# АНСАМБЛИ РЕШАЮЩИХ ПРАВИЛ В МАШИННОМ ОБУЧЕНИИ

Н.Ю. Золотых (ВМК ННГУ)

zolotykh@vmk.unn.ru

HPC-форум 2011

Нижний Новгород

1 ноября 2011

# План

- Что такое машинное обучение?
- Деревья решений
- Бустинг и Co
  - AdaBoost
  - Gradient Boosting Trees
- Баггинг и Co
  - Баггинг
  - Random forests

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

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

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

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

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

- На практике фаза обучения может предшествовать фазе работы алгоритма (например, детектирование лиц на фотокамере)
- или обучение может проходить в процессе функционирования алгоритма (определение спама).

# Обучение с учителем

Множество $\mathscr{X}$ — *объекты, примеры, ситуации, входы* (samples)

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

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

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

Зависимость известна только на объектах из *обучающей выборки*:

$$\{(x_1, y_1),\ (x_2, y_2),\ \ldots,\ (x_N, y_N)\}$$

Пара $(x_i, y_i) \in \mathscr{X} \times \mathscr{Y}$ — *прецедент*.

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

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

# Признаковые описания

*Вход:*

$$x = (\xi_1, \xi_2, \ldots, \xi_d) \in \mathscr{X} = Q_1 \times Q_2 \times \ldots \times Q_d,$$

где $Q_j = \mathbf{R}$ или $Q_j$ — конечно

$\xi_j$ — $j$-й *признак* (*свойство*, *атрибут*) объекта $x$.

- Если $Q_j$ конечно, то $j$-й признак — *номинальный* (*категориальный* или *фактор*).
  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, $Q = \{\text{холодно}, \text{прохладно}, \text{тепло}, \text{жарко}\}$
- Если $Q_j = \mathbf{R}$, то признак *количественный*.

*Выход:* $y \in \mathscr{Y}$

- $\mathscr{Y} = \mathbf{R}$ — *задача восстановления регрессии*
- $\mathscr{Y} = \{1, 2, \ldots, K\}$ — *задача классификации*.

## Пример 1. Распознавание рукописных символов (цифр)

Научиться распознавать рукописный символ по его изображению.

Изображение — битовая матрица размера $32 \times 32$:

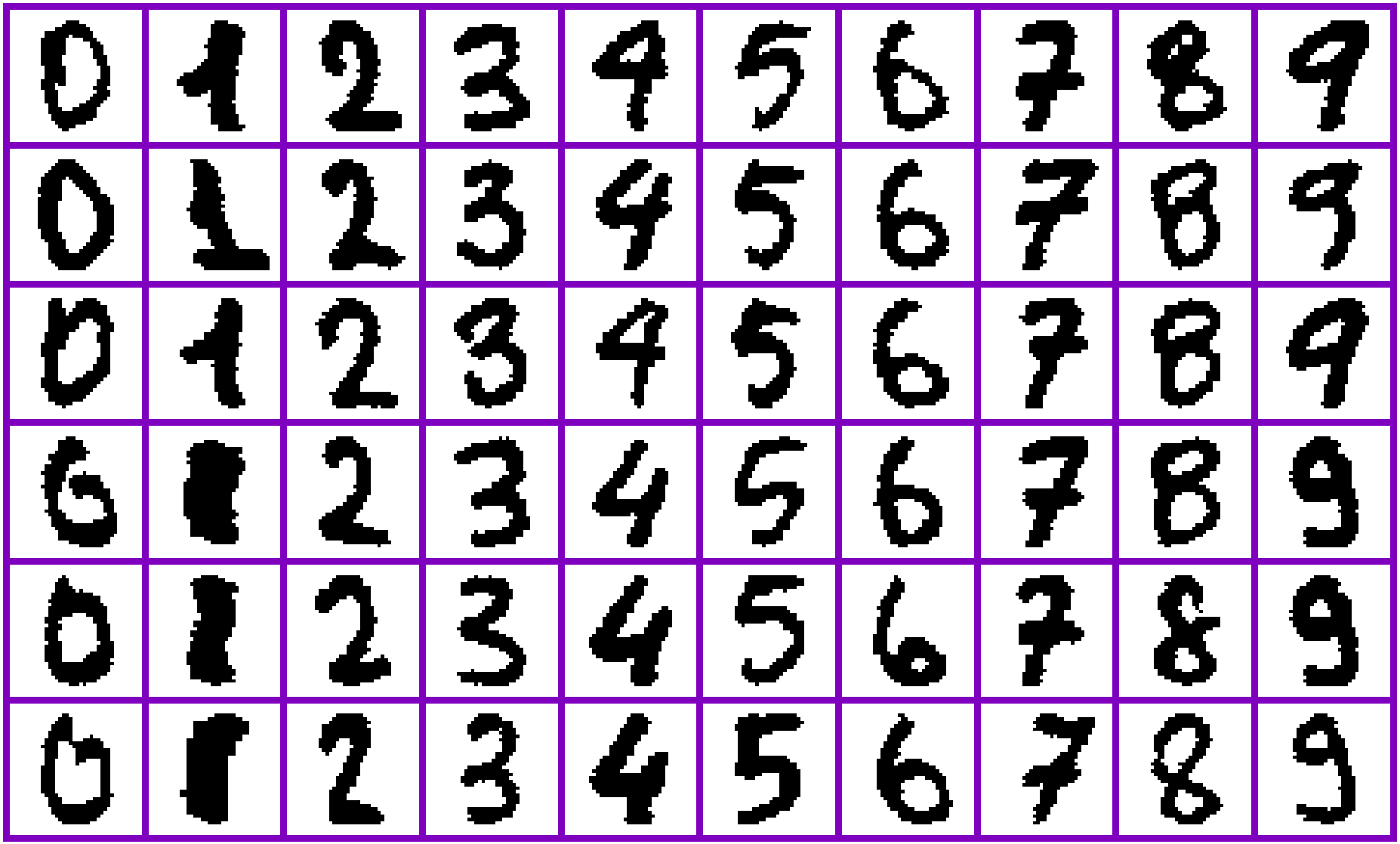
$$x \in \mathscr{X} = \{0,1\}^{32\times 32} = \{0,1\}^{1024}$$

$$\mathscr{Y} = \{0,1,2,\ldots,9\}$$

Это задача классификации.

optdigit http://www.ics.uci.edu/~mlearn/MLRepository.html — 1934 прецедента.

Некоторые объекты из обучающей выборки



## Пример 2. Оценка стоимости дома

Предположим, что имеются данные о жилых загородных домах в некоторой местности.

Для каждого дома известна его цена, состояние, жилая площадь, количество этажей, количество комнат, время постройки, удаленность до основных магистралей, наличие инфраструктуры, экологическая обстановка в районе и т. п.

Требуется научиться оценить цену по остальной информации.

Объектами являются дома, входами — их характеристики, а выходом — цена дома.

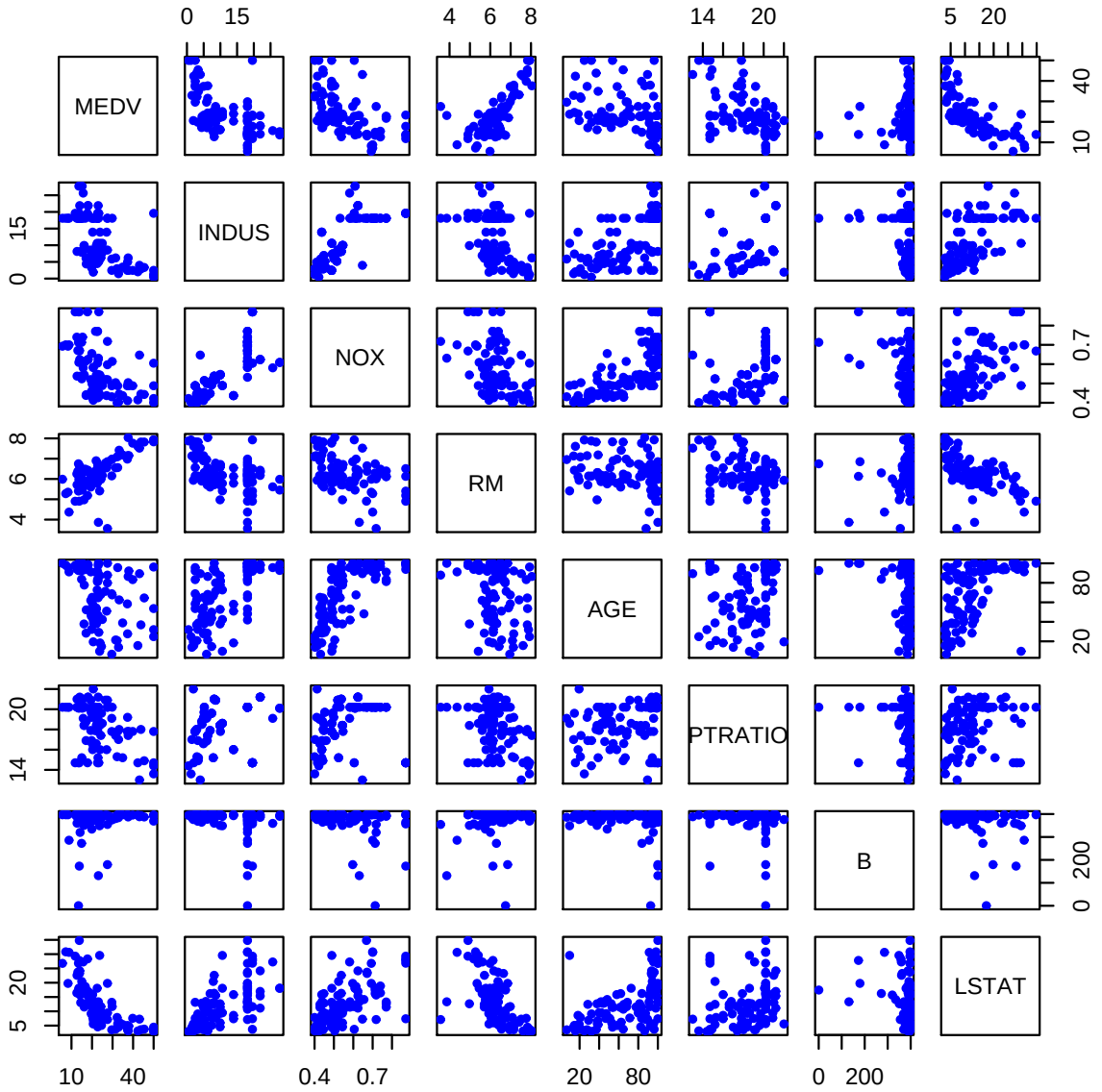
Это задача восстановления регрессии.

Boston Housing Data http://archive.ics.uci.edu/ml/datasets/Housing

Информация агрегирована: территория поделена на участки и дома, стоящие на одном участке, собраны в группы. Нужно оценить среднюю цену дома. Таким образом, объектами являются сами эти группы. Их общее количество — 506.
Признаки

1. CRIM — уровень преступности на душу населения,
2. ZN — процент земли, застроенной жилыми домами (только для участков площадью свыше 25000 кв. футов),
3. INDUS — процент деловой застройки,
4. CHAS — 1, если участок граничит с рекой; 0 в противном случае (бинарный признак),
5. NOX — концентрация оксида азота, деленная на $10^7$,
6. RM — среднее число комнат (по всем домам рассматриваемого участка),
7. AGE — процент домов, построенных до 1940 г. и занимаемых владельцами,
8. DIS — взвешенное расстояние до 5 деловых центров Бостона,
9. RAD — индекс удаленности до радиальных магистралей,
10. TAX — величина налога в $10000,
11. PTRATIO — количество учащихся, приходящихся на одного учителя (по городу),
12. B = $1000(\mathtt{AA} - 0.63)^2$, где AA — доля афро-американцев,
13. LSTAT — процент жителей с низким социальным статусом.

Диаграммы рассеяния для каждой пары переменных MEDV, INDUS, NOX, RM, AGE, PTRATIO, B. Значение переменной MEDV нужно научиться предсказывать по значениям остальных переменных. Изображены только по 100 случайных точек.



# Машинное обучение и HPC

Объемы исходных данных, подаваемых на вход алгоритмов обучения, могут быть весьма значительными.

Коллекция UCI Machine Learning Repository http://archive.ics.uci.edu/ml содержит такие большие наборы данных, как

- «Netflix Prize» (более 100 млн. объектов, 17770 признаков),
- «Bag of Words» (8000000 объектов, 100000 признаков),
- «KDD Cup 1999 Data» (4000000 объектов, 42 признака).

Объемы входных данных в задачах, возникающих при оценке качества интернет-рекламы, поисковом ранжировании, могут быть такими, что их можно хранить только распределенно.

# Некоторые методы машинного обучения

- Линейный и квадратичный дискриминантный анализ
- Логистическая регрессия
- Метод $k$ ближайших соседей
- Наивный байесовский классификатор
- Деревья решений (C4.5, CART и др.)
- Персептрон и нейронные сети
- Машина опорных векторов (SVM)
- Ансамбли решающих правил (бустинг, баггинг)
- . . .

См., например,

Top 10 algorithms in data mining // Knowl. Inf. Syst. 2008. № 14. P. 1–37

# 2. Деревья решений

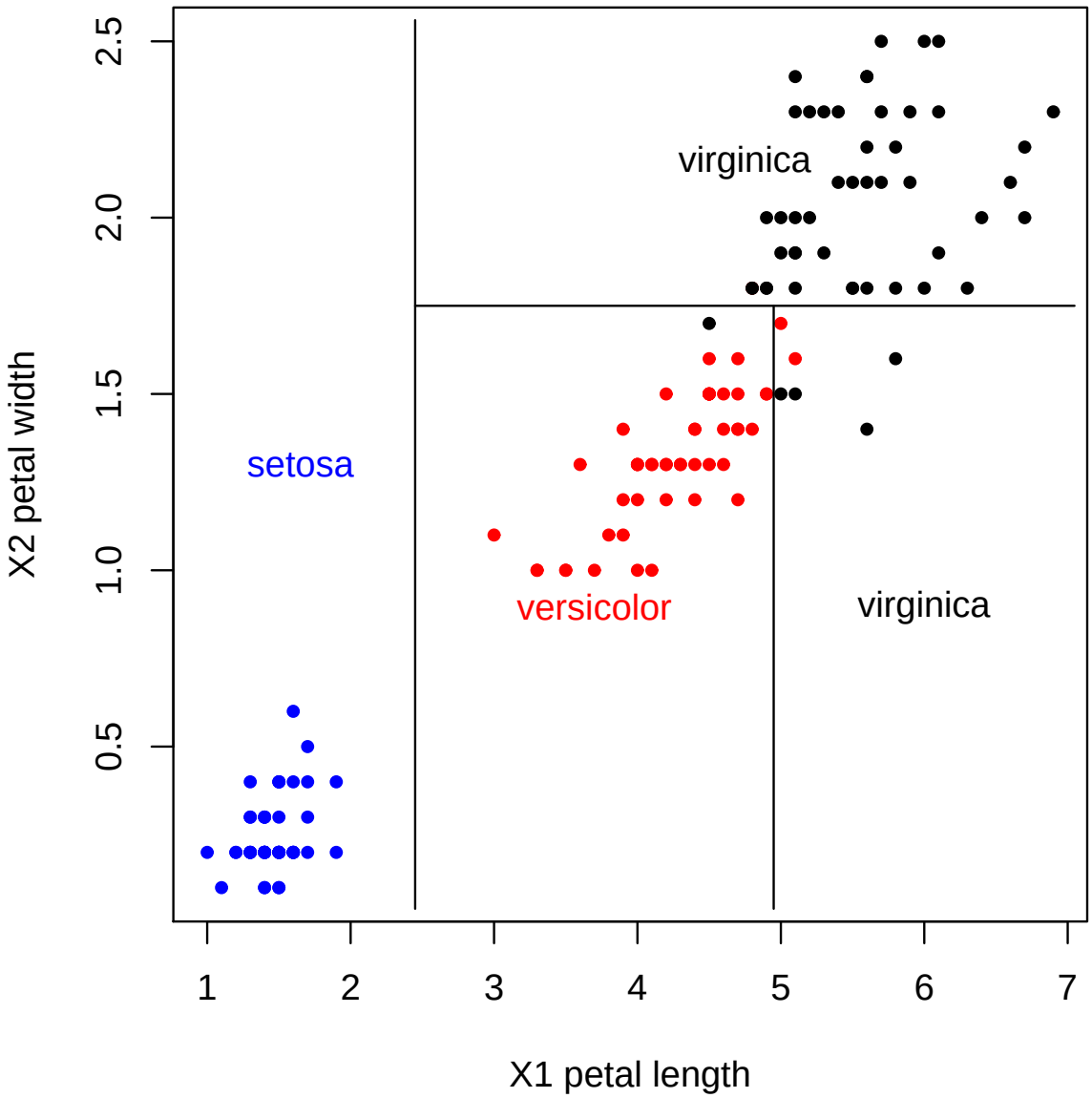
Пространство признаков разбивается на параллелепипеды со сторонами, параллельными осям координат (ящики).

В каждом ящике ответ аппроксимируется с помощью некоторой простой модели, например, константой.

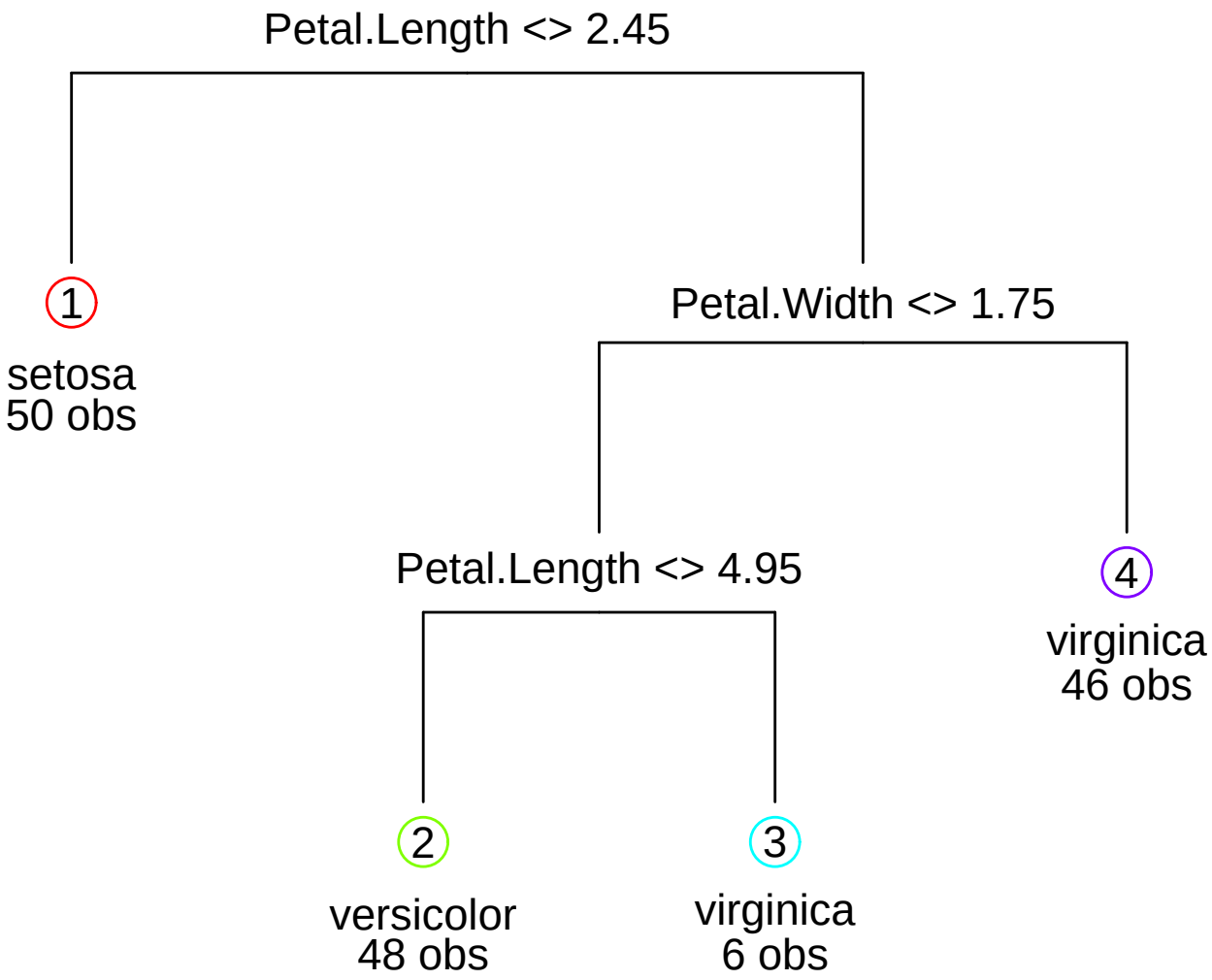
Используются только рекурсивные гильотинные разбиения.

Задача классификации цветов ириса (Fisher, 1936).

$X_1$, $X_2$ — длина и ширина чашелистика.



Дерево решений:



## Популярные алгоритмы построения деревьев решений

- See5/C5.0 [Quinlan et., 1997] $\leftarrow$ C4.5 [Quinlan, 1993] $\leftarrow$ ID3 [Quinlan, 1979] $\leftarrow$ CLS [Hunt, Marin, Stone, 1966]
- CART — Classification and Regression Trees [Breiman, Friedman, Olshen, Stone, 1984]

# Алгоритм CART

Разбиения (splits) имеют вид:

- $\xi_j \leq c$ для количественных признаков;
- $\xi_j \in L$, где $L \subset \{1, 2, \ldots, M_j\}$ для качественных признаков.

Дерево строим рекурсивно.

Пусть на текущем шаге имеется разбиение пространства признаков на области $R_1, R_2, \ldots, R_M$.

- Выбираем область $R_m$.
- Выбираем $j$ и $c$ (или $L$) так, чтобы добиться максимального уменьшения *загрязнения* (impurity) $Q_m$ ($m = 1, 2, \ldots, M$).
- Строим разбиение (split) и повторяем действия.

Способы измерить «загрязнение»:

- Для задачи восстановления регрессии:

$$Q_m = \sum_{x_i \in R_m} (y_i - f(x_i))^2.$$

- Для задачи классификации:
  - Misclassification (ошибка отнесения к другому классу)

$$Q_m = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i \neq k(m)) = 1 - \max_k p_{km} = 1 - p_{k(m),\, m}.$$

  - Индекс К. Джини

$$Q_m = \sum_{k \neq k'} p_{mk} p_{mk'} = \sum_{k=1}^{K} p_{mk}(1 - p_{mk}) = 1 - \sum_{k=1}^{K} p_{mk}^2.$$

  - Энтропия

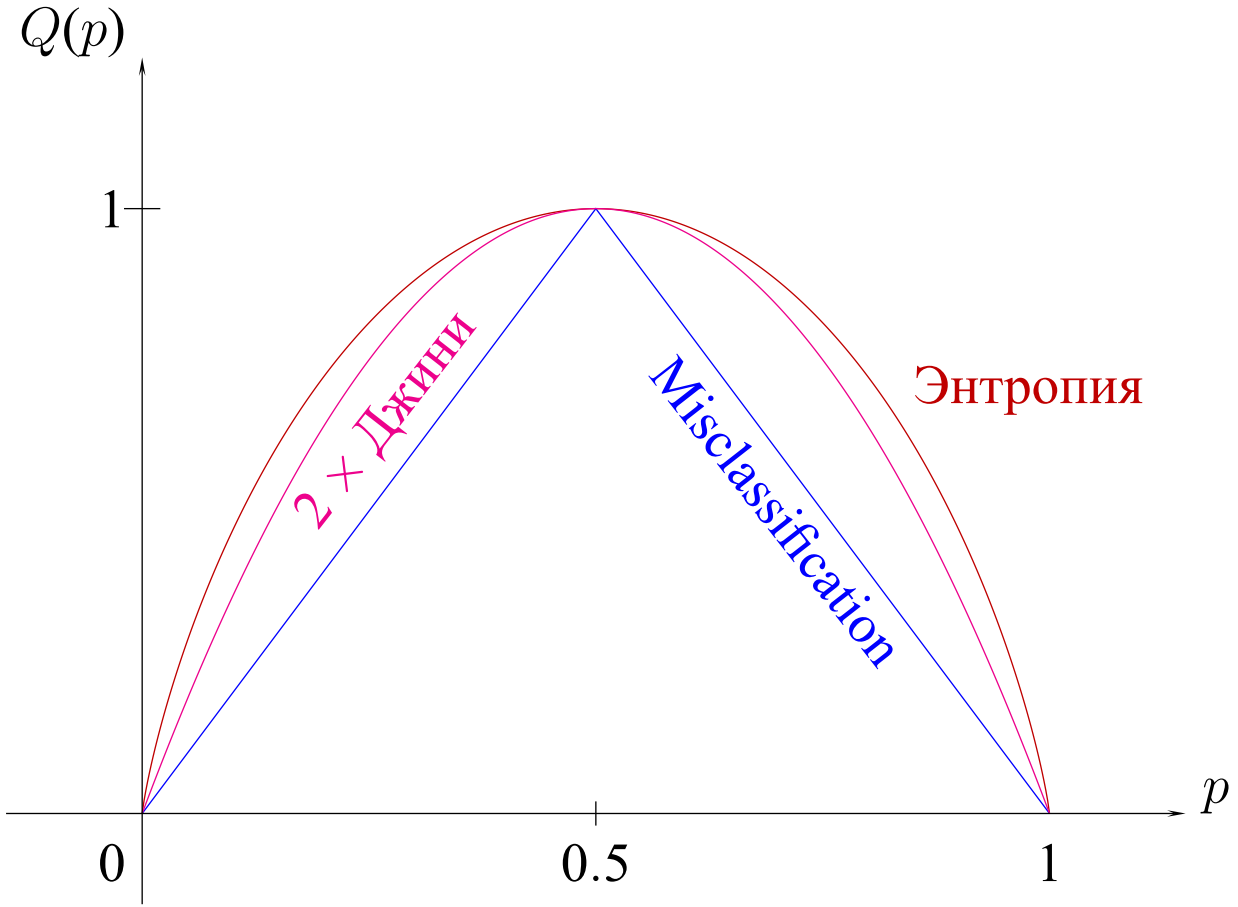
$$Q_m = -\sum_{k=1}^{K} p_{mk} \log p_{mk}.$$

Если $K = 2$, то эти функции равны соответственно

$$1 - \max\{p, 1-p\}, \quad p(1-p), \quad -p\log p - (1-p)\log(1-p),$$

где $p$ — доля объектов 1-го класса, попавших в область $R_m$.

После нормировки:




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.

Алгоритм CART использует другую важную идею — отсечения (pruning)

# 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

*Основной недостаток* — плохая предсказательная (обобщающая) способность.

# 3. Ансамбли решающих правил

*Ансамбль*, или *комитет*, решающих правил,
или *аркинг* (arcing — adaptive reweighting and combining).
Рассмотрим задачу классификации на $K$ классов.

$$\mathscr{Y} = \{1, 2, \ldots, K\}.$$

Пусть имеется $M$ классификаторов («экспертов») $f_1, f_2, \ldots, f_M$

$$f_m : \ \mathscr{X} \to \mathscr{Y}, \qquad f_m \in \mathscr{F}, \qquad (m = 1, 2, \ldots, M)$$

Построим новый классификатор:
простое голосование:

$$f(x) = \max_{k=1,\ldots,K} \sum_{m=1}^{M} I(f_m(x) = k),$$

взвешенное (выпуклая комбинация классификаторов, или «смесь экспертов»):

$$f(x) = \max_{k=1,\ldots,K} \sum_{m=1}^{M} \alpha_m \cdot I(f_m(x) = k), \qquad \alpha_m \geq 0, \qquad \sum_{m=1}^{M} \alpha_m = 1,$$

В задаче восстановления регрессии

простое голосование:

$$f(x) = \frac{1}{M} \sum_{m=1}^{M} f_m(x),$$

взвешенное голосование («смесь экспертов»):

$$f(x) = \sum_{m=1}^{M} \alpha_m \cdot f_m(x), \qquad \alpha_m \geq 0, \qquad \sum_{m=1}^{M} \alpha_m = 1.$$

Пример: $K = 2$, $M = 3$.

Решение принимается с использованием простого голосования.

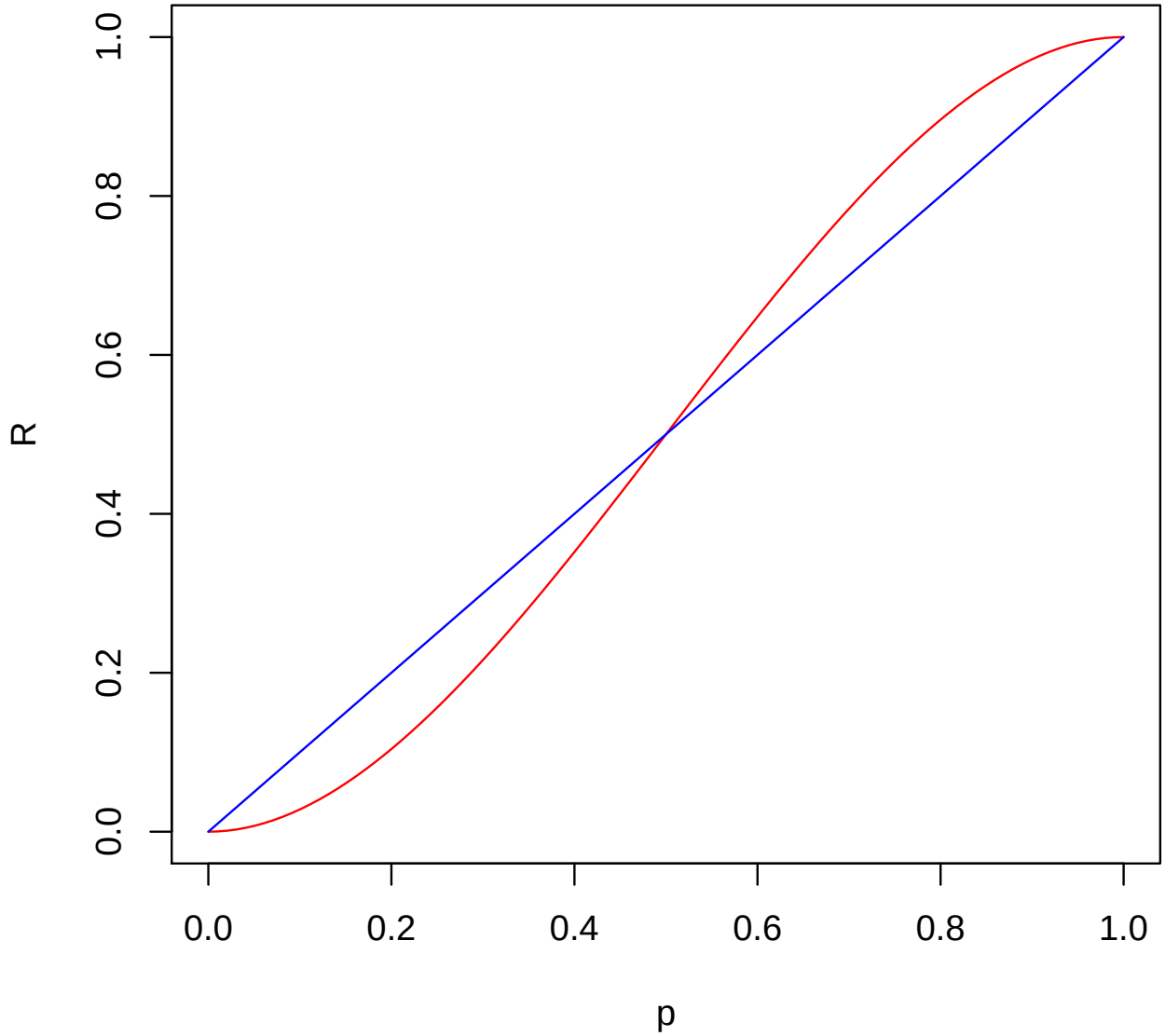
*Пусть классификаторы независимы* (на практике недостижимое требование!).

$p$ — вероятность ошибки каждого отдельного классификатора.

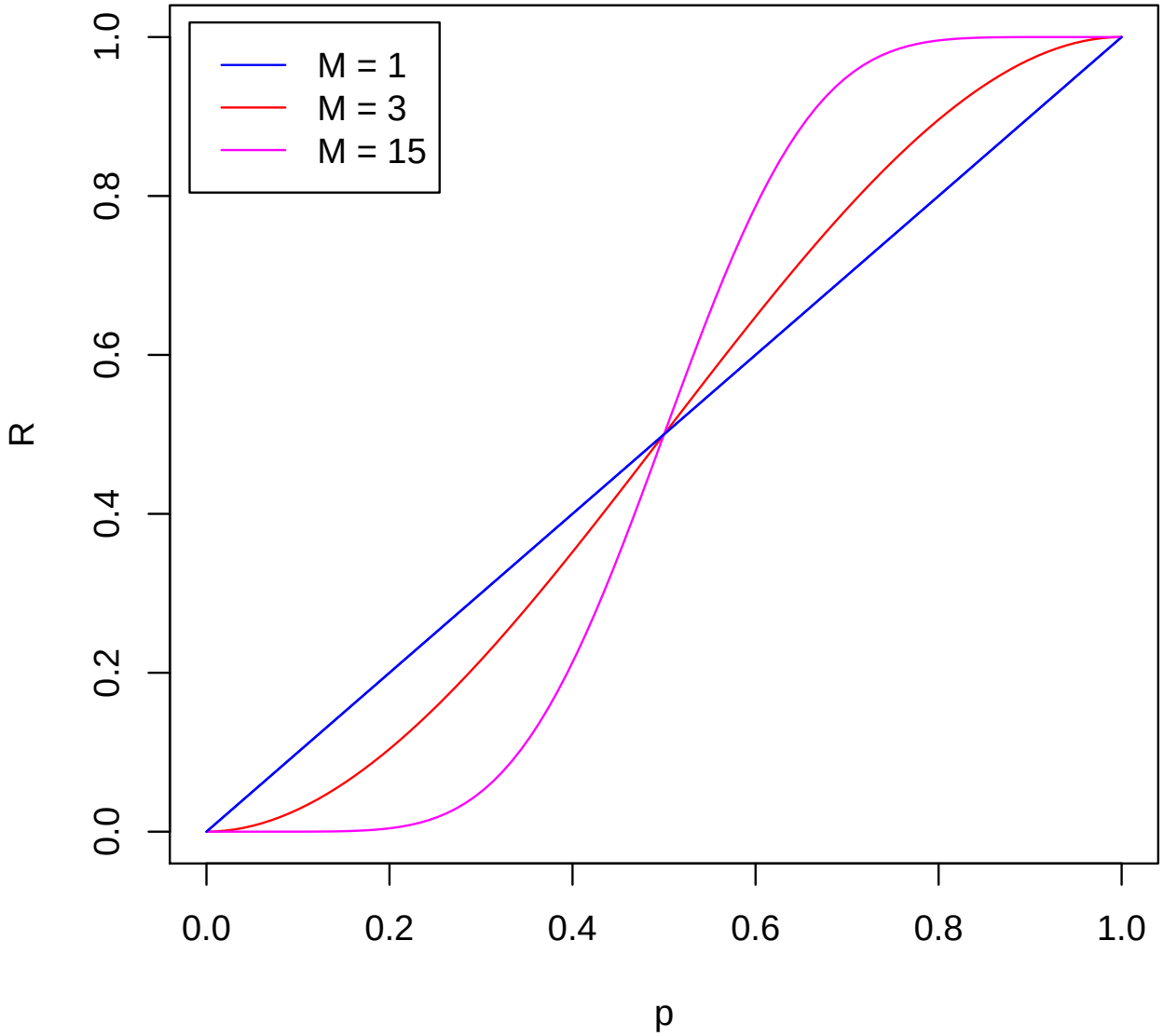
Тогда вероятность ошибки общего решения (ожидаемый риск) равен

$$R = p^3 + 3p^2(1 - p) = 3p^2 - 2p^3.$$

$$R = 3p^2 - 2p^3$$



$M = 1, 3, 15$



$K = 2$

- $p > \frac{1}{2}$ — плохой классификатор,
- $p = \frac{1}{2}$ — toss up a coin
- $p = \frac{1}{2} - \varepsilon$ — 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*: эксперты учатся на ошибках других.

# 4. Бустинг

To boost — *улучшать*, повышать, рекламировать.

Попробуем строить последовательность решающих правил, каждый из которых осведомлен об ошибках предыдущих.

Пусть только два класса $\mathscr{Y} = \{-1, 1\}$.

## 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 $w_1, w_2, \ldots, w_N$.

На первой итерации $w_i = 1/N$ $(i = 1, 2, \ldots, N)$ 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.

Как учитывать веса $w_i$?

— 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

$$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$$

$i$-й элемент с вероятностью $w_i$.

**begin** AdaBoost

Положить $w_i \leftarrow 1/N \qquad (i = 1, 2, \ldots, N)$

**for** $m = 1, 2, \ldots, M$

Найти классификатор $h_m \in \mathscr{F}_m$, минимизирующий ошибку

$$\text{err}_m = \sum_{i=1}^{N} w_i \cdot I\left(y_i \neq h_m(x_i)\right)$$

Вычислить $\beta_m \leftarrow \dfrac{1}{2} \ln \dfrac{1 - \text{err}_m}{\text{err}_m}$

$$w_i' \leftarrow w_i \cdot e^{-\beta_m y_i h_m(x_i)} = w_i \cdot \left(\sqrt{\frac{\text{err}_m}{1 - \text{err}_m}}\right)^{-y_i h_m(x_i)} \qquad (i = 1, 2, \ldots, N)$$

$$w_i \leftarrow w_i' \Big/ \sum_{i=1}^{N} w_i' \qquad (i = 1, 2, \ldots, N)$$

**end**

**return** $f = \text{sign}\, g$, где $g = \sum_{m=1}^{M} \beta_m h_m$

**end**

**Теорема 1** *Если* $\mathrm{err}_m \leq \dfrac{1}{2} - \varepsilon_m$, *то для эмпирической ошибки* $\widehat{R}(f)$ *итогового классификатора, построенного алгоротмом AdaBoost, справедливо*

$$\widehat{R}(f) \leq \prod_{m=1}^{M} 2\sqrt{\mathrm{err}_m(1-\mathrm{err}_m)} = \prod_{m=1}^{M} \sqrt{1-4\varepsilon_m^2} \leq e^{-2\sum\limits_{m=1}^{M}\varepsilon_m^2}.$$

**Теорема 2** *Для ошибки предсказания* $R(f)$ *алгоритма AdaBoost справедливо*

$$R(f) = \widehat{R}(f) + \widetilde{O}\left(\sqrt{\frac{M \cdot v}{N}}\right),$$

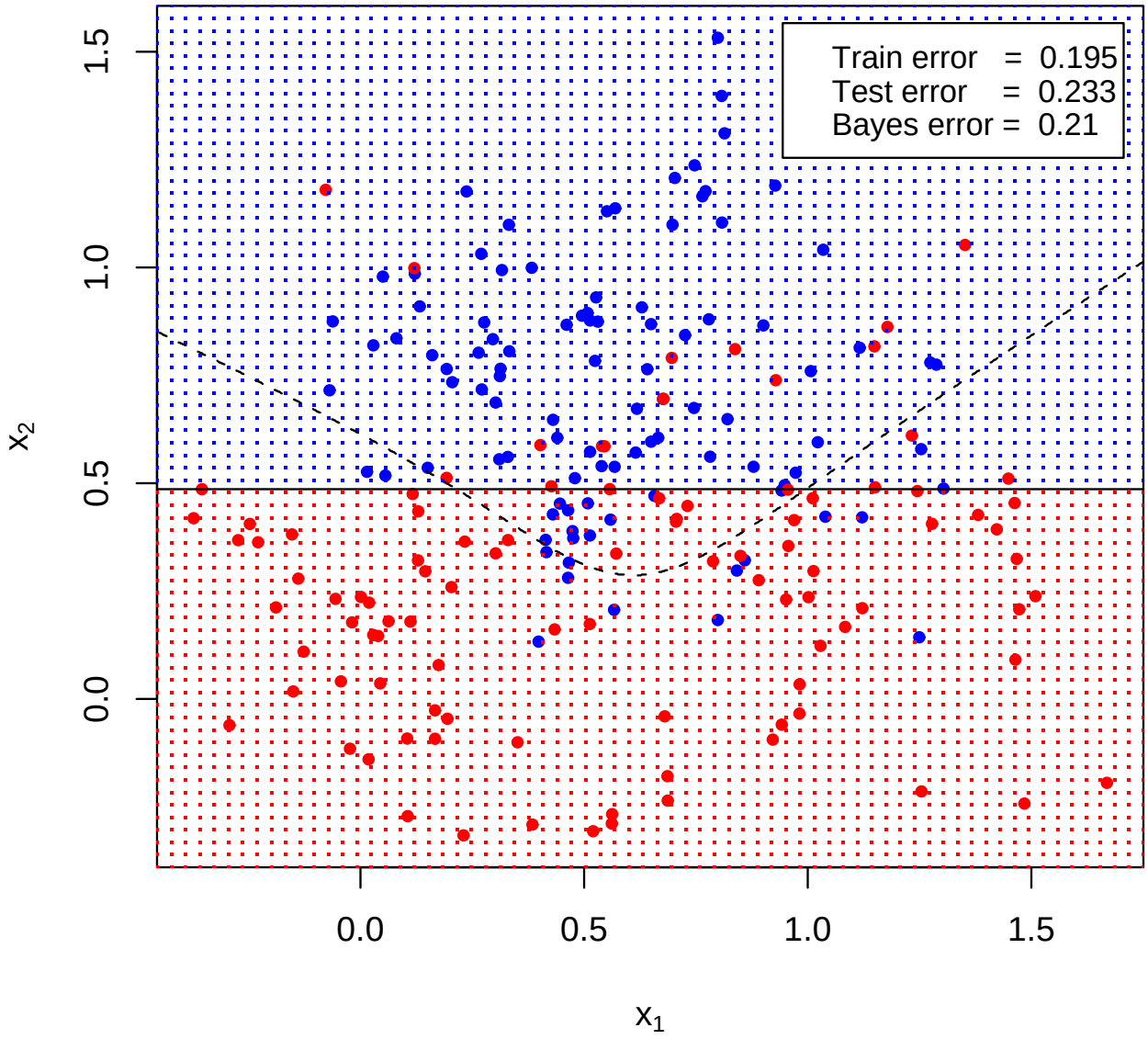
*где* $v = \mathrm{VC}(h_m)$ *— размерность Вапника–Червоненкиса для класса* $\mathscr{F}_m$ $(m = 1, 2, \ldots, M)$.

Большое $M$ — возможно переобучение.

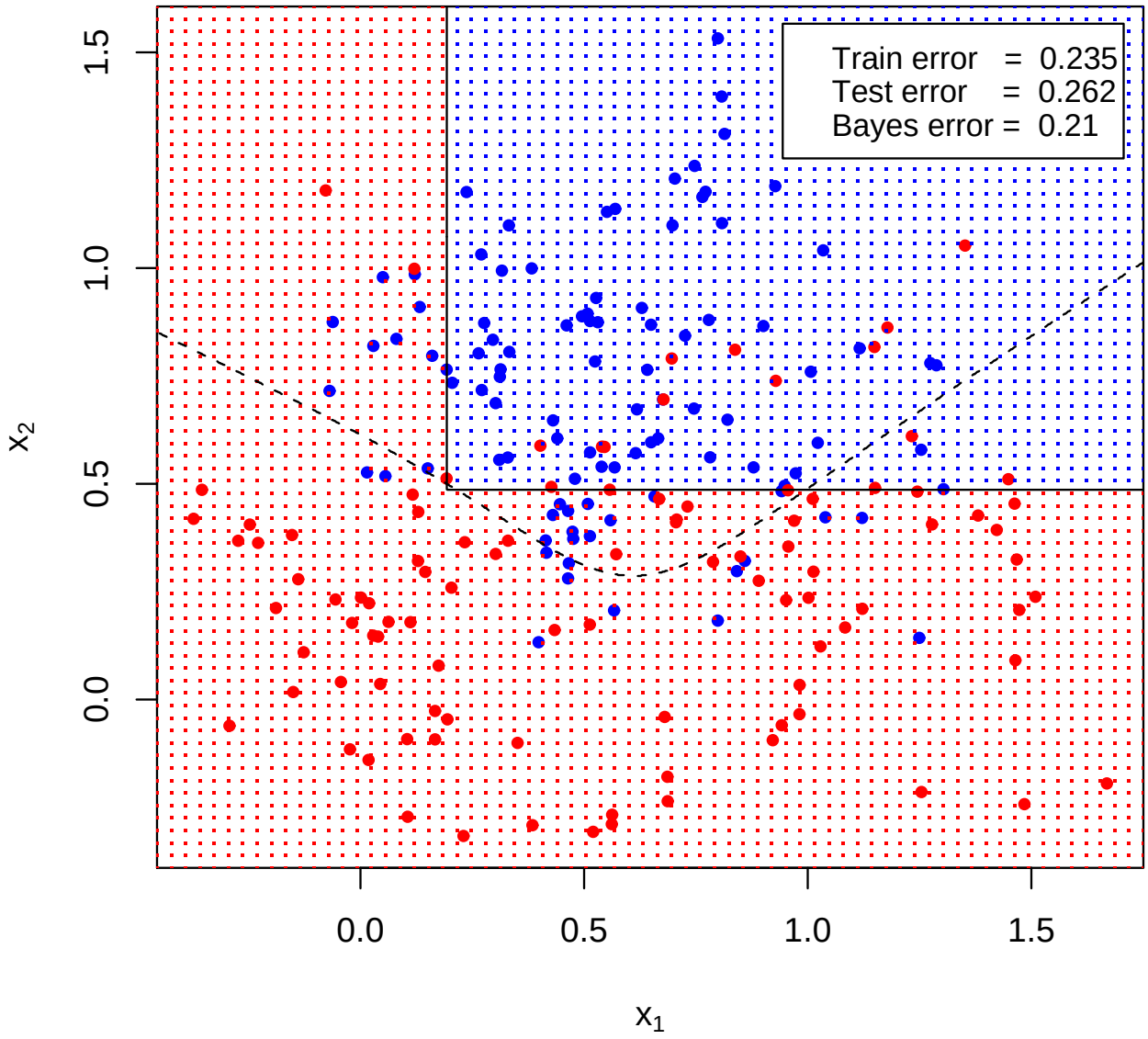
## Пример

Слабые классификаторы — деревья решений высоты 1 (*stumps*)

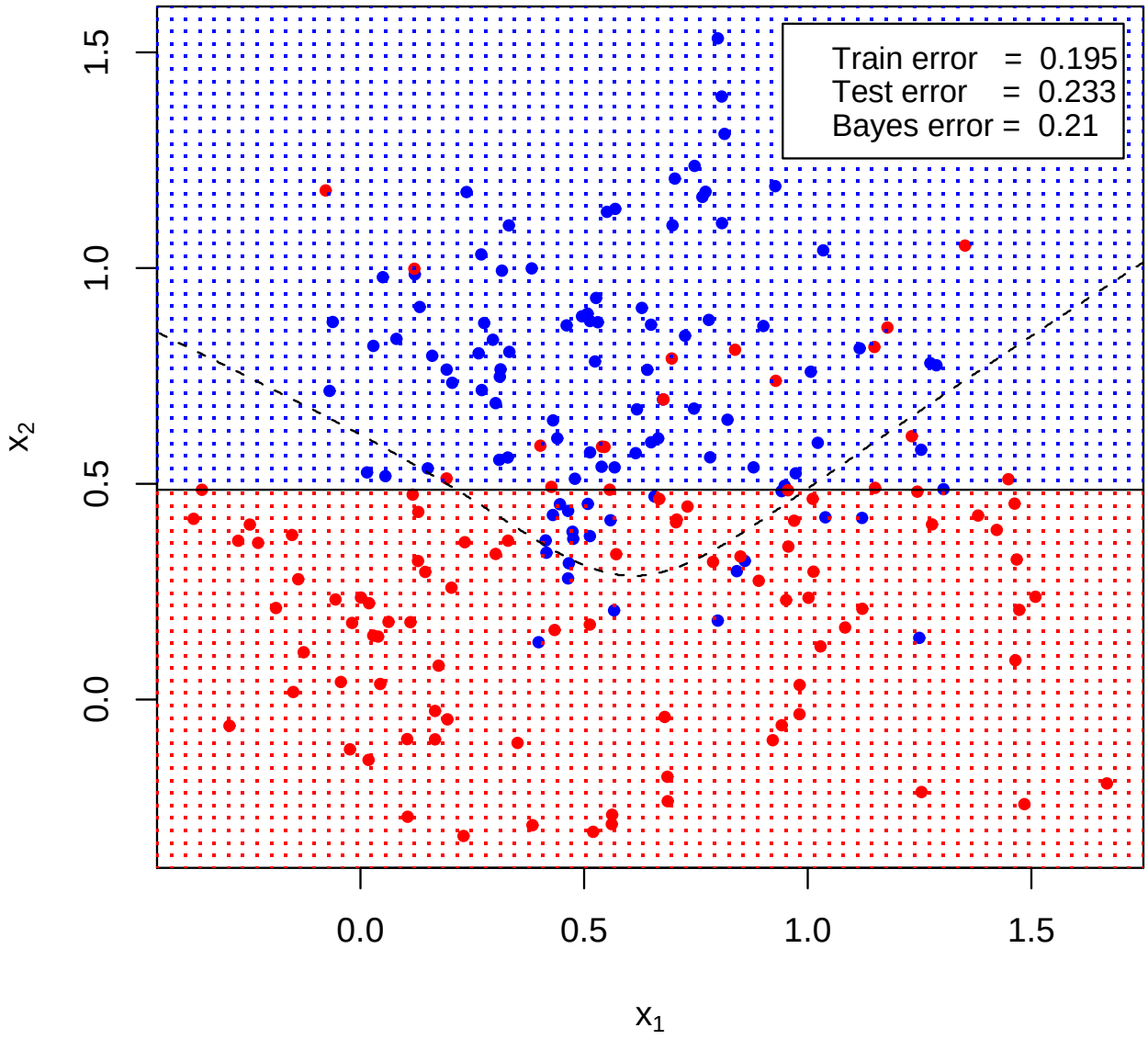
$M = 1$


Train error = 0.195
Test error = 0.233
Bayes error = 0.21
0.0
0.5
1.0
1.5
$x_1$
$x_2$

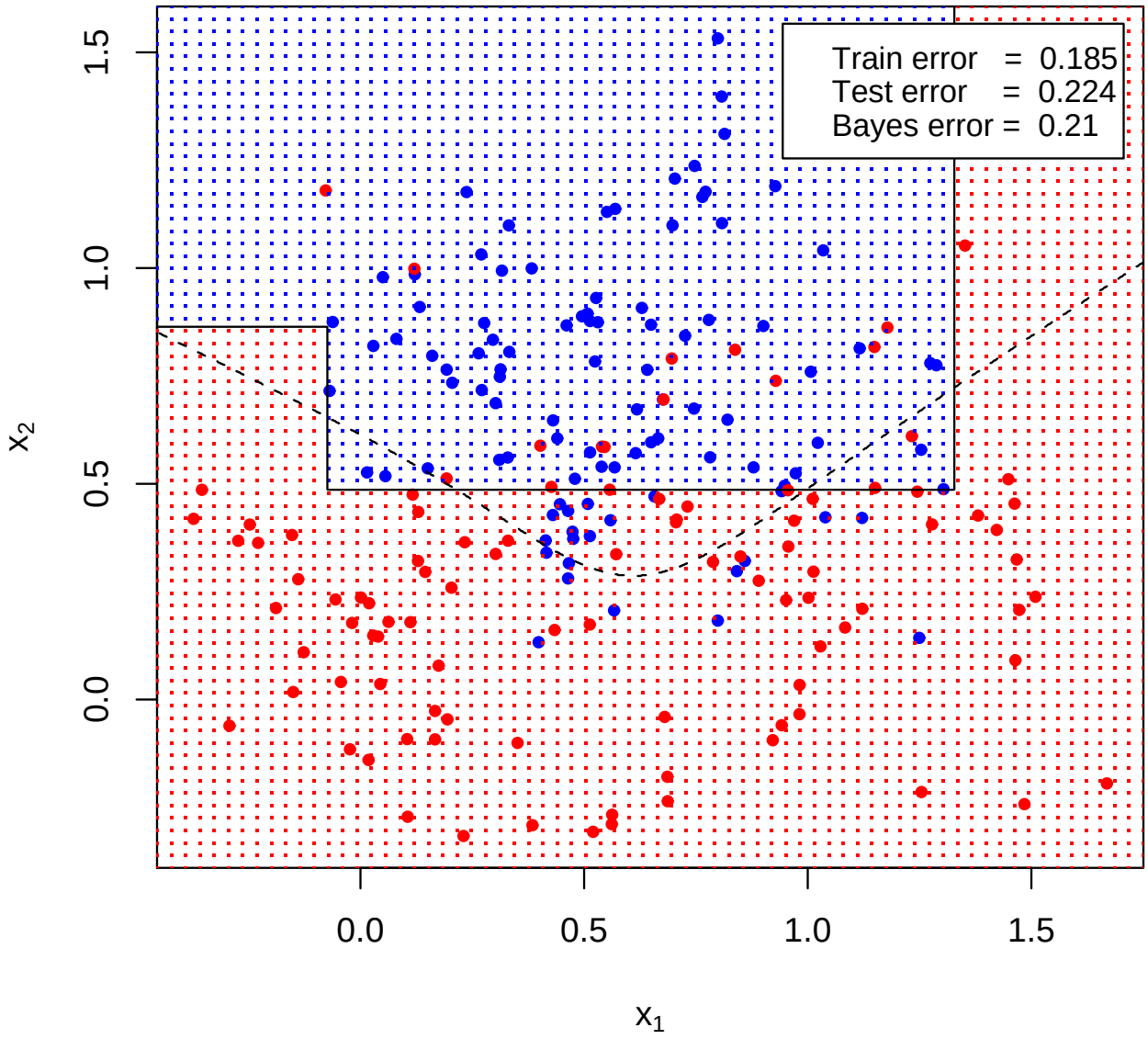
$M = 5$


Train error = 0.235
Test error = 0.262
Bayes error = 0.21
0.0
0.5
1.0
1.5
$x_1$
$x_2$

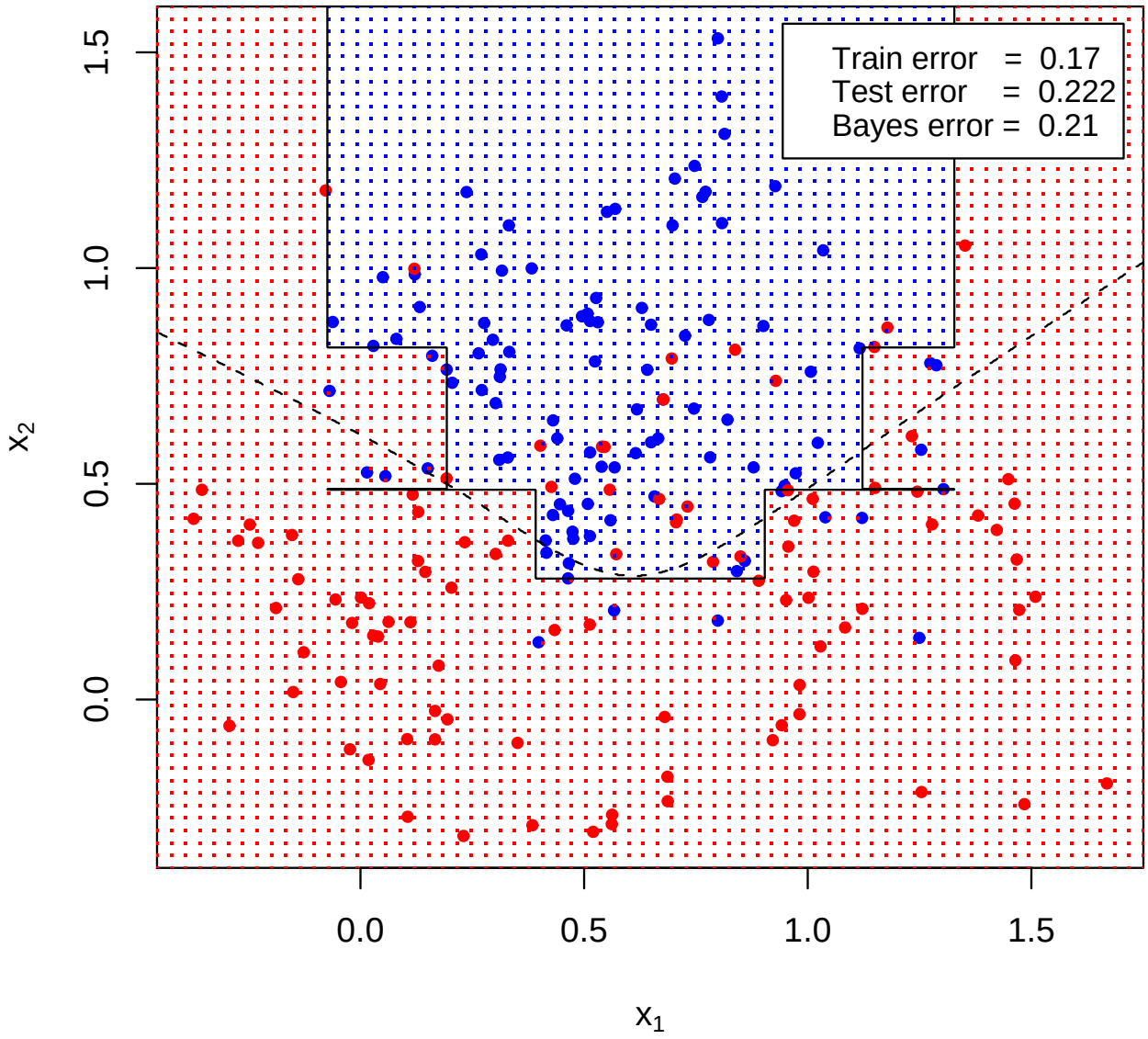
$M = 10$


Train error = 0.195
Test error = 0.233
Bayes error = 0.21
0.0
0.5
1.0
1.5
$x_1$
$x_2$

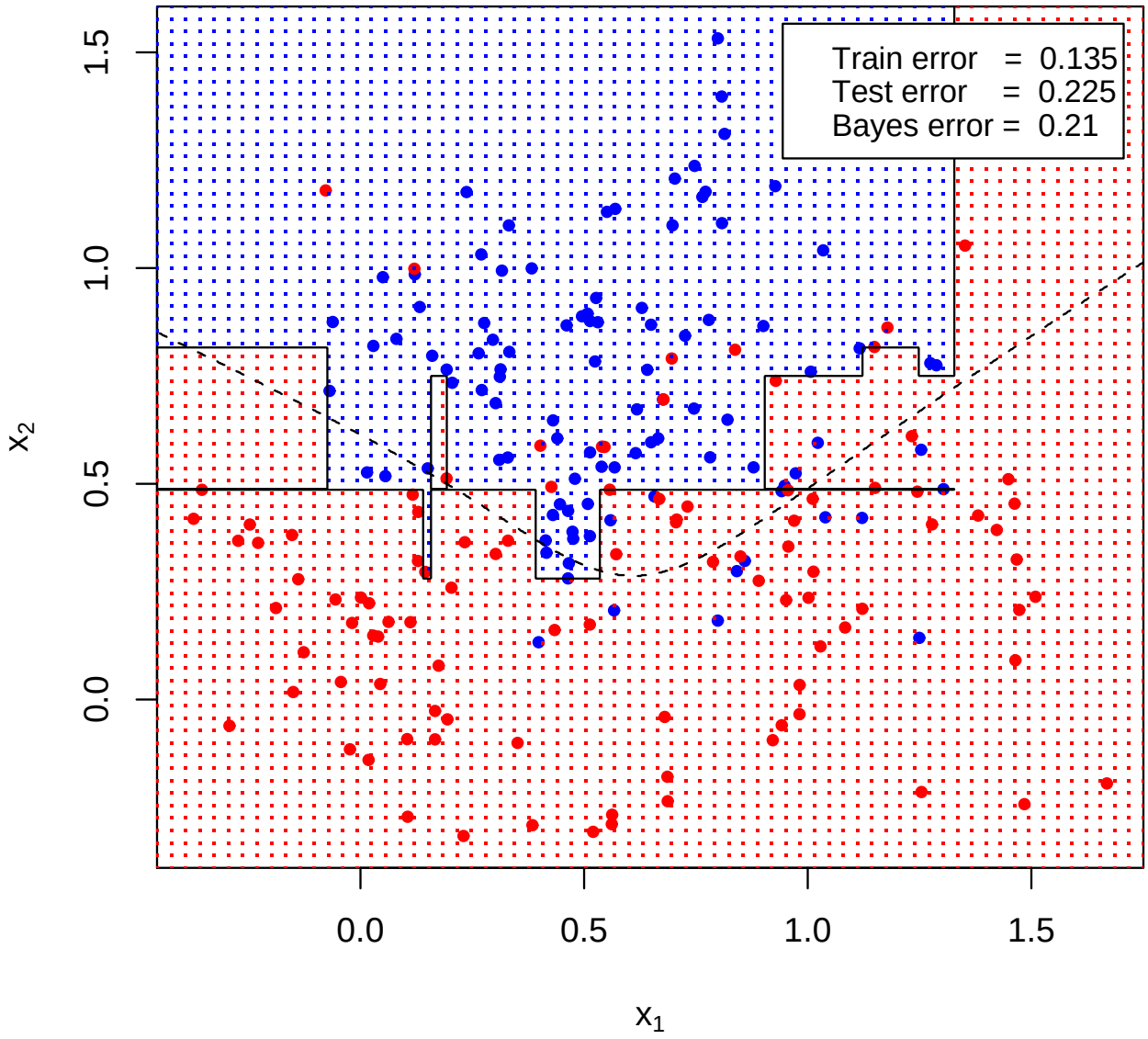
$M = 15$


Train error = 0.185
Test error = 0.224
Bayes error = 0.21
0.0
0.5
1.0
1.5
$x_1$
$x_2$

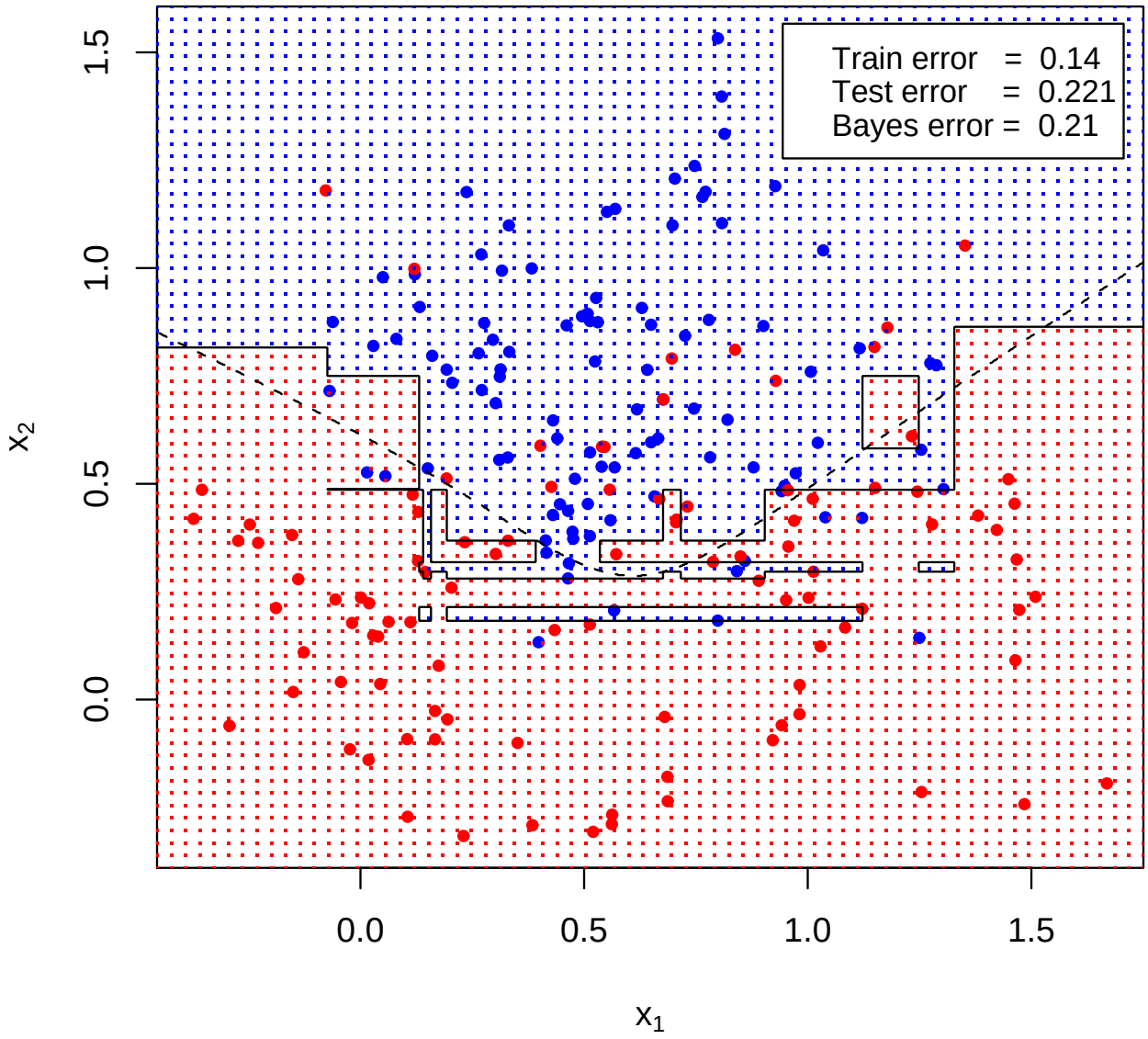
$M = 20$


Train error = 0.17
Test error = 0.222
Bayes error = 0.21
0.0
0.5
1.0
1.5
$x_1$
$x_2$

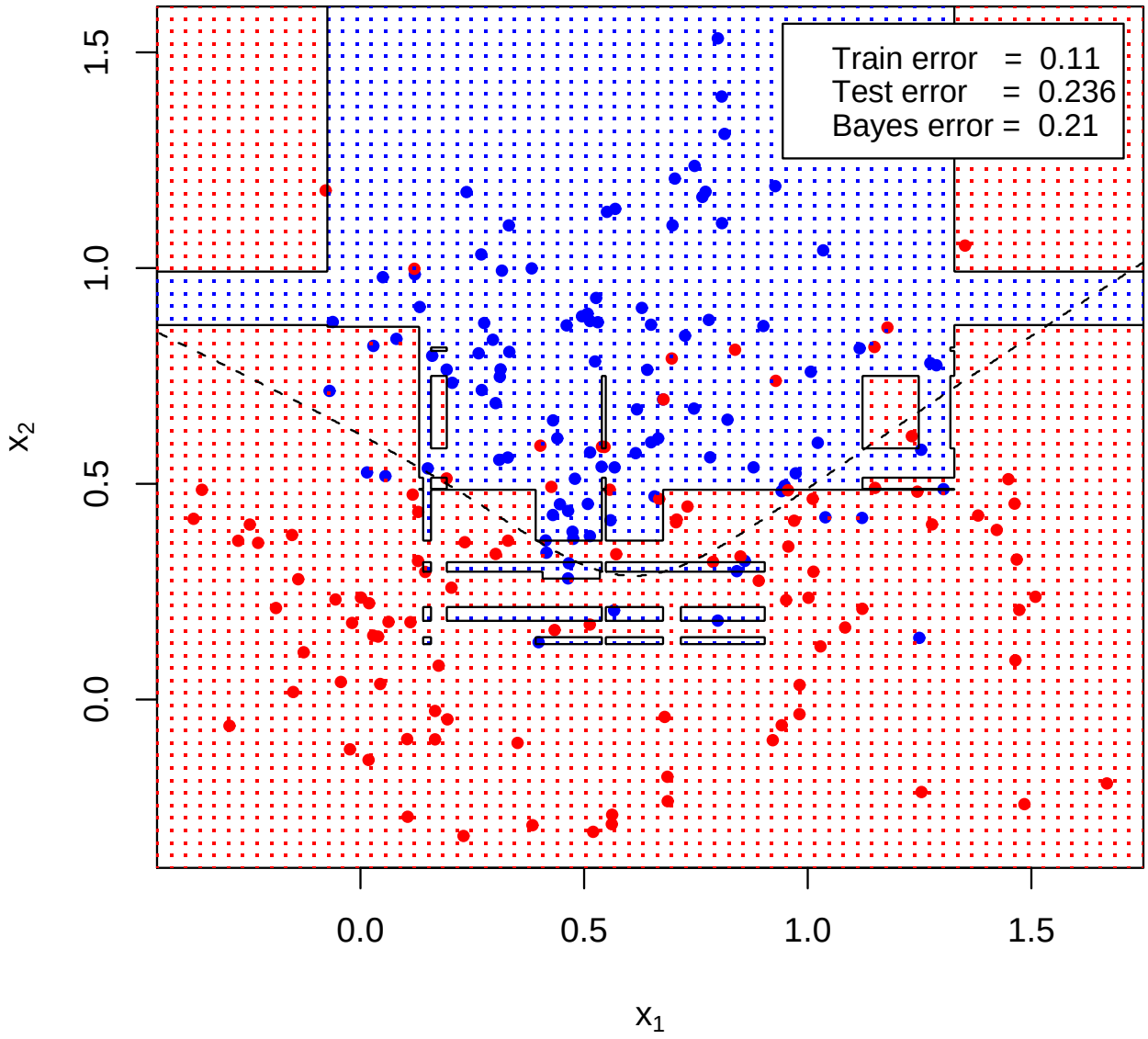
$M = 30$


Train error = 0.135
Test error = 0.225
Bayes error = 0.21
0.0
0.5
1.0
1.5
$x_1$
$x_2$

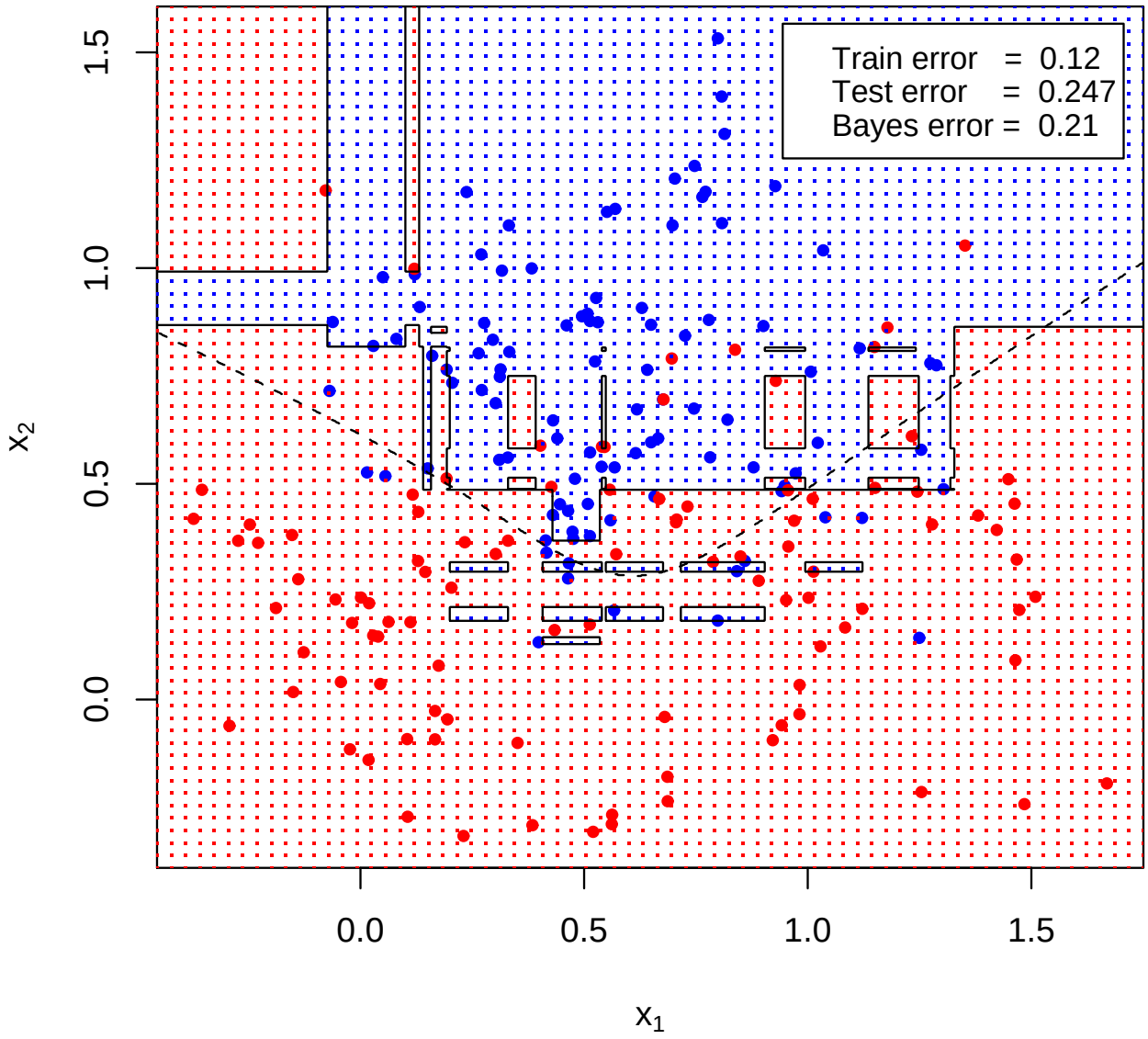
$M = 40$


Train error = 0.14
Test error = 0.221
Bayes error = 0.21
0.0
0.5
1.0
1.5
$x_1$
$x_2$

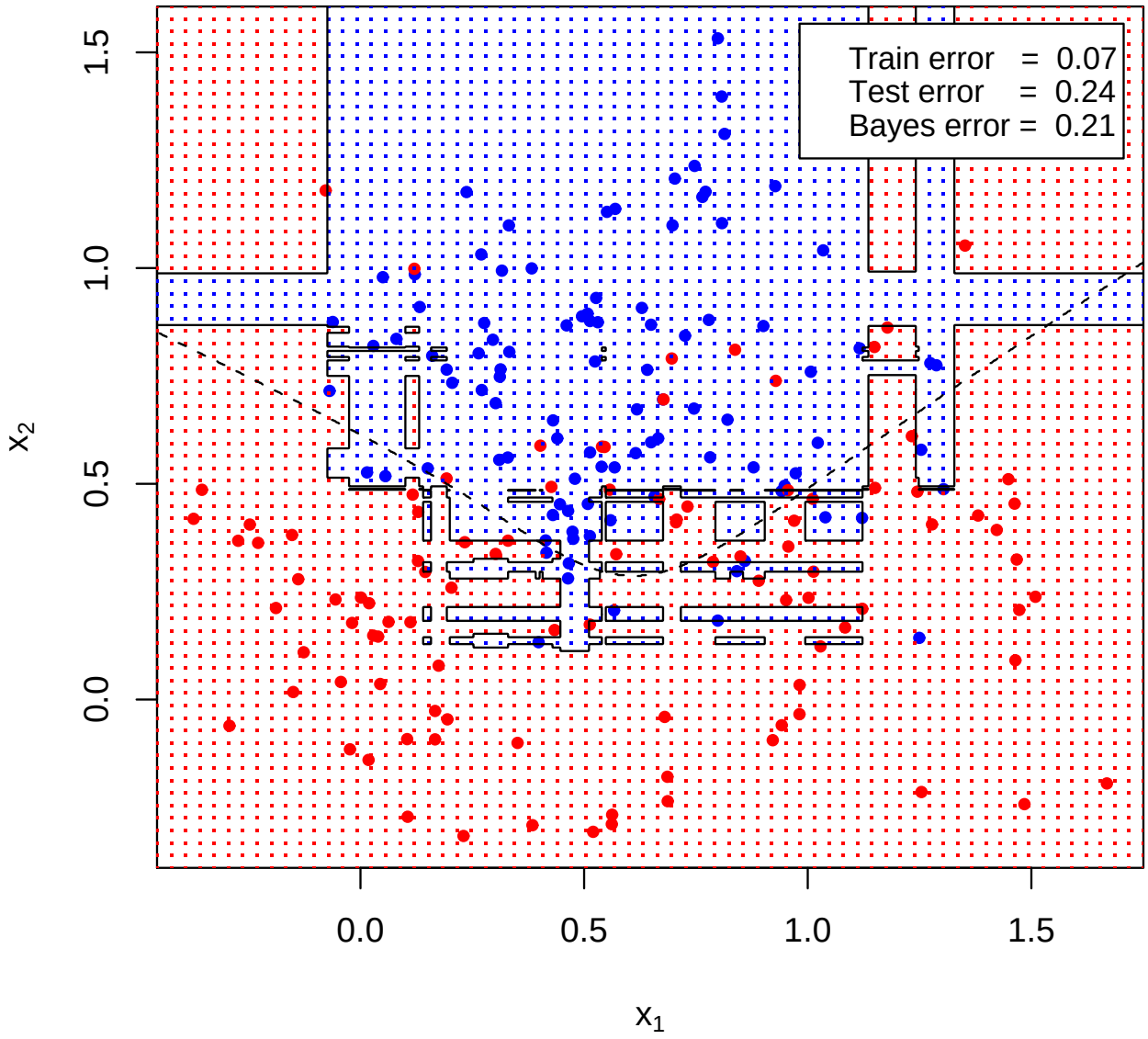
$M = 50$


Train error = 0.11
Test error = 0.236
Bayes error = 0.21
0.0
0.5
1.0
1.5
$x_1$
$x_2$

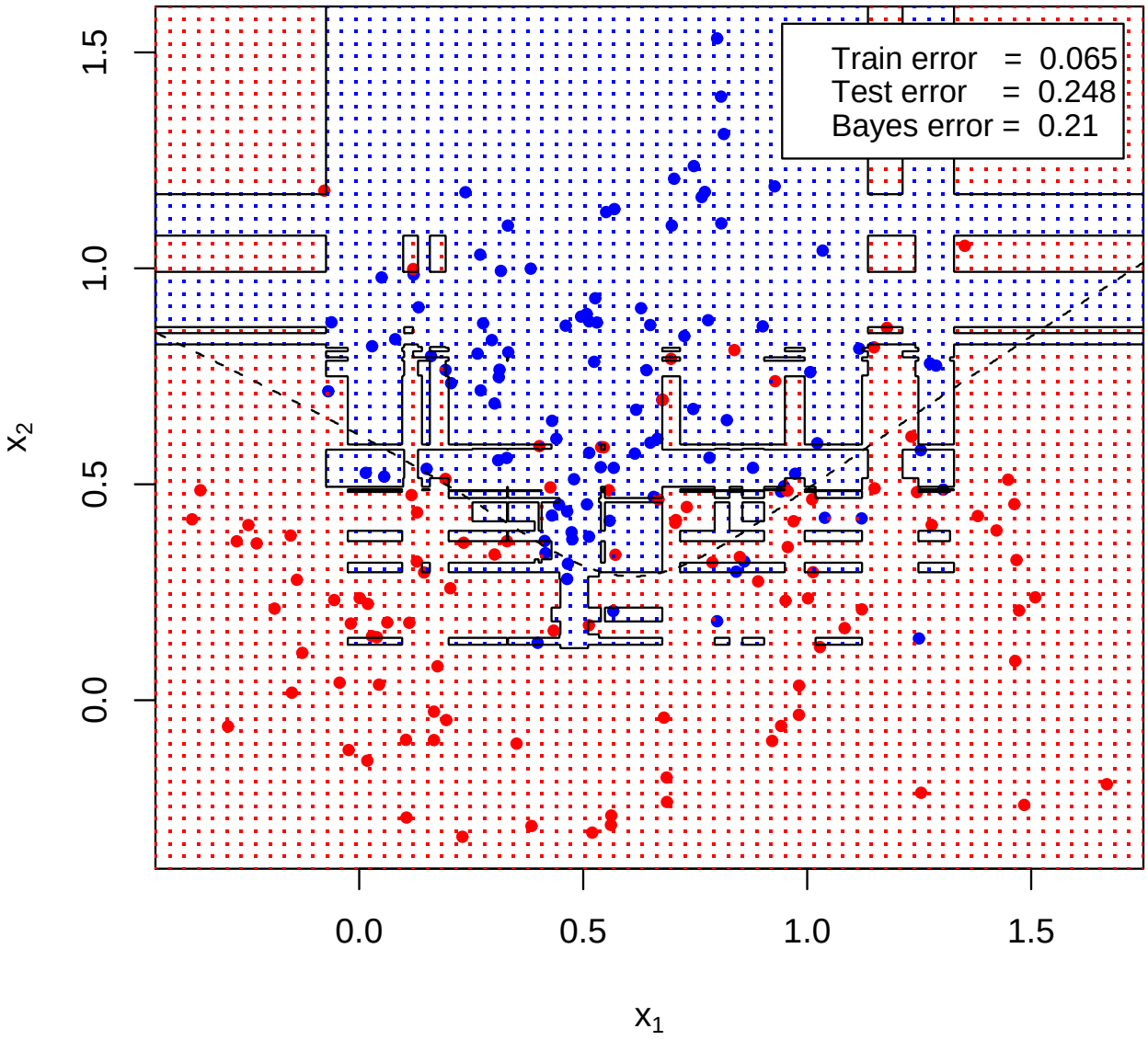
$M = 60$


Train error = 0.12
Test error = 0.247
Bayes error = 0.21
0.0
0.5
1.0
1.5
$x_1$
$x_2$

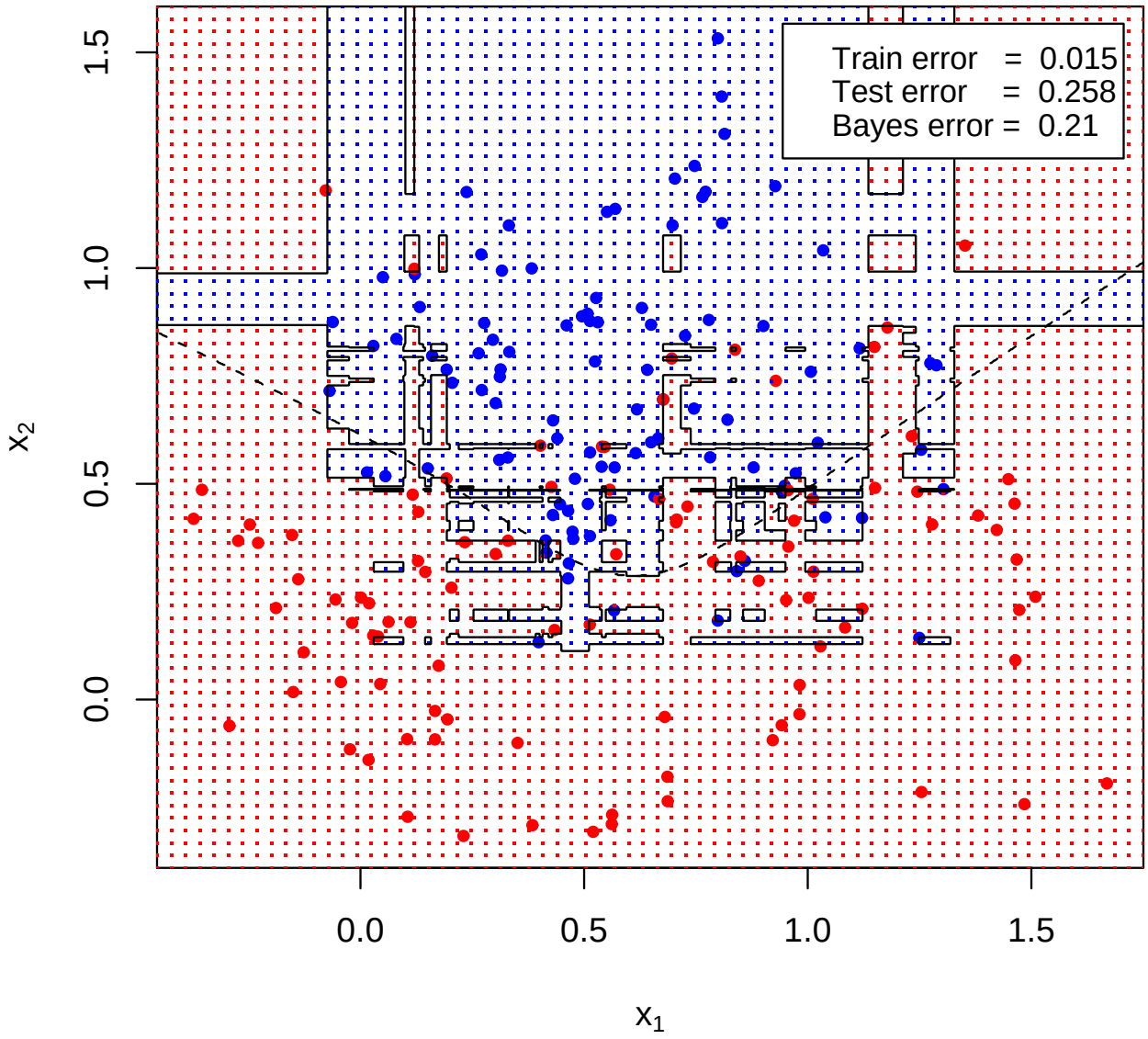
$M = 100$


Train error = 0.07
Test error = 0.24
Bayes error = 0.21
0.0
0.5
1.0
1.5
$x_1$
$x_2$

$M = 150$


Train error = 0.065
Test error = 0.248
Bayes error = 0.21
0.0
0.5
1.0
1.5
$x_1$
$x_2$

$M = 200$


Train error = 0.015
Test error = 0.258
Bayes error = 0.21
0.0
0.5
1.0
1.5
$x_1$
$x_2$

# Пример. Классификации рукописных цифр

Выборка размера 1934 разбита случайно поровну на обучающую и тестовую.

В качестве слабых классификаторов использовались деревья решений, высоты $\leq 10$.

| $M$ | *Ошибка* | |
|---|---|---|
| | *на обучающей выборке* | *на тестовой выборке* |
| 5 | 0.070 | 0.121 |
| 10 | 0.017 | 0.074 |
| 20 | 0.001 | 0.056 |
| 30 | 0 | 0.048 |
| 40 | 0 | 0.044 |

| *Алгоритм* | *Ошибка* | |
|---|---|---|
| | *на обучающей выборке* | *на тестовой выборке* |
| Машина опорных векторов | 0 % | 2.1 % |
| Метод ближайшего соседа | 0 % | 3.1 % |
| Нейронная сеть | 0 % | 4.7 % |
| AdaBoost | 0 % | 4.8 % |

AdaBoost.M1 — вариант алгоритма для задачи классификации с $K > 2$

AdaBoost.MH — другой алгоритм для задачи классификации, $K > 2$

Real AdaBoost — алгоритм для восстановления регрессии

Алгоритм AdaBoost и его варианты применяется с большим успехом в различных прикладных задачах.

Например, прорыв в решении задачи детектирования лица на изображении был осуществлен путем комбинирования каскадного алгоритма детектирования с AdaBoost [Viola, Jones, 2001].

# 5. Градиентный бустинг деревьев решений

GBT — Gradient Boosting Trees [Friedman, 1999]

- *Задача классификации*, $\mathscr{Y} = \{-1, 1\}$. Решающее правило: $f(x) = \operatorname{sign} h(x)$.

  $h$ характеризует «надежность» предсказания.

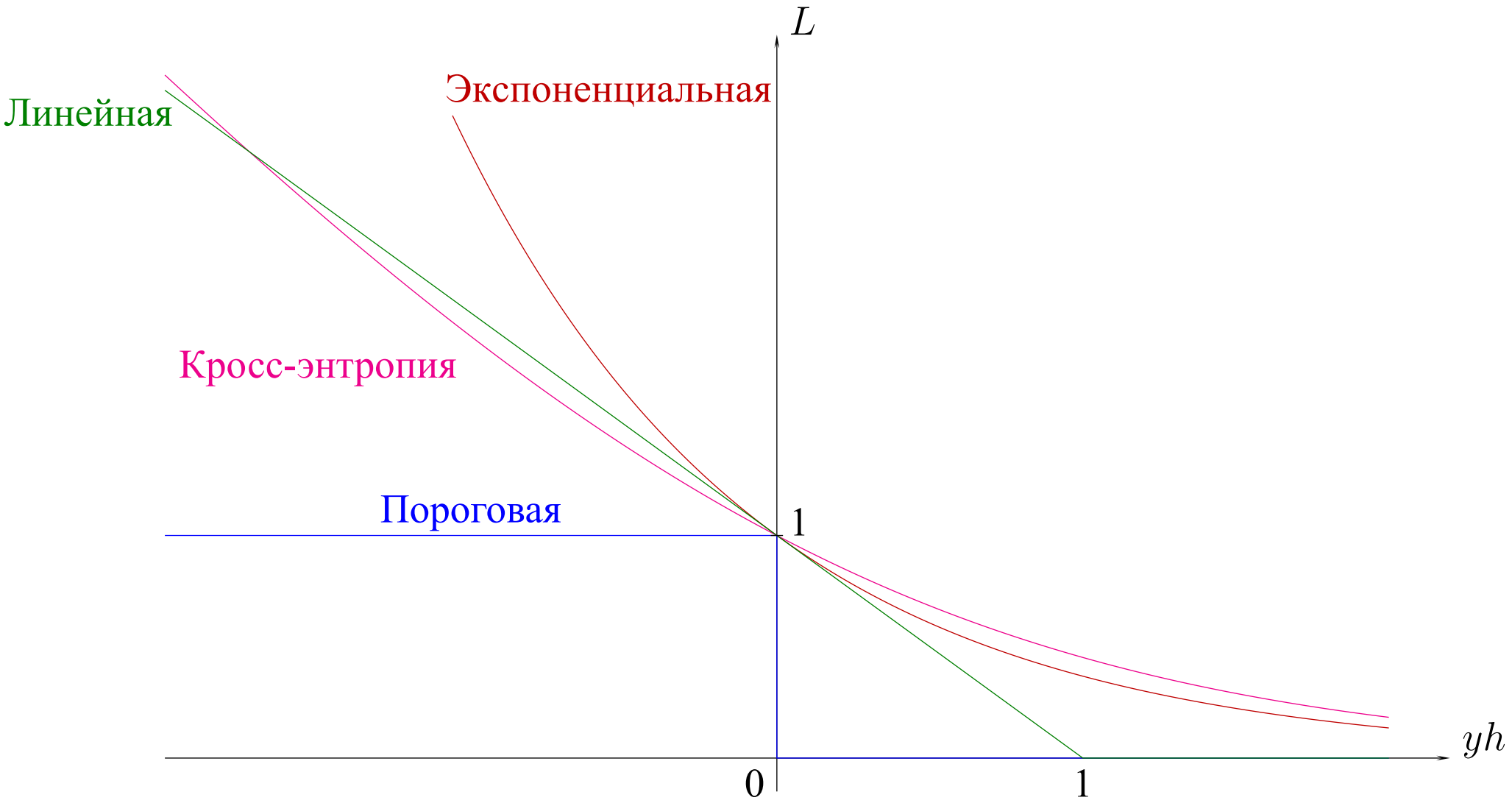
  Штрафные функции (функции потерь):

  - Пороговая функция: $L(y, h) = I(yh < 0)$
  - Линейная функция: $L(y, h) = (1 - yh)I(yh < 1)$
  - Кросс-энтропия: $L(y, h) = \log_2(1 + e^{-yh})$
  - Экспоненциальная функция: $L(y, h) = e^{-yh}$

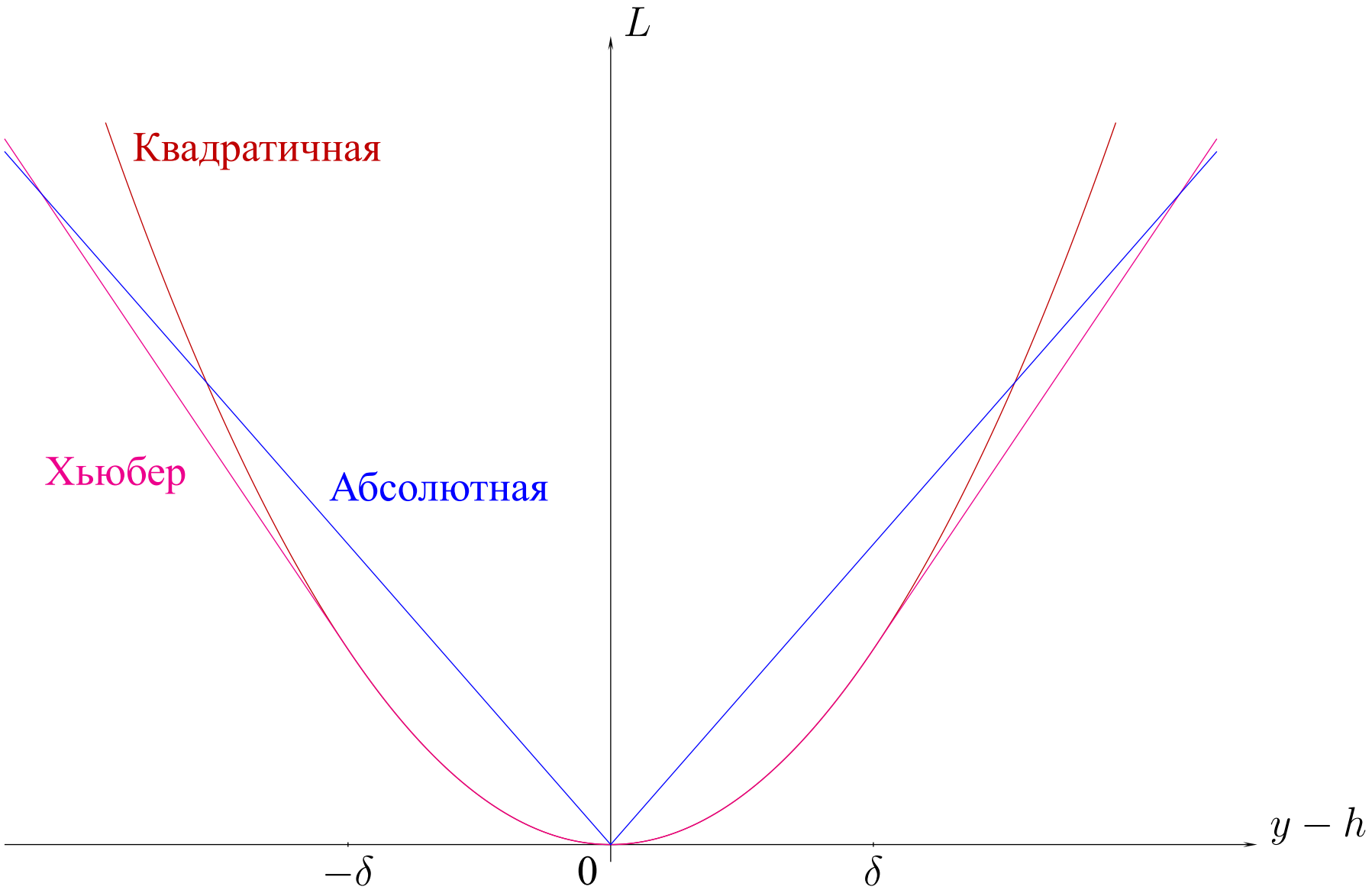
- *Задача восстановления регрессии*. Решающее правило: $f(x) = h(x)$.

  Штрафные функции:

  - Квадратичная функция: $L(y, h) = (y - h)^2$
  - Абсолютная: $L(y, h) = 2|y - h|$
  - Функция Хьюбера: $(y - h)^2 \cdot I(|y - h| \leq \delta) + \delta \cdot (2|y - h| - \delta) \cdot I(|y - h| > \delta)$


L
Экспоненциальная
Линейная
Кросс-энтропия
Пороговая
1
yh
0
1

$\delta = 1.3$



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

$$L(h) = \sum_{i=1}^{N} L\big(y_i,\ h(x_i)\big),$$

где $h$ — функция известного вида.

Опуская последнее ограничение, получаем задачу минимизации

$$\min_{\mathbf{h}} L(\mathbf{h}),$$

где $\mathbf{h} = \big(h(x_1), h(x_2), \ldots, h(x_N)\big) \in \mathbf{R}^N$.

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

$$\mathbf{h}_M = \mathbf{h}_0 + \sum_{m=1}^{M} \mathbf{s}_m, \qquad \mathbf{s}_m \in \mathbf{R}^N.$$

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:

$$g_{im} = \left.\frac{\partial L(y_i,\, h(x_i))}{\partial h(x_i)}\right|_{h(x_i)=h_{m-1}(x_i)} \qquad (i = 1, 2, \ldots, N).$$

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 $h_0 \leftarrow \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{N} L(y_i,\, \gamma)$

**for** $m = 1, 2, \ldots, M$

**for** $i = 1, 2, \ldots, N$

$$g_{im} \leftarrow - \left. \frac{\partial L\left(y_i,\, h(x_i)\right)}{\partial h(x_i)} \right|_{h=h_{m-1}}$$

**end**

Обучить базовую модель $T_m$ на выборке $(x_1, g_{1m}), (x_2, g_{2m}), \ldots, (x_N, g_{Nm})$.

Найти длину шага: $\rho_m = \underset{\rho}{\operatorname{argmin}}\, L\big(h_{m-1} - \rho T_m(x)\big)$.

Обновить модель: $h_m = h_{m-1} - \rho_m T_m(x)$.

**end**

**return** $f_M$

# 6. Баггинг

*Bagging* — от bootstrap aggregation [Breiman, 1994]

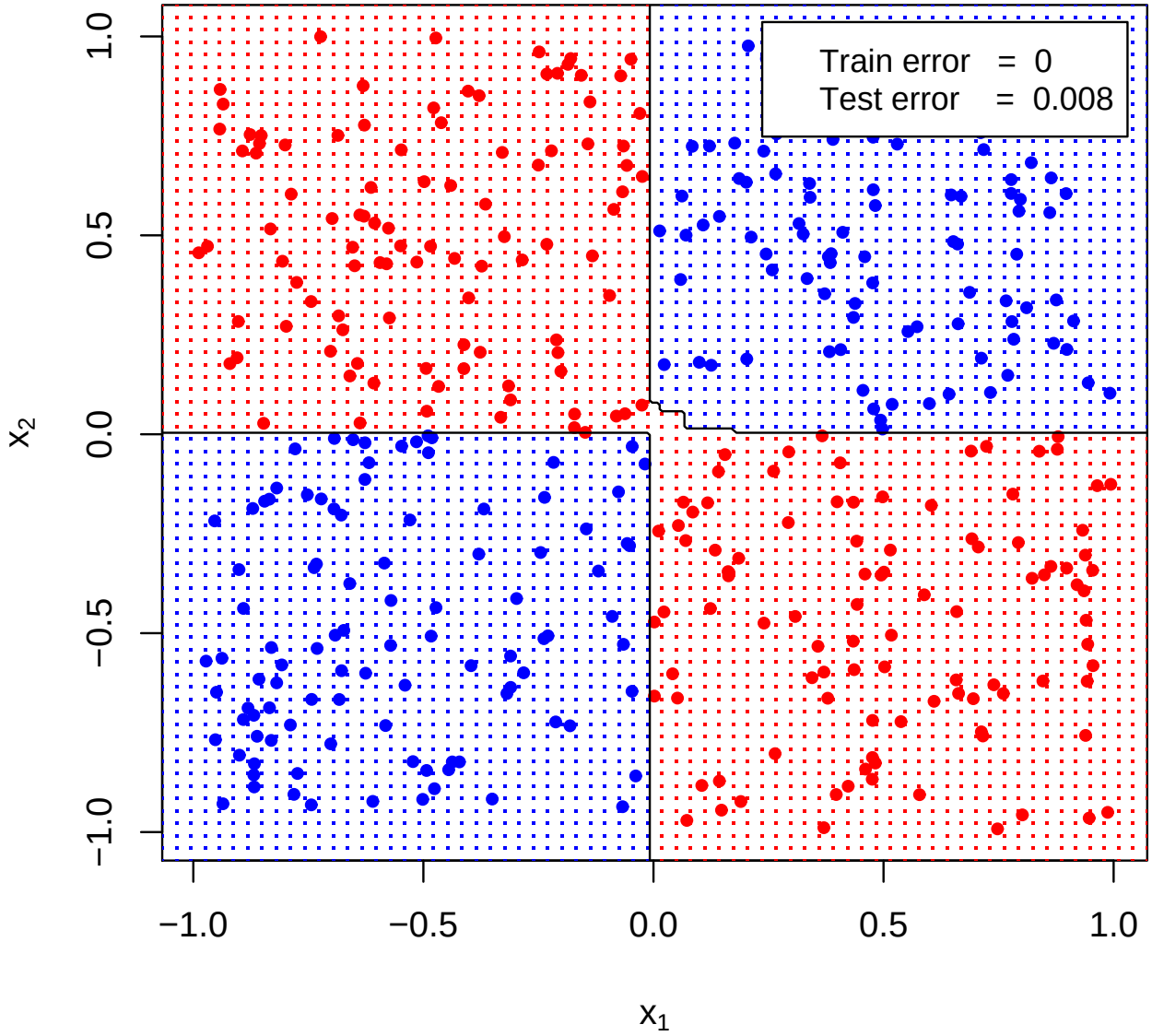
Классификатор $f_m$ обучается на bootstrap–выборке ($m = 1, 2, \ldots, M$).

Финальный классификатор $f$ — функция голосования:

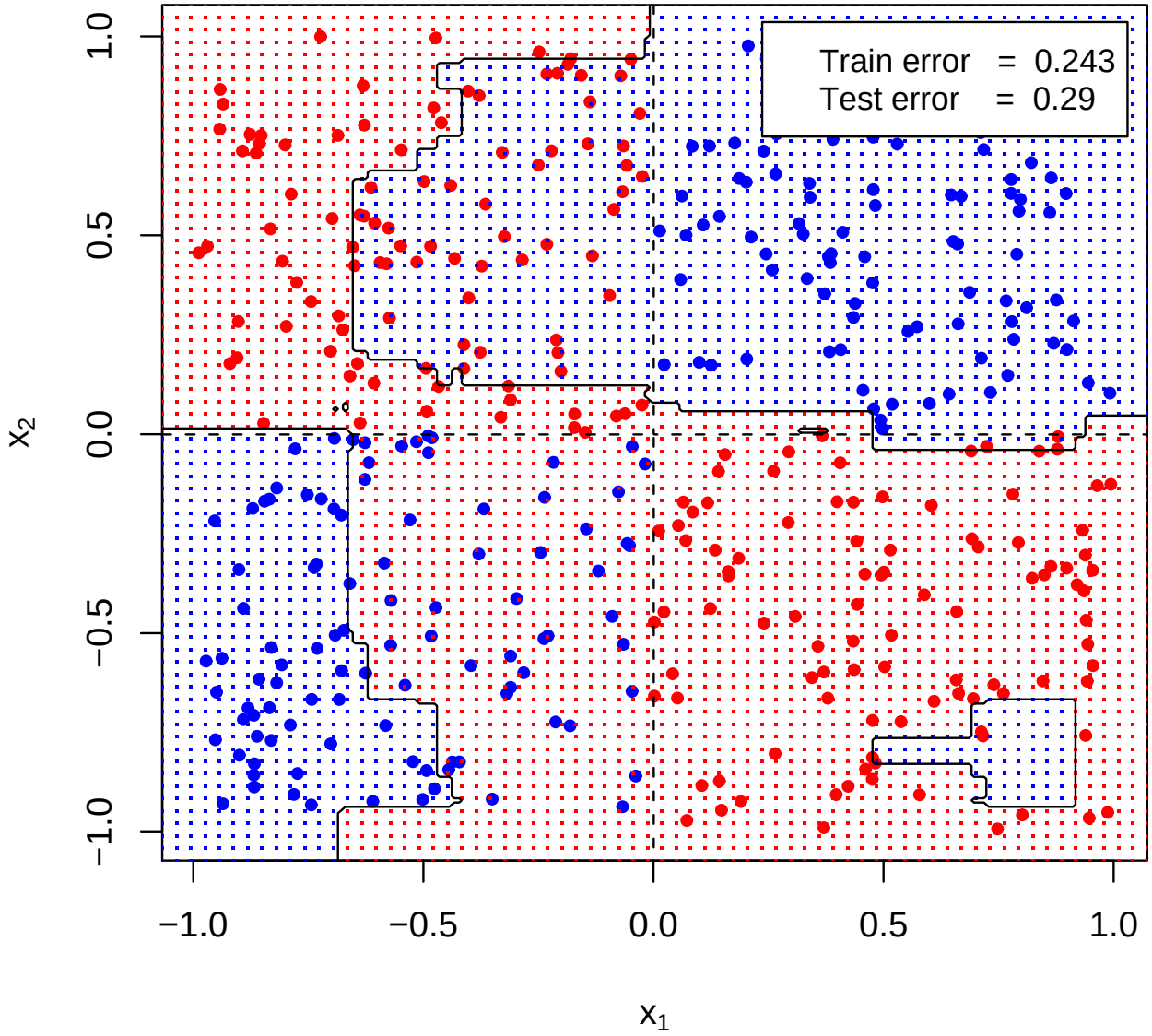
$$f(x) = \max_{k=1,\ldots,K} \sum_{m=1}^{M} I\big(f_m(x) = k\big).$$

Позволяет бороться с неустойчивостью классификаторов $f_m$.

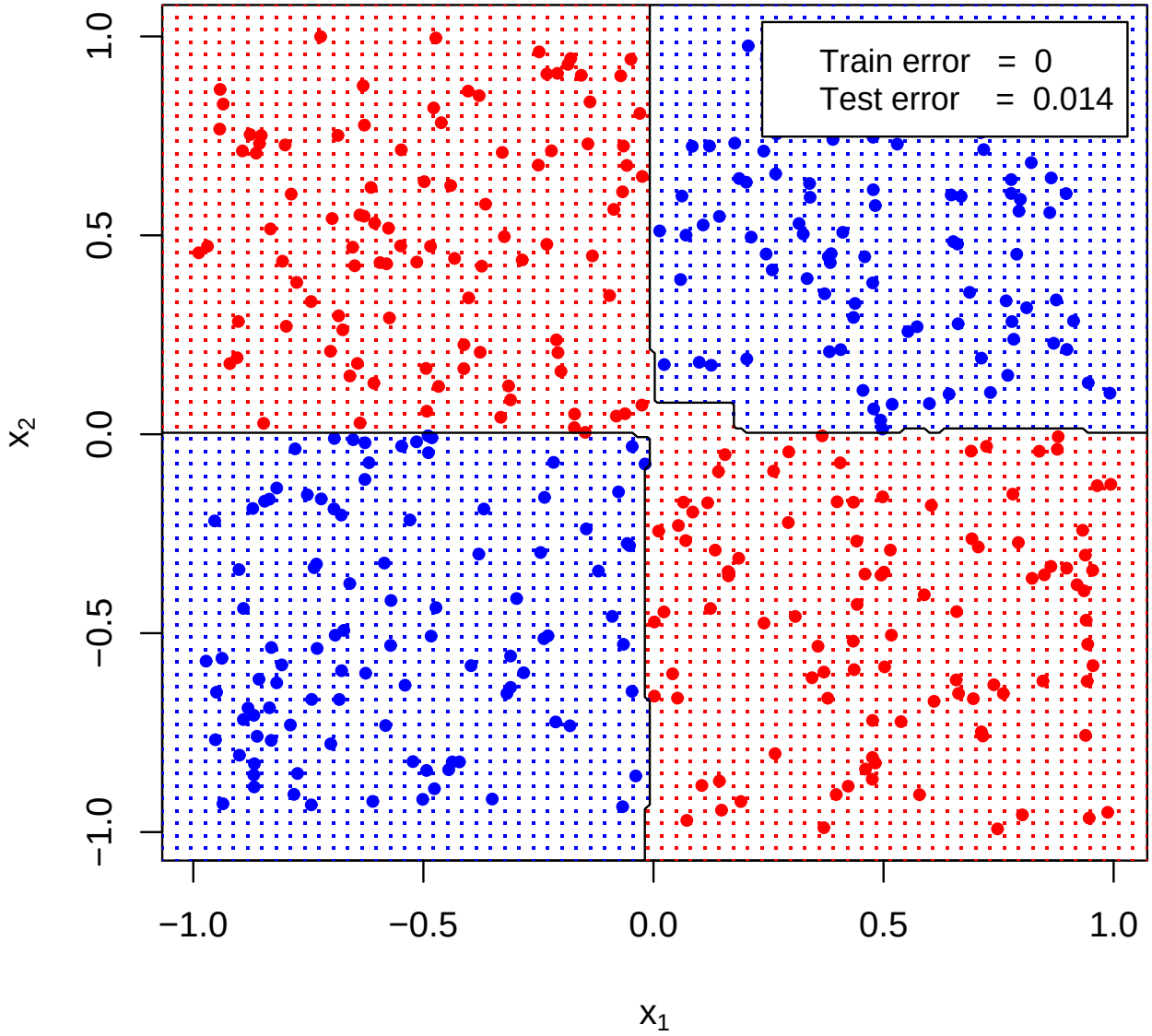
# AdaBoost — деревья решений высоты 2



# Bagging — деревья решений высоты 2



# Bagging — деревья решений высоты 3



# 7. Случайный лес

Random forests [Breiman, 2001]

Ансамбль параллельно обучаемых независимых деревьев решений.

Независимое построение определенного количества деревьев:

Генерация случайной подвыборки из обучающей выборки (50–70% от размера всей обучающей выборки) и 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).

# 8. 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

| *Задача* | $N$ | $d$ (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 |