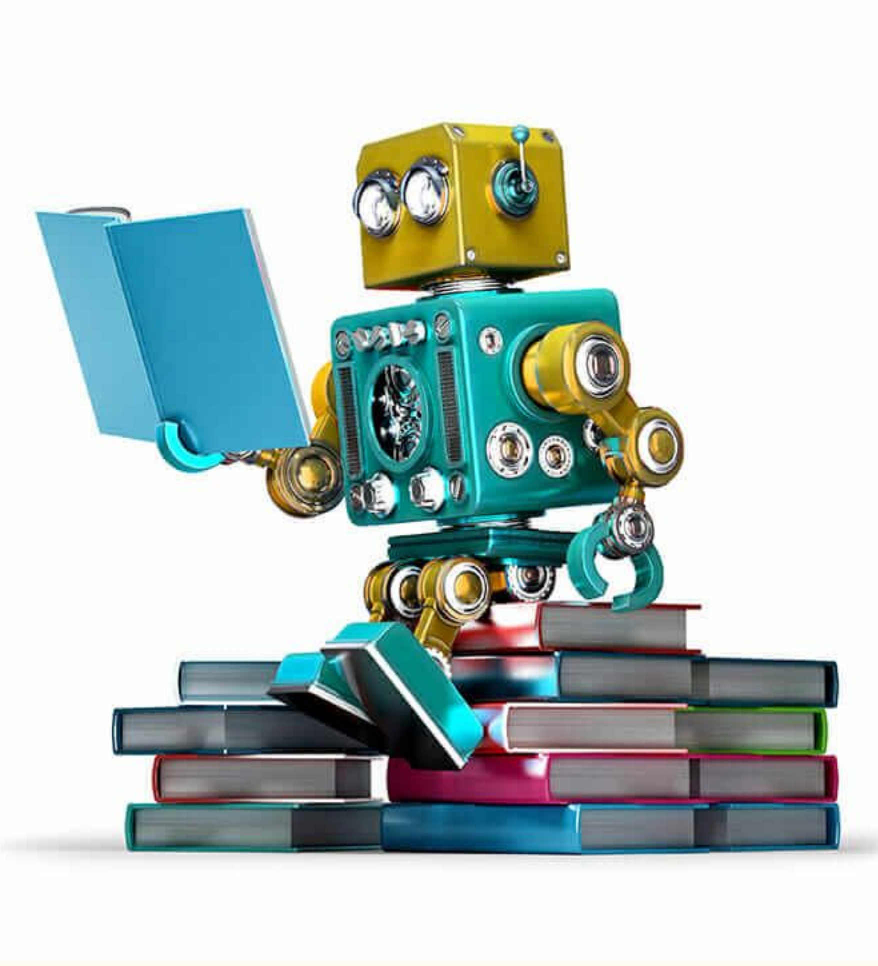
Модели, ансамбли моделей

Татьяна Гайнцева
МФТИ, ШАД, DLSchool

@atmyre

План

1. Зоопарк моделей
    a. Наивный байес
    b. Линейные модели
        i. Линейная регрессия
        ii. SVM
        iii. Логистическая регрессия
    c. Логические алгоритмы классификации
        i. Закономерность и информативность
        ii. Решающие списки
        iii. Решающие деревья
2. Композиции алгоритмов
    a. Bootstrap, Bagging
    b. Random Forest
    c. Gradient Boosting



Модели

Градиентный спуск

Градиентный спуск

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket
1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Thayer)	female	38	1	0	PC 17599
3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 310128

Пусть ваша модель -- некоторая **гладкая** функция от входных данных с некоторыми параметрами

Например, такая:

$$Y = \sigma(w_1^2 \cdot Pclass + (w_2 - 3) \cdot Age)$$

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

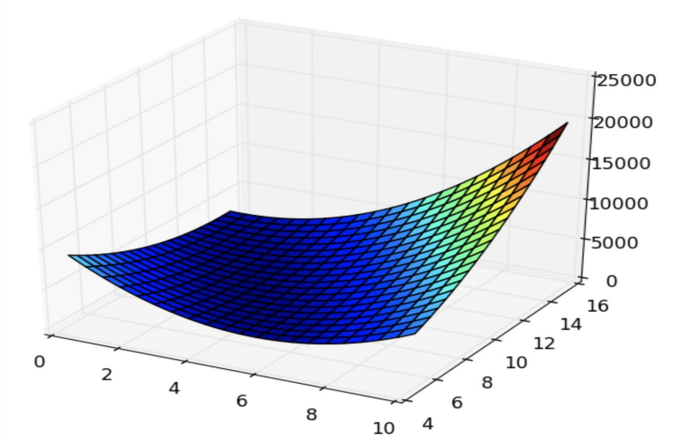
Как найти оптимальные параметры w?

Градиентный спуск

$$Y = \sigma(w_1^2 \cdot Pclass + (w_2 - 3) \cdot Age)$$

Зададим функцию потерь:

$$L = (Y - Y_{true})^2 = (\sigma(w_1^2 \cdot Pclass + (w_2 - 3) \cdot Age) - Y_{true})^2$$



Как найти минимум?

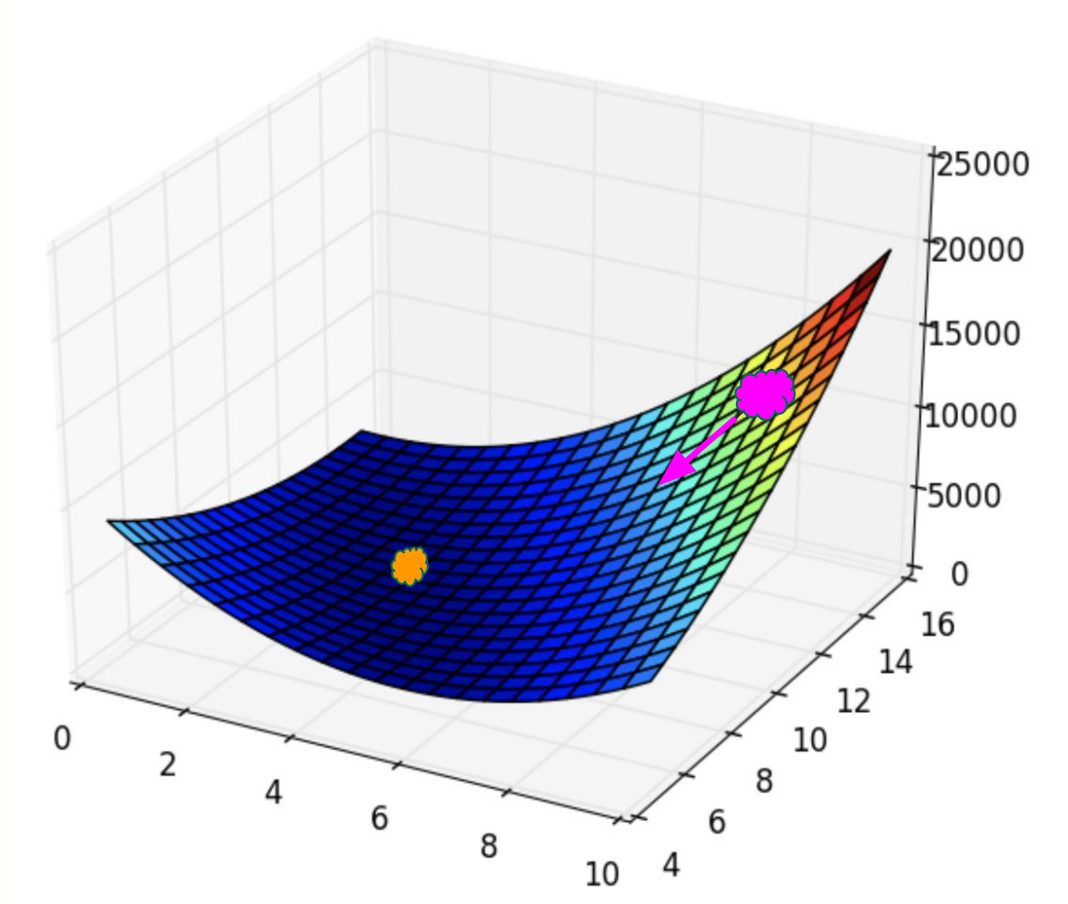
Градиентный спуск

Идея:

1. Возьмем рандомную точку w = (8, 14)

2. Поймем, в каком направлении из нее нужно двигаться, чтобы прийти к минимуму функции потерь

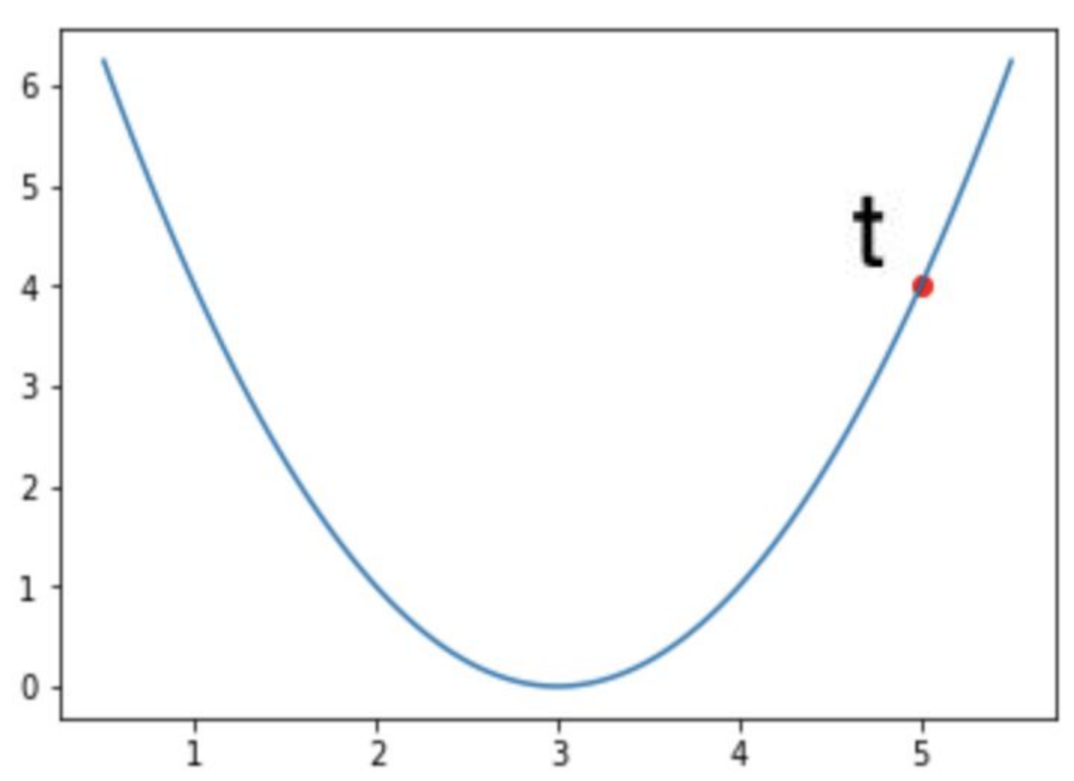
Т.е. найдем направление наискорейшего убывания функции потерь



Градиентный спуск

$$y = (x-3)^2$$

$$\frac{dy}{dx} = 2(x-3)$$

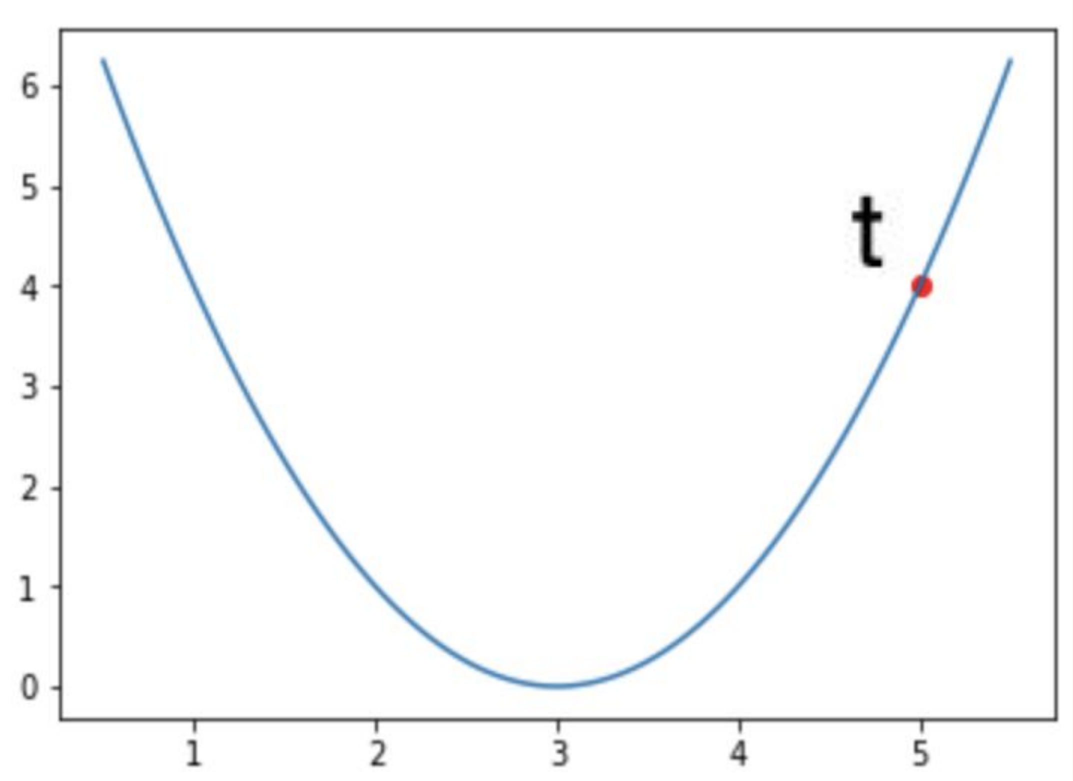


Градиентный спуск

$$y = (x - 3)^2$$

$$t = (5, 4)$$

$$\frac{dy}{dx} = 2(x - 3)$$



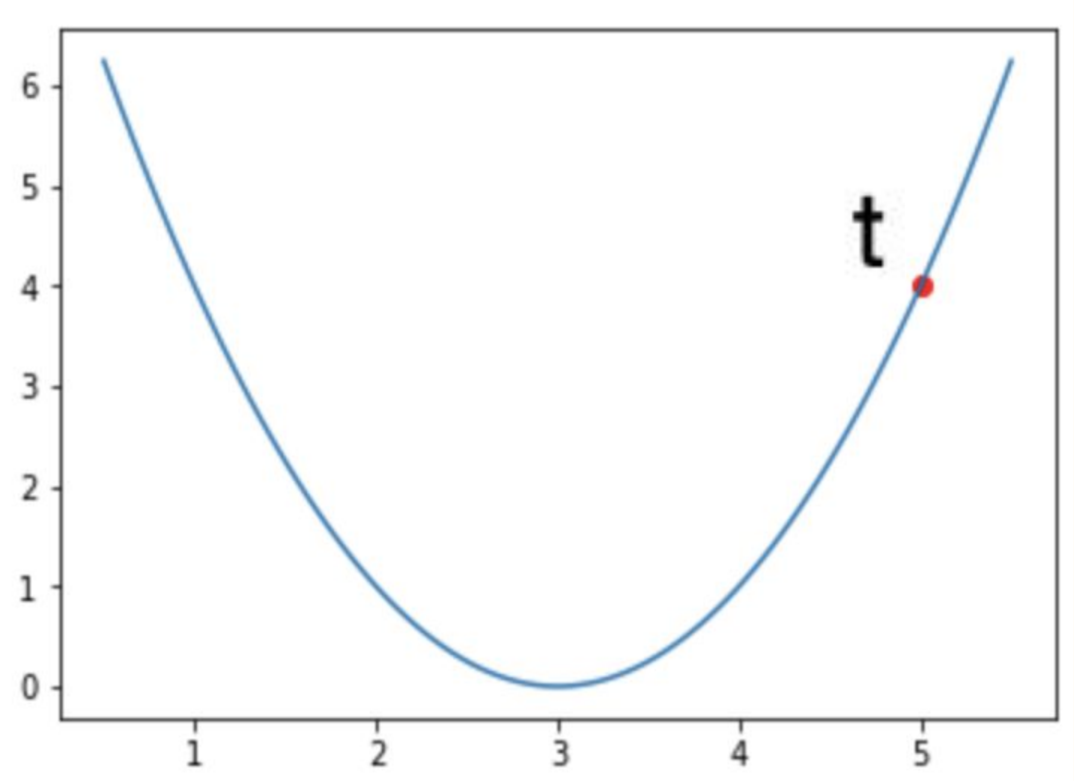
Градиентный спуск

$$y = (x-3)^2$$

$$t = (5, 4)$$

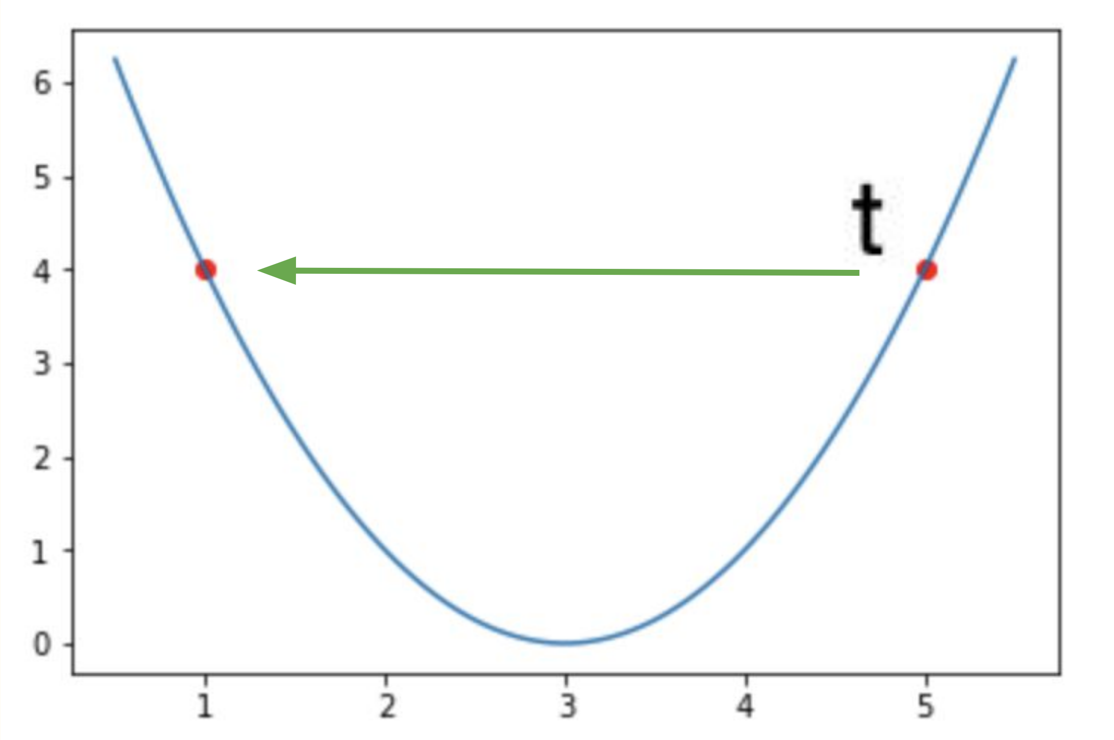
$$\frac{dy}{dx} = 2(x-3)$$

$$\frac{dy}{dx}(t) = 2(5-3) = 4$$



Градиентный спуск

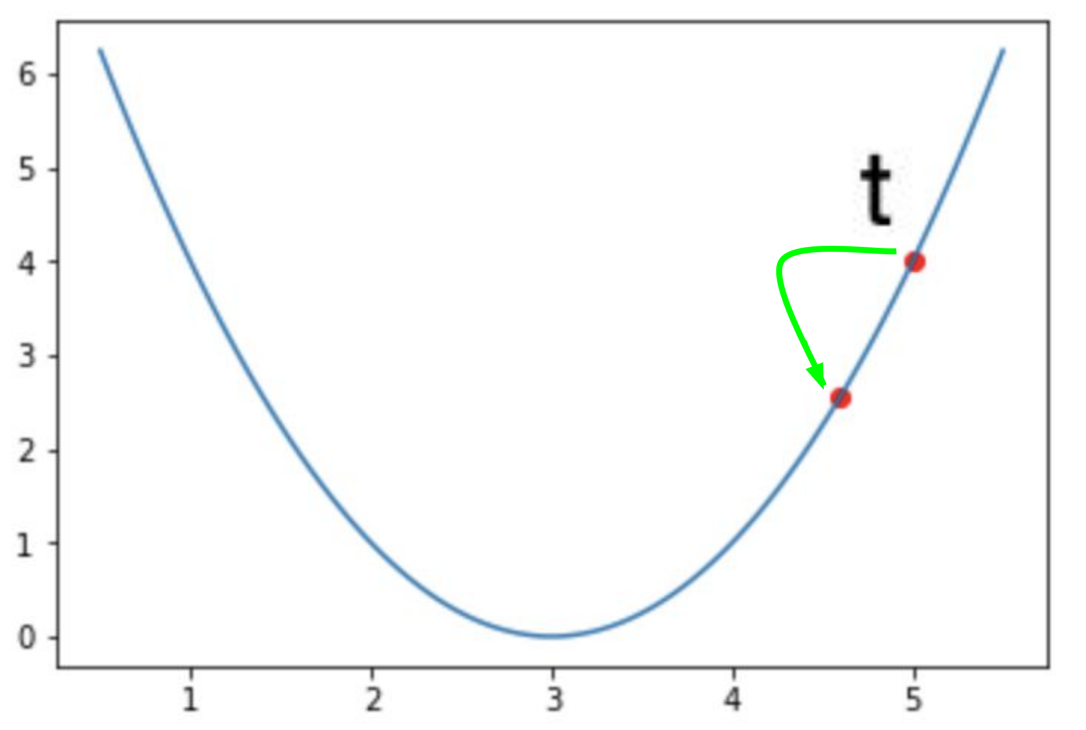
$$t - \frac{dy}{dx}(t) = 1$$



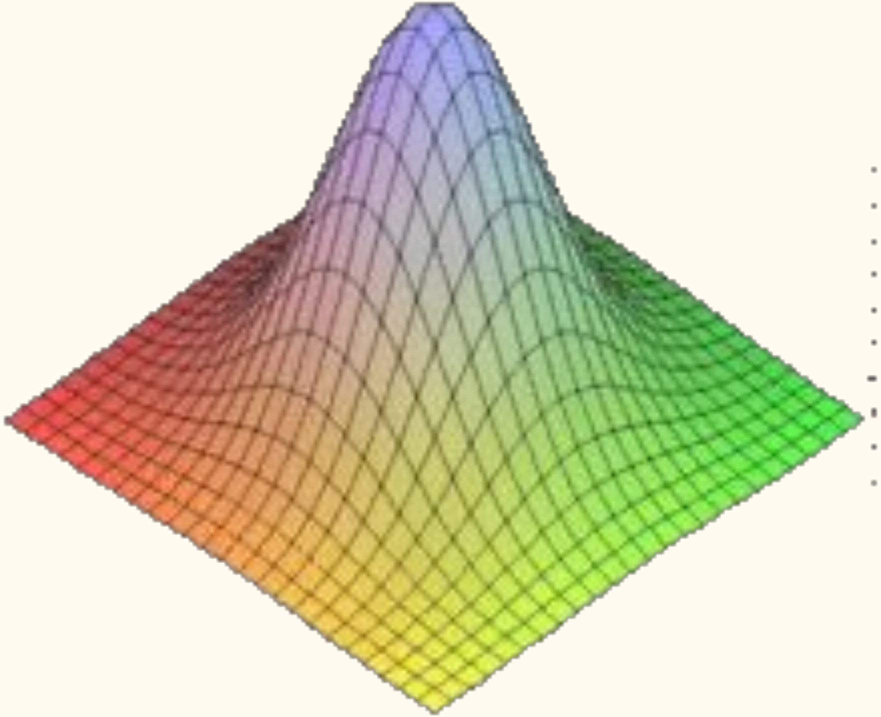
Градиентный спуск

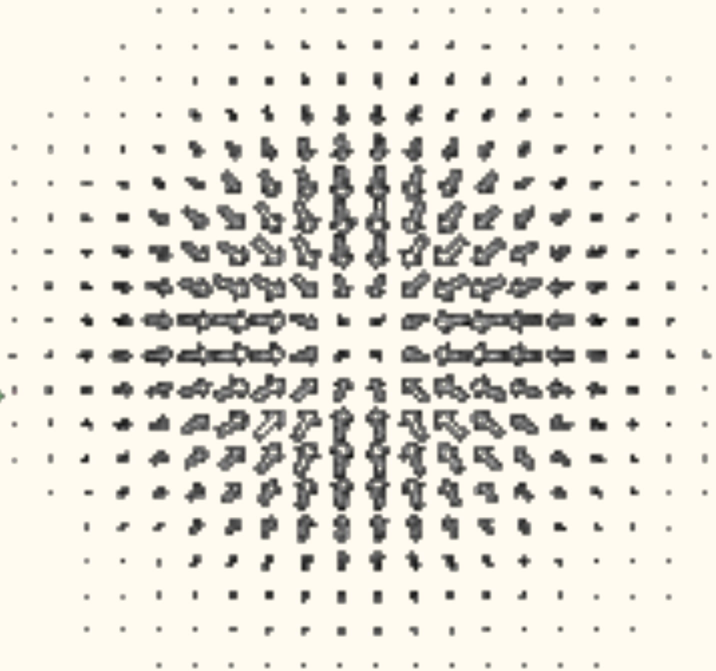
$$\alpha = 0.1$$

$$t - \alpha \frac{dy}{dx}(t) = 4.6$$



Градиентный спуск





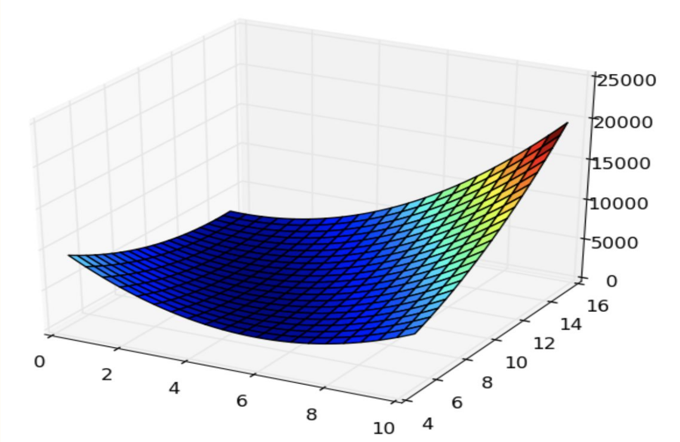
Градиент

Градиентный спуск

$$Y = \sigma(w_1^2 \cdot Pclass + (w_2 - 3) \cdot Age)$$

Зададим функцию потерь:

$$L = (Y - Y_{true})^2 = (\sigma(w_1^2 \cdot Pclass + (w_2 - 3) \cdot Age) - Y_{true})^2$$



Как найти минимум?

Градиентный спуск

1. Возьмем рандомную точку w = (8, 14)

2. Поймем, в каком направлении из нее нужно двигаться, чтобы прийти к минимуму: **возьмем градиент функции потерь по w:**

$$dw = (dw_1, dw_2) = \left(\frac{dL}{dw_1}, \frac{dL}{dw_1} \right)$$

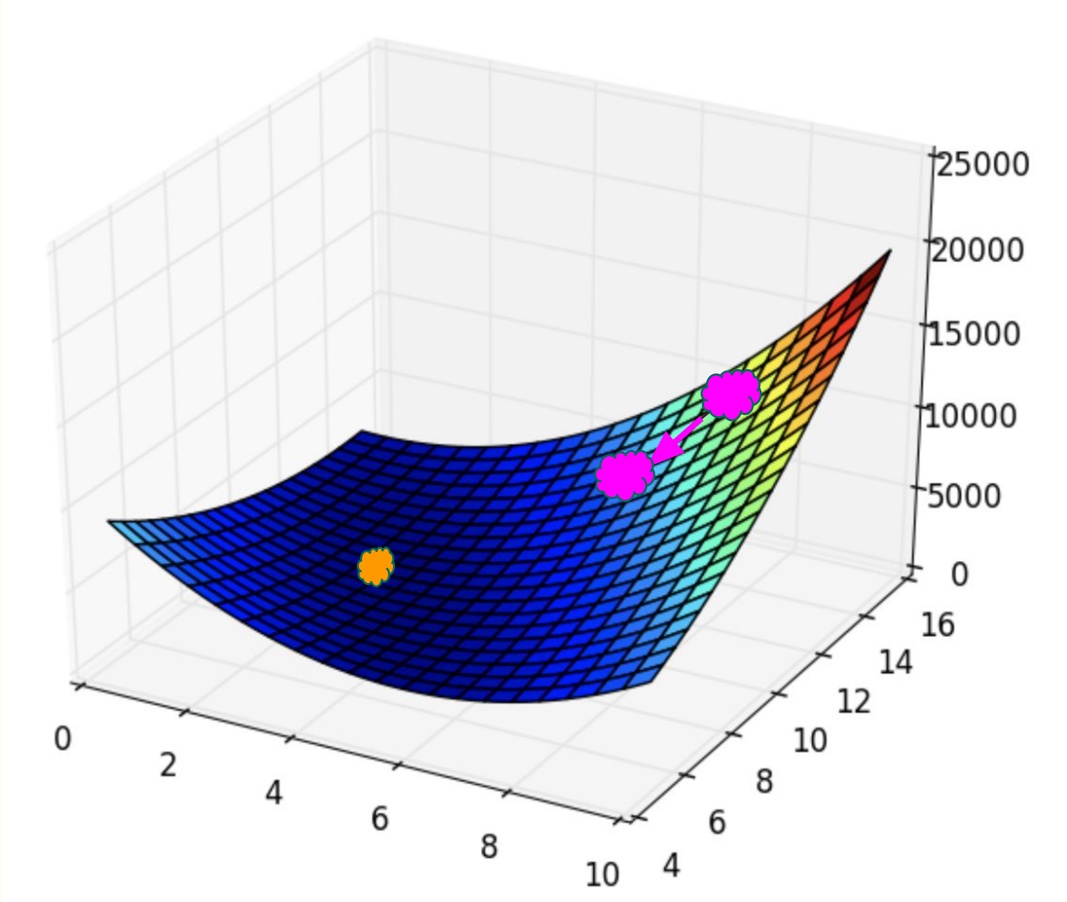
$$\frac{dL}{dw_1} = \frac{d[(Y - Y_{true})^2]}{dw_1} = \frac{d[(Y - Y_{true})^2]}{dY} \cdot \frac{dY}{d\sigma} \cdot \frac{d\sigma}{dw_1} = 3$$

Градиентный спуск

1. Возьмем рандомную точку w = (8, 14)

2. Поймем, в каком направлении из нее нужно двигаться, чтобы прийти к минимуму: **возьмем градиент функции потерь по w: dw = (3, 4)**

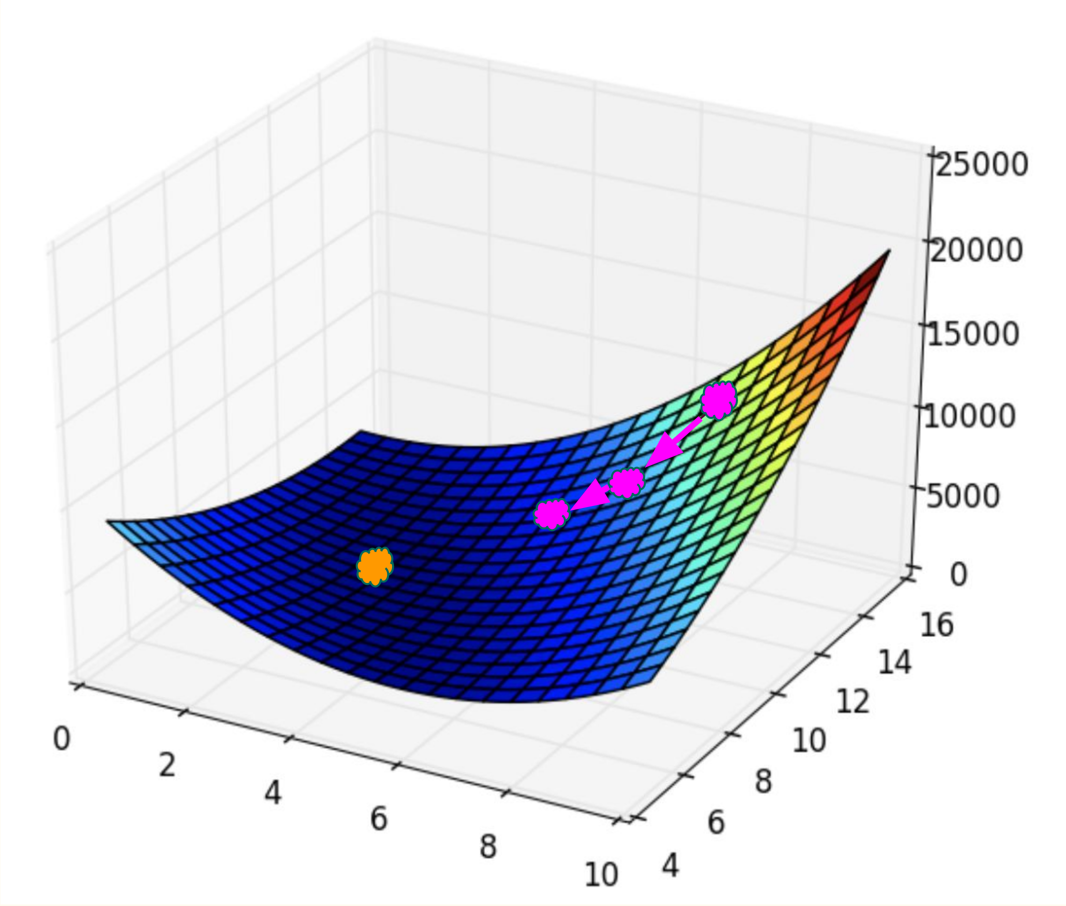
3. Обновим вес:
$w = w - \alpha \cdot dw$
$w = (8, 14) - 0.1 \cdot (3, 4) = 7.7, 13.6$



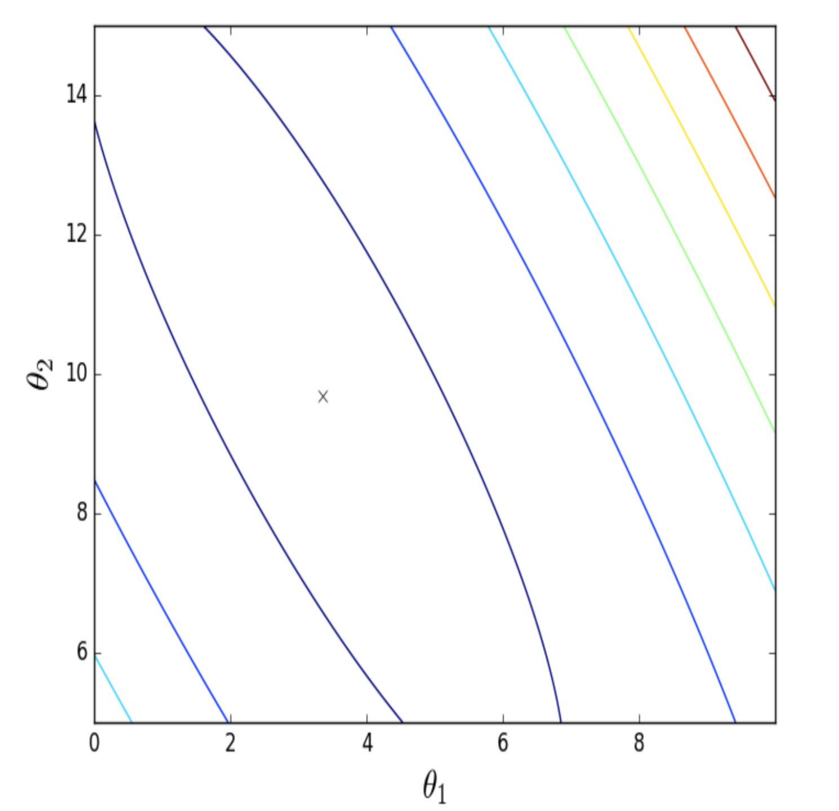
Градиентный спуск

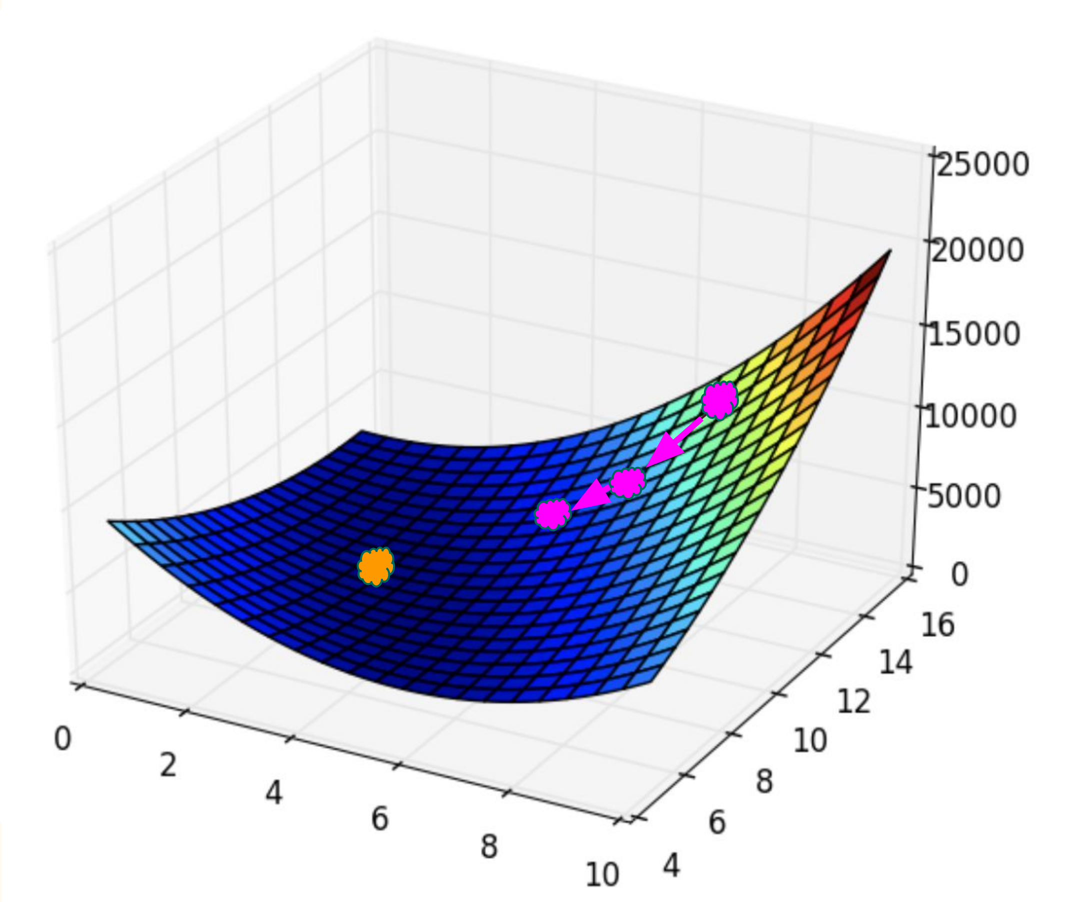
Теперь мы стоим в новой точке. Повторим процесс из нее, получим еще новую точку.

Повторяем, пока не дойдем до почти минимума или не надоест



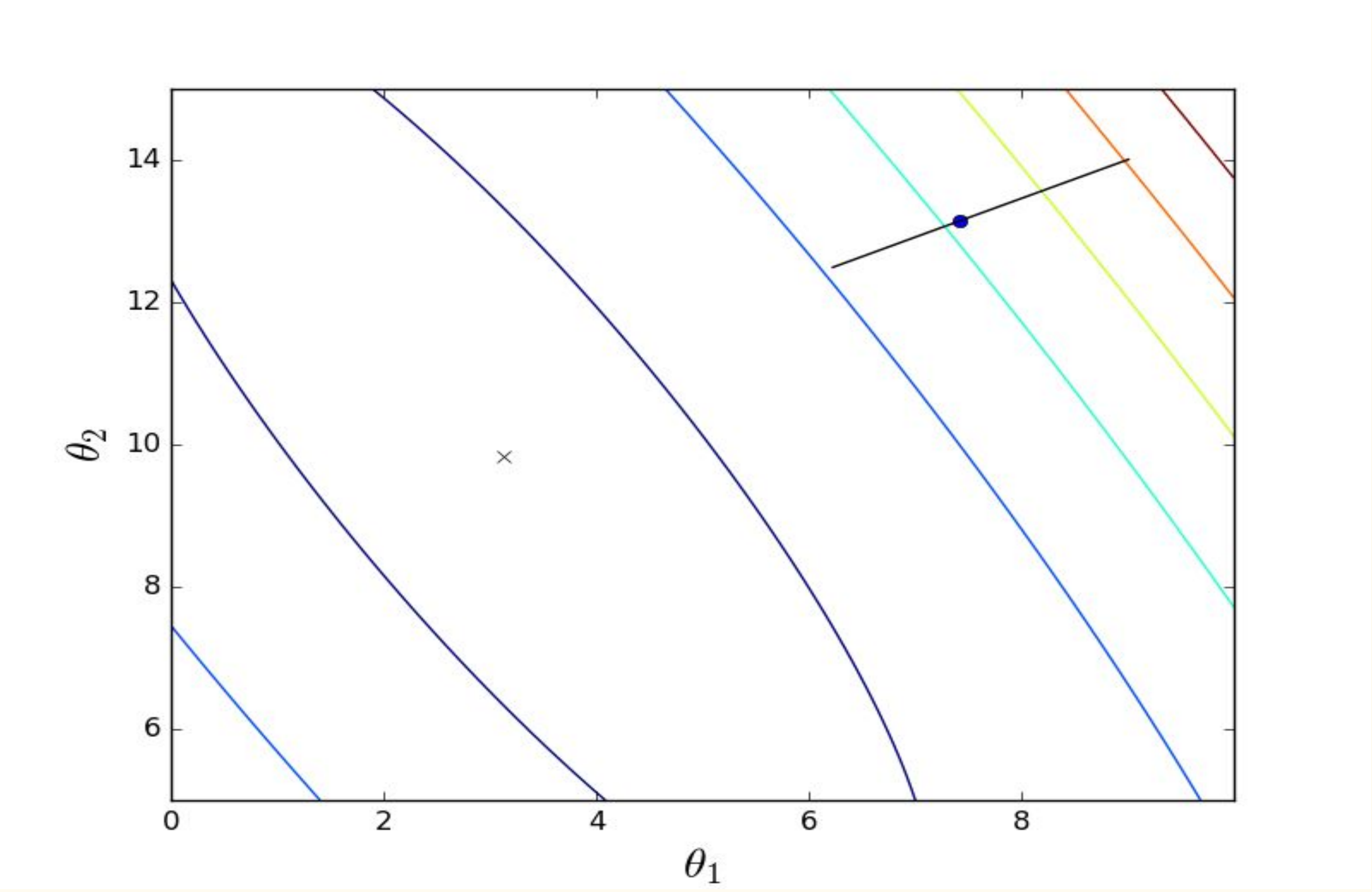
Градиентный спуск



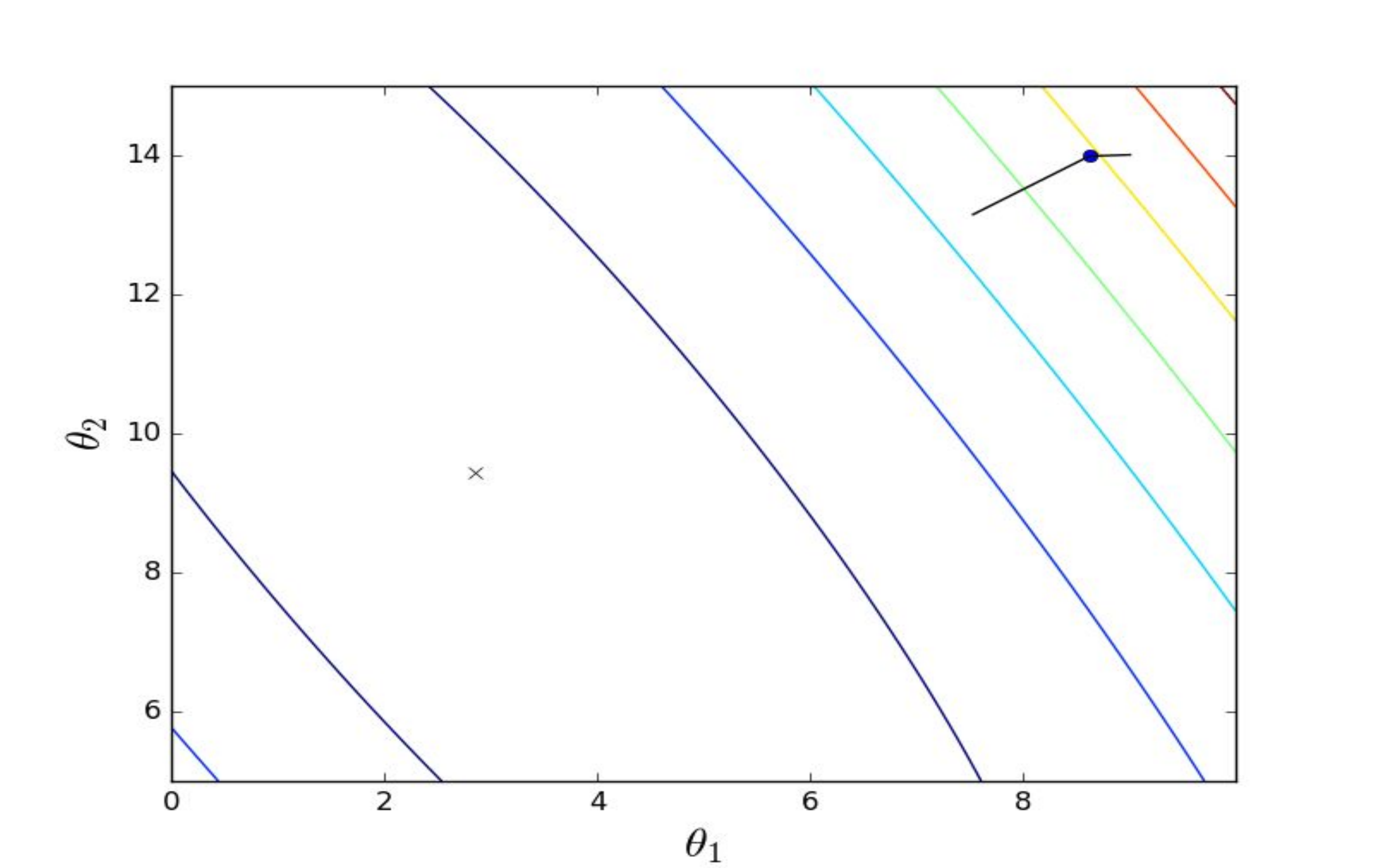


Вид сверху

Градиентный спуск



Стохастический градиентный спуск



Градиентный спуск

Пусть наша модель -- гладкая функция с параметрами w:

$$Y = \sigma(w_1^2 \cdot Pclass + (w_2 - 3) \cdot Age)$$

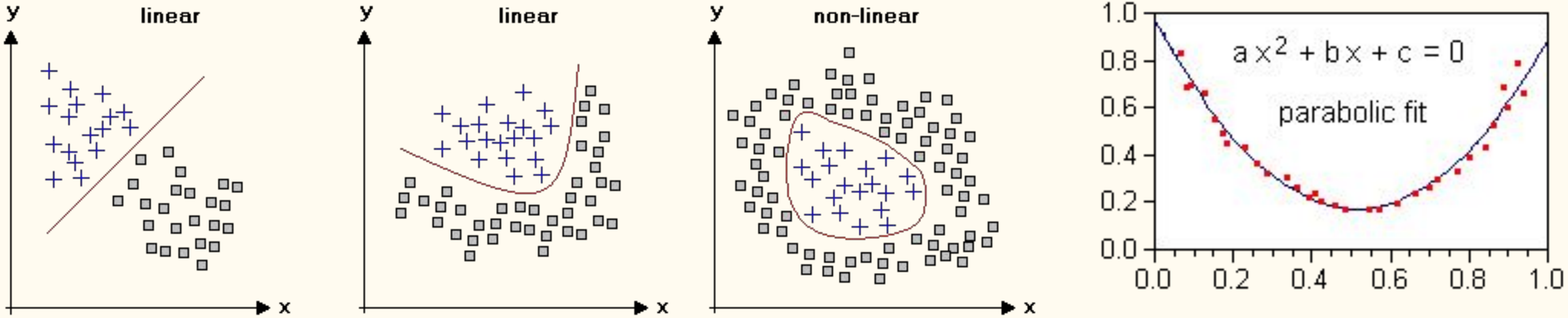
Тогда мы можем найти оптимальные веса, для которых функция потерь на выборке минимальна, с помощью градиентного спуска:

1. Выбираем рандомное начальное приближение w = (8, 14)
2. Повторяем, пока не сойдется:
    a. Вычислить градиент dw по выборке
    b. w = w - α·dw

Линейные модели

Линейная регрессия

Линейная модель



$$Y_i = \beta_0 + \beta_1 \phi_1(X_{i1}) + \cdots + \beta_p \phi_p(X_{ip}) + \varepsilon_i \qquad i = 1, \ldots, n$$

http://www.statistics4u.com

Линейная модель

$$Y_i = \beta_0 + \beta_1\phi_1(X_{i1}) + \cdots + \beta_p\phi_p(X_{ip}) + \varepsilon_i \qquad i = 1, \ldots, n$$

Максимизация правдоподобия выборки == минимизация следующей функции потрерь:

$$S = \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1\phi_1(X_{i1}) - \cdots - \beta_p\phi_p(X_{ip}))^2.$$

Линейная регрессия

https://habr.com/company/ods/blog/323890/

Линейная регрессия

Постановка задачи:

$$y = w_1 x_1 + w_2 x_2 + \cdots + w_d x_d + w_0$$

Функция потерь для линейной регрессии:

$$MSE = \frac{1}{N} \sum (y_i - (Wx_i + b))^2$$

Линейная регрессия

Постановка задачи:

$$Y = wX^T + b$$

$$(y_1, y_2, \dots, y_n) = (w_1, w_2, \dots, w_d)\begin{bmatrix} x_{11} & x_{21} & x_{31} & \dots & x_{n1} \\ x_{12} & x_{22} & x_{32} & \dots & x_{n2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{1d} & x_{2d} & x_{3d} & \dots & x_{nd} \end{bmatrix} + (b_1, b_2, \dots, b_n)$$

Линейная регрессия

Постановка задачи:

$$Y = wX^T$$

$$(y_1, y_2, \dots, y_n) = (w_1, w_2, \dots, w_d, b)\begin{bmatrix} x_{11} & x_{21} & x_{31} & \dots & x_{n1} \\ x_{12} & x_{22} & x_{32} & \dots & x_{n2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{1d} & x_{2d} & x_{3d} & \dots & x_{nd} \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix}$$

Линейная регрессия

Решение 1 :

$$W_{ans} = (W^T W)^{-1} W^T y$$

Решение 2 :

Градиентный спуск:

1. Инициализация W случайными значениями
2. Итеративно для всех сэмплов x из X_train:
3. Вычислить функцию потерь на x
4. Вычислить производные grad_w по каждому w из W
5. Обновить веса w = w - lr * grad_w
6. Полученное W = (w) искомое

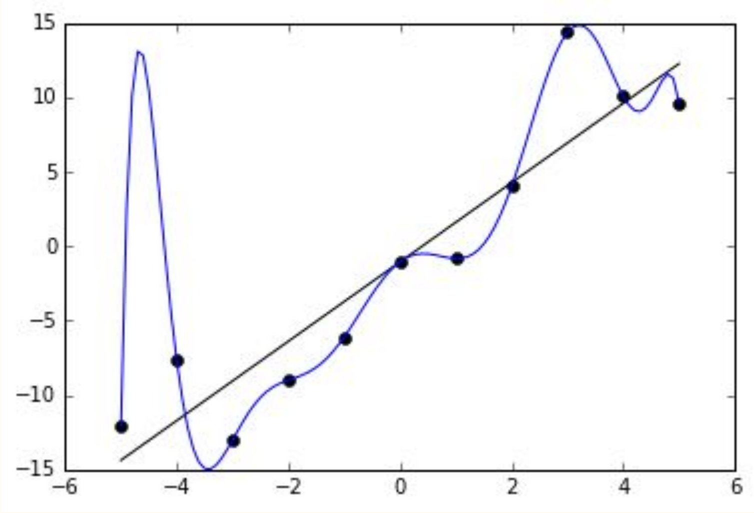
Функция потерь для линейной регрессии:

$$MSE = \frac{1}{N} \sum (y_i - (Wx_i + b))^2$$

Линейная регрессия

При решении задачи регрессии методом *градиентного спуска*:

Переобучение:



Как бороться?

Линейная регрессия

Регуляризация!

L1 (Lasso regression):

$$MSE = \frac{1}{N}\sum(y_i - (Wx_i + b))^2 + \lambda \sum |W_i|$$

L2 (Ridge regression):

$$MSE = \frac{1}{N}\sum(y_i - (Wx_i + b))^2 + \lambda \sum (W_i)^2$$

Регуляризация

L1 (Lasso regression):

$$MSE = \frac{1}{N}\sum(y_i - (Wx_i + b))^2 + \lambda\sum|W_i|$$

$$\sum(y_i - (wx_i + b_i))^2 + \lambda\sum|w_i| \rightarrow min$$

$$\Updownarrow$$

$$\begin{cases} \sum(y_i - (wx_i + b_i))^2 \rightarrow min \\ \sum|w_i| \leqslant c \end{cases}$$

Регуляризация

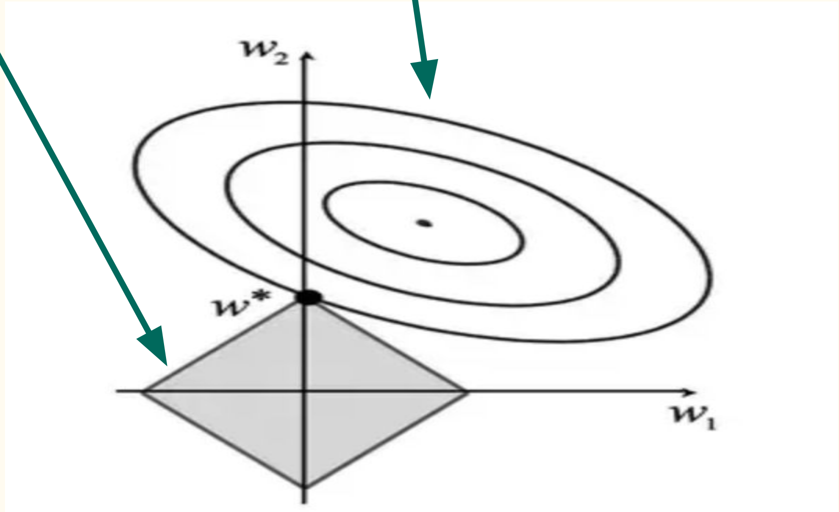
$$\begin{cases} \boxed{\sum (y_i - (wx_i + b_i))^2 \rightarrow min} \\ \boxed{\sum |w_i| \leqslant c} \end{cases}$$

Нарисуем проекции линий уровня первой формулы и области, ограниченной второй, на двумерную плоскость.

Регуляризация

$$\begin{cases} \boxed{\sum (y_i - (wx_i + b_i))^2 \to min} \\ \boxed{\sum |w_i| \leqslant c} \end{cases}$$

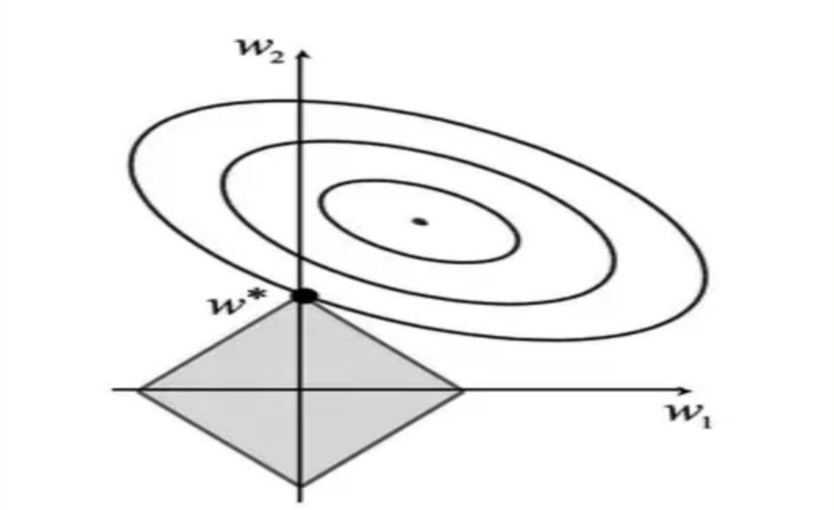
Нарисуем проекции линий уровня первой формулы и области, ограниченной второй, на двумерную плоскость.

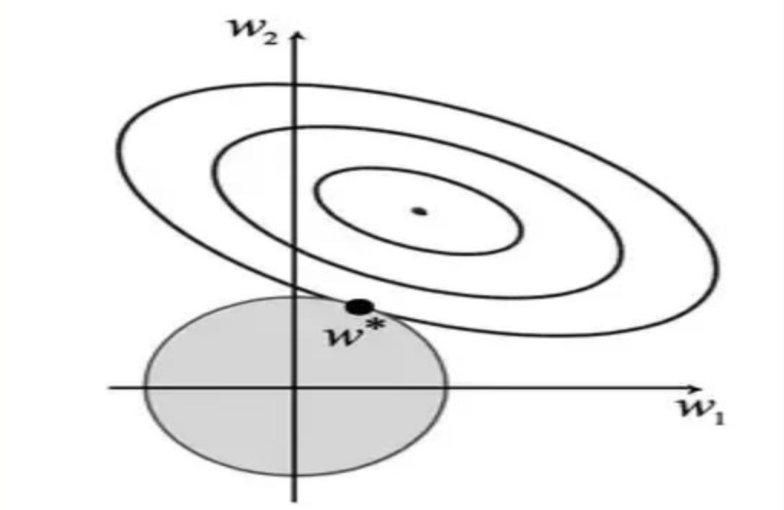


Регуляризация

$$\begin{cases} \sum (y_i - (wx_i + b_i))^2 \to min \\ \sum |w_i| \leqslant c \end{cases}$$

$$\begin{cases} \sum (y_i - (wx_i + b_i))^2 \to min \\ \sum (w_i)^2 \leqslant c \end{cases}$$





Логистическая регрессия

Логистическая регрессия

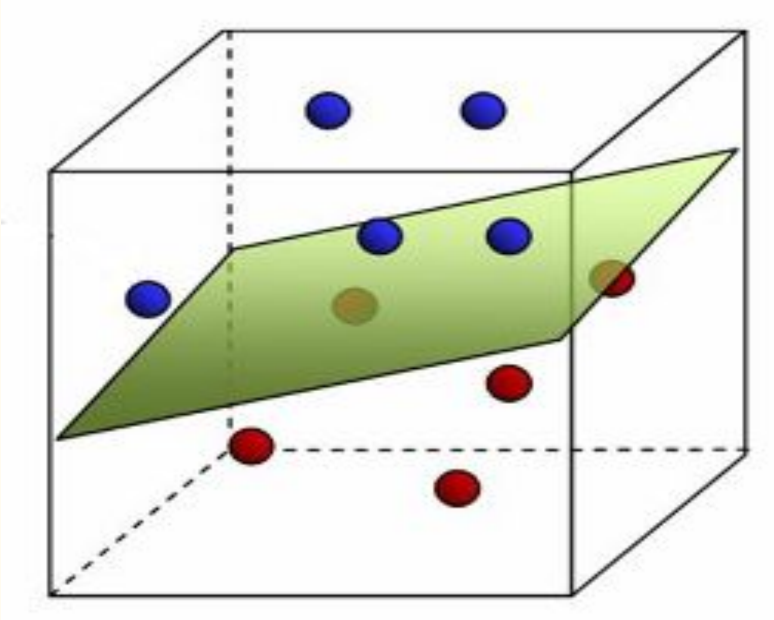
Постановка задачи:

Классификация на 2 класса: {-1, 1}

Модель: **Y = sign (wX)**

Для одного примера x можно записать:

y = sign(<w, x>)



Логистическая регрессия

Y = sign (wX)

$$(y_1, y_2, \dots, y_n) = sign\left((w_1, w_2, \dots, w_d, b)\begin{bmatrix} x_{11} & x_{21} & x_{31} & \dots & x_{n1} \\ x_{12} & x_{22} & x_{32} & \dots & x_{n2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{1d} & x_{2d} & x_{3d} & \dots & x_{nd} \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix}\right)$$

Логистическая регрессия

Y = sign (wX + b)

$$(y_1, y_2, \dots, y_n) = sign\left((w_1, w_2, \dots, w_d)\begin{bmatrix} x_{11} & x_{21} & x_{31} & \dots & x_{n1} \\ x_{12} & x_{22} & x_{32} & \dots & x_{n2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{1d} & x_{2d} & x_{3d} & \dots & x_{nd} \end{bmatrix} + (b_1, b_2, \dots, b_n)\right)$$

Логистическая регрессия

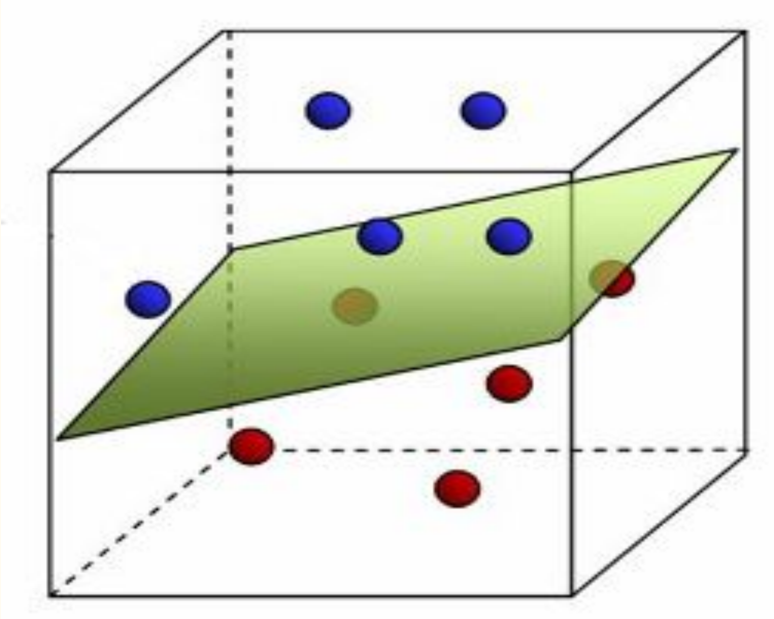
Постановка задачи:

Классификация на 2 класса: {-1, 1}

Модель: **Y = sign (wX + b)**

Для одного примера x можно записать:

y = sign(<w, x> + b)



Логистическая регрессия

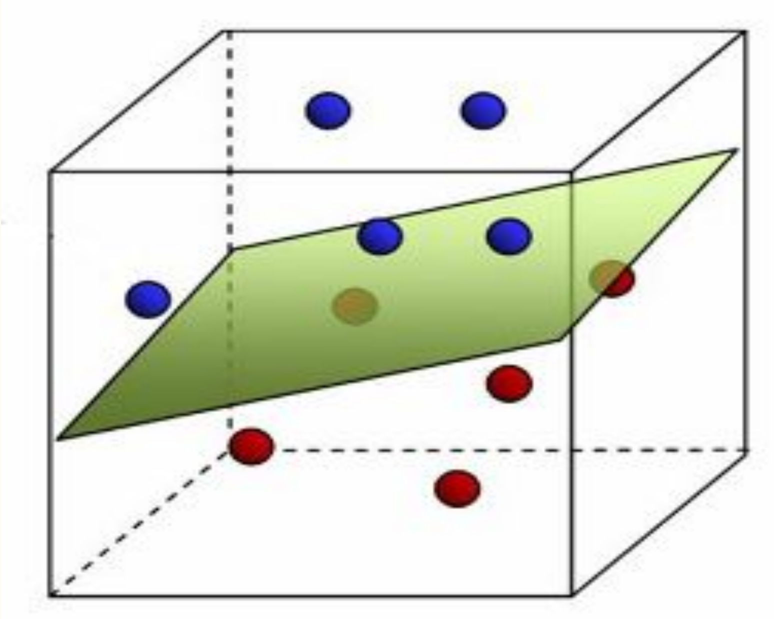
Постановка задачи:

Классификация на 2 класса: {-1, 1}

Модель: **Y = sign (WX - b)**

Для одного примера x можно записать:

y = sign(<w, x> - b)



Логистическая регрессия

Как задать функцию потерь?

Хочется оценивать количество ошибок классификатора, т.е. долю неверно угаданных ответов.

Логистическая регрессия

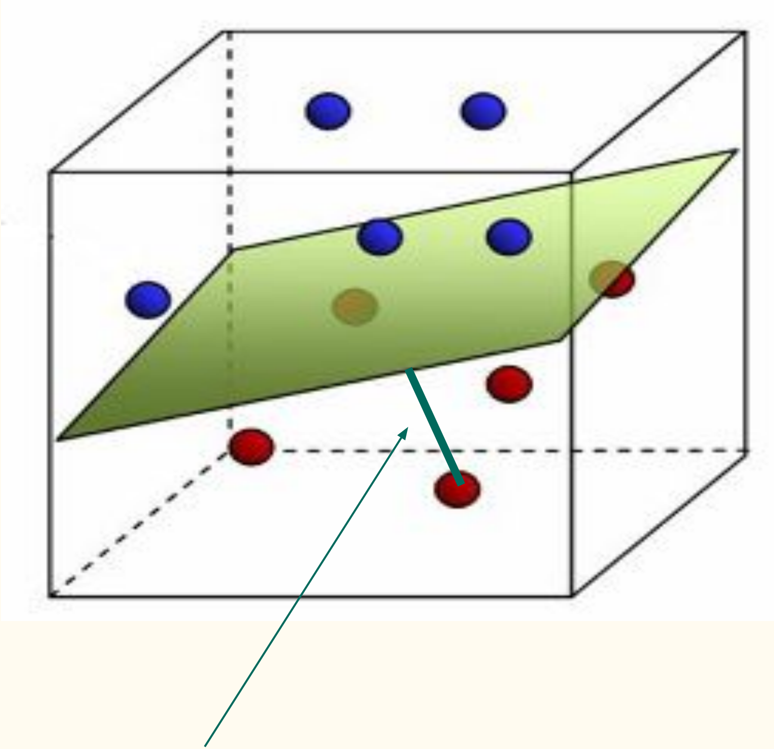
Введем понятие margin (отступ) на объекте:

$$M(x_i) = y_i < w, x_i >$$

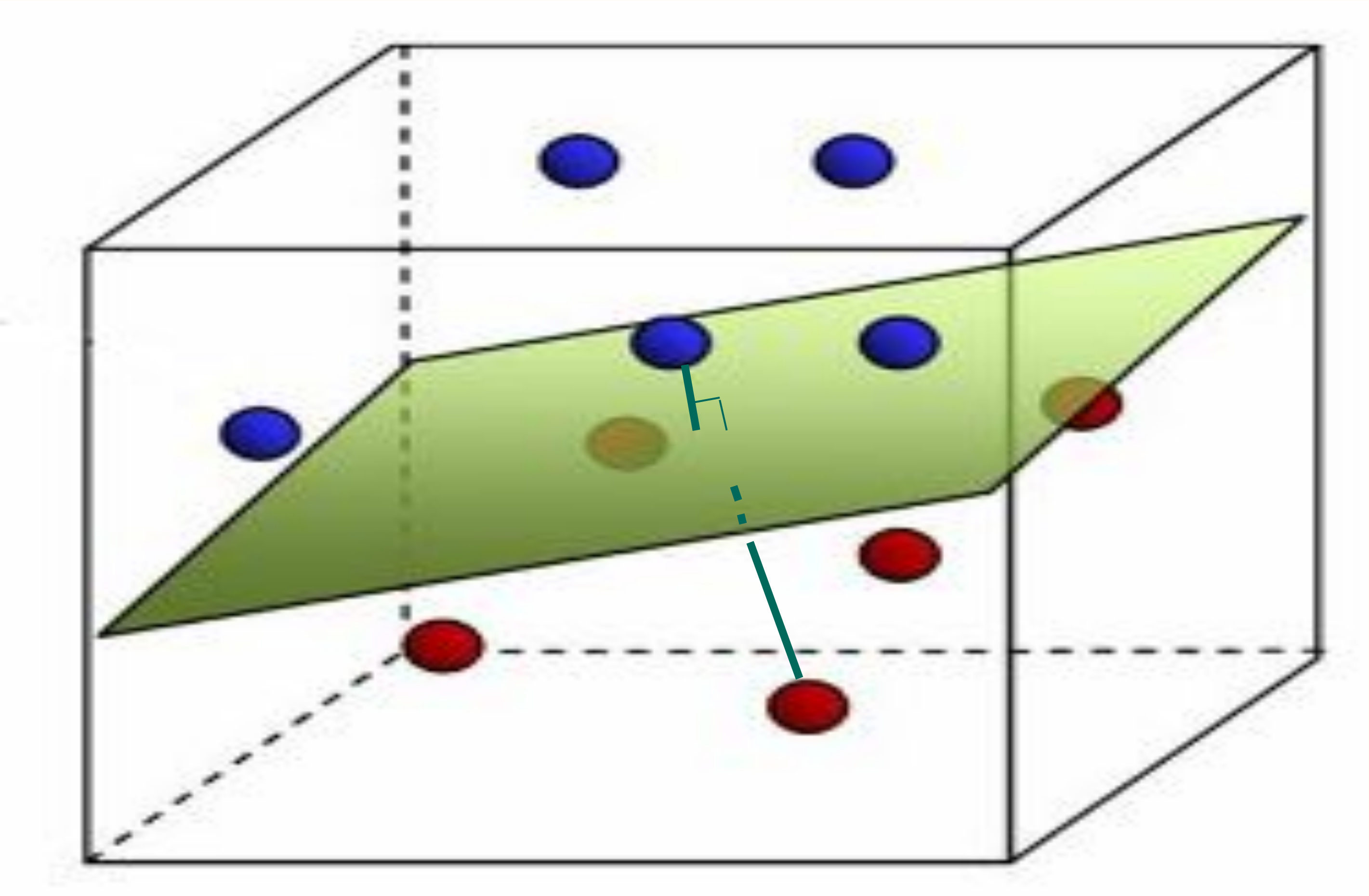
<0, если объект классифицирован неверно
>0, если объект классифицирован верно

Тогда хочется:

$$\sum [M(x_i) < 0] \to min$$



|margin|



Логистическая регрессия

Но эта функция не дифференцируема((9(

Заменим эту функцию ее оценкой сверху:

$$\sum [M(x_i) < 0] \to min$$

$$[M(x_i) < 0] \leqslant log_2(1 + e^{-M(x_i)}) = log_2(1 + exp\{-y_i < w, x_i >\})$$

Логистическая регрессия

Функция потерь:

$$L(x_i, y_i) = log_2(1 + exp\{-y_i < w, x_i >\}) \to min$$

С помощью градиентного спуска находим такие w, чтобы L -> min

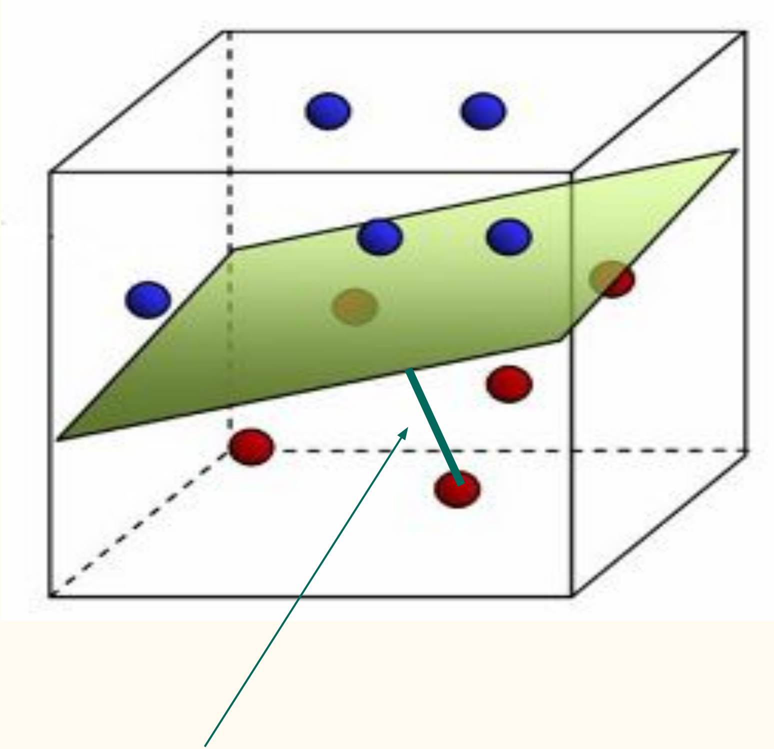
Логистическая регрессия

Ответ: Y = sign (WX)

Но! мы также можем посчитать вероятности:
Вспомним про margin:

$$M(x_i) = y_i < w, x_i >$$

<0, если объект классифицирован неверно
>0, если объект классифицирован верно



margin

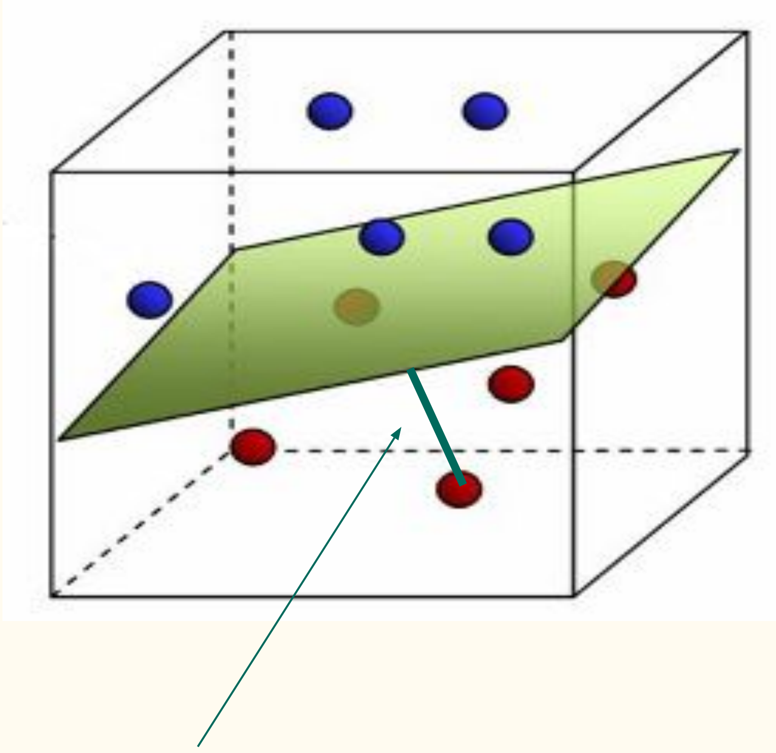
Логистическая регрессия

Ответ: Y = sign (WX)

Но! мы также можем посчитать вероятности:
Вспомним про margin:

$$M(x_i) = y_i < w, x_i >$$

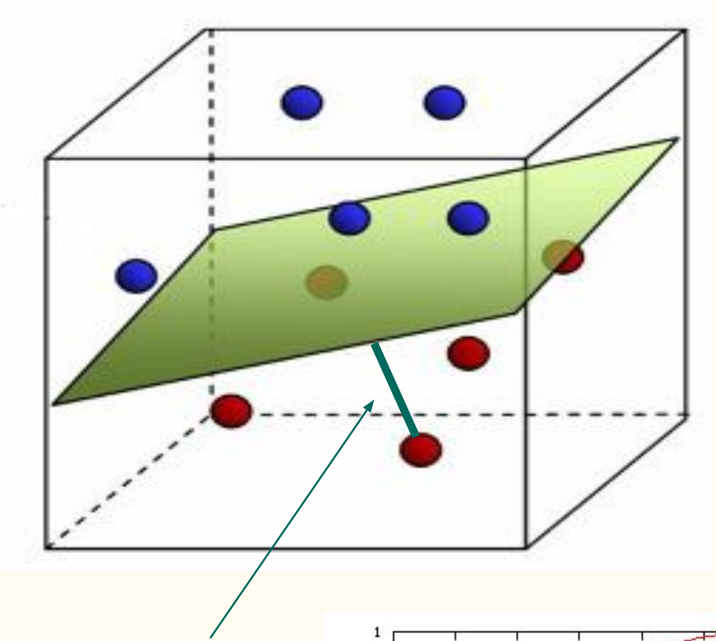
<0, если объект классифицирован неверно
>0, если объект классифицирован верно



margin

Чем больше margin, тем больше вероятность, что объект принадлежит классифицированному классу

Логистическая регрессия



margin

Ответ: Y = sign (WX)

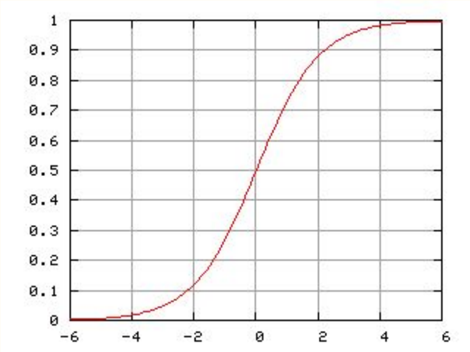
Margin:

$$M(x_i) = y_i < w, x_i >$$

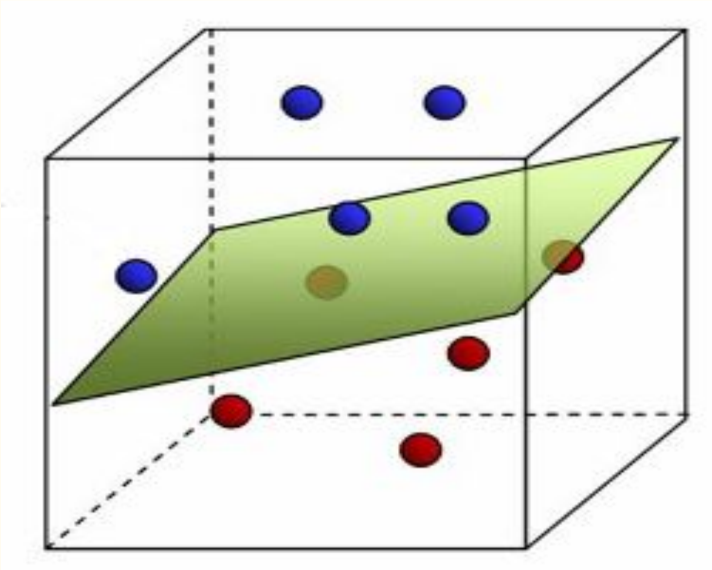
Вероятность принадлежности классу:

$$P = \sigma(y_i < w, x_i >)$$

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



Логистическая регрессия



Постановка задачи:

Классификация на 2 класса: {-1, 1}

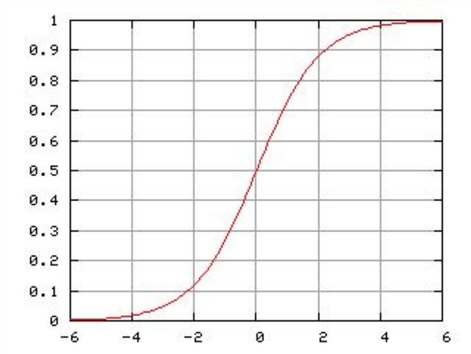
Модель: **Y = sign (WX)**

Для одного примера x можно записать:
y = sign(<w, x>)

Вероятность принадлежности классу:

$$P = \sigma(y_i < w, x_i >)$$

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



SVM

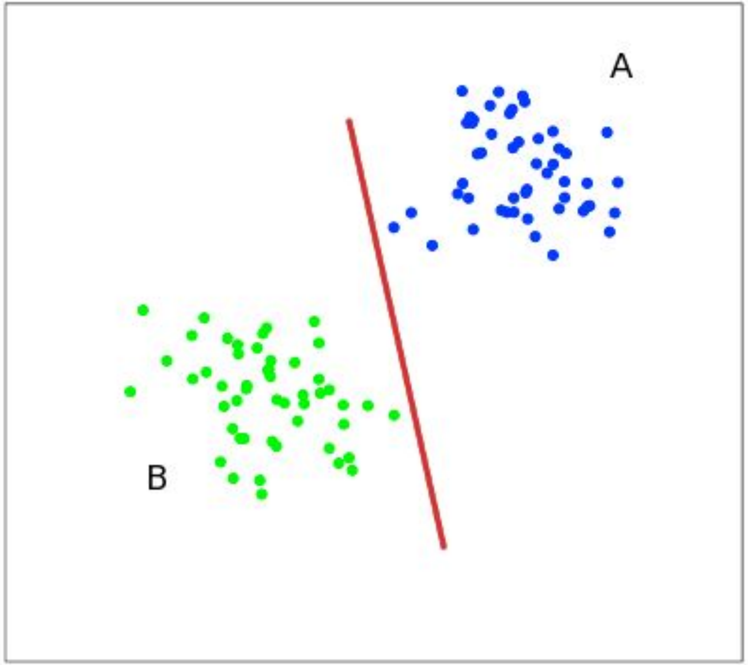
SVM

Постановка задачи:

Классификация на 2 класса: {-1, 1}

Строим линейную поверхность, которая разделяет два класса

$y = \mathrm{sign}(\langle w, x \rangle + b)$



http://www.machinelearning.ru/wiki/index.php?title=SVM

SVM

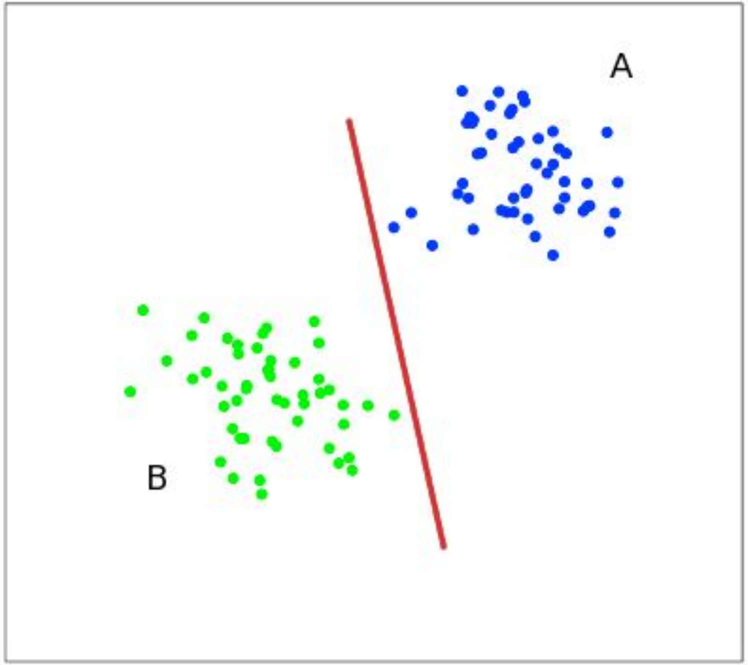
Хотим, чтобы для всех X ответы были правильные. Что это значит?
Это значит, что ответ нашей модели

sign(<w, x> + b)

и правильный ответ

y

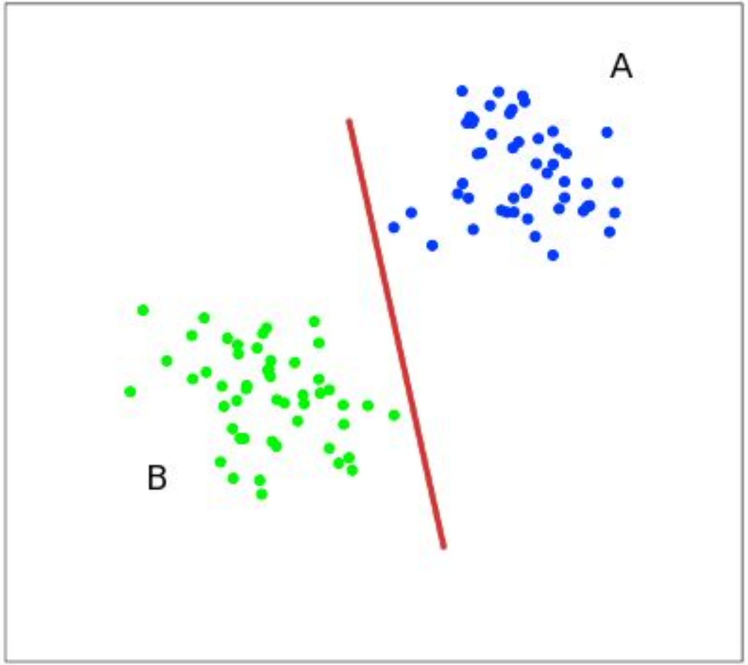
должны быть одного знака



SVM

То есть,

$$y(\langle w, x\rangle + b) > 0$$

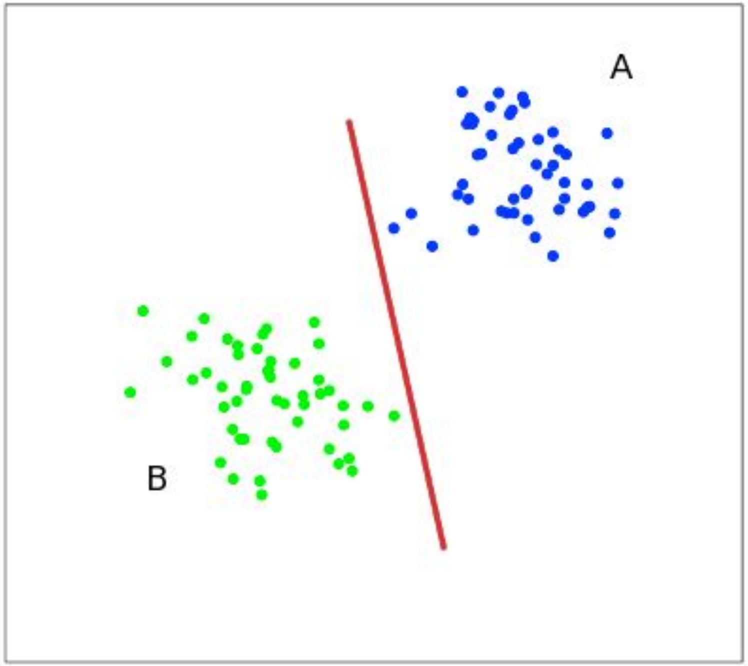


SVM

Домножив левую и правую чать на нужное число, получим:

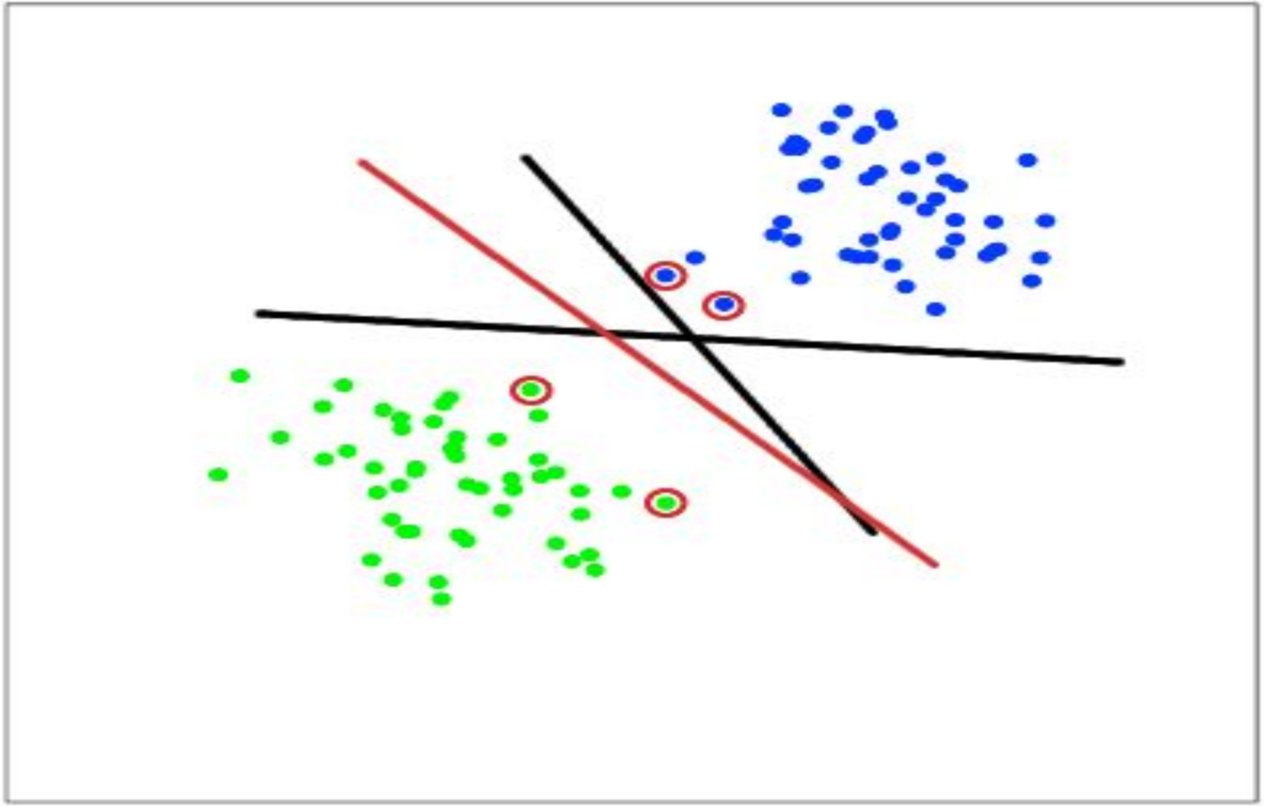
Постановка задачи:

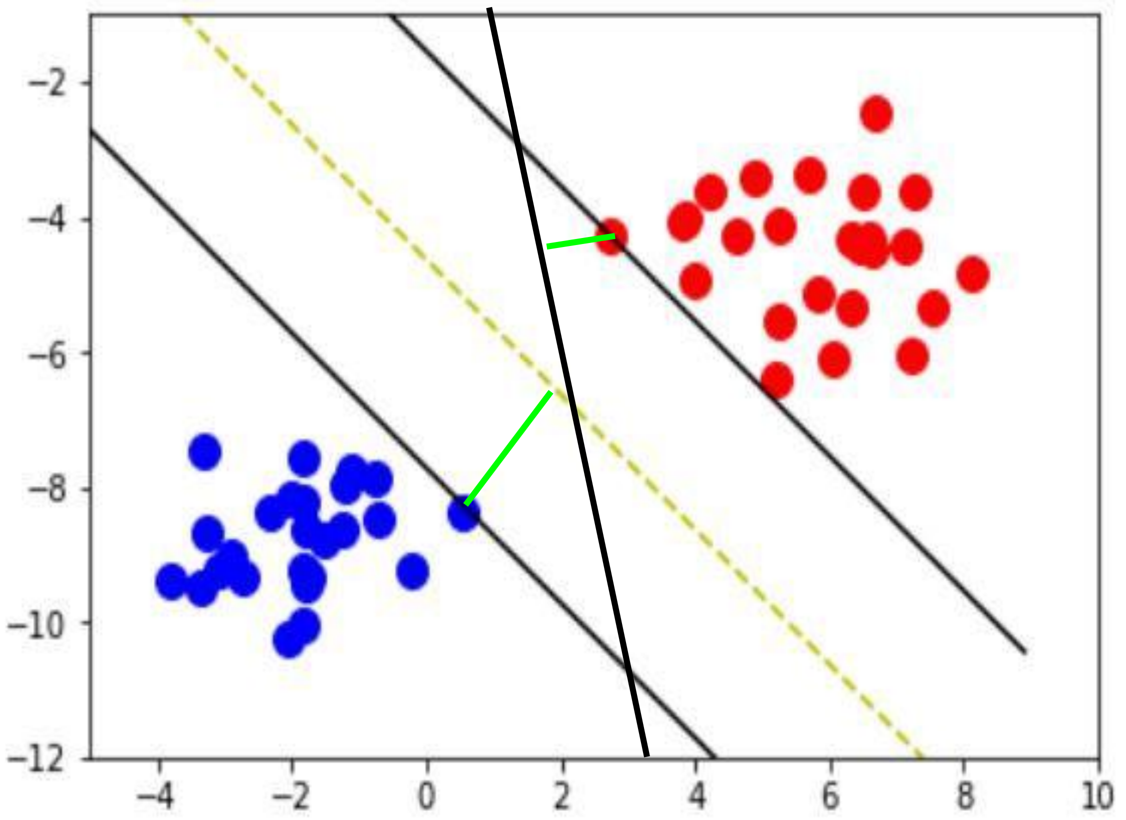
$$y(\langle w, x\rangle + b) > 1$$



http://www.machinelearning.ru/wiki/index.php?title=SVM

SVM

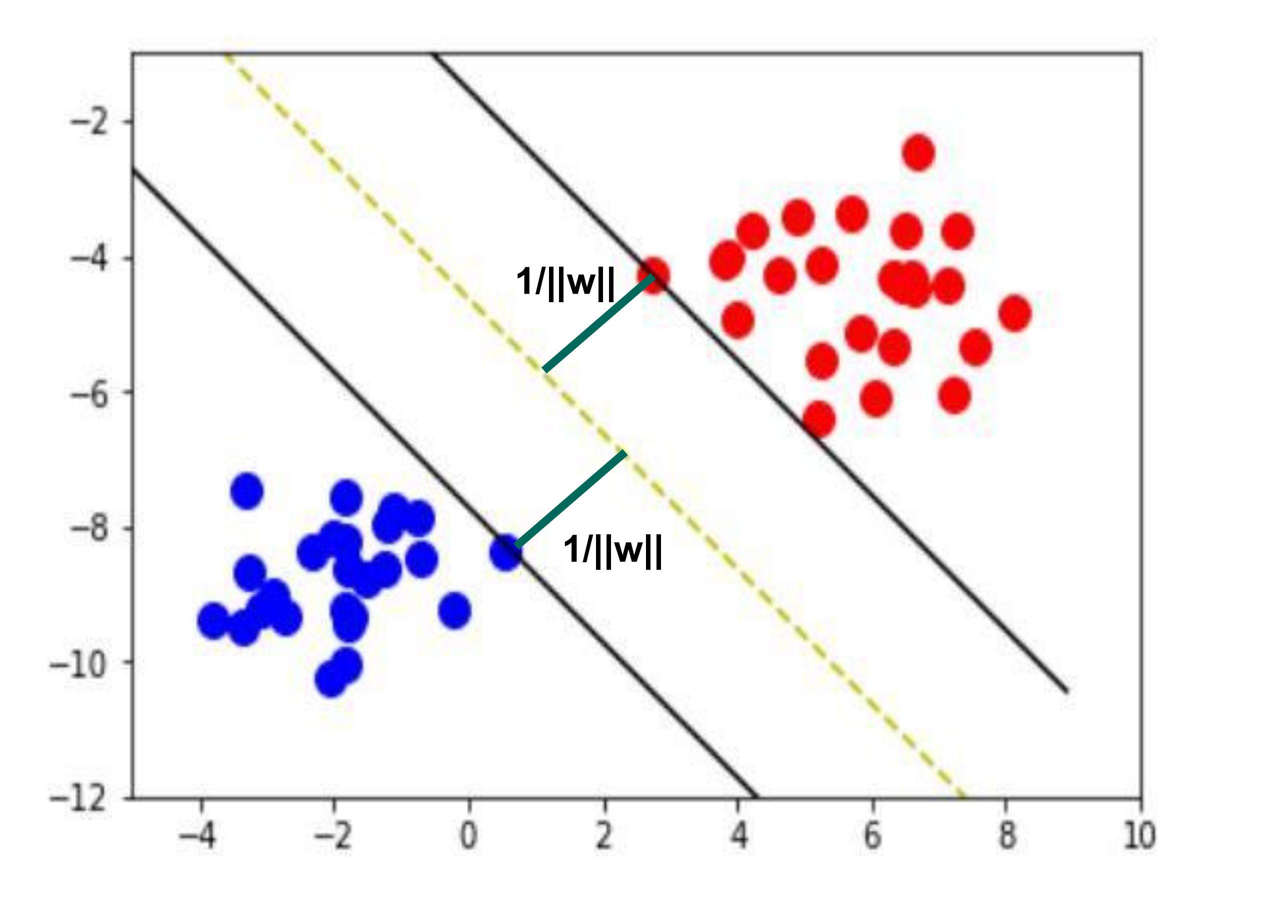



−2
−4
−6
−8
−10
−12
−4
−2
0
2
4
6
8
10

SVM

Разделяющие плоскости:

$$\langle w, x \rangle + b \pm 1 = 0$$



SVM

Расстояние между

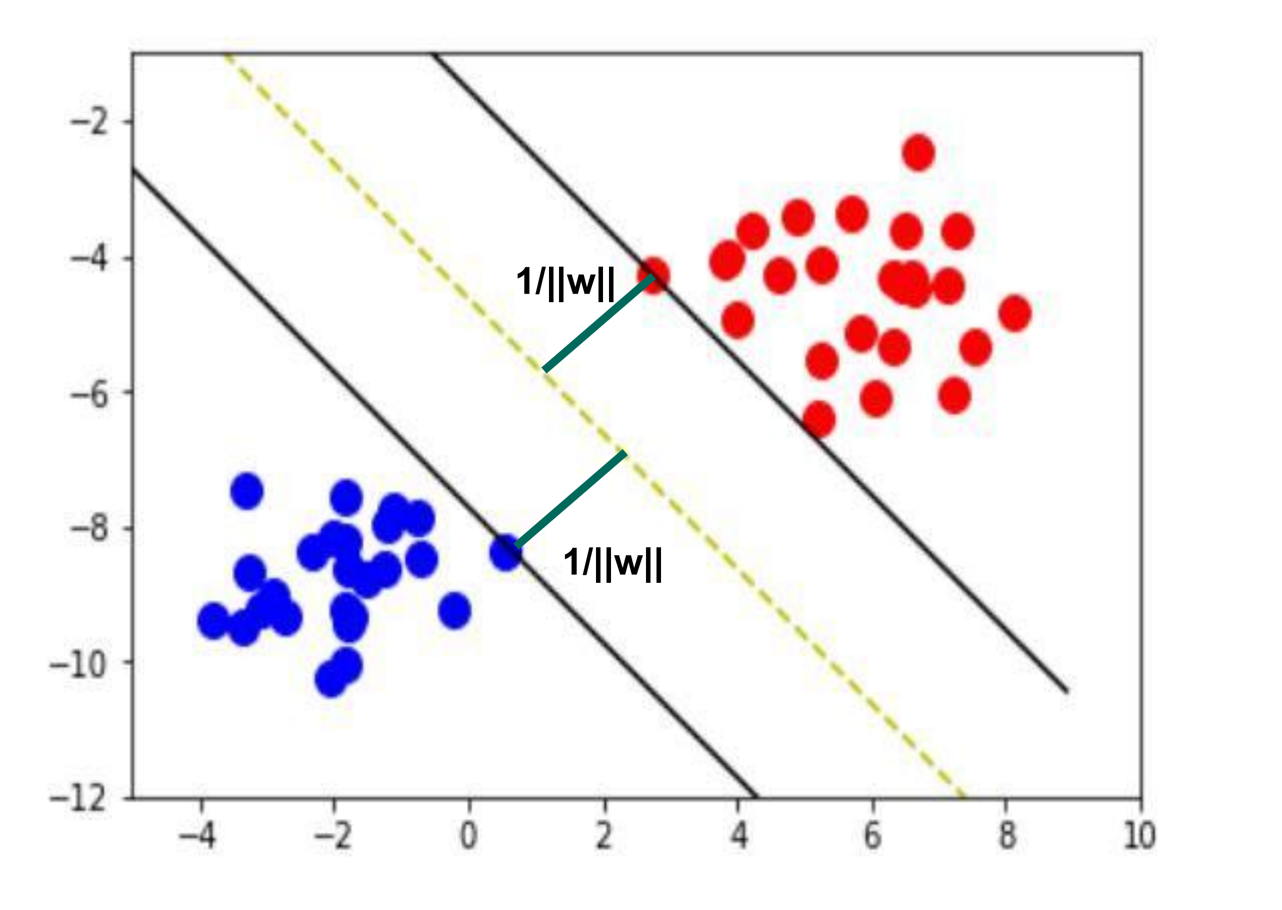
<w, x> + b + 1 = 0

и

<w, x> + b - 1 = 0

равно

2 / ||w||

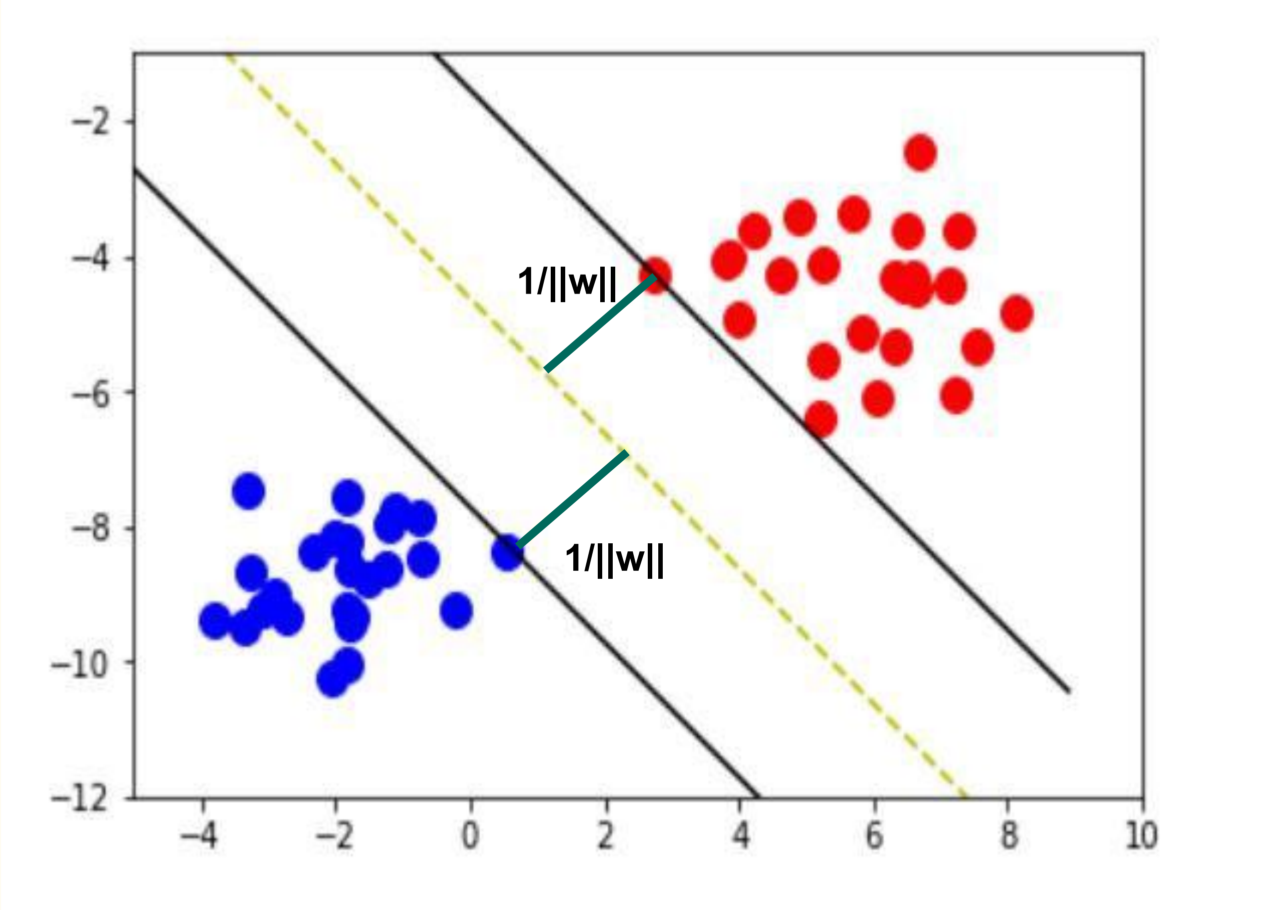


SVM

Задача оптимизации:

$$\begin{cases} y(\langle w, x\rangle + b) > 1 \\ 2 / \|w\| \rightarrow \max \end{cases}$$

$$\begin{cases} y(\langle w, x\rangle + b) > 1 \\ \|w\| \rightarrow \min \end{cases}$$



SVM

Ответ: **y = sign(<w, x> + b)**

Функция <x, w> -- **ядро**

С разными функциями ядер можно получать разные разделяющие плоскости

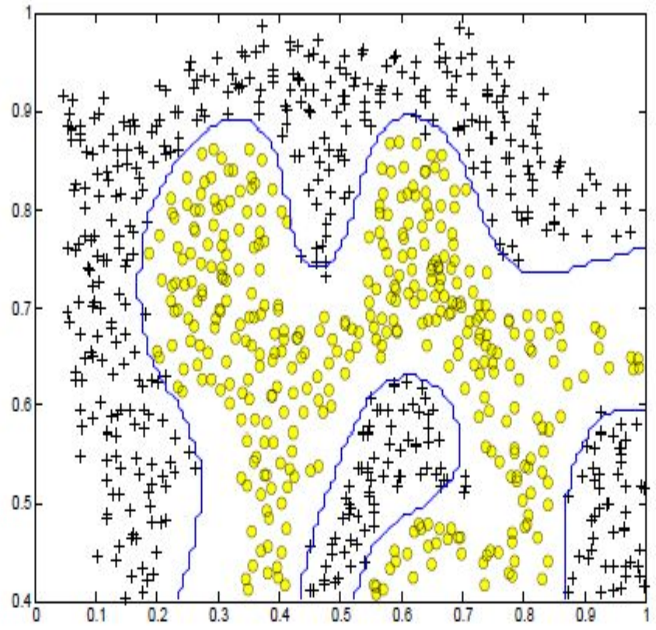


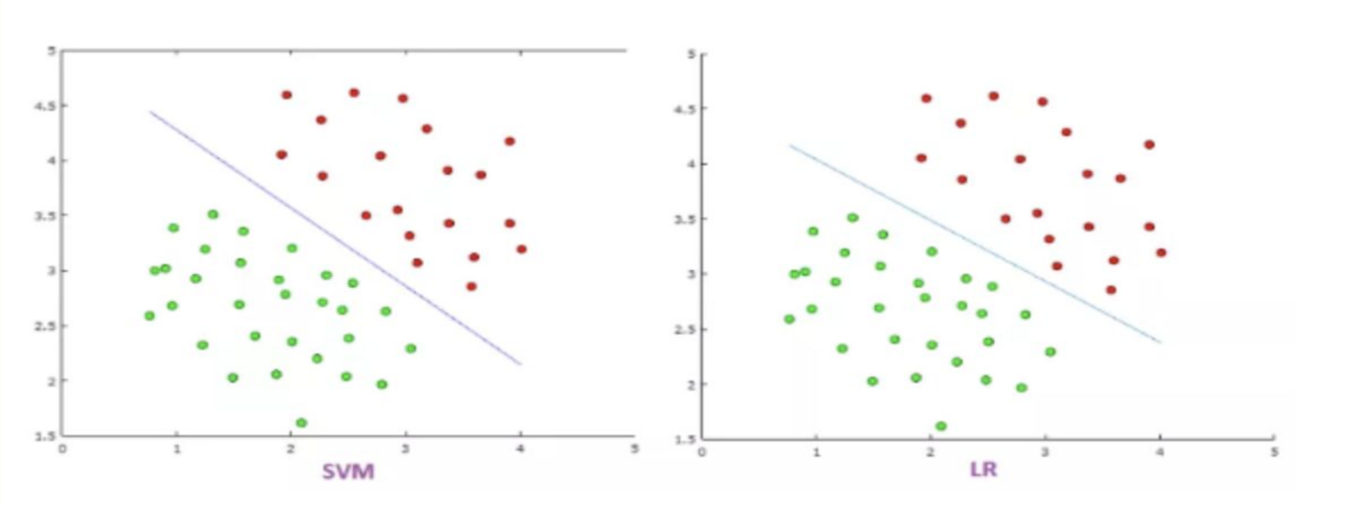
Figure 5: SVM (Gaussian Kernel) Decision Boundary (Example Dataset 2)

SVM vs LogReg

- SVM пытается построить разделение максимально далеко от обоих классов, т.е. для SVM оба класса равнозначны. У LogReg это не так.

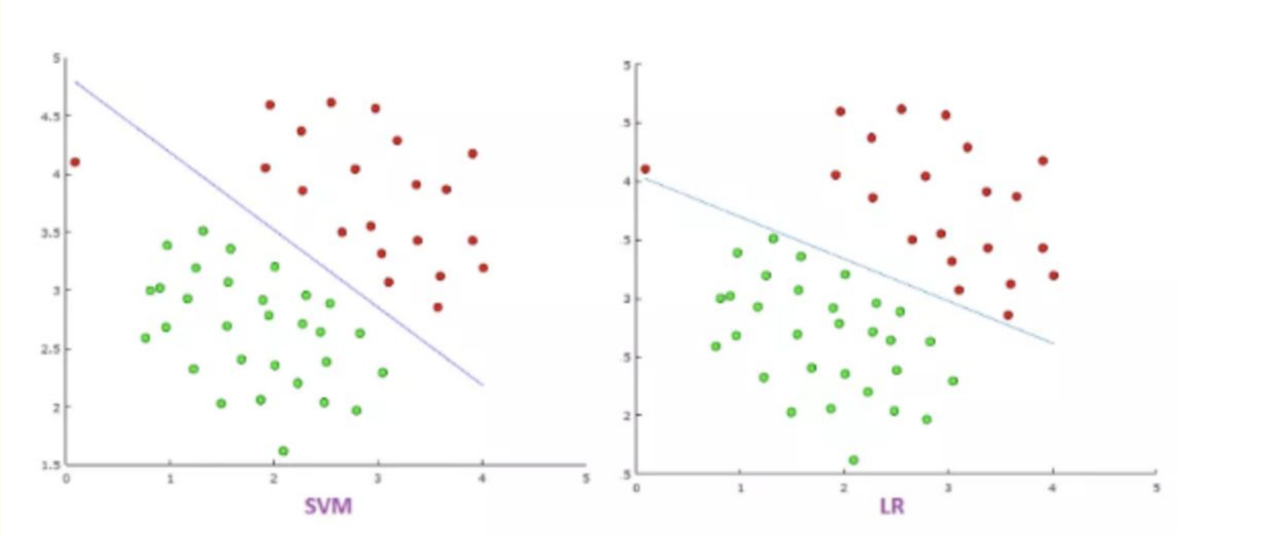
SVM vs LogReg

- SVM пытается построить разделение максимально далеко от обоих классов, т.е. для SVM оба класса равнозначны. У LogReg это не так.
- SVM менее чувствительна к шуму:



SVM vs LogReg

- SVM пытается построить разделение максимально далеко от обоих классов, т.е. для SVM оба класса равнозначны. У LogReg это не так.
- SVM менее чувствительна к шуму:

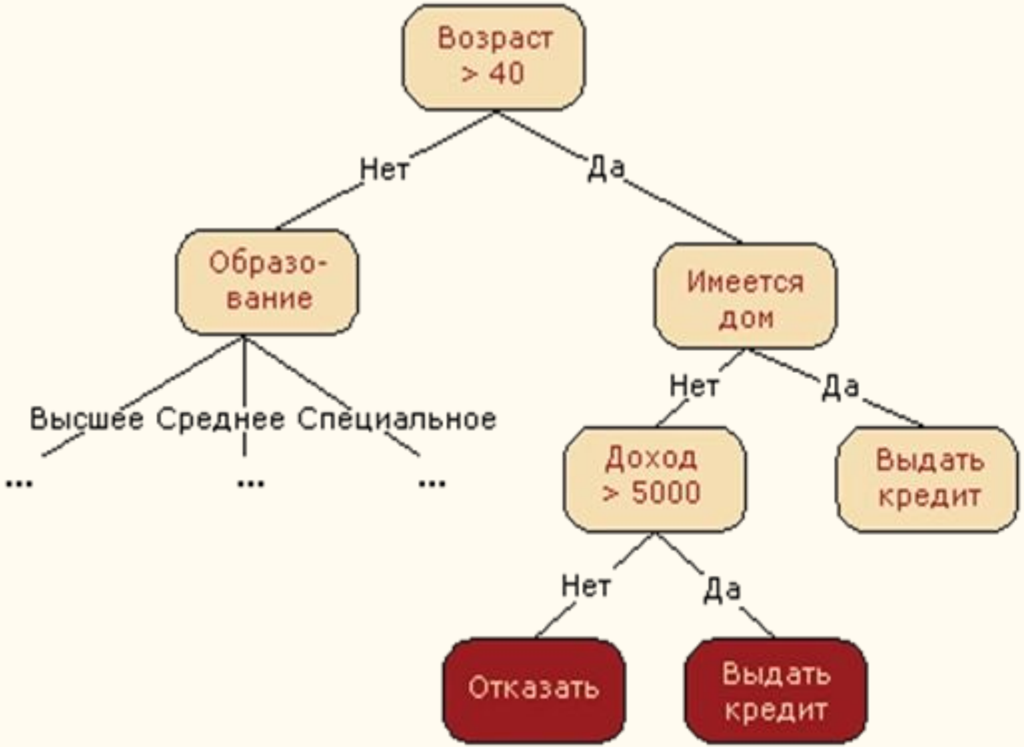


SVM vs LogReg

- SVM пытается построить разделение максимально далеко от обоих классов, т.е. для SVM оба класса равнозначны. У LogReg это не так.
- SVM менее чувствительна к шуму
- SVM умеет выдавать только ответы в виде {0, 1}, LogReg умеет выдавать вероятности принадлежности классам. Это хорошо, когда у нас мало данных и/или мы не очень уверены в надежности классификатора.

Логические методы

Решающее дерево



Логические методы классификации

Логическая закономерность – предикат **R:X**→{0,1}:

1. Интерпретируем
   a. записывается на естественном языке
   b. зависит от небольшого числа параметров (1-7)
2. Информативен

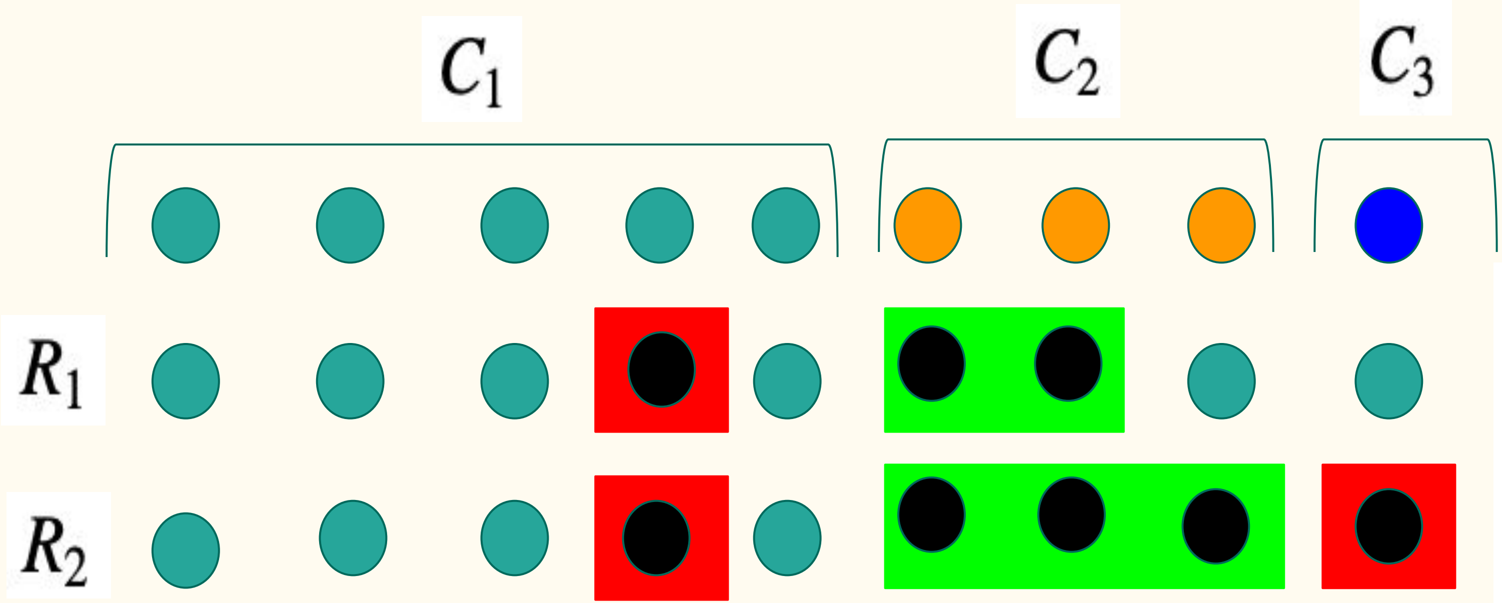
$$p_c(R) = \#\{x_i\colon R(x_i){=}1 \text{ и } y_i{=}c\} \to \max;$$
$$n_c(R) = \#\{x_i\colon R(x_i){=}1 \text{ и } y_i{\neq}c\} \to \min;$$

То есть, как можно лучше отделяет один класс от всех остальных

http://www.machinelearning.ru

Логические методы классификации

$$p_c(R) = \#\{x_i : R(x_i)=1 \text{ и } y_i=c\} \to \max;$$
$$n_c(R) = \#\{x_i : R(x_i)=1 \text{ и } y_i\neq c\} \to \min;$$

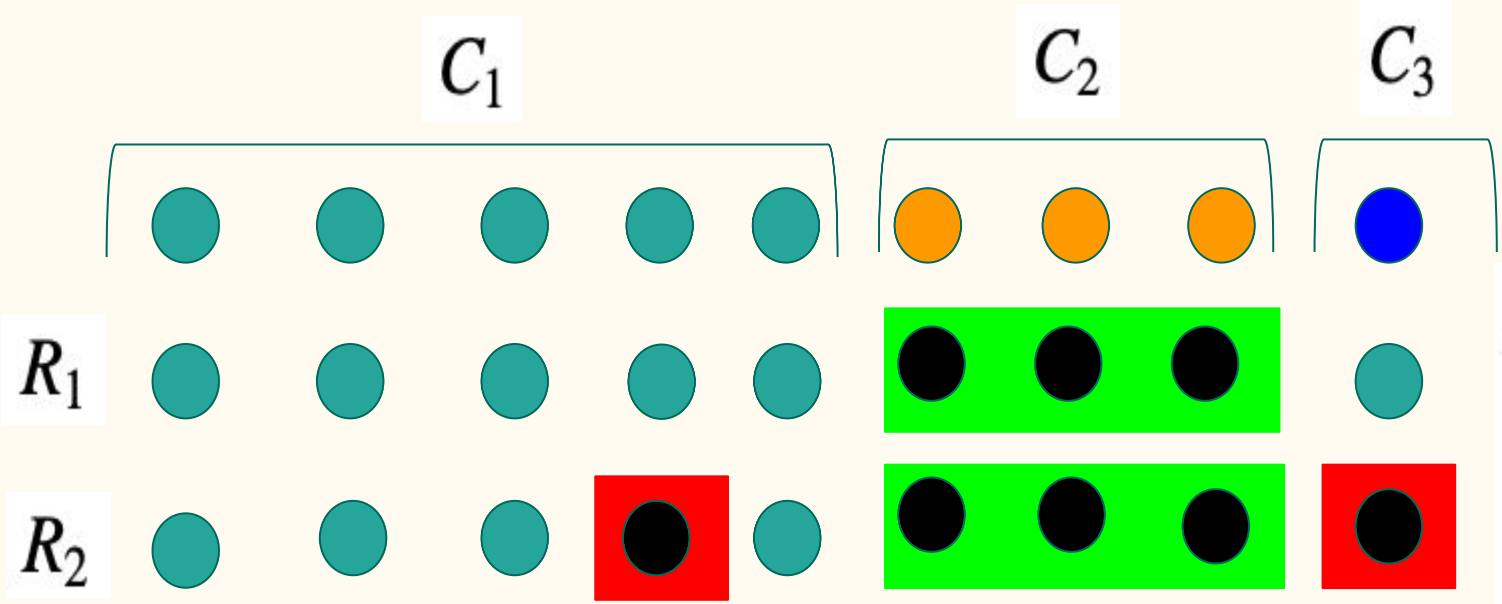


$$p_{c_2}(R_1) = 2$$
$$n_{c_2}(R_1) = 1$$
$$p_{c_2}(R_2) = 3$$
$$n_{c_2}(R_2) = 2$$

Логические методы классификации

$p_c(R) = \#\{x_i : R(x_i)=1 \text{ и } y_i=c\} \to \max;$

$n_c(R) = \#\{x_i : R(x_i)=1 \text{ и } y_i\neq c\} \to \min;$



$p_{c2}(R_1) = 3$
$n_{c2}(R_1) = 0$
$p_{c_2}(R_2) = 3$
$n_{c_2}(R_2) = 2$

http://www.mchinelearning.ru

Логические методы классификации

Пример информативности:

Если в анкете указан домашний телефон
и зарплата $> \$2000$ **и** сумма кредита $< \$5000$
то кредит можно выдать, риск дефолта 5%.

R = (“наличие домашнего телефона” == 1 **∧** “зарплата” > 2000 **∧** “сумма кредита” < 5000)

C = 1

Закономерности: наборы правил

1. Пороговое условие (решающий пень, decision stump):

$$R(x) = \left[f_j(x) \leqslant a_j\right] \text{ или } \left[a_j \leqslant f_j(x) \leqslant b_j\right].$$

2. *Конъюнкция* пороговых условий:

$$R(x) = \bigwedge_{j \in J} \left[a_j \leqslant f_j(x) \leqslant b_j\right].$$

3. *Синдром* — выполнение не менее d условий из $|J|$, (при $d = |J|$ это конъюнкция, при $d = 1$ — дизъюнкция):

$$R(x) = \left[\sum_{j \in J} \left[a_j \leqslant f_j(x) \leqslant b_j\right] \geqslant d\right],$$

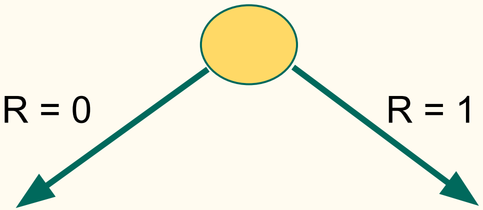
Построение дерева

Наша выборка X

PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292
893	3	Wilkes, Mrs. James (Ellen Needs)	female	47	1	0	363272	7
894	2	Myles, Mr. Thomas Francis	male	62	0	0	240276	9.6875

Строим дерево:

R -- самый информативный для датасета **X** признак



$\mathbf{X^1}$

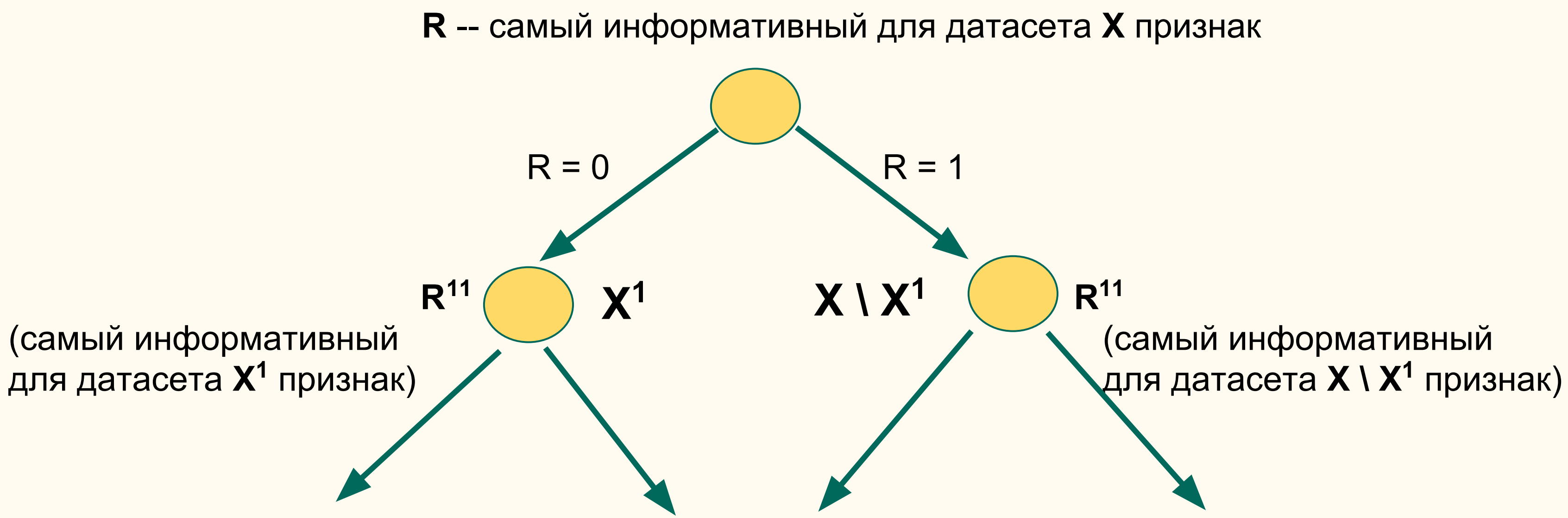
(те примеры из X, для которых R(X) = 0)

$\mathbf{X \setminus X^1}$

(те примеры из X, для которых R(X) = 1)

Построение дерева

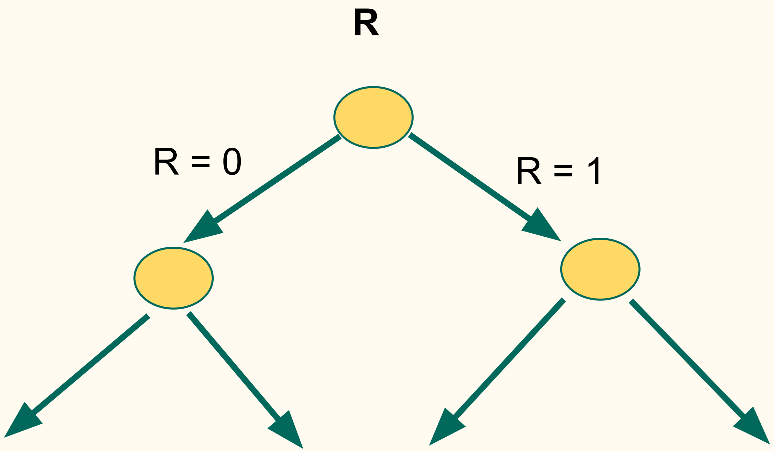
Строим дерево:



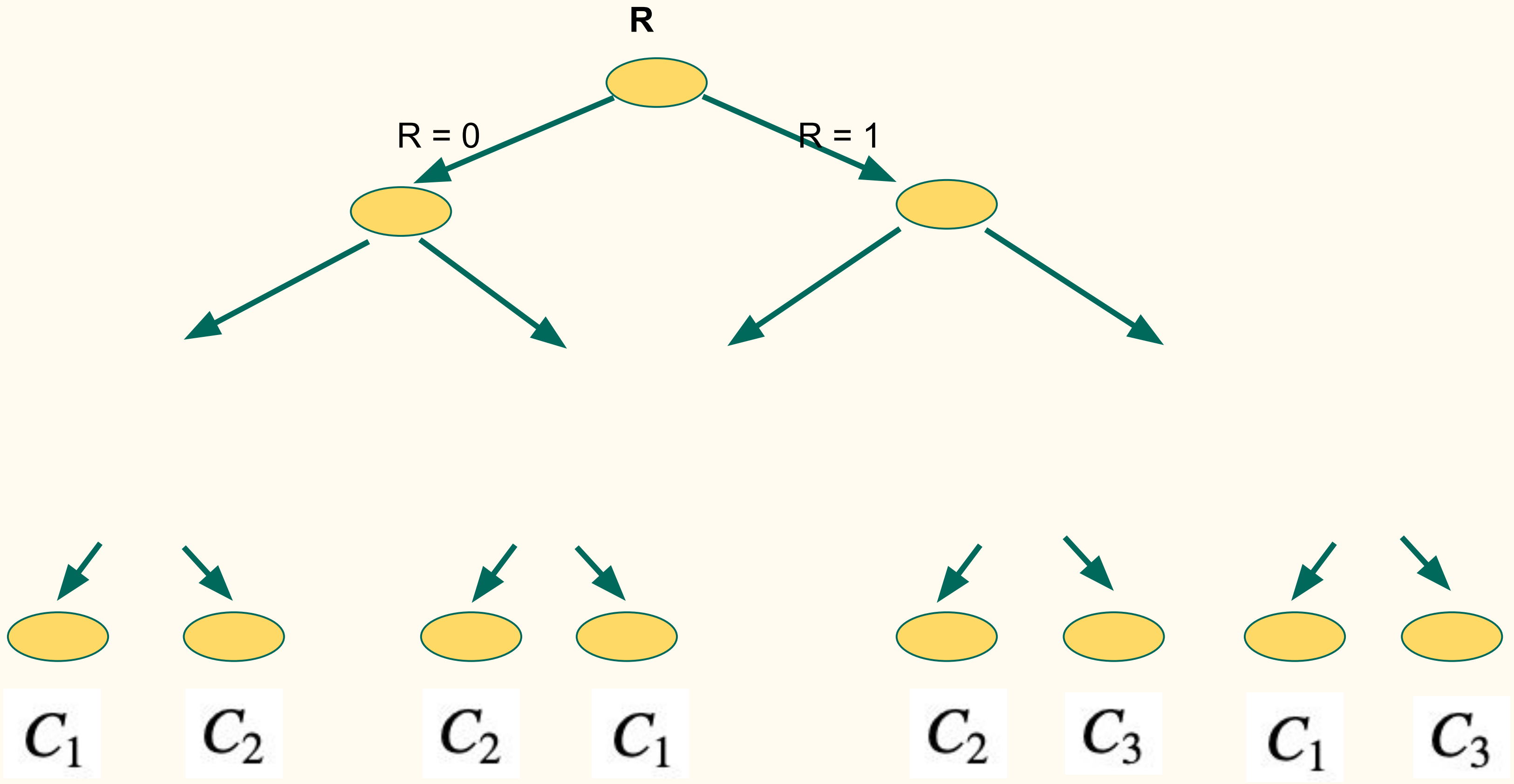
Построение дерева

Вопросы:

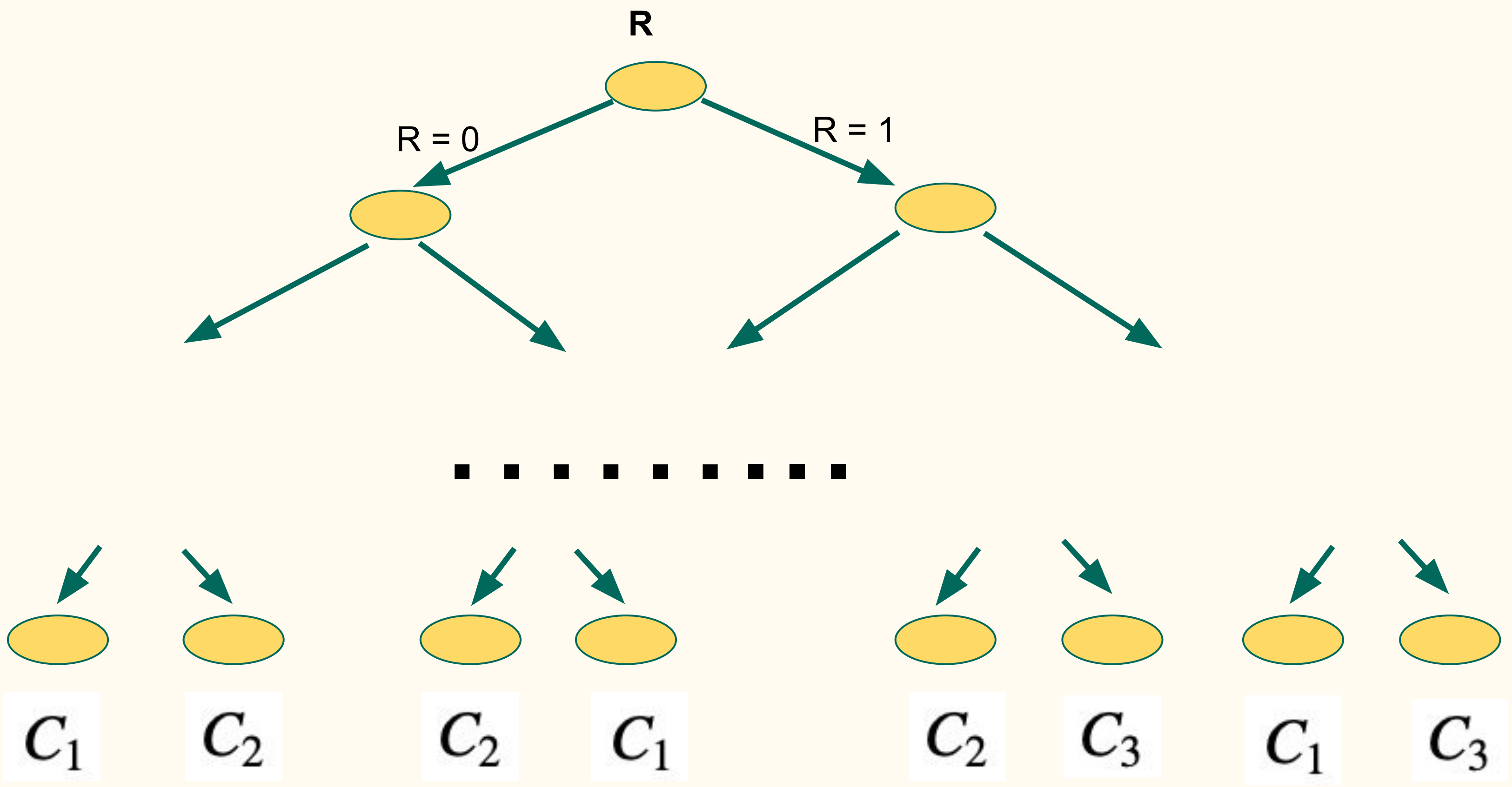
1. Как выбрать “самый информативный признак”
2. Когда остановиться?
3. Как классифицировать объект полученным деревом?



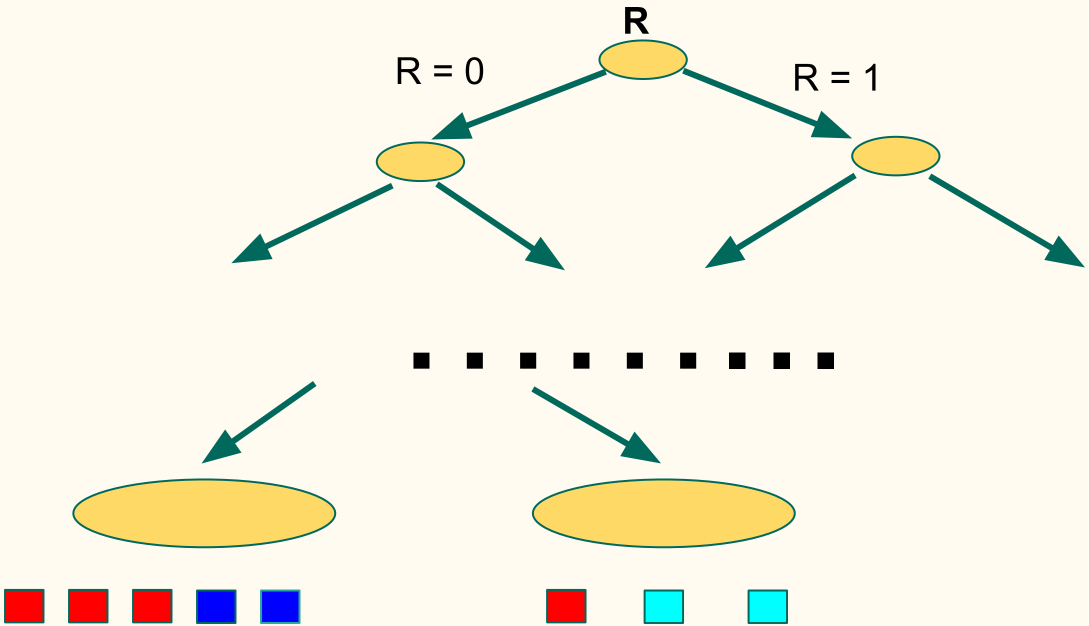
Построение дерева: когда остановиться?



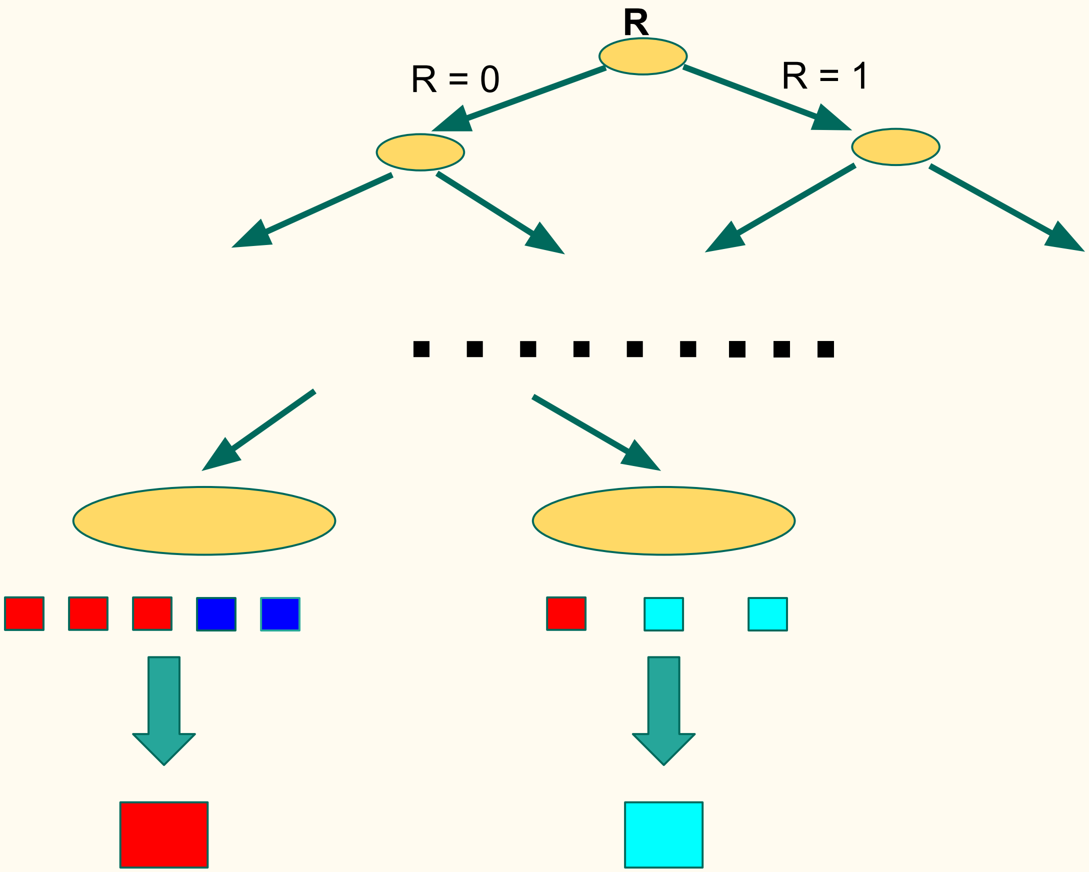
Построение дерева: когда остановиться?



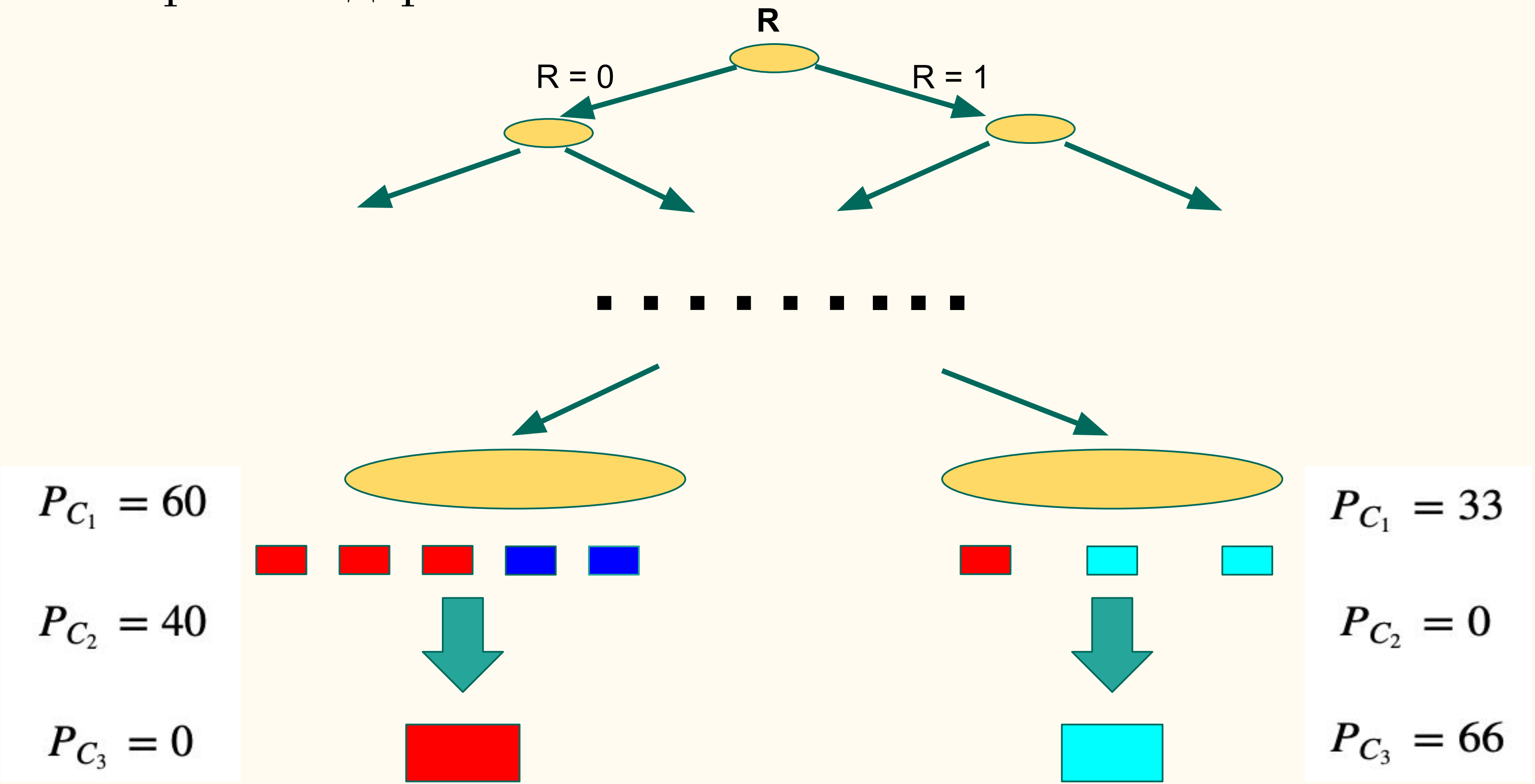
Построение дерева: когда остановиться?



Построение дерева: как классифицировать?



Построение дерева



Оценка качества информативности

Проблема: надо сравнивать закономерности R.

Как свернуть два критерия в один критерий информативности?

$$\boxed{\begin{cases} p(R) \to \max \\ n(R) \to \min \end{cases} \quad \overset{?}{\Longrightarrow} \quad I(p,n) \to \max}$$

Очевидные, но не всегда адекватные свёртки:

- $I(p,n) = \dfrac{p}{p+n} \to \max$ (precision);
- $I(p,n) = p - n \to \max$ (accuracy);

Пример:

при $P = 200$, $N = 100$ и различных p и n.

Простые эвристики не всегда адекватны:

p	n	$p-n$	$p-5n$	$\frac{p}{P}-\frac{n}{N}$	$\frac{p}{n+1}$	$\text{IStat}\cdot\ell$	$\text{IGain}\cdot\ell$	$\sqrt{p}-\sqrt{n}$
50	0	**50**	50	0.25	50	22.65	23.70	7.07
100	50	**50**	−150	0	1.96	2.33	1.98	2.93
50	9	41	**5**	0.16	**5**	7.87	7.94	4.07
5	0	5	**5**	0.03	**5**	2.04	3.04	2.24
100	0	**100**	100	**0.5**	100	52.18	53.32	10.0
140	20	**120**	40	**0.5**	6.67	37.09	37.03	7.36

Более адекватные свертки

сколько информации мы получим о разделении объектов на два класса, если узнаем **R**?

- энтропийный критерий прироста информации:

$$\mathrm{IGain}(p, n) = h\left(\tfrac{P}{\ell}\right) - \tfrac{p+n}{\ell} h\left(\frac{p}{p+n}\right) - \tfrac{\ell-p-n}{\ell} h\left(\frac{P-p}{\ell-p-n}\right) \to \max,$$

  где $h(q) = -q \log_2 q - (1-q)\log_2(1-q)$

- критерий Джини (Gini impurity):

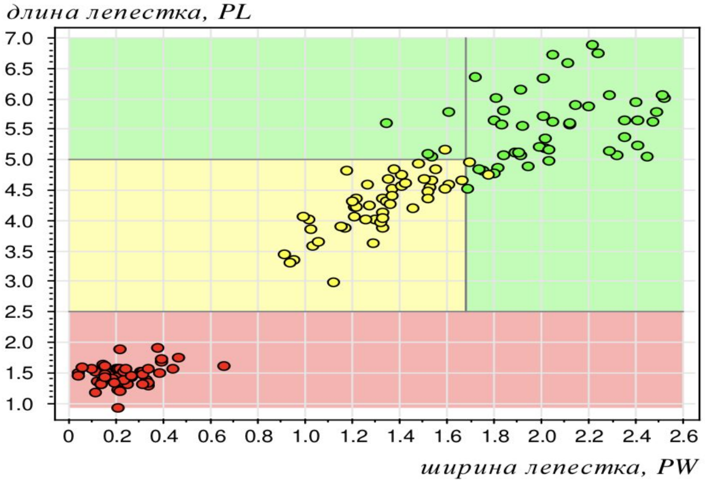
  $\mathrm{IGini}(p, n) = \mathrm{IGain}(p, n)$ при $h(q) = 4q(1-q)$

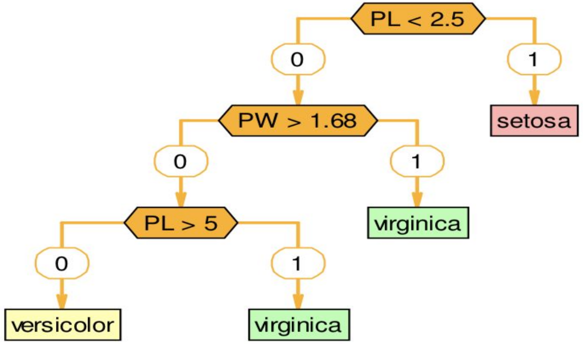
"загрязненность" распределения

http://www.machinelearning.ru

Пример решающего дерева

Задача Фишера о классификации цветков ириса на 3 класса, в выборке по 50 объектов каждого класса, 4 признака.





На графике: в осях двух самых информативных признаков (из 4) два класса разделились без ошибок, на третьем 3 ошибки.

http://www.machinelearning.ru

Построение решающего дерева

1: **ПРОЦЕДУРА** LearnID3 $(U \subseteq X^\ell)$;
2: **если** все объекты из U лежат в одном классе $c \in Y$ **то**
3: **вернуть** новый лист v, $c_v := c$;
4: найти предикат с максимальной информативностью:
$\beta := \arg\max\limits_{\beta \in \mathscr{B}} I(\beta, U)$;
5: разбить выборку на две части $U = U_0 \cup U_1$ по предикату β:
$U_0 := \{x \in U \colon \beta(x) = 0\}$;
$U_1 := \{x \in U \colon \beta(x) = 1\}$;
6: **если** $U_0 = \varnothing$ или $U_1 = \varnothing$ **то**
7: **вернуть** новый лист v, $c_v :=$ Мажоритарный класс(U);
8: создать новую внутреннюю вершину v: $\beta_v := \beta$;
построить левое поддерево: $L_v :=$ LearnID3 (U_0);
построить правое поддерево: $R_v :=$ LearnID3 (U_1);
9: **вернуть** v;

http://www.machinelearning.ru

Решающие деревья: + и -

+

- Интерпретируемость, простота
- Гибкость: можно менять множества закономерностей
- Допустимы данные с пропусками
- Сложность построения линейна по длины выборки

-

- Жадный алгоритм выбора правила для вершины переусложняает дерево
- Чем дальше вершина от корня, тем слабее статистическая надежность решающего правила
- Высокая чувствительность к шуму, составу выборки

`sklearn.tree`.DecisionTreeClassifier

class `sklearn.tree.` **`DecisionTreeClassifier`** (*criterion='gini'*, *splitter='best'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features=None*, *random_state=None*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *min_impurity_split=None*, *class_weight=None*, *presort=False*)

[source]

A decision tree classifier.

Read more in the User Guide.

Parameters:

criterion : string, optional (default="gini")

> The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

splitter : string, optional (default="best")

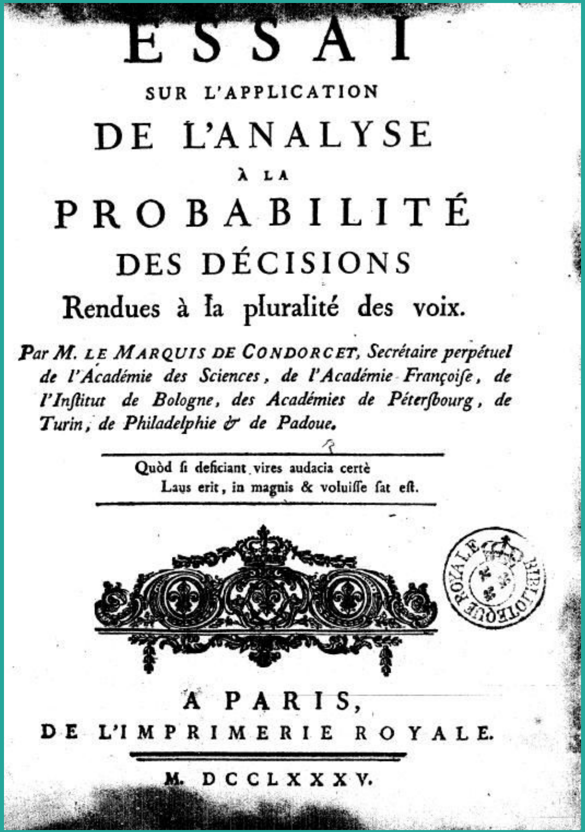
> The strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.

max_depth : int or None, optional (default=None)

> The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

Композиции алгоритмов

Если каждый член жюри имеет независимое мнение, и если вероятность правильного решения члена жюри больше 0.5, то тогда вероятность правильного решения присяжных в целом возрастает с увеличением количества членов жюри и стремится к единице.
Если же вероятность быть правым у каждого из членов жюри меньше 0.5, то вероятность принятия правильного решения присяжными в целом монотонно уменьшается и стремится к нулю с увеличением количества присяжных.


ESSAI
SUR L'APPLICATION
DE L'ANALYSE
À LA
PROBABILITÉ
DES DÉCISIONS
Rendues à la pluralité des voix.

Par M. LE MARQUIS DE CONDORCET, Secrétaire perpétuel de l'Académie des Sciences, de l'Académie Françoise, de l'Institut de Bologne, des Académies de Pétersbourg, de Turin, de Philadelphie & de Padoue.

Quòd si deficiant vires audacia certè
Laus erit, in magnis & voluisse sat est.

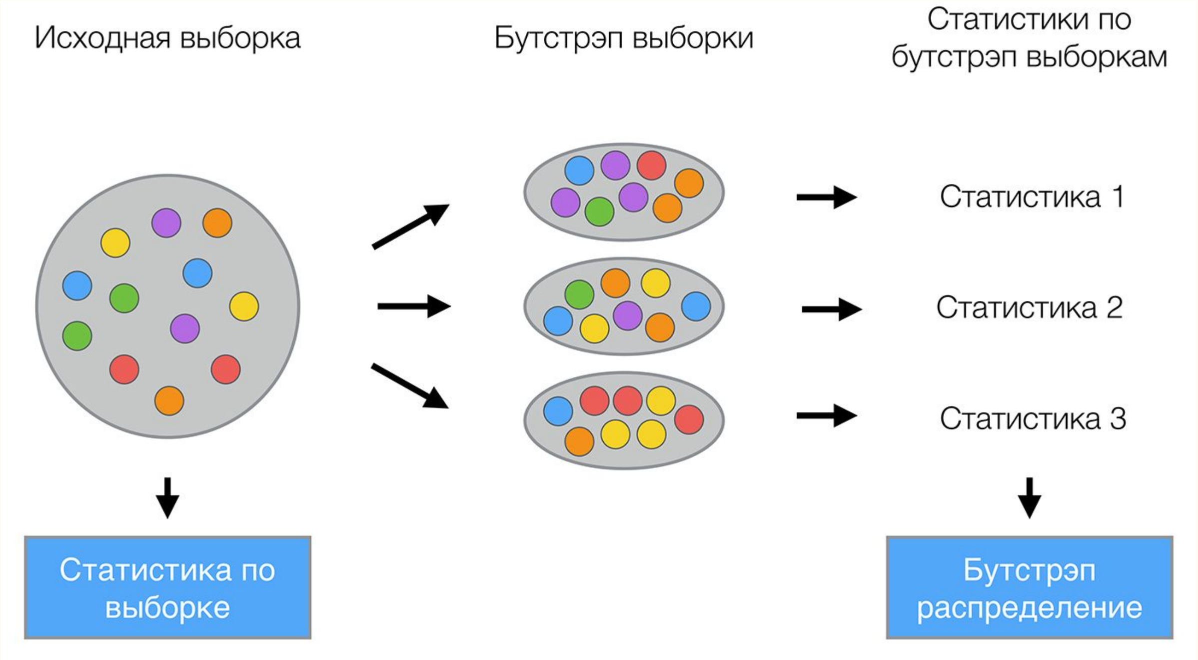
A PARIS,
DE L'IMPRIMERIE ROYALE.
M. DCCLXXXV.

принцип Кондорсе, 1784

Собралось около 800 человек, которые попытались угадать вес быка на ярмарке. Бык весил 1198 фунтов. Ни один крестьянин не угадал точный вес быка, но если посчитать среднее от их предсказаний, то получим 1197 фунтов.

Гальтон, 1906 год

Bootstrap



https://habr.com/company/ods/blog/324402/

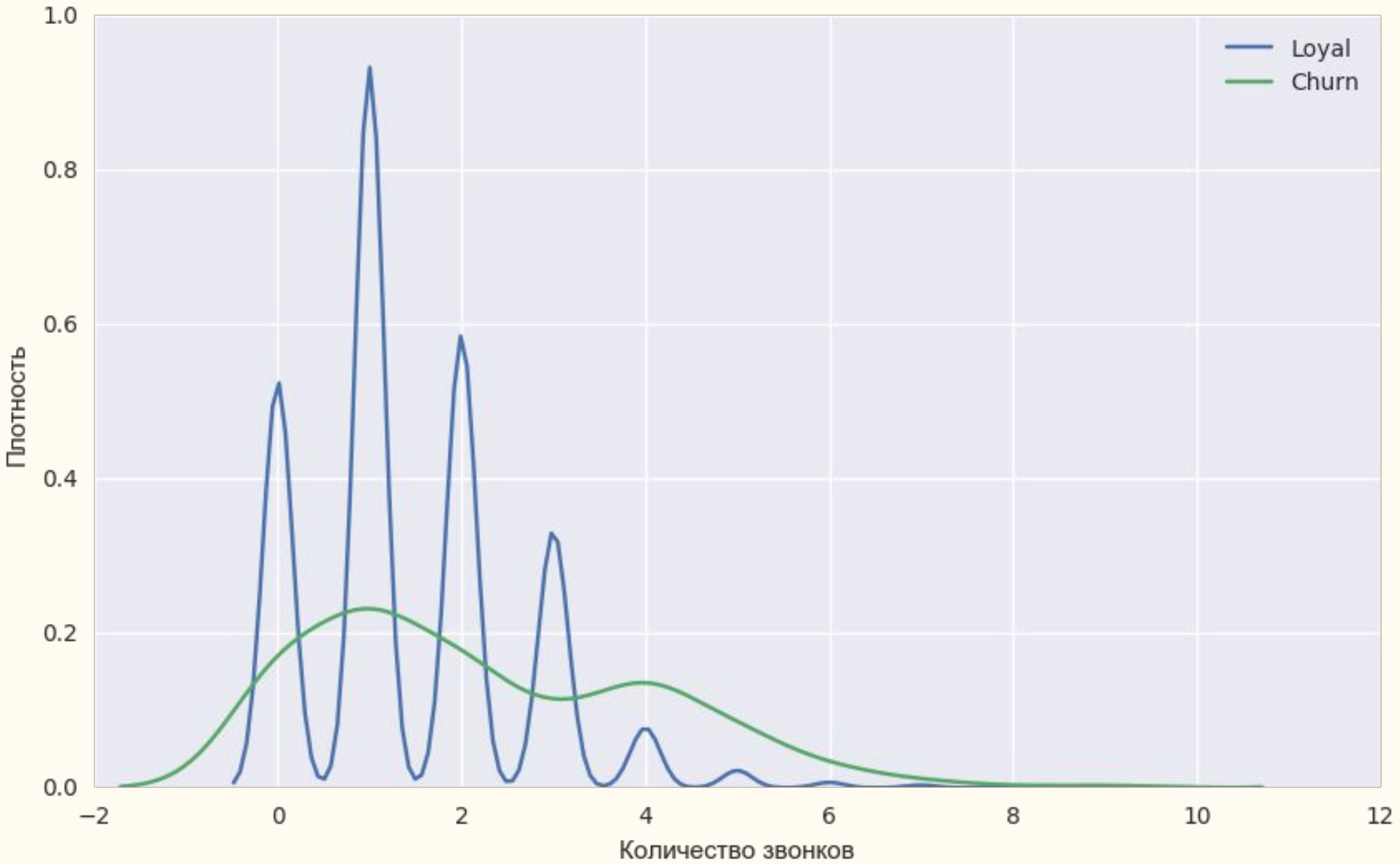
Бинарная классификация оттока клиентов

```
import pandas as pd
from matplotlib import pyplot as plt
plt.style.use('ggplot')
plt.rcParams['figure.figsize'] = 10, 6
import seaborn as sns
%matplotlib inline

telecom_data = pd.read_csv('data/telecom_churn.csv')

fig = sns.kdeplot(telecom_data[telecom_data['Churn'] == False]['Customer service call
s'], label = 'Loyal')
fig = sns.kdeplot(telecom_data[telecom_data['Churn'] == True]['Customer service call
s'], label = 'Churn')
fig.set(xlabel='Количество звонков', ylabel='Плотность')
plt.show()
```

Мало данных, одна из главных фич — количество звонков в сервисный центр



https://habr.com/company/ods/blog/324402/

Оценим, сколько в среднем делает звонков каждая из групп.

Данных мало, поэтому искать среднее не совсем правильно, применим **bootstap** и сделаем **интервальную оценку среднего**

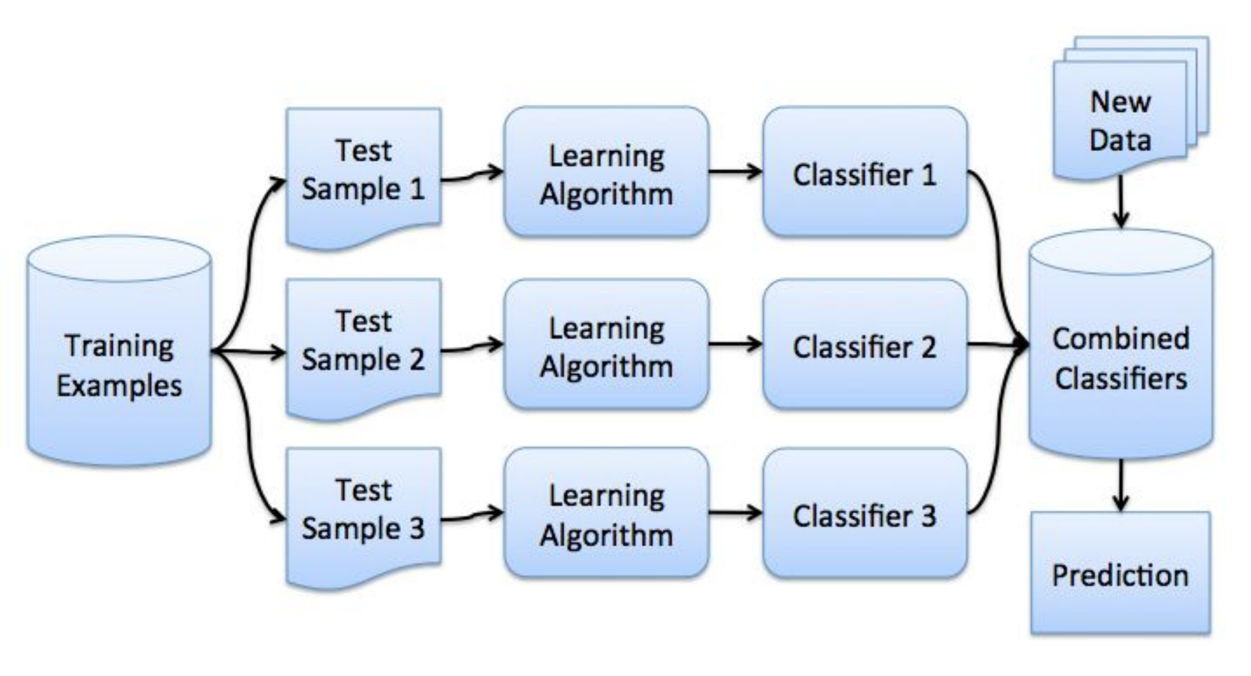
```
import numpy as np
def get_bootstrap_samples(data, n_samples):
    # функция для генерации подвыборок с помощью бутстрэпа
    indices = np.random.randint(0, len(data), (n_samples, len(data)))
    samples = data[indices]
    return samples
def stat_intervals(stat, alpha):
    # функция для интервальной оценки
    boundaries = np.percentile(stat, [100 * alpha / 2., 100 * (1 - alpha / 2.)])
    return boundaries
```

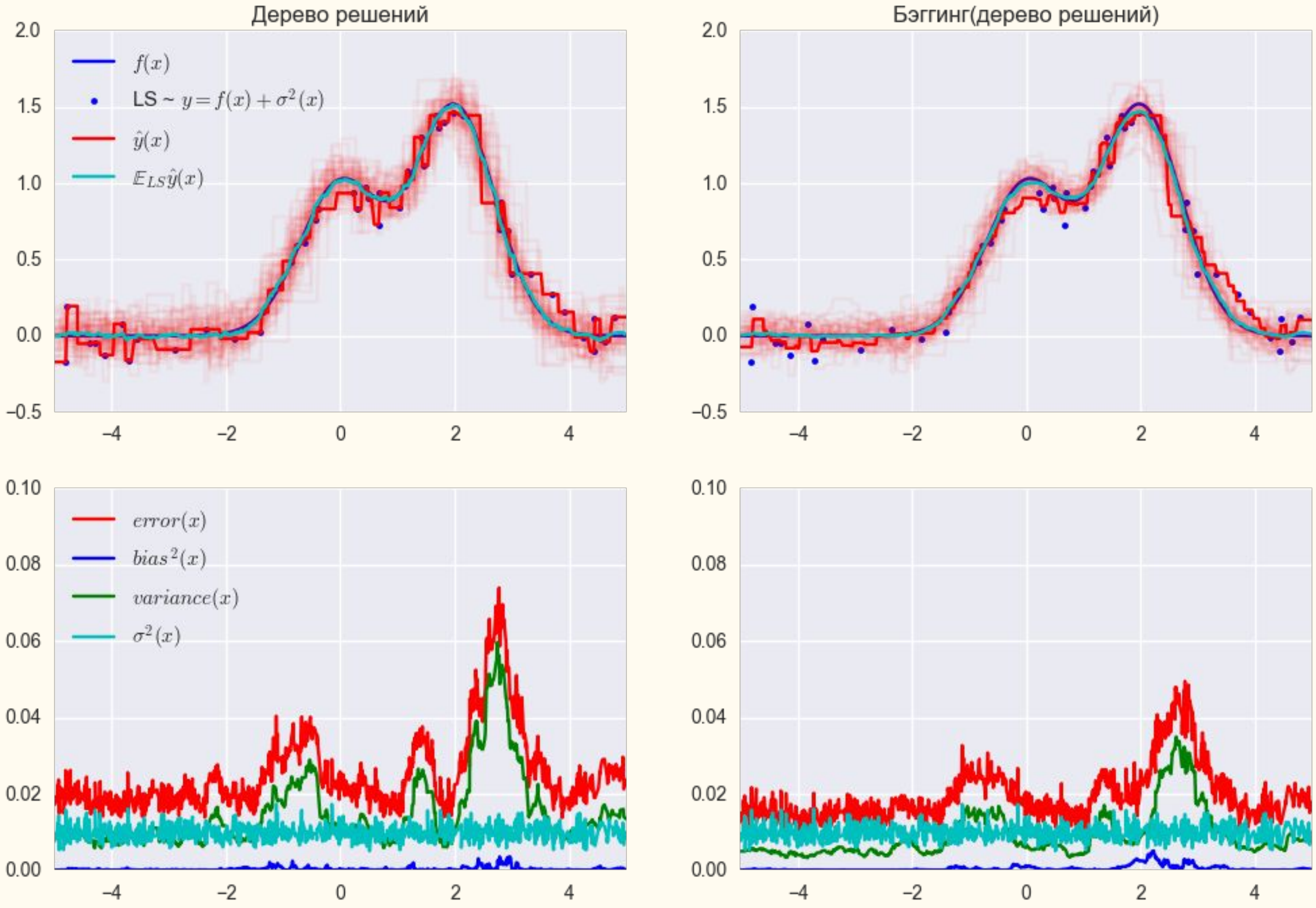
С 95% вероятностью среднее число звонков от лояльных клиентов будет лежать в промежутке между 1.40 и 1.50, в то время как наши бывшие клиенты звонили в среднем от 2.06 до 2.40 раз

Bagging (**b**ootstrap **agg**regat**ing**)

Дана train sample X. С помощью **bootstrap** сгенерируем из неё M выборок. Теперь на каждой выборке обучим свой классификатор. Итоговый классификатор будет усреднять ответы всех этих алгоритмов:

$$a(x) = \frac{1}{M}\sum_{i=1}^{M} a_i(x).$$

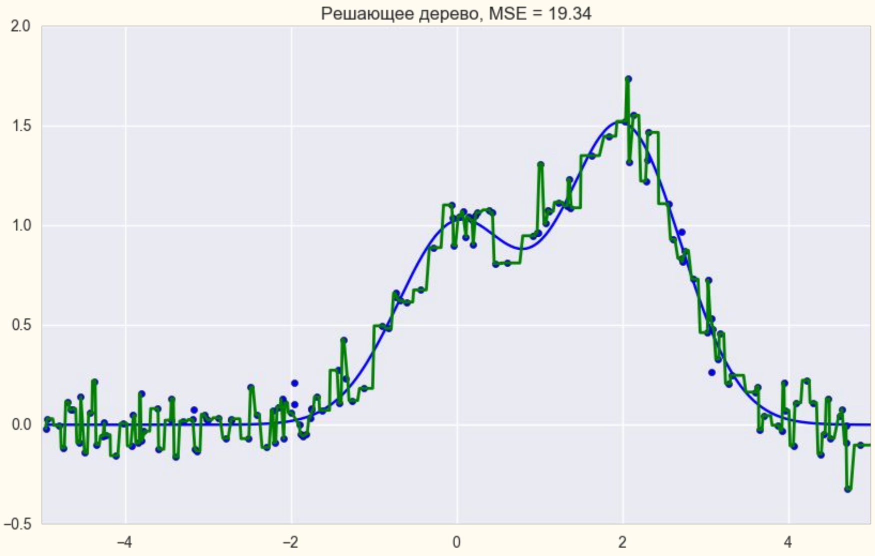


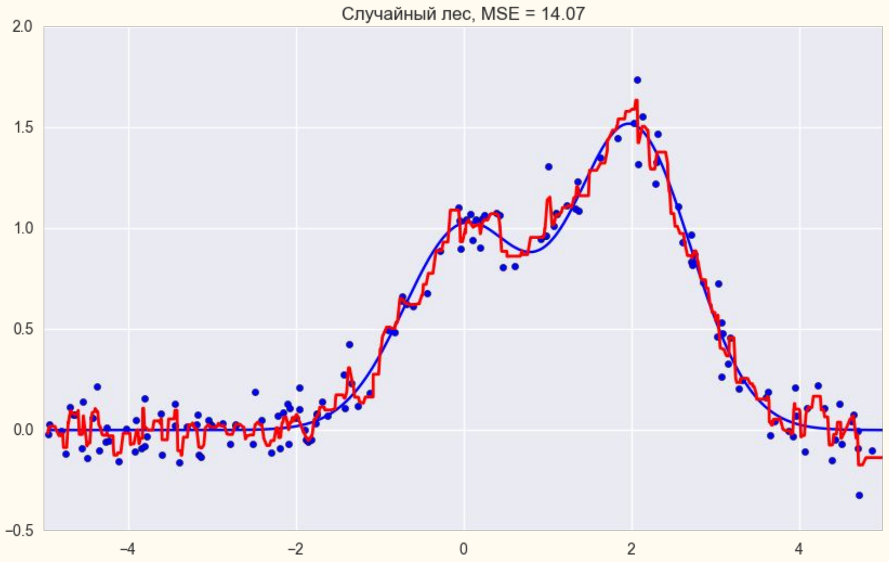


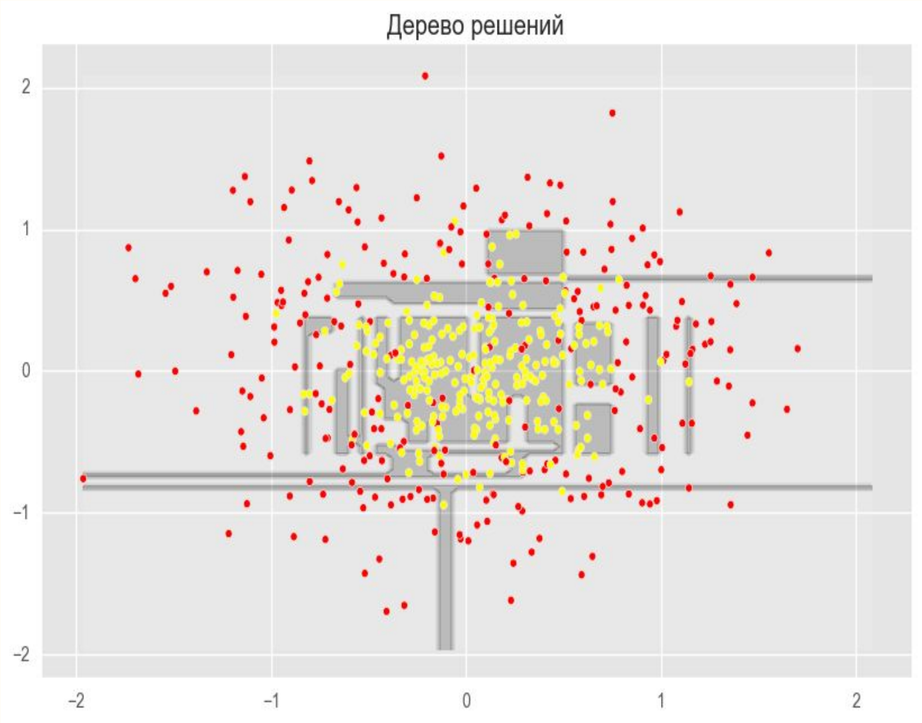
Дерево решений
Бэггинг(дерево решений)
$f(x)$
LS ~ $y = f(x) + \sigma^2(x)$
$\hat{y}(x)$
$\mathbb{E}_{LS}\hat{y}(x)$
$error(x)$
$bias^2(x)$
$variance(x)$
$\sigma^2(x)$

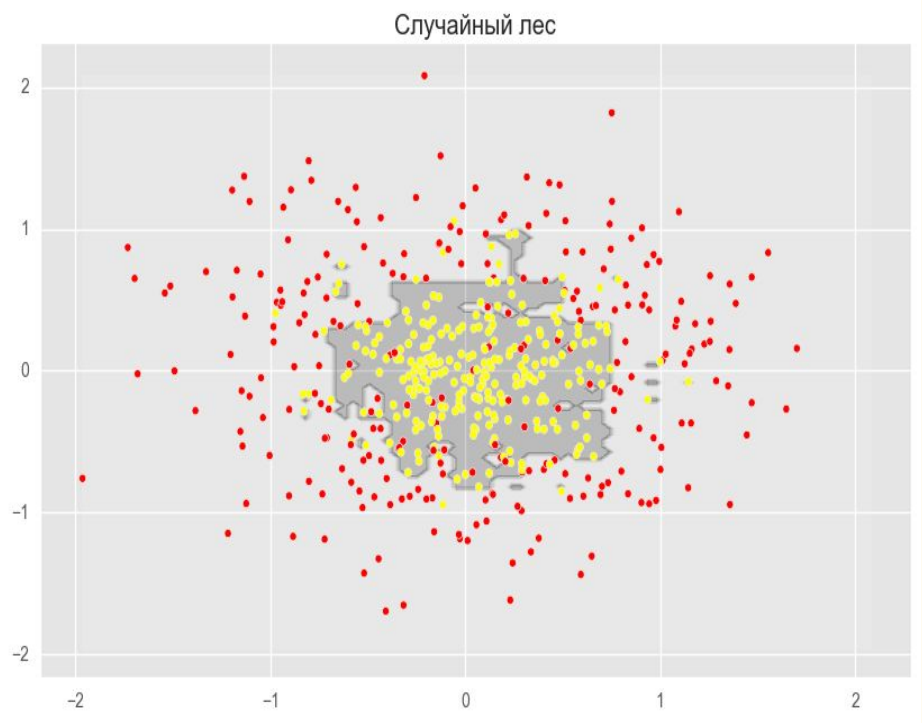
Random Forest

Случайный лес — это бэггинг над решающими деревьями, при обучении которых для каждого разбиения признаки выбираются из некоторого случайного подмножества признаков






Дерево решений
2
1
0
−1
−2
−2
−1
0
1
2


Случайный лес
2
1
0
−1
−2
−2
−1
0
1
2

3.2.4.3.1. sklearn.ensemble.RandomForestClassifier

class `sklearn.ensemble.` **`RandomForestClassifier`** (*n_estimators=10*, *criterion='gini'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *min_impurity_split=None*, *bootstrap=True*, *oob_score=False*, *n_jobs=1*, *random_state=None*, *verbose=0*, *warm_start=False*, *class_weight=None*) [source]

A random forest classifier.

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default).

Read more in the User Guide.

Parameters:

n_estimators : integer, optional (default=10)

The number of trees in the forest.

criterion : string, optional (default="gini")

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain. Note: this parameter is tree-specific.

max_features : int, float, string or None, optional (default="auto")

The number of features to consider when looking for the best split:

- If int, then consider max_features features at each split.
- If float, then max_features is a percentage and int(max_features * n_features) features are considered at each split.
- If "auto", then max_features=sqrt(n_features).
- If "sqrt", then max_features=sqrt(n_features) (same as "auto").

3.2.4.3.2. `sklearn.ensemble`.RandomForestRegressor

class `sklearn.ensemble.` **`RandomForestRegressor`** (*n_estimators=10*, *criterion='mse'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features='auto'*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *min_impurity_split=None*, *bootstrap=True*, *oob_score=False*, *n_jobs=1*, *random_state=None*, *verbose=0*, *warm_start=False*) [source]

A random forest regressor.

A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and use averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True (default).

Read more in the User Guide.

Parameters:

n_estimators : integer, optional (default=10)

The number of trees in the forest.

criterion : string, optional (default="mse")

The function to measure the quality of a split. Supported criteria are “mse” for the mean squared error, which is equal to variance reduction as feature selection criterion, and “mae” for the mean absolute error.

New in version 0.18: Mean Absolute Error (MAE) criterion.

max_features : int, float, string or None, optional (default="auto")

XGBoost

XGBoost — библиотека градиентного бустинга на деревьях решений с открытым исходным кодом.

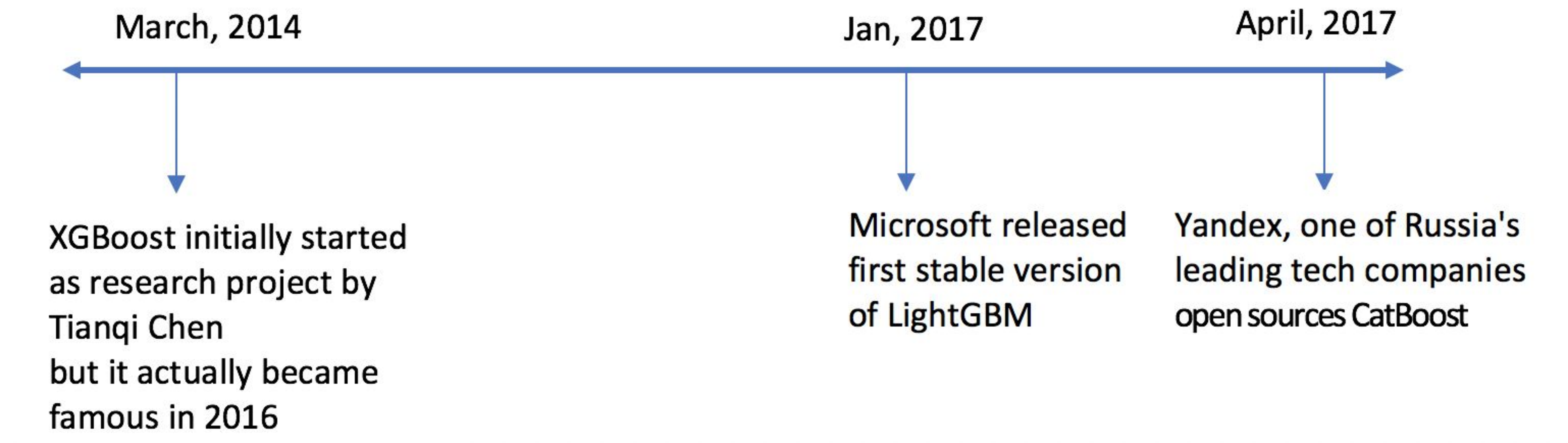
LightGBM

LightGBM — библиотека градиентного бустинга на деревьях решений с открытым исходным кодом.

CatBoost

CatBoost — библиотека градиентного бустинга на деревьях решений с открытым исходным кодом.

https://tech.yandex.ru/catboost/



https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db