

# Лекция 7

## Наивный байес, KNN. Работа с текстами. Работа с выбросами.

Юлия Конюшенко

ТГ: [@ko\\_iulia](https://t.me/ko_iulia)

[koniushenko.iun@phystech.edu](mailto:koniushenko.iun@phystech.edu)

## Лекция 1.

ML

→ обучение с учителем

обучение без учителя

- категоризация
- снижение размерн.
- визуализация

- классифр.
- регрессия
- ранжирование

## Лекция 2.

1) линейная регрессия

Обучение  $\equiv$  минимизация MSE

почему именно MSE? → есть вероятностная интерпретация

2) градиентного спуск

$$a(x) = (w, x)$$

$$MSE$$

$$w^{(k)} = w^{(k-1)} - \eta \triangleright Q(w^{(k-1)})$$

стochastic  $\rightarrow$  по 1 объекту  
mini-batch  $\rightarrow$  по батчу

## Лекция 3. Метрики качества и функционалы

$$MSE \rightarrow RMSE \rightarrow R^2$$

ошибки

$$MAE \rightarrow MSLE \rightarrow MAPE \rightarrow SMAPE$$

квантильная регрессия

онлайн / офлайн / бизнес метрики

Признаки переобучения, регуляризация

- разница качества на train/test
- большие веса

+ оптимумы MSE и

MAE

среднее медиана

$$\begin{aligned} \cdot L_2: & + \sum w_i^2 \\ \cdot L_1: & + \sum |w_i| \end{aligned}$$

## Лекция 4.

### 1) Оценивание качества модели

- отложенная выборка
- кросс-валидация

K-fold

complete

leave-one-out

K = 5

K = 7

K = 10

### 2) Способы кодирования категориальных признаков

- one-hot encoding

- стеммы
  - сжатие
  - подсет на отложенной выборке

### 3) Линейные модели классификации

$$a(x, w) = \text{sign} \left( \sum_{j=1}^n w_j x_j \right)$$

$$Q(a, X) = \frac{1}{n} \sum_{i=1}^n [M_i < 0] \rightarrow \min, \text{ где } M_i = y_i \cdot (w, x_i) - \text{отступ}$$

Для оптимизации используют верхние оценки эмпирического риска  
 Разные ф-ии потерь соответствуют различным типам моделей

Оптимизируются градиентным спуском

## Лекция 5

1. логистическая перспектива - это линейный классификатор!

$$a(x, w) = \tilde{\sigma}(w^T x), \quad \tilde{\sigma}(z) = \frac{1}{1 + e^{-z}}$$

лог. пот.

$$Q(w) = - \sum_{i=1}^e ([y_i = +1] \log(a(x_i, w)) + [y_i = -1] \log(1 - a(x_i, w)))$$

$$L(a, x) = \sum_{i=1}^e \log(1 + e^{-y_i(a(x_i, w))})$$

! логистическая функция потерь корректно предсказывает вероятности

## 2. Перцептрон

$$a(x, w) = [(w, x) > 0]$$

3. Метод опорных векторов

→ линейно разделимая выборка  
→ линейно неразделимая выборка  
2 задачи оптимизационные

безусловное задание оптимизационное:

$$\frac{1}{2} \|w\|^2 + C \sum_{i=1}^e \max(0, 1 - y_i((w, x_i) + w_0)) \rightarrow \min_{w, w_0}$$

коопротивное между линейной разделяющей поисковой и минимизирующей суммарной ошибкой

## Лекция 6

### 1) Метрики качества классификации

- accuracy
- матрица ошибок
- precision
- recall
- f-мера
- roc - curve
- pr-rec - curve,

### 2) Многоклассовая классификация

- One - vs - all
- all - vs - all
- multiclass vs multi label
- micro и macro усреднение метрик

### 3) Уловые методы

методы решения нелинейных задач классификации

# НАИВНЫЙ БАЙЕСОВСКИЙ КЛАССИФИКАТОР

# НАИВНЫЙ БАЙЕСОВСКИЙ КЛАССИФИКАТОР

**Наивный байесовский классификатор** – это алгоритм классификации, основанный на теореме Байеса с допущением о независимости признаков.

Пример: фрукт может считаться яблоком, если:

- 1) он красный
- 2) круглый
- 3) его диаметр составляет порядка 8 см

Предполагаем, что признаки вносят независимый вклад в вероятность того, что фрукт является яблоком.



# ТЕОРЕМА БАЙЕСА

Теорема Байеса:

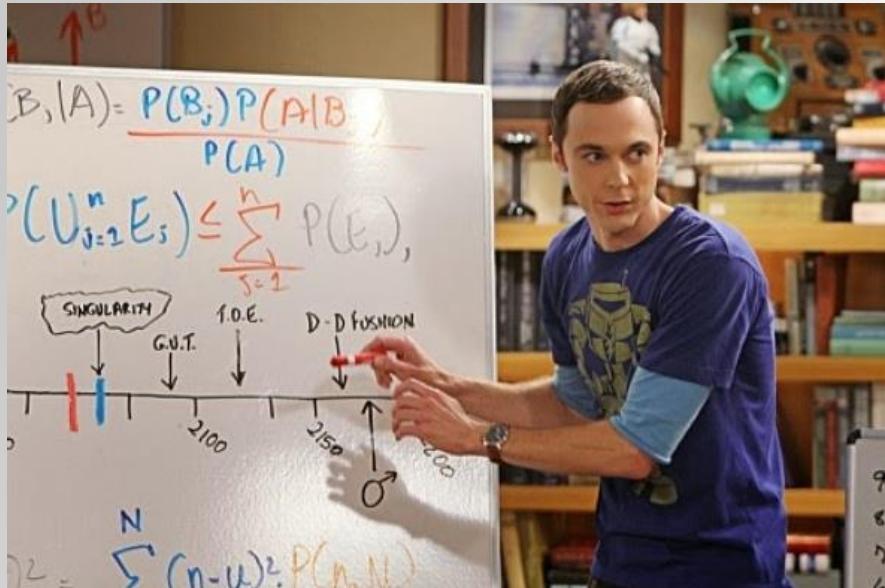
$$P(c|x) = \frac{P(x|c) \cdot P(c)}{P(x)}$$

- $P(c|x)$  - вероятность того,

что объект со значением признака  $x$

принадлежит классу  $c$ .

- $P(c)$  – априорная вероятность класса  $c$ .
- $P(x|c)$  - вероятность того, что значение признака равно  $x$  при условии, что объект принадлежит классу  $c$ .
- $P(x)$  – априорная вероятность значения признака  $x$ .



# ПРИМЕР РАБОТЫ БАЙЕСОВСКОГО АЛГОРИТМА

Пример: на основе данных о погодных условиях необходимо определить, состоится ли матч.

- Преобразуем набор данных в следующую таблицу:

Weather	No	Yes
Overcast	0	4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

# ПРИМЕР РАБОТЫ БАЙЕСОВСКОГО АЛГОРИТМА

Решим задачу с помощью теоремы Байеса:

$$P(Yes|Sunny) = P(Sunny|Yes) \cdot P(Yes) / P(Sunny)$$

Таблица частот		
Weather	No	Yes
Overcast	0	4
Rainy	3	2
Sunny	2	3
Grand Total	5	9
$=5/14$		$=9/14$
<b>0.36</b>		<b>0.64</b>
$=4/14$		<b>0.29</b>
$=5/14$		<b>0.36</b>
$=5/14$		<b>0.36</b>

- $P(Sunny|Yes) = \frac{3}{9}, P(Sunny) = \frac{5}{14}, P(Yes) = \frac{9}{14}.$
- $P(Yes|Sunny) = \frac{3}{9} \cdot \frac{9}{14} : \frac{5}{14} = \frac{3}{5} = 0,6 \Rightarrow 60\%.$

# БАЙЕСОВСКИЙ АЛГОРИТМ ДЛЯ КЛАССИФИКАЦИИ

Аналогичным образом с помощью наивного байесовского алгоритма можно прогнозировать несколько различных классов на основе множества признаков.

- + классификация быстрая и простая
- + в случае, если выполняется предположение о независимости, классификатор показывает очень высокое качество
- если в тестовых данных присутствует категория, не встречавшаяся в данных для обучения, модель присвоит ей нулевую вероятность

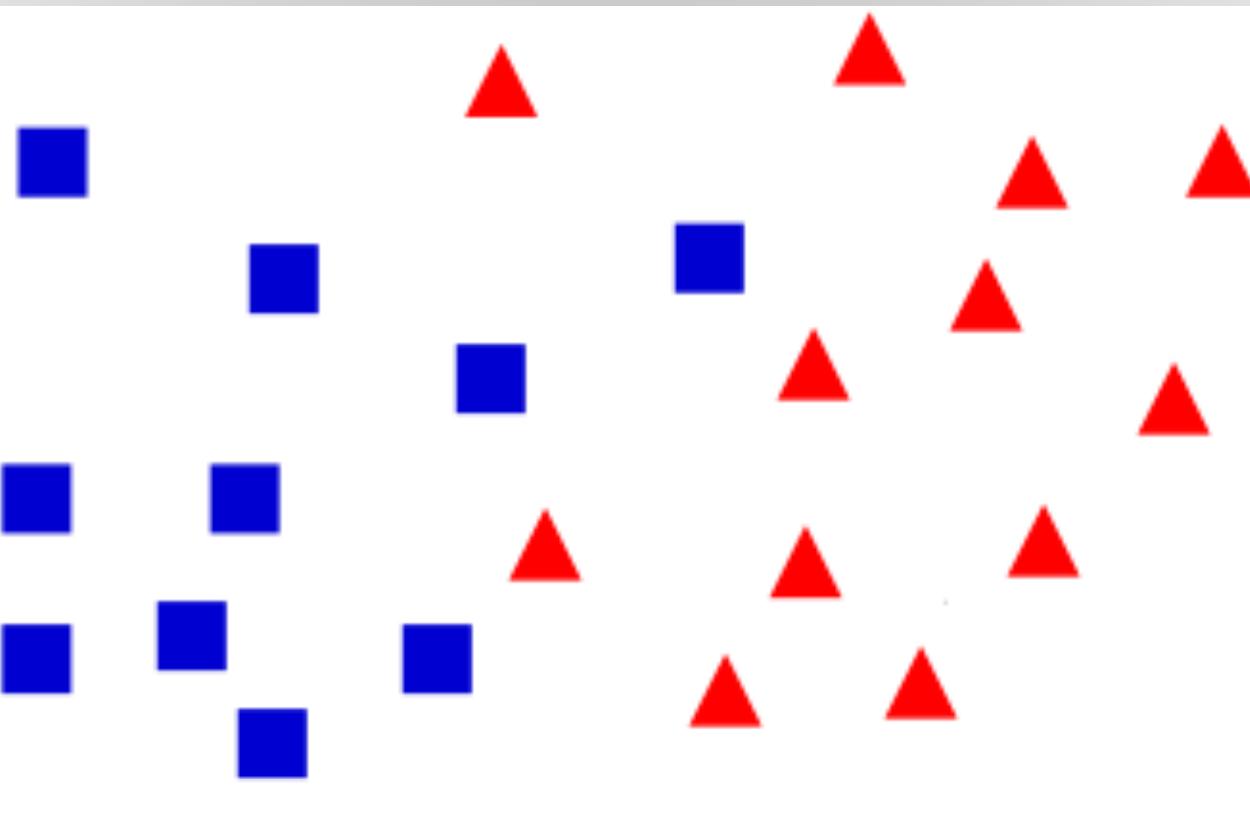
# НАИВНЫЙ БАЙЕСОВСКИЙ АЛГОРИТМ

[https://scikit-learn.org/stable/modules/naive\\_bayes.html](https://scikit-learn.org/stable/modules/naive_bayes.html)

# МЕТОД БЛИЖАЙШИХ СОСЕДЕЙ

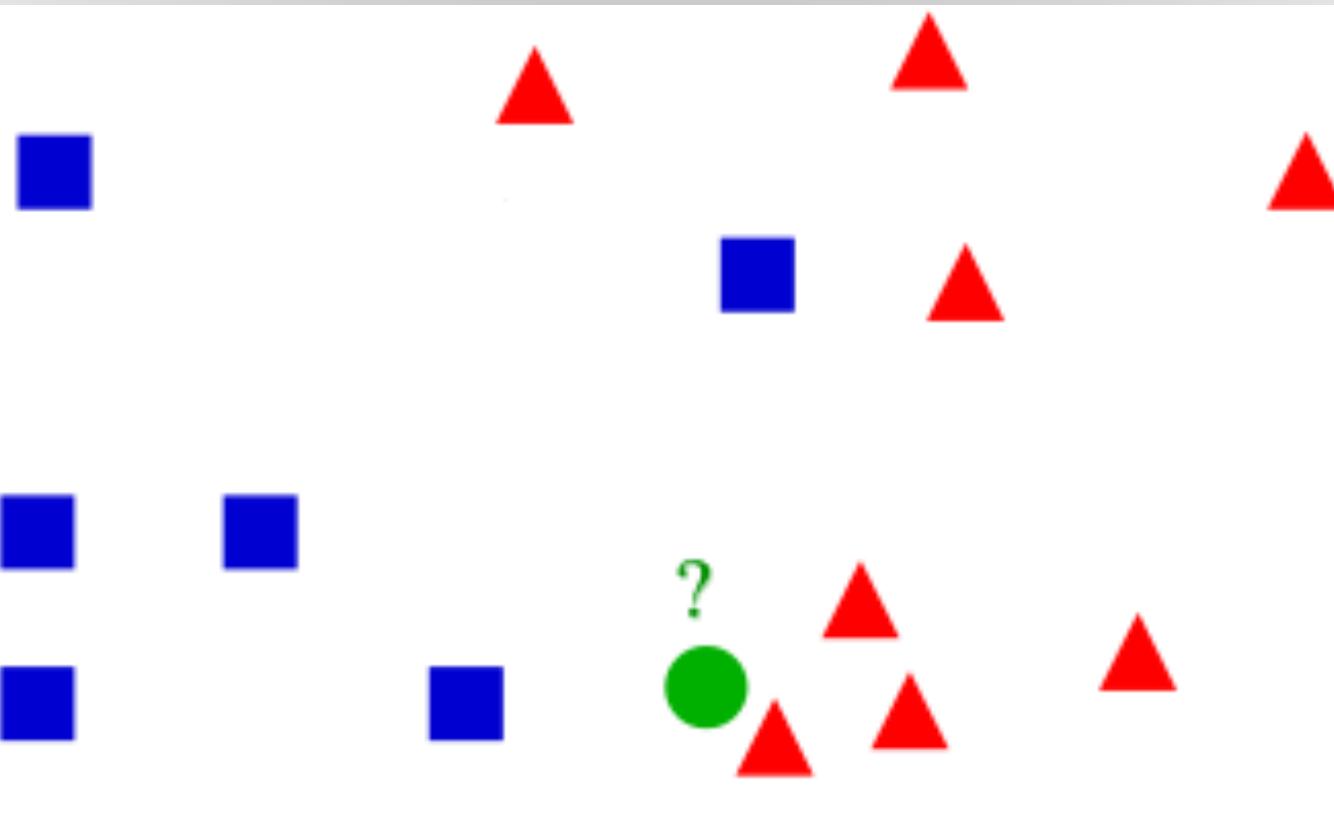
# МЕТОД БЛИЖАЙШИХ СОСЕДЕЙ

**Идея:** схожие объекты находятся близко друг к другу в пространстве признаков.



# МЕТОД БЛИЖАЙШИХ СОСЕДЕЙ

*Как классифицировать новый объект?*



# МЕТОД БЛИЖАЙШИХ СОСЕДЕЙ

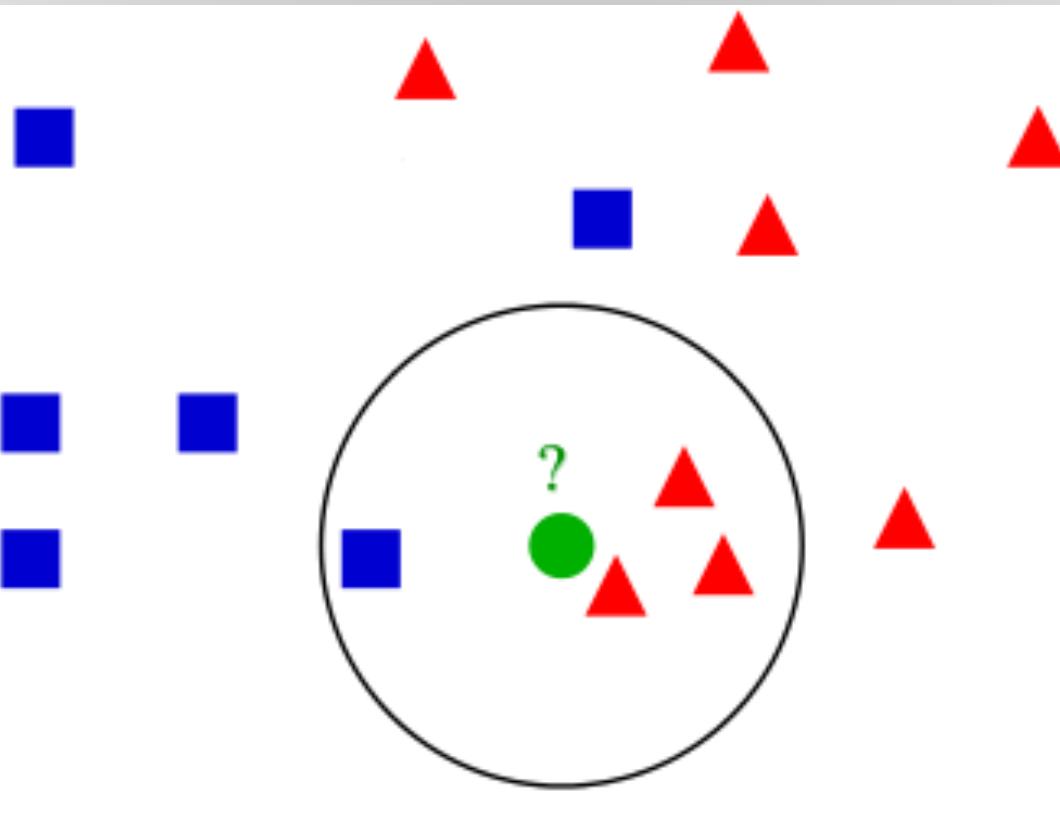
Чтобы классифицировать новый объект, нужно:

- Вычислить расстояние до каждого из объектов обучающей выборки.
- Выбрать  $k$  объектов обучающей выборки, расстояние до которых минимально.
- Класс классифицируемого объекта — это класс, наиболее часто встречающийся среди  $k$  ближайших соседей.

# МЕТОД БЛИЖАЙШИХ СОСЕДЕЙ

Число ближайших соседей  $k$  – гиперпараметр метода.

Например, для  $k = 4$  получим:



То есть объект будет отнесён к классу *треугольников*.

# ФОРМАЛИЗАЦИЯ МЕТОДА

Пусть  $k$  – количество соседей. Для каждого объекта  $u$  возьмём  $k$  ближайших к нему объектов из тренировочной выборки:

$$x_{(1;u)}, x_{(2;u)}, \dots, x_{(k;u)}.$$

Тогда класс объекта  $u$  определяется следующим образом:

$$a(u) = \operatorname{argmax}_{y \in Y} \sum_{i=1}^k [y(x_{(i;u)}) = y].$$

# ФОРМАЛИЗАЦИЯ МЕТОДА

Пусть  $k$  – количество соседей. Для каждого объекта  $u$  возьмём  $k$  ближайших к нему объектов из тренировочной выборки:

$$x_{(1;u)}, x_{(2;u)}, \dots, x_{(k;u)}.$$

Тогда класс объекта  $u$  определяется следующим образом:

$$a(u) = \operatorname{argmax}_{y \in Y} \sum_{i=1}^k [y(x_{(i;u)}) = y].$$

Ближайшие объекты – это объекты, расстояние от которых до данного объекта наименьшее по некоторой метрике  $\rho$ .

# ФОРМАЛИЗАЦИЯ МЕТОДА

Пусть  $k$  – количество соседей. Для каждого объекта  $u$  возьмём  $k$  ближайших к нему объектов из тренировочной выборки:

$$x_{(1;u)}, x_{(2;u)}, \dots, x_{(k;u)}.$$

Тогда класс объекта  $u$  определяется следующим образом:

$$a(u) = \operatorname{argmax}_{y \in Y} \sum_{i=1}^k [y(x_{(i;u)}) = y].$$

*Ближайшие объекты* – это объекты, расстояние от которых до данного объекта наименьшее по некоторой метрике  $\rho$ .

- В качестве метрики  $\rho$  как правило используют евклидово расстояние, но можно использовать и другие метрики.
- Перед использованием метода необходимо масштабировать данные, иначе признаки с большими числовыми значениями будут доминировать при вычислении расстояний.

# КАЛИБРОВКА ВЕРОЯТНОСТЕЙ

**Калибровка вероятностей** - приведение ответов алгоритма к значениям, близким к вероятностям объектов принадлежать конкретному классу.

Зачем это нужно?

- Вероятности гораздо проще интерпретировать
- Вероятности могут дать дополнительную информацию о результатах работы алгоритма

# КАЛИБРОВКА ПЛАТТА

- Пусть есть два класса,  $Y = \{+1, -1\}$

**Задача:** для классификатора  $a(x)$ , предсказывающего значения из отрезка  $[0, 1]$ , либо предсказывающего класс (+1 или -1), сделать калибровку, чтобы предсказания были вероятностями  $p(y = +1|x)$ .

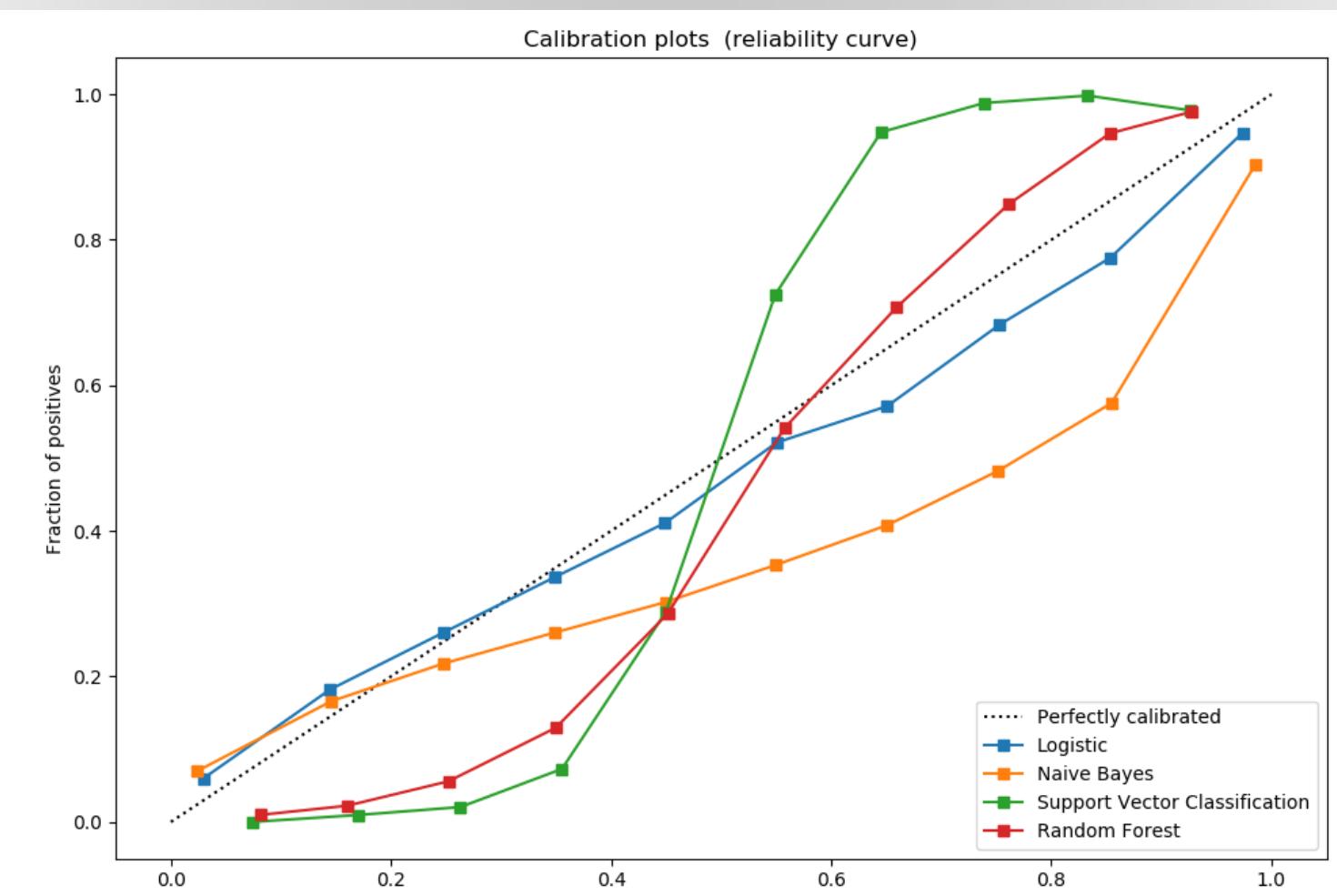
# КАЛИБРОВКА ПЛАТТА

- Пусть есть два класса,  $Y = \{+1, -1\}$

**Задача:** для классификатора  $a(x)$ , предсказывающего значения из отрезка  $[0, 1]$ , либо предсказывающего класс (+1 или -1), сделать калибровку, чтобы предсказания были вероятностями  $p(y = +1|x)$ .

**Идея:** обучаем логистическую регрессию на ответах классификатора  $a(x)$ .

# ПРИМЕР ИЗ SKLEARN



# КАЛИБРОВКА ПЛАТТА

- Пусть есть два класса,  $Y = \{+1, -1\}$

**Задача:** для классификатора  $a(x)$ , предсказывающего значения из отрезка  $[0, 1]$ , либо предсказывающего класс (+1 или -1), сделать калибровку, чтобы предсказания были вероятностями  $p(y = +1|x)$ .

**Идея:** *обучаем логистическую регрессию на ответах классификатора  $a(x)$ .*

# КАЛИБРОВКА ПЛАТТА

- Пусть есть два класса,  $Y = \{+1, -1\}$

**Задача:** для классификатора  $a(x)$ , предсказывающего значения из отрезка  $[0, 1]$ , либо предсказывающего класс (+1 или -1), сделать калибровку, чтобы предсказания были вероятностями  $p(y = +1|x)$ .

**Идея:** *обучаем логистическую регрессию на ответах классификатора  $a(x)$ .*

- $\pi(x; \alpha; \beta) = \sigma(\alpha \cdot a(x) + \beta) = \frac{1}{1+e^{-(\alpha \cdot a(x) + \beta)}}$

# КАЛИБРОВКА ПЛАТТА

- Пусть есть два класса,  $Y = \{+1, -1\}$

**Задача:** для классификатора  $a(x)$ , предсказывающего значения из отрезка  $[0, 1]$ , либо предсказывающего класс (+1 или -1), сделать калибровку, чтобы предсказания были вероятностями  $p(y = +1|x)$ .

**Идея:** *обучаем логистическую регрессию на ответах классификатора  $a(x)$ .*

- $\pi(x; \alpha; \beta) = \sigma(\alpha \cdot a(x) + \beta) = \frac{1}{1+e^{-(\alpha \cdot a(x) + \beta)}}$
- Находим  $\alpha$  и  $\beta$ , минимизируя логистическую функцию потерь (*то есть обучаем логистическую регрессию*):

$$-\sum_{y_i=-1} \log(1 - \pi(x; \alpha; \beta)) - \sum_{y_i=+1} \log(\pi(x; \alpha; \beta)) \rightarrow \min_{\alpha, \beta}$$

# ТЕРМИНОЛОГИЯ

- **документ** = текст
- **корпус** – набор документов
- **токен** – формальное определение “слова”; токен может не иметь смыслового значения (например, “12fdh” или “авыдшл”), но обычно отделен от остальных токенов пробелами или знаками препинания

# ТОКЕНИЗАЦИЯ ТЕКСТА

Чтобы работать с текстом, необходимо разбить его на токены. В простейшем случае токены – это слова (а также наборы букв, знаки препинания и т.д.).

Text  
“The cat sat on the mat.”



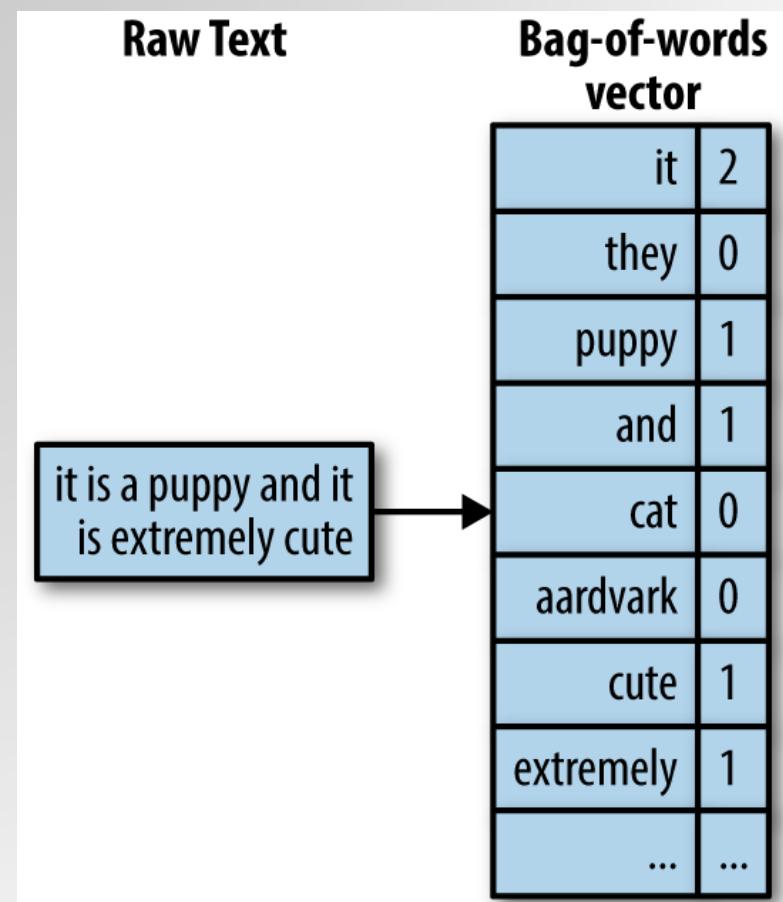
Tokens

“the”, “cat”, “sat”, “on”, “the”, “mat”, “.”

# МЕТОДЫ КОДИРОВАНИЯ ТЕКСТОВЫХ ДАННЫХ

# BAG OF WORDS (МЕШОК СЛОВ)

- По корпусу создадим словарь из всех встречающихся в нем слов (можно убрать общеупотребительные часто встречающиеся слова и очень редкие слова).
- Каждое слово закодируем вектором, в котором стоит единица на месте, соответствующем месту этого слова в словаре, все остальные компоненты вектора – 0.
- Для кодирования документа сложим коды всех его слов.



# BAG OF WORDS (ПРИМЕР)

Пусть корпус состоит из следующих документов:

- D1 - “I am feeling very happy today”
- D2 - “I am not well today”
- D3 - “I wish I could go to play”

Кодировка этих документов будет такой:

	I	am	feeling	very	happy	today	not	well	wish	could	go	to	play
D1	1	1	1	1	1	1	0	0	0	0	0	0	0
D2	1	1	0	0	0	1	1	1	0	0	0	0	0
D3	2	0	0	0	0	0	0	0	1	1	1	1	1

Какие проблемы есть у этого подхода?

# BAG OF WORDS

Используя *bag of words (BOW)*, мы теряем информацию о порядке слов в документе.

Пример: векторы документов “I have no cats” и “No, I have cats” будут идентичны.

+ часто встречающиеся слова будут меняться

# N-GRAM BAG OF WORDS

В качестве слов в словаре можно использовать:

- N-граммы из букв (наборы букв длины N в слове)
- N-граммы из слов (наборы фраз длины N в документе)

*Такой подход поможет учесть сходственные слова и опечатки.*

$N = 1$  : This is a sentence

unigrams:  
this,  
is,  
a,  
sentence

$N = 2$  : This is a sentence

bigrams:  
this is,  
is a,  
a sentence

$N = 3$  : This is a sentence

trigrams:  
this is a,  
is a sentence

## TF-IDF

- слова, которые редко встречаются в корпусе, но присутствуют в документе, могут оказаться важными для характеристики документа.
- слова, которые встречаются во всех документах, наоборот, не важны.

## TF-IDF

**Tf-Idf (term frequency – inverse document frequency):**

- $tf(t, d)$  - частота вхождения слова  $t$  в документ  $d$ :

$$tf(t, d) = \frac{n_t}{\sum_k n_k} = \frac{\text{число вхождений слова } t \text{ в документ}}{\text{общее число слов в документе}}$$

$tf(t, d)$  показывает важность слова  $t$  в документе  $d$ .

## TF-IDF

- $tf(t, d)$  - частота вхождения слова  $t$  в документ  $d$ :

$$tf(t, d) = \frac{n_t}{\sum_k n_k} = \frac{\text{число вхождений слова } t \text{ в документ}}{\text{общее число слов в документе}}$$

$tf(t, d)$  показывает важность слова  $t$  в документе  $d$ .

- $idf(t, D)$  - величина, обратная частоте, с которой слово  $t$  встречается в документах корпуса  $D$ .

$$idf(t, D) = \log \frac{|D|}{|\{d_i \in D \mid t \in d_i\}|},$$

$|D|$  – число документов в корпусе,

$|\{d_i \in D \mid t \in d_i\}|$  - число документов, в которых встречается слово  $t$

Учёт  $idf$  уменьшает вес часто используемых в корпусе слов.

## TF-IDF

Tf-idf слова  $t$  в документе  $d$  из корпуса  $D$ :

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D),$$

Пример:

Дана коллекция  $D$  из  $10000000 = 10^7$  документов, в 1000 из них встречается слово “заяц”. В данном документе  $d$  из коллекции 100 слов, и слово “заяц” встречается 3 раза.

$$tf(\text{заяц}, d) = \frac{3}{100} = 0,03$$

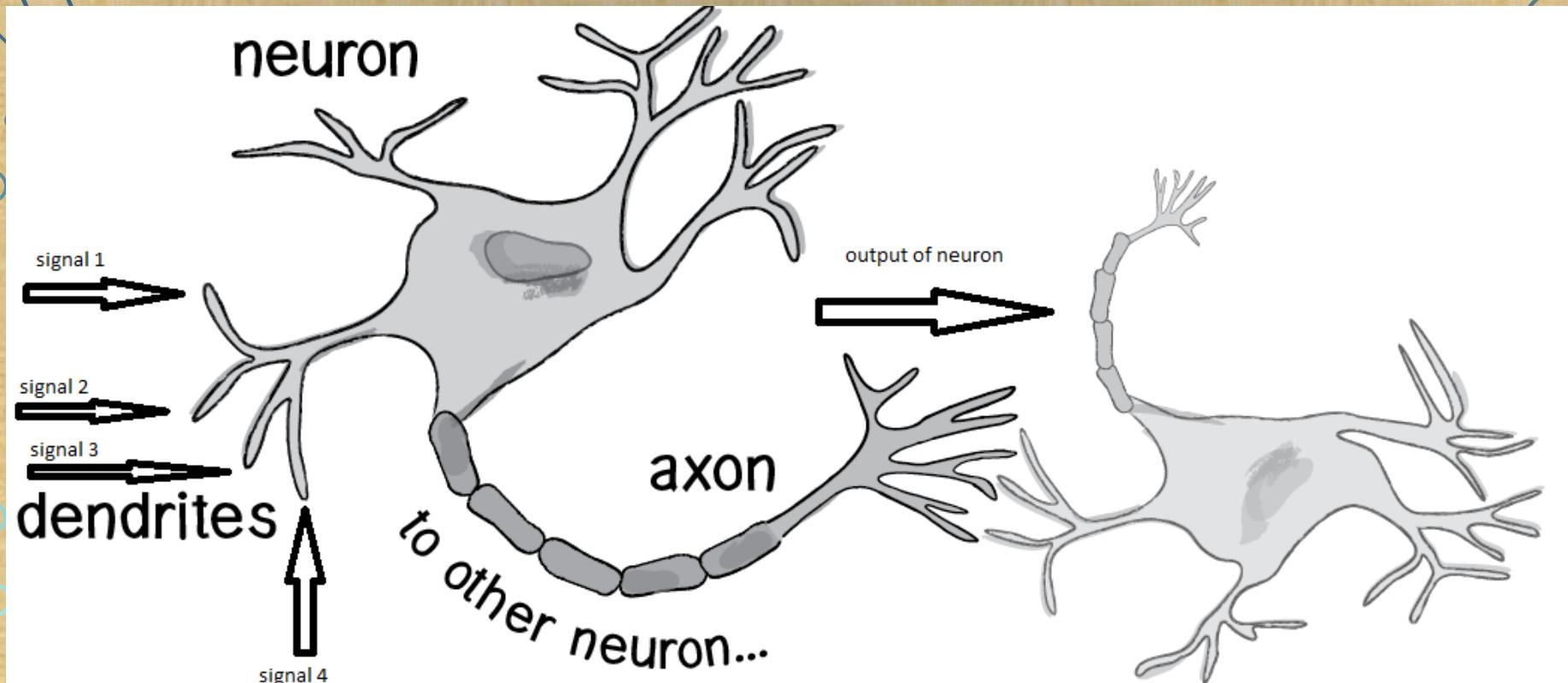
$$idf(\text{заяц}, D) = \log\left(\frac{10^7}{10^3}\right) = 4$$

Поэтому  $tfidf(\text{заяц}, d, D) = 0,03 \cdot 4 = 0,12$ .

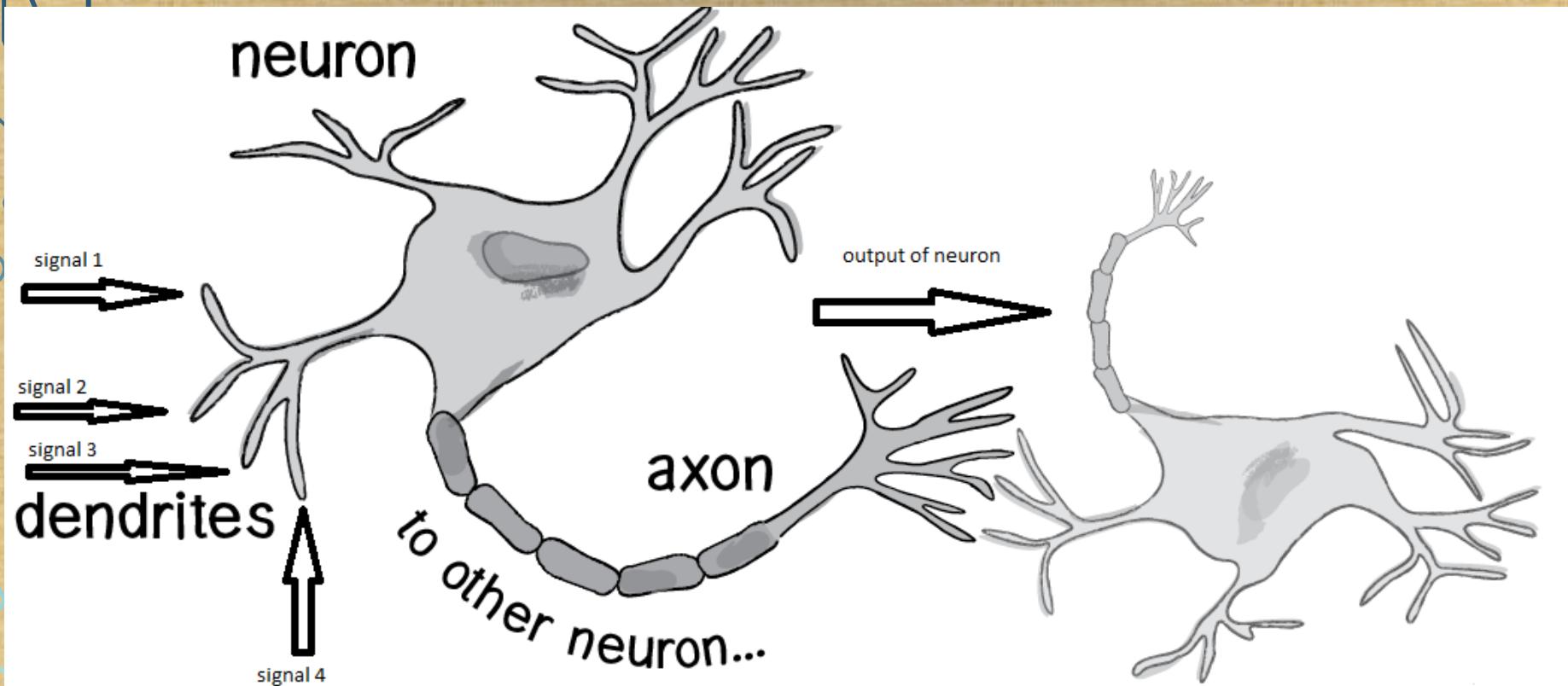
может ли  
tf-idf сработать  
лучше, чем  
bag of words?

# НЕЙРОННЫЕ СЕТИ (ЗАБЕГАЯ ВПЕРЕД...)

## СХЕМА НЕЙРОНА В МОЗГЕ



# СХЕМА НЕЙРОНА В МОЗГЕ



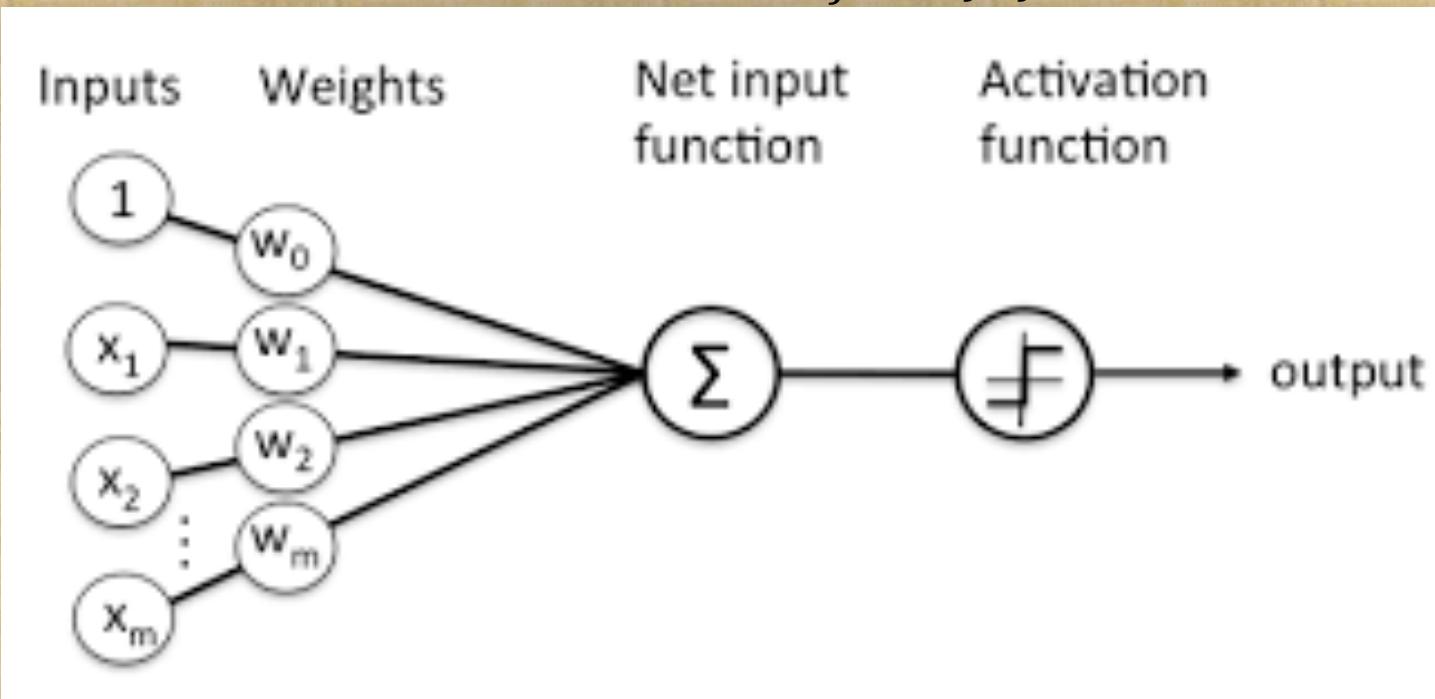
Линейная модель:

$$a(x, w) = \sigma(w_0 + \sum_{j=1}^n w_j x_j),$$

$\sigma$  – функция активации

# МОДЕЛЬ НЕЙРОНА (ОДНОСЛОЙНАЯ НЕЙРОННАЯ СЕТЬ)

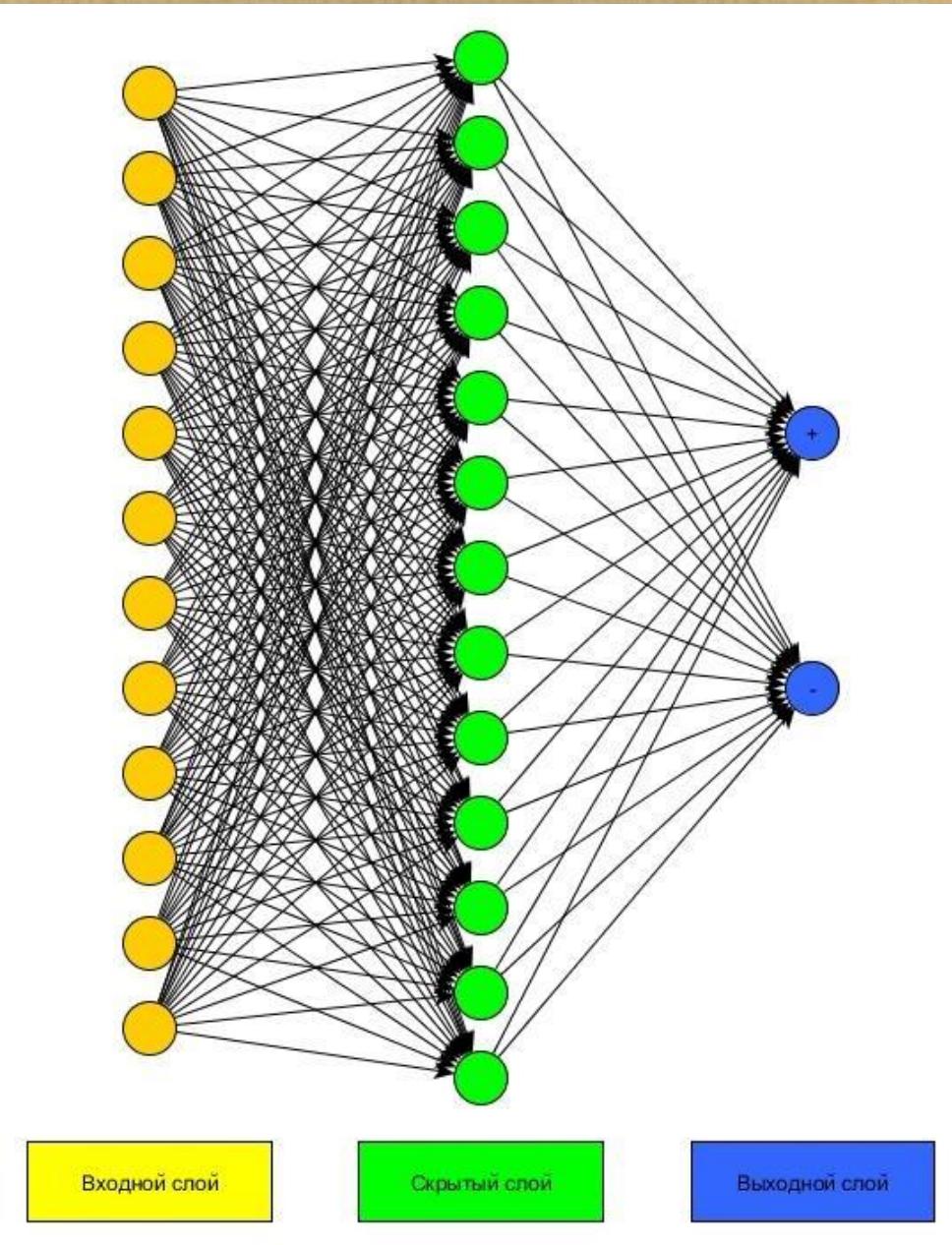
$$a(x, w) = \sigma(w_0 + \sum_{j=1}^n w_j x_j)$$



Функции активации:

- $\sigma(z) = sign(z) \Rightarrow a(x, w) = sign[w_0 + \sum_{j=1}^n w_j x_j]$
- $\sigma(z) = \frac{1}{1+\exp(-z)} \Rightarrow a(x, w) = \frac{1}{1+\exp(-(w_0 + \sum_{j=1}^n w_j x_j))}$

# ДВУХСЛОЙНАЯ НЕЙРОННАЯ СЕТЬ



# ЗАДАЧА ОПТИМИЗАЦИИ

Решаем задачу ( $L$  – функция потерь на объекте)

$$Q(w) = \sum_{i=1}^l L(w; x_i; y_i) \rightarrow \min_w$$

Градиентный спуск:

- Инициализируем  $w$ :  $w^{(0)}$
- На  $N$ -м шаге:  $w^{(N)} := w^{(N-1)} - \eta \nabla L(w^{(N-1)}; x; y)$

# WORD2VEC

Цель: для каждого слова из текста получить такой числовой вектор, чтобы векторы похожих по смыслу слов были “близки”.

# WORD2VEC

Цель: для каждого слова из текста получить такой числовой вектор, чтобы векторы похожих по смыслу слов были “близки”.

Идея: слова, встречающиеся в похожих контекстах, имеют близкие значения.

- В 2013 году Томас Миколов и его коллеги предложили word2vec – нейронную сеть, которую можно быстро обучить на огромном объеме текстов для получения векторов слов.

# СТРУКТУРА АЛГОРИТМА

- Цель: получить числовые векторы слов
- Как измерить расстояние между словами/текстами?
- Функция потерь?
- Признаки и целевая переменная?
- Алгоритм?
- Где взять результат?

# ВЕРОЯТНОСТНАЯ ПОСТАНОВКА ЗАДАЧИ

- будем предсказывать вероятность слова по его окружению (контексту)
- будем учить такие вектора слов, чтобы *вероятность, присваиваемая моделью слову была близка к вероятности встретить это слово в этом окружении в реальном тексте (softmax)*:

$$p(w_0|w_c) = \frac{e^{\rho(w_0, w_c)}}{\sum_{w_i \in V} e^{\rho(w_i, w_c)}},$$

где  $w_0$  - вектор целевого слова,  $w_c$  - вектор контекста,  
 $\rho(w_i, w_j)$  – расстояние между словами  $w_i$  и  $w_j$ .

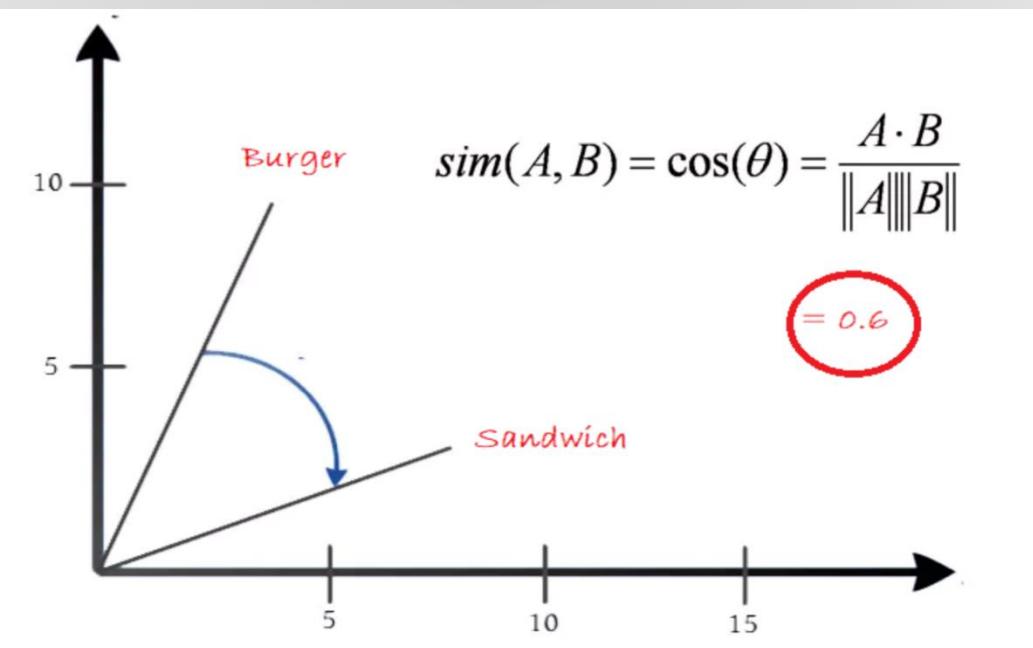
# КОСИНУСНОЕ РАССТОЯНИЕ

- Скалярное произведение векторов  $x$  и  $y$ :

$$(x, y) = \|x\| \cdot \|y\| \cdot \cos(x, y)$$

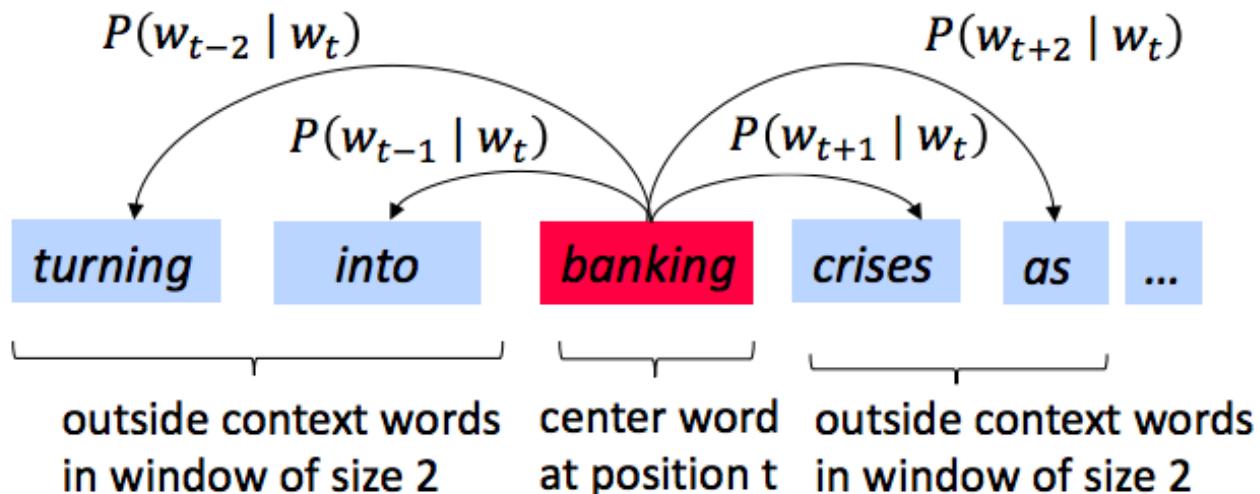
*В качестве расстояния между словами используется косинусное расстояние:*

$$\rho(w_i, w_j) = \frac{(w_i, w_j)}{\|w_i\| \cdot \|w_j\|}$$



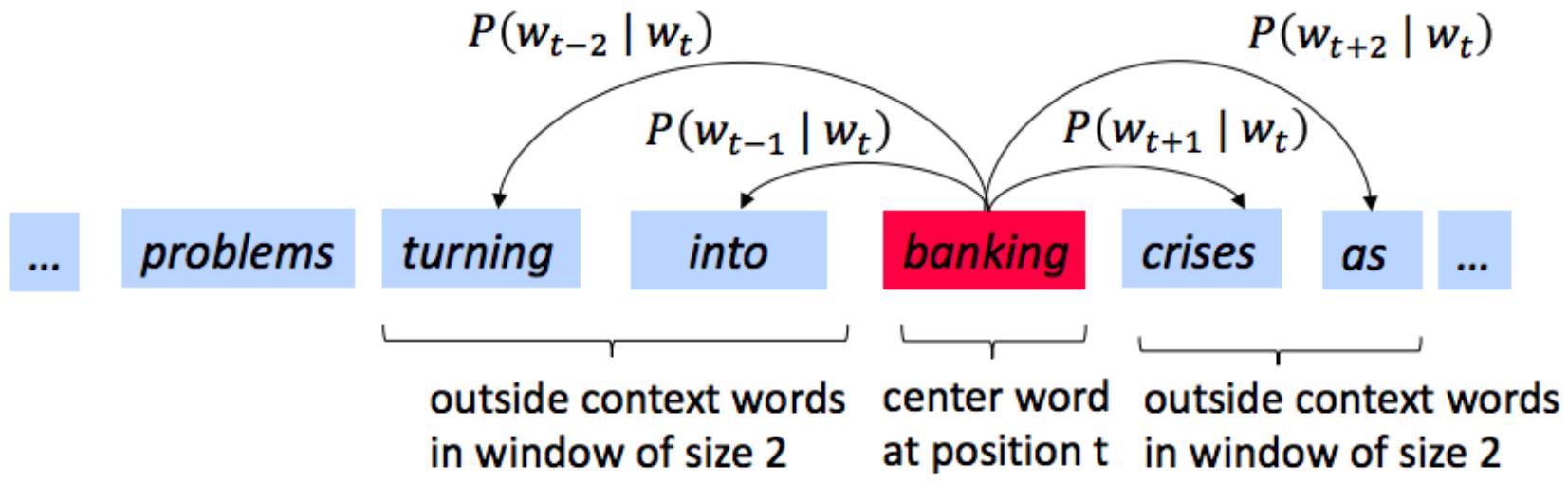
# ПРАВДОПОДОБИЕ

Для каждого слова из контекста можно вычислить вероятности:



# ПРАВДОПОДОБИЕ

Для каждого слова из контекста можно вычислить вероятности:



Тогда правдоподобие для текста длины  $n$  будет выглядеть так (2C – размер контекста):

$$\prod_{j=1}^n \prod_{\substack{k=-C \\ k \neq 0}}^C p(w_{j+k} | w_j)$$

# ФУНКЦИЯ ПОТЕРЬ

Для набора текстов  $\{x_1, \dots, x_l\}$ , где текст  $x_i$  имеет длину  $n_i$ , можно определить *логарифм правдоподобия*:

$$\sum_{i=1}^l \sum_{j=1}^{n_i} \sum_{\substack{k=-C, \\ k \neq 0}}^C \log p(w_{j+k} | w_j) \rightarrow \max_{\{w\}}$$

Тогда функция потерь, которую алгоритм будет минимизировать в процессе обучения:

$$-\sum_{i=1}^l \sum_{j=1}^{n_i} \sum_{\substack{k=-C, \\ k \neq 0}}^C \log p(w_{j+k} | w_j) \rightarrow \min_{\{w\}}$$

# СТРУКТУРА АЛГОРИТМА

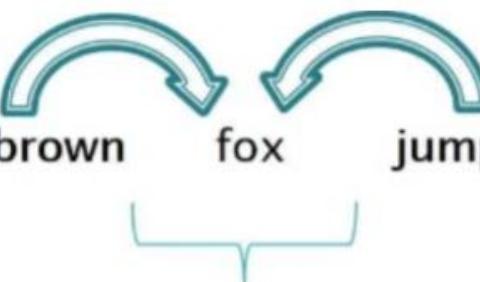
- Цель: получить числовые векторы слов
- Расстояние между словами/текстами: косинусное расстояние
- Функция потерь: минус логарифм правдоподобия
- Признаки и целевая переменная?
- Алгоритм?
- Где взять результат?

# WORD2VEC

Есть две разные модели word2vec – CBOW и Skip-gram.

- CBOW (Continuous Bag of Words) предсказывает вероятность слова по данному контексту
- Skip-gram (“словосочетание с пропуском”) предсказывает по данному слову вектор контекста

**CBOW:** The quick brown fox jumps over the lazy dog



**Skip-gram:** The

quick brown fox jumps over

предсказываются  
контекстные слова

the lazy dog

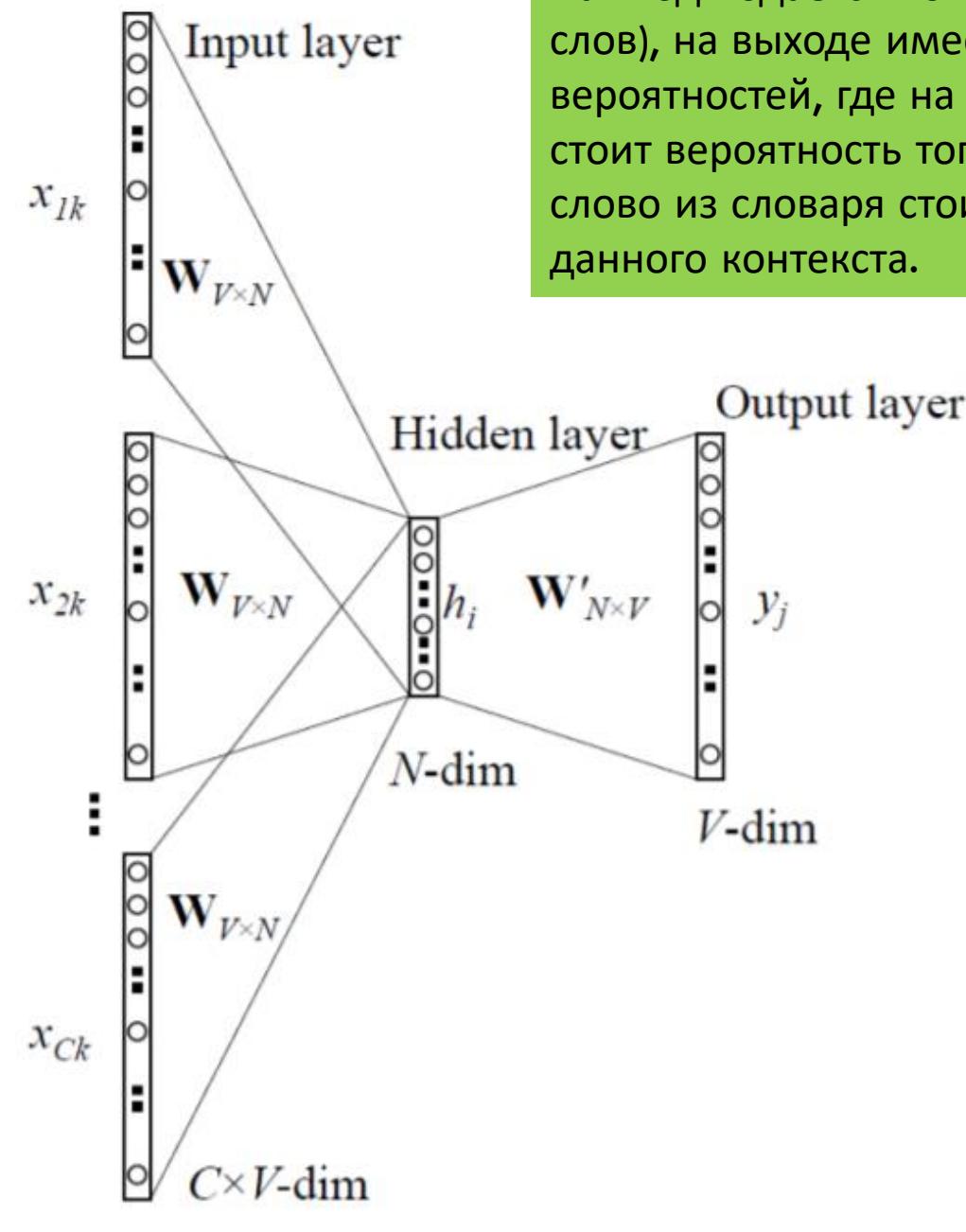
предсказываются  
контекстные слова

# МОДЕЛЬ СВОУ

- Обучаем модель с помощью *нейронной сети*.
- Скользящим окном ширины  $2C + 1$  (слово и его контекст) проходим по всей коллекции
- **Вход (признаки):** one-hot представление слов контекста (векторы длины  $V$ )
- **Ответ (целевая переменная):** one-hot представление слова
- **Выход:** распределение вероятностей на словах коллекции (вектор длины  $V$ )
- **Оптимизируемый функционал – минус логарифм правдоподобия:**

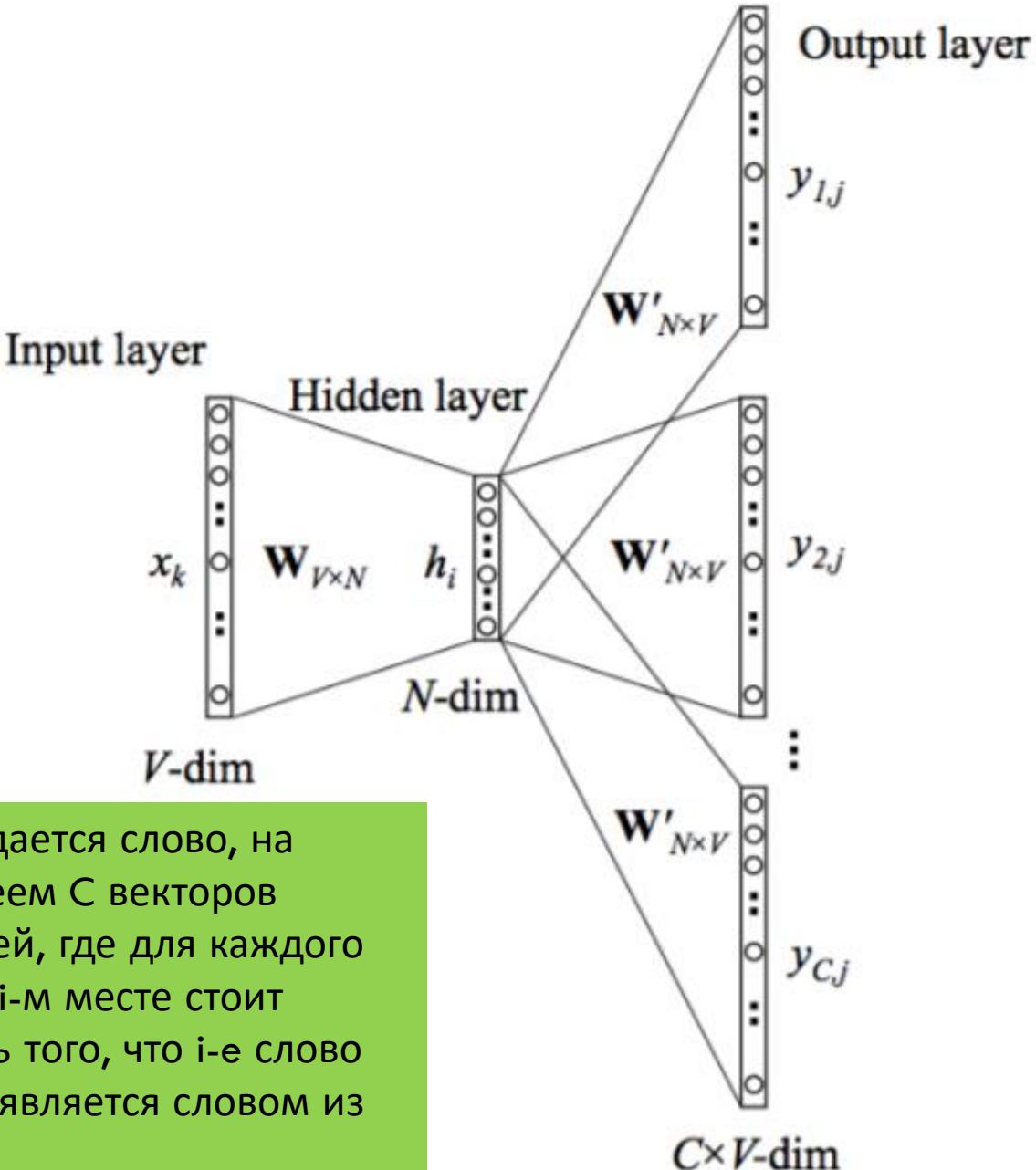
$$-\sum_{i=1}^l \sum_{j=1}^{n_i} \sum_{\substack{k=-C, \\ k \neq 0}}^C \log p(w_{j+k} | w_j) \rightarrow \min_w$$

# МОДЕЛЬ СВОУ



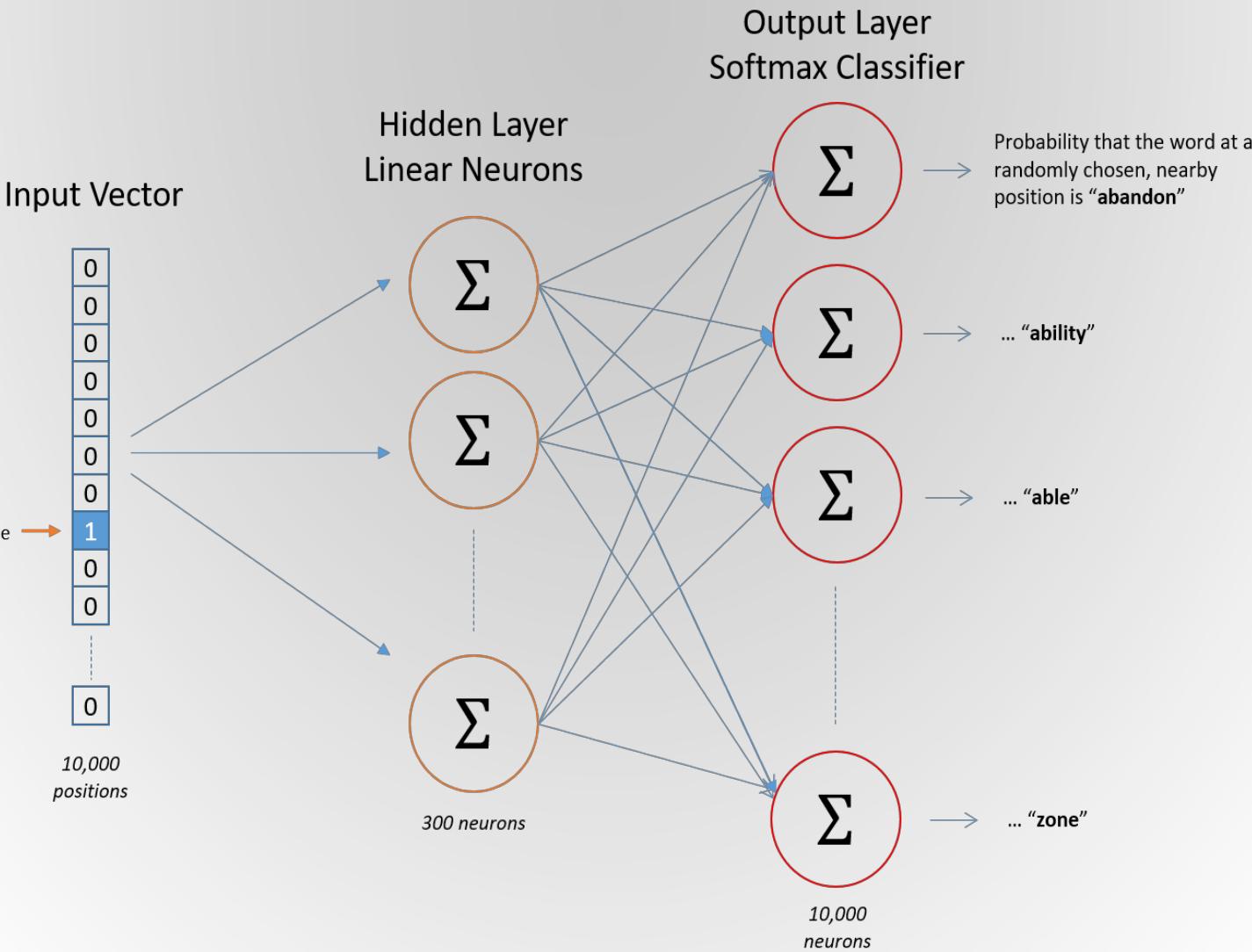
На вход подается контекст (С слов), на выходе имеем вектор вероятностей, где на  $i$ -м месте стоит вероятность того, что  $i$ -е слово из словаря стоит внутри данного контекста.

# МОДЕЛЬ SKIP-GRAM



# МОДЕЛЬ SKIP-GRAM

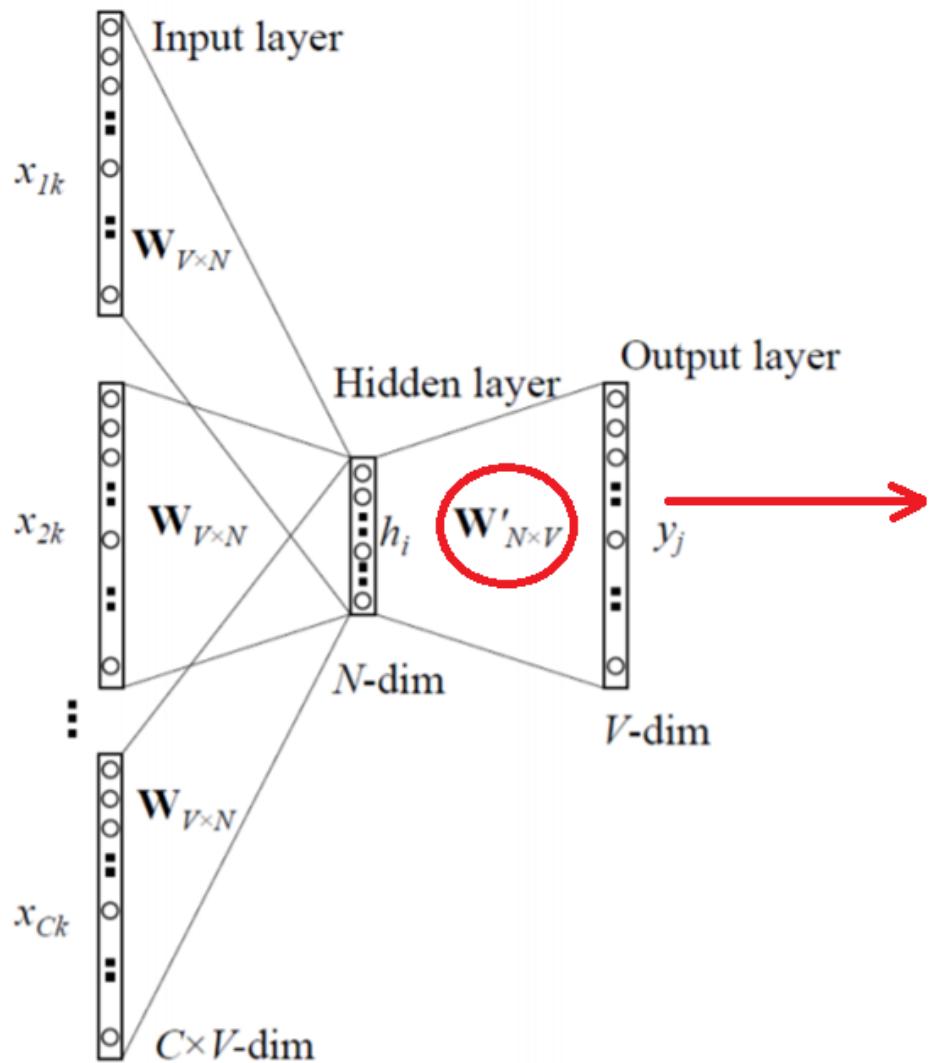
На скрытом слое активации нет, на выходном слое – softmax.



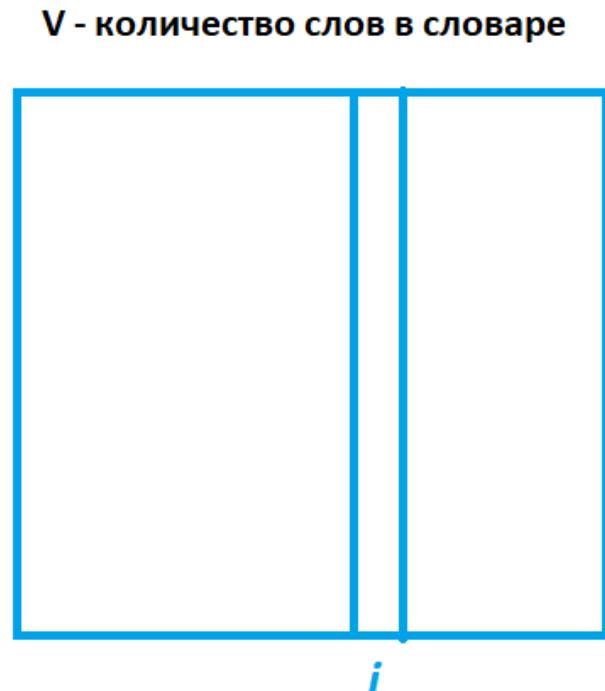
# СТРУКТУРА АЛГОРИТМА

- Цель: получить числовые векторы слов
- Расстояние между словами/текстами: косинусное расстояние
- Функция потерь: минус логарифм правдоподобия
- Признаки: one-hot представления слов контекста, целевая переменная: one-hot представление слова
- Алгоритм: полносвязная нейронная сеть с одним скрытым слоем (на внутреннем слое нет функции активации, на выходном – softmax)
- Где взять результат?

# ВЕКТОРЫ СЛОВ



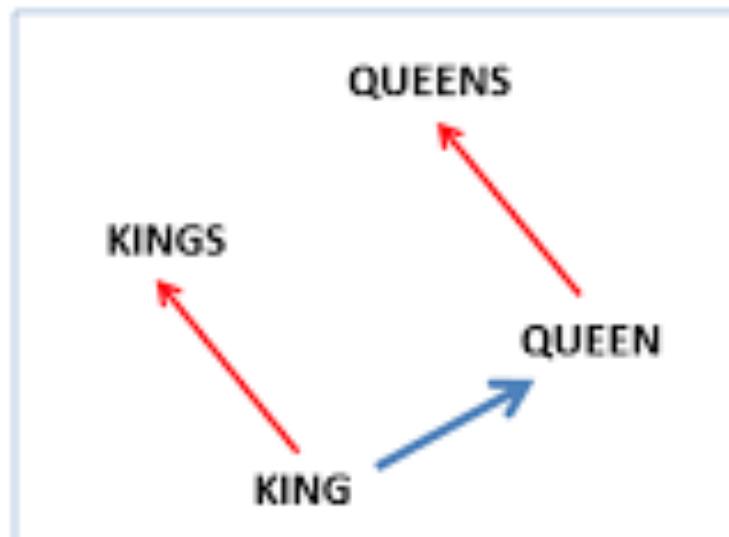
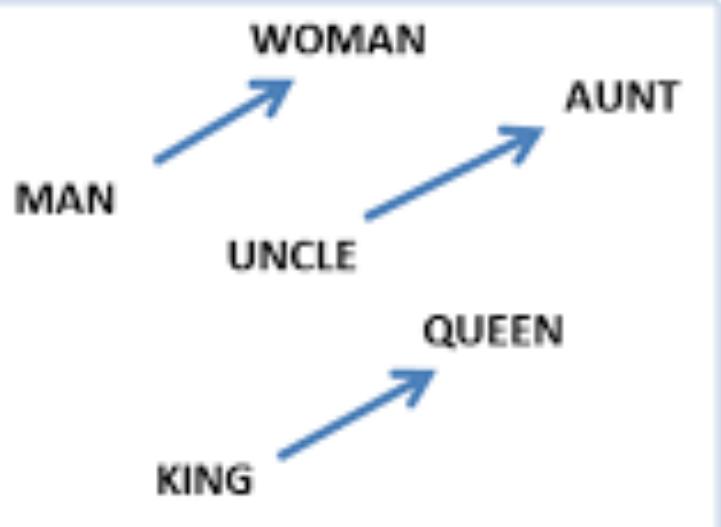
$N$  - количество нейронов на скрытом слое



Числа, стоящие в  $i$ -м столбце в полученной матрице весов – это word2vec-вектор длины  $N$ , представляющий  $i$ -е слово из словаря.

# ВЕКТОРНЫЕ СООТНОШЕНИЯ МЕЖДУ СЛОВАМИ

За счёт использования косинусного расстояния между векторами слов, к векторам слов, полученных в результате применения word2vec, можно применять векторные операции сложения и вычитания, которые будут иметь смысл:



# СТРУКТУРА АЛГОРИТМА

- Цель: получить числовые векторы слов
- Расстояние между словами/текстами: косинусное расстояние
- Функция потерь: минус логарифм правдоподобия
- Признаки: one-hot представления слов контекста, целевая переменная: one-hot представление слова
- Алгоритм: полносвязная нейронная сеть с одним скрытым слоем (на внутреннем слое нет функции активации, на выходном – softmax)
- Word2vec-векторы слов – это числа, стоящие в столбцах матрицы весов на выходе из скрытого слоя обученной нейронной сети.

# ТРАНСФОРМЕРЫ

- В октябре **2018** года Google выпустила модель кодирования текстовых данных под названием **BERT** –

*Bidirectional Encoder Representations from Transformers.*

- Такой способ кодировать тексты даёт **state-of-the-art** результаты во многих задачах машинного обучения, связанных с обработкой естественного языка.

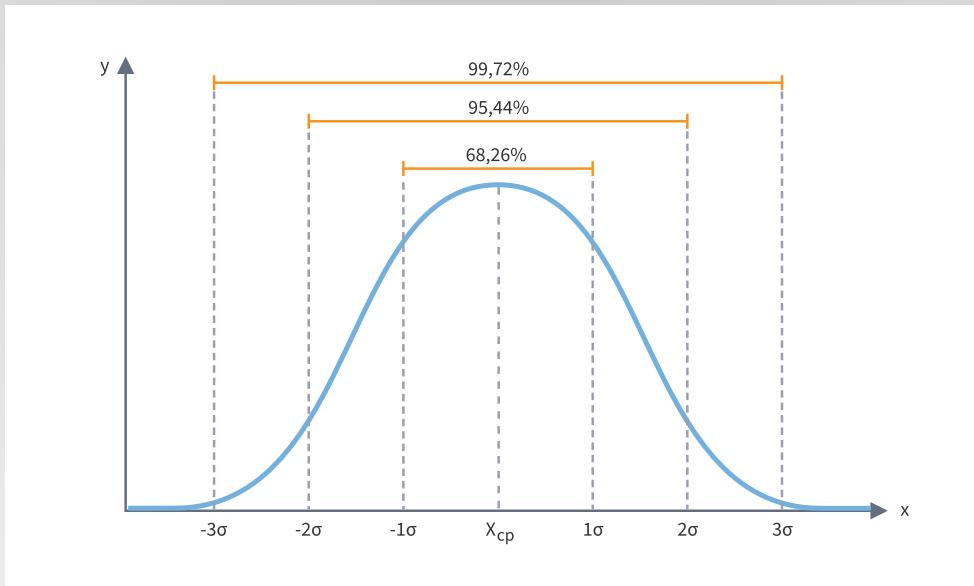
Почитать про трансформеры и механизм **attention** можно [здесь](#).

# РАБОТА С ВЫБРОСАМИ

1. Статистические методы (правило трех сигм, интерквартильный размах).
2. Методы машинного обучения.

# 1. ПРАВИЛО ТРЕХ СИГМ

- Для случайных величин, распределенных по нормальному закону, вероятность того, что случайная величина отклонится от своего математического ожидания более чем на три стандартных отклонения, практически равна нулю.

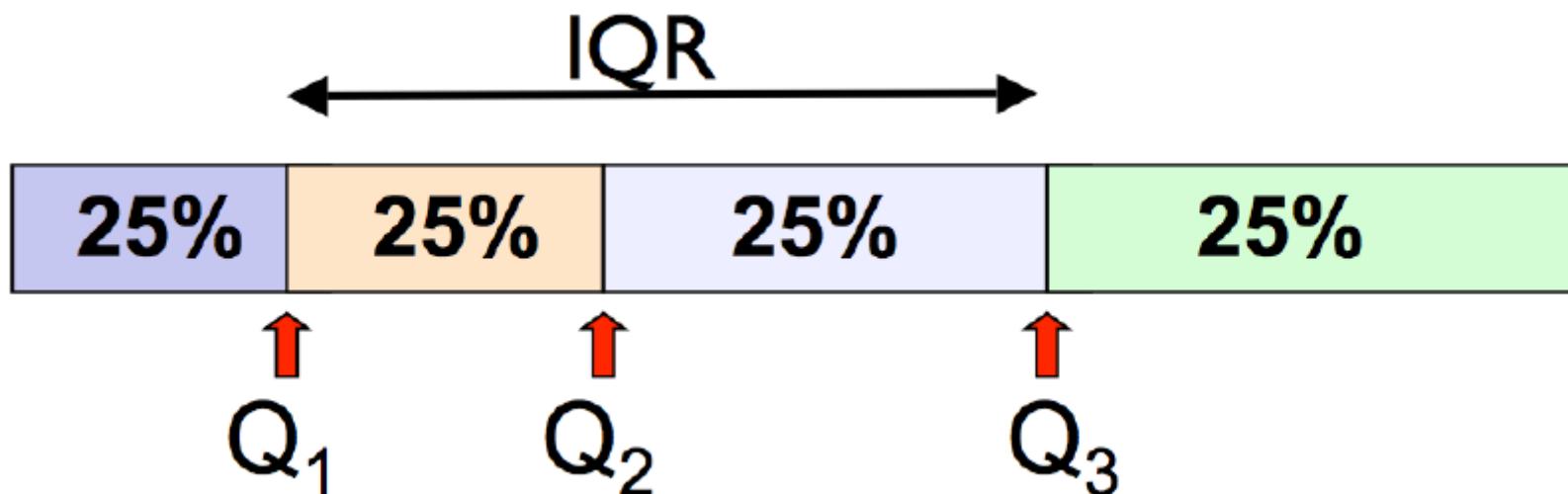


- Выбросами объявляются объекты, имеющие стандартное отклонение  $\geq 3\sigma$  от математического ожидания.

## 2. ИНТЕРКВАРТИЛЬНЫЙ РАЗМАХ

Пусть  $Q_1$  – первая (25%) квартиль распределения,  
 $Q_3$  – третья (75%) квартиль распределения.

- Величина  $IQR = Q_3 - Q_1$  называется **интерквартильным размахом**.



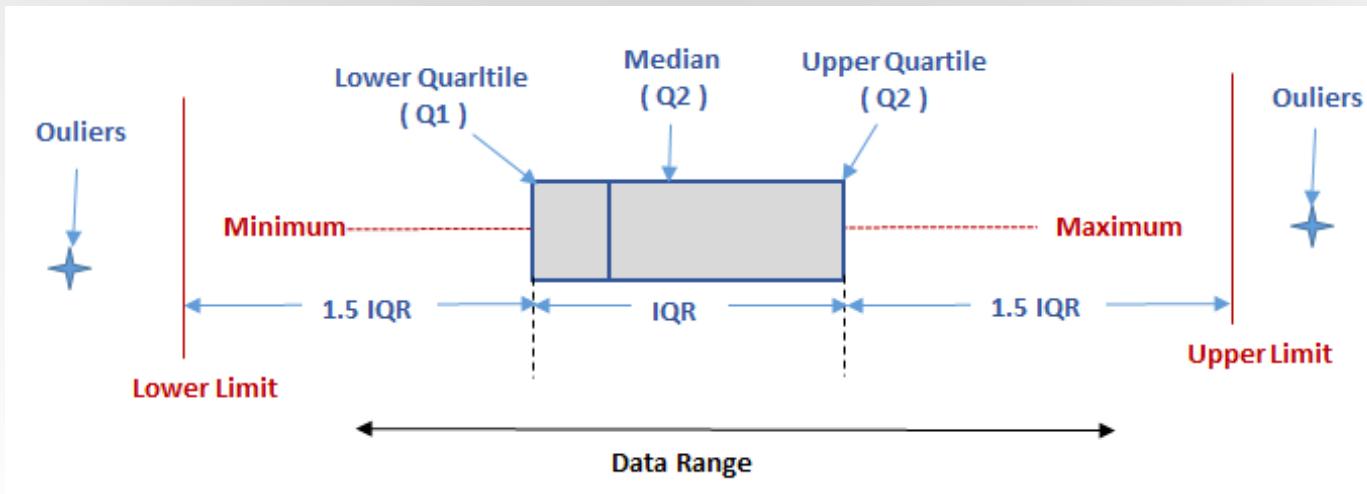
## 2. ИНТЕРКВАРТИЛЬНЫЙ РАЗМАХ

- **Слабые выбросы** – это значения, которые меньше 25%-квартили минус 1,5IQR или больше 75%-квартили плюс 1,5IQR:

$$x < Q1 - 1,5 \cdot IQR \text{ или } x > Q3 + 1,5 \cdot IQR$$

- **Сильные выбросы** – это значения, которые меньше 25%-квартили минус 3IQR или больше 75%-квартили плюс 3IQR:

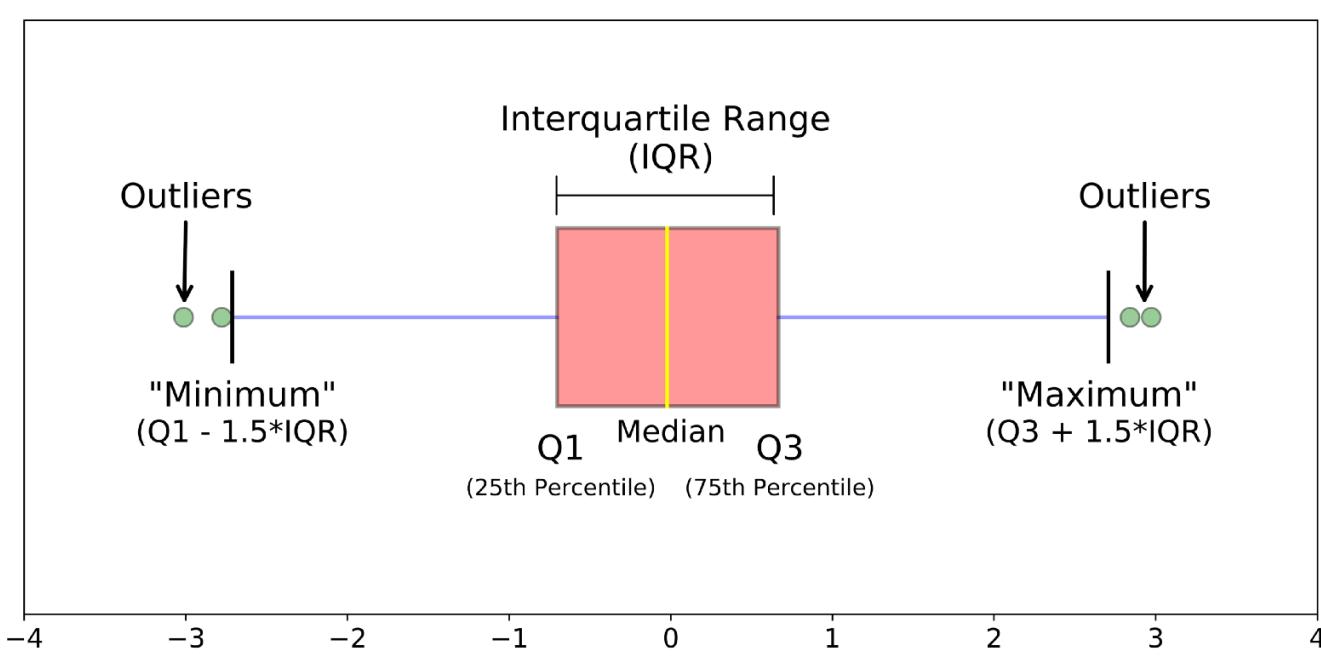
$$x < Q1 - 3 \cdot IQR \text{ или } x > Q3 + 3 \cdot IQR$$



# ЯЩИК С УСАМИ

Ящик с усами – это диаграмма, которая показывает:

- одномерное распределение вероятностей (квартили)
- границы попадания “нормальных” точек
- выбросы



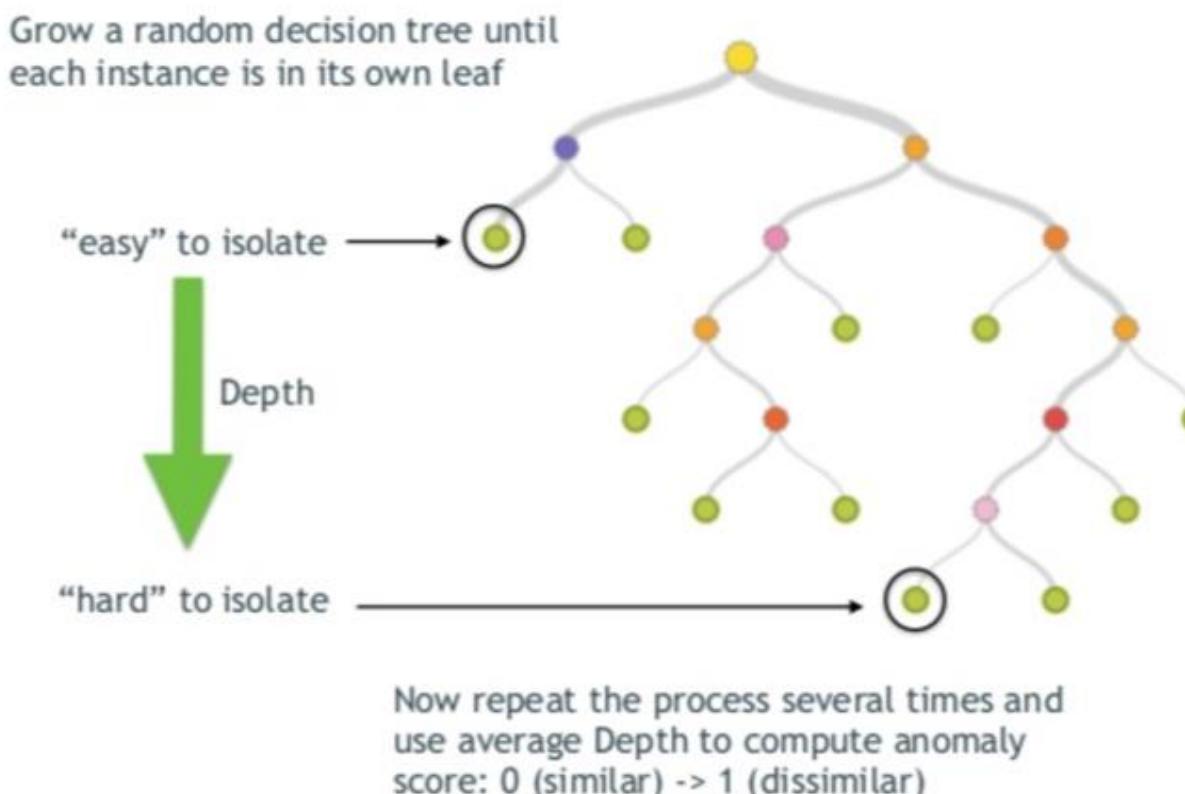
# ISOLATION FOREST

- Строим лес, состоящий из  $N$  деревьев. Каждый признак и порог выбираем случайно. Останавливаемся, когда в вершине 1 объект или когда построили дерево максимальной глубины.

*Идея: чем сильнее объект отличается от большинства, тем раньше он будет отделен от основной выборки случайными разбиениями => выбросы – объекты, которые оказались на небольшой глубине.*

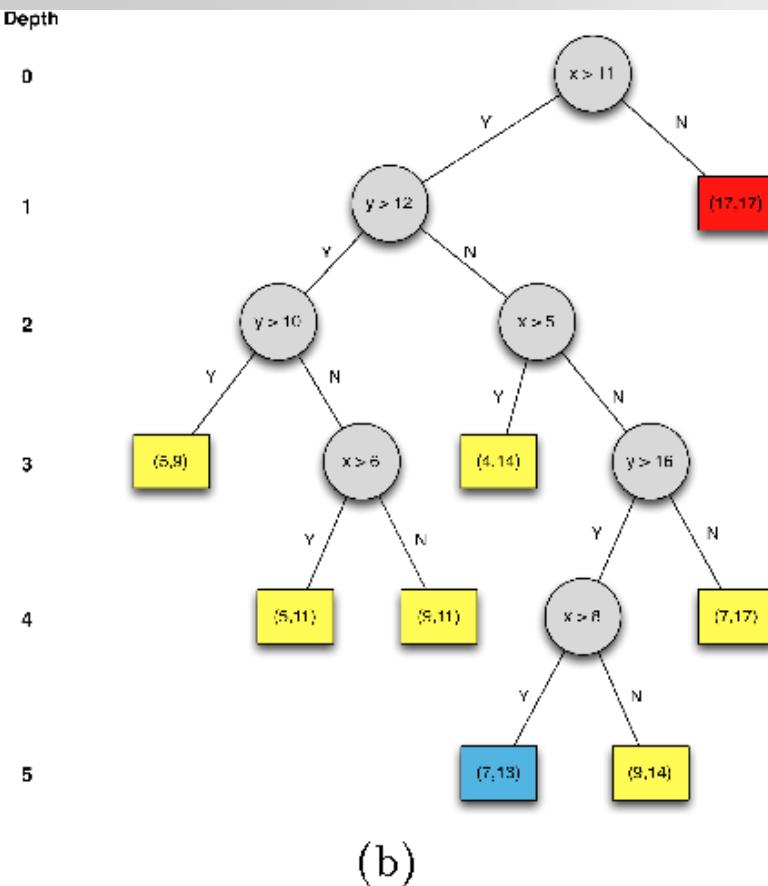
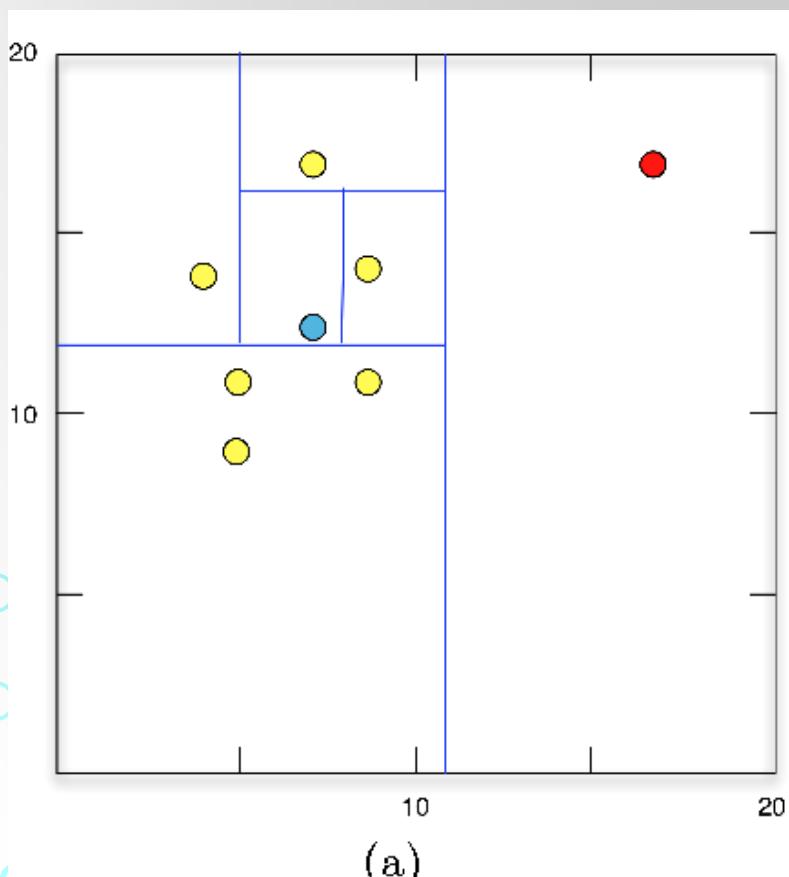
# ISOLATION FOREST

Идея: чем сильнее объект отличается от большинства, тем раньше он будет отделен от основной выборки случайными разбиениями => выбросы – объекты, которые оказались на небольшой глубине.



# ISOLATION FOREST

Идея: чем сильнее объект отличается от большинства, тем раньше он будет отделен от основной выборки случайными разбиениями => выбросы – объекты, которые оказались на небольшой глубине.



# ISOLATION FOREST

- Если объект единственный в листе, то его оценка аномальности в дереве – это глубина листа  $h_n(x) = k$ .
- Оценка аномальности объекта в Isolation Forest:

$$a(x) = 2^{-\frac{a}{b}},$$

где  $a = \frac{1}{N} \sum h_n(x)$  – средняя глубина, где  $N$  – число деревьев в лесе,

$b = c(l)$  – средняя длина пути, посчитанная по всем объектам и всем деревьям в лесе, построенном по выборке размера  $l$ .

# ПОИСК АНОМАЛИЙ С ПОМОЩЬЮ МОДЕЛЕЙ МЛ

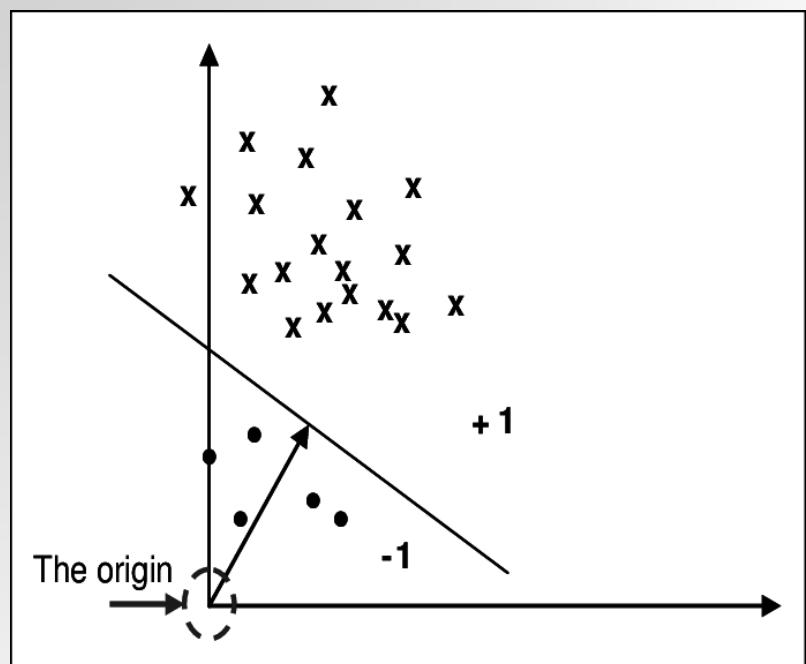
Идея: можно настроить модель машинного обучения так, чтобы на нормальных объектах она принимала значения, близкие к нулю (или, например, положительные значения). Тогда если прогноз на объекте сильно отличается от прогноза на обучающей выборке, то такой объект можно считать аномальным.

# ONE-CLASS SVM

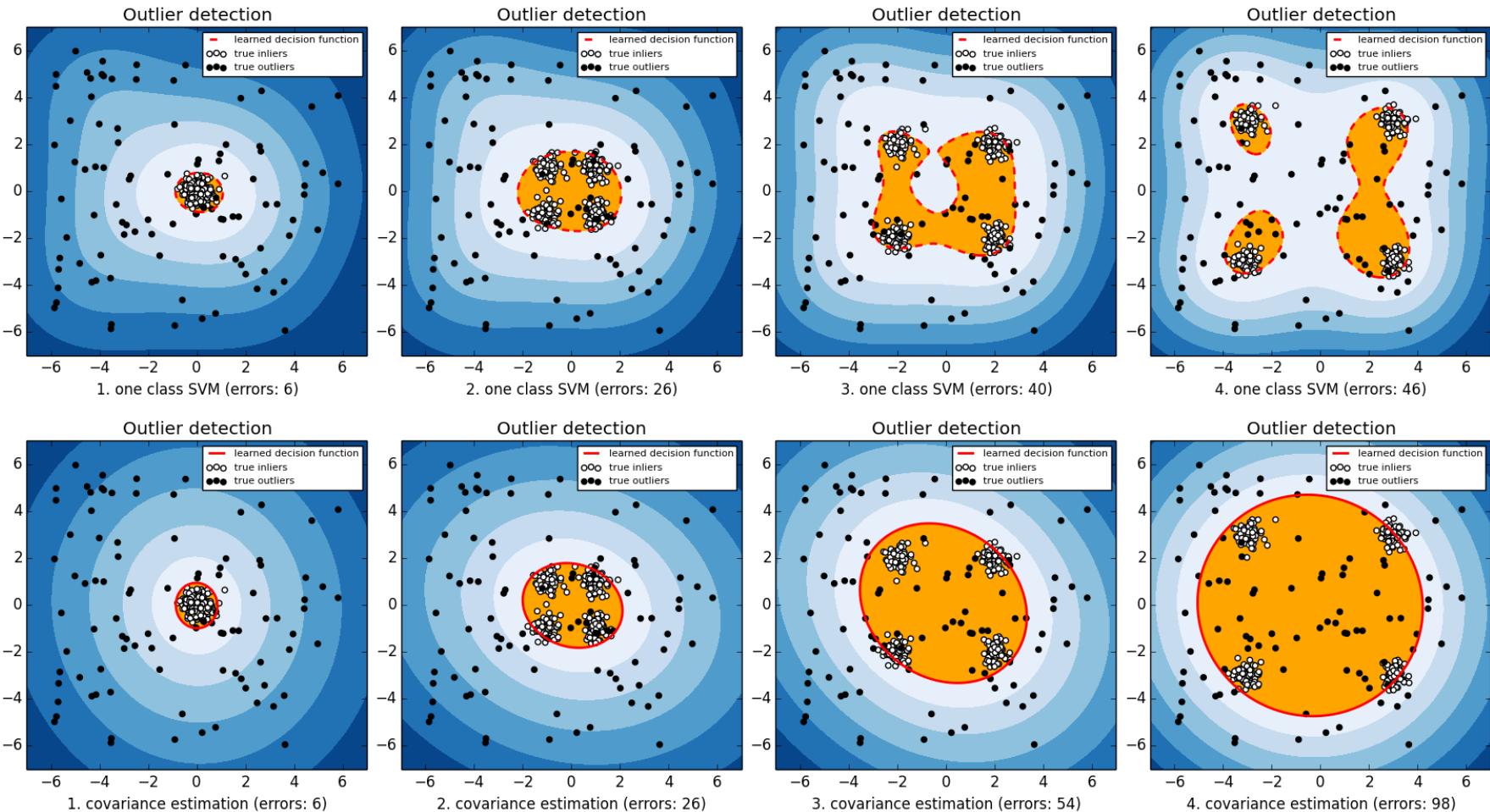
Метод строит линейную функцию  $a(x) = \text{sign}(w, x)$  так, чтобы она отделяла выборку от начала координат с максимальным отступом, а именно:

- $a(x)$  отделяет как можно больше объектов выборки от нуля:  $a(x) = +1$  на области как можно меньшего объема, содержащей как можно больше объектов выборки
- имеет большой отступ от 0.

Тогда объекты с  $a(x) = -1$  – это аномалии.

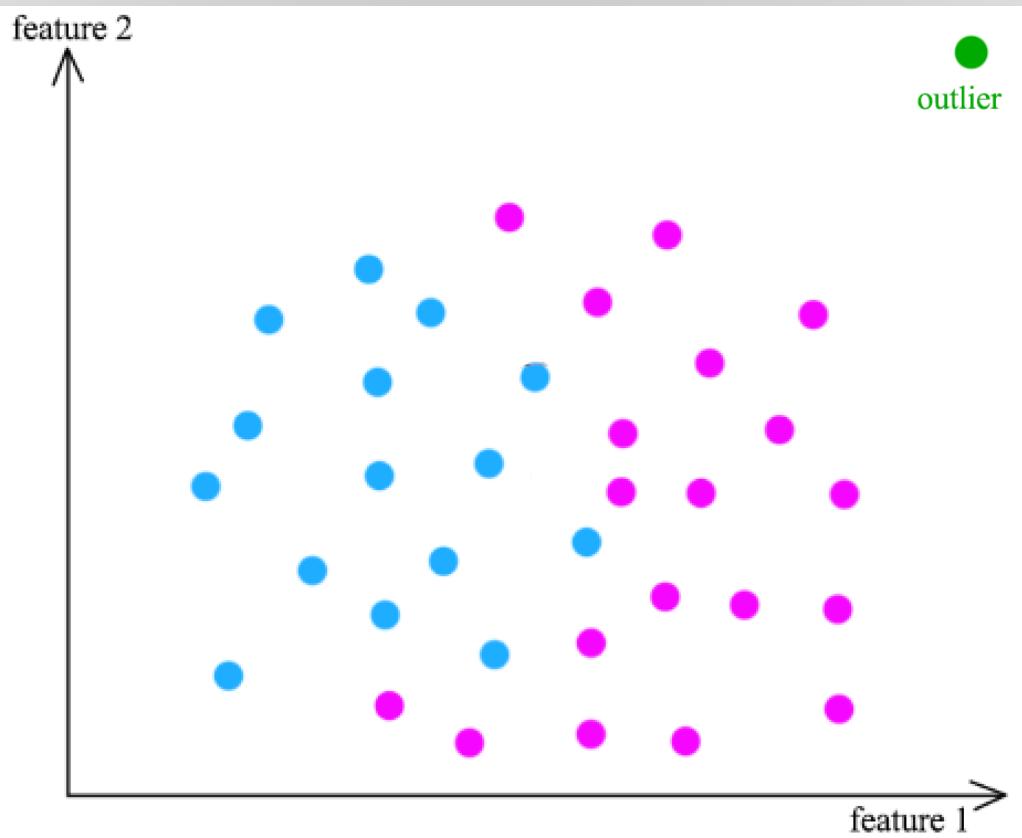


# ONE-CLASS SVM С RBF-ЯДРОМ



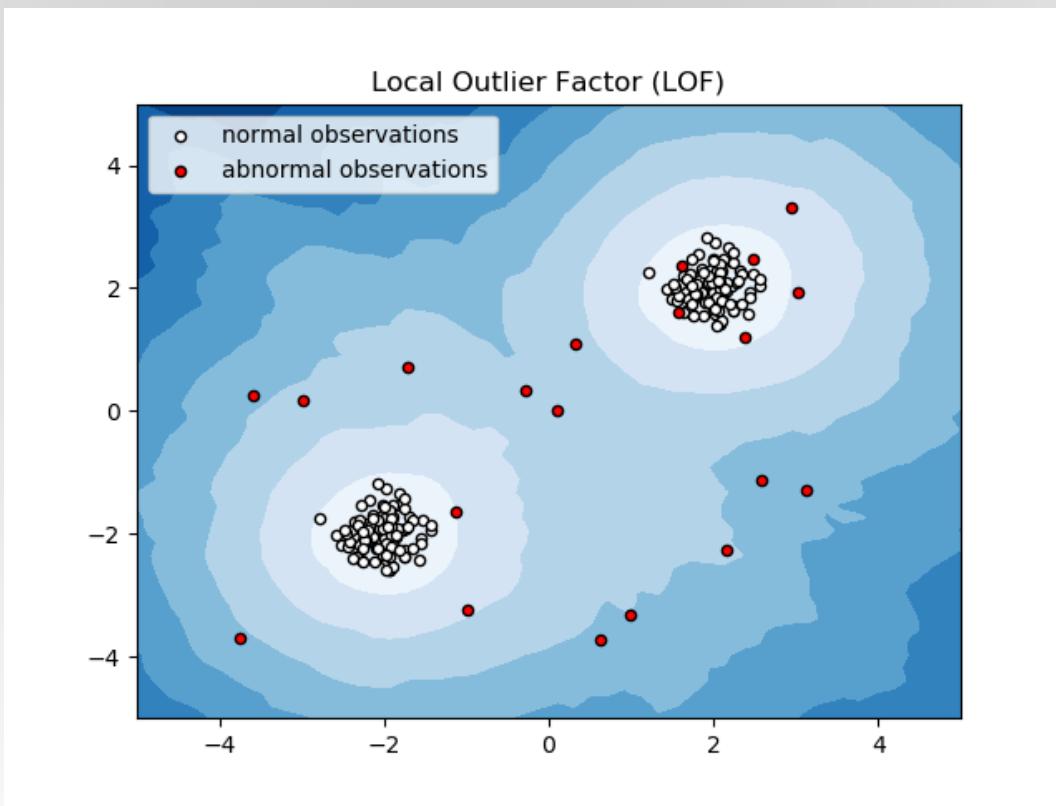
# ПОИСК ВЫБРОСОВ С ПОМОЩЬЮ KNN

- Вычисляем среднее расстояние от каждой точки до её ближайших  $k$  соседей
- Точки с наибольшим средним расстоянием – выбросы



# LOCAL OUTLIER FACTOR

- Задаем плотность распределения в точке, используя k ближайших соседей
- Точки, плотность распределения в которых значительно меньше, чем у соседей – выбросы.



# ССЫЛКИ

- <https://dyakonov.org/2017/04/19/поиск-аномалий-anomaly-detection/>
- [https://scikit-learn.org/stable/modules/outlier\\_detection.html](https://scikit-learn.org/stable/modules/outlier_detection.html)
- <https://github.com/yzhao062/pyod>