



# Лекция 2

# Метрические алгоритмы

# Содержание



1. Знакомимся с метрическими алгоритмами
2. Учимся оценивать качество и подбирать параметры
3. Улучшаем алгоритм
4. Алгоритмы на текстах и домашнее задание 1
5. Практический кейз: Рекомендательные системы

# Часть 1

## Ваш первый алгоритм

---



# Какие бывают алгоритмы



- Параметрические (имеют фиксированное число параметров) Имеют четкие предположения о структуре данных, работают быстро.
- Непараметрические (число параметров растет с размером датасета). Не имеют никаких предположений о структуре данных, работают очень медленно на больших данных.

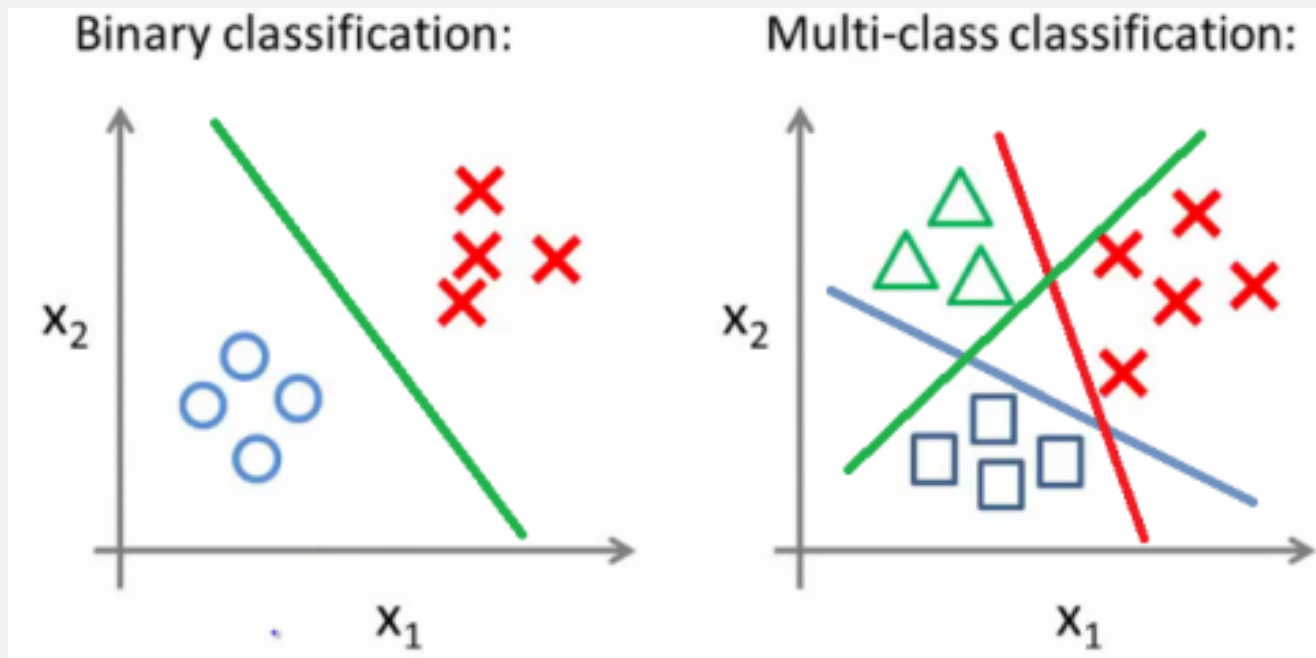
Сегодня вся лекция про **непараметрические** алгоритмы

# Гипотеза компактности



Гипотеза:

Похожие объекты лежат в одном классе



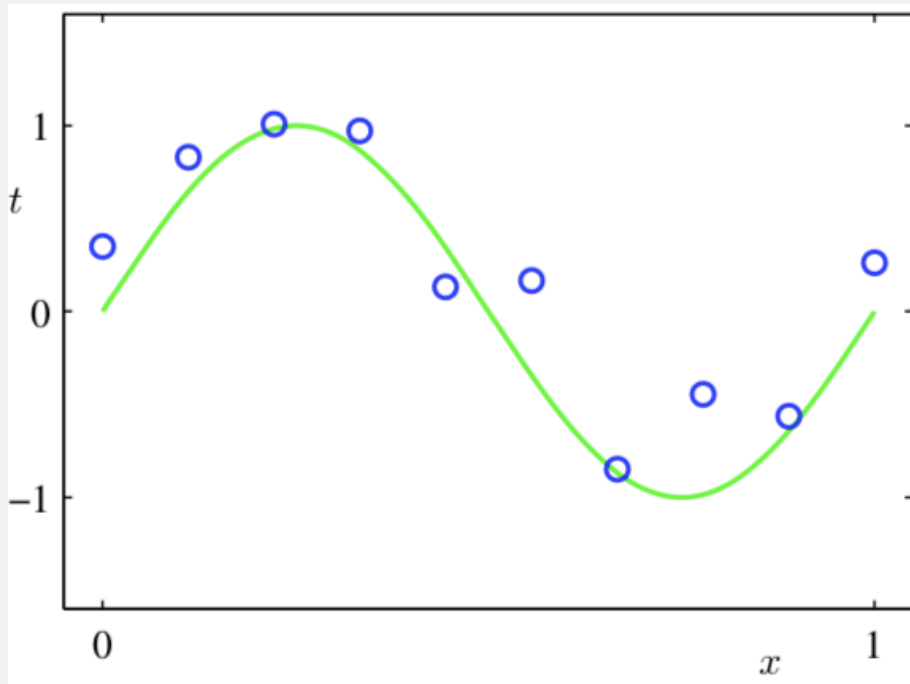
Источник: [medium.com/@b.terryjack](https://medium.com/@b.terryjack)

# Гипотеза непрерывности



Гипотеза:

Похожие объекты имеют похожий ответ



Источник: Bishop



Пусть обучающая выборка размера  $N$ .

Общая формула для классификации:

$$a(x, X_{train}) = \operatorname{argmax}_c \sum_{i=1}^N w(x, x_i) I[y_i = c], x_i \in X_{train}$$

Общая формула для регрессии:

$$a(x, X_{train}) = \frac{\sum_{i=1}^N w(x, x_i) y_i}{\sum_{i=1}^N w(x, x_i)}, x_i \in X_{train}$$

**Непараметрический, ленивый алгоритм**



- Если ненулевой вес только у ближайшего объекта, то алгоритм называют алгоритмом ближайшего соседа
- Если ненулевые веса для  $k$  ближайших объектов, то алгоритм называют алгоритмом  $k$  ближайших соседей (k-nearest neighbors, knn).

Пусть  $x_i$  —  $i$ -тый ближайший сосед объекта  $x$

- $w(x, x_i) = 1/k$
- $w(x, x_i) = \frac{k+1-i}{k}$
- $w(x, x_i) = \alpha^i, \alpha \in (0,1)$





Проблема: в прошлом варианте никак не учитываем величину расстояния.

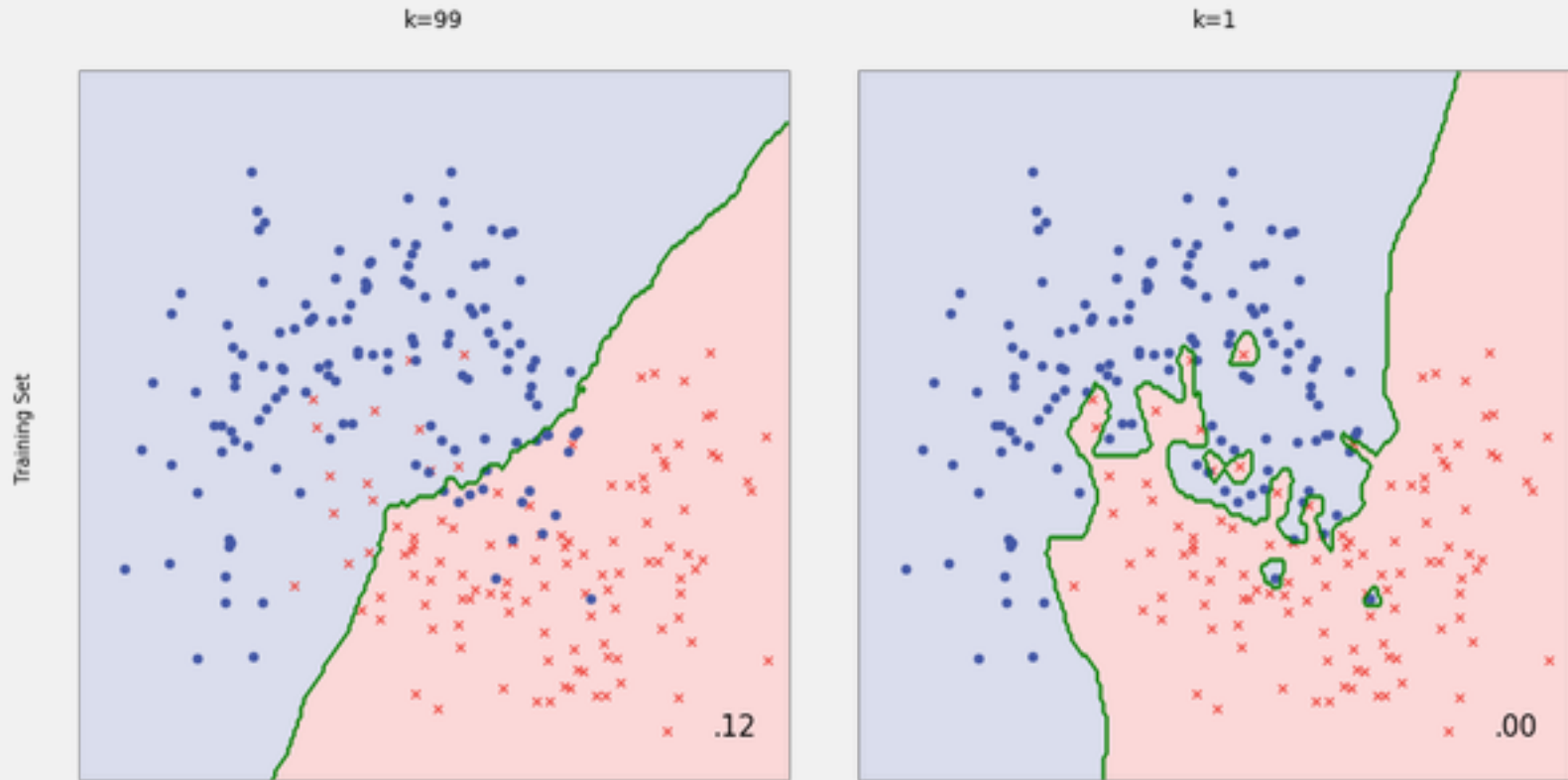
$$w(x, x_i) = K(\rho(x, x_i)),$$

где  $K(x)$  — любая монотонно убывающая функция.

Примеры:

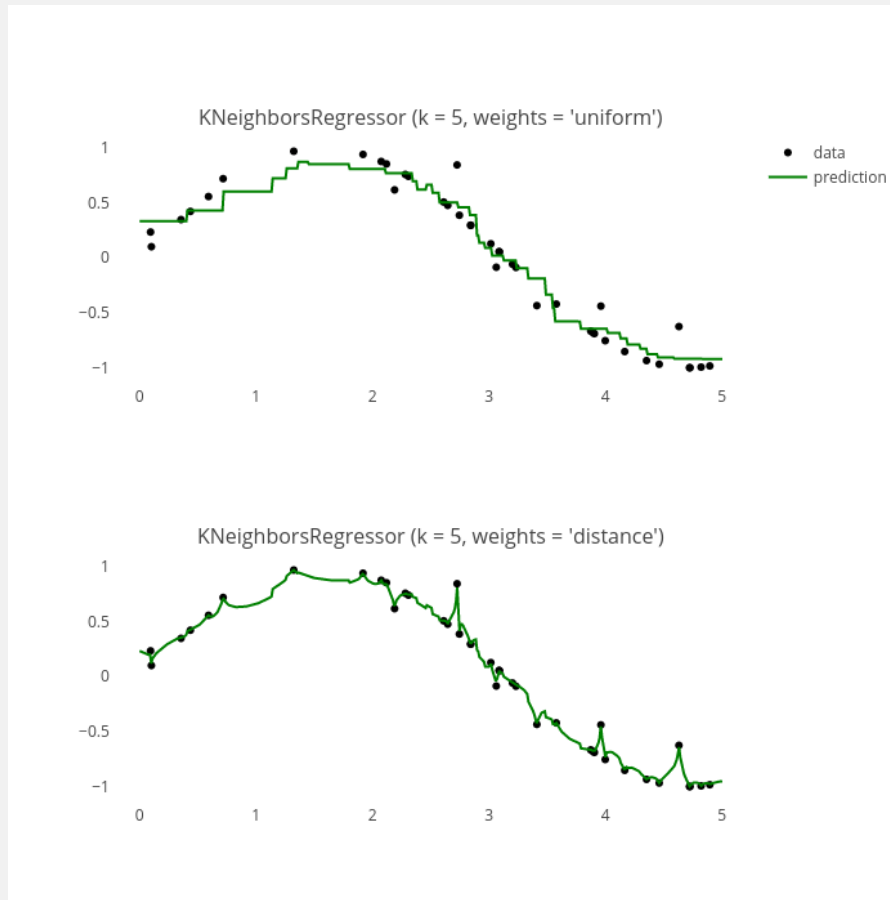
- $K(x) = \frac{1}{x + \beta}$
- $K(x) = \exp(-x)$
- $K(x) = \alpha^x, \alpha \in (0, 1)$

# Пример классификации



Источник: [cambridgecoding.wordpress.com/2016/03/24/](http://cambridgecoding.wordpress.com/2016/03/24/)

# Пример регрессии



Источник: [plot.ly/python](https://plot.ly/python)

## Часть 2

# Оцениваем качество, выбираем параметры

---



# Структурные параметры



**?** *Параметры метрических алгоритмов настраивать на обучающей выборке? На тестовой? Почему?*

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

**!** *Переобучение возникает из-за излишней сложности модели*

Параметры, которые нельзя настраивать на обучающей выборке, будем называть **структурными**.

Обучающую выборку нужно разделить на обучающую и **валидационную**. На ней настраиваем структурные параметры!

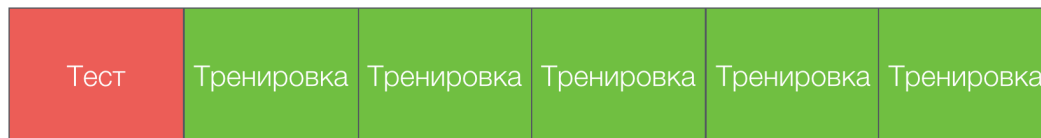
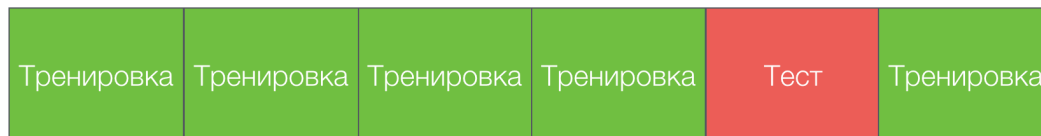
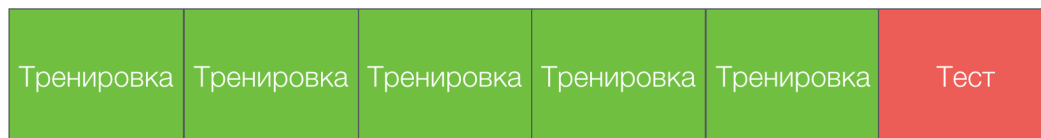
# Скольльзящий контроль



Качество зависит от объектов в валидации!

Решение — скользящий контроль (cross-validation).

В пределе, когда только 1 объект в тесте — LOO (leave one out).



# Оцениваем качество регрессии



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

Функции потерь:

1. **Квадратичная** ( $Q$  — MSE)

$$L(a, x, y) = (a(x) - y)^2$$

2. **Абсолютная** ( $Q$  — MAE)

$$L(a, x, y) = |a(x) - y|$$

3. **Логарифмическая** ( $Q$  — MSLE)

$$L(a, x, y) = (\log(a(x) + 1) - \log(y + 1))^2$$

4. **Абсолютная-процентная** ( $Q$  — MAPE)

$$L(a, x, y) = \frac{|a(x) - y|}{y}$$

# Оцениваем качество классификации



- **Accuracy** (точность) — процент правильно классифицированных объектов  $L(a, x, y) = [a(x) = y]$
- **Precision** (аккуратность) — процент правильно классифицированных объектов класса 1 среди всех объектов, которым алгоритм присвоил метку 1.
- **Recall** (полнота) — процент правильно классифицированных объектов класса 1 среди всех объектов класса 1
- **F1-score** — среднее гармоническое Precision и Recall

$$F1 = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$



Тестовая выборка содержит 10 объектов класса 1 и 990 объектов класса 0. Какая точность у константного алгоритма?



Почему именно среднее гармоническое?



# Не всегда нужно разбивать случайно



Нельзя никогда забывать, какую задачу мы решаем!

Если выборка маленькая, то нужно сохранять баланс классов — stratified валидация.



*Как сделать валидацию в случае:*

- *Спам-фильтра*
- *Предсказания объема продаж на следующую неделю*
- *Предсказания стоимости квартир для всего дома целиком*

# Часть 3

## Улучшаем алгоритм

---





Аксиомы:

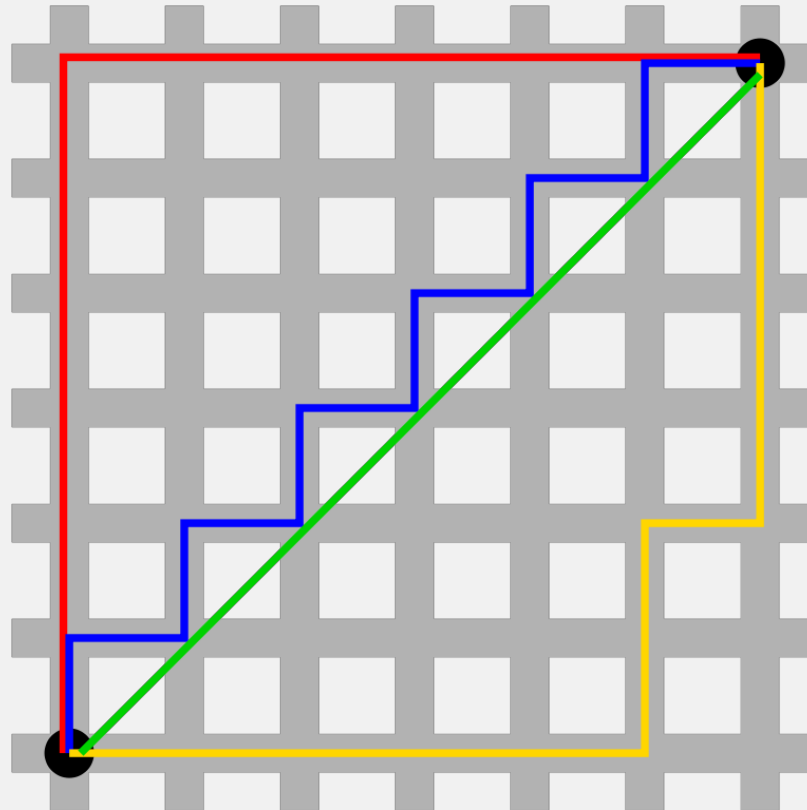
1.  $\rho(x, y) = 0$ , т.и.т.д  $x = y$
2.  $\rho(x, y) = \rho(y, x)$
3.  $\rho(x, z) \leq \rho(x, y) + \rho(y, z)$

Пусть  $D$  — число признаков. Метрика Минковского (при  $p \in (0, 1)$  не метрика):

$$\rho(x, y) = \left( \sum_{j=1}^D |x_j - y_j|^p \right)^{\frac{1}{p}}$$

- $p = 2$  — Евклидово расстояние
- $p = 1$  — Манхэттенское расстояние
- $p = \infty$  — Расстояние Чебышева (максимальное расстояние между двумя признаками)

# Манхэттенское расстояние



Источник: [en.wikipedia.org/wiki/Taxicab\\_geometry](https://en.wikipedia.org/wiki/Taxicab_geometry)

# Нормировка признаков



$$\rho(x, y) = \left( \sum_{j=1}^D |x_j - y_j|^p \right)^{\frac{1}{p}}$$

Нельзя так считать расстояния с признаками разных масштабов!

Два вида нормировки:

1. Стандартизация —  $x^j = \frac{x^j - \text{mean}(x^j)}{\text{std}(x^j)}$

2. Нормализация —  $x^j = \frac{x^j - \min(x^j)}{\max(x^j) - \min(x^j)}$



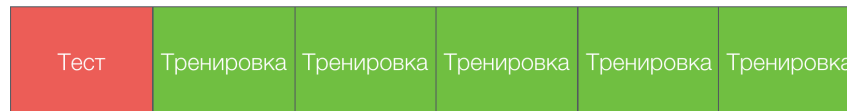
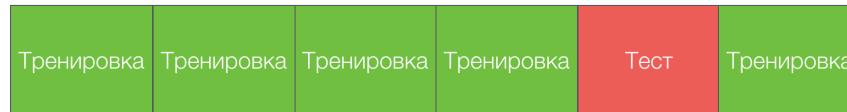
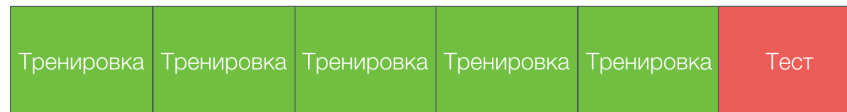
*В каком диапазоне будет лежать признак теперь?*

# Расстояния на категориальных признаках



1. Расстояние Хэмминга — число категориальных признаков, которые имеют разные значения.
2. Счетчики — среднее значение признака/целевой переменной с такой категорией.

При кодировании признака с помощью целевой переменной нельзя использовать целевую переменную данного объекта!



# Косинусное расстояние



По определению скалярного произведения считаем угол между векторами:

$$\text{sim}(x, y) = \cos \alpha = \frac{x \cdot y}{|x| |y|}$$

$$\text{rho}(x, y) = 1 - \text{sim}(x, y)$$



Косинусное расстояние часто используют для текстов. Почему?

# Расстояние Джаккарда



Как померить расстояние между множествами?  
Например, предложение — мешок (множество) слов.

$$\rho(X, Y) = 1 - \frac{X \cap Y}{X \cup Y}$$



*Приведите пример задачи, где удобно использовать расстояние над множествами*



# Расстояние Левенштейна



Минимальное количество операций вставки одного символа, удаления одного символа и замены одного символа на другой, необходимых для превращения одной строки в другую.


$$\rho(kitten, sitting) = 3$$

? В каких задачах часто применяется расстояние Левенштейна?



Посчитали расстояние по вещественным признакам, по категориальным, строковое, для множеств... Как теперь все объединить?

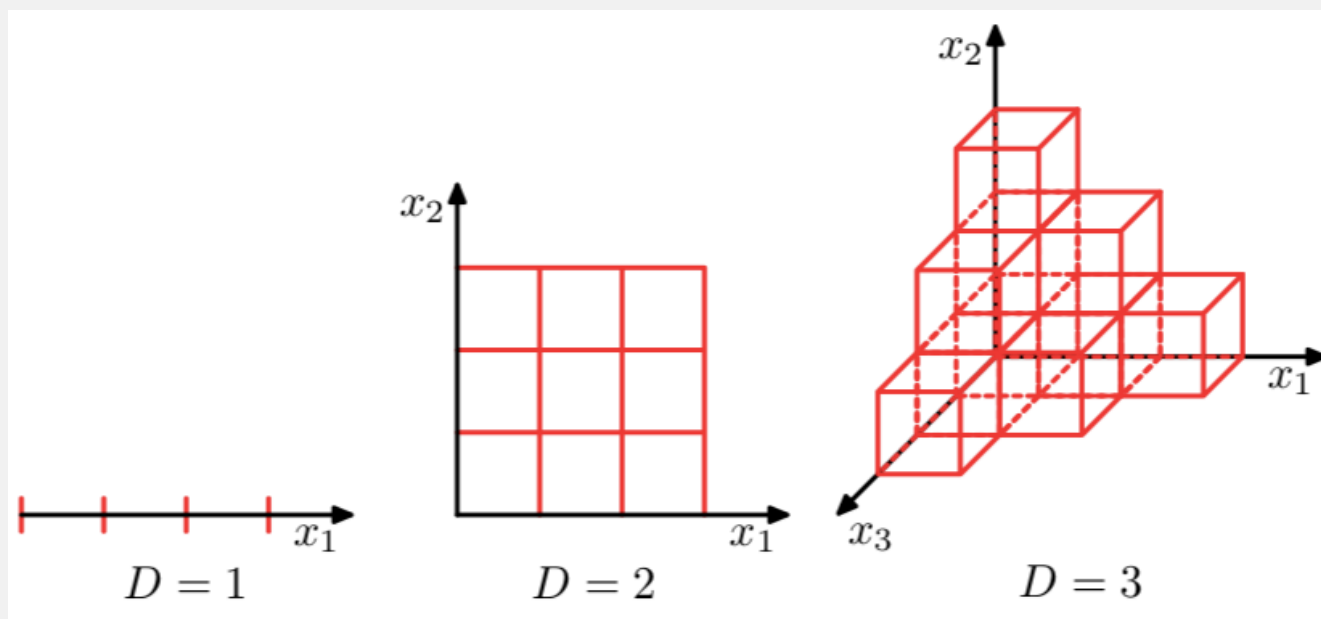
$$\rho(x, y) = c_1\rho_1(x, y) + c_2\rho_2(x, y) + \dots$$

**?** *Как найти коэффициенты?*

# Проклятие размерности



В пространстве большой размерности объекты сильно удалены друг от друга.

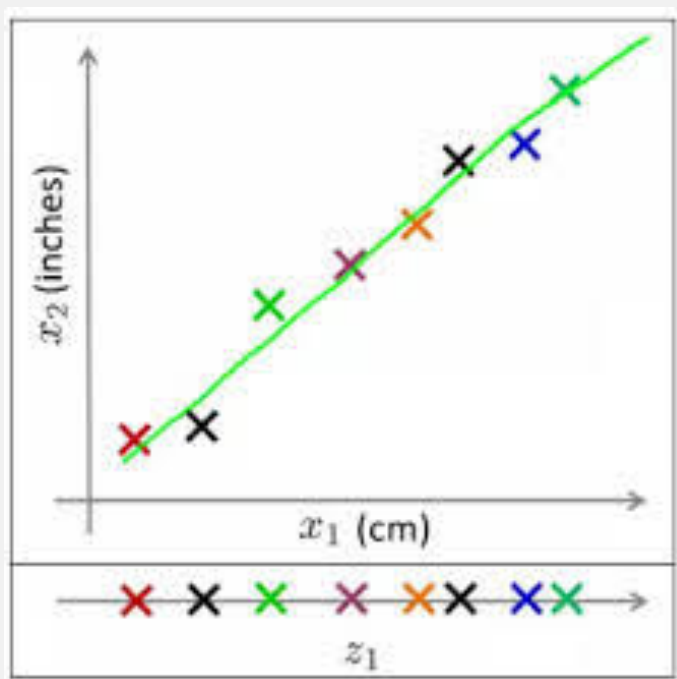


Источник: Bishop

# Уменьшаем размерность



- Методы снижения размерности
- Отбор признаков



Источник: [analyticsvidhya.com/blog/2015/07/dimension-reduction-methods](https://analyticsvidhya.com/blog/2015/07/dimension-reduction-methods)



Задача: найти и удалить вредные признаки.

*Какие признаки для нас вредные?*

- Перебрать все варианты и посмотреть качество (лучший, если признаков мало)
- Посчитать корреляцию с целевой функцией и удалить шумные
- Посчитать корреляцию всех пар признаков и удалить скоррелированные
- Последовательно удалять худшие
- Последовательно добавлять лучшие



Алгоритм:

- Наглядный, понятный
- Идеально работает, если правильно выбрана метрика
- Ленивый алгоритм, совсем не учится, тяжело работает на бигдате
- Позволяет делать **беспризнаковое распознавание**
- На признаковом распознавании, как правило, работает хуже других алгоритмов



Какие можете придумать примеры беспризнакового распознавания?



Какая сложность обучения алгоритма ближайшего соседа?  
Предсказания одного объекта?

# Сложность алгоритма



Сложность обучения —  $O(ND)$  (запоминаем выборку)

Сложность предсказания —  $O(ND)$  (считаем все расстояния)

В таком виде это в real time системах это работать не будет!

? Зачем мы тогда все это учим?

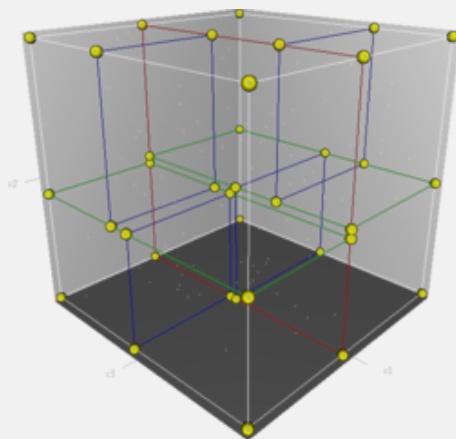
# Ускоряем базовый алгоритм



Структурируем признаковое пространство, чтобы по нему быстрее искать.

- KD-tree
- Ball tree

Если признаков мало (несколько десятков), то сложность по числу объектов логарифмическая. Если много — линейная (проклятие размерности), внедрять нельзя!





# Приближенный поиск ближайших соседей



В среднем имеют логарифмическую сложность даже для больших признаков пространств.

Примеры методов:

- ANNOY — делим пространство случайными плоскостями, строим дерево
- Navigable Small World — гуляем по графу тесного мира
- FAISS — кластеризуем пространство и ищем расстояния до центров кластеров
- LSH (Locality-sensitive hashing) — делаем хэш функцию, которая близким объектам присваивает близкие значения хэша

# Применяют ли метрические алгоритмы?



Применяют!

Все большие поисковые/рекомендательные системы состоят из двух компонент:

- Грубый отбор кандидатов
- Использование финальной модели

Быстрый приближенный поиск ближайших соседей идеально подходит под задачу выборов кандидатов.

Величину  $\rho(x, y)$  можно подавать в финальную модель!

## Часть 4

# Алгоритмы на текстах и домашнее задание 1

---



# Обработка естественного языка



- Категоризация новостей
- Автоответы в почте
- Машинный перевод
- Информационный поиск
- Чат боты

? Как мы представляем текст в виде вектора?

# Представление текста



В виде вектора размера размерность словаря, где единицы стоят в тех ячейках, которые соответствуют слову, которое есть в тексте

Можно рассматривать не слова, а

- N-граммы слов
- N-граммы букв

? В чем достоинства и недостатки каждого подхода?



Можно вставлять не просто единички, а значение, которое показывает, насколько слово важно для данного текста

Важность слова в **тексте** \* Важность слова **вообще**

$Tfidf = TF * IDF$

**TF** — *term frequency*, **IDF** — *inverse document frequency*

$TF(w, d) = \frac{n_w}{n_d}$ , где  $n_w$  — сколько раз встретилось слово в документе,

$n_d$  — длина документа

$IDF(w) = \log \frac{|D|}{|D_w|}$ ,  $|D|$  — число документов,  $|D_w|$  — число

документов, в которых есть слово  $w$

И что, это все руками считать?



## `sklearn.model_selection.KFold`

```
class sklearn.model_selection.KFold(n_splits=5, shuffle=False, random_state=None)
```

[\[source\]](#)

K-Folds cross-validator

## `sklearn.model_selection.cross_val_score`

```
sklearn.model_selection.cross_val_score(estimator, X, y=None, groups=None, scoring=None, cv=None, n_jobs=None, verbose=0, fit_params=None, pre_dispatch='2*n_jobs', error_score=nan)
```

[\[source\]](#)

Evaluate a score by cross-validation



## sklearn.feature\_extraction.text.CountVectorizer

```
class sklearn.feature_extraction.text.CountVectorizer(input='content', encoding='utf-8', decode_error='strict', strip_accents=None, lowercase=True, preprocessor=None, tokenizer=None, stop_words=None, token_pattern='(?u)|b|w|w+|b', ngram_range=(1, 1), analyzer='word', max_df=1.0, min_df=1, max_features=None, vocabulary=None, binary=False, dtype=<class 'numpy.int64'>)
```

[\[source\]](#)

Convert a collection of text documents to a matrix of token counts

## sklearn.feature\_extraction.text.TfidfVectorizer

```
class sklearn.feature_extraction.text.TfidfVectorizer(input='content', encoding='utf-8', decode_error='strict', strip_accents=None, lowercase=True, preprocessor=None, tokenizer=None, analyzer='word', stop_words=None, token_pattern='(?u)|b|w|w+|b', ngram_range=(1, 1), max_df=1.0, min_df=1, max_features=None, vocabulary=None, binary=False, dtype=<class 'numpy.float64'>, norm='l2', use_idf=True, smooth_idf=True, sublinear_tf=False)
```

[\[source\]](#)

Convert a collection of raw documents to a matrix of TF-IDF features.

Возвращает разреженную матрицу! В домашней работе для простоты работайте с обычной, так что выбирайте разумно `max_features`





**Спасибо за  
внимание!**