

# Data Mining in Action

Лекция 2. Алгоритмы supervised learning



# Наиболее часто используемые методы

- I. Линейные модели
- II. Решающие деревья
- III. Ансамбли решающих деревьев

# I. Линейные модели

# Линейные модели

1. Идея линейной классификации
2. Функции потерь
3. Градиентный спуск и стохастический градиент
4. Регуляризация
5. Стандартные линейные классификаторы
6. Линейная регрессия
7. Библиотеки

## Пример: выходить из дома или нет

Признаки (1/0):

1. Вы свободны в данный момент
2. Вам хочется где-то поесть
3. Вам хочется спать
4. Вам хочется увидеться с друзьями

## Пример: выходить из дома или нет

Признаки (1/0):

1. Вы свободны в данный момент **+1**
2. Вам хочется где-то поесть
3. Вам хочется спать
4. Вам хочется увидеться с друзьями

## Пример: выходить из дома или нет

Признаки (1/0):

1. Вы свободны в данный момент **+1**
2. Вам хочется где-то поесть **+2**
3. Вам хочется спать
4. Вам хочется увидеться с друзьями

## Пример: выходить из дома или нет

Признаки (1/0):

1. Вы свободны в данный момент **+1**
2. Вам хочется где-то поесть **+2**
3. Вам хочется спать **-3**
4. Вам хочется увидеться с друзьями

## Пример: выходить из дома или нет

Признаки (1/0):

1. Вы свободны в данный момент **+1**
2. Вам хочется где-то поесть **+2**
3. Вам хочется спать **-3**
4. Вам хочется увидеться с друзьями **+4**

# Пример: выходить из дома или нет

Признаки (1/0):

1. Вы свободны в данный момент **+1**
2. Вам хочется где-то поесть **+2**
3. Вам хочется спать **-3**
4. Вам хочется увидеться с друзьями **+4**

**Порог для решающего правила: +1**

**Если сумма больше – выходим :)**

# Более серьезный пример: дать ли кредит

## Признаки (1/0):

- Работоспособный возраст
- Имеет счет в вашем банке
- Много просрочек по платежам за другие кредиты
- Нет просрочек по платежам за другие кредиты, причем другие кредиты есть

# Скоринговые карты

| ПОКАЗАТЕЛЬ                | ДИАПАЗОН ЗНАЧЕНИЙ   |
|---------------------------|---------------------|
| Возраст заемщика          | До 35 лет           |
|                           | От 35 до 45 лет     |
|                           | От 45 и старше      |
| Образование               | Высшее              |
|                           | Среднее специальное |
|                           | Среднее             |
| Состоит ли в браке        | Да                  |
|                           | Нет                 |
| Наличие кредита в прошлом | Да                  |
|                           | Нет                 |
| Стаж работы               | До 1 года           |
|                           | От 1 до 3 лет       |
|                           | От 3 до 6 лет       |
|                           | Свыше 6 лет         |
| Наличие автомобиля        | Да                  |
|                           | Нет                 |

# Скоринговые карты

| ПОКАЗАТЕЛЬ                | ДИАПАЗОН ЗНАЧЕНИЙ   | СКОРИНГ-БАЛЛ |
|---------------------------|---------------------|--------------|
| Возраст заемщика          | До 35 лет           | 7,60         |
|                           | От 35 до 45 лет     | 29,68        |
|                           | От 45 и старше      | 35,87        |
| Образование               | Высшее              | 29,82        |
|                           | Среднее специальное | 20,85        |
|                           | Среднее             | 22,71        |
| Состоит ли в браке        | Да                  | 29,46        |
|                           | Нет                 | 9,38         |
| Наличие кредита в прошлом | Да                  | 40,55        |
|                           | Нет                 | 13,91        |
| Стаж работы               | До 1 года           | 15,00        |
|                           | От 1 до 3 лет       | 18,14        |
|                           | От 3 до 6 лет       | 19,85        |
|                           | Свыше 6 лет         | 23,74        |
| Наличие автомобиля        | Да                  | 51,69        |
|                           | Нет                 | 15,93        |

# Подбор весов признаков и порога

**Почему нельзя продолжать также:**

- Сложно настраивать вручную
- Требуется эксперт в области
- Требуется проверка на данных и уточнение весов (эксперт может что-то не учесть)

# Подбор весов признаков и порога

**Почему нельзя продолжать также:**

- Сложно настраивать вручную
- Требуется эксперт в области
- Требуется проверка на данных и уточнение весов (эксперт может что-то не учесть)

**Решение** – автоматизируем подбор параметров: придумаем функцию от параметров, которую надо минимизировать, и используем методы численной оптимизации

## Формализуем линейный классификатор

$$a(x) = \begin{cases} 1, & \text{если } f(x) > 0 \\ -1, & \text{если } f(x) \leq 0 \end{cases}$$

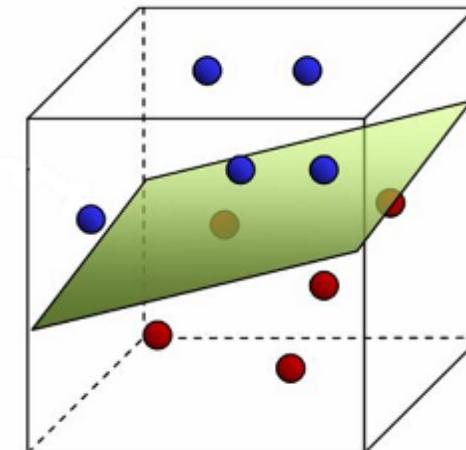
$$f(x) = w_0 + w_1x_1 + \cdots + w_nx_n$$

# Формализуем линейный классификатор

$$a(x) = \begin{cases} 1, & \text{если } f(x) > 0 \\ -1, & \text{если } f(x) \leq 0 \end{cases}$$

$$f(x) = w_0 + w_1x_1 + \cdots + w_nx_n = w_0 + \langle w, x \rangle$$

Геометрическая интерпретация:  
разделяем классы плоскостью

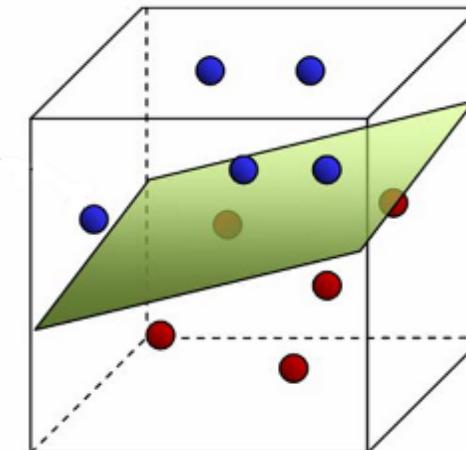


# Формализуем линейный классификатор

$$a(x) = \begin{cases} 1, & \text{если } f(x) > 0 \\ -1, & \text{если } f(x) \leq 0 \end{cases}$$

$$f(x) = \langle w, x \rangle$$

Геометрическая интерпретация:  
разделяем классы плоскостью



# Как выглядит код: применение модели

```
import numpy as np

def f(x):
    return np.dot(w, x) + w0

def a(x):
    return 1 if f(x) > 0 else 0
```

## Отступ (margin)

Отступом алгоритма  $a(x) = \text{sign}\{f(x)\}$  на объекте  $x_i$  называется величина  $M_i = y_i f(x_i)$   
( $y_i$  - класс, к которому относится  $x_i$ )

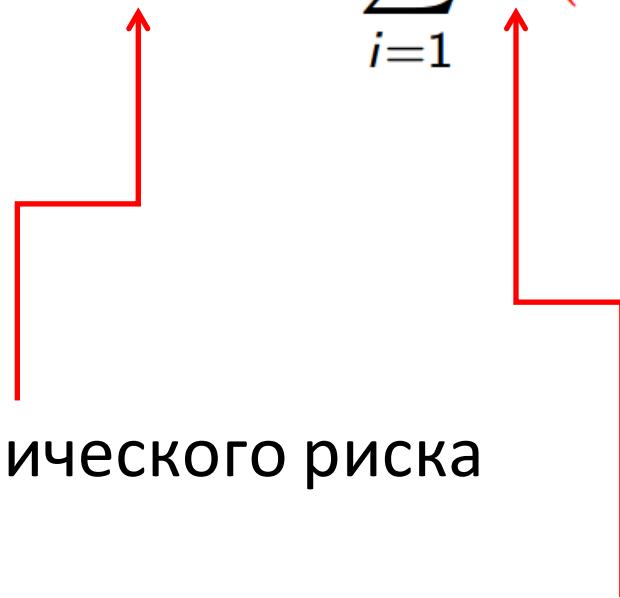
$$M_i \leq 0 \Leftrightarrow y_i \neq a(x_i)$$
$$M_i > 0 \Leftrightarrow y_i = a(x_i)$$

## Функция потерь

$$Q(w) = \sum_{i=1}^{\ell} [M_i(w) < 0]$$

## ФУНКЦИЯ ПОТЕРЬ

$$Q(w) = \sum_{i=1}^{\ell} [M_i(w) < 0] \leq \tilde{Q}(w) = \sum_{i=1}^{\ell} \mathcal{L}(M_i(w)) \rightarrow \min_w;$$

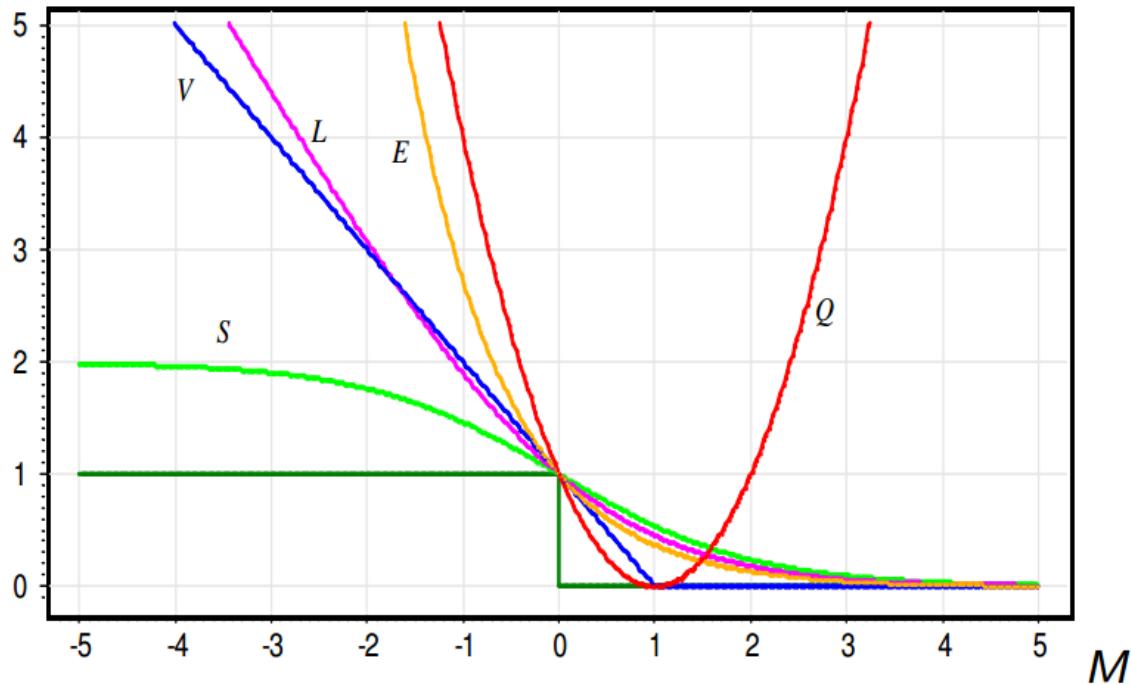


Функция эмпирического риска

Функция потерь

# ФУНКЦИЯ ПОТЕРЬ

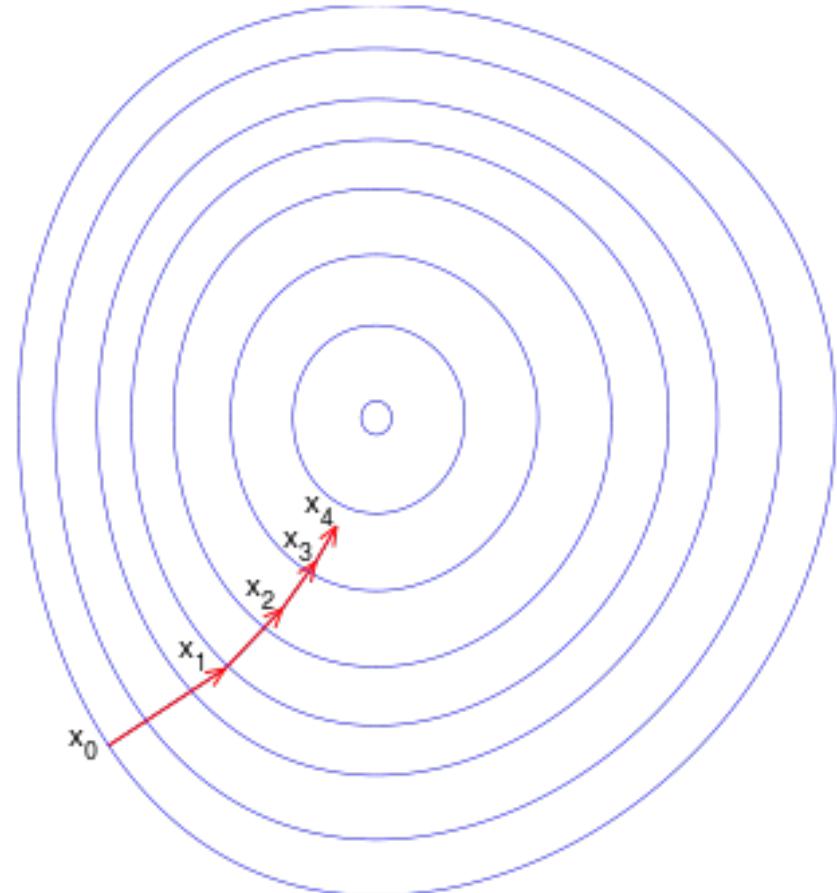
$$Q(w) = \sum_{i=1}^{\ell} [M_i(w) < 0] \leq \tilde{Q}(w) = \sum_{i=1}^{\ell} \mathcal{L}(M_i(w)) \rightarrow \min_w;$$



$$\begin{aligned}Q(M) &= (1 - M)^2 \\V(M) &= (1 - M)_+ \\S(M) &= 2(1 + e^M)^{-1} \\L(M) &= \log_2(1 + e^{-M}) \\E(M) &= e^{-M}\end{aligned}$$

# Градиентный спуск

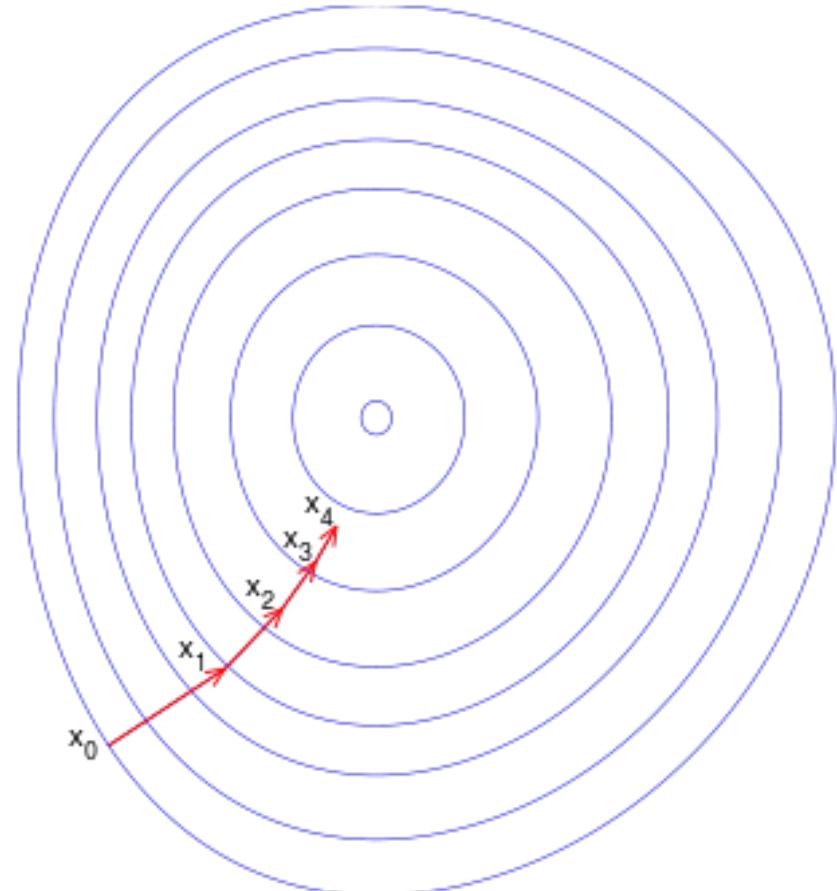
$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n), \quad n \geq 0.$$



# Градиентный спуск

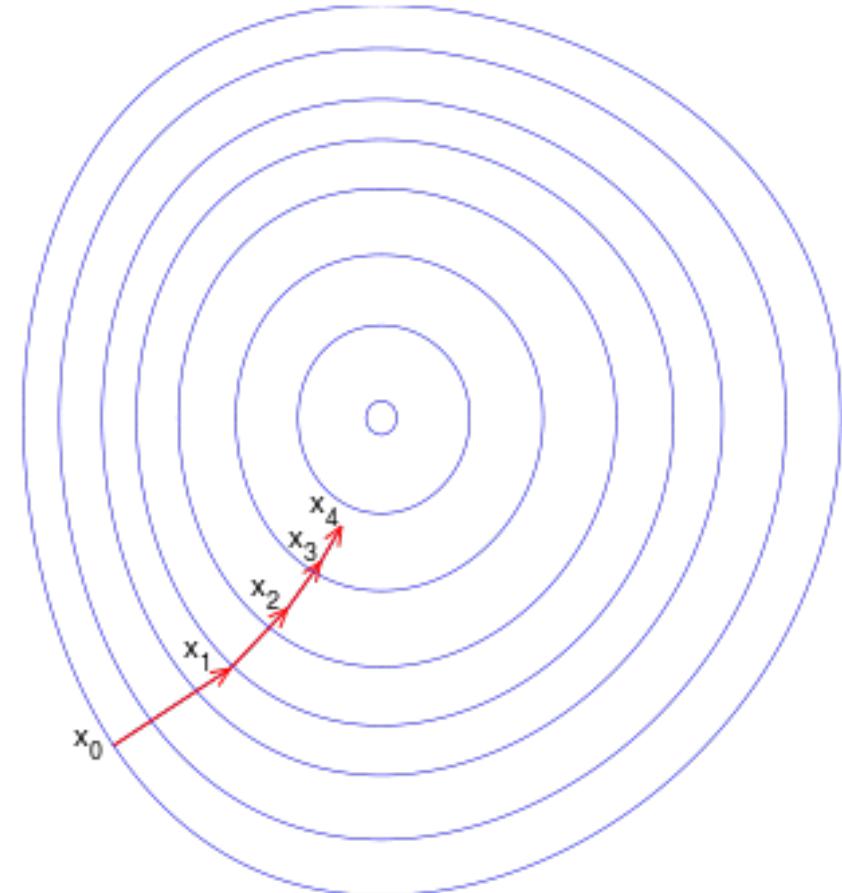
$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n), \quad n \geq 0.$$

$$\nabla_w \tilde{Q} = \sum_{i=1}^l \nabla L(M_i)$$



# Градиентный спуск

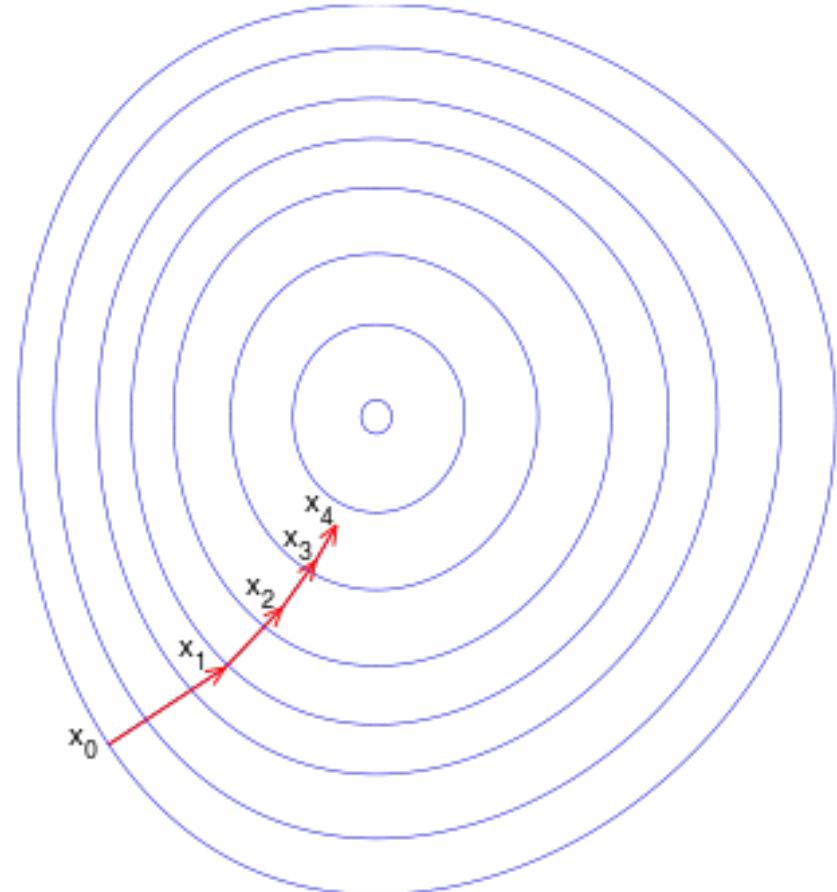
$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n), \quad n \geq 0.$$



$$\begin{aligned}\nabla_w \tilde{Q} &= \sum_{i=1}^l \nabla L(M_i) \\ \nabla \tilde{Q} &= \sum_{i=1}^l L'(M_i) \frac{\partial M_i}{\partial w}\end{aligned}$$

# Градиентный спуск

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n), \quad n \geq 0.$$



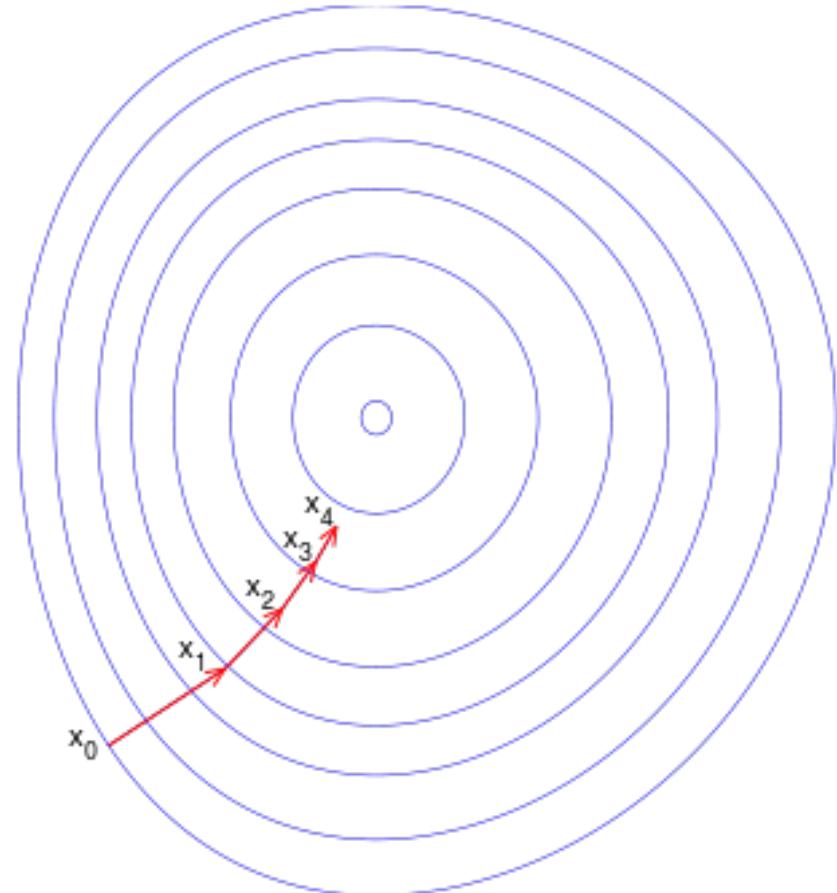
$$\nabla_w \tilde{Q} = \sum_{i=1}^l \nabla L(M_i)$$

$$\nabla \tilde{Q} = \sum_{i=1}^l L'(M_i) \frac{\partial M_i}{\partial w}$$

$$\frac{\partial M_i}{\partial w} = y_i x_i$$

# Градиентный спуск

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n), \quad n \geq 0.$$



$$\nabla_w \tilde{Q} = \sum_{i=1}^l \nabla L(M_i)$$

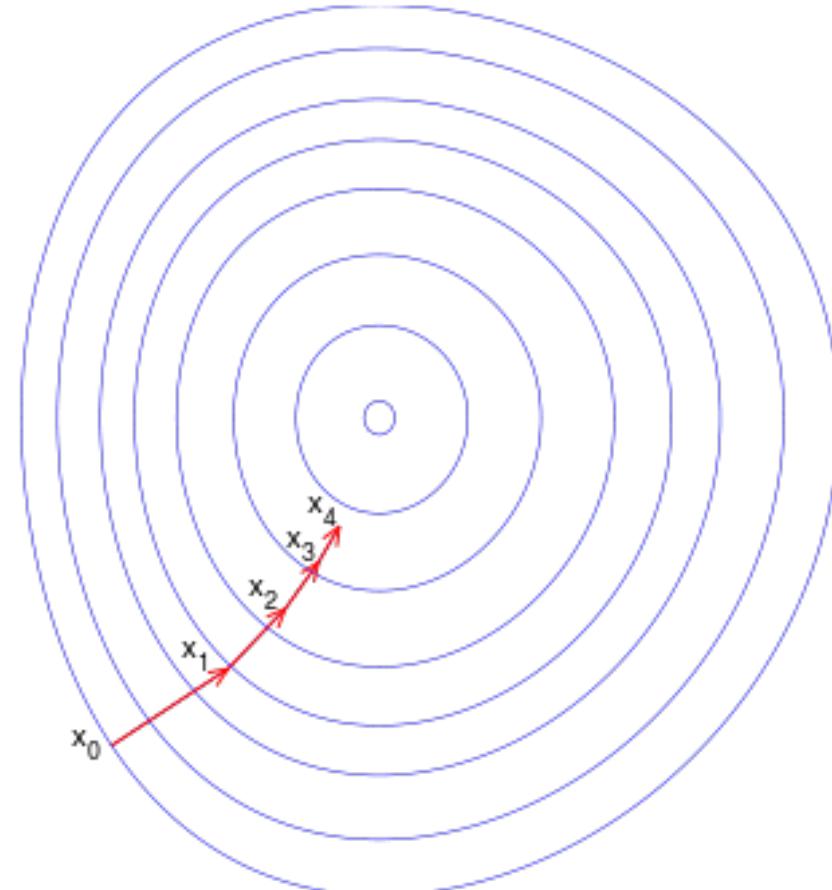
$$\nabla \tilde{Q} = \sum_{i=1}^l L'(M_i) \frac{\partial M_i}{\partial w}$$

$$\frac{\partial M_i}{\partial w} = y_i x_i$$

$$\nabla \tilde{Q} = \sum_{i=1}^l y_i x_i L'(M_i)$$

# Градиентный спуск

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n), \quad n \geq 0.$$



$$\nabla_w \tilde{Q} = \sum_{i=1}^l \nabla L(M_i)$$

$$\nabla \tilde{Q} = \sum_{i=1}^l L'(M_i) \frac{\partial M_i}{\partial w}$$

$$\frac{\partial M_i}{\partial w} = y_i x_i$$

$$\nabla \tilde{Q} = \sum_{i=1}^l y_i x_i L'(M_i)$$

$$w_{n+1} = w_n - \gamma_n \sum_{i=1}^l y_i x_i L'(M_i)$$

# Стохастический градиент

$$w_{n+1} = w_n - \gamma_n \sum_{i=1}^l y_i x_i L'(M_i)$$

$$w_{n+1} = w_n - \gamma_n y_i x_i L'(M_i)$$

$x_i$  – случайный элемент обучающей выборки

# Как выглядит код: обучение модели

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

# Как выглядит код: обучение модели

```
from random import randint           $L(M) = \max\{0, 1 - M\} = (1 - M)_+$ 

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

# Как выглядит код: обучение модели

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

# Как выглядит код: обучение модели

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

# Как выглядит код: обучение модели

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

# Как выглядит код: обучение модели

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

# Как выглядит код: обучение модели

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

# Как выглядит код: обучение модели

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

# Как выглядит код: обучение модели

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

# Как выглядит код: обучение модели

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0  $\gamma_n = \frac{1}{\alpha + n}$ 

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

# Как выглядит код: обучение модели

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

$$\gamma_n = \frac{1}{\alpha + n}$$

$$\gamma_n = \frac{1}{\sqrt{\alpha + n}}$$

# Как выглядит код: обучение модели

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

$$\gamma_n = \frac{1}{\alpha + n}$$

$$\gamma_n = \frac{1}{\sqrt{\alpha + n}}$$

$$\gamma_n = (\alpha + n)^{-\beta}$$

# Как выглядит код: обучение модели

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

$$\gamma_n = \frac{1}{\alpha + n}$$

$$\gamma_n = \frac{1}{\sqrt{\alpha + n}}$$

$$\gamma_n = (\alpha + n)^{-\beta}$$

$$\gamma_n = \tau \beta_n$$

# Как выглядит код: обучение модели

```
from random import randint

def loss(x, answer):
    return max([0, 1 - answer * f(x)])

def der_loss(x, answer):
    return -1.0 if 1 - answer * f(x) > 0 else 0.0

def fit(X_train, y_train):
    for k in range(10000):
        rand_index = randint(0, len(X_train))
        x = X_train[rand_index]
        y = y_train[rand_index]

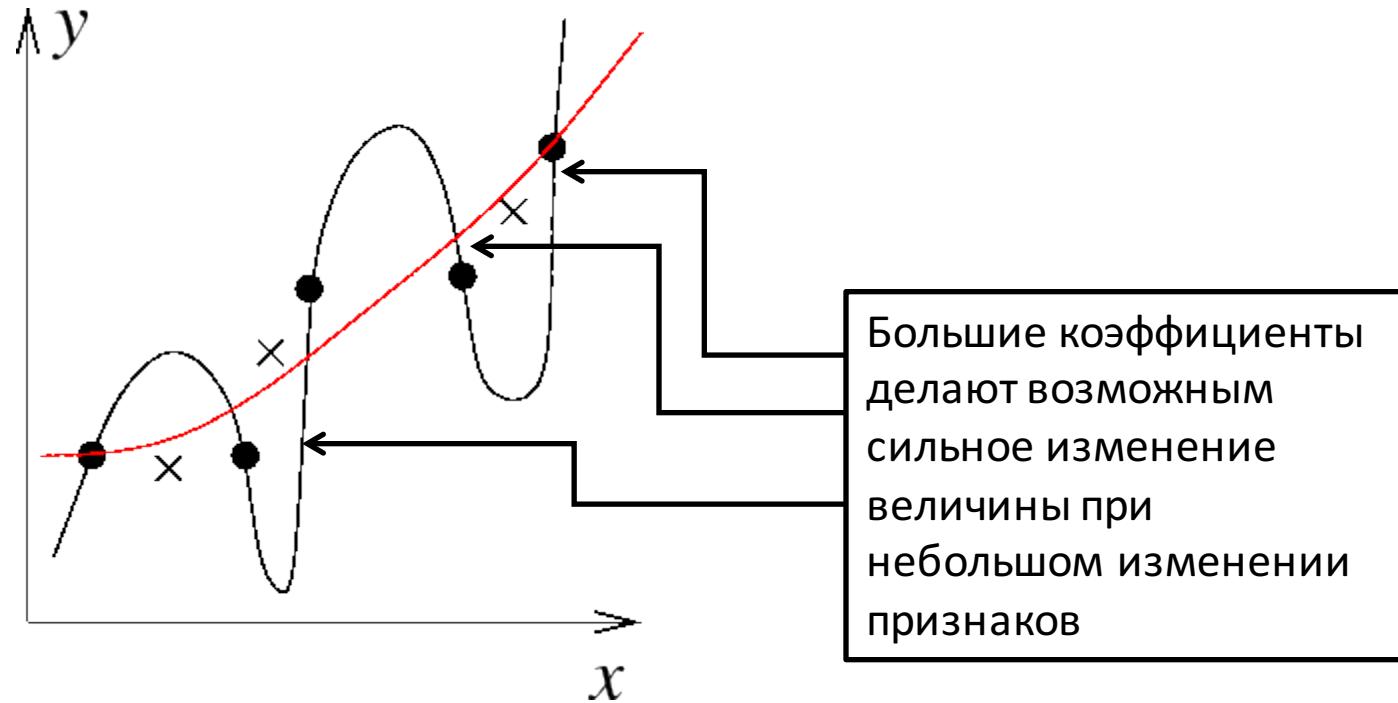
        step = 0.01

        w -= x * step * y * der_loss(x, y)
```

$$w_{n+1} = w_n - \gamma_n y_i x_i L'(M_i)$$

# Регуляризация

- Переобучение в задаче обучения с учителем как правило означает большие коэффициенты:



- Идея: добавить ограничение на коэффициенты

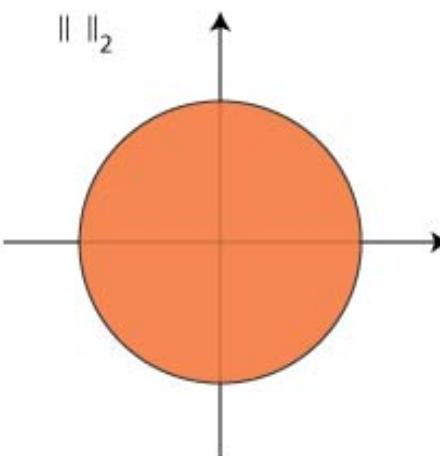
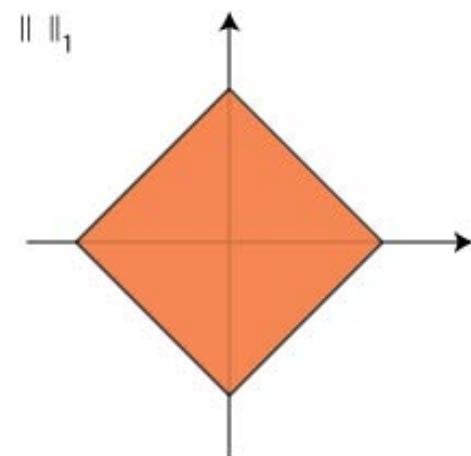
# Регуляризация

$$\left\{ \begin{array}{l} \tilde{Q} = \sum_{i=1}^l L(M_i) \rightarrow \min \\ \sum_{k=1}^m |w_k| \leq \tau \end{array} \right.$$

*l1 – регуляризация*

$$\left\{ \begin{array}{l} \tilde{Q} = \sum_{i=1}^l L(M_i) \rightarrow \min \\ \sum_{k=1}^m |w_k|^2 \leq \tau \end{array} \right.$$

*l2 – регуляризация*

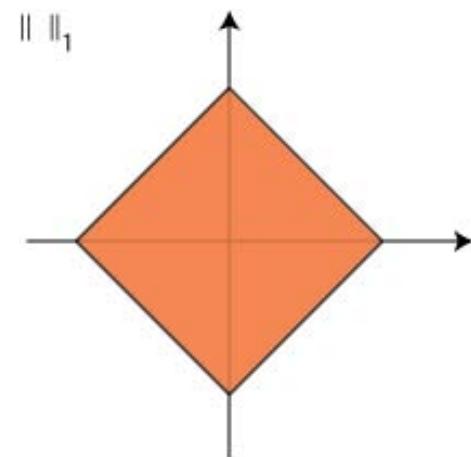


# Регуляризация

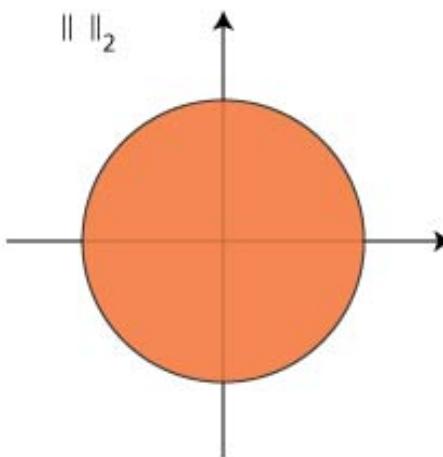
$$\sum_{i=1}^l L(M_i) + \gamma \sum_{k=1}^m |w_k| \rightarrow \min$$

$$\sum_{i=1}^l L(M_i) + \gamma \sum_{k=1}^m w_k^2 \rightarrow \min$$

*l1 – регуляризация*



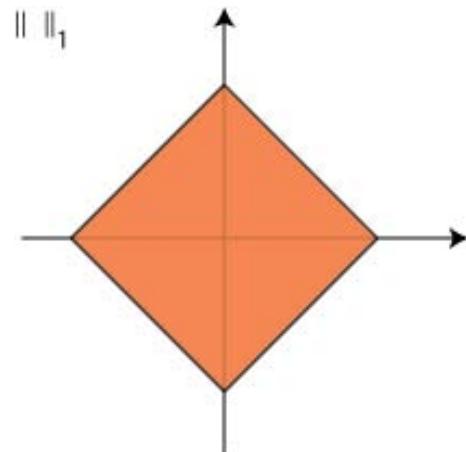
*l2 – регуляризация*



# Регуляризация

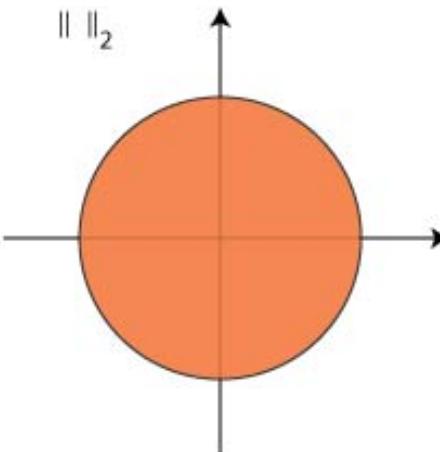
$$\sum_{i=1}^l L(M_i) + \gamma \sum_{k=1}^m |w_k| \rightarrow \min$$

*l1 – регуляризация*



$$\sum_{i=1}^l L(M_i) + \gamma \sum_{k=1}^m {w_k}^2 \rightarrow \min$$

*l2 – регуляризация*



**Вопрос:**  
вы заметили, что  
в регуляризатор  
не включается  
вес  $w_o$ ?

## Различия между $\ell_1$ и $\ell_2$

- **Разреженность** –  $\ell_1$ -регуляризация делает вектор весов более разреженным (содержащим больше нулей)
- В случае линейной классификации это означает **отбор признаков**: признаки с нулевыми весами не используются в классификации

## Упражнение

Выпишете, как поменяется правило обновления весов признаков в линейном классификаторе с помощью SGD при добавлении регуляризатора

# Стандартные линейные классификаторы

| Классификатор  | Функция потерь                        | Регуляризатор  |
|--|---------------------------------------|--|
| SVM (Support vector machine, метод опорных векторов) | $L(M) = \max\{0, 1 - M\} = (1 - M)_+$ | $\sum_{k=1}^m w_k^2$                                       |
| Логистическая регрессия                              | $L(M) = \log(1 + e^{-M})$             | Обычно<br>$\sum_{k=1}^m w_k^2$ или<br>$\sum_{k=1}^m  w_k $ |

# Обязательно ли функция потерь – функция от отступа?

Пример:

$$y_i \in \{0, 1\} \quad Q = - \sum_{i=1}^{\ell} y_i \ln p_i + (1 - y_i) \ln(1 - p_i) \rightarrow \min_w$$

$$p_i = \sigma(\langle w, x_i \rangle) = \frac{1}{1 + e^{-\langle w, x_i \rangle}}$$

# Обязательно ли функция потерь – функция от отступа?

Пример:

$$y_i \in \{0, 1\} \quad Q = - \sum_{i=1}^{\ell} y_i \ln p_i + (1 - y_i) \ln(1 - p_i) \rightarrow \min_w$$

$$p_i = \sigma(\langle w, x_i \rangle) = \frac{1}{1 + e^{-\langle w, x_i \rangle}}$$

Упражнение:

Показать, что это та же оптимизационная задача, что и в логистической регрессии

## Общий случай

$$Q = \sum_{i=1}^{\ell} L(y_i, f(x_i)) + \gamma V(w) \rightarrow \min_w$$

Функция потерь

Коэффициент  
регуляризации

Регуляризатор

## Общий случай

$$Q = \sum_{i=1}^{\ell} L(y_i, f(x_i)) + \gamma V(w) \rightarrow \min_w$$

Функция потерь

Регуляризатор

Коэффициент  
регуляризации

Упражнение:

Как будет меняться качество на обучающей и на тестовой выборке с ростом коэффициента регуляризации в SVM и в логистической регрессии в sklearn? Выясните, почему результат такой.

# Линейная регрессия

$$a(x) = \langle w, x \rangle + w_0$$

# Линейная регрессия

$$a(x) = \langle w, x \rangle + w_0$$

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

# Линейная регрессия

$$a(x) = \langle w, x \rangle + w_0$$

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

$$L(y_i, a(x_i)) = (y_i - a(x_i))^2$$

# Линейная регрессия

$$a(x) = \langle w, x \rangle + w_0$$

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

$$L(y_i, a(x_i)) = (y_i - a(x_i))^2$$

$$L(y_i, a(x_i)) = |y_i - a(x_i)|$$

## Библиотеки

- libSVM
- liblinear
- sklearn.linear\_models
- Vowpal Wabbit

# Итог

1. Идея линейной классификации
2. Функции потерь
3. Градиентный спуск и стохастический градиент
4. Регуляризация
5. Стандартные линейные классификаторы
6. Линейная регрессия
7. Библиотеки

## Упражнение

В реализации линейного классификатора на Python, приведенной в этих слайдах, лектор коварно допустил ошибки.

Попробуйте их найти и исправить до следующей лекции :)

# Бонус-трек: Классификация музыки по жанрам



# Проблема

Захотели послушать определенный жанр, открываем плейлист или папку с музыкой, а там...

|  |      |
|--|------|
| <b>Imagine Dragons</b> – Selene                                  | 4:04 |
| <b>Imagine Dragons</b> – Radioactive                             | 3:08 |
| <b>Imagine Dragons</b> – Paradise                                | 4:38 |
| <b>Imagine Dragons</b> – Demons                                  | 3:35 |
| <b>Metallica</b> – The Day That Never Comes                      | 7:56 |
| <b>ESC 2014 - Austria</b> – Conchita Wurst - Rise Like A Phoenix | 3:05 |
| <b>Bon Jovi</b> – Someday I'll Be Saturday Night                 | 4:39 |
| <b>Бумбокс</b> – Летний Дождь                                    | 3:19 |
| <b>Бумбокс</b> – Сандали   | 4:42 |
| <b>Бумбокс</b> – Полина  | 4:21 |
| <b>Бумбокс</b> – Пошла вон                                       | 3:25 |
| <b>Бумбокс</b> – Скажи как мне жить                              | 3:24 |
| <b>Marty Robbins</b> – Big iron                                  | 3:56 |
| <b>Jack Shaindin</b> – Let's go sunning                          | 1:41 |
| <b>Frank Sinatra</b> – Blue Moon                                 | 2:51 |
| <b>The Ink Spots</b> – I Don't Want To Set The World On Fire     | 3:04 |
| <b>Peggy Lee</b> – Why Don't You Do Right                        | 2:27 |
| <b>The Ink Spots</b> – Maybe                                     | 2:50 |
| <b>John Lee Hooker</b> – Boom Boom Boom                          | 2:44 |

| <input type="checkbox"/> | Имя                                       | Дата изменения   | Тип        | Размер    |
|--------------------------|---|------------------|------------|-----------|
|                          | The Piano Guys – Happy end (Piano C...    | 05.07.2013 12:05 | Файл "MP3" | 7 509 КБ  |
|                          | The Rolling Stones – Rain Fall Down       | 30.06.2013 11:49 | Файл "MP3" | 7 362 КБ  |
|                          | The White Stripes – Ball and Biscuit      | 22.06.2013 18:59 | Файл "MP3" | 10 247 КБ |
|                          | This Will Destroy You – A Three-Legge...  | 21.12.2013 17:56 | Файл "MP3" | 17 228 КБ |
|                          | This Will Destroy You – Quiet             | 21.12.2013 17:56 | Файл "MP3" | 7 604 КБ  |
|                          | Tiarah – Diane (Therapy! cover)           | 25.04.2014 13:02 | Файл "MP3" | 9 195 КБ  |
|                          | Toe – Goodbye                             | 24.06.2013 10:31 | Файл "MP3" | 9 978 КБ  |
|                          | Toe – Kodoku no Hatsumei                  | 24.06.2013 10:31 | Файл "MP3" | 5 556 КБ  |
|                          | Toe – Long Tomorrow                       | 24.06.2013 10:31 | Файл "MP3" | 14 141 КБ |
|                          | toe – Ordinary Days                       | 24.06.2013 10:31 | Файл "MP3" | 7 100 КБ  |
|                          | Toe – Path                                | 24.06.2013 10:31 | Файл "MP3" | 5 778 КБ  |
|                          | Toe – The Future Is Now                   | 24.06.2013 10:31 | Файл "MP3" | 7 725 КБ  |
|                          | toe – You Go                              | 24.06.2013 10:32 | Файл "MP3" | 6 221 КБ  |
|                          | Tony Hawk's Pro Skater 2 – Track 02       | 25.04.2014 13:08 | Файл "MP3" | 4 382 КБ  |
|                          | Tony Hawk's Pro Skater 2 – Track 10       | 25.04.2014 13:09 | Файл "MP3" | 7 049 КБ  |
|                          | Tony Hawk's Pro Skater 4 OST – 3          | 25.04.2014 13:07 | Файл "MP3" | 4 518 КБ  |
|                          | Trent Reznor and Atticus Ross – In Mo...  | 22.06.2013 18:59 | Файл "MP3" | 11 599 КБ |
|                          | Trent Reznor and Atticus Ross – Intrig... | 22.06.2013 18:59 | Файл "MP3" | 10 334 КБ |
|                          | Uma2rman – А знаешь, все еще буде...      | 11.03.2014 2:44  | Файл "MP3" | 3 204 КБ  |
|                          | Vorony_stereo                             | 16.03.2014 22:31 | Файл "MP3" | 3 679 КБ  |
|                          | Woodkid – I Love You                      | 07.11.2013 7:49  | Файл "MP3" | 9 064 КБ  |
|                          | Woodkid – Iron                            | 26.06.2013 15:16 | Файл "MP3" | 6 139 КБ  |
|                          | Woodkid – Run Boy Run                     | 26.06.2013 15:16 | Файл "MP3" | 8 384 КБ  |
|                          | Woodkid – The Great Escape                | 26.06.2013 15:16 | Файл "MP3" | 7 730 КБ  |
|                          | Woodkid – Wasteland                       | 26.06.2013 15:17 | Файл "MP3" | 5 627 КБ  |
|                          | Zero 7 – In the Waiting Line (OST Hou...  | 19.11.2013 13:41 | Файл "MP3" | 7 382 КБ  |
|                          | Алина Орлова – Menulis (1)                | 09.08.2013 22:08 | Файл "MP3" | 3 793 КБ  |
|                          | Алина Орлова – Menulis                    | 01.08.2013 15:47 | Файл "MP3" | 3 793 КБ  |
|                          | Би-2 – Падает снег (feat. Ю. Чичерин...)  | 11.03.2014 2:46  | Файл "MP3" | 5 712 КБ  |

...одна большая свалка :)

# Несколько светлых идей

- Навести порядок
- Не делать это вручную для 1000+ аудиозаписей
- Определение жанра – в чистом виде задача классификации
- Остается придумать признаки
- По запросам в гугле про классификацию музыки узнаем, что область имеет название Music Information Retrieval
- Находим много интересных статей :)

# Сначала найдем датасет

The screenshot shows a web browser displaying the Marsyas website at [marsyas.info/download/data\\_sets/](http://marsyas.info/download/data_sets/). The page title is "Data Sets". On the left, there is a vertical navigation menu with sections: "About" (Overview, Projects, Videos, Pictures, Webdemos, Publications, Licence), "Download" (github, git Repository, Installation, Binaries, Data Sets, Dashboard, Vamp Plugins), and "Documentation" (Documentation Links, User manual, Marsyas Cookbook, Marsyas Blog, Library Reference). The main content area features a section titled "GTZAN Genre Collection" with a detailed description of the dataset, mentioning its use in a 2002 paper and its collection from 2000-2001. It also describes the dataset's composition of 1000 tracks, 10 genres, and 100 tracks per genre. A "Download Now" button with a downward arrow icon is present. Below this, there is a section titled "Music Speech" with a similar dataset description.

[http://marsyas.info/download/data\\_sets/](http://marsyas.info/download/data_sets/)

## Что в датасете

- Десять жанров
- По 100 wav-файлов для каждого жанра
- В каждом файле первые 30 секунд аудиозаписи

# Как посмотреть на датасет

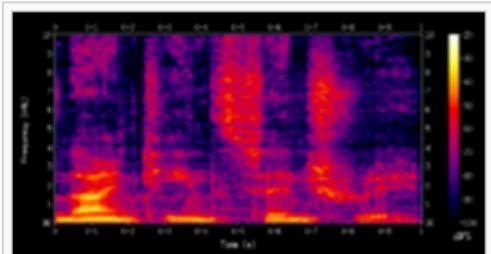
Гуглим, как люди визуализируют звук, и сразу узнаем про спектрограммы.  
Вот, что пишет про это Вики:

## Спектрограмма

Материал из Википедии — свободной энциклопедии

[править | править исходный текст]

Спектрограмма (сонограмма) — изображение, показывающее зависимость спектральной плотности мощности сигнала от времени. Спектрограммы применяются для идентификации речи, анализа звуков животных, в различных областях музыки, радио- и гидролокации, обработке речи, сейсмологии и в других областях.



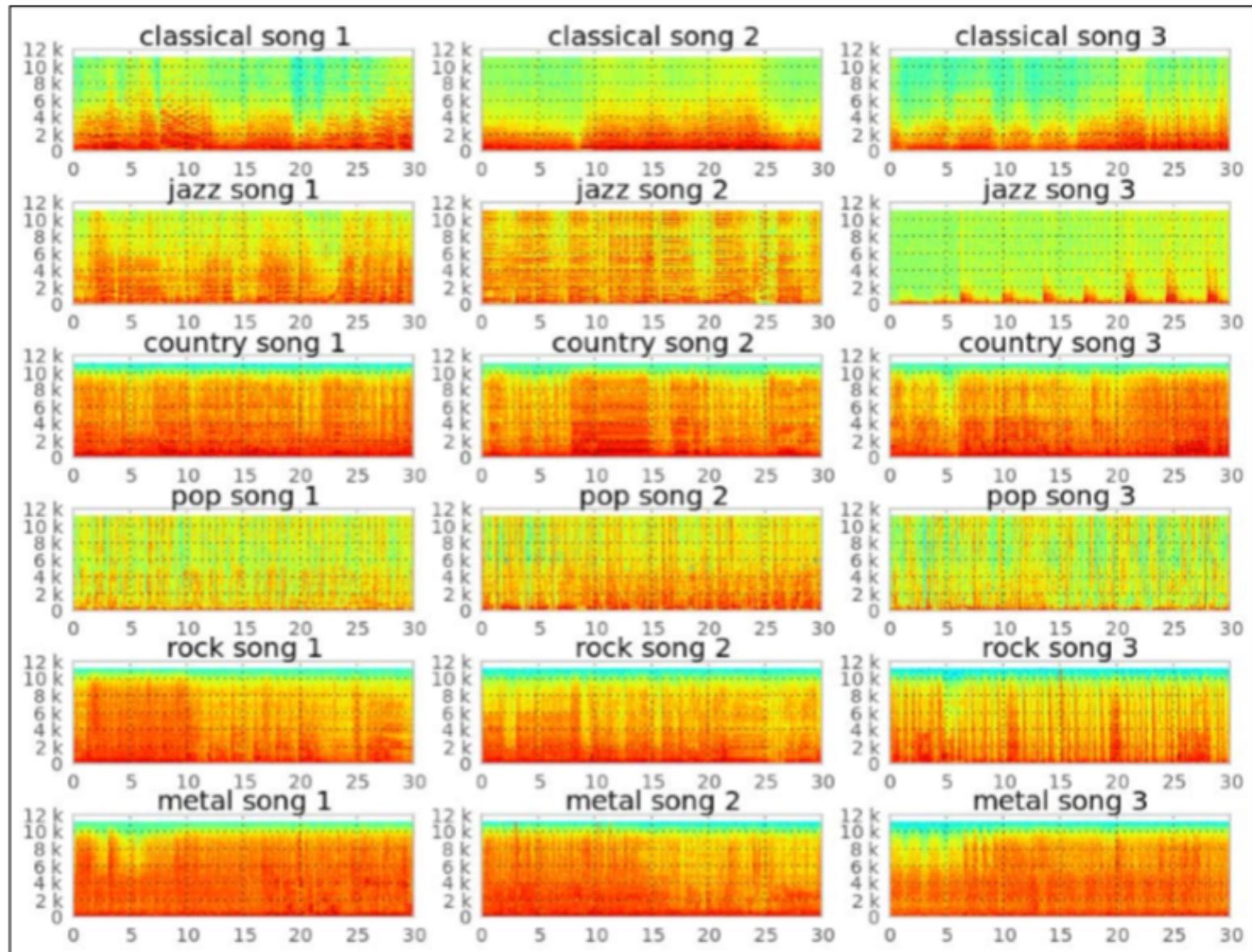
Спектрограмма мужского голоса

### Представление [править | править исходный текст]

Наиболее распространенным представлением спектрограммы является двумерная диаграмма: на горизонтальной оси представлено время, по вертикальной оси — частота; третье измерение с указанием амплитуды на определенной частоте в конкретный момент времени представлено интенсивностью или цветом каждой точки изображения.



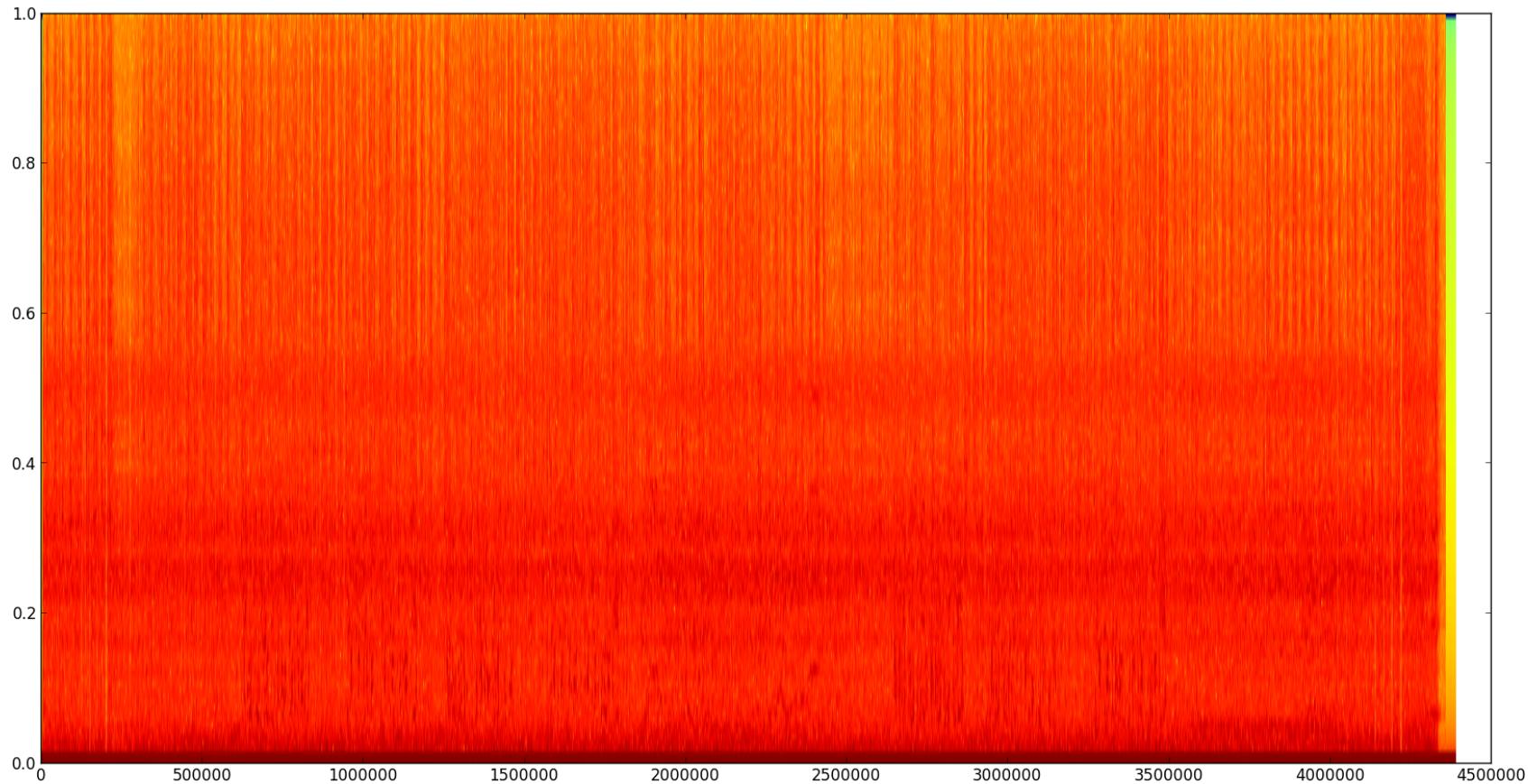
# Спектрограммы для разных жанров



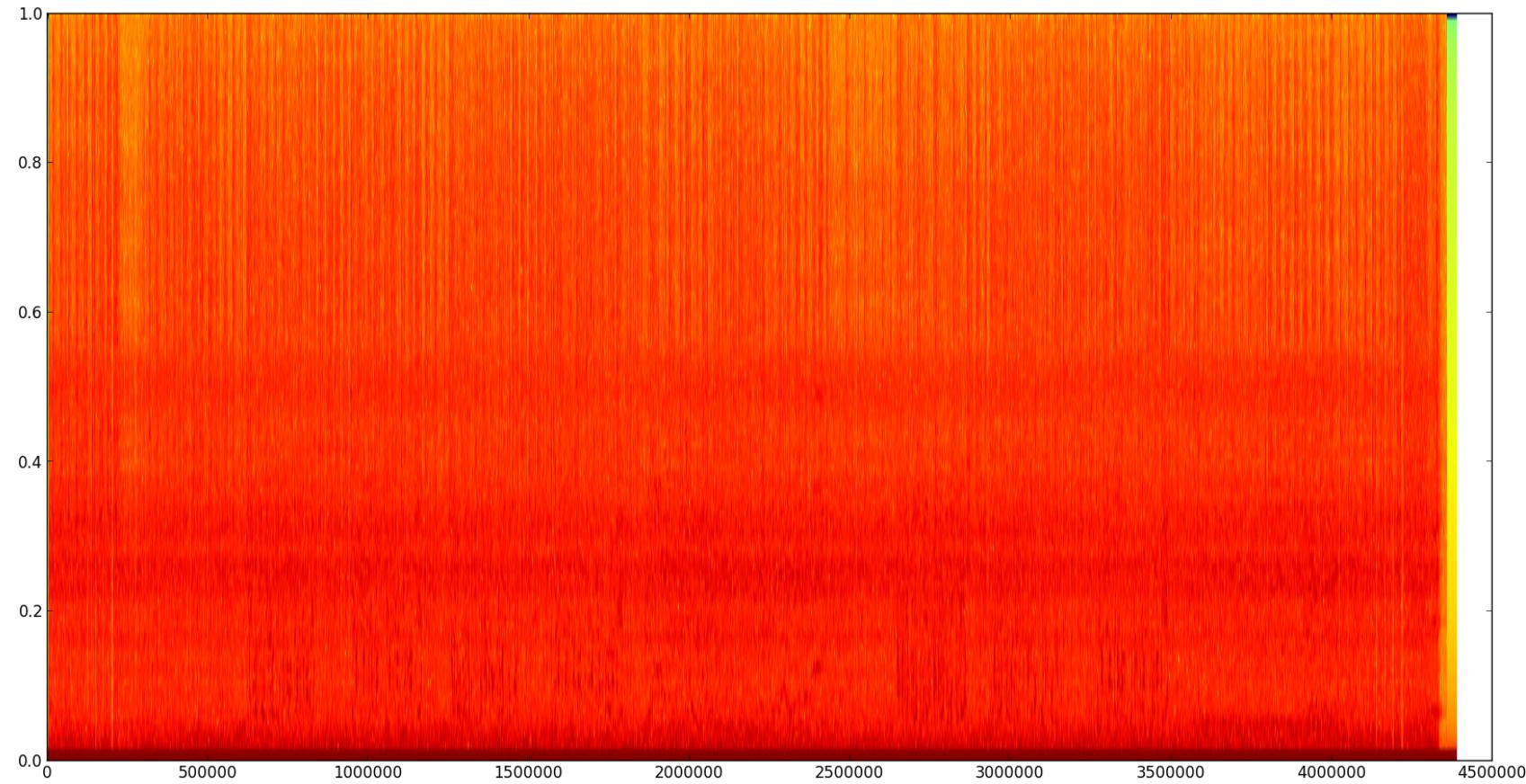
# Как построить?

```
>>> import scipy
>>> from matplotlib.pyplot import specgram
>>> sample_rate, X = scipy.io.wavfile.read(wave_filename)
>>> print sample_rate, X.shape
22050, (661794,)
>>> specgram(X, Fs=sample_rate, xextent=(0,30))
```

Какой жанр?



Какой жанр?

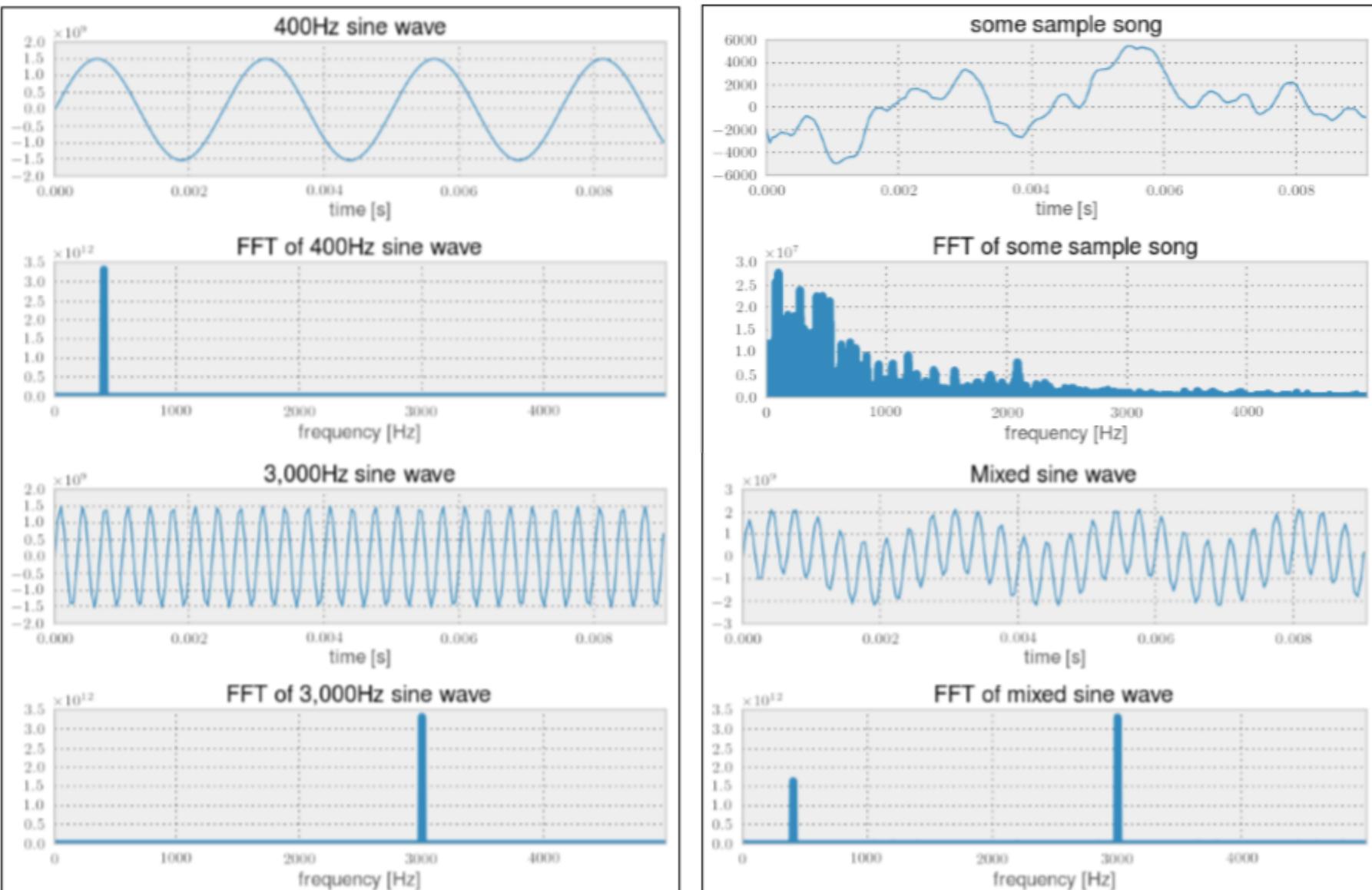


Метал (Metallica – St.Anger)

## Как бы построить один набор чисел?

- В обработке сигналов часто используют разложение в ряд Фурье
- Несколько основных гармоник уже неплохо описывают сигнал
- Почему бы и здесь не сделать так?

# Вспоминаем второй курс (если он уже позади :)



# Как сделать?

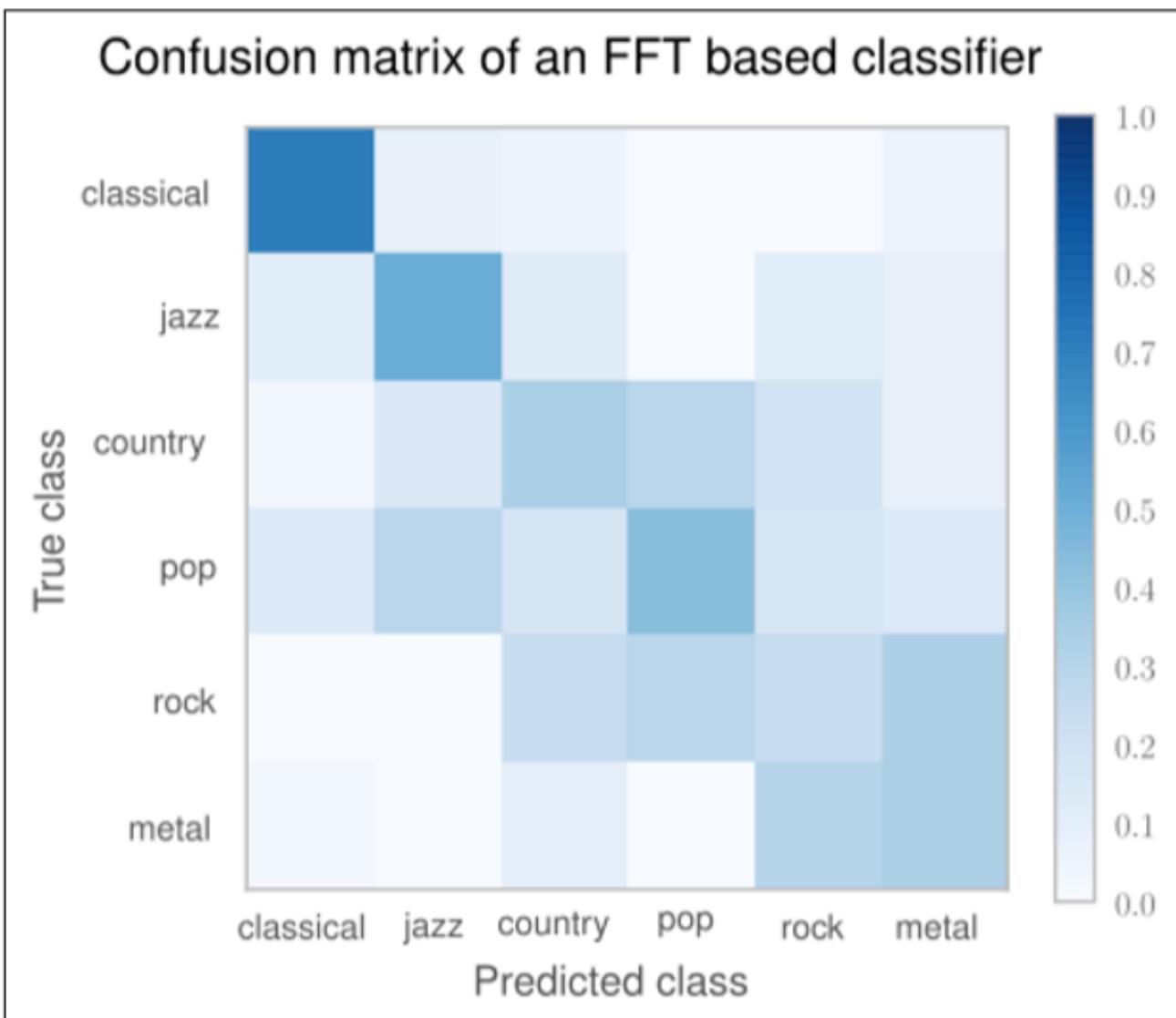
```
def create_fft(fn):
    sample_rate, X = scipy.io.wavfile.read(fn)
    fft_features = abs(scipy.fft(X) [:1000])
    base_fn, ext = os.path.splitext(fn)
    data_fn = base_fn + ".fft"
    np.save(data_fn, fft_features)

def read_fft(genre_list, base_dir=GENRE_DIR):
    X = []
    y = []
    for label, genre in enumerate(genre_list):
        genre_dir = os.path.join(base_dir, genre, "*.fft.npy")
        file_list = glob.glob(genre_dir)
        for fn in file_list:
            fft_features = np.load(fn)

            X.append(fft_features [:1000])
            y.append(label)

    return np.array(X), np.array(y)
```

# Результат 1



## Повышаем качество

- Круто? Не очень. Читаем статьи.
- Узнаем, например, про Mel Frequency Cepstral Coefficients (MFCC) – преобразование Фурье логарифма спектра
- Используем в качестве признаков

Как обычно, есть готовый пакет

```
>>>from scikits.talkbox.features import mfcc  
>>>sample_rate, X = scipy.io.wavfile.read(fn)  
>>>ceps, mspec, spec = mfcc(X)  
>>> print(ceps.shape)  
(4135, 13)
```

**Hint:**

```
x = np.mean(ceps[int(num_ceps*1/10):int(num_ceps*9/10)], axis=0)
```

# Итоговый код построения фич

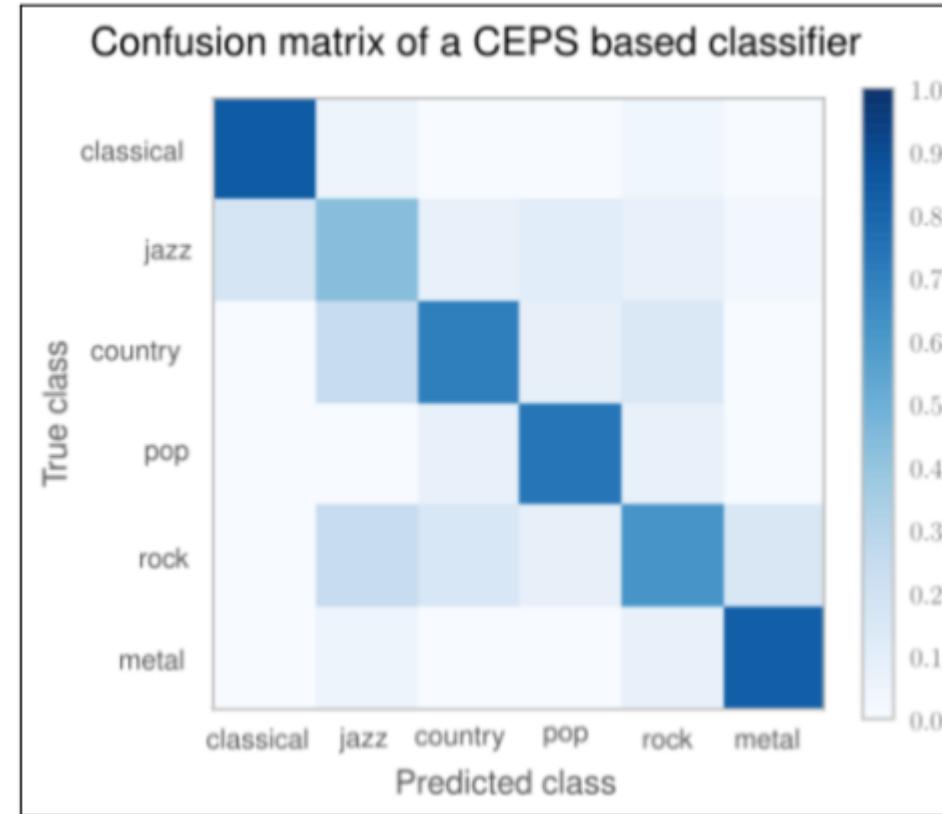
```
def write_ceps(ceps, fn):
    base_fn, ext = os.path.splitext(fn)
    data_fn = base_fn + ".ceps"
    np.save(data_fn, ceps)
    print("Written %s" % data_fn)

def create_ceps(fn):
    sample_rate, X = scipy.io.wavfile.read(fn)
    ceps, mspec, spec = mfcc(X)
    write_ceps(ceps, fn)

def read_ceps(genre_list, base_dir=GENRE_DIR):
    X, Y = [], []
    for label, genre in enumerate(genre_list):
        for fn in glob.glob(os.path.join(
            base_dir, genre, "*.ceps.npy")):
            ceps = np.load(fn)
            num_ceps = len(ceps)
            X.append(np.mean(
                ceps[int(num_ceps*1/10):int(num_ceps*9/10)], axis=0))
            Y.append(label)

    return np.array(X), np.array(Y)
```

# Результат 2



# Как можно погрузиться в тему:

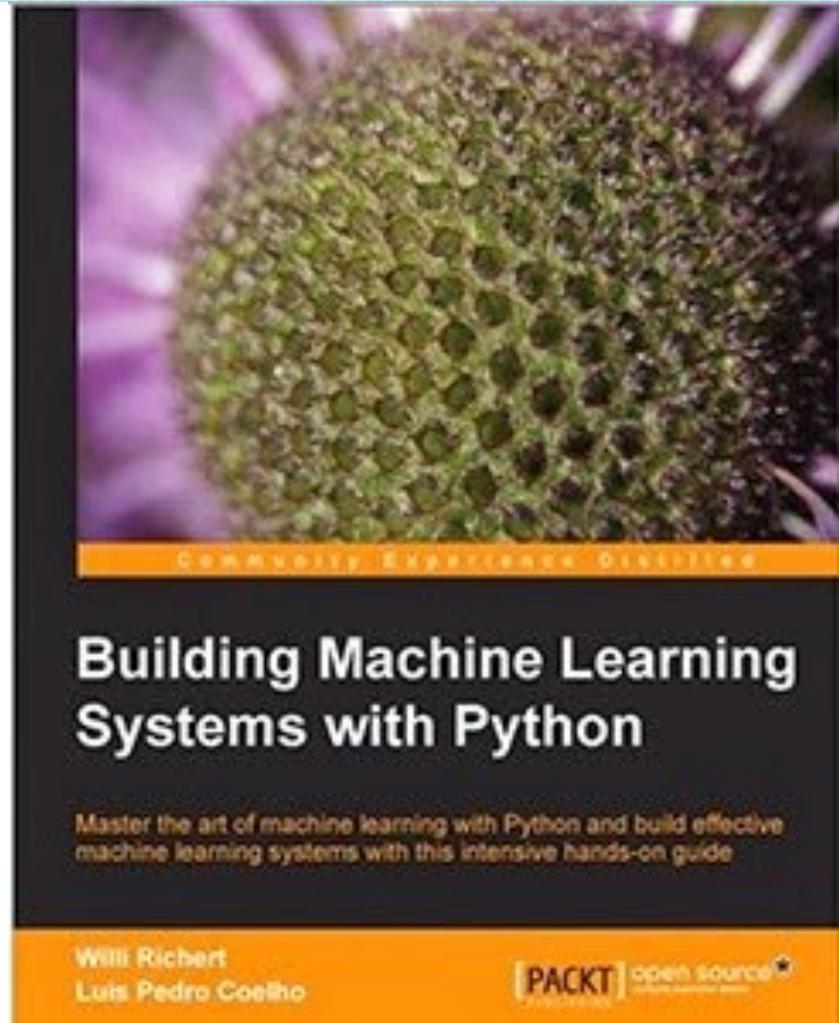
1. Воспроизвести все описанные действия и построить классификатор музыки по жанрам
2. Разобрать статью с описанием других признаков, реализовать их и сравнить результаты на кросс-валидации
3. \* Реализовать скачивание плейлиста из vk (или из другого источника), конвертацию mp3 в wav и классификацию по жанрам. При необходимости, сделать свою обучающую выборку.
4. \*\* Почитать статьи и посмотреть, что исследуют в области Music Information Retrieval

## Что еще можно придумать?

- Можно сделать подбор плейлиста из музыки одного жанра или из разных
- Можно сделать поиск похожих мелодий и рекомендации песен по звучанию, советуя те песни, которые ближе в пространстве признаков
- Можно сделать автоматическое составление плейлиста с плавным переходом от одного «класса» музыки к другому (например, чтобы постепенно исправить плохое настроение)

# Если захочется повторить

- Конвертация из mp3 в wav из консоли:  
ffmpeg, sox
- Выкачивание музыки из vk на Python:  
<http://habrahabr.ru/post/157925/>
- Интересная книжка по машинному обучению на Python – Building Machine Learning Systems with Python



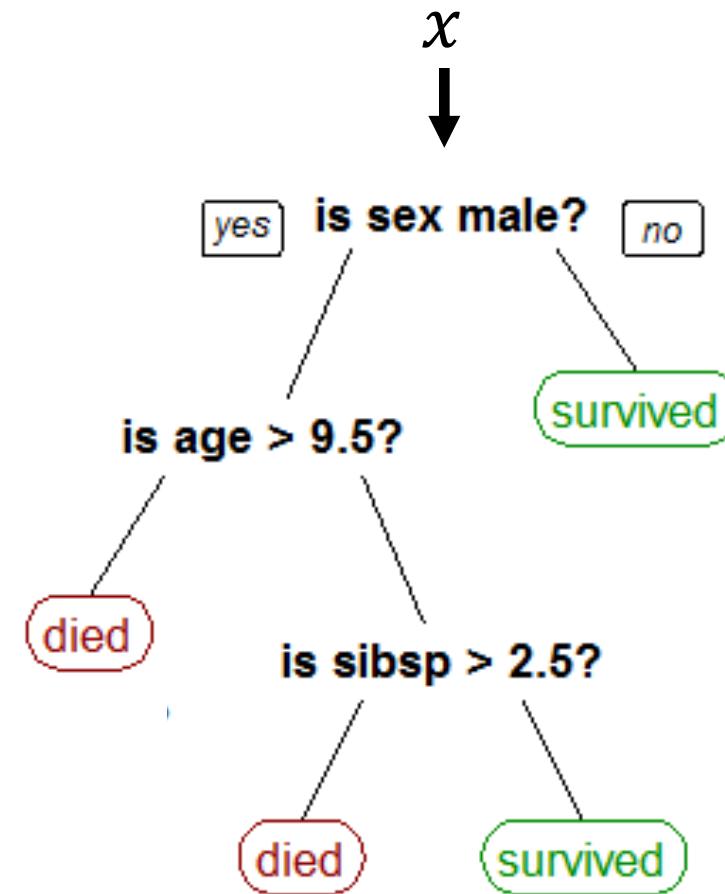
## II. Решающие деревья

# План

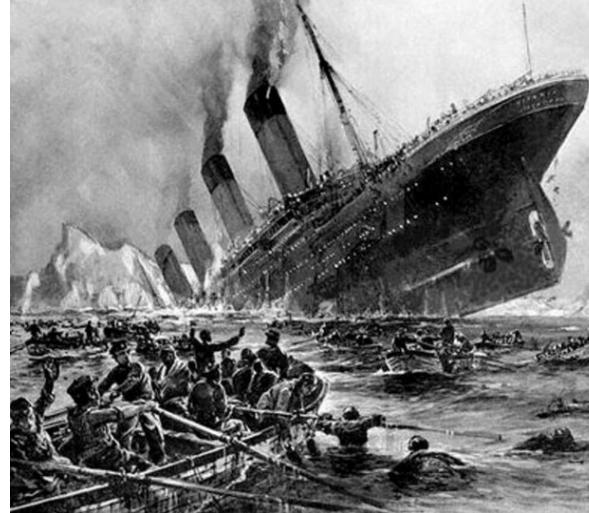
1. Что такое решающие деревья
2. Решающие деревья в классификации и регрессии
3. Как строить решающие деревья
4. Дополнительные темы

1. Что такое решающие деревья

# Решающее дерево



# Датасет

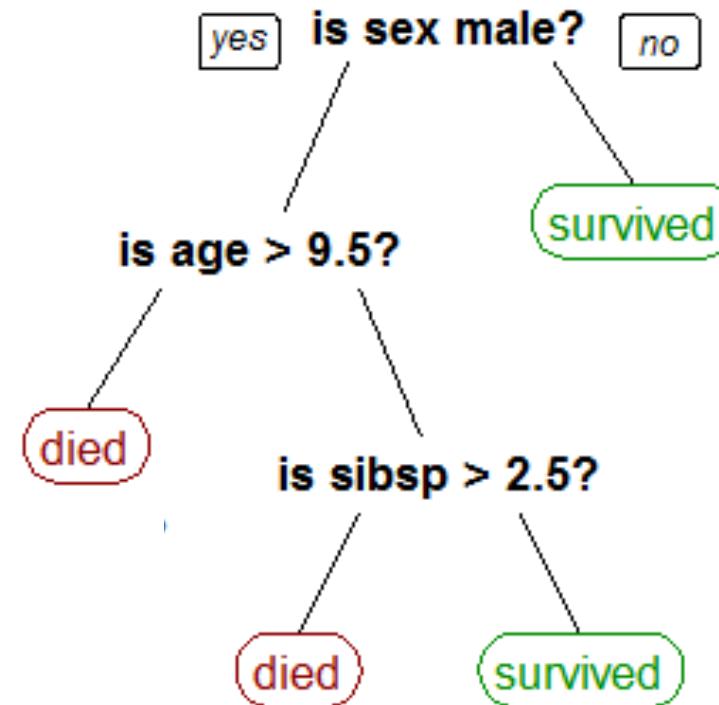


**«Titanic Dataset»** - список пассажиров Титаника, для которых даны возраст, пол, количество членов семьи на борту и другие признаки.

**Целевые значения:** выжил пассажир или нет  
(задача классификации)

# Решающее дерево

$$x = (9, \text{ male}, 3)$$

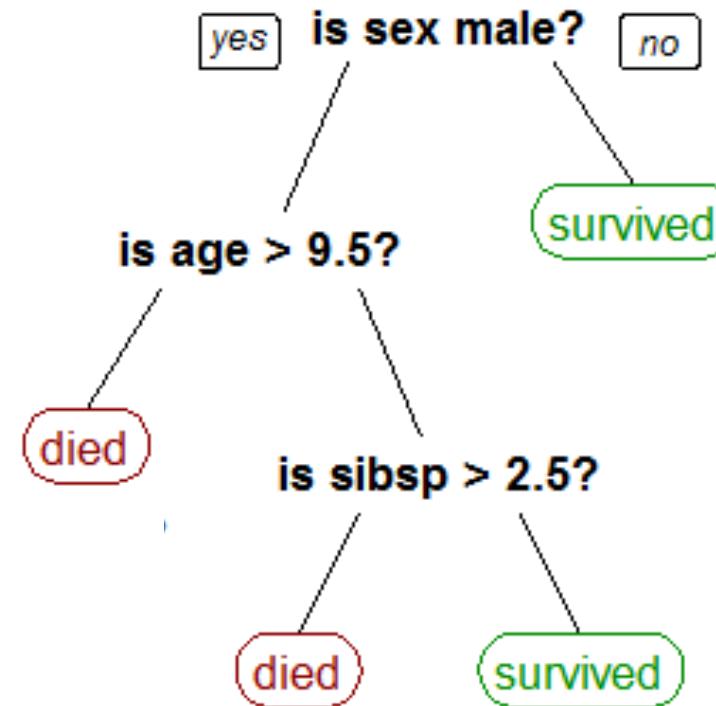


# Решающее дерево

age    sex    sibsp

$$x = (9, \text{ male}, 3)$$

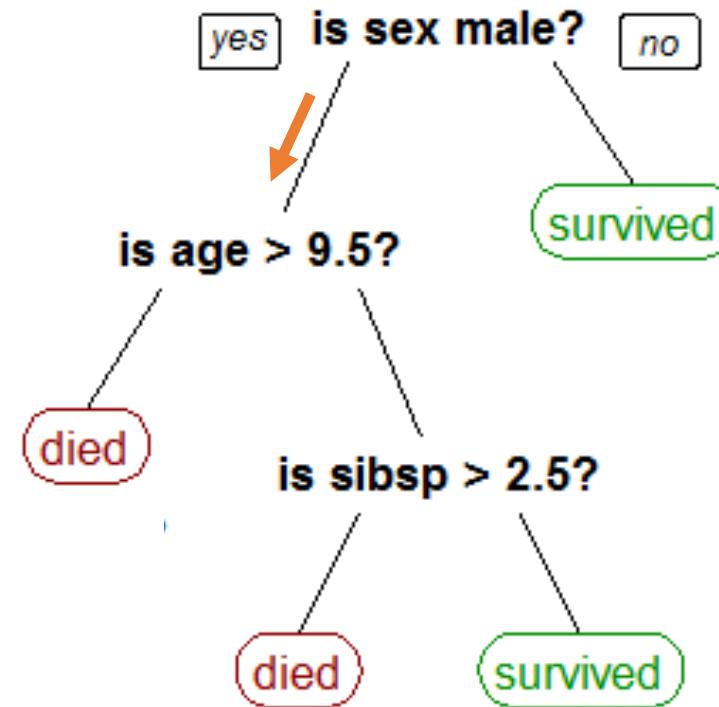
↓



# Решающее дерево

age sex sibsp

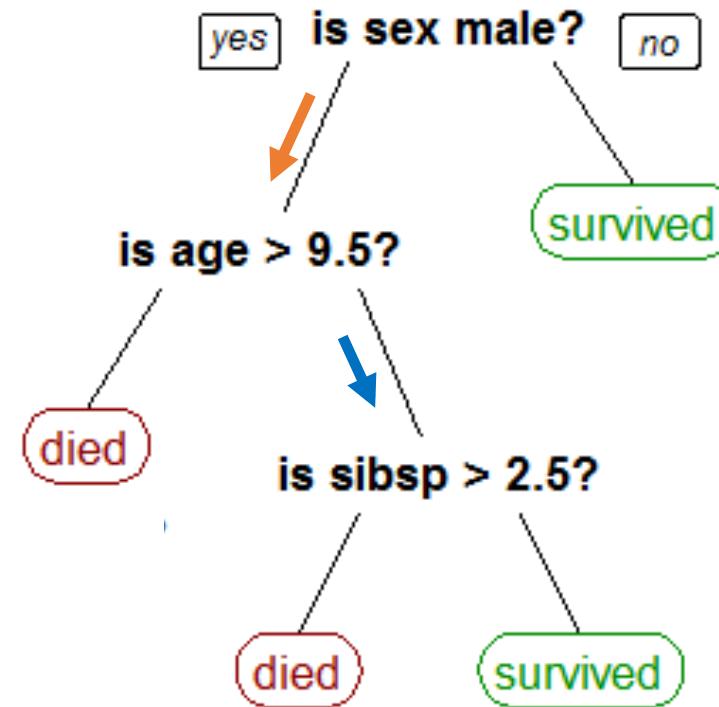
$$x = (9, \text{male}, 3)$$



# Решающее дерево

age sex sibsp

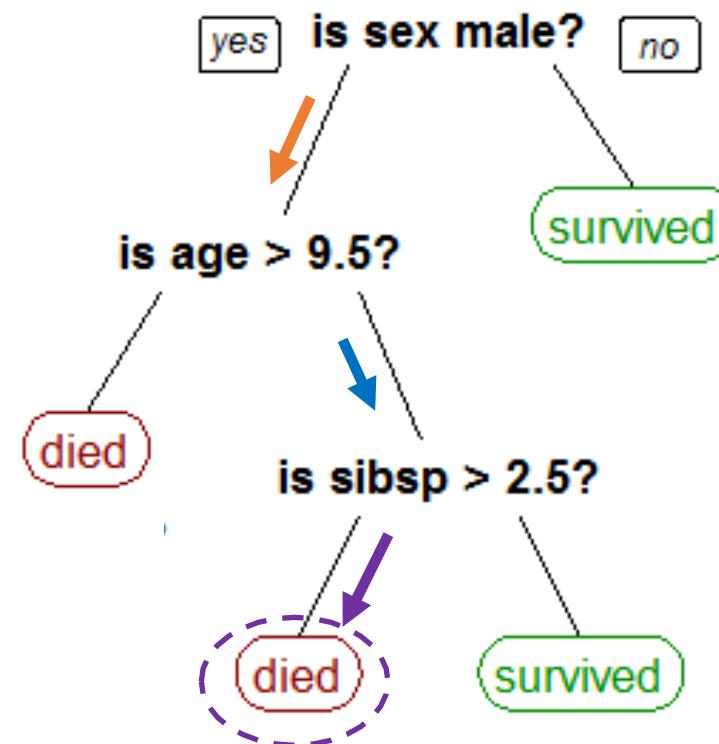
$$x = (9, \text{male}, 3)$$



# Решающее дерево

age      sex      sibsp

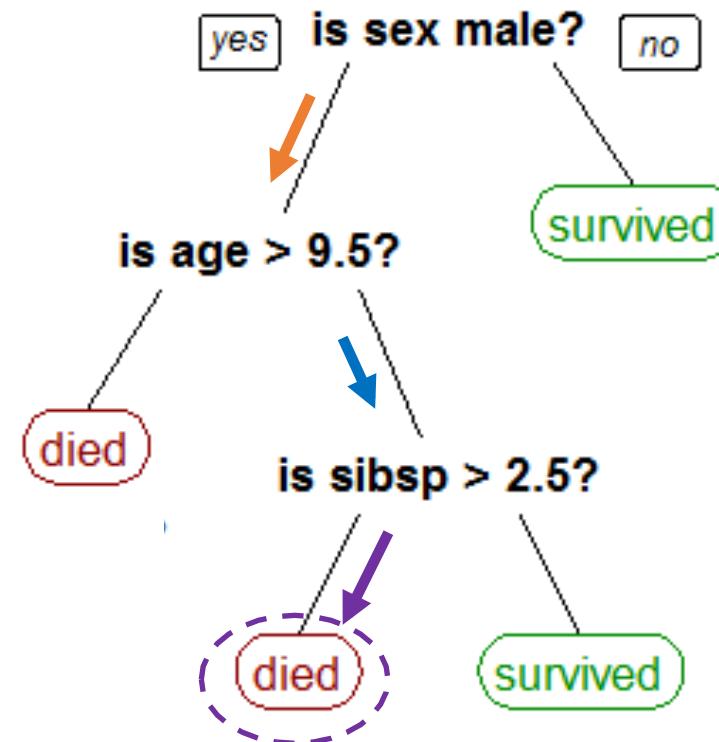
$$x = (9, \text{male}, 3)$$



# Решающее дерево

age      sex      sibsp

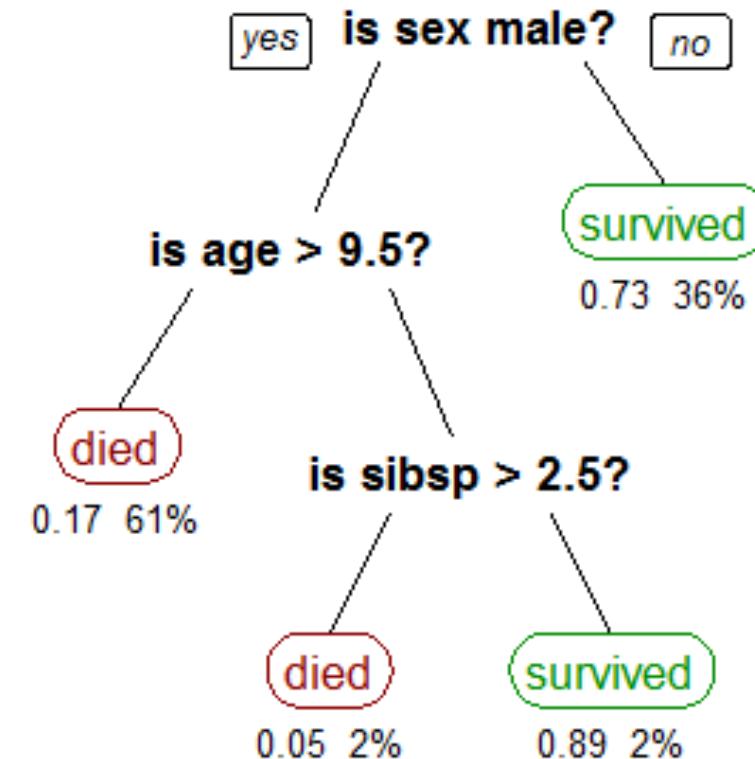
$$x = (9, \text{male}, 3)$$



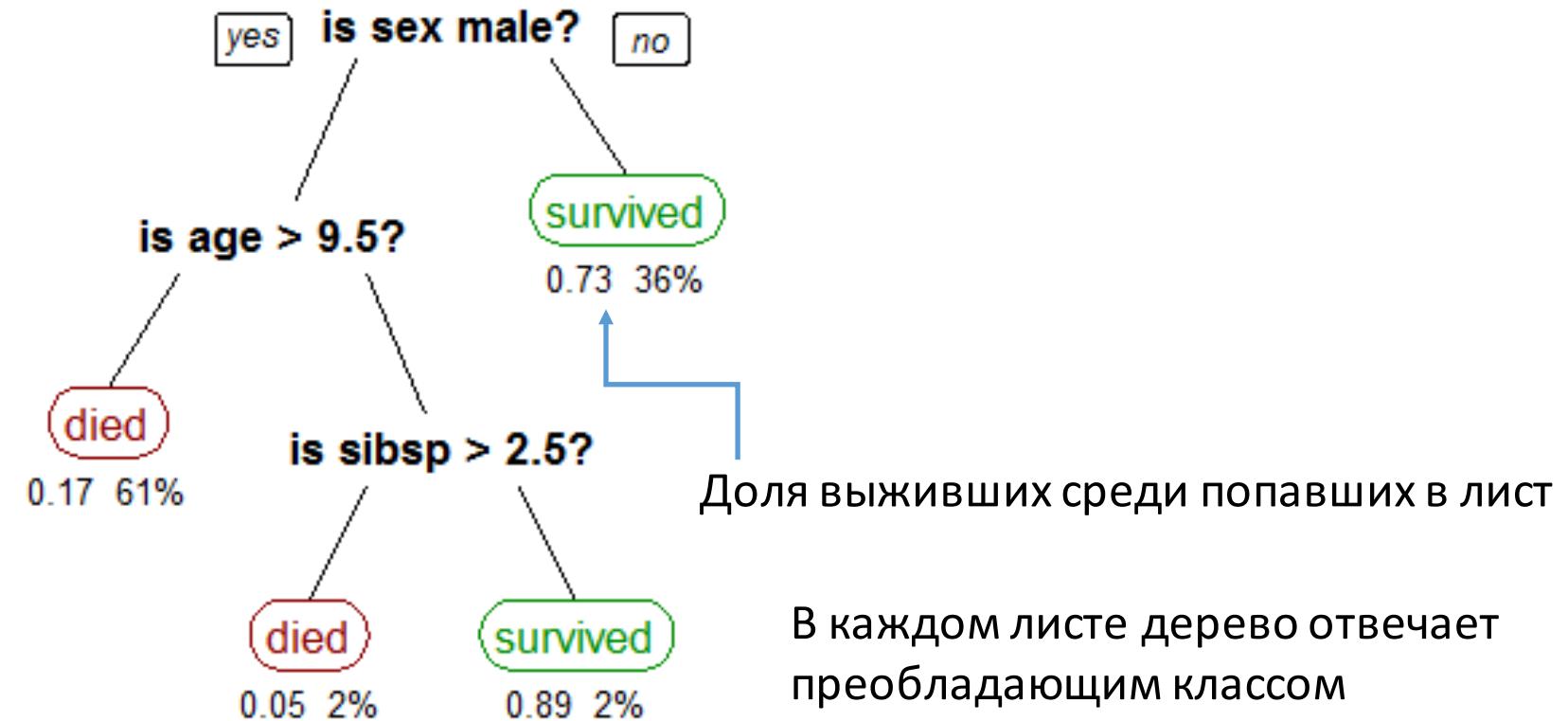
$$y = \text{died}$$

## 2. Решающие деревья в классификации и регрессии

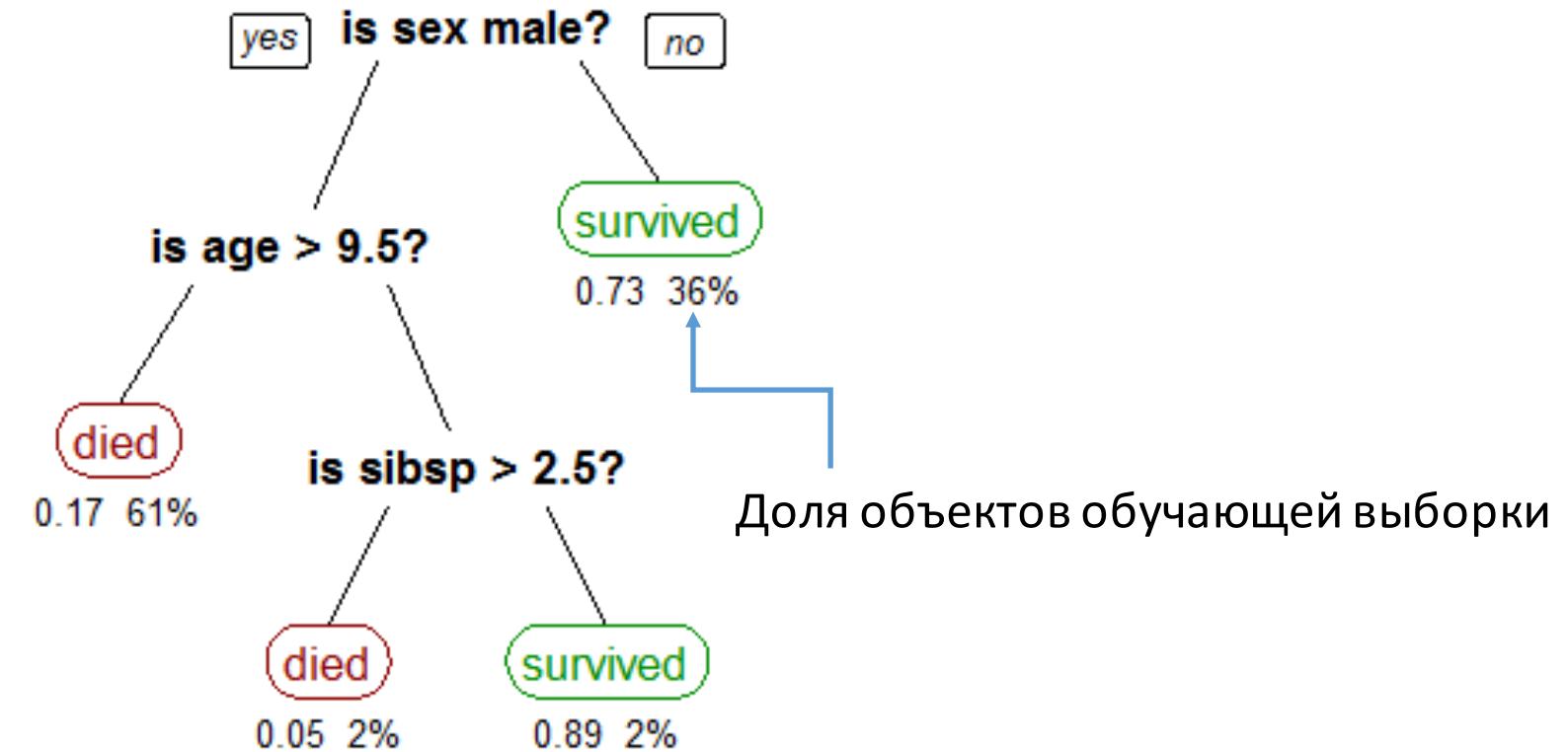
# Решающее дерево: классификация



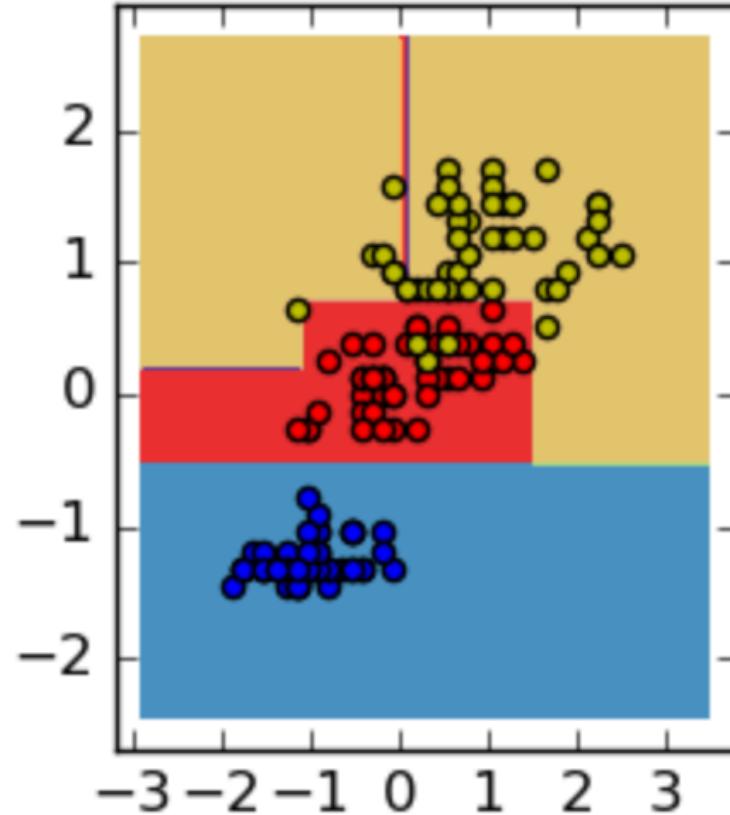
# Решающее дерево: классификация



# Решающее дерево: классификация

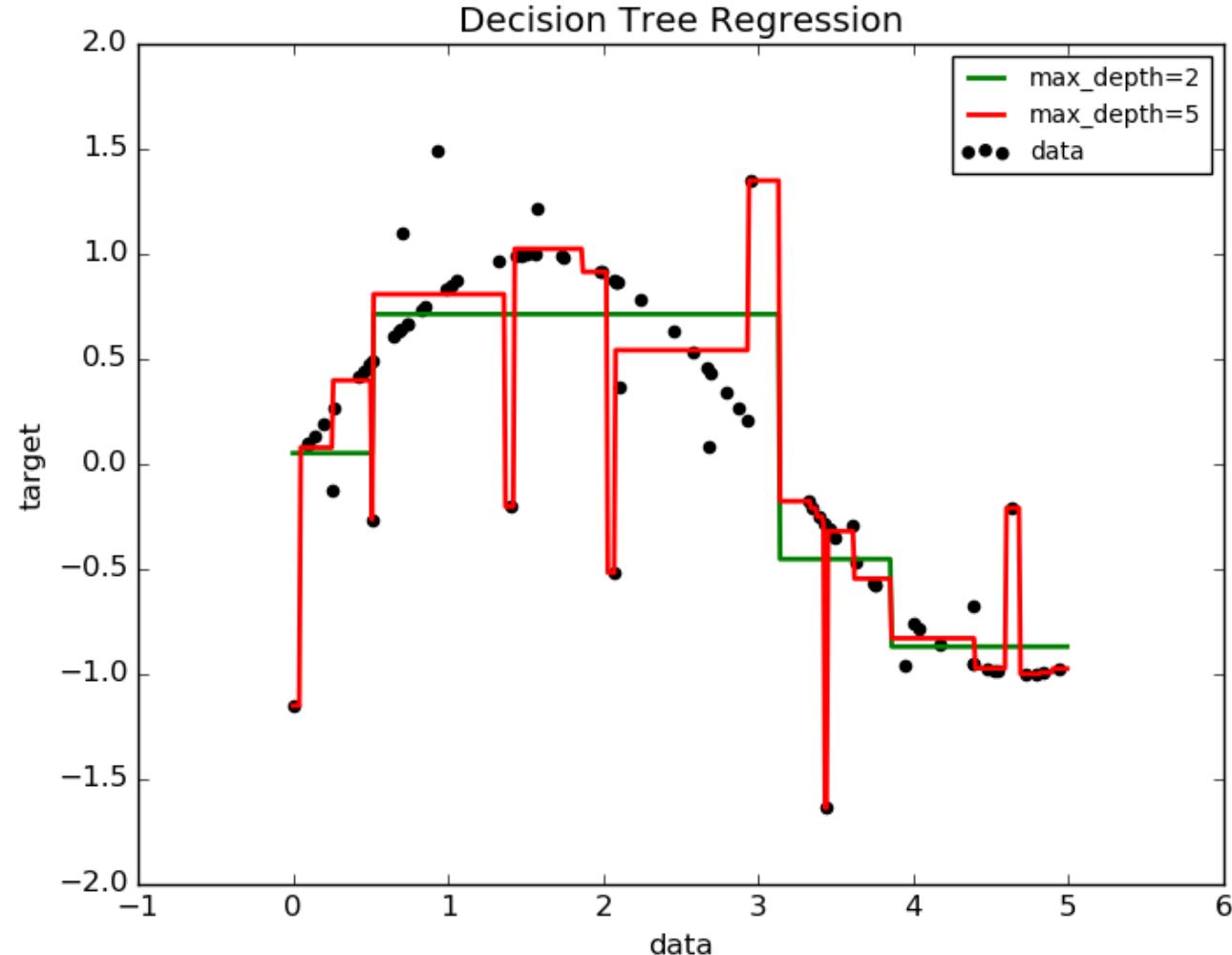


# Решающее дерево: классификация



Пример: 3 класса и 2 признака

# Решающее дерево: регрессия



Пример: восстановление зависимости  $y(x) = \sin x$  с помощью решающих деревьев глубины 2 и глубины 5

В каждом листе дерево отвечает некоторой константой

### 3. Как строить решающие деревья

# Рекурсивное построение

Строим разбиение  
выборки по значению  
одного из признаков

$$x^{(j)} < t$$

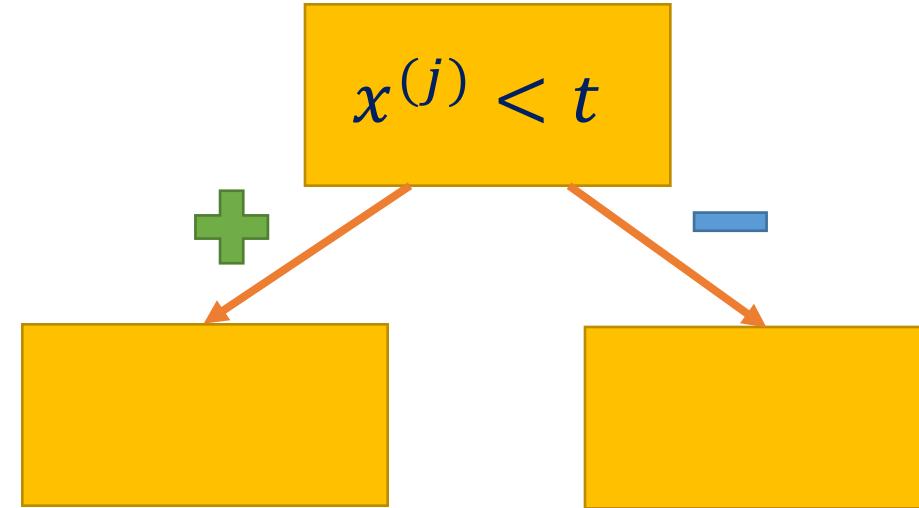
# Рекурсивное построение

Строим разбиение  
выборки по значению  
одного из признаков

$$x^{(j)} < t$$

Фактически нужно  
только выбрать  $j$  и  $t$   
наилучшим образом

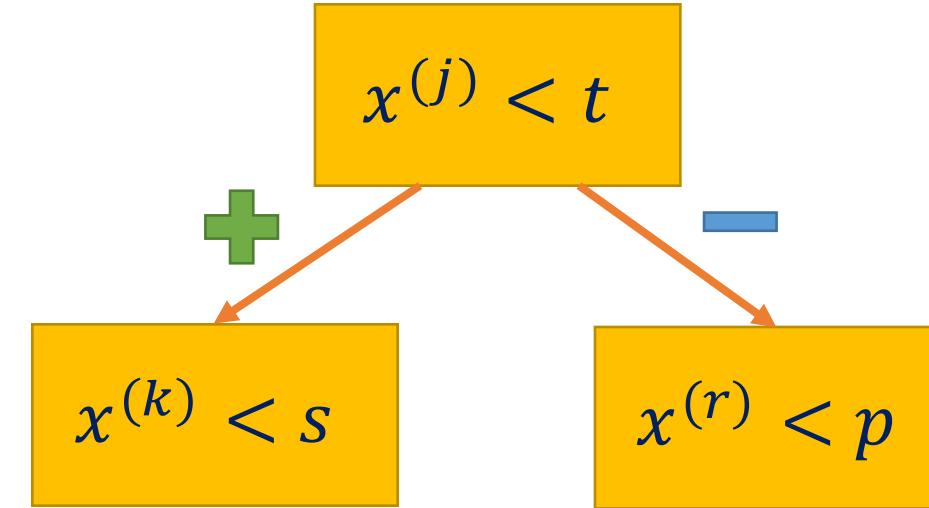
# Рекурсивное построение



Выборка делится  
по этому условию  
на две части

# Рекурсивное построение

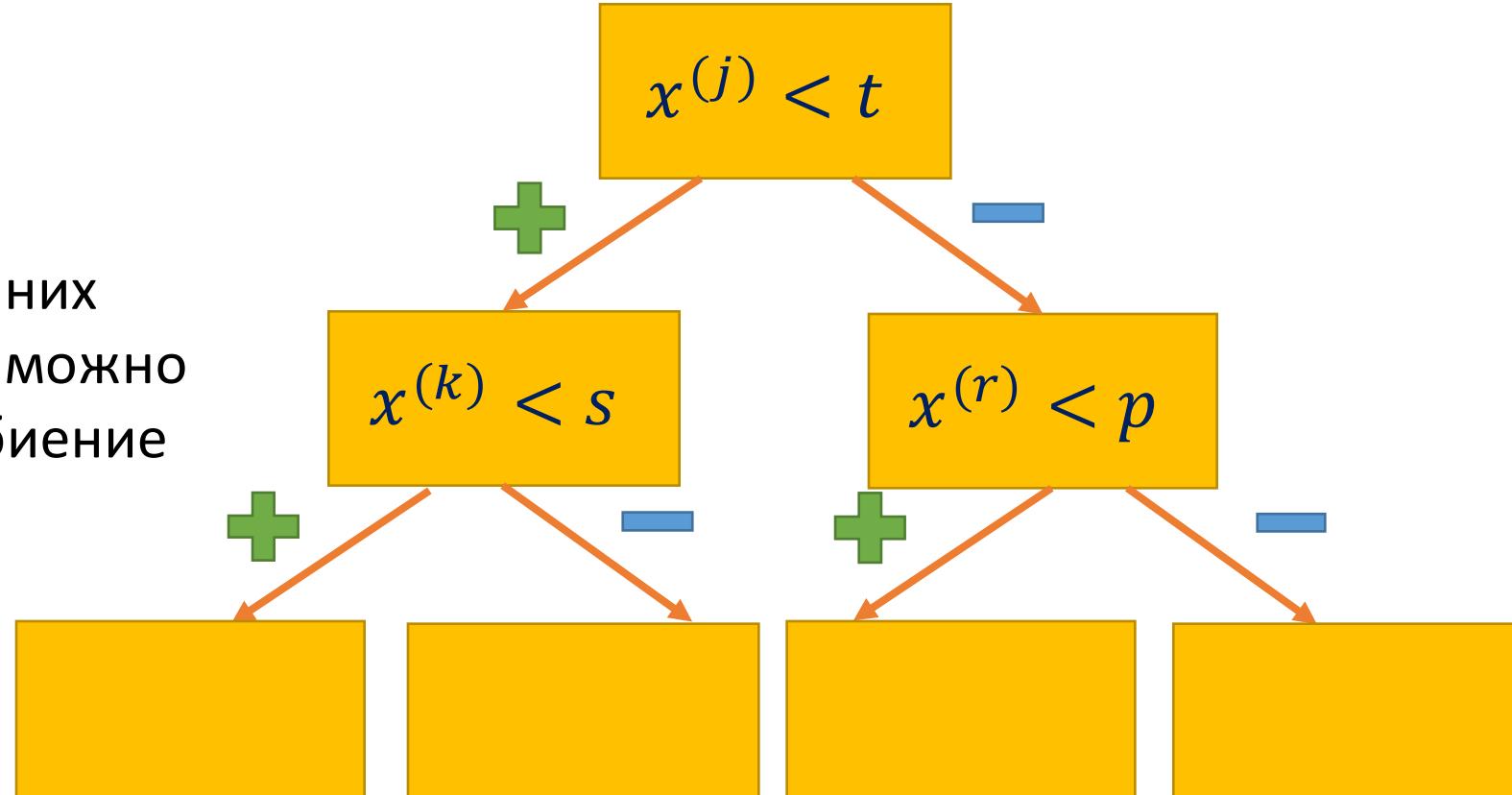
В каждой из них  
теперь тоже можно  
сделать разбиение



Выборка делится  
по этому условию  
на две части

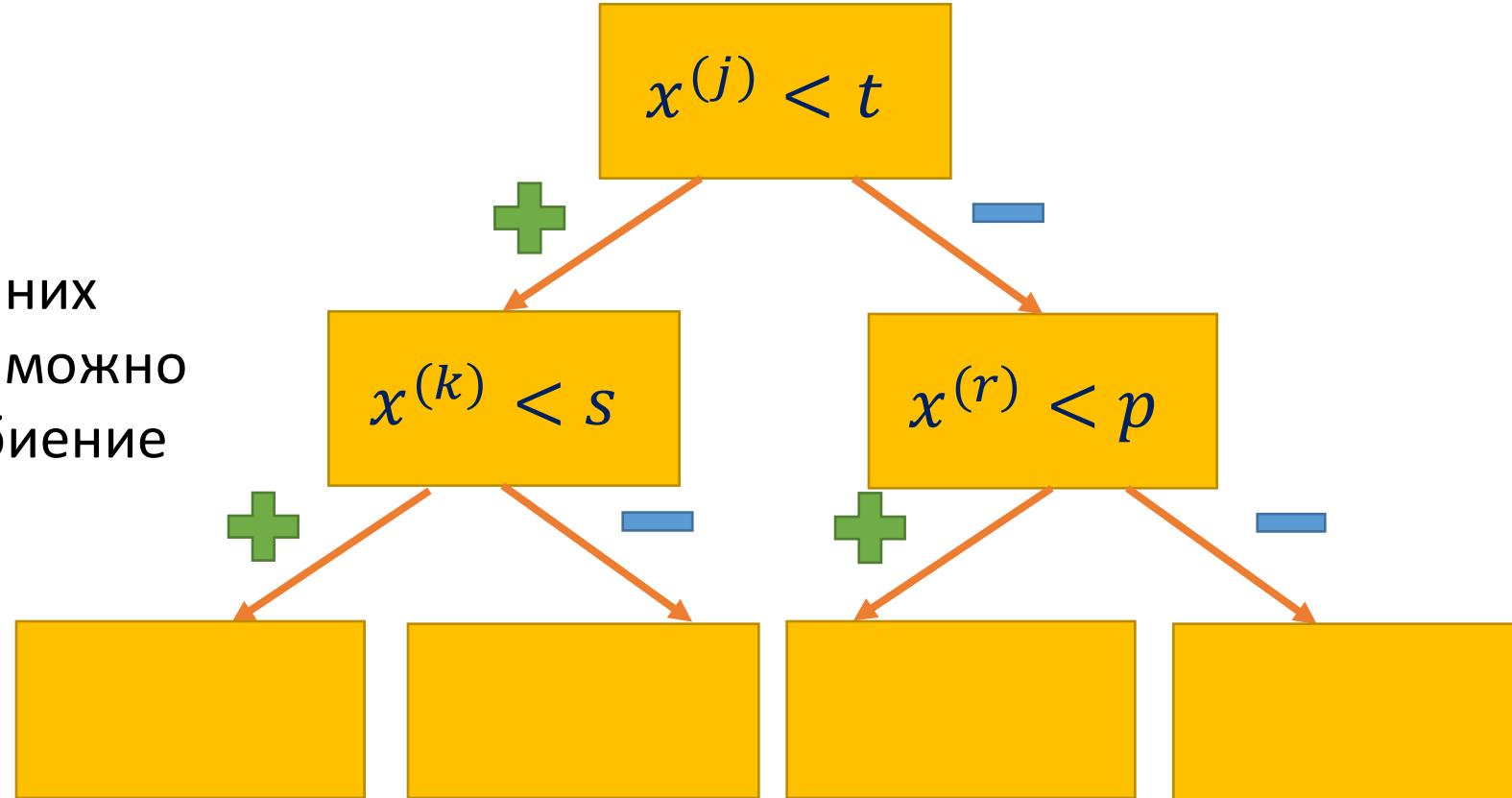
# Рекурсивное построение

В каждой из них  
теперь тоже можно  
сделать разбиение



# Рекурсивное построение

В каждой из них  
теперь тоже можно  
сделать разбиение



Процесс можно продолжать в тех узлах, в  
которые попадает достаточно много объектов

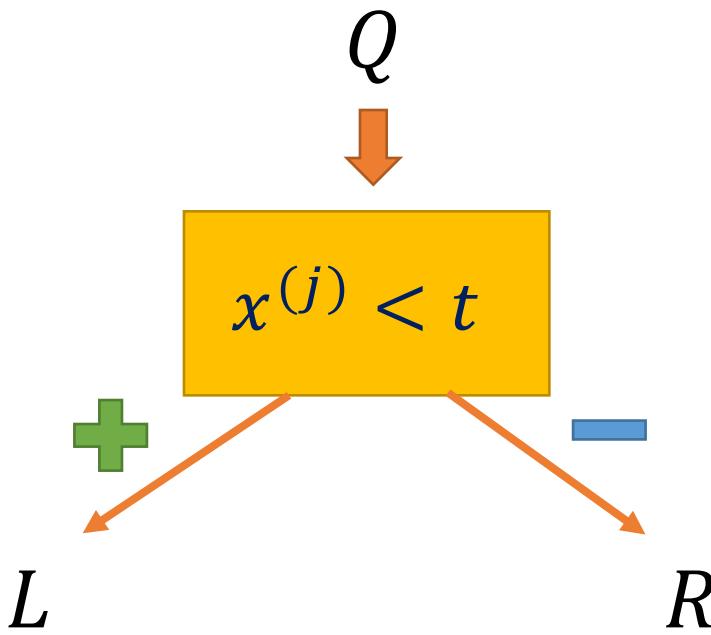
# Выбор разбиения

$Q$

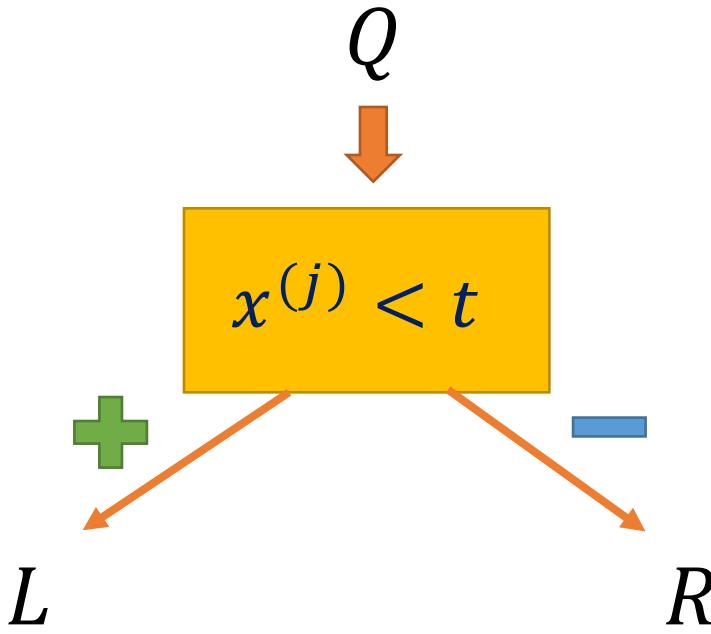


$$x^{(j)} < t$$

# Выбор разбиения

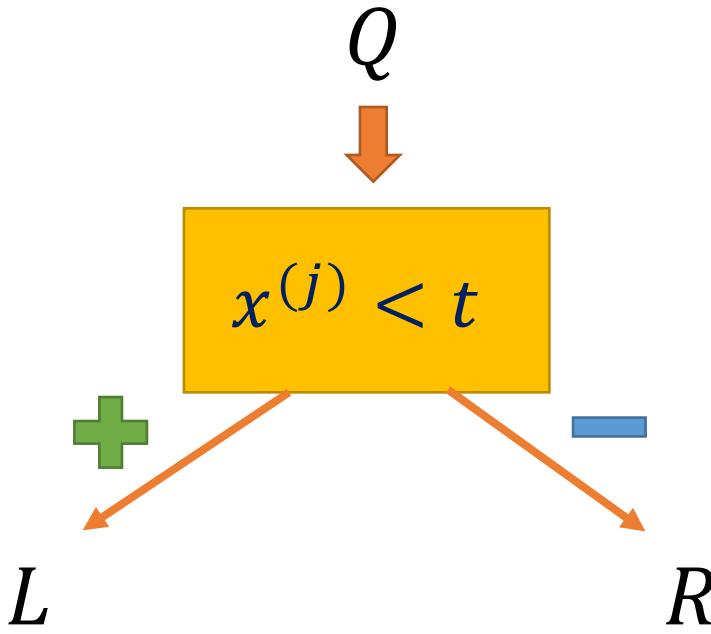


# Выбор разбиения



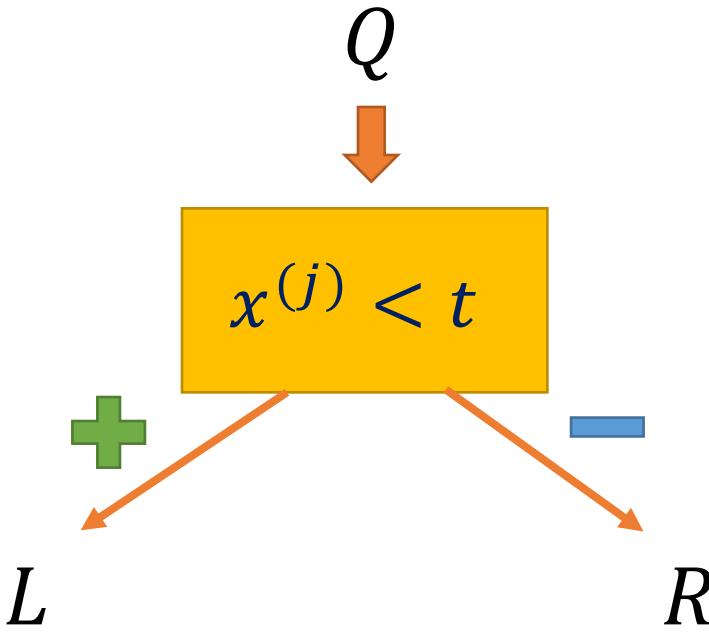
$$G(j, t) = \frac{|L|}{|Q|} H(L) + \frac{|R|}{|Q|} H(R)$$

# Выбор разбиения



$$G(j, t) = \frac{|L|}{|Q|} H(L) + \frac{|R|}{|Q|} H(R) \rightarrow \min_{j,t}$$

# Выбор разбиения



$$G(j, t) = \frac{|L|}{|Q|} H(L) + \frac{|R|}{|Q|} H(R) \rightarrow \min_{j,t}$$

$H(R)$  - мера «неоднородности» множества  $R$

# Критерии построения разбиений

$H(R)$  – мера «неоднородности» множества  $R$

# Критерии построения разбиений

$H(R)$  – мера «неоднородности» множества  $R$

Пусть мы решаем задачу классификации на 2 класса,  
 $p_0, p_1$  – доли объектов классов 0 и 1 в  $R$

1) Misclassification criteria:  $H(R) = 1 - p_{max}$

2) Entropy criteria:  $H(R) = -p_0 \ln p_0 - p_1 \ln p_1$

3) Gini criteria:  $H(R) = 1 - p_0^2 - p_1^2 = 2p_0p_1$

# Критерии построения разбиений

$H(R)$  – мера «неоднородности» множества  $R$

Пусть мы решаем задачу классификации на  $K$  классов,  
 $p_1, \dots, p_K$  – доли объектов классов 1, ...,  $K$  в  $R$

1) Misclassification criteria:  $H(R) = 1 - p_{max}$

2) Entropy criteria:

$$H(R) = - \sum_{k=1}^K p_k \ln p_k$$

3) Gini criteria:

$$H(R) = \sum_{k=1}^K p_k(1 - p_k)$$

# Критерии построения разбиений

$H(R)$  – мера «неоднородности» множества  $R$

Чтобы решать задачу регрессии, достаточно взять среднеквадратичную ошибку в качестве  $H(R)$ :

$$H(R) = \frac{1}{|R|} \sum_{x_i \in R} (y_i - \bar{y})^2$$

# Критерии построения разбиений

$H(R)$  – мера «неоднородности» множества  $R$

Чтобы решать задачу регрессии, достаточно взять среднеквадратичную ошибку в качестве  $H(R)$ :

$$H(R) = \frac{1}{|R|} \sum_{x_i \in R} (y_i - \bar{y})^2$$

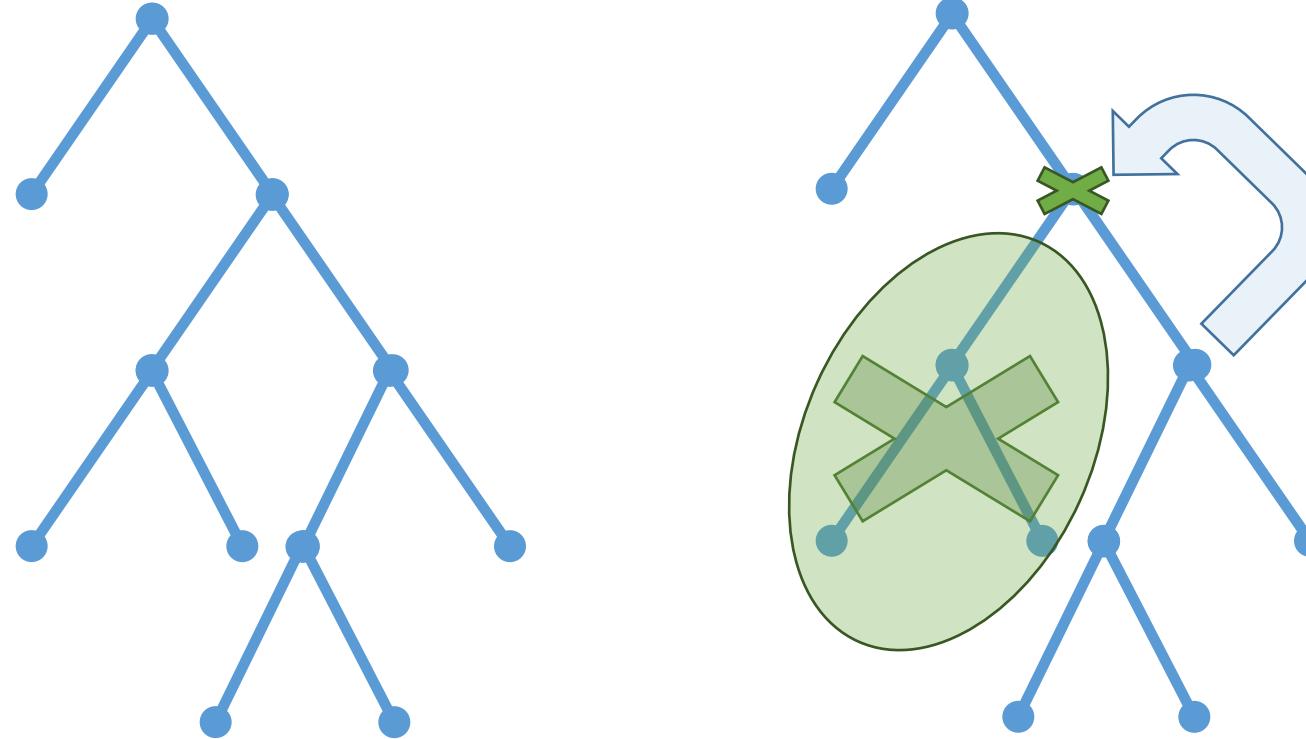
$$\bar{y} = \frac{1}{|R|} \sum_{x_i \in R} y_i$$

## 4. Дополнительные темы

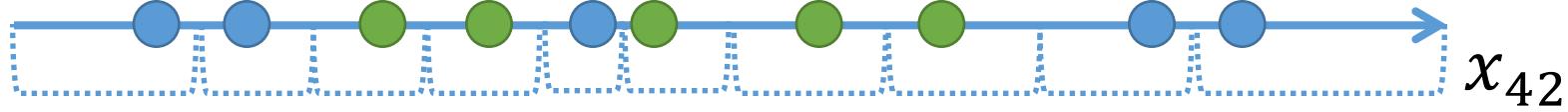
# Pruning

- Pre-prunning:
  - Ограничиваем рост дерева до того как оно построено
  - Если в какой-то момент информативность признаков в разбиении меньше порога – не разбиваем вершину
- Post-prunning:
  - Упрощаем дерево после того как дерево построено

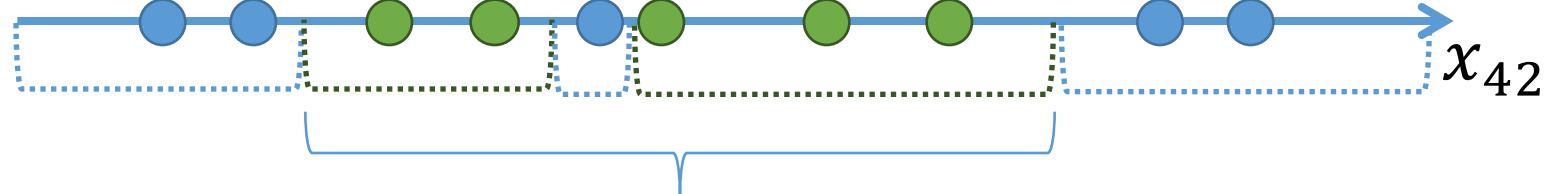
# Post-pruning



# Бинаризация



$$\varphi(x) = [a_k < x \leq a_{k+1}]$$



# Вариации алгоритма построения

- C4.5
- C5.0
- CART

# Итог

1. Что такое решающие деревья
2. Решающие деревья в классификации и регрессии
3. Как строить решающие деревья
4. Дополнительные темы

### **III. Ансамбли деревьев**

# План

1. Введение в построение ансамблей
2. Random Forest
3. Gradient Boosted Decision Trees (GBDT)
4. Библиотеки

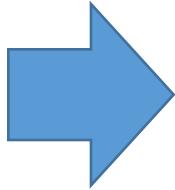
# Bagging

Bagging = Bootstrap aggregation

# Бутстреп

Выборка:

| №   | Значение |
|-----|----------|
| 1   | 2.4      |
| 2   | 3.2      |
| 3   | 2.5      |
| 4   | 1.8      |
| 5   | 1.9      |
| ... | ...      |
| N   | 2.2      |



Хотим вычислить какую-то величину  $X$  по данным наблюдениями.

Было бы здорово вычислить  $X$  на многих выборках из распределения, а потом усреднить, но их у нас нет

# Бутстреп

## Решение:

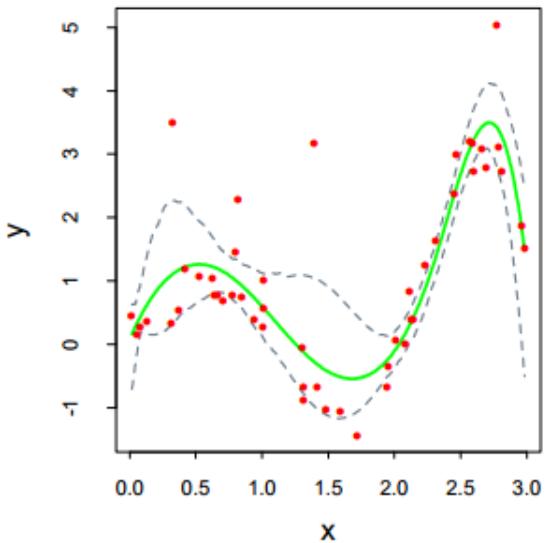
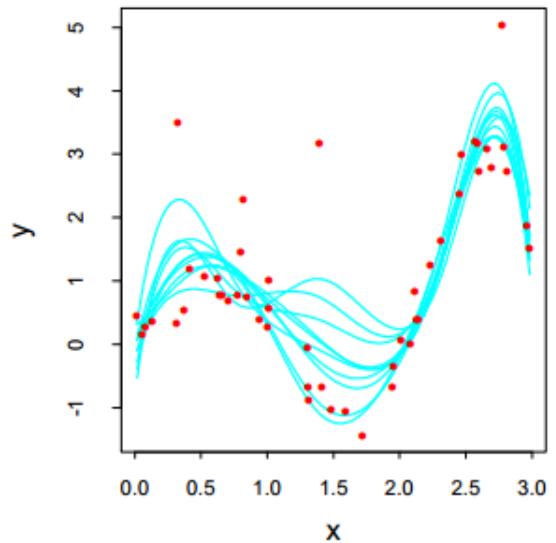
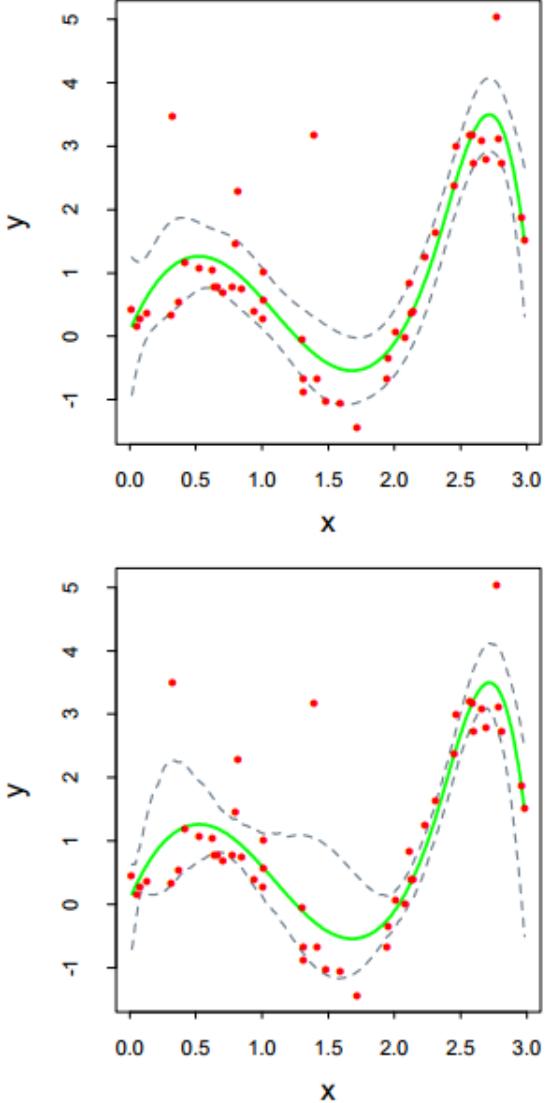
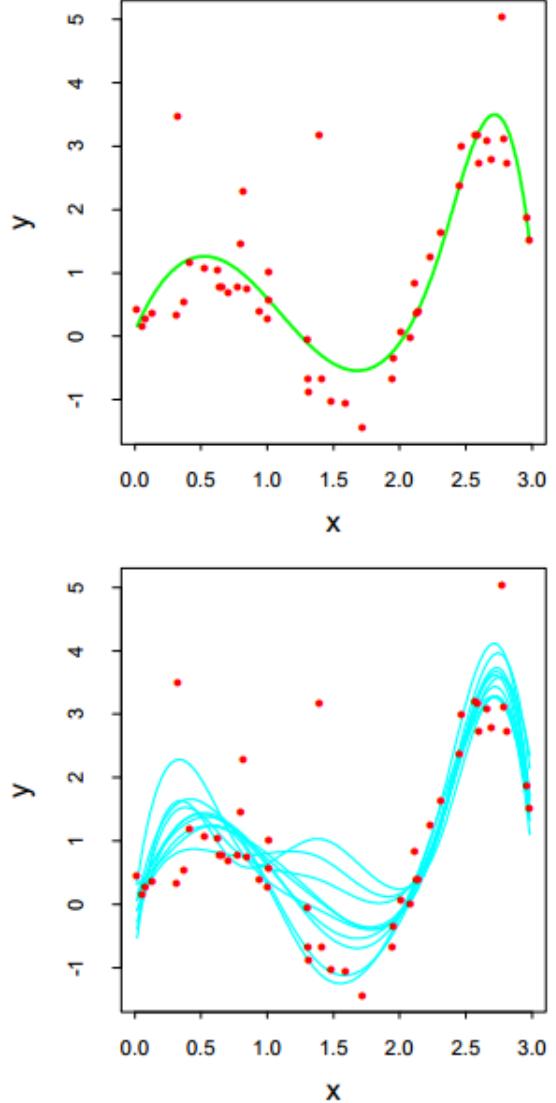
1. Выбираем наугад одно наблюдение из имеющихся.
2. Повторяем пункт 1 столько раз, сколько у нас есть наблюдений. При этом некоторые из них мы можем выбрать повторно
3. Считаем интересующие нас величины по новой выборке. Запоминаем результат.
4. Повторяем пункты 1-3 много раз и усредняем

# Bagging

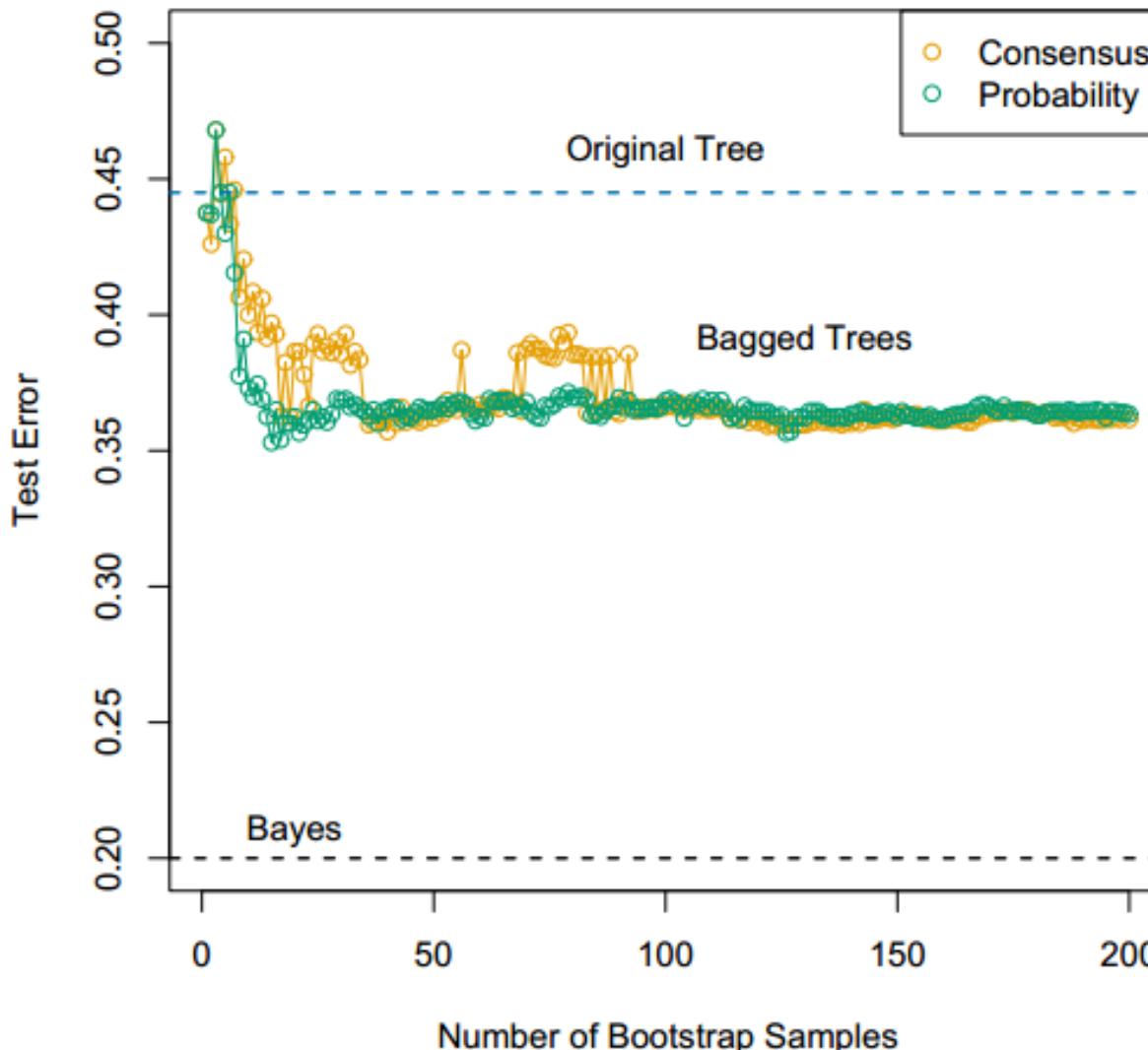
Bagging = Bootstrap aggregation

По схеме выбора с возвращением, генерируем  $M$  обучающих выборок такого же размера, обучаем на них модели и усредняем

# Bagging



# Бэггинг в классификации

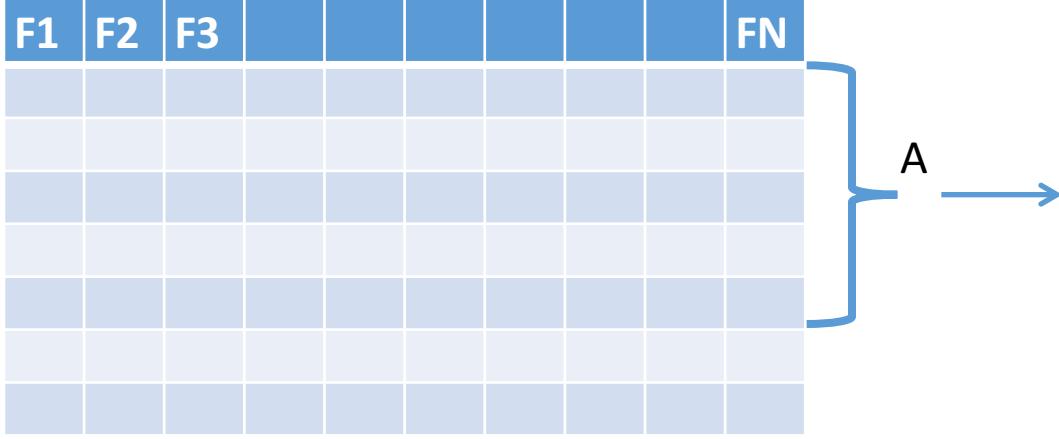


## Вариации: Pasting, RSM

- RSM – Random Subspace Method, выбираем не объекты, а признаки
- Pasting – выбираем объекты без возвращения

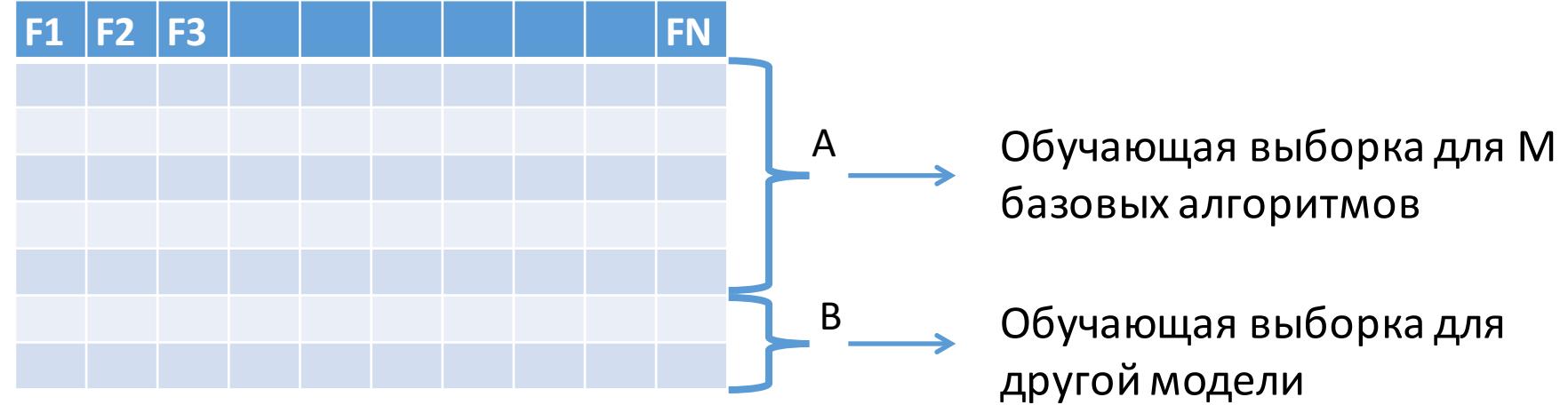
# Stacking

# Stacking

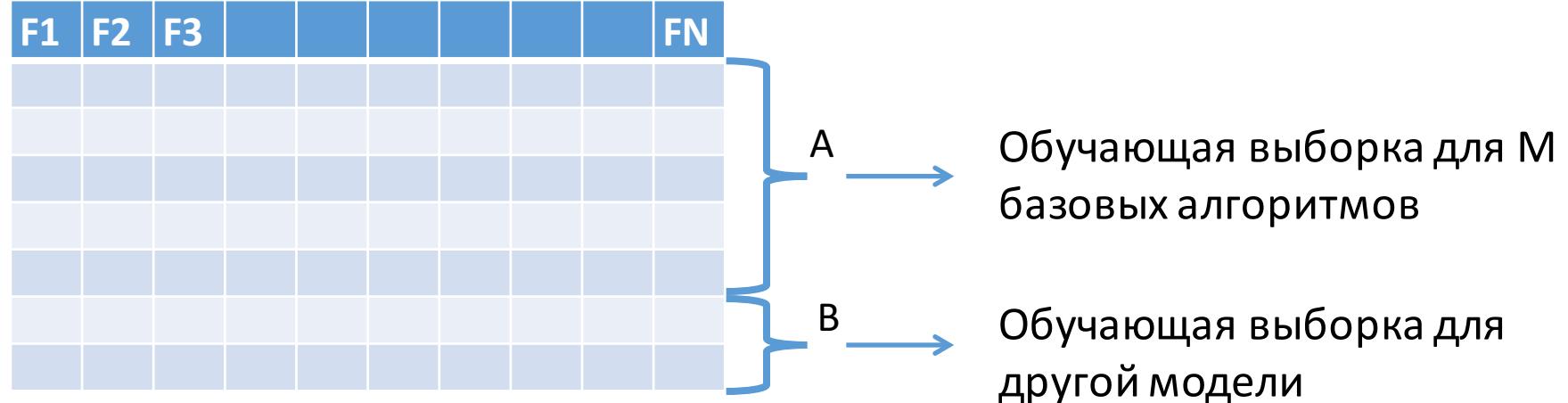


Обучающая выборка для M  
базовых алгоритмов

# Stacking

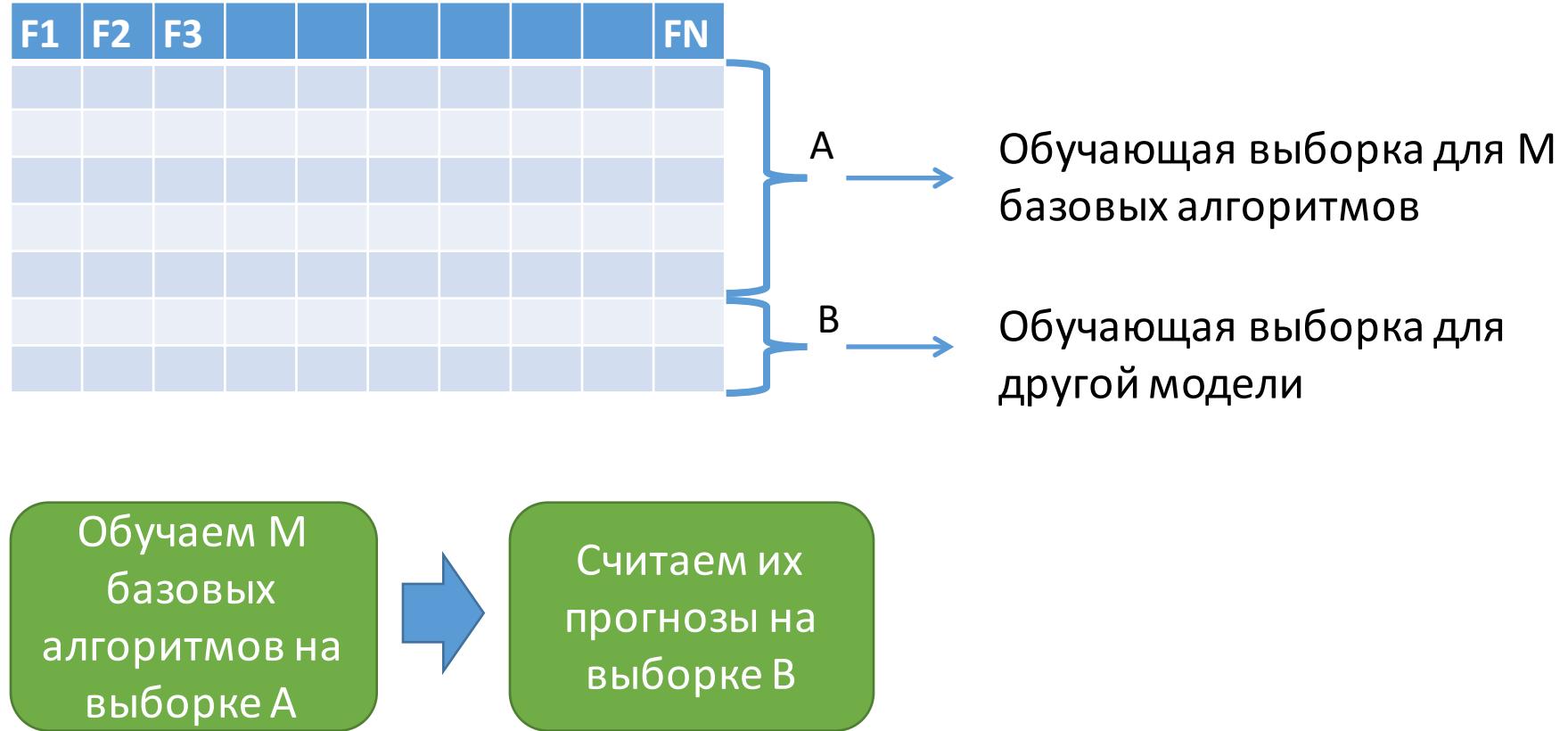


# Stacking

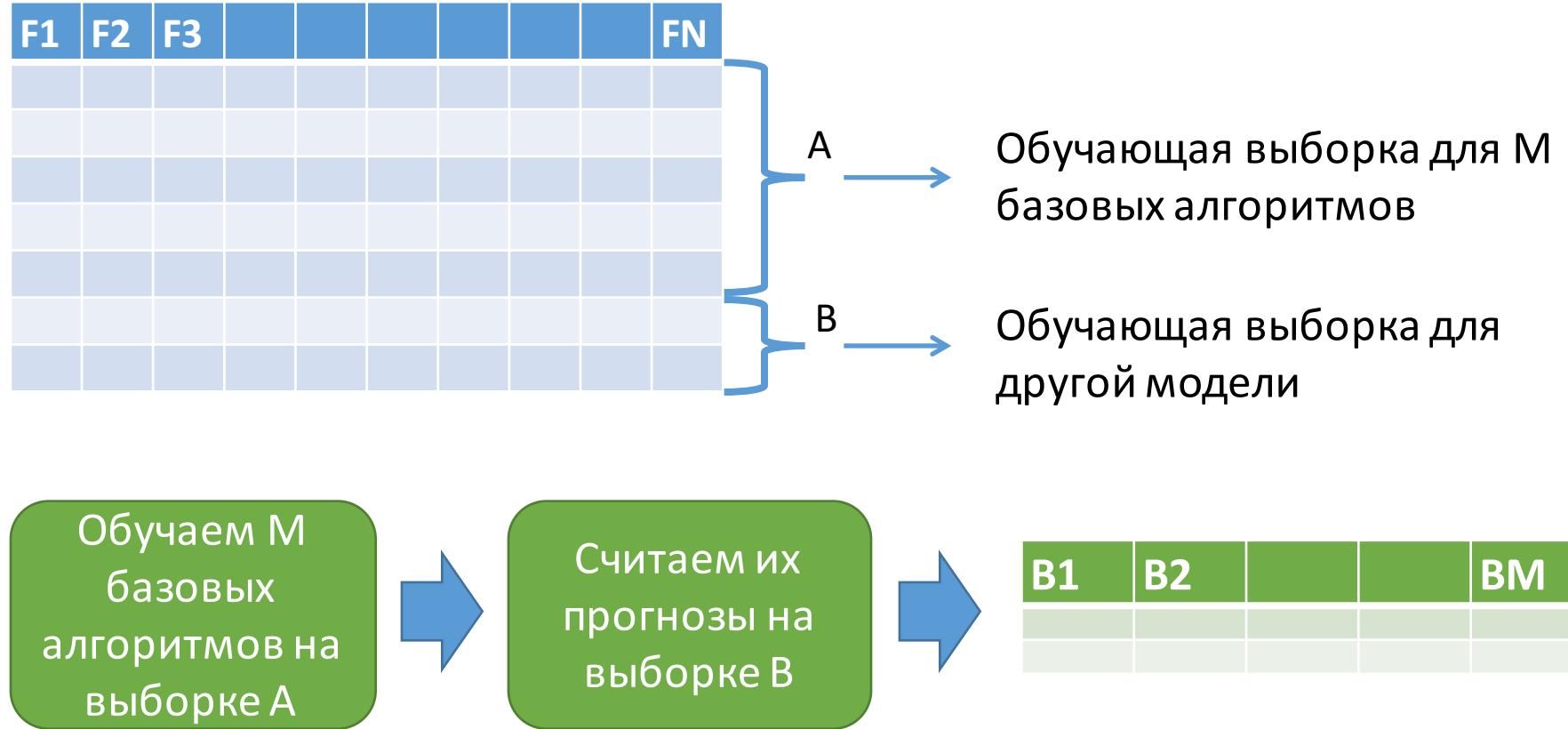


Обучаем M базовых алгоритмов на выборке A

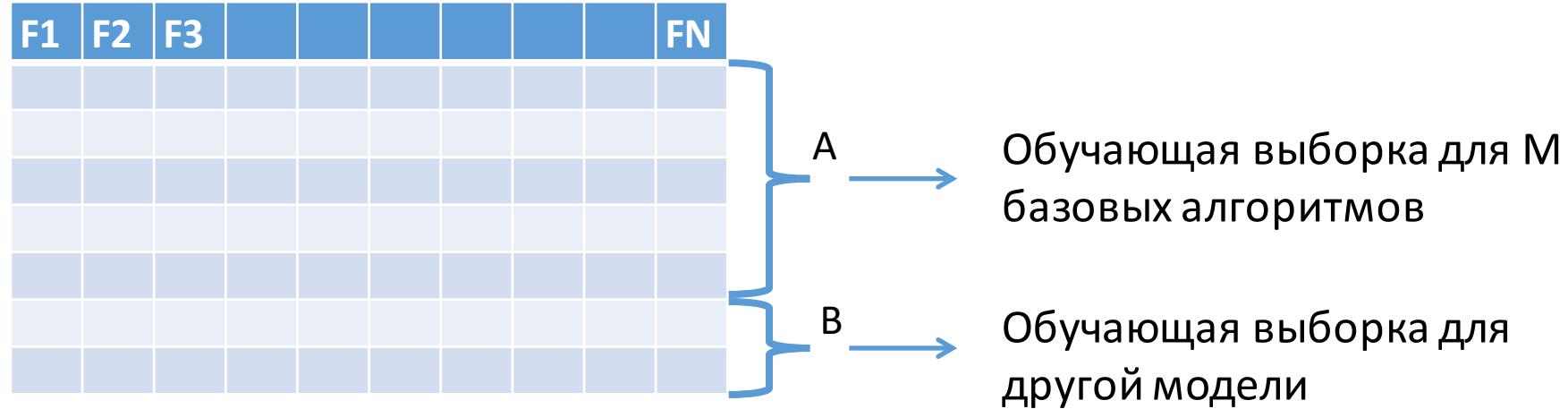
# Stacking



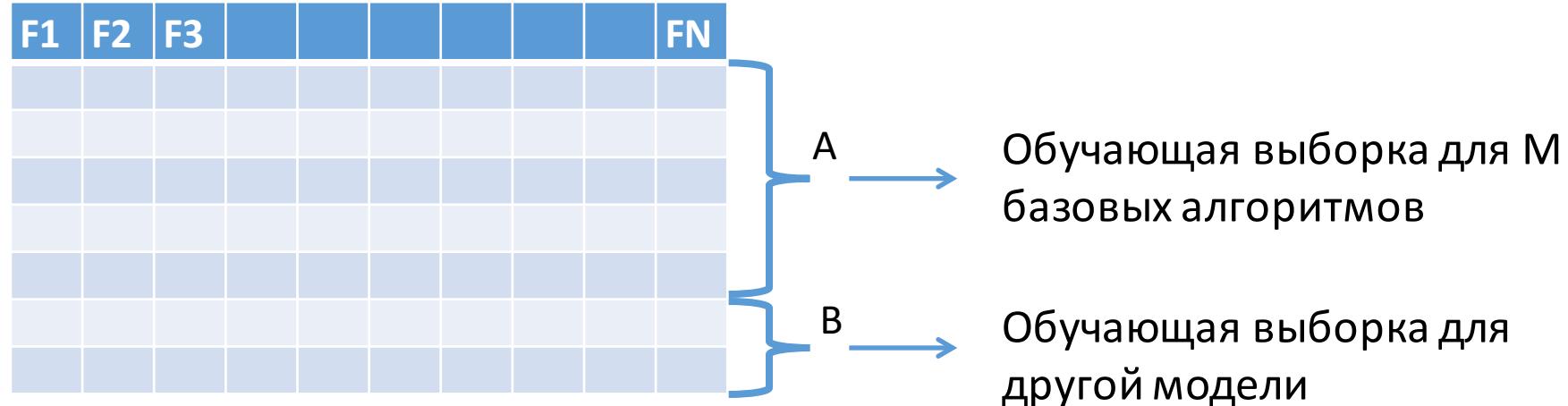
# Stacking



# Stacking



# Stacking



# Blending

Смесь нескольких сильных классификаторов:

$$a(x) = \sum_{t=1}^T \alpha_t b_t(x)$$

+ веса неотрицательны и дают в сумме единицу

# Blending

Смесь нескольких сильных классификаторов:

$$a(x) = \sum_{t=1}^T \alpha_t b_t(x)$$

+ веса неотрицательны и дают в сумме единицу

Преимущества и недостатки:

- Очень прост идеально, хорошо работает, логичен

# Blending

Смесь нескольких сильных классификаторов:

$$a(x) = \sum_{t=1}^T \alpha_t b_t(x)$$

+ веса неотрицательны и дают в сумме единицу

Преимущества и недостатки:

- Очень прост идеально, хорошо работает, логичен
- Иногда надо перебирать веса или использовать дискретную оптимизацию

# Blending

Смесь нескольких сильных классификаторов:

$$a(x) = \sum_{t=1}^T \alpha_t b_t(x)$$

+ веса неотрицательны и дают в сумме единицу

Преимущества и недостатки:

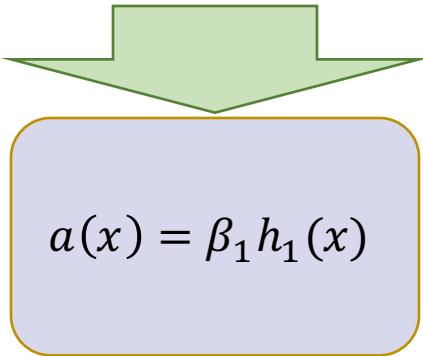
- Очень прост идеально, хорошо работает, логичен
- Иногда надо перебирать веса или использовать дискретную оптимизацию
- Не всегда композиция в виде взвешенной суммы – то, что надо. Иногда нужна более сложная композиция

# Boosting

Бустинг – жадное построение  
взвешенной суммы базовых  
алгоритмов  $h_k(x)$

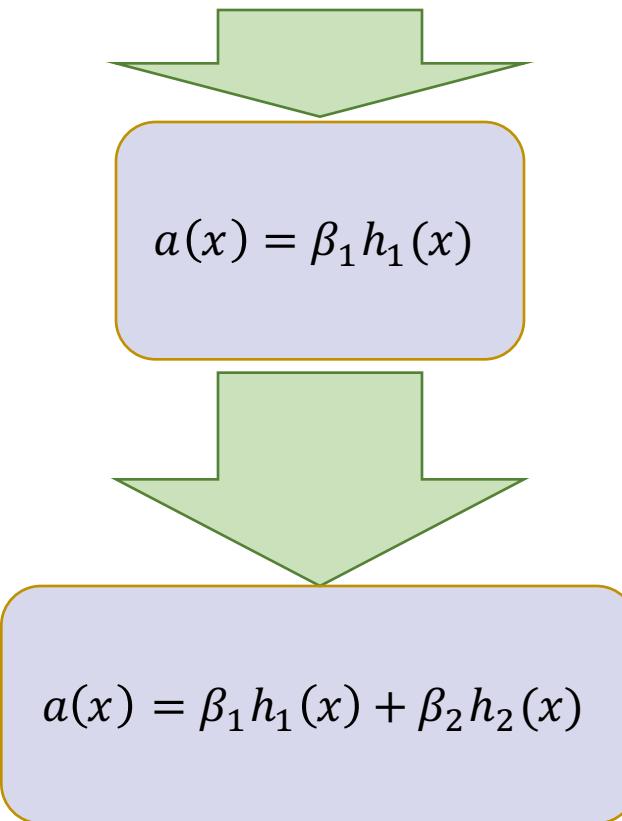
# Boosting

Бустинг – жадное построение  
взвешенной суммы базовых  
алгоритмов  $h_k(x)$


$$a(x) = \beta_1 h_1(x)$$

# Boosting

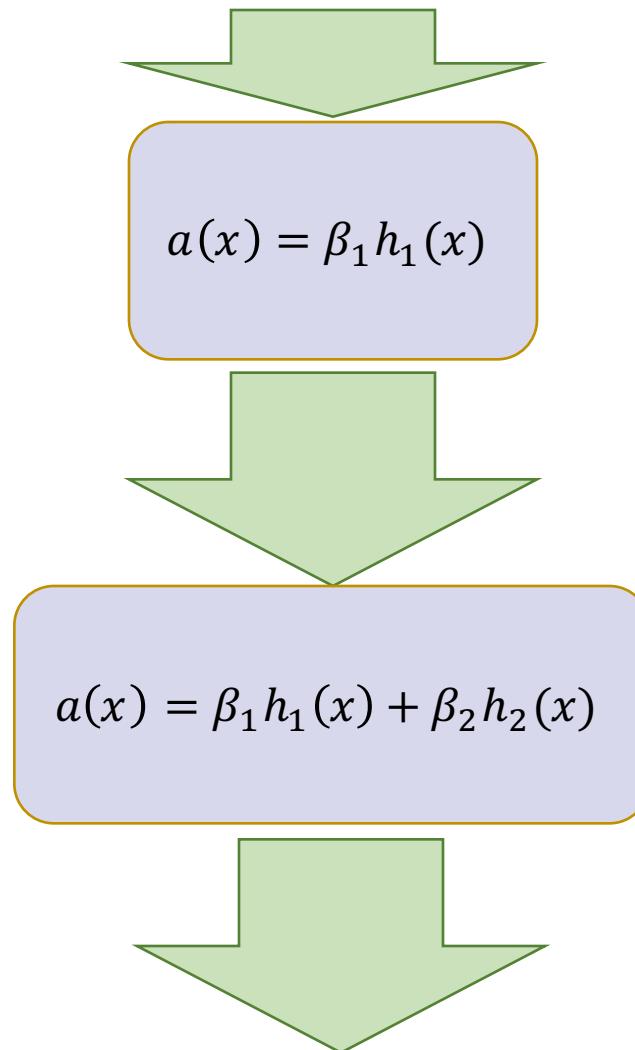
Бустинг – жадное построение  
взвешенной суммы базовых  
алгоритмов  $h_k(x)$



# Boosting

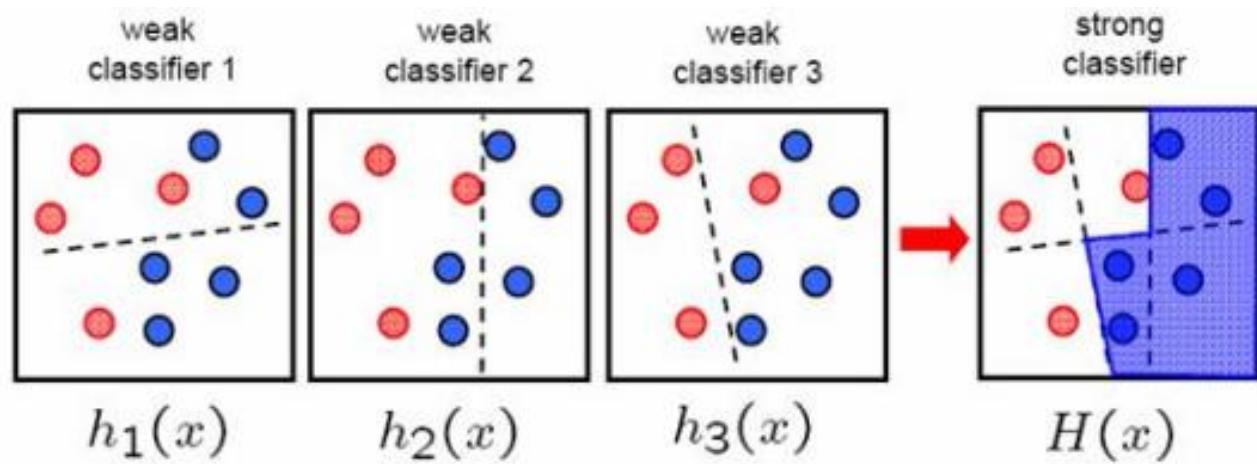
Бустинг – жадное построение  
взвешенной суммы базовых  
алгоритмов  $h_k(x)$

$$a(x) = \sum_{t=1}^T \beta_t h_t(x)$$

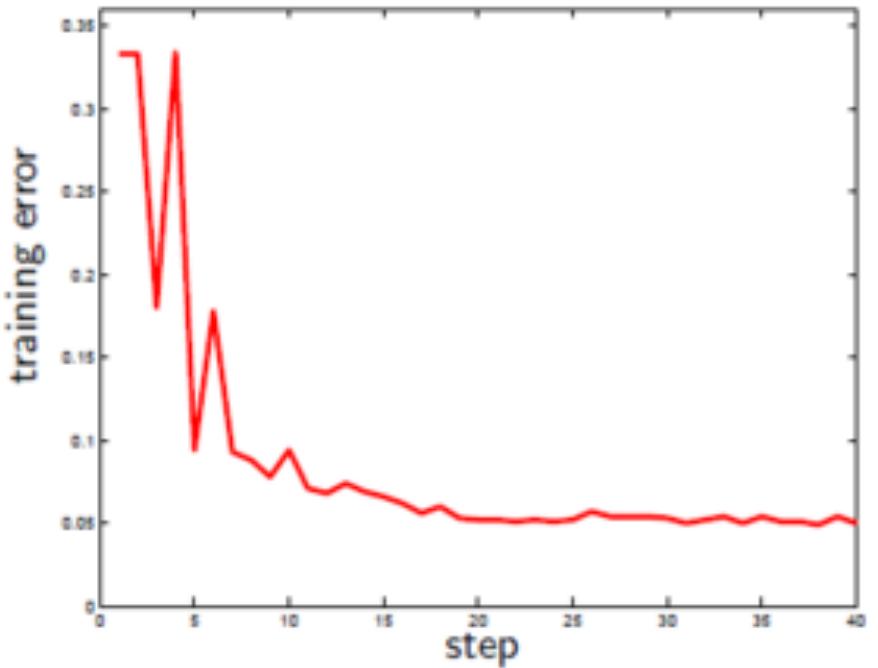
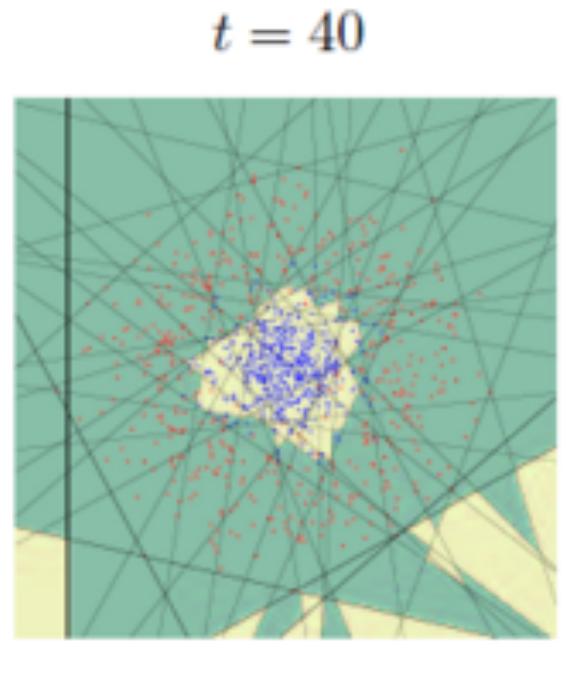


# «Слабые» алгоритмы

$h_k(x)$  – как правило, решающие деревья небольшой глубины или линейные модели



# Пример: бустинг над линейными классификаторами

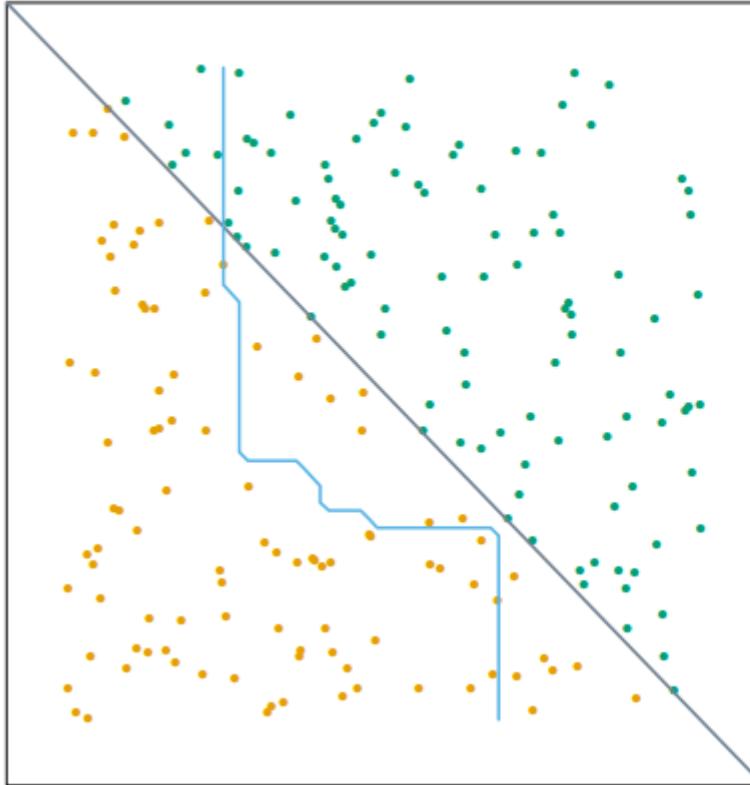


# Алгоритмы бустинга

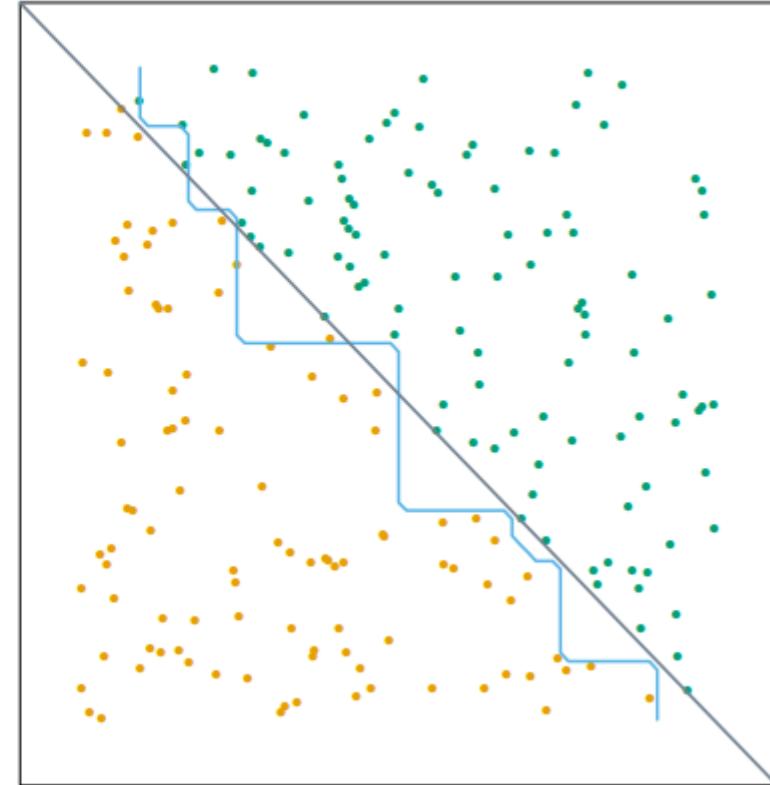
- Основные алгоритмы:
  - Градиентный бустинг
  - Адаптивный бустинг (AdaBoost)
- Вариации AdaBoost:
  - AnyBoost (произвольная функция потерь)
  - BrownBoost
  - GentleBoost
  - LogitBoost
  - ....

# Бэггинг и бустинг

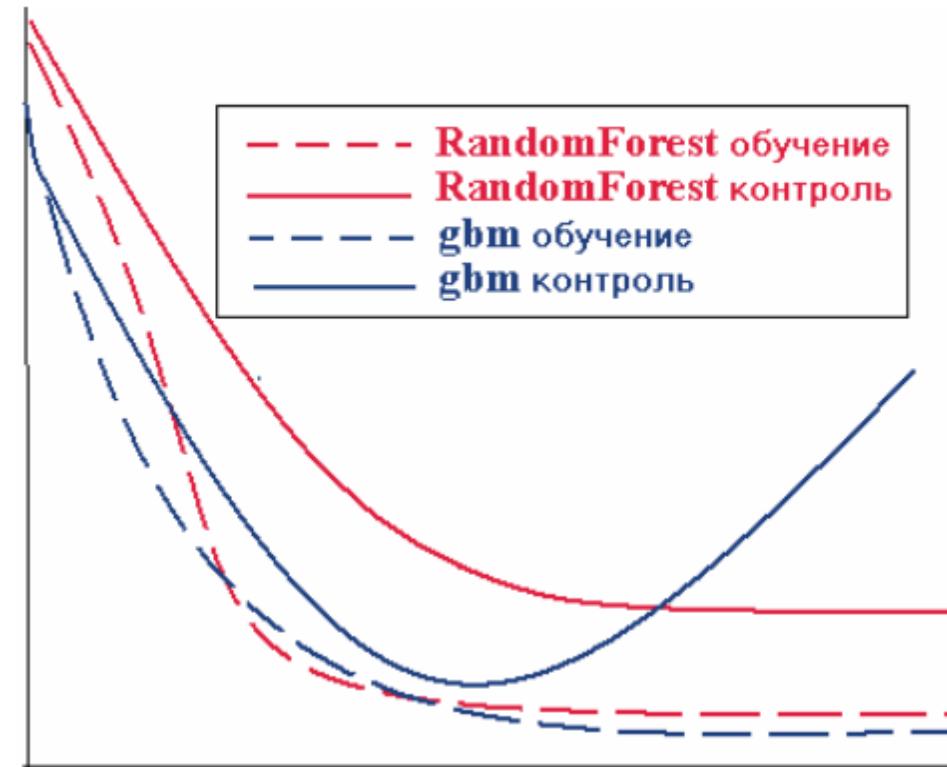
Bagged Decision Rule



Boosted Decision Rule



# Бэггинг и бустинг: переобучение

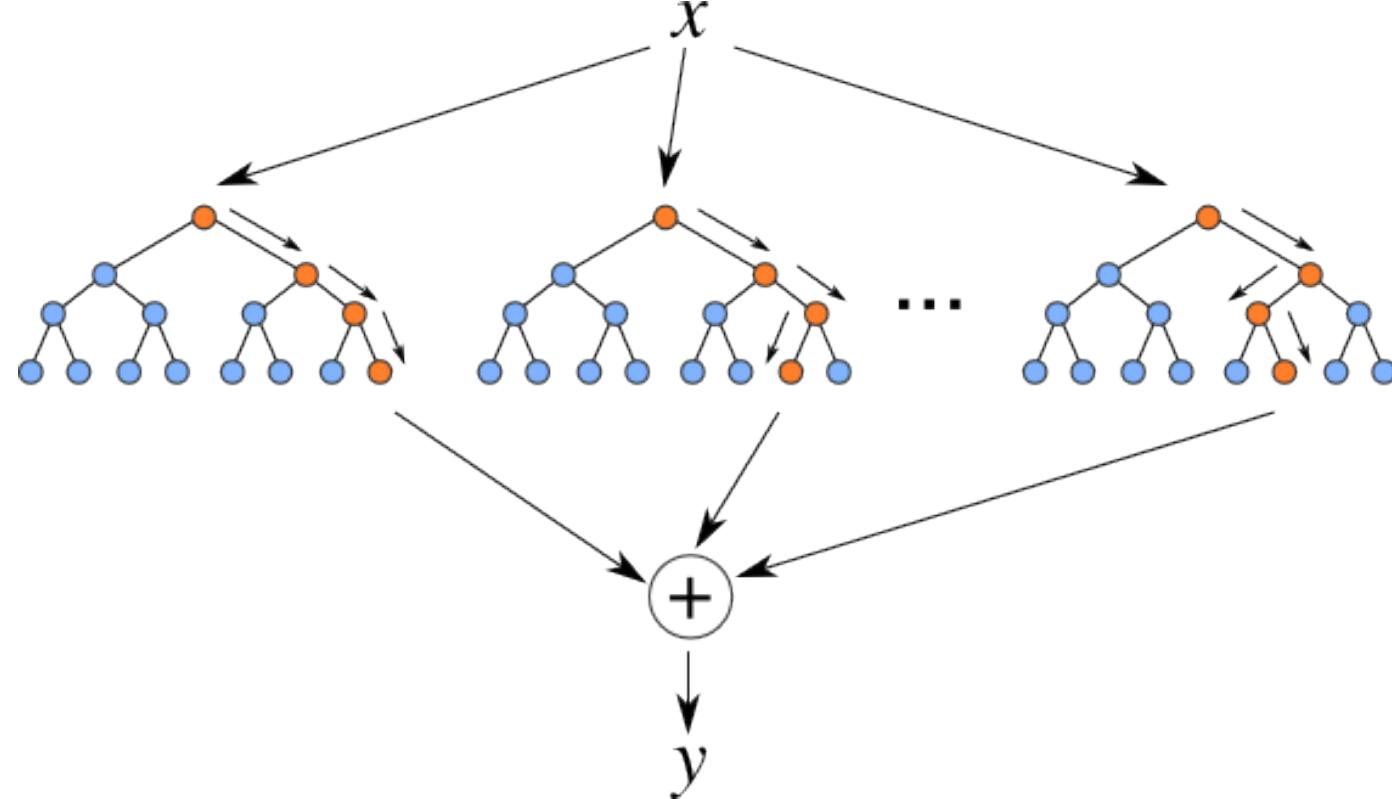


# Преимущества и недостатки бустинга

- Позволяет очень точно приблизить восстанавливаемую функцию или разделяющую поверхность классов
- Плохо интерпретируем
- Композиции могут содержать десятки тысяч базовых моделей и долго обучаться
- Переобучение на выбросах при избыточном количестве классификаторов

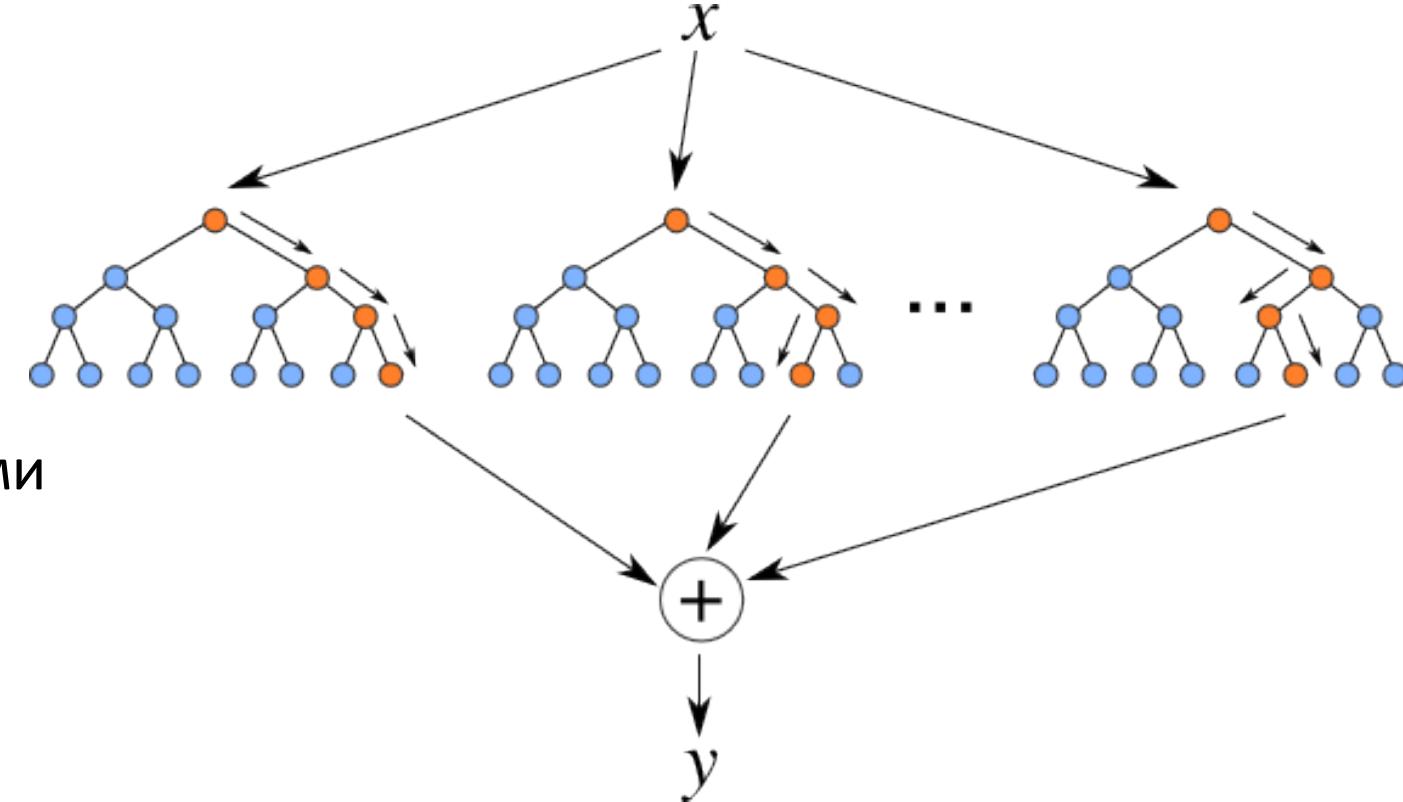
# Random Forest

1. Генерируем  $M$  выборок на основе имеющейся (по схеме выбора с возвращением)



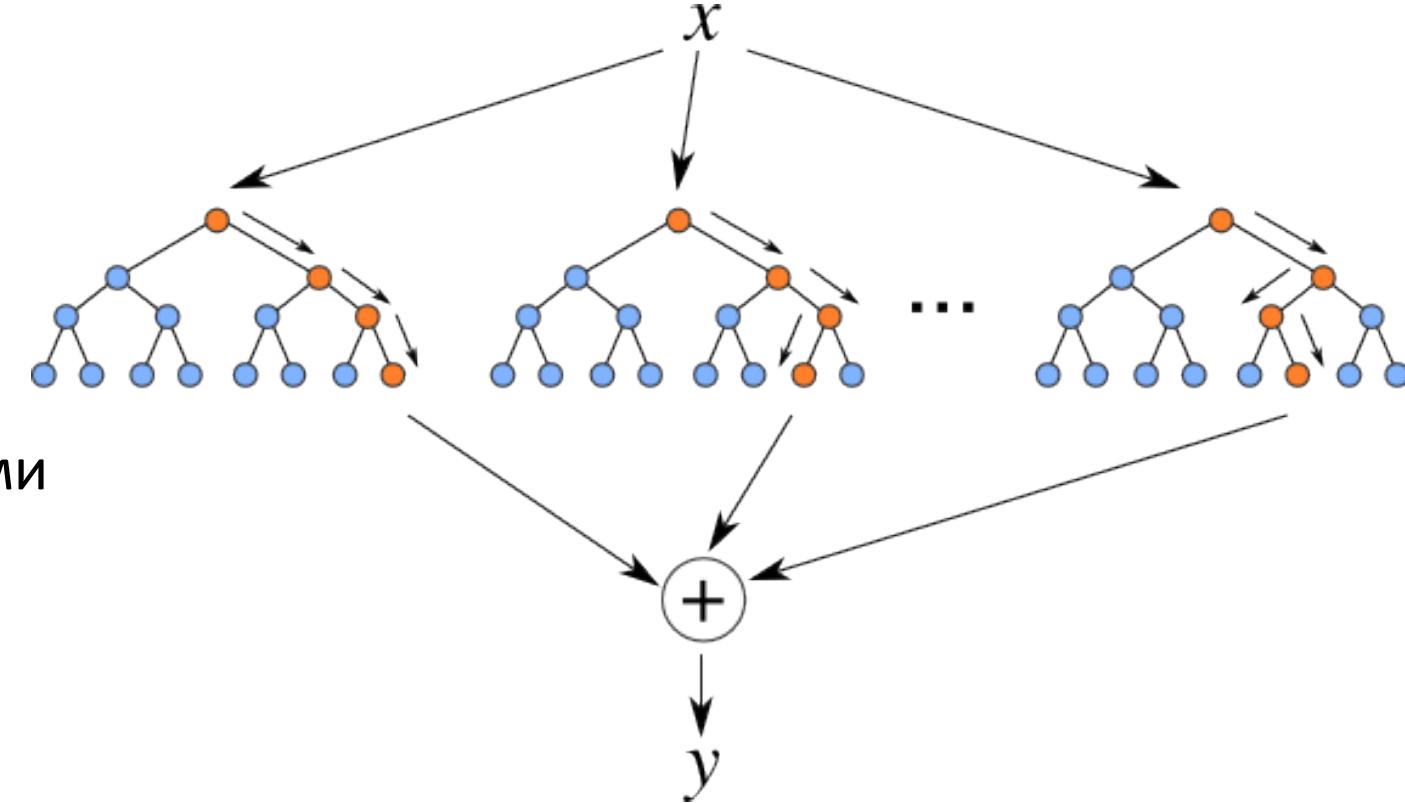
# Random Forest

1. Генерируем  $M$  выборок на основе имеющейся (по схеме выбора с возвращением)
2. Строим на них деревья с рандомизированными разбиениями в узлах: выбираем  $k$  случайных признаков и ищем наиболее информативное разбиение по ним



# Random Forest

1. Генерируем  $M$  выборок на основе имеющейся (по схеме выбора с возвращением)
2. Строим на них деревья с рандомизированными разбиениями в узлах: выбираем  $k$  случайных признаков и ищем наиболее информативное разбиение по ним
3. При прогнозе усредняем ответ всех деревьев

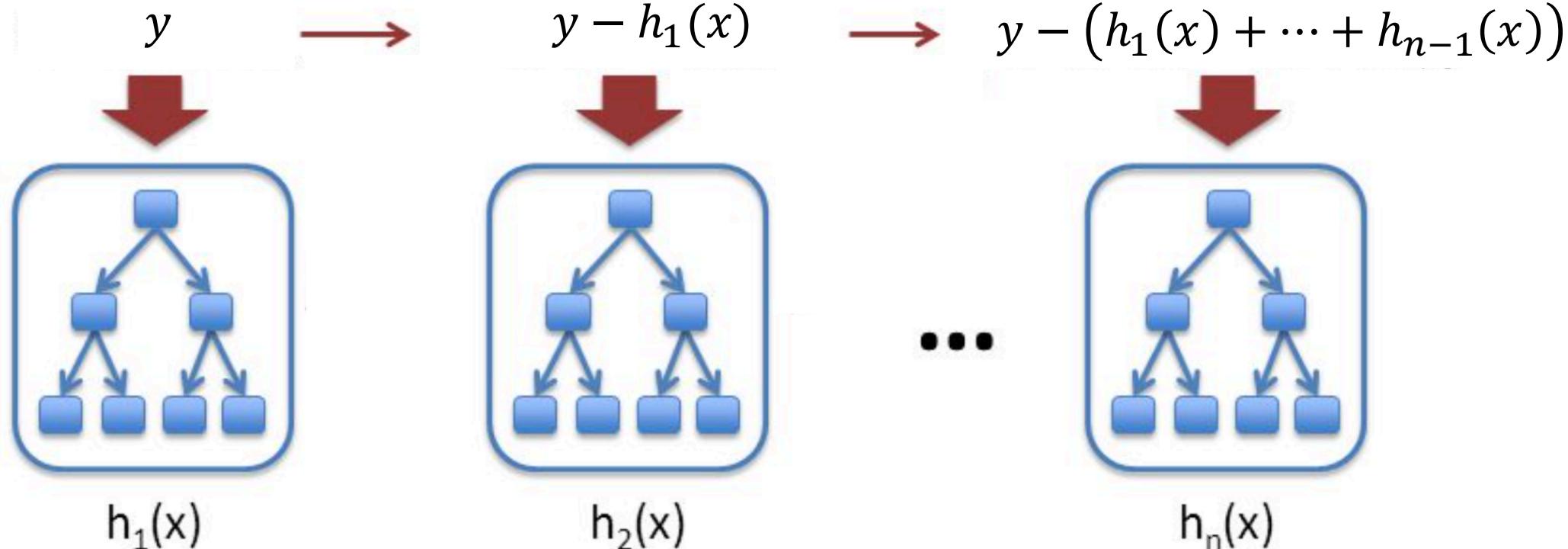


# Идея Gradient Boosted Decision Trees (GBDT)

$$h(x) = h_1(x) + \dots + h_n(x)$$

# Идея Gradient Boosted Decision Trees (GBDT)

$$h(x) = h_1(x) + \dots + h_n(x)$$



# Gradient Boosted Decision Trees

- Каждое новое дерево  $h_k(x)$  обучаем на ответы  $y_i - h_i$   
 $h_i$  - прогноз всей композиции на  $i$ -том объекте на предыдущей итерации
- Коэффициент  $\alpha_k$  перед новым деревом подбираем с помощью численной оптимизации ошибки

Где здесь градиент

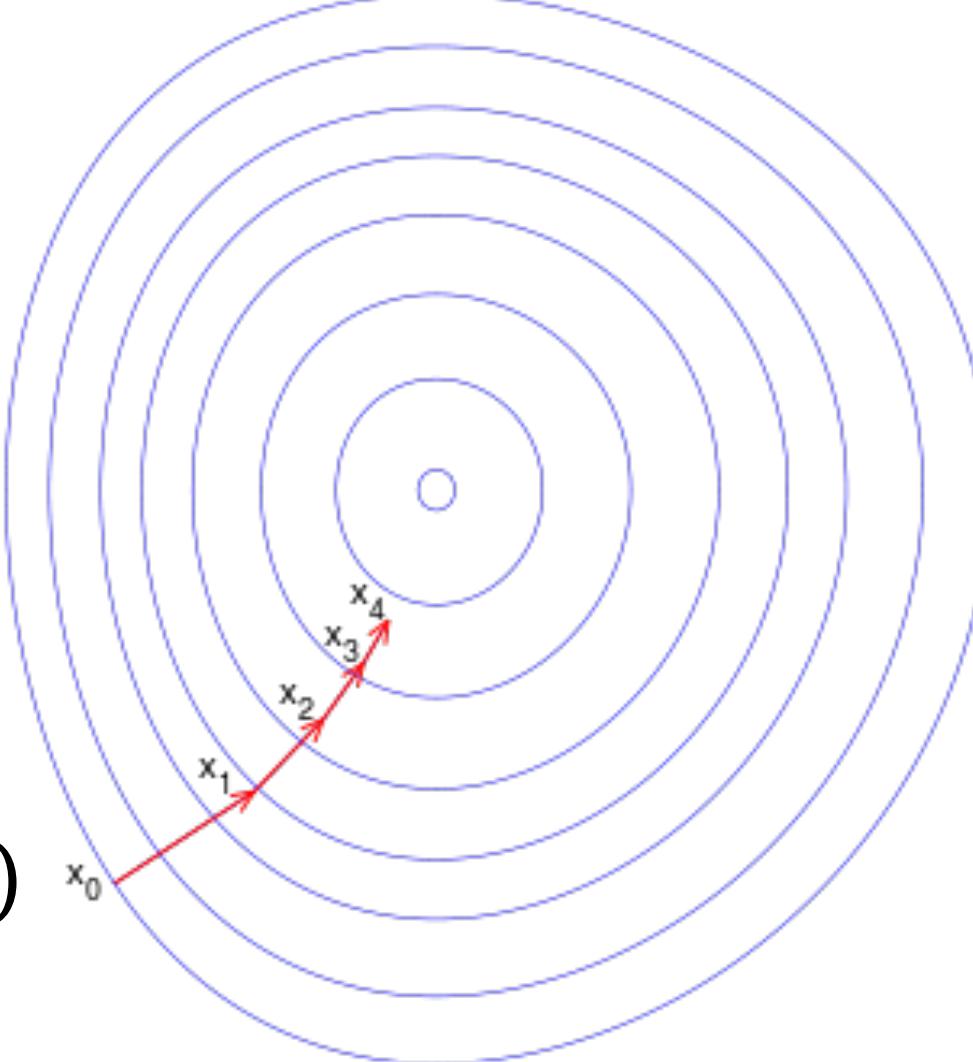
$$L(y_i, h(x_i)) = (y_i - h(x_i))^2$$

$$Q = \sum_{i=1}^N L(y_i, h(x_i)) = \sum_{i=1}^N (y_i - h(x_i))^2$$

## Напоминание: градиентный спуск

$$\nabla F(x_k) = \begin{pmatrix} \frac{\partial F}{\partial x_{k1}} \\ \vdots \\ \frac{\partial F}{\partial x_{kn}} \end{pmatrix}$$

$$x_{k+1} = x_k - \eta \nabla F(x_k)$$



Где здесь градиент

$$L(y_i, h(x_i)) = (y_i - h(x_i))^2$$

$$Q = \sum_{i=1}^N L(y_i, h(x_i)) = \sum_{i=1}^N (y_i - h(x_i))^2$$

$$\frac{\partial Q}{\partial h_i} = \sum_{i=1}^N \frac{\partial}{\partial h_i} (y_i - h_i)^2 = 2(y_i - h_i)$$

Где здесь градиент

$$\frac{\partial Q}{\partial h_i} = \sum_{i=1}^N \frac{\partial}{\partial h_i} (y_i - h_i)^2 = 2(y_i - h_i)$$

$$h(x_i) = \sum_{k=1}^n \alpha_k h_k(x_i)$$

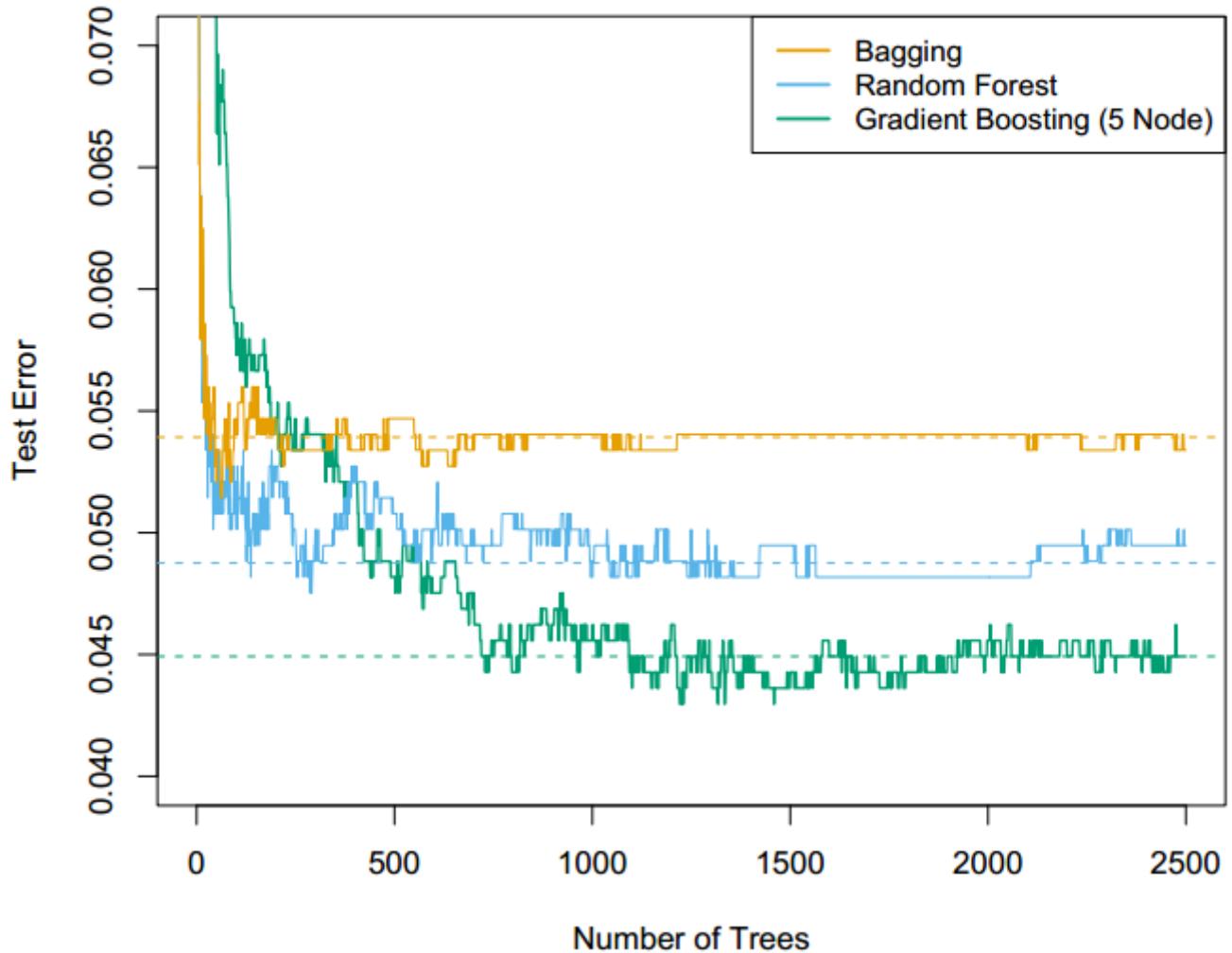
Новый  $h_k(x)$  будем обучать на ответы  $y_i - h_i$

# Gradient Boosted Decision Trees

- Каждое новое дерево  $h_k(x)$  обучаем на ответы  $y_i - h_i$   
 $h_i$  - прогноз всей композиции на  $i$ -том объекте на предыдущей итерации
- Коэффициент  $\alpha_k$  перед новым деревом подбираем с помощью численной оптимизации ошибки  $Q$

# GBDT и RF

Spam Data



# Распараллеливание

**Вопрос для обсуждения:**

Какой из ансамблей деревьев больше подходит для распараллеливания? Как это делать в одном и в другом случае?

## Библиотеки

- Scikit-learn:
  - `sklearn.ensemble.RandomForestClassifier`
  - `sklearn.ensemble.RandomForestRegressor`
- XGBoost
- LightGBM
- H2O

# Итог

1. Общие методы построения ансамблей
2. Random Forest
3. Gradient Boosted Decision Trees (GBDT)
4. Библиотеки

Сегодня на лекции мы обсудили:

- I. Линейные модели
- II. Решающие деревья
- III. Ансамбли решающих деревьев

На следующей лекции:  
методы unsupervised learning