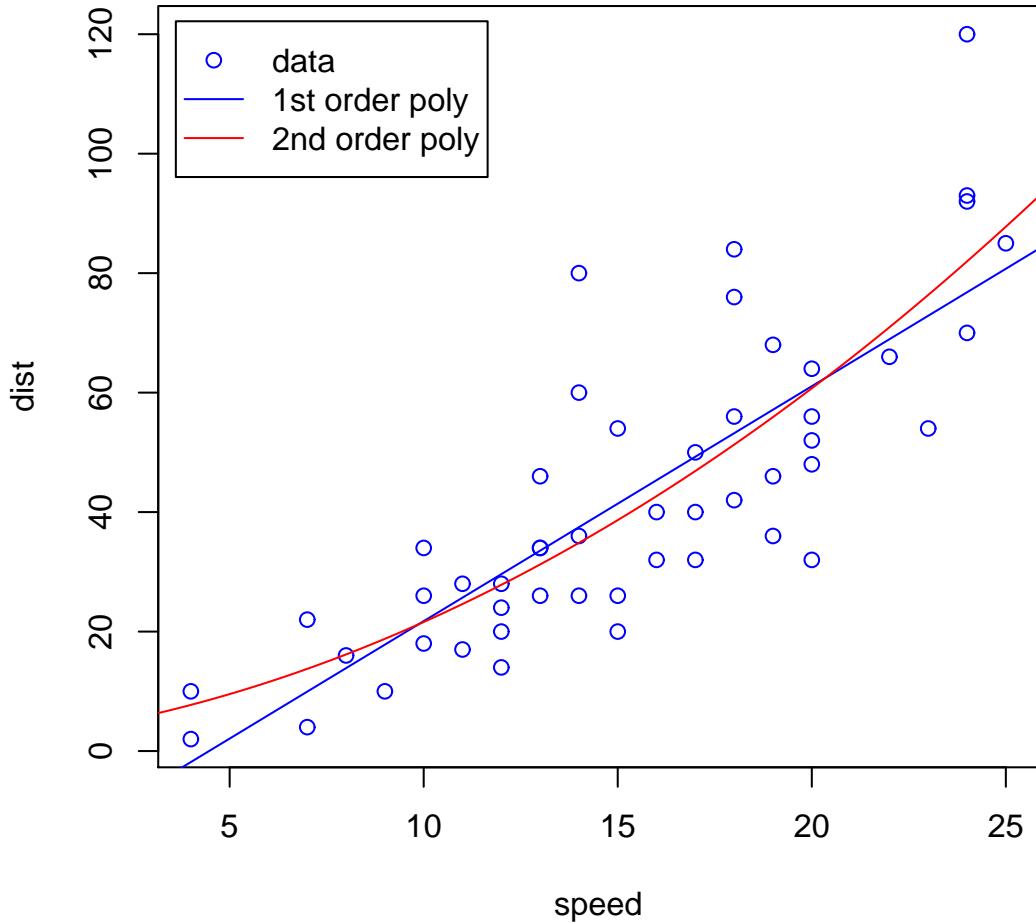
Зависимость длины тормозного пути (dist) от начальной скорости автомобиля (speed) [Ezekiel M. Methods of Correlation Analysis. Wiley. 1930],  $N = 50$ .

В качестве модели рассмотрим

$$\text{dist} = \beta_0 + \beta_1 \times \text{speed}.$$



Найдены значения  $\hat{\beta}_0 = -17.579$ ,  $\hat{\beta}_1 = 3.932$ .



Синий график: регрессия в виде  $\text{dist} = \beta_0 + \beta_1 \times \text{speed}$ .

Красный график: регрессия в виде  $\text{dist} = \beta_0 + \beta_1 \times \text{speed} + \beta_2 \times \text{speed}^2$ .

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	-17.5791	6.7584	-2.601	0.0123 *
speed	3.9324	0.4155	9.464	1.49e-12 ***
---				
Signif. codes:	0 ‘***’	0.001 ‘**’	0.01 ‘*’	0.05 ‘.’
	0.1 ‘ ’	1		

Residual standard error: 15.38 on 48 degrees of freedom

Multiple R-squared: 0.6511, Adjusted R-squared: 0.6438

F-statistic: 89.57 on 1 and 48 DF, \pvalue: 1.49e-12

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	2.47014	14.81716	0.167	0.868
speed	0.91329	2.03422	0.449	0.656
I(speed^2)	0.09996	0.06597	1.515	0.136

Residual standard error: 15.18 on 47 degrees of freedom

Multiple R-squared: 0.6673, Adjusted R-squared: 0.6532

F-statistic: 47.14 on 2 and 47 DF, \pvalue: 5.852e-12

Для  $\text{dist} = \beta_0 + \beta_1 \times \text{speed}$

Коэффициент детерминации и подправленный коэффициент детерминации:

$$r^2 = 0.6511, \quad r_a^2 = 0.6438.$$

Таким образом, регрессия объясняет около 65% изменчивости данных.

Стандартные ошибки и стандартные коэффициенты:

$$\text{se } \beta_0 = 6.7584, \quad \text{se } \beta_1 = 0.4155, \quad t_0 = -2.601, \quad t_1 = 9.464.$$

Задача имеет  $N - d - 1 = 48$  степеней свободы.

Сравнивая значения  $t_0$  и  $t_1$  с квантилями  $t$ -распределения Стьюдента с 48 степенями свободы, получаем, что для гипотезы  $\beta_0 = 0$  значение p-value меньше 0.0123, а для гипотезы  $\beta_1 = 0$  оно равно  $1.49 \times 10^{-12}$ .

Если считать, что уровень надежности равен, например,  $\alpha = 0.01$ , то гипотезу  $\beta_0 = 0$  принимаем, гипотезу  $\beta_1 = 0$  отвергаем.

Остаточная стандартная ошибка равна  $\hat{\sigma} = 15.38$ .

Значение  $F$ -статистики  $F = 89.57$  сравниваем с квантилями  $F$ -распределения  $F_{1,48}$ .

Получаем p-value, равное  $1.490 \times 10^{-12}$  — для указанного уровня надежности модель статистически значима.

Для  $\text{dist} = \beta_0 + \beta_1 \times \text{speed} + \beta_2 \times \text{speed}^2$ :

Найдены значения  $\hat{\beta}_0 = 2.47014$ ,  $\hat{\beta}_1 = 0.91329$ ,  $\hat{\beta}_2 = 0.09996$ .

Имеем

$$r^2 = 0.6673, \quad r_a^2 = 0.6532,$$

$$\text{se } \beta_0 = 14.81716, \quad \text{se } \beta_1 = 2.03422, \quad \text{se } \beta_2 = 0.06597,$$

$$t_0 = 0.167, \quad t_1 = 0.449, \quad t_2 = 1.515.$$

Задача имеет  $N - d - 1 = 47$  степеней свободы.

Значение p-value для гипотезы  $\beta_0 = 0$  меньше 0.868.

Для гипотезы  $\beta_1 = 0$  оно равно 0.656,

а для гипотезы  $\beta_2 = 0$  равно 0.136.

Таким образом, при уровне надежности  $\alpha = 0.01$  гипотезу  $\beta_2 = 0$  принимаем.

Коэффициент  $\beta_2$  не является статистически значимым.

Остаточная стандартная ошибка равна  $\hat{\sigma} = 15.18$ .

Значение F-статистики  $F = 47.14$  нужно сравнить с квантилями F-распределения Фишера  $F_{2,47}$ .

Это сравнение приводит к p-value, равному  $5.852 \times 10^{-12}$ .

Таким образом, для указанного уровня надежности модель статистически значима.

Сравниваем обе модели (1-я квадратичная; 2-я линейная)

$$F = \frac{(\text{RSS}_2 - \text{RSS}_1)/(d_1 - d_2)}{\text{RSS}_1/(N - d_1 - 1)} = \frac{(11353.52 - 10824.72)/(2 - 1)}{10824.72/(50 - 2 - 1)} = 2.296027$$

Заметим, что  $\sqrt{F} = 1.515265$  совпадает с t-статистикой для  $\beta_2$

Вычисляем p-value = 0.13640241 —  $\beta_2$  незначим.

### 3.2.2. Пример. Boston

Все данные:

$$N = 506, d = 13 \text{ RSS} = 11078.78 \text{ RSS}/N = 21.90$$

Residuals:

	Min	1Q	Median	3Q	Max
	-15.595	-2.730	-0.518	1.777	26.199

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	3.646e+01	5.103e+00	7.144	3.28e-12 ***
CRIM	-1.080e-01	3.286e-02	-3.287	0.001087 **
ZN	4.642e-02	1.373e-02	3.382	0.000778 ***
INDUS	2.056e-02	6.150e-02	0.334	0.738288
CHAS	2.687e+00	8.616e-01	3.118	0.001925 **
NOX	-1.777e+01	3.820e+00	-4.651	4.25e-06 ***
RM	3.810e+00	4.179e-01	9.116	< 2e-16 ***
AGE	6.922e-04	1.321e-02	0.052	0.958229
DIS	-1.476e+00	1.995e-01	-7.398	6.01e-13 ***
RAD	3.060e-01	6.635e-02	4.613	5.07e-06 ***
TAX	-1.233e-02	3.760e-03	-3.280	0.001112 **
PTRAT	-9.527e-01	1.308e-01	-7.283	1.31e-12 ***
B	9.312e-03	2.686e-03	3.467	0.000573 ***

LSTAT -5.248e-01 5.072e-02 -10.347 < 2e-16 \*\*\*  
---  
Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.745 on 492 degrees of freedom  
Multiple R-squared: 0.7406, Adjusted R-squared: 0.7338  
F-statistic: 108.1 on 13 and 492 DF, \pvalue: < 2.2e-16

Выделим обучающую ( $N = 405$ ) и тестовую ( $N_{\text{test}} = 101$ ) выборки  
на обучающей  $\text{RSS} = 8655.383$ ,  $\text{RSS}/N = 21.37132$   
на тестовой  $\text{RSS}_{\text{test}} = 2522.615$ ,  $R \approx \text{RSS}_{\text{test}}/N_{\text{test}} = 24.97638$ ,  $\text{se } R = 0.4997638$ .

$$(\text{se } R)^2 = \frac{1}{N_{\text{test}}} \text{Var}(Y - \hat{Y}) = \frac{1}{N_{\text{test}}} \left( \frac{1}{N_{\text{test}} - 1} \sum_{i=1}^{N_{\text{test}}} (y^{(i)} - \hat{y}^{(i)})^2 \right)$$

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )
(Intercept)	32.353120	5.775840	5.601	4.01e-08 ***
CRIM	-0.116261	0.034297	-3.390	0.00077 ***
ZN	0.049895	0.015084	3.308	0.00103 **
INDUS	0.032491	0.073167	0.444	0.65724
CHAS	2.054215	1.059339	1.939	0.05320 .
NOX	-17.501306	4.194206	-4.173	3.71e-05 ***
RM	4.261255	0.500527	8.514	3.67e-16 ***
AGE	-0.003699	0.015249	-0.243	0.80845
DIS	-1.389752	0.216254	-6.426	3.80e-10 ***
RAD	0.310786	0.072337	4.296	2.19e-05 ***
TAX	-0.013453	0.004167	-3.229	0.00135 **
PTRATIO	-0.913321	0.148451	-6.152	1.89e-09 ***
B	0.008908	0.002929	3.042	0.00251 **
LSTAT	-0.475633	0.061656	-7.714	1.02e-13 ***

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.705 on 391 degrees of freedom

Multiple R-squared: 0.7432, Adjusted R-squared: 0.7346

F-statistic: 87.03 on 13 and 391 DF, \pvalue: < 2.2e-16

Возьмем только значимые (\*\*\*) переменные: CRIM + NOX + RM + DIS + RAD + PTRATIO + LSTAT

На обучающей выборке  $\text{RSS} = 9437.129$ ,  $\text{RSS}/N = 23.30155$

На тестовой выборке  $\text{RSS} = 2798.908$ ,  $\text{RSS}/N = 27.71196$  se = 0.5264215

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )	
(Intercept)	35.55914	5.71594	6.221	1.25e-09	***
CRIM	-0.11438	0.03527	-3.243	0.00128	**
NOX	-20.55633	3.86533	-5.318	1.75e-07	***
RM	4.59477	0.48776	9.420	< 2e-16	***
DIS	-1.06844	0.17807	-6.000	4.44e-09	***
RAD	0.11284	0.04673	2.415	0.01619	*
PTRATIO	-1.13134	0.14049	-8.053	9.54e-15	***
LSTAT	-0.51548	0.05926	-8.699	< 2e-16	***
---					
Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1					

Residual standard error: 4.876 on 397 degrees of freedom

Multiple R-squared: 0.72, Adjusted R-squared: 0.715

F-statistic: 145.8 on 7 and 397 DF, \pvalue: < 2.2e-16

$$F = \frac{(\text{RSS}_2 - \text{RSS}_1)/(d_1 - d_2)}{\text{RSS}_1/(N - d_1 - 1)} = \frac{(9437.129 - 8655.383)/(13 - 7)}{8655.383/(405 - 13 - 1)} = 5.885792$$

Распределение  $F(d_1 - d_2, N - d_1 - 1) = F(6, 391)$ .

Находим p-value:  $6.783518e - 06$ , т. е. гипотеза о незначимости 6 переменных не принимается (для разумного уровня значимости  $\alpha$ )

### **3.2.3. Анализ остатков**

Ранее мы сделали предположение, что остатки должны быть независимы (некоррелированы) и иметь нормальное распределение.

Многие рассмотренные выше тесты достаточно устойчивы по отношению к отклонениям от таких предположений. Однако перед тем, как исследовать статистическую значимость модели и строить доверительные интервалы, рекомендуется провести анализ остатков.

«Визуальные» способы:

- Гистограмма остатков — должна быть близка к нормальному распределению;
- график остатков — без зависимостей и нелинейностей;
- $QQ$ -график (график «квантиль–квантиль»)

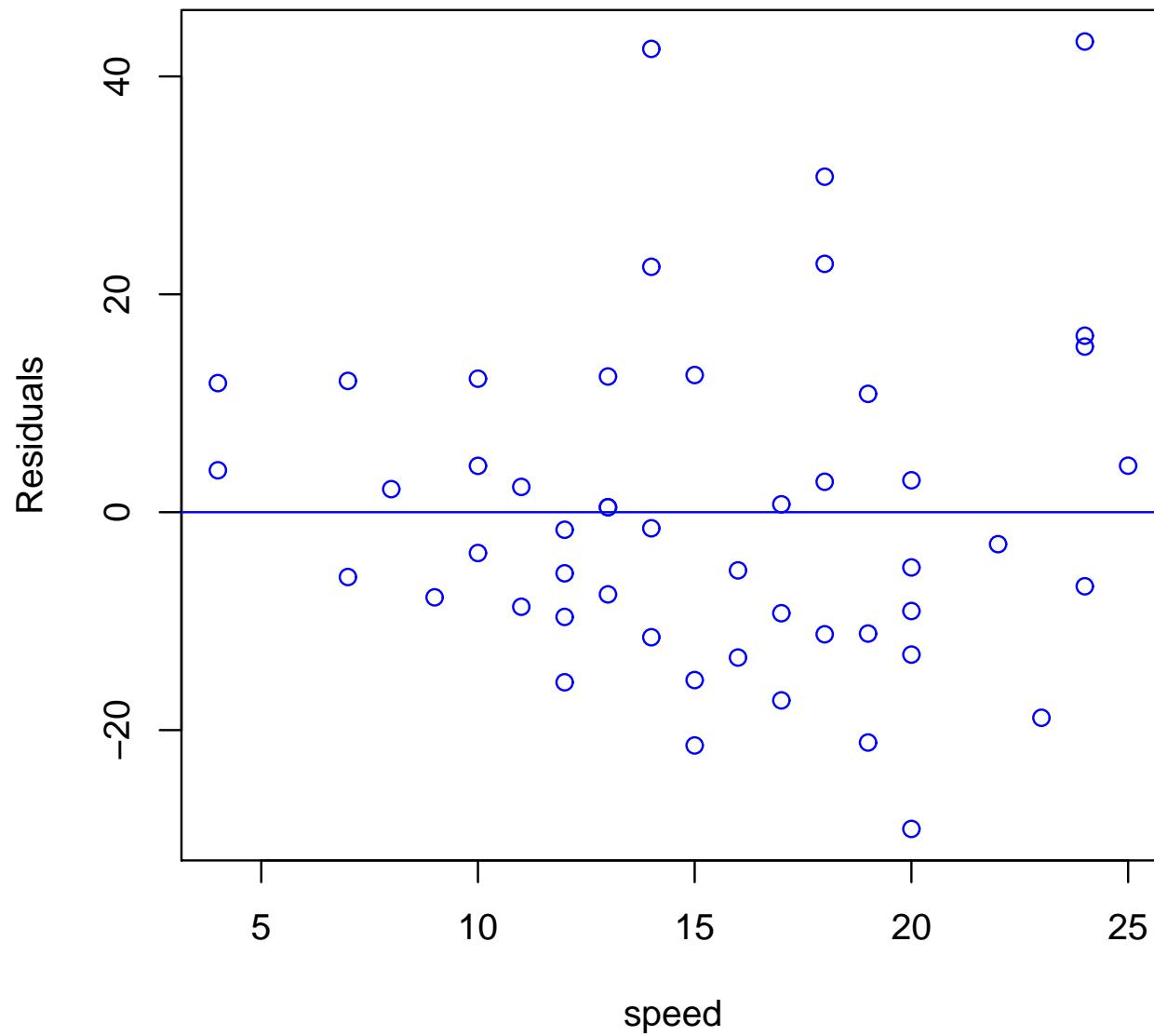
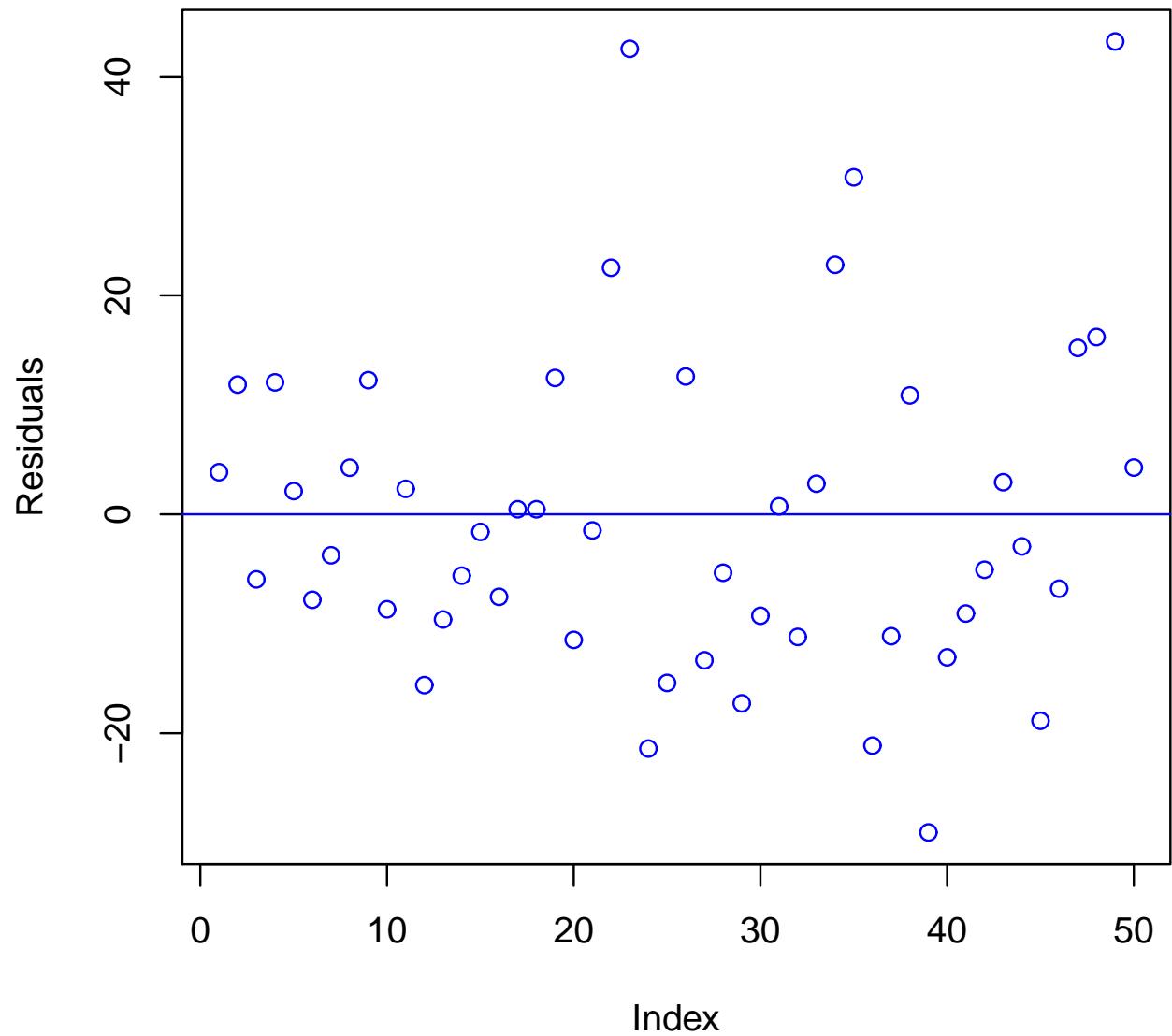
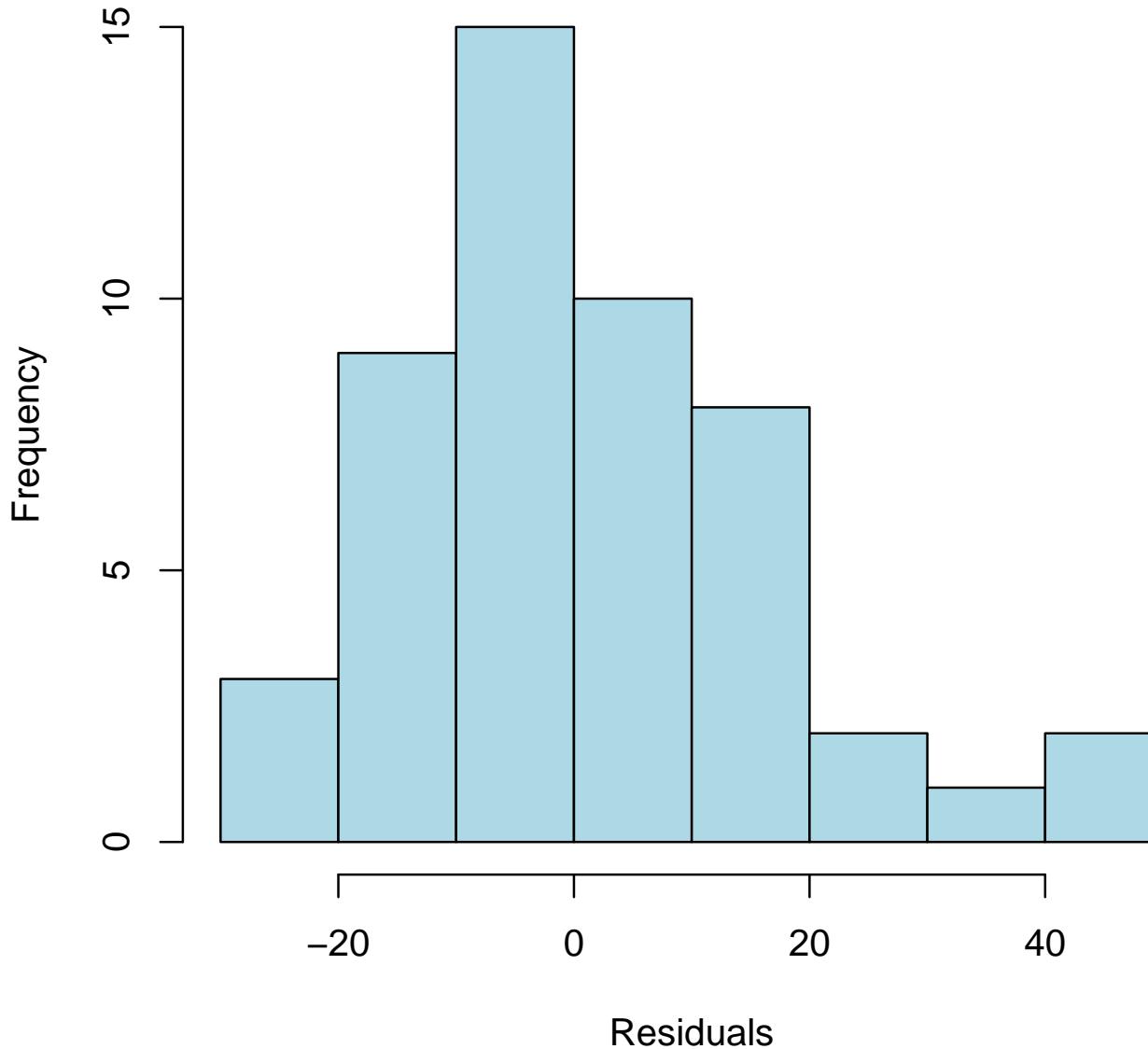
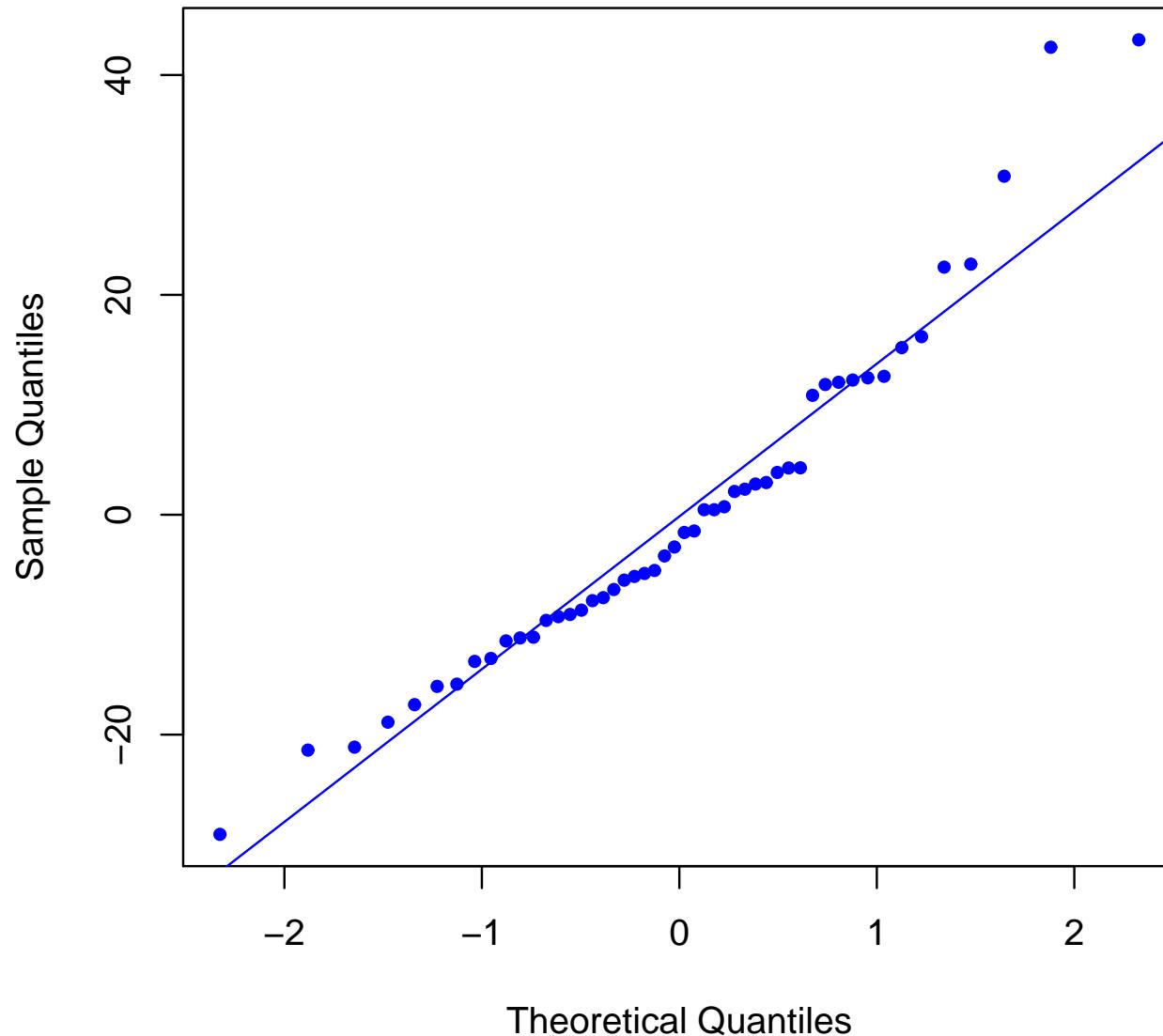


График остатков в задаче восстановления зависимости длины тормозного пути от начальной скорости.





Гистограмма остатков в задаче восстановления зависимости вида  $\text{dist} = \beta_0 + \beta_1 \times \text{speed}$  длины тормозного пути от начальной скорости.



QQ-график в задаче восстановления зависимости длины тормозного пути от начальной скорости.

Теоретическая проверка нормальности остатков:

- общие критерии согласия:  $\chi^2$ , критерий Колмогорова–Смирнова и т. п.
- специализированные, например, *критерий Шапиро–Уилка*

Нуль-гипотеза в критерии Шапиро–Уилка заключается в том, что статистически наблюдаемая случайная величина имеет нормальное распределение.

В задаче с определением длины тормозного пути автомобиля статистика Шапиро–Уилка оказалось равной  $W = 0.9451$ .

Соответствующее  $p$ -value равно 0.02153. При уровне значимости  $\alpha = 0.01$  гипотезу о нормальности распределения остатков принимаем.

## Критерий Уайта

Проверка на гомоскедастичность (постоянство дисперсии) остатков.

Гипотеза  $H_0: \sigma_1 = \sigma_2 = \dots = \sigma_N$

Гипотеза  $H_1: H_0$  не выполнено

К исходной модели применяется метод наименьших квадратов и находятся остатки  $e_1, e_2, \dots, e_N$ .

Затем строится модель, описывающая квадраты этих остатков через исходные переменные, их квадраты и попарные произведения:

$$e^2 = \gamma_0 + \sum_{j=1}^d \gamma_j x_j + \sum_{j \leq j'} \gamma_{jj'} x_j x_{j'}.$$

При гипотезе  $H_0$  величина  $nr^2$  асимптотически имеет распределение  $\chi^2(d' - 1)$ , где

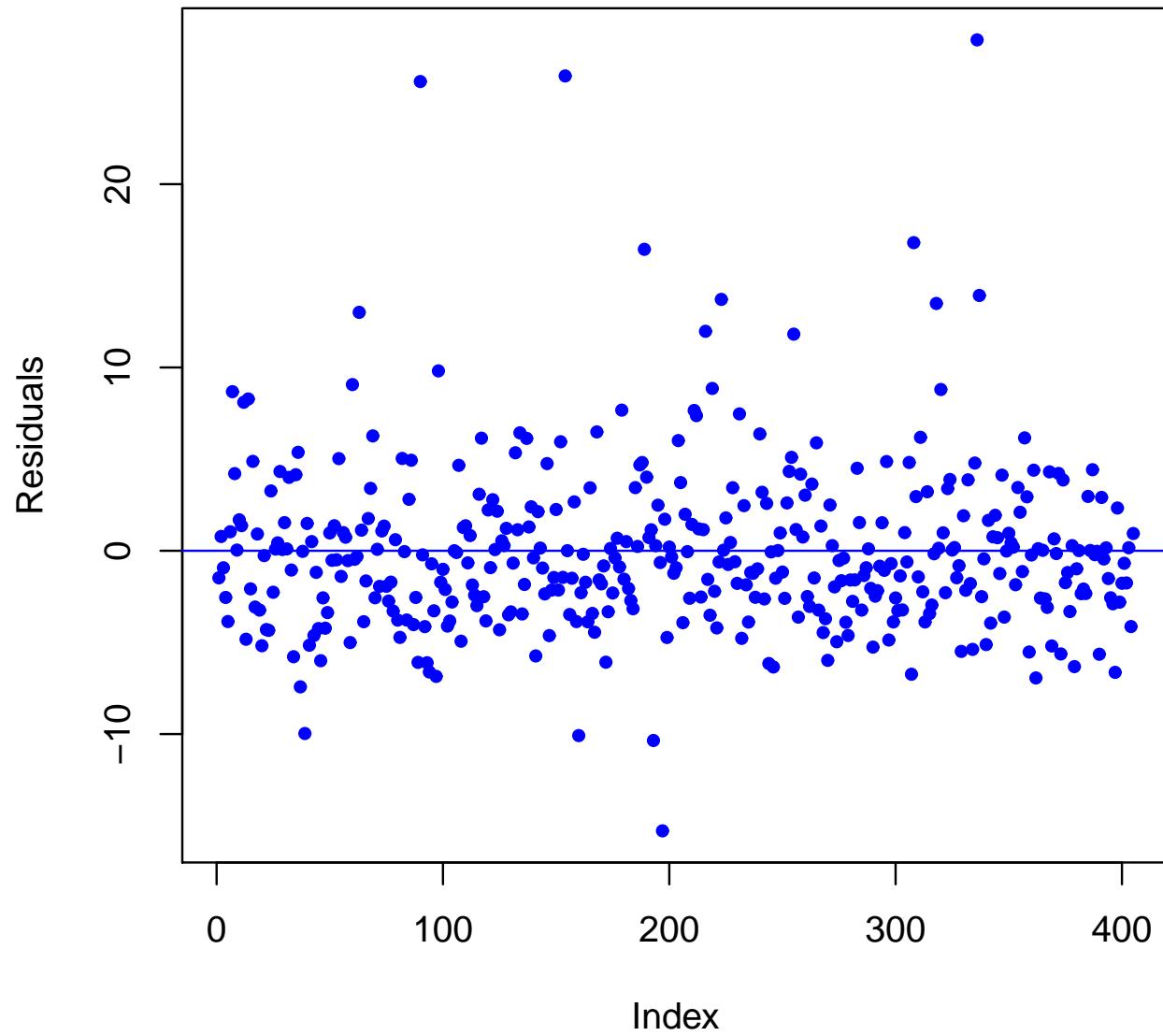
$d' = 1 + d + \frac{d(d+1)}{2}$  — число переменных новой модели,

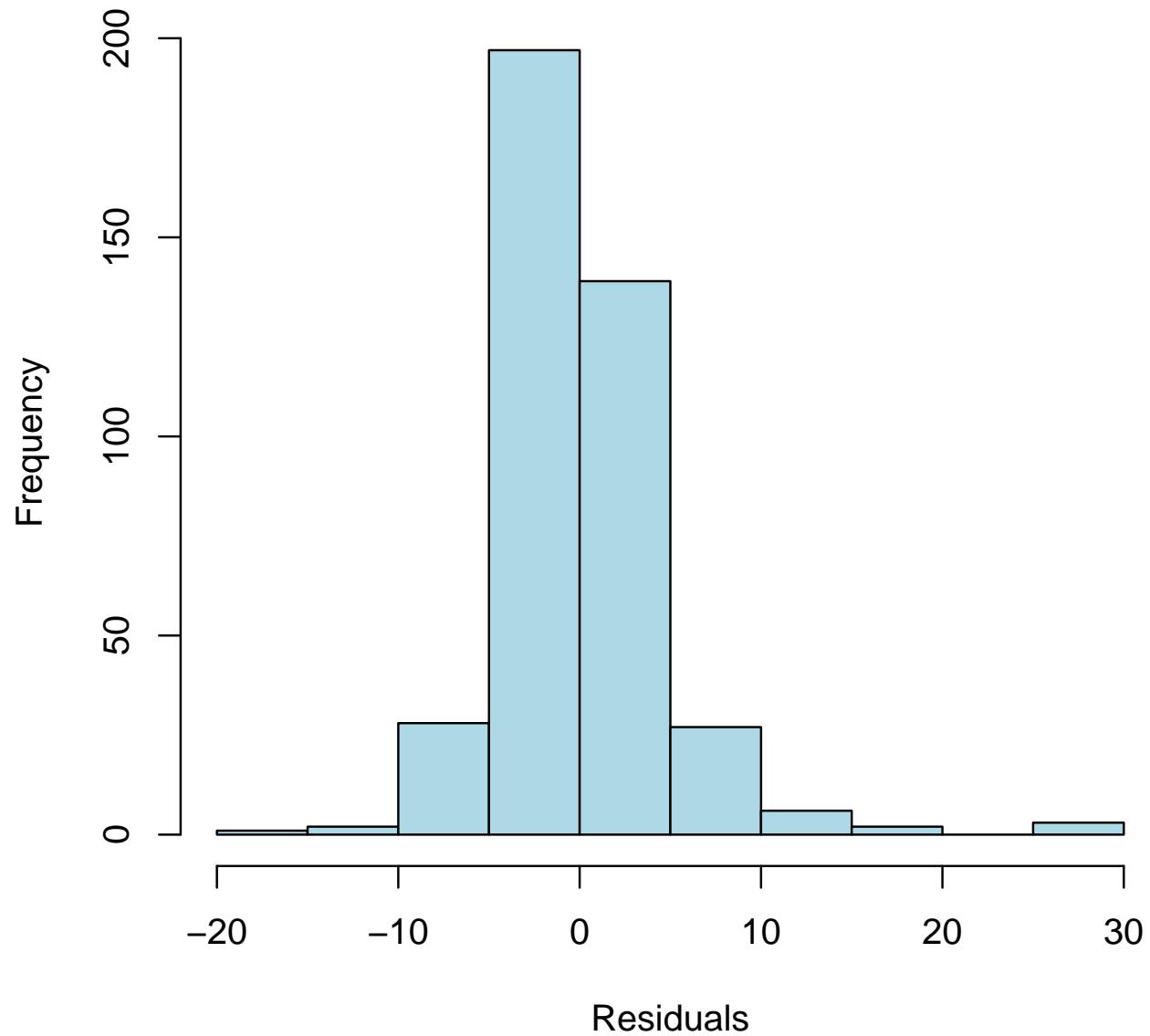
$r^2$  — коэффициент детерминации новой модели.

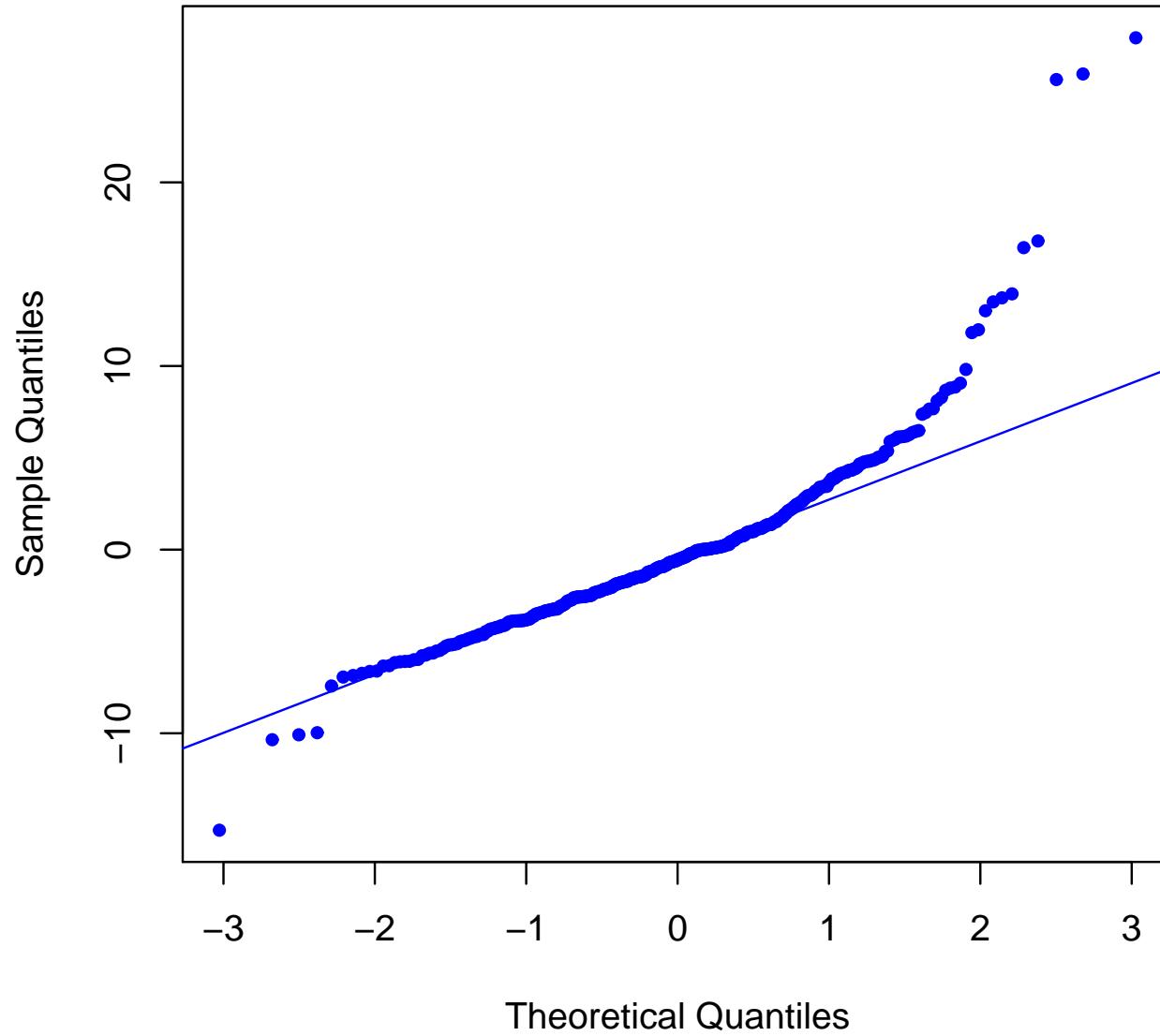
Анализ остатков для линейной модели  $\text{dist} = \beta_0 + \beta_1 \times \text{speed}$ :

```
bptest(dist ~ speed, data=cars) # BP = 3.2149, df = 1, \pvalue = 0.07297  
White = 3.2157, df = 2, \pvalue = 0.2003
```

Boston





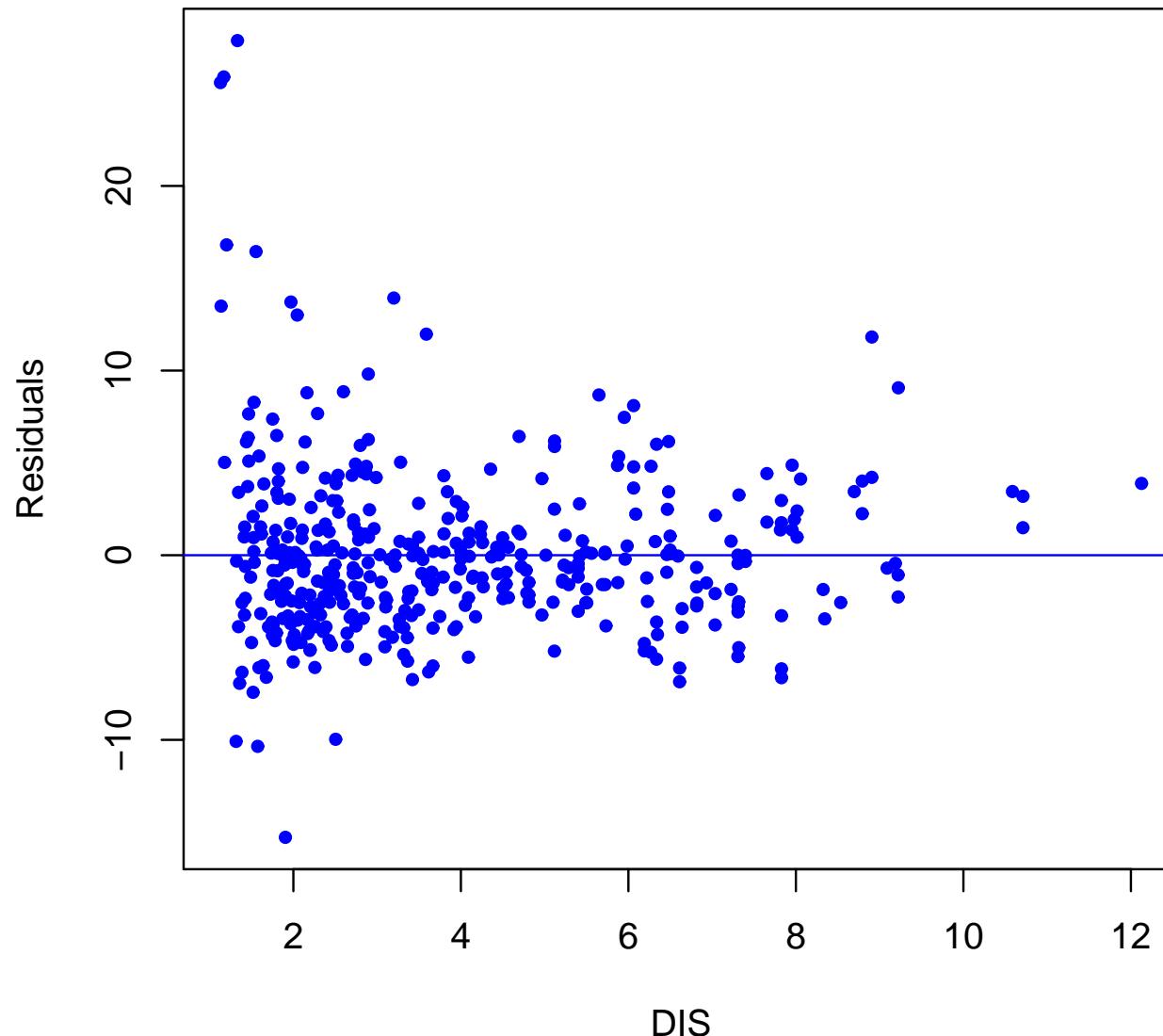


Критерий Шапиро-Уилка  $W = 0.8739$ ,  $p\text{-value} < 2.2\text{e-}16$  Отвергаем

Тест на гомоскедастичность

Тест Бройша–Пагана = 51.0623, df = 13, p-value= 1.958e-06

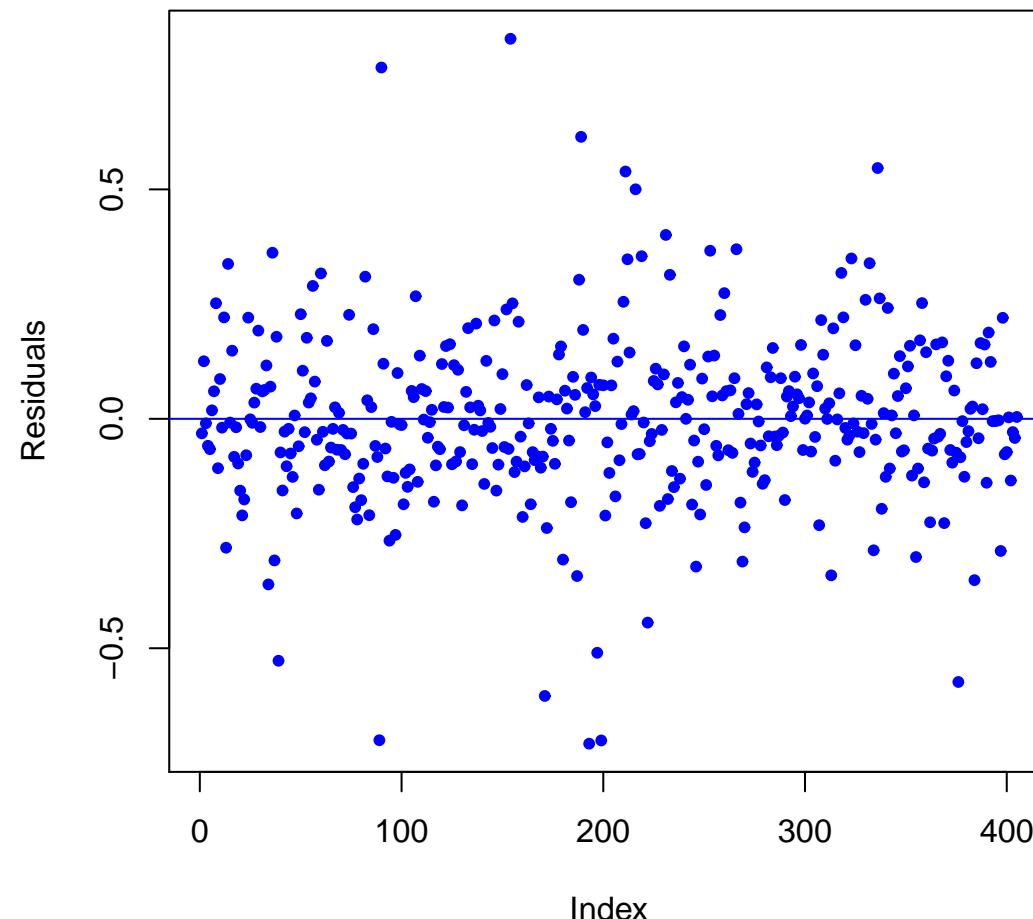
Критерий Уайта: White-test = 285.8917, df = 103, p-value< 2.2e-16



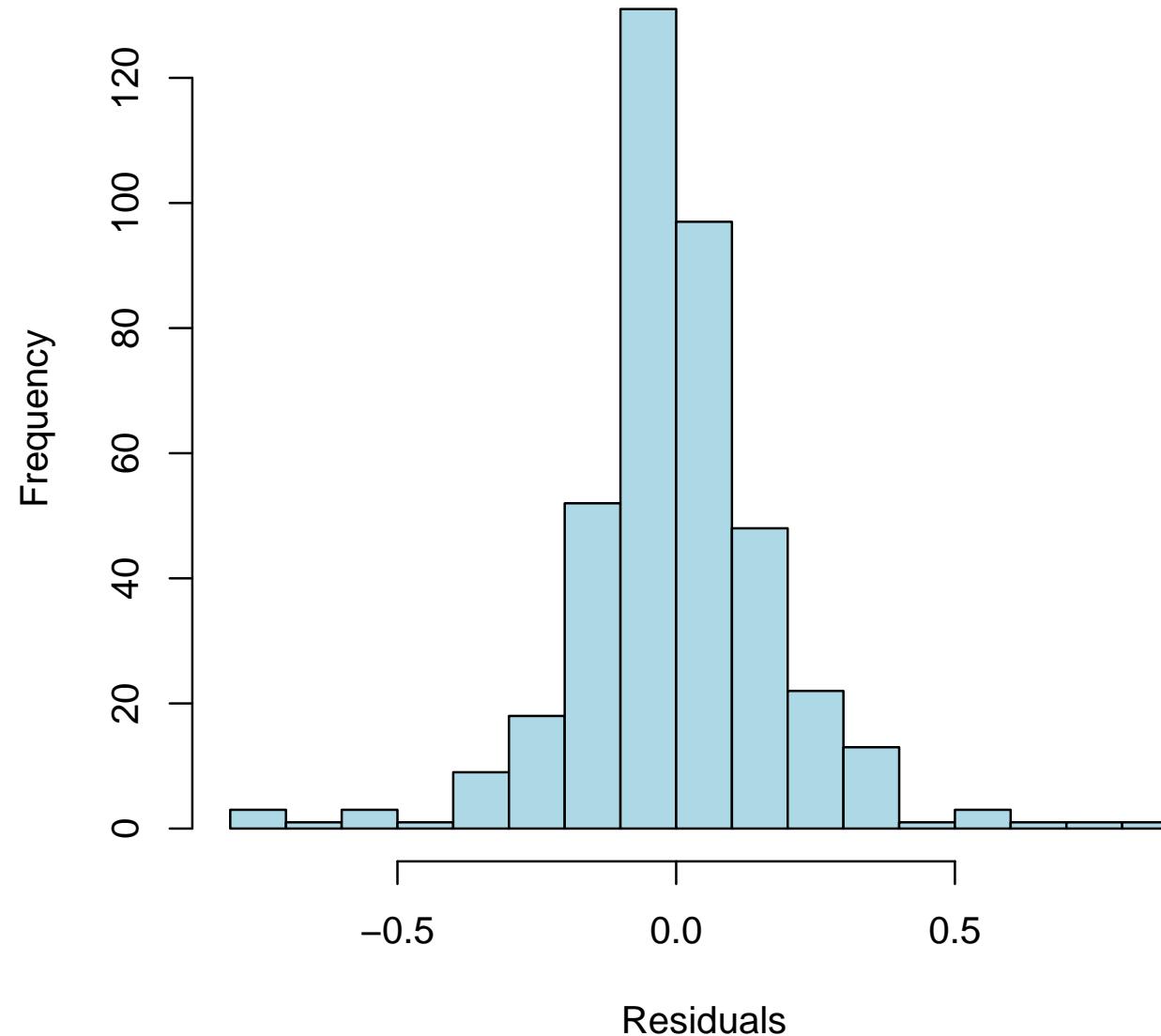
В оригинальной статье [Harrison and Rubinfeld, 1978] рассматривается модель

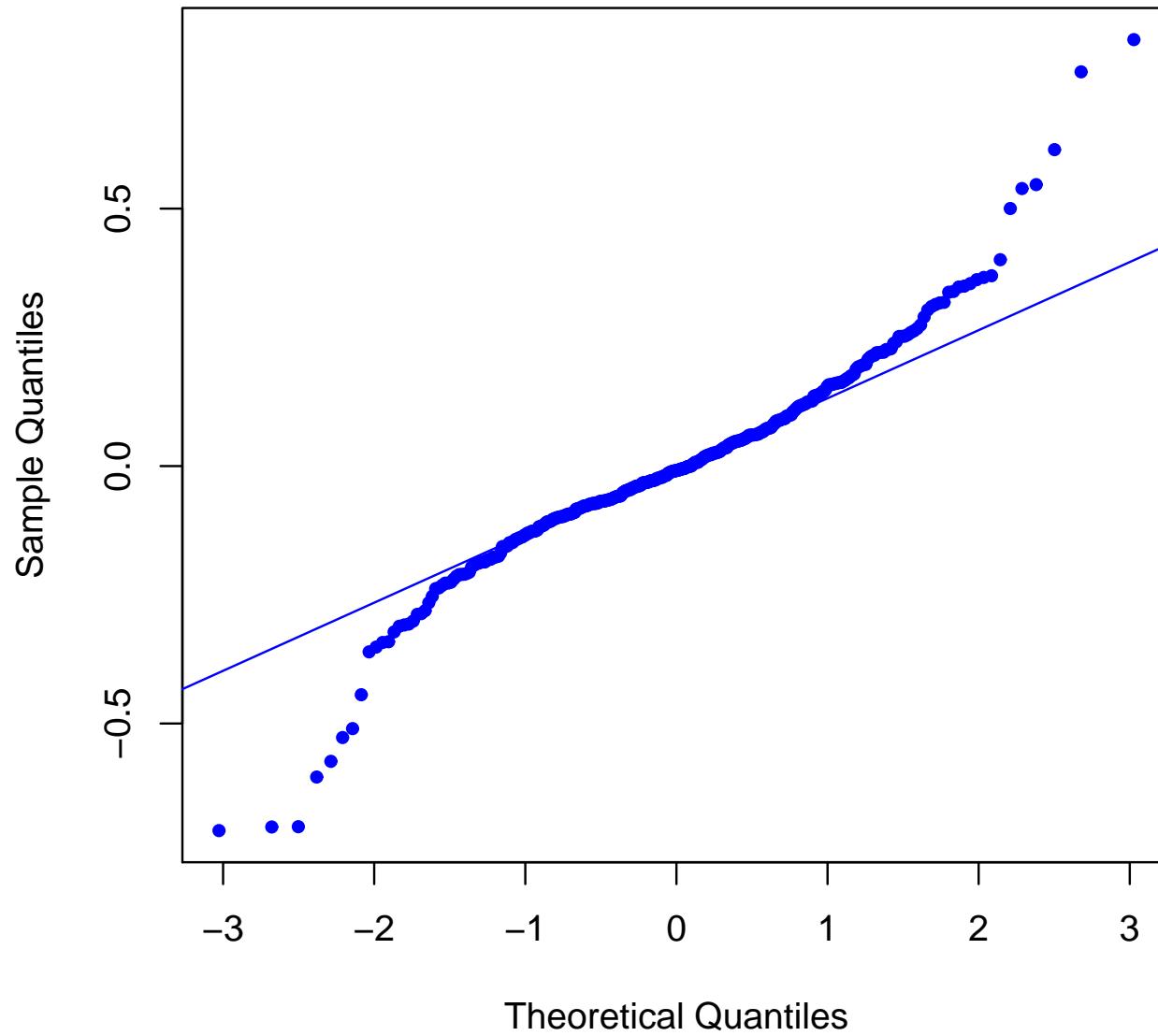
$$\begin{aligned}\ln \text{MEDV} = & \beta_0 + \beta_1 \text{CRIM} + \beta_2 \text{ZN} + \beta_3 \text{INDUS} + \beta_4 \text{CHAS} + \beta_5 \text{NOX}^2 + \beta_6 \text{RM}^2 + \beta_7 \text{AGE} + \beta_8 \ln \text{DIS} + \\ & + \beta_9 \ln \text{RAD} + \beta_{10} \text{TAX} + \beta_{11} \text{PTRAT} + \beta_{12} \text{B} + \beta_{13} \ln \text{LSTAT} + E.\end{aligned}$$

$$\text{RSS } / N_{train} = 16.05832, \quad \text{RSS } / N_{test} = 14.77086$$



Но также не проходит тест на нормальность. Shapiro-Wilk normality test  $W = 0.9439$ ,  
 $p\text{-value} = 2.989\text{e-}11$

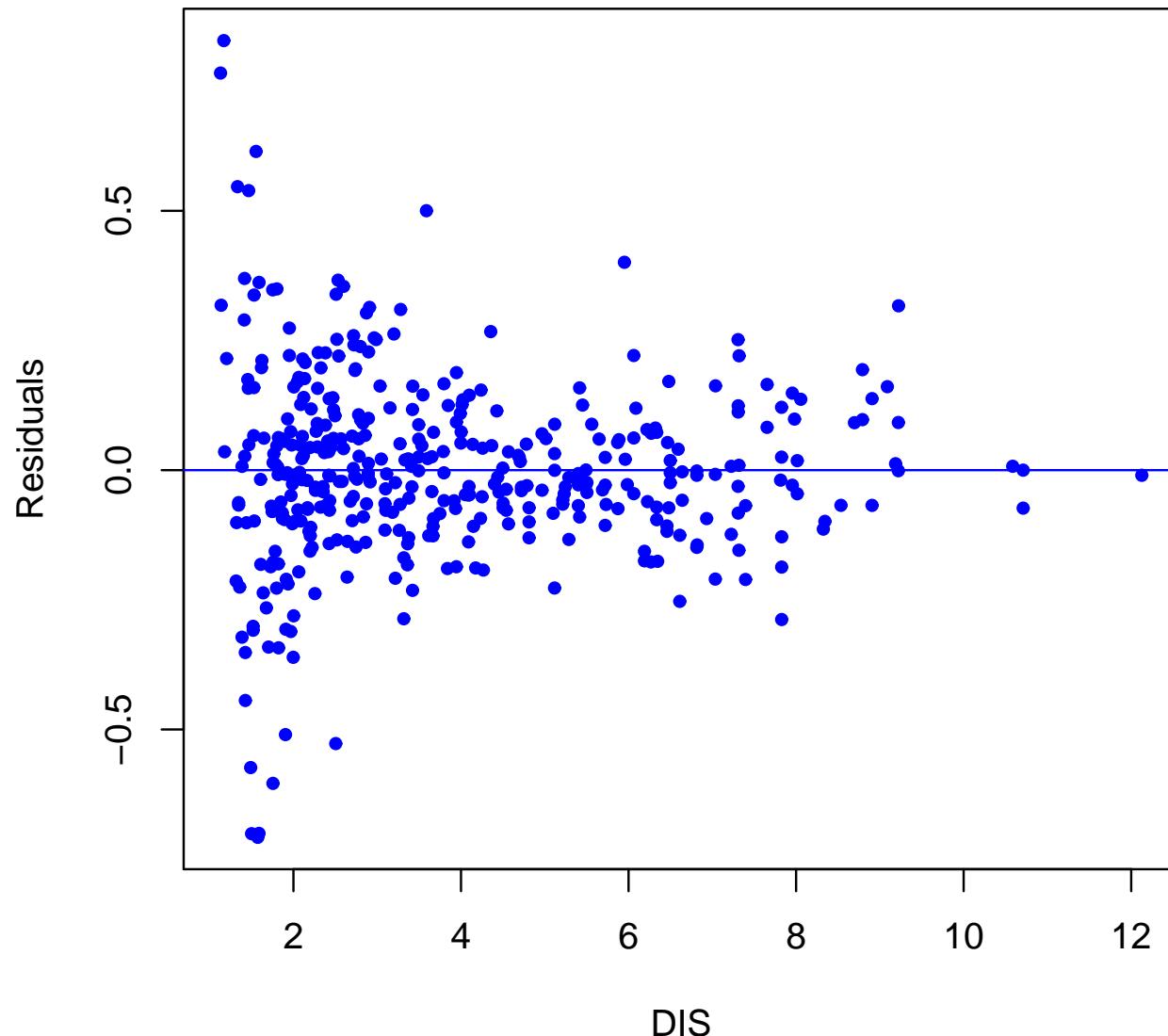




Тест на гомоскедастичность

Тест Брайша–Пагана = 76.1122, df = 13, p-value= 5.905e-11

Критерий Уайта =



Для выявления корреляции между остатками (точнее: сериальной корреляции) часто используется критерий Дарбина–Уотсона. Статистика Дарбина–Уотсона вычисляется по формуле

$$D = \frac{\sum_{i=1}^{N-1} (e^{(i+1)} - e^{(i)})^2}{\sum_{i=1}^N e^{(i)2}}.$$

**Упражнение 3.5** Доказать, что  $0 \leq D \leq 4$ .

Если  $D < D_L(\alpha)$  или  $D > 4 - D_L(\alpha)$ , то с достоверностью  $\alpha$  принимается гипотеза о наличии отрицательной или соответственно положительной корреляции остатков.

Если  $D_L(\alpha) < D < D_U(\alpha)$  или  $4 - D_U(\alpha) < D < 4 - D_L(\alpha)$ , то критерий не позволяет принять решение о наличии или отсутствии корреляции остатков.

Если  $D_U(\alpha) < D < 4 - D_U(\alpha)$ , то гипотеза о наличии корреляции остатков отклоняется.

Таблицы критических значений  $D_L(\alpha)$  и  $D_U(\alpha)$  для различных  $\alpha$ ,  $N$ ,  $d$  приведены, например, в [Дрейпер, Смит Т. 1, С. 211].

### 3.3. Переобучение

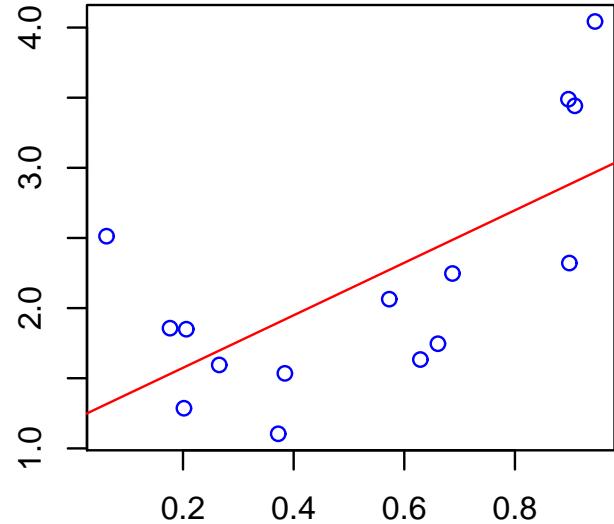
*Переобучение* — это явление, при котором обучающий алгоритм выдает хорошие результаты на обучающей выборке, но имеет очень плохие обобщающие свойства.

Типичное поведение ошибки на обучающей и тестовой выборках с ростом «сложности» модели:

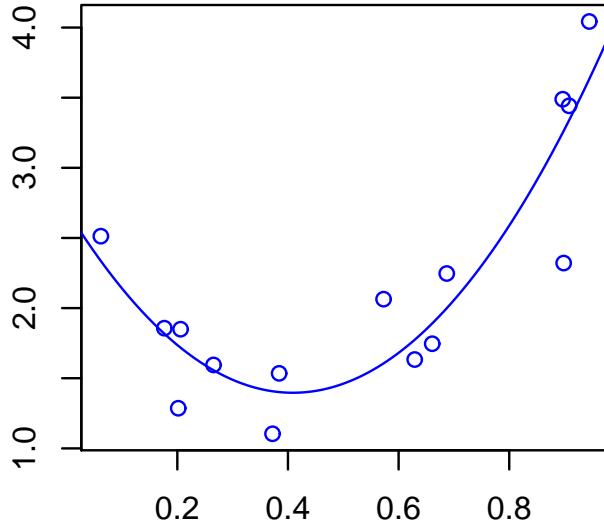
- Обычно на обучающей выборке с ростом сложности модели ошибка уменьшается.
- На тестовой выборке сперва с ростом сложности модели ошибка уменьшается, но с некоторого момента ошибка начинает расти: сказывается синдром переобучения.

Когда модель слишком сложна, она, как правило, хорошо приспосабливается к конкретным обучающим данным, улавливая какие-то специфичные для них особенности, но не присущие всей генеральной совокупности, поэтому на тестовых данных ошибка может быть большой.

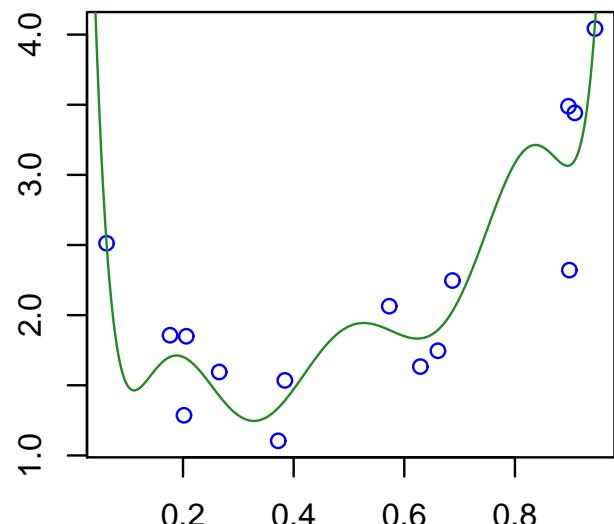
$$y = 8x^2 - 6.4x + 2.5 + N(0, 0.4)$$



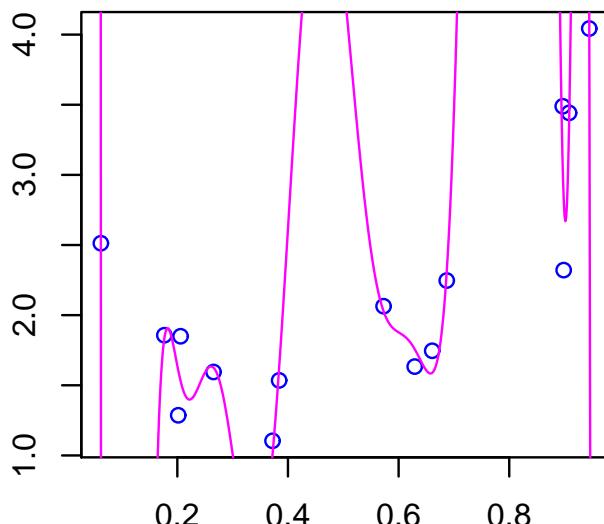
Degree = 1



Degree = 2

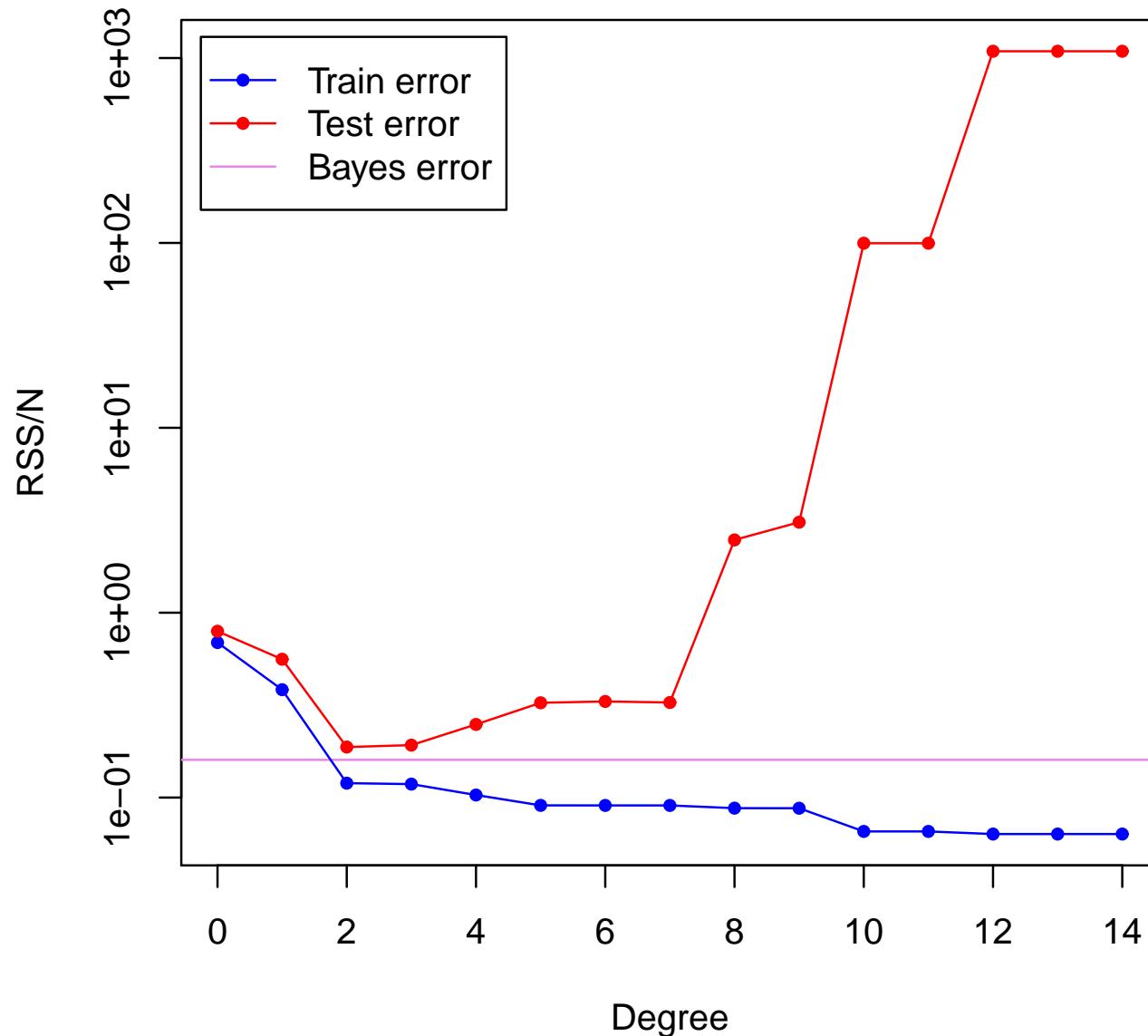


Degree = 8



Degree = 13

$\text{RSStrain}/\text{Ntrain} = 0.1196395$   $\text{RSStest}(3)/\text{Ntest} = 0.1876475$   $\sigma^2 = 0.16$



Риск



средний риск  $R$



эмпирический риск  $\hat{R}$

Емкость модели

### 3.3.1. Причины переобучения (в задаче восстановления регрессии)

- *Мультиколлинеарность*:  $\mathbf{X}$  плохо обусловлена (т. е. столбцы матрицы  $\mathbf{X}$  образуют систему, близкую к линейно зависимой).

$$\text{cond}_2 \mathbf{X} = \frac{\max \sqrt{\lambda_j(\mathbf{X}^\top \mathbf{X})}}{\min \sqrt{\lambda_j(\mathbf{X}^\top \mathbf{X})}} = \frac{\max \sigma_j}{\min \sigma_j}$$

(отношение максимального сингулярного числа матрицы  $\mathbf{X}$  к минимальному)

- *Наличие неинформативных признаков*
- *Слишком мало данных.*

Даже если все признаки информативны, данных может оказаться слишком мало.

## 3.4. Сокращение числа параметров и «усадка» коэффициентов

Рассмотрим некоторые методы по уменьшению числа признаков (уменьшения размерности пространства признаков) и «усадке» коэффициентов.

Зачем?

- Борьба с *переобучением*: чем меньше параметров (и входных переменных) или чем меньше по величине параметры, тем проще модель и снижается возможность переобучения
- *Интерпретация*: чем меньше параметров (и входных переменных), тем проще интерпретировать модель

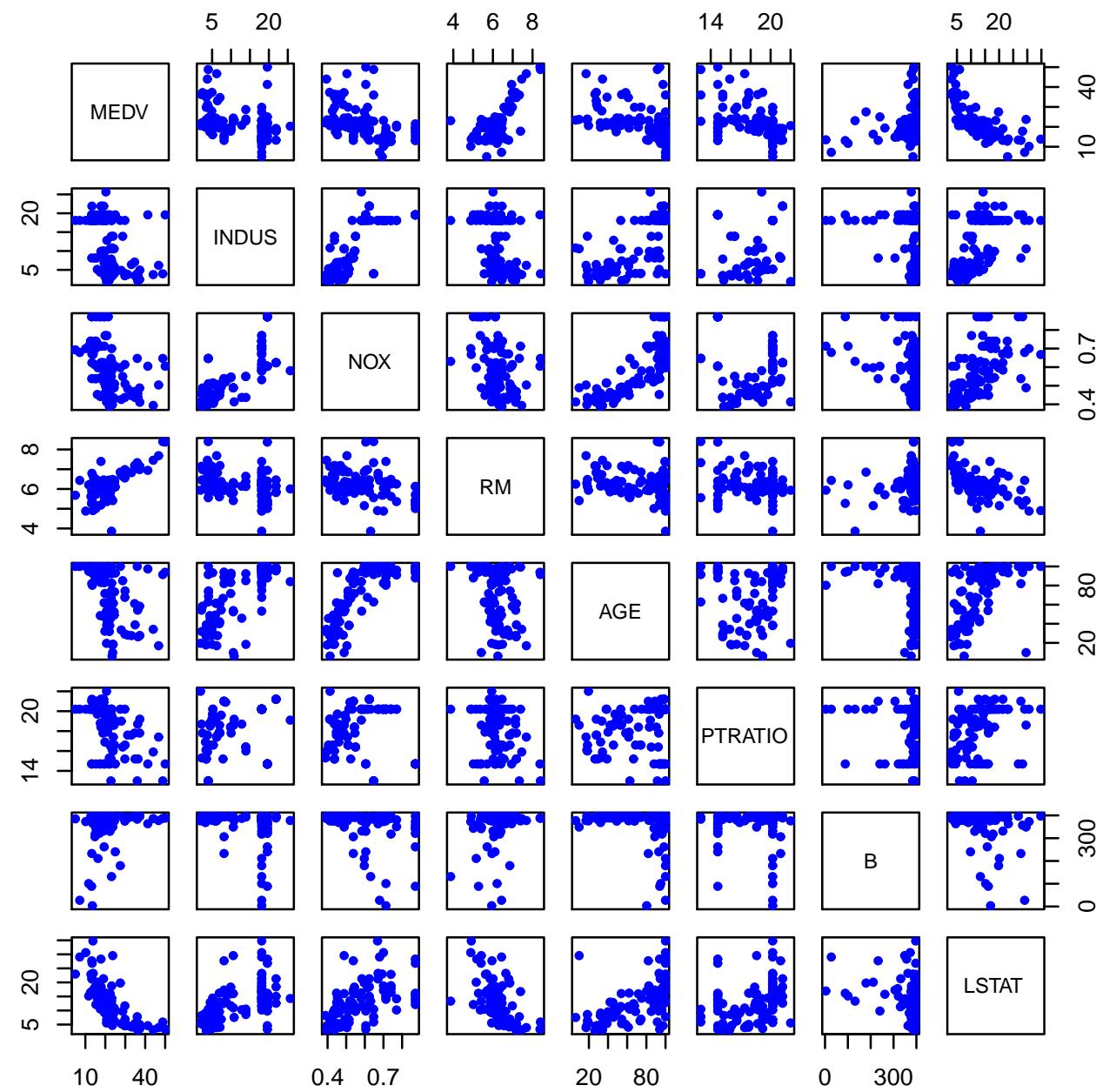
### **3.4.1. Выбор подмножества признаков**

Для каждого  $k \in \{0, 1, \dots, d\}$  найдем подмножество входных параметров мощности  $k$ , для которого  $\text{RSS}(\beta)$  минимально.

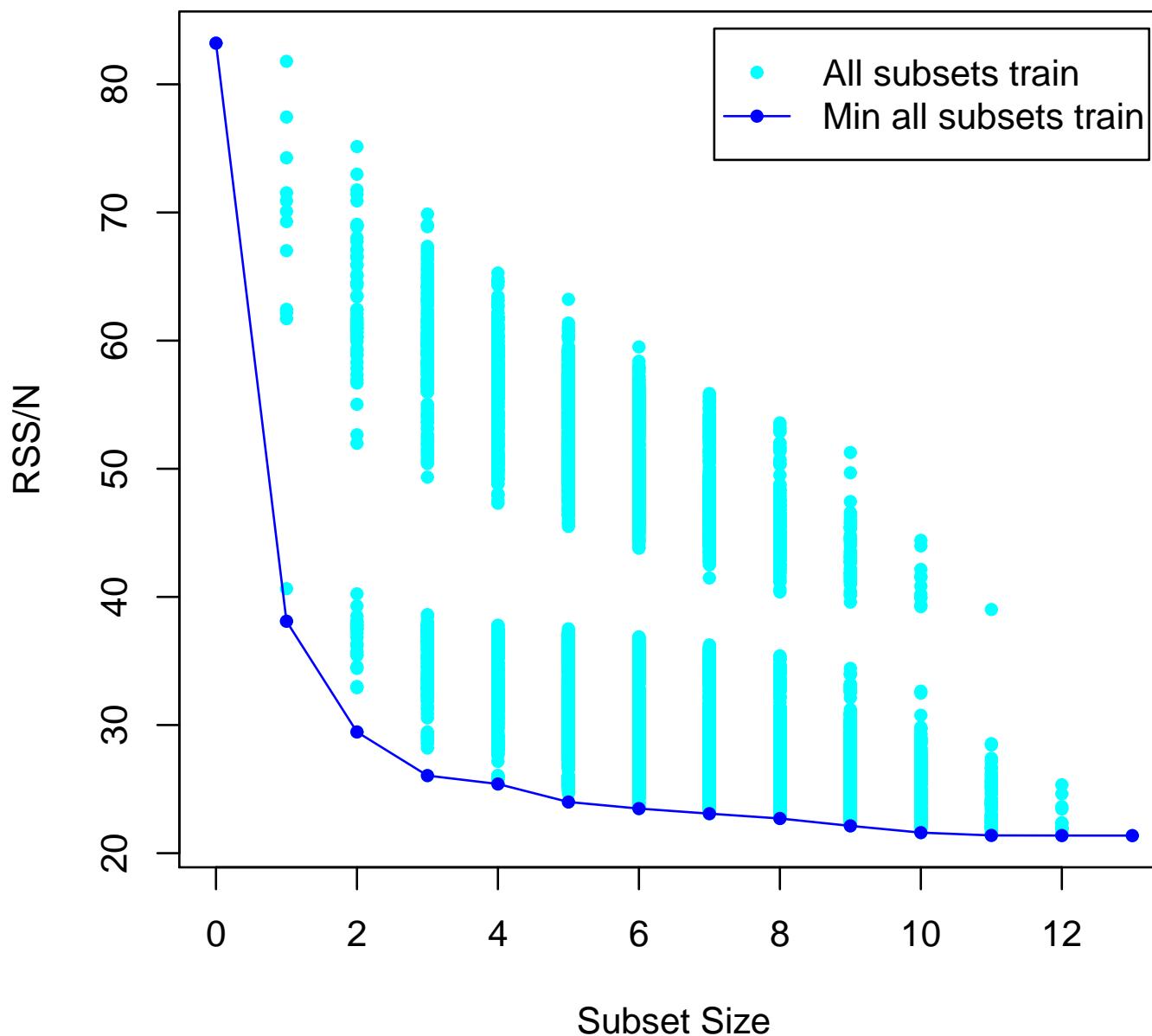
Большой перебор!

Оптимальное подмножество признаков мощности  $k$  не обязательно содержится в оптимальном подмножестве мощности  $k + 1$

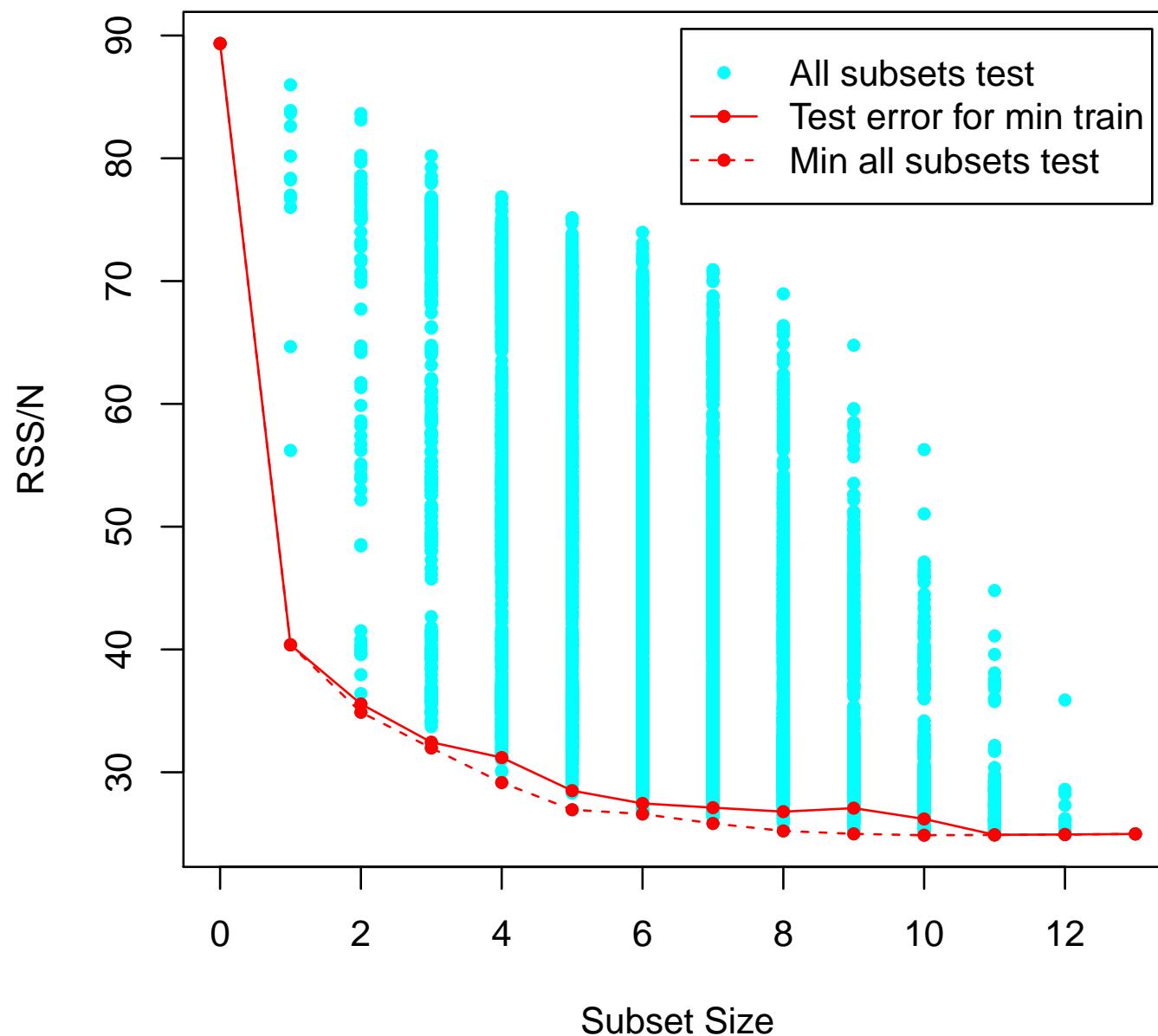
## Boston. 13 входных переменных



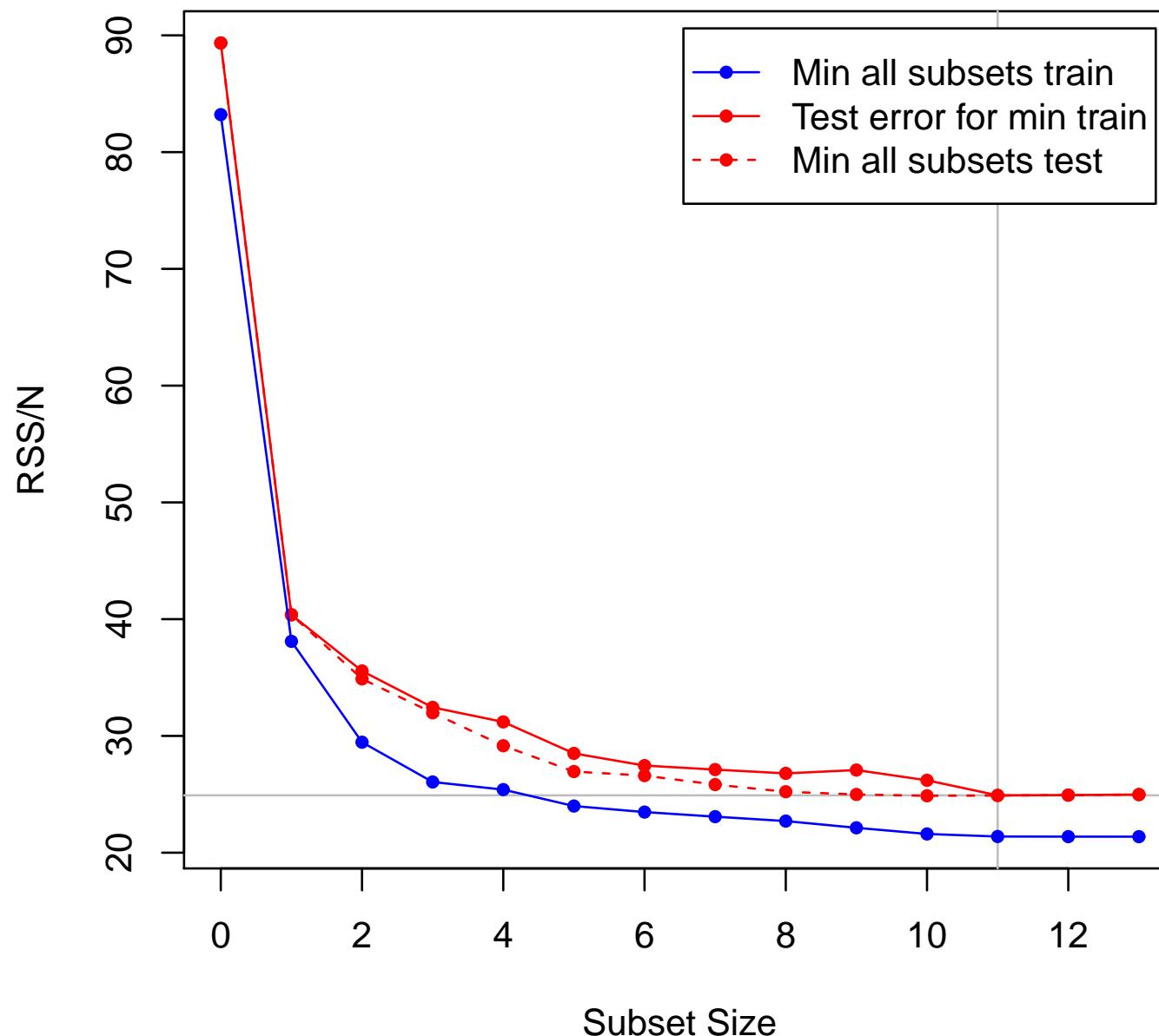
На обучающей выборке:



На тестовой выборке:

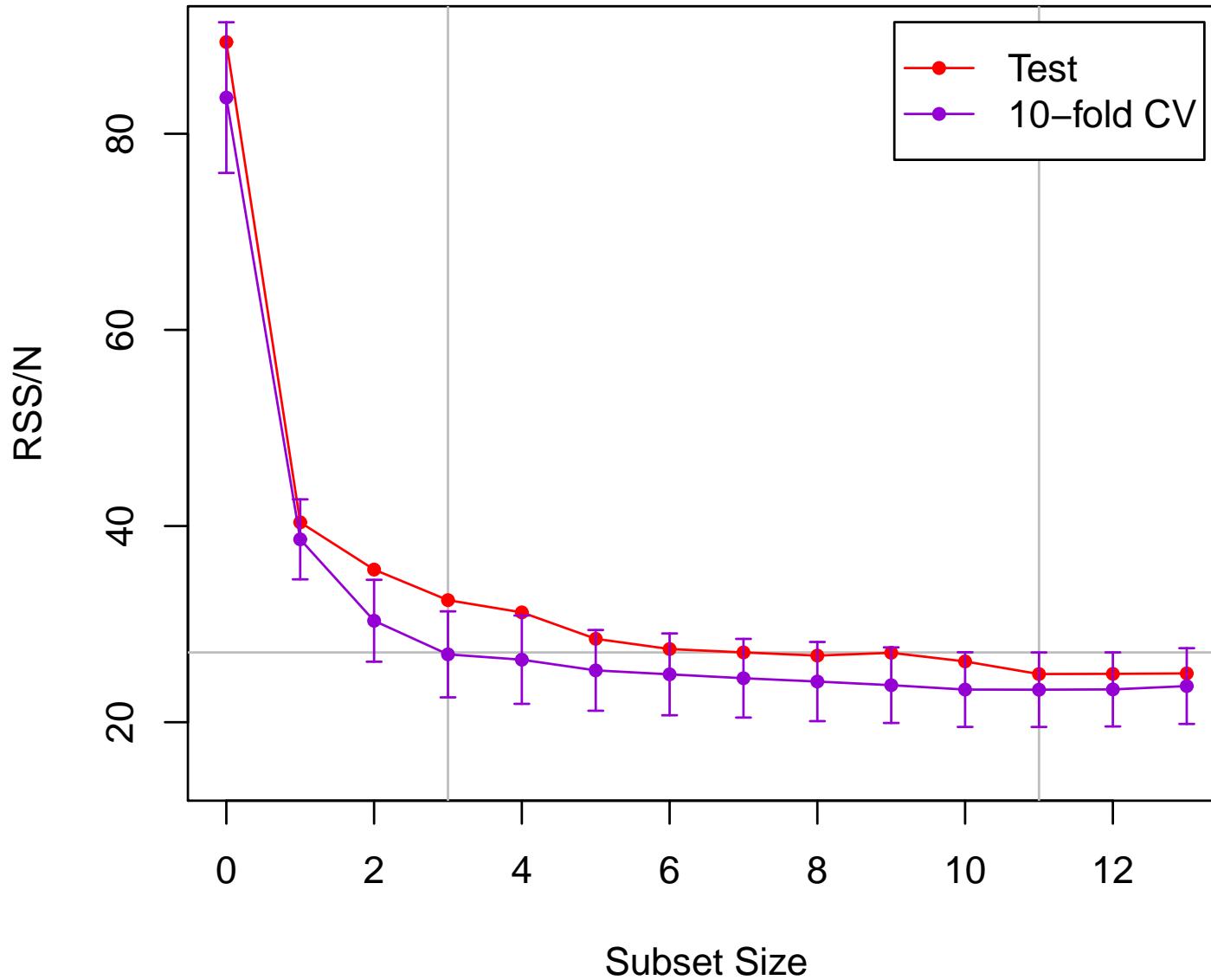


Оба графика testminmin = 24.87363(10) testmin = 24.90995(11) (se = 0.4990987), trainmin = 21.37132(13)



0 MEDV ~ NULL  
 1 MEDV ~ LSTAT  
 2 MEDV ~ RM\* + LSTAT  
 3 MEDV ~ RM + PTRAT\* + LSTAT  
 4 MEDV ~ RM + PTRAT + B\* + LSTAT  
 5 MEDV ~ NOX\* + RM + DIS\* + PTRAT + LSTAT  
 6 MEDV ~ NOX + RM + DIS + PTRAT + B\* + LSTAT  
 7 MEDV ~ ZN\* + NOX + RM + DIS + PTRAT + B + LSTAT  
 8 MEDV ~ CRIM\* + NOX + RM + DIS + RAD\* + PTRAT + B + LSTAT  
 9 MEDV ~ CRIM + ZN\* + NOX + RM + DIS + RAD + TAX\* + PTRAT + LSTAT  
 10 MEDV ~ CRIM + ZN + NOX + RM + DIS + RAD + TAX + PTRAT + B\* + LSTAT  
 11 MEDV ~ CRIM + ZN + CHAS\* + NOX + RM + DIS + RAD + TAX + PTRAT + B + LSTAT  
 12 MEDV ~ CRIM + ZN + INDUS\* + CHAS + NOX + RM + DIS + RAD + TAX + PTRAT + B + LSTAT  
 13 MEDV ~ CRIM + ZN + INDUS + CHAS + NOX + RM + AGE\* + DIS + RAD + TAX + PTRAT + B + LSTAT

Intercept)	CRIM	ZN	INDUS	NOX	RM
32.454444463	-0.119515776	0.050716190	0.048085599	-16.873831778	4.268249228
DIS	RAD	TAX	PTRAT	B	LSTAT
-1.374503346	0.323300559	-0.014337215	-0.942511171	0.009136954	-0.488350735



```
0 MEDV ~ NULL
1 MEDV ~ LSTAT
2 MEDV ~ RM* + LSTAT
3 MEDV ~ RM + PTRAT* + LSTAT
4 MEDV ~ RM + PTRAT + B* + LSTAT
5 MEDV ~ NOX* + RM + DIS* + PTRAT + LSTAT
6 MEDV ~ NOX + RM + DIS + PTRAT + B* + LSTAT
7 MEDV ~ CRIM* + ZN* + NOX + RM + DIS + PTRAT + LSTAT
8 MEDV ~ CRIM + ZN + NOX + RM + DIS + PTRAT + B* + LSTAT
9 MEDV ~ CRIM + ZN + NOX + RM + DIS + RAD* + TAX* + PTRAT + LSTAT
10 MEDV ~ CRIM + ZN + NOX + RM + DIS + RAD + TAX + PTRAT + B* + LSTAT
11 MEDV ~ CRIM + ZN + INDUS* + NOX + RM + DIS + RAD + TAX + PTRAT + B + LSTAT
12 MEDV ~ CRIM + ZN + INDUS + CHAS* + NOX + RM + DIS + RAD + TAX + PTRAT + B + LSTAT
13 MEDV ~ CRIM + ZN + INDUS + CHAS + NOX + RM + AGE* + DIS + RAD + TAX + PTRAT + B + LSTAT
```

```
[1] "alpha      cv.error   cv.se    test.err"
[1] "opt.cv.error: 3.000000 26.918245 4.382135 32.446126"
[1] "min.cv.error: 11.000000 23.314292 3.798790 24.909954"
> modelsj[i.opt]
[[1]]
(Intercept)          RM        PTRAT        LSTAT
14.9532399  4.9839559 -0.9273550 -0.5522889

> sersj[i.opt]
```

```
[1] 3.739824
> modelsj[i.min]
[[1]]
(Intercept)          CRIM           ZN           INDUS          NOX
32.4544444463 -0.119515776  0.050716190  0.048085599 -16.873831778
              RM          DIS           RAD           TAX          PTRAT
4.268249228 -1.374503346  0.323300559 -0.014337215 -0.942511171
              B          LSTAT
0.009136954 -0.488350735
```

```
> sersj[i.min]
[1] 3.09097
```

## Forward stepwise

Вместо того, чтобы проводить поиск среди всех подмножеств, можно рассмотреть только некоторые из них.

Можно, начав с оптимального набора параметров мощности 0, последовательно добавлять такую входную переменную, которая максимальным образом улучшает приближение.

Предположим, что текущая модель имеет  $k$  входных переменных. Имеем соответствующее значение  $\text{RSS}_k$ . Рассмотрим статистику

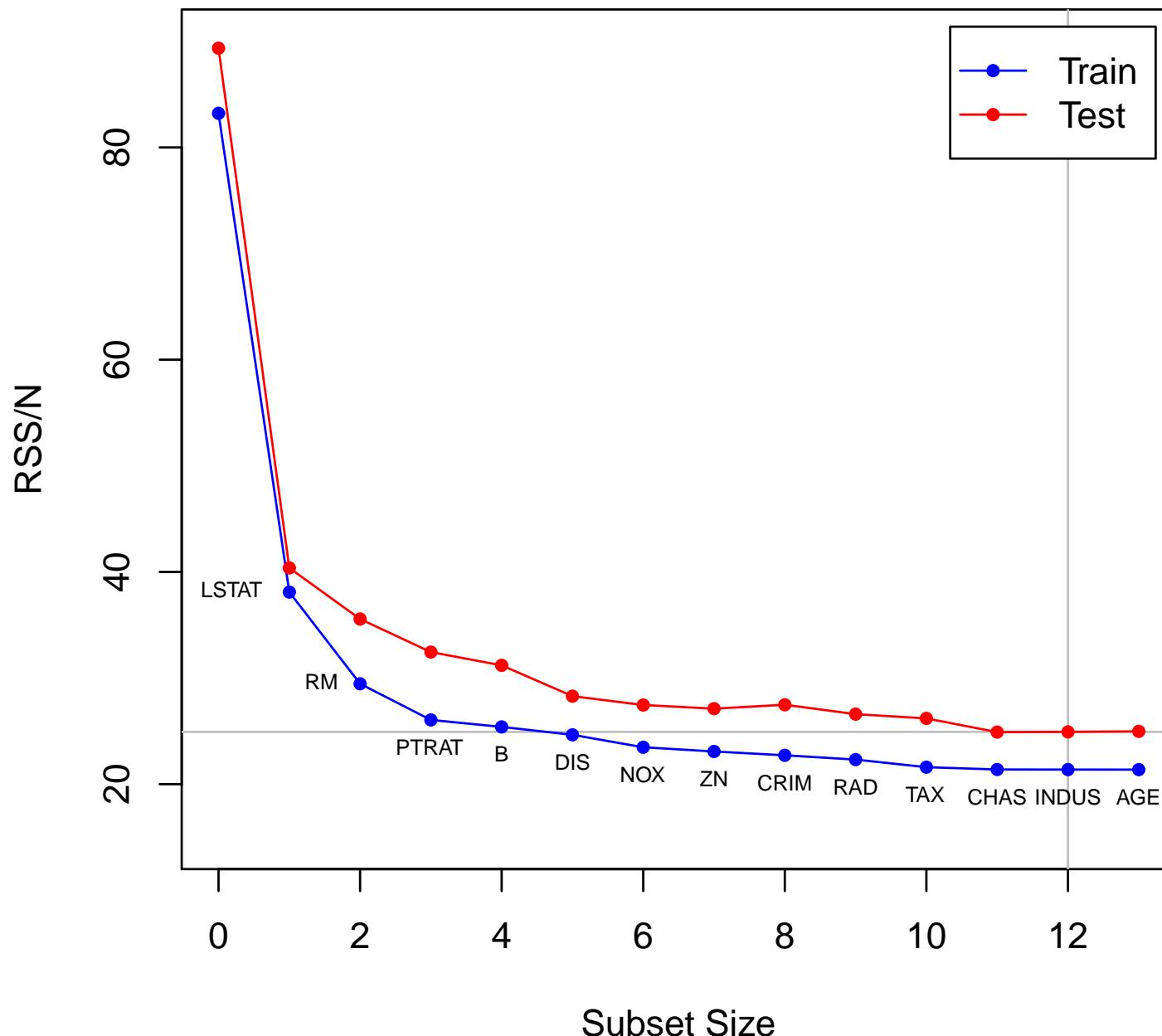
$$F = \frac{\text{RSS}_k - \text{RSS}_{k+1}}{\text{RSS}_{k+1} / (N - k - 2)}.$$

Добавим к набору входных переменных ту, для которой соответствующее значение  $\text{RSS}_{k+1}$  (и, следовательно,  $F$ ) максимально.

Как правило, процедуру добавления новых входных переменных заканчивают, когда  $F$  достигнет процентиля уровня 90% или 95% для  $F(1, N - k - 2)$  распределения.

[Другая стратегия — начав с рассмотрения всего набора входных переменных, последовательно исключать их из этого набора (backward stepwise).]

На всей выборке:



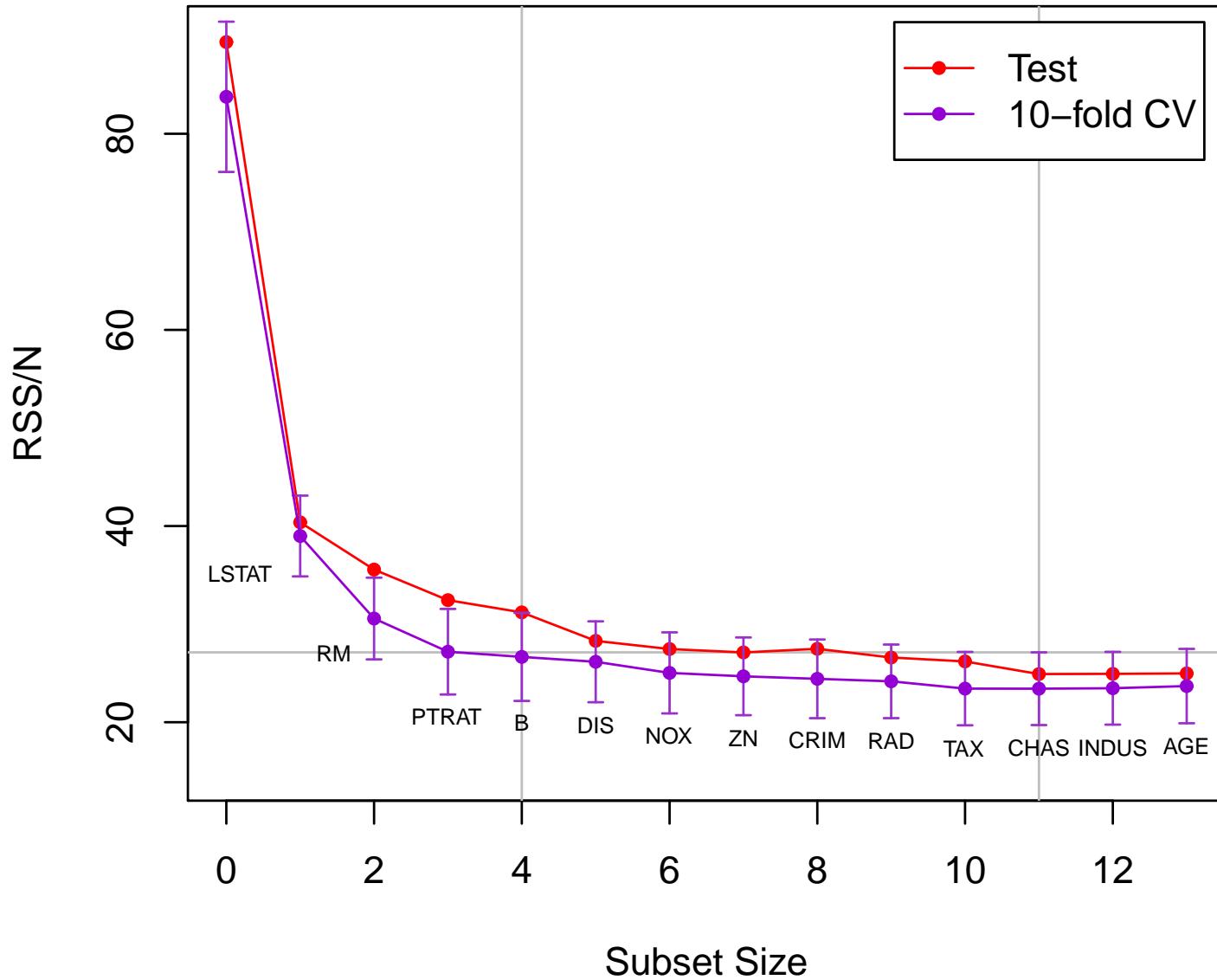
trainerror(12) = 21.37453 testerror(12) = 24.93032 se = 2.995196

```
# (Intercept)          CRIM           ZN           INDUS          CHAS          NOX
# 32.50597493 -0.11633355  0.05036234  0.03227048  2.05067862 -17.77111519
#       PTRAT            B           LSTAT
# -0.91637028  0.00886231 -0.48073216
```

Значения  $F$ -статистики: [1] 477.15382733 117.92466979 52.39653260 10.32469114  
12.06838722 19.91189830 6.80818253 6.31483614 7.09650624 13.06682614 [11]  
4.00876620 0.19502565 0.05884747

Соответствующие  $p$ -value: [1] 0.000000e+00 0.000000e+00 2.330025e-12 1.418964e-03  
5.690409e-04 1.057010e-05 9.417005e-03 1.236951e-02 8.039617e-03 3.395480e-04 [11]  
4.595095e-02 6.590098e-01 8.084541e-01

Останавливаемся на модели из 11 переменных (которая уже встречалась при полном переборе) или из 12 (как и по обучающей выборке)



```

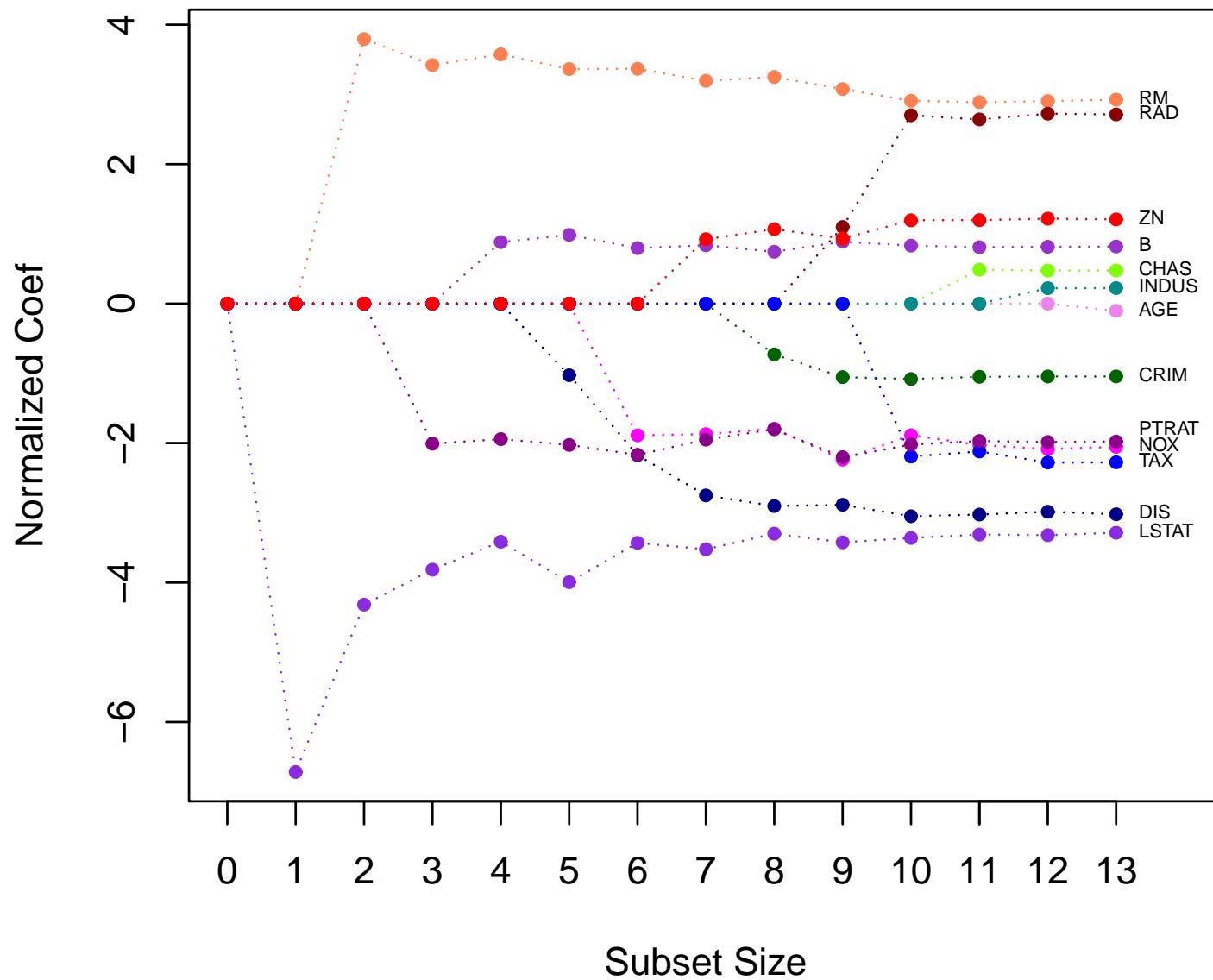
> print.all.numbers(serule, 0:p, cvlm, cvse, RSSftest/Ntest)
[1] "alpha      cv.error  cv.se    test.err"
[1] "opt.cv.error: 4.000000 26.658153 4.487114 31.198681"
[1] "min.cv.error: 11.000000 23.417400 3.707207 24.909954"
> models[i.opt]
[[1]]
(Intercept)          RM          PTRAT          B          LSTAT
8.844995085  5.208485224 -0.897163402  0.009592995 -0.494341896

> sers[i.opt]
[1] 3.827101
> models[i.min]
[[1]]
(Intercept)          CRIM          ZN          CHAS          NOX
32.383528817 -0.116845639  0.049464484  2.102846695 -17.271681307
          RM          DIS          RAD          TAX          PTRAT
4.209503086 -1.391759173  0.302601405 -0.012533028 -0.909537315
          B          LSTAT
0.008799863 -0.479332141

> sers[i.min]

```

[1] 2.996839



### 3.4.2. «Ридж» (гребневая) регрессия (регуляризация)

(*ridge regression*)

Добавим к RSS штрафную функцию, чувствительную к абсолютной величине коэффициентов  $\beta_j$ :

$$\hat{\beta}^{\text{ridge}} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^N \left( y^{(i)} - \beta_0 - \sum_{j=1}^d x_j^{(i)} \beta_j \right)^2 + \lambda \sum_{j=1}^d \beta_j^2 \right\},$$

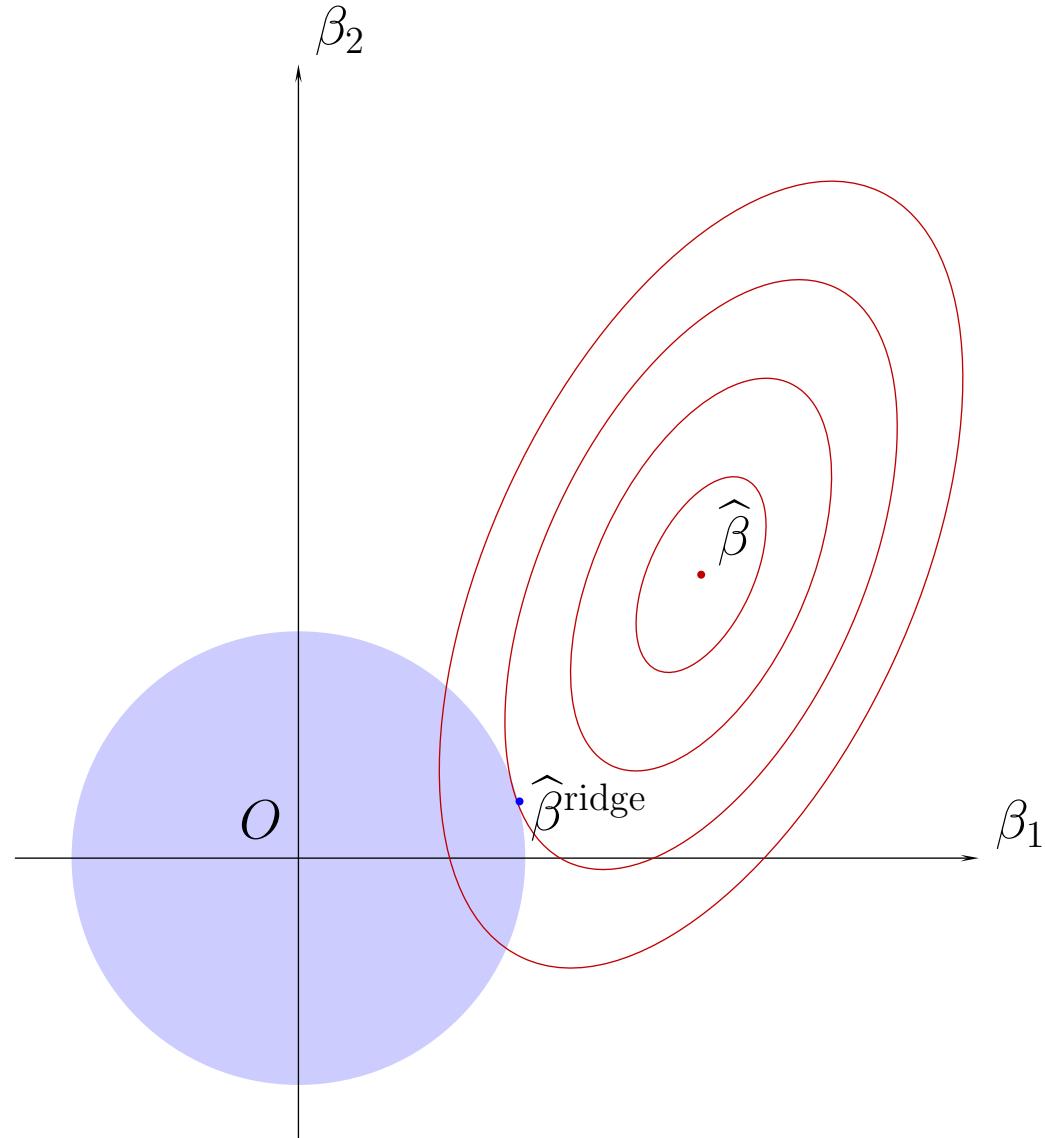
где  $\lambda$  — неотрицательный параметр:

чем больше  $\lambda$ , тем больше чувствительность штрафа к величине коэффициентов

Понятно, что задача эквивалентна следующей задаче условной минимизации:

$$\hat{\beta}^{\text{ridge}} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^N \left( y^{(i)} - \beta_0 - \sum_{j=1}^p x_j^{(i)} \beta_j \right)^2 \right\}, \text{ при условии } \sum_{j=1}^d \beta_j^2 \leq s.$$

Параметр  $s$  определяется по  $\lambda$ , и наоборот.



Если в линейной регрессионной модели много коррелированных переменных, то параметры модели определяются с трудом и имеют большую дисперсию.

Например,  $X_j$  и  $X_{j'}$  сильно зависимы.

Тогда возможна ситуация:

$\beta_j$  велик и положителен, а  $\beta_{j'}$  велик по абсолютному значению и отрицателен:

так как  $X_j$  и  $X_{j'}$  сильно коррелируют, то вклад  $\beta_j x_j$  компенсируется  $\beta_{j'} x_{j'}$ .

Таким образом, добавляемое ограничение на величину коэффициентов  $\beta_j$  должно предотвратить подобную ситуацию.

Коэффициент  $\beta_0$  может быть устранен из штрафной функции, так как он не соответствует ни одной переменной  $x_j$ .

Слагаемое  $\beta_0$  можно выкинуть и из целевой функции.

Действительно, решение задачи наименьших квадратов можно разбить на два этапа:

1. *Центрирование данных:*

- каждое  $x_j^{(i)}$  заменяется на  $x_j^{(i)} - \bar{x}_j$  ( $i = 1, 2, \dots, N$ ;  $j = 1, 2, \dots, d$ )
- в качестве  $\beta_0$  выбирается  $\bar{y}$ , где

$$\bar{x}_j = \frac{1}{N} \sum_{i=1}^N x_j^{(i)}, \quad \bar{y} = \frac{1}{N} \sum_{i=1}^N y^{(i)}.$$

2. Гребневая регрессия (без коэффициента  $\beta_0$ )

Пусть центрирование произведено, следовательно,  $\mathbf{X}$  имеет  $d$  (а не  $d + 1$ ) столбцов

$$\text{RSS}^{\text{ridge}}(\beta, \lambda) = (\mathbf{y} - \mathbf{X}\beta)^\top (\mathbf{y} - \mathbf{X}\beta) + \lambda\beta^\top\beta,$$

$$\text{RSS}^{\text{ridge}}(\beta, \lambda) \rightarrow \min_{\beta}$$

Дифференцируя, находим:

$$\frac{\partial \text{RSS}^{\text{ridge}}}{\partial \beta} = -2\mathbf{X}^\top(\mathbf{y} - \mathbf{X}\beta) + 2\lambda\beta, \quad \frac{\partial^2 \text{RSS}^{\text{ridge}}}{\partial \beta \partial \beta^\top} = 2\mathbf{X}^\top\mathbf{X} + 2\lambda\mathbf{I}.$$

откуда

$$\widehat{\beta}^{\text{ridge}} = (\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y} \tag{*}$$

Итак, решение  $\widehat{\beta}^{\text{ridge}}$  задачи гребневой линейной регрессии также является функцией, линейно зависящей от  $\mathbf{y}$ .

Если  $\lambda > 0$ , то матрица  $\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I}$  невырождена (положительно определена), даже если столбцы в  $\mathbf{X}$  линейно зависимы

(\*) представляет собой стандартную *регуляризацию*

Итак, от системы  $\mathbf{X}\beta = \mathbf{y}$  мы перешли к  $(\mathbf{X}^\top\mathbf{X} + \lambda\mathbf{I})\beta = \mathbf{X}^\top\mathbf{y}$

### 3.4.3. Отступление. Регуляризация (А.Н. Тихонов)

Система  $(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})\beta = \mathbf{X}^\top \mathbf{y}$  называется *регуляризованной* к  $\mathbf{X}\beta = \mathbf{y}$ .

$\lambda$  — *параметр регуляризации*,  $\lambda > 0$ .

Обозначим  $\widehat{\beta}(\lambda)$  — (единственное) решение системы  $(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})\beta = \mathbf{X}^\top \mathbf{y}$ .

#### Теорема 3.6 (А.Н. Тихонов)

Пусть  $\lambda_n$  — некоторая последовательность,  $\lambda_n > 0$  и  $\lambda_n \rightarrow 0$ .

Тогда  $\widehat{\beta}(\lambda_n)$  стремится к нормальному псевдорешению системы  $\mathbf{X}\beta = \mathbf{y}$   
(из всех псевдорешений нормальное псевдорешение имеет минимальную норму).

Если  $\lambda$  близко к нулю, то регуляризованная система близка к вырожденной, поэтому на практике решение  $\widehat{\beta}(\lambda)$  будет найдено с большой ошибкой.

Метод регуляризации заключается в решении регуляризованных систем для конечного числа значений  $\lambda_n$  и дальнейшем выборе того решения, для которого норма невязки минимальна.

### **3.4.4. Гребневая регрессия и сингулярное разложение**

Выразим  $\widehat{\beta}^{\text{ls}}$  и  $\widehat{\mathbf{y}}^{\text{ls}}$  (без регуляризации), используя *SVD*:

$$\begin{aligned}\widehat{\beta}^{\text{ls}} &= (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} = ((\mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^\top)^\top \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^\top)^{-1} (\mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^\top)^\top \mathbf{y} = \\ &= (\mathbf{V} \boldsymbol{\Sigma} \mathbf{U}^\top \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^\top)^{-1} \mathbf{V} \boldsymbol{\Sigma} \mathbf{U}^\top \mathbf{y} = (\mathbf{V}^\top)^{-1} \boldsymbol{\Sigma}^{-2} \mathbf{V}^{-1} \mathbf{V} \boldsymbol{\Sigma} \mathbf{U}^\top \mathbf{y} = \mathbf{V} \boldsymbol{\Sigma}^{-1} \mathbf{U}^\top \mathbf{y}\end{aligned}$$

$$\widehat{\mathbf{y}}^{\text{ls}} = \mathbf{X} \widehat{\beta}^{\text{ls}} = \mathbf{U} \boldsymbol{\Sigma} \mathbf{V}^\top \mathbf{V} \boldsymbol{\Sigma}^{-1} \mathbf{U}^\top \mathbf{y} = \mathbf{U} \mathbf{U}^\top \mathbf{y} = \sum_{j=1}^d \mathbf{u}_j \mathbf{u}_j^\top \mathbf{y}$$

Заметим, что  $\mathbf{u}_j(\mathbf{u}_j^\top \mathbf{y})$  есть проекция вектора  $\mathbf{y}$  на вектор  $\mathbf{u}_j$ , а вся сумма (как мы уже видели) есть проекция вектора  $\mathbf{y}$  на подпространство, натянутое на  $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_d$ .

Аналогично, если проводится регуляризация:

$$\widehat{\mathbf{y}}^{\text{ridge}} = \mathbf{X} \widehat{\beta}^{\text{ridge}} = \mathbf{X} (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{y} = \mathbf{U} \boldsymbol{\Sigma} (\boldsymbol{\Sigma} + \lambda \mathbf{I})^{-1} \boldsymbol{\Sigma} \mathbf{U}^\top \mathbf{y} = \sum_{j=1}^d \mathbf{u}_j \frac{\sigma_j^2}{\sigma_j^2 + \lambda} \mathbf{u}_j^\top \mathbf{y}$$

$$\hat{\mathbf{y}}^{\text{ridge}} = \mathbf{U}\Sigma(\Sigma + \lambda\mathbf{I})^{-1}\Sigma\mathbf{U}^\top\mathbf{y} = \sum_{j=1}^d \mathbf{u}_j \frac{\sigma_j^2}{\sigma_j^2 + \lambda} \mathbf{u}_j^\top \mathbf{y}$$

Итак, как и для линейной регрессии, линейная гребневая регрессия вычисляет проекцию  $\mathbf{u}(\mathbf{u}_j^\top \mathbf{y})$  вектора  $\mathbf{y}$  на вектор  $\mathbf{u}_j$ , но затем домножает эту проекцию на

$$\frac{\sigma_j^2}{\sigma_j^2 + \lambda} \leq 1.$$

Чем больше  $\sigma_j$ , тем ближе этот множитель к 1.

Чем меньше  $\sigma_j$ , тем ближе этот множитель к 0.

Таким образом, наибольшему уменьшению подвергаются компоненты, соответствующие меньшим сингулярным числам  $\sigma_j$ .

### 3.4.5. Лассо (R. Tibshirani)

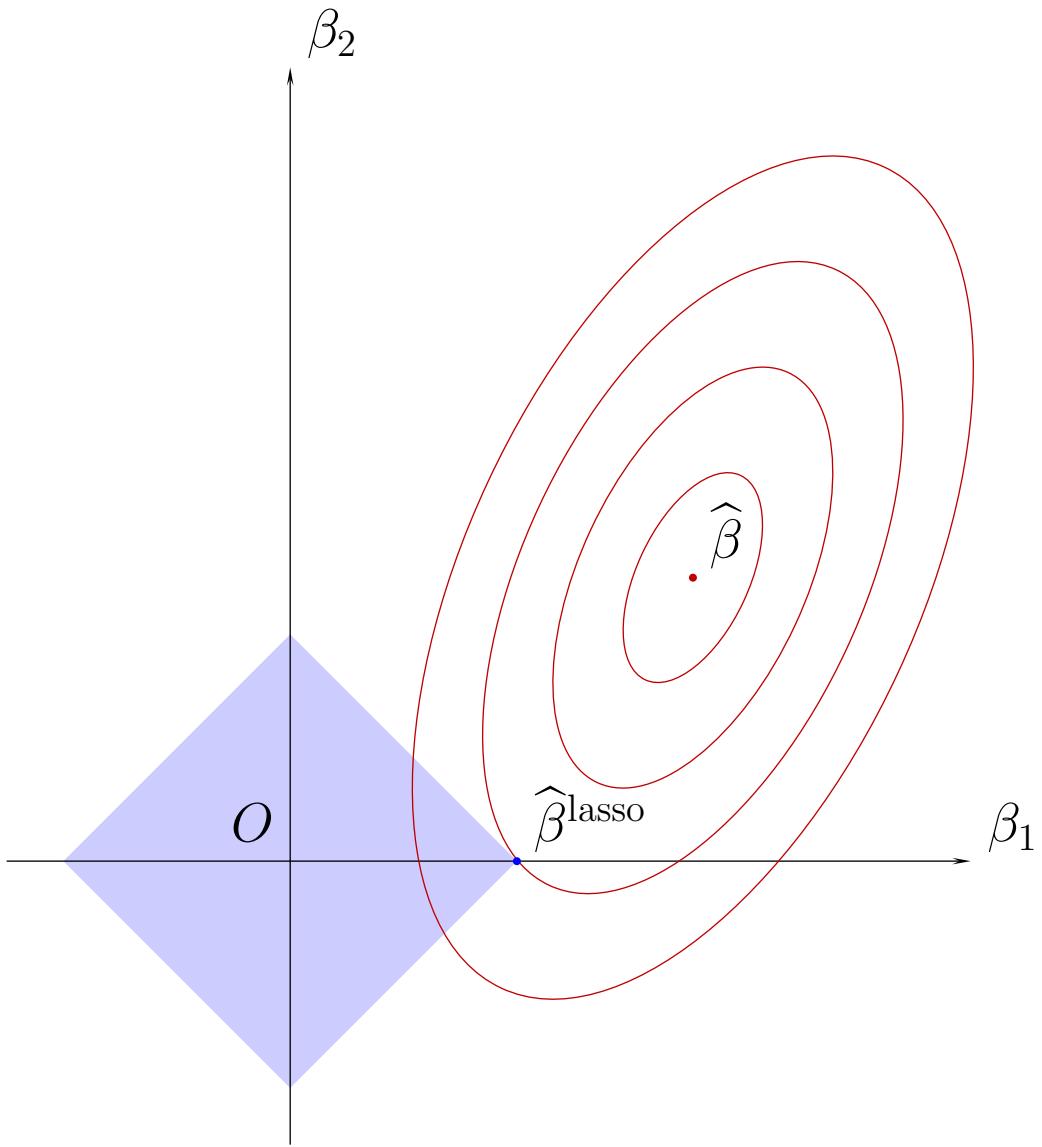
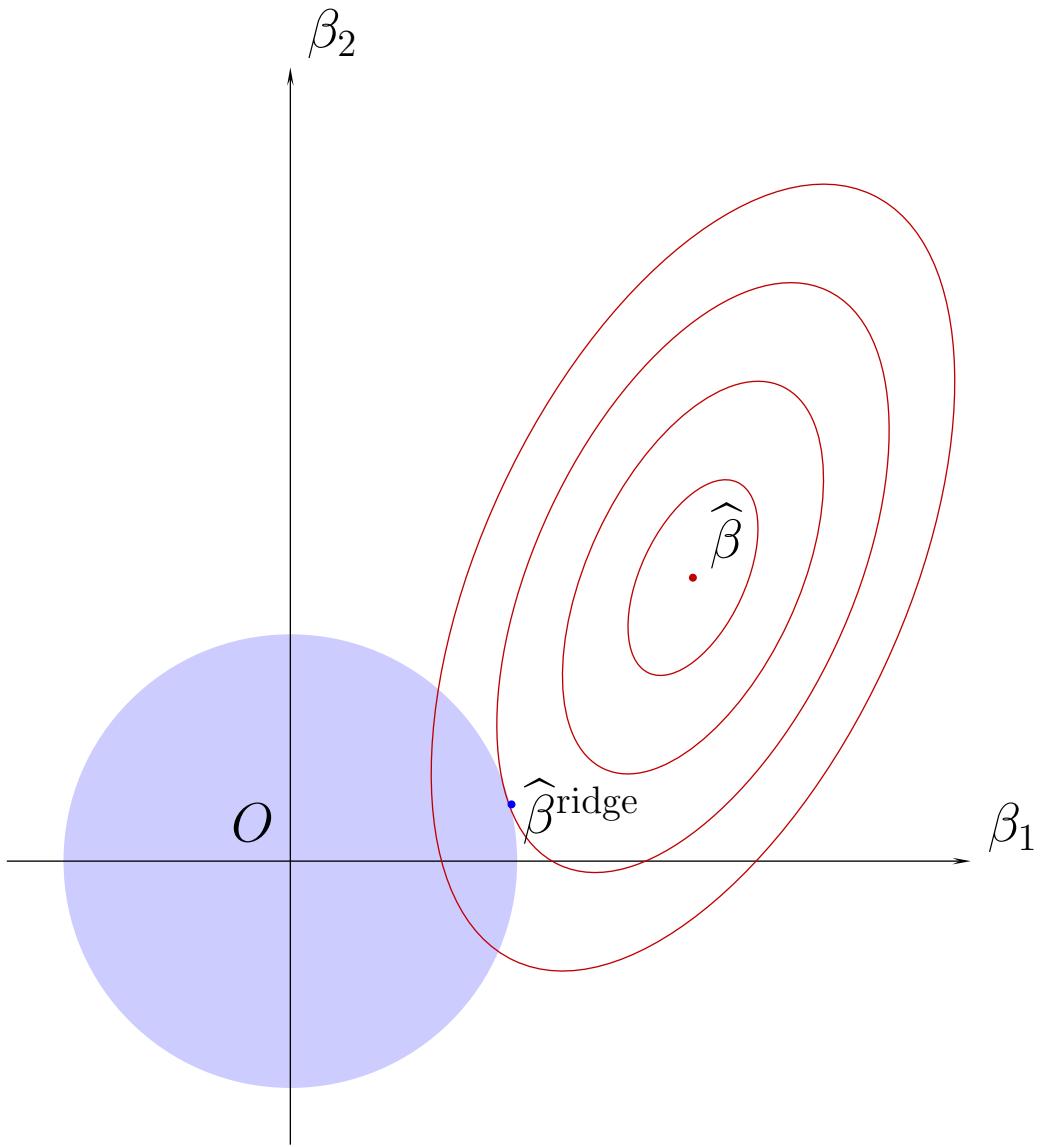
LASSO – «least absolute shrinkage and selection operator»

$$\hat{\beta}^{\text{lasso}} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^N \left( y^{(i)} - \beta_0 - \sum_{j=1}^d x_j^{(i)} \beta_j \right)^2 \right\}, \text{ при условии } \sum_{j=1}^d |\beta_j| \leq s.$$

Если  $s$  достаточно мало, то в типичной ситуации часть коэффициентов  $\hat{\beta}_j^{\text{lasso}}$  равна в точности 0.

Почему?

(При гребневой регрессии такого, как правило, не происходит)



Графики зависимости величины коэффициентов  $\widehat{\beta}_j^{\text{ridge}}$  в гребневой регрессии от  $\lambda$  в примере `boston`.

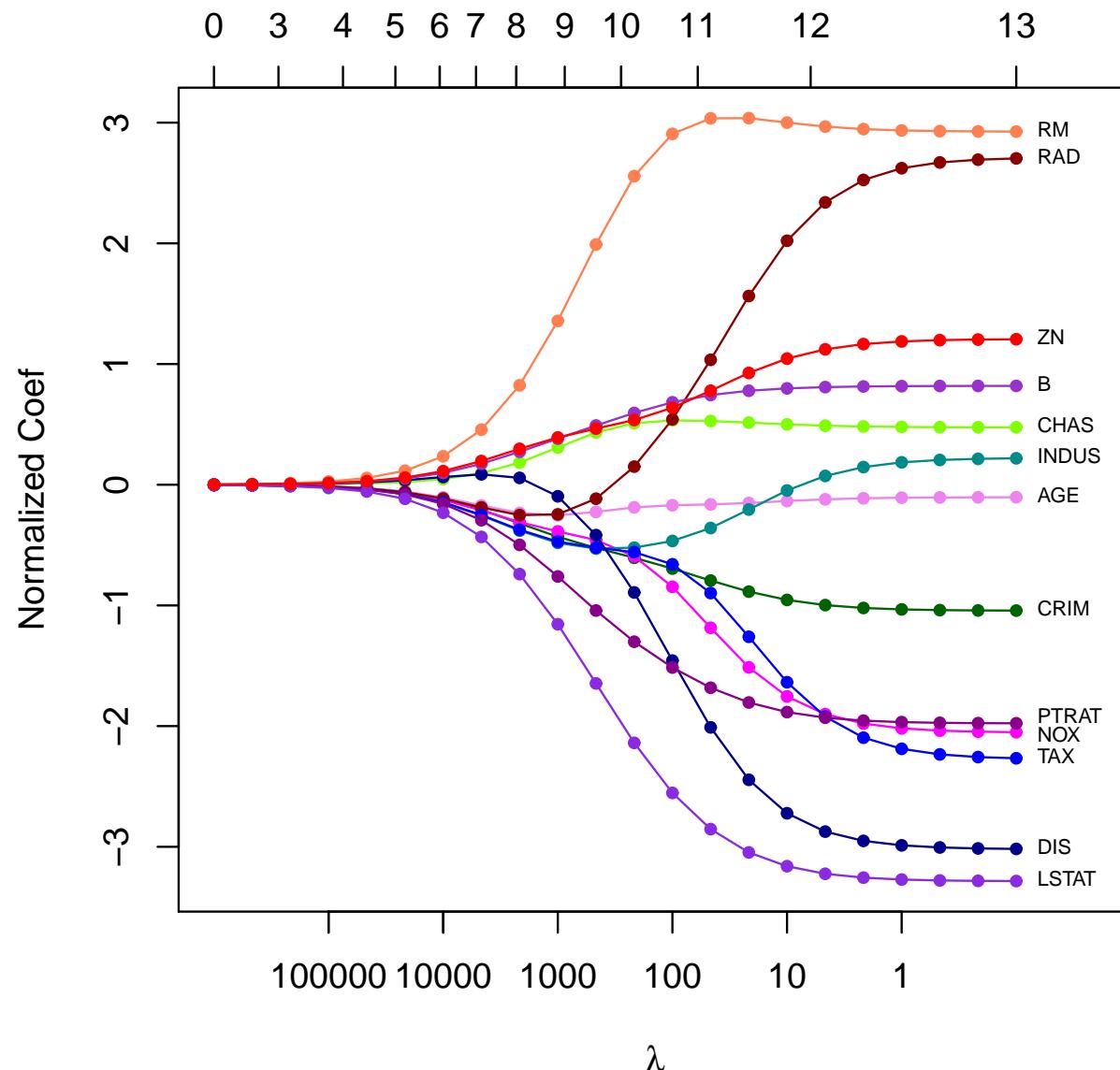
*Эффективное число степеней свободы*

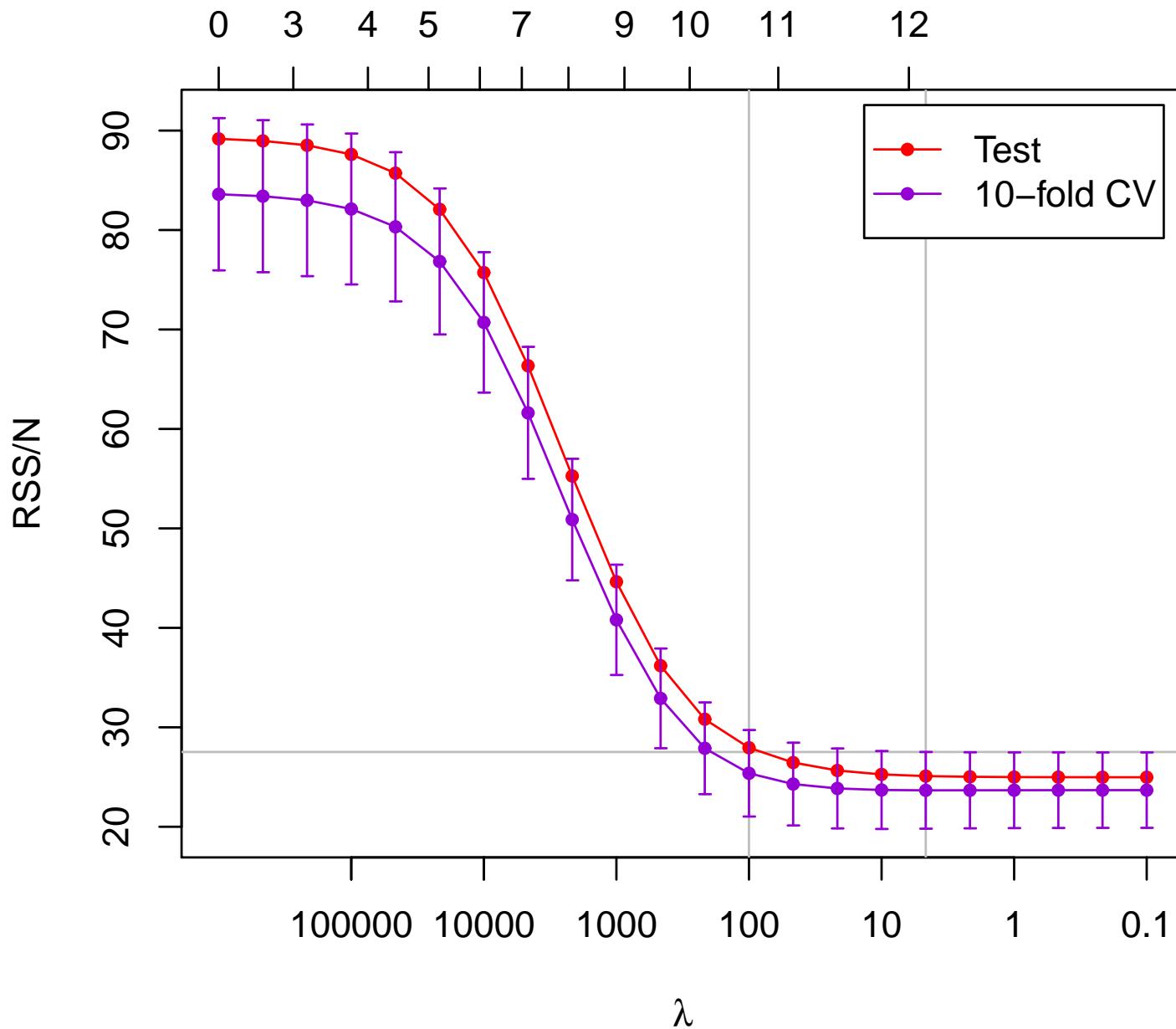
$$\text{df}(\lambda) = \text{tr} (\mathbf{X}(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^\top) = \sum_{j=1}^d \frac{\sigma_j^2}{\sigma_j^2 + \lambda}.$$

Если  $\lambda = 0$  (нет регуляризации), то  $\text{df}(\lambda) = d$ .

Если  $\lambda \rightarrow \infty$ , то  $\text{df}(\lambda) \rightarrow 0$ .

Графики зависимости величины коэффициентов  $\hat{\beta}_j^{\text{ridge}}$  в гребневой регрессии от  $\text{df}(\lambda)$  в примере boston





```

[1] "alpha      cv.error  cv.se    test.err"
[1] "opt.cv.error: 100.000000 25.377471 4.348975 39.699647"
[1] "min.cv.error: 4.641589 23.663354 3.857456 34.884854"

> model <- my.lm.ridge(MEDV~., data = boston.train, lambda = lambda[i.opt])
> model

            CRIM          ZN          INDUS         CHAS        NOX
19.189683156 -0.077377068  0.026373388 -0.067974027  2.302313172 -7.206008616
            RM          AGE          DIS          RAD          TAX        PTRAT
4.234950160 -0.006089838 -0.670412894  0.062041179 -0.003898142 -0.699415567
            B          LSTAT
0.007425473 -0.369705443

> s.error(predict.lm.ridge(model, boston.train) - boston.train$MEDV)
[1,]
[1,] 3.604228

>
> model <- my.lm.ridge(MEDV~., data = boston.train, lambda = lambda[i.min])
> model

            CRIM          ZN          INDUS         CHAS        NOX
30.362243917 -0.111084960  0.046356095  0.010604428  2.113290512
            RM          AGE          DIS          RAD

```

```
-16.194492167  4.322668332 -0.004342458 -1.322330996  0.267994607
          TAX           PTRAT            B          LSTAT
-0.011360520 -0.891331859  0.008795508 -0.466686769
> s.error(predict.lm.ridge(model, boston.train) - boston.train$MEDV)
 [,1]
[1,] 3.070772
>
```

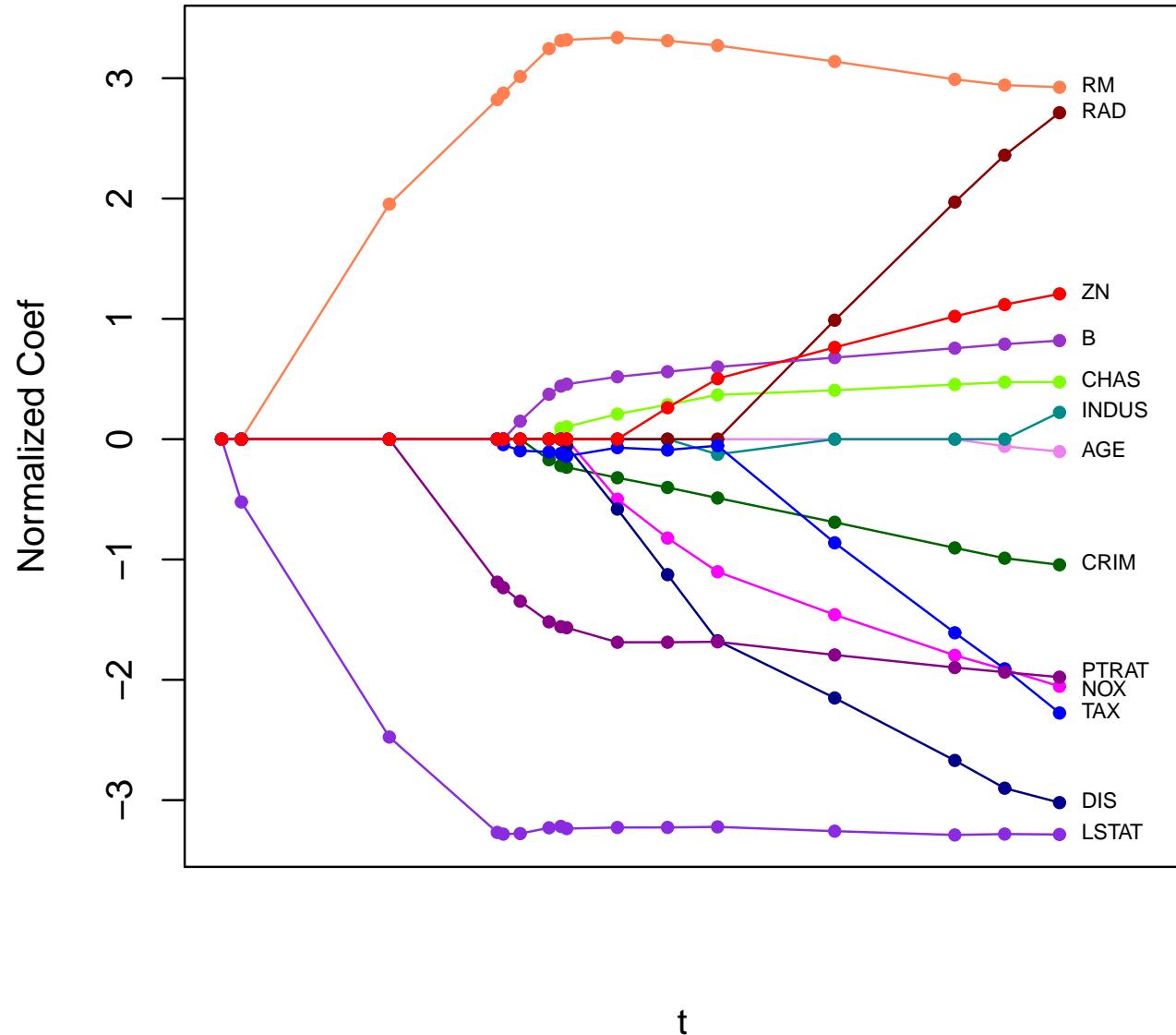
Лассо. Введем параметр

$$t = \frac{s}{\sum_{j=1}^d |\hat{\beta}_j|}$$

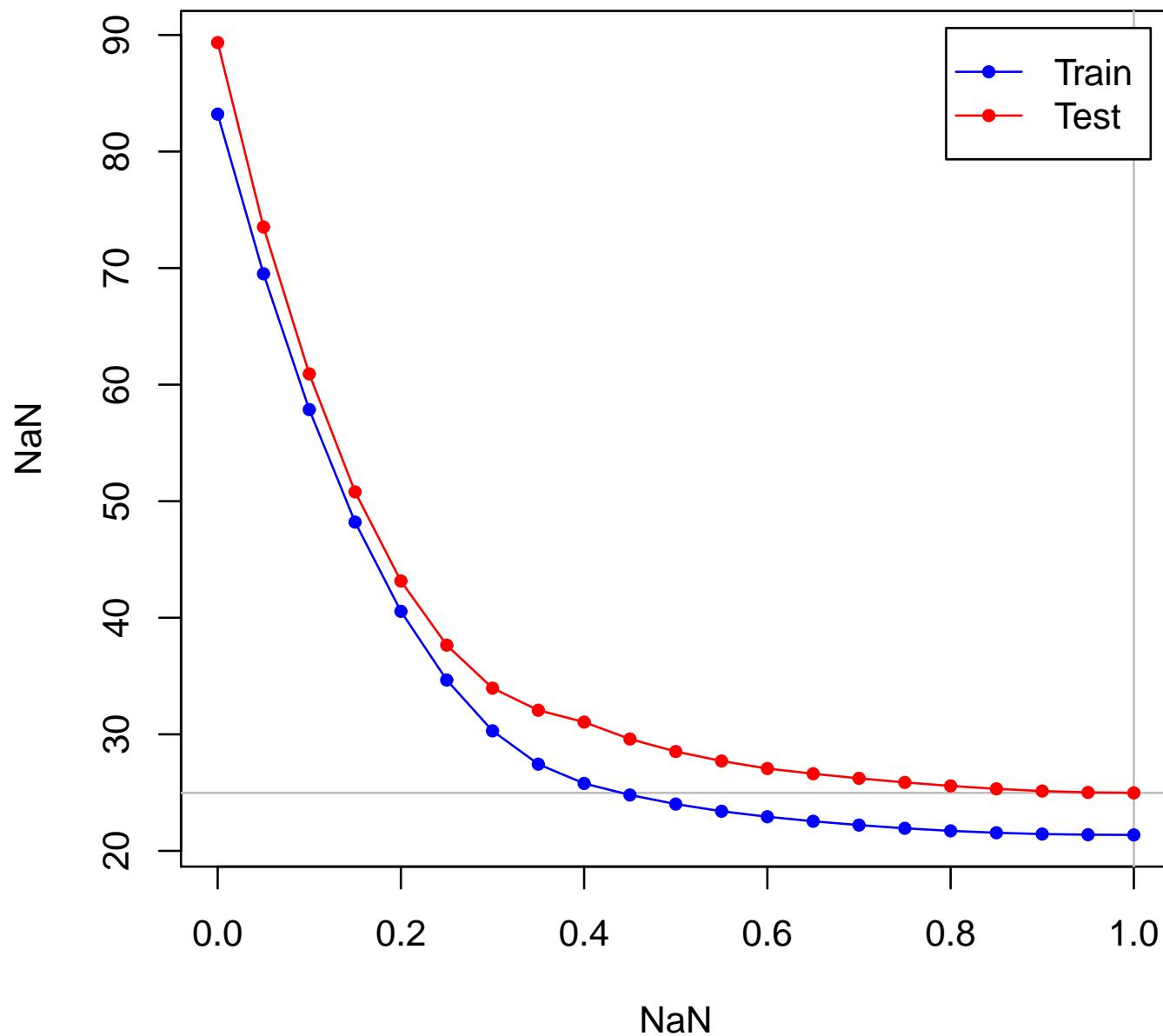
При  $t = 0$  все  $\hat{\beta}_j^{\text{lasso}}$  равны нулю.

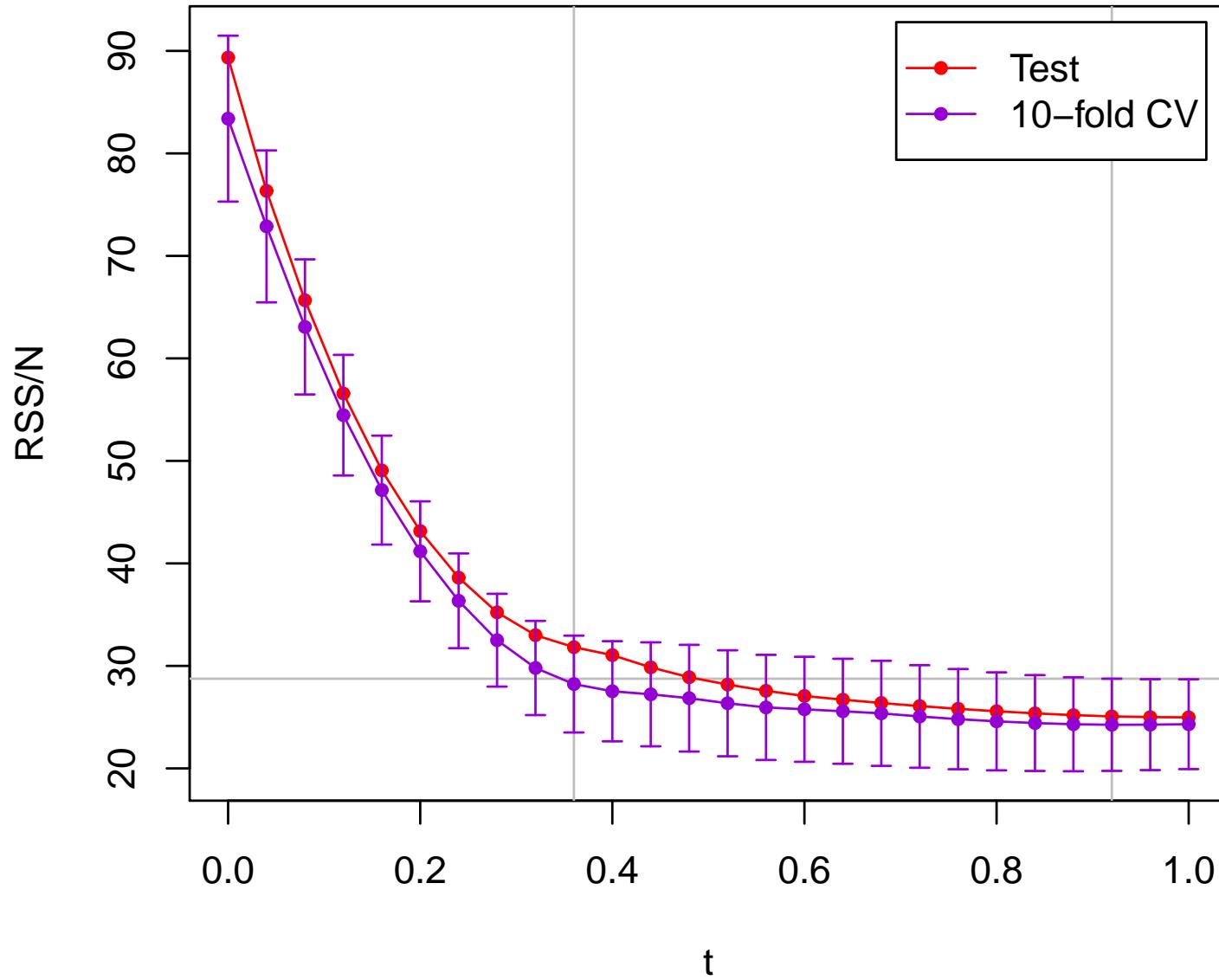
При  $t = 1$  имеем  $\hat{\beta}_j^{\text{lasso}} = \hat{\beta}_j$  ( $j = 1, 2, \dots, d$ ).

Графики зависимости величины коэффициентов  $\widehat{\beta}_j^{\text{lasso}}$  от параметра  $t = \frac{s}{\sum_{j=1}^d |\widehat{\beta}_j|}$  в примере `boston`.



На всей обучающей выборке





```

[1] "alpha      cv.error  cv.se    test.err"
[1] "opt.cv.error: 0.360000 28.146944 5.136314 31.825188"
[1] "min.cv.error: 0.920000 24.009220 4.545165 25.073336"
>

> coef(predict.lars(lars.model, as.matrix(boston.train[, 1:13]), t[i.opt], type="norm"))
          CRIM          ZN          INDUS         CHAS          NOX
-0.0020202505 0.0000000000 0.0000000000 0.0000000000 0.0000000000
          RM          AGE          DIS          RAD          TAX
4.4266045672 0.0000000000 0.0000000000 0.0000000000 -0.0005724785
          PTRAT         B          LSTAT
-0.6302981249 0.0018838301 -0.4738472707
> pred <- predict.lars(lars.model, as.matrix(boston.test[, 1:13]), t[i.opt], type="norm")
> s.error(boston.test$MEDV - pred$fit)
[1] 3.101102
>
> coef(predict.lars(lars.model, as.matrix(boston.train[, 1:13]), t[i.min], type="norm"))
          CRIM          ZN          INDUS         CHAS          NOX
-0.107838566 0.045222939 0.000000000 2.026581810 -16.064602484
          RM          AGE          DIS          RAD          TAX
4.304359991 -0.001613487 -1.308660957 0.259449341 -0.010850565

```

PTRAT	B	LSTAT
-0.889868116	0.008499284	-0.475381318

```
> pred <- predict.lars(lars.model, as.matrix(boston.test[, 1:13]), t[i.min], ty)
> s.error(boston.test$MEDV - pred$fit)
[1] 2.376218
```

### **3.4.6. Трудоемкости методов**

Линейная задача наименьших квадратов (и гребневая регрессия) решается методами:  
 $QR$ -разложения,  
разложения Холецкого,  
 $SVD$ -разложением и др.

Эти методы требуют  $O(Nd^2 + d^3)$  операций.

Метод лассо использует квадратическое программирование.



*Глава 4*

## **Задача классификации**

## 4.1. Наивный байесовский классификатор

Пусть все компоненты случайной величины  $X = (X_1, \dots, X_d)$  дискретны.

$$\Pr \{Y = y | X = x\} = \frac{\Pr \{Y = y\} \cdot \Pr \{X = x | Y = y\}}{\sum_{k=1}^K \Pr \{Y = k\} \cdot \Pr \{X = x | Y = k\}}$$

*Основное предположение наивного байесовского классификатора:*

переменные  $X_1, \dots, X_d$  условно независимы при любом заданном  $Y = y$ :

$$\Pr \{X_1 = x_1, \dots, X_d = x_d | Y = y\} = \Pr \{X_1 = x_1 | Y = y\} \cdot \dots \cdot \Pr \{X_d = x_d | Y = y\},$$

или, короче,

$$\Pr \{X = x | Y = y\} = \prod_{j=1}^d \Pr \{X_j = x_j | Y = y\}.$$

По принципу максимума апостериорной вероятности

$$f(x) = \operatorname{argmax}_y \Pr \{Y = y | X = x\} = \operatorname{argmax}_y \left( \Pr \{Y = y\} \cdot \prod_{j=1}^d \Pr \{X_j = x_j | Y = y\} \right).$$

Вероятности заменяем частотой:

$$\Pr \{Y = y\} \approx \frac{1}{N} | \left\{ i : y^{(i)} = y \right\} |$$

$$\Pr \{X_j = x_j | Y = y\} \approx \frac{|\left\{ i : x_j^{(i)} = x_j \right\}|}{|\left\{ i : y^{(i)} = y \right\}|}$$

Задача: предсказать исход матча команды А с командой В по имеющимся данным о предыдущих матчах.

<i>№</i>	<i>Поле</i>	<i>Ключевые игроки</i>	<i>Ключевые игроки противника</i>	<i>Погода</i>	<i>Исход</i>
	$X_1$	$X_2$	$X_3$	$X_4$	$Y$
1	свое	присутствуют	присутствуют	дождь	поражение
2	чужое	отсутствуют	отсутствуют	нет дождя	победа
3	свое	присутствуют	отсутствуют	нет дождя	победа
4	свое	присутствуют	отсутствуют	дождь	ничья
5	чужое	отсутствуют	присутствуют	дождь	поражение
6	чужое	присутствуют	присутствуют	нет дождя	поражение
7	чужое	отсутствуют	присутствуют	нет дождя	поражение
8	свое	отсутствуют	отсутствуют	нет дождя	ничья
9	свое	присутствуют	отсутствуют	дождь	победа
10	чужое	присутствуют	отсутствуют	нет дождя	победа

Априорные вероятности:  $Y$

	победа	ничья	поражение
	0.4	0.2	0.4

Условные вероятности:

Поле		
$Y$	свое	чужое
победа	0.50	0.50
ничья	1.00	0.00
поражение	0.25	0.75

Ключевые игроки		
$Y$	отсутствуют	присутствуют
победа	0.25	0.75
ничья	0.50	0.50
поражение	0.50	0.50

Ключевые игроки противника		
$Y$	отсутствуют	присутствуют
победа	1	0
ничья	1	0
поражение	0	1

Погода		
$Y$	дождь	нет дождя
победа	0.25	0.75
ничья	0.50	0.50
поражение	0.50	0.50

Нужно предсказать  $Y$ ,  
если  $X_1 = \text{свое}$ ,  $X_2 = \text{присутствуют}$ ,  $X_3 = \text{отсутствуют}$ ,  $X_4 = \text{дождь}$ .

$$\Pr \{\text{победа} | \text{свое, присутствуют, отсутствуют, дождь}\} = \\ = \frac{\Pr \{\text{победа}\} \cdot \Pr \{\text{свое} | \text{победа}\} \cdot \Pr \{\text{присутствуют} | \text{победа}\} \cdot \Pr \{\text{отсутствуют} | \text{победа}\} \cdot \Pr \{\text{дождь} | \text{победа}\}}{\Pr \{\text{свое, присутствуют, отсутствуют, дождь}\}} =$$

$$= \frac{0.4 \cdot 0.5 \cdot 0.75 \cdot 1 \cdot 0.25}{0.0375 + 0.05 + 0} = \frac{0.0375}{0.0875} = 0.4286$$

$$\Pr \{\text{ничья} | \text{свое, присутствуют, отсутствуют, дождь}\} = \\ = \frac{\Pr \{\text{ничья}\} \cdot \Pr \{\text{свое} | \text{ничья}\} \cdot \Pr \{\text{присутствуют} | \text{ничья}\} \cdot \Pr \{\text{отсутствуют} | \text{ничья}\} \cdot \Pr \{\text{дождь} | \text{ничья}\}}{\Pr \{\text{свое, присутствуют, отсутствуют, дождь}\}} =$$

$$= \frac{0.2 \cdot 1 \cdot 0.5 \cdot 1 \cdot 0.5}{0.0375 + 0.05 + 0} = \frac{0.05}{0.0875} = 0.5714$$

$$\Pr \{Y = \text{поражение} | \text{свое, присутствуют, отсутствуют, дождь}\} = \\ = \frac{\Pr \{\text{пораж.}\} \cdot \Pr \{\text{свое} | \text{пораж.}\} \cdot \Pr \{\text{присутствуют} | \text{пораж.}\} \cdot \Pr \{\text{отсутствуют} | \text{пораж.}\} \cdot \Pr \{\text{дождь} | \text{пораж.}\}}{\Pr \{\text{свое, присутствуют, отсутствуют, дождь}\}} =$$

$$= \frac{0.4 \cdot 0.25 \cdot 0.5 \cdot 0 \cdot 0.5}{0.0375 + 0.05 + 0} = 0$$

На обучающей выборке ошибка 0.1 (неправильно 9-й  $\rightarrow$  ничья)

#### 4.1.1. Сглаживание Лапласа

Пусть нужно предсказать  $Y$ ,

если  $X_1 = \text{нейтральное}$ ,  $X_2 = \text{присутствуют}$ ,  $X_3 = \text{отсутствуют}$ ,  $X_4 = \text{нет дождя}$ .

$$\Pr \{X_1 = \text{нейтральное} | Y = y\} \approx \frac{|\{i : x_1^{(i)} = \text{нейтральное}\}|}{|\{i : y^{(i)} = y\}|} = 0$$

$\Pr \{X_1 = \text{нейтральное}, X_2 = \text{присутствуют}, X_3 = \text{отсутствуют}, X_4 = \text{нет дождя}\} =$

$= \Pr \{\text{нейтральное, присутствуют, отсутствуют, нет дождя} | \text{победа}\} \cdot \Pr \{\text{победа}\} +$

$+ \Pr \{\text{нейтральное, присутствуют, отсутствуют, нет дождя} | \text{ничья}\} \cdot \Pr \{\text{ничья}\} +$

$+ \Pr \{\text{нейтральное, присутствуют, отсутствуют, нет дождя} | \text{поражение}\} \cdot \Pr \{\text{поражение}\} = 0$

$$\Pr \{\text{победа} | \text{нейтральное, присутствуют, отсутствуют, нет дождя}\} = \frac{0}{0}$$

$$\Pr \{\text{ничья} | \text{нейтральное, присутствуют, отсутствуют, нет дождя}\} = \frac{0}{0}$$

$$\Pr \{\text{поражение} | \text{нейтральное, присутствуют, отсутствуют, нет дождя}\} = \frac{0}{0}$$

*Сглаживание Лапласа:*

$$\Pr \{X_j = x_j | Y = y\} \approx \frac{|\{i : x_j^{(i)} = x_j\}| + 1}{|\{i : y^{(i)} = y\}| + s_j},$$

где  $s_j$  — количество значений, которые принимает признак  $x_j$ .

(т.е. притворяемся, что видели каждый признак 1 раз больше, чем на самом деле)

Легко видеть, что тогда

$$\sum_{x_j} \frac{|\{i : x_j^{(i)} = x_j\}| + 1}{|\{i : y^{(i)} = y\}| + s_j} = 1,$$

*Сглаживание Лидстоуна:*

$$\Pr \{X_j = x_j | Y = y\} \approx \frac{|\{i : x_j^{(i)} = x_j\}| + \alpha}{|\{i : y^{(i)} = y\}| + \alpha s_j},$$

где  $s_j$  — количество значений, которые принимает признак  $x_j$ ,  $\alpha \geq 0$ .

Априорные вероятности  $Y$  (не изменяются):

	победа	ничья	поражение
	0.4	0.2	0.4

	Поле				Ключевые игроки	
$Y$	свое	чужое	нейтральное	$Y$	отсутствуют	присутствуют
победа	3/7	3/7	1/7	победа	0.25	0.75
ничья	3/5	1/5	1/5	ничья	0.50	0.50
поражение	2/7	4/5	1/7	поражение	0.50	0.50

	Ключевые игроки противника	
$Y$	отсутствуют	присутствуют
победа	5/6	1/6
ничья	3/4	1/4
поражение	1/6	5/6

	Погода	
$Y$	дождь	нет дождя
победа	2/3	1/3
ничья	1/2	1/2
поражение	1/2	1/2

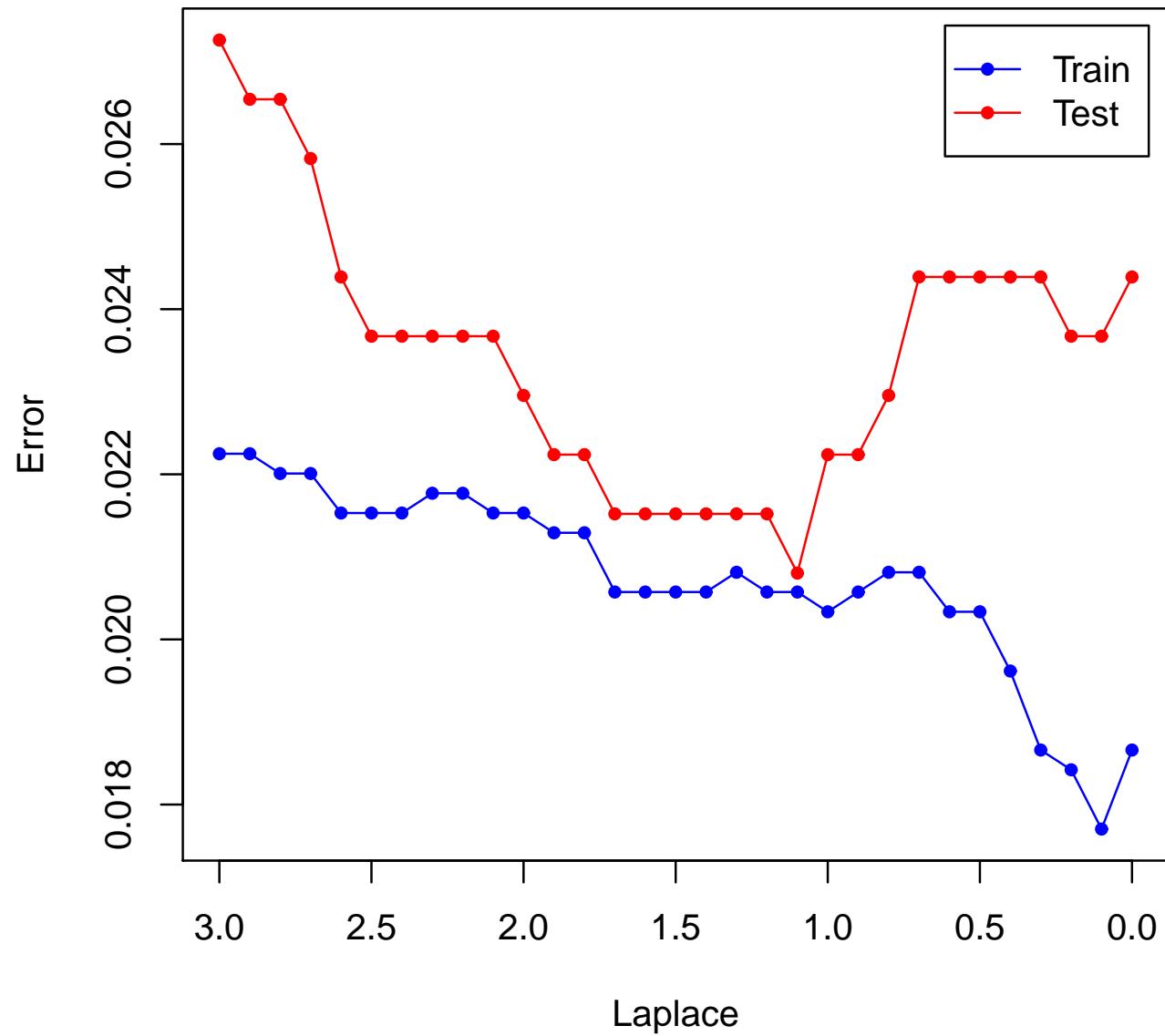
Предсказать  $Y$ ,  
если  $X_1$  = нейтральное,  $X_2$  = присутствуют,  $X_3$  = отсутствуют,  $X_4$  = нет дождя.

$$\Pr \{ \text{победа} | \text{нейтральное, присутствуют, отсутствуют, нет дождя} \} = \\ = \frac{\Pr \{ \text{победа} \} \cdot \Pr \{ \text{нейтральное} | \text{победа} \} \cdot \Pr \{ \text{присутствуют} | \text{победа} \} \cdot \Pr \{ \text{отсутствуют} | \text{победа} \} \cdot \Pr \{ \text{нет дождя} | \text{победа} \}}{\Pr \{ \text{нейтральное, присутствуют, отсутствуют, нет дождя} \}} \\ = \frac{\frac{2/5 \cdot 1/7 \cdot 2/3 \cdot 5/6 \cdot 2/3}{0.0212 + 0.0075 + 0.0024}}{0.0311} = 0.6817$$

$$\Pr \{ \text{ничья} | \text{нейтральное, присутствуют, отсутствуют, нет дождя} \} = \\ = \frac{\Pr \{ \text{ничья} \} \cdot \Pr \{ \text{нейтральное} | \text{ничья} \} \cdot \Pr \{ \text{присутствуют} | \text{ничья} \} \cdot \Pr \{ \text{отсутствуют} | \text{ничья} \} \cdot \Pr \{ \text{нет дождя} | \text{ничья} \}}{\Pr \{ \text{нейтральное, присутствуют, отсутствуют, нет дождя} \}} = \\ = \frac{\frac{1/5 \cdot 1/5 \cdot 1/2 \cdot 3/4 \cdot 1/2}{0.0212 + 0.0075 + 0.0024}}{0.0311} = 0.2416$$

$$\Pr \{ \text{поражение} | \text{нейтральное, присутствуют, отсутствуют, нет дождя} \} \\ = \frac{\Pr \{ \text{пораж.} \} \cdot \Pr \{ \text{нейтральное} | \text{пораж.} \} \cdot \Pr \{ \text{присутствуют} | \text{пораж.} \} \cdot \Pr \{ \text{отсутствуют} | \text{пораж.} \} \cdot \Pr \{ \text{нет дождя} | \text{пораж.} \}}{\Pr \{ \text{нейтральное, присутствуют, отсутствуют, нет дождя} \}} = \\ = \frac{\frac{2/5 \cdot 1/7 \cdot 1/2 \cdot 1/6 \cdot 1/2}{0.0212 + 0.0075 + 0.0024}}{0.0311} = 0.0767$$

## sms-spam



TN	FN
FP	TP

TN – true negative FP – false positive FN – false negative TP – true positive

FP – *ложные тревоги* (ошибки 1-го рода или  $\alpha$ -ошибки)

FN – *промахи* (ошибки 2-го рода или  $\beta$ -ошибки)

$\alpha = FPR = \frac{FP}{TN+FP}$  – вероятность ошибки первого рода (*уровень значимости*)

$\beta = FNR = \frac{FN}{FN+TP}$  – вероятность ошибки второго рода

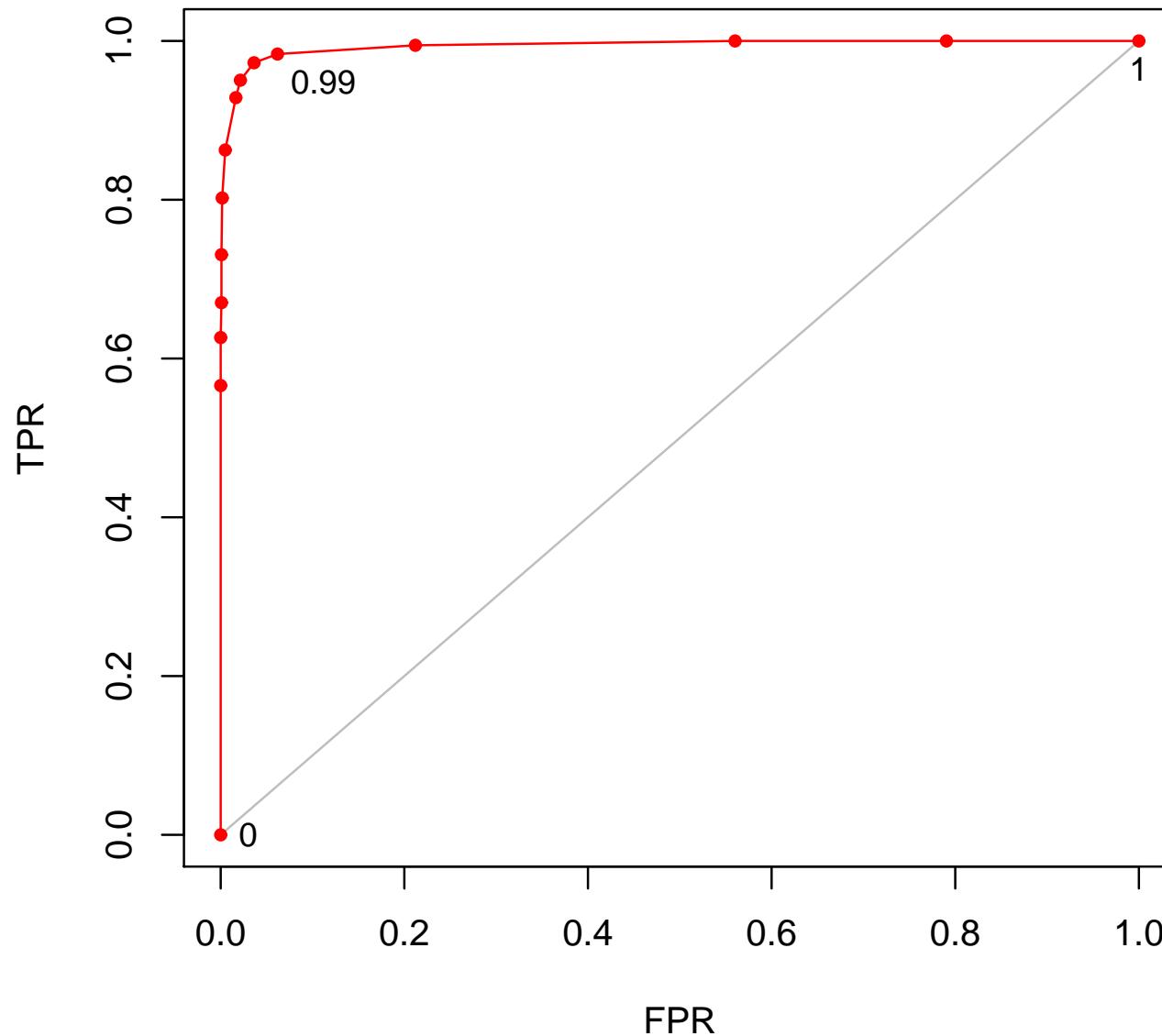
$TNR = \frac{TN}{TN+FP} = 1 - FPR$  – *специфичность* (вероятность предсказать отсутствие болезни, при условии, что ее нет)

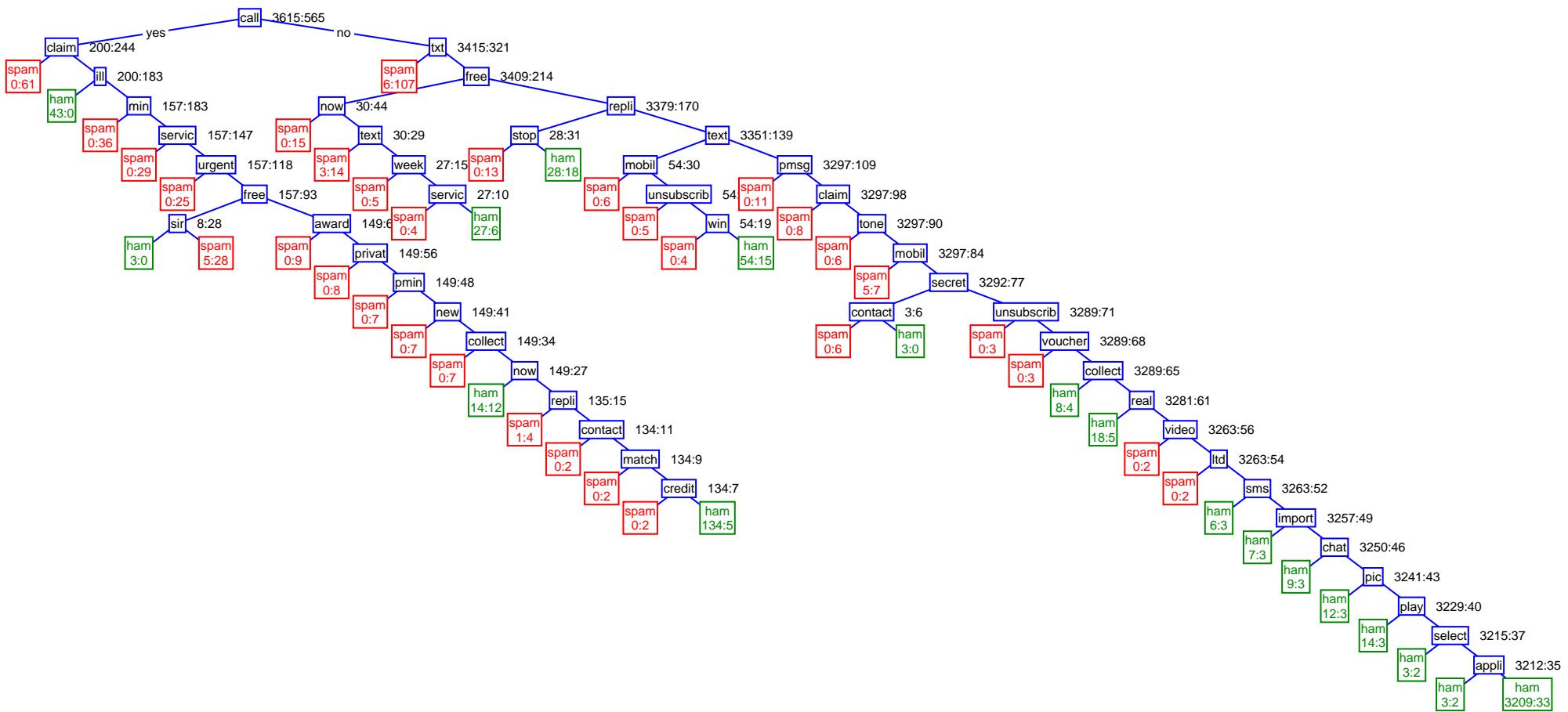
$TPR = \frac{TP}{FN+TP} = 1 - FNR$  – *чувствительность* (вероятность предсказать болезнь, при условии, что она есть)

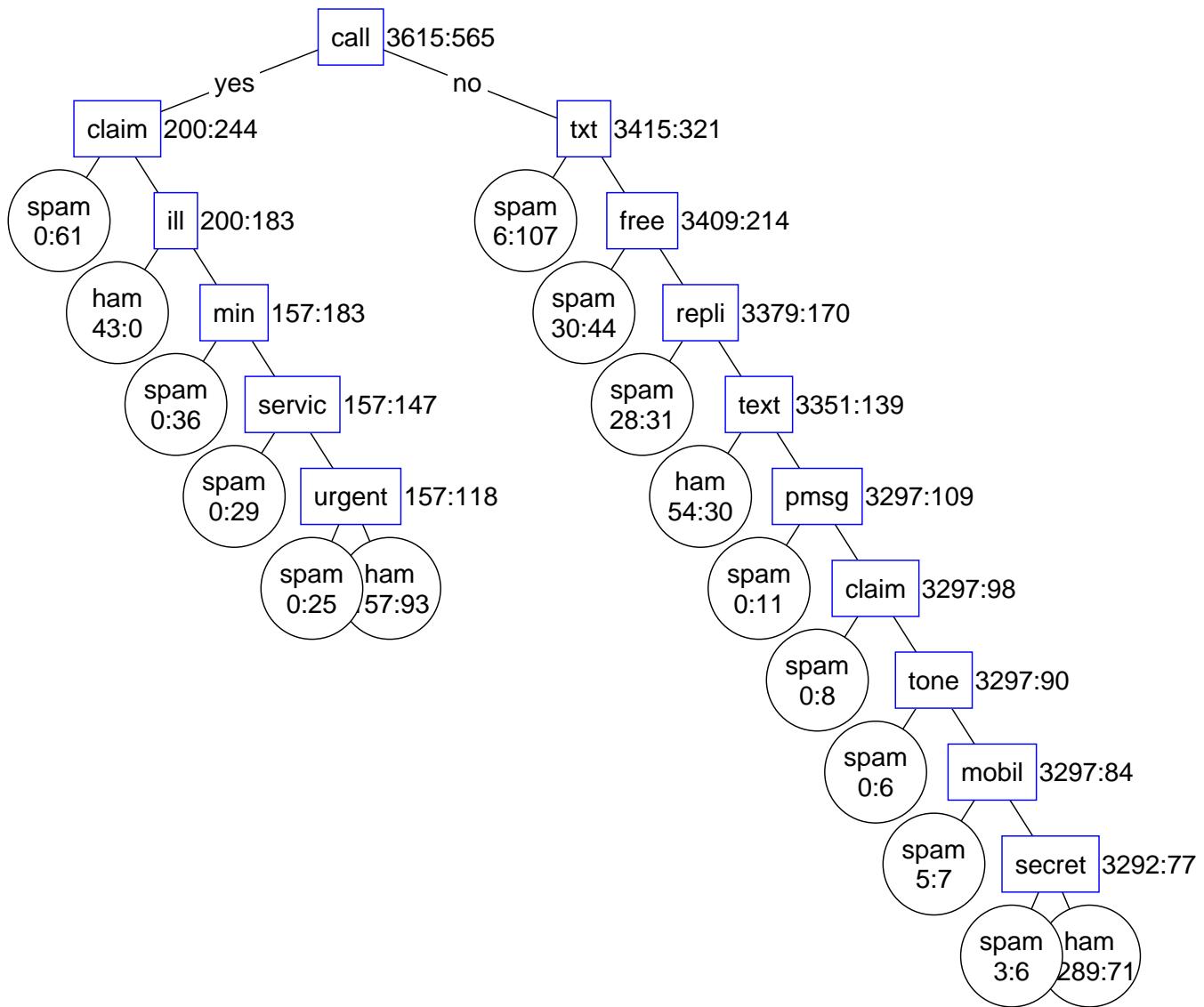
ROC-крявая (receiver operating characteristic) – график зависимости TPR от FPR при изменении порога в дискриминантной функции бинарного классификатора

AUC (area under curve) – площадь под кривой

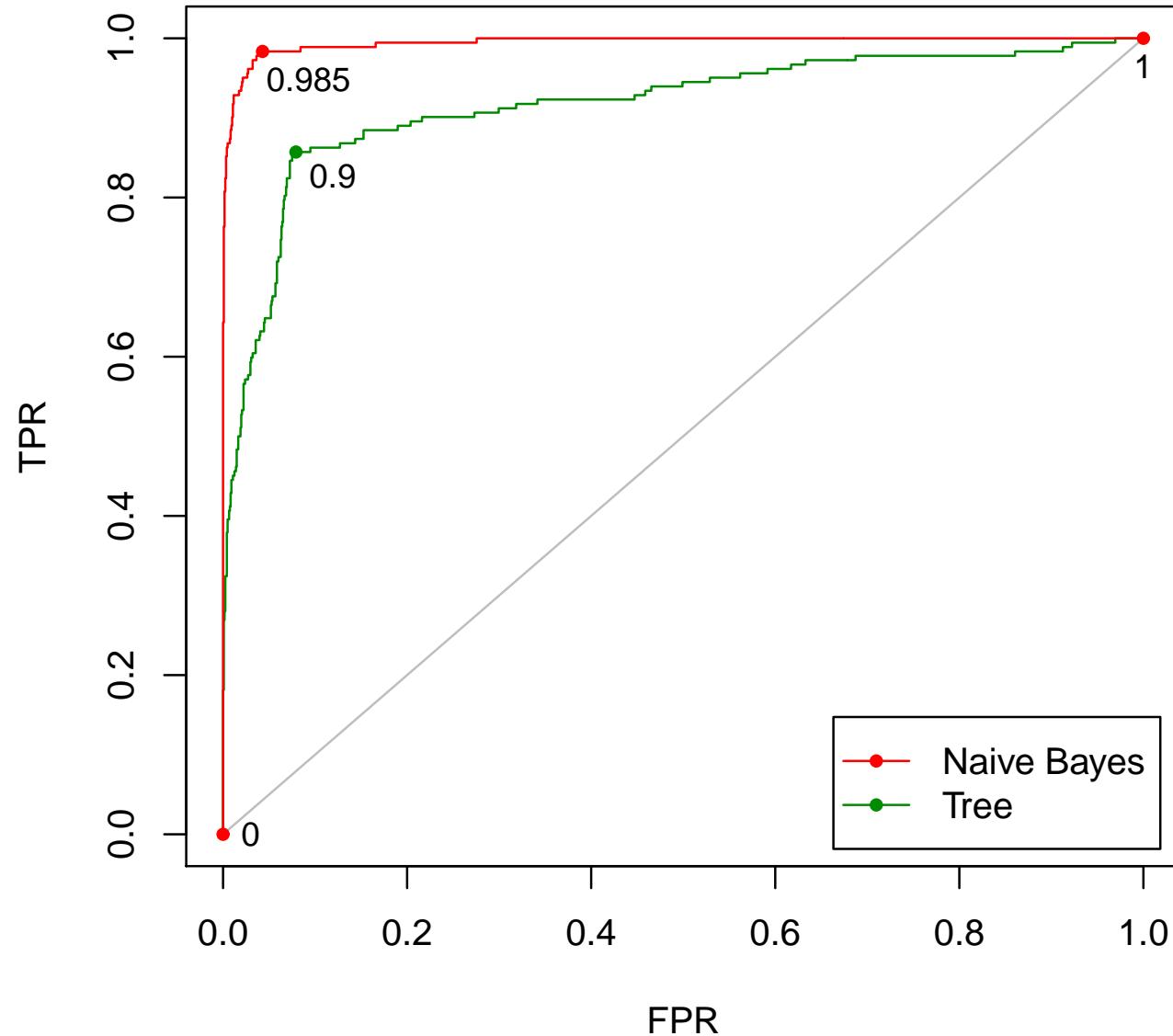
ROC-кривая. Laplace = 1 test.FPR[threshold.seq == 0.99] = 0.04290429 1 -  
test.TPR[threshold.seq == 0.99] = 0.01648352 test.Err[threshold.seq == 0.99] = 0.03945481  
AUC = 0.994646



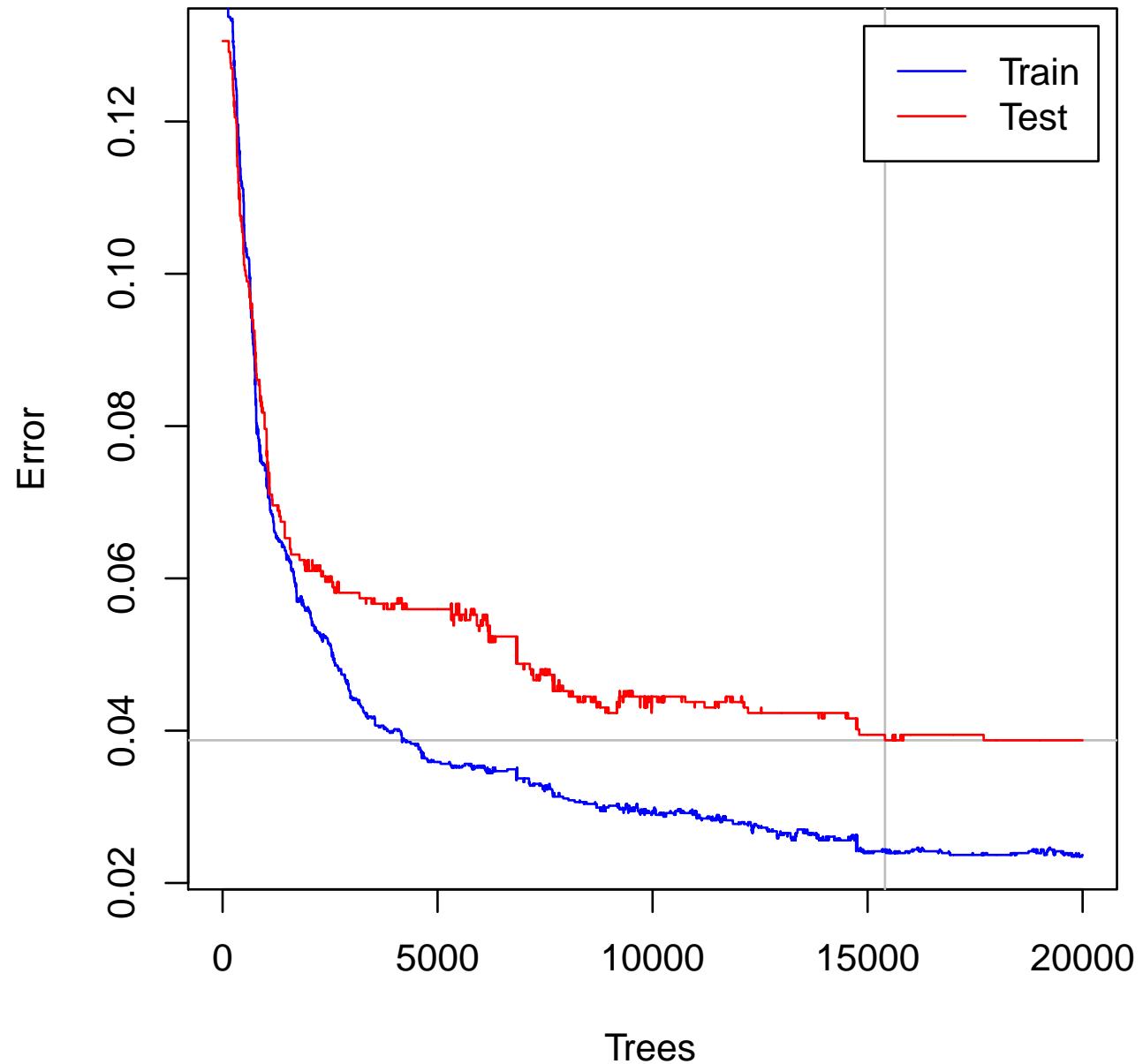


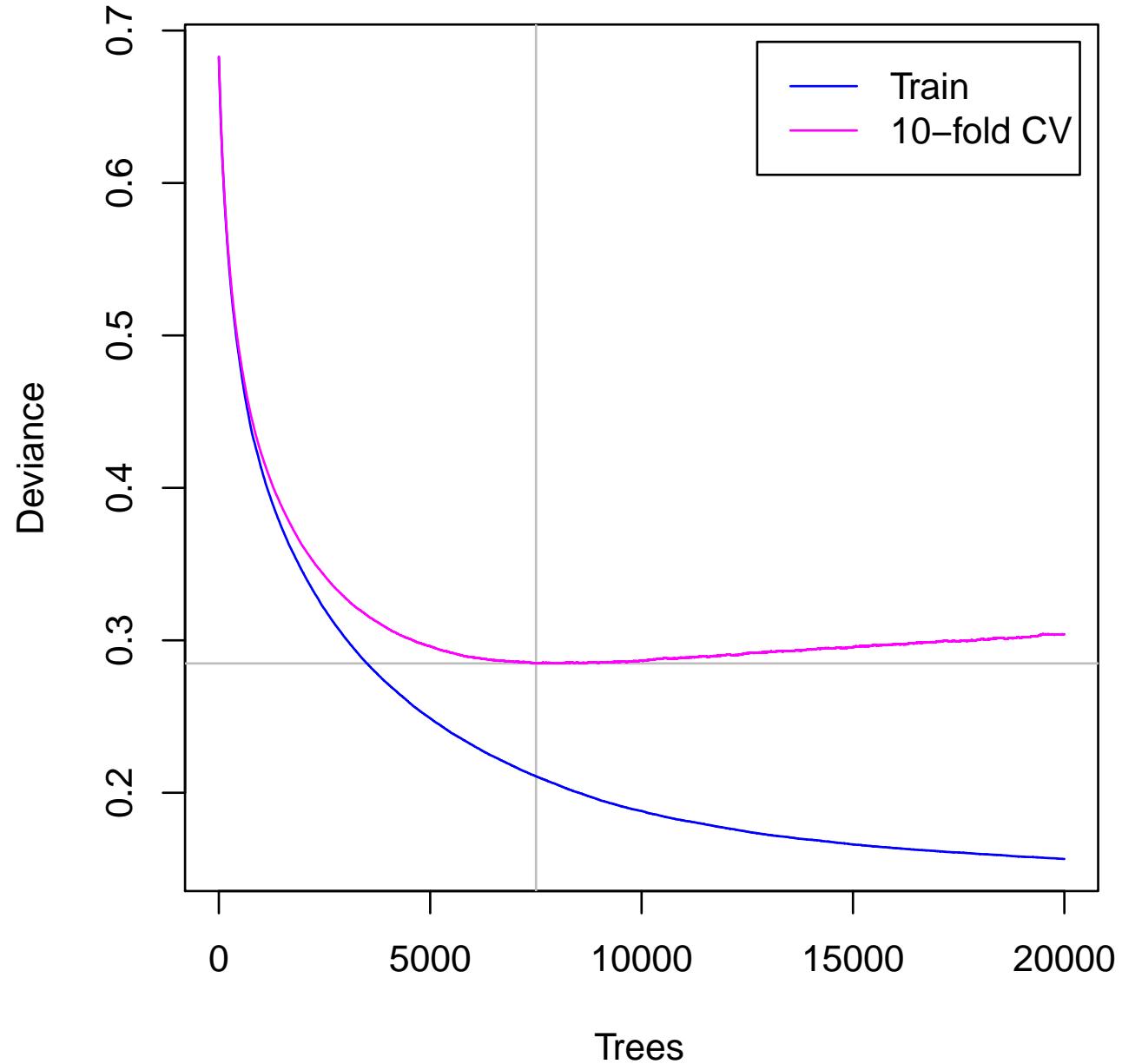


Дерево решений ROC-кривая. tree.test.FPR[tree.threshold.seq == 0.9] = 0.07920792 1 - tree.test.TPR[tree.threshold.seq == 0.9] = 0.1428571 tree.test.Err[tree.threshold.seq == 0.9] = 0.08751793 AUC = 0.916712



GBT





## 4.1.2. Наивный Байес для количественных признаков

Два подхода:

- дискретизация;
- оценка параметров распределения.

Рассмотрим второй подход.

$$\Pr \{Y = y | X = x\} = \frac{p(x) \cdot \Pr y}{p(x)} = \frac{p(x|y) \cdot \Pr y}{\sum_{y'} p(x|y') \cdot \Pr y'}$$

Основное предположение наивного байесова классификатора:

$$p(x|y) = p(x_1|y) p(x_2|y) \dots p(x_d|y)$$

Предположим, что  $p(x_j|y)$  нормальное:

$$p(x_j|y) = \frac{1}{\sqrt{2\pi}\sigma_j} e^{-\frac{(x-\mu_j)^2}{2\sigma_j^2}}$$

Оцениваем  $\mu_j$ ,  $\sigma_j$  по выборке.

## Методы решения задачи классификации:

- Дискриминантные (разделительные) (discriminant):
  - напрямую строим  $f(x)$ ;
  - моделируем (оцениваем) апостериорную вероятность  $\Pr(y|x)$  и используем метод максимума апостериорной вероятности;
- Генеративные (generative) — моделируем (оцениваем) правдоподобие  $p(x|y)$  и априорную вероятность  $\Pr(y)$  и используем метод максимума апостериорной вероятности.

*Внимание:* так называемые методы дискриминантного анализа являются генеративными, а не дискриминантными!

К какому методу относится метод  $k$  ближайших соседей?

## 4.2. Дискриминантный анализ

### 4.2.1. Линейный дискриминантный анализ (LDA)

Нужно сравнить две апостериорные вероятности:

$$\Pr(y \mid x) = \frac{p(x \mid y) \Pr y}{p(x)} > \Pr(y' \mid x) = \frac{p(x \mid y') \Pr y'}{p(x)} \quad (*)$$

Линейный дискриминантный анализ (LDA) делает два предположения:

- объекты каждого класса распределены по нормальному закону:

$$p(x \mid y) = \frac{1}{\sqrt{(2\pi)^d \det \Sigma_y}} e^{-\frac{1}{2}(x - \mu_y)^\top \Sigma_y^{-1} (x - \mu_y)}$$

- матрицы ковариации  $\Sigma = \Sigma_y$  одинаковы для всех классов

Подставляя выражения для  $p(x | y)$  и  $p(x | y')$  в (\*) и логарифмируя:

$$p(x | y) = \frac{1}{\sqrt{(2\pi)^d \det \Sigma_y}} e^{-\frac{1}{2}(x - \mu_y)^\top \Sigma_y^{-1} (x - \mu_y)}$$

$$\Pr(y | x) = \frac{p(x | y) \Pr y}{p(x)} > \Pr(y' | x) = \frac{p(x | y') \Pr y'}{p(x)} \quad (*)$$

$$-\frac{1}{2}(x - \mu_y)^\top \Sigma_y^{-1} (x - \mu_y) + \ln \Pr y > -\frac{1}{2}(x - \mu_{y'})^\top \Sigma_{y'}^{-1} (x - \mu_{y'}) + \ln \Pr y'$$

Откуда  $(\mu_y - \mu_{y'})^\top \Sigma^{-1} x > \frac{1}{2}\mu_y^\top \Sigma^{-1} \mu_y - \frac{1}{2}\mu_{y'}^\top \Sigma^{-1} \mu_{y'} - \ln \Pr y + \ln \Pr y'$

т. е. классы разделяет гиперплоскость  $w^\top x = c$ :  $w^\top x > c$ ,

где  $w = (\mu_y - \mu_{y'})^\top \Sigma^{-1}$ , а  $c$  – некоторая константа.

Введем *линейную дискриминантную функцию*:

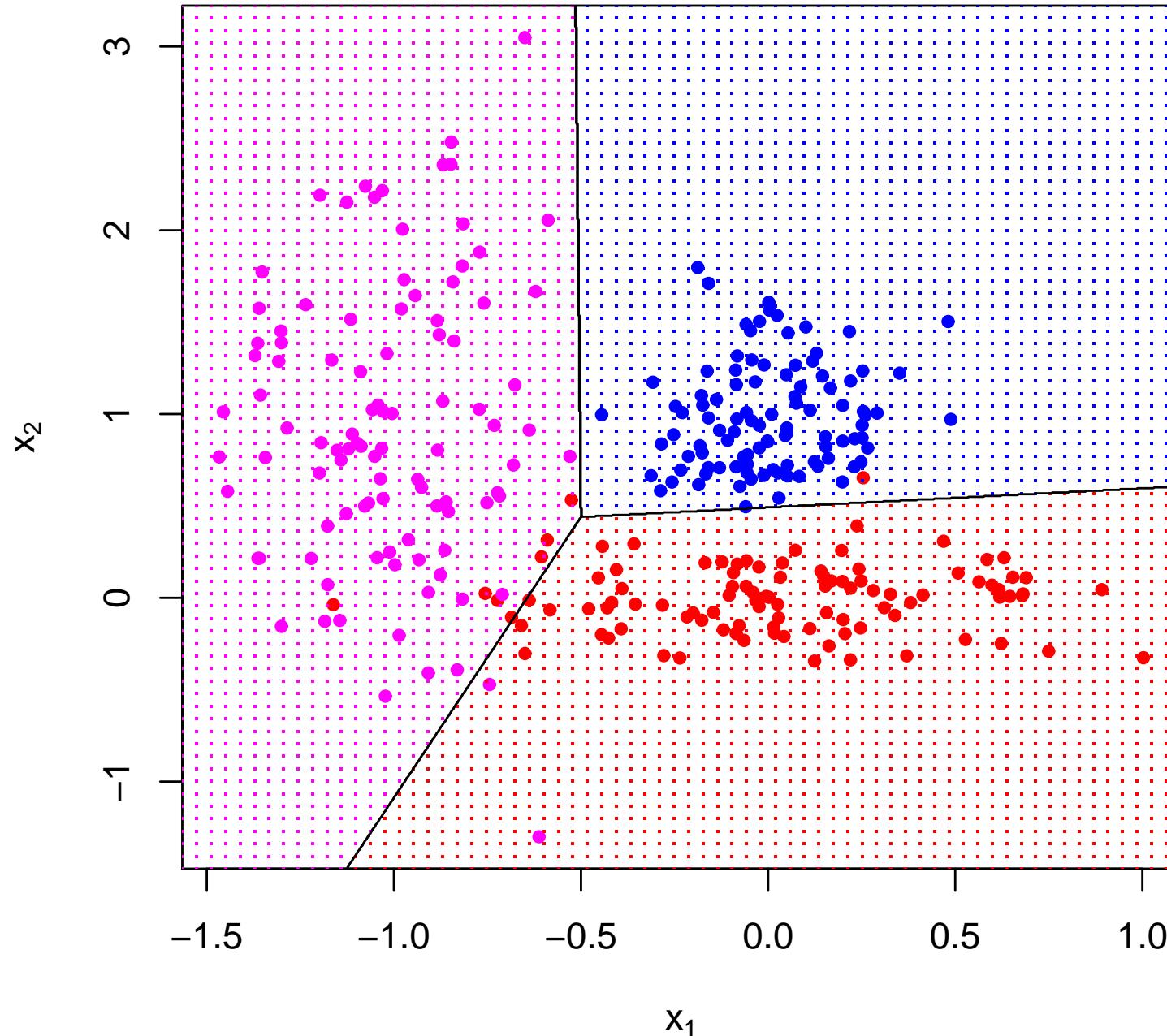
$$\delta_y(x) = -\frac{1}{2}(x - \mu_y)^\top \Sigma_y^{-1} (x - \mu_y) + \frac{1}{2}x^\top \Sigma^{-1} x + \ln \Pr y = \mu_y^\top \Sigma^{-1} x - \frac{1}{2}\mu_y^\top \Sigma^{-1} \mu_y + \ln \Pr y$$

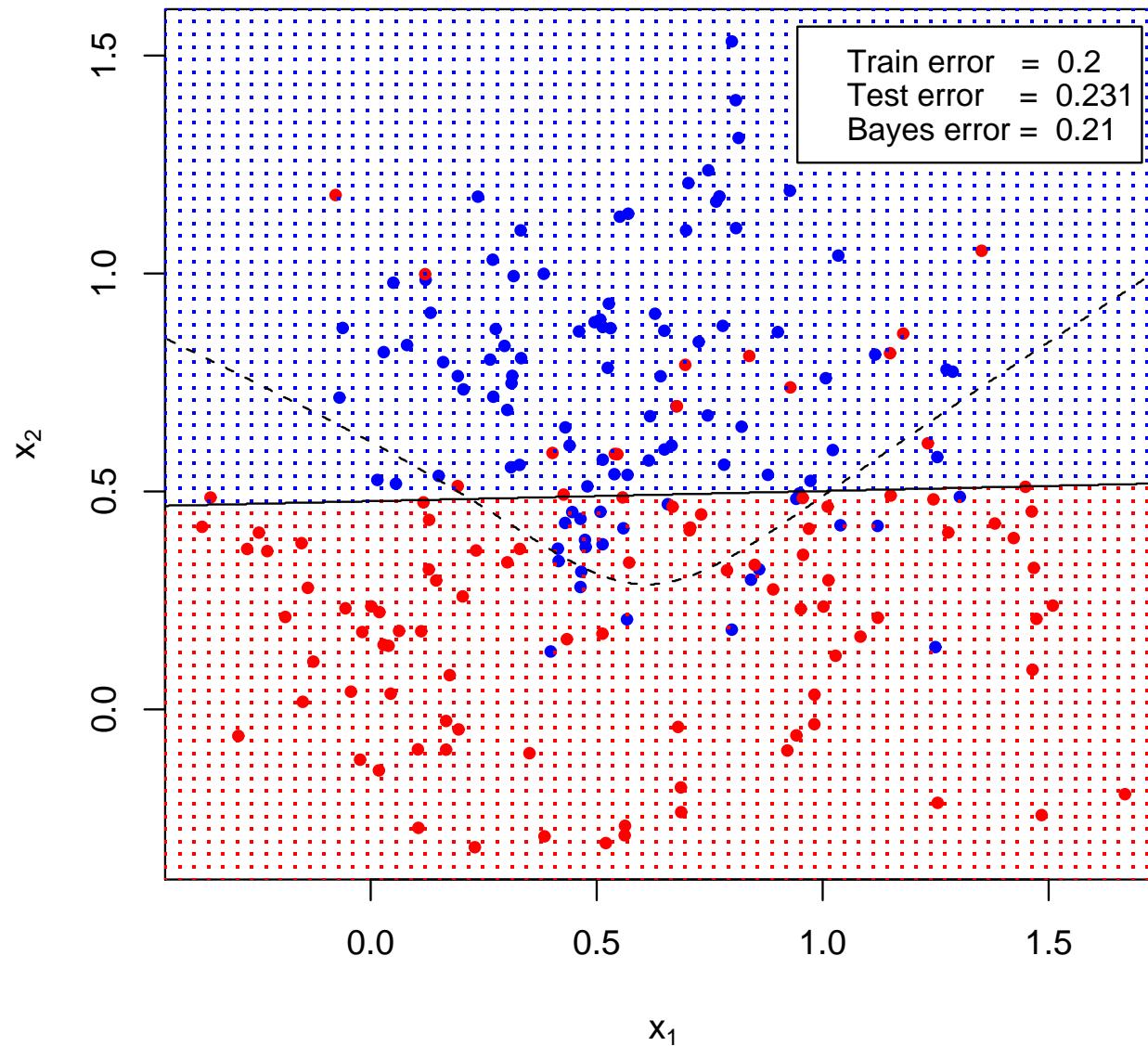
Классификатор:  $f(x) = \operatorname{argmax}_y \delta_y(x)$ .

где  $\rho(x, x') = \sqrt{(x - x')^\top \Sigma^{-1} (x - x')}$  – это *расстояние Махalanобиса*.

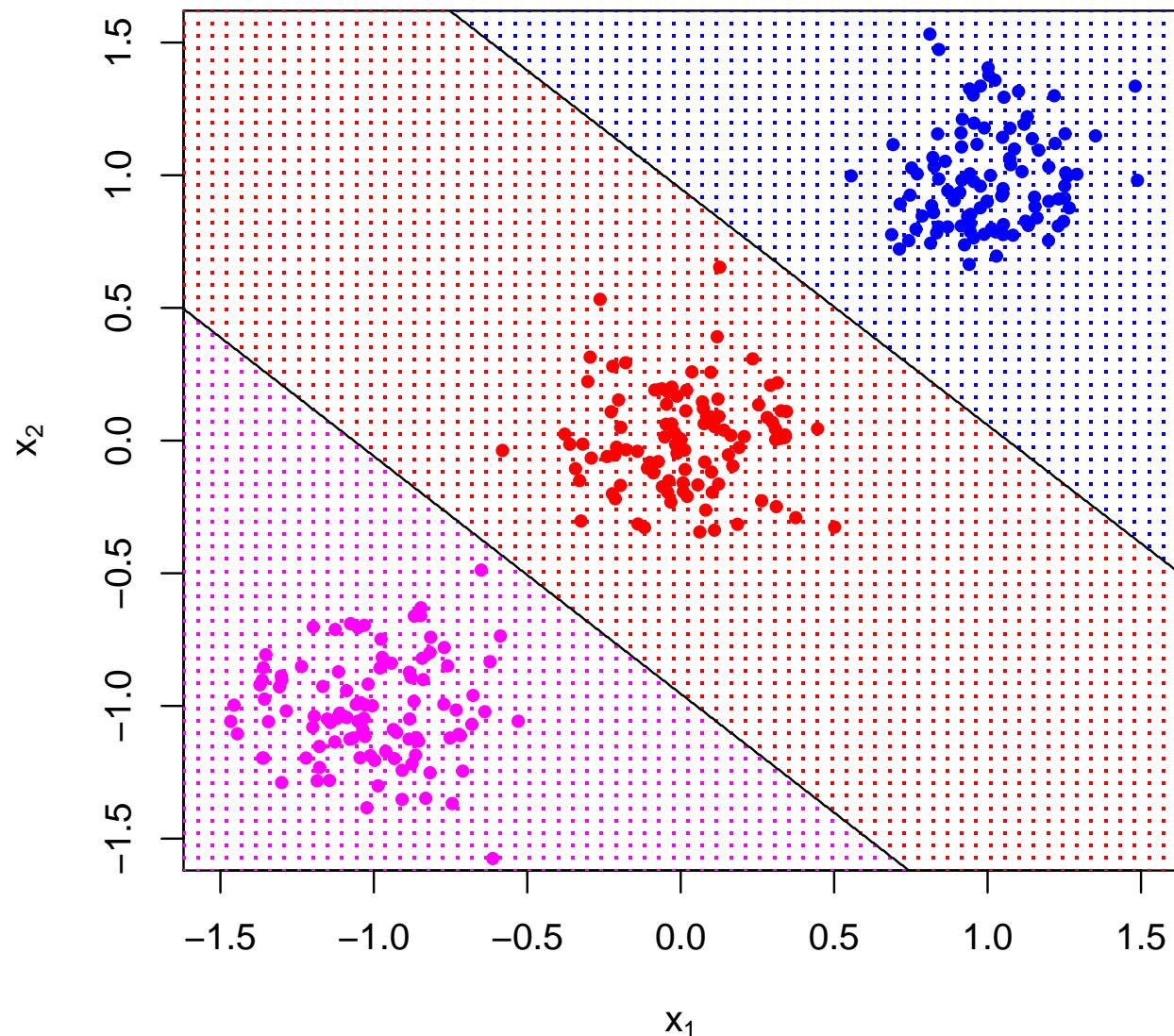
На практике мы не знаем параметров нормального распределения, но можем оценить их по обучающей выборке:

$$\widehat{\Pr}(y) = \frac{N_y}{N}, \quad \widehat{\mu}_y = \sum_{y^{(i)}=k} \frac{x^{(i)}}{N_y}, \quad \widehat{\Sigma} = \frac{1}{N - K} \sum_{k=1}^K \sum_{y^{(i)}=k} (x^{(i)} - \mu_k)(x^{(i)} - \mu_k)^\top.$$





# LDA



**Замечание 4.1** Пусть также  $\hat{\beta}_0, \hat{\beta}$  — параметры, полученные с помощью линейной регрессии с критерием

$$\sum_{i=1}^N (y^{(i)} - \beta_0 - \beta^\top x^{(i)})^2 \rightarrow \min.$$

Можно показать, что вектор  $\hat{\beta}$  коллинеарен вектору  $\hat{\Sigma}^{-1}(\hat{\mu}_2 - \hat{\mu}_1)$ .

Таким образом, разделяющие гиперплоскости, полученные с помощью линейной регрессии и методом *LDA*, параллельны (но при  $N_1 \neq N_2$  не совпадают друг с другом).

Это уже не так при числе классов, большем 2.

Если имеется более 2-х классов, то, напомним, линейная регрессия при классификации может «не замечать» некоторых из них.

*LDA*-метод свободен от этого недостатка.

## 4.2.2. Квадратичный дискриминантный анализ

Рассмотрим теперь

$$p(x|y) = \frac{1}{(2\pi)^{d/2} \sqrt{\det \Sigma_y}} e^{-\frac{1}{2}(x-\mu_y)^\top \Sigma_y^{-1}(x-\mu_y)},$$

не предполагая, что  $\Sigma_y$  равны между собой.

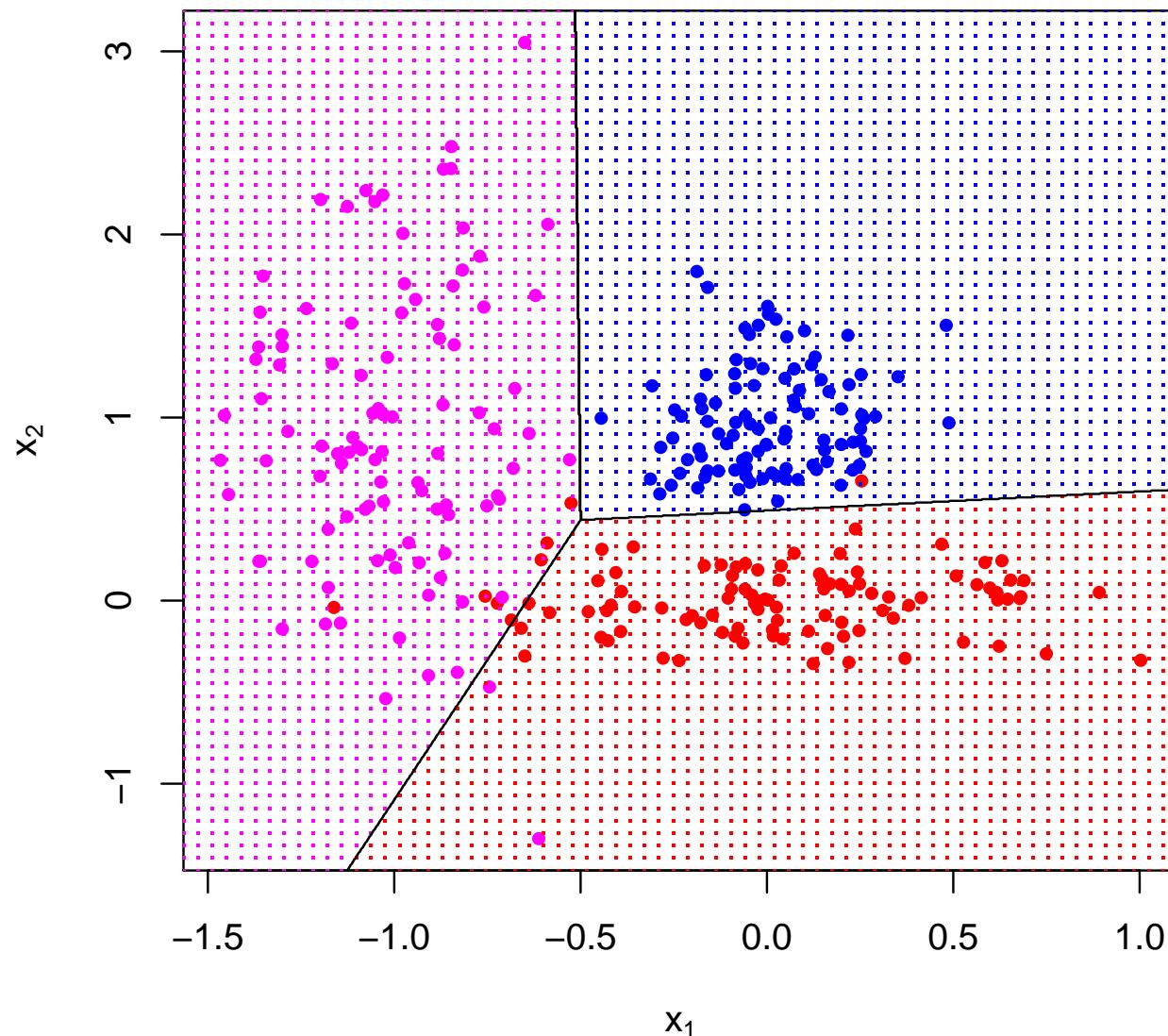
Проводя аналогичные рассуждения, придем к понятию *квадратичной дискриминантной функции*

$$\delta_y(x) = -\frac{1}{2} \ln \det \Sigma_y - \frac{1}{2}(x - \mu_y)^\top \Sigma_y^{-1}(x - \mu_y) + \ln \Pr(y)$$

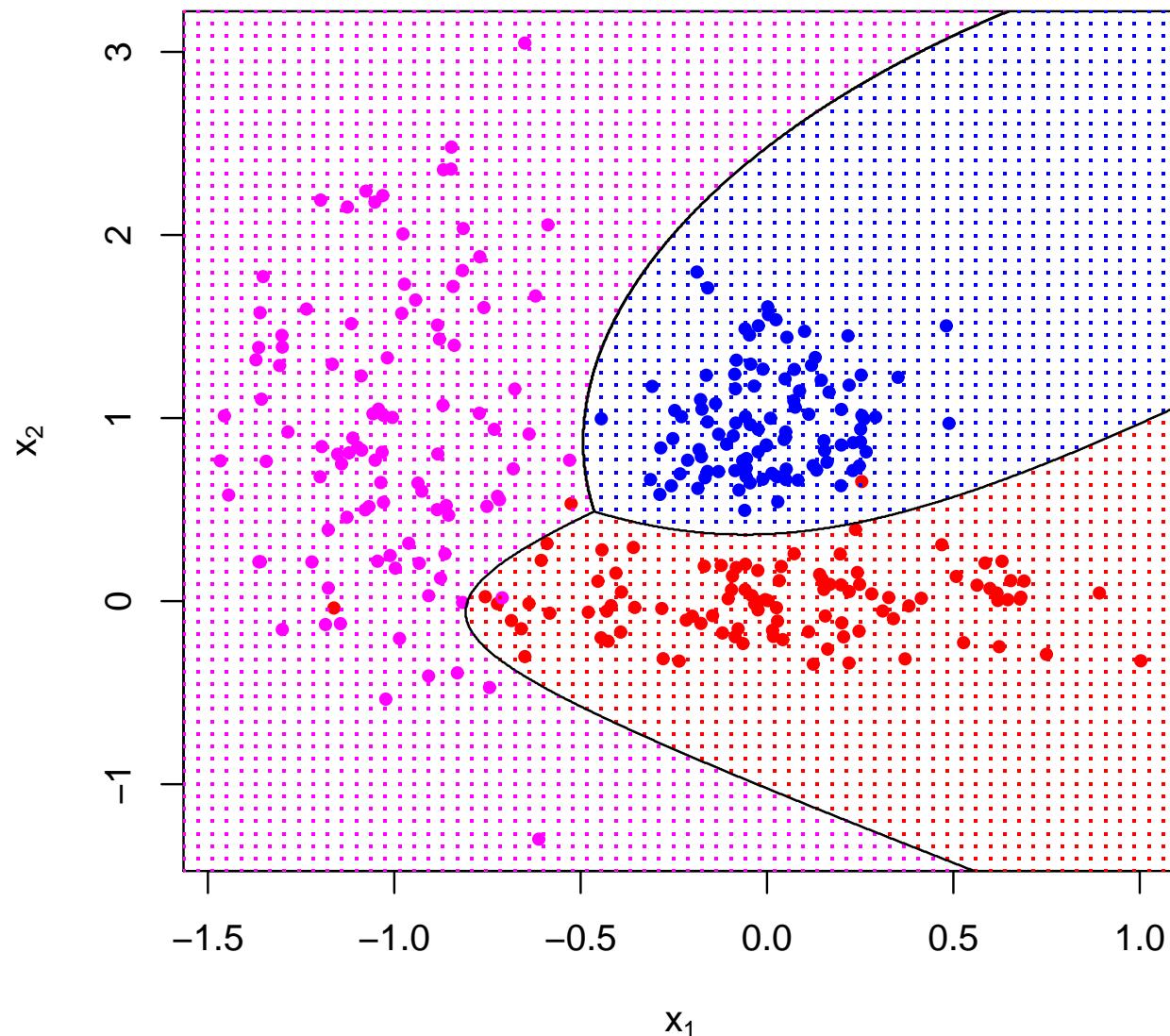
Поверхность, разделяющая два класса  $y$  и  $k$  описывается уравнением 2-го порядка  $\delta_y(x) = \delta_k(x)$ .

*Квадратичным дискриминантным анализом (quadratic discriminant analysis, QDA)* называется построение решающего правила с помощью квадратичной дискриминантной функции.

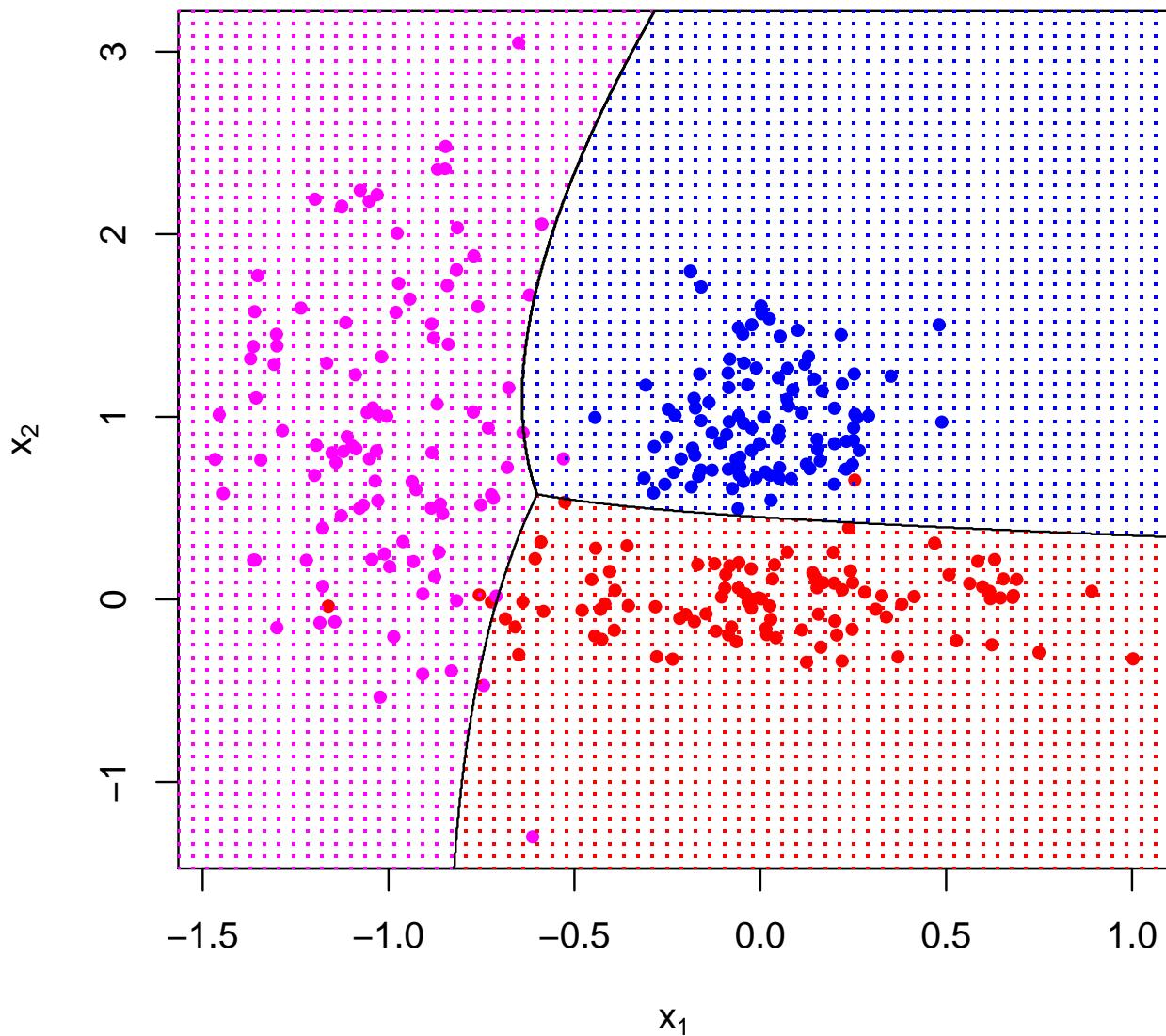
# LDA



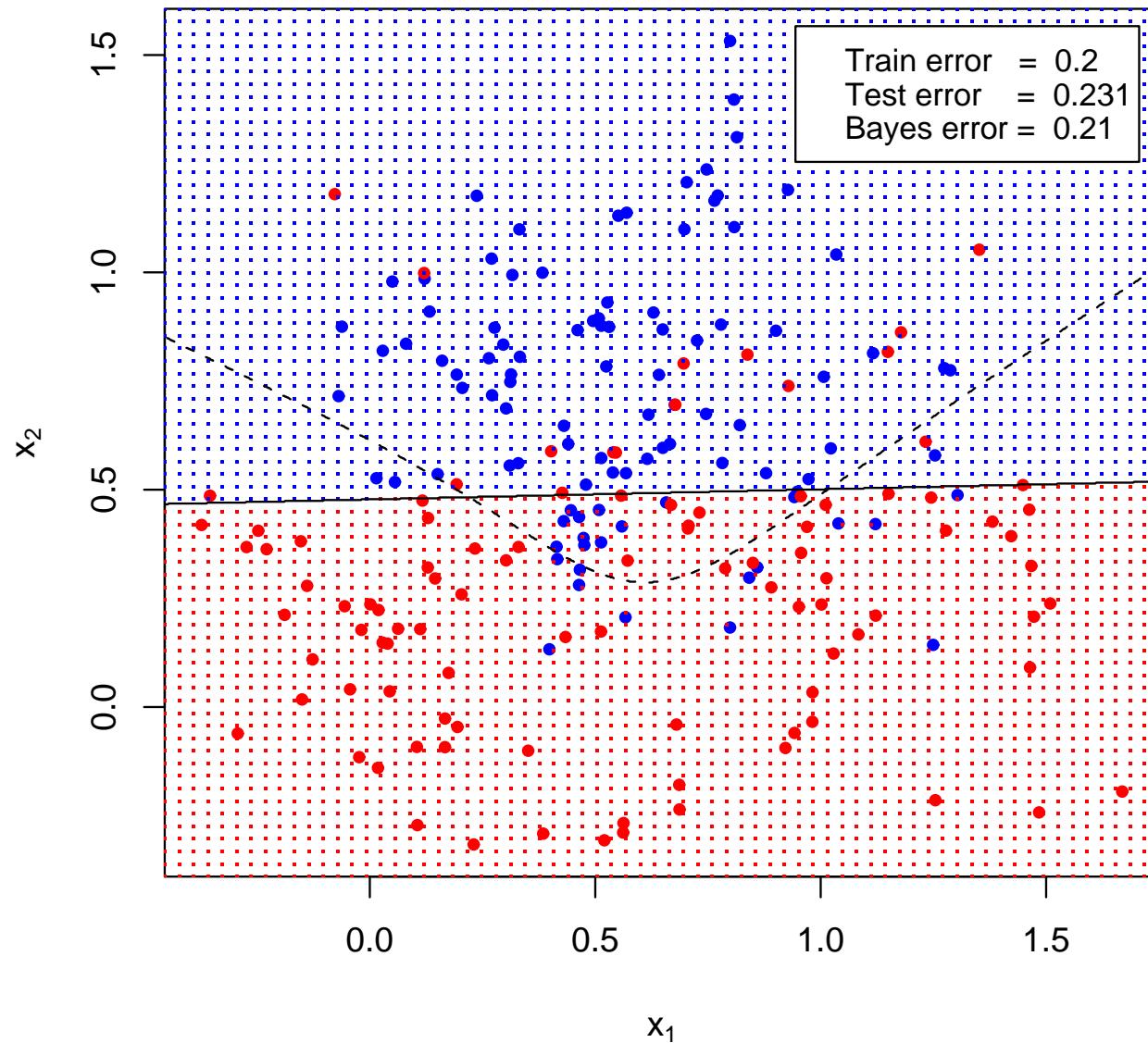
# QDA



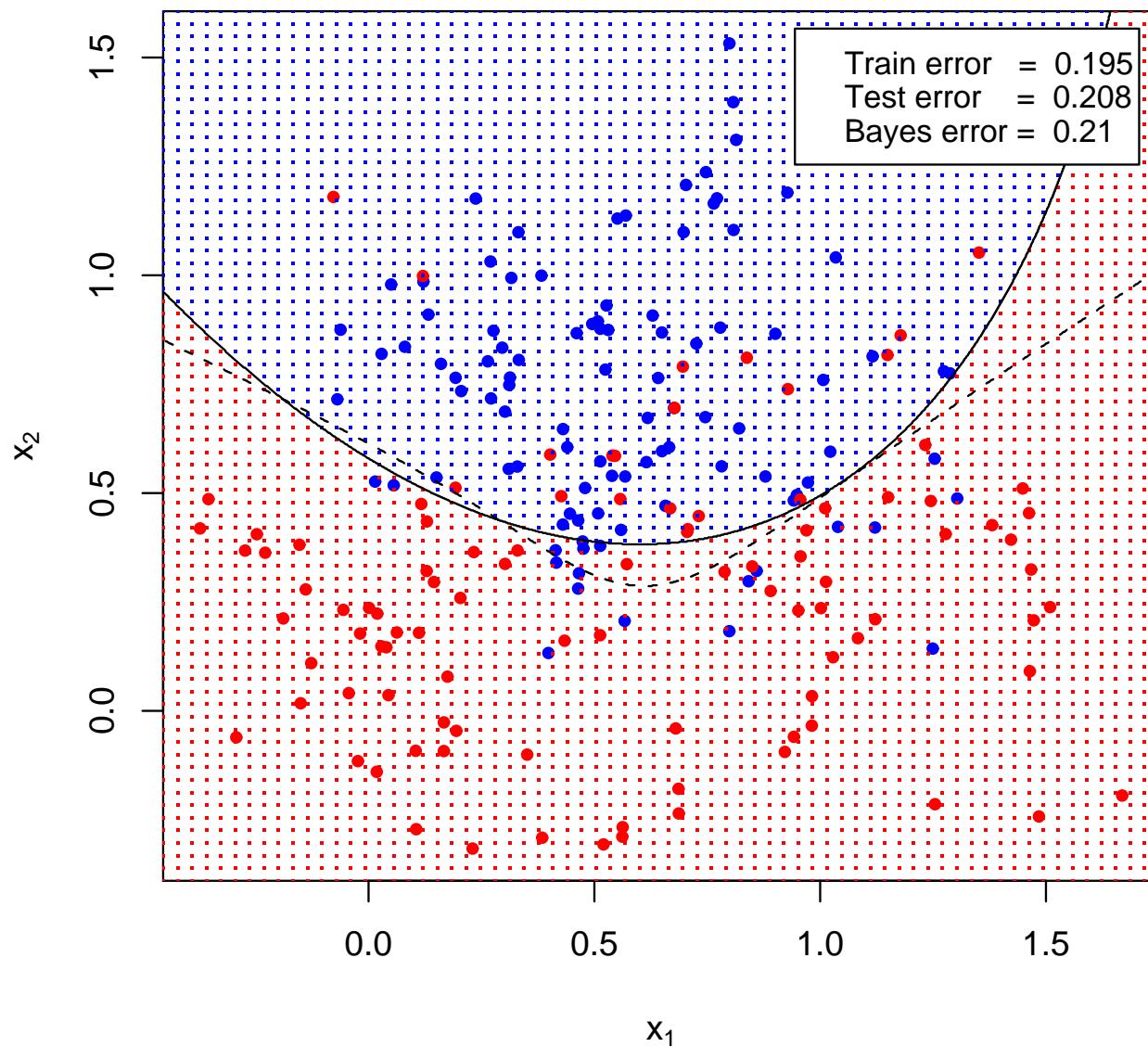
$$\text{LDA } y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2 + \beta_{11} x_1^2 + \beta_{22} x_2^2$$



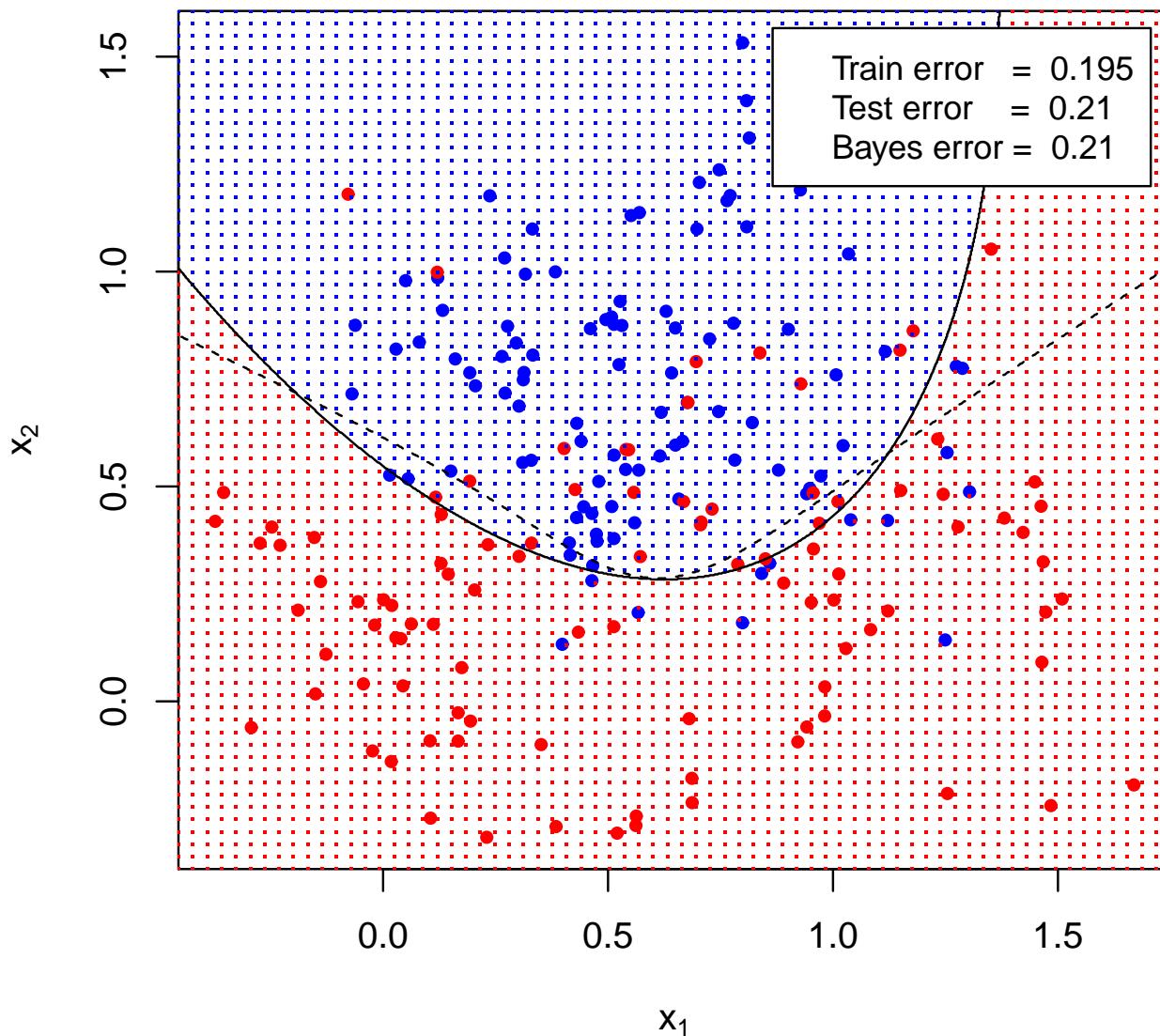
# LDA



# QDA



$$\text{LDA } y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_{12} x_1 x_2 + \beta_{11} x_1^2 + \beta_{22} x_2^2$$



### 4.2.3. Другой взгляд на LDA

Максимизация отношения межклассовой дисперсии к внутриклассовой эквивалентна LDA.

Максимизация соотношения Релея:

$$\max_a \frac{a^\top \mathbf{B} a}{a^\top \mathbf{W} a}$$

или

$$\max_a a^\top \mathbf{B} a \quad \text{при ограничении } a^\top \mathbf{W} a = 1$$

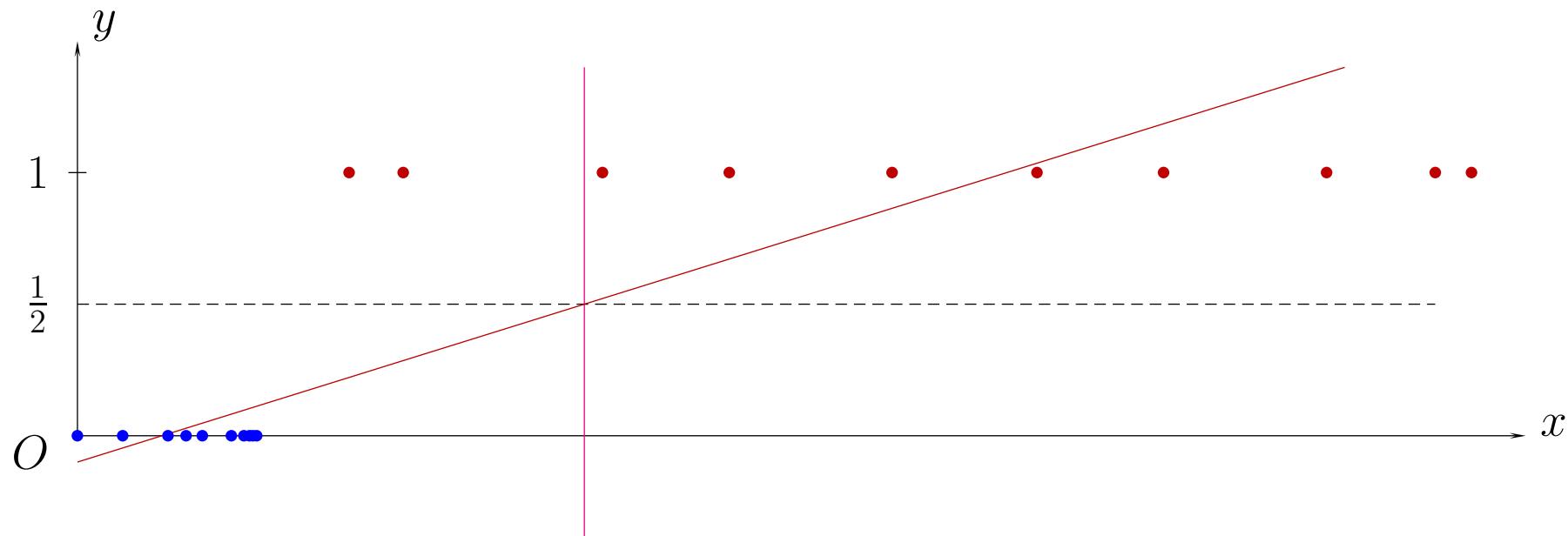
— это *обобщенная задача на собственные значения*

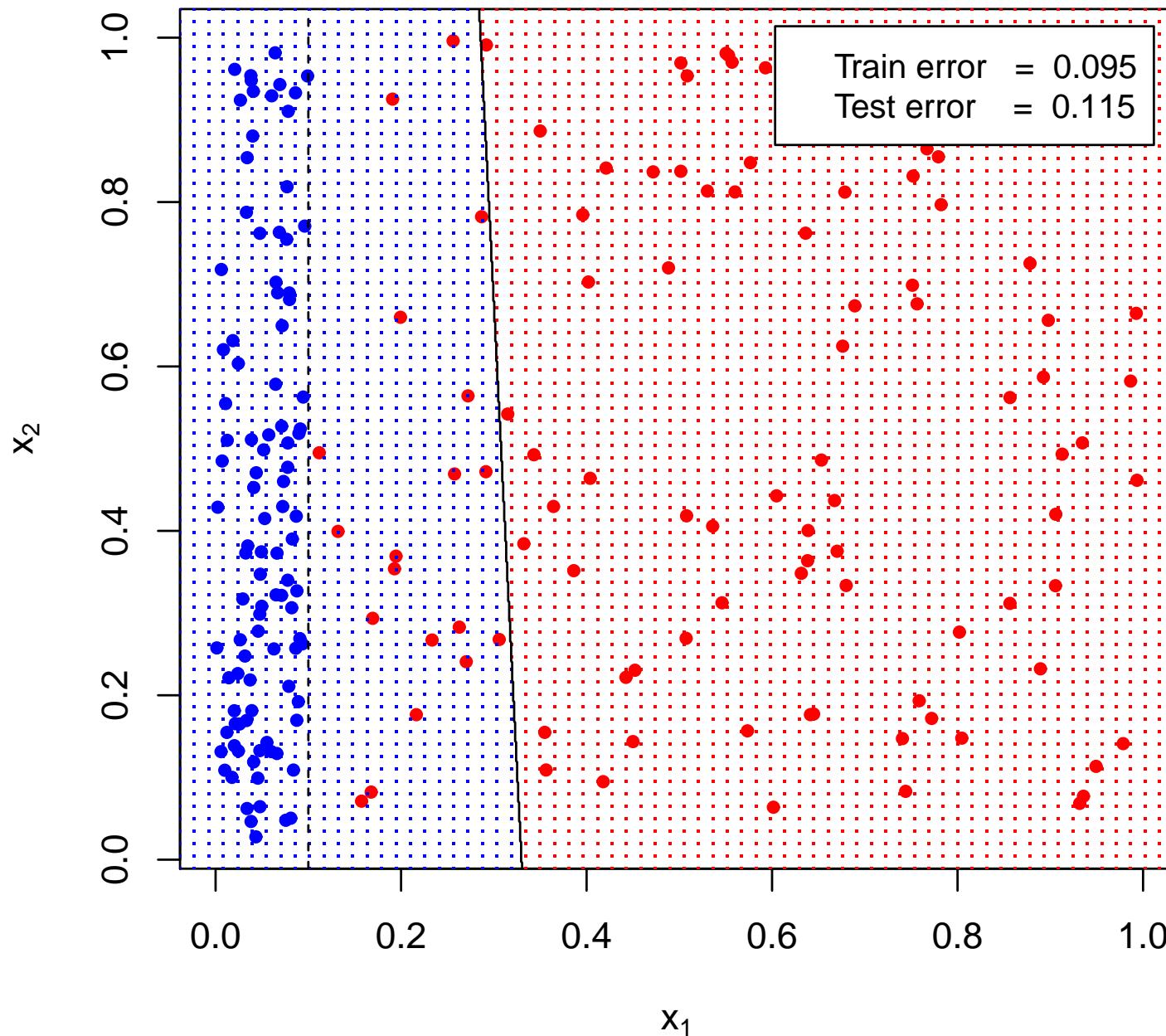
## 4.3. МНК для задачи классификации

(дискриминантный метод)

Нельзя ли использовать линейную регрессию (МНК) для решения задачи классификации?

$$K = 2$$





$K > 2$

Сопоставим каждому классу  $k$  *характеристический, или индикаторный, вектор*  $(y_1, y_2, \dots, y_K)$ , в котором  $y_k = 1$ , а  $y_i = 0$  при  $i \neq k$ .

Собрав вместе индикаторные векторы объектов обучающей выборки, получим матрицу  $\mathbf{Y}$  размера  $N \times K$ , называемую *индикаторной*.

Таким образом,  $\mathbf{Y}$  состоит только из нулей и единиц и каждая строка имеет ровно одну единицу.

Как обычно,  $\mathbf{X}$  – матрица размера  $N \times (d + 1)$ , первый столбец которой состоит из единиц, а последующие представляют собой векторы из обучающей выборки.

Применяя метод наименьших квадратов одновременно к каждому столбцу матрицы  $\mathbf{Y}$ , получаем значения

$$\widehat{\mathbf{Y}} = \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}.$$

Для каждого столбца  $y_k$  матрицы  $\widehat{\mathbf{Y}}$  получим свой столбец коэффициентов  $\widehat{\beta}_k$ . Соберем их в матрицу  $\widehat{\mathbf{B}}$  размера  $(d + 1) \times K$ .

Имеем

$$\widehat{\mathbf{B}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}.$$

Объект  $x$  будем классифицировать согласно следующему правилу:

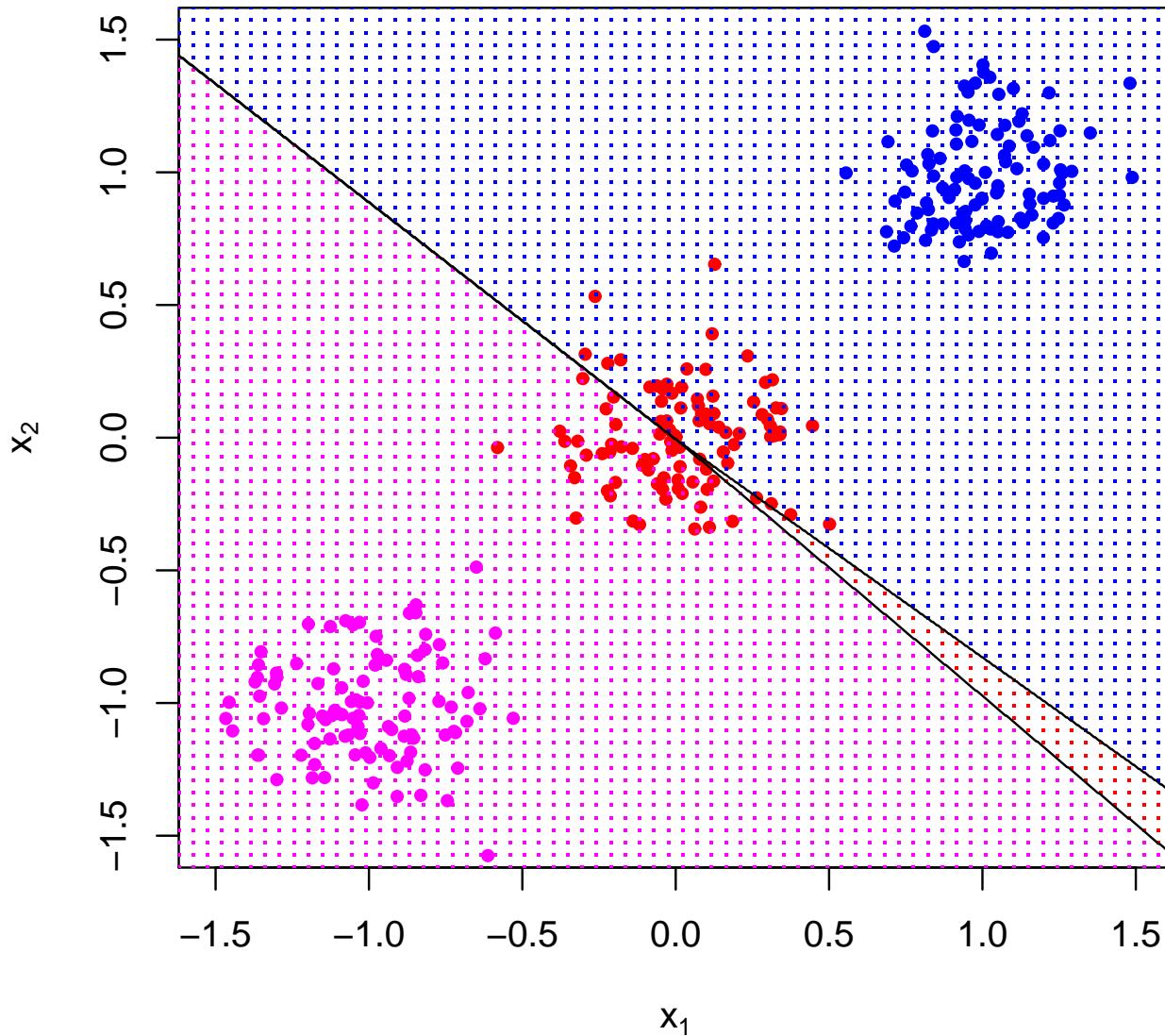
Вычислим вектор-строку длины  $K$

$$g(x) = (1, x) \widehat{\mathbf{B}}.$$

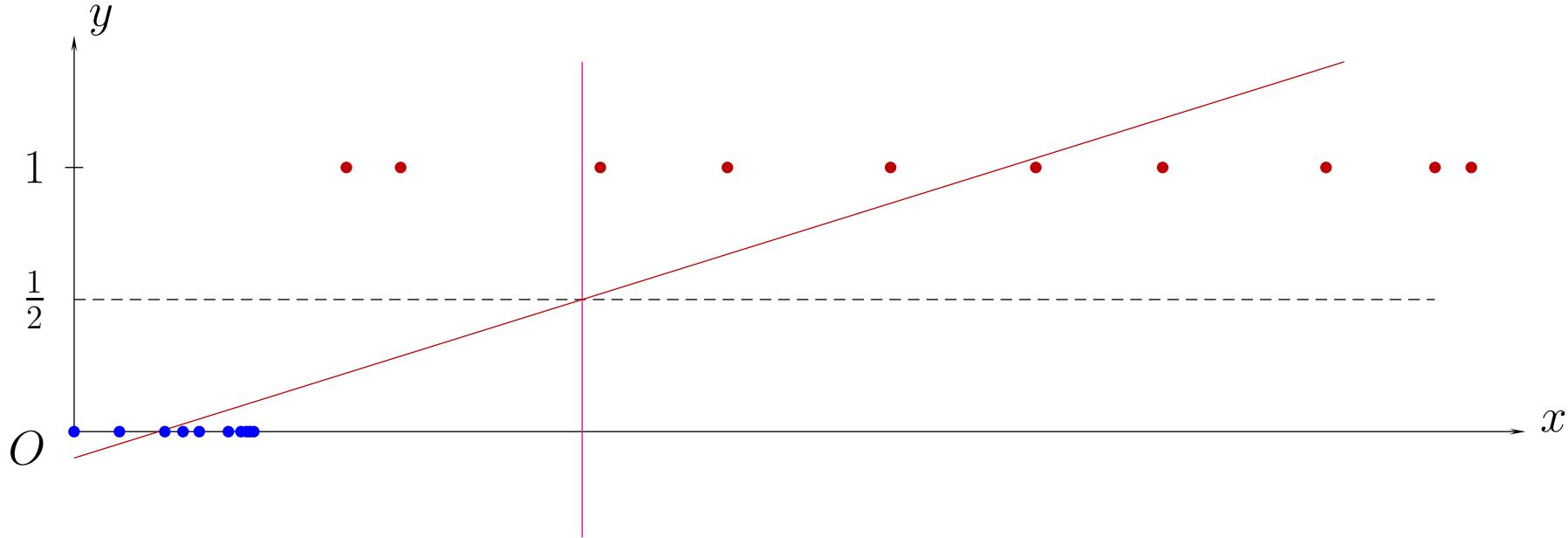
Отнесем  $x$  к классу

$$f(x) = \operatorname{argmax}_k g_k(x).$$

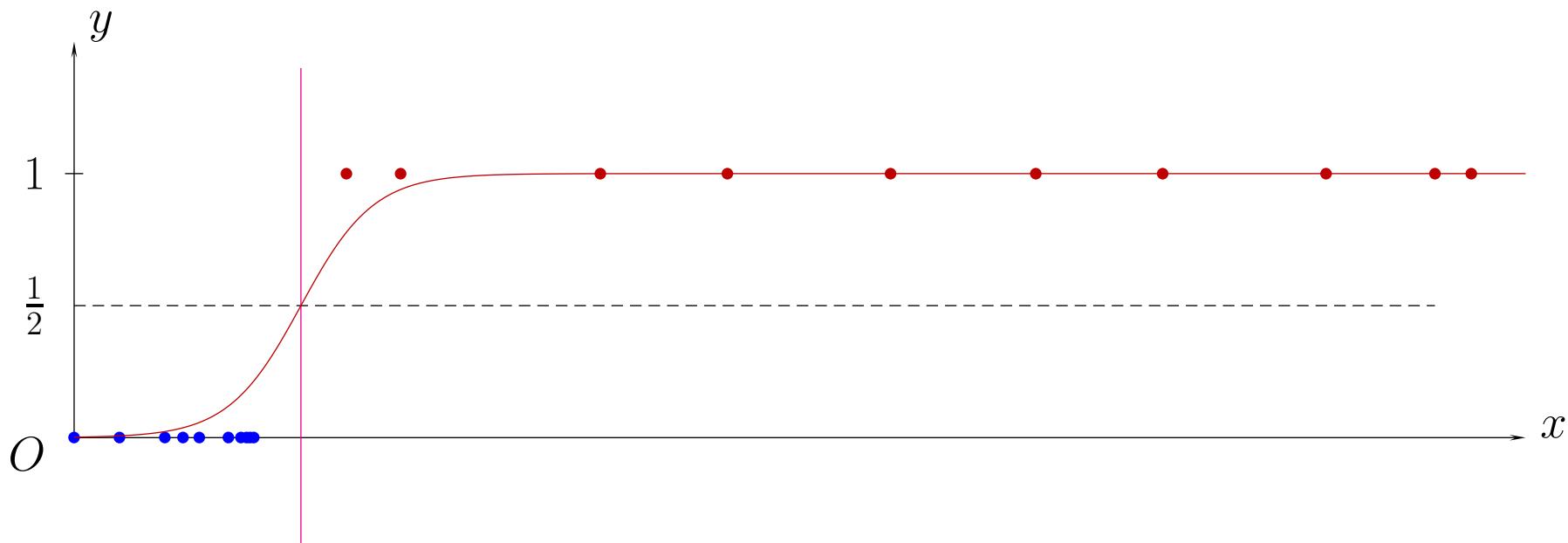
При  $K \geq 3$  линейная регрессия может «не замечать» некоторых хорошо отделенных классов.



МНК:



Хочется:



## 4.4. Логистическая регрессия

(Дискриминантный метод)

Рассмотрим задачу классификации на 2 класса:  $\mathcal{Y} = \{0, 1\}$

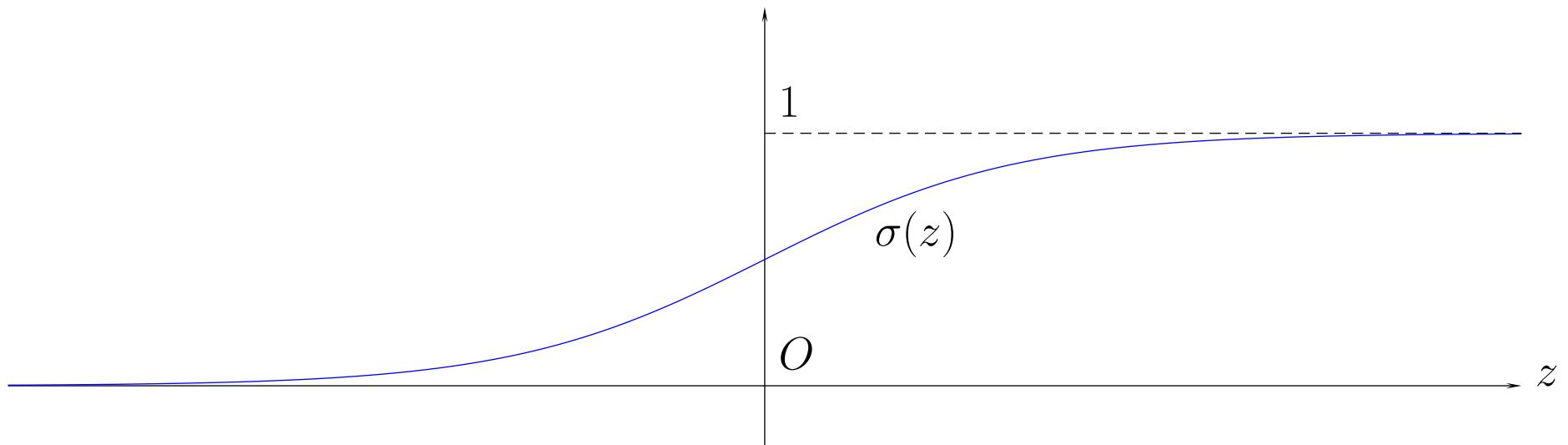
Пусть

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

где

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

— логистическая функция (элементарный сигмоид или логит-функция)



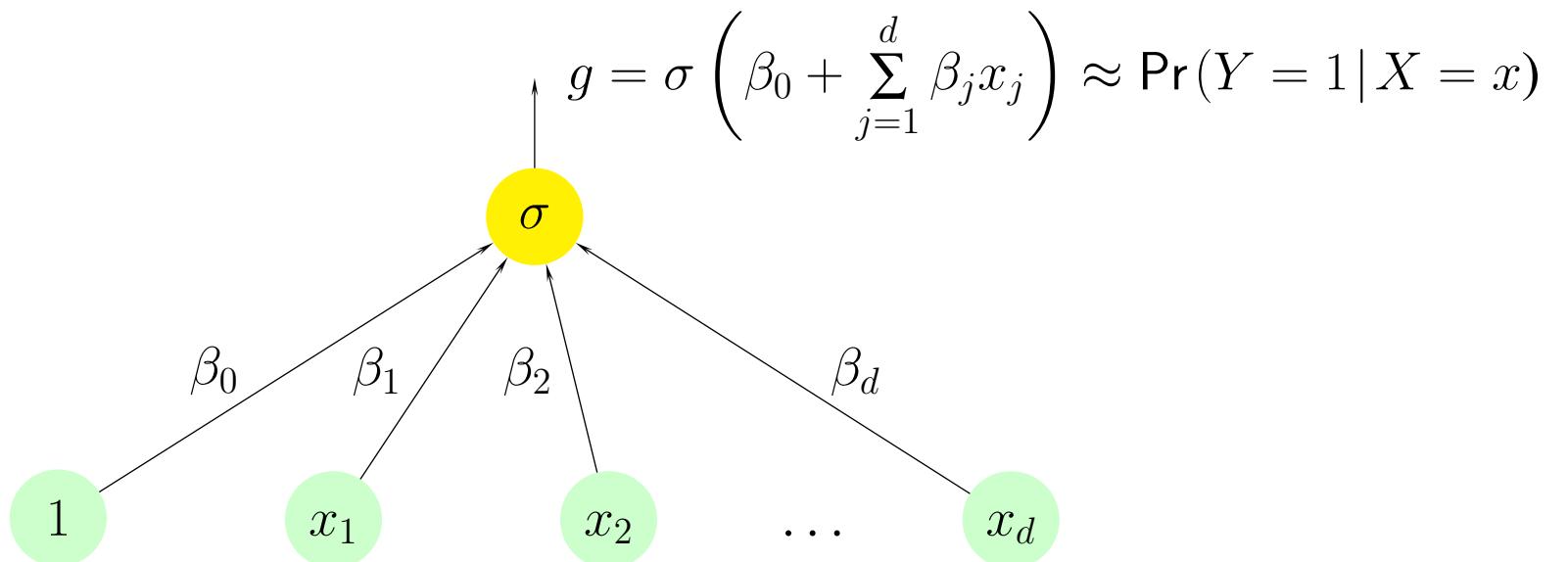
$$\Pr(Y = 1 | x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \dots + \beta_d x_d)}} = \sigma(\beta_0 + \beta^\top x),$$

тогда

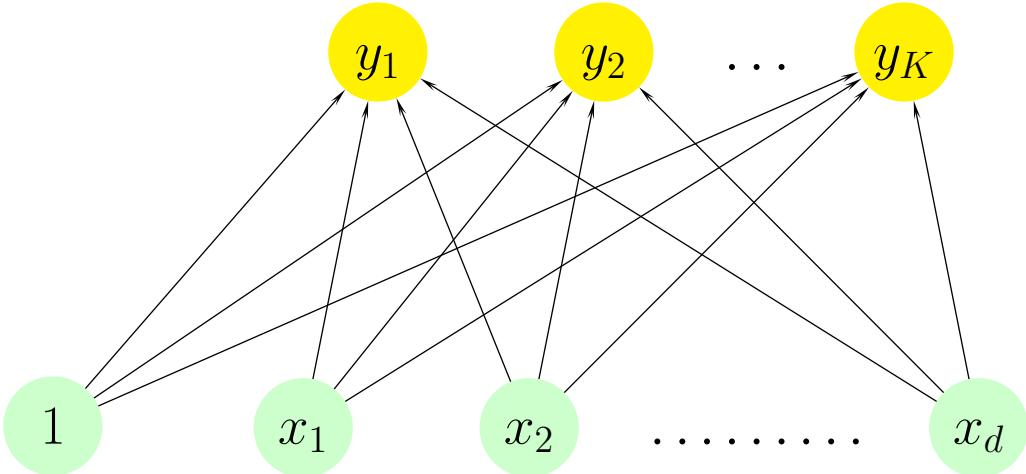
$$\Pr(Y = 0 | x) = 1 - \Pr(Y = 1 | x) = \frac{1}{1 + e^{\beta_0 + \beta_1 x_1 + \dots + \beta_d x_d}} = \sigma(-\beta_0 - \beta^\top x),$$

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:



$$y_k = \frac{\exp\left(\beta_{k0} + \sum_{j=1}^d \beta_{kj} x_j\right)}{\sum_{\ell=1}^K \exp\left(\beta_{\ell0} + \sum_{j=1}^d \beta_{\ell j} x_j\right)} \approx \Pr(k|x) \quad (k = 1, 2, \dots, K)$$

Разделяющие поверхности (между каждой парой классов) снова линейные:

$$\Pr(Y = k | X = x) = \Pr(Y = k' | X = x) \Leftrightarrow \beta_{k0} + \beta_k^T x = \beta_{k'0} + \beta_{k'}^T x$$

**Замечание 4.2** Можно преобразовать и переобозначить:

$$\Pr(Y = k | X = x) = \frac{e^{\beta_{k0} + \beta_k^T x}}{1 + \sum_{\ell=1}^{K-1} e^{\beta_{\ell0} + \beta_{\ell}^T x}} \quad (k = 1, 2, \dots, K-1),$$

$$\Pr(Y = K | X = x) = \frac{1}{1 + \sum_{\ell=1}^{K-1} e^{\beta_{\ell0} + \beta_{\ell}^T x}}$$

#### 4.4.1. Расчет параметров

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

В логистической регрессии параметры обычно подбираются с помощью метода максимального правдоподобия.

Максимизации подвергается логарифмическая функция правдоподобия

$$\ell(\beta) = \sum_{i=1}^N \ln \Pr \left\{ Y = y^{(i)} \mid X = x^{(i)}, \beta \right\} \rightarrow \max,$$

где

$$\beta = (\beta_{10}, \beta_1, \beta_{20}, \beta_2, \dots, \beta_{K-1,0}, \beta_{K-1}), \quad \Pr(Y = k \mid X = x, \beta) = \Pr(Y = k \mid X = x).$$

Возможна регуляризация:

$$\sum_{i=1}^N \ln \Pr \left\{ Y = y^{(i)} \mid X = x^{(i)}, \beta \right\} - \lambda \|\beta\|^2 \rightarrow \max$$

где вместо квадрата евклидовой нормы можно рассматривать 1-норму  $\|\beta\|_1$

Вначале рассмотрим случай  $K = 2$ :  $\mathcal{Y} = \{0, 1\}$ .

$$g(x, \beta) = \Pr \{Y = 1 \mid X = x, \beta\} = \frac{1}{1 + e^{-\beta_0 - \beta^\top x}}.$$

Требуется максимизировать логарифмическую функцию правдоподобия

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

**Упражнение 4.3** Докажите, что

$$\frac{\partial \ell(\beta)}{\partial \beta_0} = \sum_{i=1}^N \left( y^{(i)} - g(x^{(i)}, \beta) \right), \quad \frac{\partial \ell(\beta)}{\partial \beta_j} = \sum_{i=1}^N \left( y^{(i)} - g(x^{(i)}, \beta) \right) x_j^{(i)}.$$

Теперь можем воспользоваться методом градиентного спуска или более продвинутым методом оптимизации (сопряженных градиентов, BFGS, L-BFGS и др.).

Теперь рассмотрим случай  $\mathcal{Y} = \{1, 2, \dots, K\}$ .

$$g(x, \beta) = \Pr \{Y = k | X = x, \beta\} = \frac{e^{\beta_{k0} + \beta_k^\top x}}{\sum_{\ell=1}^K e^{\beta_{\ell0} + \beta_\ell^\top x}}.$$

Требуется максимизировать логарифмическую функцию правдоподобия

$$\ell(\beta) = \sum_{i=1}^N \sum_{k=1}^K I(y^{(i)} = k) \ln g(x^{(i)}, \beta) = \sum_{i=1}^N \sum_{k=1}^K I(y^{(i)} = k) \left( \beta_{k0} + \beta_k^\top x^{(i)} - \ln \sum_{\ell=1}^K e^{\beta_{\ell0} + \beta_\ell^\top x^{(i)}} \right).$$

**Упражнение 4.4** Докажите, что

$$\frac{\partial \ell(\beta)}{\partial \beta_{k0}} = \sum_{i=1}^N \left( I(y^{(i)} = k) - g(x^{(i)}, \beta) \right), \quad \frac{\partial \ell(\beta)}{\partial \beta_k} = \sum_{i=1}^N x^{(i)} \left( I(y^{(i)} = k) - g(x^{(i)}, \beta) \right).$$

**Замечание 4.5** Легко видеть, что максимизация логарифмической функции правдоподобия эквивалентна минимизации эмпирического риска

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

если в качестве штрафной функции рассмотреть *кросс-энтропию*:

$$L(g(x, \beta), y) = -y \ln g(x, \beta) - (1 - y) \ln (1 - g(x, \beta)).$$

Аналогично для случая  $K$  классов:

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

если в качестве штрафной функции рассмотреть *кросс-энтропию*:

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

#### 4.4.2. Регуляризация

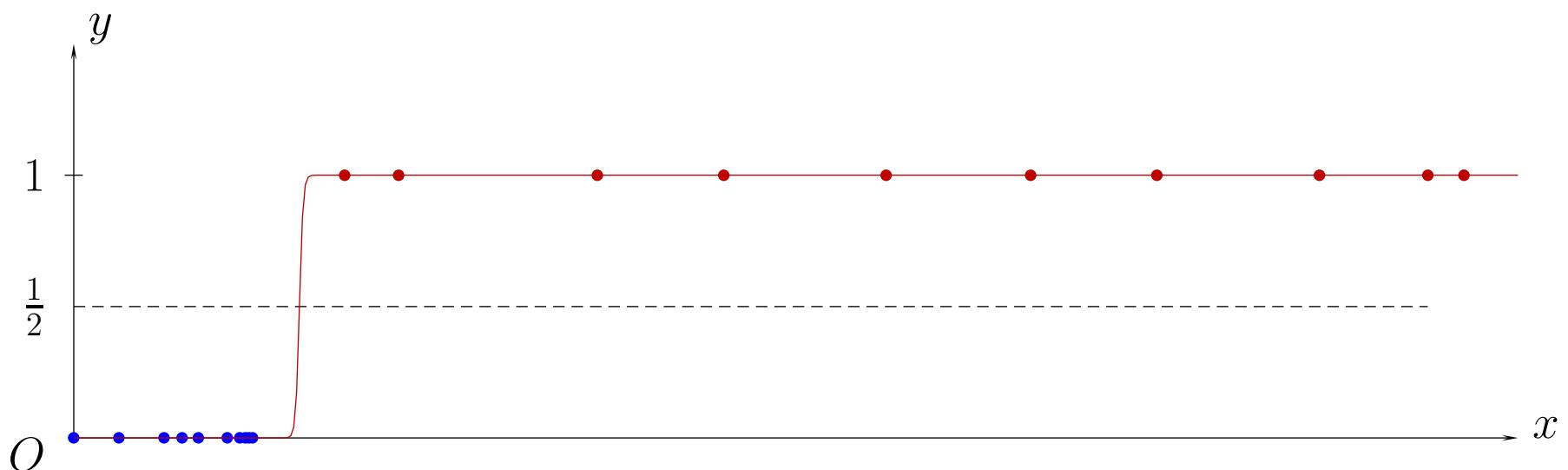
В  $L_2$  норме:

$$-\frac{1}{N} \sum_{i=1}^N \sum_{k=1}^K I(y^{(i)} = k) \ln g(x^{(i)}, \beta) + \lambda \|\beta\|_2 \rightarrow \min,$$

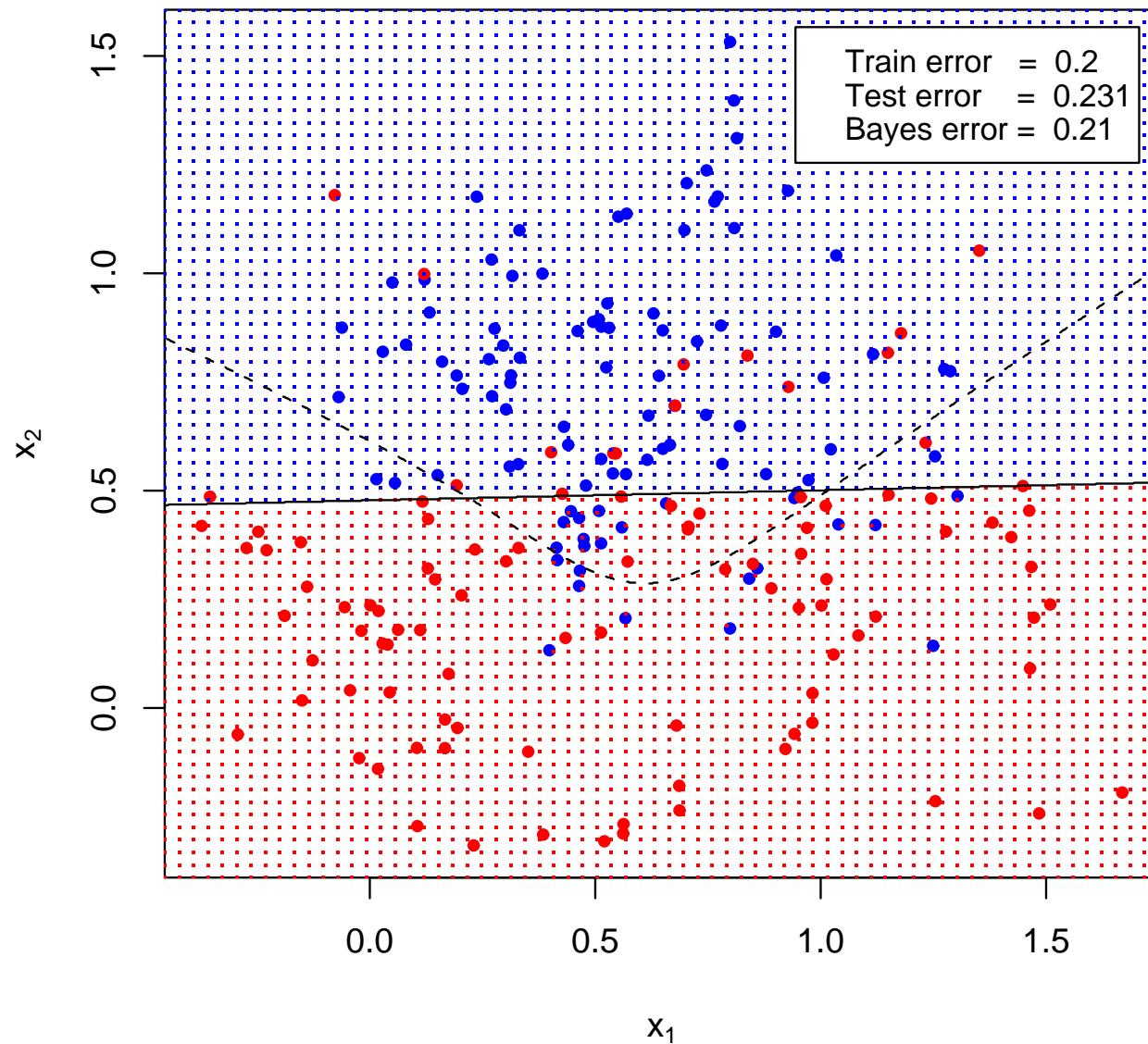
В  $L_1$  норме:

$$-\frac{1}{N} \sum_{i=1}^N \sum_{k=1}^K I(y^{(i)} = k) \ln g(x^{(i)}, \beta) + \lambda \|\beta\|_1 \rightarrow \min,$$

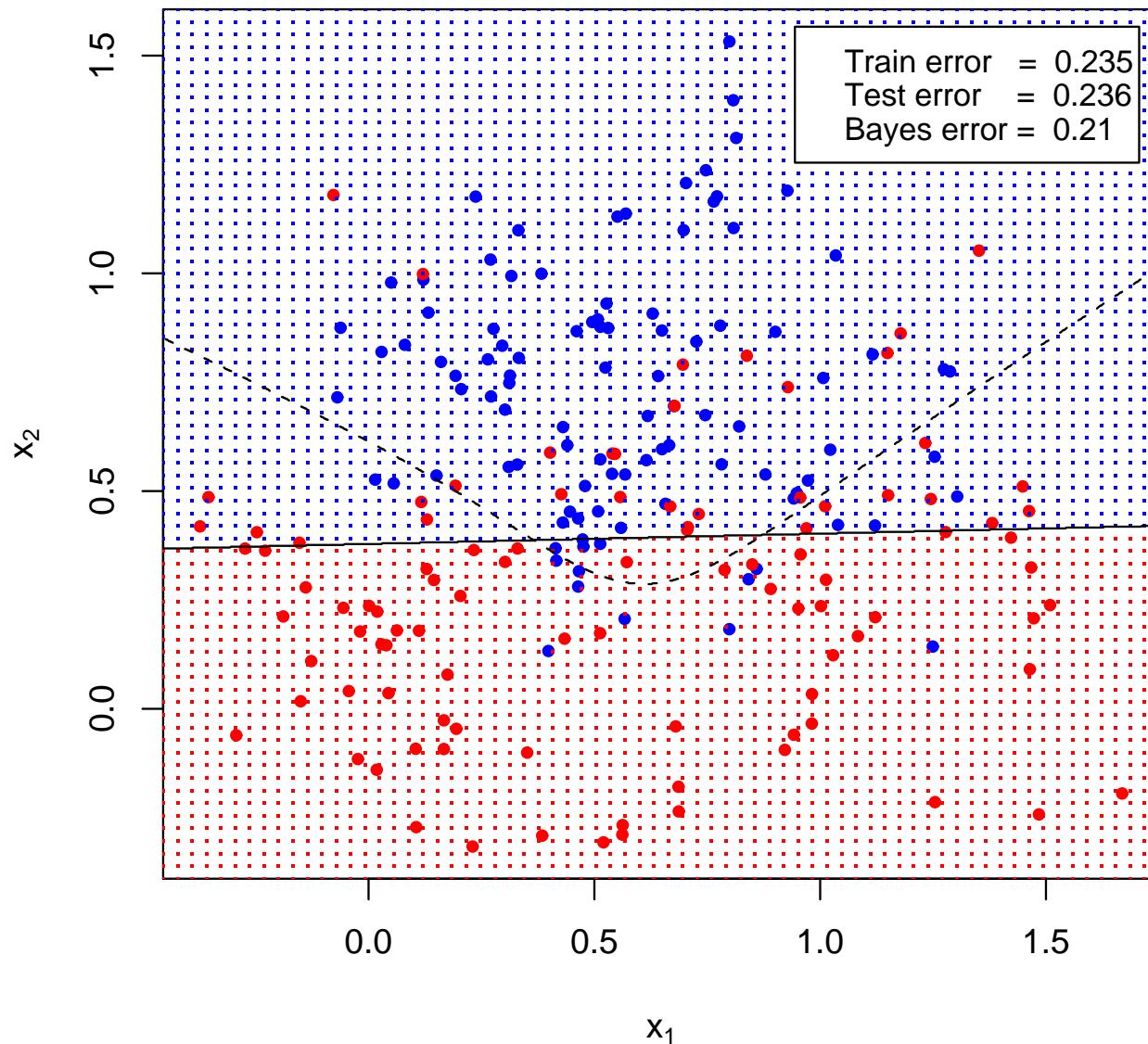
Без регуляризации, если классы линейно отделимы:  $\beta \rightarrow -\infty$



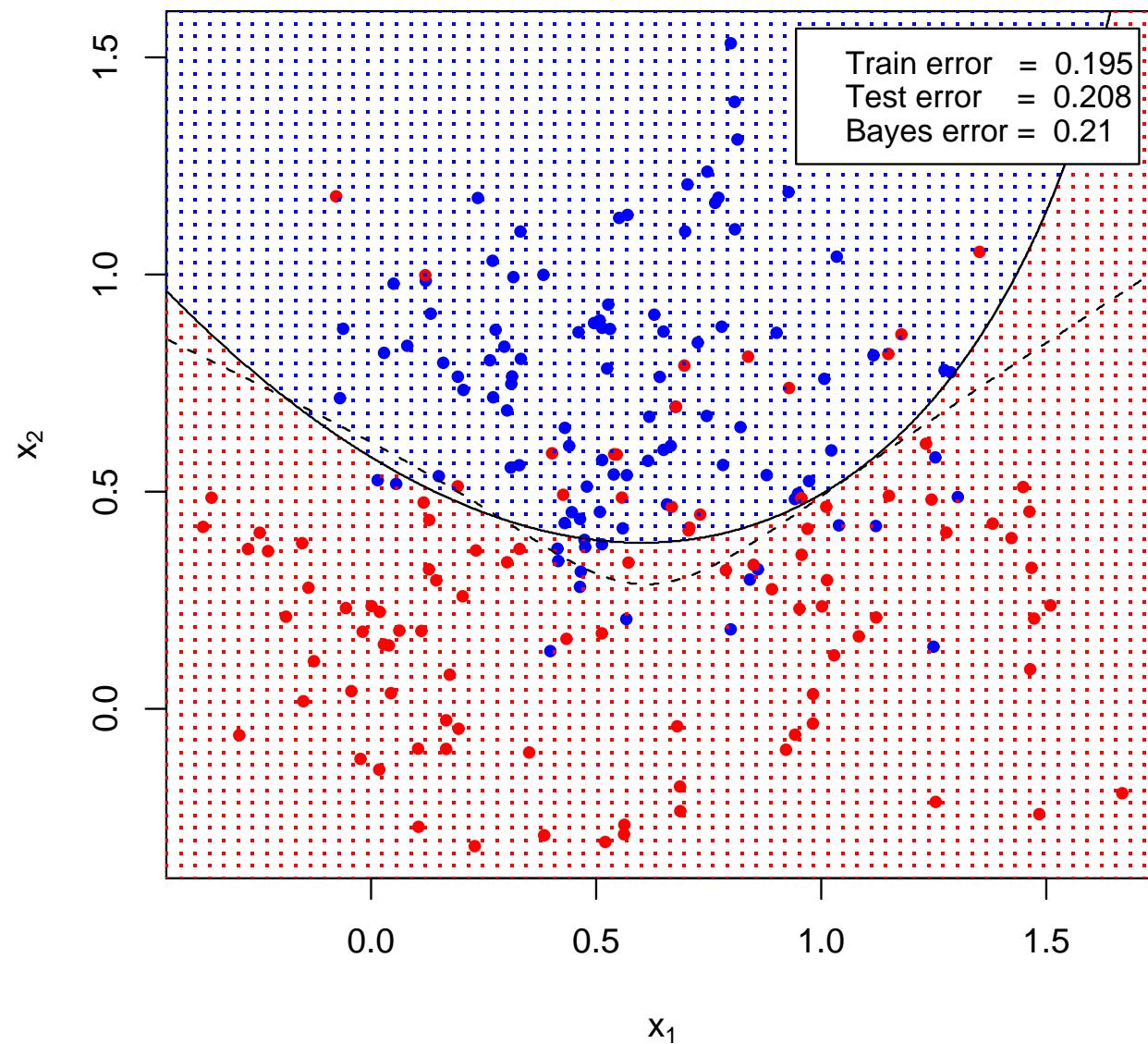
# LDA



## Логистическая регрессия



# QDA



*Глава 5*

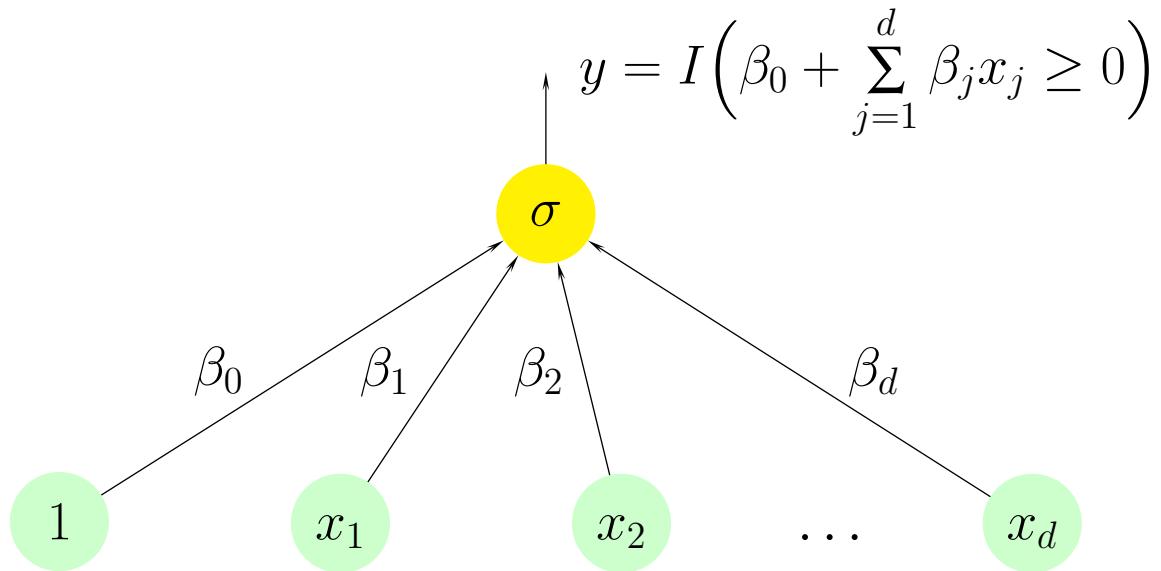
## **Нейронные сети**

## 5.1. Персептрон Розенблатта

Модель нейрона и нейронной сети [W. S. McCulloch, W. Pitts, 1943]

Идеи обучения нейронной сети [D. O. Hebb, 1949]

Алгоритм обучения, программа [1958], нейрокомпьютер MARK 1 [F. Rosenblatt, 1960]



$K = 2$ . Разделяющая поверхность — гиперплоскость  $\beta_0 + \sum_{j=1}^d \beta_j x_j = \beta_0 + \beta^\top x = 0$ .

Решающее правило —

$$f(x) = \begin{cases} 1, & \beta_0 + \beta^\top x \geq 0, \\ 0, & \beta_0 + \beta^\top x < 0 \end{cases}$$

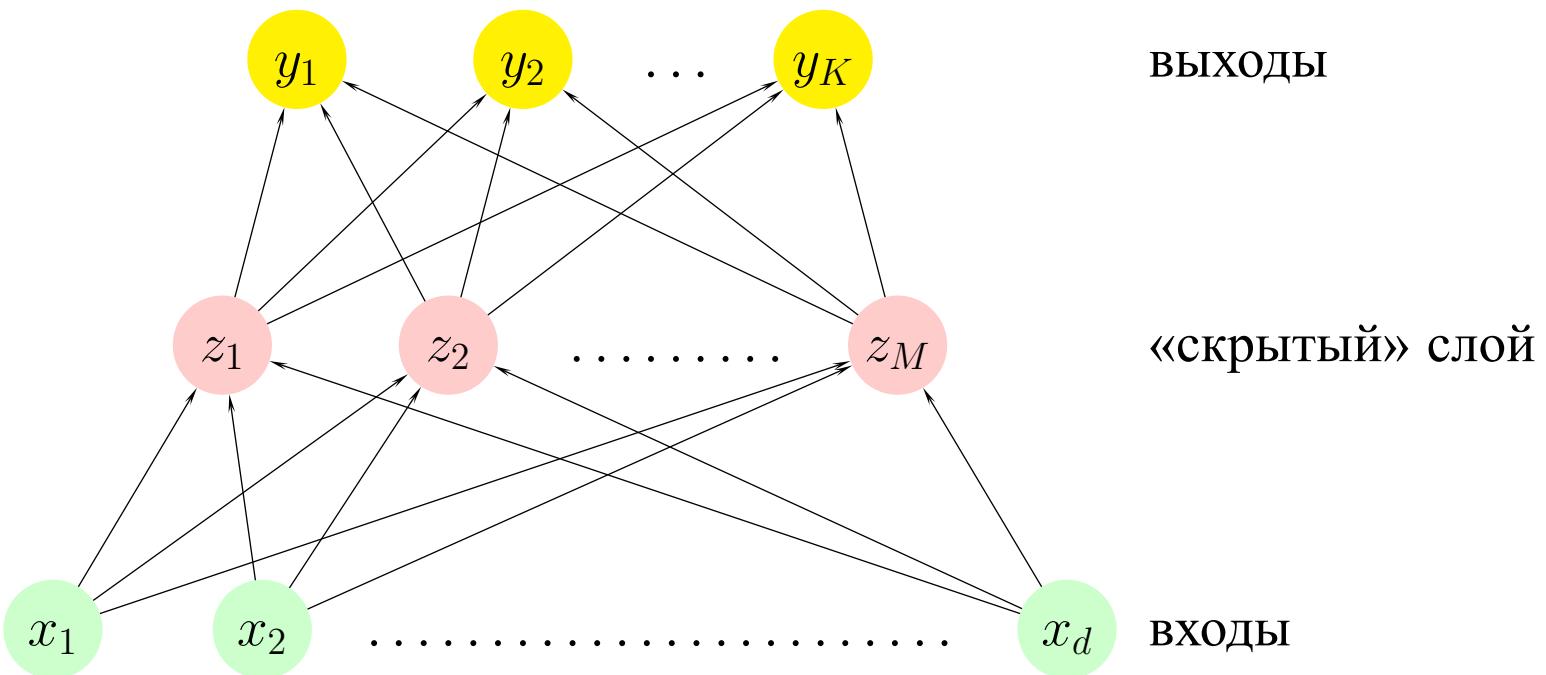
## 5.2. Нейронные сети

В мозге человека порядка  $10^{12}$  нейронов и  $10^{15}$  нейронных связей.

Нейронная сеть — (очень простая) модель мозга, т. е. множества взаимодействующих между собой нейронов.

*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

Пример «трехслойной» нейронной сети:



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

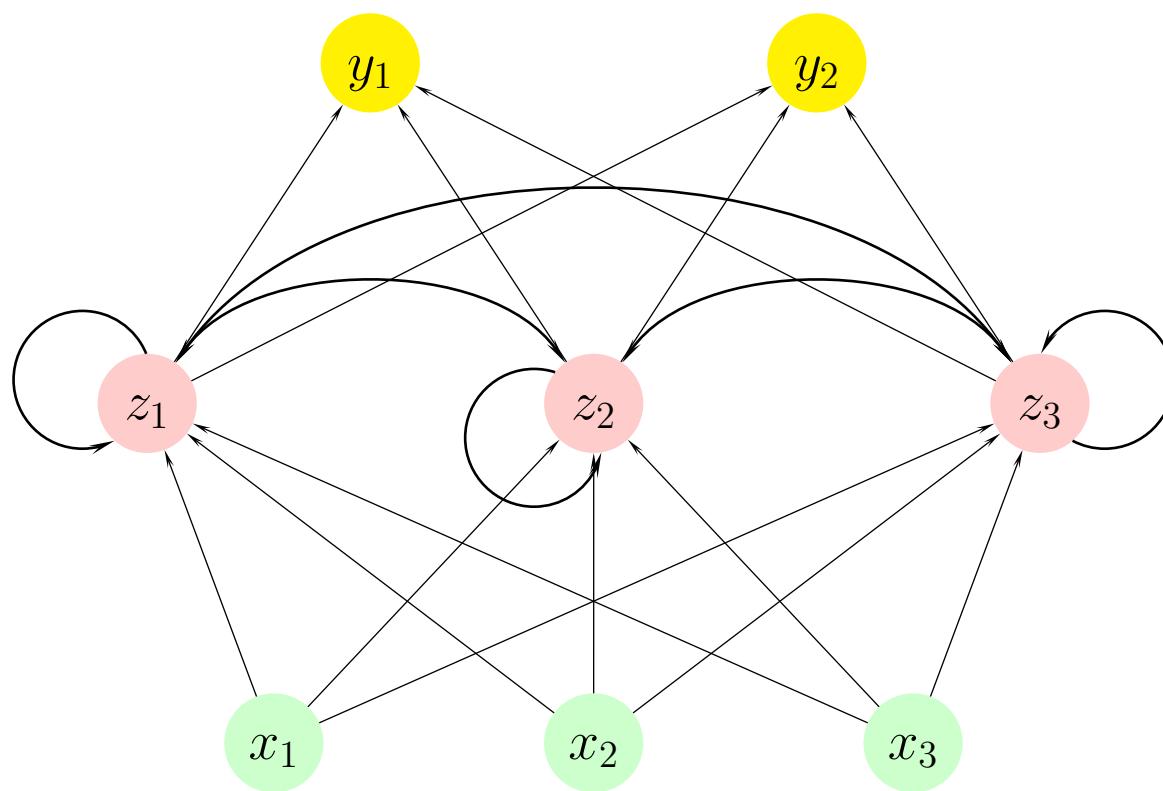
Небольшая классификация нейронных сетей:

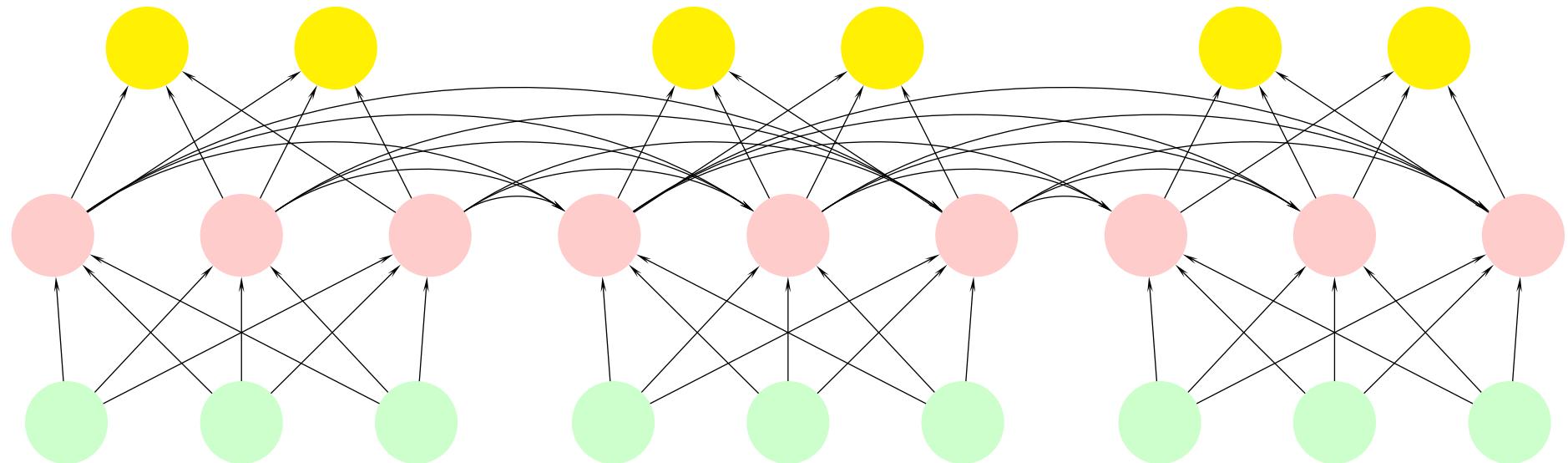
- сети прямого распространения (feed-forward NN) — отсутствуют орцикли;
- рекуррентные нейронные сети, или сети с обратной связью (recurrent NN) — присутствуют орцикли.

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

Если узлы сети можно разбить на группы (слои), которые удается пронумеровать так, что дуги будут вести только от вершин из  $i$ -го слоя в вершины  $(i + 1)$ -го слоя, то нейронная сеть называется *многоуровневой* (или *многослойным персепtronом*).

## Пример рекуррентной сети



 $t - 1$  $t$  $t + 1$

В качестве функций активации обычно берут пороговые и близкие к ним функции:  
*Пороговая функция* (в чистом виде обычно не используется, так как разрывна):

$$g(x_1, x_2, \dots, x_q) = \begin{cases} 1, & \text{если } \beta_0 + \beta_1 x_1 + \dots + \beta_q x_q \geq 0, \\ 0, & \text{если } \beta_0 + \beta_1 x_1 + \dots + \beta_q x_q < 0, \end{cases}$$

*Сигмоидальная (логит или логистическая) функция* (наиболее популярная):

$$g(x_1, x_2, \dots, x_q) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \dots + \beta_q x_q)}}$$

*Арктангенс:*

$$g(x_1, x_2, \dots, x_q) = \frac{1}{2} + \frac{1}{\pi} \operatorname{arctg}(\beta_0 + \beta_1 x_1 + \dots + \beta_q x_q)$$

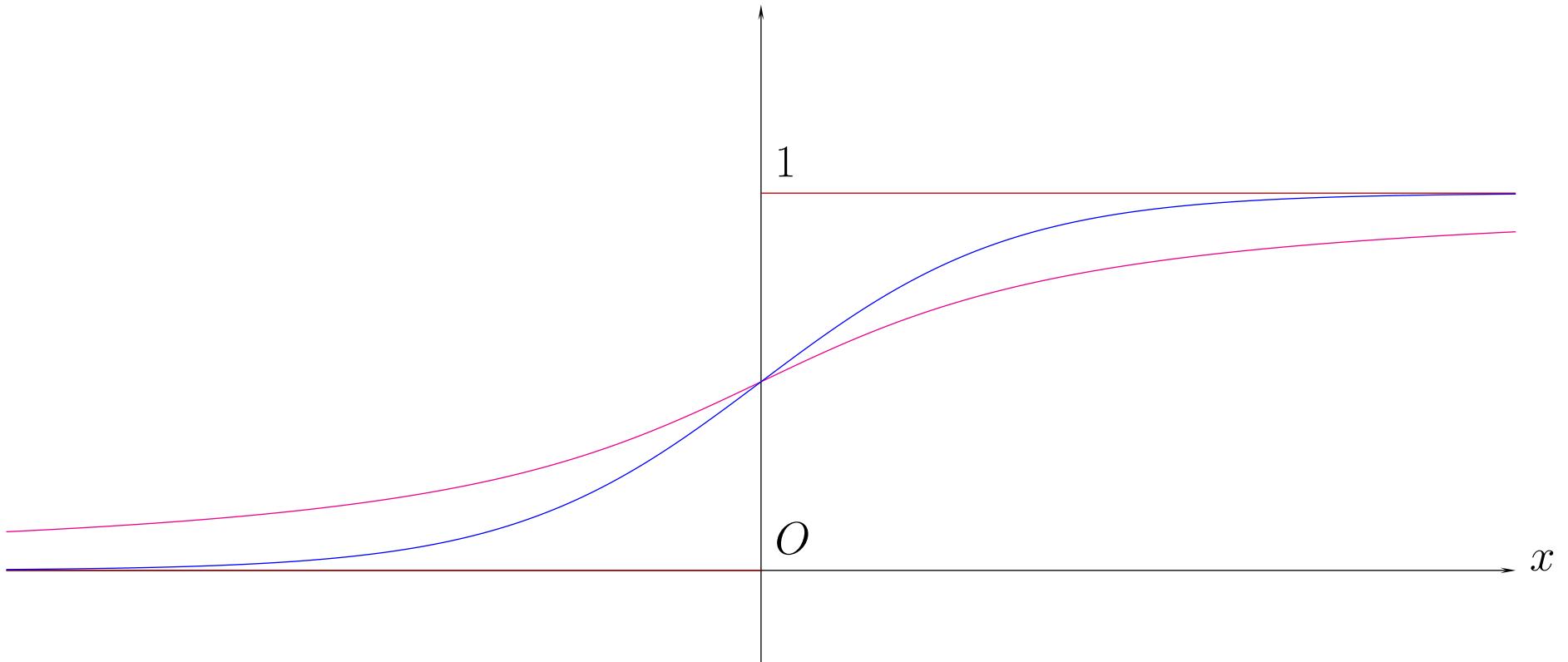
В специальном виде нейронных сетей — радиальных нейронных сетях (radial basis function network — RBF network) — используют радиальную функцию (Гауссиан)

$$g(x) = e^{-\beta \|x - c\|}.$$

Элементарная пороговая функция  $y = I(x \geq 0) = \begin{cases} 1, & \text{если } x \geq 0, \\ 0, & \text{если } x < 0, \end{cases}$

Элементарный сигмоид (логит или логистическая функция)  $y = \sigma(x) = \frac{1}{1 + e^{-x}}$

Арктангенс:  $y = \frac{1}{2} + \frac{1}{\pi} \arctg x$



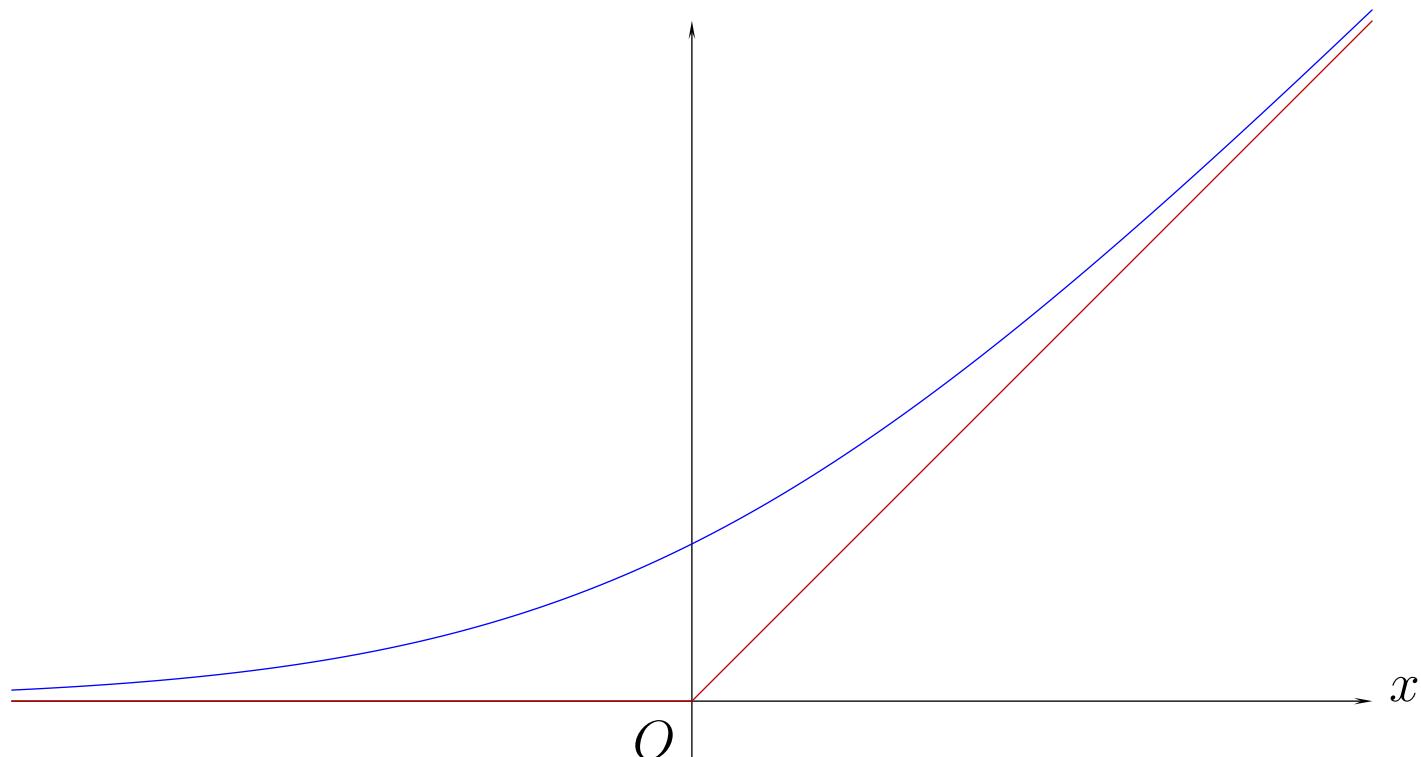
Используют и другие функции, например, *положительную срезку линейной функции* (linear rectifier):

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

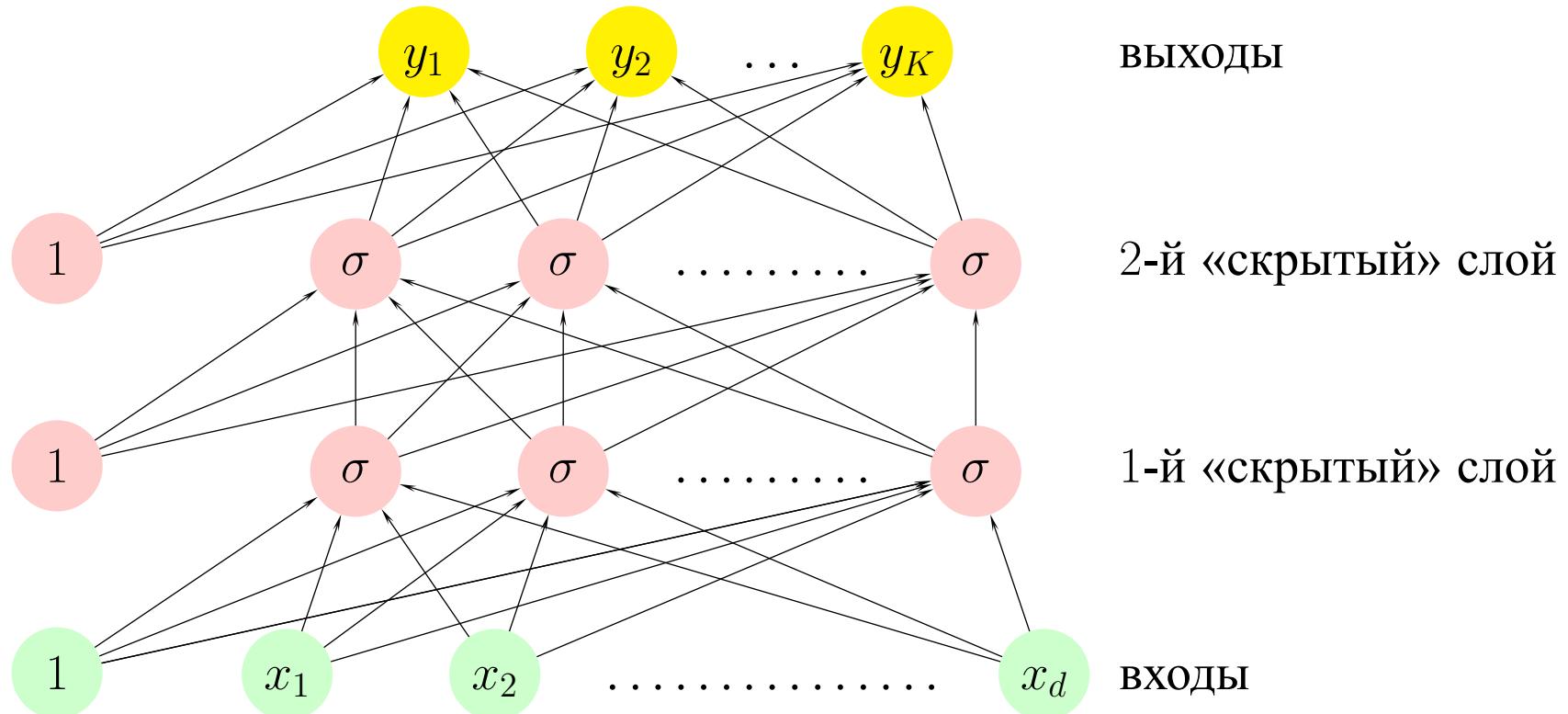
где  $(x)_+ = \max \{0, x\}$ , или ее сглаженный вариант *softplus*:

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

— эти функции успешно применяются для решения проблемы исчезающего градиента (см. ниже).



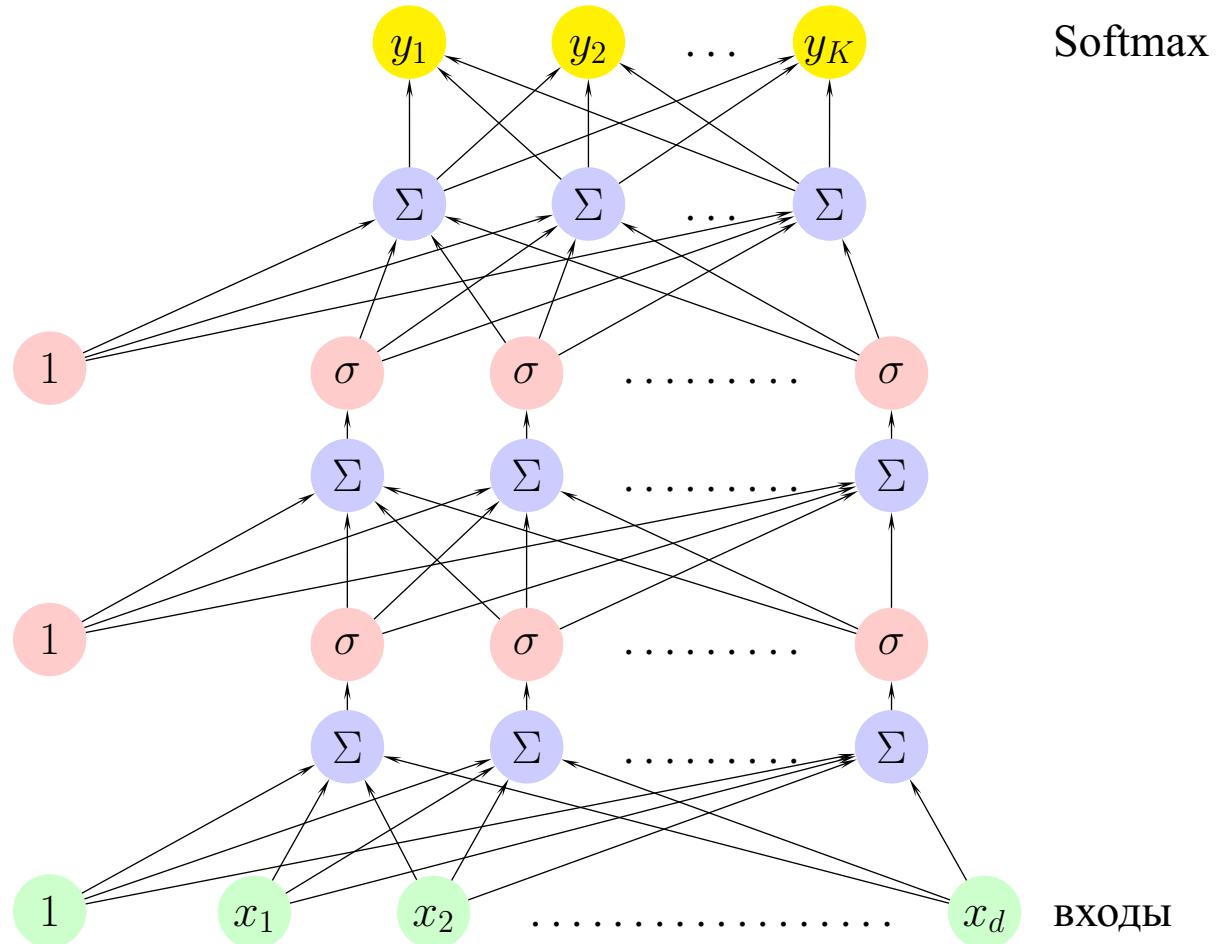
Нейронные сети часто изображают с дополнительными «единичными» нейронами. Веса указываются рядом с соответствующими дугами.



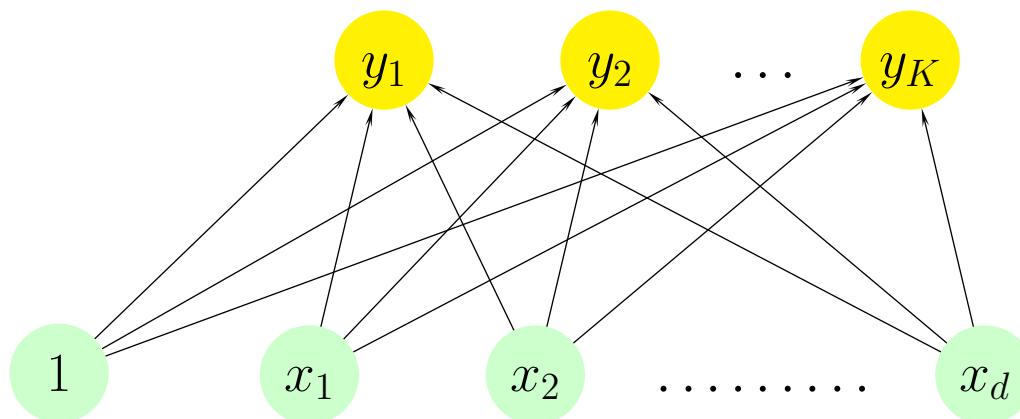
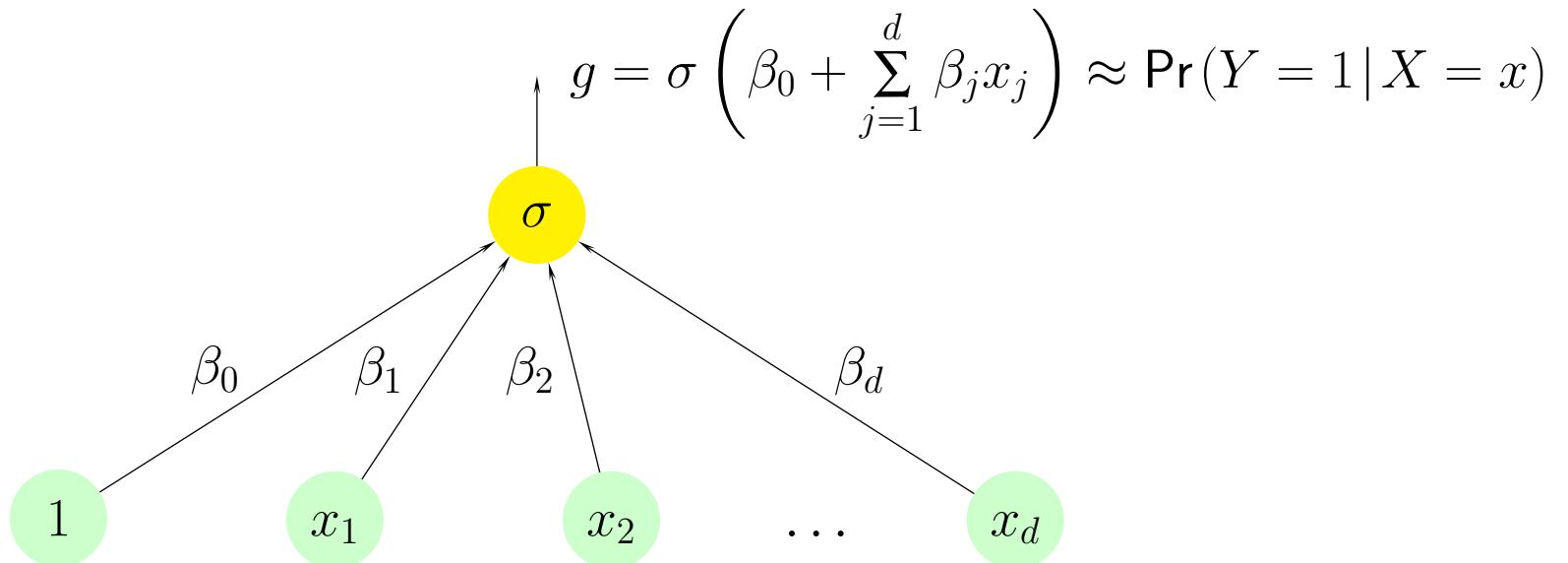
Таким образом, выходы из каждого узла (нейрона) умножаются на соответствующие веса и складываются.

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

Иногда отдельно изображают суммирующие элементы и элементы, вычисляющие  $\sigma$ :

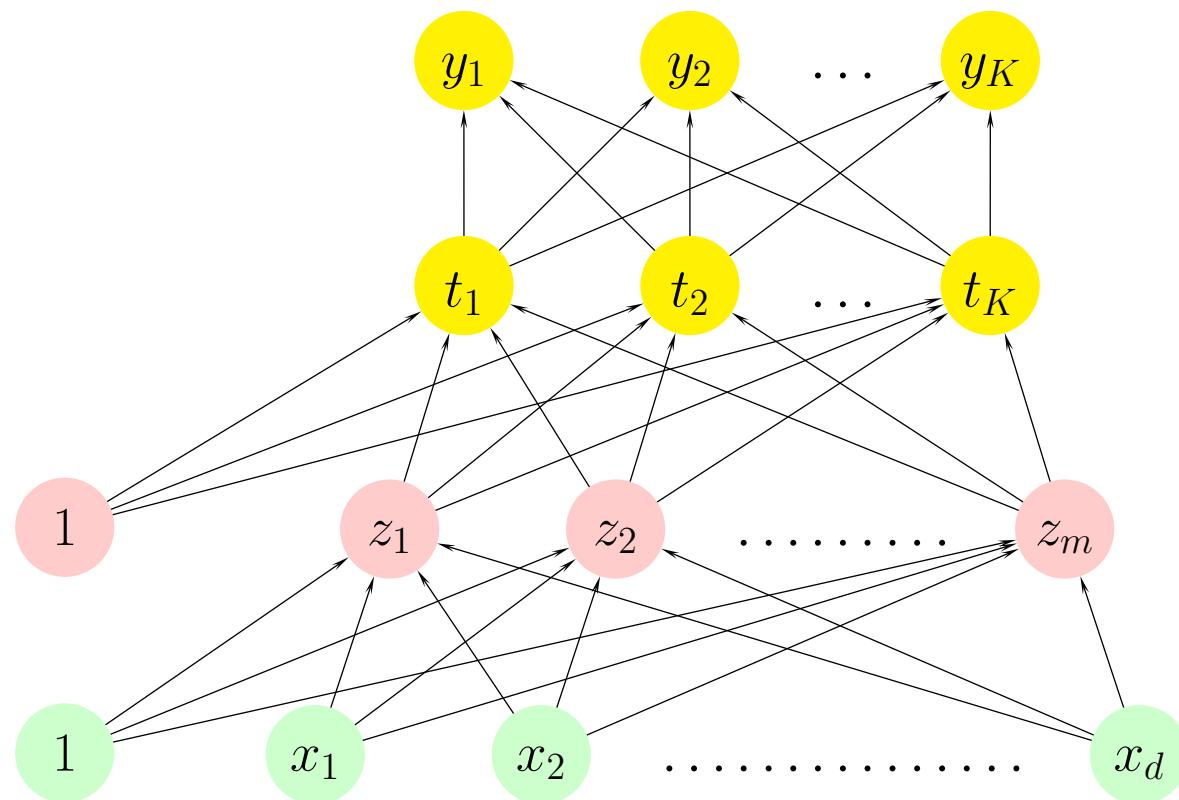


Двухслойная нейронная сеть (с логистической функцией активации  $\sigma$ ) — это логистическая регрессия.



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

## Трехслойная («ванилла») нейронная сеть для задачи классификации

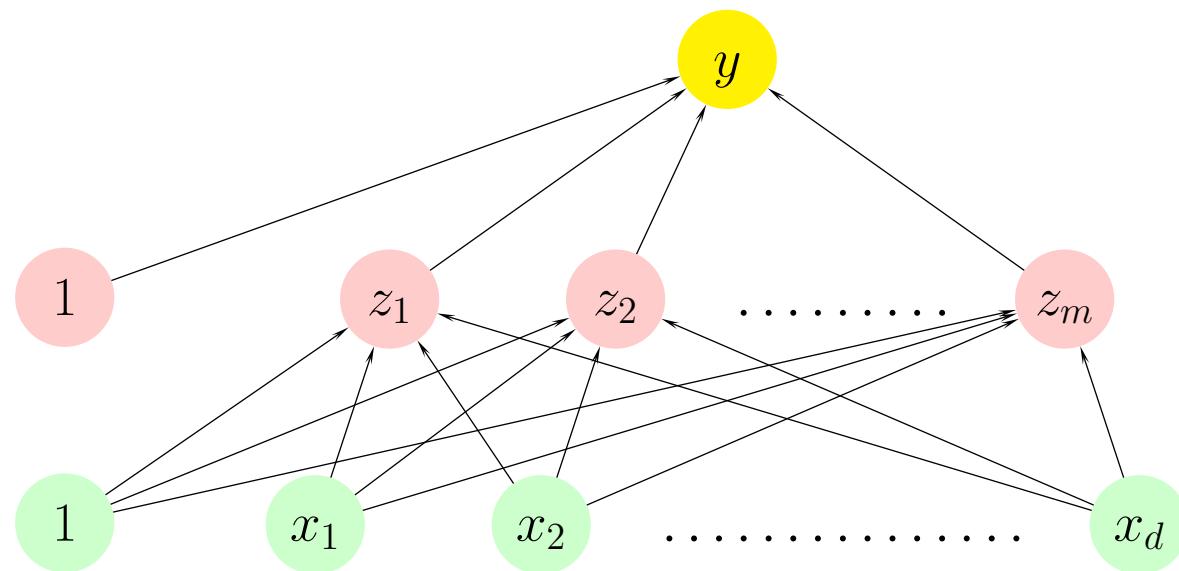


$$y_k = \frac{e^{t_k}}{\sum_{\ell} e^{t_{\ell}}} \approx \Pr(k|x)$$

$$t_k = \gamma_{k0} + \sum_{i=1}^m \gamma_{ki} z_i$$

$$z_i = \sigma \left( \beta_{i0} + \sum_{j=1}^d \beta_{ij} x_j \right)$$

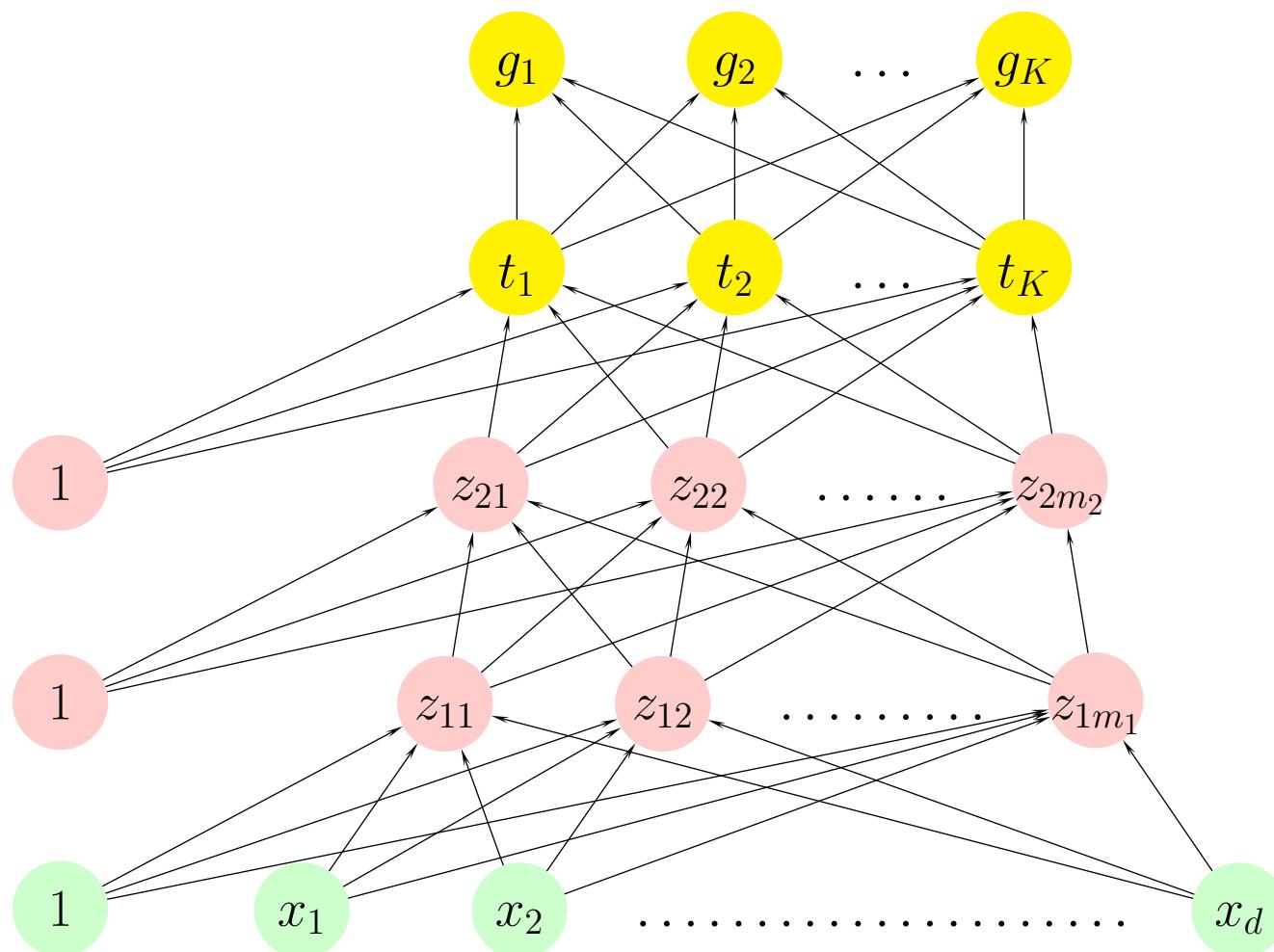
Трехслойная («ванилла») нейронная сеть для задачи восстановления регрессии



$$y = \gamma_0 + \sum_{i=1}^m \gamma_i z_i$$

$$z_i = \sigma\left(\beta_{i0} + \sum_{j=1}^d \beta_{ij} x_j\right)$$

## Четырехслойная нейронная сеть (2 скрытых слоя) для задачи классификации



$$g_k = \frac{e^{t_k}}{\sum_{\ell=1}^K e^{t_\ell}} \approx \Pr(k|x)$$

$$t_k = \gamma_{k0} + \sum_{i=1}^{m_2} \gamma_{ki} z_{2i}$$

$$z_{2i} = \sigma \left( \beta_{2i0} + \sum_{j=1}^{m_1} \beta_{2ij} z_{1j} \right)$$

$$z_{1i} = \sigma \left( \beta_{1i0} + \sum_{j=1}^d \beta_{1ij} x_j \right)$$

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

## Почему нейронные сети?

- $\&$ ,  $\vee$  — пороговые функции. На самом деле из 16 булевых функций двух переменных не являются пороговыми только  $x \oplus$  (сложение по модулю 2) и его отрицание.
- 3-слойных нейронных сетей с пороговыми функциями активации достаточно для представления всех булевых функций, зависящих от  $d$ -переменных (а сколько всего элементов требуется?)
- 3-слойных нейронных сетей с пороговыми функциями активации достаточно, чтобы выделять выпуклые многогранные области (конъюнкция полупространств), а 4-слойных — чтобы выделять невыпуклые, в т.ч. многосвязные области (дизъюнкция многограных областей).
- 4-слойные нейронные сети с сигмоидальными функциями активации выделяют «сглаженные» многогранные области.

## Отступление. История вопроса

- Теорема Вейерштрасса: любую непрерывную функцию можно аппроксимировать полиномами (но сколько их достаточно?). Есть обобщения этой теоремы Стоуна и Горбаня. Есть еще ряды Фурье.
- Колмогоров, Арнольд: любая непрерывная функция  $f(x_1, x_2, \dots, x_d)$  от  $d$  аргументов представима в виде

$$f(x_1, x_2, \dots, x_d) = \sum_{k=1}^{2d+1} h_k \left( \sum_{j=1}^d \varphi_{jk}(x_j) \right),$$

где  $h_k, \varphi_{jk}$  — непрерывные функции, причем  $\varphi_{jk}$  не зависят от  $f$  (13-я проблема Гильберта). Таким образом, 3 слоев достаточно (но функции активации не пороговые и не близкие к ним.)

## «Универсальная теорема об аппроксимации»

**Теорема 5.1** Пусть  $\sigma$  — ограниченная, монотонно возрастающая, непрерывная функция. Тогда для любой непрерывной функции  $f^* : [0, 1]^d \rightarrow \mathbf{R}$ , для любого  $\varepsilon > 0$  существуют  $M$ ,  $\alpha_m$  ( $m = 1, 2, \dots, M$ ),  $w_{mj}$  ( $m = 1, 2, \dots, M$ ;  $j = 0, 1, \dots, d$ ), такие, что функция

$$f(x_1, \dots, x_d) = \sum_{m=1}^M \alpha_m \sigma \left( w_{m0} + \sum_{j=1}^d w_{mj} x_j \right)$$

является  $\varepsilon$ -аппроксимацией функции  $f$ , т. е.  $|f(x) - f^*(x)| \leq \varepsilon$ .

Таким образом, 3 слоев достаточно.

Но чему равно  $M$  — число узлов скрытого слоя?

(есть результаты, например, в терминах первых моментов преобразования Фурье)

## 5.3. Обучение нейронной сети (backpropagation)

[H. J. Kelley, 1960; A. E. Bryson, 1961; S. Dreyfus, 1962; S. Linnainmaa, 1970; **А. И. Галушкин**, 1974; **P. Werbos**, 1974; С. И. Барцев, В. А. Охонин, 1986; **D. E. Rumelhart et. al.**, 1986]

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

В задачах восстановления регрессии, например, минимизируют сумму квадратов

$$R(w) = \frac{1}{2N} \sum_{i=1}^N \left( y^{(i)} - f(x^{(i)}) \right)^2 \quad (\text{a})$$

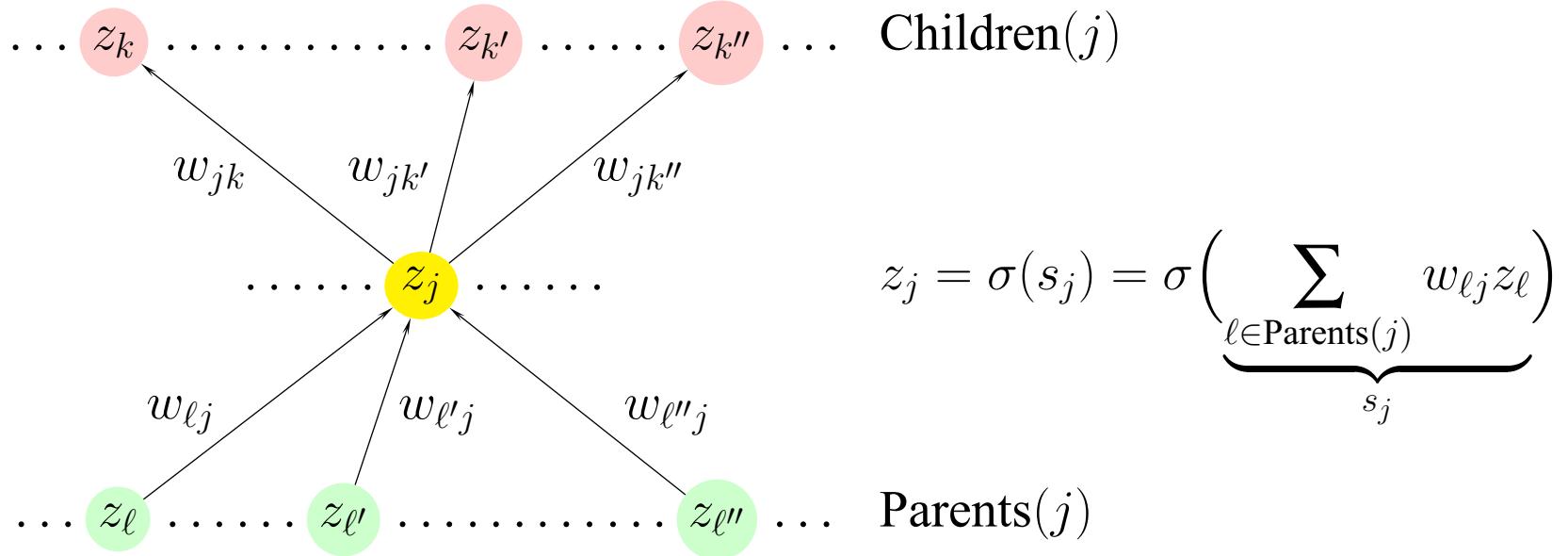
В задачах классификации обычно минимизируют *кросс-энтропию* (это логарифмическая функция правдоподобия!)

$$R(w) = -\frac{1}{N} \sum_{i=1}^N \sum_{k=1}^K y_k^{(i)} \log f_k(x^{(i)}), \quad y_k^{(i)} = 1 \Leftrightarrow y^{(i)} = k \quad (\text{b})$$

Задача минимизации  $R(w)$  решается численными методами: например, методом градиентного спуска, квазиньютоновскими методами и т. п.

Нужно уметь вычислять производную.

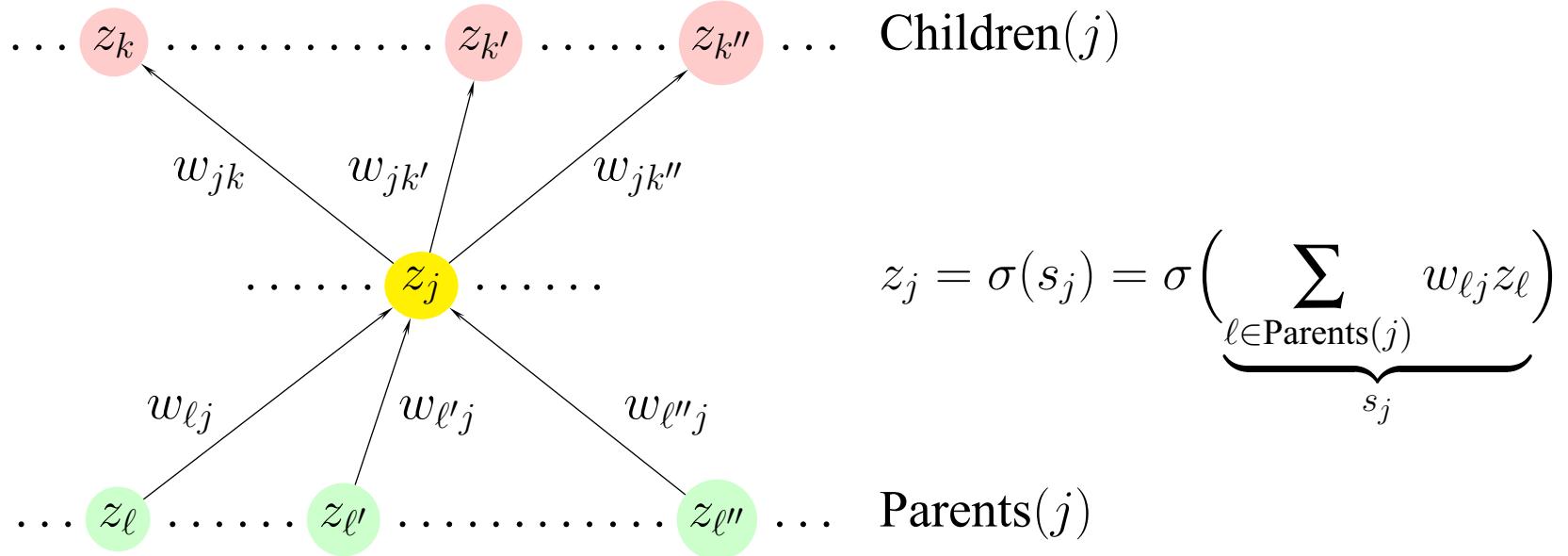
Алгоритм *обратного распространения ошибки* (*back propagation*) — это метод вычисления градиента  $\partial R / \partial w$ . Главное — цепное правило!



$$\delta_j = \frac{\partial R^{(i)}}{\partial s_j} \text{ — «ошибка» в } j\text{-м узле,}$$

$$(a) \quad R(w) = \frac{1}{N} \sum_{i=1}^N R^{(i)}(w), \quad R^{(i)}(w) = \frac{1}{2} \sum_{j \in \text{Outputs}} \left( y_j^{(i)} - f_j(x^{(i)}) \right)^2$$

$$(b) \quad R(w) = \frac{1}{N} \sum_{i=1}^N R^{(i)}(w), \quad R^{(i)}(w) = - \sum_{j \in \text{Outputs}} y_j^{(i)} \ln z_j = - \sum_{j \in \text{Outputs}} y_j^{(i)} \ln \frac{e^{s_j}}{\sum_m e^{s_m}}$$



Если  $j \in \text{Outputs}$ :

(a, b)

$$\delta_j = \frac{\partial R^{(i)}}{\partial s_j} = z_j - y_j^{(i)},$$

Если  $j \notin \text{Outputs}$ :

$$\delta_j = \frac{\partial R^{(i)}}{\partial s_j} = \sum_{k \in \text{Children}(j)} \frac{\partial R^{(i)}}{\partial s_k} \cdot \frac{\partial s_k}{\partial z_j} \cdot \frac{\partial z_j}{\partial s_j} = \sum_{k \in \text{Children}(j)} \delta_k w_{jk} \sigma'(s_j) = \sigma'(s_j) \sum_{k \in \text{Children}(j)} \delta_k w_{jk}$$

Для всех  $j$ :

$$\frac{\partial R^{(i)}}{\partial w_{\ell j}} = \frac{\partial R^{(i)}}{\partial s_j} \cdot \frac{\partial s_j}{\partial w_{\ell j}} = \delta_j z_\ell$$

Batch и On-line Back Propagation ( $\rho_r \rightarrow 0$ ,  $\sum \rho_r = \infty$ ,  $\sum \rho_r^2 < \infty$ , например,  $\rho_r = \frac{1}{r}$ ):

$$w_{\ell j}^{(r+1)} = w_{\ell j}^{(r)} - \rho \sum_{i=1}^N \frac{\partial R^{(i)}}{\partial w_{\ell j}}, \quad w_{\ell j}^{(r+1)} = w_{\ell j}^{(r)} - \rho \frac{\partial R^{(i)}}{\partial w_{\ell j}}$$

**procedure** BackPropagation (on-line)

Инициализировать  $w_{\ell j}$  (например, случайными значениями)

**repeat**

**for**  $i = 1, \dots, N$

Прямой ход: подать на вход  $x^{(i)}$ , вычислить все  $z_j$

Обратный ход:

**for**  $j \in \text{Outputs}$

$$(a, b) \quad \delta_j \leftarrow z_j - y_j^{(i)}$$

**for** каждого уровня, начиная с предпоследнего

**for** каждого узла  $j$  текущего уровня

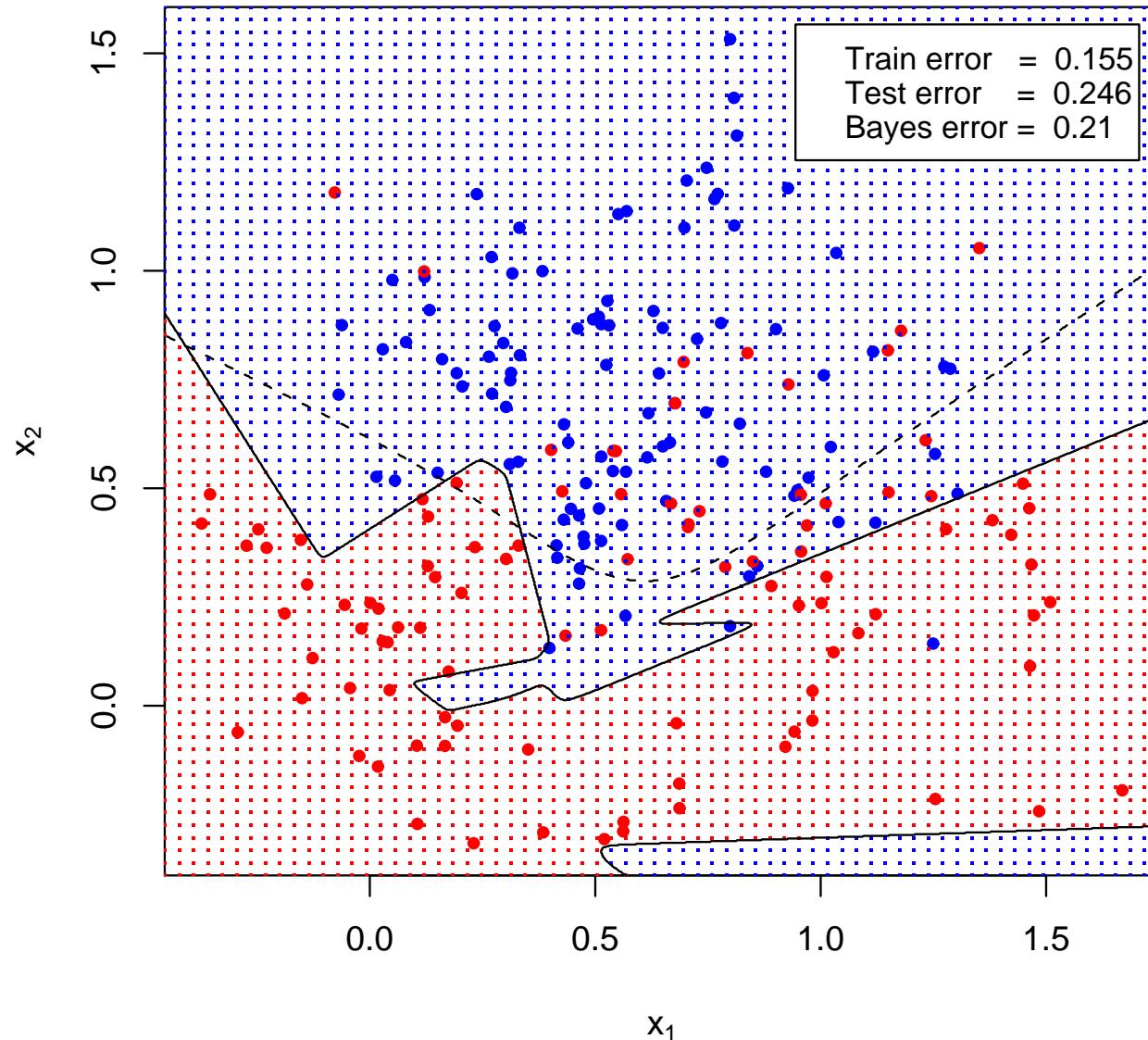
$$\delta_j \leftarrow \sigma'(s_j) \sum_{k \in \text{Children}(j)} w_{jk} \delta_k$$

**for** каждой дуги  $(\ell, j)$

$$w_{\ell j} \leftarrow w_{\ell j} - \rho \delta_j z_l$$

**return**  $w_{\ell j}$

Классификация на основе трехслойной нейронной сети с 10 скрытыми узлами.  
Построенный классификатор сильно зависит от начальных значений.



## 5.4. Переобучение

Регуляризация — *weight decay*

Добавим штраф к функции ошибок:  $R(w) + \lambda J(w)$ , где

$$J(w) = \sum_{i i'} w_{i i'}^2, \quad (\star)$$

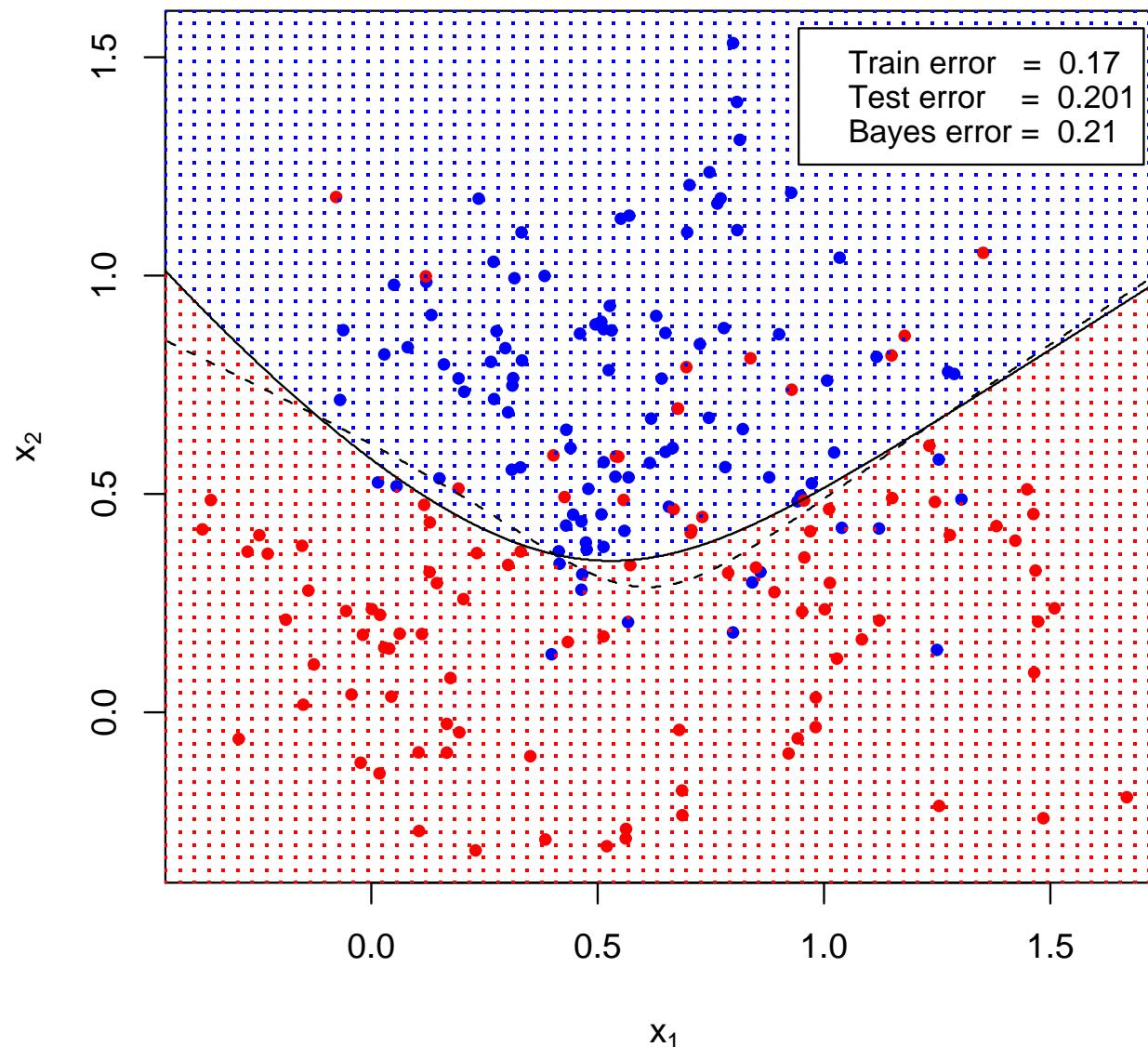
$$\lambda \geq 0$$

Вариант:

$$J(w) = \sum_{i i'} \frac{w_{i i'}^2}{1 + w_{i i'}^2},$$

— большие коэффициенты подвергаются большему сокращению, чем это делает  $(\star)$

Классификация на основе трехслойной нейронной сети с 10 скрытыми узлами и коэффициентом  $\lambda = \text{decay weight} = 0.1$



В примере `digits` выборка размера 1934 была случайным образом разбита на две группы по 967 объектов в каждой.

Для обучения использовалась трехслойная нейронная сеть с 15 элементами в скрытом слое.

100 итераций алгоритма BFGS

decay weight	<i>Ошибка</i>	
	<i>на обучающей выборке</i>	<i>на тестовой выборке</i>
0	0.000	0.111
0.1	0.000	0.078
0.2	0.000	0.035
0.3	0.000	0.037
0.4	0.000	0.039
0.5	0.000	0.037
0.6	0.000	0.035
0.7	0.000	<b>0.029</b>
0.8	0.000	0.036
0.9	0	0.037
1.0	0	0.033
5.0	0.007	0.039
10.0	0.027	0.047
50.0	0.249	0.267

Рукописные цифры, которые были не правильно классифицированы нейронной сетью с decay weight = 0.7. Цифра рядом с каждым изображением — ответ нейронной сети



## 5.5. Глубокое обучение

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

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

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

### **Некоторые подходы в глубоком обучении**

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

### **Проблемы с большими нейронными сетями**

- Переобучение
- Исчезающий градиент

## **5.5.1. 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)

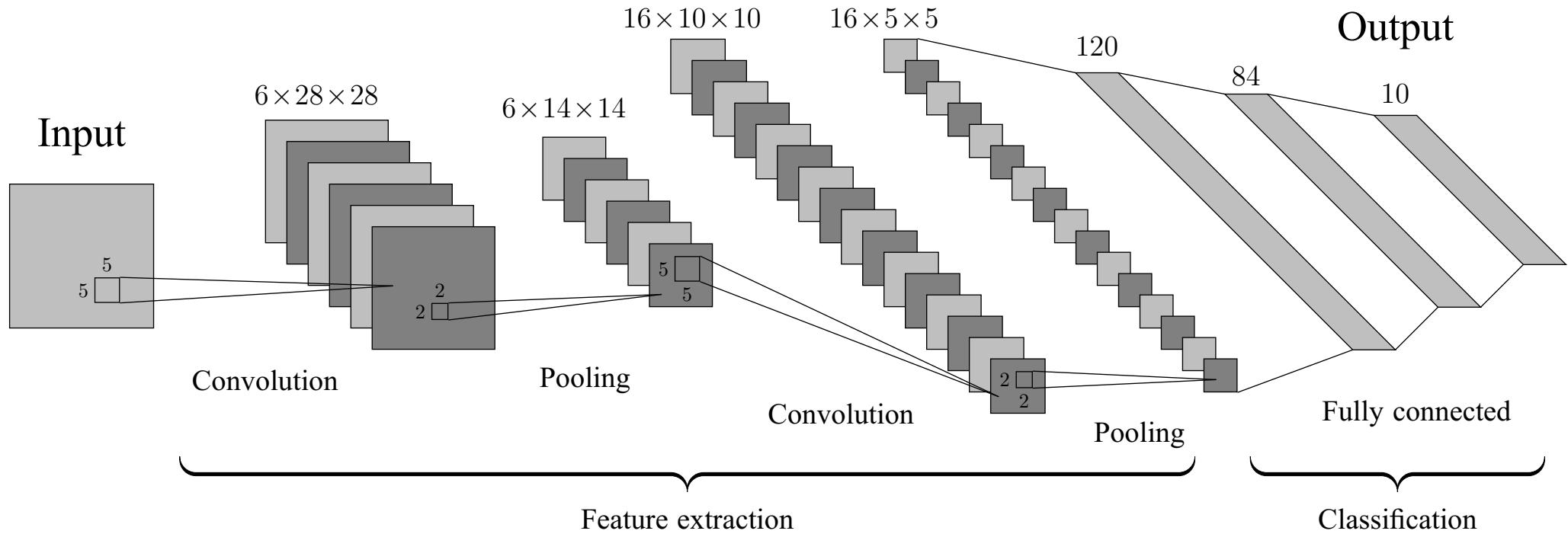
$z_{pq}$  — узлы следующего слоя

Параметры фильтра — это теперь веса нейронной сети.

Отличия от полносвязной сети (полносвязного слоя):

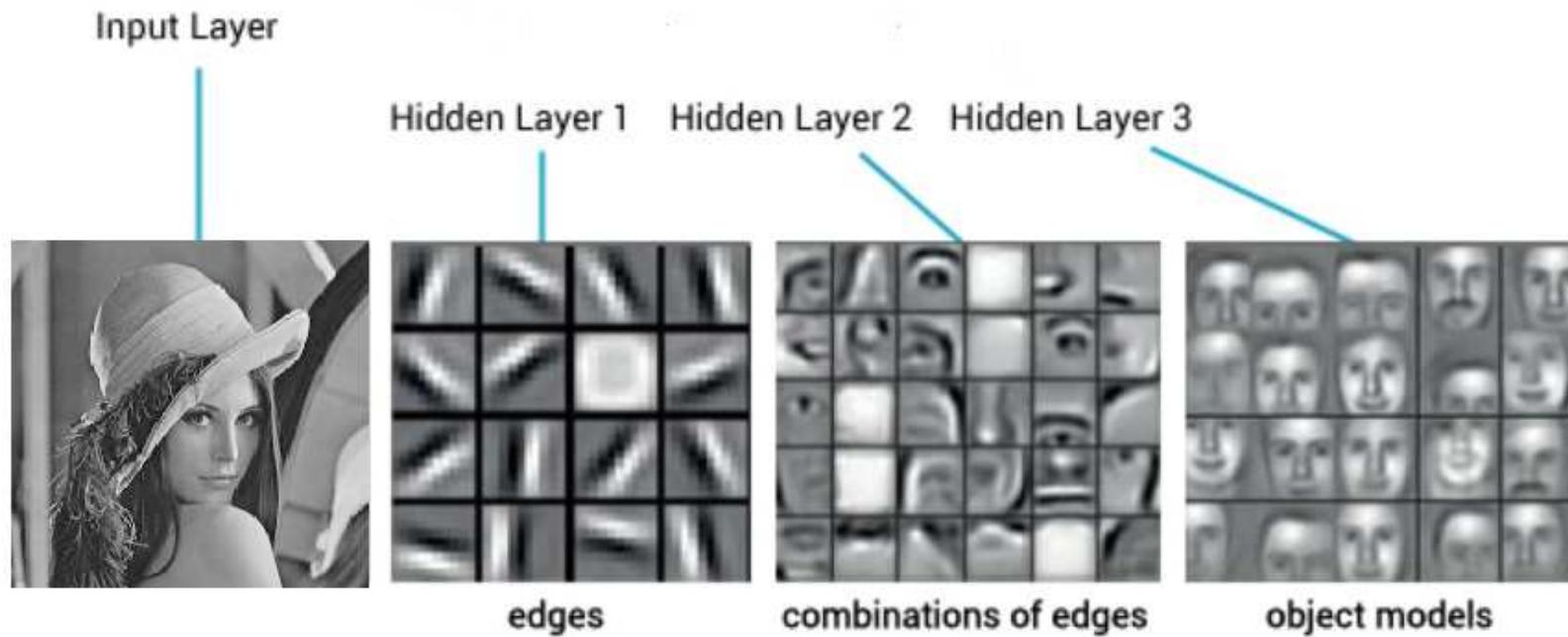
- Нет соединения каждого узла одного слоя со всеми узлами следующего.
- Веса становятся *разделяемыми*.

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



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

## Выделение признаков в слоях нейронной сети (схема)



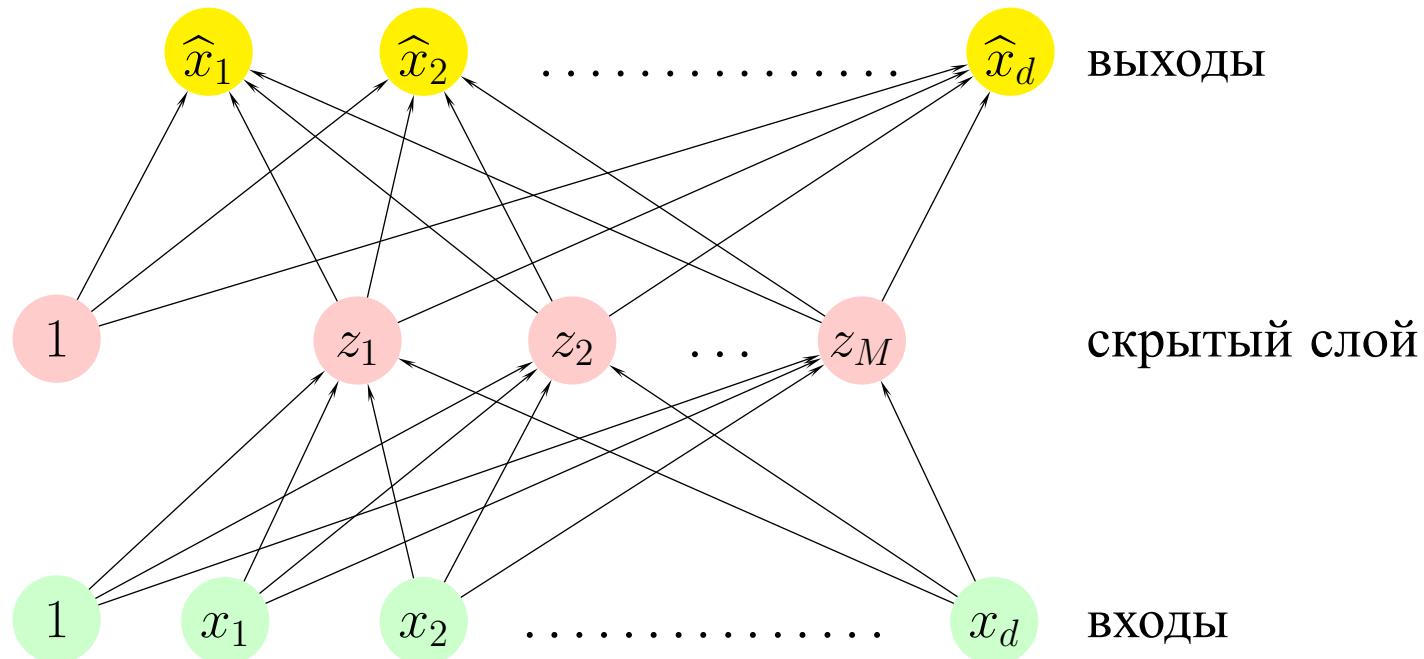
Другие примеры:

(внимание: количество слоев — вещь условная)

- 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
- ...

## 5.5.2. Автокодировщики

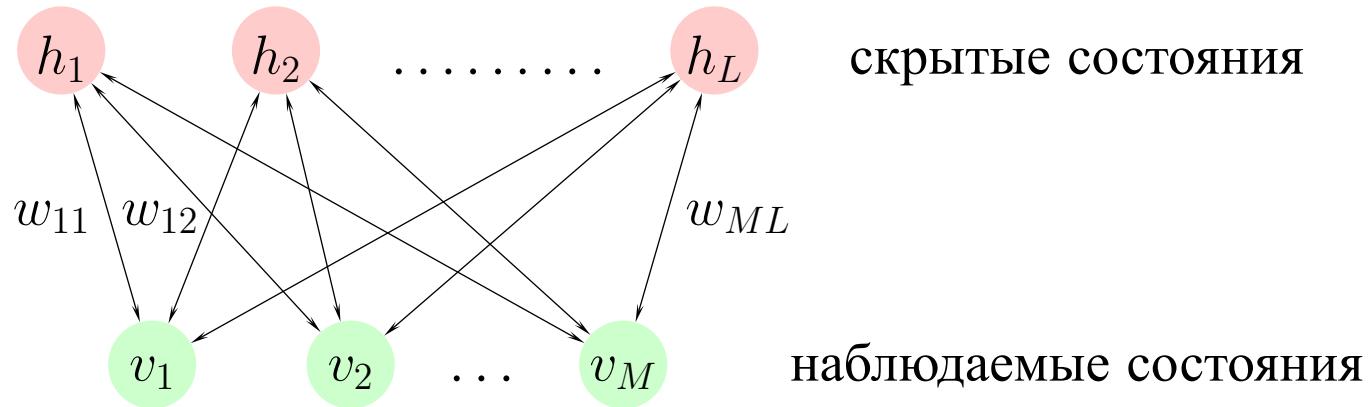
Автокодировщик (*autoencoder*) — метод сжатия (с потерями) на основе обучения без учителя.



$$x_1 \approx \hat{x}_1, \quad x_2 \approx \hat{x}_2, \quad \dots, \quad x_d \approx \hat{x}_d$$

$M < d$  или используется разреженность

### 5.5.3. Ограниченная машина Больцмана



*Метод сравнительного отклонения (Contrastive Divergence)*

**begin**

Наблюдаемым состояниям присваиваются значения  $v = (v_1, v_2, \dots, v_M)$

По ним вычисляются значения скрытого слоя  $h = (h_1, h_2, \dots, h_L)$

По  $h$  вычисляются новые значения наблюдаемого слоя  $v' = (v'_1, v'_2, \dots, v'_M)$

По  $v'$  вычисляются новые значения скрытого слоя  $h' = (h'_1, h'_2, \dots, h'_L)$

$$W \leftarrow W + \alpha(vh^\top - v'h'^\top)$$

**end**

$$v_j, h_i \in \{0, 1\}, \quad \Pr \{h_i = 0\} = \sigma \left( \sum_{j=1}^M w_{ij} v_j \right), \quad \Pr \{v'_i = 0\} = \sigma \left( \sum_{i=1}^L w_{ij} h_i \right)$$

## 5.6. Программное обеспечение

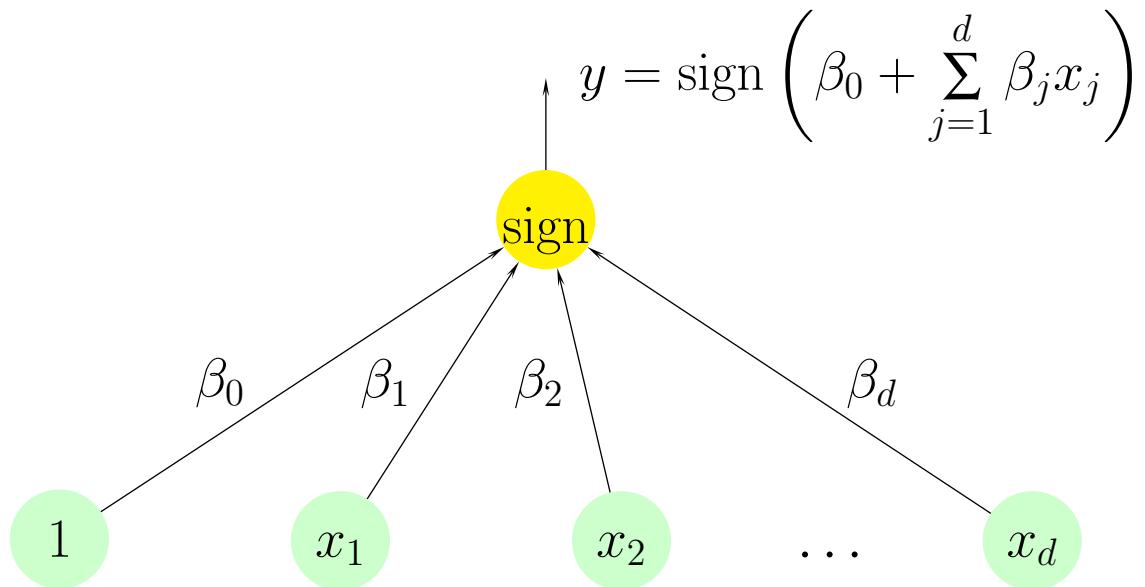
№	Пакет/библиотека	Интерфейс	ОС	FCNN	CNN	RNN	AE	RBM
1	TensorFlow	Python, Java, Go	Linux, Windows, Mac OS, Android	+	+	+	+	+
2	Theano	Python	Linux, Windows, Mac OS	+	+	+	+	+
3	Keras	Python, R	Linux, Vagrant	+	+	+	+	+
4	Torch	Lua, C	Linux, iOS, Android	+	+	+	+	+
5	Caffe	C++, Python, Matlab	Linux, Windows, OS X	+	+	+	+	-
6	MXNet	C++, Python, R, Scala, Julia, Perl, MATLAB, JavaScript	Linux, Windows, Mac OS	+	+	+	+	+
7	Matlab Neural Networks Toolbox	Matlab	Linux, Windows, OS X	+	+	+	+	-
8	Wolfram Mathematica	Java, C++	Linux, Windows, OS X	+	+	-	+	+

## 5.7.\* Персепtron Розенблатта

Модель нейрона и нейронной сети [W. S. McCulloch, W. Pitts, 1943]

Идеи обучения нейронной сети [D. O. Hebb, 1949]

Алгоритм обучения, программа [1958], нейрокомпьютер MARK 1 [F. Rosenblatt, 1960]



Будем строить линейную разделяющую поверхность (гиперплоскость).

Рассмотрим случай  $K = 2$  классов, которые будем кодировать числами 1 и -1.

Разделяющую гиперплоскость зададим уравнением  $\beta_0 + \beta^\top x = 0$ , а решающее правило – формулой  $f(x) = \text{sign}(\beta_0 + \beta^\top x)$ .

Для нахождения гиперплоскости будем минимизировать кусочно-линейную функцию

$$D(\beta, \beta_0) = - \sum_{i \in \mathcal{M}} y^{(i)} (\beta_0 + \beta^\top x^{(i)}),$$

где  $\mathcal{M} = \mathcal{M}(\beta, \beta_0)$  — множество индексов тех объектов, которые были классифицированы неправильно (т.е.  $\beta_0 + \beta^\top x^{(i)} < 0$ ).

Каждое слагаемое неотрицательно и пропорционально расстоянию от точки до гиперплоскости  $\beta_0 + \beta^\top x = 0$ .

$$D(\beta, \beta_0) = - \sum_{i \in \mathcal{M}} y^{(i)} (\beta_0 + \beta^\top x^{(i)}),$$

$$\frac{\partial D(\beta, \beta_0)}{\partial \beta_0} = - \sum_{i \in \mathcal{M}} y^{(i)}. \quad \frac{\partial D(\beta, \beta_0)}{\partial \beta} = - \sum_{i \in \mathcal{M}} y^{(i)} x^{(i)},$$

Воспользуемся алгоритмом *стохастического градиентного спуска*:

вместо вычисления градиента (как в обычном градиентном спуске), будем находить одно из слагаемых  $y^{(i)}x^{(i)}$  (и  $y^{(i)}$ ).

Объекты в обучающей выборке обходим в некотором порядке (например, случайно).

Если на  $i$ -м объекте текущее решающее правило ошибается, т. е.  $i \in \mathcal{M}$ , то обновляем веса ( $\mathcal{M}$  обновляется после каждого изменения коэффициентов  $\beta_0, \beta$ ):

$$\begin{pmatrix} \beta_0^{(r+1)} \\ \beta^{(r+1)} \end{pmatrix} \leftarrow \begin{pmatrix} \beta_0^{(r)} \\ \beta^{(r)} \end{pmatrix} + \rho \begin{pmatrix} y^{(i)} \\ y^{(i)}x^{(i)} \end{pmatrix},$$

где  $\rho$  — некоторый положительный параметр (например,  $\rho = 1$ ).

Обучение происходит до тех пор, пока  $\beta_0, \beta$  не стабилизируются (или не пройдет слишком много итераций).

**Теорема 5.2** (A. B. J. Novikoff, 1962) Если классы линейно отделимы, то за конечное число шагов алгоритм построит разделяющую гиперплоскость.

ДОКАЗАТЕЛЬСТВО. Пусть к  $x^{(i)}$  приписана первая координата, равная 1, и  $\beta^{(0)} = 0$ ,  $\rho = 1$ ,  $\|x^{(i)}\| \leq \gamma$ . Пусть классы разделяются гиперплоскостью  $\beta^{*\top} x = 0$ , причем

$$\beta^{*\top} x^{(i)} \geq \alpha > 0 \Leftrightarrow y^{(i)} = 1, \quad \beta^{*\top} x^{(i)} \leq -\alpha < 0 \Leftrightarrow y^{(i)} = -1, \quad \|\beta^*\| = 1$$

$$\beta^{*\top} \beta^{(1)} = \beta^{*\top} (\beta^{(0)} + \rho y^{(i_1)} x^{(i_1)}) = \beta^{*\top} y^{(i_1)} x^{(i_1)} \geq \alpha$$

$$\beta^{*\top} \beta^{(2)} = \beta^{*\top} (\beta^{(1)} + \rho y^{(i_2)} x^{(i_2)}) = \beta^{*\top} \beta^{(1)} + \beta^{*\top} y^{(i_2)} x^{(i_2)} \geq 2\alpha$$

$$\beta^{*\top} \beta^{(r)} \geq r\alpha$$

$$\|\beta^{(1)}\|^2 = \|x^{(i_1)} y^{(i_1)}\|^2 \leq \gamma^2,$$

$$\|\beta^{(2)}\|^2 = \|\beta^{(1)} + x^{(i_2)} y^{(i_2)}\|^2 = \underbrace{\|\beta^{(1)}\|^2}_{\leq \frac{\gamma^2}{r}} + 2 \underbrace{\beta^{(1)\top} x^{(i_2)} y^{(i_2)}}_{< 0} + \underbrace{\|x^{(i_2)}\|^2}_{\leq \gamma^2} < 2\gamma^2$$

$$\|\beta^{(r)}\|^2 < \sum_{\ell=1}^r \|x^{(i_\ell)}\|^2 \leq r\gamma^2$$

$$\cos \angle(\beta^{(*)}, \beta^{(r)}) = \frac{\beta^{*\top} \beta^{(r)}}{\|\beta^*\| \cdot \|\beta^{(r)}\|} > \frac{r\alpha}{\sqrt{r}\gamma} = \frac{\sqrt{r}\alpha}{\gamma} \rightarrow \infty \text{ (если } r \rightarrow \infty \text{)} — \text{противоречие!} \blacksquare$$

Количество итераций:

$$\frac{\sqrt{r}\alpha}{\gamma} < 1 \quad \Rightarrow \quad r_{\max} = \frac{\gamma^2}{\alpha^2}.$$

## Недостатки

1. Персепtron работает крайне медленно.
2. Если данные линейно не разделимы, то не известно, какую именно разделяющую гиперплоскость из возможных найдет алгоритм: это зависит от начального значения параметров  $\beta$ ,  $\beta_0$ , от порядка обхода точек и от величины  $\rho$ .
3. Если данные линейно не разделимы, то алгоритм может зациклиться, причем, как правило, наблюдается большая длина цикла, и зацикливание трудно распознать.

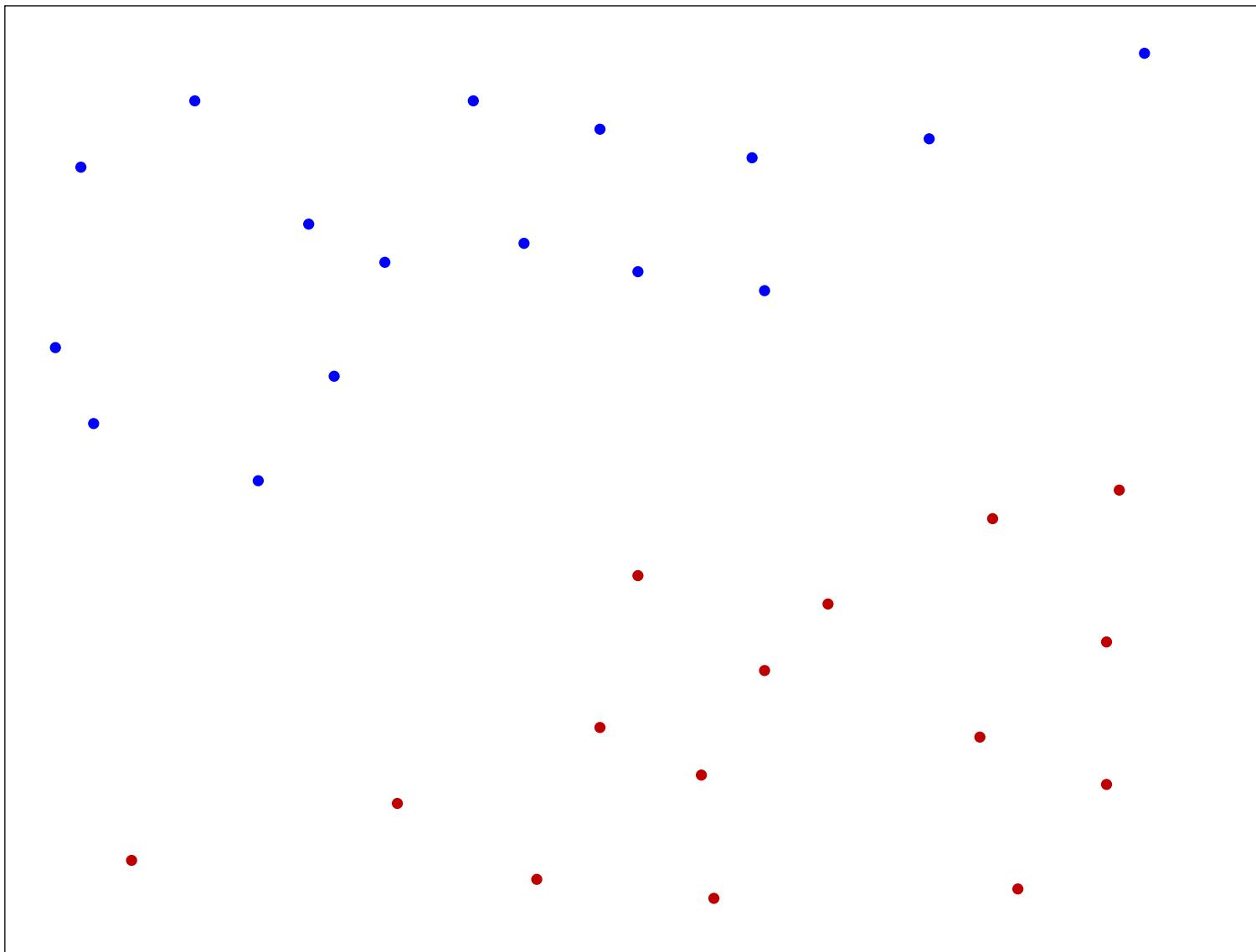
# *Глава 6*

## **Машина опорных векторов**

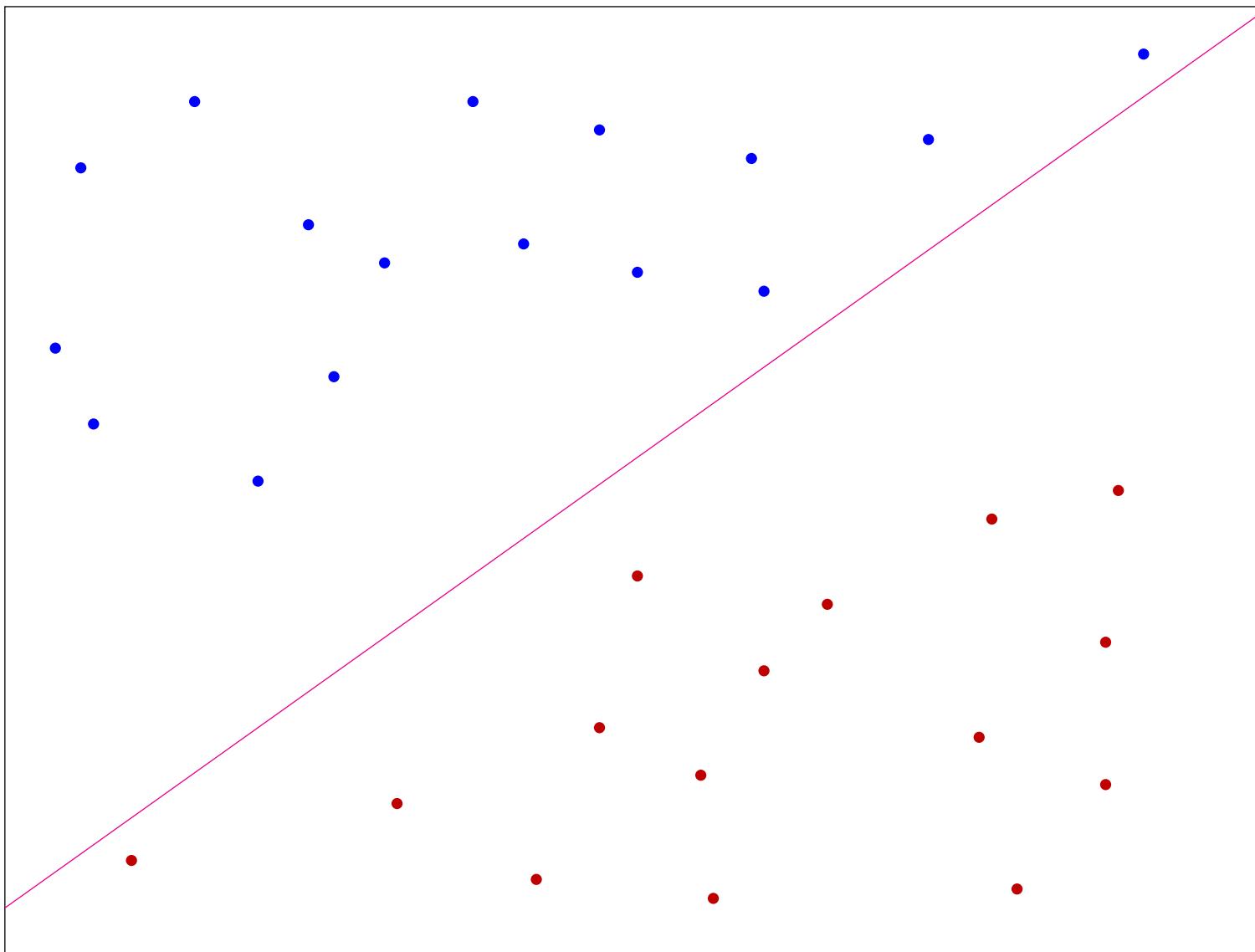
SVM – Support Vector Machine

- Метод обобщенного портрета (оптимальная разделяющая гиперплоскость) — 60–70 гг. В. Н. Вапник и др., см. В. Н. Вапник, А. Я. Червоненкис «Теория распознавания образов». М.: Наука, 1974
- Добавлены ядра – [Cortes, Vapnik, 1995]

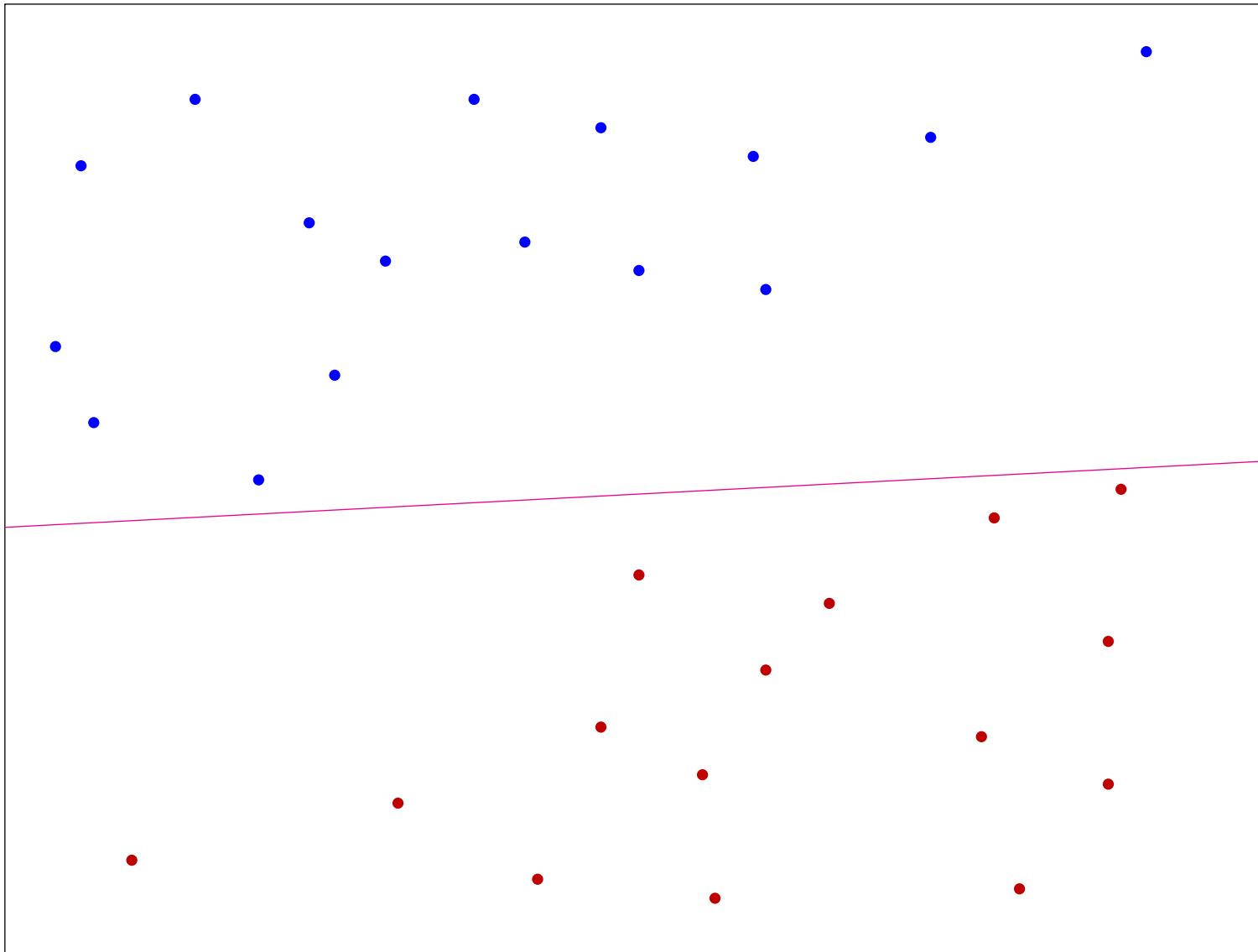
## **6.1. Оптимальная разделяющая гиперплоскость**



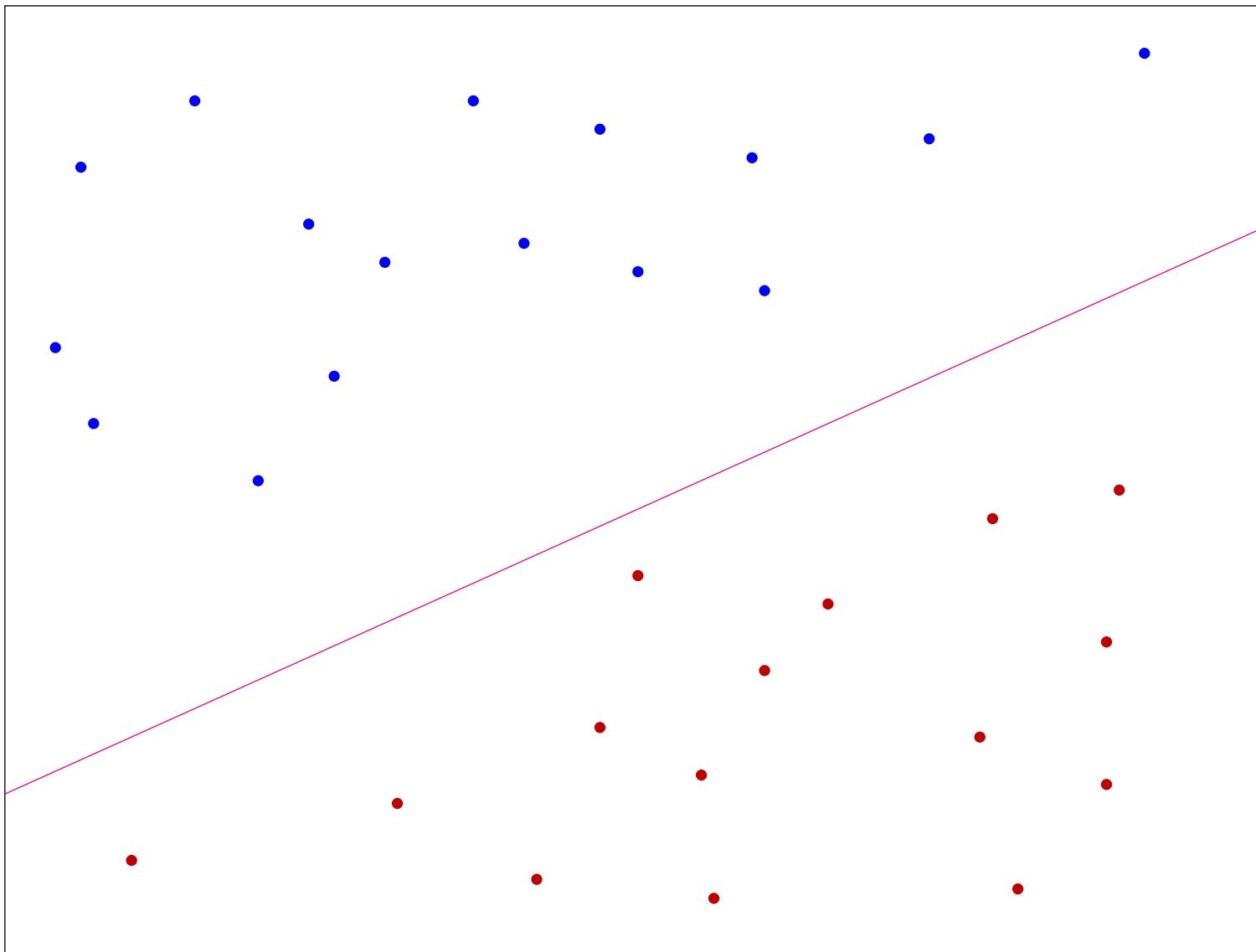
Два класса



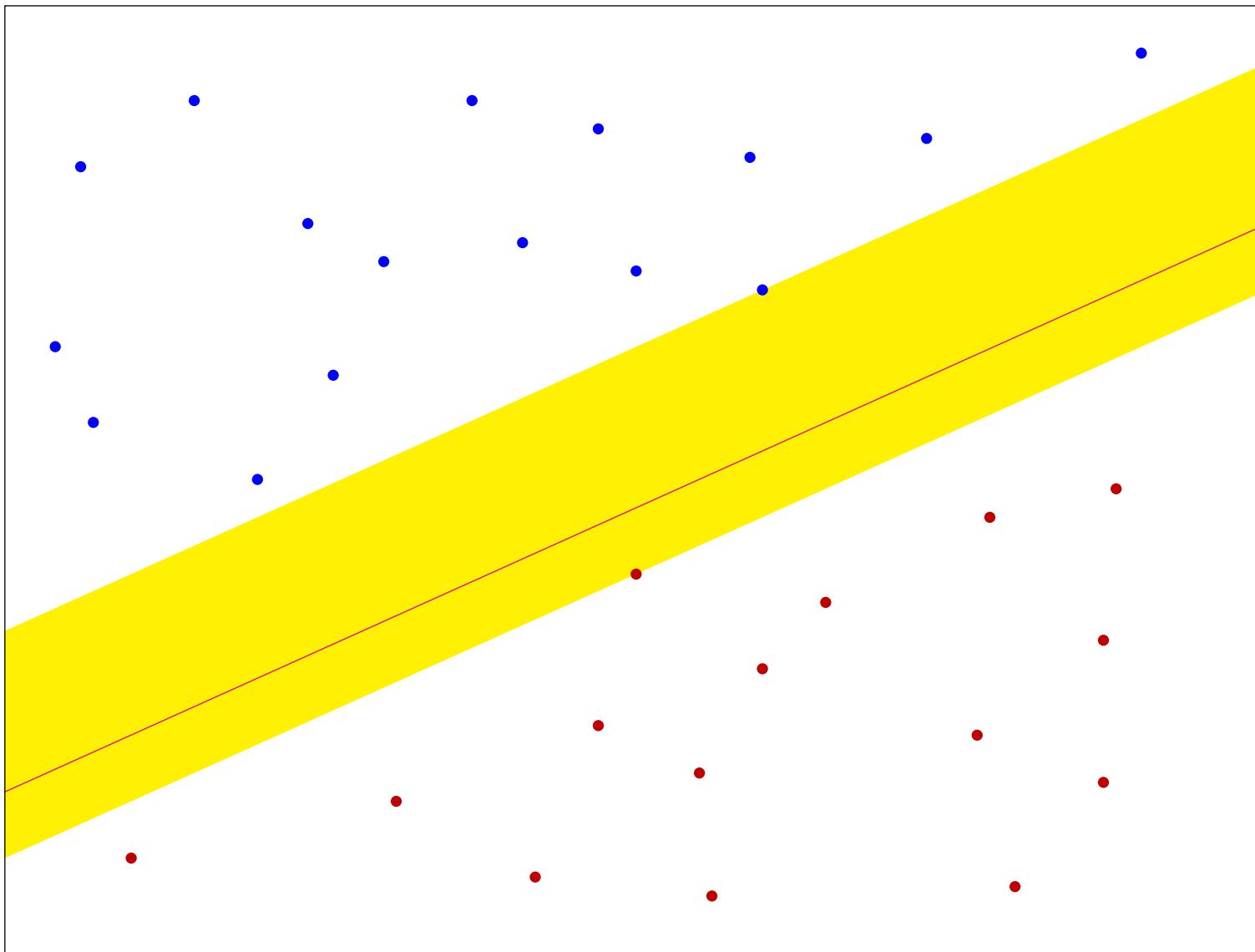
Одна из возможных разделяющих прямых



Другая из возможных разделяющих прямых



Еще одна из возможных разделяющих прямых (гиперплоскостей)

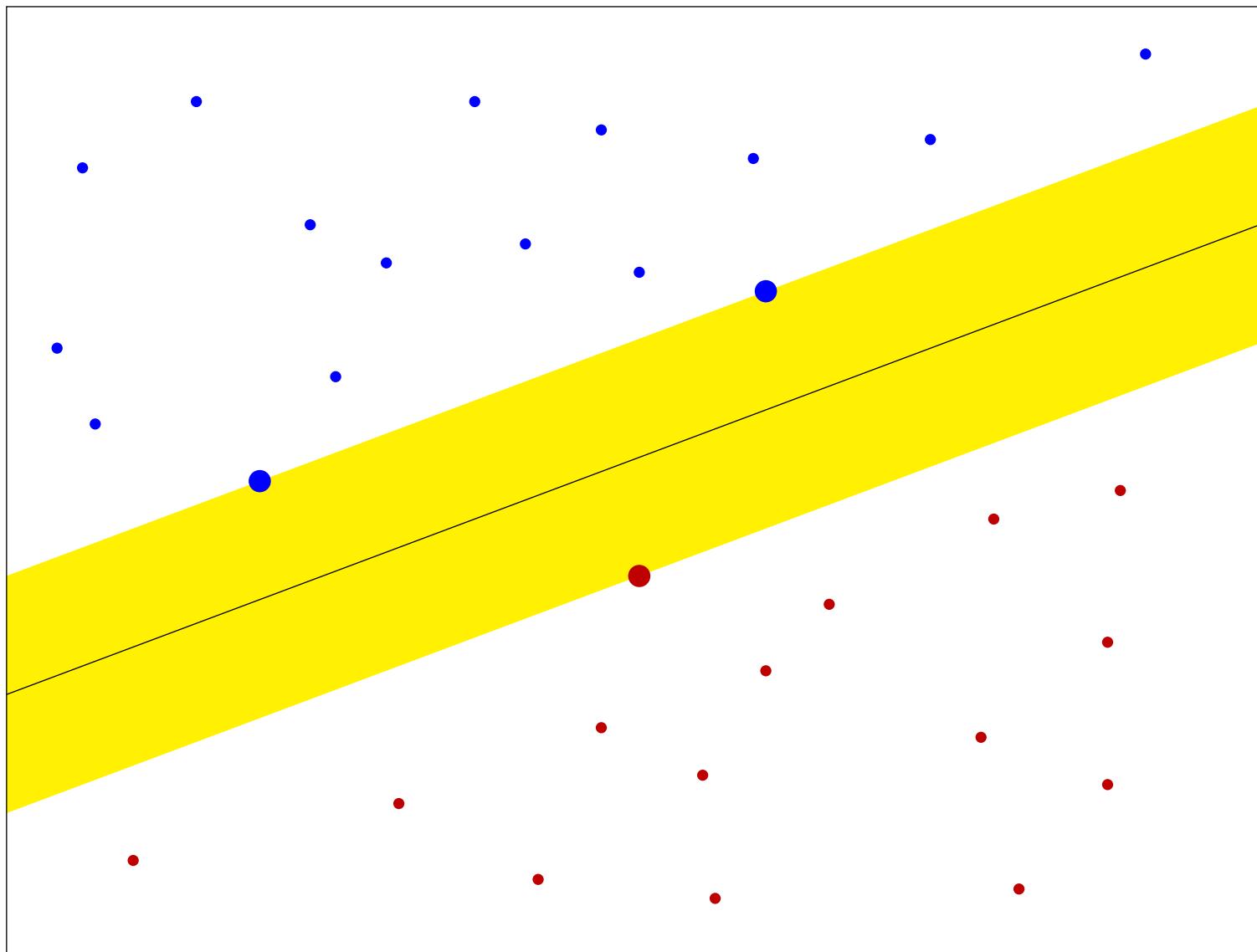


Желтая – разделяющая («нейтральная») полоса

*Оптимальная разделяющая гиперплоскость* — это гиперплоскость, максимизирующая ширину разделяющей полосы и лежащая в середине этой полосы.

Иными словами, оптимальная разделяющая гиперплоскость максимизирует *зазор* (*margin*) между плоскостью и данными из обучающей выборки.

Если классы линейно разделимы и каждый содержит не менее одного элемента, то оптимальная разделяющая гиперплоскость единственна.



Задача поиска оптимальной гиперплоскости эквивалентна следующей:

$$\max_{\beta, \beta_0} C$$

при ограничениях

$$y^{(i)}(\beta^\top x^{(i)} + \beta_0) \geq C \quad (i = 1, 2, \dots, N)$$

$$\|\beta\| = 1$$

или, что эквивалентно,

$$\min_{\beta, \beta_0} \frac{1}{2} \|\beta\|^2$$

при ограничениях

$$y^{(i)}(\beta^\top x^{(i)} + \beta_0) \geq 1 \quad (i = 1, 2, \dots, N).$$

Во втором случае зазор равен  $2/\|\beta\|$ .

Получили задачу выпуклого программирования (минимизация квадратичной функции при линейных ограничениях).

Запишем функцию Лагранжа

$$L(\beta_0, \beta, \lambda) = \frac{1}{2} \|\beta\|^2 + \sum_{i=1}^N \lambda_i \left( y^{(i)} (\beta^\top x^{(i)} + \beta_0) - 1 \right).$$

Приравнивая частные производные (по  $\beta_0$  и  $\beta$ ) нулю, получим:

$$0 = \sum_{i=1}^N \lambda_i y^{(i)}, \quad \beta = \sum_{i=1}^N \lambda_i y^{(i)} x^{(i)}.$$

Откуда

$$L(\lambda) = L(\beta_0, \beta, \lambda) = \frac{1}{2} \sum_{i=1}^N \sum_{k=1}^N \lambda_i \lambda_k y^{(i)} y^{(k)} x^{(i)\top} x^{(k)} - \sum_{i=1}^N \lambda_i.$$

Тем самым задача сведена к минимизации  $L(\lambda)$  при ограничениях  $\lambda_i \geq 0$  ( $i = 1, 2, \dots, N$ ).

На практике обычно решают именно эту задачу.

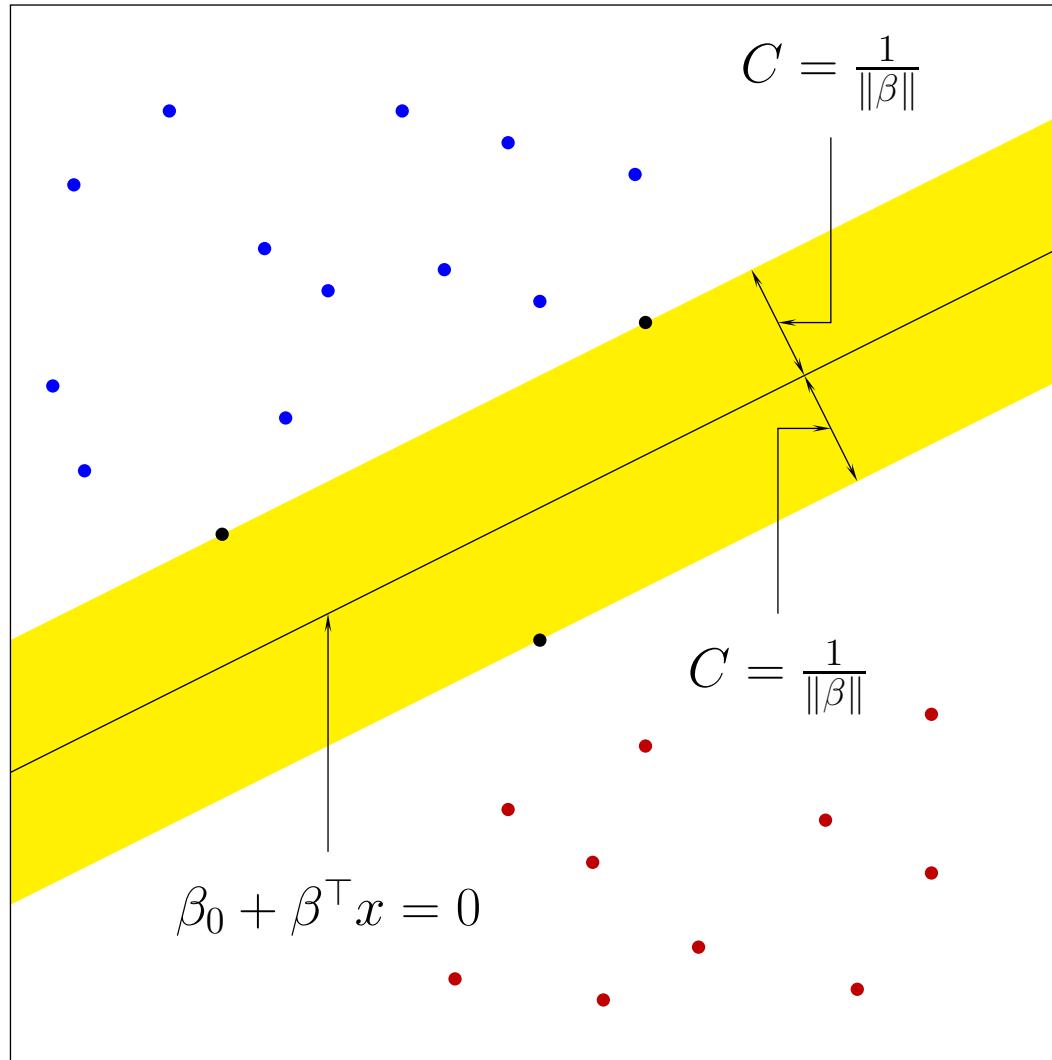
Решение задачи удовлетворяет условию Куна–Таккера (*дополняющей нежесткости*):

$$\lambda_i \left( y^{(i)} (\beta^\top x^{(i)} + \beta_0) - 1 \right) = 0 \quad (i = 1, 2, \dots, N),$$

откуда следует, что

- если  $\lambda_i > 0$ , то  $y^{(i)}(\beta^\top x^{(i)} + \beta_0) = 1$ , т. е.  $x^{(i)}$  лежит на границе разделяющей полосы
- если  $y^{(i)}(\beta^\top x^{(i)} + \beta_0) > 1$ , т. е.  $x^{(i)}$  не лежит на границе разд. полосы, то  $\lambda_i = 0$

Точки, для которых  $y^{(i)}(\beta^\top x^{(i)} + \beta_0) = 1$ , называются *опорными точками* или *опорными векторами*.



По построению, ни один объект из обучающей выборки не может попасть внутрь разделяющей полосы. Однако в нее могут попасть тестовые объекты. Тем не менее, даже в этом случае решающее правило, основанное на оптимальной разделяющей гиперплоскости, показывает, как правило, хорошие результаты.

Расположение оптимальной разделяющей гиперплоскости полностью определяется только опорными точками и не зависит от других точек обучающей выборки. Такая робастность отличает метод, основанный на оптимальных разделяющих гиперплоскостях, от *LDA*, в котором учитываются даже точки, расположенные далеко от границы. Конечно, для нахождения самих опорных точек мы должны принимать во внимание все точки обучающей выборки. Однако если данные действительно подчиняются нормальному закону с равными матрицами ковариации, то *LDA* — лучший выбор.

Разделяющая гиперплоскость, найденная с помощью логистической регрессии, часто близка к оптимальной разделяющей гиперплоскости. Это можно объяснить некоторой схожестью этих походов: на логистическую регрессию можно смотреть как на взвешенный метод наименьших квадратов, причем веса тем больше, чем ближе точка к границе.

Что делать, если данные линейно не разделимы?

## 6.2. Случай «перекрывающихся» классов

Предположим, что классы линейно-неразделимы. Рассмотрим задачу

$$\max_{\beta, \beta_0, \xi_i} C$$

при ограничениях

$$\|\beta\| = 1, \quad y^{(i)}(\beta^\top x^{(i)} + \beta_0) \geq C(1 - \xi_i), \quad \xi_i \geq 0 \quad (i = 1, 2, \dots, N), \quad \sum_{i=1}^n \xi_i \leq \Xi,$$

где  $\Xi$  — некоторая константа (параметр метода).

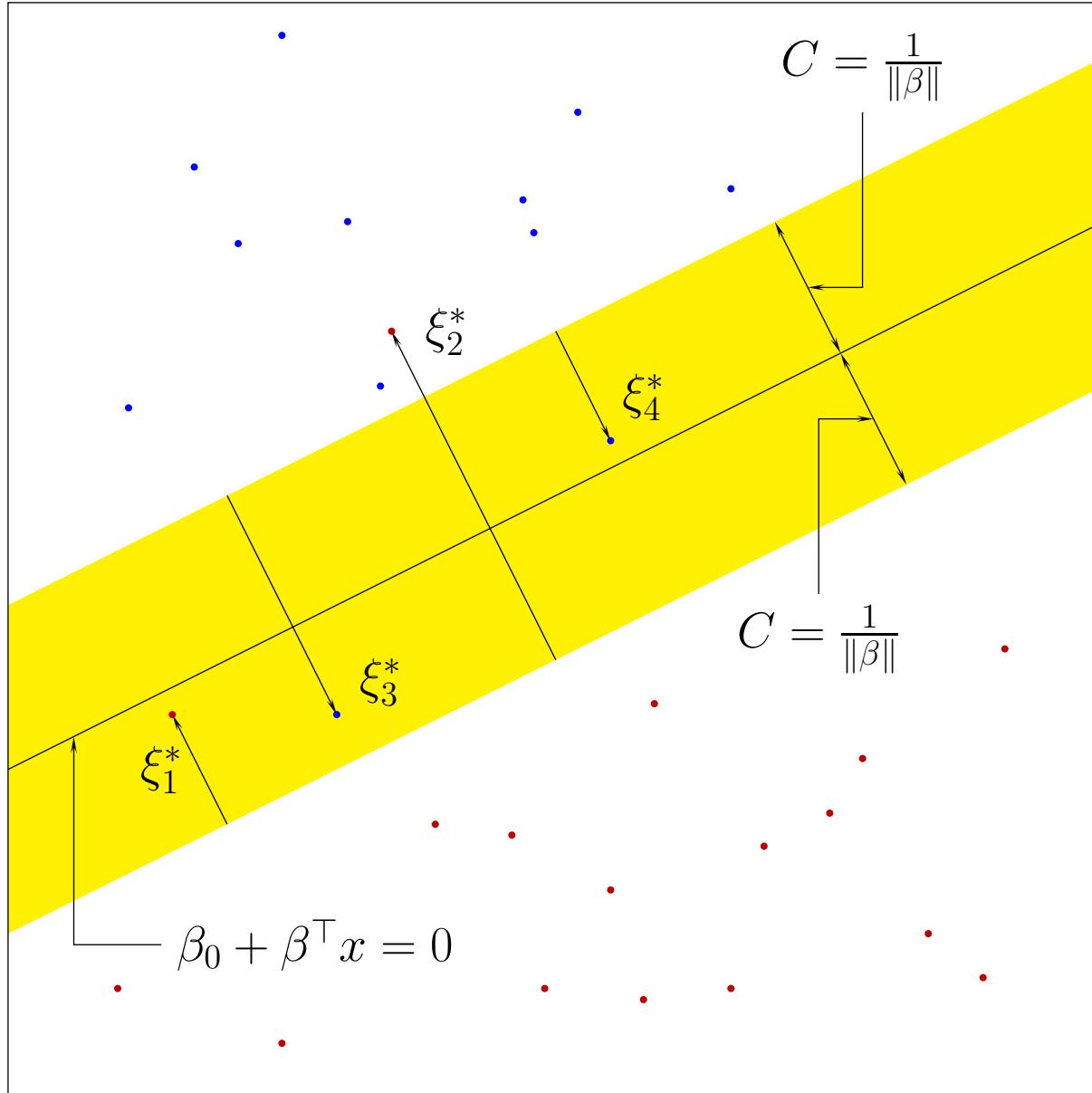
Случаю линейно-отделимых классов соответствует  $\Xi = 0$ .

$\xi_i$  пропорционально расстоянию, на которое  $x^{(i)}$  заходит за границу разделяющей полосы.

В частности,  $i$ -й объект будет классифицирован не правильно  $\Leftrightarrow \xi_i > 1$ .

Чем меньше  $\Xi$ , тем меньше объектов будет классифицировано неправильно.

Но параметр  $\Xi$  должен быть достаточно велик, чтобы задача была совместной.



Эквивалентная запись задачи:

$$\min_{\beta, \beta_0, \xi_i} \frac{1}{2} \|\beta\|^2,$$

при ограничениях

$$y^{(i)}(\beta^\top x^{(i)} + \beta_0) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad (i = 1, 2, \dots, N), \quad \sum_{i=1}^n \xi_i \leq \Xi.$$

Задача заключается в минимизации квадратичной положительно определенной функции при линейных ограничениях.

Запишем задачу в виде

$$\min_{\beta, \beta_0, \xi_i} \frac{1}{2} \|\beta\|^2 + \gamma \sum_{i=1}^n \xi_i,$$

при ограничениях

$$y_i(\beta^\top x^{(i)} + \beta_0) \geq 1 - \xi_i, \quad \xi_i \geq 0 \quad (i = 1, 2, \dots, N),$$

где  $\gamma = 1/\Xi$ . Случай линейно отделимых областей соответствует значению  $\gamma = \infty$ .

Функция Лагранжа для этой задачи имеет вид

$$L_P = \frac{1}{2} \|\beta\|^2 + \gamma \sum_{i=1}^n \xi_i - \sum_{i=1}^N \lambda_i \left( y^{(i)} (\beta^\top x^{(i)} + \beta_0) - (1 - \xi_i) \right) - \sum_{i=1}^N \mu_i \xi_i.$$

Положим производные относительно неизвестных  $\beta_0, \beta, \xi_i$  равными нулю:

$$0 = \sum_{i=1}^N \lambda_i y^{(i)}, \quad \beta = \sum_{i=1}^N \lambda_i y^{(i)} x^{(i)}, \quad \lambda_i = \gamma - \mu_i \quad (i = 1, 2, \dots, N).$$

Подставляя эти формулы в  $L_P$ , получим двойственную функцию Лагранжа

$$L_D = \frac{1}{2} \sum_{i=1}^N \sum_{k=1}^N \lambda_i \lambda_k y^{(i)} y^{(k)} x^{(i)\top} x^{(k)} - \sum_{i=1}^N \lambda_i.$$

$L_D$  необходимо минимизировать при ограничениях

$$0 \leq \lambda_i \leq \gamma, \quad \sum_{i=1}^N \lambda_i y^{(i)} = 0.$$

Условия Куна–Таккера (дополняющей нежесткости):

$$\lambda_i \left( y^{(i)} (\beta^\top x^{(i)} + \beta_0) - (1 - \xi_i) \right) = 0,$$

$$\mu_i \xi_i = 0$$

Пусть  $\beta, \beta_0, \xi_i, \lambda_i$  и т. д. — оптимальные значения соответствующих неизвестных.

Если  $\lambda_i \neq 0$ , то  $y^{(i)}(\beta^\top x^{(i)} + \beta_0) = 1 - \xi_i$ , т. е.  $i$ -е неравенство выполнено как равенство.

Таким образом, в формуле

$$\beta = \sum_{i=1}^N \lambda_i y^{(i)} x^{(i)},$$

в правой части остаются только слагаемые, соответствующие точкам, для которых  $y^{(i)}(x^{(i)^\top} \beta + \beta_0) = 1 - \xi_i$ .

Эти точки (объекты) называются *опорными векторами*, так как  $\beta$  зависит только от них.

Среди этих точек некоторые могут лежать на границе разделяющей полосы ( $\xi_i = 0$ ).

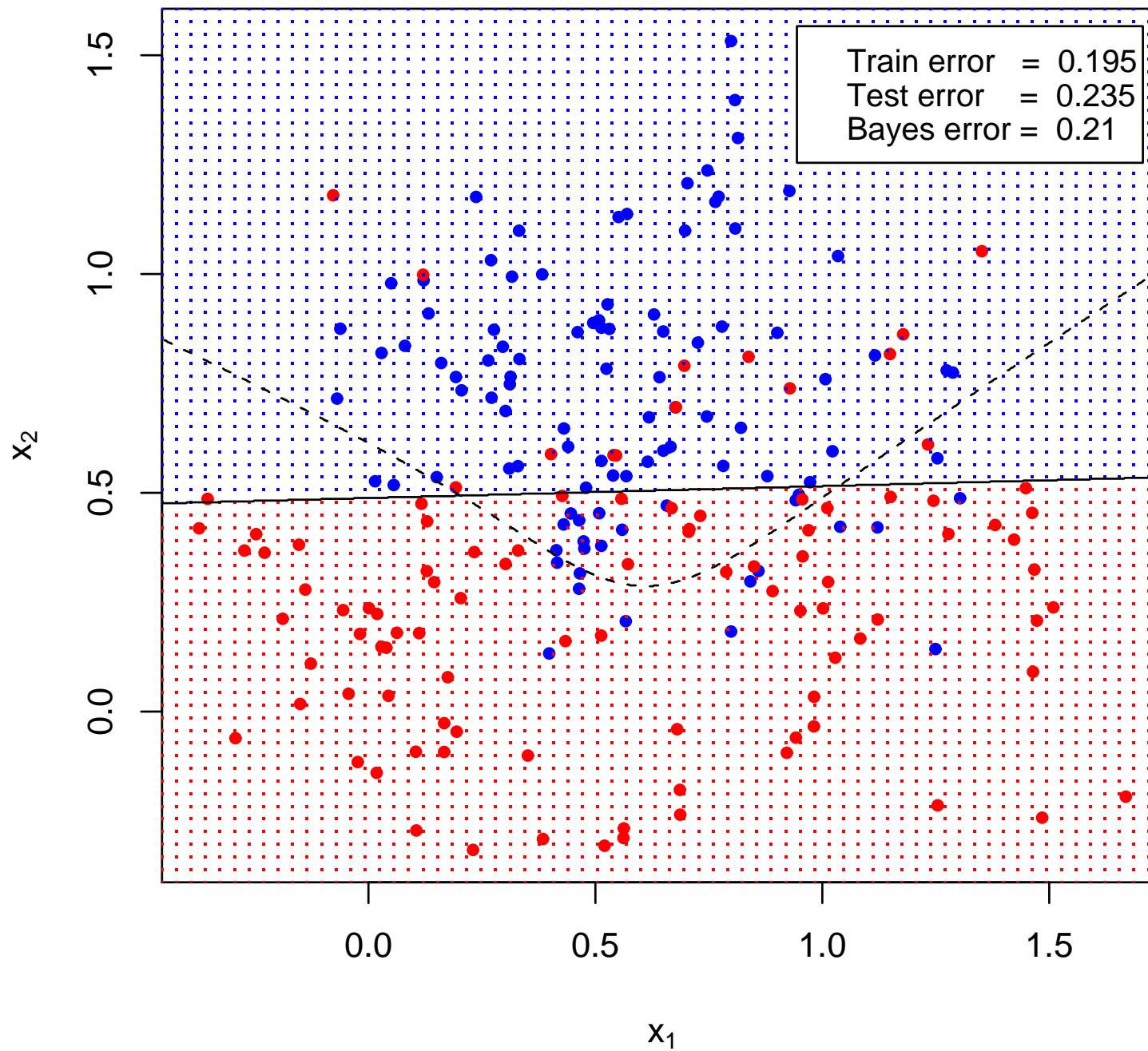
Для них  $0 < \lambda_i < \gamma$ . Для остальных опорных точек  $\xi_i > 0$  и  $\lambda_i = \gamma$ .

Любая точка на границе разделяющей полосы может использоваться для определения  $\beta_0$ . На практике в качестве  $\beta_0$  берется среднее из всех значений, определяемых по опорным векторам.

Итак, на расположение разделяющей гиперплоскости влияют только опорные точки.

Это выгодно отличает данный классификатор от *LDA*, в котором граница областей определяется матрицами ковариации и расположением центроидов, и, следовательно, зависит от всех точек.

В рассматриваемом отношении классификатор опорных векторов больше похож на логистическую регрессию.



## 6.3. Ядра и спрямляющие пространства

Перейдем от исходного пространства  $\mathcal{X}$  в другое, называемое *спрямляющим*,  $\mathcal{H}$  с помощью некоторого отображения

$$h(x) = \left( h_1(x), \dots, h_M(x) \right),$$

где  $h_m(x)$  – базисные функции (новые признаки) ( $m = 1, 2, \dots, M$ ).

Новый классификатор определяется теперь функцией

$$f(x) = \text{sign} (\beta^\top h(x) + \beta_0).$$

В формуле

$$\beta = \sum_{i=1}^N \lambda_i y^{(i)} x^{(i)},$$

заменяем  $x^{(i)}$  на  $h(x^{(i)})$ :

$$\beta = \sum_{i=1}^N \lambda_i y^{(i)} h(x^{(i)}),$$

Двойственная функция Лагранжа

$$L_D = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \lambda_i \lambda_j y^{(i)} y^{(j)} x^{(i)\top} x^{(j)} - \sum_{i=1}^N \lambda_i.$$

примет вид

$$L_D = \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \lambda_i \lambda_j y^{(i)} y^{(j)} \langle h(x^{(i)}), h(x^{(j)}) \rangle - \sum_{i=1}^N \lambda_i.$$

Функция  $f(x)$  запишется как

$$f(x) = \text{sign} (\beta^\top h(x) + \beta_0) = \text{sign} \left( \sum_{i=1}^N \lambda_i y^{(i)} \langle h(x), h(x^{(i)}) \rangle + \beta_0 \right).$$

Мы видим, что  $h(x)$  встречается только в скалярном произведении  $\langle h(x), h(x^{(i)}) \rangle$ !

Таким образом, для определения классификатора опорных векторов нам достаточно уметь вычислять лишь функцию

$$K(x, x') = \langle h(x), h(x') \rangle.$$

Итак, мы можем заменить скалярное произведение функцией  $K(x, x')$  и, более того, вообще явно не строить спрямляющего пространства  $\mathcal{H}$ , а подбирать функцию  $K$ .

Можно совсем отказаться от построения новых признаков, а попробовать построить модель, в которой описываются взаимоотношения между объектами с помощью функции  $K(x, x')$ .

Рассмотрим необходимые и достаточные условия, которым должна удовлетворять функция  $K(x, x')$ .

Функция  $K : \mathcal{X} \times \mathcal{X} \rightarrow \mathbf{R}$  называется **ядром**, если она представима в виде  $K(x, x') = \langle h(x), h(x') \rangle$  при некотором отображении  $h : \mathcal{X} \rightarrow \mathcal{H}$ , где  $\mathcal{H}$  – евклидово (или гильбертово) пространство со скалярным произведением  $\langle \cdot, \cdot \rangle$ .

**Теорема 6.1 (Мерсер)** *Функция  $K(x, x')$  является ядром тогда и только тогда, когда она симметрична, т. е.  $K(x, x') = K(x', x)$ , и неотрицательно определена, т. е.*

$$\int_{\mathcal{X}} \int_{\mathcal{X}} K(x, x') g(x) g(x') dx dx' \geq 0$$

для всех  $g(x)$ , для которых  $\int_{\mathcal{X}} g(x)^2 dx$  ограничено.

Примеры ядер:

полином степени  $d$ :  $K(x, x') = (1 + \langle x, x' \rangle)^d$ ,

радиальная функция:  $K(x, x') = e^{-\|x-x'\|^2/c}$ ,

сигмоидальная (масштабированная логистическая) функция:

$$K(x, x') = \tanh(\kappa_1 \langle x, x' \rangle + \kappa_2), \quad \text{где} \quad \tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

## Пример

Рассмотрим пространство признаков размерности 2 с двумя входами  $x_1, x_2$  и полиномиальным ядром степени 2:

$$K(x, x') = (1 + \langle x, x' \rangle)^2 = (1 + x_1 x'_1 + x_2 x'_2)^2 = 1 + 2x_1 x'_1 + 2x_2 x'_2 + (x_1 x'_1)^2 + (x_2 x'_2)^2 + 2x_1 x'_1 x_2 x'_2.$$

Мы видим, что  $M = 6$  и можно положить

$$h_1(x) = 1, \quad h_2(x) = \sqrt{2}x_1, \quad h_3(x) = \sqrt{2}x_2, \quad h_4(x) = x_1^2, \quad h_5(x) = x_2^2, \quad h_6(x) = \sqrt{2}x_1 x_2.$$

Тогда  $K(x, x') = \langle h(x), h(x') \rangle$ .

Исследуем роль параметра  $\gamma$  в

$$\min_{\beta, \beta_0, \xi_i} \frac{1}{2} \|\beta\|^2 + \gamma \sum_{i=1}^n \xi_i,$$

учитывая, что классы в спрямляющем пространстве, как правило, можно разделить.

Большое значение  $\gamma$  препятствует росту  $\xi_i$ , что, как правило, приводит к переобучению и крайне извилистой границе между областями.

Маленькое значение  $\gamma$  способствует росту  $\xi_i$  и обычно приводит к более «прямой» разделяющей поверхности.

На SVM можно смотреть как на регуляризованный метод минимизации эмпирического риска, если в качестве штрафной функции рассматривается

$$L(g, y) = [1 - yg]_+,$$

где

$f(x) = \text{sign } g(x)$  — классификатор,

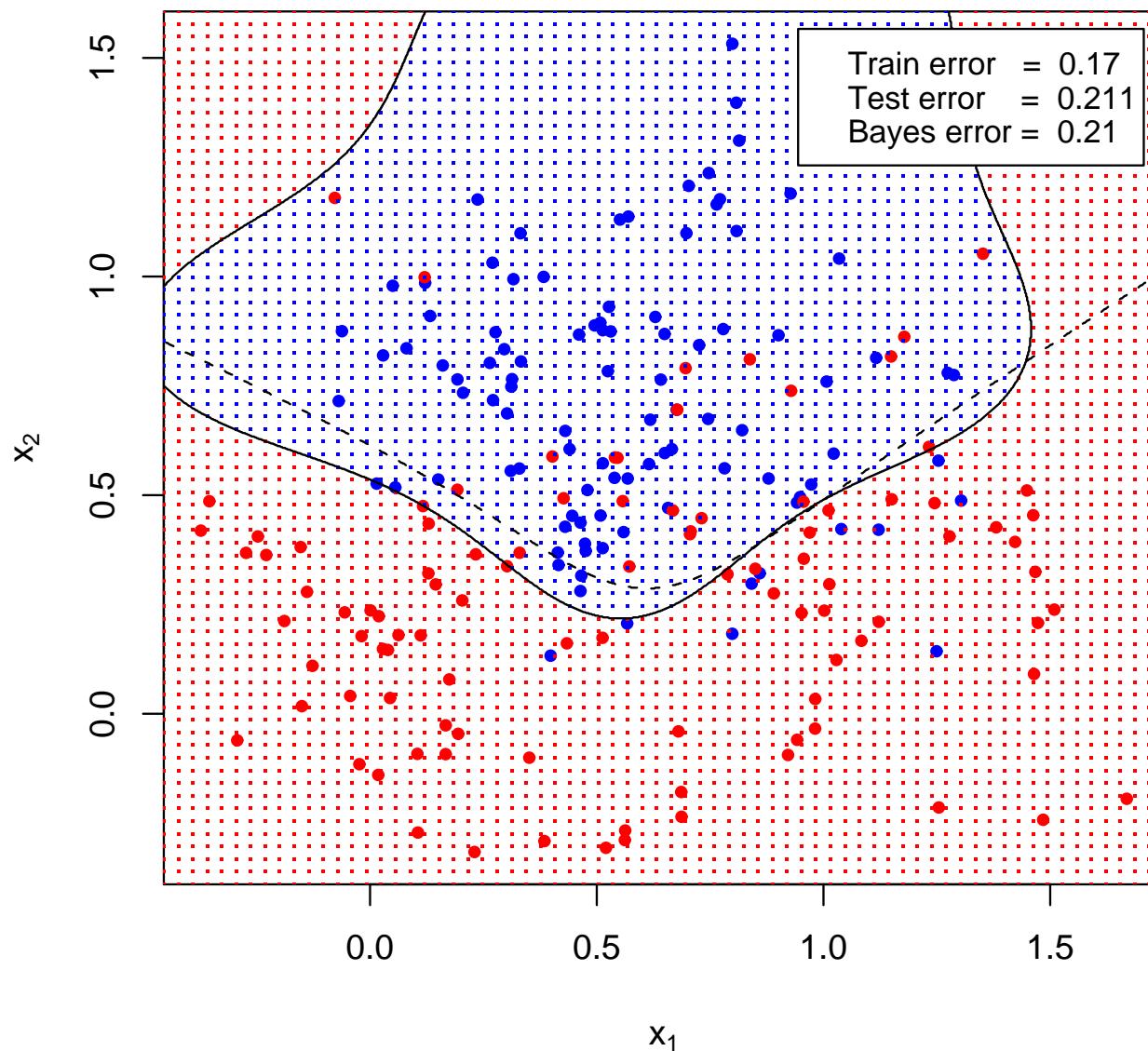
$g(x) = h(x)^\top \beta + \beta_0$  — отступ (margin) объекта  $x$ .

SVM эквивалентен минимизации

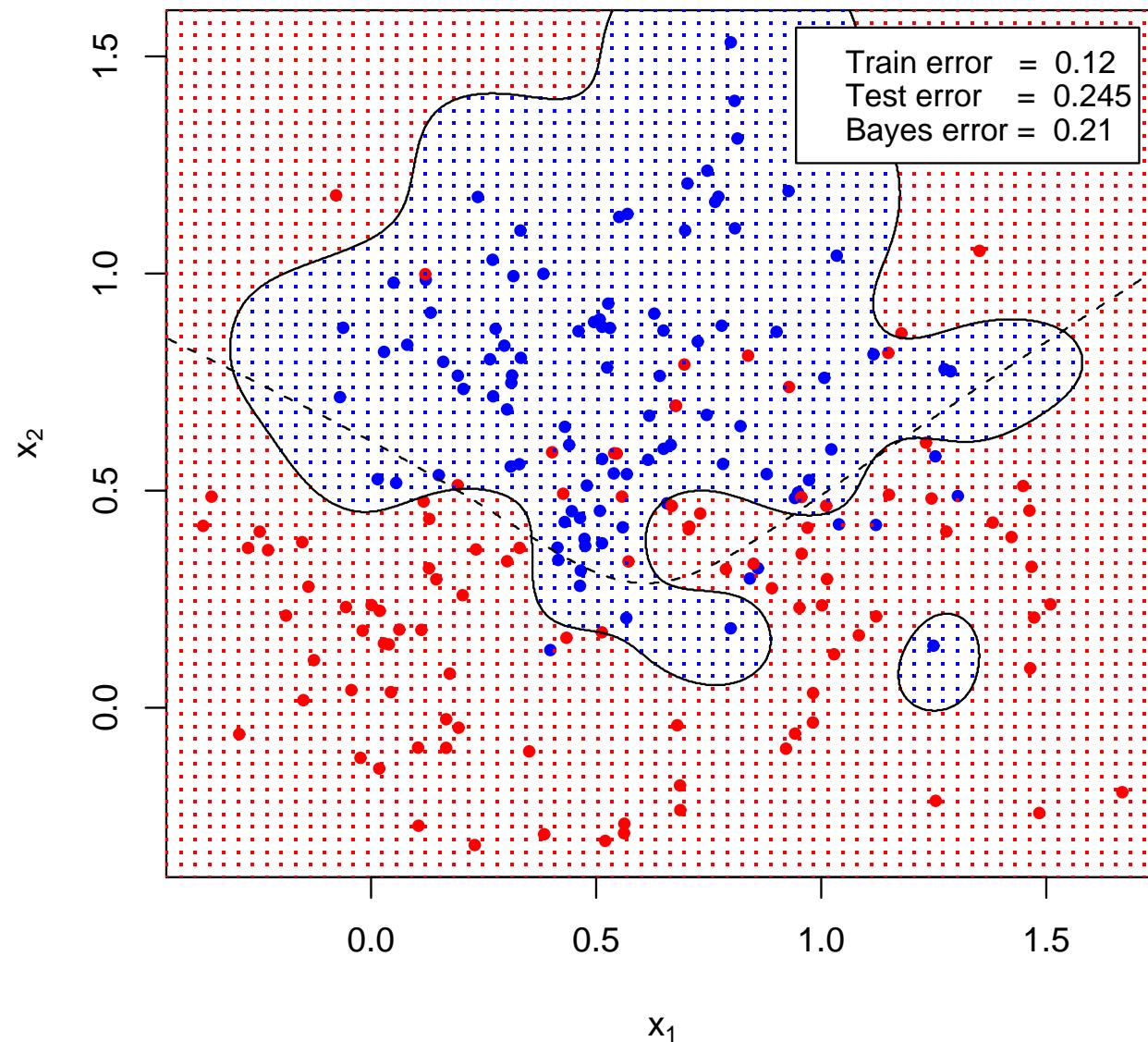
$$\sum_{i=1}^N [1 - y^{(i)} (h(x^{(i)})^\top \beta + \beta_0)]_+ + \frac{1}{2\gamma} \|\beta\|^2.$$

Целевая функция имеет вид «потери + штраф».

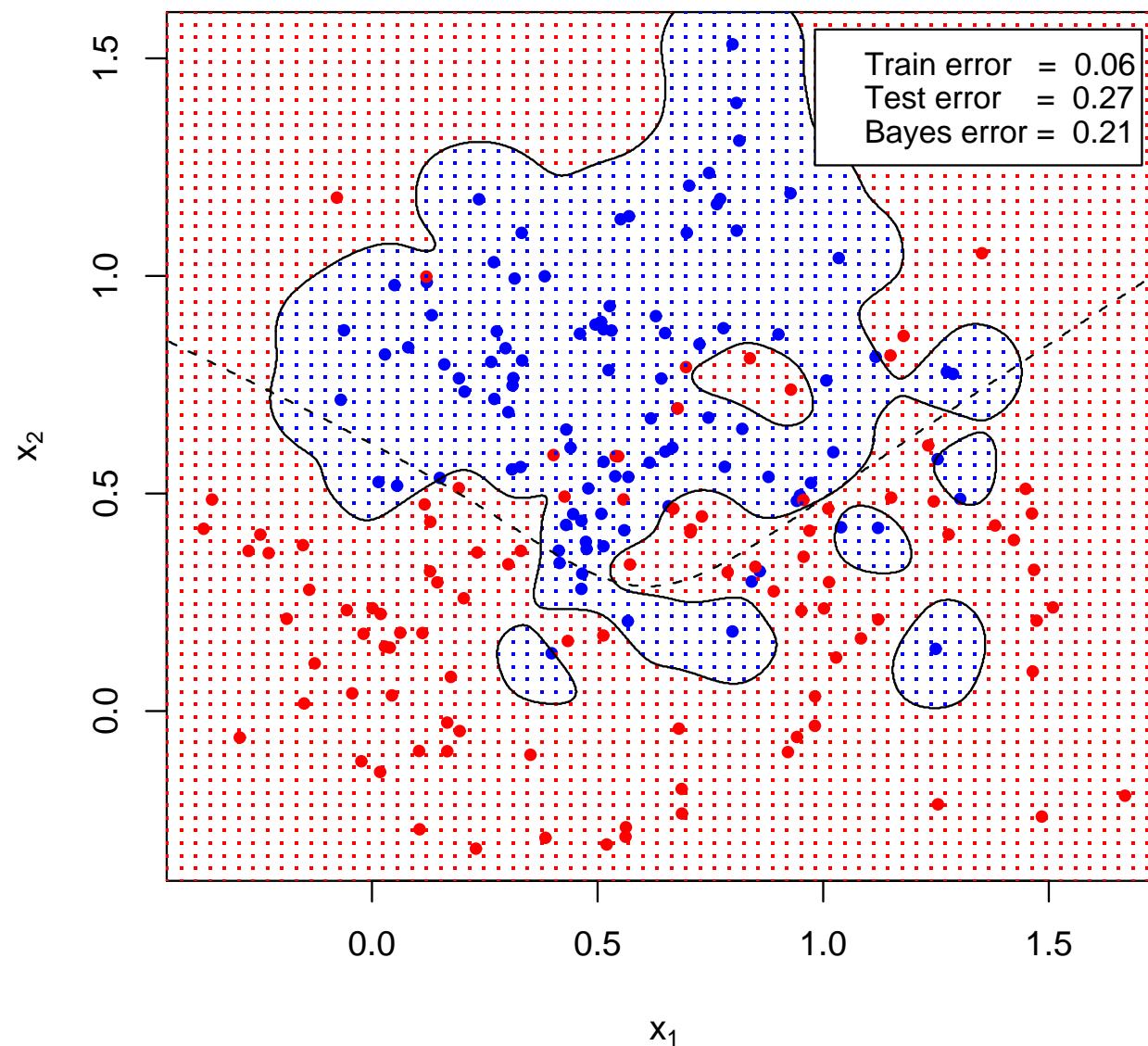
## SVM радиальное ядро, $\gamma = 1/2$



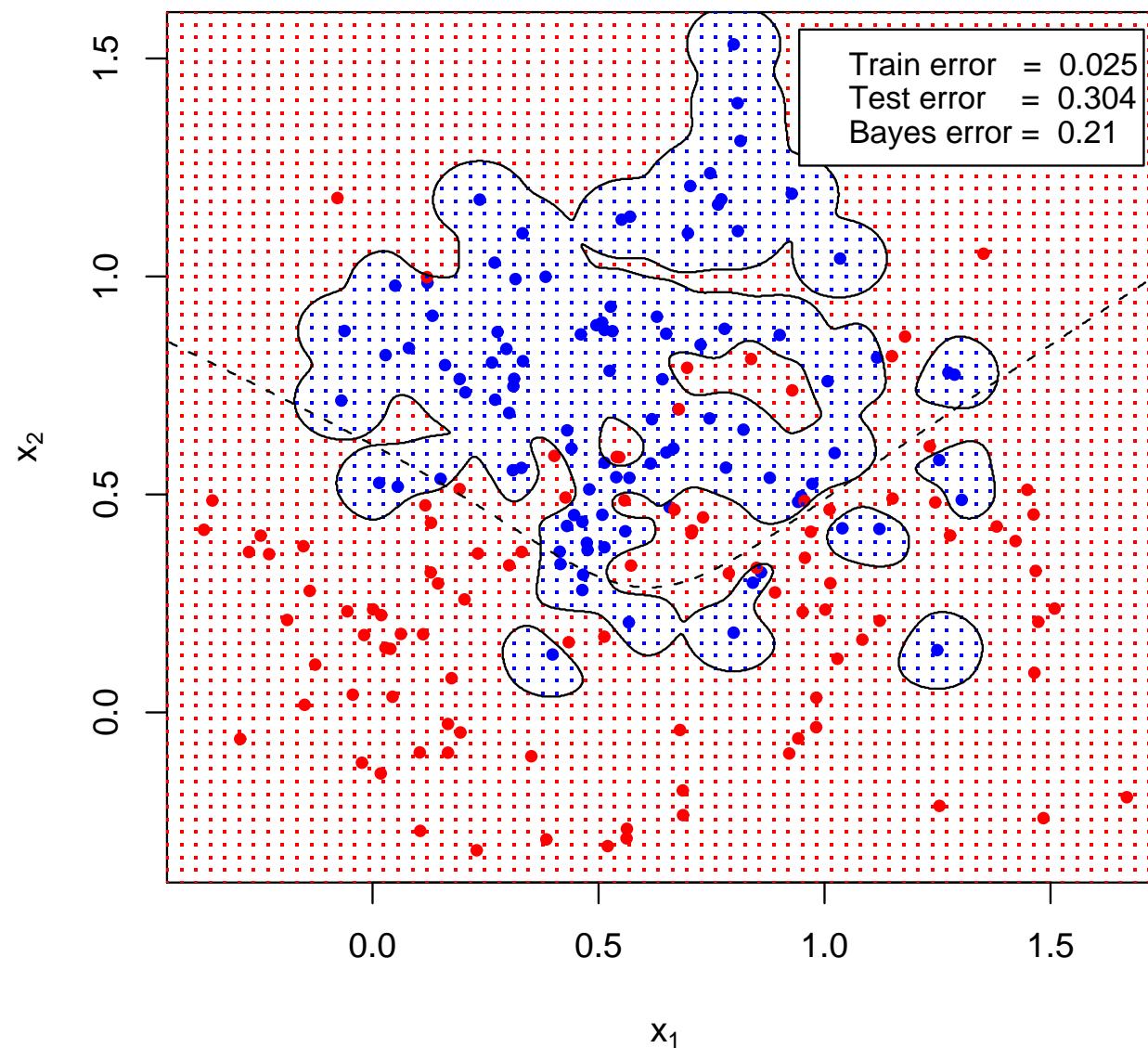
## SVM радиальное ядро, $\gamma = 5$



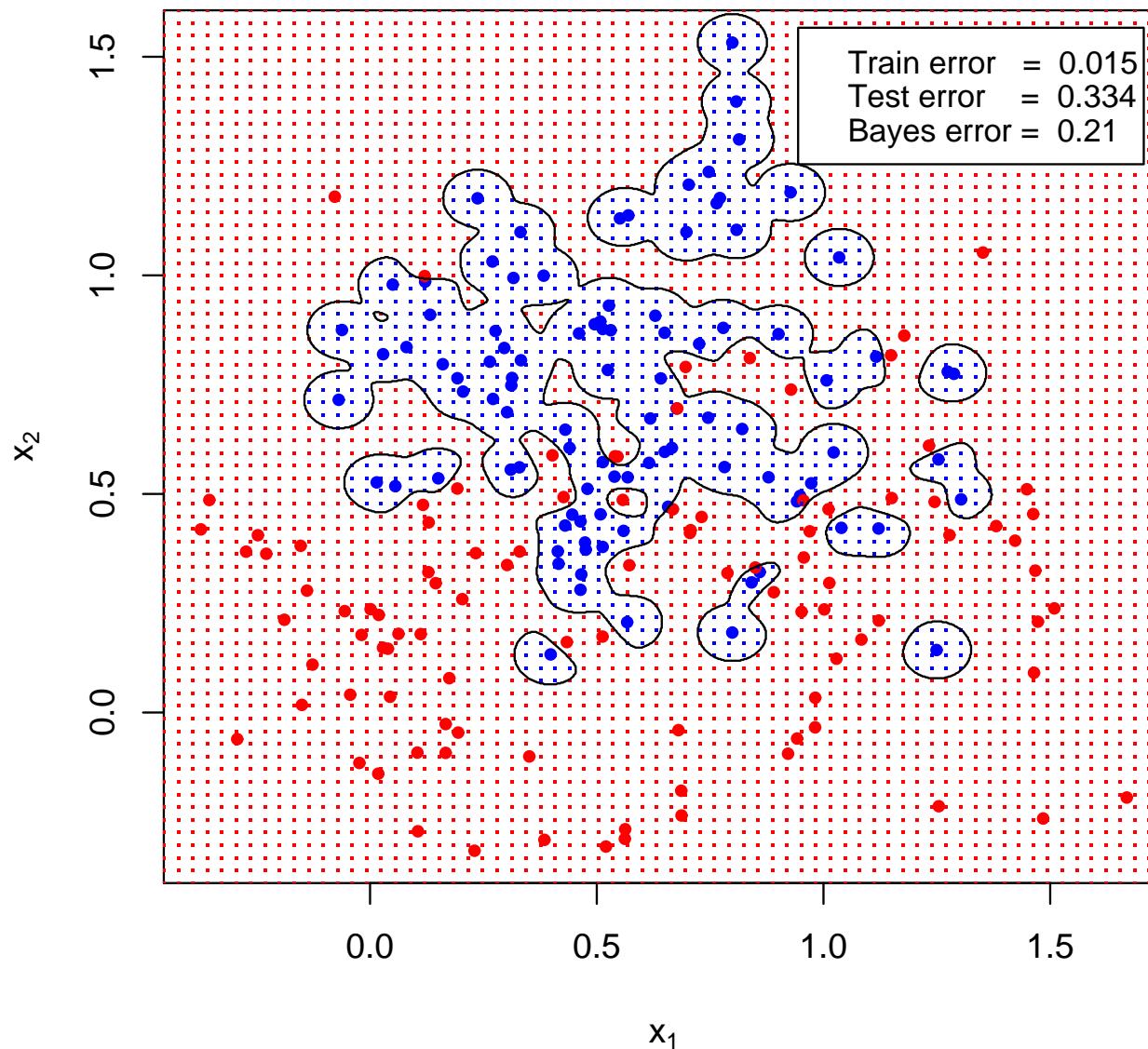
## SVM радиальное ядро, $\gamma = 20$



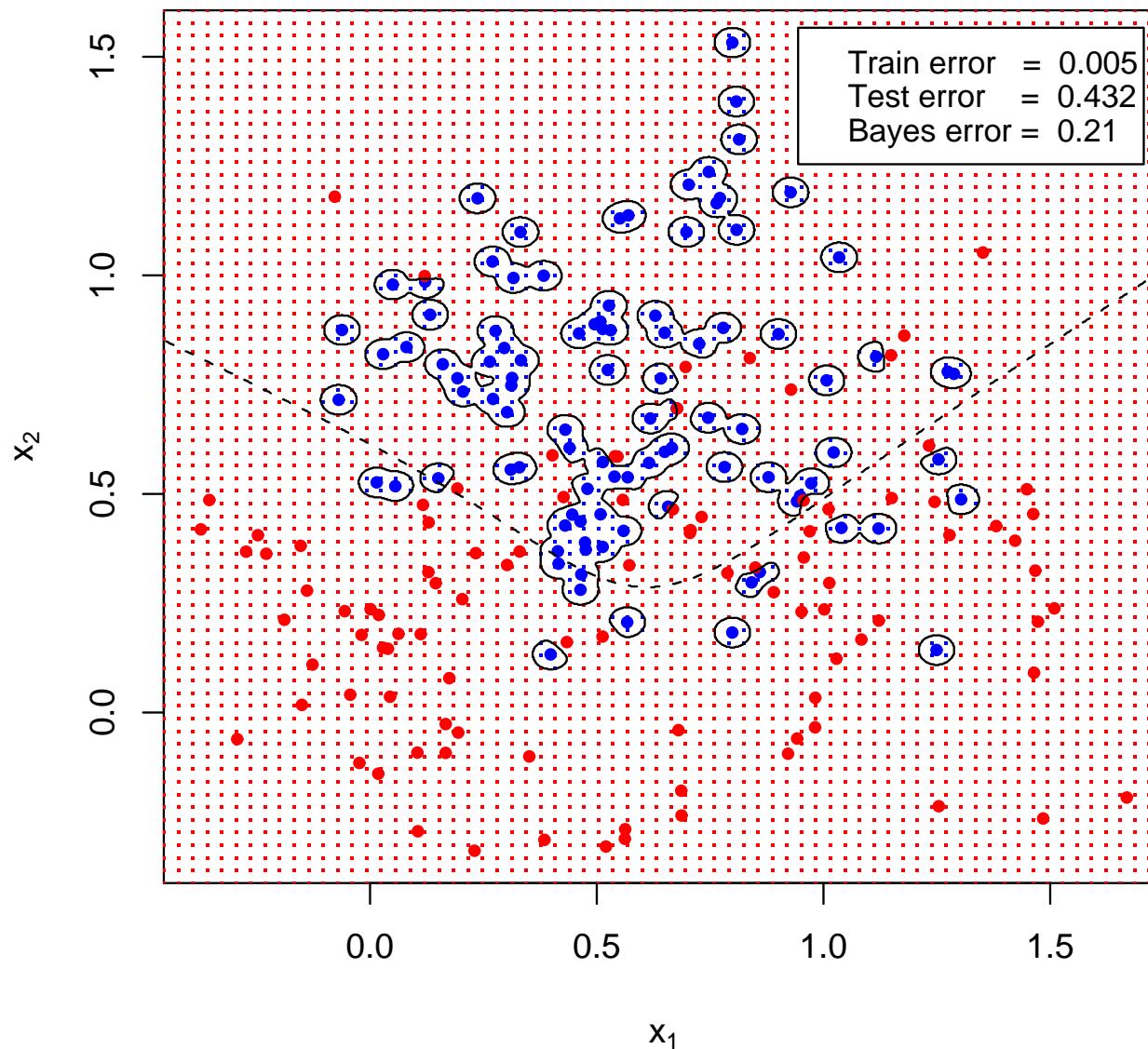
## SVM радиальное ядро, $\gamma = 50$



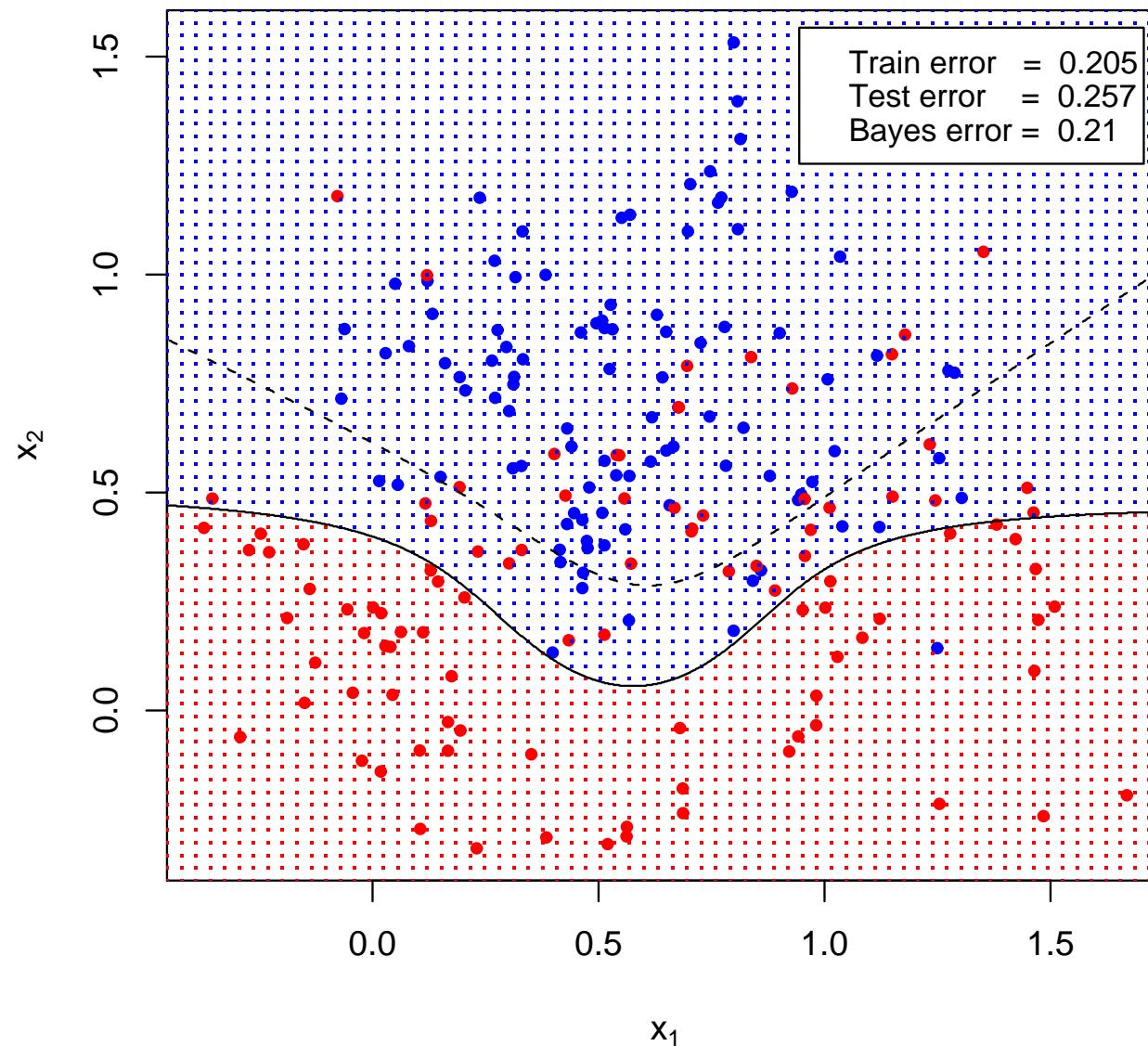
## SVM радиальное ядро, $\gamma = 100$



SVM радиальное ядро,  $\gamma = 500$



## SVM полиномиальное ядро (полином 3-й степени)

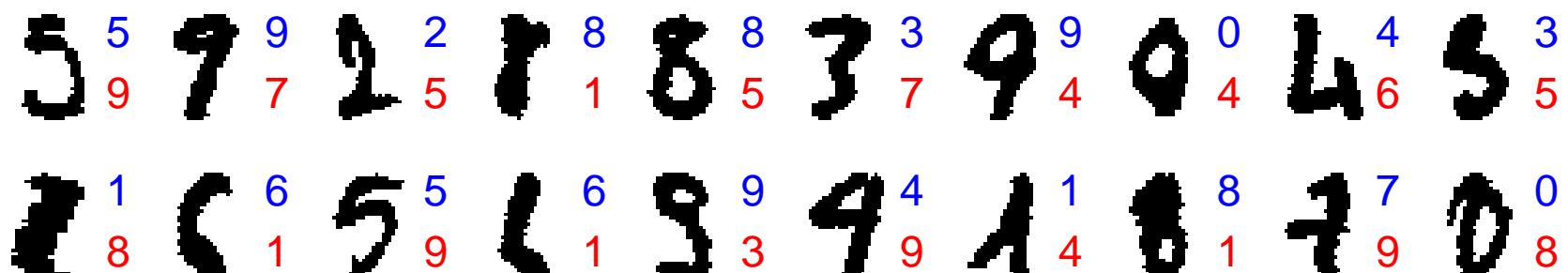


Задача классификации рукописных цифр. Выборка размера 1934 была случайным образом разбита на две группы по 967 объектов в каждой.  $\gamma = 1/1024$

Ошибки на обучающей и тестовой выборках приведена в следующей таблице.

Ядро	Ошибка	
	на обучающей выборке	на тестовой выборке
Линейное	0	0.021
Радиальное	0.011	0.030
Полином 3 степени	0.090	0.094
Сигмоидальное	0.131	0.125

Все случаи неправильной классификации цифр из тестовой выборки в случае линейного ядра.



## 6.4. SVM и восстановление регрессии

Как SVM можно адаптировать к решению задачи восстановления регрессии?

Сначала рассмотрим линейную регрессионную модель

$$f(x) = \beta^\top x + \beta_0.$$

Для восстановления  $\beta, \beta_0$  рассмотрим задачу минимизации функции

$$H(\beta, \beta_0) = \sum_{i=1}^N V\left(y^{(i)} - f(x^{(i)})\right) + \frac{\alpha}{2}\|\beta\|^2,$$

$$V(t) = V_\varepsilon(t) = \begin{cases} 0, & \text{если } |t| < \varepsilon, \\ |t| - \varepsilon & \text{в противном случае.} \end{cases}$$

$V_\varepsilon(t)$  — функция « $\varepsilon$ -нечувствительности», игнорирующая ошибки, меньшие  $\varepsilon$ .

Можно провести аналогию с SVM-классификатором, в которой точки, расположенные далеко от разделяющей полосы (с «правильной» стороны) не рассматриваются при построении классификатора.

В случае с регрессией такую роль играют точки с маленькой ошибкой  $|y^{(i)} - f(x^{(i)})|$ .

Можно показать, что решение  $\widehat{\beta}$ ,  $\widehat{\beta}_0$ , минимизирующее функцию  $H(\beta, \beta_0)$ , можно представить в виде

$$\widehat{\beta} = \sum_{i=1}^N (\widehat{\alpha}_i^* - \widehat{\alpha}_i) x^{(i)}, \quad \widehat{f}(x) = \sum_{i=1}^N (\widehat{\alpha}_i^* - \widehat{\alpha}_i) \langle x, x^{(i)} \rangle + \beta_0,$$

где  $\widehat{\alpha}_i$  и  $\widehat{\alpha}_i^*$  являются решением следующей задачи квадратичного программирования:

$$\min_{\alpha_i, \alpha_i^*} \left( \varepsilon \sum_{i=1}^N (\alpha_i^* + \alpha_i) - \sum_{i=1}^N y^{(i)} (\alpha_i^* - \alpha_i) + \frac{1}{2} \sum_{i,j=1}^N (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) \langle x^{(i)}, x_j \rangle \right)$$

при ограничениях

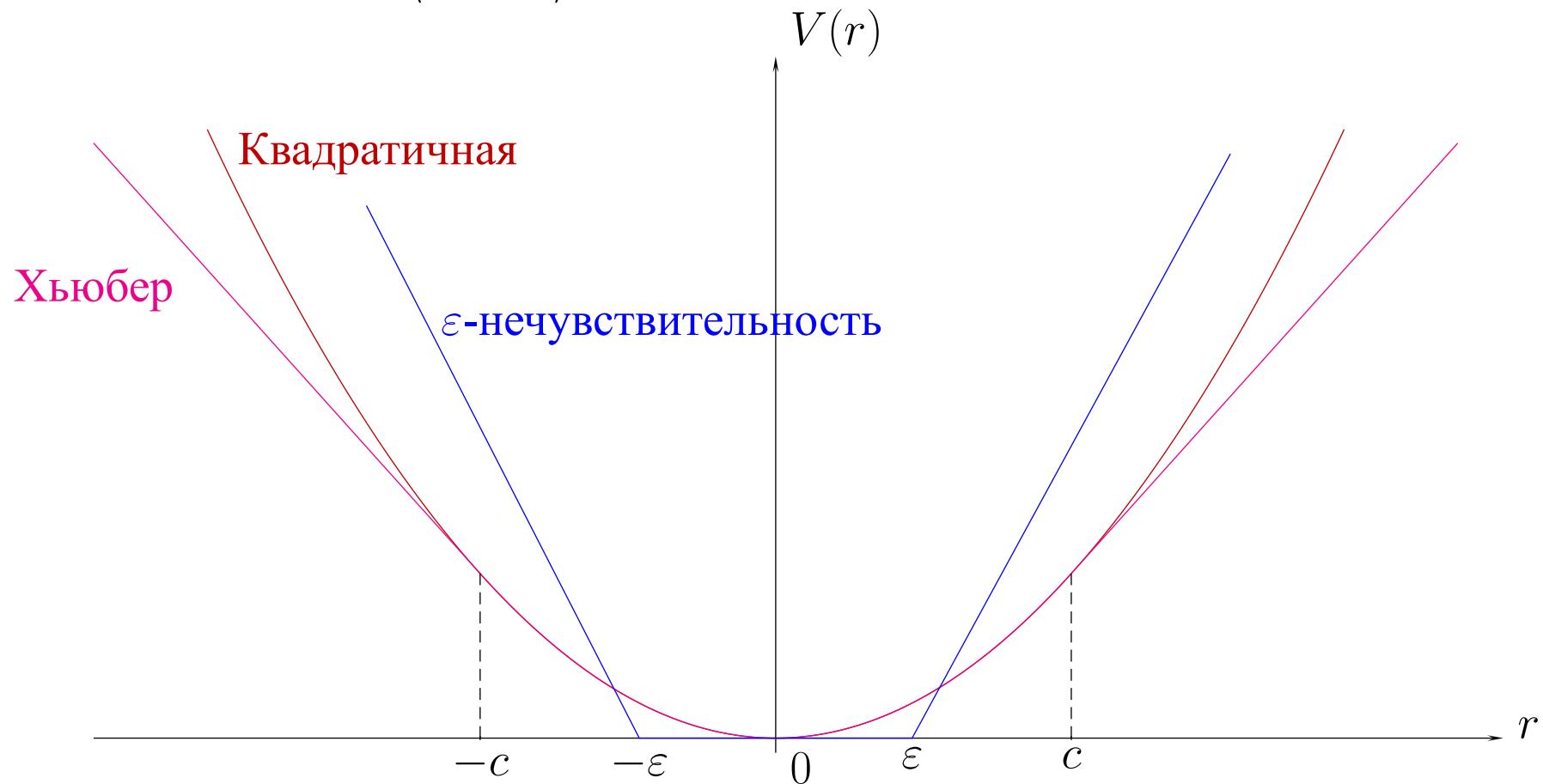
$$0 \leq \alpha_i, \alpha_i^* \leq \frac{1}{\lambda}, \quad \sum_{i=1}^N (\alpha_i^* - \alpha_i) = 0, \quad \alpha_i \alpha_i^* = 0.$$

Как и в случае задачи классификации, здесь решение зависит только от скалярных произведений  $\langle x^{(i)}, x_j \rangle$ , поэтому мы можем использовать аппарат спрямляющих пространств и ядер.

В качестве функции  $V(t)$ , можно выбрать другую меру ошибки, например, квадратичную  $V(t) = t^2$  или функцию Хьюбера

$$V_H(r) = \begin{cases} r^2/2, & \text{если } |r| < c, \\ c|r| - c^2/2 & \text{в противном случае,} \end{cases}$$

но важно свести все к задаче квадратичного программирования, содержащей только скалярные произведения  $\langle x^{(i)}, x_j \rangle$ .



## 6.5. Регрессия и ядра

SVM — не единственная модель, в которой могут использоваться ядра.

Рассмотрим, например, задачу аппроксимации функции  $f(x)$  (т. е. задачу восстановления регрессии) при заданных базисных функциях  $h_1(x), \dots, h_m(x)$ :

$$f(x) = \sum_{m=1}^M \beta_m h_m(x) + \beta_0.$$

Будем минимизировать регуляризованную функцию потерь

$$H(\beta, \beta_0) = \sum_{i=1}^N V(y^{(i)} - f(x^{(i)})) + \frac{\lambda}{2} \|\beta\|^2.$$

Решение задачи минимизации имеет вид

$$\hat{f}(x) = \sum_{m=1}^M \hat{\beta}_m h_m(x) + \hat{\beta}_0 = \sum_{i=1}^N \hat{\alpha}_i K(x, x^{(i)}),$$

где

$$K(x, x') = \sum_{m=1}^M h_m(x) h_m(x').$$

Пусть, например,  $V(r) = r^2$ .

Пусть  $\mathbf{H} - (N \times M)$ -матрица, в которой  $(i, m)$ -й элемент есть  $h_m(x^{(i)})$ .

Предположим, что  $M > N$  и  $M$  велико.

Для простоты будем предполагать, что  $\beta_0 = 0$ .

Тогда

$$H(\beta) = (\mathbf{y} - \mathbf{H}\beta)^\top (\mathbf{y} - \mathbf{H}\beta) + \lambda \|\beta\|^2.$$

Минимизирует функцию  $H(\beta)$  вектор  $\widehat{\beta}$ , который можно определить из условий

$$-\mathbf{H}^\top (\mathbf{y} - \mathbf{H}\widehat{\beta}) + \lambda \widehat{\beta} = 0.$$

После очевидных преобразований получаем

$$\mathbf{H}\widehat{\beta} = (\mathbf{H}\mathbf{H}^\top + \lambda I)^{-1} \mathbf{H}\mathbf{H}^\top \mathbf{y}.$$

Матрица  $\mathbf{H}\mathbf{H}^\top$  размера  $N \times N$  содержит скалярные произведения для всех пар векторов  $\langle x^{(i)}, x_j \rangle$ .

Таким образом,  $(\mathbf{H}\mathbf{H}^\top)_{ij} = K(x^{(i)}, x_j)$ .

Легко показать, что

$$\widehat{f}(x) = h(x)^\top \beta = \sum_{i=1}^N \widehat{\alpha}_i K(x, x^{(i)}),$$

где  $\hat{\alpha} = (\mathbf{H}\mathbf{H}^\top + \mathbf{I})^{-1}\mathbf{y}$ .

Как и в SVM, нет необходимости вычислять  $M$  значений функций  $h_1(x), \dots, h_M(x)$  (и даже определять сами эти функции).

Достаточно вычислить только значения  $K(x^{(i)}, x_j)$ .

Удачный выбор функций  $h_m$  (например, если в качестве них выбраны собственные функции ядра) позволяет вычислить все эти значения за время  $N^2/2$ , а не  $N^2M$ , как при прямом умножении матриц.



## *Глава 7*

# **Деревья решений**

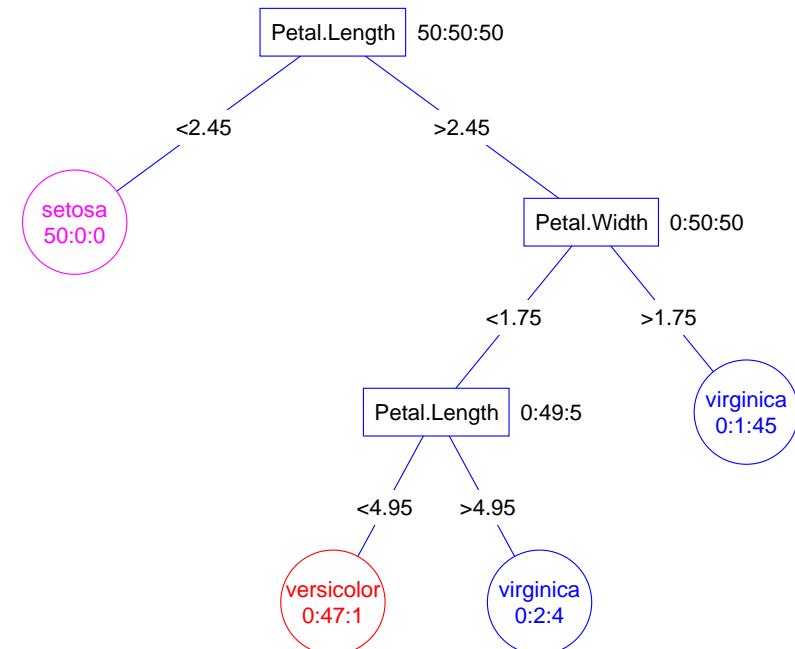
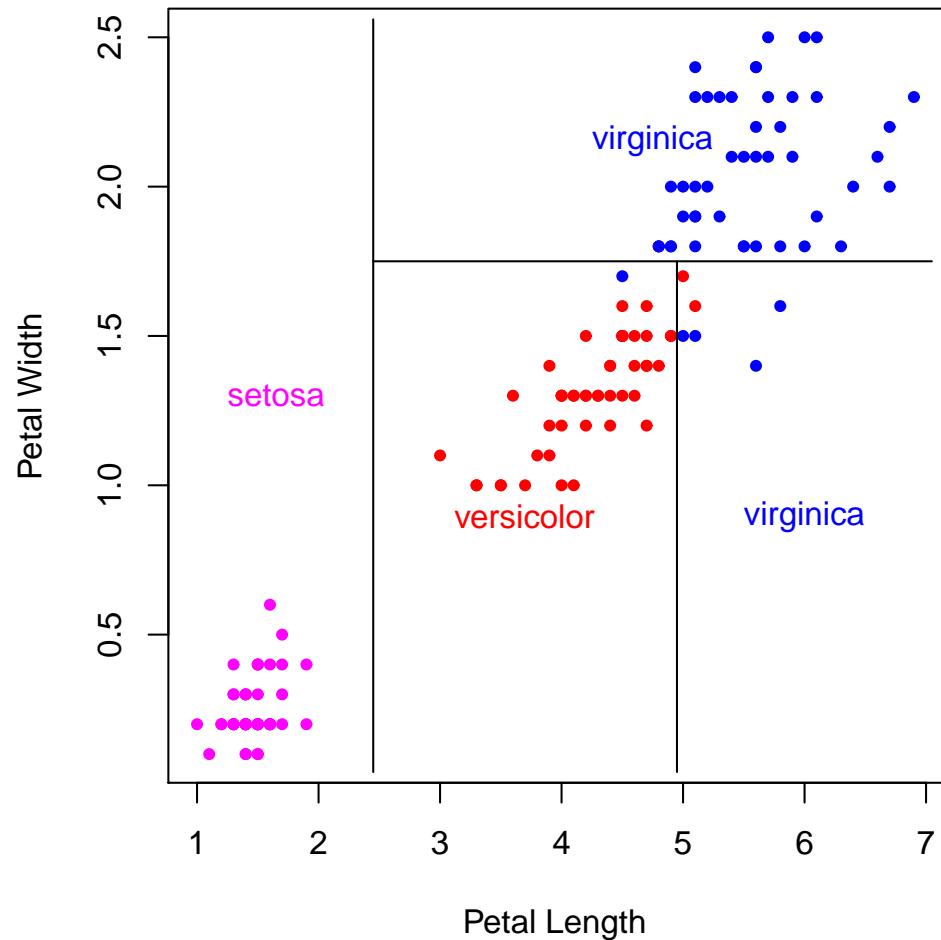
Пространство признаков разбивается на параллелепипеды со сторонами, параллельными осям координат (ящики).

В каждом ящике ответ аппроксимируется с помощью некоторой простой модели, обычно константой (как для задачи классификации, так и для задачи восстановления регрессии).

Используются только рекурсивные гильотинные разбиения.

Задача классификации цветов ириса (Fisher, 1936).

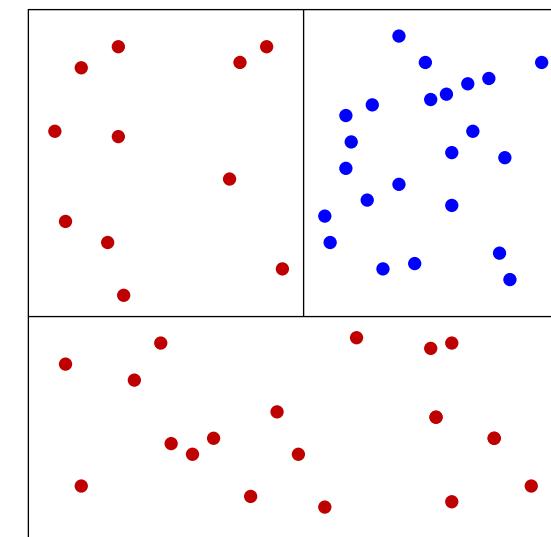
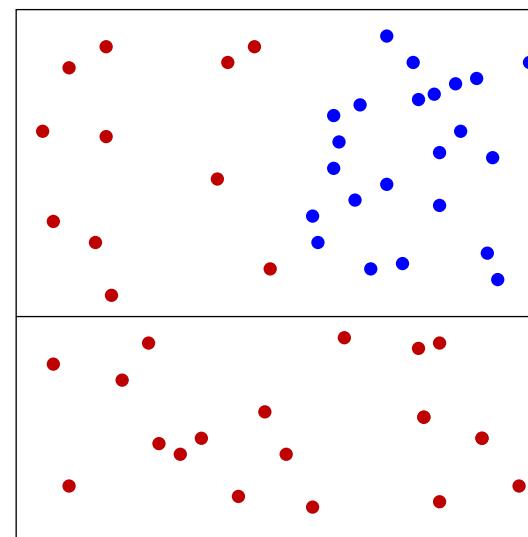
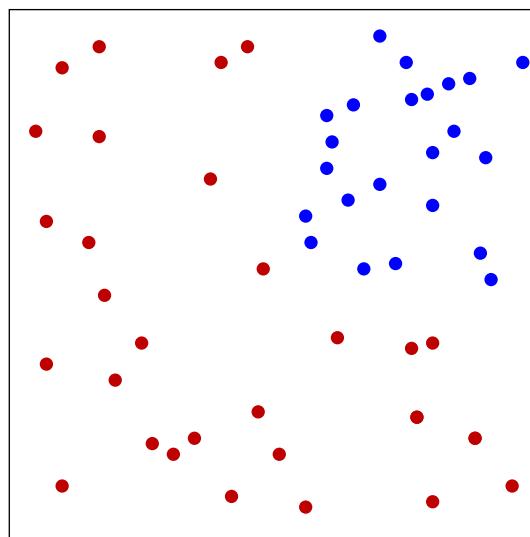
$x_1, x_2$  — длина и ширина чашелистика.



Каждому узлу дерева соответствует «ящик» в пространстве признаков.  
Этот ящик может разбиваться далее на следующих ярусах.

## 7.1. Популярные алгоритмы построения деревьев решений

- See5/C5.0 [Quinlan et., 1997] ← C4.5 [Quinlan, 1993] ← ID3 [Quinlan, 1979] ← CLS [Hunt & Marin & Stone & 1966]
- CART – Classification and Regression Trees [Breiman & Friedman & Olshen & Stone, 1984] ← CHAID [Kass, 1980] ← THAID [Morgan & Messenger 1973] ← AID [Morgan & Sonquist, 1963]
  - это *жадные рекурсивные* алгоритмы, на каждом шаге разбивающие очередной ящик, чтобы добиться максимального уменьшения взвешенной *неоднородности*:



## 7.2. Алгоритм CART

Разбиения (splits) имеют вид:

- $x_j \leq c$  для количественных признаков;
- $x_j \in L$ , где  $L \subset \{1, 2, \dots, M_j\}$  для качественных признаков.

Дерево строим рекурсивно.

Пусть на текущем шаге имеется разбиение пространства признаков на области  $R_1, R_2, \dots, R_M$ .

- Выбираем область  $R_m$ .
- Выбираем  $j$  и  $c$  (или  $L$ ) так, чтобы добиться максимального уменьшения взвешенной неоднородности (impurity) (загрязненности, примесности, хаоса)  $Q_m$  ( $m = 1, 2, \dots, M$ ) (т. е. максимального увеличения «прироста информации»).
- Строим разбиение (split) и повторяем действия.

Способы измерить «неоднородности»:

- Для задачи восстановления регрессии:

$$Q_m = \frac{1}{N_m} \sum_{x^{(i)} \in R_m} \left( y^{(i)} - f(x^{(i)}) \right)^2 = \frac{1}{N_m} \sum_{x^{(i)} \in R_m} \left( y^{(i)} - c_m \right)^2,$$

где  $N_m = \sum I(x^{(i)} \in R_m)$  — количество  $x^{(i)} \in R_m$ .

Взвешенная неоднородность:

$$\widehat{Q}_m = \frac{N_{m1}}{N_m} Q_{m1} + \frac{N_{m2}}{N_m} Q_{m2} = \frac{1}{N_m} \left( \sum_{x^{(i)} \in R_{m1}} \left( y^{(i)} - c_{m1} \right)^2 + \sum_{x^{(i)} \in R_{m2}} \left( y^{(i)} - c_{m2} \right)^2 \right) \rightarrow \min,$$

где  $R_m = R_{m1} \cup R_{m2}$ ,  $N_{m1} = \sum I(x^{(i)} \in R_{m1})$ ,  $N_{m2} = \sum I(x^{(i)} \in R_{m2})$ .

- Для задачи классификации:

- Ошибка классификации:

$$Q_m = \frac{1}{N_m} \sum_{x^{(i)} \in R_m} I\left(y^{(i)} \neq k(m)\right) = 1 - \max_k p_{km} = 1 - p_{k(m), m},$$

$p_{km}$  — доля объектов  $k$ -го класса в  $R_m$ ,  $k(m)$  — класс, преобладающий в  $R_m$ .

- Индекс К. Джини (вероятность, что два наугад взятых элемента из  $R_m$  принадлежат разным классам):

$$Q_m = \sum_{k \neq k'} p_{mk} p_{mk'} = \sum_{k=1}^K p_{mk} (1 - p_{mk}) = 1 - \sum_{k=1}^K p_{mk}^2.$$

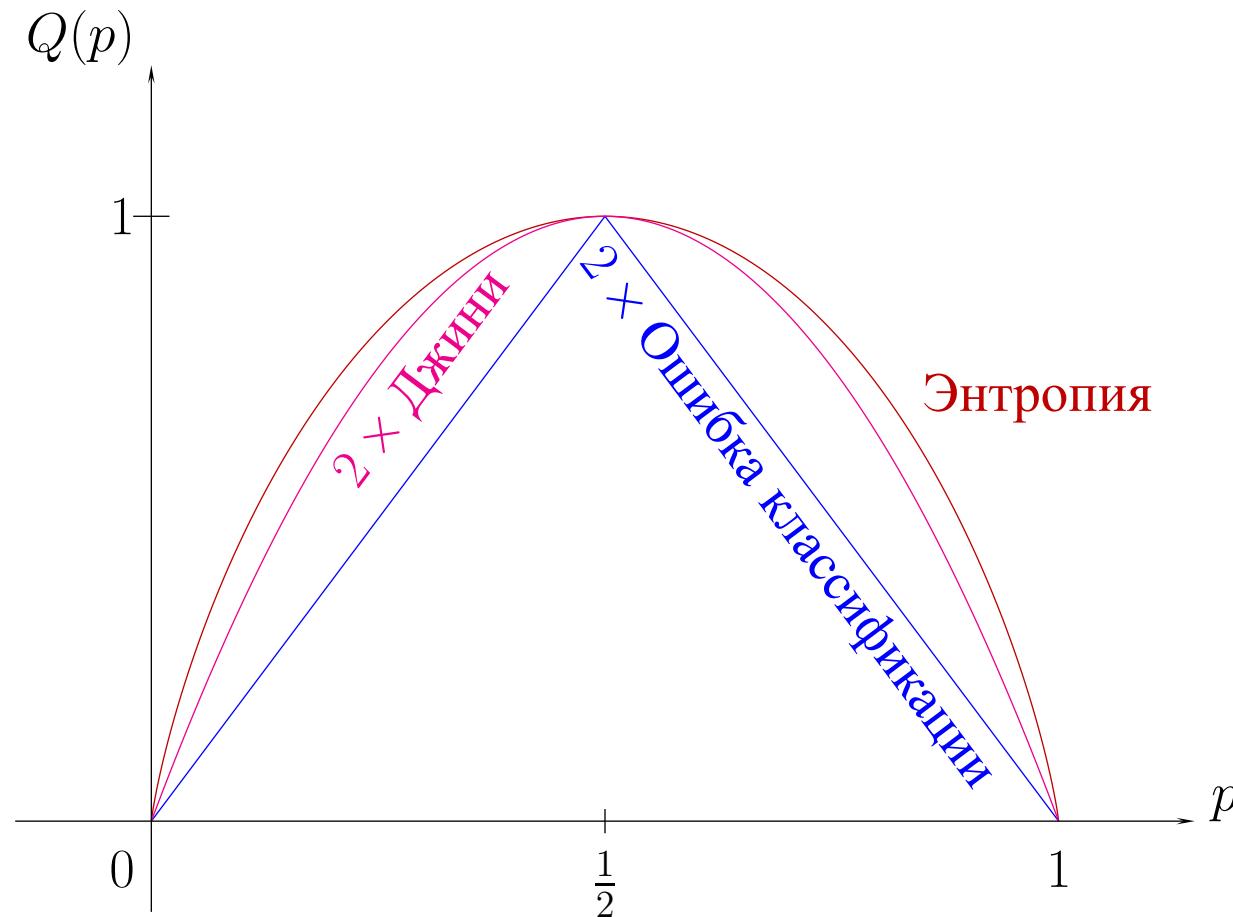
- Энтропия (количество информации):

$$Q_m = - \sum_{k=1}^K p_{mk} \log p_{mk}.$$

Если  $K = 2$ , то эти функции равны соответственно

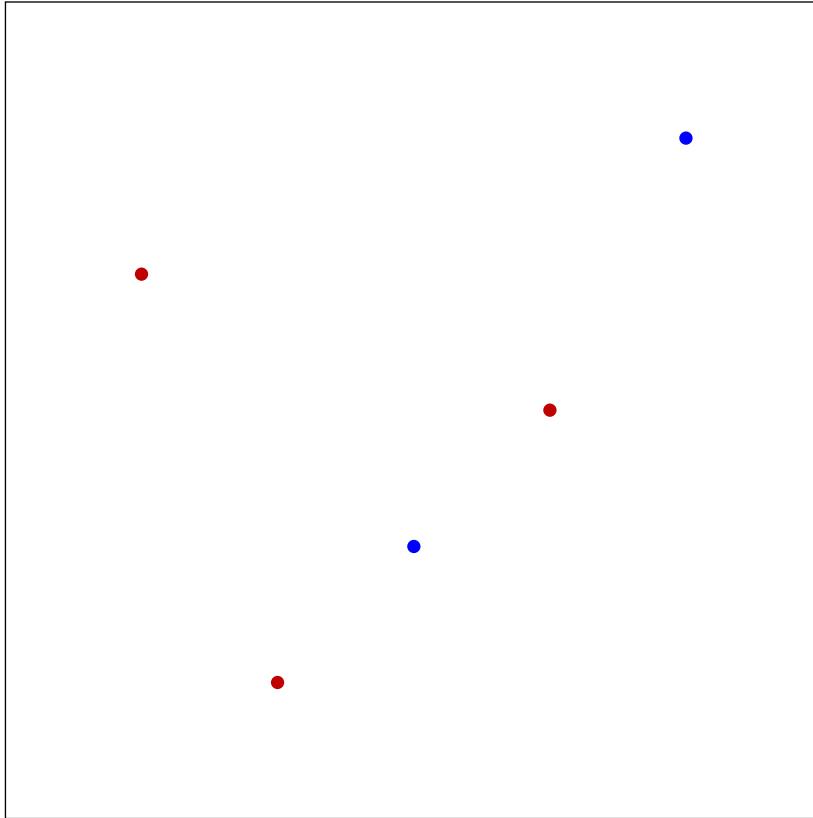
$$1 - \max \{p, 1 - p\}, \quad 2p(1 - p), \quad -p \log p - (1 - p) \log(1 - p),$$

где  $p = p_{1m}$  — доля объектов 1-го класса, попавших в ящик  $R_m$ .



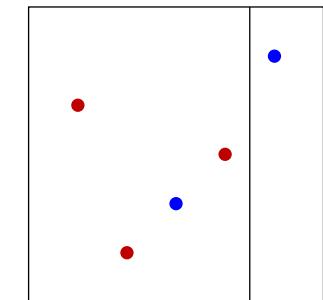
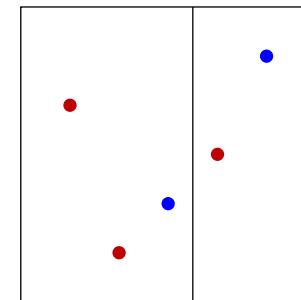
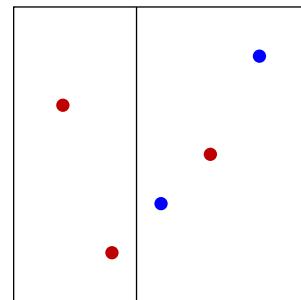
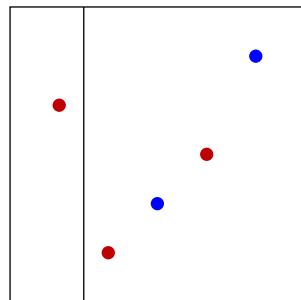
Функции похожи друг на друга. Индекс Джини и энтропия являются гладкими функциями и поэтому более податливы для численной оптимизации.

Каждая из трех приведенных функций равна нулю тогда и только тогда, когда в узле присутствуют объекты только одного класса.



$$Q_{\text{misclass}} = \frac{2}{5} = 0.4, \quad Q_{\text{Gini}} = 1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2 = \frac{12}{25} = 0.4800,$$

$$Q_{\text{entropy}} = -\frac{2}{5} \log \frac{2}{5} - \frac{3}{5} \log \frac{3}{5} = 0.9710.$$



$$N_m \hat{Q}_{\text{misclass}} = 1 \cdot 0 + 4 \cdot \frac{1}{2} = 2$$

$$N_m \hat{Q}_{\text{Gini}} = 1 \cdot 0 + 4 \cdot \frac{1}{2} = 2$$

$$N_m \hat{Q}_{\text{entropy}} = 1 \cdot 0 + 4 \cdot 1 = 4$$

$$2 \cdot 0 + 3 \cdot \frac{1}{3} = 1$$

$$2 \cdot 0 + 3 \cdot \frac{4}{9} = \frac{4}{3}$$

$$2 \cdot 0 + 3 \cdot 0.9183 = 2.7549$$

$$3 \cdot \frac{1}{3} + 2 \cdot \frac{1}{2} = 2$$

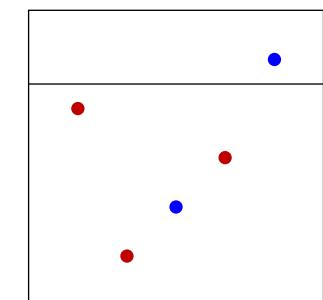
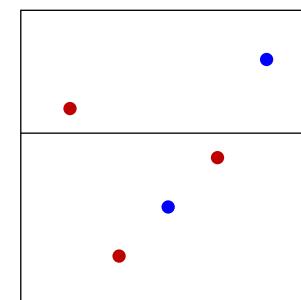
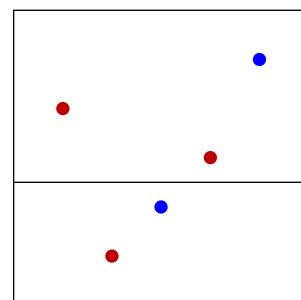
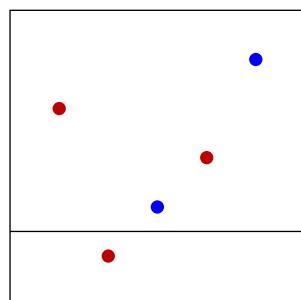
$$3 \cdot \frac{4}{9} + 2 \cdot \frac{1}{2} = \frac{7}{3}$$

$$3 \cdot 0.9183 + 2 \cdot 1 = 4.7549$$

$$4 \cdot \frac{1}{4} + 1 \cdot 0 = 1$$

$$4 \cdot \frac{3}{8} + 1 \cdot 0 = \frac{3}{2}$$

$$4 \cdot 0.8113 + 1 \cdot 0 = 3.2452$$



$$N_m \hat{Q}_{\text{misclass}} = 1 \cdot 0 + 4 \cdot \frac{1}{2} = 2$$

$$N_m \hat{Q}_{\text{Gini}} = 1 \cdot 0 + 4 \cdot \frac{1}{2} = 4$$

$$N_m \hat{Q}_{\text{entropy}} = 1 \cdot 0 + 4 \cdot 1 = 4$$

$$2 \cdot \frac{1}{2} + 3 \cdot \frac{1}{3} = 2$$

$$2 \cdot \frac{1}{2} + 3 \cdot \frac{4}{9} = \frac{7}{3}$$

$$2 \cdot 1 + 3 \cdot 0.9183 = 4.7549$$

$$3 \cdot \frac{1}{3} + 2 \cdot \frac{1}{2} = 2$$

$$3 \cdot \frac{4}{9} + 2 \cdot \frac{1}{2} = \frac{3}{2}$$

$$3 \cdot 0.9183 + 2 \cdot 1 = 4.7549$$

$$4 \cdot \frac{1}{4} + 1 \cdot 0 = 1$$

$$4 \cdot \frac{3}{8} + 1 \cdot 0 = \frac{3}{4}$$

$$4 \cdot 0.8113 + 1 \cdot 0 = 3.2452$$

**Замечание 7.1** Дадим еще две интерпретации индексу Джини:

- Вместо того, чтобы в листе  $t$  классифицировать объект по большинству голосов, мы можем относить объект к классу  $k$  с вероятностью  $p_{mk}$ .

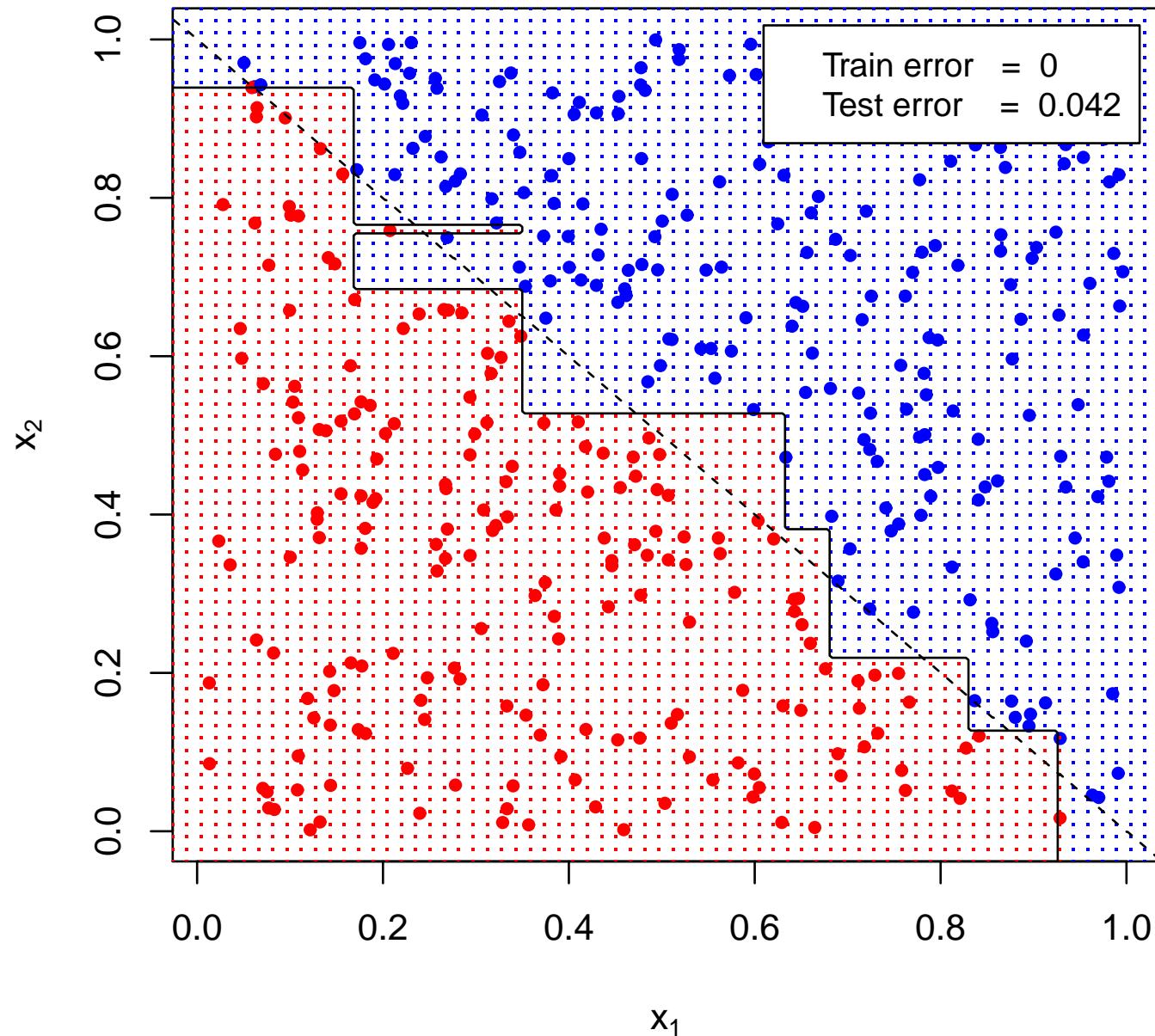
В этом случае средняя ошибка на обучающей выборке равна индексу Джини

$$\sum_{k \neq k'} p_{mk} p_{mk'}.$$

Разумеется, средняя ошибка на обучающей выборке является аппроксимацией средней ошибки на тестовой выборке (для объектов, попадающих в  $R_m$ )

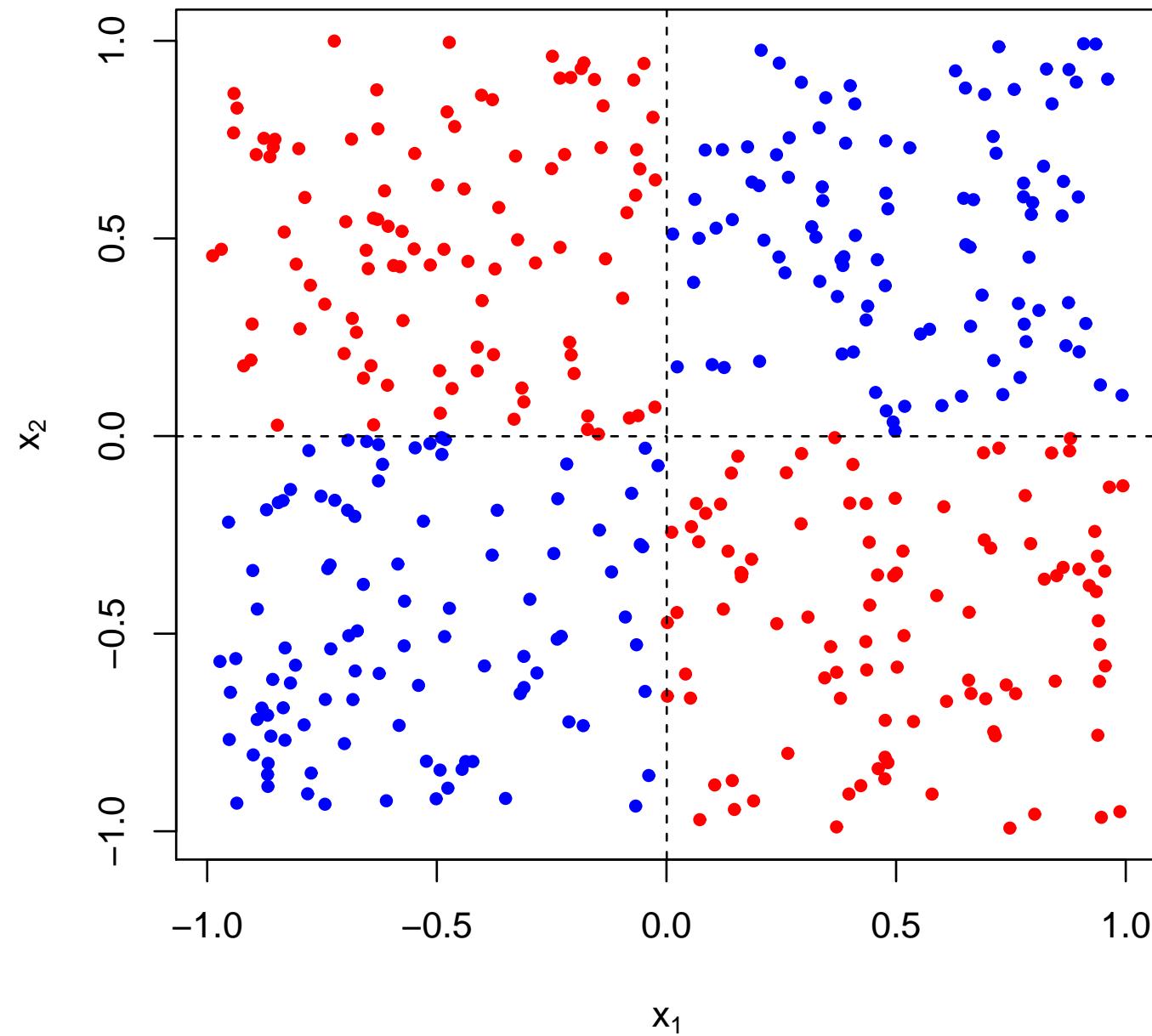
- Кодируем единицей объекты  $k$ -го класса и нулем — все остальные. Тогда выборочная дисперсия этой случайной величины в  $R_m$  равна  $p_{mk}(1 - p_{mk})$ . Сумма по всем классам  $k$  снова дает индекс Джини.

Высота = 6

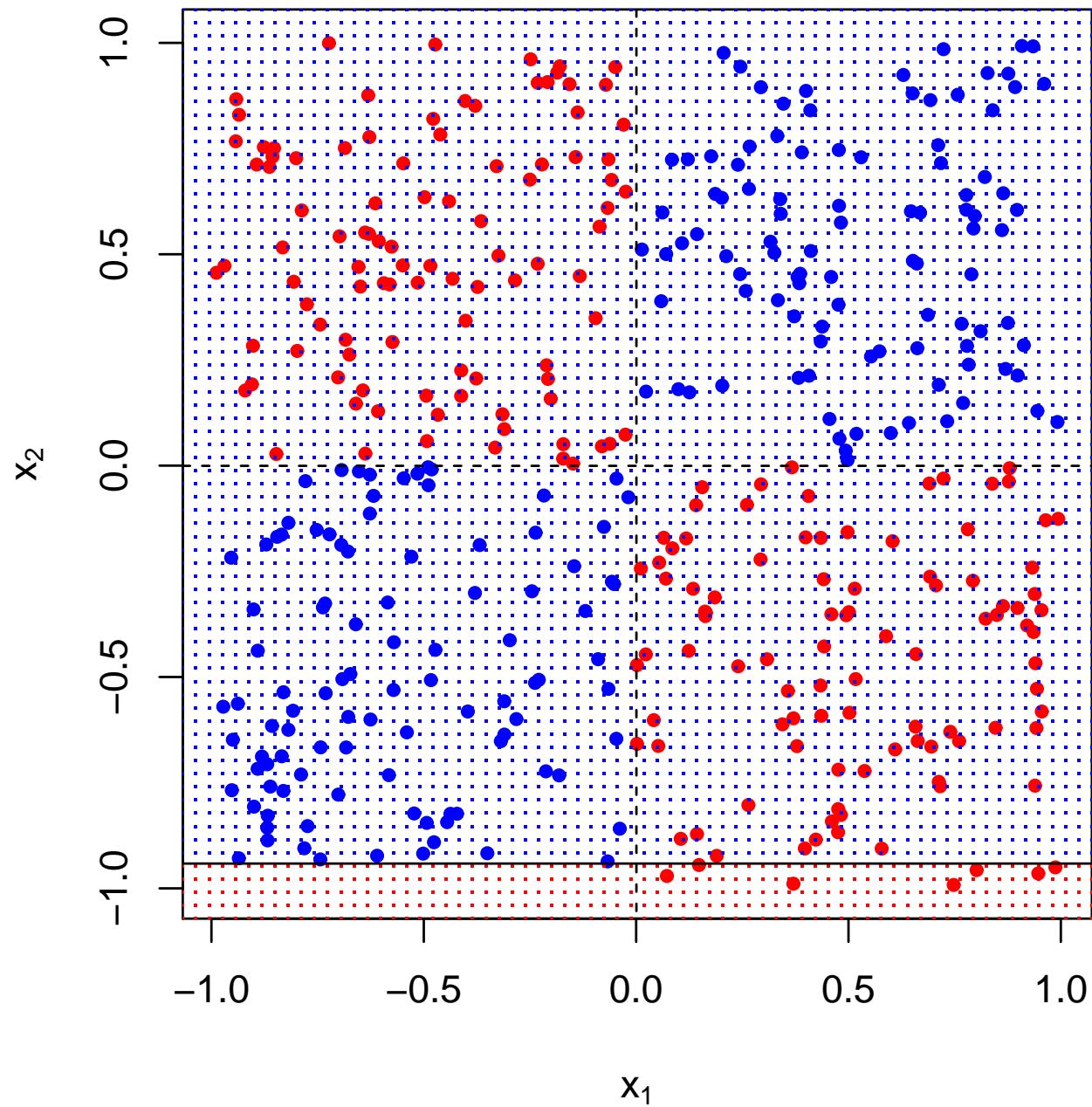


У алгоритм CART проблемы с некоторыми простыми распределениями.

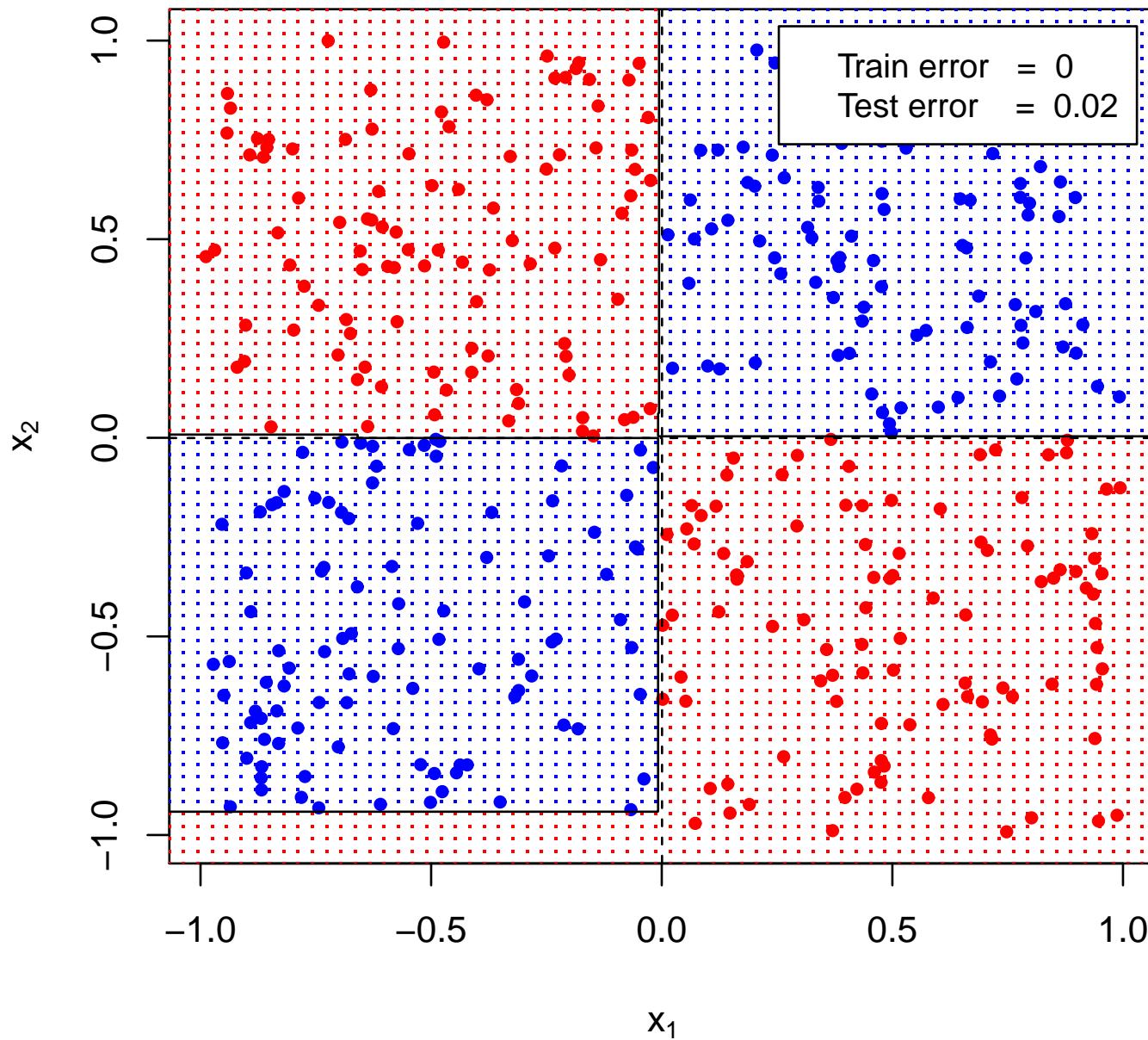
Например, с функцией xor, хотя для нее можно построить хорошее дерево решений.



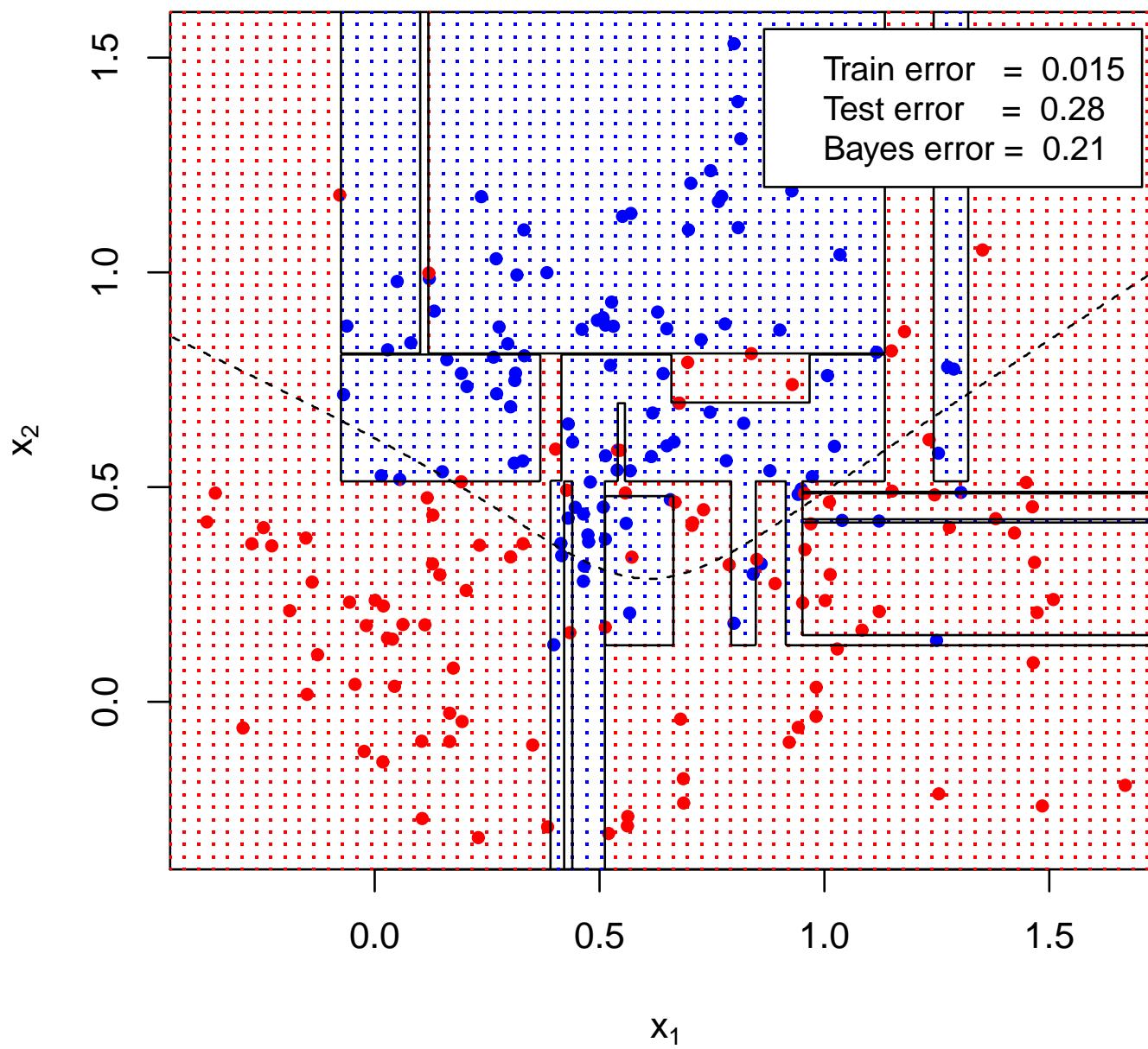
Высота = 2, 3, 4



Высота = 5



Высота = 10



— переобучение

## 7.3. Обрезка деревьев

Обрезка, или стрижка, деревьев (pruning) — боремся с переобучением.

$T' \subseteq T \Leftrightarrow$  дерево  $T'$  получается из  $T$  отсечениями (выбираем неконцевую вершину и удаляем оба ее поддерева)

$$Q_\alpha(T) = Q(T) + \alpha \cdot |T|$$

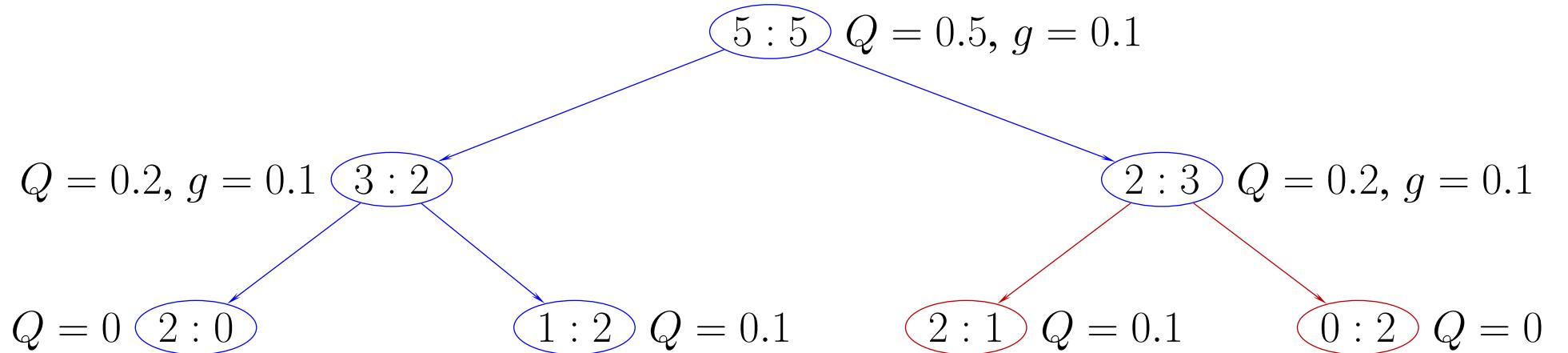
$|T|$  — число листьев в дереве  $T$

Минимизируем  $Q_\alpha(T')$  на множестве всех поддеревьев  $T'$ , получаемых из  $T$  отсечениями.

Для любого  $\alpha$  существует единственное тупиковое минимальное дерево  $T(\alpha) \subset T$ , т. е.

- 1) на  $T(\alpha)$  достигается минимум  $Q_\alpha(T)$ ;
- 2) из любого другого дерева, на котором достигается минимум  $Q_\alpha(T)$ , отсечениями можно получить  $T(\alpha)$ .

Найдем  $0 = \alpha_0 < \alpha_1 < \alpha_2 < \dots < \alpha_s$ ,  $T \supset T_1 \supset T_2 \supset \dots \supset T_s$ ,  
для которых  $T_i = T(\alpha)$ , где  $\alpha_{i-1} < \alpha \leq \alpha_i$ .



$T$  — все дерево,  $T'$  — синее поддерево.

$$Q_\alpha(T) = 0.2 + 4\alpha, \quad Q_\alpha(T') = 0.3 + 3\alpha.$$

$$Q_\alpha(T') < Q_\alpha(T) \Leftrightarrow \alpha > 0.1$$

В общем случае ( $t$  — вершина,  $T_t$  — выходящее из нее поддерево):

$$\begin{aligned} Q_\alpha(T') < Q_\alpha(T) &\Leftrightarrow Q(T') + \alpha|T'| < Q(T) + \alpha|T'| \Leftrightarrow \\ &\Leftrightarrow Q(t) + \alpha < Q(T_t) + |T_t| \cdot \alpha \Leftrightarrow \alpha > g_T(t) \equiv \frac{Q(t) - Q(T_t)}{|T_t| - 1}, \end{aligned}$$

так как

$$|T| = |T'| + |T_t| - 1, \quad Q(T) = Q(T') + Q(T_t) - Q(t).$$

Процедура построения последовательности  $T \supset T_1 \supset T_2 \supset \dots \supset T_s$

**begin**

$T_0 \leftarrow T$

$\alpha_0 \leftarrow 0$

$k = 0$

**while** число узлов в  $T_k$  больше 1

Для каждого нетерминального узла  $t$  дерева  $T_k$  вычислить  $g_{T_k}(t)$

$\alpha_k \leftarrow \min_t g_{T_k}(t)$  (минимум берется по всем нетерминальным узлам  $t$  дерева  $T_k$ )

Обойти сверху вниз все узлы  $t'$  дерева  $T_k$  и обрезать те, в которых  $g_{T_k}(t) = \alpha_{k+1}$

Построенное дерево обозначить  $T_{k+1}$

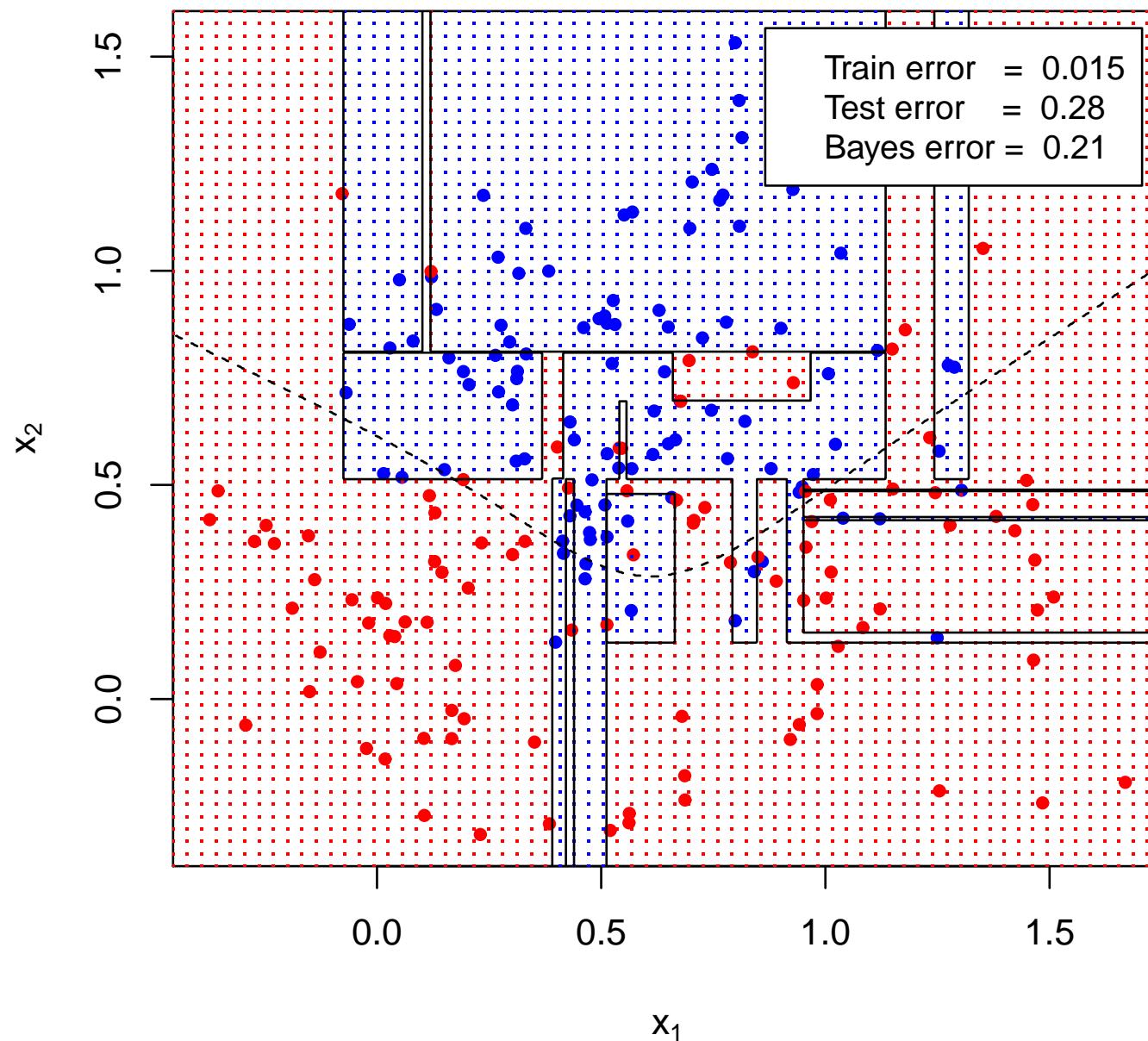
$k \leftarrow k + 1$

**end**

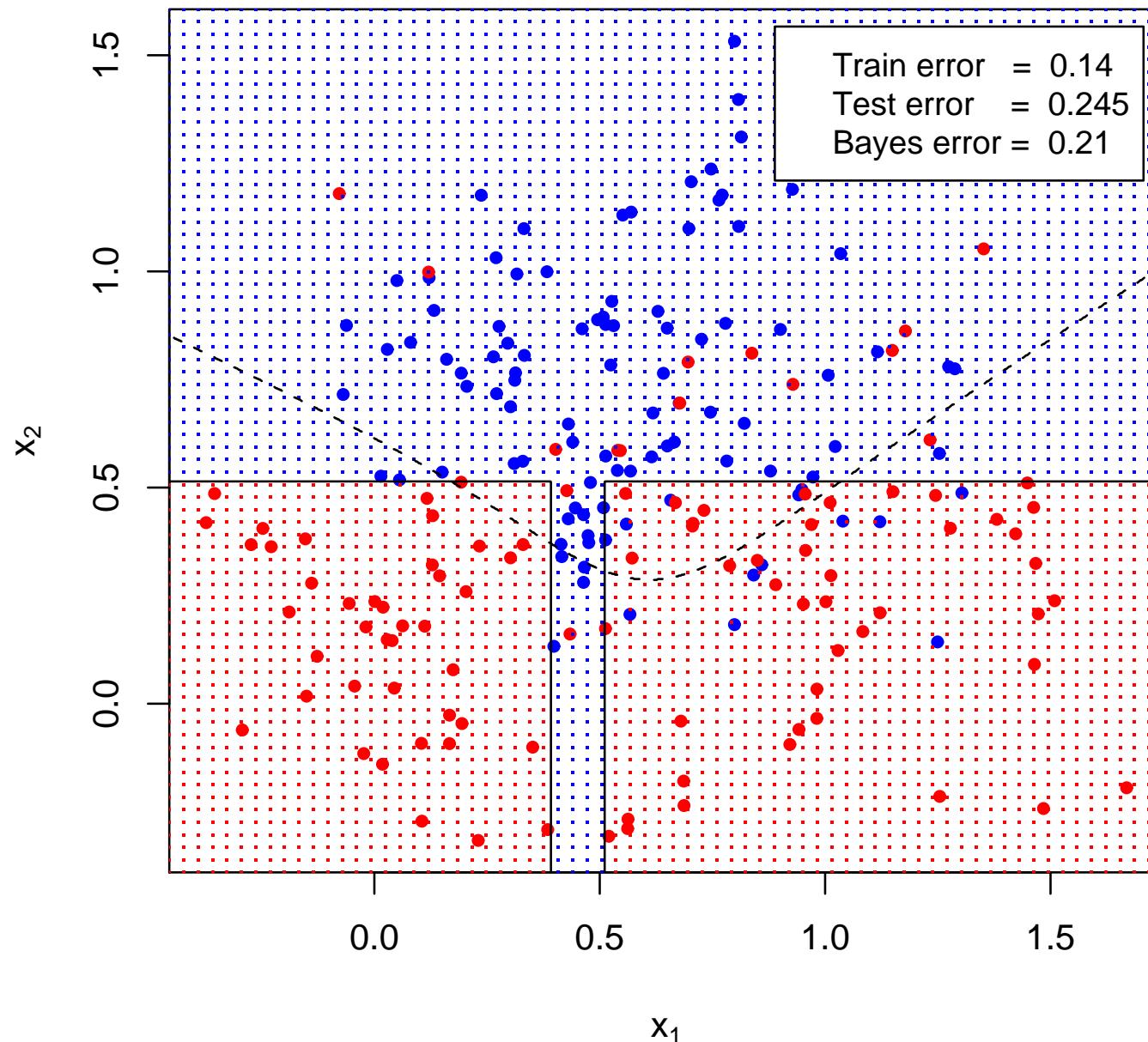
**end**

Среди всех  $k$  выбираем такое, для которого  $T(\alpha_k)$  дает наименьшую CV-ошибку.

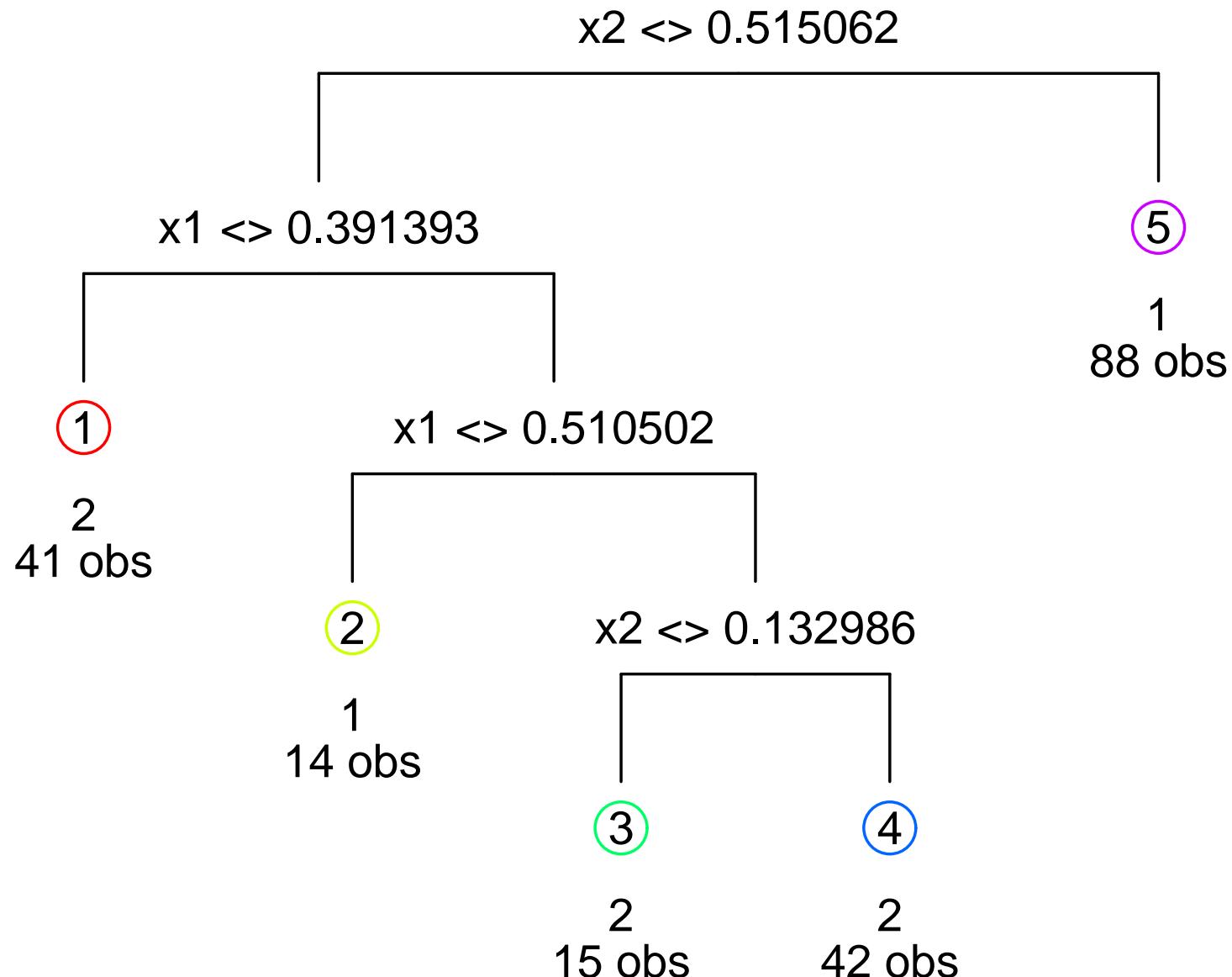
## Дерево решений глубины 10 — переобучение



## Оптимальное дерево после проведения отсечений — 5 листьев



Оптимальное дерево после проведения отсечений — 5 листьев



## Пример fg1

По стеклянным осколкам требуется определить их происхождение (B. German).

$N = 214$ ,  $d = 9$ ,  $K = 6$

*Входные признаки:*

RI — показатель преломления

плюс 8 значений процентного содержания оксидов/диоксидов следующих элементов:

Na, Mg, Al, Si, K, Ca, Ba, Fe

*Классы:*

WinF — оконное термополированное стекло (флоат-стекло) (70)

WinNF — оконное нетермополированное стекло (76)

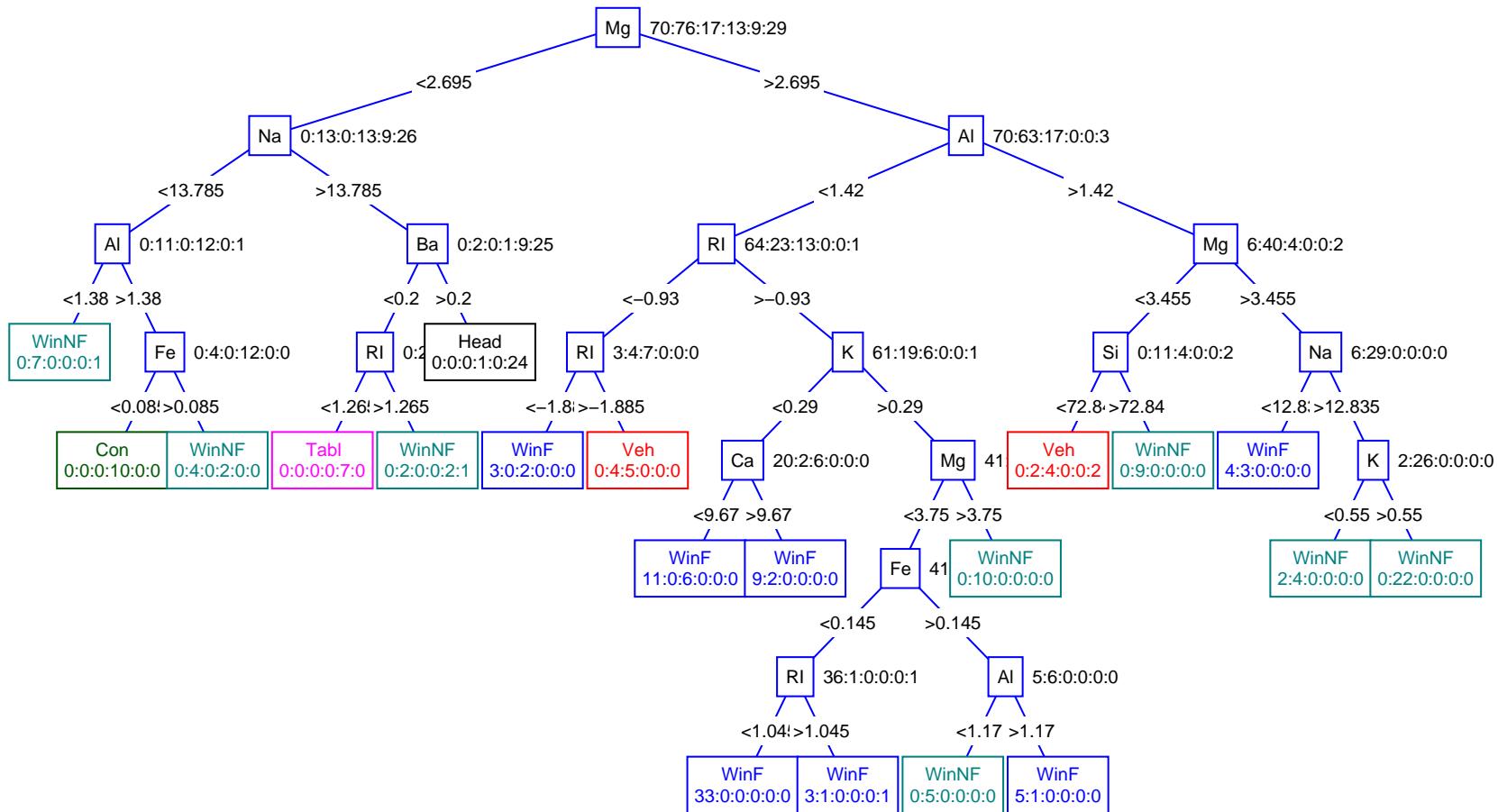
Veh — автомобильные окна (17)

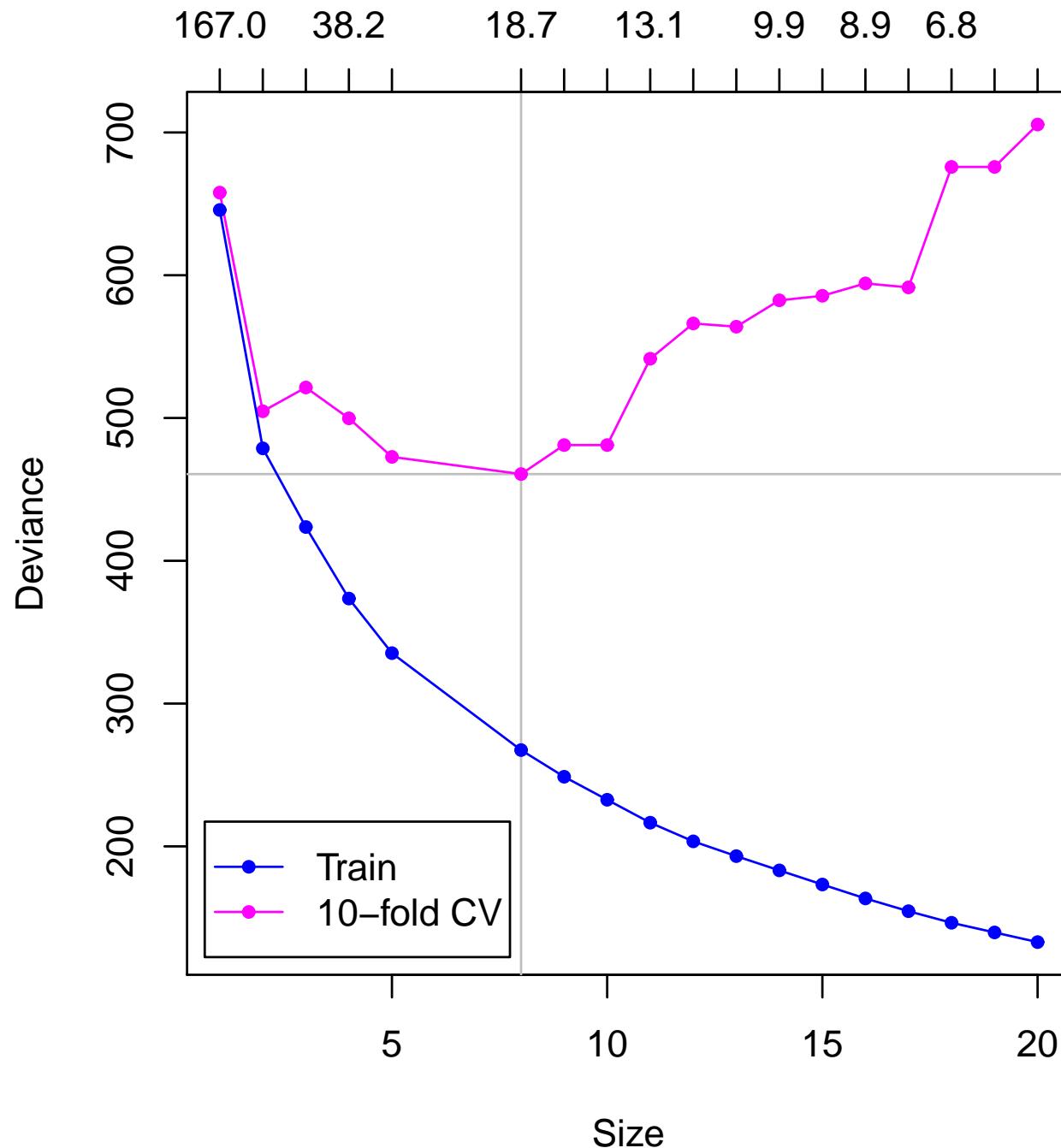
Con — сосуд (13)

Tabl — посуда (9)

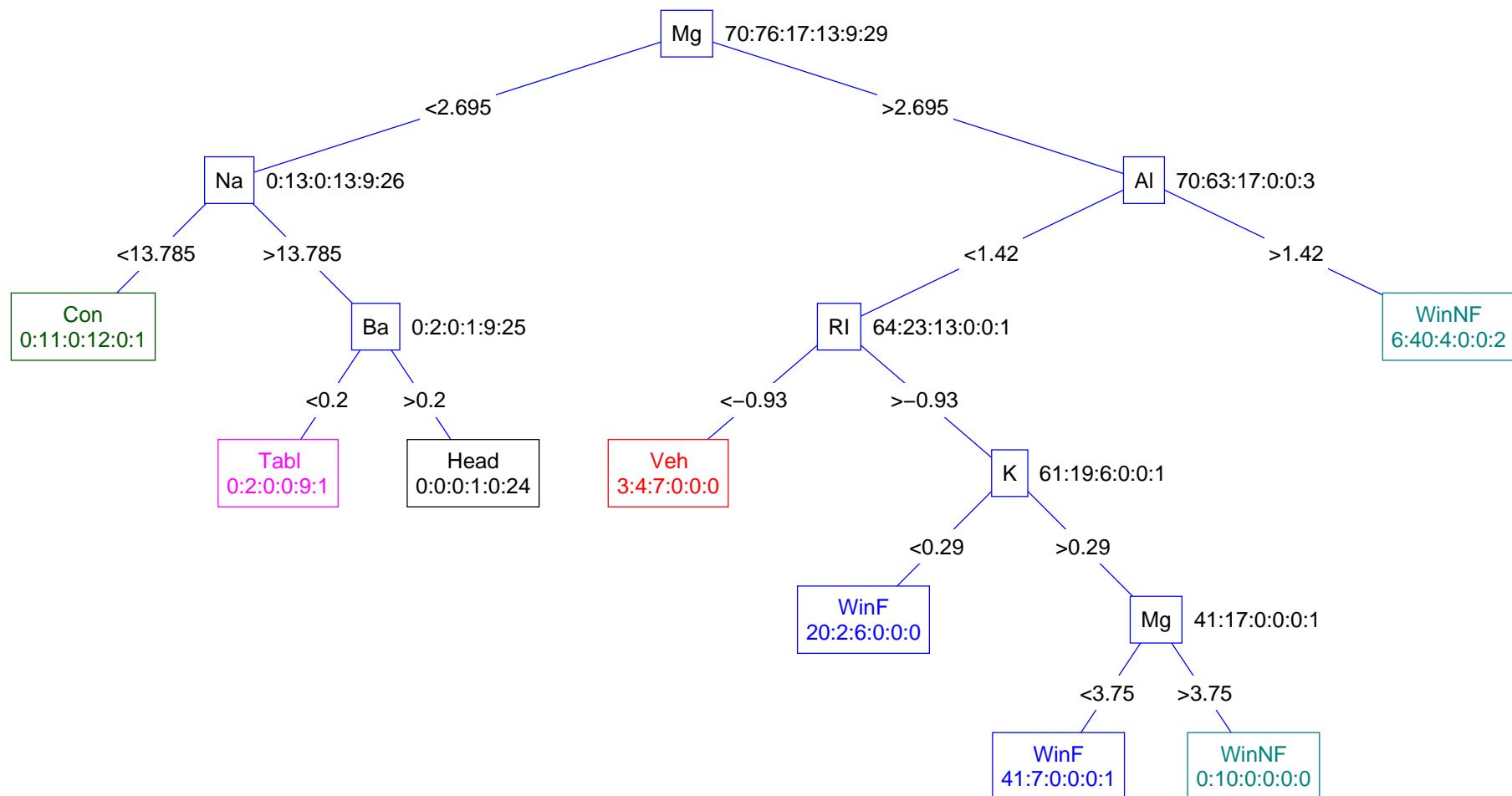
Head — автомобильные фары (29)

Мера неоднородности — энтропия.

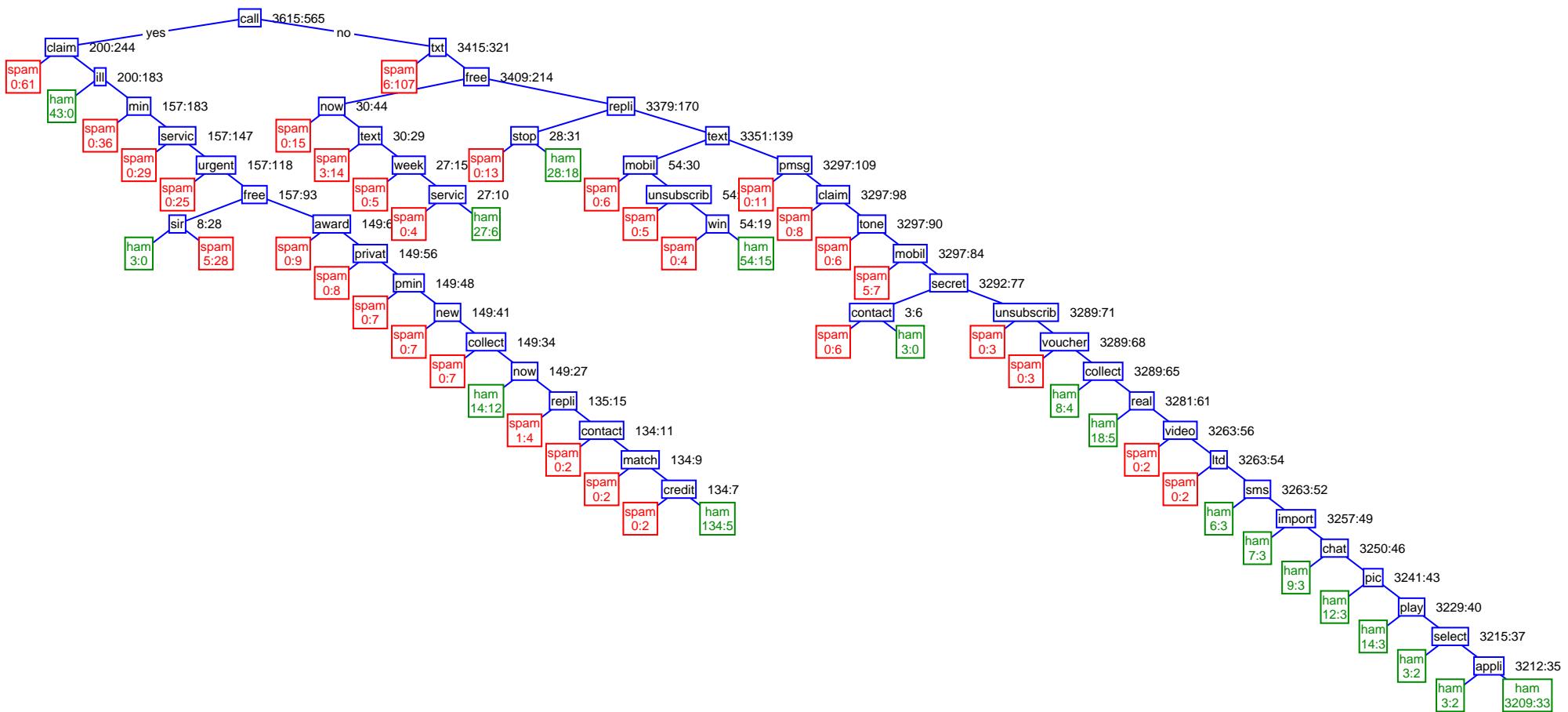


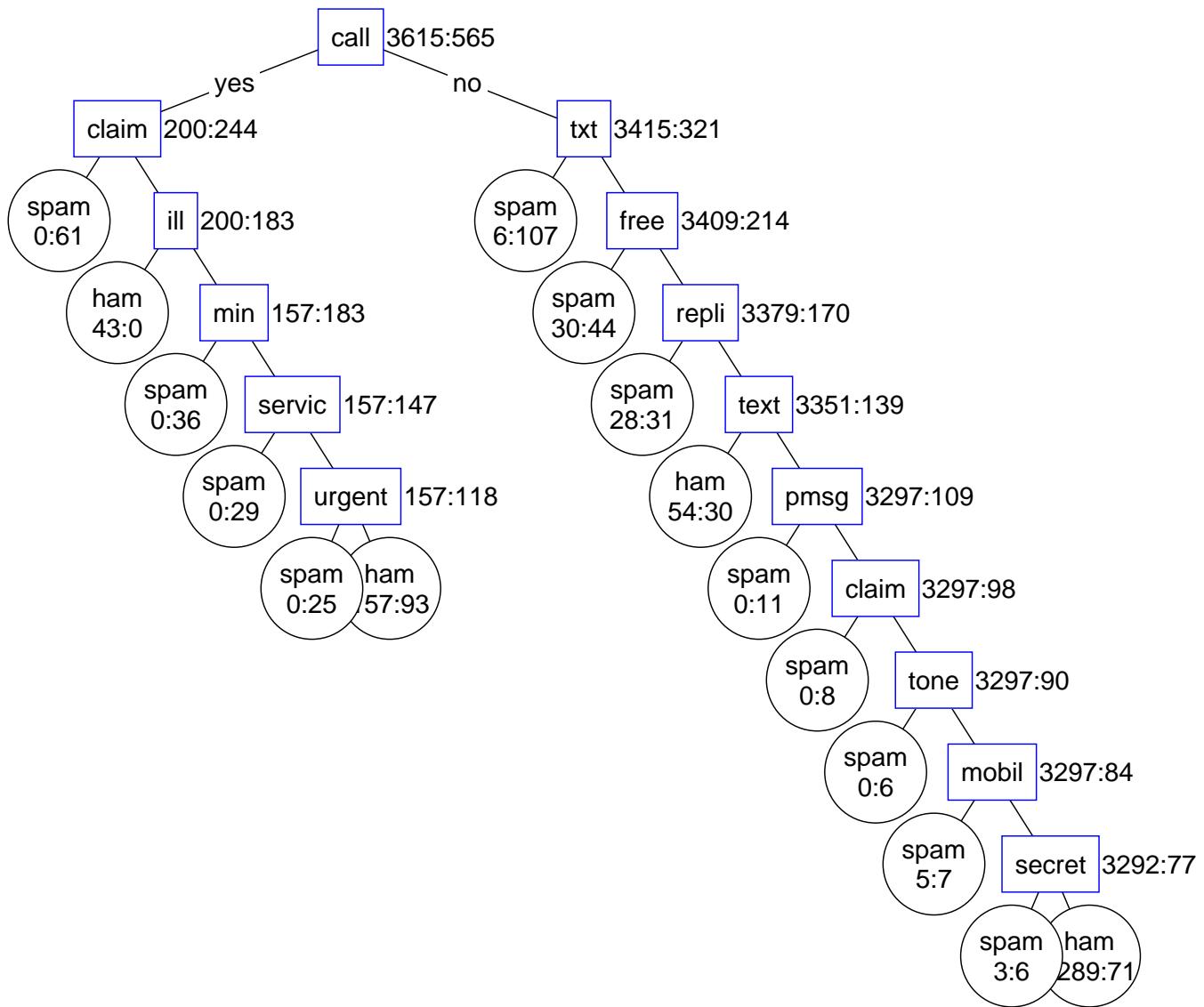


## «Оптимальное» дерево — 8 листьев

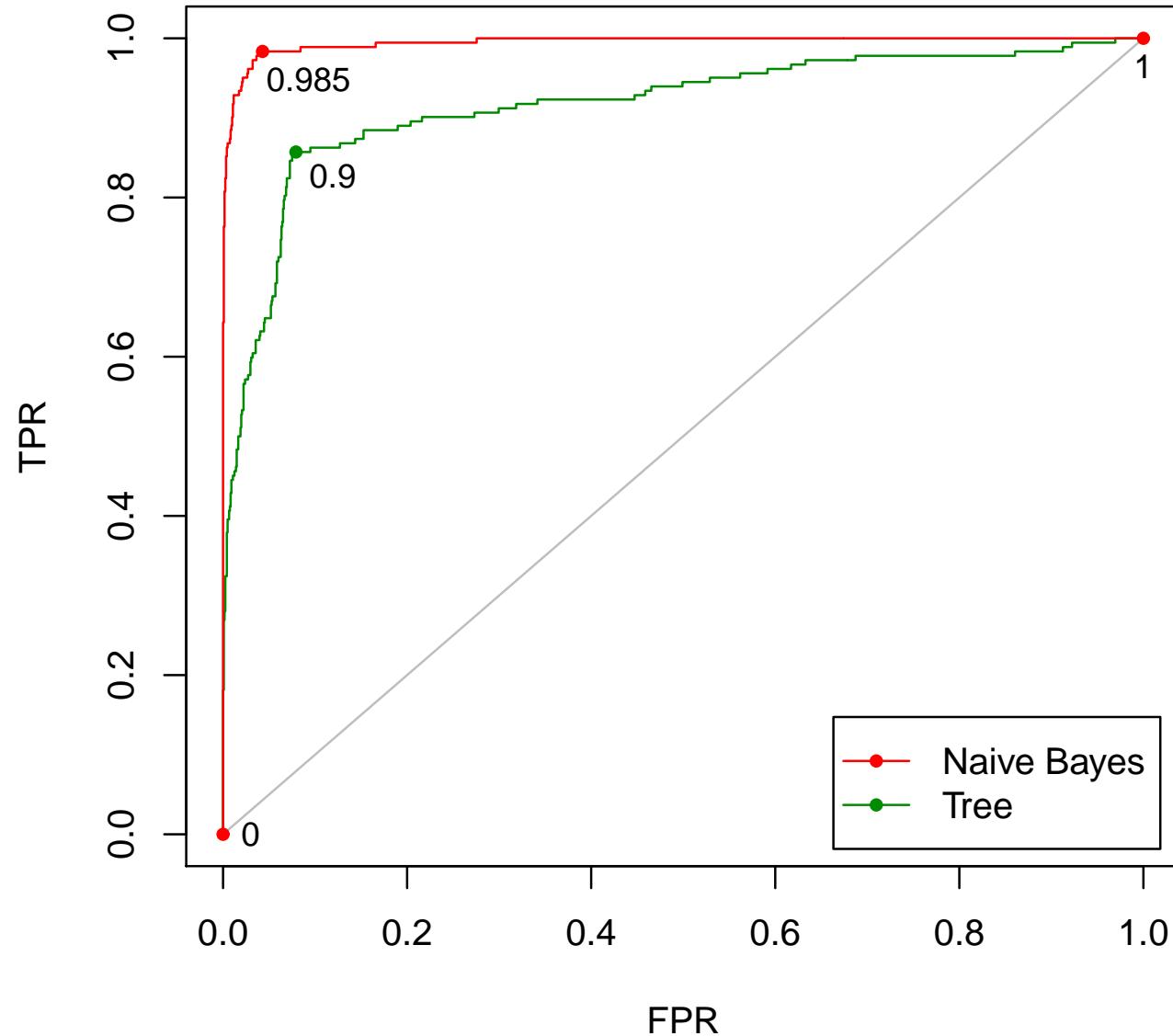


# SMS





Дерево решений ROC-кривая. tree.test.FPR[tree.threshold.seq == 0.9] = 0.07920792 1 - tree.test.TPR[tree.threshold.seq == 0.9] = 0.1428571 tree.test.Err[tree.threshold.seq == 0.9] = 0.08751793 AUC = 0.916712



## 7.4. Некоторые отличия C4.5 от CART

- C4.5 для каждого значения номинального признака строит свою ветвь (т. е. деревья *не* бинарные).
- При обрезке используется не перекрестный контроль, а пессимистическая верхняя оценка для биномиального распределения.

## 7.5. Обрезка в C4.5

Пусть в ящике  $R_m$  попало  $n$  объектов из обучающей выборки, из них  $n'$  — не принадлежат большинству. Таким образом, вероятность ошибки в заданном ящике  $\approx n'/n$ . Но насколько эта оценка точна? Не будет ли она слишком завышенной («пессимистичной»)?

Пусть  $p$  — настоящая вероятность ошибки, тогда

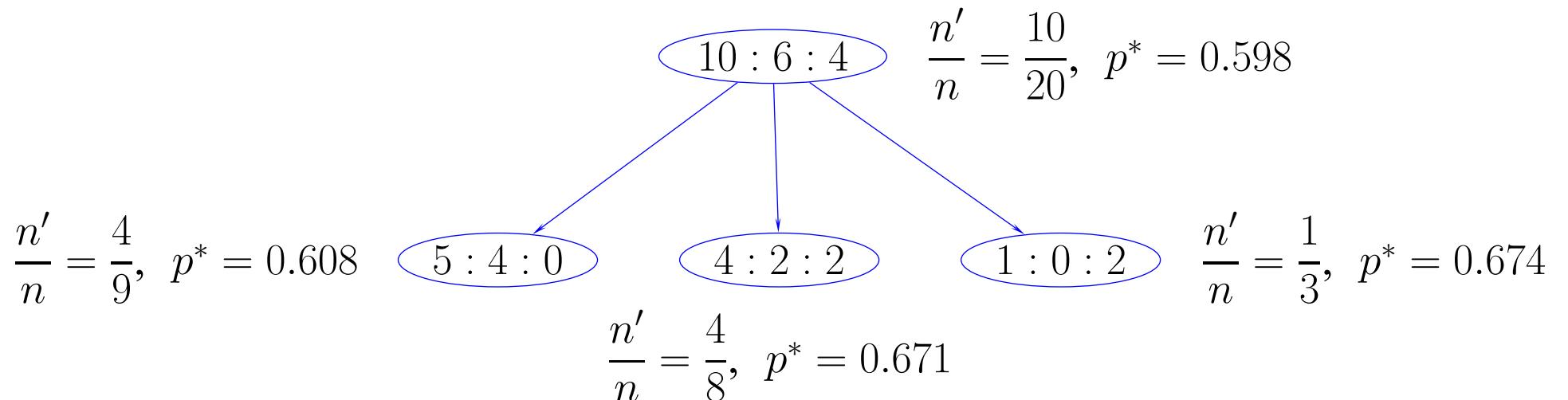
$$\Pr \{ \leq n' \text{ объектов не принадлежат большинству} \} = \sum_{i=0}^{n'} \binom{n}{i} p^i (1-p)^{n-i}. \quad (*)$$

В качестве верхней оценки  $p^*$  для вероятности ошибки берем такое значение  $p$ , для которого правая часть равенства  $(*)$  равна выбранному уровню значимости  $\alpha$ . Рекомендуется  $\alpha = 0.25$ .

Т. е. интервал  $[0, p^*]$  является доверительным интервалом при оценивании вероятности  $p$  с уровнем доверия  $1 - \alpha$ . Иными словами,  $p^*$  — верхняя оценка значения вероятности  $p$  при уровне доверия  $1 - \alpha$ .

За один проход от листьев к корню удаляются те узлы, взвешенная сумма оценок которых больше оценки для их родителя.

— «Когда не видно разницы, зачем платить больше?»



$$\frac{9}{20} \cdot 0.608 + \frac{8}{20} \cdot 0.671 + \frac{3}{20} \cdot 0.674 > 0.598 \quad \text{— все листья отсекаем.}$$

Для вычисления  $p^*$  используются аппроксимации (обратной) функции биномиального распределения. Например,

$$p^* = \frac{n' + \frac{1}{2} + \frac{u^2}{2} + u \sqrt{n' + \frac{1}{2} - \frac{1}{n} \left( n' + \frac{1}{2} \right)^2 + \frac{u^2}{4}}}{n + u^2}.$$

[Blyth C.R. Approximate binomial confidence limits // JASA. V. 81, 1986. P. 843–855]

Здесь  $u = u_{1-\alpha}$  — квантиль нормального распределения.

Если, например,  $\alpha = 0.25$ , то  $u = u_{0.75} = 0.674$ .

## 7.6. Достоинства и недостатки деревьев решений

*Достоинства:*

- Поддерживают работу с входными переменными разных (смешанных) типов
- Возможность обрабатывать данные с пропущенными значениями
- Устойчивы к выбросам
- Нечувствительность к монотонным преобразованиям входных переменных
- Поддерживают работу с большими выборками
- Возможность интерпретации построенного решающего правила

*Недостатки:*

- *Основной недостаток* — плохая предсказательная (обобщающая) способность.

## *Глава 8*

# **Ансамбли решающих правил**

*Ансамбль, или комитет, решающих правил (функций).*

Рассмотрим задачу классификации на  $K$  классов.

$$\mathcal{Y} = \{1, 2, \dots, K\}.$$

Пусть имеется  $M$  классификаторов («экспертов»)  $f_1, f_2, \dots, f_M$

$$f_m : \mathcal{X} \rightarrow \mathcal{Y}, \quad f_m \in \mathcal{F}, \quad (m = 1, 2, \dots, M)$$

Построим новый классификатор:

простое голосование:

$$f(x) = \operatorname{argmax}_{k=1, \dots, K} \sum_{m=1}^M I(f_m(x) = k),$$

взвешенное (выпуклая комбинация классификаторов, или «смесь экспертов»):

$$f(x) = \operatorname{argmax}_{k=1, \dots, K} \sum_{m=1}^M \alpha_m \cdot I(f_m(x) = k), \quad \alpha_m \geq 0, \quad \sum_{m=1}^M \alpha_m = 1,$$

или

$$f(x) = \operatorname{argmax}_{k=1, \dots, K} \sum_{m=1}^M \alpha_m(x) \cdot I(f_m(x) = k), \quad \alpha_m(x) \geq 0, \quad \sum_{m=1}^M \alpha_m(x) = 1.$$

В задаче восстановления регрессии

простое голосование:

$$f(x) = \frac{1}{M} \sum_{m=1}^M f_m(x),$$

взвешенное голосование («смесь экспертов»):

$$f(x) = \sum_{m=1}^M \alpha_m(x) \cdot f_m(x), \quad \alpha_m(x) \geq 0, \quad \sum_{m=1}^M \alpha_m(x) = 1.$$

Пример:  $K = 2$ ,  $M = 3$ .

Решение принимается с использованием простого голосования.

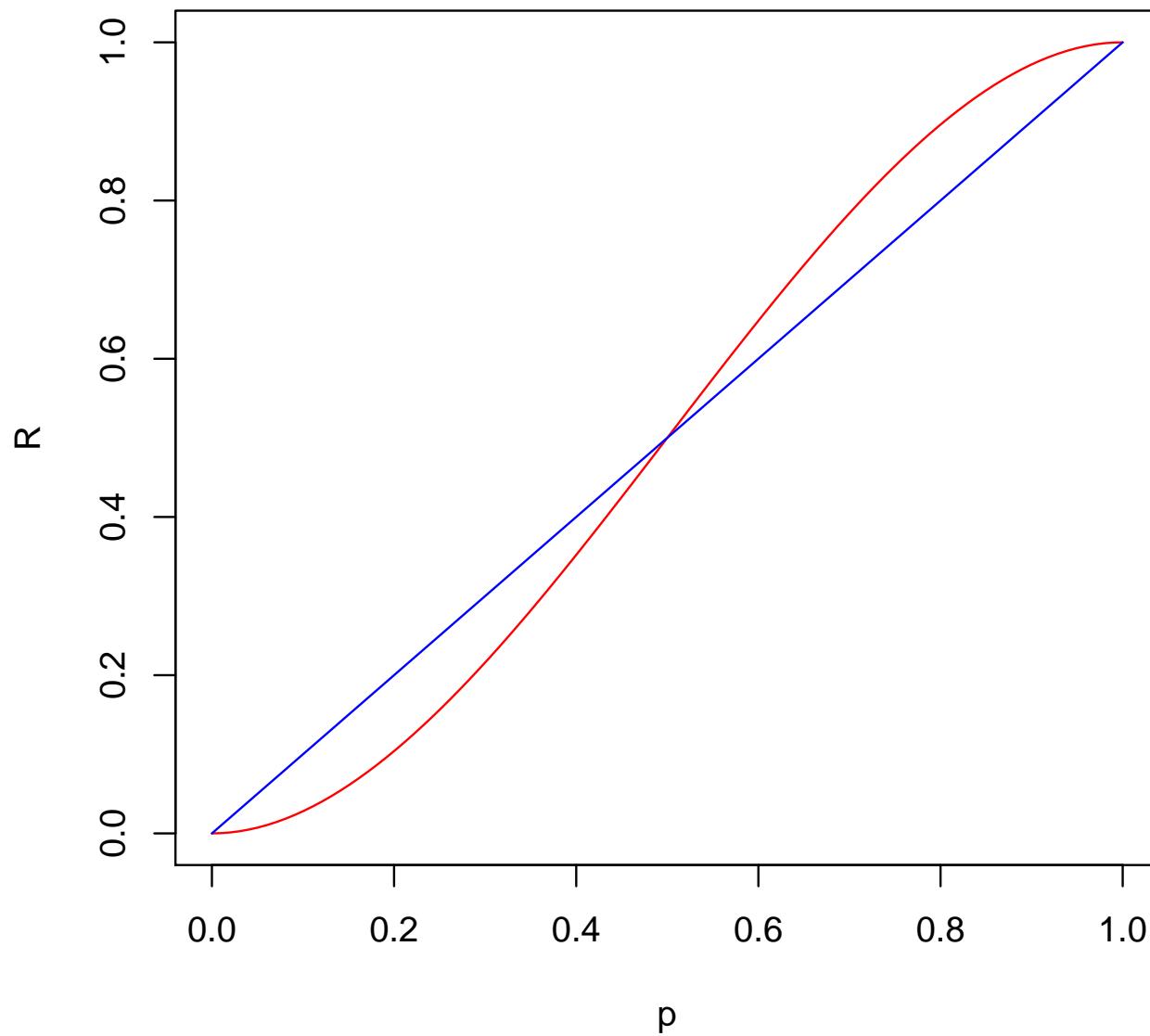
*Пусть классификаторы независимы* (на практике недостижимое требование!).

$p$  — вероятность ошибки каждого отдельного классификатора.

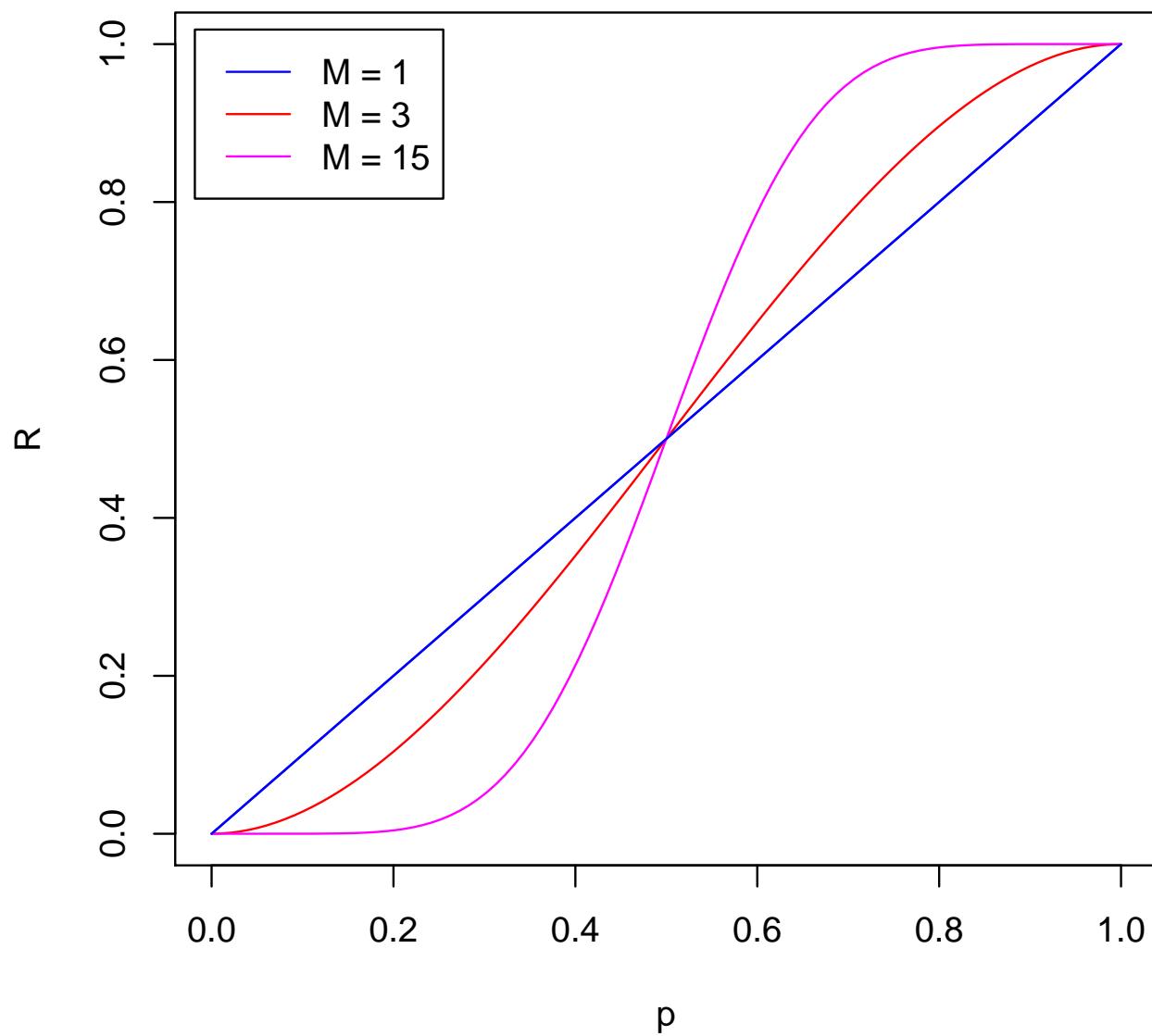
Тогда вероятность ошибки общего решения (ожидаемый риск) равен

$$R = p^3 + 3p^2(1 - p) = 3p^2 - 2p^3.$$

$$R = 3p^2 - 2p^3$$



$$M = 1, 3, 15$$



$K = 2$

- $p > \frac{1}{2}$  — плохой классификатор,
- $p = \frac{1}{2}$  — 50/50,
- $p = \frac{1}{2} - \varepsilon$  — хороший, но слабый классификатор ( $\varepsilon$  мало),
- $p = \varepsilon$  — сильный классификатор.

Можно ли научиться комбинировать слабые классификаторы, чтобы получить сильный [Kearns, Valiant, 1988]?

Два известных подхода:

- *Баггинг* и т. п.: пытаемся снизить зависимость экспертов друг от друга.
- *Бустинг* и т. п.: эксперты учатся на ошибках других.

## 8.1. Баггинг

*Bagging* – от bootstrap aggregation [Breiman, 1994]

Классификатор  $f_m$  обучается на bootstrap-выборке (повторной выборке) ( $m = 1, 2, \dots, M$ ).

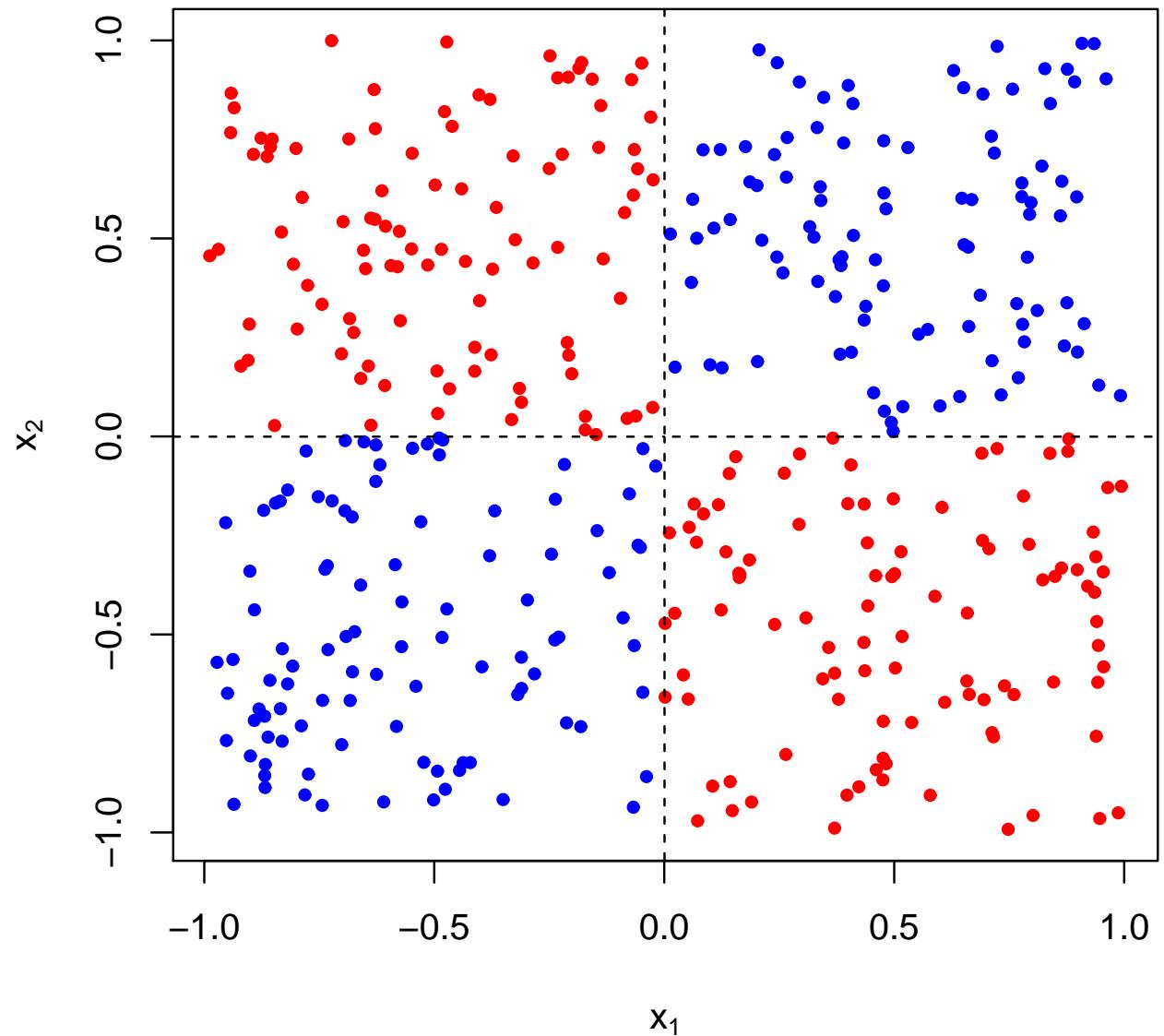
Финальный классификатор  $f$  – функция голосования:

для задачи восстановления регрессии:

$$f(x) = \frac{1}{M} \sum_{m=1}^M f_m(x),$$

для задачи классификации:

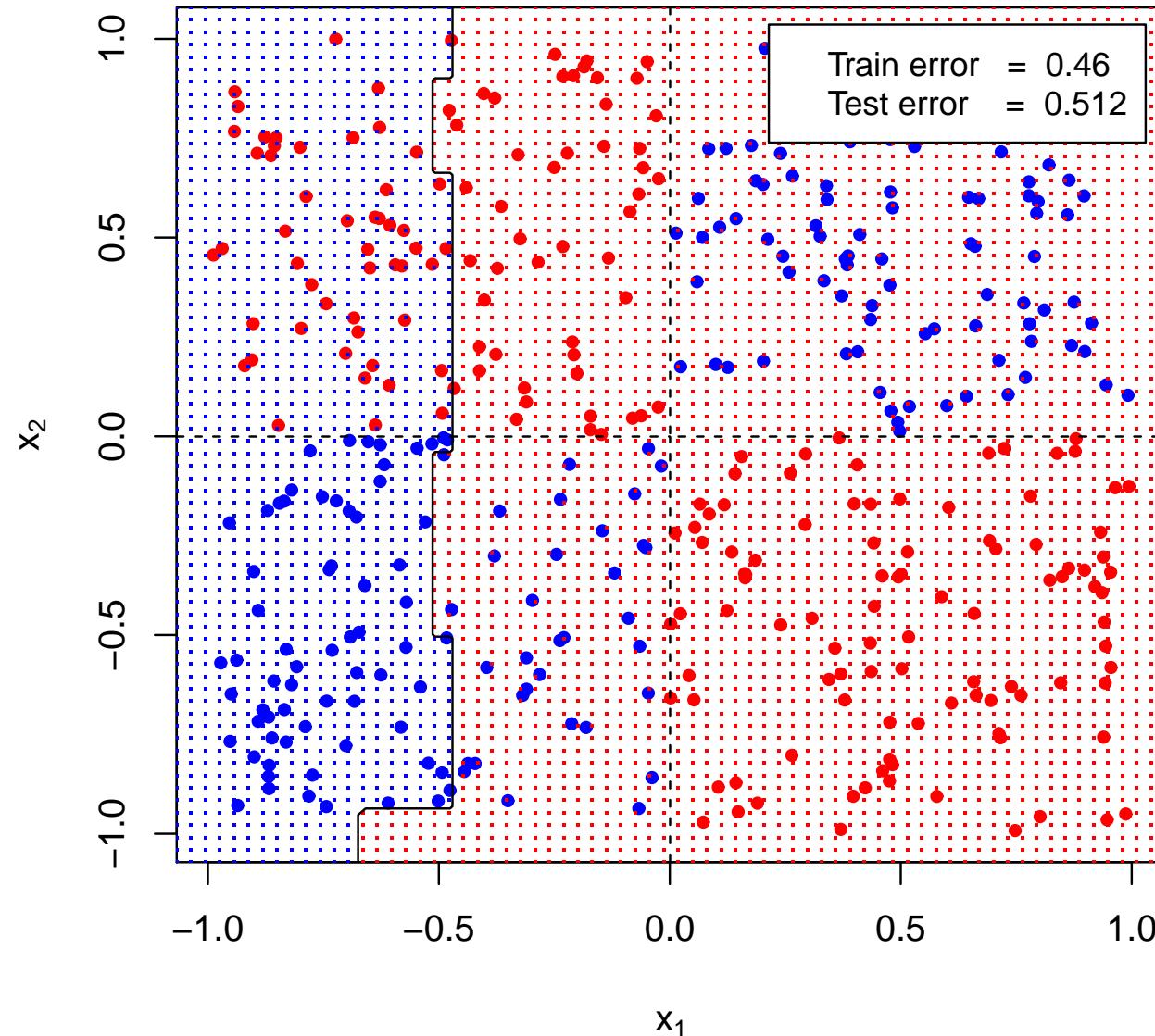
$$f(x) = \operatorname{argmax}_{k=1, \dots, K} \sum_{m=1}^M I(f_m(x) = k).$$



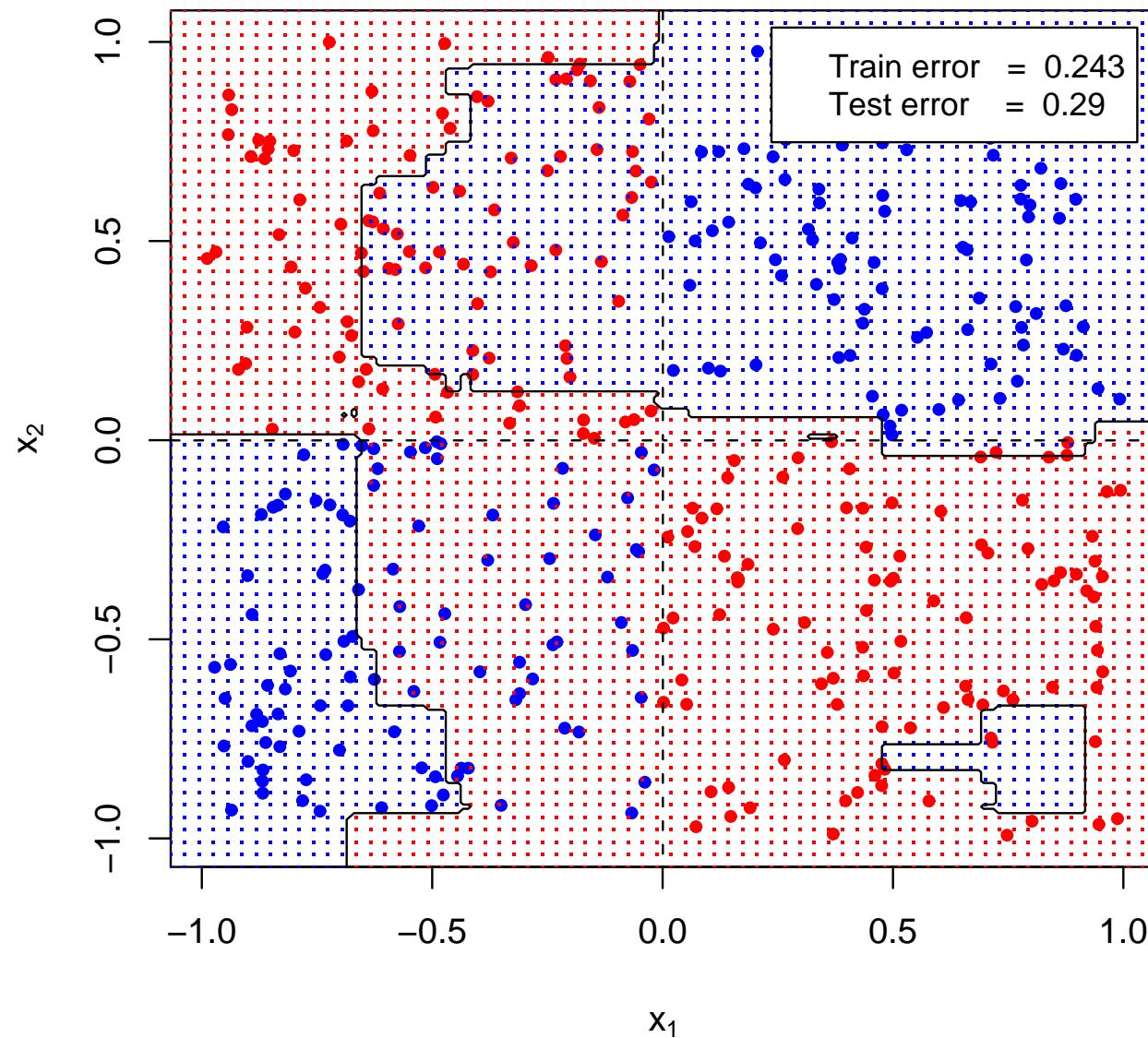
xor: низких деревьев (высоты 1 или 2) должно быть много.

Еще при 1000 деревьях картина не удовлетворительная.

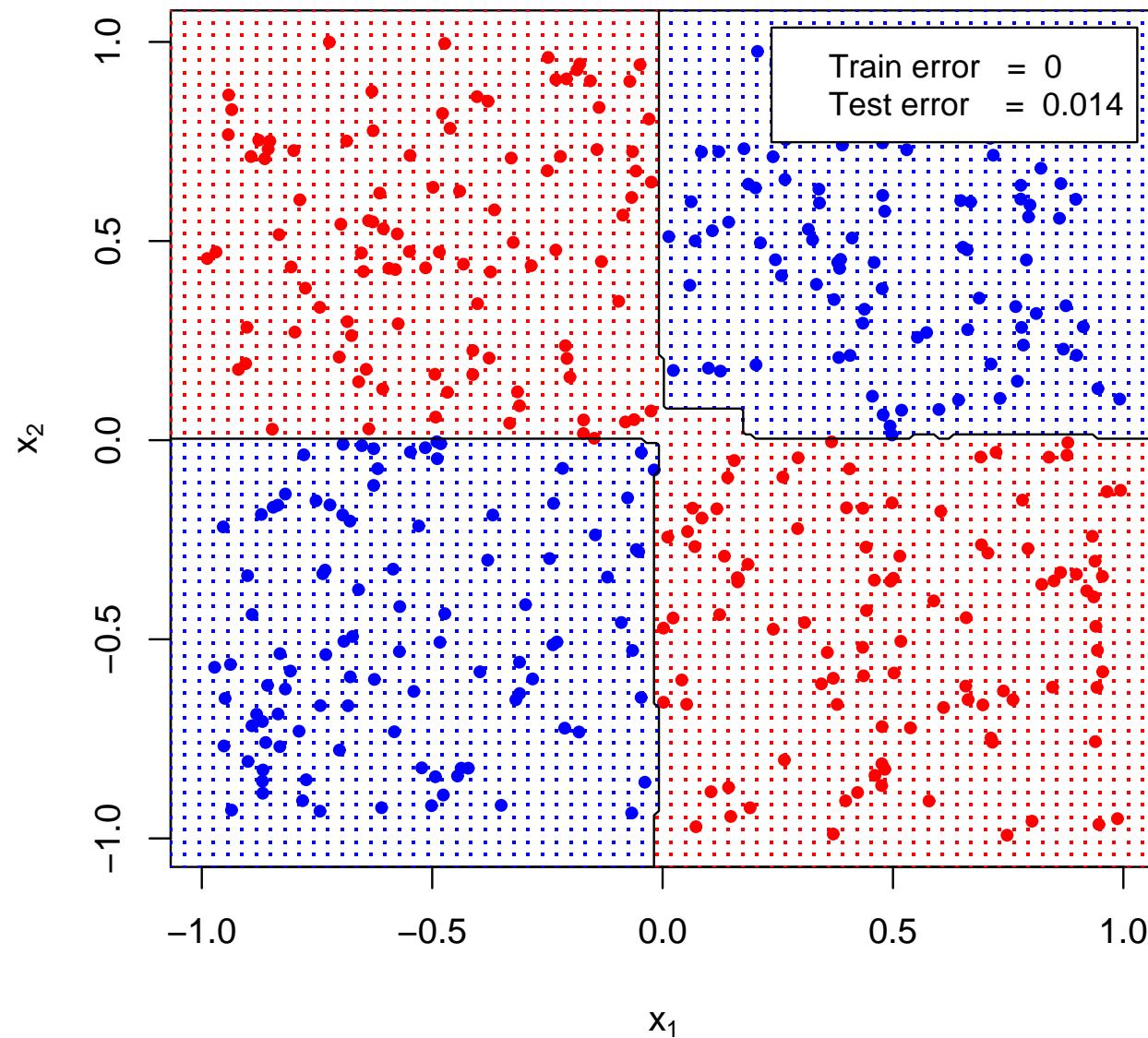
На рис.: деревья решений высоты 1 (100 деревьев).

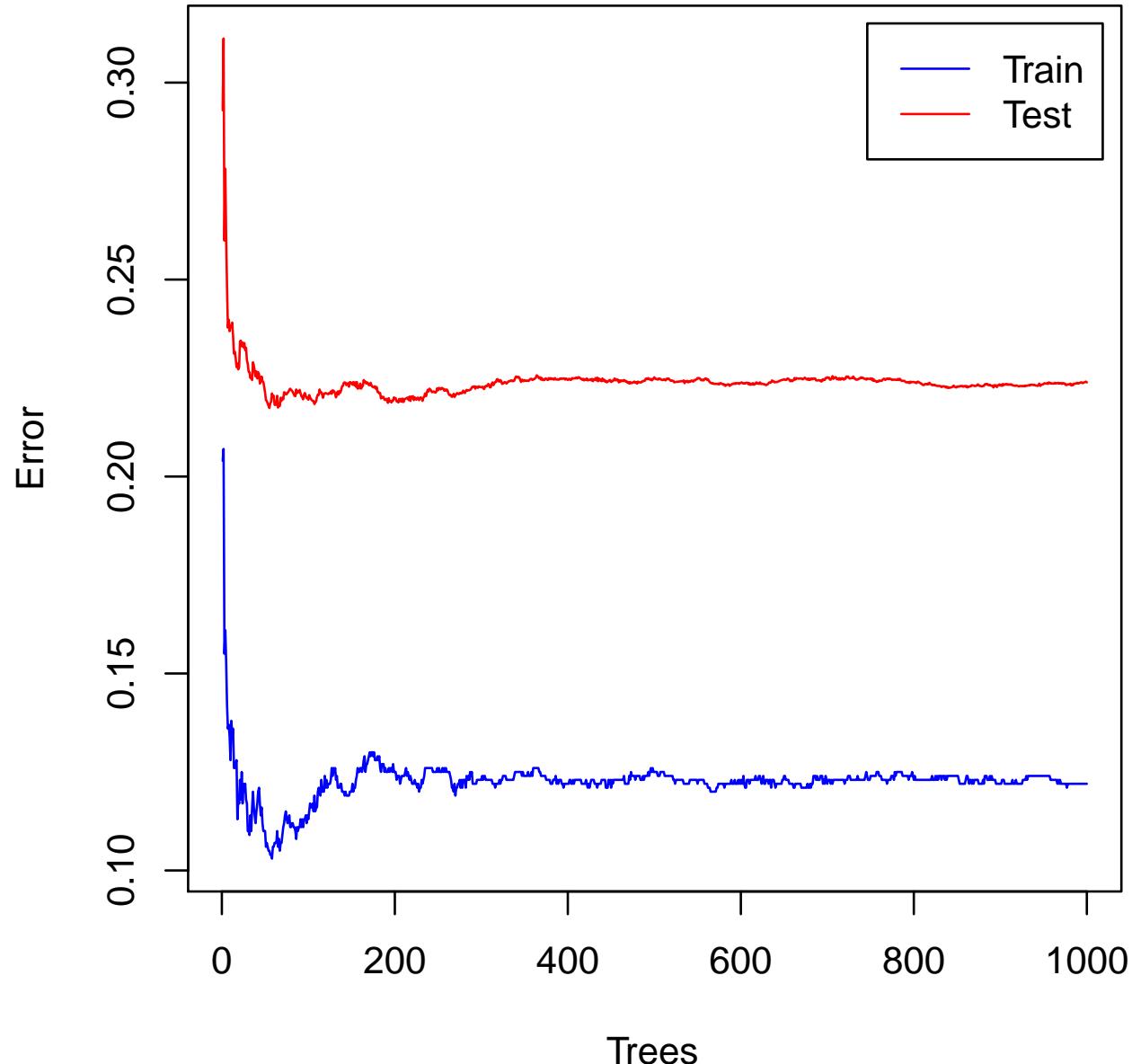


## Bagging – деревья решений высоты 2 (100 деревьев)



## Bagging – деревья решений высоты 3 (100 деревьев)





Переобучается ли баггинг?

## 8.2. Случайный лес

Развитие идеи баггинга: Random forests [Breiman, 2001]

Ансамбль параллельно обучаемых «независимых» деревьев решений.

Независимое построение определенного количества  $M$  (например, 500) деревьев:

- генерация случайной bootstrap-выборки из обучающей выборки (50–70% от размера всей обучающей выборки);
- построение дерева решений по данной подвыборке: в каждом новом узле дерева переменная для разбиения выбирается не из всех признаков, а из случайно выбранного их подмножества небольшой мощности.

**procedure** RandomForests

**for**  $m = 1, 2, \dots, M$

**begin**

По обучающей выборке построить бутстрэп-выборку

Построить дерево  $f_m$ , рекурсивно применяя следующую процедуру,

пока не будет достигнут минимальный размер sz:

**begin**

Построить случайный набор из  $p$  признаков

Выбрать из него лучшую переменную и построить 2 дочерних узла

**end**

**end**

Для задачи восстановления регрессии **return**  $f = \frac{1}{M} \sum_{m=1}^M f_m$

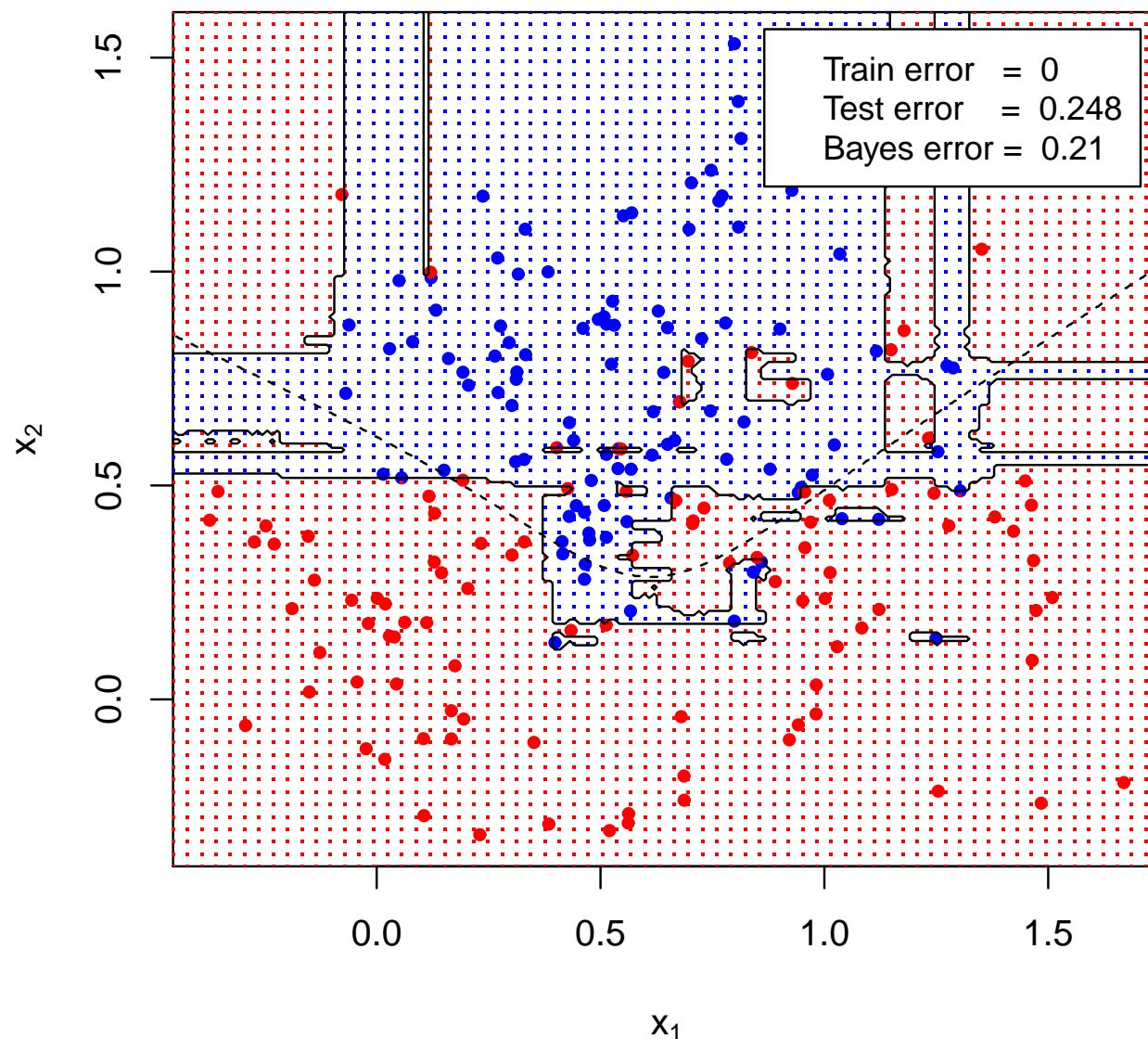
Для задачи классификации **return**  $f = \operatorname{argmax}_k \sum_{m=1}^M I(f_m = k)$

**end**

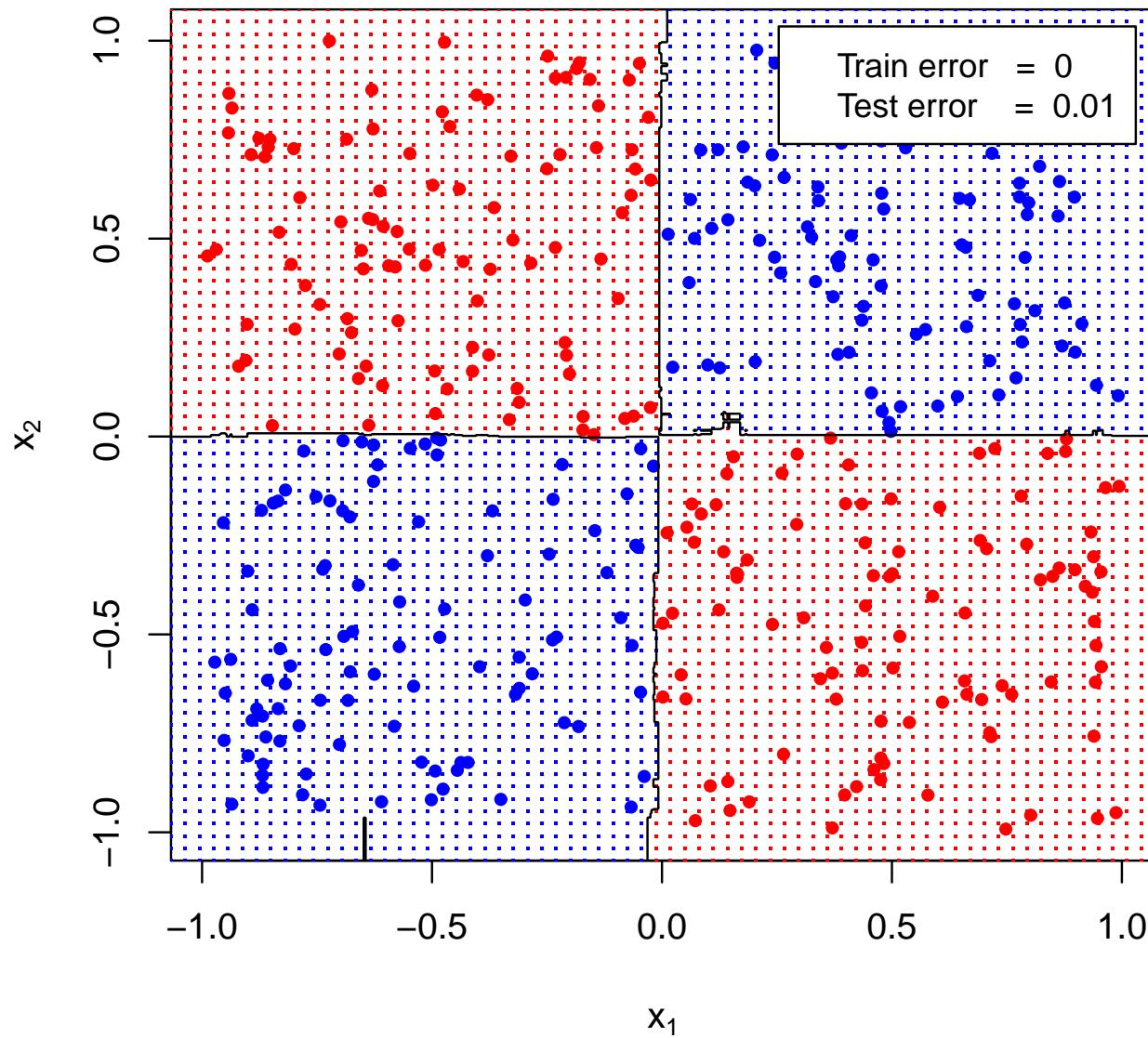
Для задачи восстановления регрессии, например,  $p = \sqrt{d}$ , sz = 3

Для задачи классификации, например,  $p = d/3$ , sz = 1

Random forest: 500 деревьев

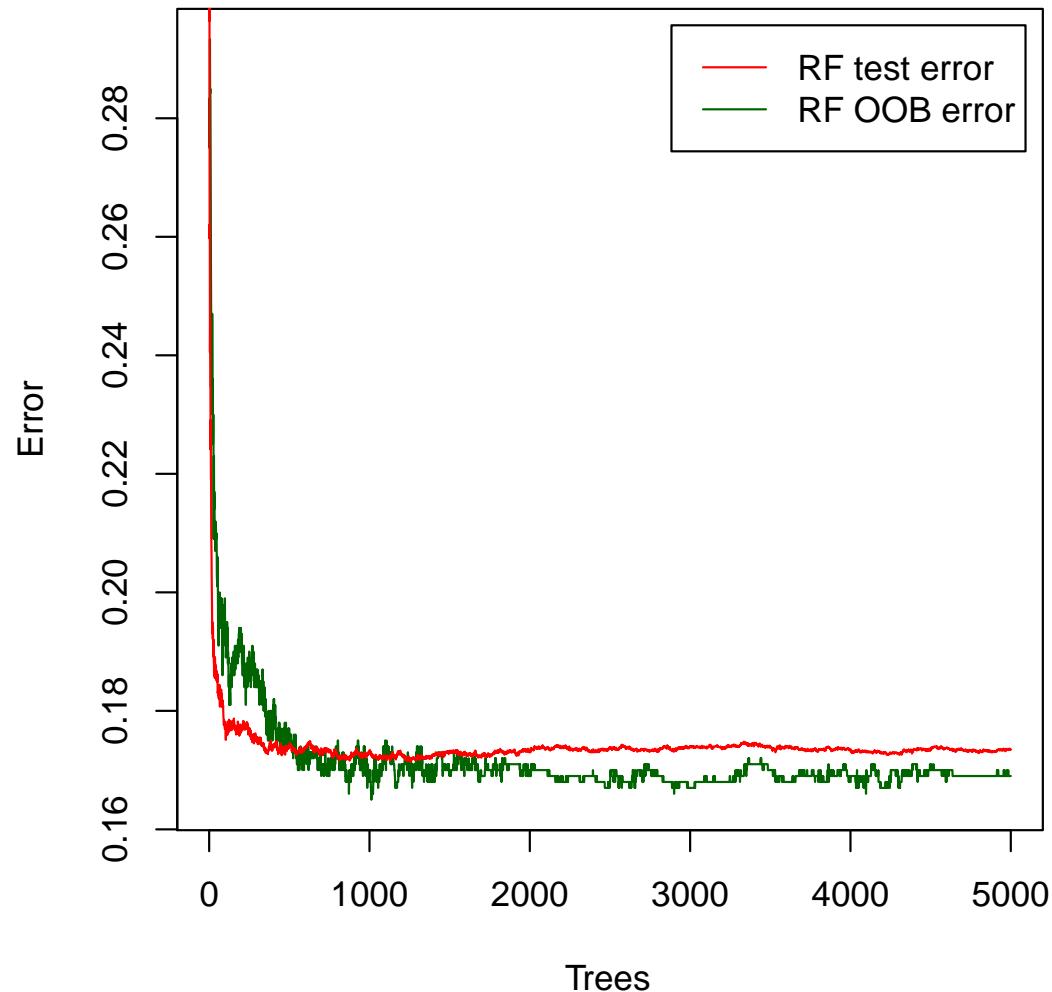


## Random Forest (50 деревьев)



## Out of bag error

OOB ошибка – средняя ошибка предсказаний классификатора (RF) на обучающей выборке, усредненная по всем слабым классификаторам (деревьям), при обучении которых данных объект не вошел в бутстрэп-выборку.



## **Extremely randomized trees**

[Geurts, Ernst, Wehenkel, 2006]

Основные отличия от Random Forests:

- Бутстрэп-выборки не используются. Для обучения каждого дерева используется целиком все обучающее множество.
- При ветвлении, как и в RF, выбираем случайное подмножество (неконстантных в данном ящике) признаков. Затем для каждой такой переменной строим случайное разбиение и выбираем из них лучшее.

## 8.3. Бустинг

To boost — улучшать, повышать, рекламировать.

Попробуем строить последовательность решающих правил, каждый из которых осведомлен об ошибках предыдущих.

Пусть только два класса:  $\mathcal{Y} = \{-1, 1\}$ .

### Простая схема [Schapire, 1990]

3 классификатора

$f_1$  обучается на  $N$  прецедентах

$f_2$  обучается на  $N$  прецедентах, таких, что  $f_1$  ровно на половине дает верный ответ

$f_3$  обучается на  $N$  прецедентах, на которых  $f_1(x) \neq f_2(x)$

**return**  $f = \text{sign} \left( \sum_{m=1}^3 f_m \right)$

Откуда брать данные для новых обучающих выборок?

— Например, из исходной выборки путем изъятия с возвращением (бутстрэп-выборка)

## AdaBoost [Freund, Schapire, 1995]

(от Adaptive Boosting)

Будем использовать веса  $w_1, w_2, \dots, w_N$ .

На первой итерации  $w_i = 1/N$  ( $i = 1, 2, \dots, N$ ) и алгоритм построения  $f_1$  работает в обычном режиме.

На  $m$ -й итерации увеличиваются веса тех прецедентов, на которых на  $(m - 1)$ -й итерации была допущена ошибка, и уменьшаются веса тех прецедентов, которые на предыдущей итерации были классифицированы правильно.

На  $m$ -й итерации ищем классификатор  $f_m \in \mathcal{F}_m$ , минимизирующий ошибку

$$\text{err}_m = \sum_{i=1}^N w_i \cdot I(y^{(i)} \neq f_m(x^{(i)})) = \sum_{y^{(i)} \neq f_m(x^{(i)})} w_i.$$

— Некоторые алгоритмы обучения принимают на вход веса  $w_i$ .

Если это не возможно, то на каждой итерации генерировать бутстрэп выборку, изымая (с возвращением) из обучающей выборки

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

$i$ -й элемент с вероятностью  $w_i$ .

**procedure** AdaBoost

Положить  $w_i \leftarrow 1/N \quad (i = 1, 2, \dots, N)$

**for**  $m = 1, 2, \dots, M$

Найти  $f_m \in \mathcal{F}_m$ , минимизирующую ошибку  $\text{err}_m = \sum_{i=1}^N w_i \cdot I(y^{(i)} \neq f_m(x^{(i)}))$

**if**  $\text{err}_m > \frac{1}{2}$

**break**

Вычислить  $\beta_m \leftarrow \frac{1}{2} \ln \frac{1 - \text{err}_m}{\text{err}_m}$

$w'_i \leftarrow w_i \cdot e^{-\beta_m y^{(i)} f_m(x^{(i)})} = w_i \cdot \left( \sqrt{\frac{1 - \text{err}_m}{\text{err}_m}} \right)^{-y^{(i)} f_m(x^{(i)})} \quad (i = 1, 2, \dots, N)$

$w_i \leftarrow w'_i / \sum_{j=1}^N w'_j \quad (i = 1, 2, \dots, N)$

**end**

**return**  $f = \text{sign } h$ , где  $h = \sum_{m=1}^M \beta_m f_m$

**end**

**Теорема 8.1** Если  $\text{err}_m \leq \frac{1}{2} - \varepsilon_m$ , то для эмпирической ошибки  $\widehat{R}(f)$  итогового классификатора, построенного алгоритмом AdaBoost, справедливо

$$\widehat{R}(f) \leq \prod_{m=1}^M 2\sqrt{\text{err}_m(1 - \text{err}_m)} = \prod_{m=1}^M \sqrt{1 - 4\varepsilon_m^2} \leq e^{-2 \sum_{m=1}^M \varepsilon_m^2}.$$

**Теорема 8.2** Для ошибки предсказания  $R(f)$  алгоритма AdaBoost справедливо

$$R(f) = \widehat{R}(f) + \tilde{O}\left(\sqrt{\frac{M \cdot v}{N}}\right),$$

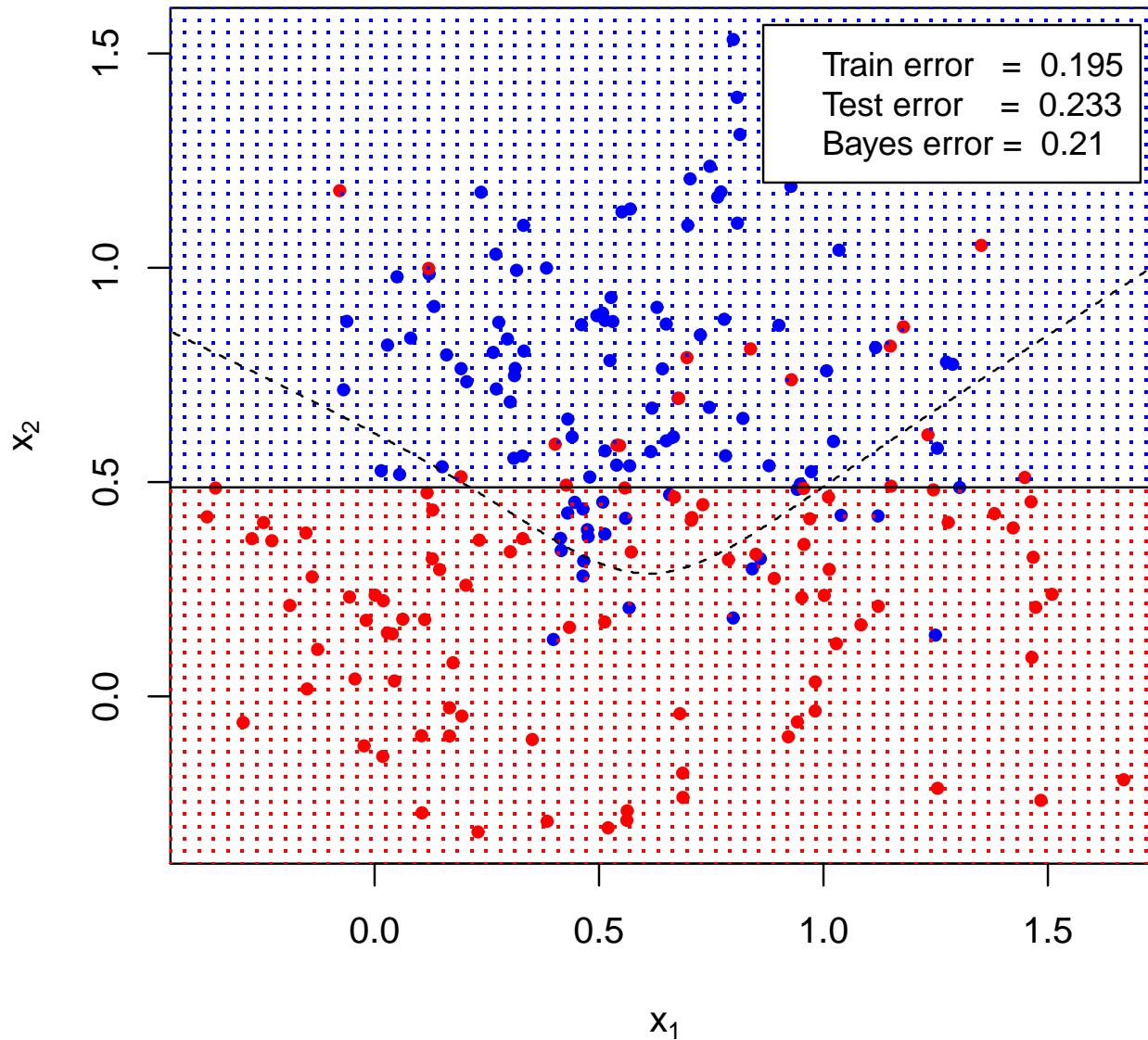
где  $v = \text{VC}(\mathcal{F}_m)$  — размерность Вапника–Червоненкиса для класса  $\mathcal{F}_m$  ( $m = 1, 2, \dots, M$ ).

Большое  $M$  — возможно переобучение.

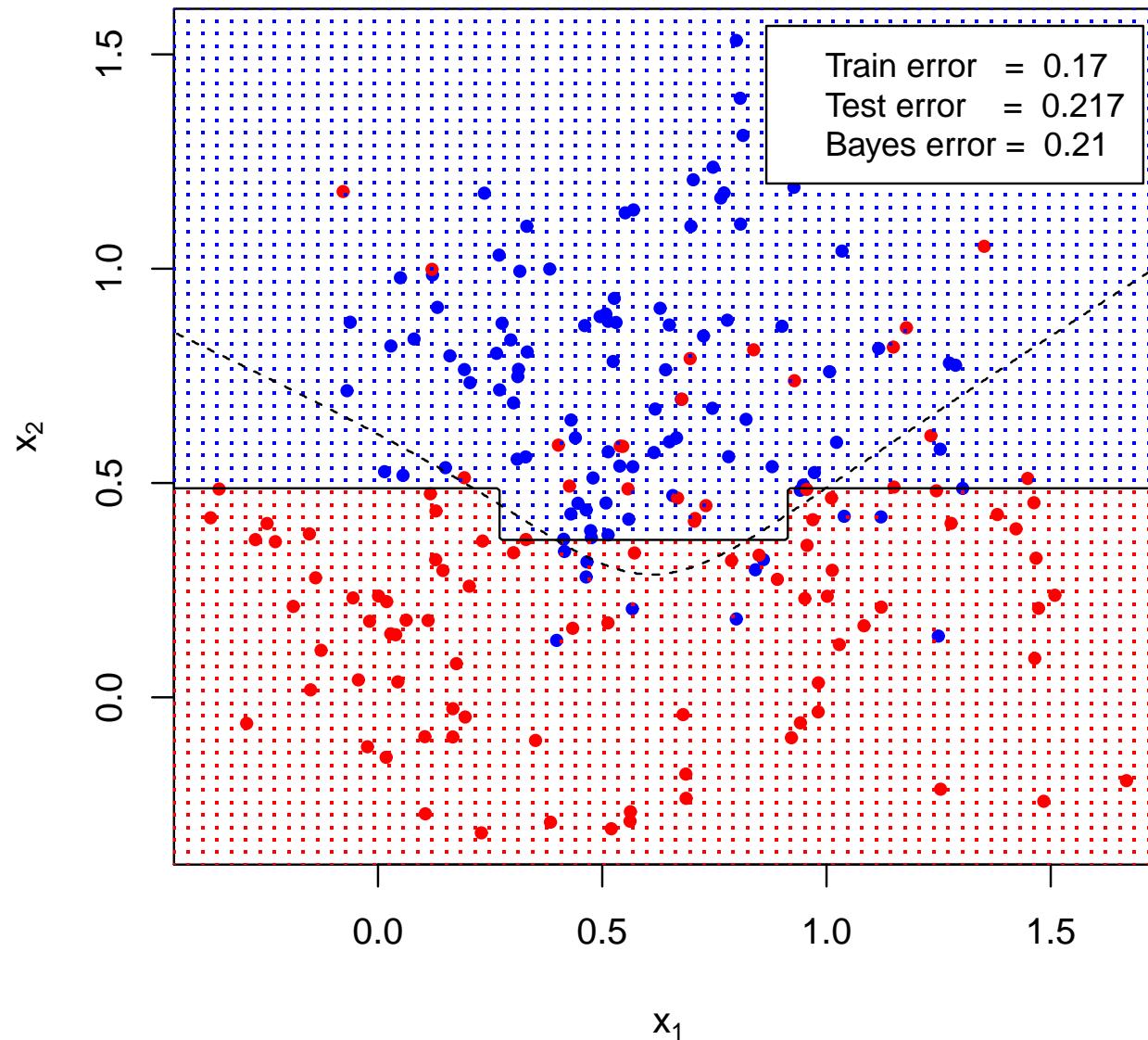
## Пример

Слабые классификаторы — деревья решений высоты 1 (*stumps*)

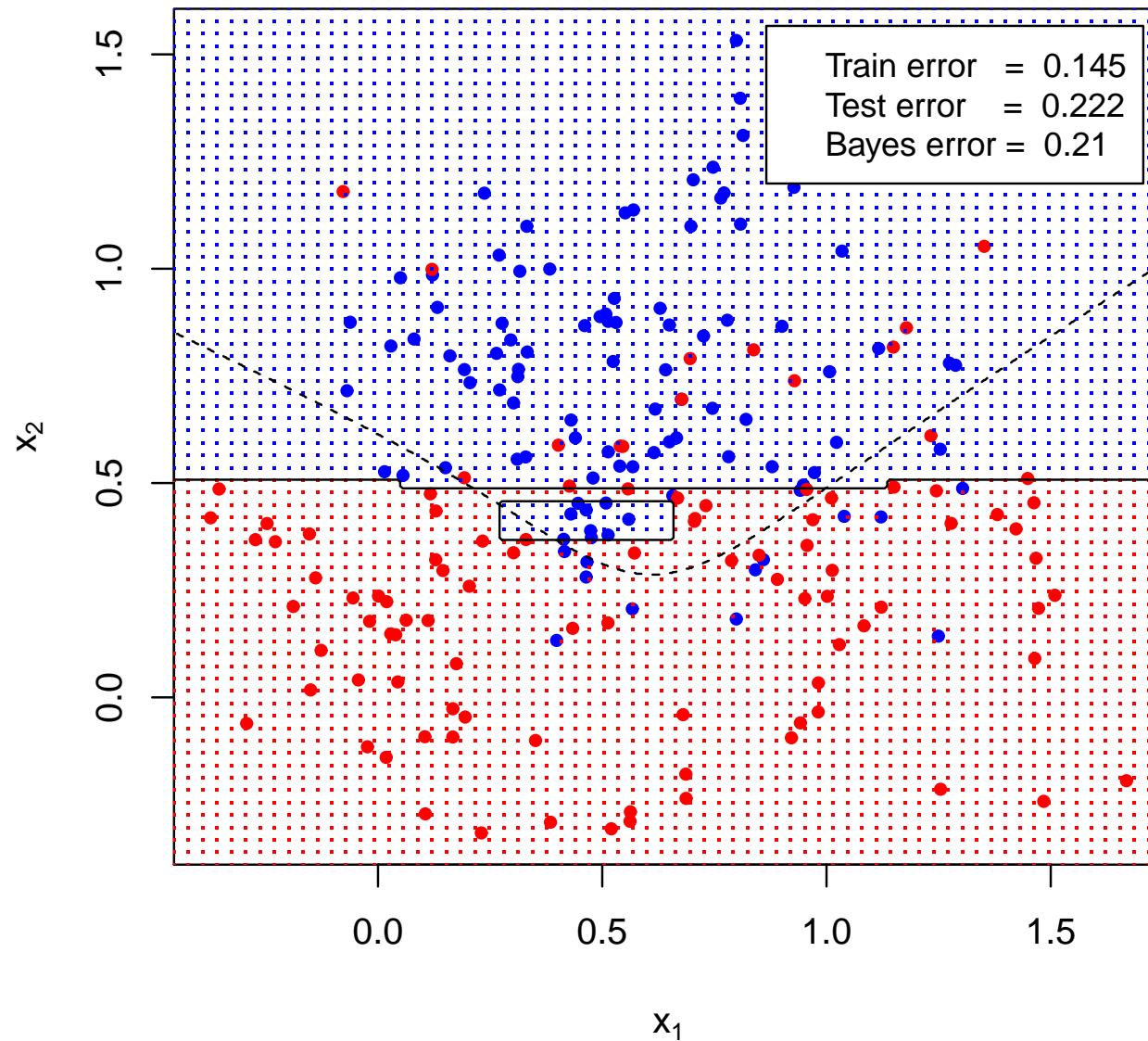
$$M = 1$$



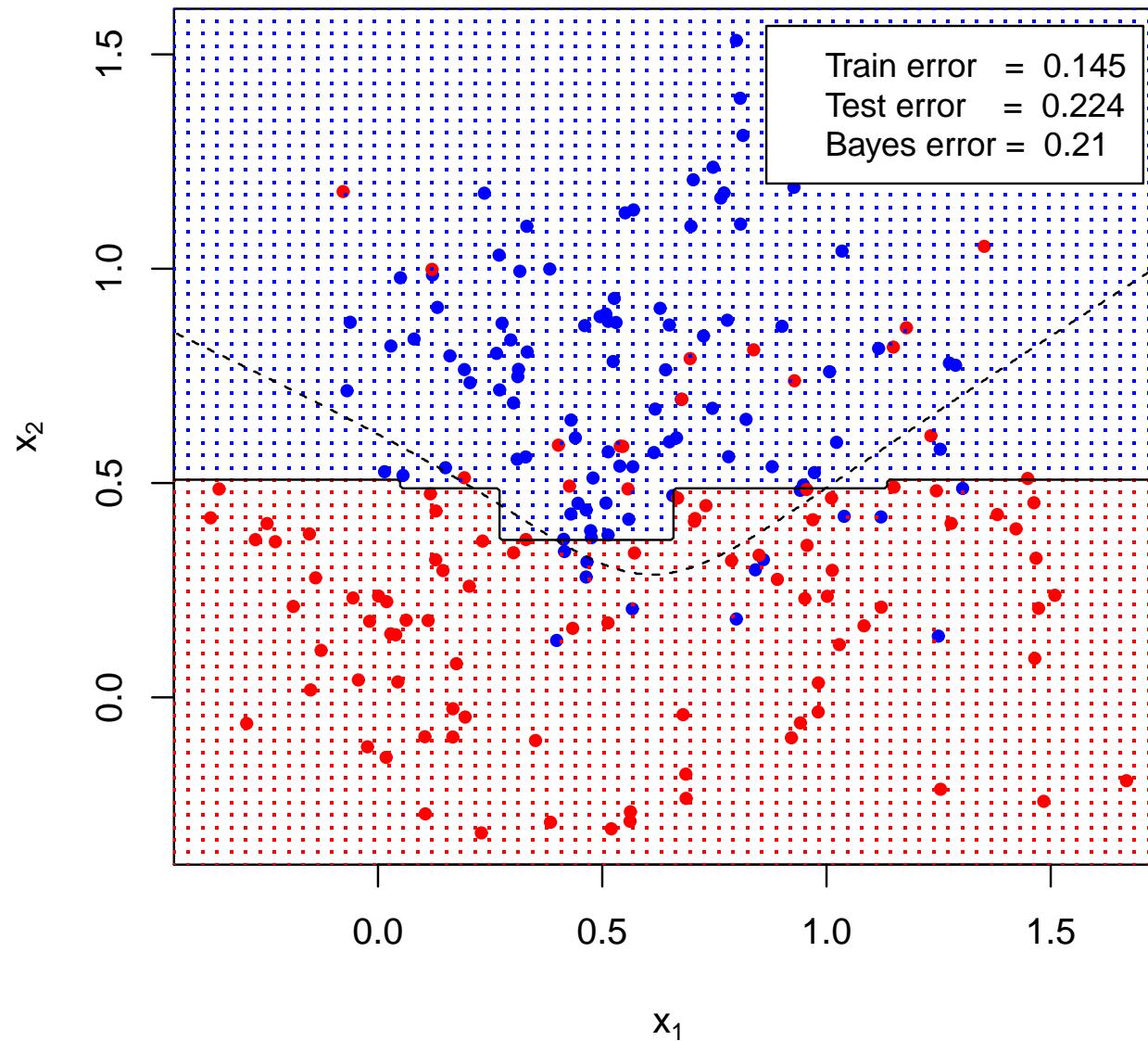
$M = 25$



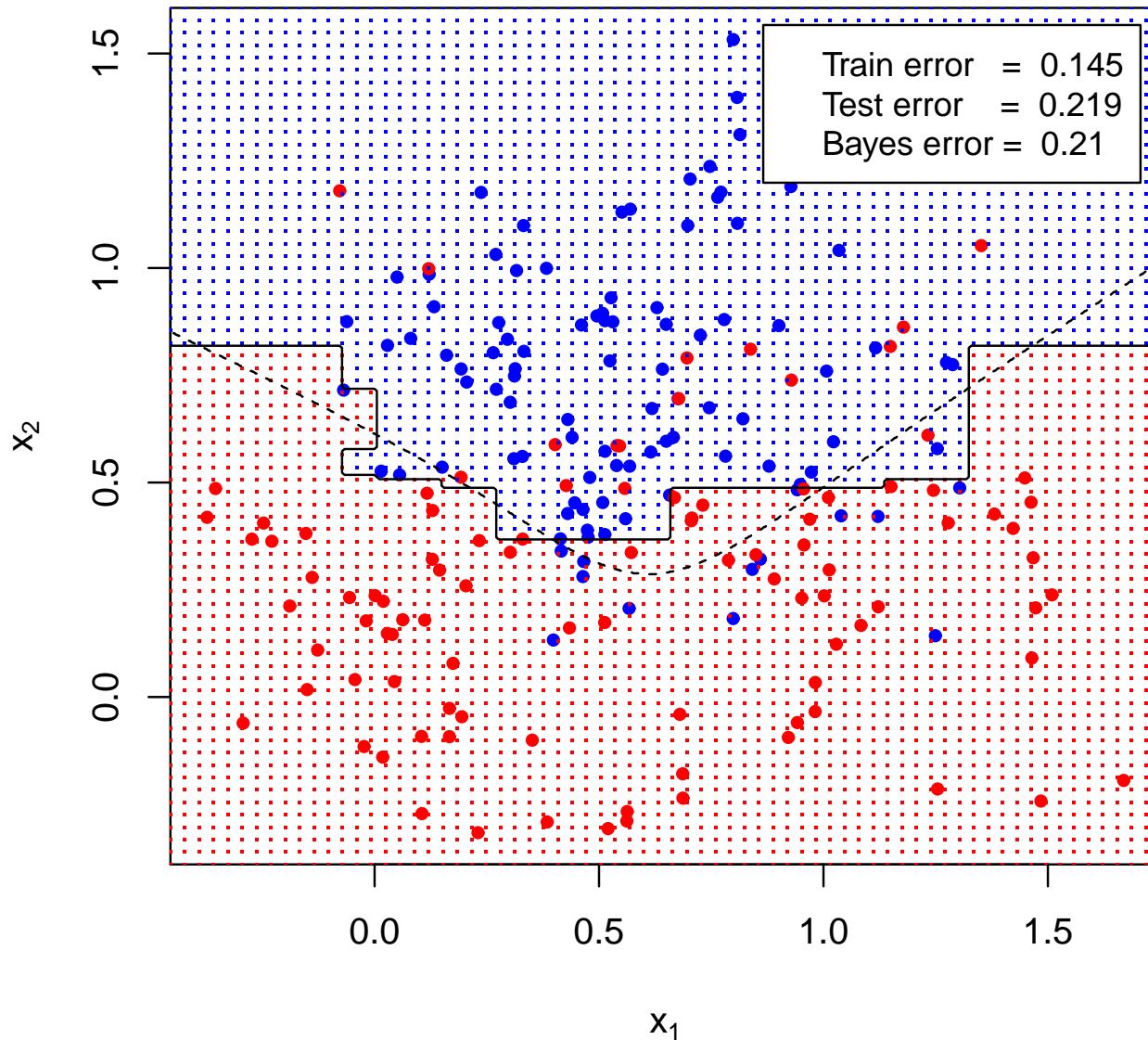
$$M = 50$$



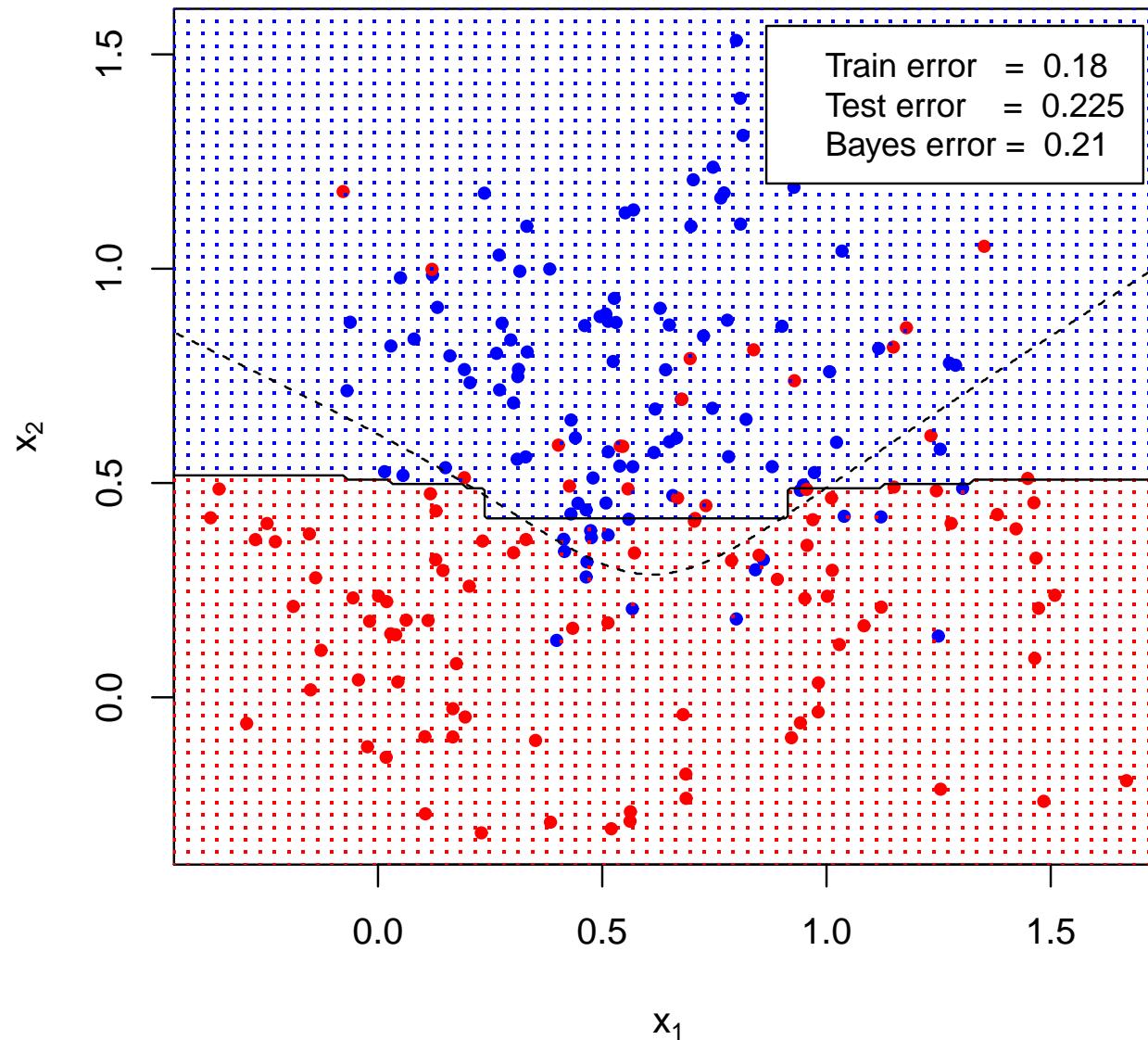
$M = 75$



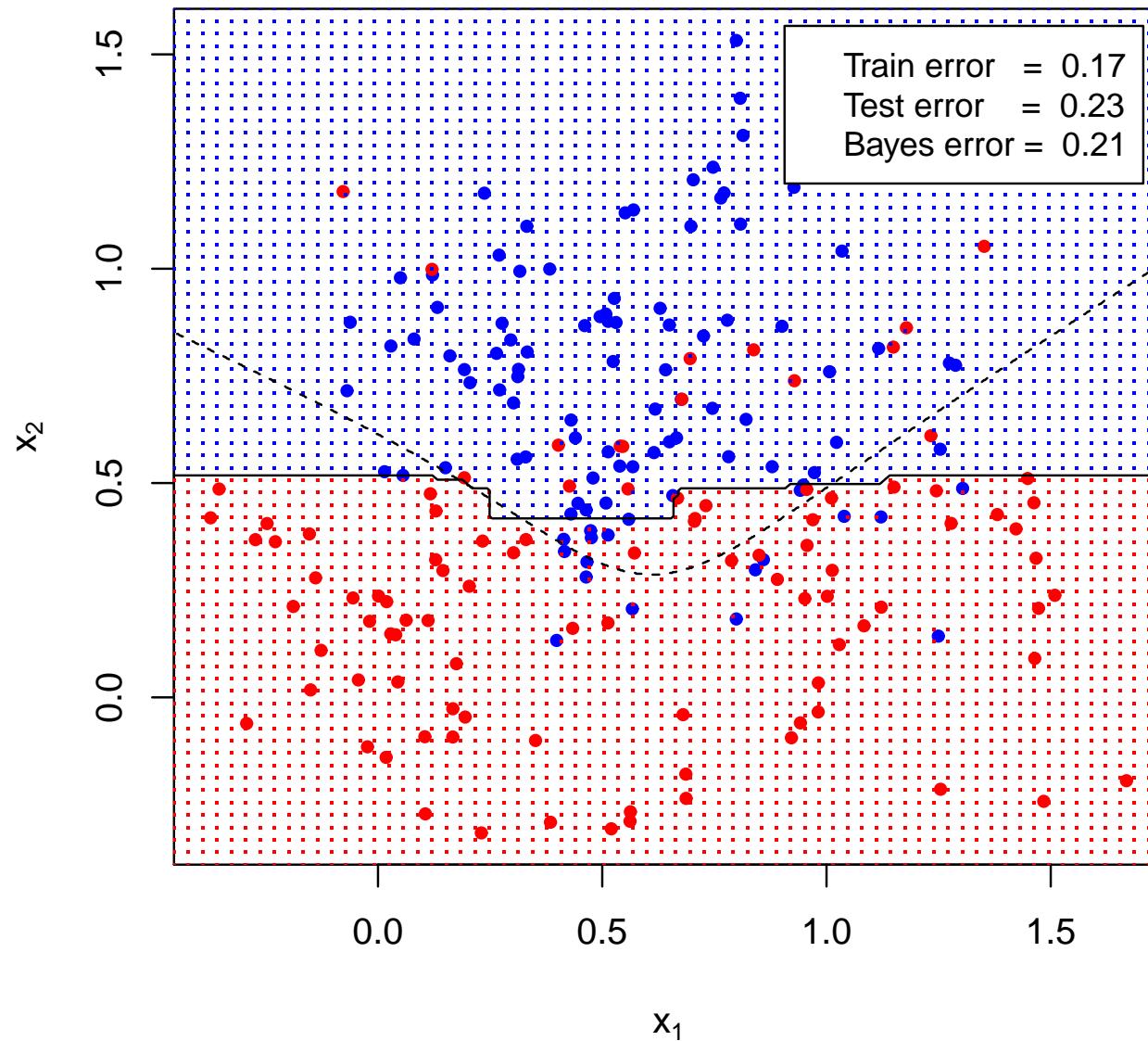
$M = 100$



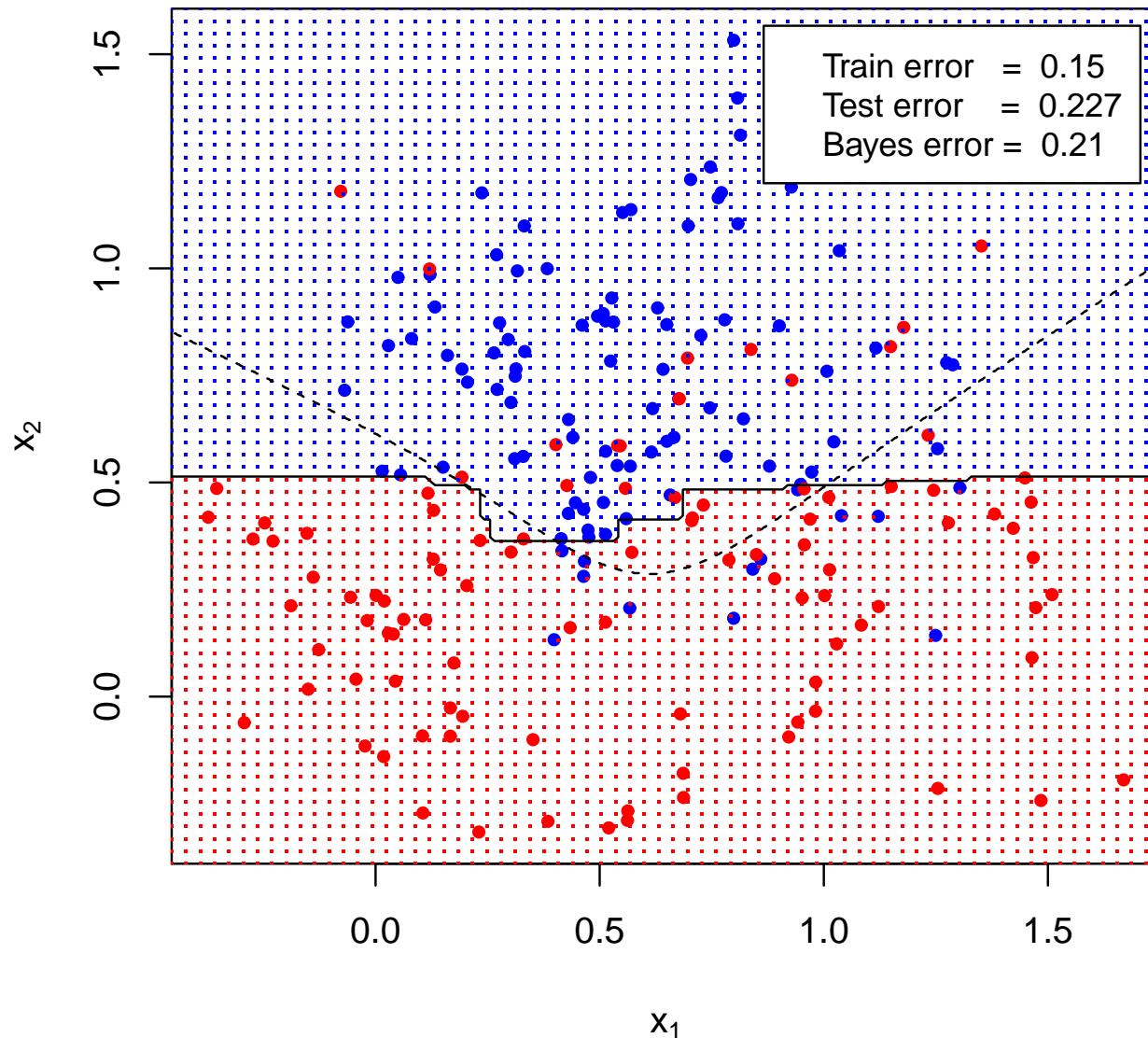
$M = 150$



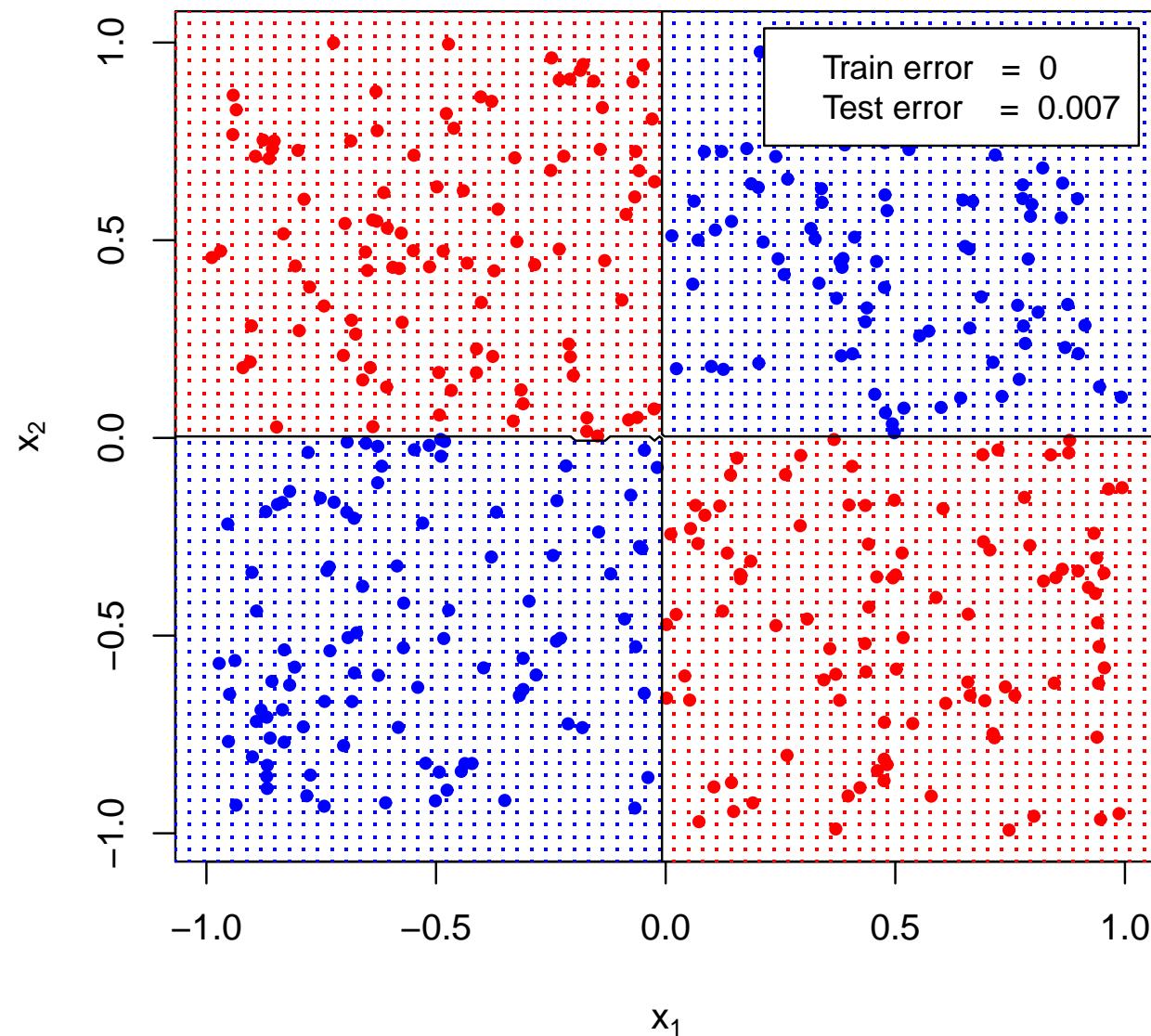
$M = 200$



$M = 1000$



## 50 деревья решений высоты 2

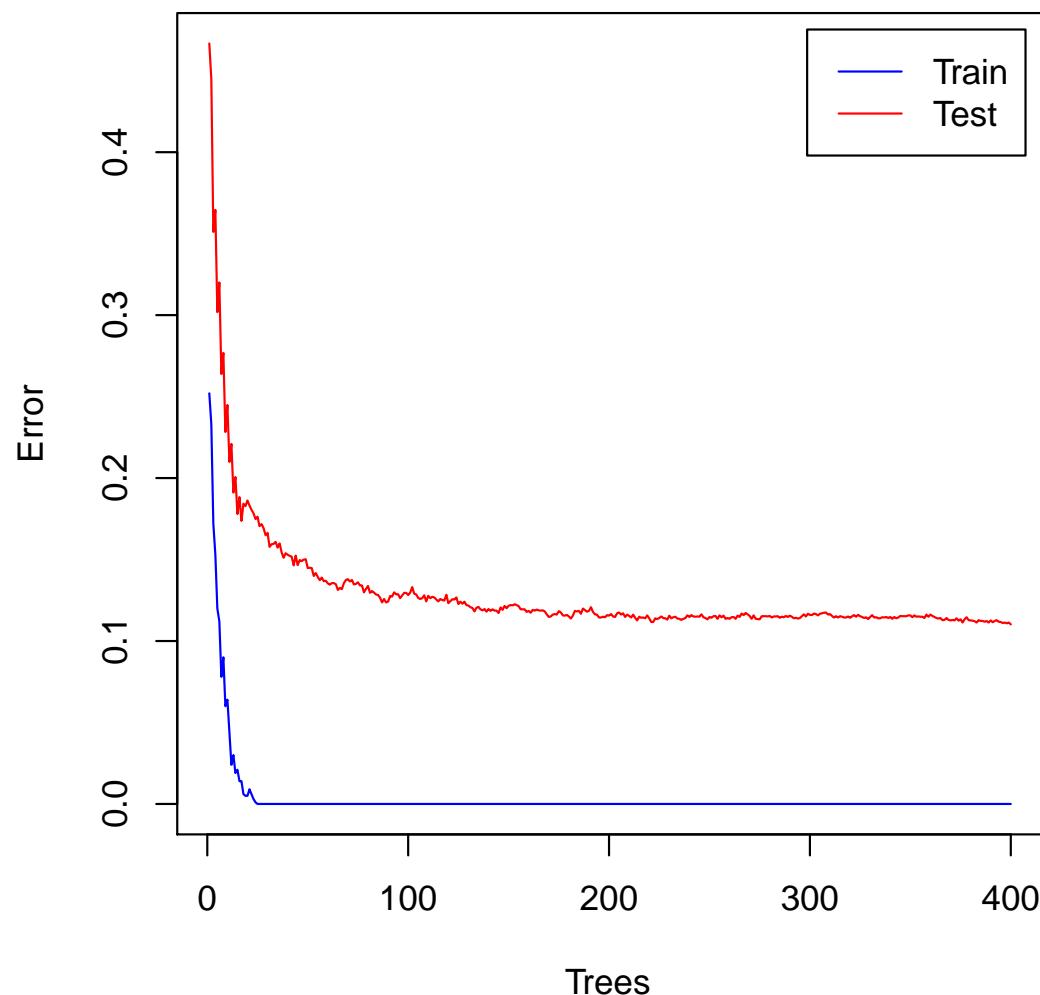


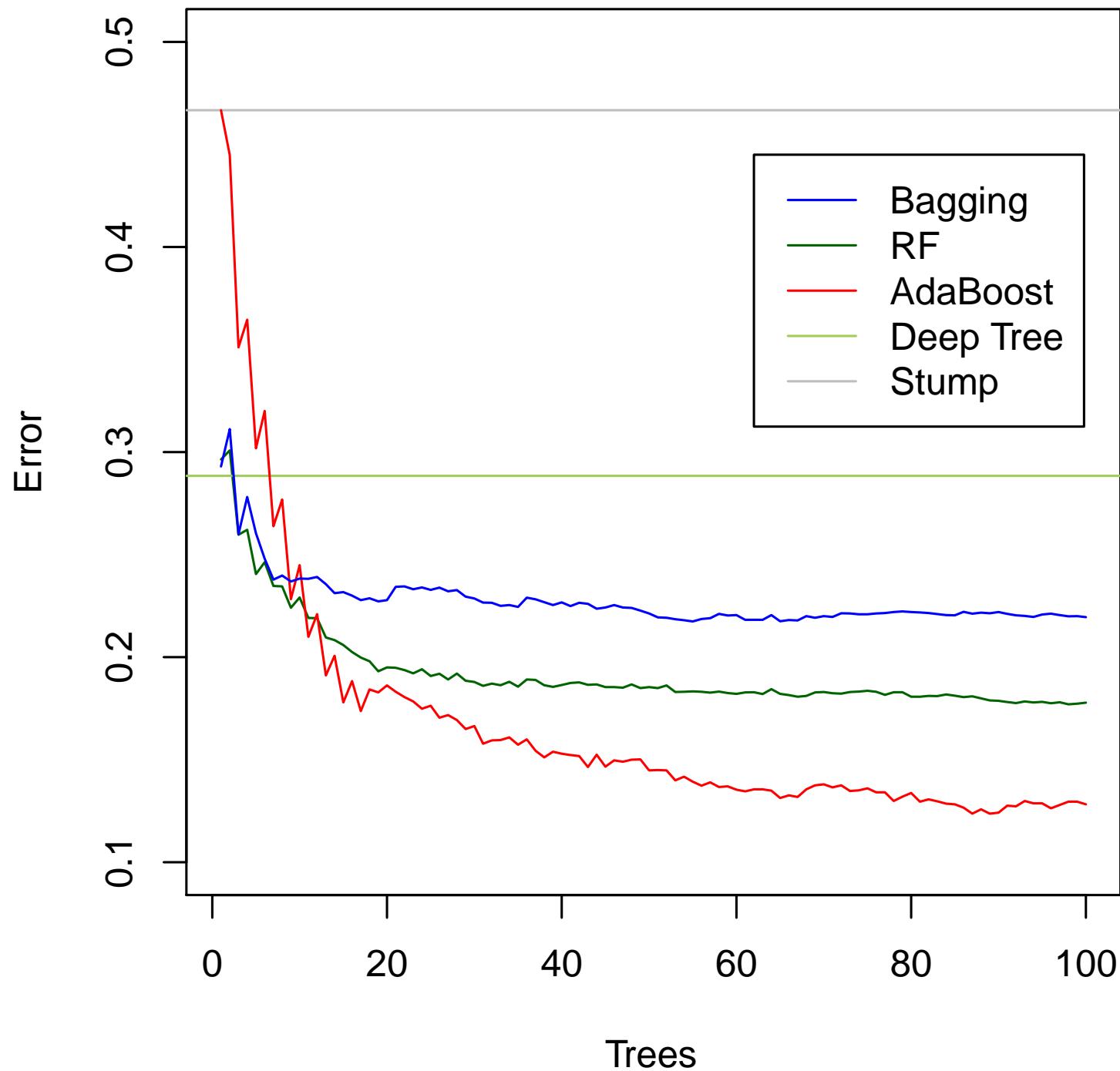
(Если брать деревья решений высоты 1 — не хватает даже 1000)

## Пример

$d = 10, \mathcal{X} = \mathbf{R}^d, \mathcal{Y} = \{-1, 1\}, N = 1000$  — обучающая выборка,  $N_{\text{test}} = 10000$

$$y = 1 \Leftrightarrow \sum_{j=1}^d x_j^2 \leq \chi^2_{10}(0.5) \approx 9.34$$





<i>Алгоритм</i>	<i>Ошибка</i> <i>на обучающей выборке на тестовой выборке</i>	
Одиночное дерево	0.01	0.29
Stump	0.43	0.47
Bagging (100 деревьев)	0.10	0.22
RF (100 деревьев)	0.00	0.18
AdaBoost (100 stumps)	0	0.12
GBT (100 stumps)	0.26	0.32

Одиночное дерево: 138 листьев

## Пример. Классификации рукописных цифр

M	<i>Ошибка</i>	
	<i>на обучающей выборке</i>	<i>на тестовой выборке</i>
5	0.070	0.121
10	0.017	0.074
20	0.001	0.056
30	0	0.048
40	0	0.044

<i>Алгоритм</i>	<i>Ошибка</i>	
	<i>на обучающей выборке</i>	<i>на тестовой выборке</i>
Машина опорных векторов	0 %	2.1 %
Метод ближайшего соседа	0 %	3.1 %
Нейронная сеть	0.0 %	2.9 %
AdaBoost	0 %	4.8 %

AdaBoost.M1, AdaBoost.M2, RealBoost, SAMME, SAMME.R — различные варианты алгоритма AdaBoost для задачи классификации с  $K > 2$

## 8.4. Бустинг – настройка аддитивной модели

Аддитивная модель:

$$f(x) = \text{sign } h(x), \quad h(x) = \sum_{m=1}^M \beta_m b(x, \theta_m)$$

Минимизация эмпирического риска (может быть неприподъемной задачей):

$$\min_{\beta, \theta} \sum_{i=1}^N L \left( \sum_{m=1}^M \beta_m b(x^{(i)}, \theta_m), y^{(i)} \right).$$

**begin** Прямой пошаговый алгоритм (forward stagewise additive modeling)

Положить  $h_0(x) \equiv 0$

**for**  $m = 1, 2, \dots, M$

Вычислить  $(\beta_m, \theta_m) = \underset{\beta, \theta}{\operatorname{argmin}} \sum_{i=1}^N L(h_{m-1}(x_i) + \beta b(x^{(i)}, \theta), y^{(i)})$

Положить  $h_m \equiv h_{m-1} + \beta_m b(\cdot, \theta_m)$

**end**

**return**  $h \equiv h_M$

**end**

AdaBoost эквивалентен прямому пошаговому алгоритму, если  $L(h, y) = e^{-yh}$ :

$$\begin{aligned}
 (\beta_m, b_m) &\leftarrow \operatorname{argmin}_{\beta, b} \sum_{i=1}^N L\left(h_{m-1}(x^{(i)}) + \beta b(x^{(i)}), y^{(i)}\right) \\
 \sum_{i=1}^N L\left(h_{m-1}(x^{(i)}) + \beta b(x^{(i)}), y^{(i)}\right) &= \sum_{i=1}^N \underbrace{e^{-y^{(i)}h_{m-1}(x^{(i)})}}_{w_i'^{(m)}} \cdot e^{-\beta y^{(i)}b(x^{(i)})} = \sum_{i=1}^N w_i'^{(m)} e^{-\beta y^{(i)}b(x^{(i)})} = \\
 &= e^{-\beta} \cdot \sum_{y^{(i)}=b(x^{(i)})} w_i'^{(m)} + e^{\beta} \cdot \sum_{y^{(i)} \neq b(x^{(i)})} w_i'^{(m)} = (e^{\beta} - e^{-\beta}) \cdot \sum_{y^{(i)} \neq b(x^{(i)})} w_i'^{(m)} + e^{-\beta} \cdot \sum_{i=1}^N w_i'^{(m)}.
 \end{aligned}$$

Поэтому  $b_m = \operatorname{argmin}_b \sum_{y^{(i)} \neq b(x^{(i)})} w_i'^{(m)}$ .

Теперь минимизируем по  $\beta$ :

$$\beta_m = \frac{1}{2} \ln \frac{1 - \text{err}_m}{\text{err}_m}, \quad \text{где} \quad \text{err}_m = \frac{\sum_{y^{(i)} \neq b_m(x^{(i)})} w_i'^{(m)}}{\sum_{i=1}^N w_i'^{(m)}}, \quad w_i'^{(m+1)} = w_i^{(m)} \cdot e^{-\beta_m y^{(i)} b_m(x^{(i)})}.$$

## 8.5. Градиентный бустинг деревьев решений

GBT – Gradient Boosting Trees [Friedman, 1999]

- Задача классификации,  $\mathcal{Y} = \{-1, 1\}$ . Решающее правило:  $f(x) = \text{sign } h(x)$ .  
 $h$  характеризует «надежность» предсказания.

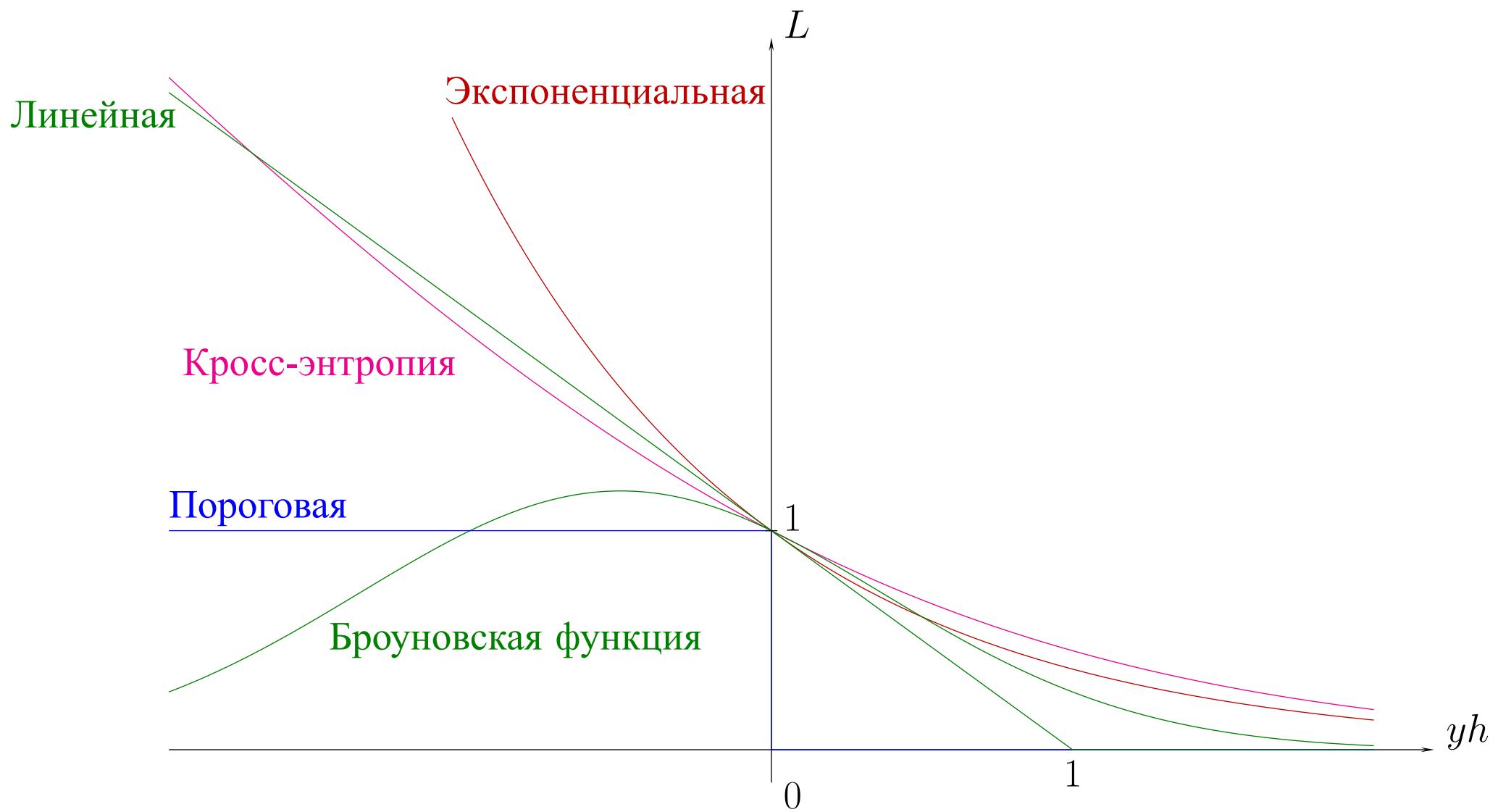
Штрафные функции (функции потерь):

- Пороговая функция:  $L(h, y) = I(yh < 0)$
- Линейная функция:  $L(h, y) = (1 - yh)I(yh < 1)$
- Кросс-энтропия:  $L(h, y) = \log_2(1 + e^{-yh})$
- Экспоненциальная функция:  $L(h, y) = e^{-yh}$
- Броуновская функция:  $L(h, y) = e^{-cyh(yh+s)}$

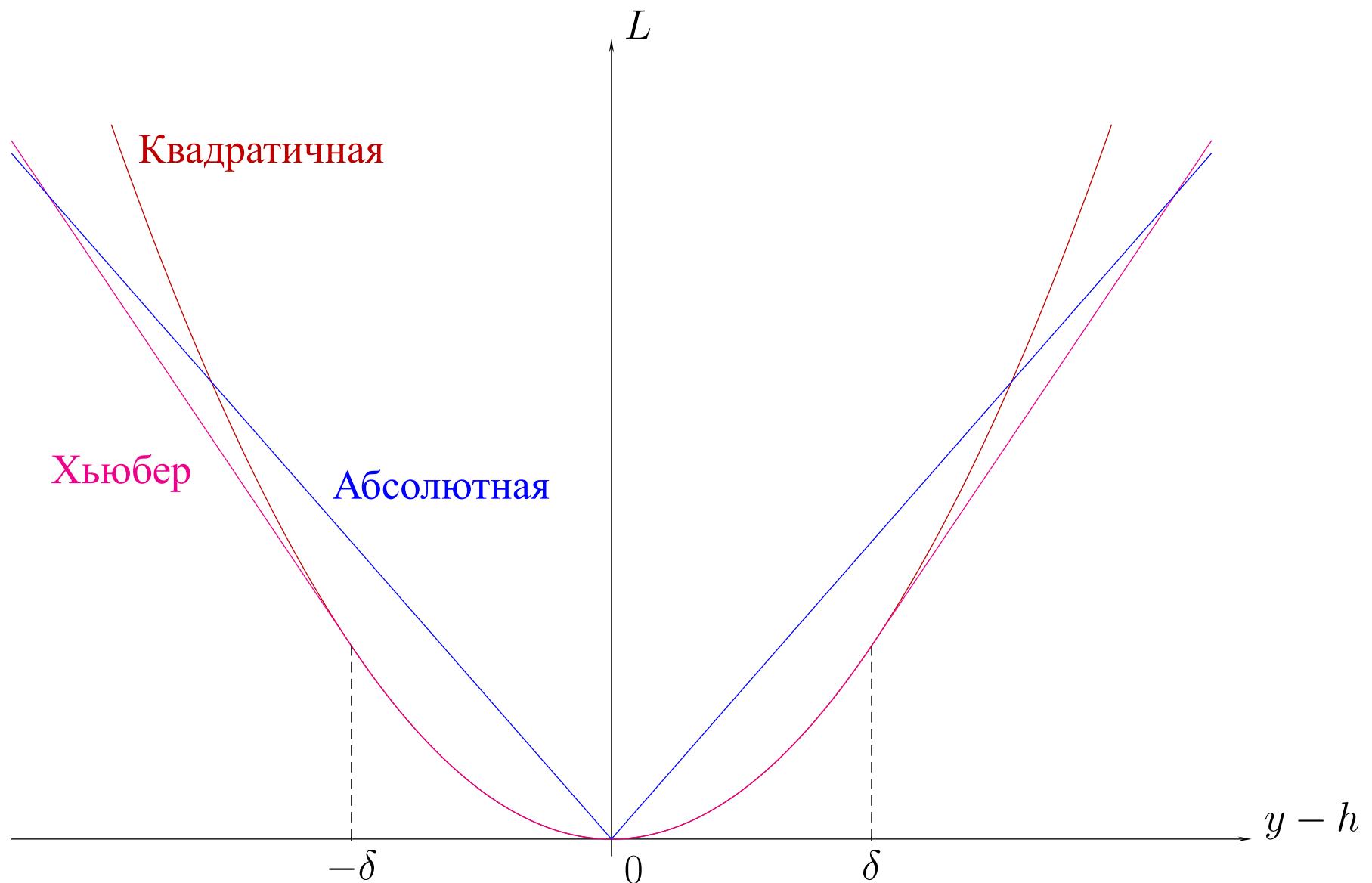
- Задача восстановления регрессии. Решающее правило:  $f(x) = h(x)$ .

Штрафные функции:

- Квадратичная функция:  $L(h, y) = (y - h)^2$
- Абсолютная:  $L(h, y) = 2|y - h|$
- Функция Хьюбера:  $(y - h)^2 \cdot I(|y - h| \leq \delta) + \delta \cdot (2|y - h| - \delta) \cdot I(|y - h| > \delta)$



$$\delta = 1.3$$



В случае произвольной гладкой штрафной функции алгоритмы для настройки модели можно получить по аналогии с соответствующими численными методами оптимизации.

Наша задача заключается в минимизации функционала

$$L(h) = \sum_{i=1}^N L\left(h(x^{(i)}), y^{(i)}\right),$$

где  $h$  — функция известного вида.

Опуская последнее ограничение, получаем задачу минимизации

$$\min_{\mathbf{h}} L(\mathbf{h}),$$

где  $\mathbf{h} = (h(x^{(1)}), h(x^{(2)}), \dots, h(x^{(N)})) \in \mathbf{R}^N$ .

На многие численные методы оптимизации можно смотреть как на процесс, который, начиная с  $\mathbf{h}_0$ , переходит от одной точки к другой, делая на  $m$ -й итерации шаг  $\mathbf{s}_m$ .

За  $M$  итераций алгоритм приходит в точку

$$\mathbf{h}_M = \mathbf{h}_0 + \sum_{m=1}^M \mathbf{s}_m, \quad \mathbf{s}_m \in \mathbf{R}^N.$$

В методе наискорейшего спуска шаг равен  $\mathbf{s}_m = -\rho_m \mathbf{g}_m$ , где  $\mathbf{g}_m$  — градиент функции  $L(\mathbf{h})$ , вычисленный в точке  $\mathbf{h}_{m-1}$ .

Компоненты вектора  $\mathbf{g}_m$  суть:

$$g_m^{(i)} = \left. \frac{\partial L(h(x^{(i)}), y^{(i)})}{\partial h(x^{(i)})} \right|_{h(x^{(i)})=h_{m-1}(x^{(i)})} \quad (i = 1, 2, \dots, N).$$

Параметр  $\rho_m$ , характеризующий длину шага, равен

$$\rho_m = \operatorname{argmin}_{\rho} L(\mathbf{h}_{m-1} - \rho \mathbf{g}_m).$$

К сожалению, градиент известен только на объектах из обучающей выборки.

Эту проблему можно решить путем обучения базовой модели (например, дерева решений) так, чтобы она наиболее точно предсказывала компоненты градиента.

Частные производные  $\frac{\partial L(y^{(i)}, h(x^{(i)}))}{\partial h(x^{(i)})}$  легко вычисляются аналитически.

## *Градиентный бустинг*

Положить  $h_0 \leftarrow \operatorname{argmin}_{\gamma} \sum_{i=1}^N L(y^{(i)}, \gamma)$

**for**  $m = 1, 2, \dots, M$

**for**  $i = 1, 2, \dots, N$

$$g_m^{(i)} \leftarrow - \frac{\partial L(y^{(i)}, h(x^{(i)}))}{\partial h(x^{(i)})} \Big|_{h=h_{m-1}}$$

**end**

    Обучить базовую модель  $g_m$  на выборке  $(x_1, g_m^{(1)}), (x_2, g_m^{(2)}), \dots, (x_N, g_m^{(N)})$ .

    Найти длину шага:  $\rho_m = \operatorname{argmin}_{\rho} L(h_{m-1} - \rho g_m(x))$ .

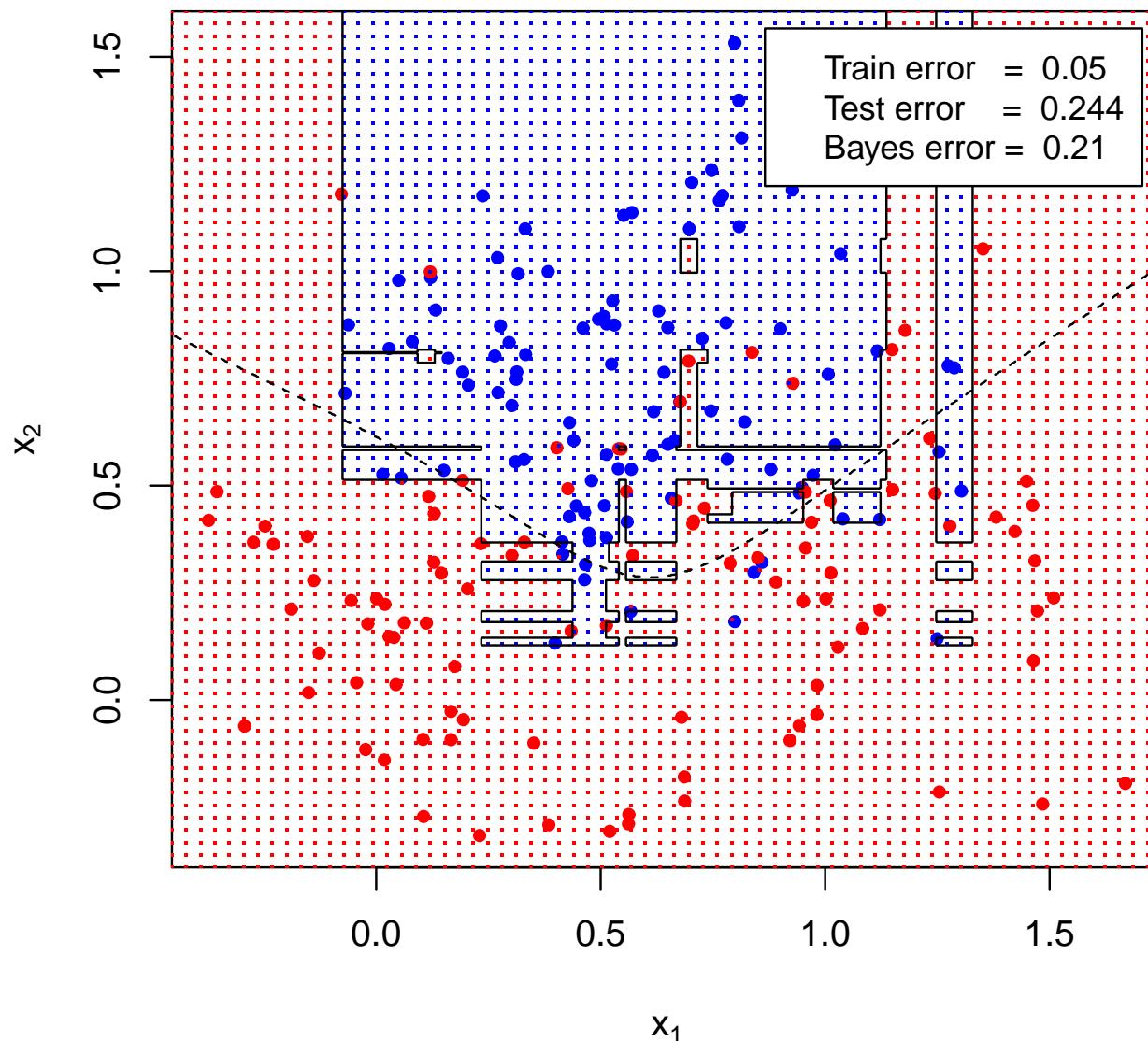
    Обновить модель:  $h_m = h_{m-1} - \gamma \rho_m g_m(x)$ .

**end**

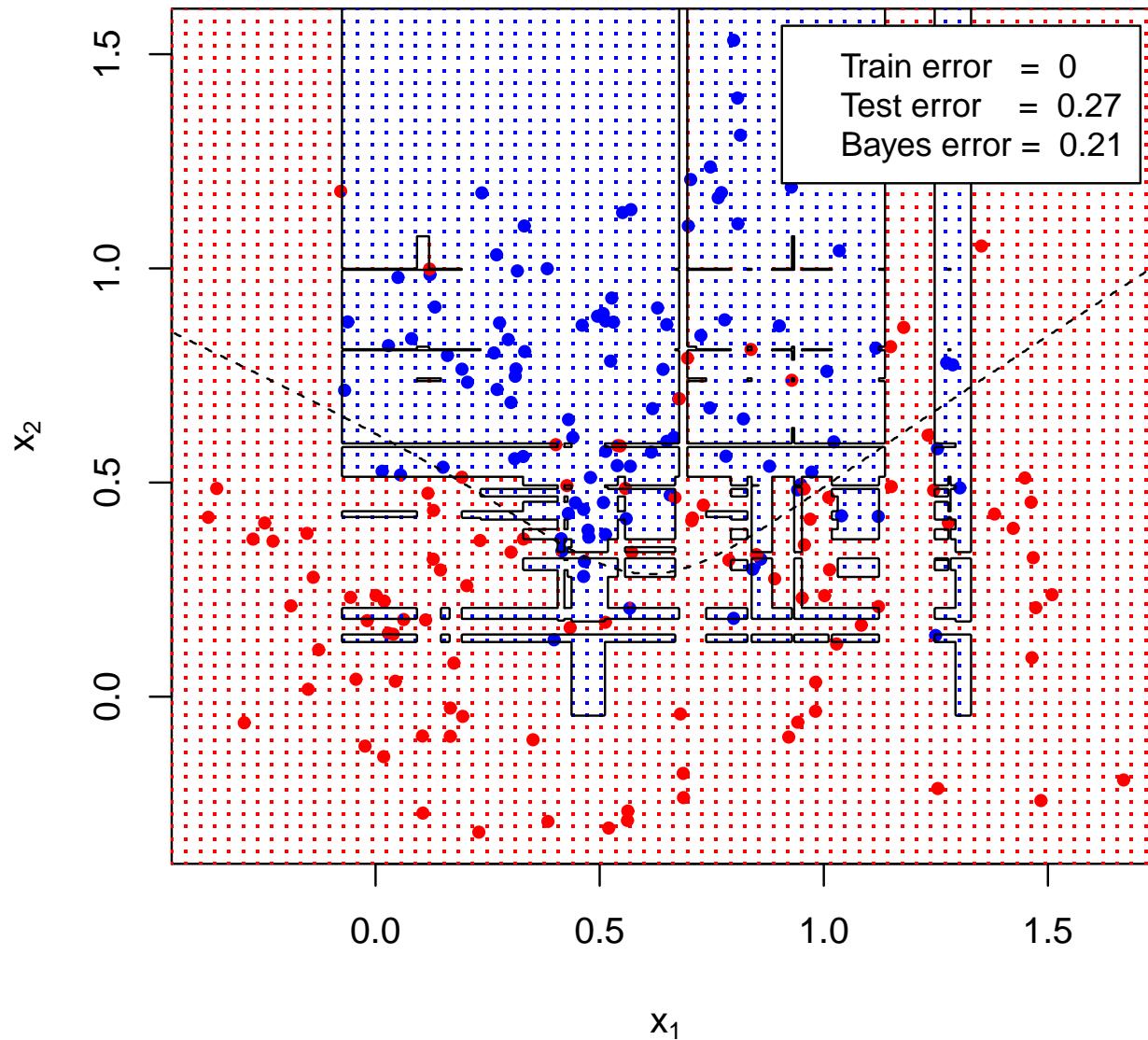
**return**  $f = \operatorname{sign}(h_M)$

$0 < \gamma \leq 1 - \text{shrinkage}$

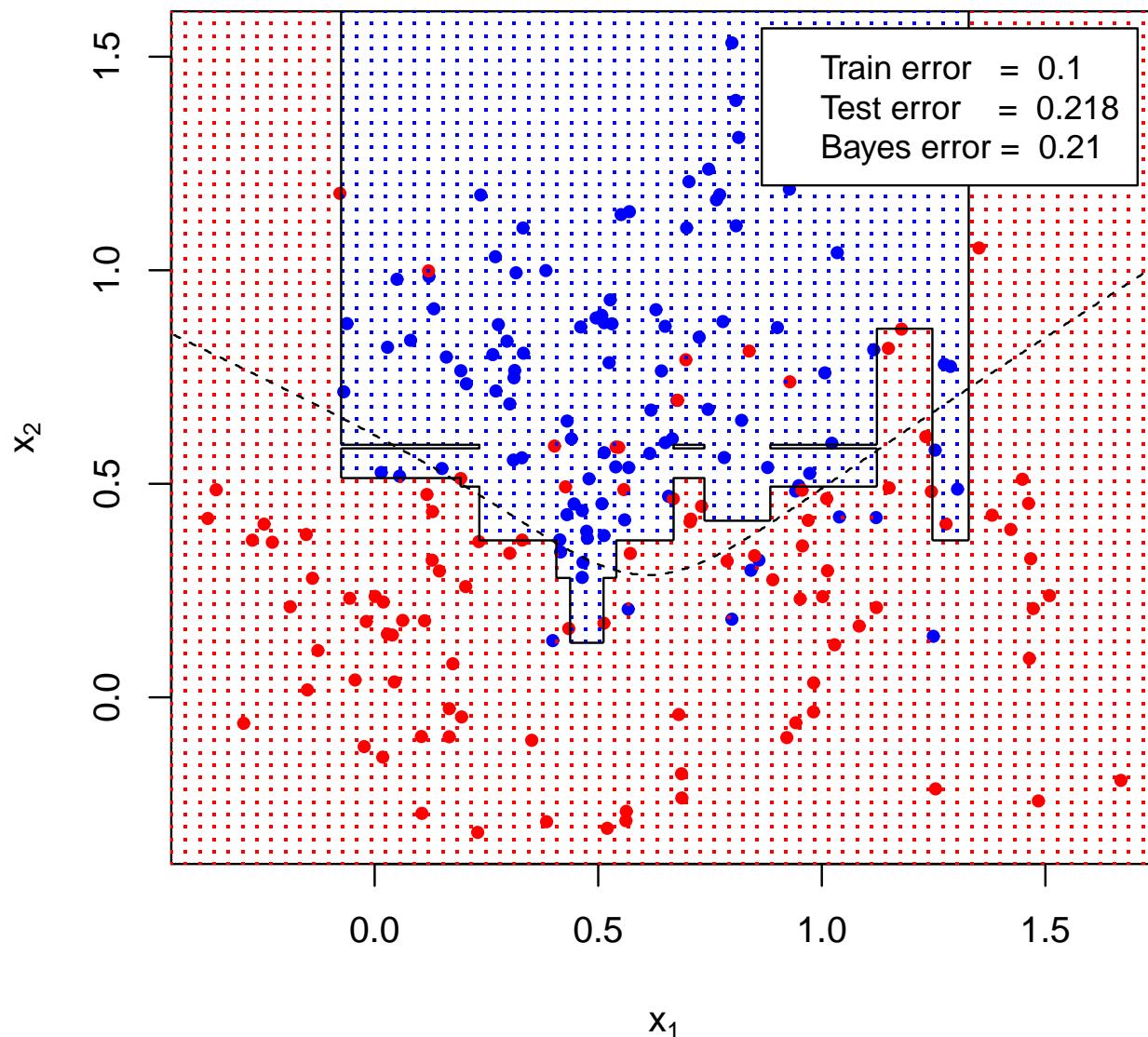
$M = 100$ , штрафная функция — кросс-энтропия, stumps



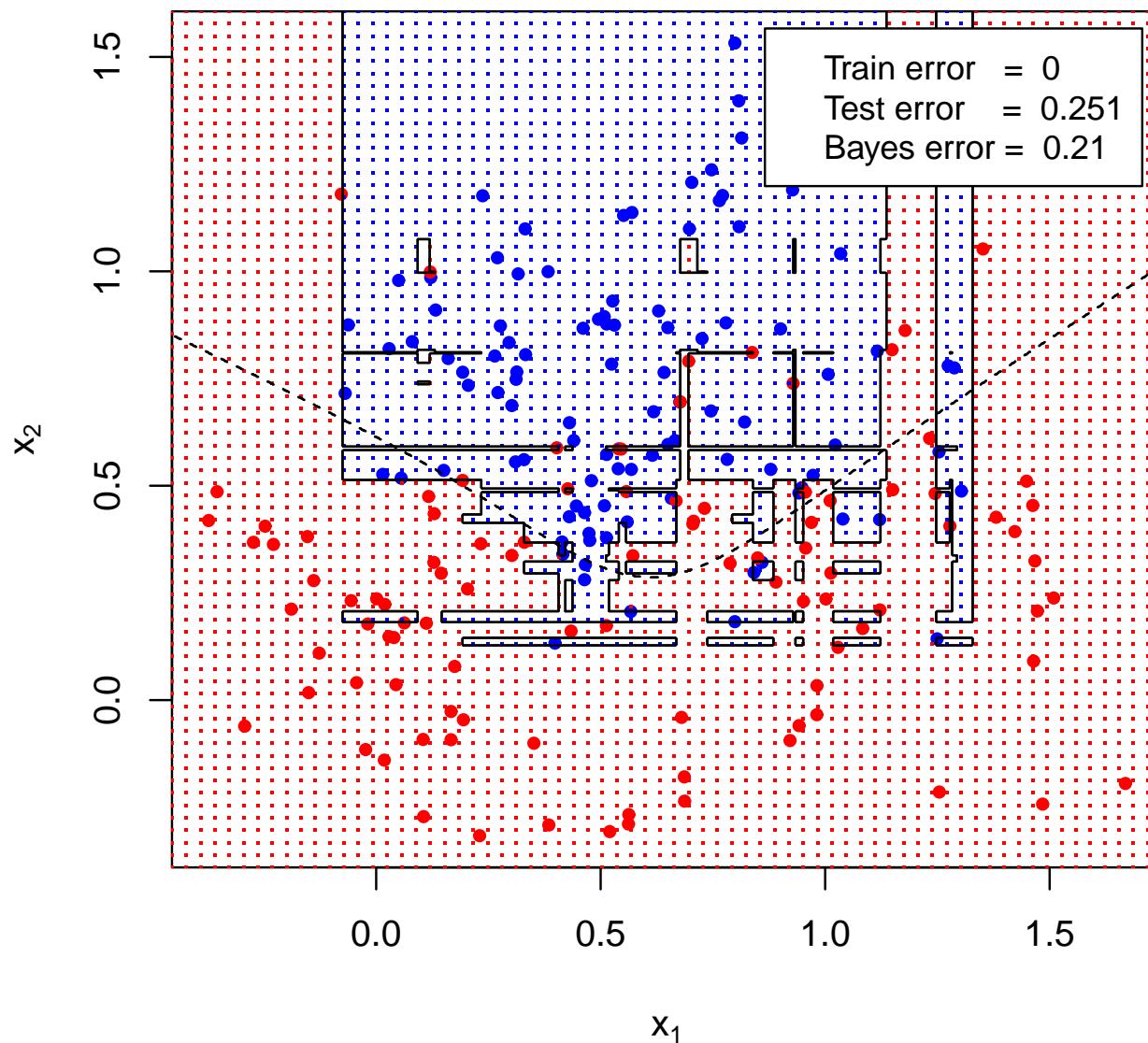
$M = 1000$



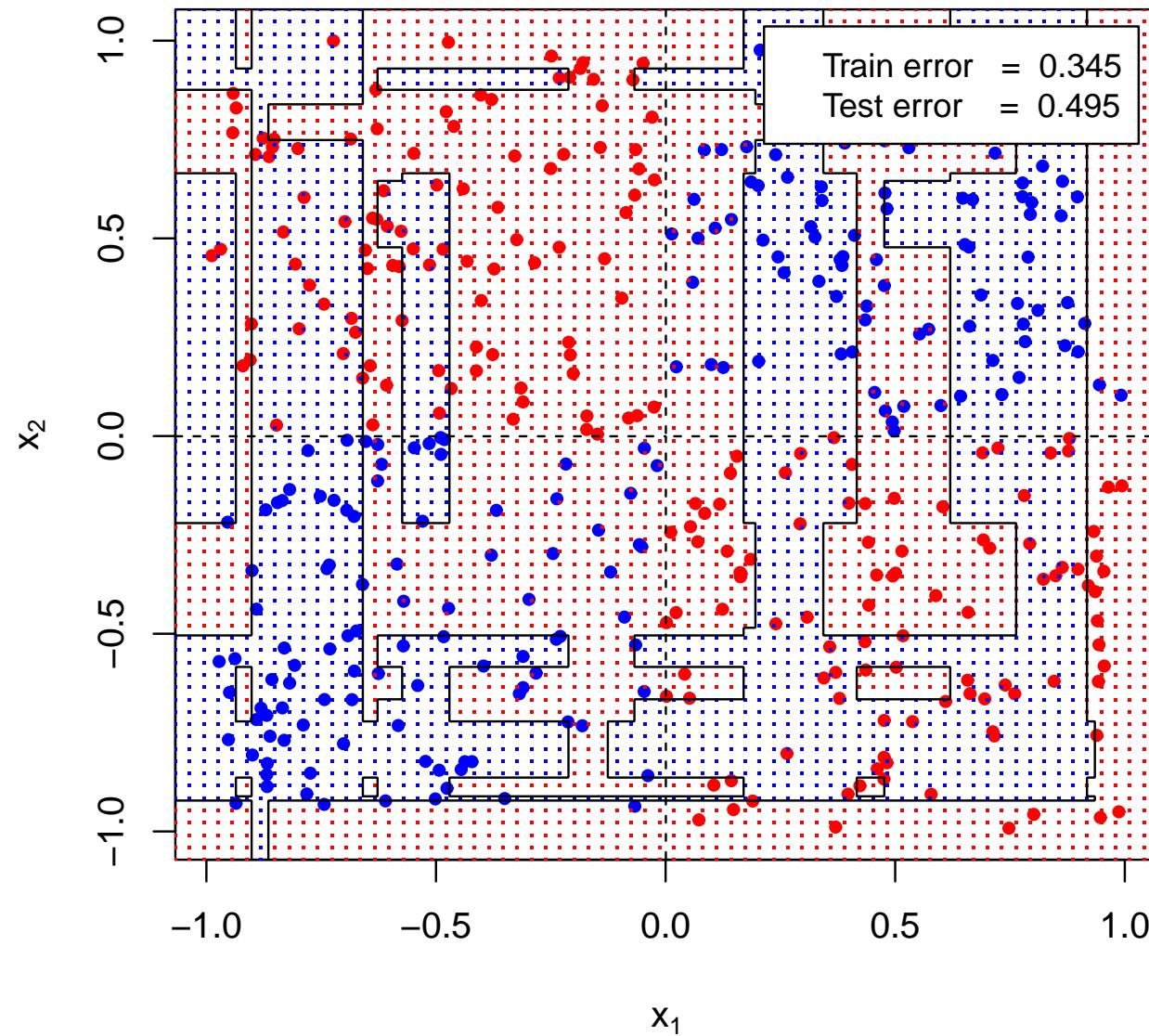
$M = 100$ , shrinkage = 0.5



$M = 1000$ , shrinkage = 0.5

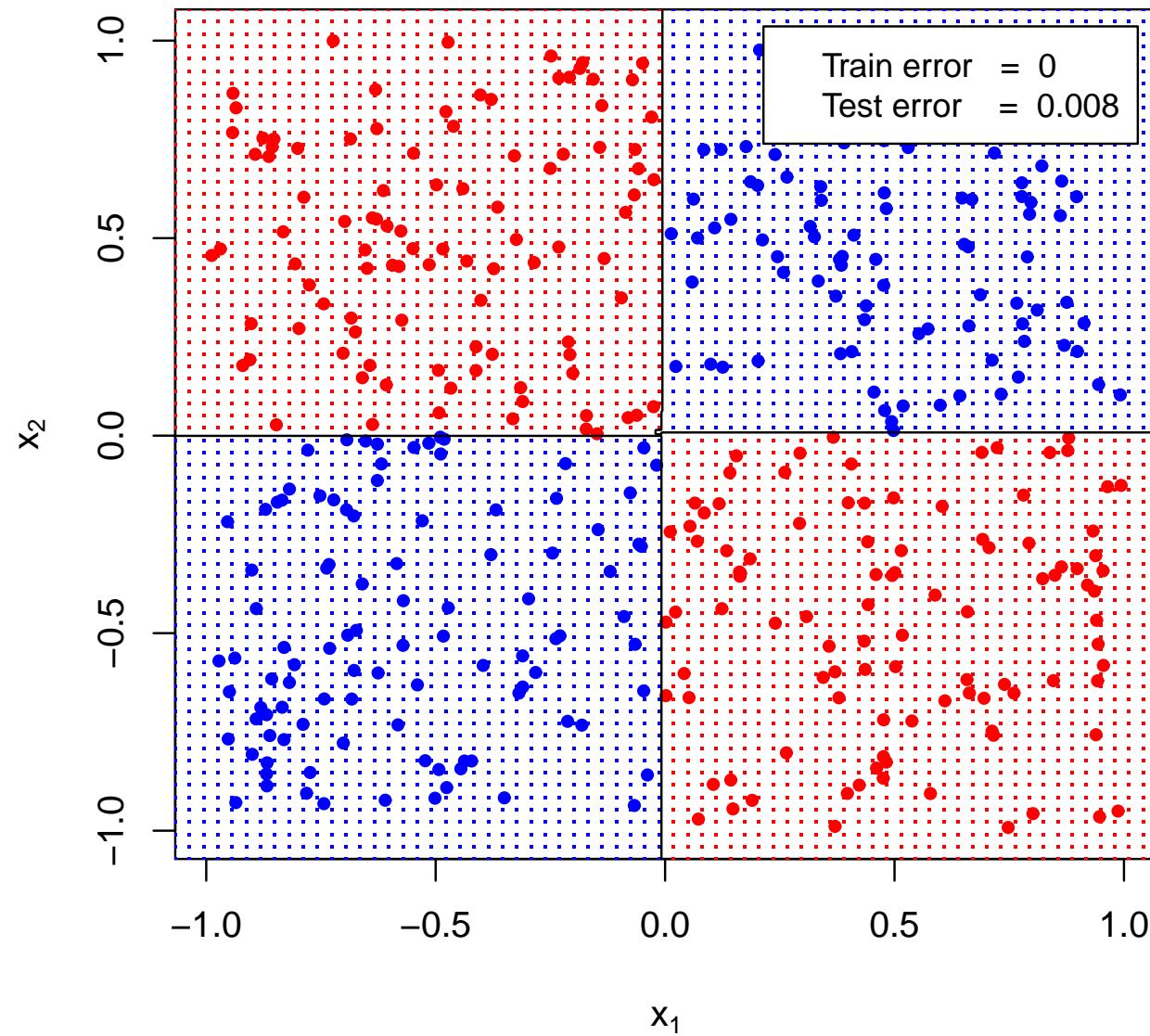


50 деревьев решений высоты 1, no shrinkage

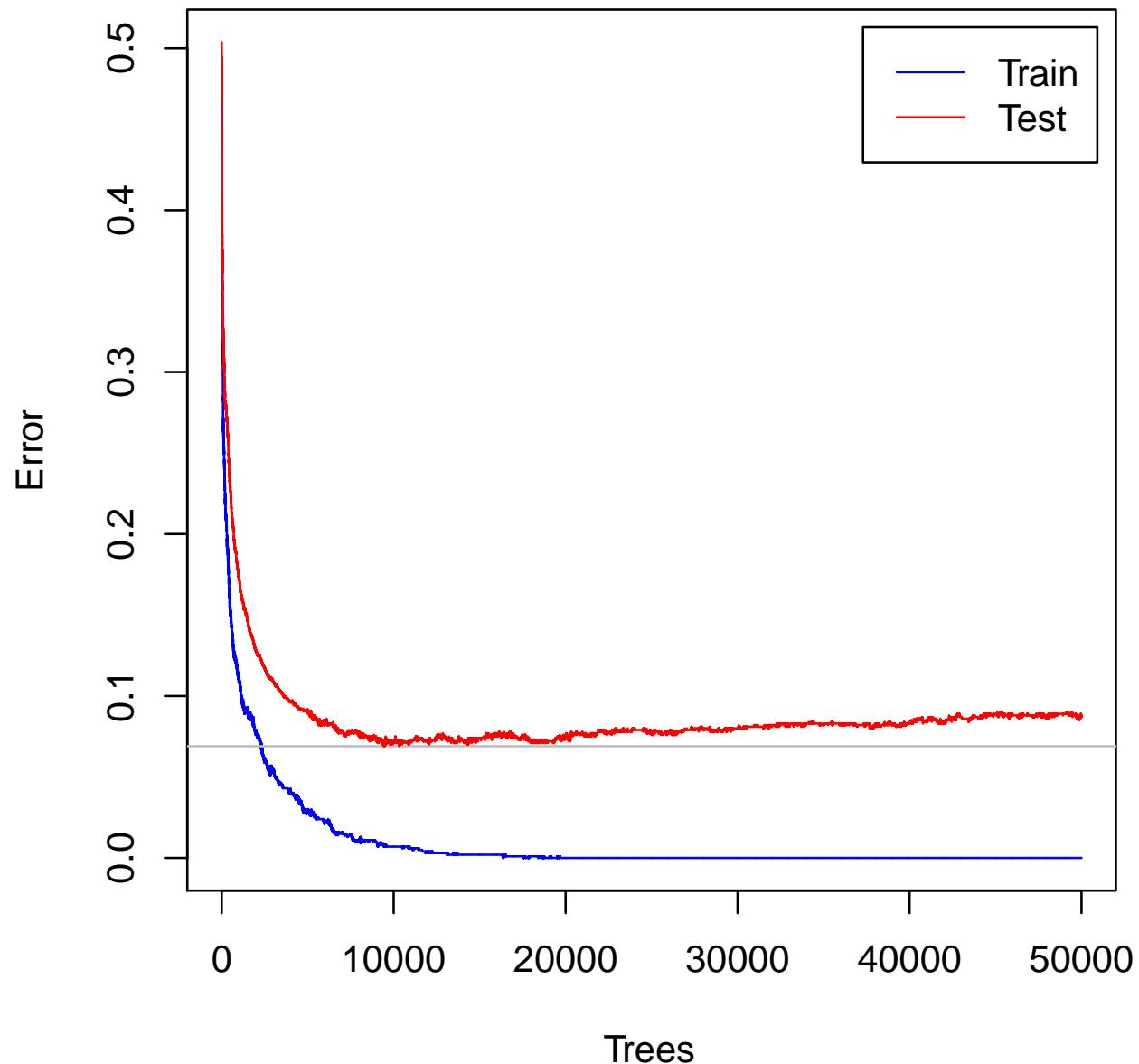


Для 1000 деревьев высоты 1 картина (и ошибки) примерно такая же

## 50 деревья решений высоты 2, no shrinkage

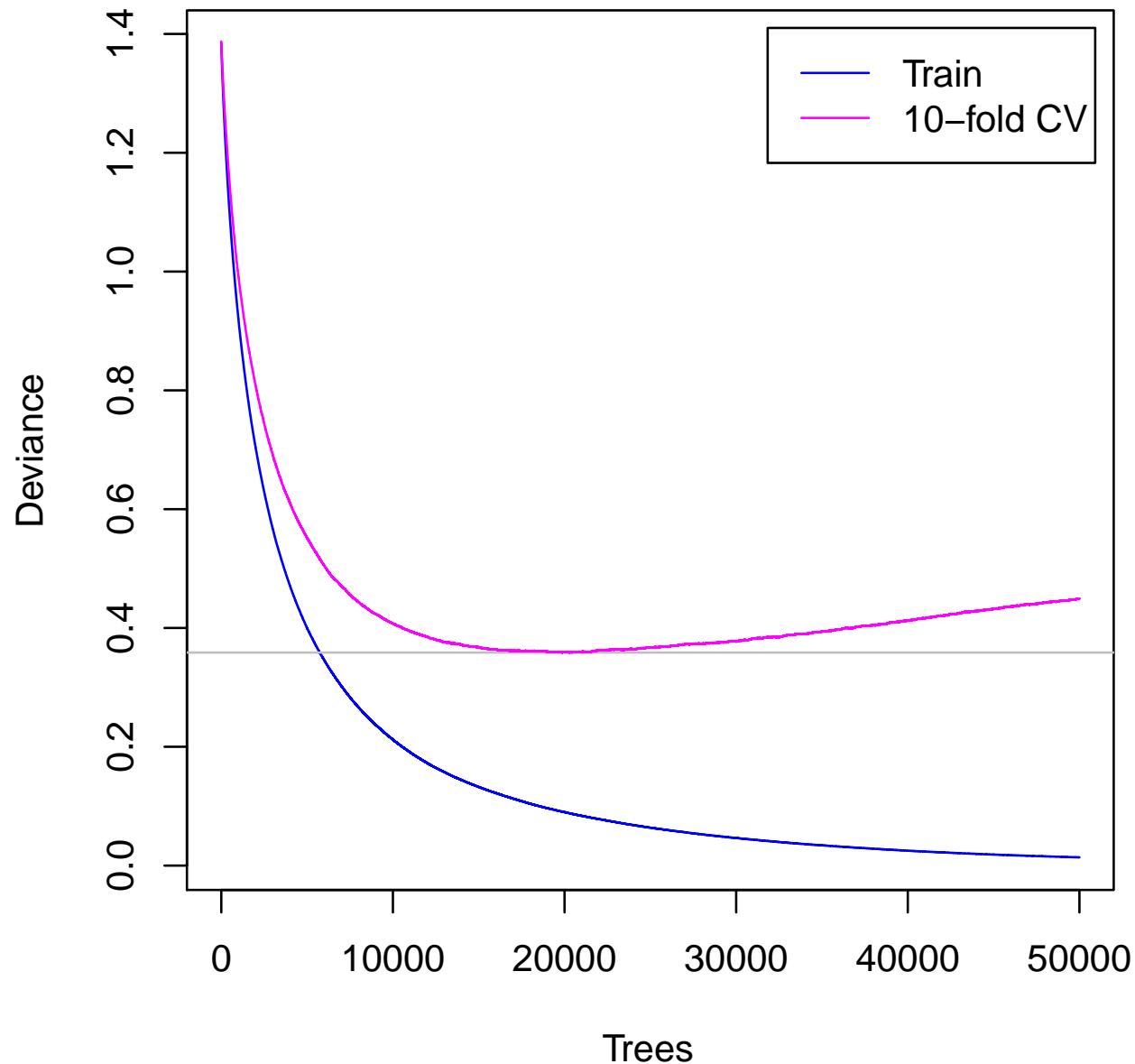


Гиперсфера. Штрафная функция — кросс-энтропия. shrinkage = 0.01, stumps



Минимум (0.069) достигается на 9447 деревьях

Как меняется  $h(x)$ ?



Минимум (0.359) достигается на 19864 деревьях

## 8.6. Эксперименты

Данные: UCI Machine Learning Repository <http://archive.ics.uci.edu/ml/>

Software: OpenCV Library <http://opencv.willowgarage.com>

(Реализация GBT – П.Н.Дружков, ННГУ)

Эксперимент – П.Н. Дружков

Задачи классификации

10-CV ошибка

Задача	N	d (колич.+ном.)	K	GBT	DTree	RF	ExtRF	SVM
Agaricus lepiota	8124	22 (0 + 22)	2	0	0.00	0	0	0
Liver disorders	345	6 (6 + 0)	2	0.25	0.31	0.22	0.25	0.28
Car evaluation	1728	6 (0 + 6)	4	0	0.051	0.036	0.039	0.050

GBT – Gradient Boosting Trees – градиентный бустинг деревьев решений,

DTree – Decision Tree – деревья решений,

RF – Random Forests – случайные леса,

ExtRF – Extremely Random Forests – экстремально случайные леса,

SVM – Support Vector Machine – машина опорных векторов

## Задачи восстановления регрессии. Средняя абсолютная 10-CV ошибка

<i>Задача</i>	<i>N</i>	<i>d</i> (колич.+ном.)	GBT	DTree	RF	ExtRF	SVM
Auto-mpg	398	7 (4 + 3)	2.00	2.24	1.88	2.15	2.98
Computer hardware	209	8 (7 + 1)	12.61	15.62	11.62	9.63	37.00
Concrete slump	103	9 (9 + 0)	2.26	2.92	2.60	2.36	1.77
Forestfires	517	12 (10 + 2)	18.74	17.26	17.79	16.64	12.90
Boston housing	506	13 (13 + 0)	2.03	2.60	2.13	2.20	4.05
Import-85	201	25 (14 + 11)	1305	1649	1290	1487	1787
Servo	167	4 (0 + 4)	0.238	0.258	0.247	0.420	0.655
Abalone	4177	8 (7 + 1)	1.470	1.603	1.492	1.498	2.091



*Глава 9*

## **Обучение без учителя**

## План

- Матрица расстояний
- Многомерное шкалирование (масштабирование)
- Алгоритмы кластеризации:
  - Методы, основанные на теории графов
  - Методы центров тяжести и медоидов
  - FOREL
- Иерархическая кластеризация:
  - Агломеративные методы
  - Разделяющие методы
- Ассоциативные правила и алгоритм Apriori
- Google PageRank

## Обучение с учителем

$X$  —  $d$ -мерная случайная величина,

$Y$  — одномерная случайная величина,

$P(X, Y)$  — совместная функция распределения.

Имеется набор  $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(N)}, y^{(N)})$  (*обучающая выборка*) из  $N$  независимых реализаций случайной величины  $(X, Y)$ .

Задачу обучения с учителем можно рассматривать как исследование функции распределения  $P(Y | X)$ .

## Обучение без учителя

$X$  —  $d$ -мерная случайная величина.

Имеется  $N$  объектов  $x^{(1)}, x^{(2)}, \dots, x^{(N)}$  из  $N$  независимых реализаций случайной величины  $X$ .

Задачу обучения без учителя можно рассматривать как исследование функции распределения  $P(X)$ .

## 9.1. Матрица различий

Многие алгоритмы кластеризации на вход принимают *матрицу различий (dissimilarity matrix)*, или *матрицу расстояний*  $D = (\delta_{ii'})$ , размера  $N \times N$ .

$\delta_{ii'}$  равен «расстоянию» от  $i$ -го объекта до  $i'$ -го.

При этом под «расстоянием» не обязательно понимается евклидово или какое-либо другое расстояние, удовлетворяющее аксиомам метрического пространства.

Например, часто не предполагается, что выполнено неравенство треугольника  $\delta_{ii'} \leq \delta_{ik} + \delta_{ki'}$ .

Однако, как правило,  $\delta_{ii} = 0$ ,  $\delta_{ii'} = \delta_{i'i} \geq 0$  ( $i = 1, 2, \dots, N$ ,  $i' = 1, 2, \dots, N$ ).

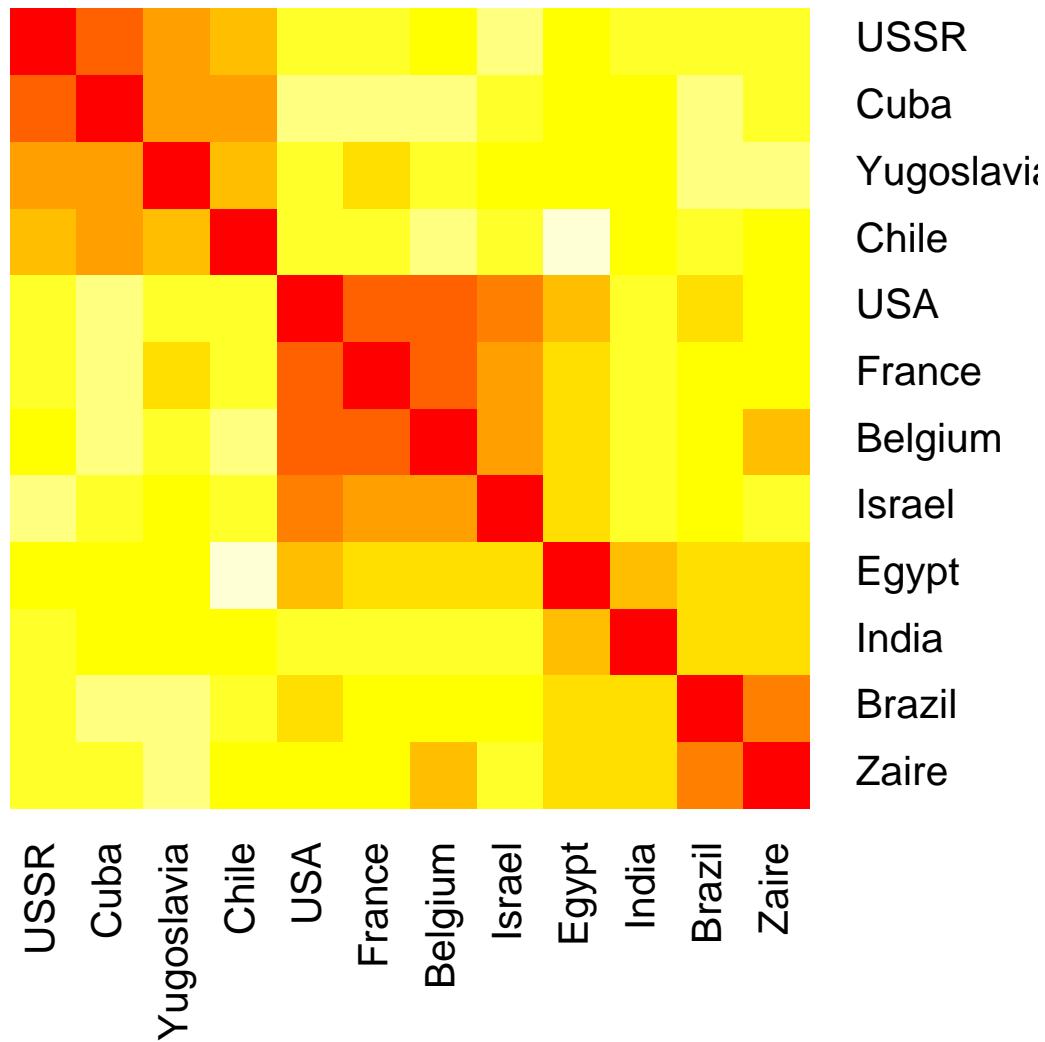
Явные признаковые описания объектов могут вообще отсутствовать.

Пример: матрица различий в политической и экономической системах 12 стран мира (на основании опроса студентов, изучающих политические науки, 1980-е гг., StatLib).

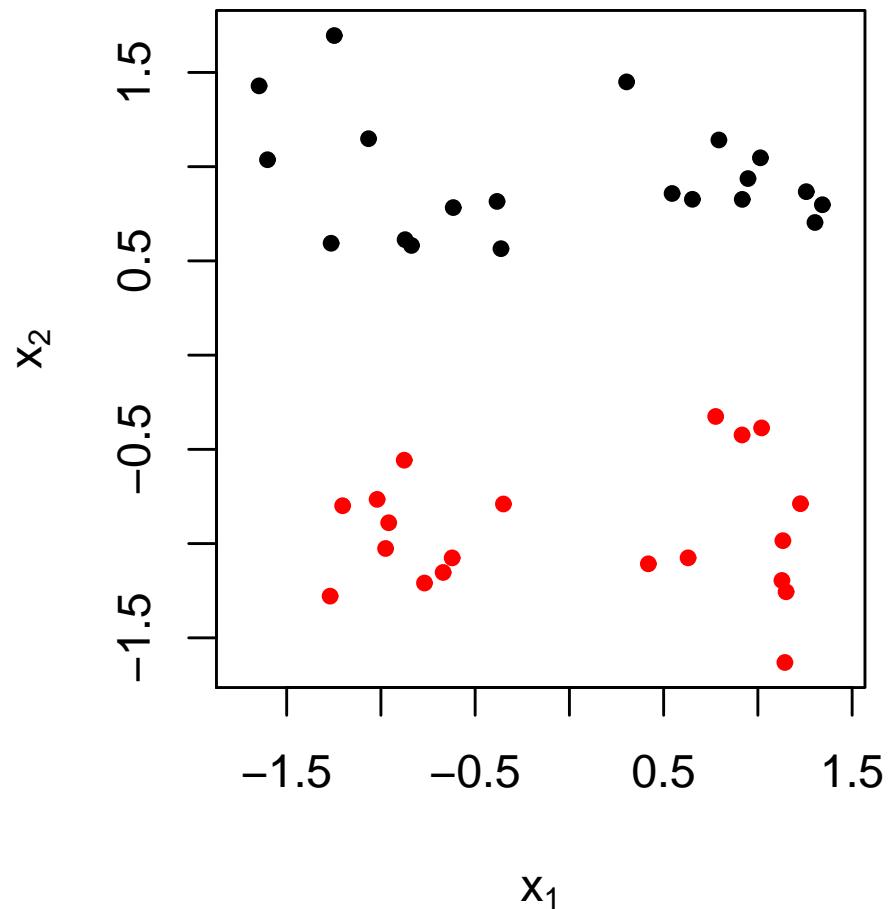
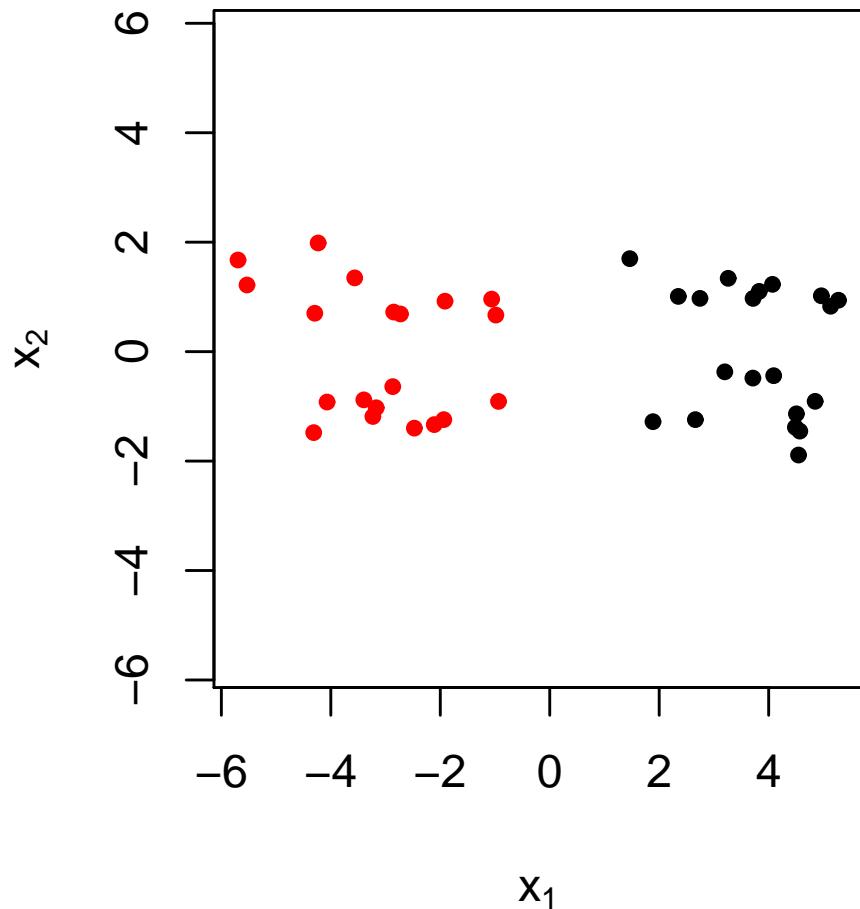
Различия в баллах (от 0 до 10):

Можно разглядеть 3 кластера (строки и столбцы переставлены).

Белый и желтый цвет — большие различия. Красный — малые.



Пусть признаковые описания  $x_j^{(i)}$  ( $i = 1, 2, \dots, N$ ,  $j = 1, 2, \dots, d$ ) известны.  
Как строить матрицу различий?



Разные подходы к построению матрицы различий.

$$\delta(x, x') = \sum_{j=1}^d w_j \delta_j(x_j, x'_j), \quad \sum_{j=1}^d w_j = 1, \quad w_j \geq 0,$$

где  $w_j$  — весовые константы, а  $\delta_j(x, x')$  — функция, характеризующая *различие в j-й координате*.

Для количественных, бинарных и порядковых признаков можно взять, например,

$$\delta_j(x, x') = (x_j - x'_j)^2.$$

Пусть качественная переменная  $j$  может принимать  $m$  значений  $s_{j1}, s_{j2}, \dots, s_{jm}$ .

Функцию  $\delta_j(x_j, x'_j) = \delta_j(s_{jr}, s_{jr'})$  определяем явным перечислением набора значений  $L_{rr'} = \delta_j(s_r, s_{r'})$ .

Матрица  $L = (L_{rr'})$  должна удовлетворять свойствам:

$$L_{rr'} = L_{r'r}, \quad L_{rr} = 0, \quad (r = 1, 2, \dots, m, \quad r' = 1, 2, \dots, m).$$

Другой подход для измерения различия между объектами (если нет качественных переменных) заключается в использовании корреляции:

$$\text{Corr}(x, x') = \frac{\sum_{j=1}^d (x_j - \bar{x})(x'_j - \bar{x}')}{\sqrt{\sum_{j=1}^d (x_j - \bar{x})^2 \sum_{j=1}^d (x'_j - \bar{x}')^2}}, \quad \text{где } \bar{x} = \frac{1}{d} \sum_{j=1}^d x_j, \quad \bar{x}' = \frac{1}{d} \sum_{j=1}^d x'_j.$$

Измеряется корреляция между объектами, а не между переменными!

Здесь наоборот: чем  $\text{Corr}(x, x')$  больше, тем различие между объектами  $x, x'$  меньше.  
Если вход стандартизован, то

$$\sum_{j=1}^d (x_j - x'_j)^2 = \langle x - x', x - x' \rangle = \langle x, x \rangle - 2\langle x, x' \rangle + \langle x', x' \rangle = 2(1 - \text{Corr}(x, x')).$$

Таким образом, кластеризация на основе евклидова расстояния (расстояние является мерой различия между парой объектов) эквивалентна кластеризации на основе корреляции (корреляция является мерой сходства между парой объектов).

Как выбирать веса  $w_j$  ( $j = 1, 2, \dots, d$ )?

Выбор  $w_j = 1/d$  не обязательно приводит к тому, что влияние различия в каждой переменной будет одинаковым.

Величину  $w_j \delta_j(x_j, x'_j)$  имеет смысл соотнести со средним значением  $\bar{\delta}$  по всем расстояниям  $\delta(x^{(i)}, x^{(i')})$  между каждой парой объектов  $x^{(i)}, x^{(i')}$  ( $i, i' = 1, 2, \dots, N$ ):

$$\bar{\delta} = \frac{1}{N^2} \sum_{i=1}^N \sum_{i'=1}^N \delta(x^{(i)}, x^{(i')}) = \frac{1}{N^2} \sum_{i=1}^N \sum_{i'=1}^N \sum_{j=1}^d w_j \delta_j(x_j^{(i)}, x_j^{(i')}) = \sum_{j=1}^d w_j \bar{\delta}_j.$$

Здесь

$$\bar{\delta}_j = \frac{1}{N^2} \sum_{i=1}^N \sum_{i'=1}^N \delta_j(x_j^{(i)}, x_j^{(i')})$$

есть среднее различие в  $j$ -м признаке.

Таким образом, относительное влияние  $w_j \delta_j(x_j, x'_j)$  равно  $w_j \bar{\delta}_j$ .

И если  $w_j = 1/\bar{\delta}_j$ , то эти влияния будут одинаковыми в каждой координате.

Заметим, что если все переменные количественные и в качестве меры различия выбрана взвешенная сумма квадратов:

$$\delta(x, x') = \sum_{j=1}^d w_j (x_j - x'_j)^2,$$

то

$$\bar{\delta}_j = \frac{1}{N^2} \sum_{i=1}^N \sum_{i'=1}^N \left( x_j^{(i)} - x_j^{(i')} \right)^2 = 2 \text{var}_j,$$

где  $\text{var}_j$  — смещенная выборочная дисперсия для  $j$ -й координаты:

$$\text{var}_j = \frac{1}{N} \sum_{i=1}^N \left( x_j^{(i)} \right)^2 - \frac{1}{N^2} \left( \sum_{i=1}^N x_j^{(i)} \right)^2.$$

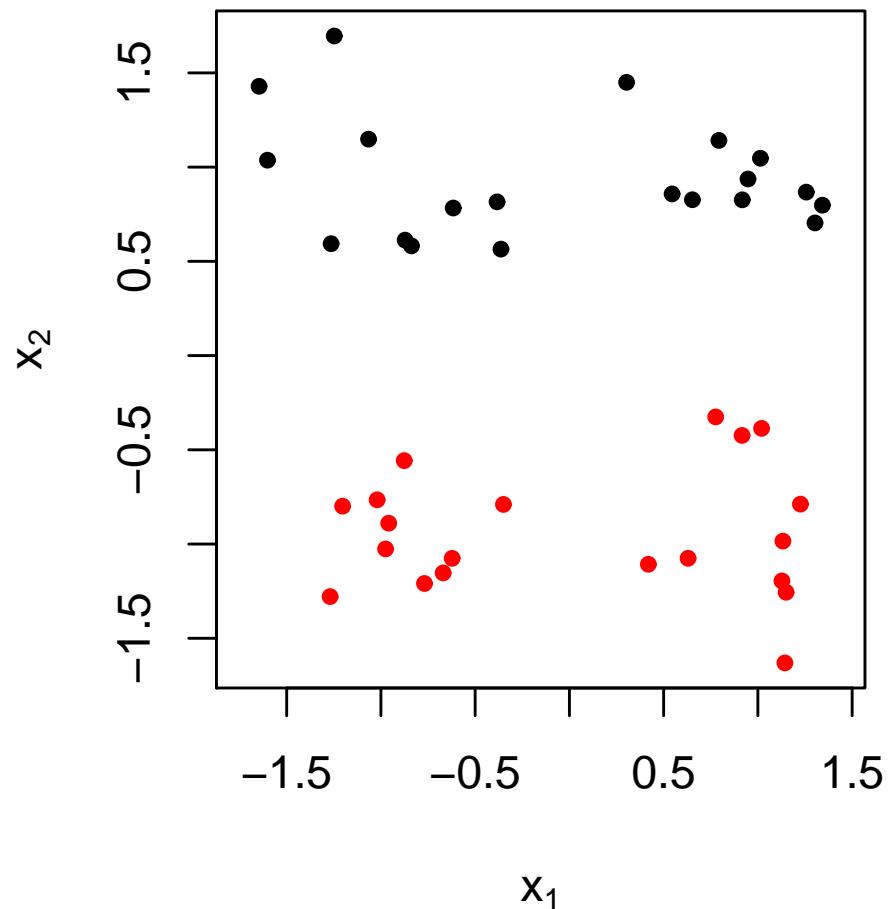
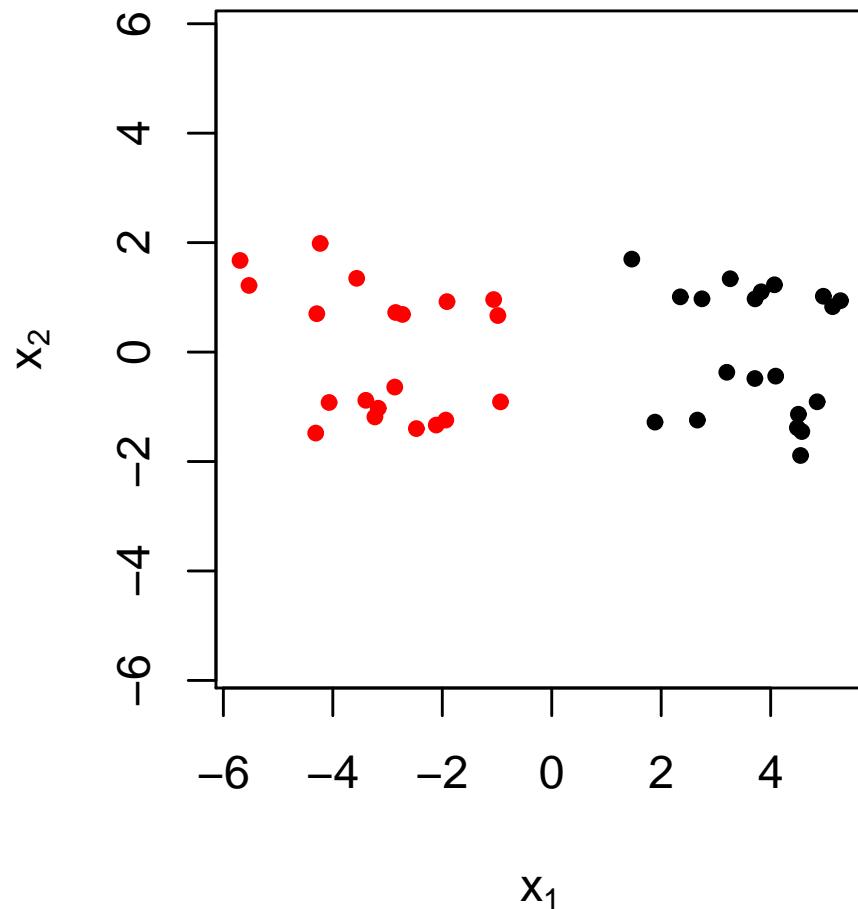
**Упражнение 9.1** Проверить, что  $\bar{\delta}_j = 2 \text{var}_j$ .

От выбора коэффициентов  $w_j$  сильно зависит ответ.

Разбиение на классы множества точек на плоскости, полученные методом медоидов.

На рисунке слева использовались равномерные веса.

На рисунке справа перед применением алгоритма данные были приведены к стандартному виду, что эквивалентно использованию весов  $w_j = 1/(2 \text{ var}_j)$ .



## 9.2. Многомерное шкалирование (масштабирование)

Пусть  $D = (\delta_{ii'})$  — матрица различий между каждой парой объектов  $x^{(1)}, x^{(2)}, \dots, x^{(N)} \in \mathbf{R}^d$ . Предполагается, что

$$\delta_{ii'} = \delta_{i'i} \geq 0, \quad \delta_i = 0 \quad (i, i' = 1, 2, \dots, N).$$

В задаче многомерного шкалирования (*multidimensional scaling*) требуется найти такие  $z^{(1)}, z^{(2)}, \dots, z^{(N)} \in \mathbf{R}^k$ , что

$$\|z^{(i)} - z^{(i')}\| \approx \delta_{ii'} \quad (i, i' = 1, 2, \dots, N).$$

Компоненты векторов  $z^{(i)}$  называются *MDS-координатами*.

Понятно, что если в качестве меры различия берется евклидово расстояние и  $k = d$ , то задача имеет *точное* тривиальное решение: в качестве  $z^{(i)}$  надо взять  $x^{(i)}$  ( $i = 1, 2, \dots, N$ ).

На самом деле, задача интересна для небольших  $k$ , например,  $k = 2$ .

В случае  $k = 2$  точки  $z^{(i)}$  можно визуализировать и тем самым попробовать решить задачу кластеризации визуально.

Подходы к решению:

*Шкалирование Краскала–Шефарда (Kruskal–Shephard scaling)* — минимизация стресс-функции

$$S(z^{(1)}, z^{(2)}, \dots, z^{(N)}) = \sum_{i \neq i'} \left( \delta_{ii'} - \|z^{(i)} - z^{(i')}\| \right)^2$$

(например, методом наискорейшего спуска).

*Шкалирование Сэммона (Sammon scaling)* — минимизация функции

$$S(z^{(1)}, z^{(2)}, \dots, z^{(N)}) = \sum_{i \neq i'} \frac{\left( \delta_{ii'} - \|z^{(i)} - z^{(i')}\| \right)^2}{\delta_{ii'}}$$

— маленькие расстояния между точками учитываются сильнее, чем большие.

*Классический метрический метод.*

Вместо «различий»  $\delta_{ii'}$  будем рассматривать «сходства»  $s_{ii'}$ :

$$s_{ii'} = \left\langle x^{(i)} - \bar{x}, x^{(i')} - \bar{x} \right\rangle, \quad \text{где} \quad \bar{x} = \frac{1}{N} \sum_{i=1}^N x^{(i)}.$$

Рассмотрим матрицу  $\mathbf{S} = (s_{ii'})$ .

Будем минимизировать функцию

$$\sum_{i \neq i'} (s_{ii'} - \left\langle z^{(i)} - \bar{z}, z^{(i')} - \bar{z} \right\rangle)^2.$$

Эта задача минимизации имеет явное решение в терминах собственных векторов матрицы  $\mathbf{S}$ .

Пусть  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_k$  суть  $k$  наибольших собственных чисел матрицы  $\mathbf{S}$ , а  $e_1, e_2, \dots, e_k$  — соответствующие собственные векторы.

$\mathbf{E}$  — матрица, составленная из столбцов  $e_1, e_2, \dots, e_k$ ,

$\mathbf{D}$  — диагональная матрица с числами  $\sqrt{\lambda_1}, \sqrt{\lambda_2}, \dots, \sqrt{\lambda_k}$  на диагонали.

Тогда в качестве решений  $z_i$  ( $i = 1, 2, \dots, k$ ) нужно взять строки матрицы  $\mathbf{ED}$ .

Если известна матрица  $S$ , по ней можно определить  $D$ , причем единственным способом.

Обратный переход неоднозначен.

Например,  $S$  можно вычислить по формуле

$$S = -\frac{1}{2} \left( T - \frac{T\mathbf{1}}{N} - \frac{\mathbf{1}T}{N} + \frac{1}{N^2} \sum_{i=1}^N \sum_{i'=1}^N t_{ii'} \right),$$

где  $\mathbf{1}$  — матрица порядка  $N$ , заполненная единицами, а  $T = (t_{ii'})$  — матрица порядка  $N$ , в которой  $t_{ii'} = \delta_{ii'}^2$ .

Классический метрический метод эквивалентен РСА.

*Неметрический метод Шефарда–Краскала* основан на минимизации стресс-функции

$$\frac{\sum_{ii'} \left( \theta(\|z^{(i)} - z^{(i')}\|) - \delta_{ii'} \right)^2}{\sum_{ii'} \delta_{ii'}^2}$$

на множестве всех  $\delta_{ii'}$  и всех монотонно возрастающих функций  $\theta$ .

Если  $\theta$  зафиксировано, то  $\delta_{ii'}$  можно найти, например, методом наискорейшего спуска.

Если зафиксированы  $\delta_{ii'}$ , то для поиска  $\theta$  можно воспользоваться методом *изотонной регрессии*: функция ищется в классе монотонных ступенчатых функций методом наименьших квадратов.

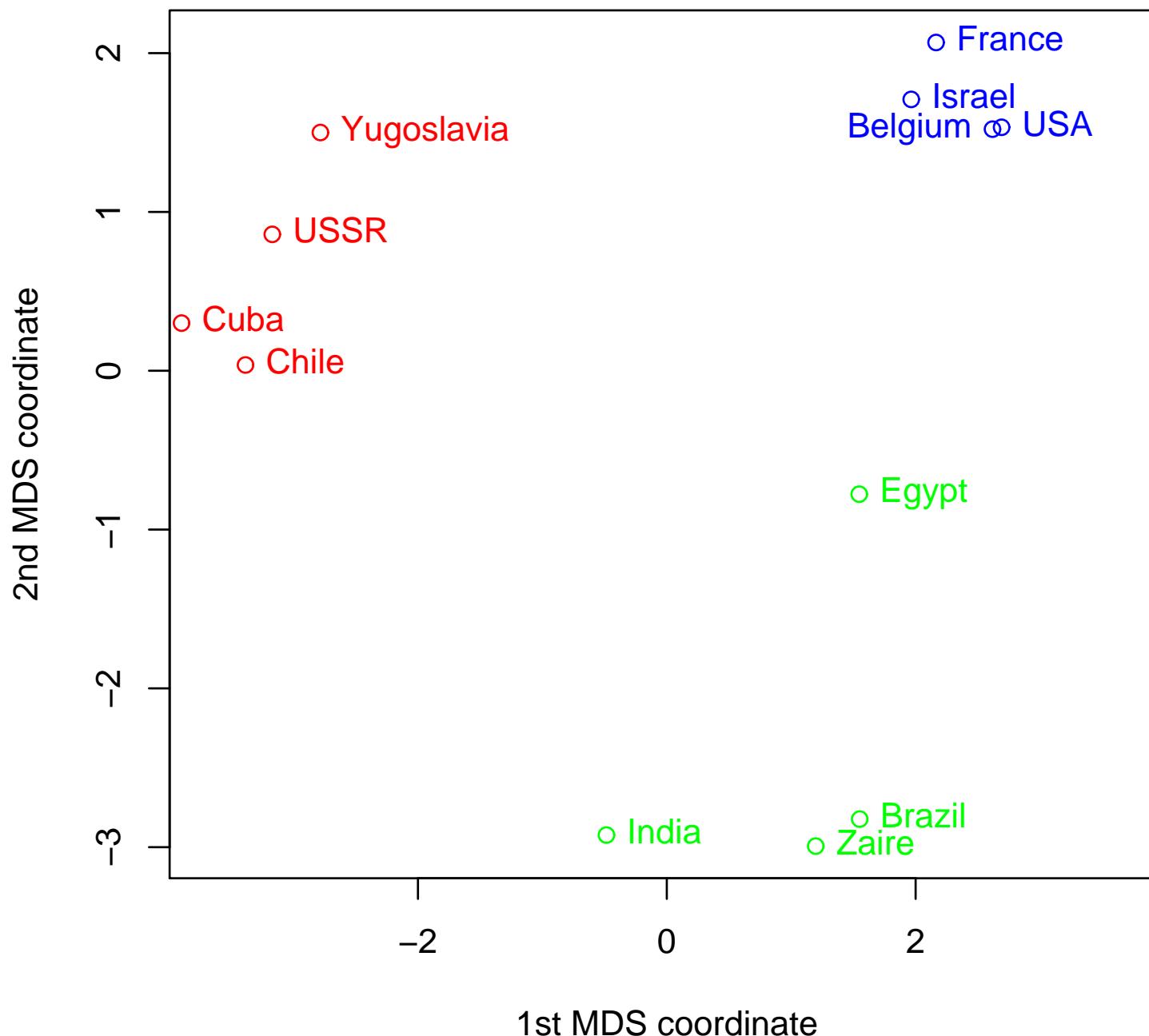
Итерации заканчиваются, когда процесс стабилизируется.

Результаты применения классического метрического метода и неметрического метода Шефарда–Краскала к данным из примера `politics`.

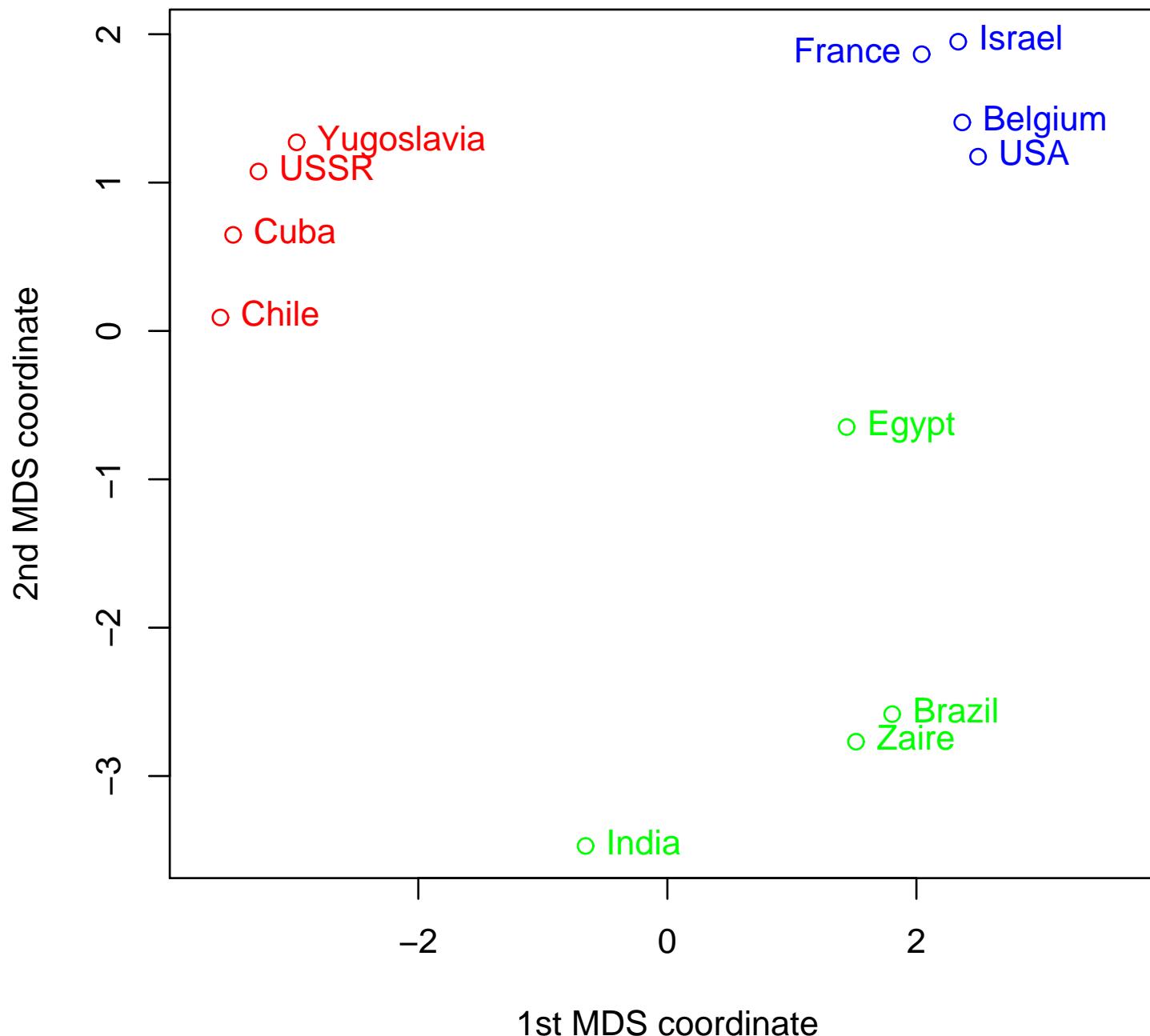
Было выбрано  $k = 2$ .

Можно выделить 3 кластера.

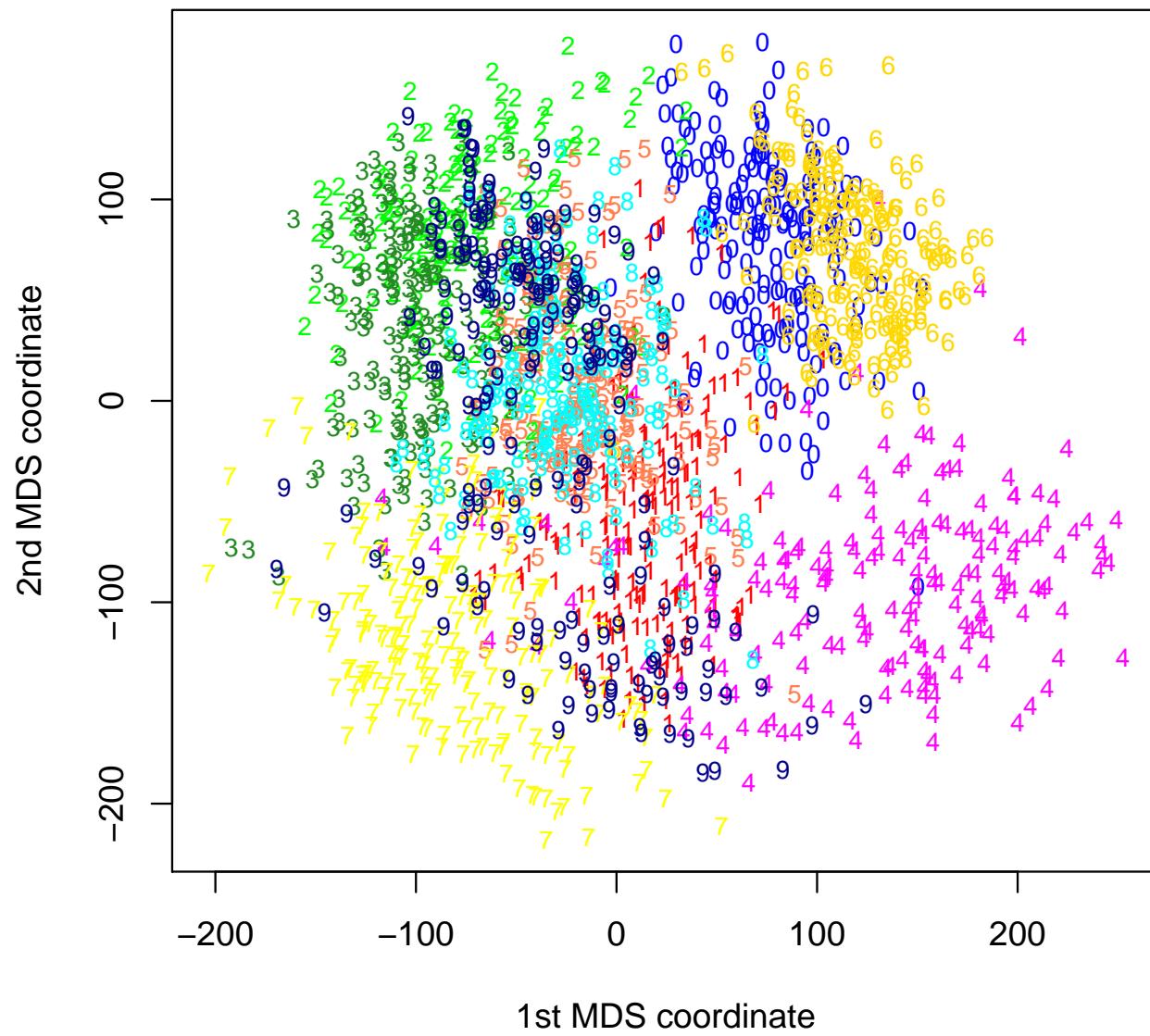
## Классический метод



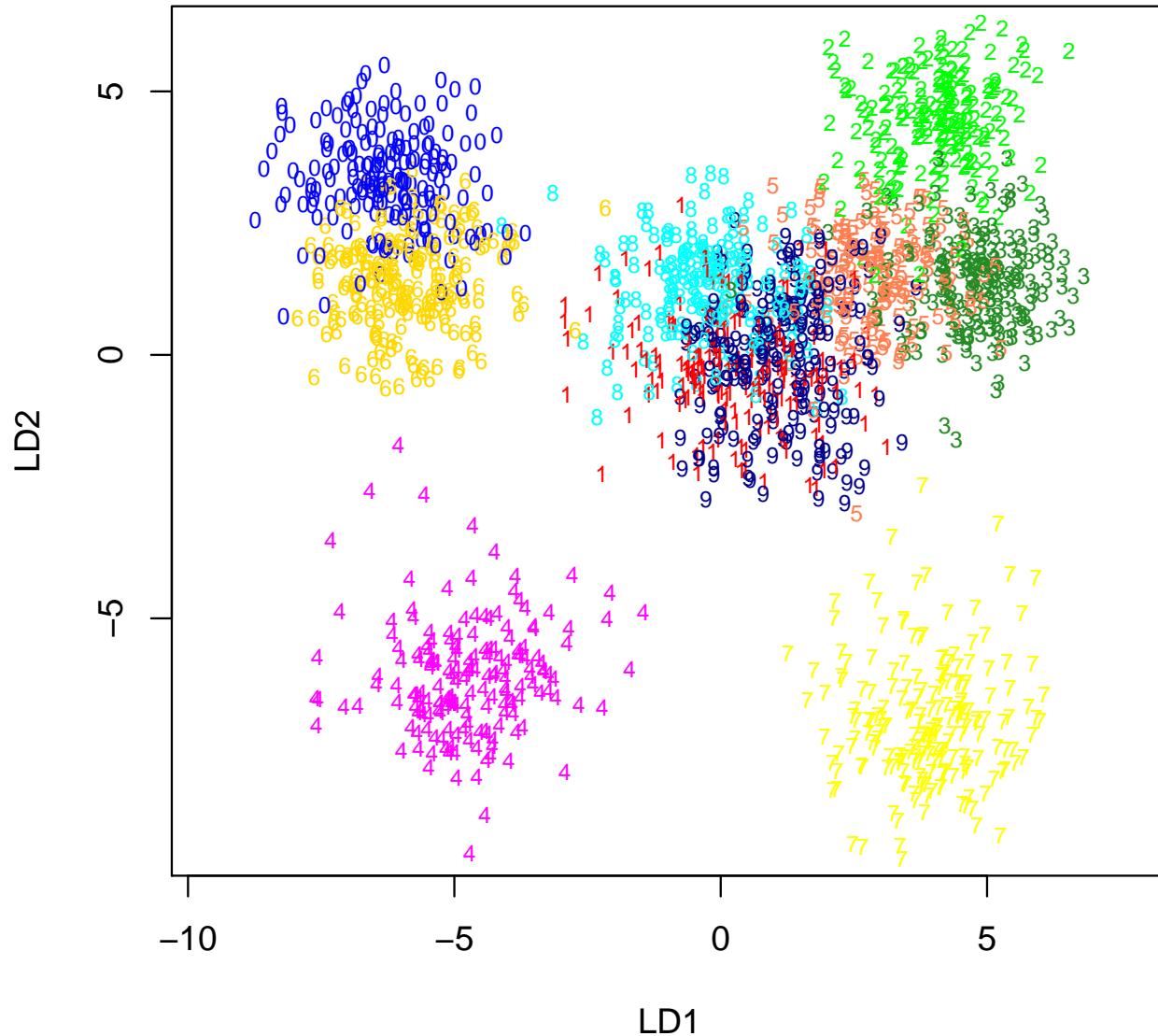
## Неметрический метод Шефарда–Краскала



Пример digits. Классический метод



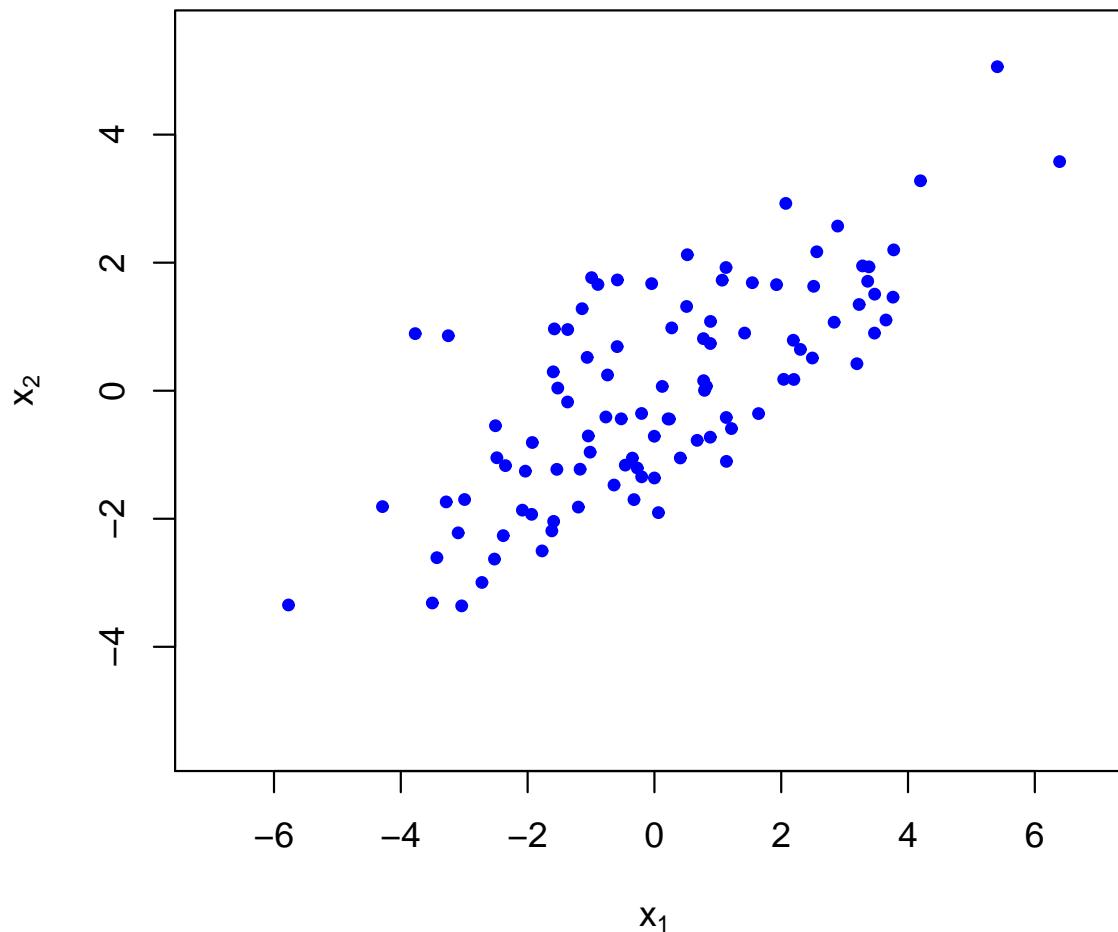
С помощью LDA тоже можно решать задачу визуализации, но там мы используем метки классов!

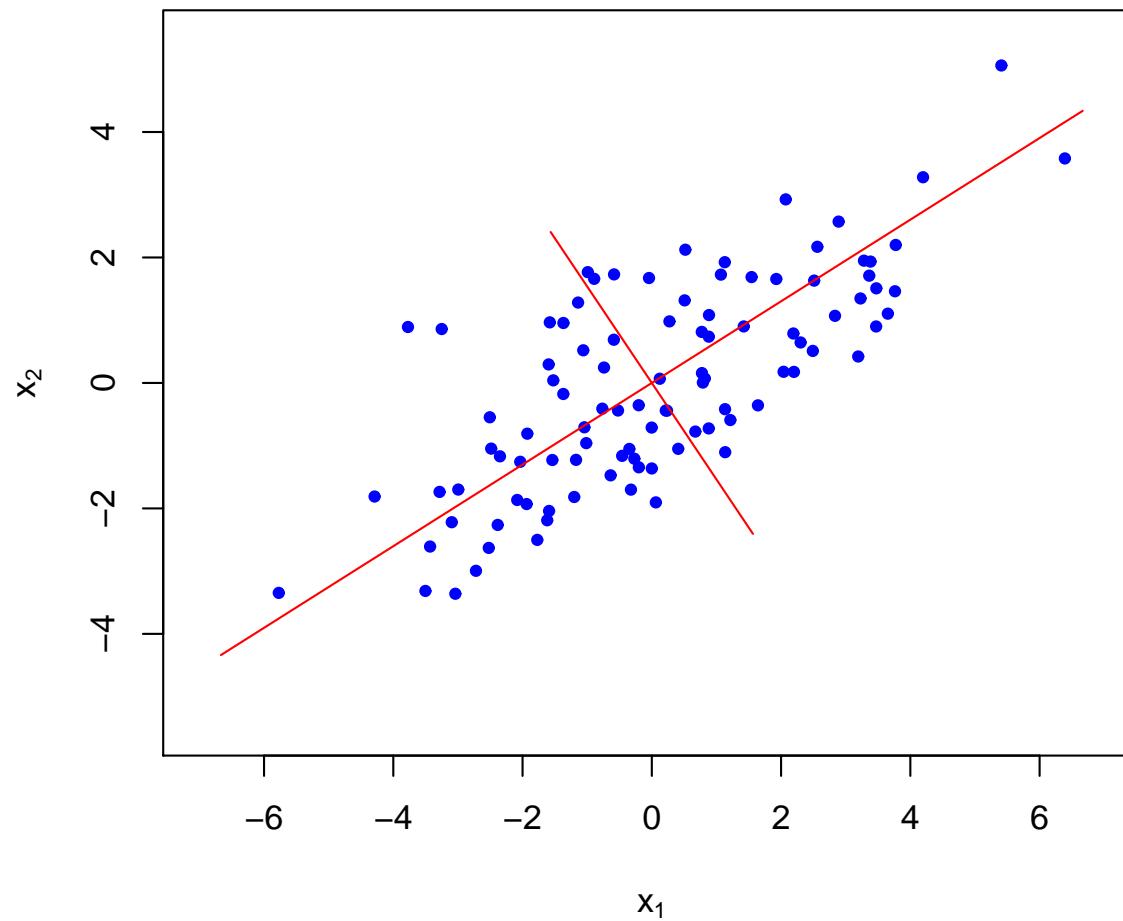


## 9.2.1. Метод главных компонент

Principal component analysis (PCA)

Неформально: пусть точки группируются «вдоль» некоторого линейного многообразия (небольшой размерности). Как его найти?





*Сингулярным, или SVD-разложением (singular value decomposition),* матрицы  $\mathbf{X}$  размера  $N \times d$  называется ее представление в виде

$$\begin{array}{cccccc} \mathbf{X} & = & \mathbf{U} & \times & \mathbf{D} & \times \mathbf{V}^{\top} \\ N \times d & & N \times d & & d \times d & d \times d \end{array}$$

где

$\mathbf{U}$  —  $N \times d$  матрица с ортонормированными столбцами ( $\mathbf{U}^{\top} \mathbf{U} = \mathbf{I}$ ),

$\mathbf{V}$  — ортогональная  $d \times d$  матрица ( $\mathbf{V}^{\top} = \mathbf{V}^{-1}$ ),

$\mathbf{D} = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_d)$  — диагональная  $d \times d$  матрица,  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_d \geq 0$ .

$\sigma_1, \sigma_2, \dots, \sigma_d$  — *сингулярные значения* (или *сингулярные числа*) матрицы  $\mathbf{X}$

Столбцы матрицы  $\mathbf{U}$  — *левые сингулярные векторы*

Столбцы матрицы  $\mathbf{V}$  — *правые сингулярные векторы*

Очевидно, что столбцы  $\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_d$  матрицы  $\mathbf{U}$  представляют собой ортонормированный базис подпространства, натянутого на столбцы  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_d$  матрицы  $\mathbf{X}$ .

Как  $\sigma_j$  зависят от данных?

Пусть, по-прежнему, данные, центрированы.

Выборочная матрица ковариаций равна  $\mathbf{S} = \mathbf{X}^\top \mathbf{X} / N$ , откуда

$$\mathbf{S} = \frac{1}{N} \mathbf{X}^\top \mathbf{X} = \frac{1}{N} \mathbf{V} \mathbf{D} \mathbf{U}^\top \mathbf{U} \mathbf{D} \mathbf{V}^\top = \mathbf{V} \cdot \mathbf{D}^2 / N \cdot \mathbf{V}^\top.$$

Итак, столбцы  $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_d$  матрицы  $\mathbf{V}$  представляют собой собственный базис матрицы ковариаций  $\mathbf{S}$ ,

$\sigma_j^2 / N$  — собственные числа этой матрицы.

Векторы  $\mathbf{v}_j$  называются также *главными* (или *основными*) *компонентами* (*principal components*) для данных  $\mathbf{X}$ .

Пусть  $\mathbf{z}_j = \mathbf{X}\mathbf{v}_j$ .

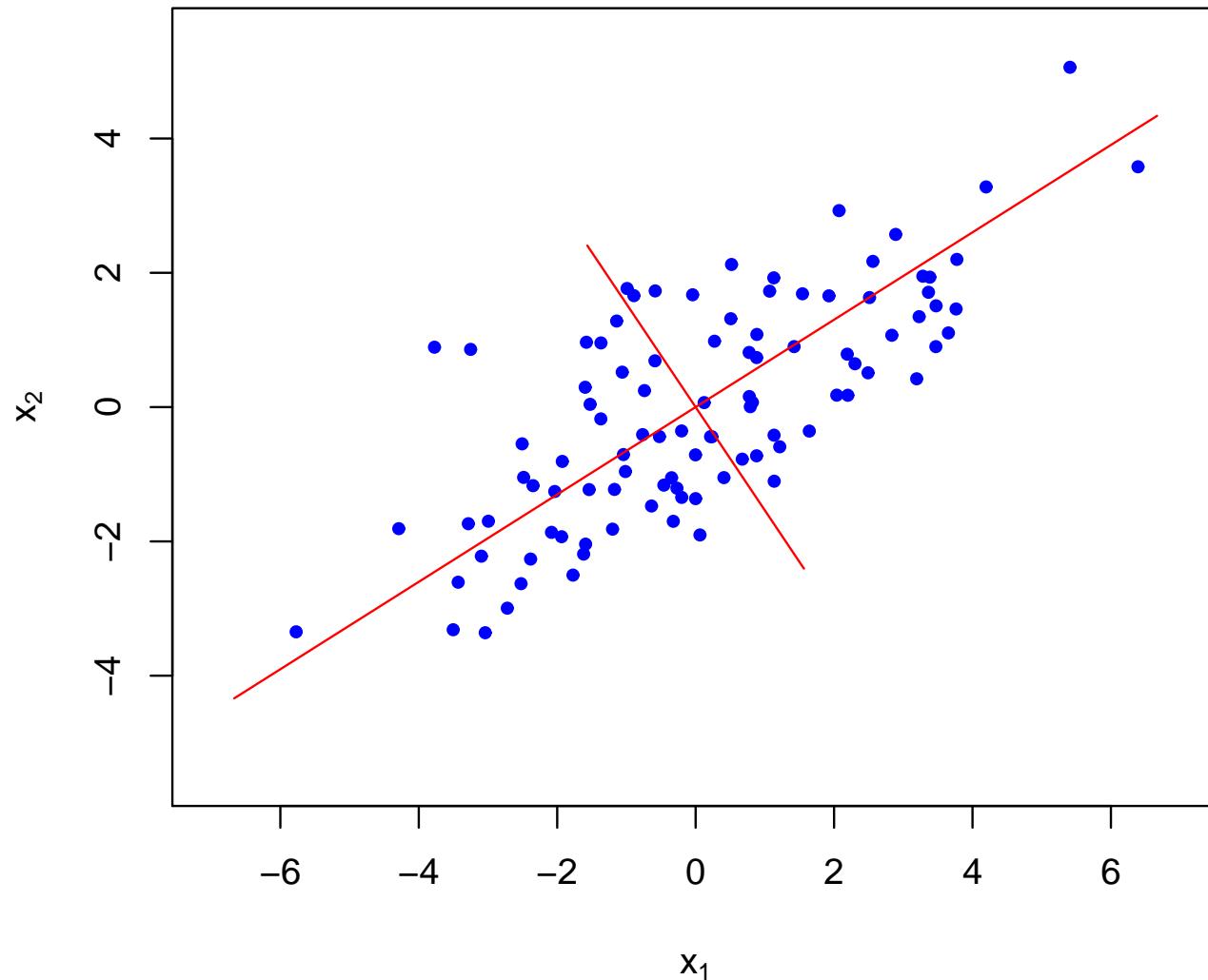
Легко проверить, что

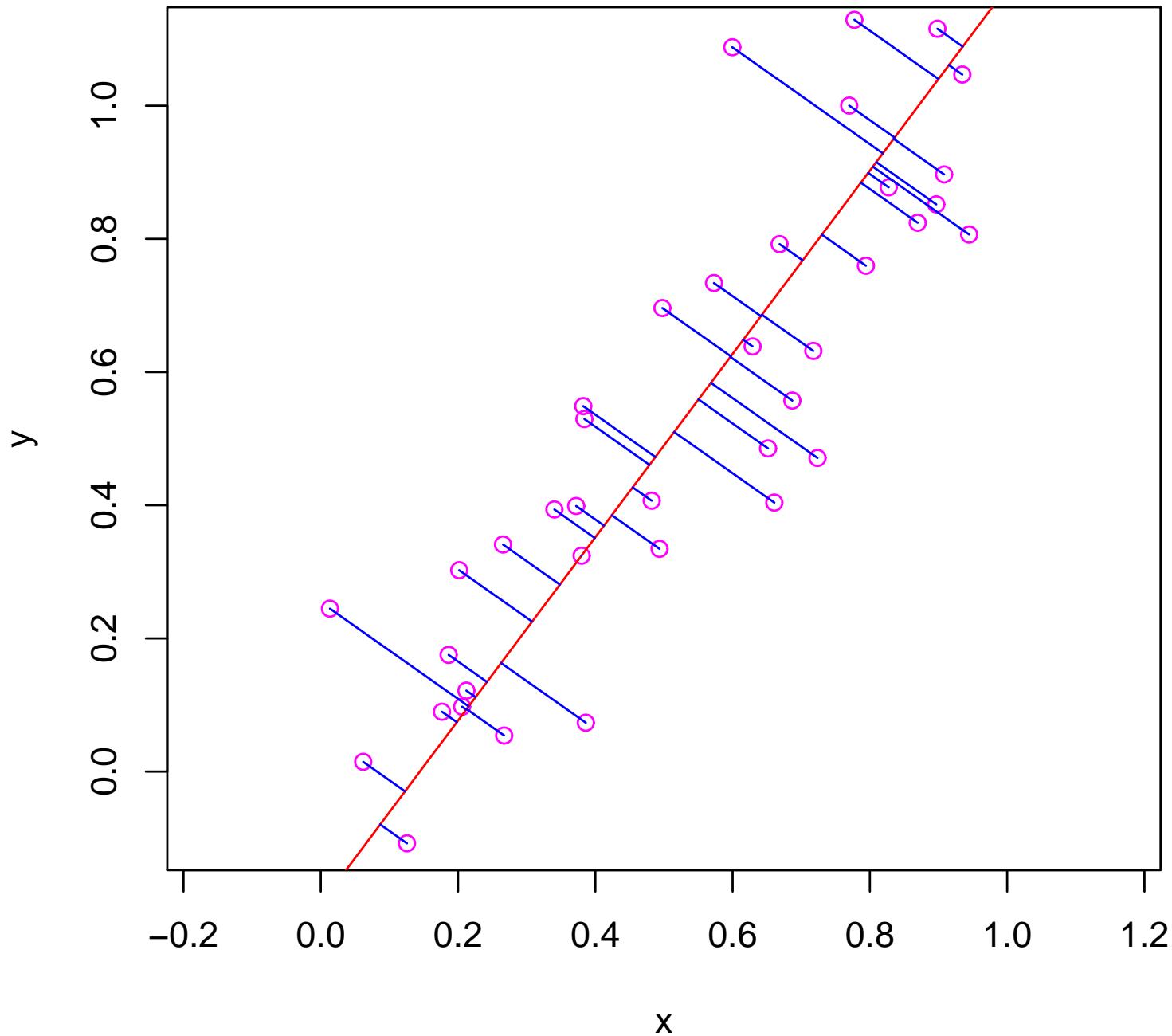
$$\text{Var}(\mathbf{z}_j) = \text{Var}(\mathbf{X}\mathbf{v}_j) = \frac{\sigma_j^2}{N}, \quad \mathbf{z}_j = \mathbf{X}\mathbf{v}_j = \sigma_j \mathbf{u}_j.$$

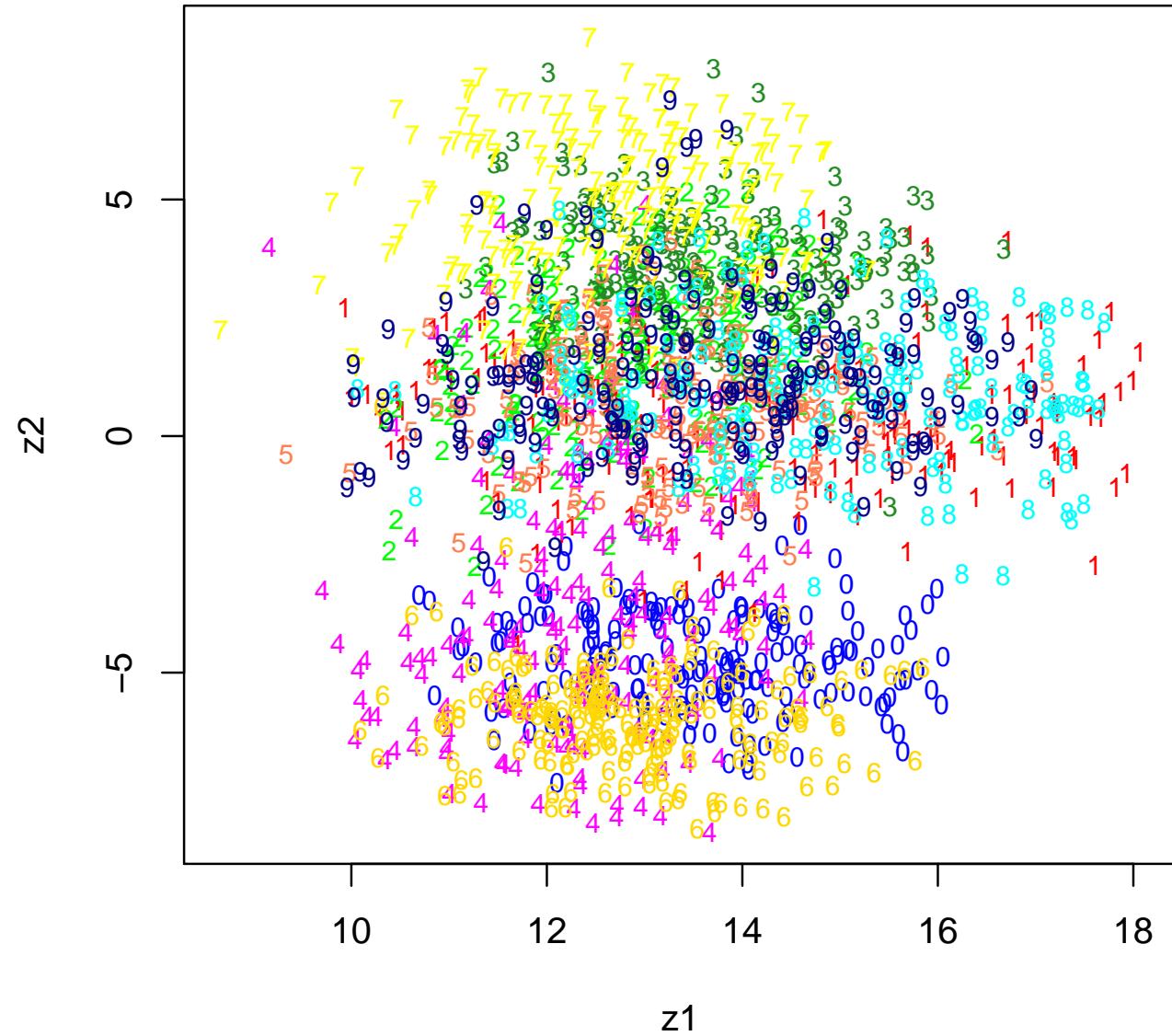
Первая главная компонента  $\mathbf{u}_1$  обладает тем свойством, что  $\mathbf{z}_1$  имеет максимальную дисперсию среди всех нормированных линейных комбинаций столбцов матрицы  $\mathbf{X}$ .

Вектор  $\mathbf{u}_j$  выбран среди всех векторов, ортогональных  $\mathbf{u}_1, \dots, \mathbf{u}_{j-1}$ , так, что  $\mathbf{z}_j$  имеет максимальную дисперсию.

Вектор  $\mathbf{z}_d$  имеет минимальную дисперсию.







## 9.2.2. Самоорганизующиеся карты Кохонена

Самоорганизующаяся карта Кохонена (*self-organized map, SOM*) является одним из методов многомерного шкалирования.

Прототипы  $m_{jj'} \in \mathbf{R}^d$  ( $j = 1, 2, \dots, K_1$ ,  $j' = 1, 2, \dots, K_2$ ),  $K = K_1 K_2$ .

$m_{jj'}$  и  $m_{kk'}$  — соседи, если  $|j - k| + |j' - k'| \leq \rho$

$N(i, i', \rho) = \{(k, k') : |j - k| + |j' - k'| \leq \rho\}$

$0 < \alpha < 1$

Инициализировать значения прототипов  $m_{jj'} \in \mathbf{R}^d$

**repeat**

**for**  $i = 1, 2, \dots, N$

Найти ближайший к  $x_i$  прототип  $m_{jj'}$  в  $\mathbf{R}^d$

**for**  $(k, k') \in N(i, i', \rho)$  (т. е. для каждого соседа  $m_{kk'}$  прототипа  $m_{jj'}$ )

$m_{kk'} \leftarrow m_{kk'} + \alpha(x_i - m_{kk'})$

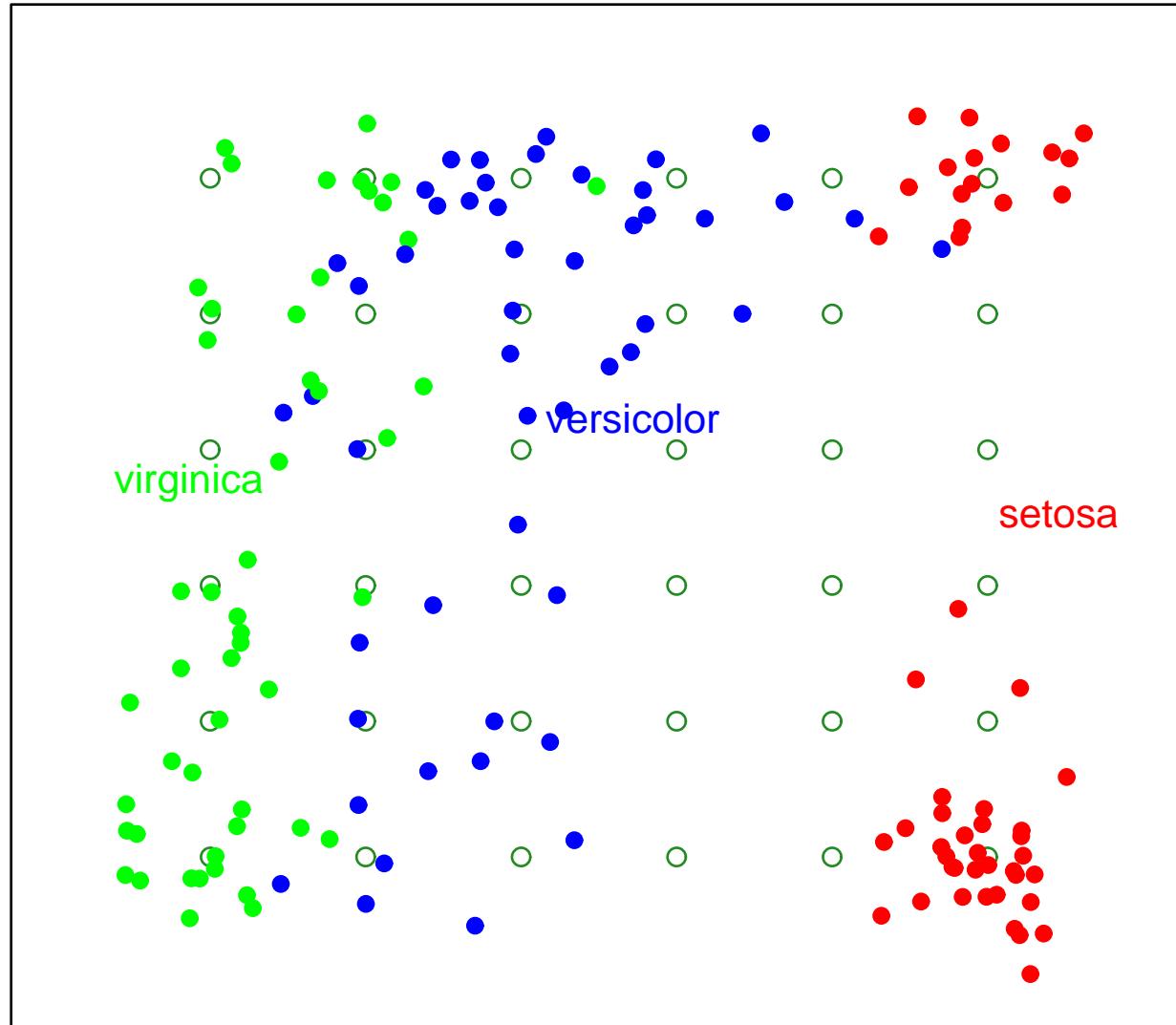
**end**

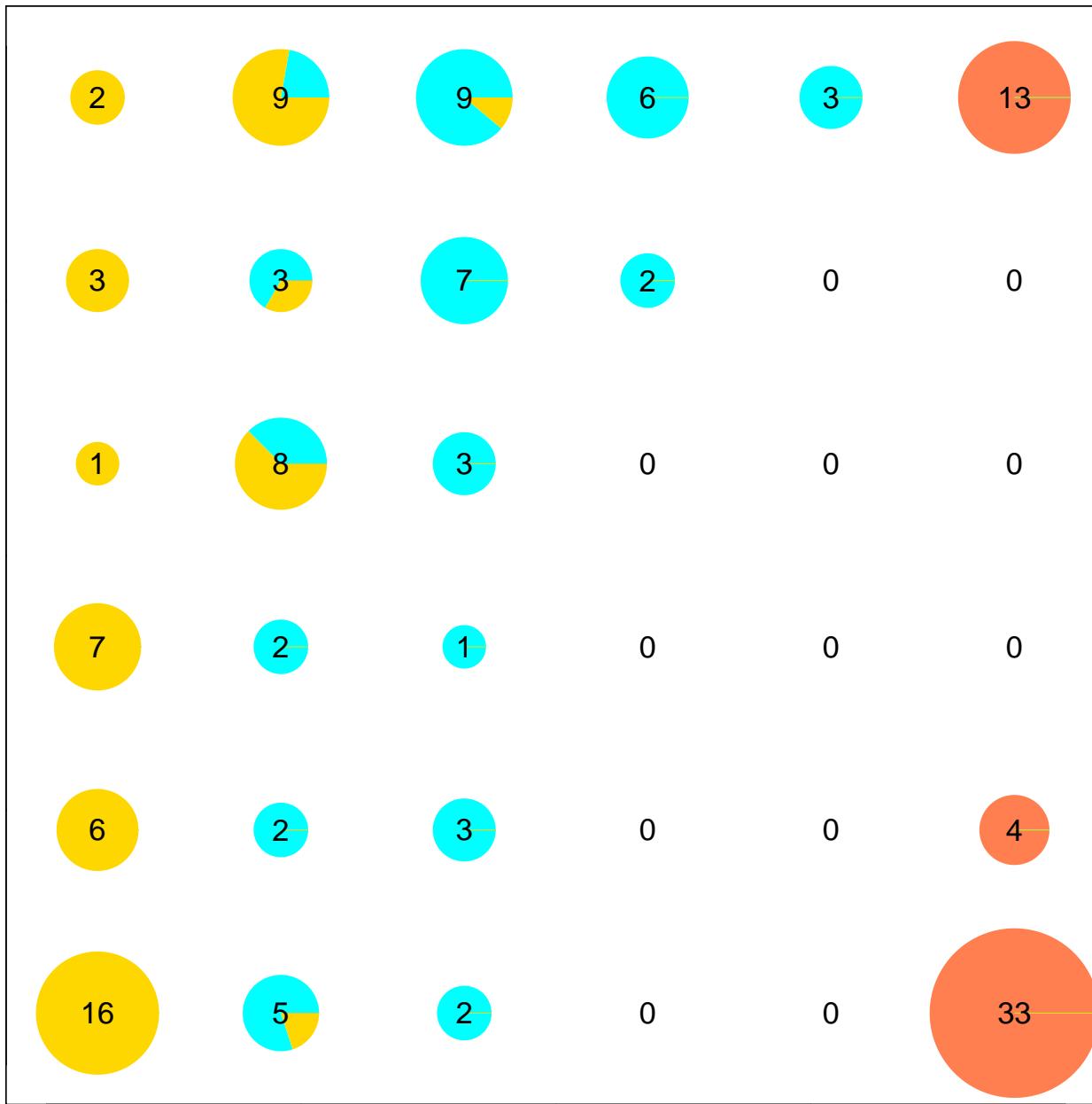
**end**

**end**

Можно использовать не прямоугольную, а гексагональную или треугольную сетку, и т. п.

Самоорганизующая карта Кохонена для данных по классификации цветов ириса. Данные содержат 150 объектов, 4 признака. Используется карта с прямоугольной сеткой  $6 \times 6$ . Точки, соответствующие объектам, случайным образом расположены вокруг узлов сетки, которым они приписаны. Точки, соответствующие цветам разных сортов, окрашены в разные цвета. Информация о сортах при обучении не использовалась.





## 9.3. Кластеризация

Алгоритмы кластеризации разбивают заданное множество объектов на группы (кластеры), в одном кластере размещая близкие, а в разных — далекие по своим характеристикам объекты.

Количество кластеров может быть известно заранее, а может определяться в процессе работы алгоритма.

Некоторые алгоритмы кластеризации на вход требуют признаковые описания  $x^{(1)}, x^{(2)}, \dots, x^{(N)} \in \mathbf{R}^d$  объектов. Для других достаточно только матрицы различий  $D$ .

Если число кластеров известно заранее, то с формальной точки зрения алгоритм кластеризации находит отображение

$$C : \{1, 2, \dots, N\} \rightarrow \{1, 2, \dots, K\},$$

где  $N$  — число имеющихся объектов, а  $K$  — число кластеров.

### 9.3.1. Кластеризация методами теории графов

1. Пусть  $D$  — матрица различий.

Рассмотрим полный граф с  $N$  вершинами. Каждой вершине соответствует объект.

Припишем ребрам графа веса  $\delta_{ii'}$ .

Пусть  $r$  — некоторый параметр. Удалим из графа все ребра, веса которых больше  $r$ .

Компоненты связности полученного таким образом графа и есть кластеры.

2. Для полного графа из  $N$  вершин, ребрам которого приписаны веса  $\delta_{ii'}$ , построим минимальное оствовное дерево.

Удалим из дерева  $K$  ребер максимального веса.

Получим разбиение множества объектов на  $K$  кластеров.

### 9.3.2. 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  $\mu_k$  центр тяжести системы точки, 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  $\mu_k$  ( $k = 1, 2, \dots, m$ ), 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  $\mu_k$  минимально.

После этого повторим итерацию.

Получаем следующий алгоритм.

**begin**

Случайно инициализировать  $\mu_k$  ( $k = 1, 2, \dots, K$ )

**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**

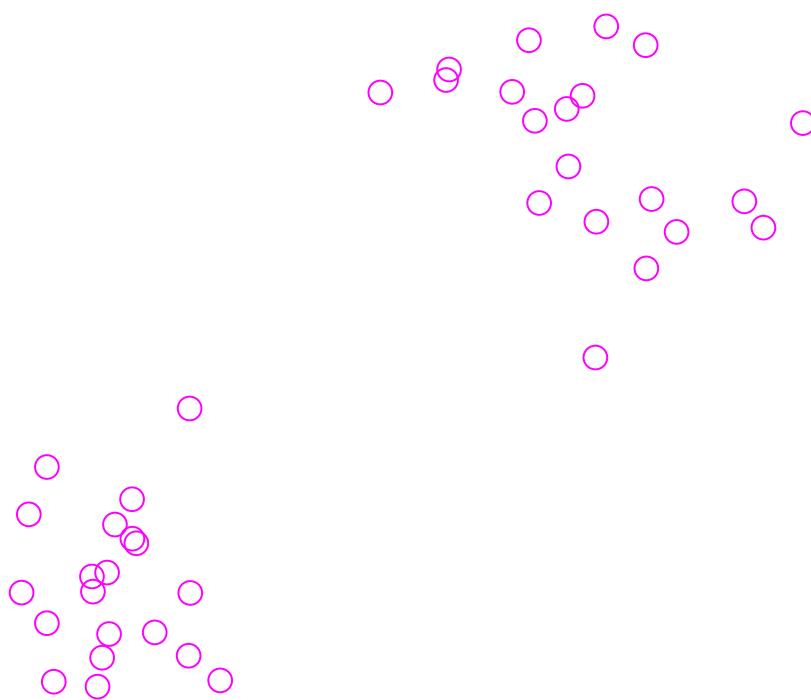
**end**

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

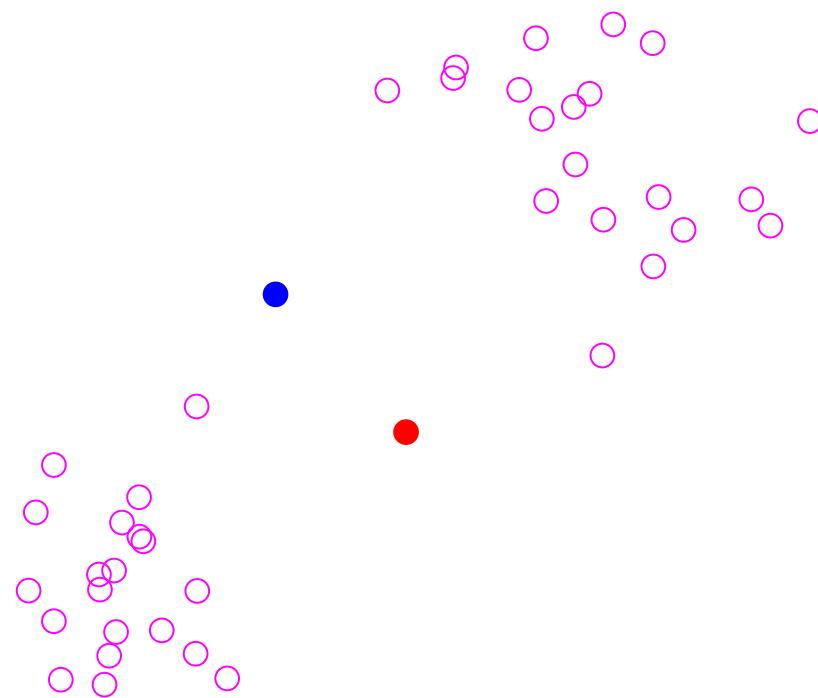
Алгоритм может не сойтись к глобальному минимуму, но всегда найдет локальный минимум.

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

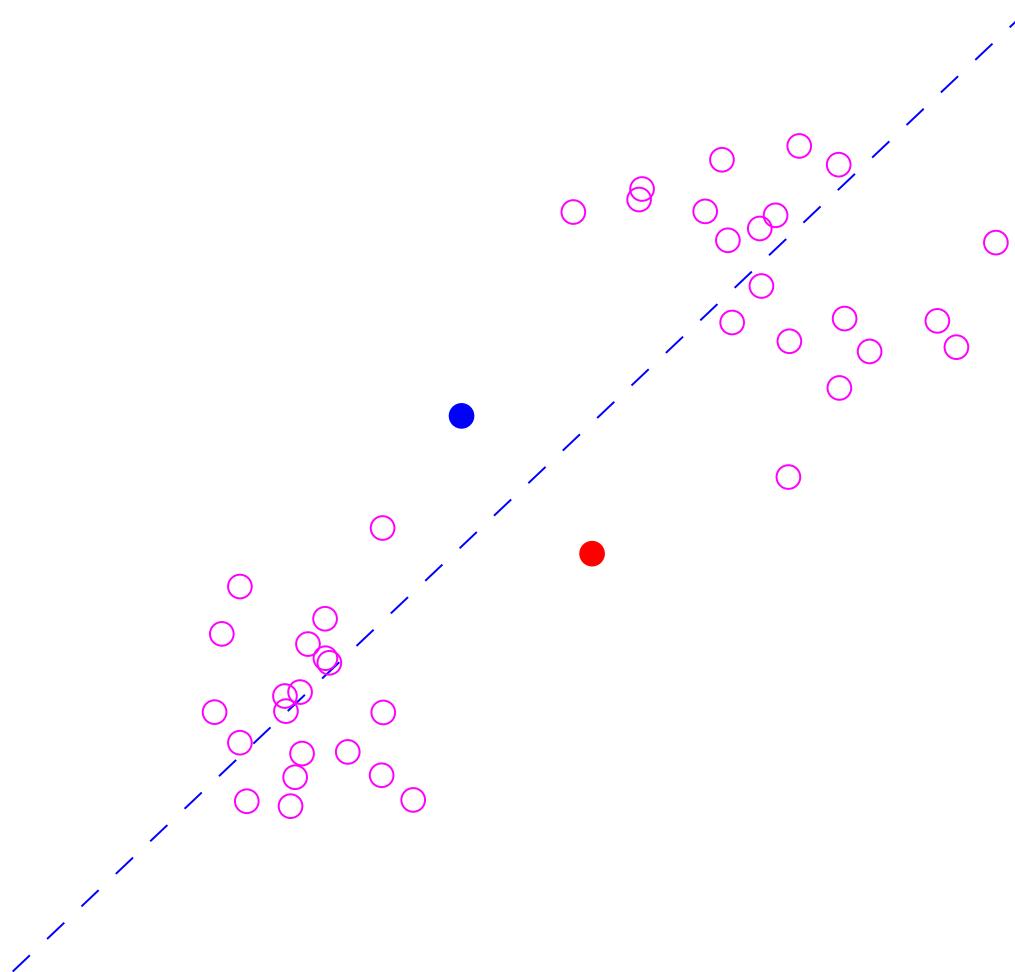
Иллюстрация,  $K = 2$ :



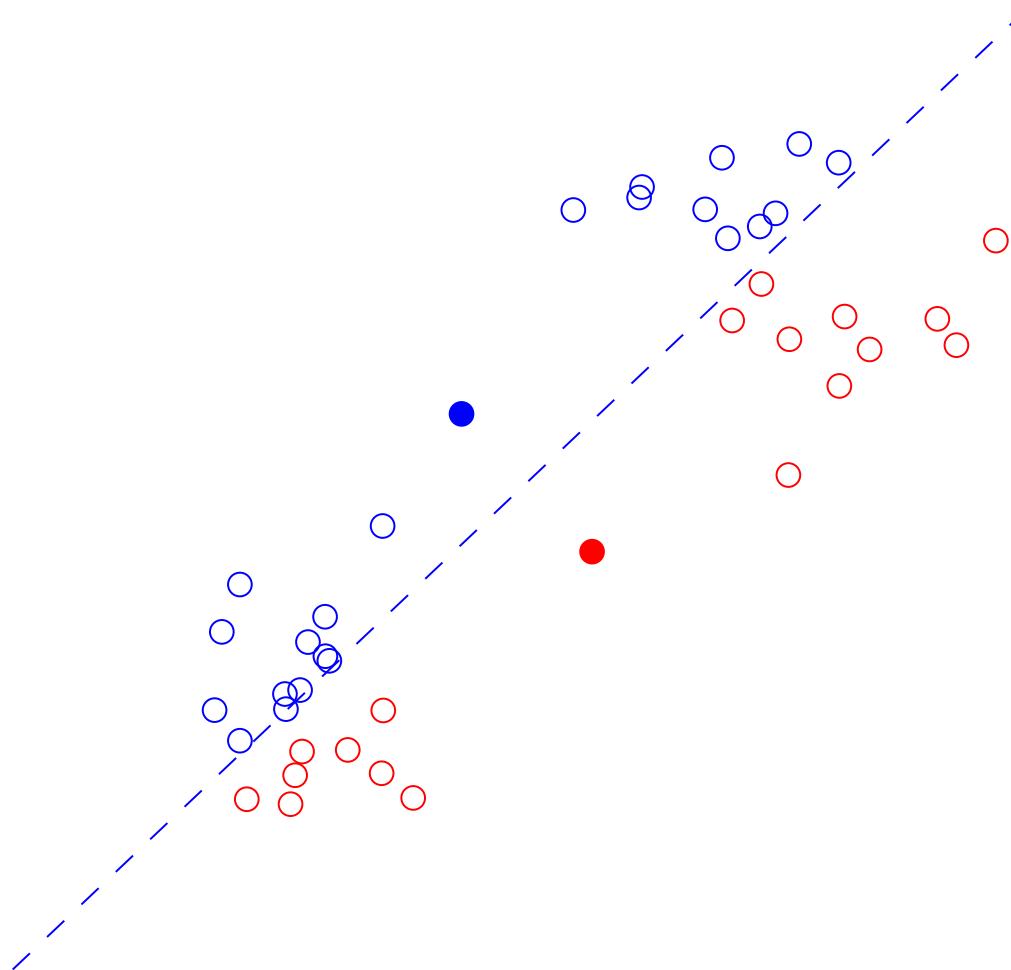
Случайно выбираем центры кластеров:



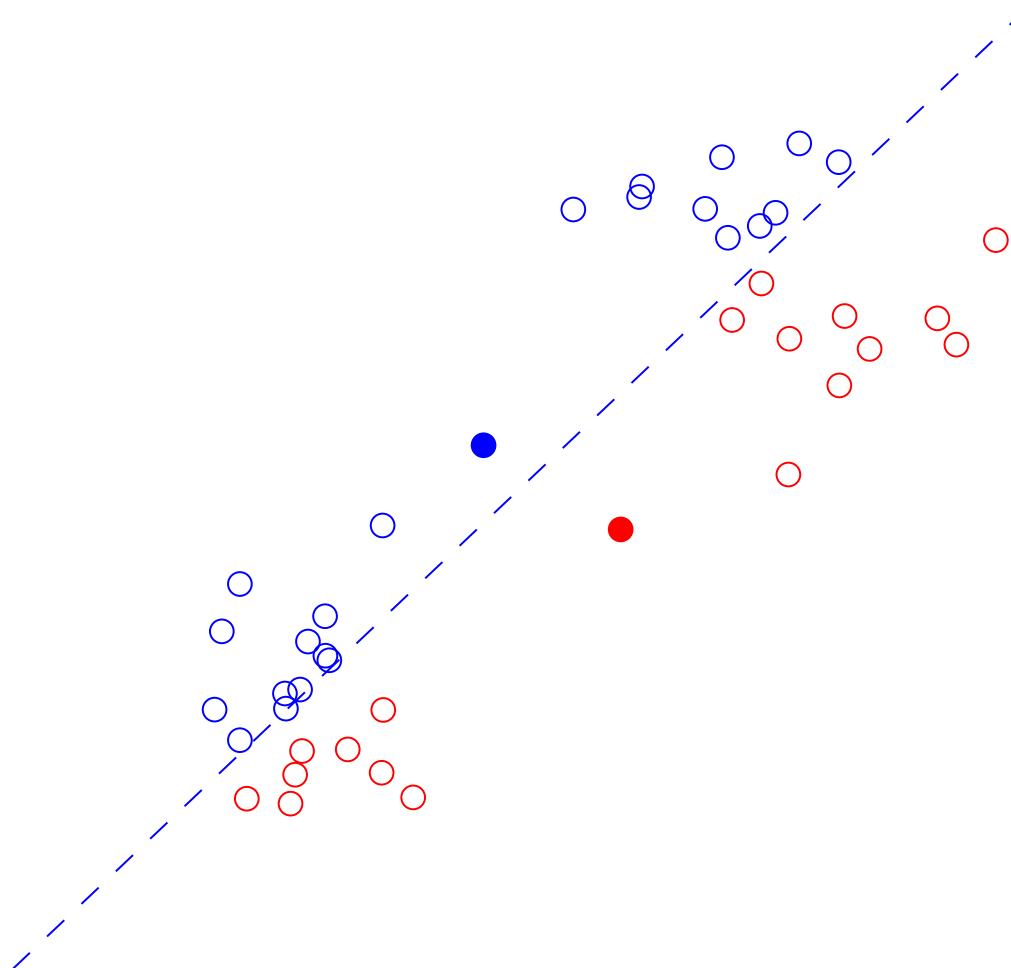
Находим ближайшие точки:



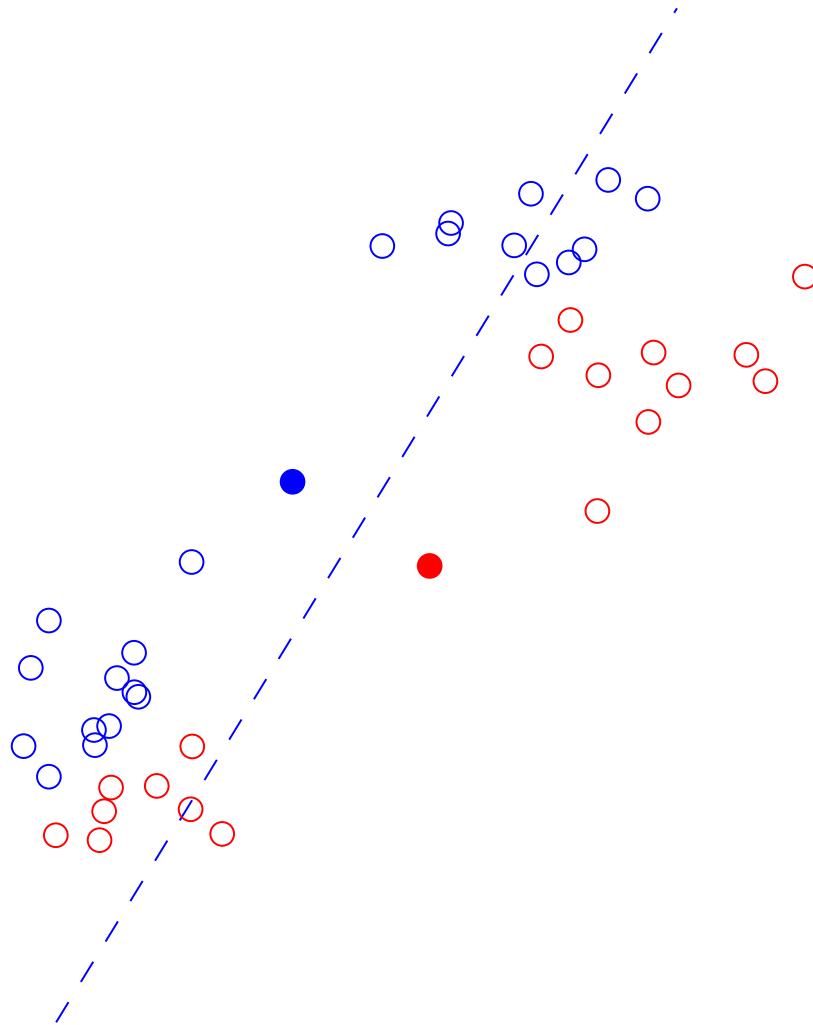
Находим ближайшие точки:



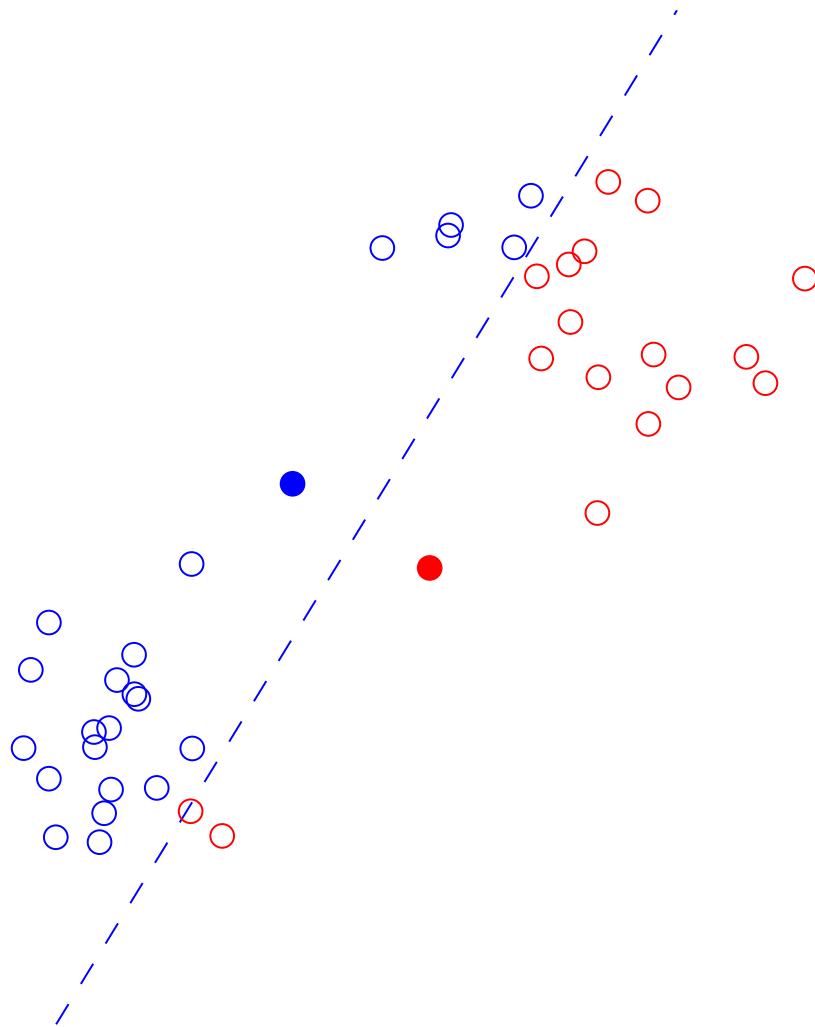
Вычисляем центры кластеров:



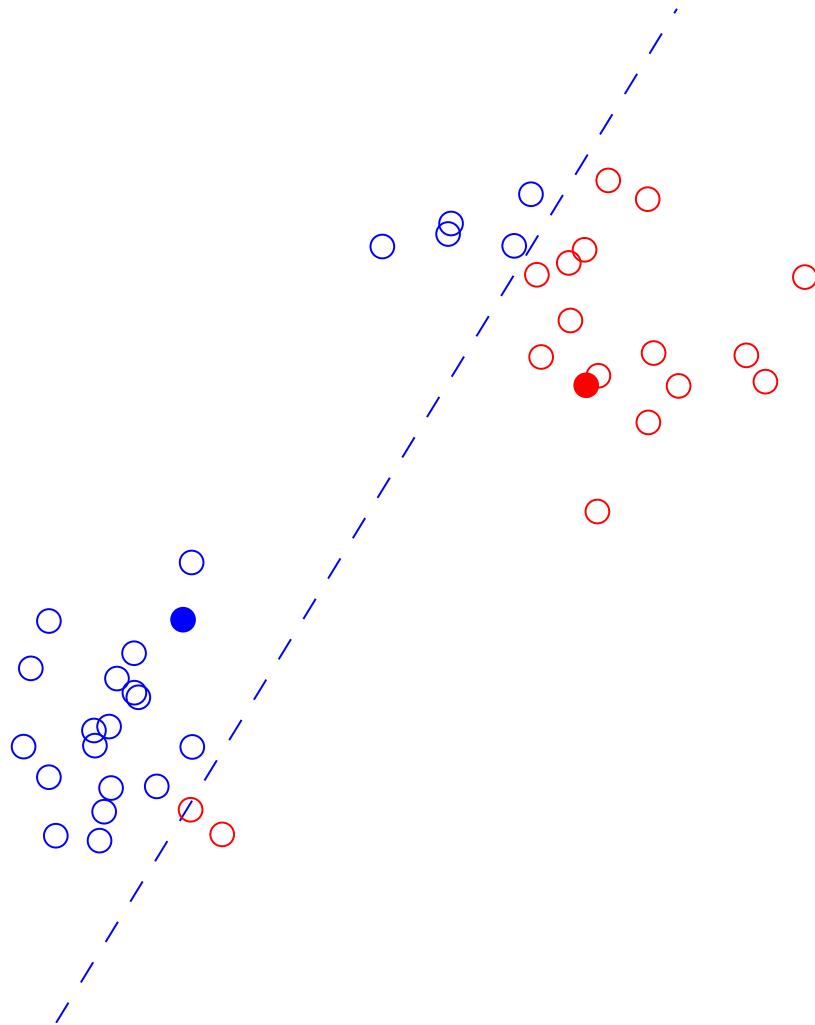
Находим ближайшие точки:



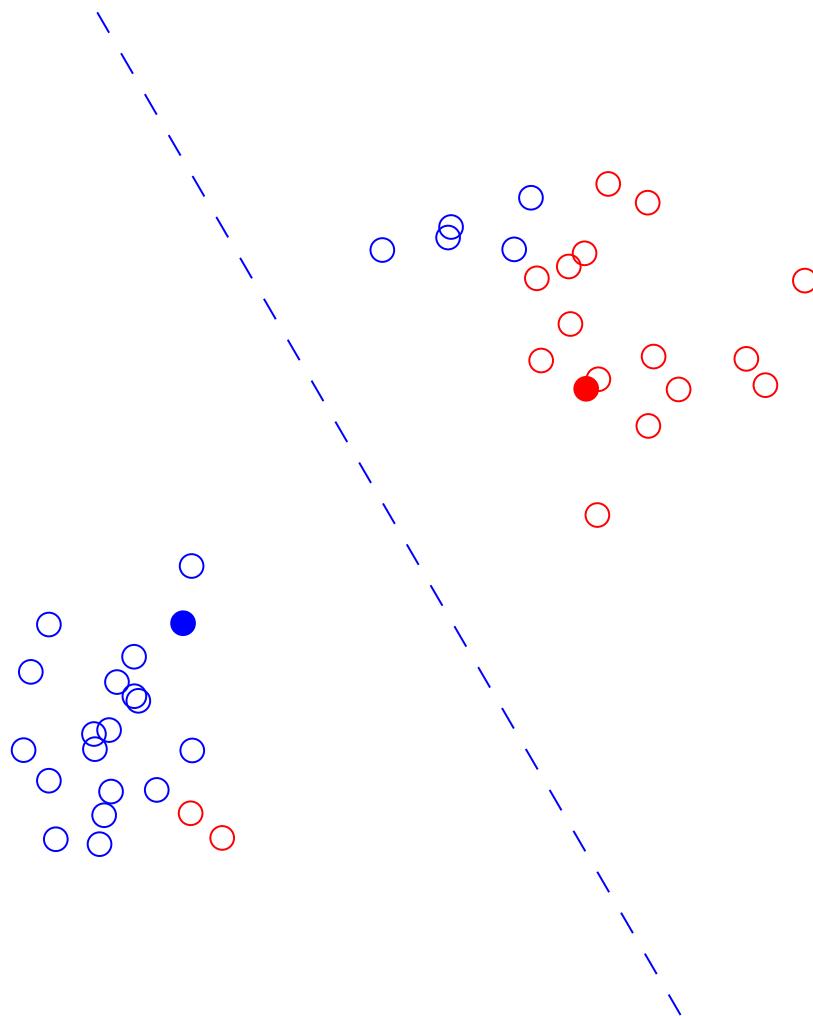
Находим ближайшие точки:



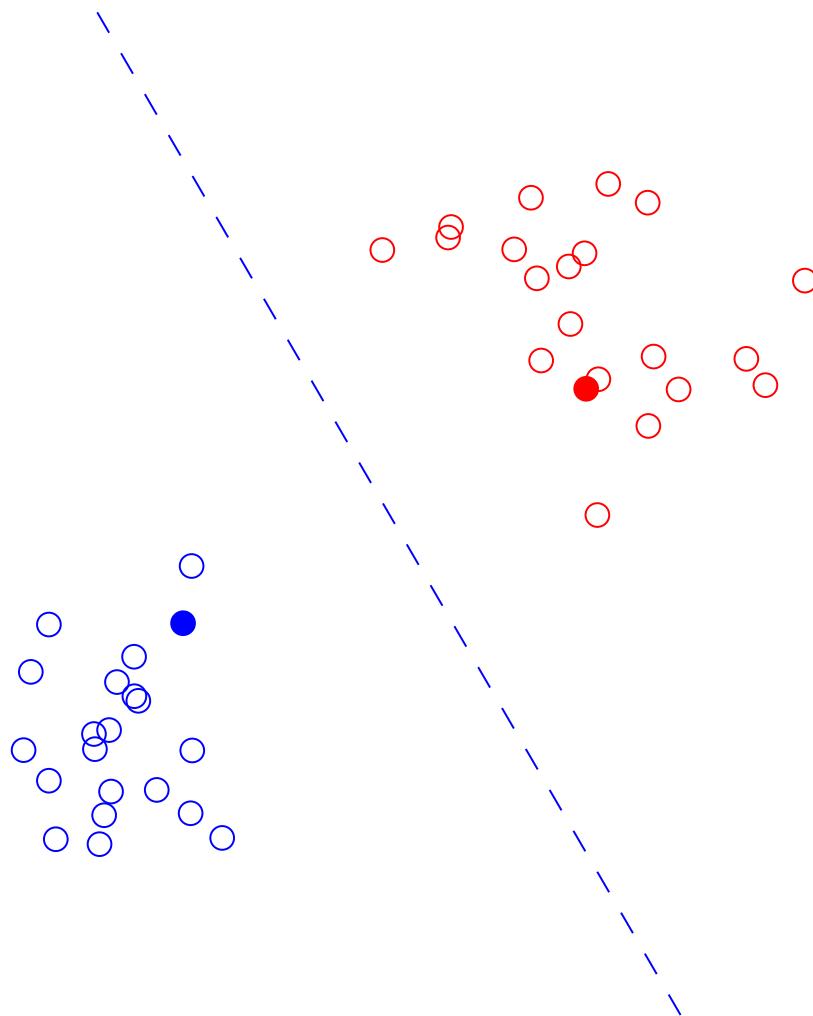
Вычисляем центры кластеров:



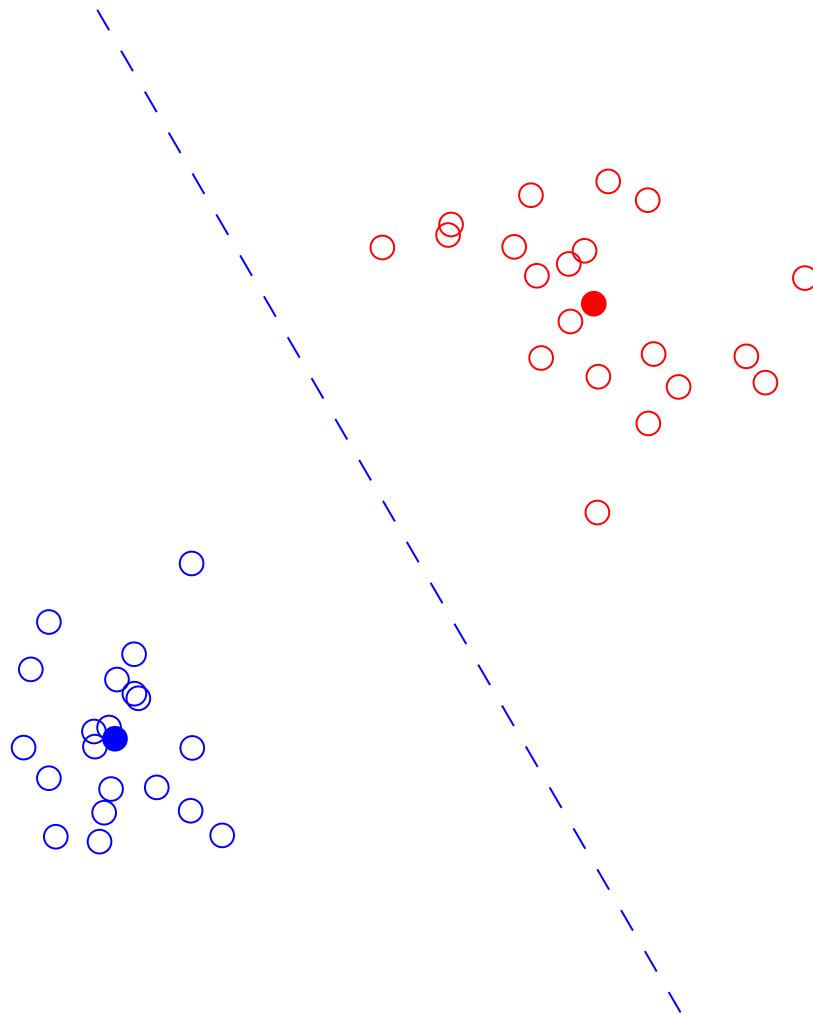
Находим ближайшие точки:



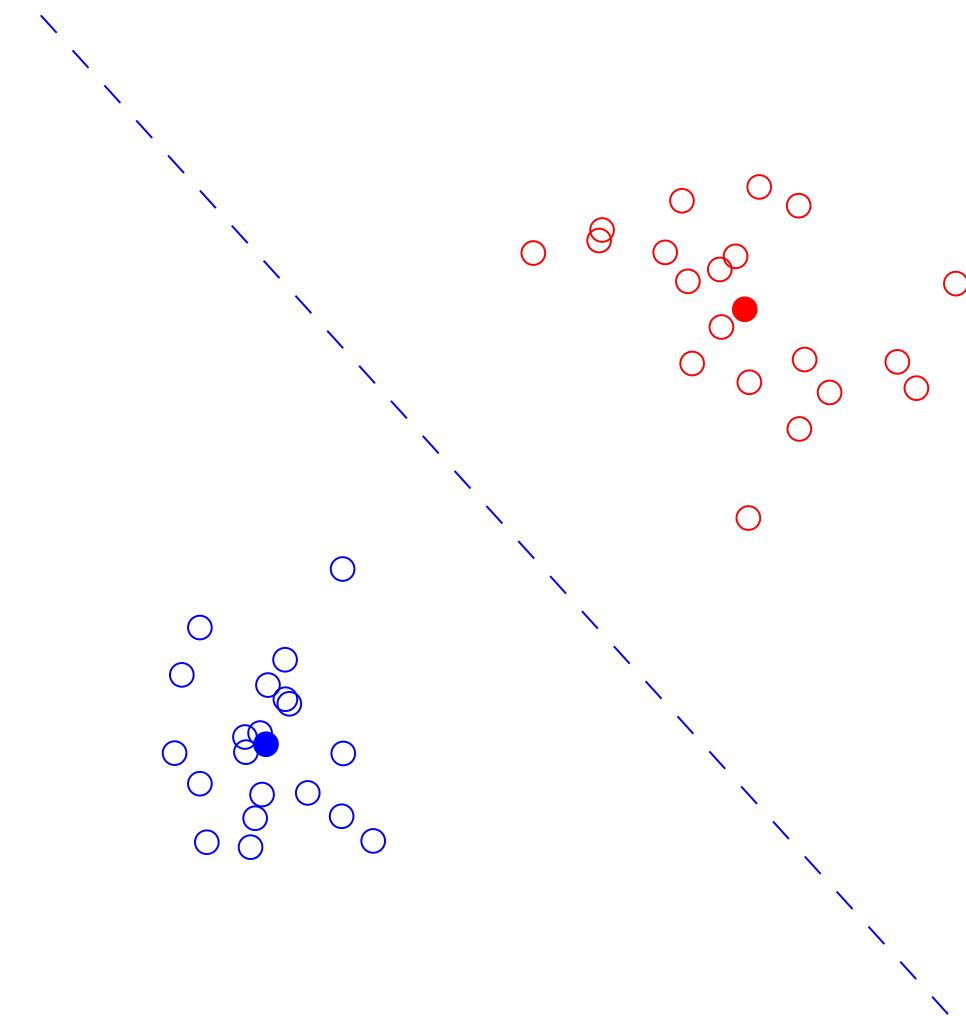
Находим ближайшие точки:



Вычисляем центры кластеров:



Находим ближайшие точки:



### 9.3.3. Метод медоидов

Метод центров тяжести работает только с явными описаниями объектов, так как находит  $\mu_k$ .

*Метод медоидов (K-medoids)* является версией метода центров тяжести, однако ему достаточно только матрицы расстояний  $D = (d_{ii'})$ .

Вместо того, чтобы работать с центрами тяжести, найдем в каждом кластере *медоид*  $x^{(i_k^*)}$ :

$$i_k^* = \operatorname{argmin}_{\{i: C(i)=k\}} \sum_{C(i')=k} \delta_{ii'}^2.$$

Обновим  $C(i)$  ( $i = 1, 2, \dots, N$ ), приписав точке  $x^{(i)}$  тот класс  $k$  ( $k = 1, 2, \dots, N$ ), для которого расстояние от  $x^{(i)}$  до  $x^{(i_k^*)}$  минимально.

После этого повторим итерацию.

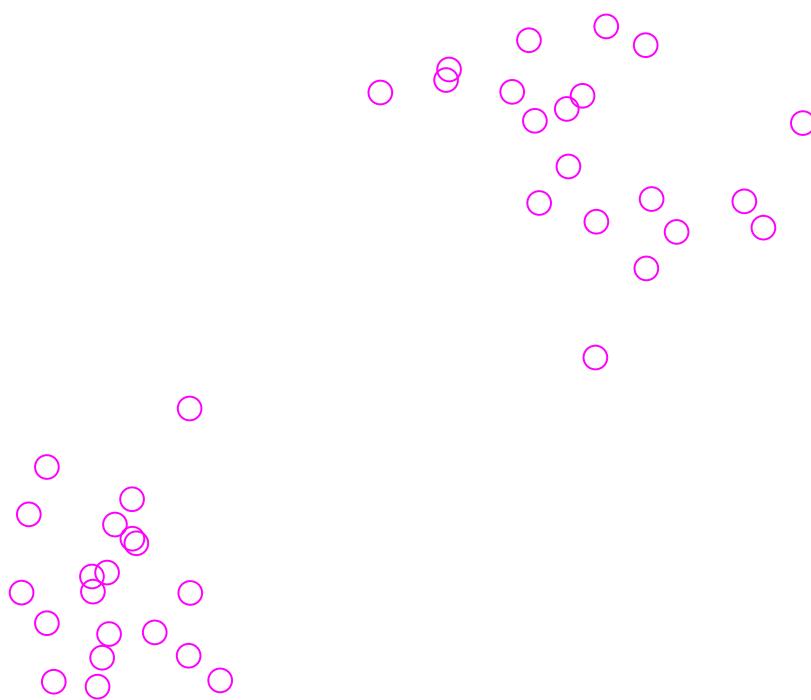
**repeat**

$$\text{Найти } i_k^* \leftarrow \operatorname{argmin}_{i: C(i)=k} \sum_{C(i')=k} \delta_{ii'}^2 \quad (k = 1, 2, \dots, K)$$

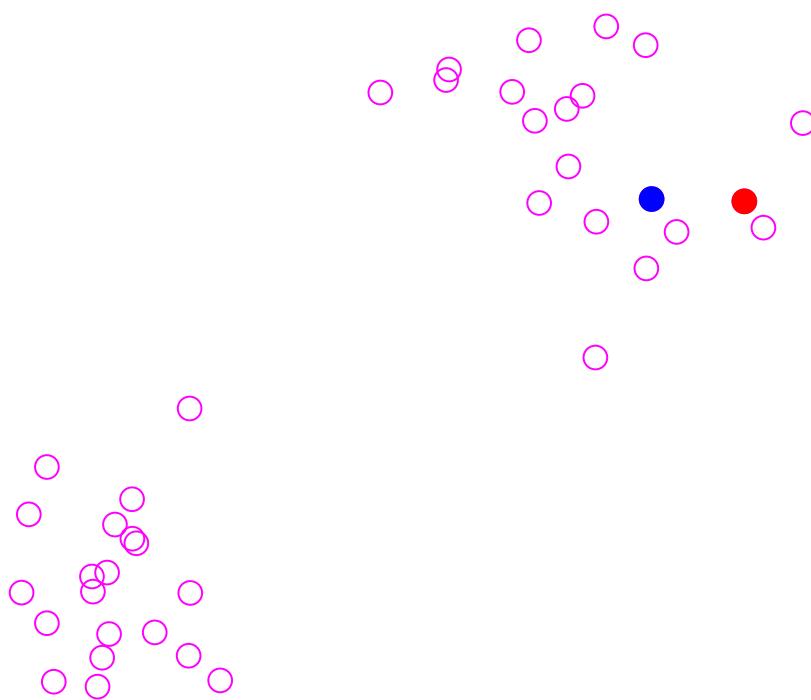
$$\text{Положить } C(i) \leftarrow \operatorname{argmin}_{1 \leq k \leq K} \delta_{i,i_k^*}^2 \quad (i = 1, 2, \dots, N)$$

**end**

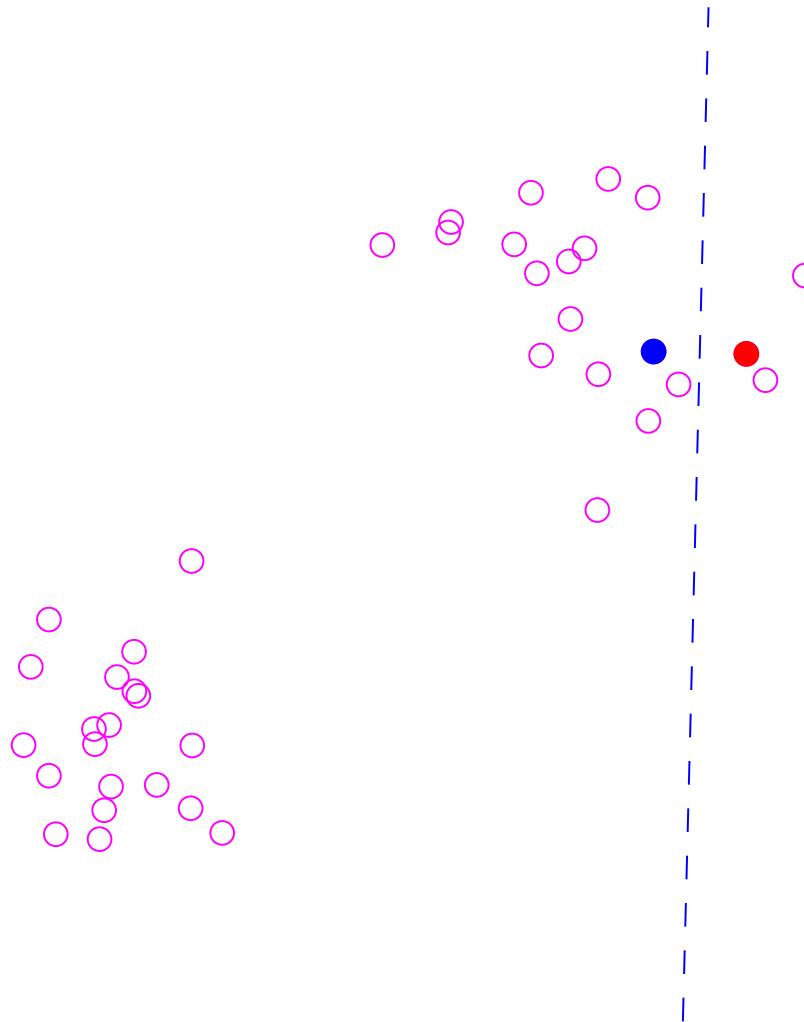
Иллюстрация,  $K = 2$ :



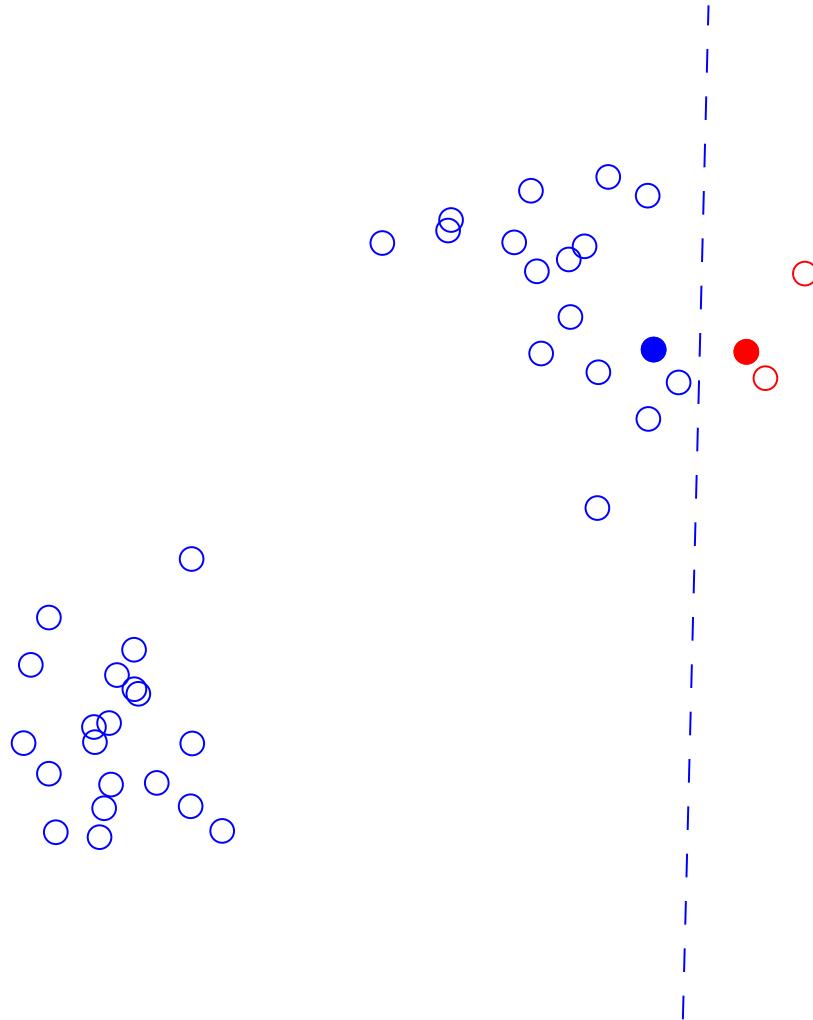
Случайно выбираем две точки из выборки:



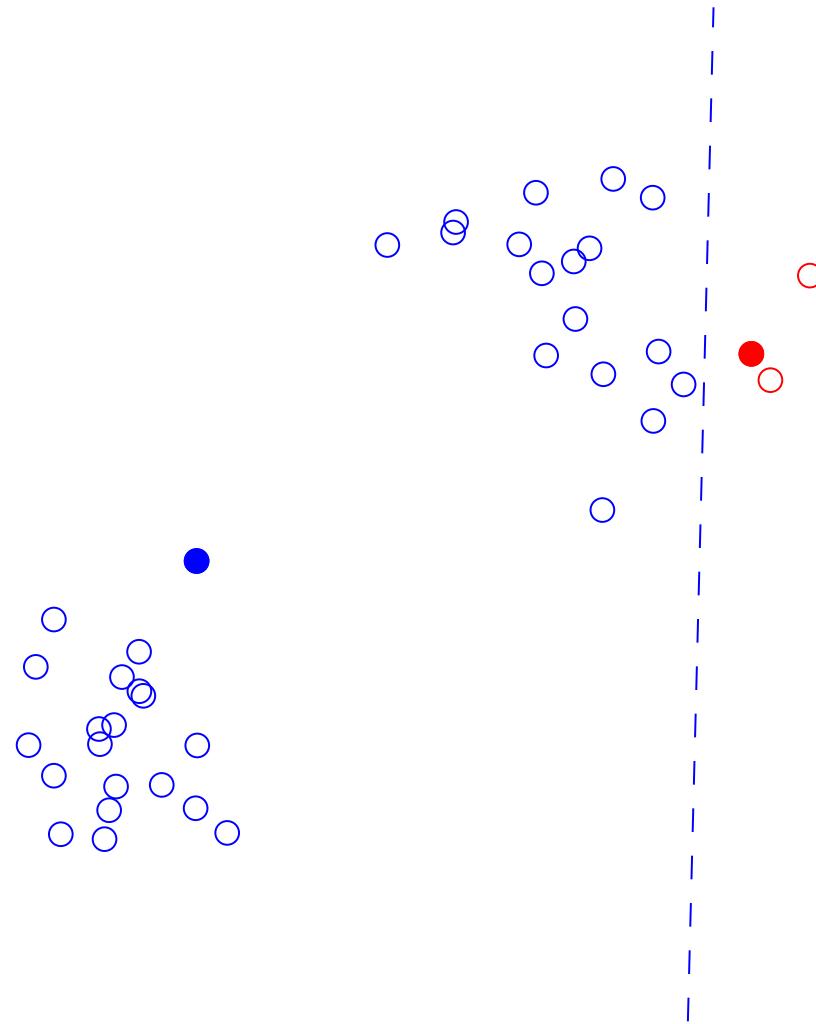
Находим ближайшие точки:



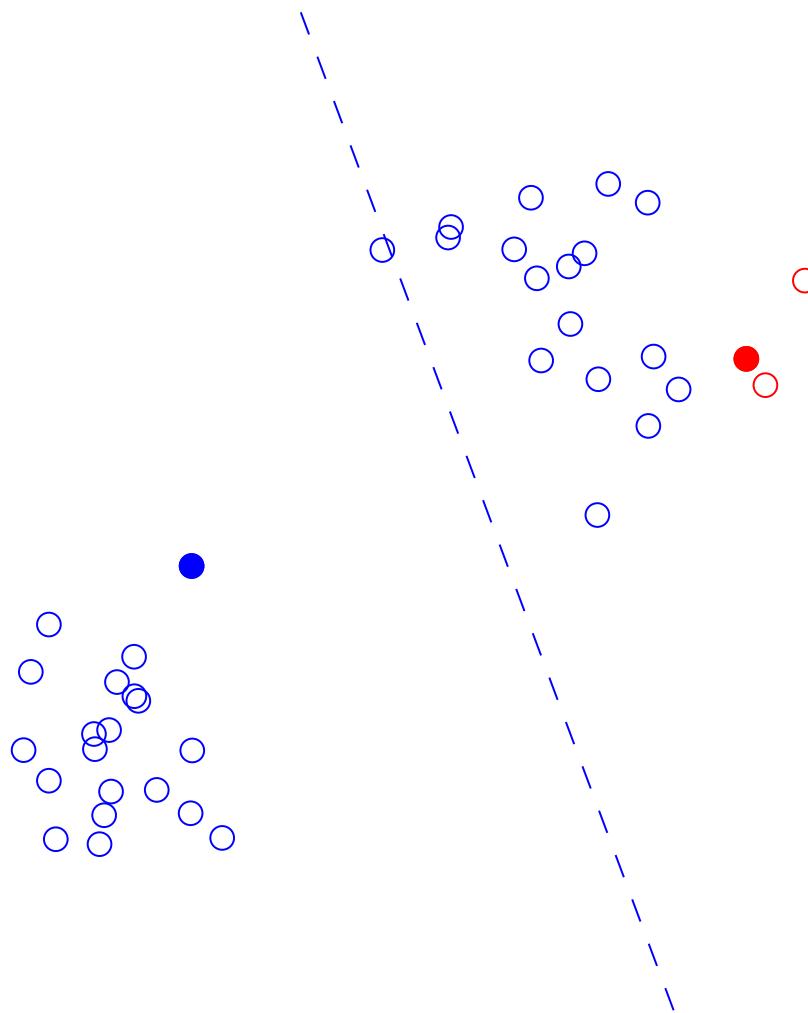
Находим ближайшие точки:



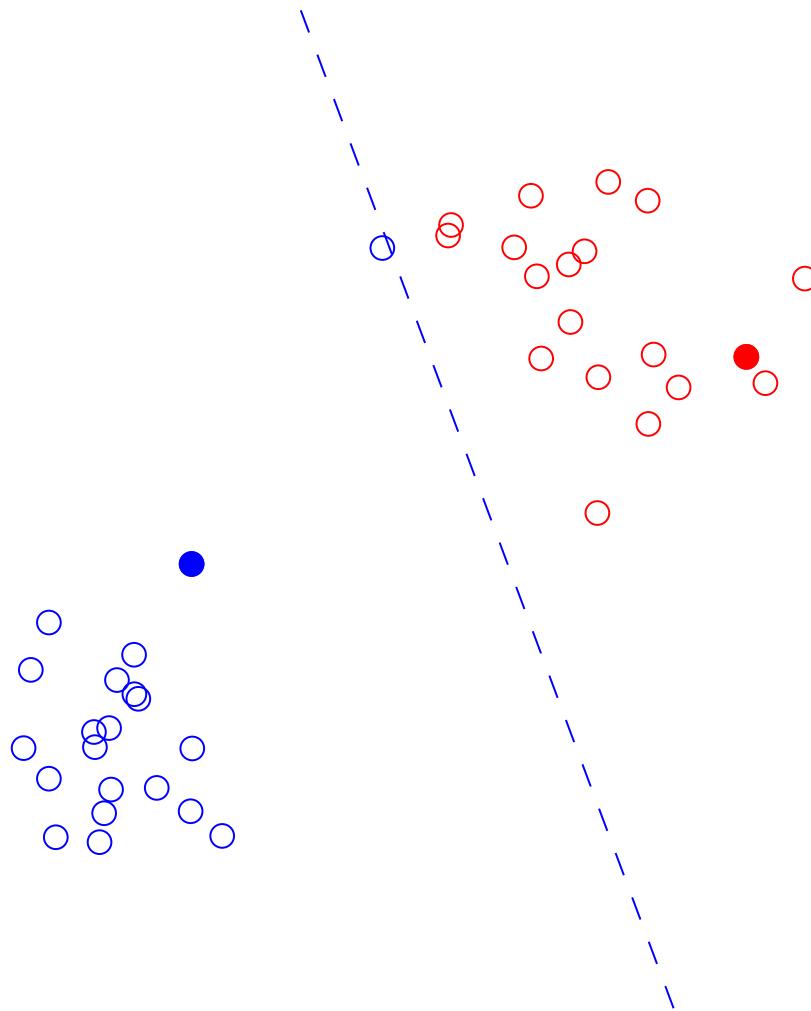
Находим медоиды:



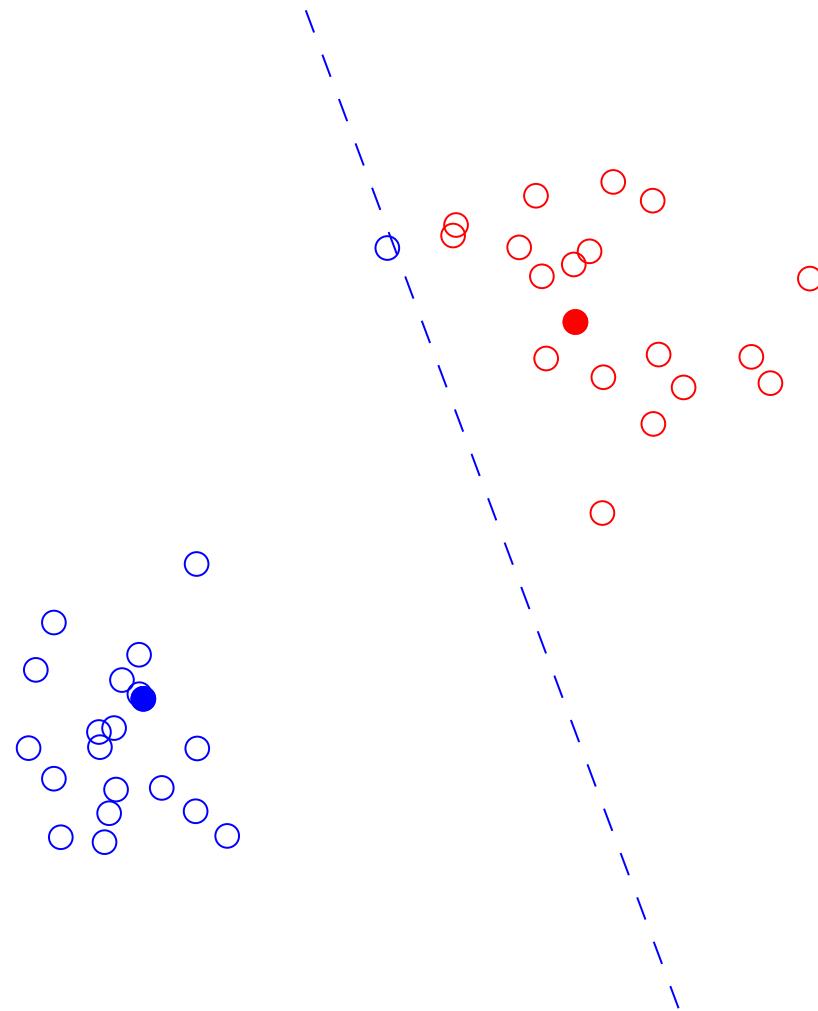
Находим ближайшие точки:



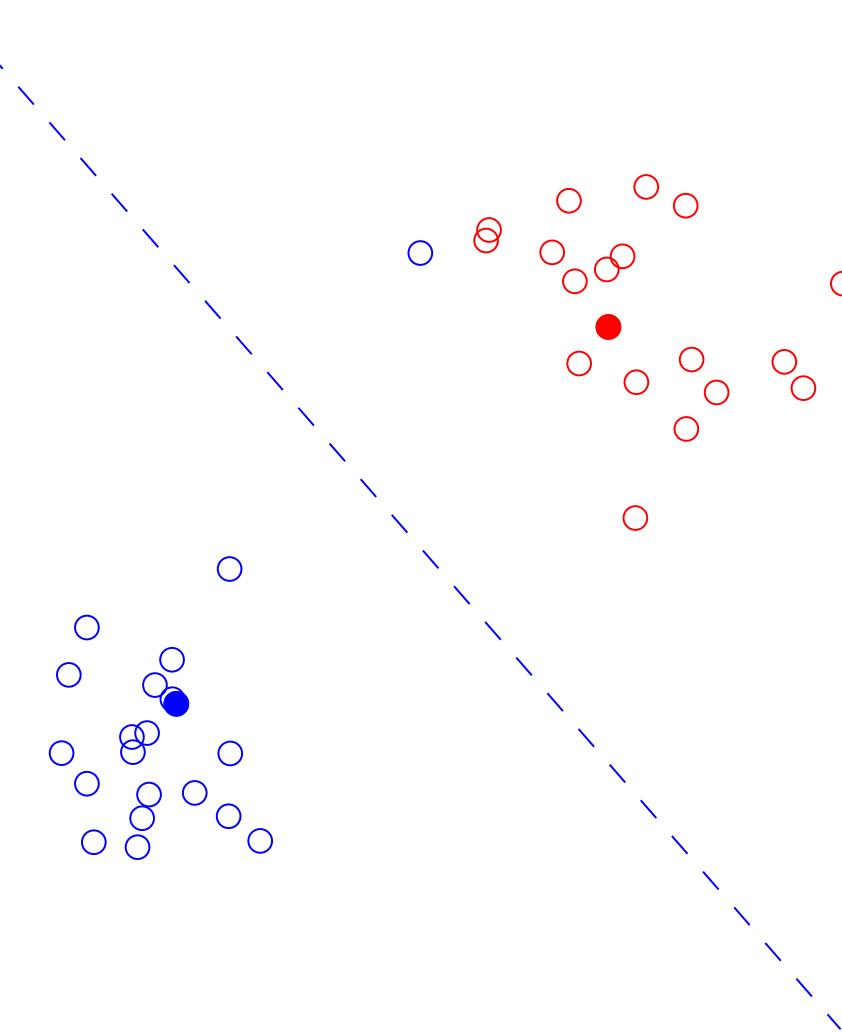
Находим ближайшие точки:



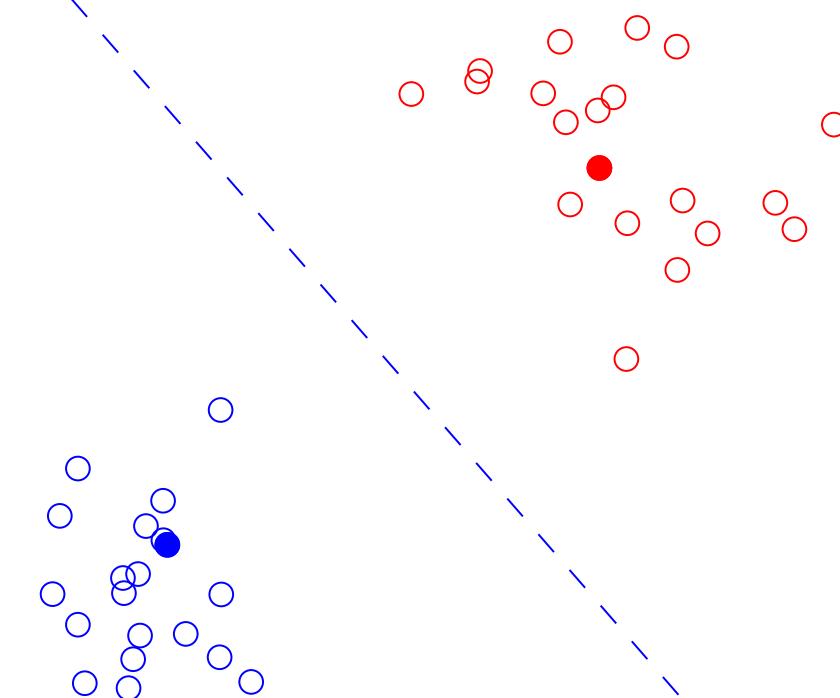
Находим медоиды:



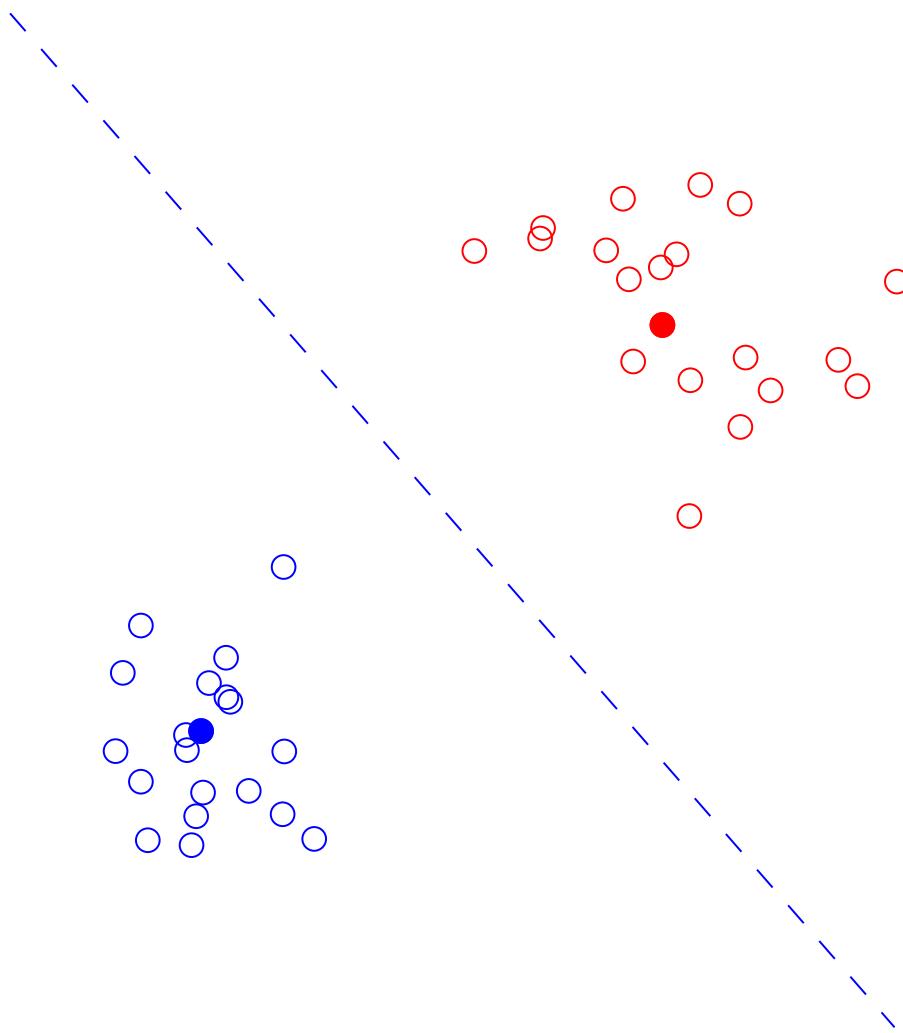
Находим ближайшие точки:



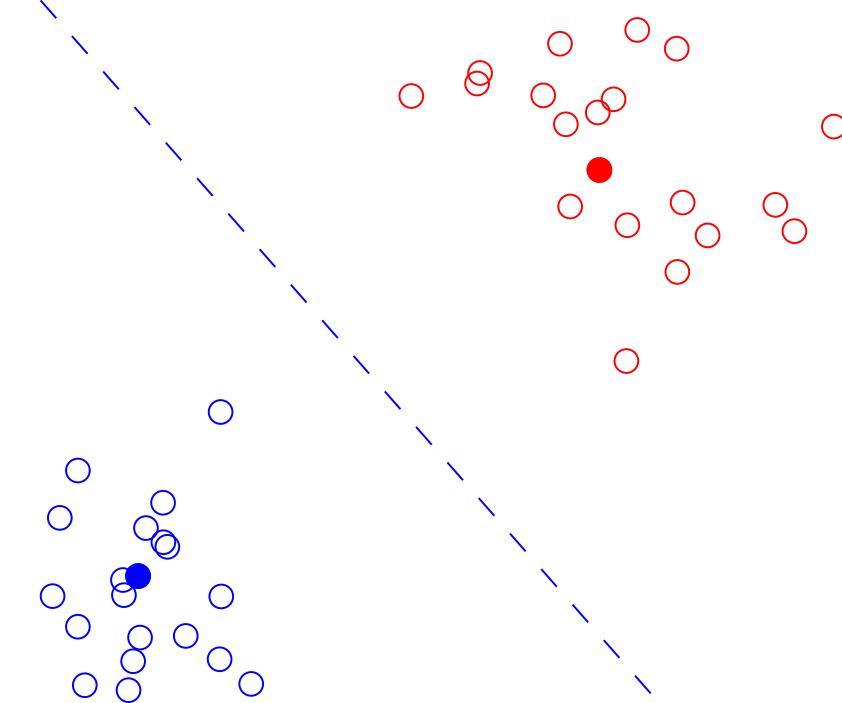
Находим ближайшие точки:



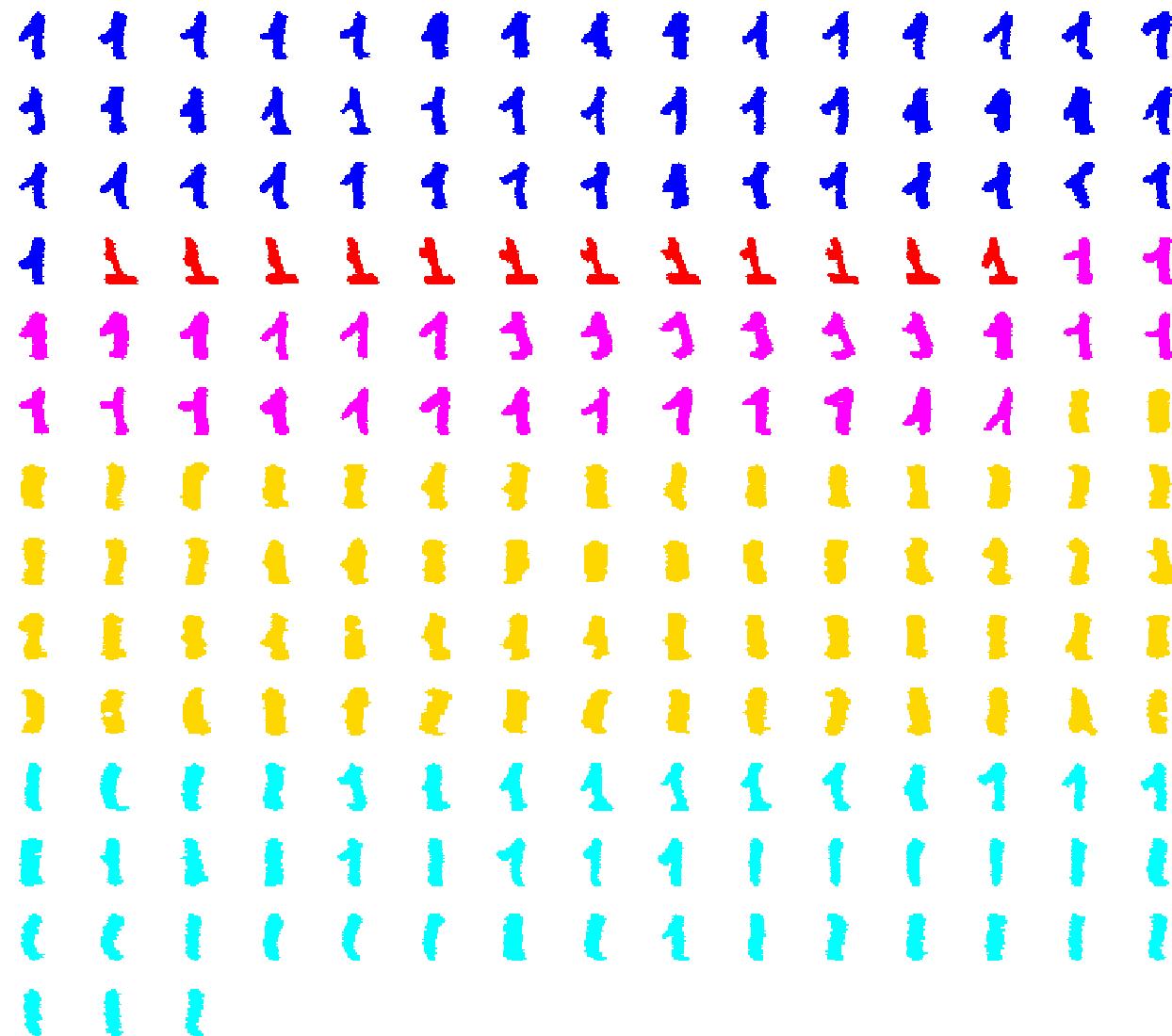
Находим медоиды:



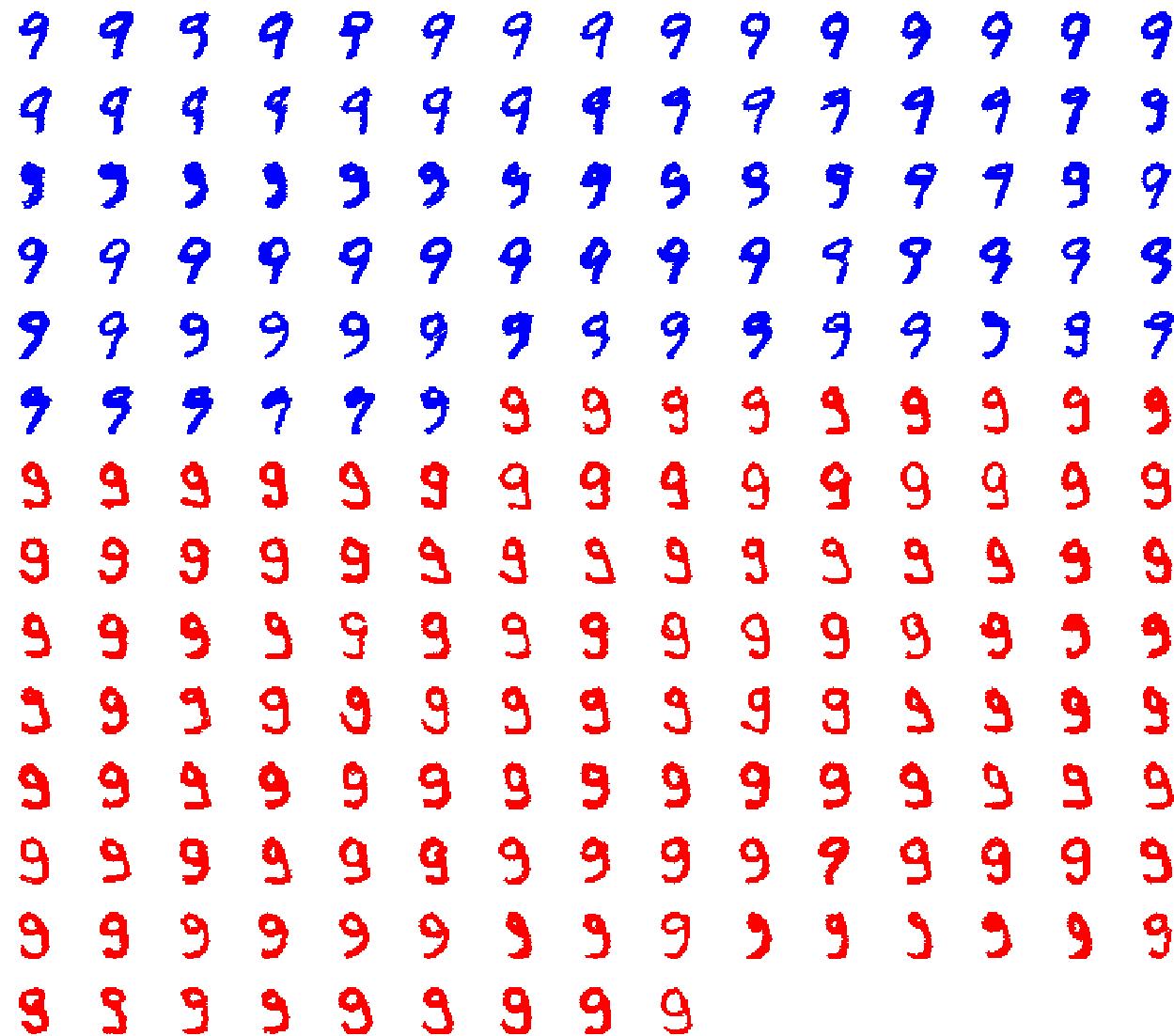
Находим ближайшие точки:



5 стилей написания цифры 1, найденные методом медоидов. Манхеттеновская метрика. Медоиды изображены первыми в своем кластере.



2 стиля написания цифры 9, найденные методом медоидов. Манхэттеновская метрика. Медоиды изображены первыми в своем кластере.



### 9.3.4. Метод нечетких множеств

Рассмотрим подход к задаче кластеризации, навеянный нечеткой логикой.

Зафиксируем некоторое натуральное  $K$  и рассмотрим множество  $U = \{1, 2, \dots, K\}$ , которое назовем универсом.

*Нечетким множеством* называется отображение  $V : U \rightarrow [0, 1]$ , для которого

$$0 \leq V(i) \leq 1, \quad (i = 1, 2, \dots, N).$$

Отображение  $V$  называется также *функцией принадлежности*, а значение  $V(i)$  — *степенью принадлежности* элемента  $i$  нечеткому множеству  $V$ .

Предположим, что кластеры являются нечеткими множествами.

Пусть тогда  $u_{ik}$  есть степень принадлежности  $i$ -го объекта  $k$ -му кластеру.

Имеем

$$\sum_{k=1}^K u_{ik} = 1, \quad 0 \leq u_{ik} \leq 1, \quad (i = 1, 2, \dots, N, \ k = 1, 2, \dots, K).$$

Обозначим  $m_k$  взвешенный центр тяжести системы точек  $x^{(1)}, x^{(2)}, \dots, x^{(N)}$  с весами  $u_{1k}, u_{2k}, \dots, u_{Nk}$ :

$$m_k = \sum_{i=1}^N u_{ik} x^{(i)}.$$

«Нечеткий» вариант метода центров тяжести заключается в итеративной минимизации

$$\min_{u_{ik}} \sum_{i=1}^N \sum_{k=1}^K u_{ik}^2 \|x^{(i)} - m_k\|^2.$$

А «нечеткий» вариант метода медоидов состоит в минимизации

$$\min_{u_{ik}} \sum_{k=1}^K \frac{\sum_{i, i'} u_{ik}^2 u_{i'k}^2 \delta_{ii'}}{\sum_i u_{ik}^2}.$$

По окончании процесса полагаем

$$C(i) = \operatorname{argmax}_k u_{ik} \quad (i = 1, 2, \dots, N).$$

### 9.3.5. Метод FOREL

Метод FOREL (от «формальный элемент») [Загоруйко–Елкина, 1967], похож на метод центров тяжести.

Он не требует знания количества кластеров, на которые нужно разбить множество объектов.

Наоборот, параметром алгоритма является число  $R$ , характеризующее близость точек внутри одного кластера.

Пусть задана точка  $x^{(0)}$  в  $\mathbf{R}^d$  и параметр  $R$ .

Найдем все объекты из тестовой выборки, попадающие в шар радиуса  $R$  с центром  $x^{(0)}$ .

Перенесем  $x^{(0)}$  в центр тяжести системы точек, найденных на предыдущем шаге.

Так будем продолжать до тех пор, пока  $x^{(0)}$  не перестанет изменяться.

После этого, если еще остались некластеризованные объекты, то повторим процесс сначала.

Точка  $x^{(0)}$  — это и есть «формальный элемент». Формальным он назван потому, что может не принадлежать множеству рассматриваемых объектов.

**begin**

Положить  $U \leftarrow \{x^{(1)}, x^{(2)}, \dots, x^{(N)}\}$  (множество некластеризованных объектов)

Инициализировать множество кластеров  $C \leftarrow \emptyset$

Инициализировать счетчик количества кластеров  $k \leftarrow 0$

**while**  $U \neq \emptyset$

$k \leftarrow k + 1$

Выбрать в  $\mathbf{R}^d$  точку  $x_0$

**repeat**

Образовать кластер  $C_k \leftarrow \{i : \|x^{(i)} - x^{(0)}\| \leq R, i = 1, 2, \dots, N\}$

Найти центр тяжести  $x^{(0)} \leftarrow \frac{1}{|C_k|} \sum_{i \in C_k} x^{(i)}$

$U \leftarrow U \setminus C_k, C \leftarrow C \cup \{c_k\}$

**end**

**end**

Возвратить множество  $C$

**end**

### 9.3.6. DBSCAN

Density-based spatial clustering of applications with noise (DBSCAN)

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

Входные параметры:  $\varepsilon, M$

Точка  $x^{(i')}$  — ядерная (core)  $\Leftrightarrow \left| \left\{ i : \text{dist}(x^{(i')}, x^{(i)}) \leq \varepsilon \right\} \right| \geq M$

Точка  $x^{(i')}$  — граничная (border)  $\Leftrightarrow x^{(i')}$  не ядерная и  $\exists$  ядерная  $x^{(i'')} : \text{dist}(x^{(i')}, x^{(i'')}) \leq \varepsilon$

Точка  $x^{(i')}$  — шум (noise)  $\Leftrightarrow$  не ядерная и не граничная (outlier)

Точка  $x^{(i')}$  достижима из ядерной точки  $x^{(i)}$   $\Leftrightarrow$  существует последовательность

$$x^{(i_0)} = x^{(i)}, x^{(i_1)}, \dots, x^{(i_{s-1})}, x^{(i_s)} = x^{(i')},$$

такая, что точки  $x^{(i_1)}, \dots, x^{(i_{s-1})}$  — ядерные и  $\text{dist}(x^{(i_\ell)}, x^{(i_{\ell+1})}) \leq \varepsilon$  ( $\ell = 0, 1, \dots, s - 1$ )

$x^{(i)}$  и  $x^{(i')}$  принадлежат одному кластеру  $\Leftrightarrow \exists$  ядерная точка  $x^{(i'')}$ , такая, что из  $x^{(i')}$  достижима как  $x^{(i)}$ , так и  $x^{(i')}$

Шумовые точки не принадлежат ни одному кластеру

**foreach**  $x \in D$

**if**  $x$  — Посещенная

**continue**

  пометить  $x$  как Посещенная

$P \leftarrow \text{Neighbourhood}(x, \varepsilon)$  ( $\varepsilon$ -окрестность точки  $x$  включая  $x$ )

**if**  $|P| < M$

    пометить  $x$  как Шум

**else**

    отнести  $x$  к новому кластеру

**foreach**  $x' \in P$

**if**  $x'$  не Посещенная

        пометить  $x'$  как Посещенная

$P' = \text{Neighbourhood}(x', \varepsilon)$

**if**  $|P'| \geq M$

$P \leftarrow P \cup P'$

**if**  $x'$  не относится ни к какому кластеру, то

        добавить  $x'$  к текущему кластеру

## 9.4. ЕМ-алгоритм

Алгоритм *Expectation–maximization* («ожидание–максимизация») — «сглаженный» вариант алгоритма центров тяжести.

Имеется смесь  $K$  нормальных распределений (пока для простоты одномерных):

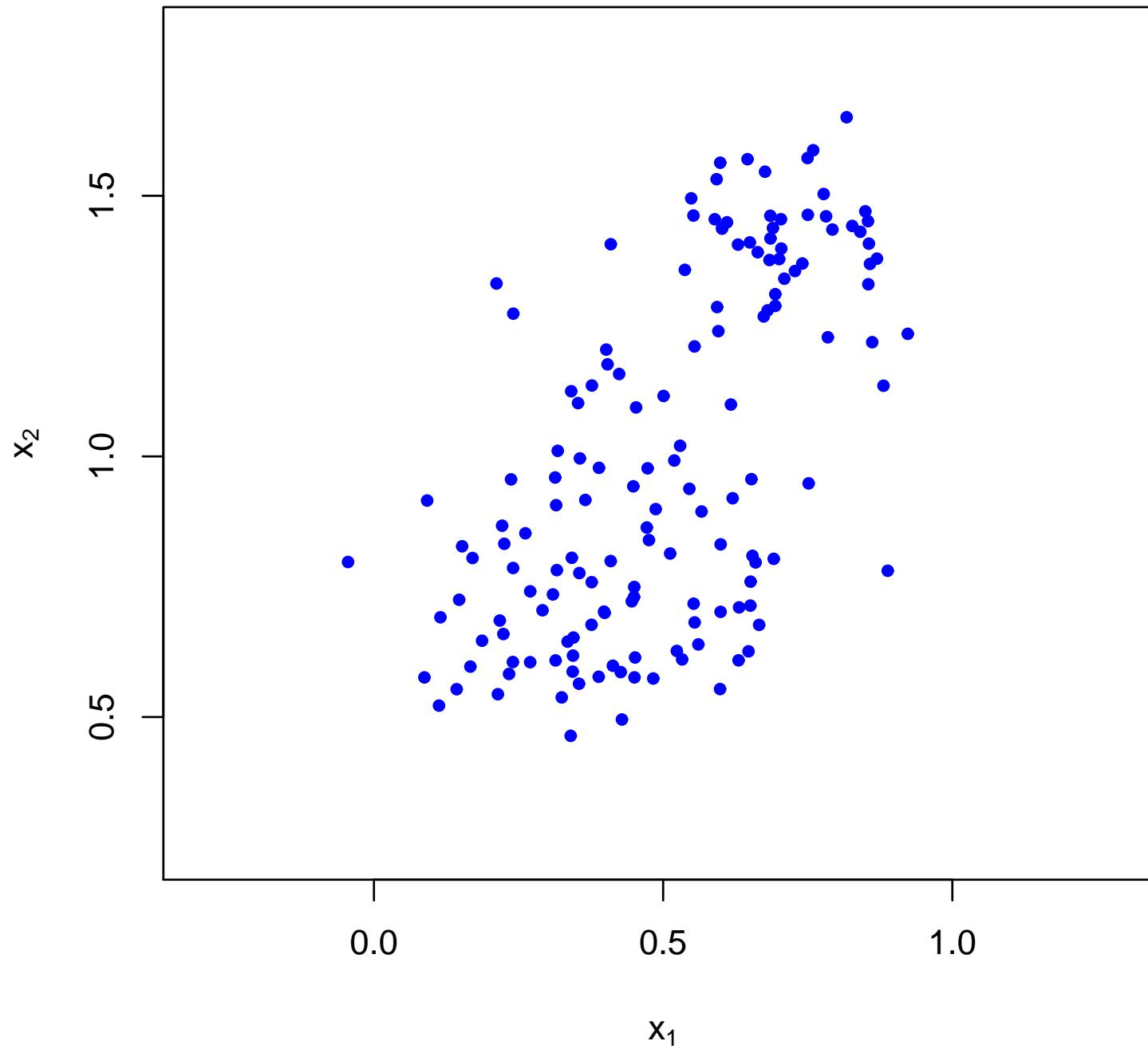
$$X = \begin{cases} X_1, & \text{если } Y = 1, \\ X_2, & \text{если } Y = 2, \\ \dots \\ X_K, & \text{если } Y = K, \end{cases} \quad \begin{array}{ll} X_1 \sim N(\mu_1, \sigma_1) \\ X_2 \sim N(\mu_2, \sigma_2) \\ \dots \\ X_K \sim N(\mu_K, \sigma_K) \end{array}$$

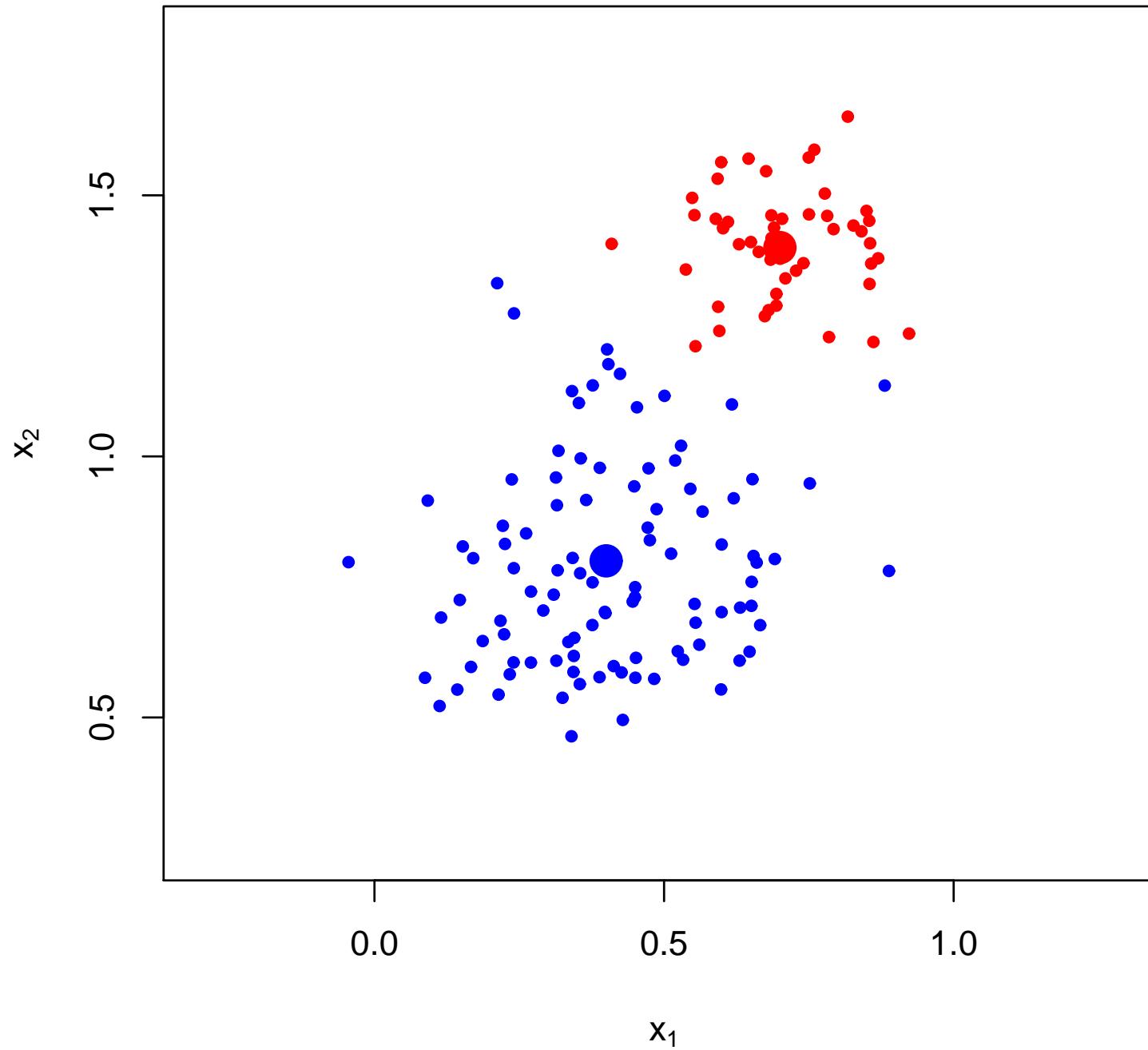
где  $Y$  — скрытая (неизвестная) случайная переменная (переменная переключения).

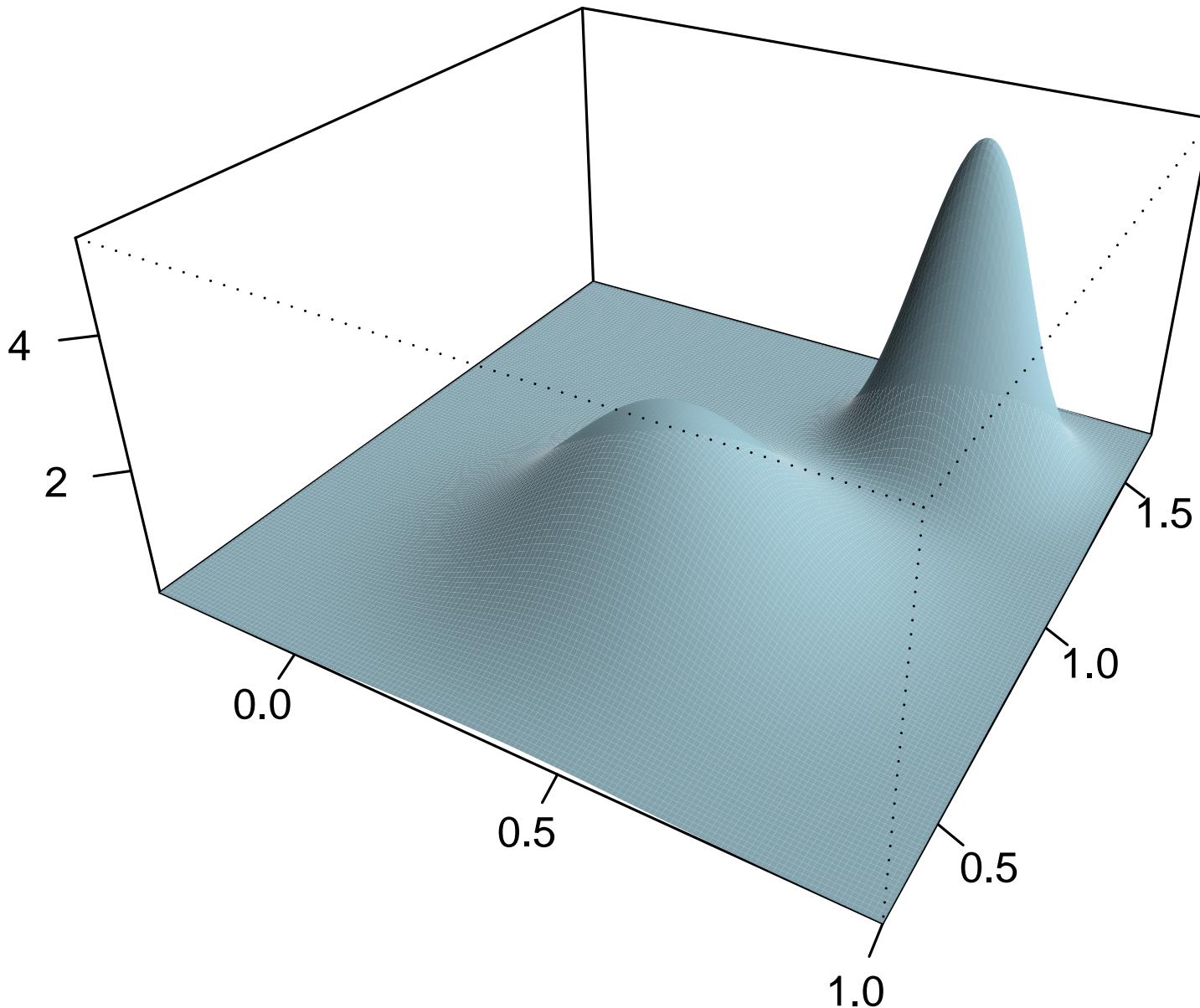
$$p_X(x) = \sum_{k=1}^K \delta_k p_{X_k}(x), \quad p_{X_k}(x) = p_{X|Y}(x) = \varphi(x, \mu_k, \sigma_k), \quad \Pr\{Y = k\} = \delta_k,$$

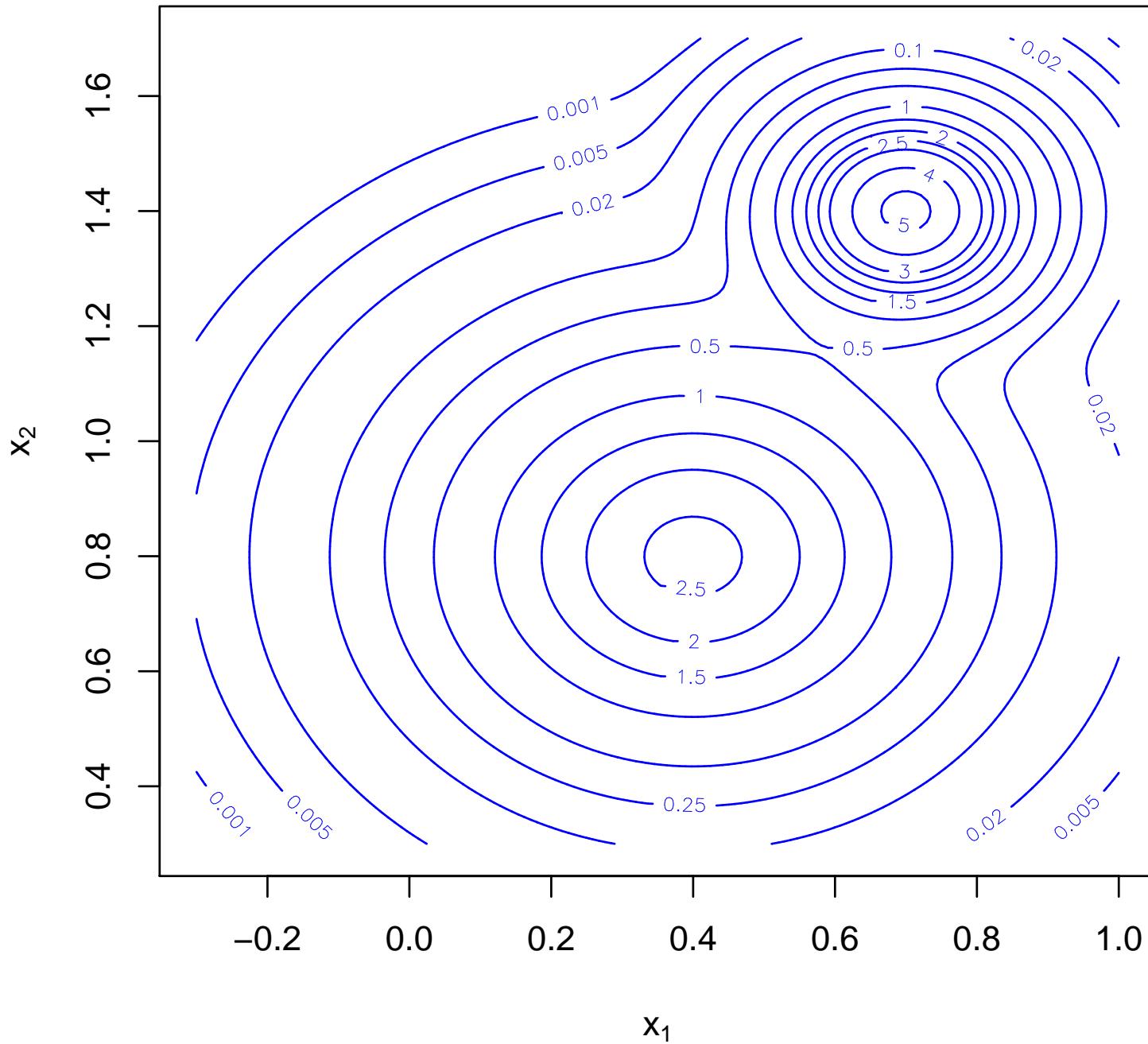
$$\varphi(x, \mu, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

Необходимо по выборке  $x^{(1)}, x^{(2)}, \dots, x^{(N)}$  восстановить  $\mu_1, \dots, \mu_K, \sigma_1, \dots, \sigma_K, \delta_1, \dots, \delta_K$ .









*Ожидание – Expectation step.*

Если знаем параметры  $\mu_1, \dots, \mu_K, \sigma_1 \dots, \sigma_K, \delta_1, \dots, \delta_K$ , то можем найти апостериорные вероятности (вероятности, что  $i$ -й объект принадлежит  $k$ -й компоненте смеси):

$$\gamma_{ik} = \Pr(k|x^{(i)}) = \frac{\Pr\{k\} p(x^{(i)}|k)}{p(x^{(i)})} = \frac{\delta_k \varphi(x^{(i)}, \mu_k, \sigma_k)}{\sum_{\ell=1}^K \delta_\ell \varphi(x^{(i)}, \mu_\ell, \sigma_\ell)} \quad (i = 1, \dots, N; k = 1, \dots, K).$$

*Максимизация – Maximization step.*

Если знаем апостериорные вероятности  $\gamma_{ik}$ , то можем оценить параметры  $\mu_1, \dots, \mu_K, \sigma_1 \dots, \sigma_K, \delta_1, \dots, \delta_K$ :

$$\mu_k = \frac{1}{\zeta_k} \sum_{i=1}^N \gamma_{ik} x^{(i)}, \quad \sigma_k^2 = \frac{1}{\zeta_k} \sum_{i=1}^N \gamma_{ik} (x^{(i)} - \mu_k)^2, \quad \delta_k = \frac{1}{N} \sum_{i=1}^N \gamma_{ik}, \quad \zeta_k = \sum_{i=1}^N \gamma_{ik}.$$

EM-алгоритм заключается в итерационном повторении этих шагов.

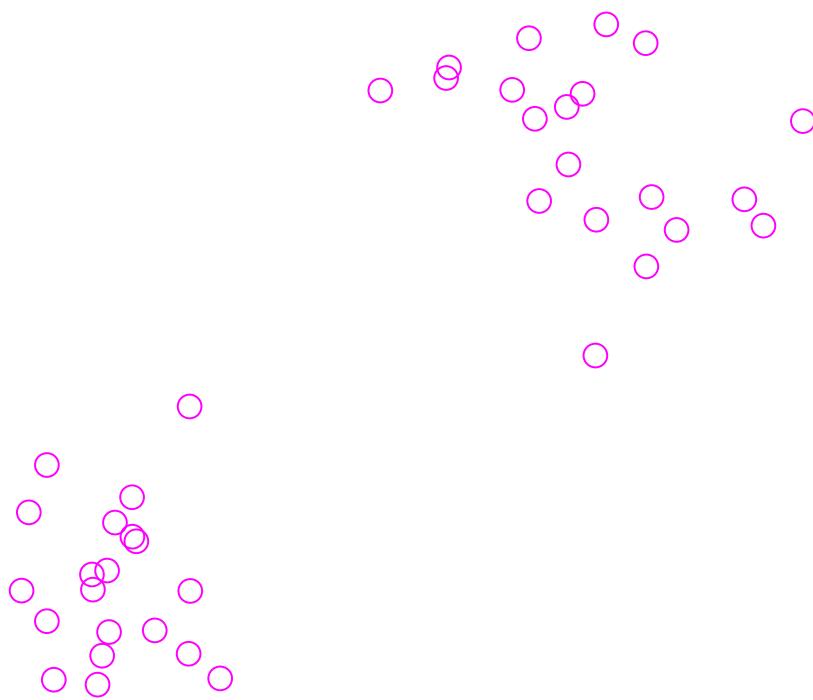
В случае смеси  $d$ -мерных для пересчета матрицы ковариации используем формулу (остальное также):

$$\sum_k = \frac{1}{\zeta_k} \sum_{i=1}^N \gamma_{ik} (x^{(i)} - \mu_k)(x^{(i)} - \mu_k)^\top \quad (k = 1, 2, \dots, K).$$

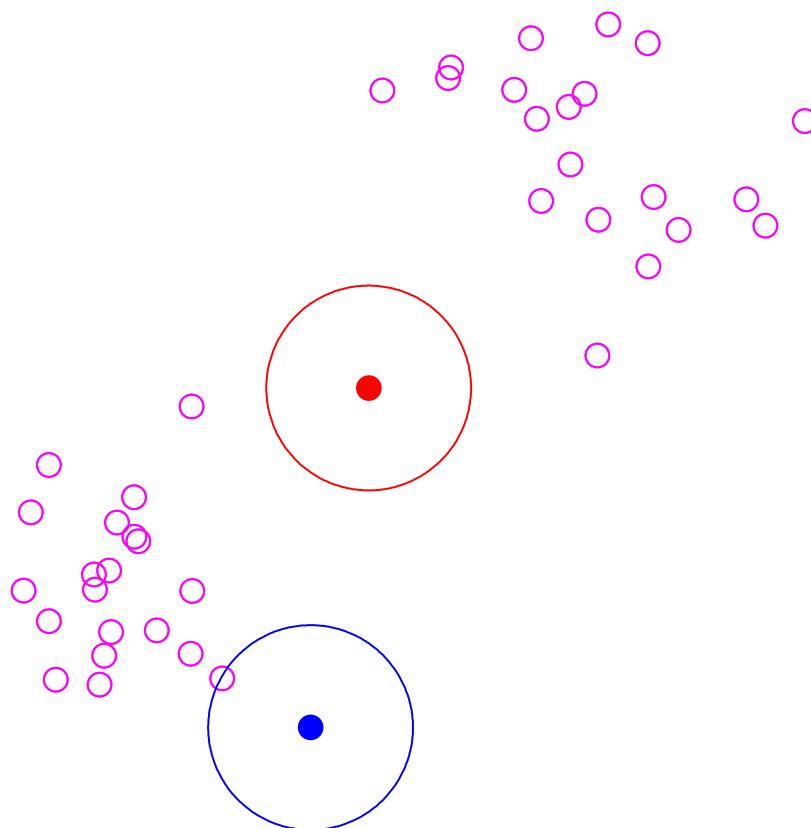
При  $\Sigma_k = 0$  получаем метод центров тяжести.

**Упражнение 9.2** Измените формулы для вычисления  $\sigma_k^2$  и  $\Sigma_k$ , чтобы они давали несмещенные оценки для дисперсии и матрицы ковариации.

Иллюстрация,  $K = 2$ :

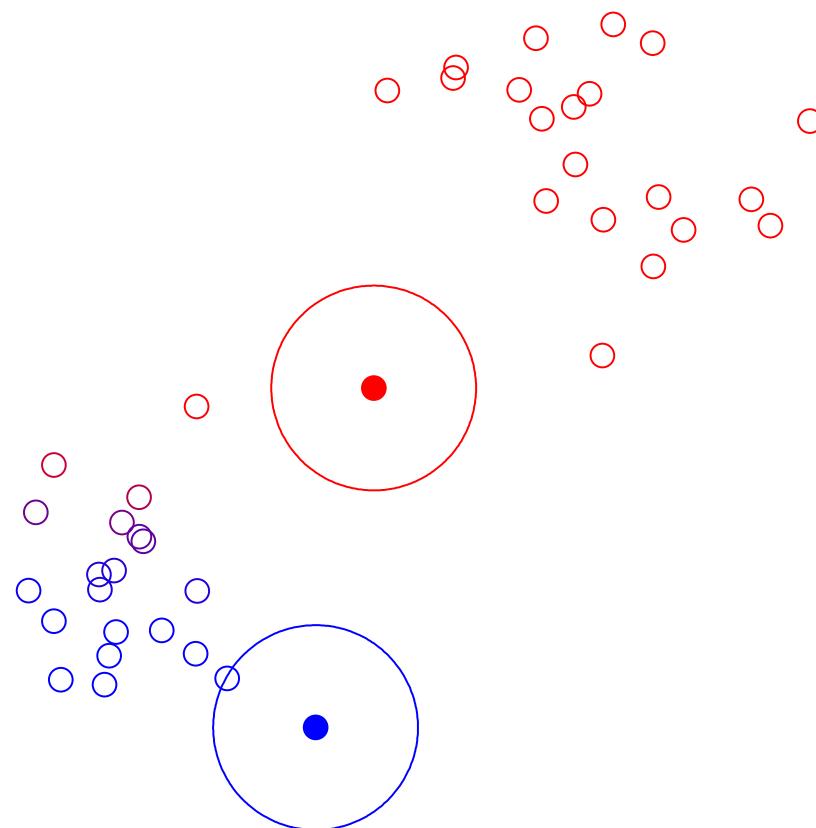


Случайно выбираем параметры компонентов смесей и априорные вероятности:



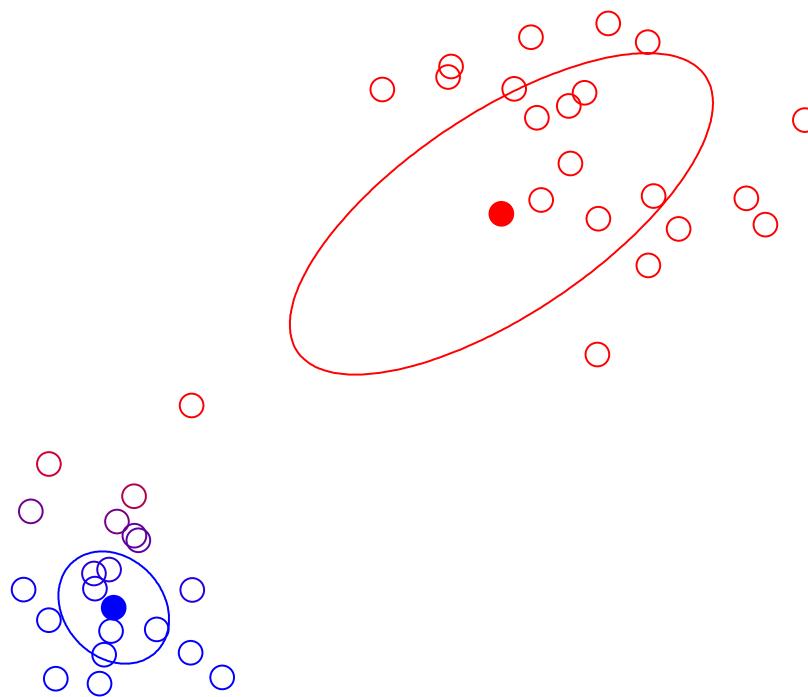
Вычисляем апостериорные вероятности:

Expectation step



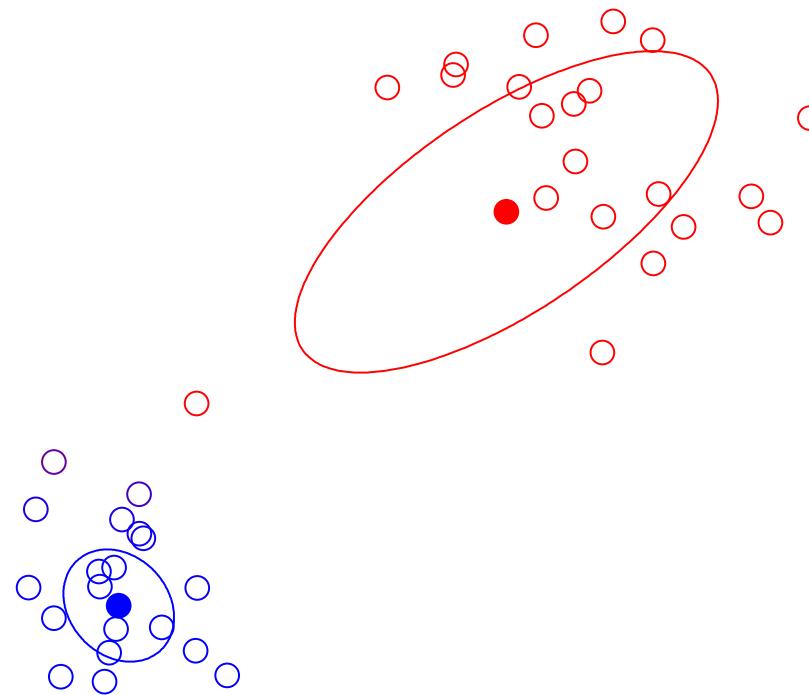
Обновляем параметры компонентов смесей и априорные вероятности:

Maximization step



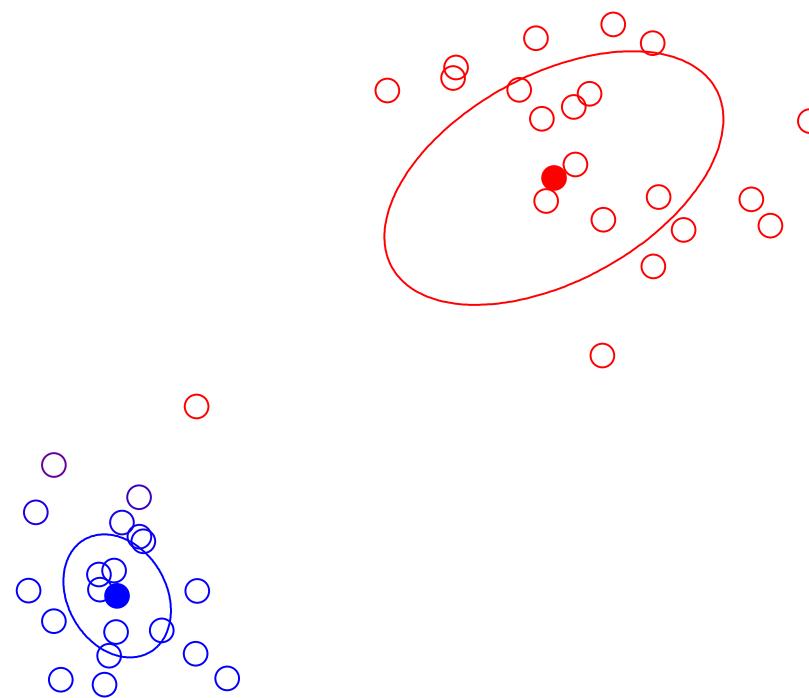
Вычисляем апостериорные вероятности:

Expectation step



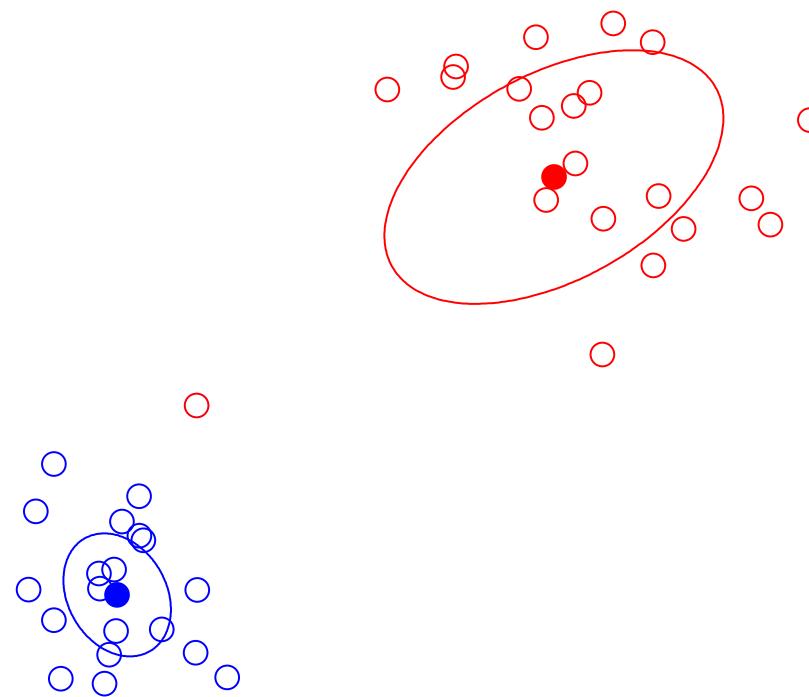
Обновляем параметры компонентов смесей и априорные вероятности:

Maximization step



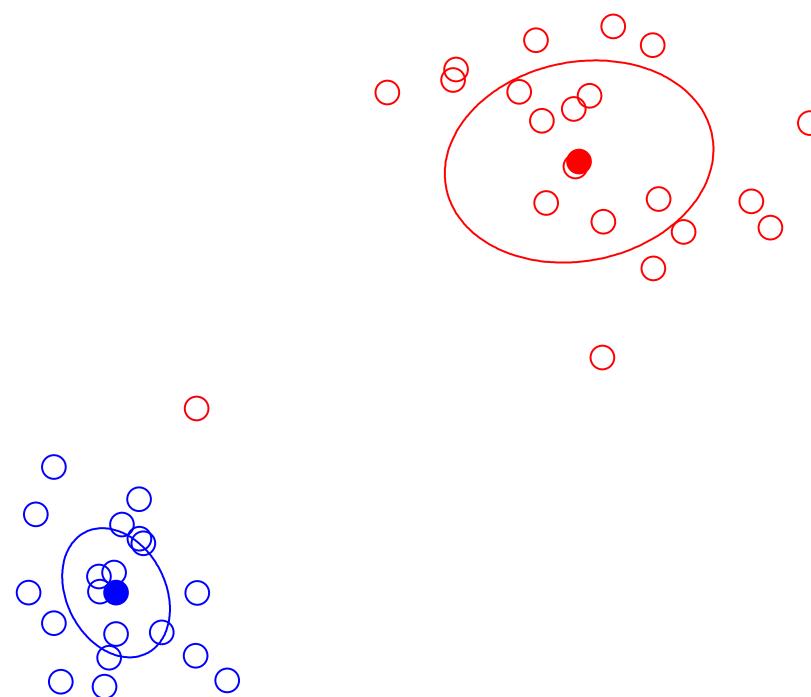
Вычисляем апостериорные вероятности:

Expectation step



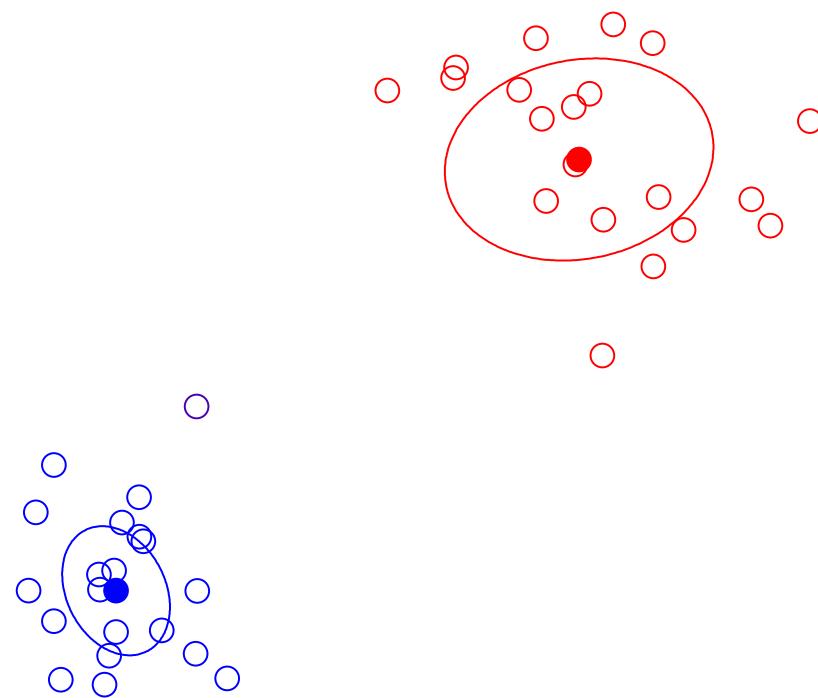
Обновляем параметры компонентов смесей и априорные вероятности:

Maximization step



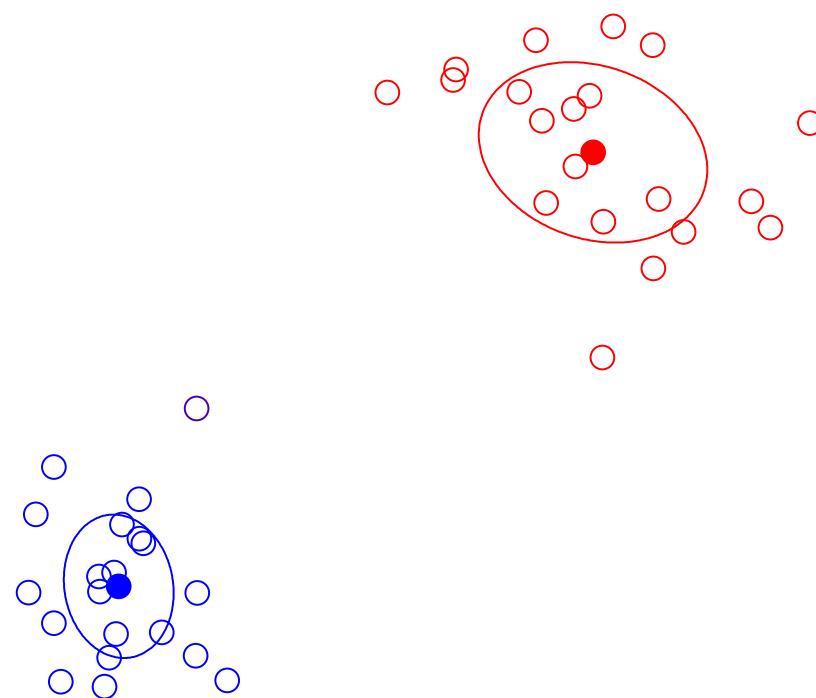
Вычисляем апостериорные вероятности:

Expectation step



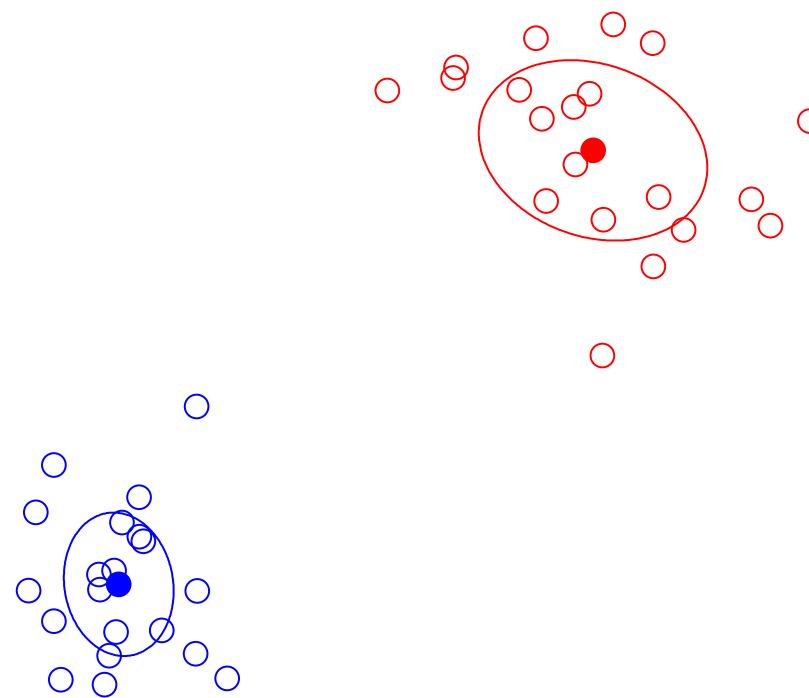
Обновляем параметры компонентов смесей и априорные вероятности:

Maximization step



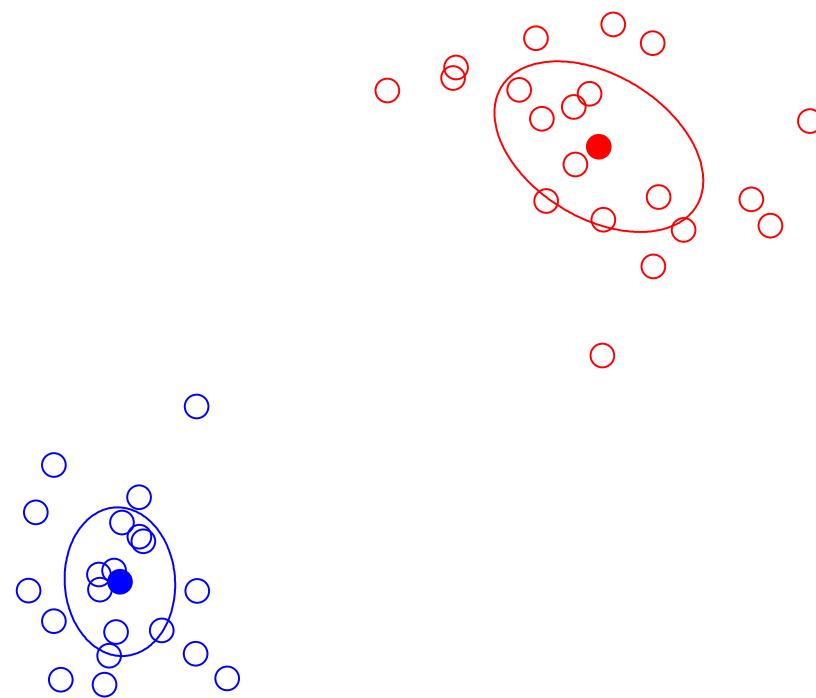
Вычисляем апостериорные вероятности:

Expectation step



Обновляем параметры компонентов смесей и априорные вероятности:

Maximization step



### **9.4.1. Варианты**

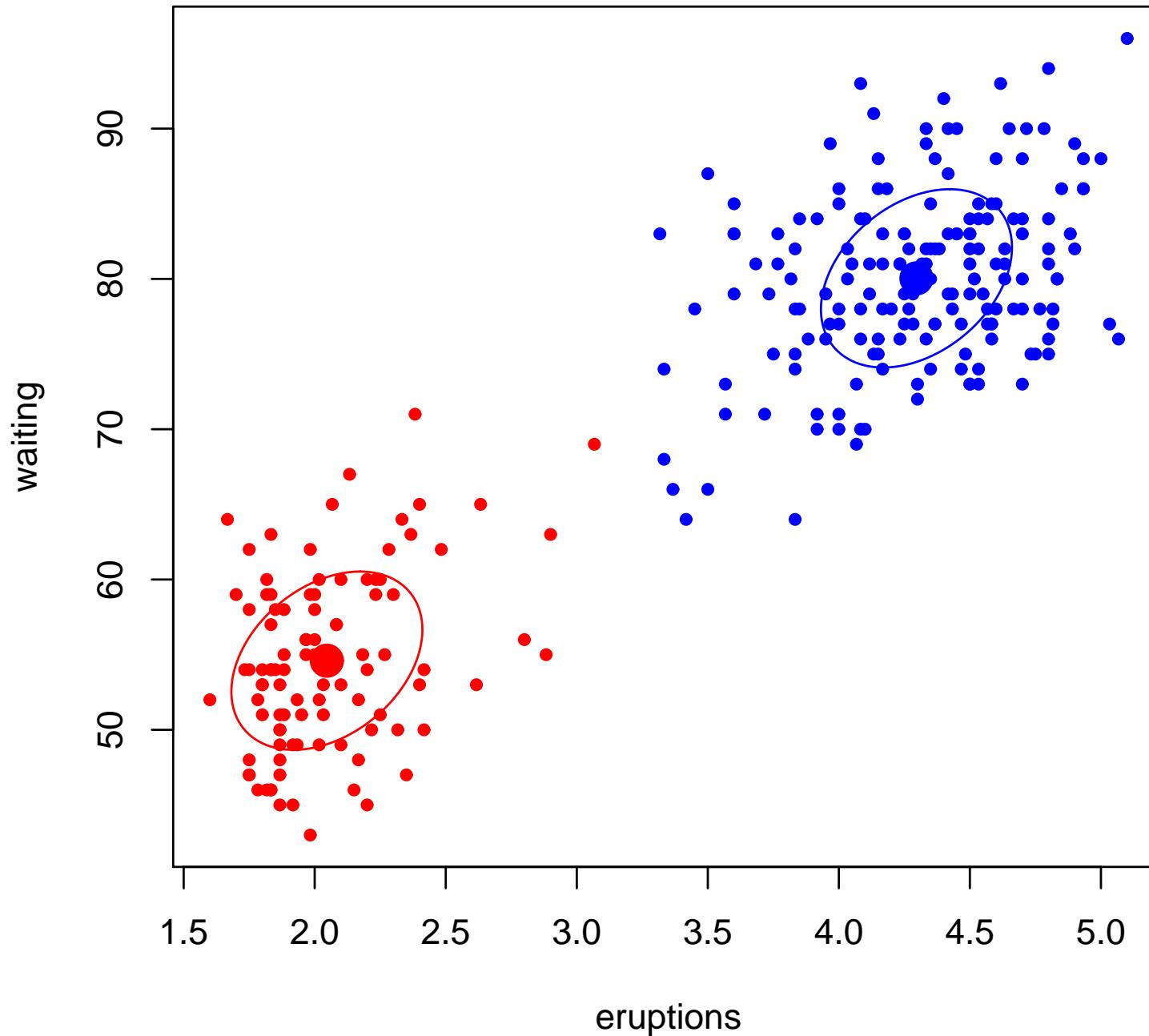
Стандартное отклонение  $\sigma_k$  может быть:

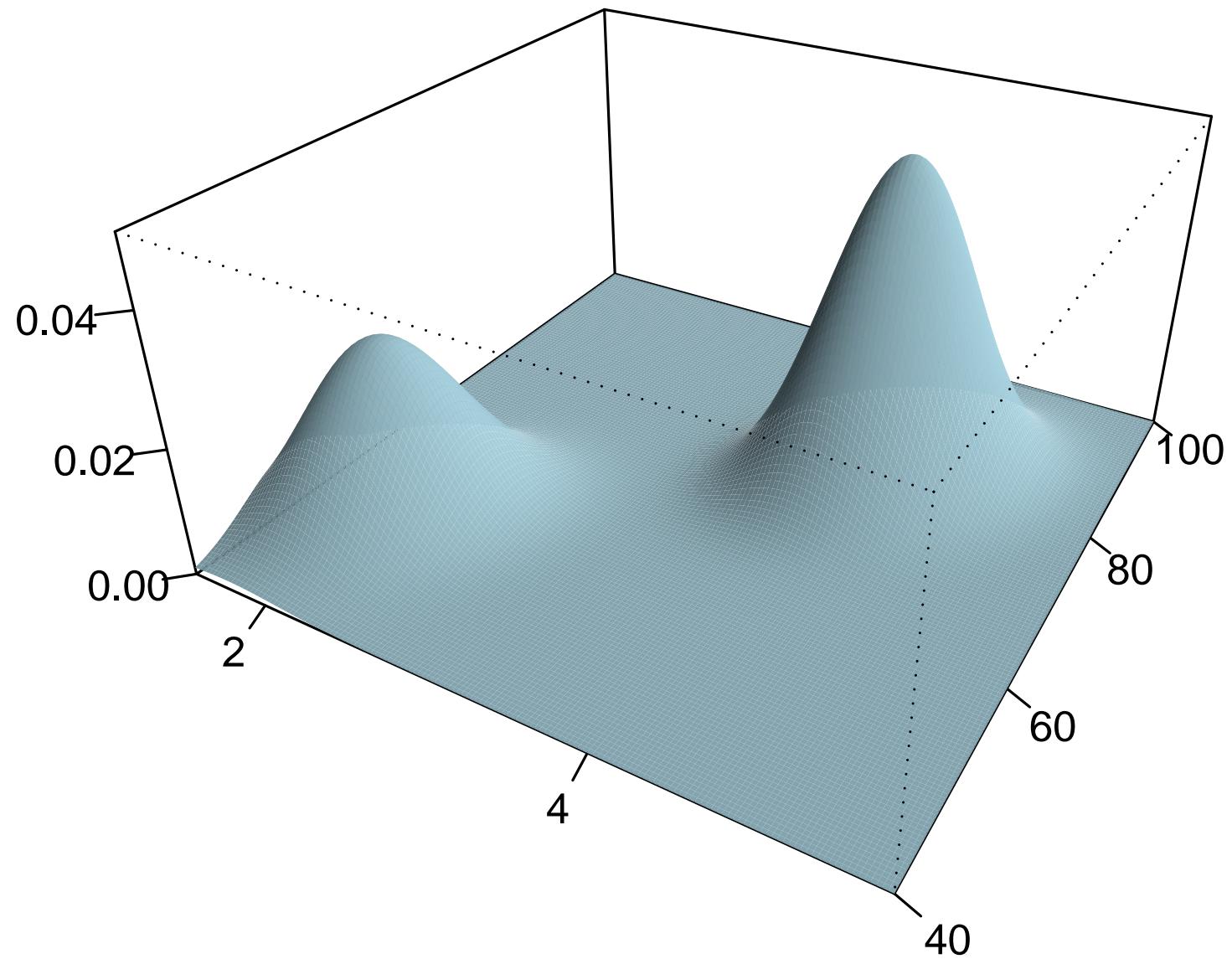
- (а) Единым для всех:  $\sigma_k = \sigma$
- (б) Разным

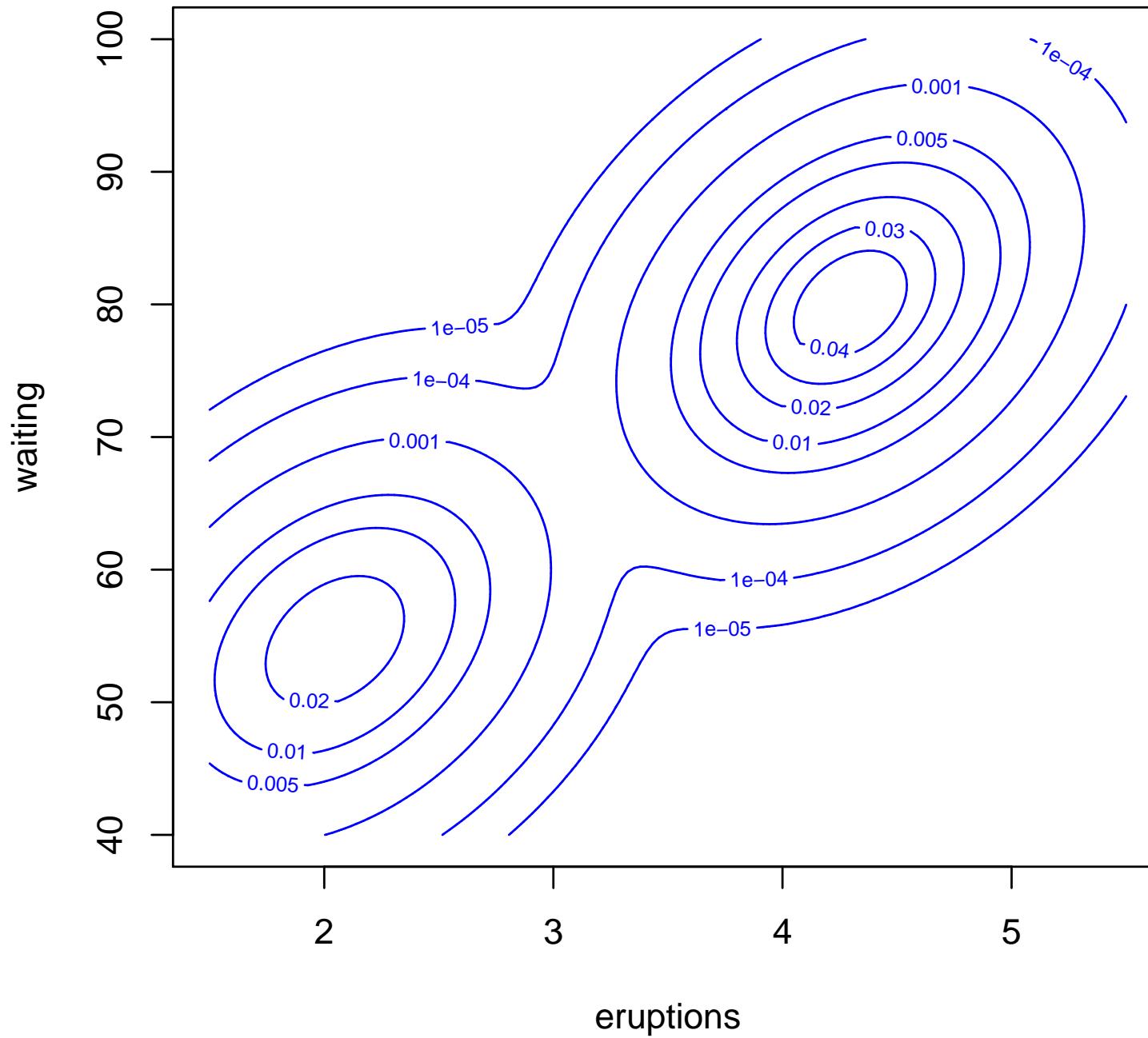
$$d \geq 2$$

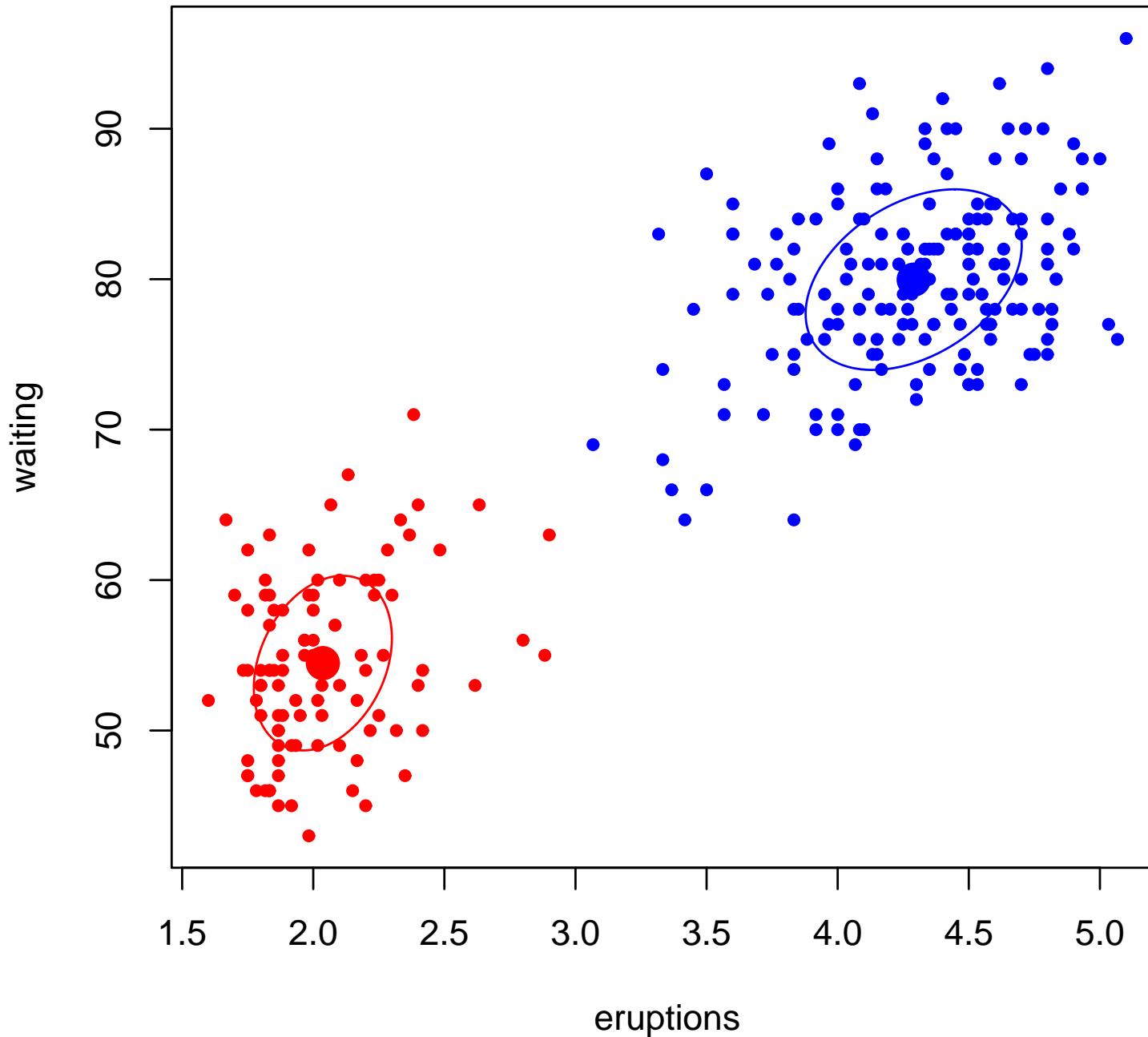
Матрица ковариации  $\Sigma_k$  может быть:

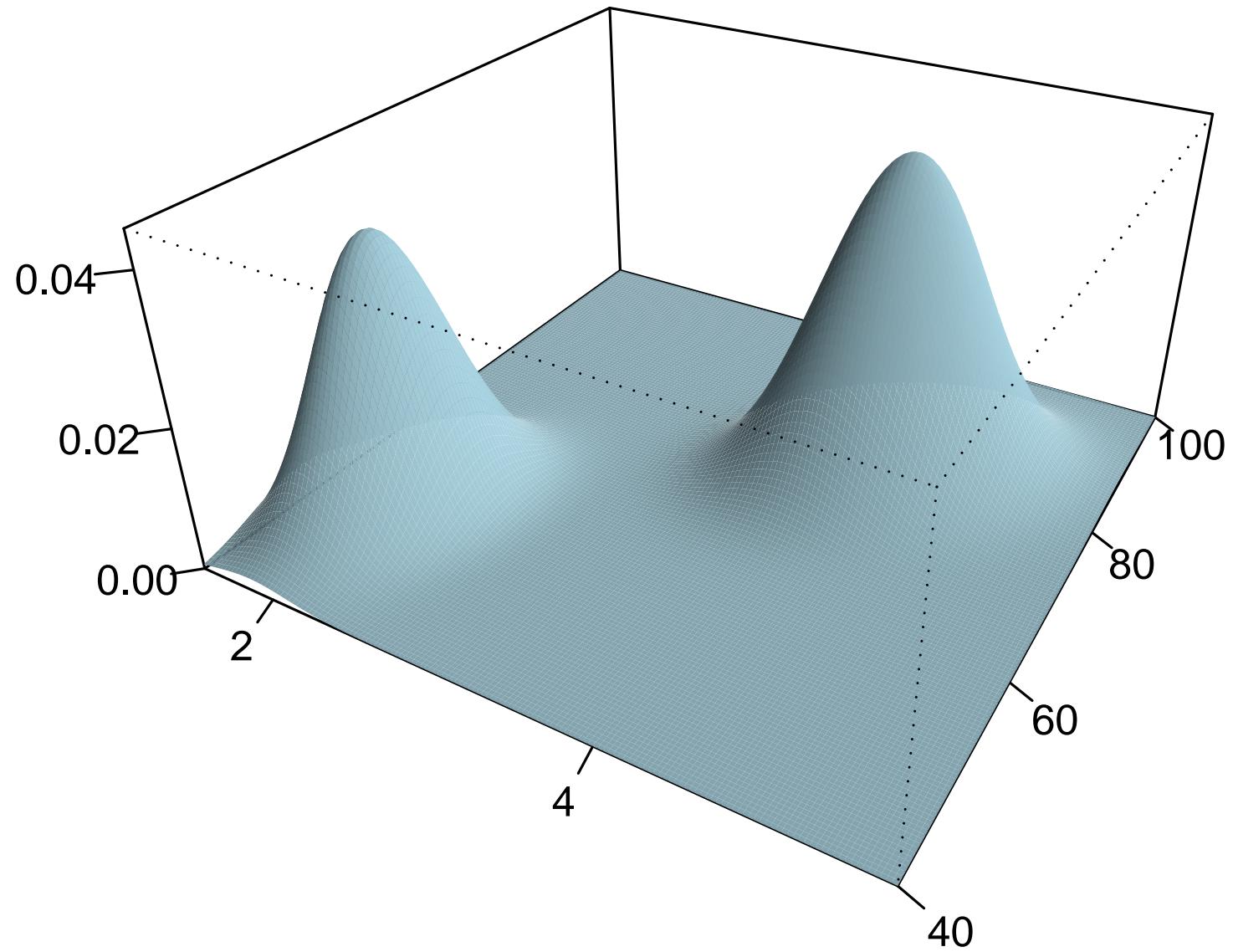
1. (а) Единой для всех:  $\Sigma_k = \Sigma$  ( $k = 1, 2, \dots, K$ )  
(б) Разной
2. (а) Скалярной:  $\Sigma_k = \lambda_k I$   
(б) Диагональной:  $\Sigma_k = D_k$   
(в) Произвольной (положительно определенной):  $\Sigma_k$

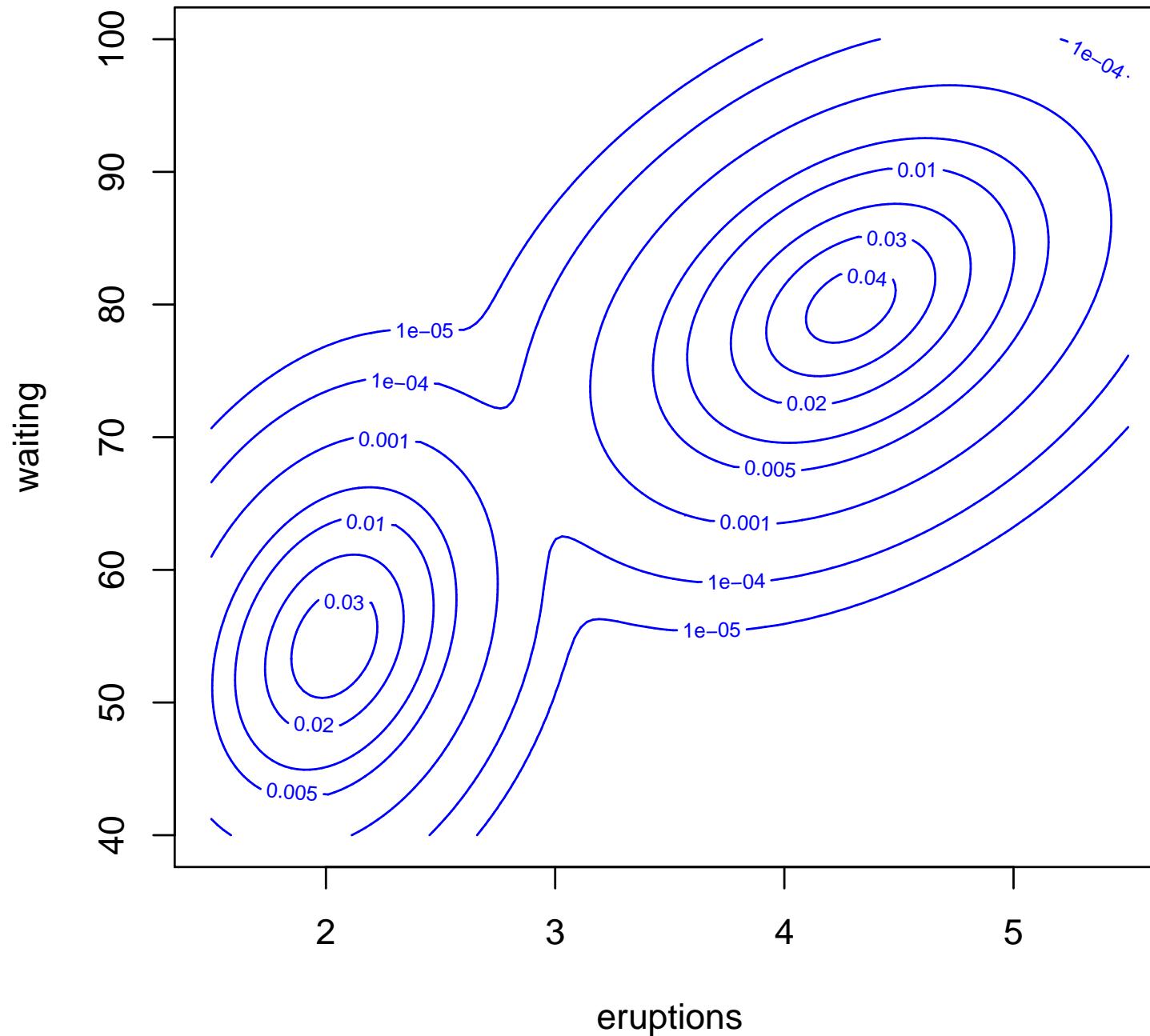


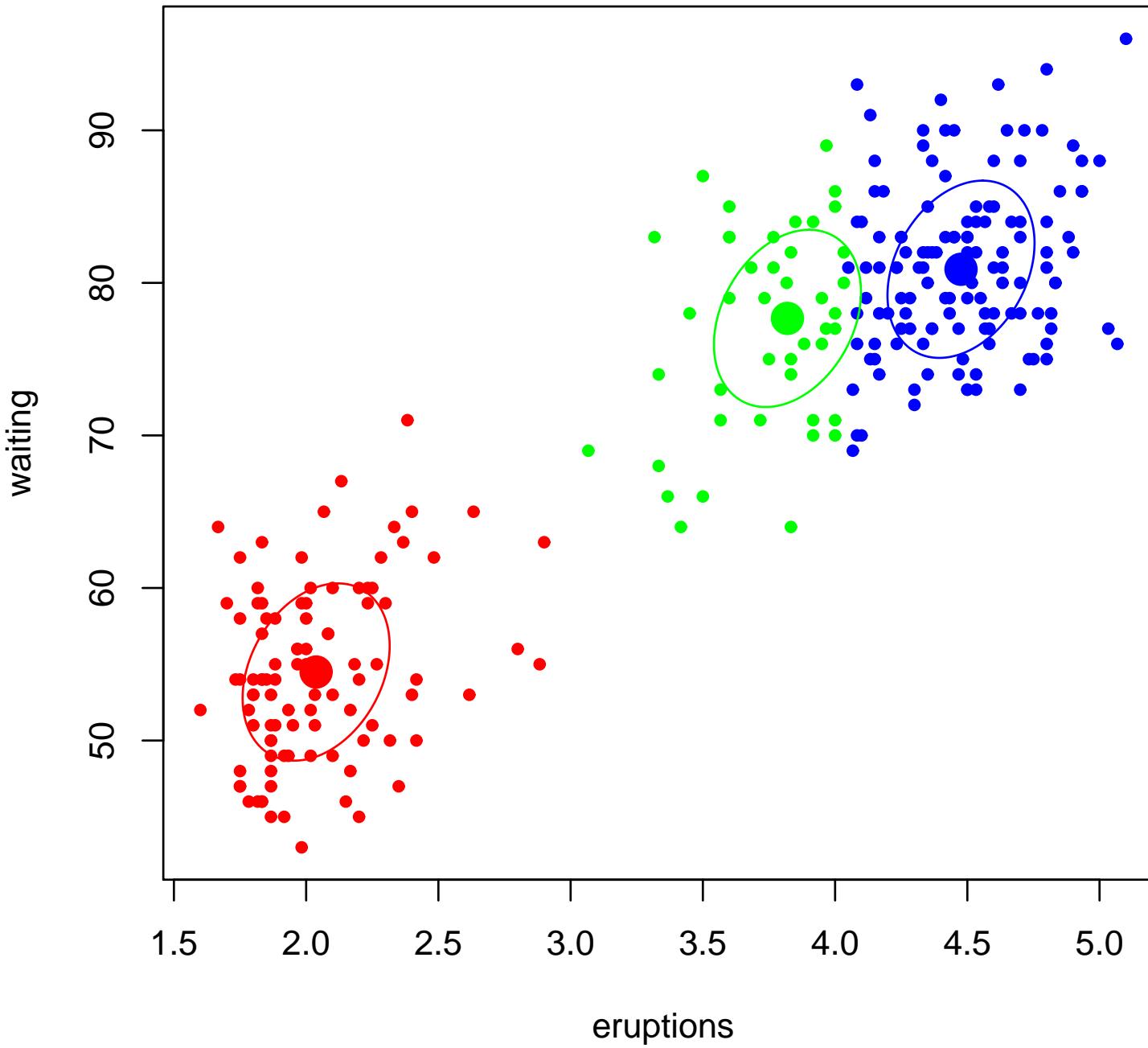


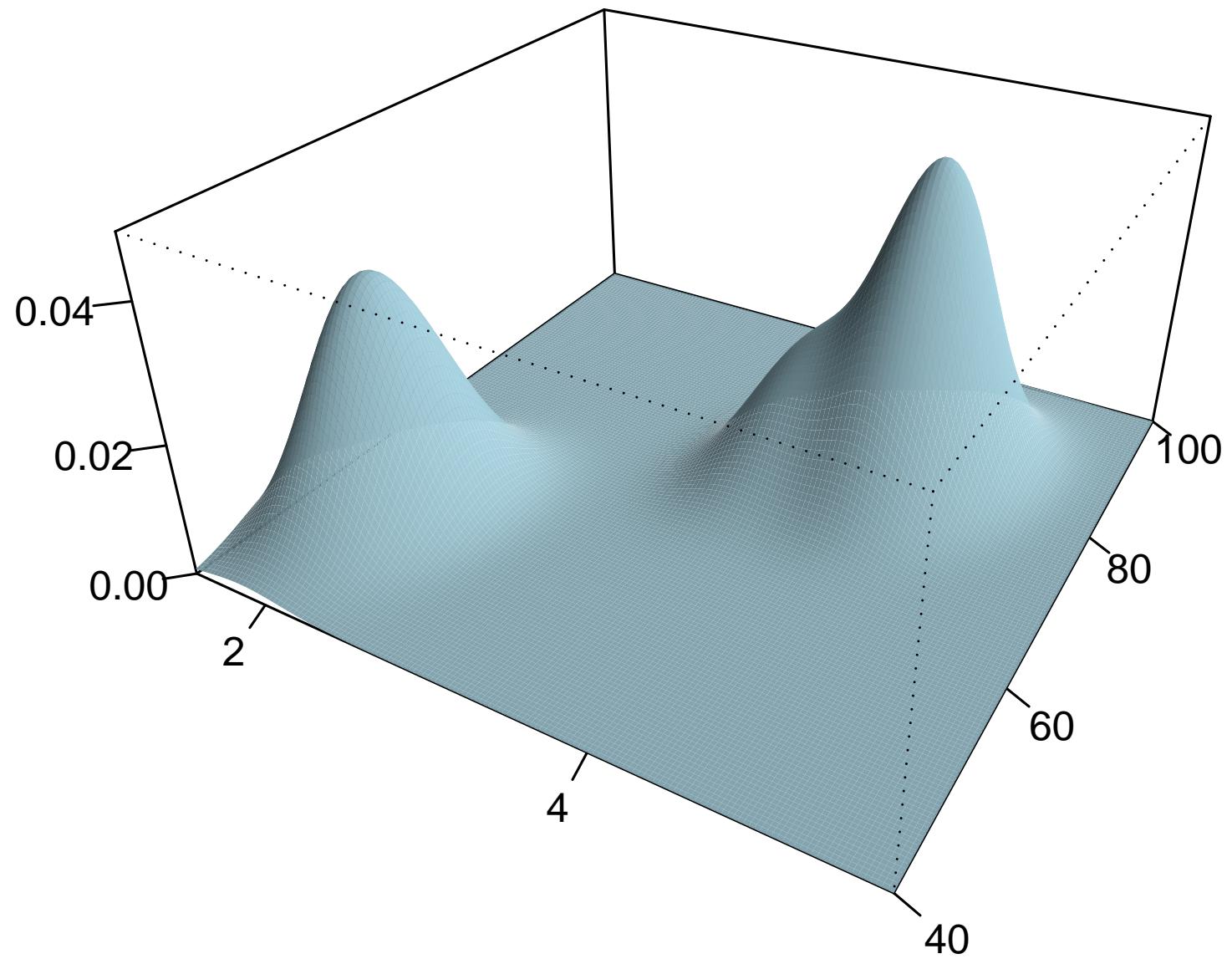


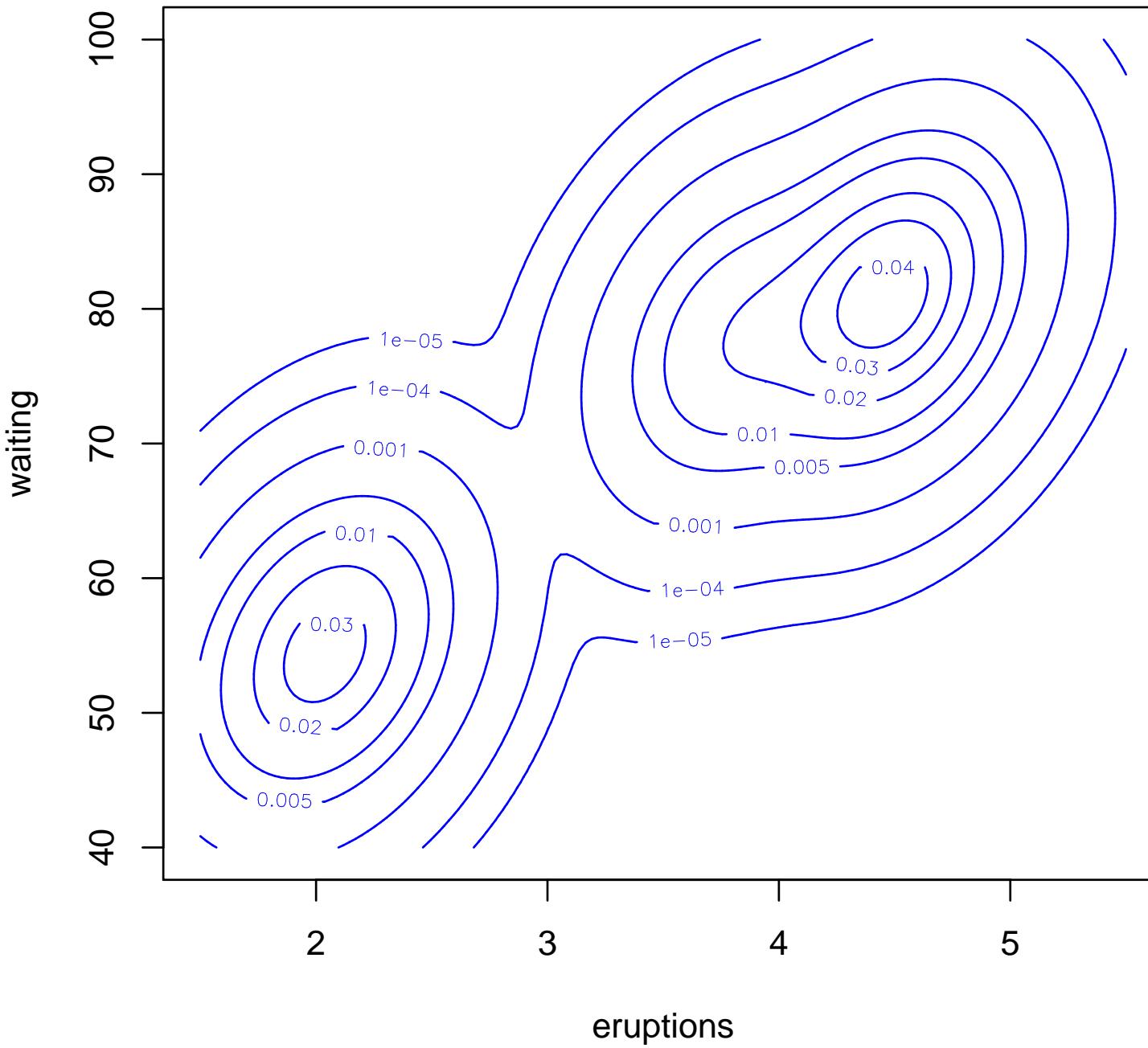


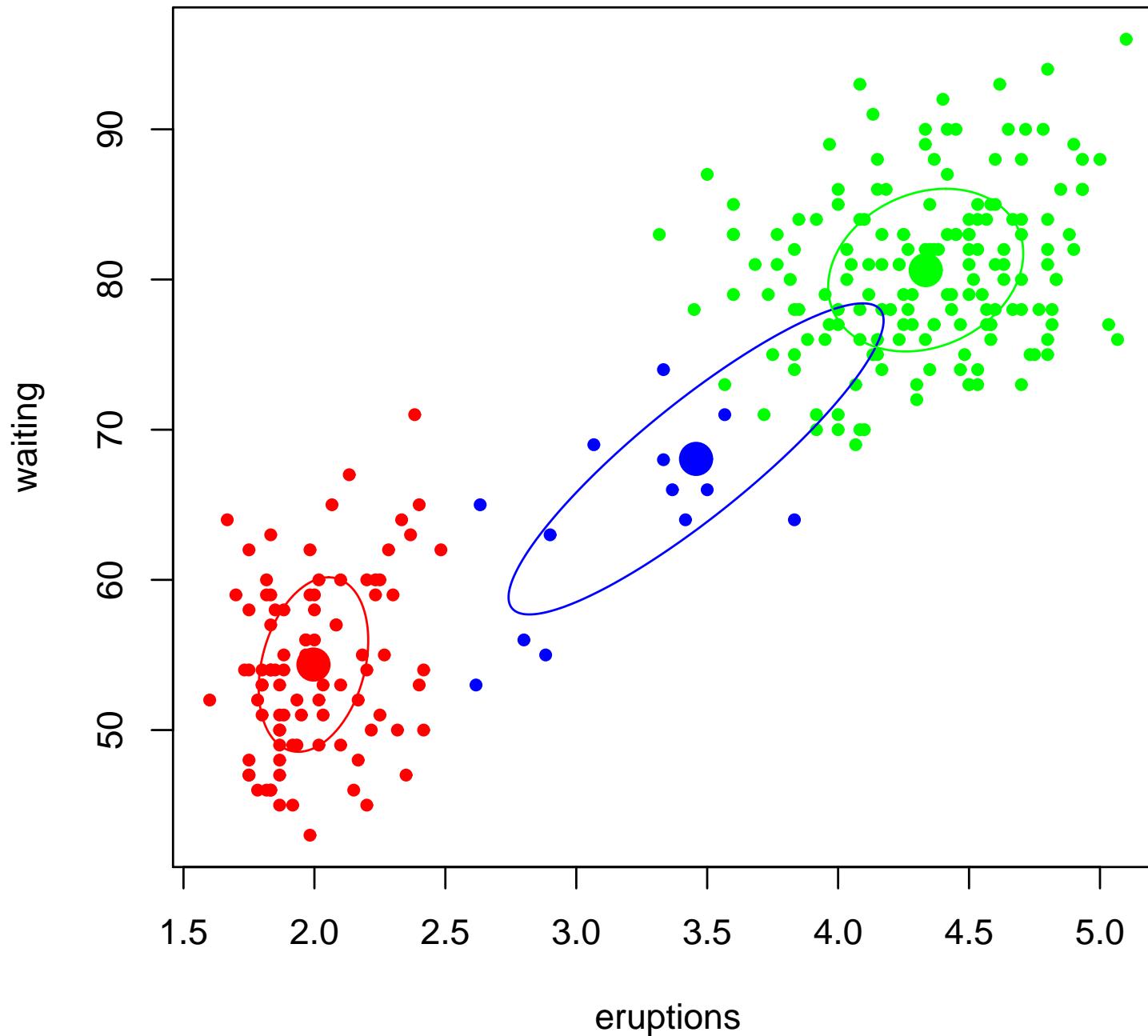


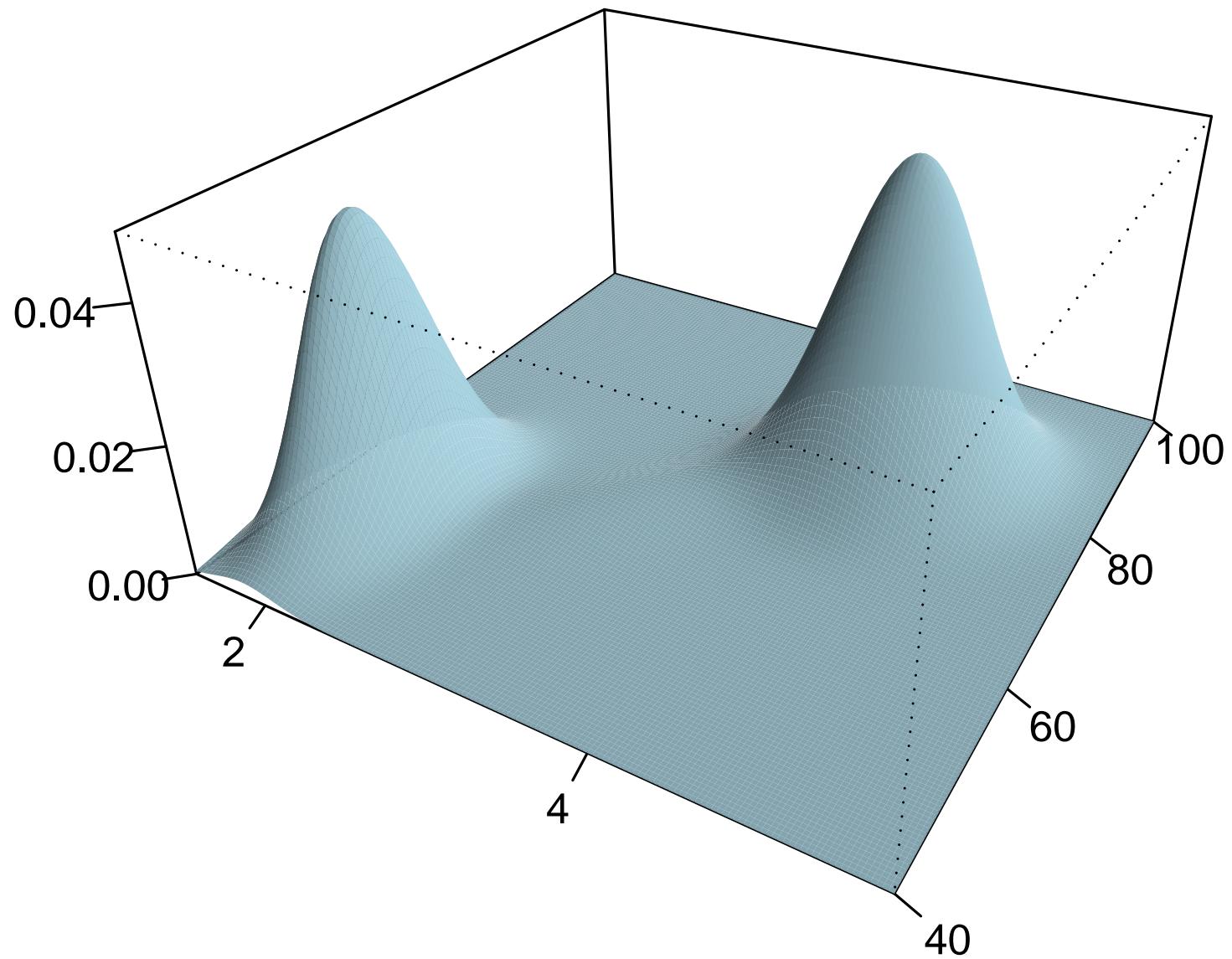


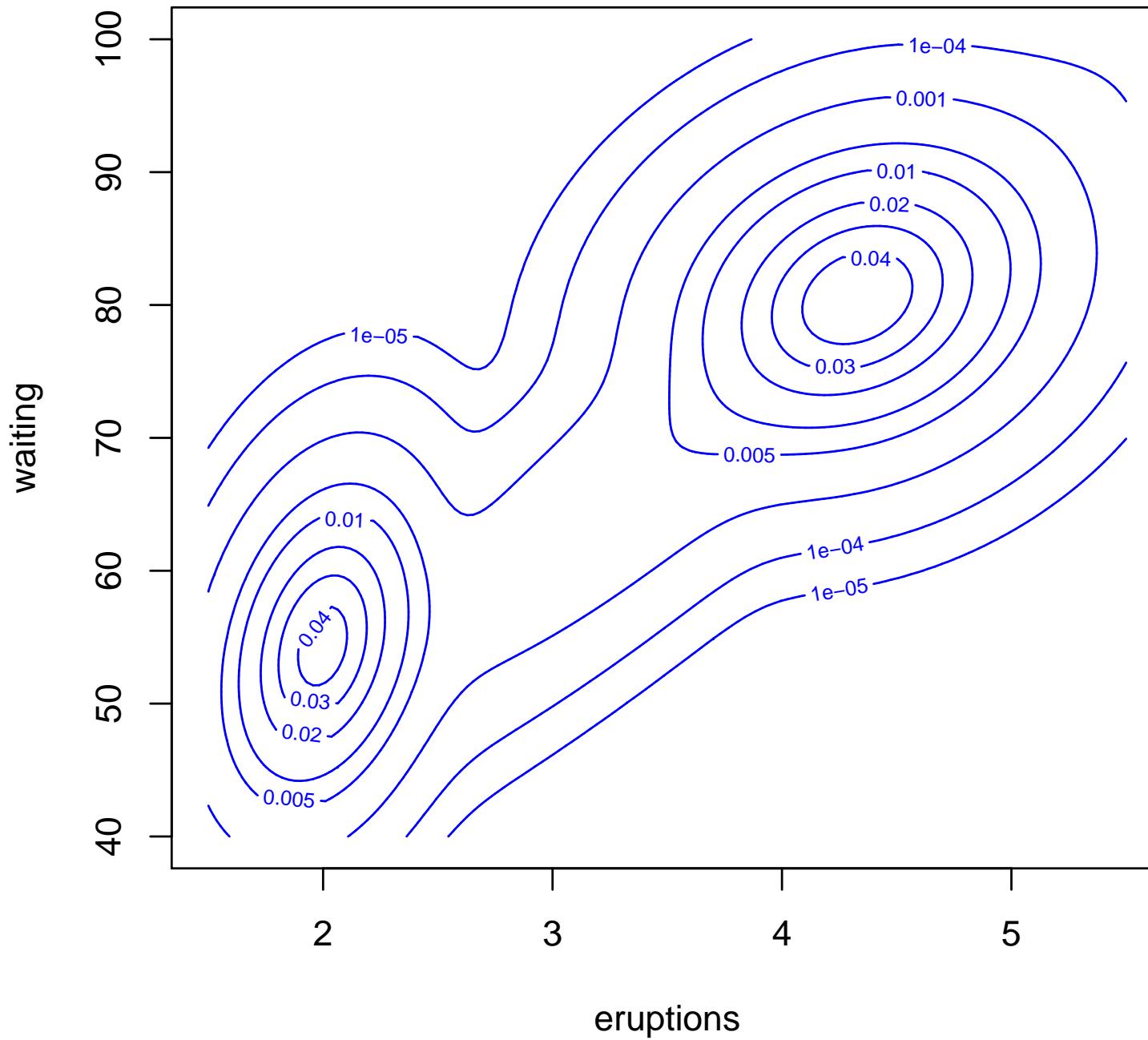


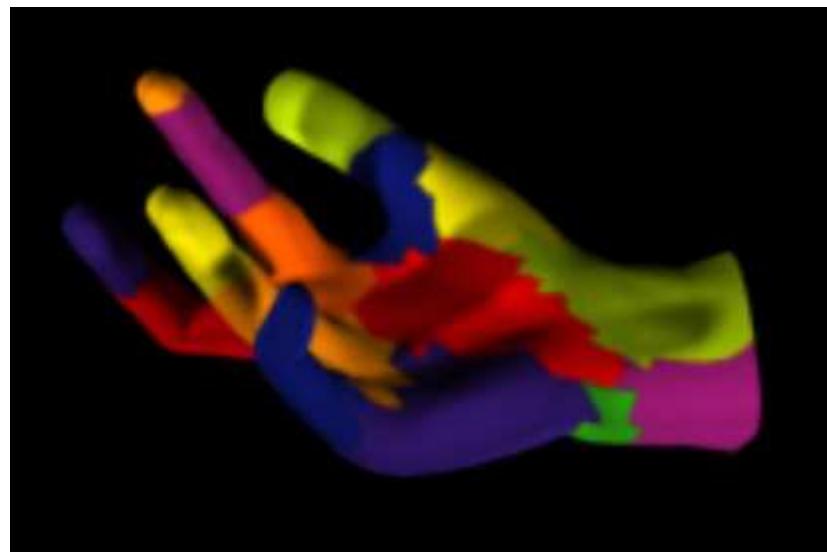
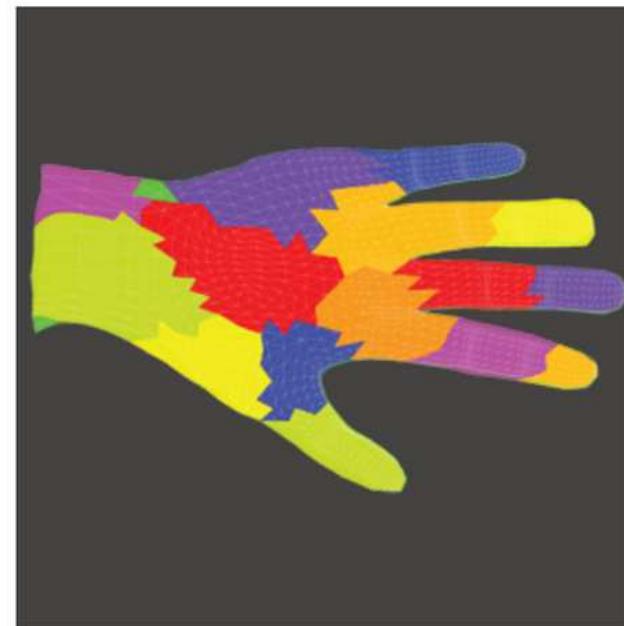




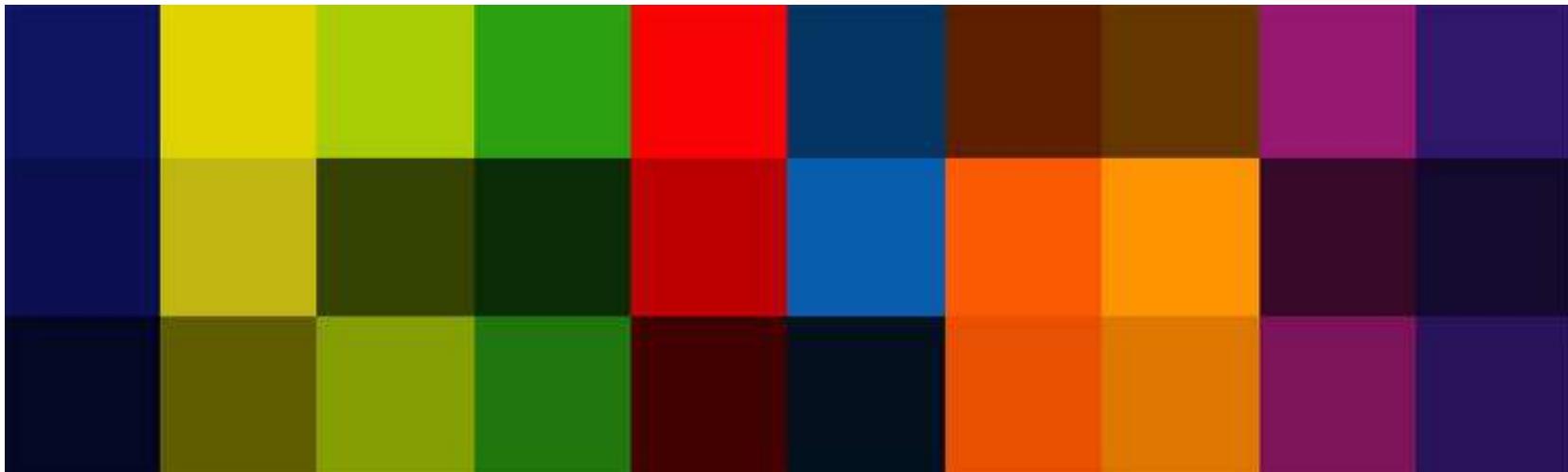








Цвета, соответствующие центрам каждой компоненты смеси нормальных распределений. В RGB:



В HSV (Hue, Saturation, Value — тон, насыщенность, значение):



## 9.4.2. ЕМ. Общий случай

Наблюдаемые значения:

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

Скрытые переменные:

$$Z = (Z^{(1)}, Z^{(2)}, \dots, Z^{(N)})$$

Логарифмическая функция правдоподобия:  $\ell(\theta, x)$

Логарифмическая функция правдоподобия, если  $z^{(i)}$  известны:  $\ell_0(\theta, x, z)$

**begin**

Установить начальные значения для  $\widehat{\theta}$ .

**repeat**

*Expectation step*

Вычислить  $Q(\theta', \widehat{\theta}) \leftarrow E(\ell_0(\theta', x, z) | x, \widehat{\theta})$  как функцию от  $\theta'$

*Maximization step*

$$\widehat{\theta} \leftarrow \operatorname{argmax}_{\theta'} Q(\theta', \widehat{\theta})$$

**end**

**end**

Можно показать, что

$$\ell(\theta', x, z) \geq \ell(\theta, x, z)$$

## 9.5. Иерархическая кластеризация (таксономия)

Алгоритмы *иерархической кластеризации*, или *таксономии*, находят иерархическое представление, такое, что кластеры на каждом уровне получаются объединением кластеров на более низком уровне.

Таким образом, речь идет о кластерах кластеров.

Можно сказать по-другому: кластеры на более низком уровне получаются дроблением кластеров на более высоком уровне.

В вершине этой классификации мы имеем один кластер, включающий все объекты.

На низшем уровне имеем  $N$  кластеров, каждый из которых включает один объект.

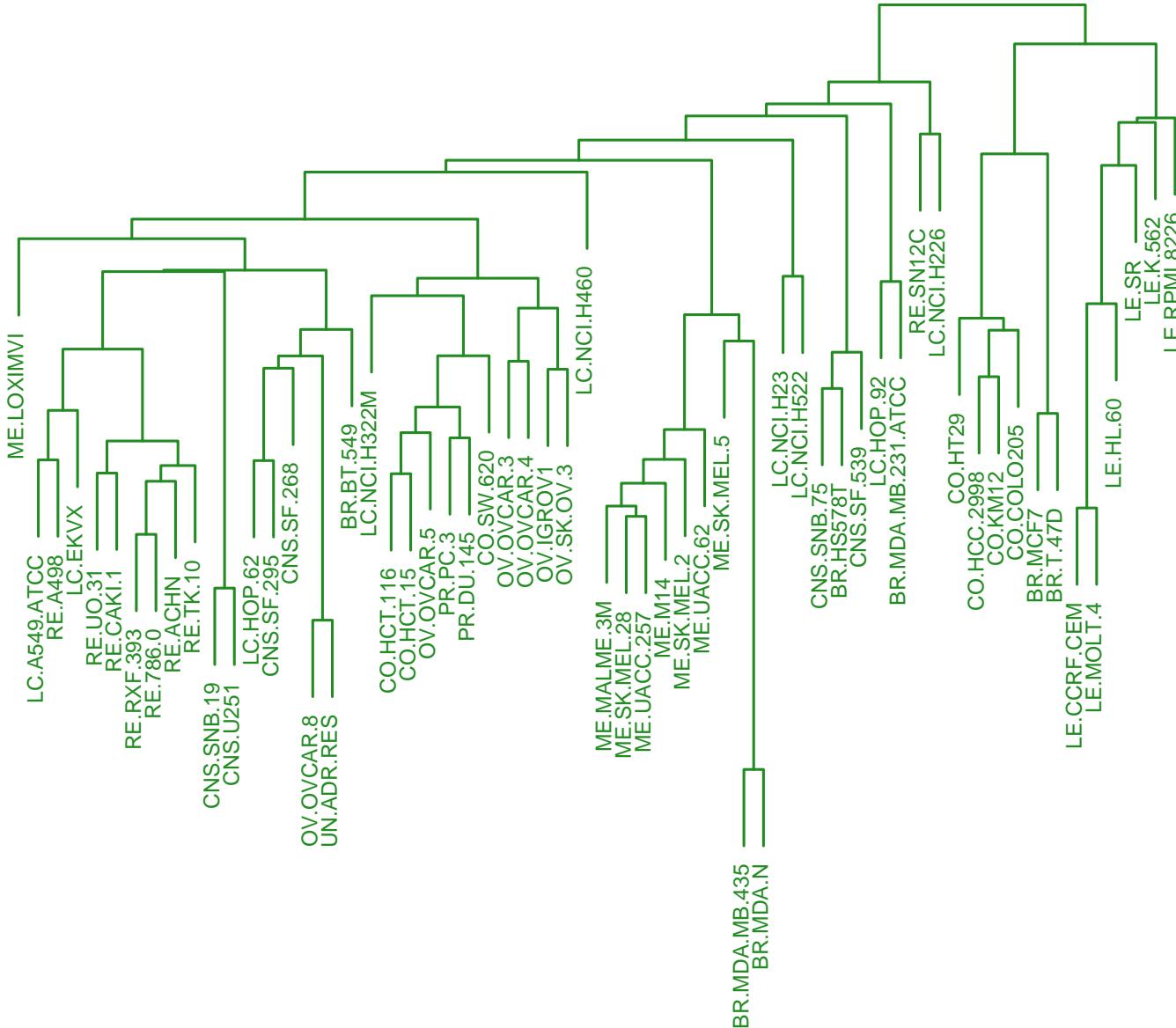
Такие иерархические структуры удобно представлять в виде корневых деревьев (дендограмм).

Примеры таксономий: номенклатура живых существ, библиографические классификаторы, различные системы научной классификации и т. п.

Алгоритмам иерархической кластеризации не нужно на вход подавать количество кластеров.

Имея дендрограмму, исследователь может сам обрезать ее на нужном уровне, получив некоторое количество кластеров.

Таксономия 60 клеток на основе анализа уровня экспрессии их генов. Иерархия составлена агglomerативным усредняющим методом.



## 9.5.1. Агломеративные методы

*Агломеративные методы*, или методы «снизу вверх», строят дерево в направлении от листьев к корню.

$\delta_{ii'}$  — мера различия («расстояние») между  $i$ -м и  $i'$ -м объектами ( $i = 1, 2, \dots, N$ ),

$\delta_{RS}$  — мера различия между кластерами  $R$  и  $S$ , которая каким-либо образом определяется исходя из попарных различий между элементами из  $R$  и из  $S$ .

Некоторые возможные способы определения  $\delta_{RS}$ :

по минимальному различию (по принципу «ближнего соседа»)

$$\delta_{RS} = \min_{i \in R, i' \in S} \delta_{ii'},$$

по максимальному различию (по принципу «дальнего соседа»)

$$\delta_{RS} = \max_{i \in R, i' \in S} \delta_{ii'},$$

по усредненному различию (по принципу «средней связи»)

$$\delta_{RS} = \frac{1}{|R| \cdot |S|} \sum_{i \in R} \sum_{i' \in S} \delta_{ii'}.$$

Агломеративные методы начинают работу с  $N$  кластеров, каждый из которых содержит один объект.

Пусть на некотором этапе имеется некоторый набор кластеров.

В этом наборе находится пара кластеров, скажем,  $R$  и  $S$ , для которой  $\delta_{RS}$  минимально.

Тогда объекты из этих групп объединяются в один кластер и т. д.

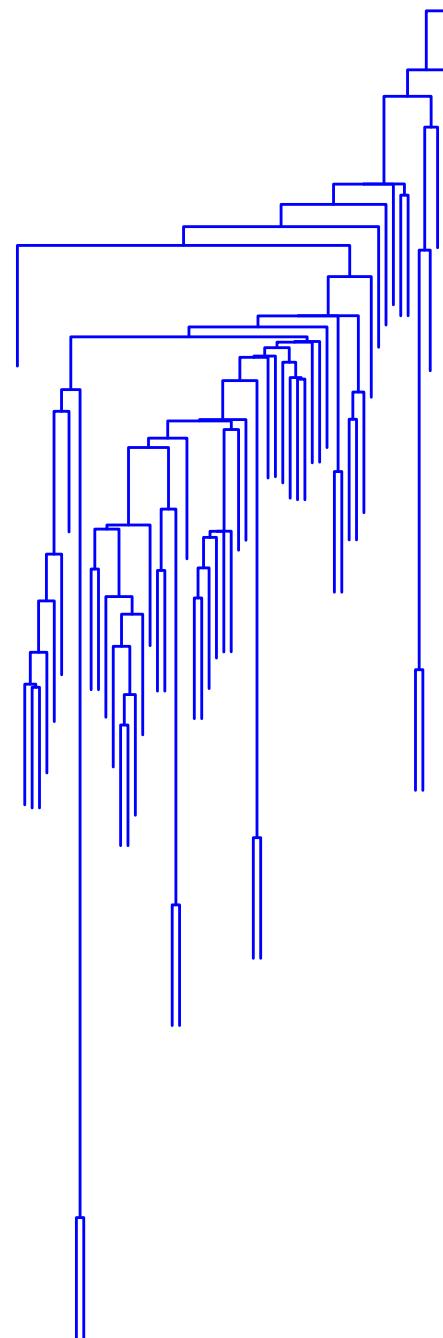
Использование разных функций  $\delta_{RS}$  приводит к различным уточнениям данного метода.

Первая из упомянутых выше функций (по минимальному различию) обычно приводит к сильно несбалансированным деревьям, длинным и плохо связанным кластерам.

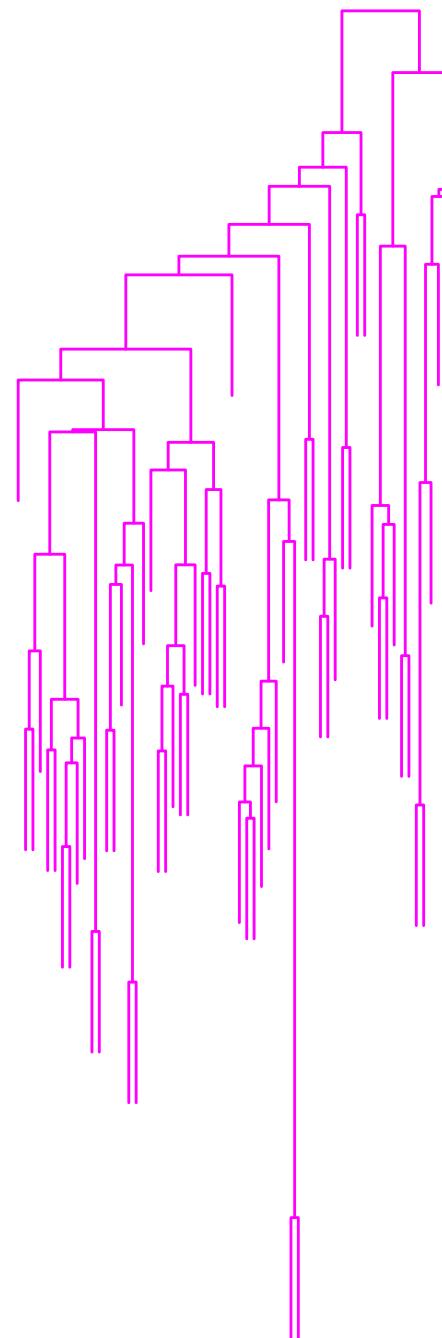
Вторая (по минимальному различию) приводит к сбалансированным деревьям и «компактным» кластерам, но, с другой стороны, даже весьма близкие объекты могут оказаться в далеких друг от друга (по иерархии) кластерах.

Третий метод занимает промежуточное положение между ними.

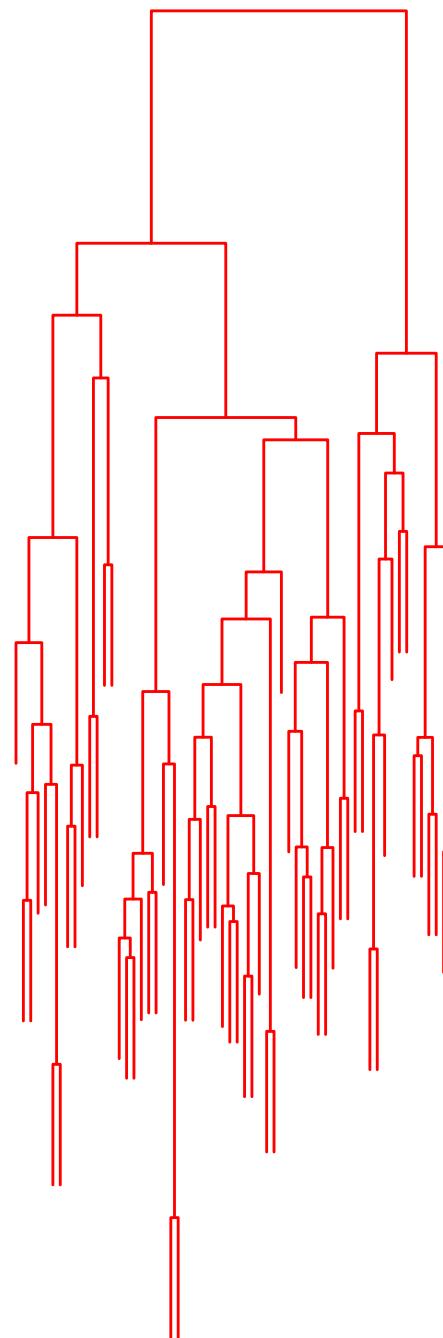
**Single**



**Average**



**Complete**



## **9.5.2. Разделяющие (дивизивные) методы**

*Разделяющие методы*, или методы «*сверху вниз*», строят дерево в направлении от корня к листьям.

*Один из подходов:*

На первом шаге ко множеству объектов применим какой-либо алгоритм (центров тяжести, медоидов и т. п.), разбивающий это множество на два кластера.

Затем разобьем каждый из полученных кластеров и т. д.

*Другой подход:* Достаточно описать, как разбить произвольный кластер  $R$ .

Сначала найдем в  $R$  объект, для которого среднее расстояние до всех остальных объектов максимально:

$$i^* = \operatorname{argmax}_{i \in R} \frac{1}{|R| - 1} \sum_{i' \in R, i' \neq i} \delta_{ii'}$$

Удалим  $i^*$  из  $R$  и поместим в  $S$ .

Далее выполняем следующие итерации.

Среди оставшихся в  $R$  объектов найдем

$$i^* = \operatorname{argmax}_{i \in R} \left( \frac{1}{|R| - 1} \sum_{i' \in R, i' \neq i} \delta_{ii'} - \frac{1}{|S|} \sum_{i' \in S} \delta_{ii''} \right)$$

Также поместим его в  $S$  и удалим из  $R$ .

Эту итерацию будем продолжать до тех пор, пока

$$\max \left( \frac{1}{|R| - 1} \sum_{i' \in R, i' \neq i} \delta_{ii'} - \frac{1}{|S|} \sum_{i' \in S} \delta_{ii''} \right) \geq 0.$$

Тогда имеем разбиение исходного кластера на два подкластера  $R$  и  $S$ .

Процедура для разбиения кластера  $R$  на два подкластера  $R$  и  $S$ :

**begin**

$$S \leftarrow \emptyset$$

$$i^* \leftarrow \operatorname{argmax}_{i \in R} \frac{1}{|R|-1} \sum_{i' \in R, i' \neq i} \delta_{ii'}$$

$$R \leftarrow R \setminus \{i^*\}$$

$$S \leftarrow S \cup \{i^*\}$$

$$\textbf{while } \max \left( \frac{1}{|R|-1} \sum_{i' \in R, i' \neq i} \delta_{ii'} - \frac{1}{|S|} \sum_{i' \in S} \delta_{ii''} \right) \geq 0$$

$$i^* = \operatorname{argmax}_{i \in R} \left( \frac{1}{|R|-1} \sum_{i' \in R, i' \neq i} \delta_{ii'} - \frac{1}{|S|} \sum_{i' \in S} \delta_{ii''} \right)$$

$$R \leftarrow R \setminus \{i^*\}$$

$$S \leftarrow S \cup \{i^*\}$$

**end**

Имеем разбиение исходного кластера на два подкластера  $R$  и  $S$ .

**end**

## 9.6. Поиск ассоциативных правил. Алгоритм Apriori

Задачу обучения без учителя можно рассматривать как исследование свойств распределения с. в.  $X \in \mathcal{X} \subseteq \mathbf{R}^d$ , при наличии выборки  $x^{(1)}, x^{(2)}, \dots, x^{(N)}$ .

Например, выделяем области в  $D \subseteq \mathcal{X}$ , для которых вероятность  $\Pr \{X \in D\}$  велика.

## Задача анализа рыночной корзины (market basket analysis) [1993]

Магазин обладает ассортиментом из  $d$  наименований товаров:  $U = \{1, 2, \dots, d\}$

Имеются данные о  $N$  покупках (транзакциях):  $t^{(1)}, t^{(2)}, \dots, t^{(N)}$ , где  $t^{(i)} \subseteq U$ .

Например,  $d \approx 10^4$ ,  $N \approx 10^8$ .

- Найти часто повторяющиеся шаблоны  $c$ .

Например, в 2 % всех транзакций встречается вместе хлеб и молоко.

- Найти «ассоциативные правила».

Например, если купили хлеб, то с вероятностью 50 % покупают молоко.

*Поддержкой* (support) шаблона  $c \subseteq U$  называется частота его встречаемости:

$$\text{supp } c = \frac{|\{i : c \subseteq t^{(i)}\}|}{N} \approx \Pr\{c \subseteq t\}$$

Пусть  $c, c'$  — два непересекающихся шаблона,  $c \subseteq U, c' \subseteq U, c \cap c' = \emptyset$

*Значимостью* (confidence) правила  $c \Rightarrow c'$  называется

$$\text{conf}(c \Rightarrow c') = \frac{\text{supp}(c \cup c')}{\text{supp } c} \approx \Pr(c' \subseteq t | c \subseteq t)$$

*Поддержка* правила  $c \Rightarrow c'$ :  $\text{supp}(c \Rightarrow c') = \text{supp}(c \cup c')$

Например,  $U = \{1, 2, 3, 4, 5\}$

$$t^{(1)} = \{2, 4, 5\}$$

$$t^{(2)} = \{1, 2, 4, 5\}$$

$$t^{(3)} = \{1, 2, 3, 4\}$$

$$t^{(4)} = \{2\}$$

Шаблон  $c = \{2\}$  встречается в каждой транзакции,  $\text{supp } \{2\} = 1$

Шаблон  $c = \{2, 4\}$  встречается в 3 транзакциях из 4,  $\text{supp } \{2, 4\} = 0.75$ .

Шаблон  $c = \{1, 2, 4\}$  встречается в 2 транзакциях из 4,  $\text{supp } \{1, 2, 4\} = 0.5$ .

Шаблон  $c = \{2, 4, 5\}$  встречается в 2 транзакциях из 4,  $\text{supp } \{2, 4, 5\} = 0.5$ .

Правило  $r = (\emptyset \Rightarrow \{2\})$        $\text{conf } r = 1, \text{supp } r = 1$

Правило  $r = (\{5\} \Rightarrow \{2, 4\})$        $\text{conf } r = 1, \text{supp } r = 0.5$

Правило  $r = (\{2, 4\} \Rightarrow \{1\})$        $\text{conf } r = \frac{0.5}{0.75} = \frac{2}{3}, \text{supp } r = 0.5$

$\delta$  — минимальная поддержка

$\gamma$  — минимальная значимость

Необходимо найти все правила  $c \Rightarrow c'$ , для которых

$\text{supp}(c \Rightarrow c') \geq \delta$ ,  $\text{conf}(c \Rightarrow c') \geq \gamma$

Алгоритм Apriori [R. Agrawal, R. Srikant, 1994]

- Вначале выделяет частые шаблоны, т. е. те, для которых  $\text{supp } c \geq \delta$
- Из частых шаблонов выделяем частые ассоциативные правила

## 9.6.1. Apriori. Обнаружение частых наборов

Основная идея — поиск в ширину (от частых шаблонов мощности 1 к частым шаблонам мощности 2, 3 и т. д.), используя следующее свойство:

*Свойство антимонотонности:*

Если  $c \subseteq c'$ , то  $\text{supp } c \geq \text{supp } c'$ .

Следствия:

- Если шаблон  $c$  частый, то все его подмножества частые
- Если шаблон  $c$  нечастый, то все его надмножества нечастые
- $\text{supp}(c \cup c') \leq \text{supp } c$  для всех  $c, c'$

Будем пытаться уменьшить объем работы с базой транзакций, пытаясь уменьшить количество новых кандидатов в частые шаблоны (функция apriori-gen)

$L_1 = \{c : c \subseteq U, |c| = 1, \text{supp } c \geq \delta\}$  (частые шаблоны мощности 1)

**for**  $m = 2, 3, \dots, d$

**if**  $L_{m-1} = \emptyset$

**break**

$C_m = \text{apriori-gen}(L_{m-1})$  (новые кандидаты в частые наборы)

    (Теперь вычисляем  $L_m = \{c \in C_m : \text{supp } c \geq \delta\}$ :)

**foreach**  $c \in C_m$

$c.count = 0$

**for**  $i = 1, 2, \dots, N$  (сканируем базу транзакций)

$C' = \{c : c \in C_m, c \subseteq t^{(i)}\}$

**foreach**  $c \in C'$

$c.count += 1$

$L_m = \{c \in C_m : c.count/N \geq \delta\}$

**return**  $\bigcup_m L_m$

В худшем случае  $2^d$  шаблонов — очень много!

Нужны хорошие структуры данных для реализации всех процедур

**function** apriori-gen ( $L$ )

Пусть шаблоны в  $L$  лексикографически упорядочены

$C = \emptyset$

Join-step:

**foreach**  $c \in L$

Пусть  $c = \{j_1, j_2, \dots, j_{m-1}\}$ ,  $j_1 < j_2 < \dots < j_{m-1}$

**for**  $j = j_{m-1} + 1, j_{m-1} + 2, \dots, d$

$c' = \{j_1, j_2, \dots, j_{m-2}, j\}$

**if**  $c' \in L$

Добавить  $\{j_1, j_2, \dots, j_{m-2}, j_{m-1}, j\}$  в  $C$

Prune-step:

**foreach**  $c \in C$

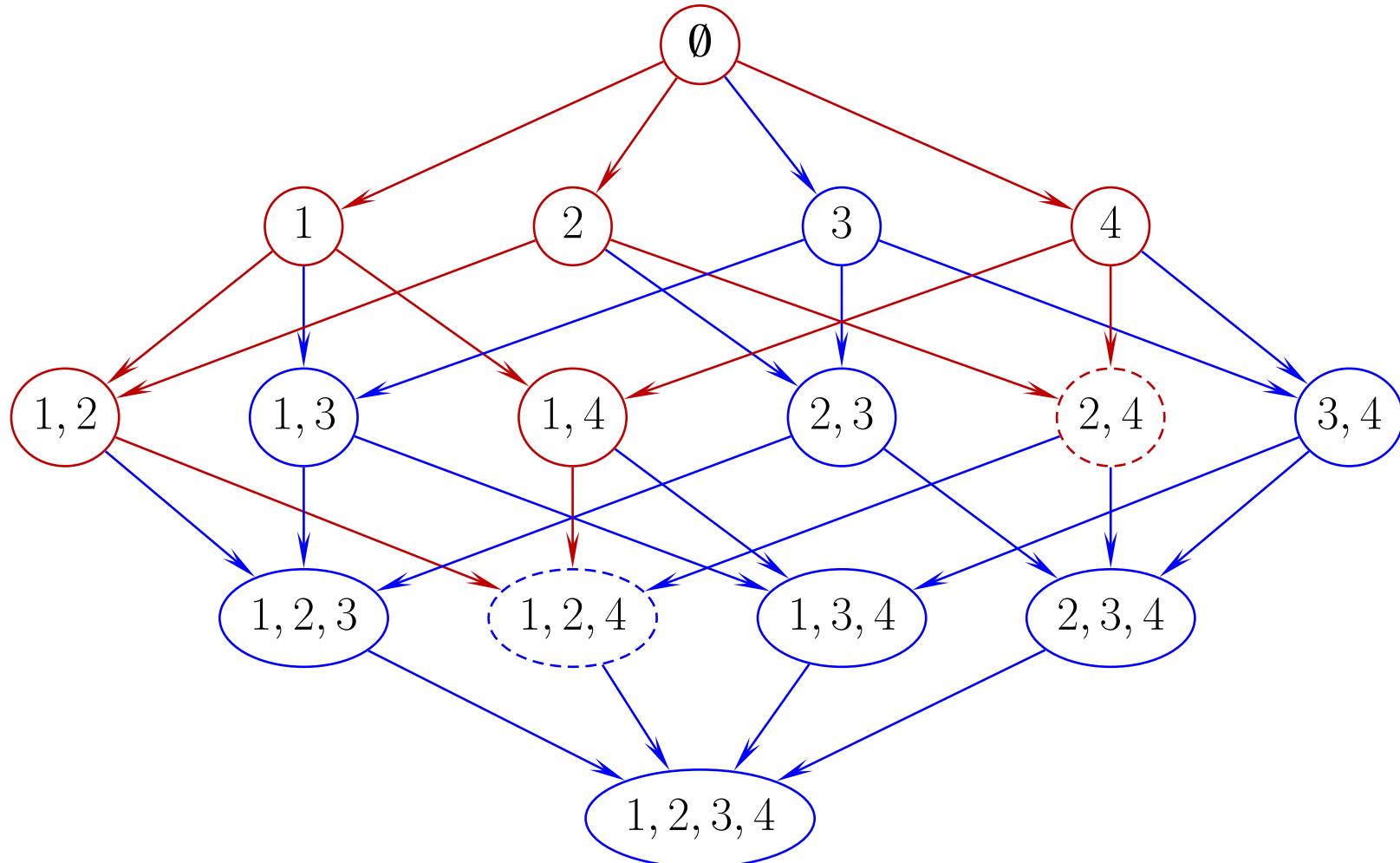
**foreach**  $j \in c$

**if**  $c \setminus \{j\} \notin L$

Удалить  $c$  из  $C$

**break**

**end**



Шаблоны, вошедшие в  $L_m$ , обведены красной сплошной линией;  
 шаблоны, вошедшие в  $C_m$ , но не в  $L_m$ , — красным пунктиром  
 (также отмечено пустое множество, так как можно считать, что  $\emptyset \in L_0$ )

$\{1, 2, 4\} = \{1, 2\} \cup \{1, 4\}$ , однако так как  $\{2, 4\} \notin L_2$  и  $\{2, 4\} \subseteq \{1, 2, 4\}$ , то  $\{1, 2, 4\} \notin C$

## 9.6.2. Apriori. Вывод ассоциативных правил

**foreach**  $c \in L$

    Assoc-Rules( $R, c, \emptyset$ )

**function** Assoc-Rules( $R, c, c'$ )

**foreach**  $j \in c$

$c = c \setminus \{j\}$

$c' = c' \cup \{j\}$

**if**  $(c \cup c').count \geq \gamma \cdot c.count$       (т. е.  $\text{conf}(c \Rightarrow c') \geq \gamma$ )

            Добавить правило  $c \Rightarrow c'$  в список  $R$

**if**  $|c| > 1$

            Assoc-Rules( $R, c, c'$ )

            (Можно не выполнять, если нужны правила с п.ч. мощности 1)

**end**

(Можно, наоборот, начинать с одного элемента в  $c$ )

## Пример. **Titanic**

$N = 2201$  пассажир (по другим данным 2208)

Дети:  $\leq 12$  лет

Признаки:

Class	Gender	Age	Survived
1st :325	Female: 470	Adult:2092	No :1490
2nd :285	Male :1731	Child: 109	Yes: 711
3rd :706			
Crew:885			

$$\delta = 0.005, \gamma = 0.8$$

Всего найдено 72 правила  $c \Rightarrow c'$ , где  $|c'| = 1$ .

Из них 12 правил, в которых правая часть  $\text{Survived}=\text{Yes}$  или  $\text{Survived}=\text{No}$

Из них 6 избыточные.

Например,  $\{\text{Class}=2\text{nd}, \text{Gender}=\text{Female}, \text{Age}=\text{Child}\} \Rightarrow \{\text{Survived}=\text{Yes}\}$  следует из  
 $\{\text{Class}=2\text{nd}, \text{Age}=\text{Child}\} \Rightarrow \{\text{Survived}=\text{Yes}\}$

Оставшиеся правила:

	lhs	rhs	support	confidence	lift
1	{Class=2nd, Age=Child}	=> {Survived=Yes}	0.011	1.000	3.096
2	{Class=1st, Gender=Female}	=> {Survived=Yes}	0.064	0.972	3.010
3	{Class=2nd, Gender=Female}	=> {Survived=Yes}	0.042	0.877	2.716
4	{Class=Crew, Gender=Female}	=> {Survived=Yes}	0.009	0.870	2.692
5	{Class=2nd, Gender=Male}	=> {Survived=No}	0.070	0.860	1.271
6	{Class=3rd, Gender=Male}	=> {Survived=No}	0.192	0.827	1.222

## 9.7. Google PageRank

[Brin S., Page L., 1998]

$N$  страниц,  $A = (a_{ij})$  —  $N \times N$ -матрица смежности

$$a_{ij} = \begin{cases} 1, & \text{если } i\text{-я страница ссылается на } j\text{-ю,} \\ 0 & \text{в противном случае.} \end{cases}$$

Исходящая степень  $i$ -й страницы (вершины):

$$u_i = \sum_{j=1}^N a_{ij}$$

Основные принципы:

- Чем больше страниц ссылаются на  $i$ -ю, тем выше ее ранг  $r_i$ .
- Чем выше ранг страницы, тем больший вес она имеет при «голосовании» за страницы, на которые ссылается.
- При этом ранг страницы равномерно распределяется на все эти страницы.

Исходя из этих принципов, составляем систему относительно  $r_i$ :

$$r_i = \frac{1 - \delta}{N} + \delta \sum_{j=1}^N \frac{a_{ji}}{u_j} r_j \quad (i = 1, 2, \dots, N) \quad (\star)$$

$\delta$  — «коэффициент демпфирования»,  $0 < \delta < 1$  (например,  $\delta = 0.85$ ).

Страницы, на которые нет ссылок, имеют ранг  $(1 - \delta)/N$ .

Следовательно, даже они «голосуют» за страницы, на которые ссылаются.

Суммируя все уравнения в  $(\star)$ , получаем  $(1 - \delta) \sum_{i=1}^N r_i = (1 - \delta)$ .

Откуда при  $\delta \neq 1$  получаем условие нормировки:

$$\sum_{i=1}^N r_i = 1.$$

Матричная запись уравнений  $(\star)$ :

$$r = \frac{1 - \delta}{N} e + \delta U^{-1} A^\top r \iff (I - \delta U^{-1} A^\top) r = \frac{1 - \delta}{N} e$$

( $e$  — вектор из единиц,  $I$  — единичная,  $U$  — диагональная матрица с  $u_j$  на диагонали).

Итак, матрица системы  $(\star)$  есть  $I - \delta U^{-1} A^\top$ . При  $0 < \delta < 1$  она имеет диагональное преобладание (по столбцам), поэтому невырождена. Система имеет единственное решение, причем все его компоненты строго положительны и в сумме равны 1.

Систему решают методом простых итераций:

$$r_i^{(m+1)} = \frac{1 - \delta}{N} + \delta \sum_{j=1}^N \frac{a_{ji}}{u_j} r_j^{(m)}$$

Так как  $\|\delta U^{-1} A^\top\|_1 = \delta < 1$ , то процесс сходится.

В [Brin S., Page L., 1998] сообщается, что при  $N \approx 322$  млн. алгоритм сошелся с приемлемой точностью за 52 итерации.

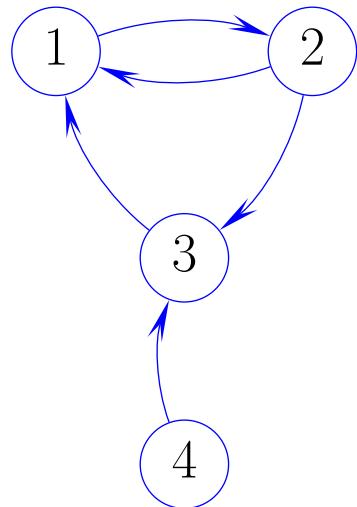
*Одна из интерпретаций* — цепь Маркова.

Ходим по ссылкам. Находясь на странице, случайно выбираем любую гиперссылку на ней и переходим по этой ссылке.

С вероятностью  $1 - \delta$  выбираем случайную страницу.

В этой интерпретации ранг страницы есть вероятность нахождения на ней при  $t \rightarrow \infty$ .

Пример. Интернет в 1984 г. :-)



$$\left\{ \begin{array}{l} r_1 = \frac{1-\delta}{4} + \delta \left( \frac{r_2}{2} + \frac{r_3}{1} \right), \\ r_2 = \frac{1-\delta}{4} + \delta \left( \frac{r_1}{1} \right), \\ r_3 = \frac{1-\delta}{4} + \delta \left( \frac{r_2}{2} + \frac{r_4}{1} \right), \\ r_4 = \frac{1-\delta}{4}. \end{array} \right.$$

$$U^{-1}A^\top = \begin{pmatrix} 0 & \frac{1}{2} & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Решение при  $\delta = 0.85$ :

$$r^{(0)} = \begin{pmatrix} 0.25 \\ 0.25 \\ 0.25 \\ 0.25 \end{pmatrix}, \quad r^{(1)} = \begin{pmatrix} 0.3562 \\ 0.2500 \\ 0.3562 \\ 0.0375 \end{pmatrix}, \quad r^{(2)} = \begin{pmatrix} 0.4466 \\ 0.3403 \\ 0.1756 \\ 0.0375 \end{pmatrix}, \quad r^{(3)} = \begin{pmatrix} 0.3314 \\ 0.4171 \\ 0.2140 \\ 0.0375 \end{pmatrix}, \quad \dots$$

$$r_1 = 0.3797, \quad r_2 = 0.3603, \quad r_3 = 0.2225, \quad r_4 = 0.0375.$$

**Замечание 9.3** При  $\delta = 1$  получаем задачу на собственные векторы:  $r = U^{-1}A^\top r$ . Требуется найти собственный вектор матрицы  $U^{-1}A^\top$ , относящийся к собственному значению 1. Для таких матриц (стохастических по столбцам) 1 всегда является максимальным собственным значением, но соответствующий собственный вектор может быть неединственным. Например, для матрицы

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Если же граф строго связан, то такой собственный вектор единственен (теорема эргодичности).

**Замечание 9.4** Вместо системы  $(\star)$  можно рассматривать

$$r_i = (1 - \delta) + \delta \sum_{j=1}^N \frac{a_{ji}}{u_j} r_j \quad (i = 1, 2, \dots, N).$$

Тогда условие нормировки запишется как  $\sum_{i=1}^N r_i = N$ .

*Глава 10*

## **Теория машинного обучения**

Истоки: В. П. Вапник, А. Я. Червоненкис [1971]

**Лемма 10.1** Пусть  $A_1, A_2, \dots, A_q$  — некоторые события (зависимые или независимые), заданные на одном вероятностном пространстве, тогда

$$\Pr(A_1 \cup A_2 \cup \dots \cup A_q) \leq \Pr A_1 + \Pr A_2 + \dots + \Pr A_q.$$

**Лемма 10.2 (С.Н. Бернштейн, Н. Chernoff, W. Hoeffding)** Пусть  $Z^{(1)}, Z^{(2)}, \dots, Z^{(N)}$  — независимые одинаково распределенные случайные величины.

$$\Pr \left\{ Z^{(i)} = 1 \right\} = \theta, \quad \Pr \left\{ Z^{(i)} = 0 \right\} = 1 - \theta$$

(схема Бернулли). Тогда

$$\Pr(|\hat{\theta} - \theta| > \gamma) \leq 2e^{-2\gamma^2 N},$$

где

$$\hat{\theta} = \frac{1}{N} \sum_{i=1}^N Z^{(i)}.$$

Рассмотрим задачу классификации на 2 класса.

$\mathcal{F}$  — некоторое множество функций  $f : \mathbf{R}^d \rightarrow \{0, 1\}$ .

$(X, Y)$  —  $(d + 1)$ -мерная с.в. с неизвестной функцией распределения  $P(x, y)$

Ожидаемый риск

$$R(f) = \mathsf{E} I(f(X) \neq Y) = \Pr I(f(X) \neq Y).$$

$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(N)}, y^{(N)})$  — обучающая выборка.

Эмпирический риск (частота ошибки) на данной выборке есть

$$\widehat{R}(f) = \frac{1}{N} \sum_{i=1}^N I(f(x^{(i)}) \neq y^{(i)}).$$

Задача: оценить  $R(f)$  в терминах  $\widehat{R}(f)$ .

*Принцип минимизации эмпирической риска:*

$$\widehat{f} = \operatorname*{argmin}_{f \in \mathcal{F}} \widehat{R}(f)$$

## 10.1. Случай конечного $\mathcal{F}$

Пусть  $\mathcal{F} = \{f_1, f_2, \dots, f_q\}$ .

Рассмотрим  $f \in \mathcal{F}$ .

Введем семейство случайных величин

$$Z^{(i)} = I(f(X^{(i)}) \neq Y^{(i)}) \quad (i = 1, 2, \dots, N).$$

$Z^{(i)}$  распределены одинаково и независимо — схема испытаний Бернулли.

По лемме 2

$$\Pr(|\widehat{R}(f) - R(f)| > \gamma) \leq 2e^{-2\gamma^2 N}$$

Откуда

$$\Pr(|\widehat{R}(f) - R(f)| \leq \gamma) \geq 1 - 2e^{-2\gamma^2 N}$$

Итак, для *конкретной*  $f$  эмпирический риск близок к ожидаемому с *большой* вероятностью.

Нас же интересует более сильное условие:

$$\Pr \left\{ \forall f \in \mathcal{F} : |\widehat{R}(f) - R(f)| \leq \gamma \right\}$$

$$\begin{aligned} \Pr \left\{ \exists f \in \mathcal{F} : |\widehat{R}(f) - R(f)| > \gamma \right\} &= \Pr(A_1 \cup \dots \cup A_q) \leq \\ &\leq \sum_{i=1}^q \Pr(A_i) \leq \sum_{i=1}^q 2e^{-2\gamma^2 N} = 2qe^{-2\gamma^2 N} \end{aligned}$$

Мы доказали утверждение:

**Утверждение 10.3 (О равномерной сходимости  $\widehat{R}$  к  $R$  в случае конечного  $\mathcal{F}$ )**

$$\Pr \left\{ \forall f \in \mathcal{F} : |\widehat{R}(f) - R(f)| \leq \gamma \right\} \geq 1 - 2qe^{-2\gamma^2 N}$$

**Следствие 10.4** Если  $N \geq \frac{1}{2\gamma^2} \ln \frac{2q}{\delta}$ , тогда

$$\Pr \left\{ \forall f \in \mathcal{F} : |\widehat{R}(f) - R(f)| \leq \gamma \right\} \geq 1 - \delta.$$

**Следствие 10.5**

$$\Pr \left\{ \forall f \in \mathcal{F} : |\widehat{R}(f) - R(f)| \leq \sqrt{\frac{1}{N} \ln \frac{2q}{\delta}} \right\} \geq 1 - \delta$$

**Следствие 10.6 (Принцип минимизации эмпирического риска)** Пусть

$$f^* = \operatorname{argmin}_{f \in \mathcal{F}} R(f), \quad \widehat{f} = \operatorname{argmin}_{f \in \mathcal{F}} \widehat{R}(f),$$

тогда

$$\Pr \left\{ R(\widehat{f}) \leq R(f^*) + 2 \sqrt{\frac{1}{N} \ln \frac{2q}{\delta}} \right\} \geq 1 - \delta.$$

ДОКАЗАТЕЛЬСТВО.

$$0 \leq R(\widehat{f}) - R(f^*) = R(\widehat{f}) - \widehat{R}(\widehat{f}) + \widehat{R}(\widehat{f}) - \widehat{R}(f^*) + \widehat{R}(f^*) - R(f^*) =$$

$$= \underbrace{R(\widehat{f}) - \widehat{R}(\widehat{f})}_{\leq \gamma} + \underbrace{\widehat{R}(\widehat{f}) - \widehat{R}(f^*)}_{\leq 0} + \underbrace{\widehat{R}(f^*) - R(f^*)}_{\leq \gamma} \leq 2\gamma.$$

Поэтому

$$\Pr \left\{ R(\widehat{f}) - R(f^*) \leq 2\gamma \right\} \geq \Pr \left\{ R(\widehat{f}) - \widehat{R}(\widehat{f}) \leq \gamma \text{ и } \widehat{R}(f^*) - R(f^*) \leq \gamma \right\} \geq 1 - \delta,$$

$$\text{где } \delta = \sqrt{\frac{1}{N} \ln \frac{2q}{\delta}}.$$

■

## 10.2. Случай бесконечного $\mathcal{F}$

Коэффициентом разнообразия (*shatter coefficient*)  $\Delta_0(\mathcal{F}, \mathbf{x})$  множества  $\mathcal{F}$  на выборке  $\mathbf{x}$  называется количество различных бинарных векторов вида

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

порождаемых всевозможными функциями  $f$  из  $\mathcal{F}$ .

*Пример.*

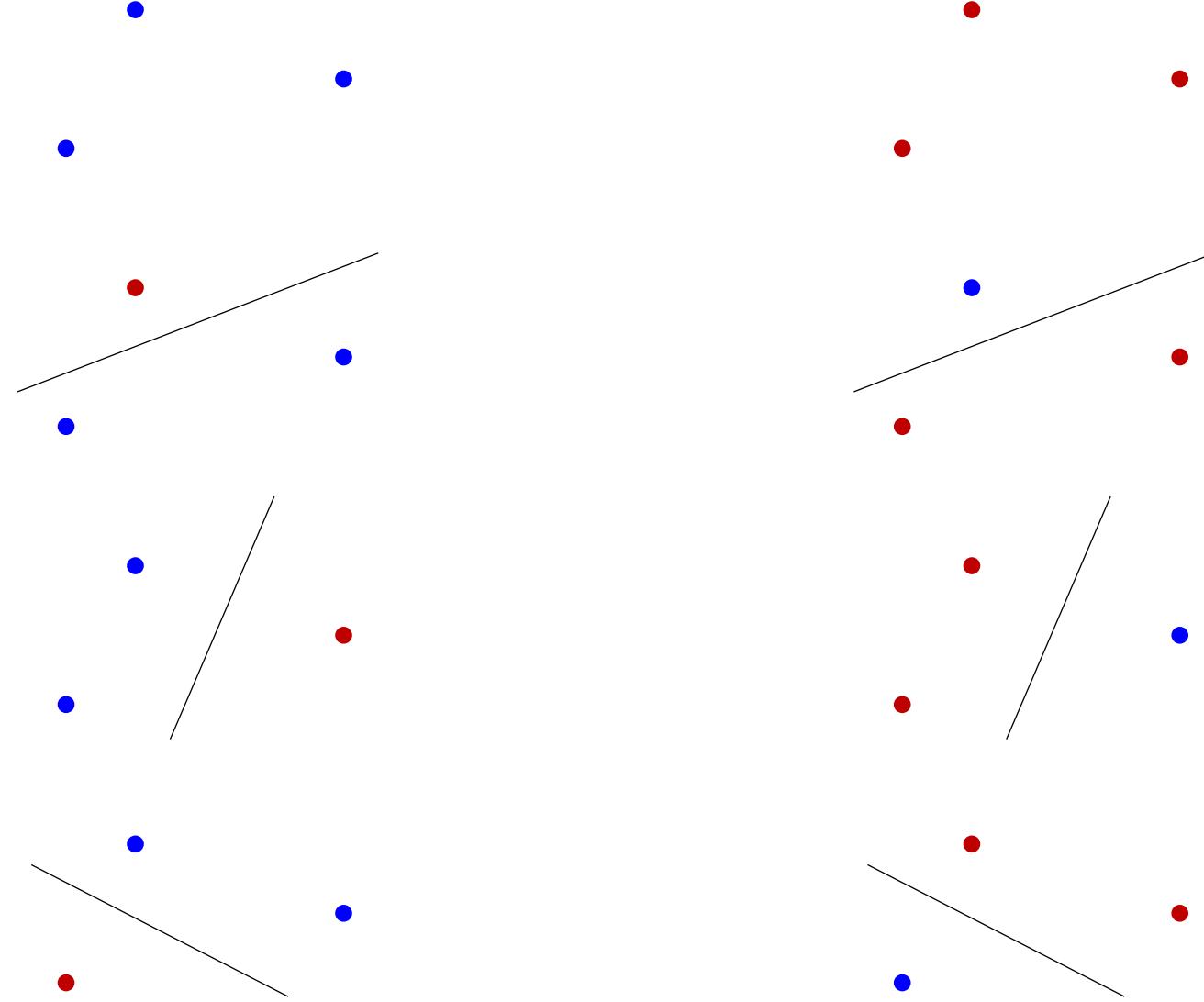
В качестве  $\mathcal{F}$  рассмотрим класс пороговых функций (линейных решающих правил)

$$\mathcal{F} = \text{TF}_d = \left\{ f(x) = I(\beta_0 + \langle \beta, x \rangle \leq 0) : \beta_0 \in \mathbf{R}, \beta \in \mathbf{R}^d \right\}$$



$$\Delta_0(\text{TF}_p, \mathbf{x}) = 8$$

(все возможные разбиения)



Если 3 точки на прямой, то  $\Delta_0(\text{TF}_d, \mathbf{x}) = 6$

Если  $N$  точек на прямой, то  $\Delta_0(\text{TF}_d, \mathbf{x}) = 2N$

Если 4 точки (в общем положении) в  $\mathbf{R}^2$ , то ...

$\Delta_0(\text{TF}_d, \mathbf{x}) = 14$

т. е. все без двух



*Функцией роста*  $\Delta(\mathcal{F}, N)$  класса  $\mathcal{F}$  называется максимальное значение  $\Delta_0(\mathcal{F}, \mathbf{x})$ , если максимум берется по всем возможным выборкам  $\mathbf{x}$  длины  $N$ :

$$\Delta(\mathcal{F}, N) = \max_{|\mathbf{x}|=N} \Delta_0(\mathcal{F}, \mathbf{x}).$$

**Утверждение 10.7** *Функция роста класса пороговых функций равна*

$$\Delta(\text{TF}_d, N) = 2 \left( \binom{N-1}{0} + \binom{N-1}{1} + \cdots + \binom{N-1}{d} \right).$$

Например,

$$\Delta(\text{TF}_2, 2) = 4, \Delta(\text{TF}_2, 3) = 8, \Delta(\text{TF}_2, 4) = 14, \Delta(\text{TF}_2, 5) = 22$$

$$\Delta(\text{TF}_3, 2) = 4, \Delta(\text{TF}_3, 3) = 8, \Delta(\text{TF}_3, 4) = 16, \Delta(\text{TF}_3, 5) = 30$$

Система из  $N$  неравенств с неизвестными  $\beta_0, \beta_1, \dots, \beta_d$ :

$$\begin{cases} \beta_0 + \langle \beta, x \rangle \leq 0, & \text{если } f(x) = 0 \\ \beta_0 + \langle \beta, x \rangle > 0, & \text{если } f(x) = 1 \end{cases}$$

Не нарушая общности, можем считать, что точка 0 — один из объектов.

Множество всех функций разобьется на два равномощных класса:

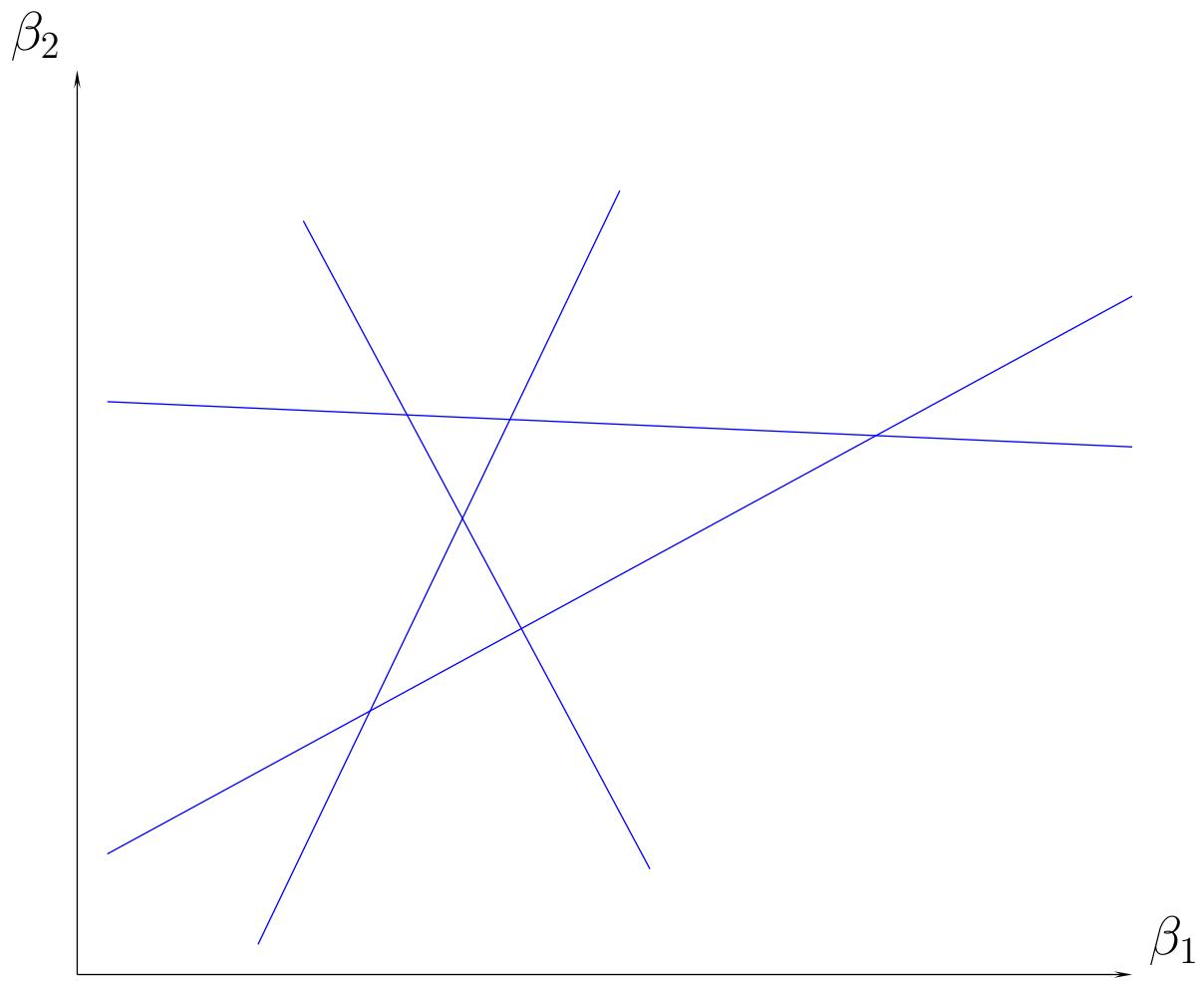
для которых  $f(0) = 0$

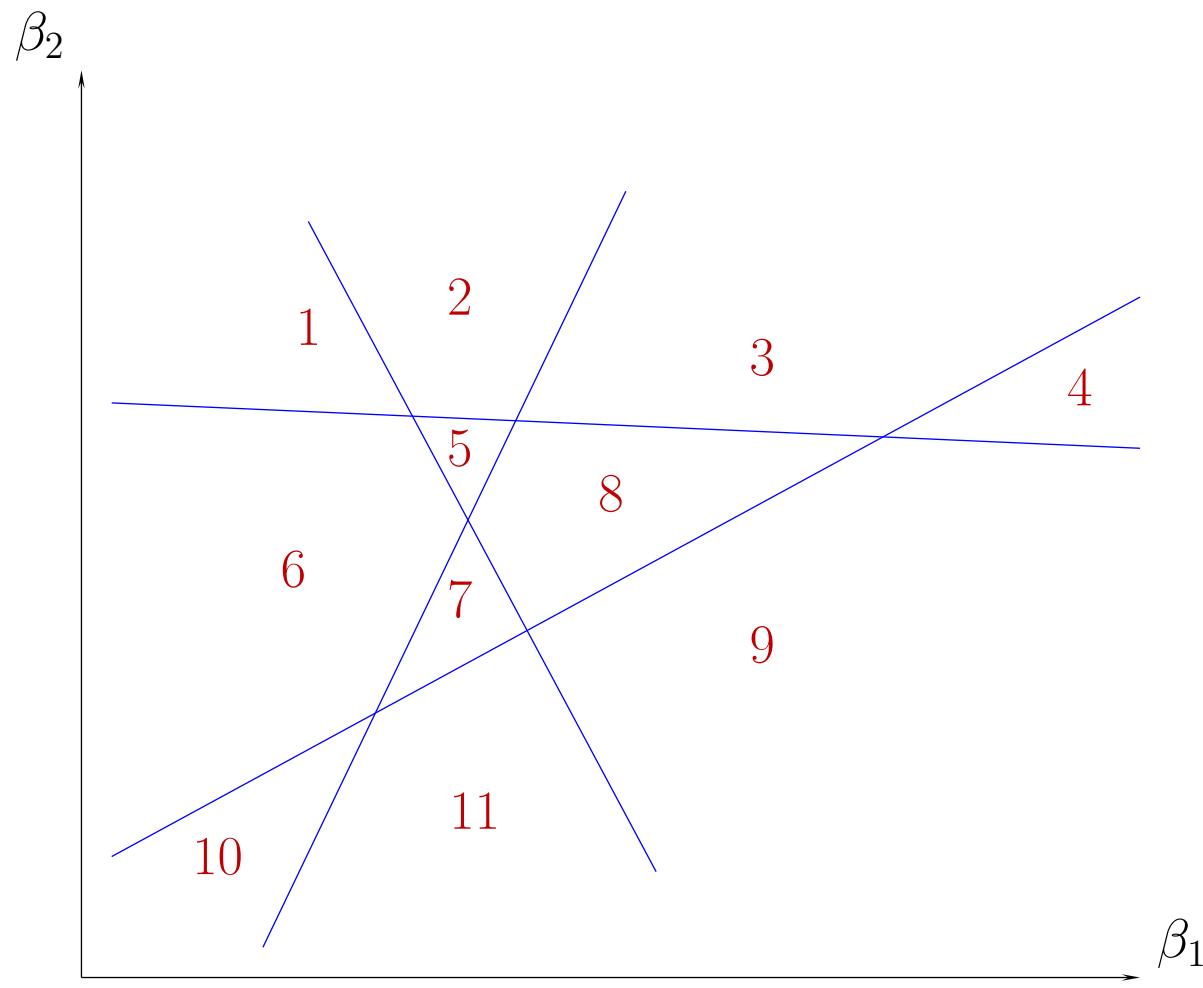
и для которых  $f(0) = 1$

Будем считать только первые, а затем результат умножим на 2.

Для них можно считать, что  $\beta_0 = -1$ :

$$\begin{cases} -1 + \langle \beta, x \rangle \leq 0, & \text{если } f(x) = 0 \\ -1 + \langle \beta, x \rangle > 0, & \text{если } f(x) = 1 \end{cases}$$





Количество областей

Найдем максимальное количество  $\Gamma(d, N)$  областей, на которые можно разбить  $d$ -мерное пространство  $N$  гиперплоскостями.

Справедливо рекуррентное соотношение

$$\Gamma(d, N) = \Gamma(d, N - 1) + \Gamma(d - 1, N - 1),$$

откуда получаем

$$\Gamma(d, N) = \binom{N}{0} + \binom{N}{1} + \cdots + \binom{N}{d}.$$

*Размерностью Вапника–Червоненкиса* (VC-размерностью), или *емкостью*,  $\text{VC}(\mathcal{F})$  класса  $\mathcal{F}$  называется максимальное  $N$ , при котором  $\Delta(\mathcal{F}, N) = 2^N$ .

Если максимального  $N$  не существует, то будем считать, что размерность Вапника–Червоненкиса равна бесконечности.

Эквивалентно: *Размерностью Вапника–Червоненкиса* называется наибольшая мощность множества в  $\mathcal{X}$ , разбиваемая (*shatter*) функциями из  $\mathcal{F}$ .

Говорят, что  $\mathcal{F}$  *разбивает* множество  $\{x^{(1)}, x^{(2)}, \dots, x^{(N)}\}$ , если для любого двоичного набора  $y \in \{0, 1\}^N$  найдется  $f \in \mathcal{F}$ , такое, что  $y = (f(x^{(1)}), f(x^{(2)}), \dots, f(x^{(N)}))$ .

Чтобы доказать, что  $\text{VC } \mathcal{F} = h$  достаточно проделать две вещи:

- 1) доказать  $\text{VC } \mathcal{F} \geq h$ ; для этого достаточно построить  $\{x^{(1)}, x^{(2)}, \dots, x^{(N)}\}$ , разбиваемое функциями из  $\mathcal{F}$ ;
- 2) доказать  $\text{VC } \mathcal{F} \leq h$ ; для этого достаточно показать, что никакое множество из  $n + 1$  точек нельзя разбить функциями из  $\mathcal{F}$ .

**Утверждение 10.8** *Если  $\mathcal{F}$  конечно, то*

$$\text{VC}(\mathcal{F}) \leq \log_2 |\mathcal{F}|.$$

**Утверждение 10.9** *Размерность Вапника–Червоненкиса класса пороговых функций равна*

$$\text{VC}(\text{TF}_d) = d + 1.$$

**Теорема 10.10 (Вапник–Червоненкис)** (*O равномерной сходимости  $\widehat{R}$  к  $R$  в случае конечной  $\text{VC } \mathcal{F}$* ) Пусть  $\mathcal{F}$  задано и  $\text{VC } \mathcal{F}$  конечно, тогда

$$\Pr \left\{ \forall f \in \mathcal{F} : |\widehat{R}(f) - R(f)| \leq \gamma \right\} \geq 1 - \delta, \quad (2)$$

$$\Pr \left\{ R(\widehat{f}) \leq R(f^*) + 2\gamma \right\} \geq 1 - \delta. \quad (3)$$

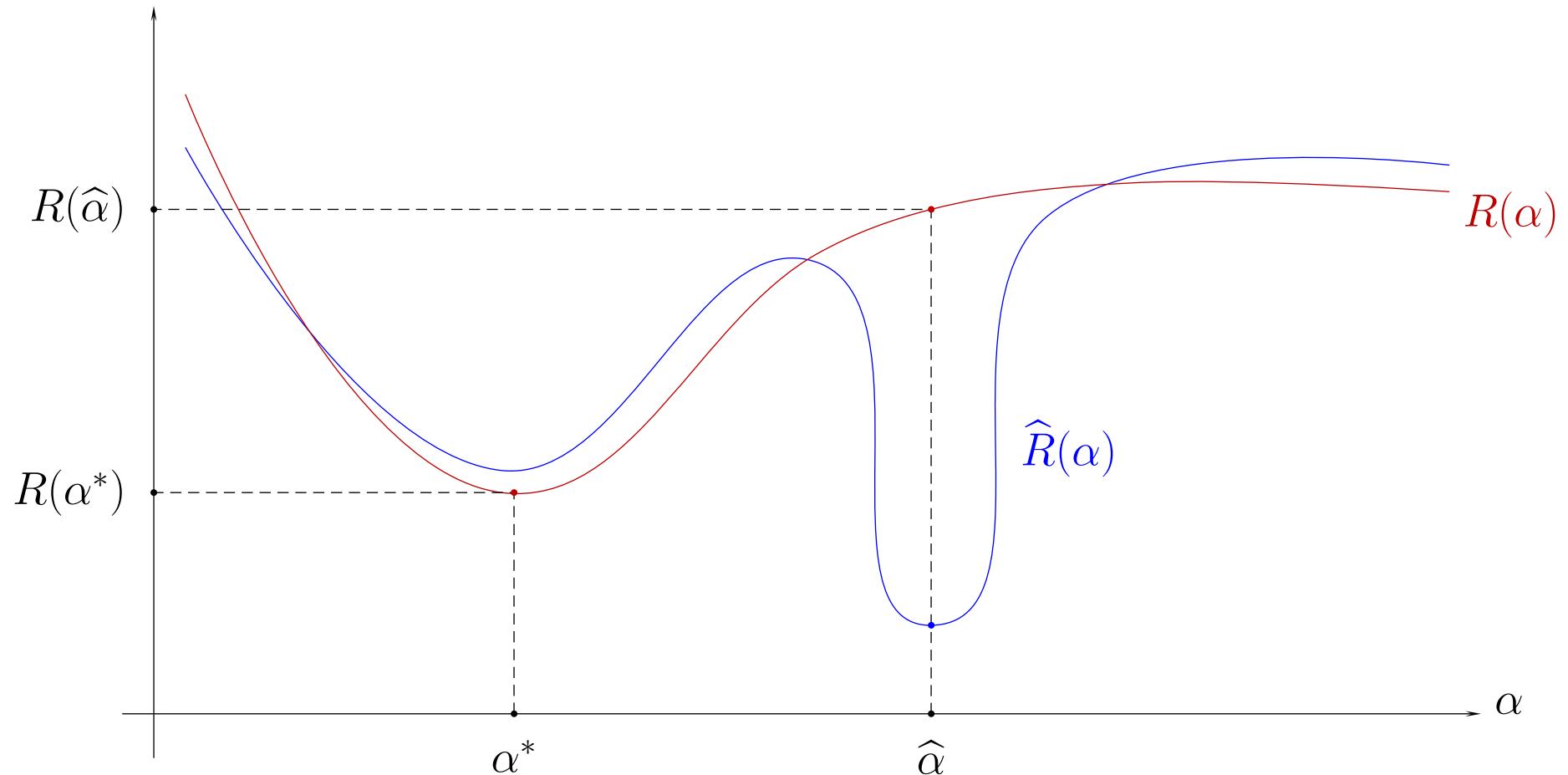
где

$$\gamma = O \left( \sqrt{\frac{\text{VC } \mathcal{F}}{N} \ln \frac{N}{\text{VC } \mathcal{F}}} + \frac{1}{N} \ln \frac{1}{\delta} \right). \quad (4)$$

В правой части неравенства  $R(\widehat{f}) \leq R(f^*) + 2\gamma$  с ростом  $\text{VC } \mathcal{F}$  первое слагаемое убывает, а второе (при  $\text{VC } \mathcal{F} < N$ ) возрастает. Сумма достигает своего минимума при некотором оптимальном значении  $\text{VC } \mathcal{F}$ .

Также можно доказать, что если  $\text{VC } \mathcal{F} = \infty$ , то равномерной сходимости нет.

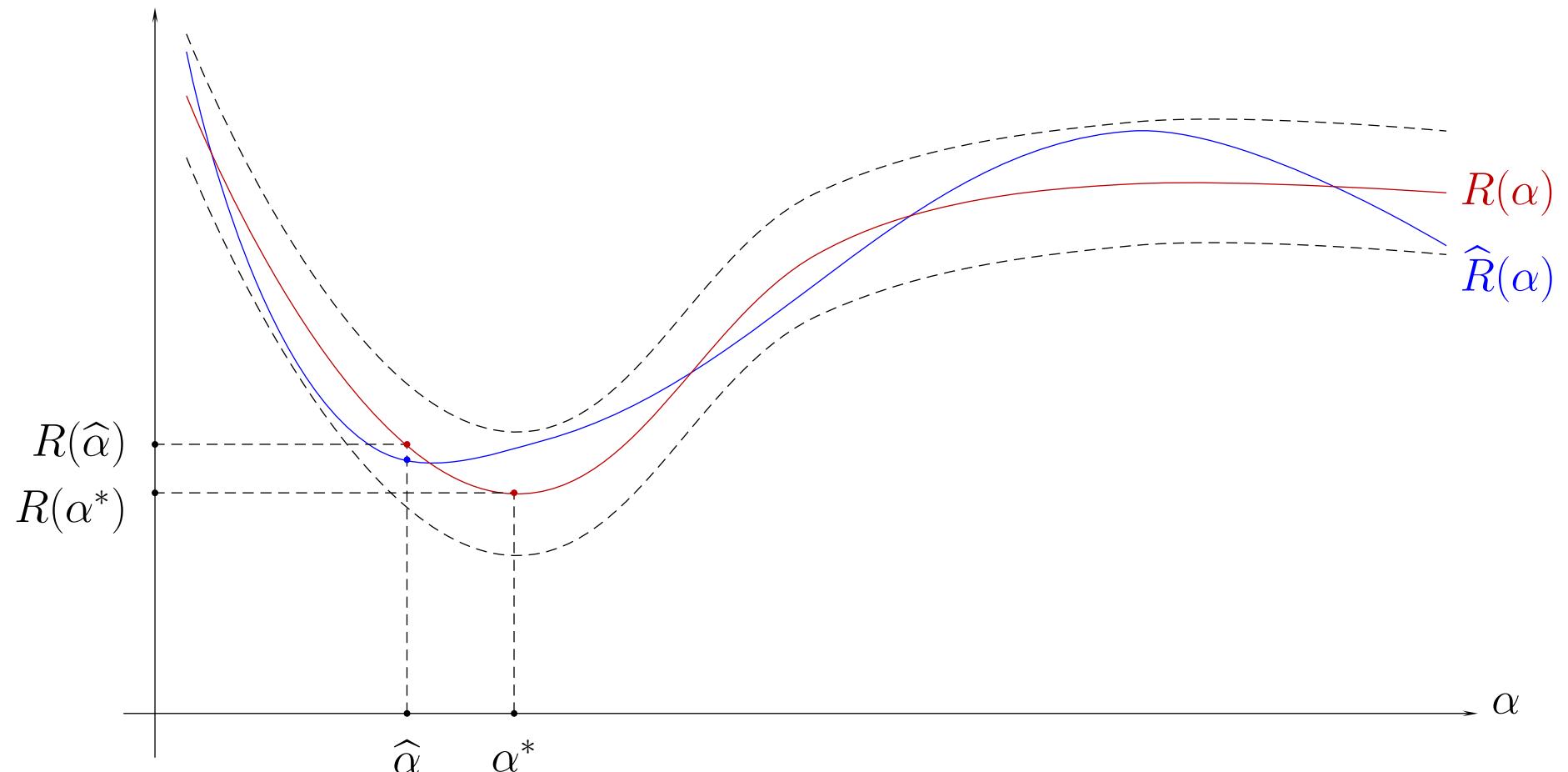
Пусть  $\mathcal{F} = \{f : f(x, \alpha), \alpha \in [0, 1]\}$  — класс решающих правил  
 $R(\alpha)$  — средний риск,  $\widehat{R}(\alpha)$  — эмпирический риск на функции  $f(x, \alpha)$



$R(\widehat{\alpha})$  далеко от минимального значения  $R(\alpha^*)$ .

$$\lim_{N \rightarrow \infty} \Pr \left\{ |\widehat{R}(f) - R(f)| \leq \gamma \right\} = 1.$$

## Равномерная сходимость



$$\lim_{N \rightarrow \infty} \Pr \left\{ \forall f : \in \mathcal{F} : |\hat{R}(f) - R(f)| \leq \gamma \right\} = 1.$$

Принцип *структурной минимизации риска*, предложенный Вапником, предназначен для отыскания оптимальной сложности. Пусть в  $\mathcal{F}$  выделена некоторая цепочка подсемейств  $\mathcal{F}_1 \subset \mathcal{F}_2 \subset \dots \subset \mathcal{F}_M = \mathcal{F}$ . Необходимо выбрать семейство с минимальным значением верхней оценки в (3).

К сожалению, оценки (2)–(4) являются слишком завышенными, чтобы их можно было использовать для практических вычислений. Однако они представляют большой интерес с теоретической точки зрения и являются стимулом для дальнейших исследований в поиске подобных оценок, в частности в теории вероятностно–приближенно–корректного обучения (probably approximately correct learning, PAC–learning).

Изложенные здесь результаты распространяются на задачу классификации с большим числом классов и на задачу восстановления регрессии.

Пусть  $\mathcal{F}$  — некоторый класс функций  $f : \mathcal{X} \rightarrow \mathcal{Y}$ . Размерностью Вапника–Червоненкиса для класса  $\mathcal{F}$  называется  $\text{VC } \mathcal{F} = \text{VC } \mathcal{F}'$ , где

$$\mathcal{F}' = \left\{ I(f(x) - y) : f \in \mathcal{F}, y \in \mathcal{Y} \right\}.$$

**Теорема 10.11 (Лемма Зауэра; Sauer, 1972; Shelah, 1972)** Пусть класс  $\mathcal{F}$  имеет конечную размерность Вапника–Червоненкиса  $\text{VC}(\mathcal{F}) = h$ , тогда

$$\Delta(\mathcal{F}, N) \leq \binom{N}{0} + \binom{N}{1} + \cdots + \binom{N}{h} < \left(\frac{eN}{h}\right)^h$$