

«Машинное обучение и анализ данных»

Метрические методы

Александр Дьяконов

21 сентября 2020 года



План

Методы, которые используют расстояния

Геометрия

Ближайший центроид

Метод k ближайших соседей

Теоретическое обоснование

Обобщения (весовые, на регрессию и т.п.)

Метрики

Приложения

Эффективные методы поиска соседей

Метрические алгоритмы

«distance-based» – анализируются расстояния

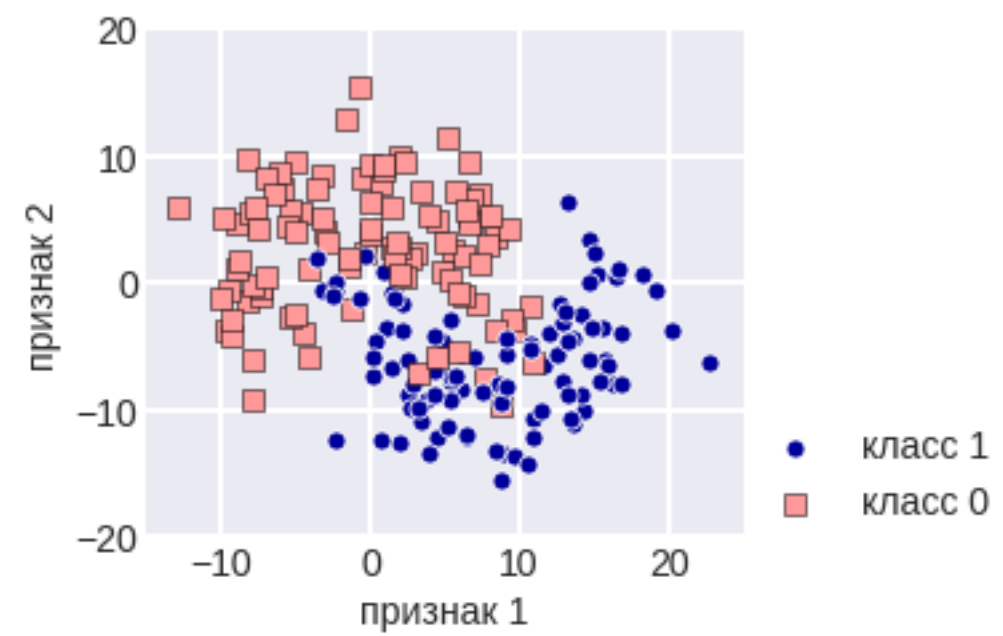
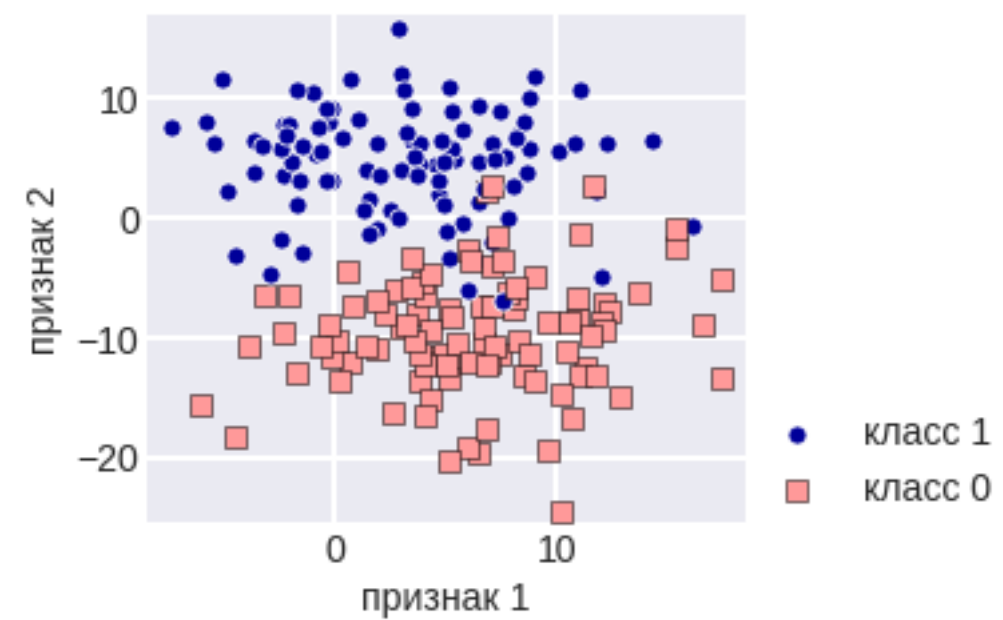
$$\rho(x, x_1), \dots, \rho(x, x_m)$$

- **Nearest centroids algorithm / Distance from Means**
 - **kNN (Nearest Neighbor)**

ещё называют:

- **«memory-based»**
- **«instance-based»**
- **«non-parametric»**

Модельные задача классификации



Ближайший центроид (Nearest centroid algorithm)

**Задача классификации на непересекающиеся классы
с вещественными признаками:**

$$Y = \{1, 2, \dots, l\}$$
$$x_i \in \mathbb{R}^n$$

центроиды:

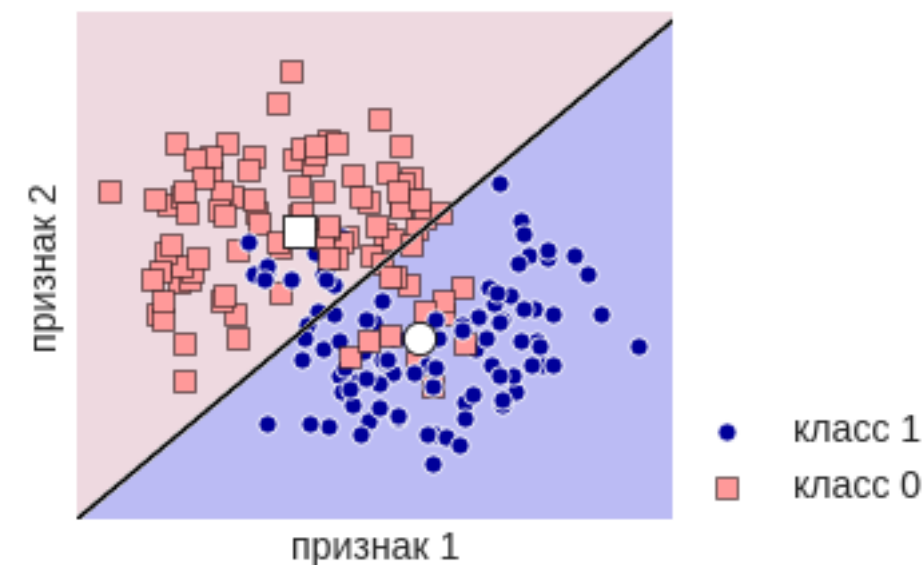
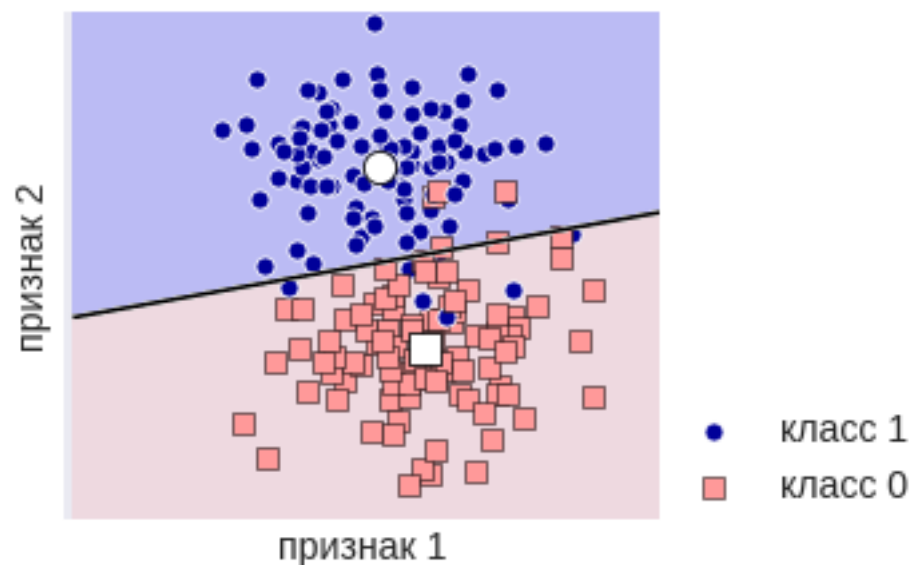
$$c_j = \frac{1}{|\{i : y_i = j\}|} \sum_{i: y_i = j} x_i$$

классификация:

$$a(x) = \arg \min_j \rho(x, c_j)$$

обобщается на случаи, когда можно вычислить «средний объект»

Ближайший центроид (Nearest centroids algorithm)



- + хранить только центроиды (их можно адаптивно менять)
- + понятие центроида можно менять («средний объект»)
- + простая реализация
- + размер модели = число классов \times описание центроида

– очень простой алгоритм

интуитивно подходит в задачах,

где объекты разных классов распределены «колоколообразно»

Минутка кода: ближайший центроид (Nearest centroids algorithm)

```
from sklearn.neighbors.nearest_centroid import NearestCentroid
model = NearestCentroid()
model.fit(X, y)
a = model.predict(X2)
```

есть параметр

`metric='euclidean'`

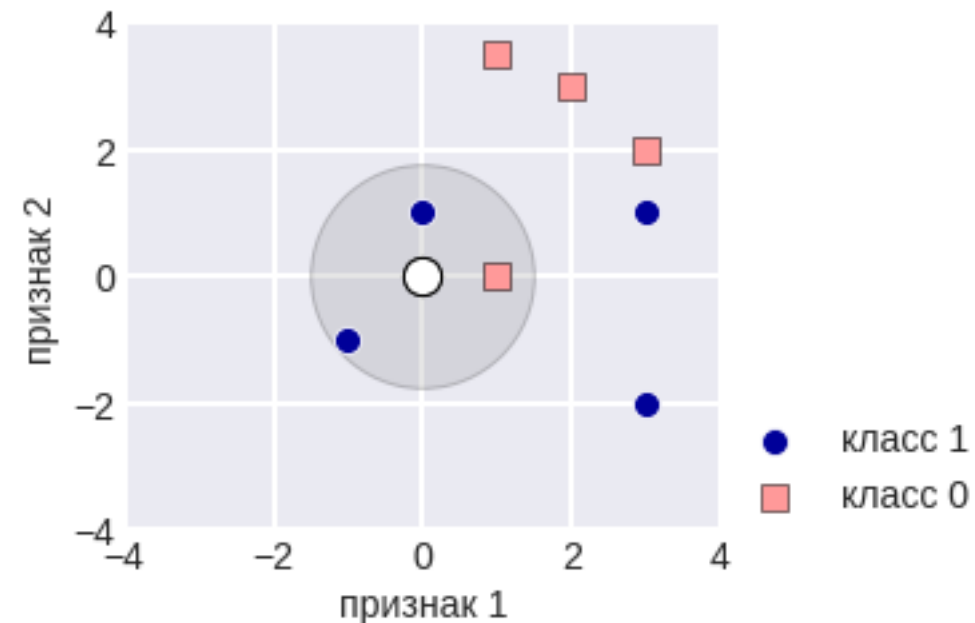
Подход, основанный на близости

Задача классификации

$$a(x) = \text{mode}(y_i \mid x_i \in N(x))$$

Задача регрессии

$$a(x) = \text{mean}(y_i \mid x_i \in N(x))$$



$N(x)$ – **окрестность (neighborhood) объекта x**
(похожие на него объекты)

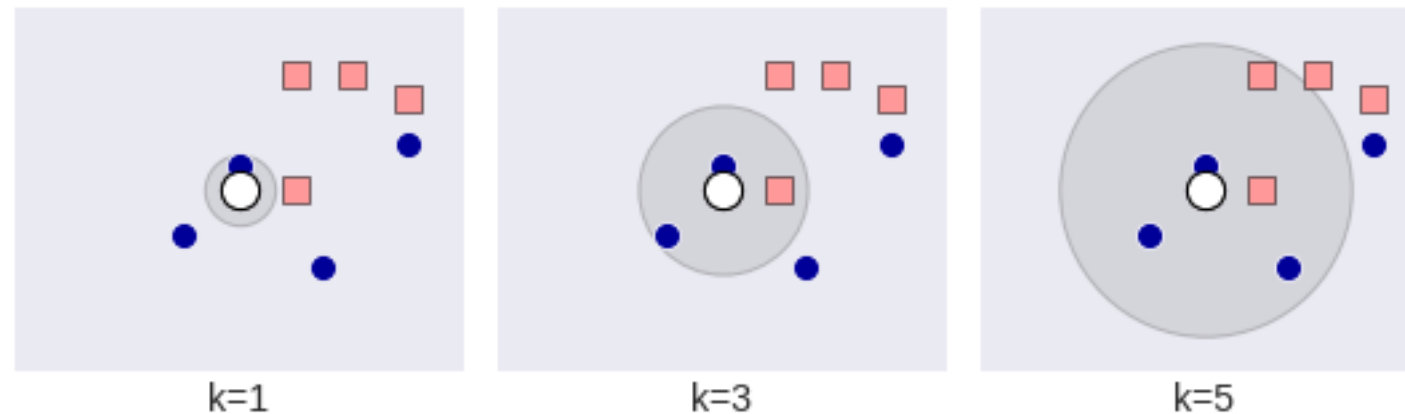
Окрестность

Если X – метрическое пространство с метрикой ρ ,
пусть нумерация объектов такая, что

$$\rho(x, x_1) \leq \dots \leq \rho(x, x_m)$$

к ближайших соседей: $N(x) = \{x_1, \dots, x_k\}$

– метод **к ближайших соседей (kNN = k nearest neighbours)**



Есть также «Fixed-Radius Near Neighbor»

$$N(x) = \{x_t \mid \rho(x_t, x) \leq R\}$$

Метод k ближайших соседей (kNN)

$k = 1$ – алгоритм ближайшего соседа (nearest neighbour algorithm)

формально нет обучения – храним всю выборку
работа алгоритма – просматриваем всю выборку
(+ вычисляем расстояние до каждого объекта обучения)

– ленивый алгоритм (lazy learning)

Гиперпараметр k можно выбрать на скользящем контроле **дальше**

Ещё гиперпараметры (потом):

- метрика (+ параметры метрики)
 - ядро (+ параметры ядра)

Обоснование 1NN

Теорема. В достаточно однородном метрическом пространстве объектов бинарной задачи классификации ошибка 1NN не выше удвоенной ошибки оптимального алгоритма.

Пусть в некоторой области вероятность встретить объект класса 0 (без огр-я общ-ти) – $p \leq 0.5$ – (это же и вероятность ошибки оптимального алгоритма), тогда

сосед	объект	
	класс 0 – p	класс 1 – $1 - p$
класс 0 – p	pp	$p(1 - p)$
класс 1 – $(1 - p)$	$p(1 - p)$	$(1 - p)(1 - p)$

вероятность ошибочной классификации

$$2p(1 - p) = 2p - 2p^2 \leq 2p$$

Термины

Нетерпеливый алгоритм (Eager learner)

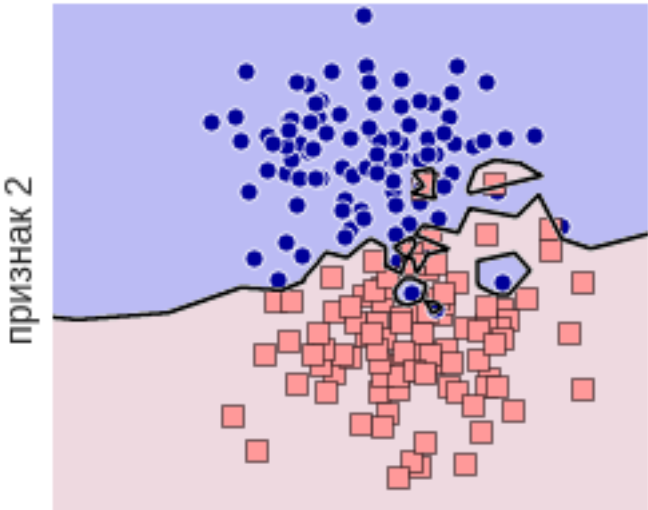
как только есть обучение – получает значения параметров (учит модель)

Ленивый алгоритм (Lazy learner)

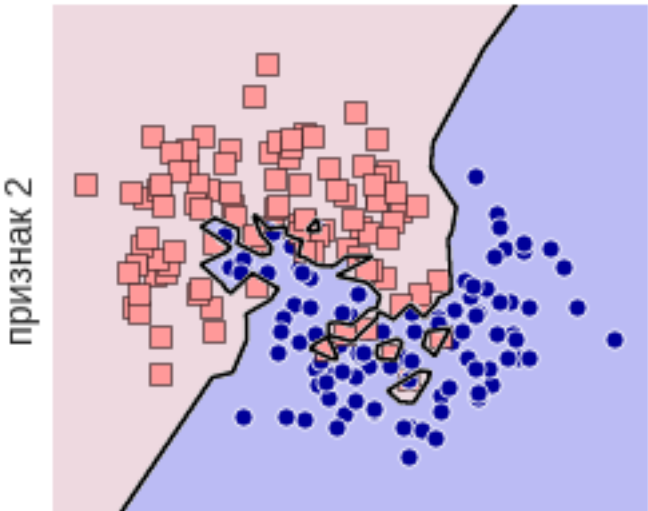
не использует обучающую выборку до классификации

Решение модельной задачи при разном числе соседей

k=1

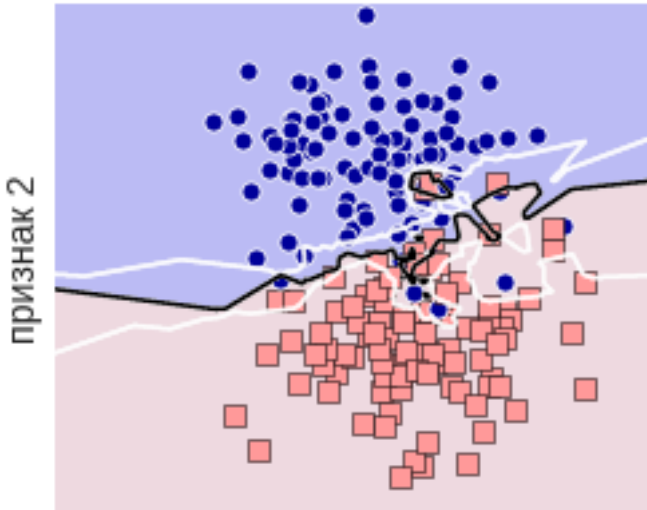


признак 1

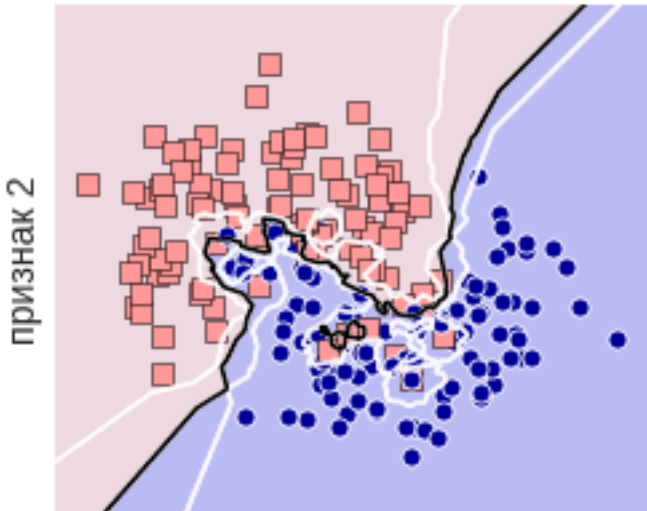


признак 1

k=3

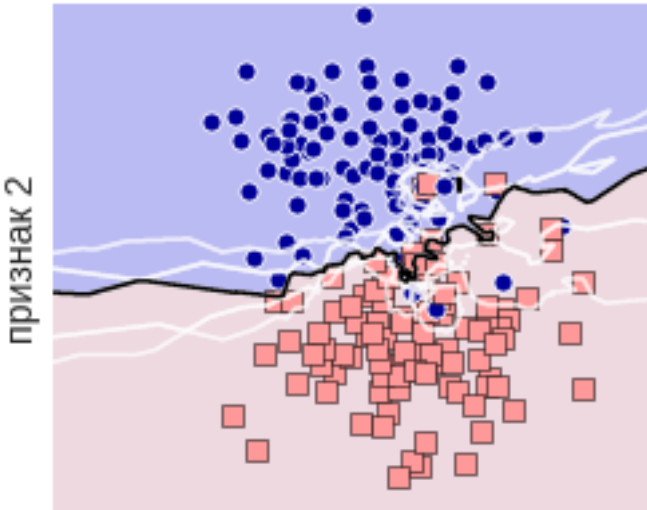


признак 1

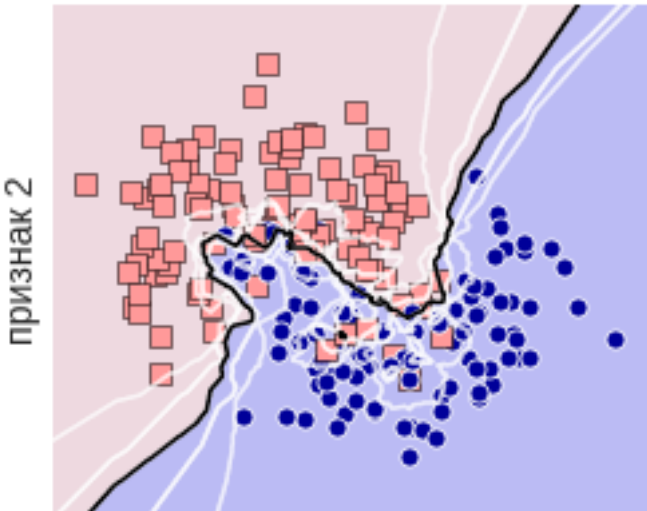


признак 1

k=5



признак 1



признак 1

Решение модельной задачи при разном числе соседей

Как увидим дальше, k отвечает за «сложность модели»

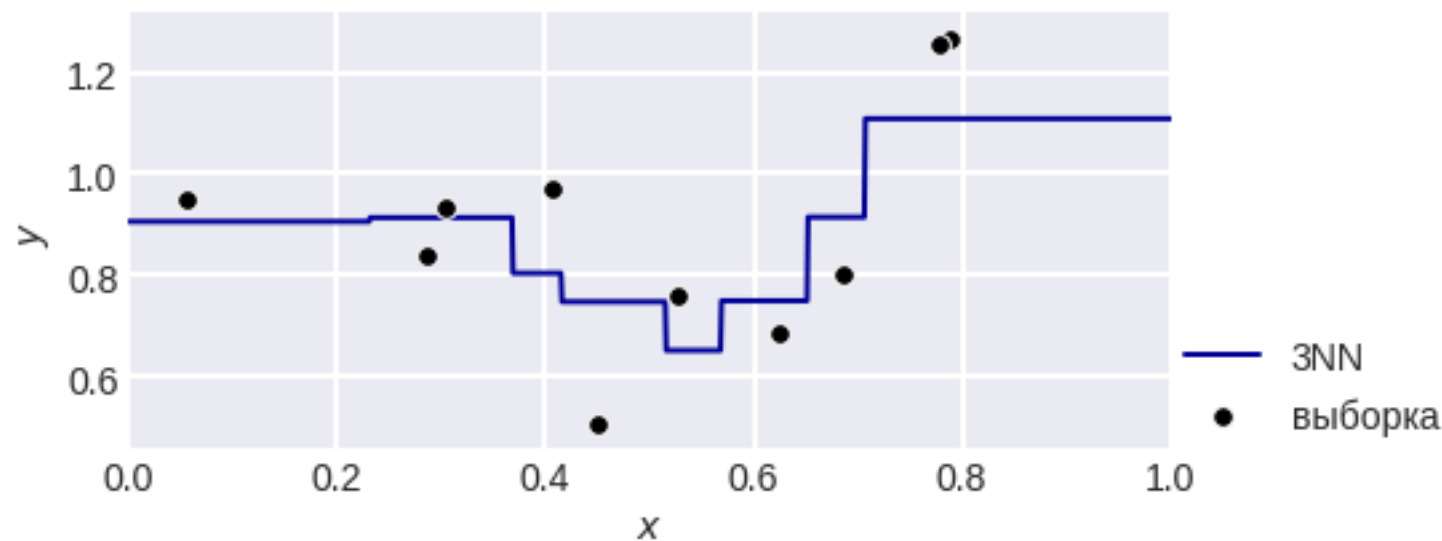
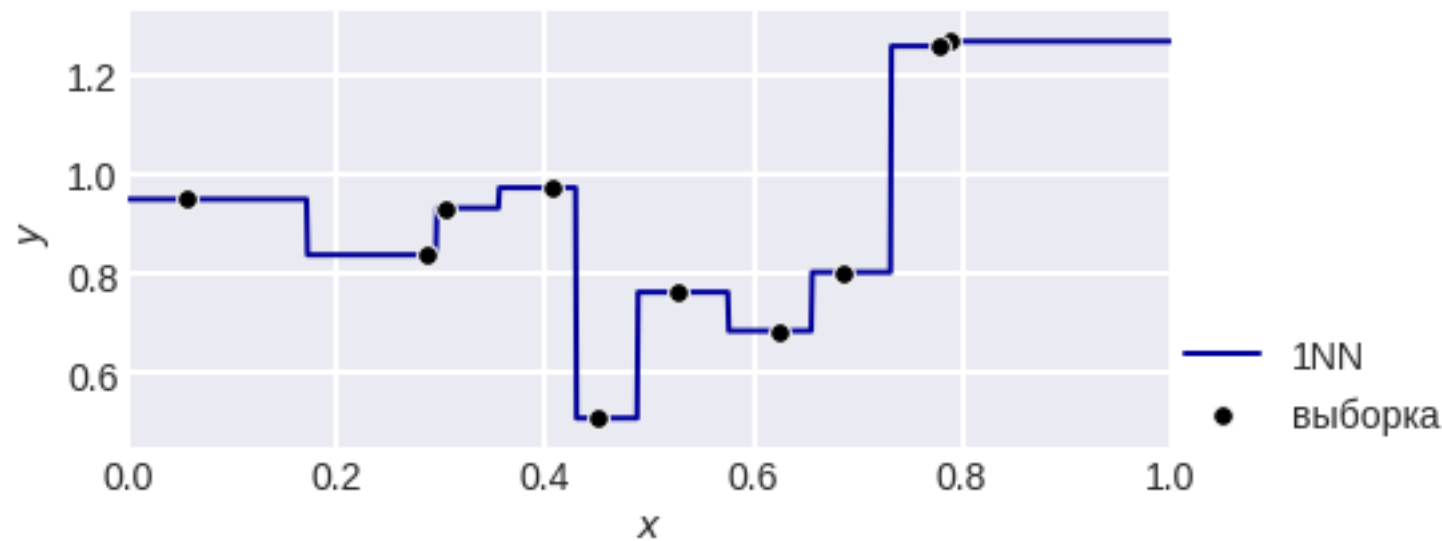
Минутка кода

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=5)
model.fit(X, y)
a = model.predict(X2)
p = model.predict_proba(X2)[:, 1]
```

параметры разберём ниже

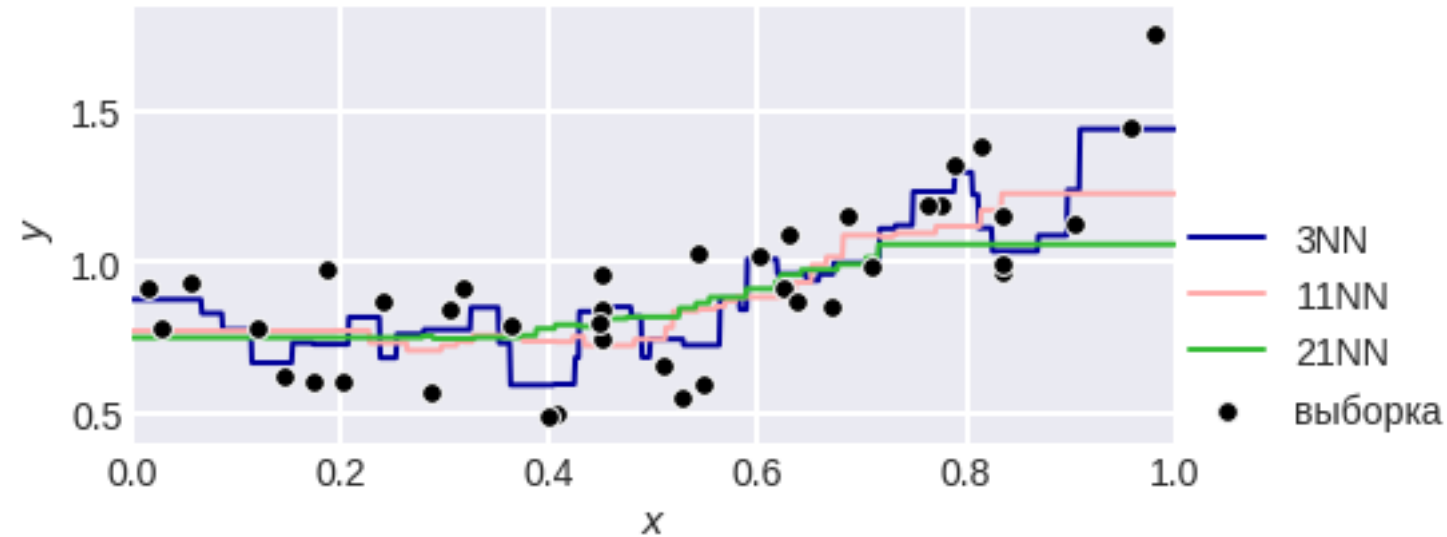
Метод ближайшего соседа

обобщается на регрессию



Метод ближайшего соседа

обобщается на регрессию



Минутка кода

```
from sklearn.neighbors import KNeighborsRegressor
model = KNeighborsRegressor(n_neighbors=3) # kNN-регрессия
model.fit(x_train, y_train) # обучение
a = model.predict(x_test) # ответ
```


Метод ближайшего соседа в регрессии

есть теоретическое обоснование

$$y = f(x) + \varepsilon \sim \{x_i, y_i\}_{i=1}^m,$$

$\mathbf{E}f^2 < +\infty$ **(больше нет ограничений)**

$a \sim k\text{NN}$ **«universally consistent»:**

Если $k \rightarrow +\infty$ и $k/n \rightarrow 0$, то
$$\mathbf{E} |a(x) - f(x)| \xrightarrow{n \rightarrow +\infty} 0$$

Кстати, это аппроксимация непрерывной функцией

Теоретическое обоснование для других метрических методов

[Fix, Hodges, 51]: classification + regularity, R^d

[Cover, Hart, 65, 67, 68]: classification + regularity, any metric

[Stone, 77]: classification, universal, R^d

[Devroye, Wagner, 77]: density estimation + regularity, R^d

[Devroye, Gyorfi, Kryzak, Lugosi, 94]: regression, universal, R^d

[Chaudhuri, Dasgupta, 14]: classification, nice metric/measure.

Везде k должен расти, но не очень быстро...

G. H. Chen, D. Shah E«xplaining the Success of Nearest Neighbor Methods in Prediction» // Publisher: Now Foundations and Trends https://devavrat.mit.edu/wp-content/uploads/2018/03/nn_survey.pdf

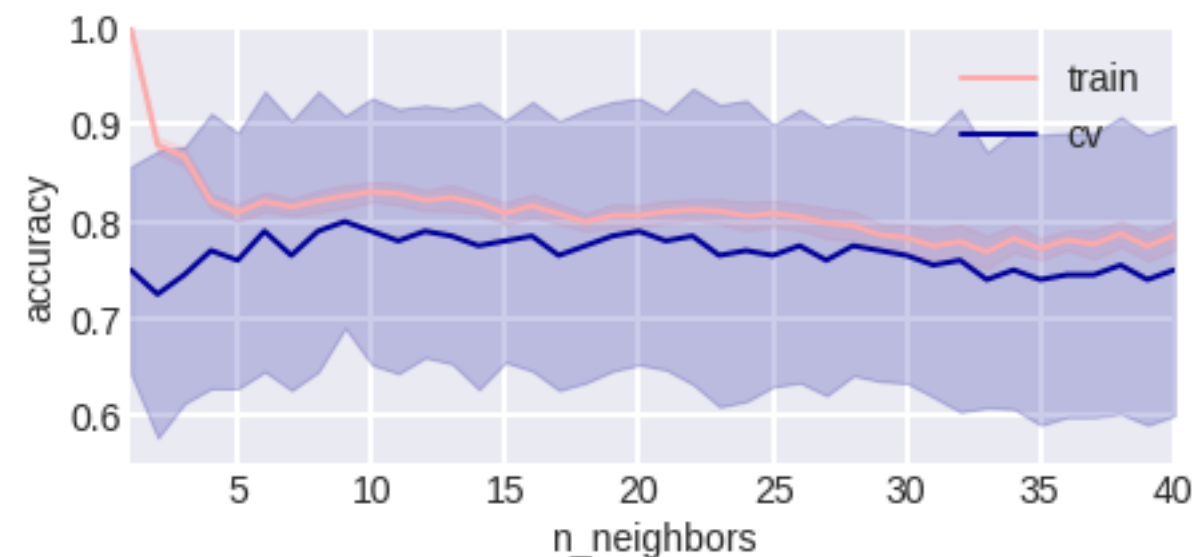
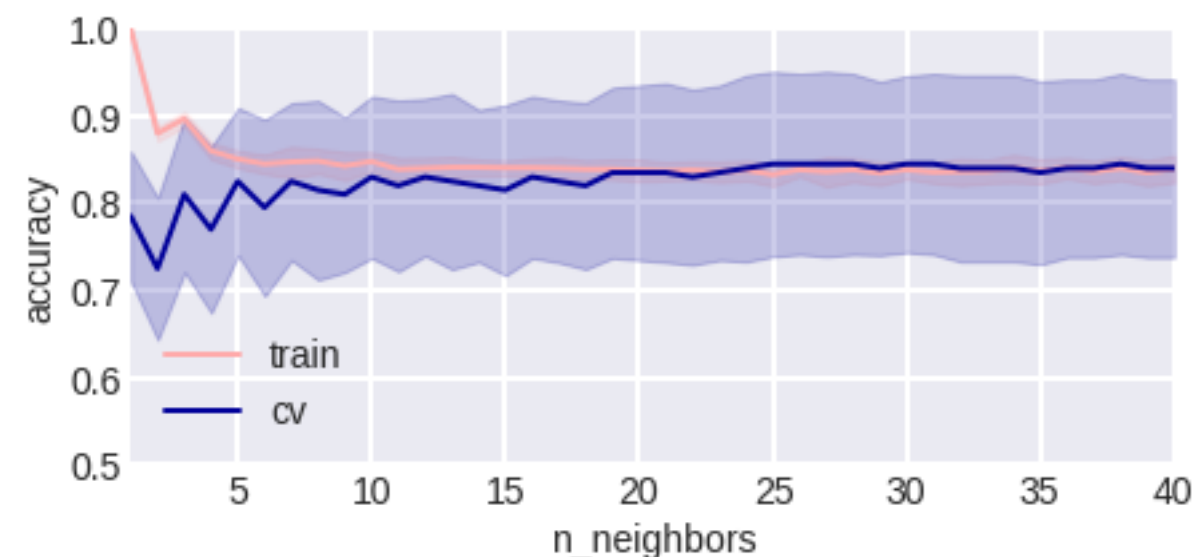
Подбор гиперпараметров специальными методами контроля

```
# cv-контроль
from sklearn.model_selection import KFold
cv = KFold(n_splits=10, shuffle=True, random_state=2)
# модель
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=5)
# параметр
param_name = "n_neighbors"
# его значения
pars = np.arange(1, 41)

# сделать тест
from sklearn.model_selection import validation_curve

train_errors, test_errors = validation_curve(model,
      X, y,
      param_name=param_name,
      param_range=pars,
      cv=cv.split(X),
      scoring='accuracy',
      n_jobs=-1)
```

будет дальше...

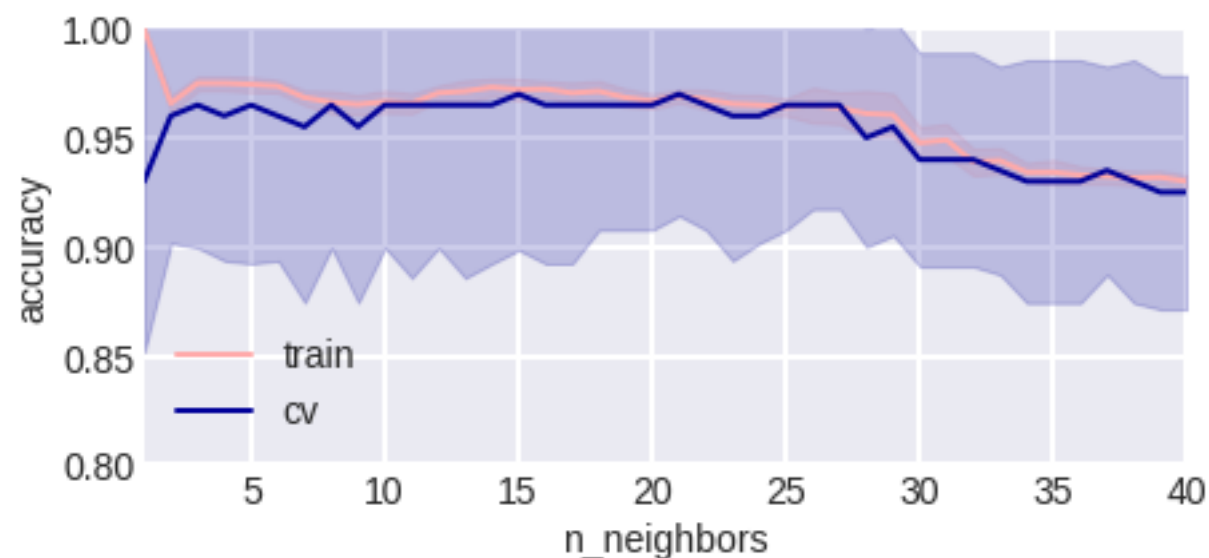


Как был получен второй график? с $k_{opt}=9$

Подбор гиперпараметров специальными методами контроля

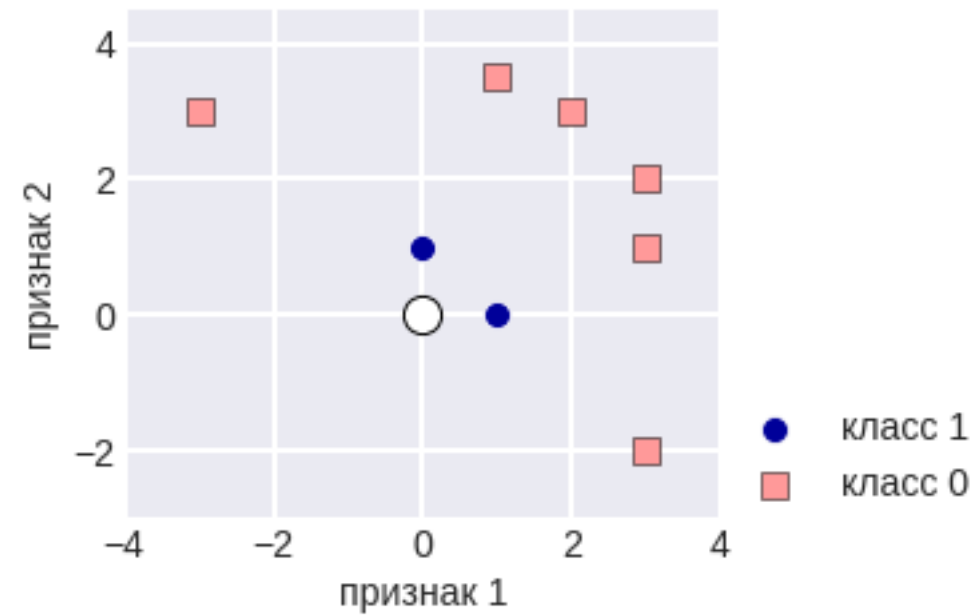
Как был получен второй график?

**Сделали дисбаланс классов –
стало невыгодно делать большие k**



**а это если взять декартово произведение нескольких полумесяцев
(выборка сбалансированная)**

Проблема



близкие соседи должны быть важнее

Весовые обобщения kNN

классика

$$\text{mode}(y_i \mid x_i \in N(x)) = \arg \max \sum_{t=1}^k I[y(x_t) = a]$$

обобщение

$$\arg \max \sum_{t=1}^k w_t I[y(x_t) = a]$$

разные весовые схемы:

$$w_1 \geq w_2 \geq \dots \geq w_k > 0$$

Весовые схемы

$$w_t = (k - t + 1)^\delta$$

$$k^\delta \geq (k - 1)^\delta \geq \dots \geq 1^\delta > 0$$

$$w_t = \frac{1}{t^\delta}$$

$$\frac{1}{1^\delta} \geq \frac{1}{2^\delta} \geq \dots \geq \frac{1}{k^\delta} > 0$$

$$w_t = K \left(\frac{\rho(x, x_t)}{h(x)} \right)$$

Последний способ хорош только на картинках...

часто веса лучше отнормировать, чтобы сумма = 1

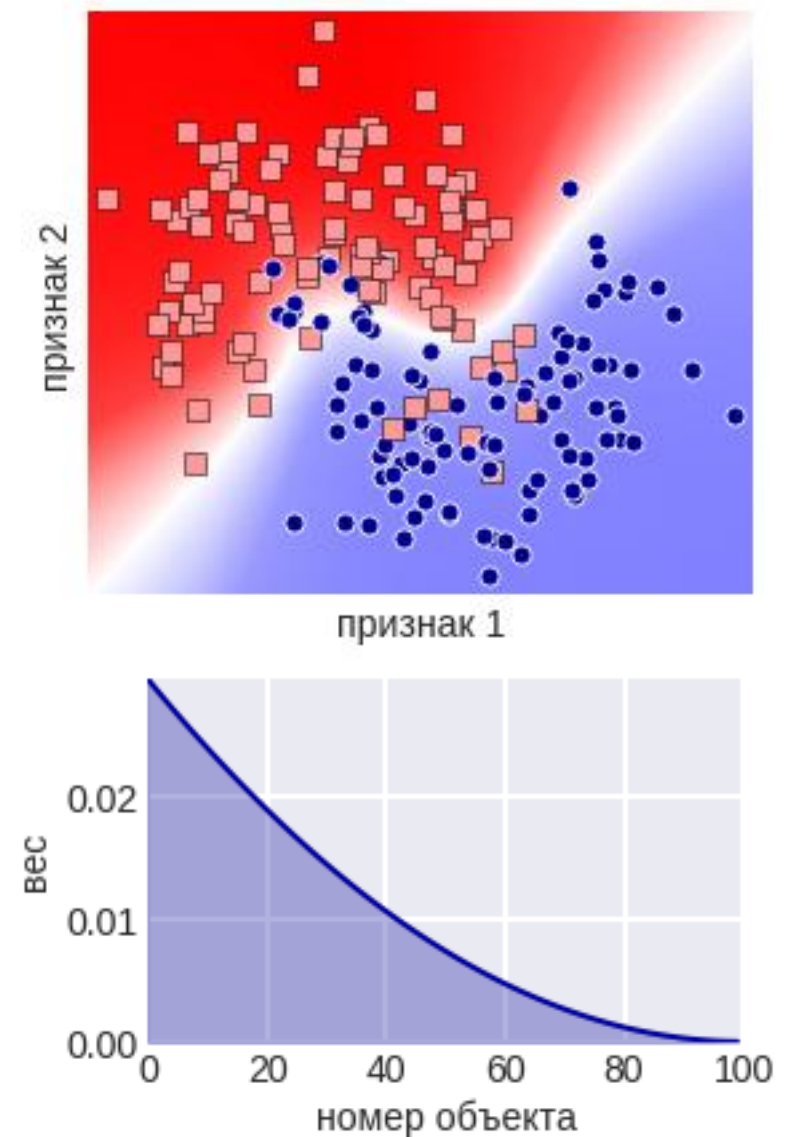
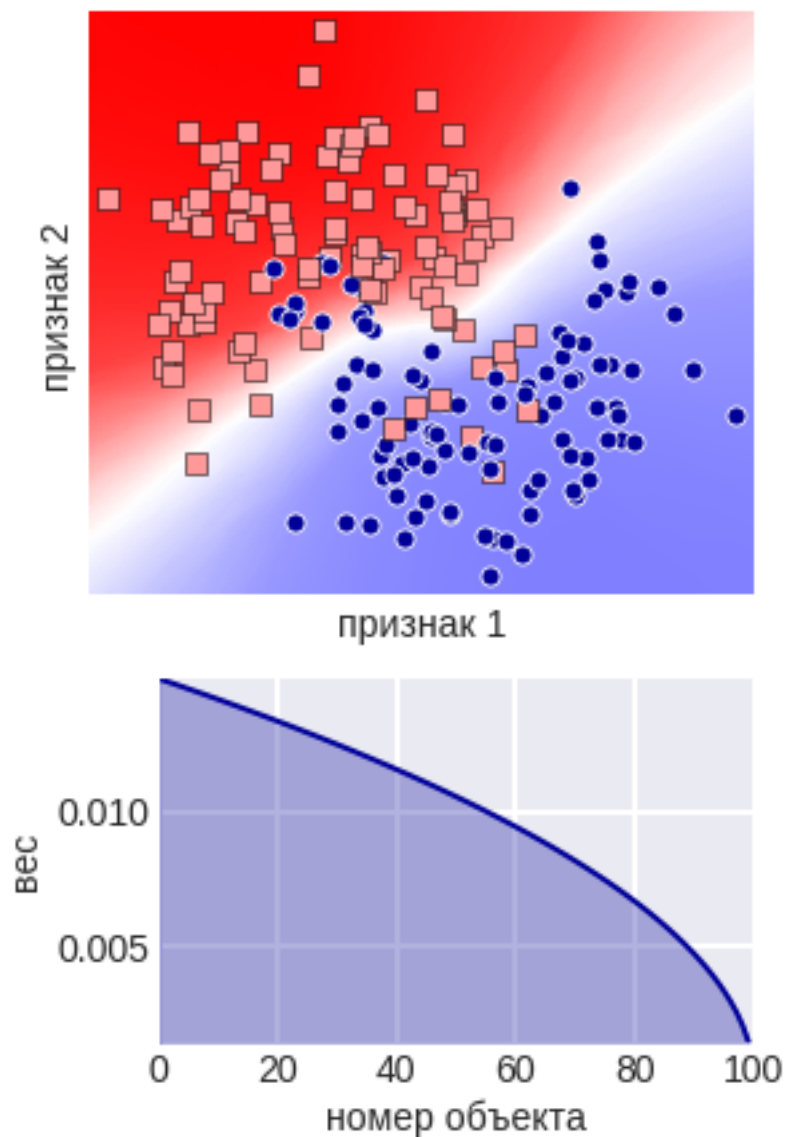
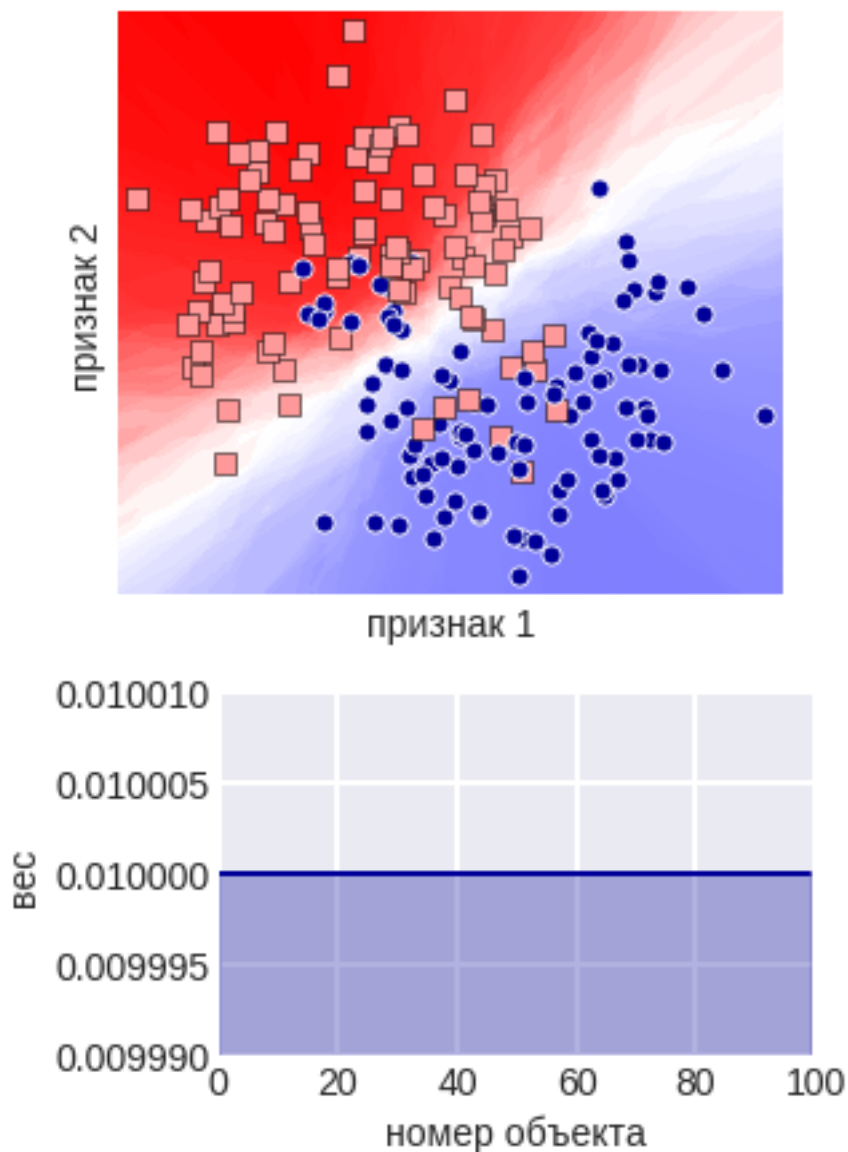
главное преимущество –

богатое пространство вероятности в задачах классификации!

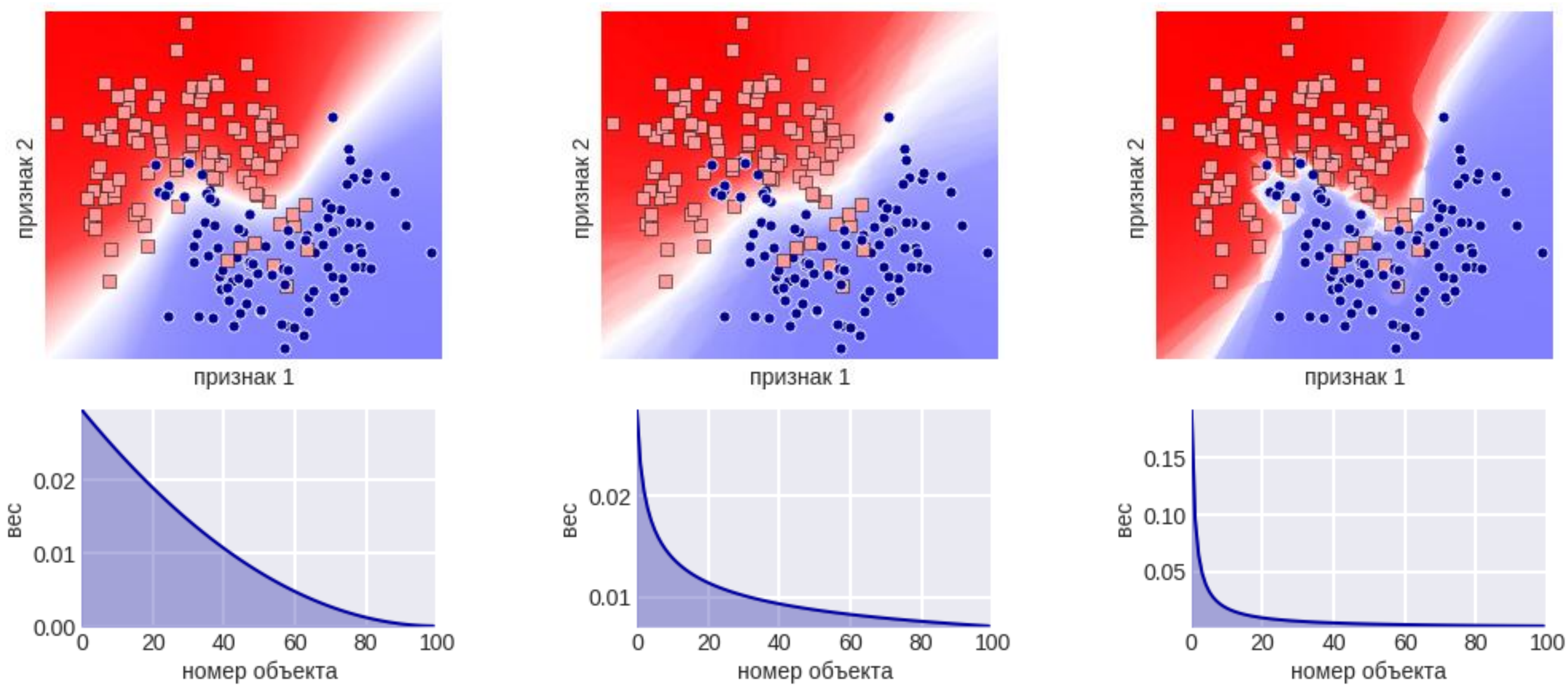
можем различать степени принадлежности у разных объектов

потом будем подробно разбирать

Весовые обобщения kNN



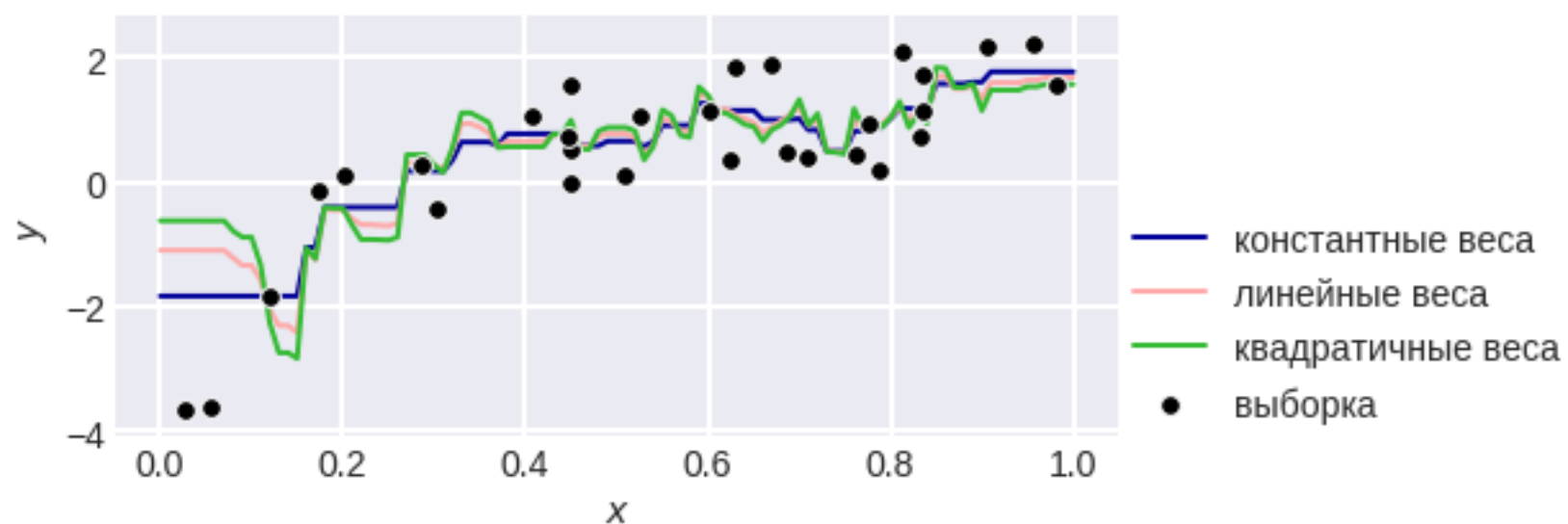
Весовые обобщения kNN



Весовые обобщения в регрессии

$$\frac{\sum_{t=1}^k w_t y(x_t)}{\sum_{t=1}^k w_t}$$

пример для 5NN



Эффект почти не заметен, дальше будет обобщение – регрессия Надарая-Ватсона

Метрики

Расстояние (метрика) на X – функция $\rho(x, z): X \times X \rightarrow \mathbb{R}$

- 1.** $\rho(x, z) \geq 0$
- 2.** $\rho(x, z) = 0 \Leftrightarrow x = z$ (**без – полуметрика/псевдометрика**)
- 3.** $\rho(x, z) = \rho(z, x)$
- 4.** $\rho(x, z) + \rho(z, v) \geq \rho(x, v)$

- **Минковского L_p**
 - **Евклидова L_2**
 - **Манхэттенская L_1**
- **Махалонобиса**

- **Canberra distance**
- **Хэмминга**
- **косинусное**
- **расстояние Джаккарда**

- **DTW**
- **Левенштейна**

Различные метрики

Евклидова (L2)

$$\sqrt{\sum_{i=1}^n (x_i - z_i)^2}$$

Общий вариант – Минковского (L_p)

$$\left(\sum_{i=1}^n |x_i - z_i|^p \right)^{1/p}$$

Предельный случай – Чебышева (L_∞)

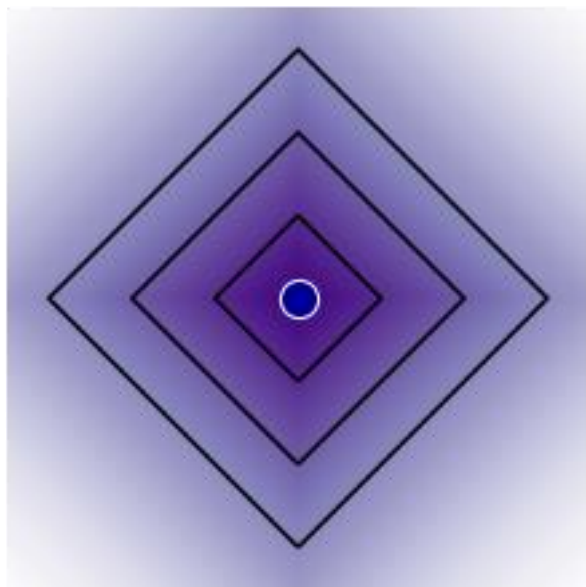
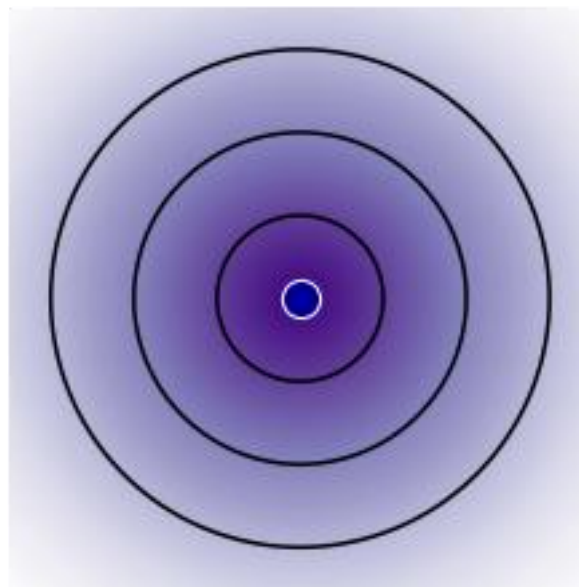
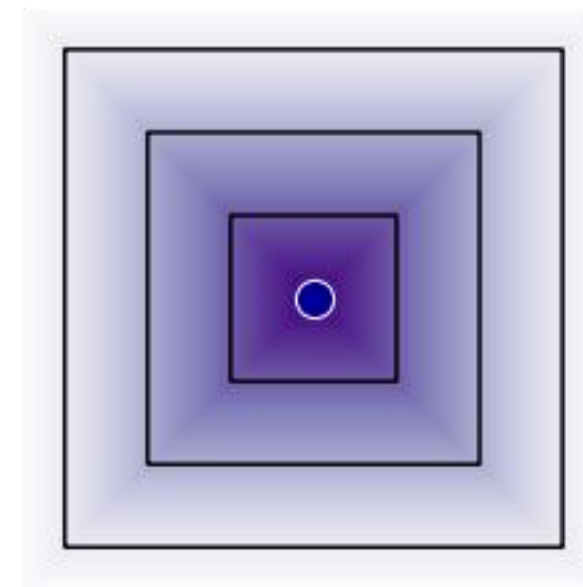
$$\left(\sum_{i=1}^n |x_i - z_i|^\infty \right)^{1/\infty} \sim \max_i |x_i - z_i|$$

Частный случай – Манхэттенская (L₁)

$$\sum_{i=1}^n |x_i - z_i|$$

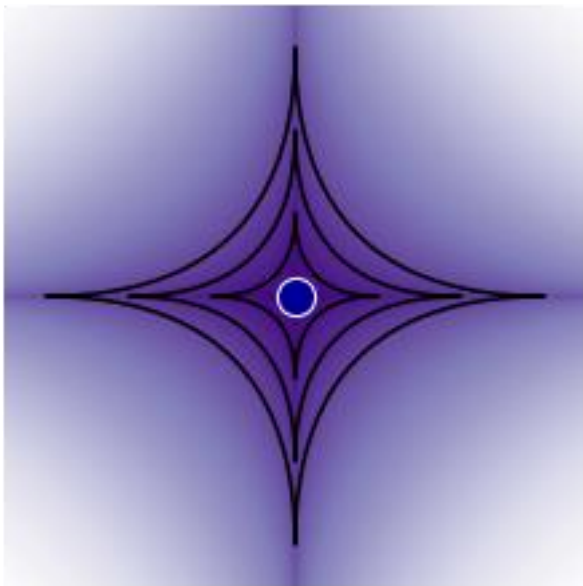
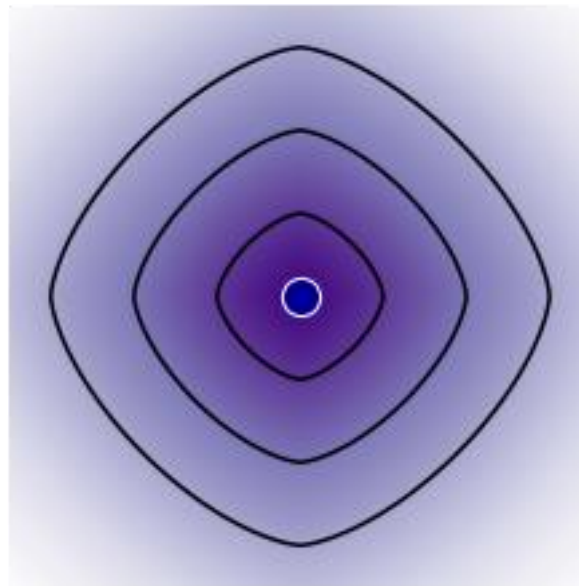
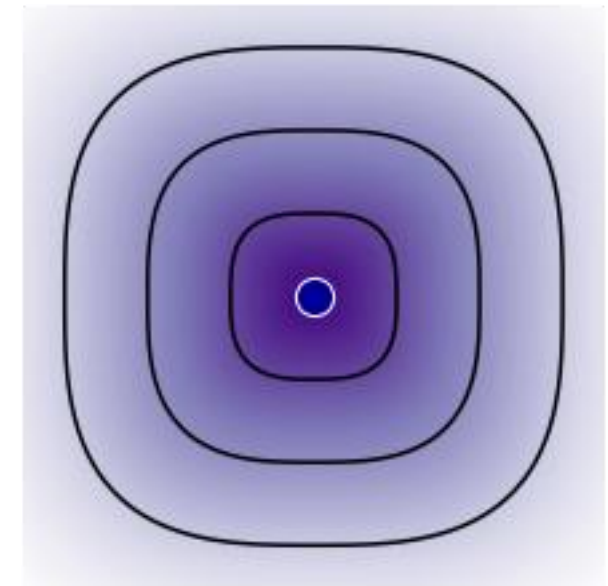
Различные метрики

$$\left(|x_1 - z_1|^p + |x_2 - z_2|^p \right)^{1/p}$$

 L_1  L_2  L_∞

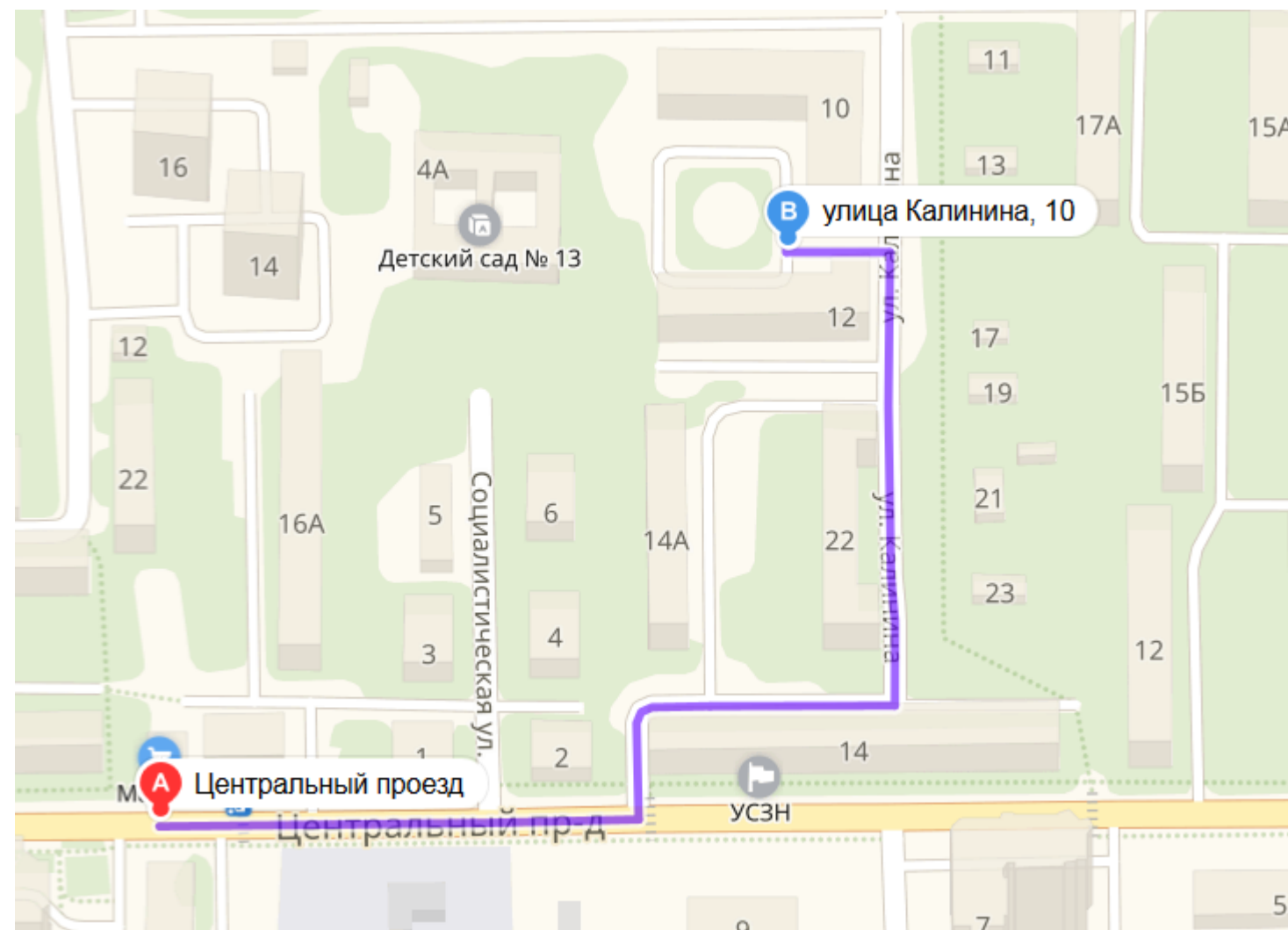
Различные метрики

$$\left(|x_1 - z_1|^p + |x_2 - z_2|^p \right)^{1/p}$$

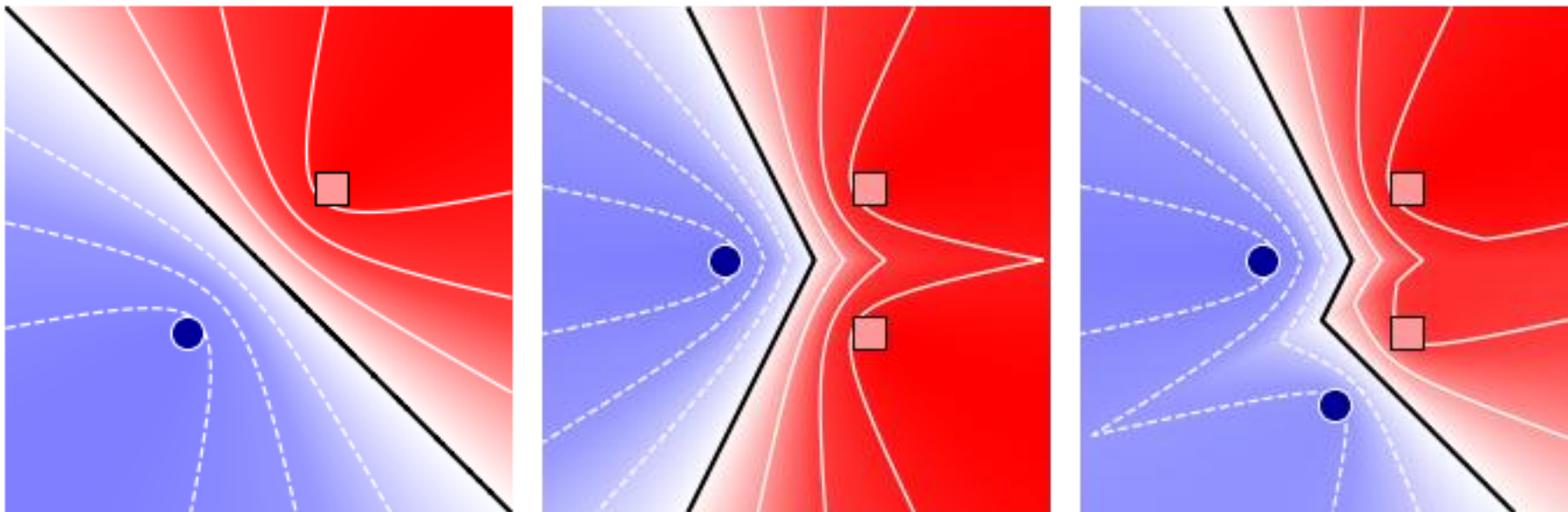
 $L_{0.5}$  $L_{1.5}$  L_3

что такое L_0 ?

Различные метрики



Разделяющие поверхности L_2



Разделяющие поверхности L_1

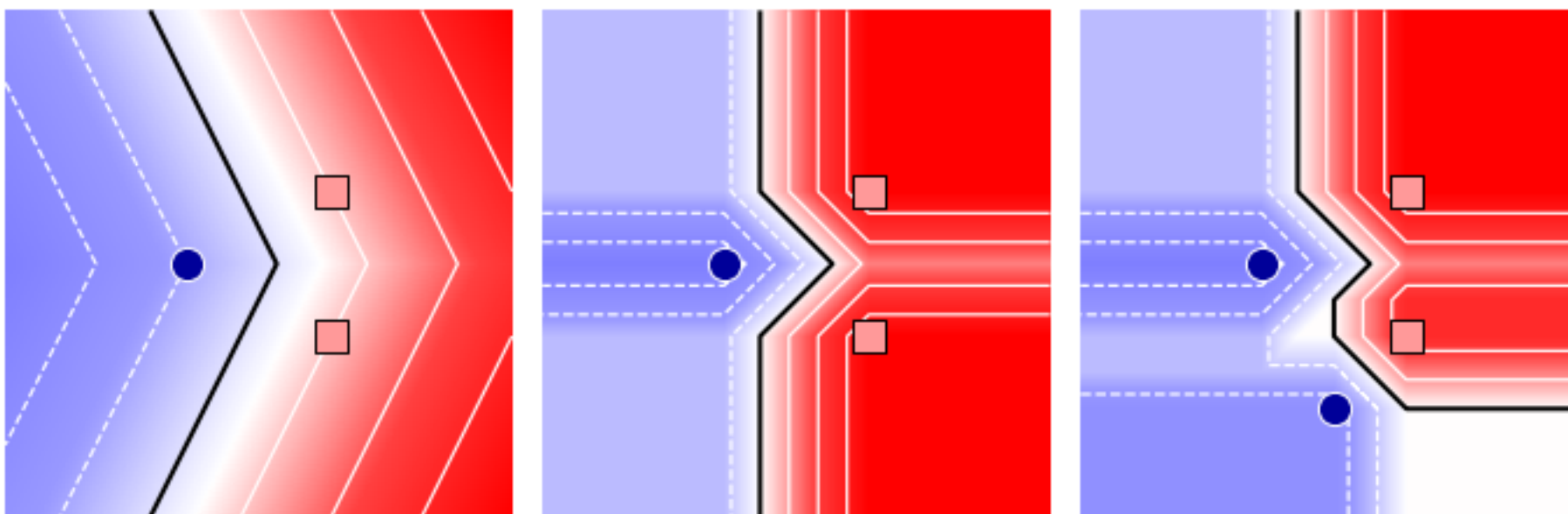
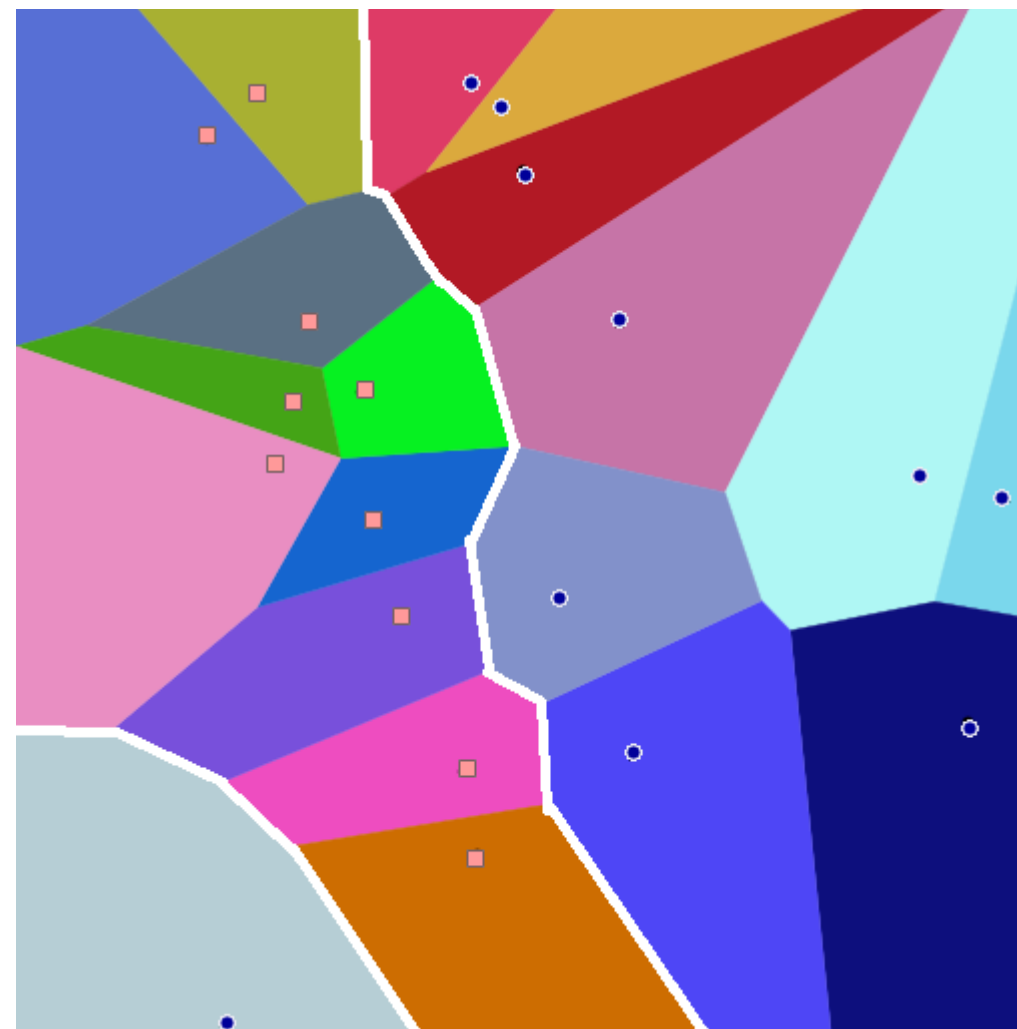
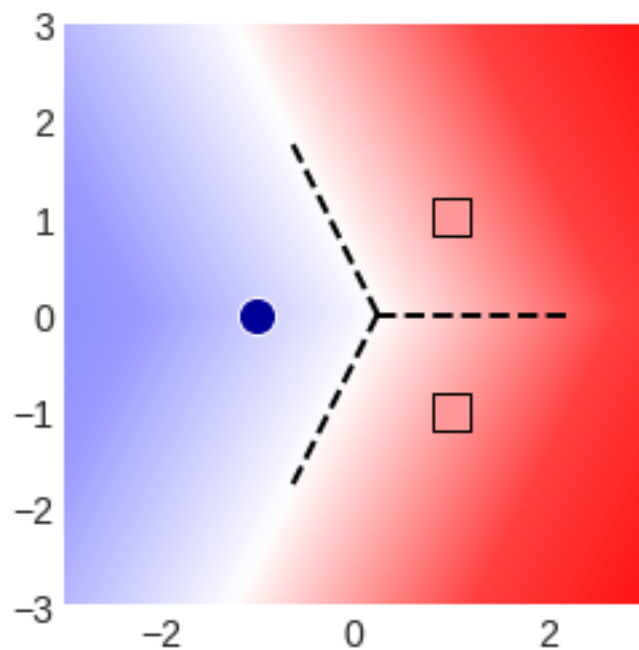
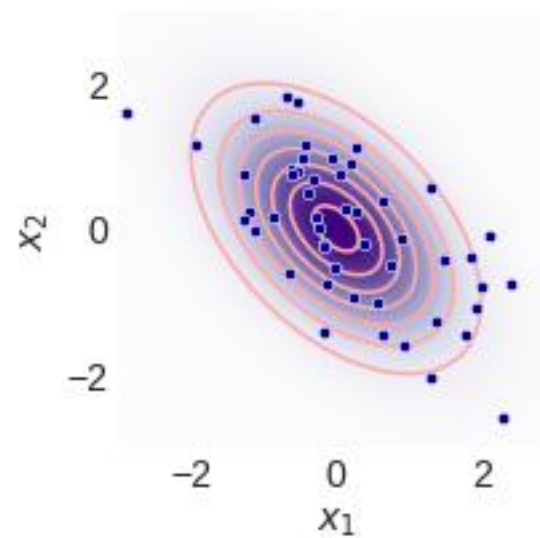


Диаграмма Вороного (Voronoi diagram)



https://en.wikipedia.org/wiki/Voronoi_diagram

Расстояние Махаланобиса (Mahalanobis distance)



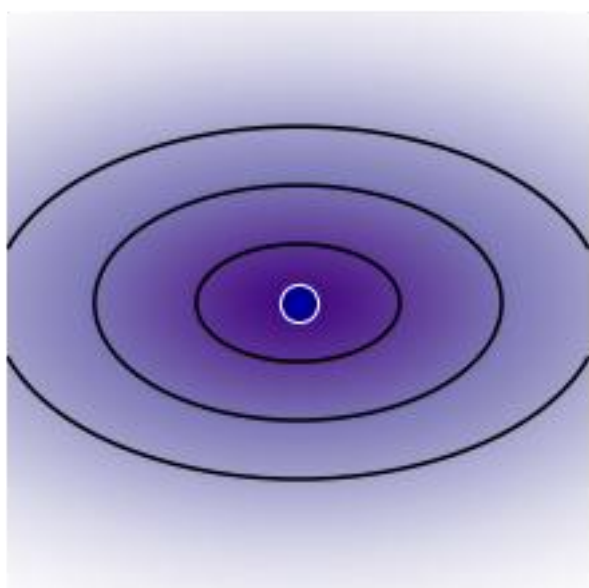
$$x \rightarrow \varphi(x) = \Sigma^{-1/2}(x - \mu)$$

стандартизует нормальные данные

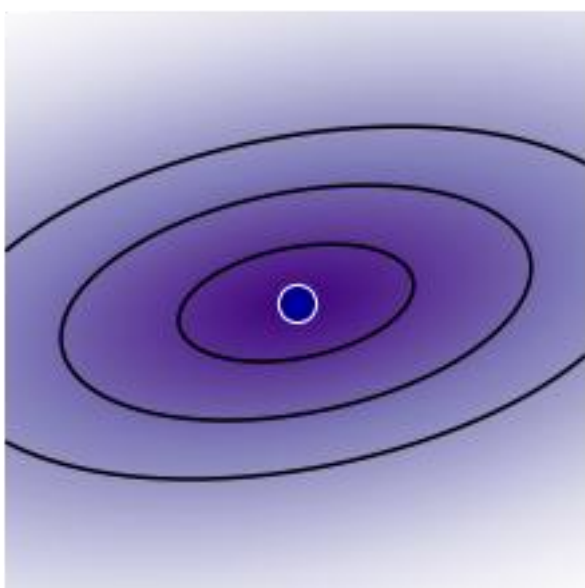
$$\text{norm}(\mu, \Sigma) \rightarrow \text{norm}(0, I)$$

$$\rho(x, z) = \rho_{L_2}(\varphi(x), \varphi(z)) = \sqrt{(\varphi(x) - \varphi(z))^T (\varphi(x) - \varphi(z))} = \sqrt{(x - z)^T \Sigma^{-1} (x - z)}$$

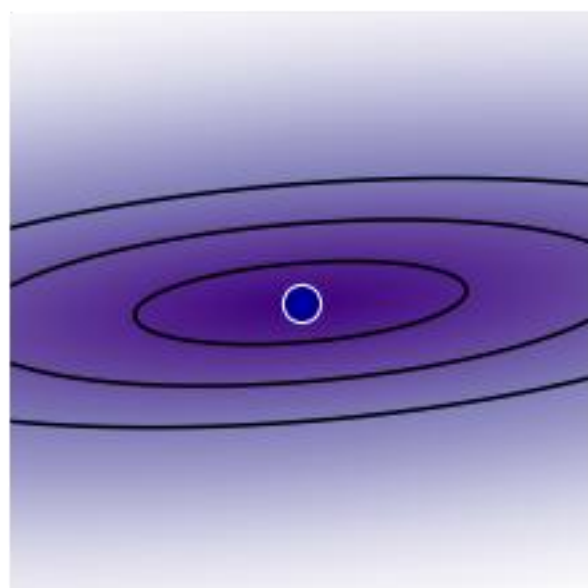
Расстояние Махаленобиса



$$\Sigma = \begin{bmatrix} 1.5 & 0 \\ 0 & 0.5 \end{bmatrix}$$



$$\Sigma = \begin{bmatrix} 2 & 0.3 \\ 0.3 & 0.5 \end{bmatrix}$$



$$\Sigma = \begin{bmatrix} 4 & 0.3 \\ 0.3 & 0.25 \end{bmatrix}$$

Расстояния

Canberra distance

https://en.wikipedia.org/wiki/Canberra_distance

$$\frac{1}{n} \sum_{i=1}^n \frac{|x_i - z_i|}{|x_i| + |z_i|}$$

Хэмминга

$$\sum_{i=1}^n I[x_i \neq z_i]$$

Косинусная мера сходство

$$\cos(x, z) = \frac{x^T z}{\|x\| \cdot \|z\|}$$

если работать с нормированными векторами,
достаточно рассматривать скалярное произведение

Расстояния

Расстояние Джаккарда (на множествах)

$$1 - \frac{|X \cap Z|}{|X \cup Z|}$$

Расстояние на множествах индуцирует
расстояние на бинарных векторах

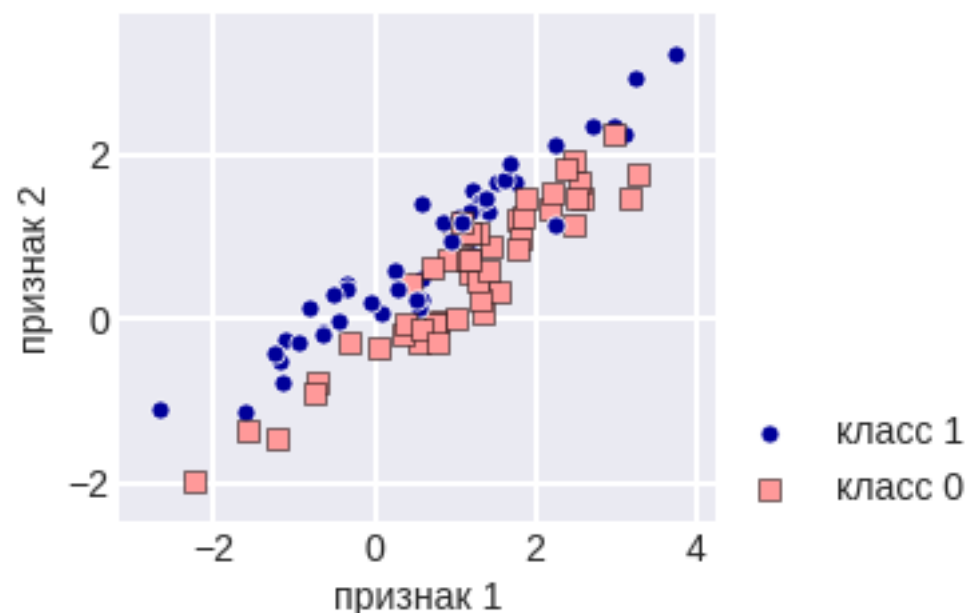
$$1 - \frac{x^T z}{x^T x + z^T z - x^T z} = \frac{x^T x + z^T z - 2x^T z}{x^T x + z^T z - x^T z}$$

тут есть много разных вариантов...

Проблема выбора метрики

- **зависимость от масштаба**
нормировка признаков
однородные признаки
смесь метрик
- **можно выбирать не метрику, а близость**
пример: косинусная мера сходства
- **часто выбор функции расстояния, как ни странно,
довольно прост...**

Обучение метрики: Metric Learning



**Признаковое пространство явно
нуждается в корректировке**

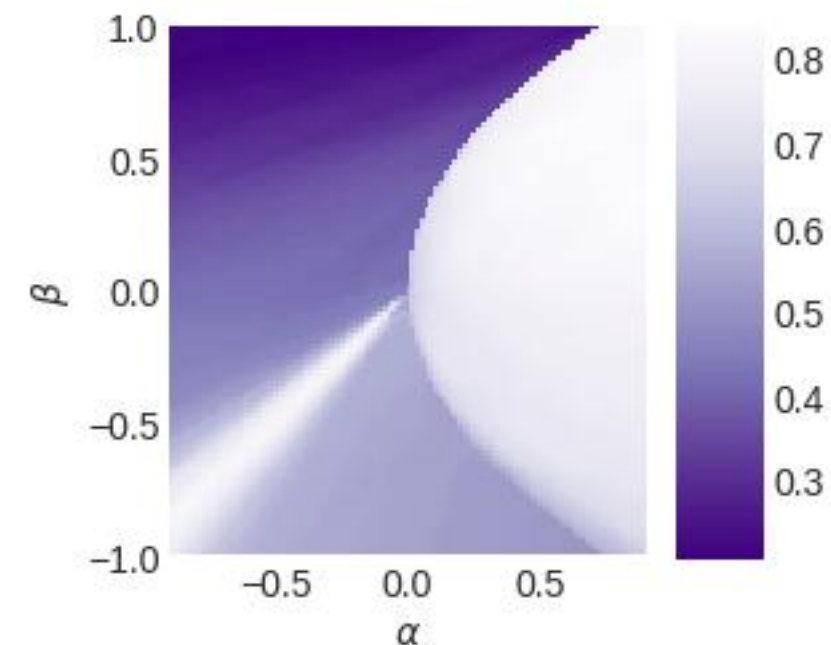
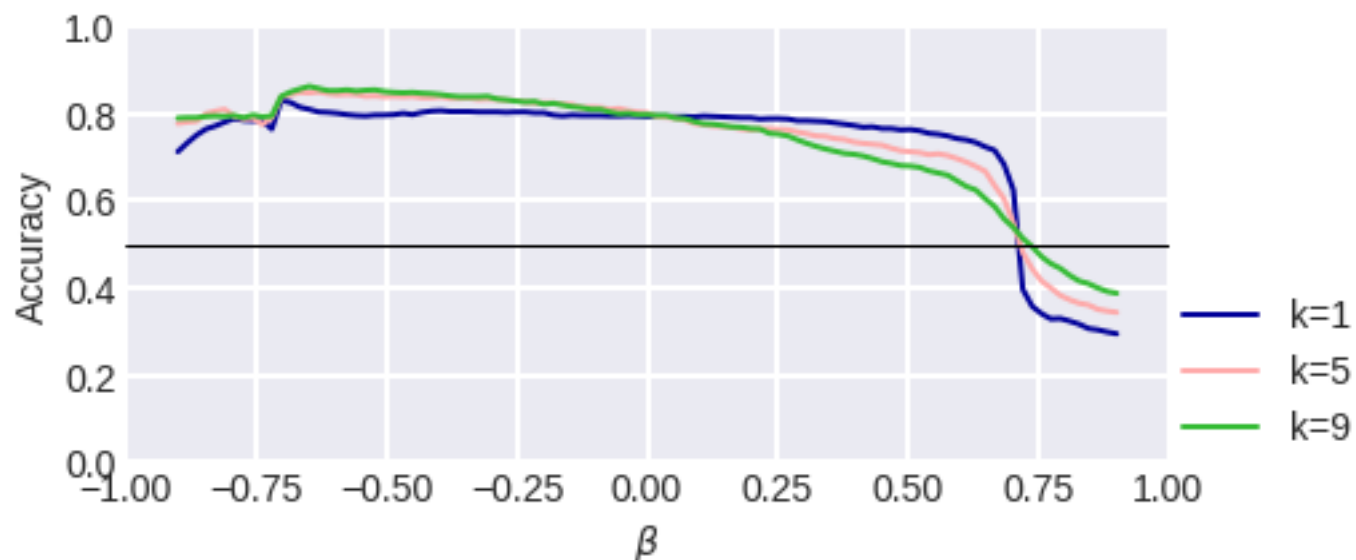
$$x \rightarrow \varphi(x) = \Sigma^{-1/2}(x - \mu)$$
$$\rho(x, z) = \sqrt{(x - z)^T \Sigma^{-1}(x - z)}$$

**Можно считать, что Σ^{-1} – матрица
параметров**

Оптимизировать

- **Качество kNN**
- **Расстояния до своих / чужих**

Обучение метрики: Metric Learning



```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
```

```
n_p, small_eps = 11, 0.000001
Q = np.zeros([n_p, n_p])
LS1 = np.linspace(-0.9, 0.9, n_p)
LS2 = np.linspace(-1, 1, n_p)
```

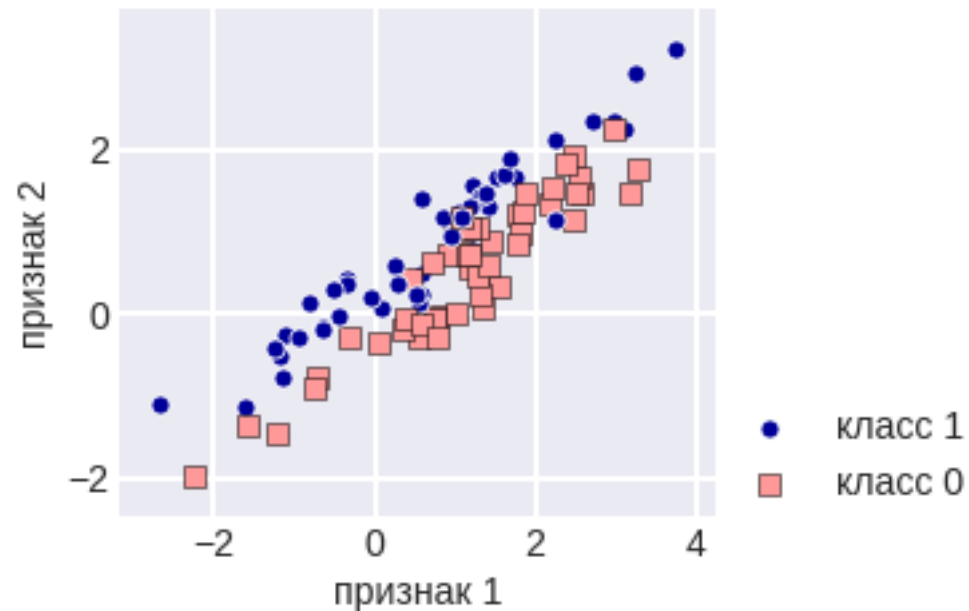
```
for i, b in enumerate(LS1):
    for j, a in enumerate(LS2):
        A = np.array([[a, b], [b, 1]])
        if np.abs(np.linalg.det(A)) < small_eps:
            A = A + small_eps * np.eye(2)
        model = KNeighborsClassifier(n_neighbors=5, metric='mahalanobis',
                                   metric_params={'V': np.linalg.inv(A)})
        model.fit(X, y)
        Q[i, j] = accuracy_score(y2, model.predict(X2))
```

$$\Sigma = \begin{bmatrix} \alpha & \beta \\ \beta & 1 \end{bmatrix}$$

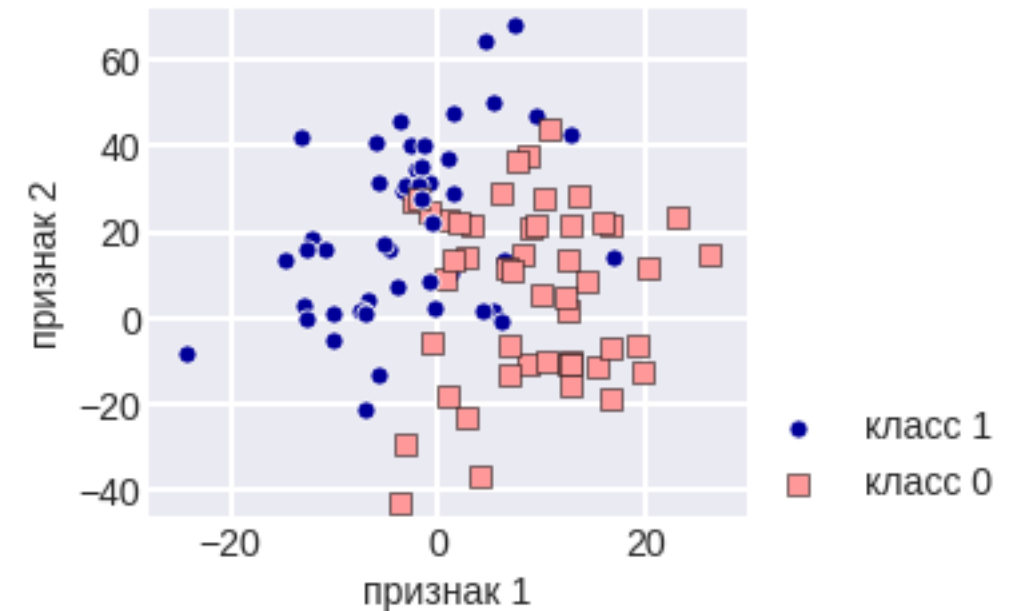
**здесь можно перебрать
параметры,
а вообще – методы
оптимизации
почему такой рис?**

Обучение метрики: Metric Learning

NeighborhoodComponentsAnalysis – улучшение качества 1NN с евклидовым расстоянием



до обучения



после обучения

```
from sklearn.neighbors import NeighborhoodComponentsAnalysis
nca = NeighborhoodComponentsAnalysis(max_iter=30, random_state=10)
nca = nca.fit(X, y)
X_embedded = nca.transform(X)
```

Обучение метрики: Metric Learning

**NeighborhoodComponentsAnalysis – математика
ищем преобразование**

$$x \rightarrow \varphi(x) = L(x - \mu)$$

$$p_{ij} = \frac{\exp(-\|Lx_i - Lx_j\|^2)}{\sum_{t \neq i} \exp(-\|Lx_i - Lx_t\|^2)}$$

если $x_i \in K_s$, то

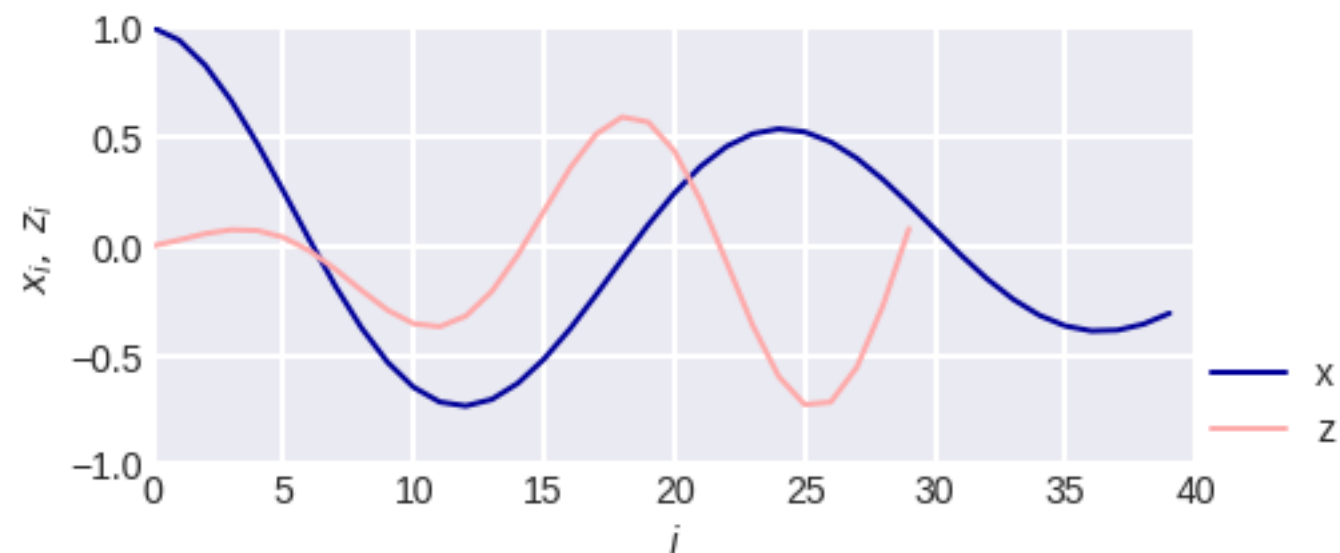
$$p_i = \sum_{x_j \in K_s} p_{ij}$$

(сумма вероятностей по объектам этого же класса)

задача оптимизации

$$\sum_{i=1}^m p_i \rightarrow \max$$

Метрики на временных рядах



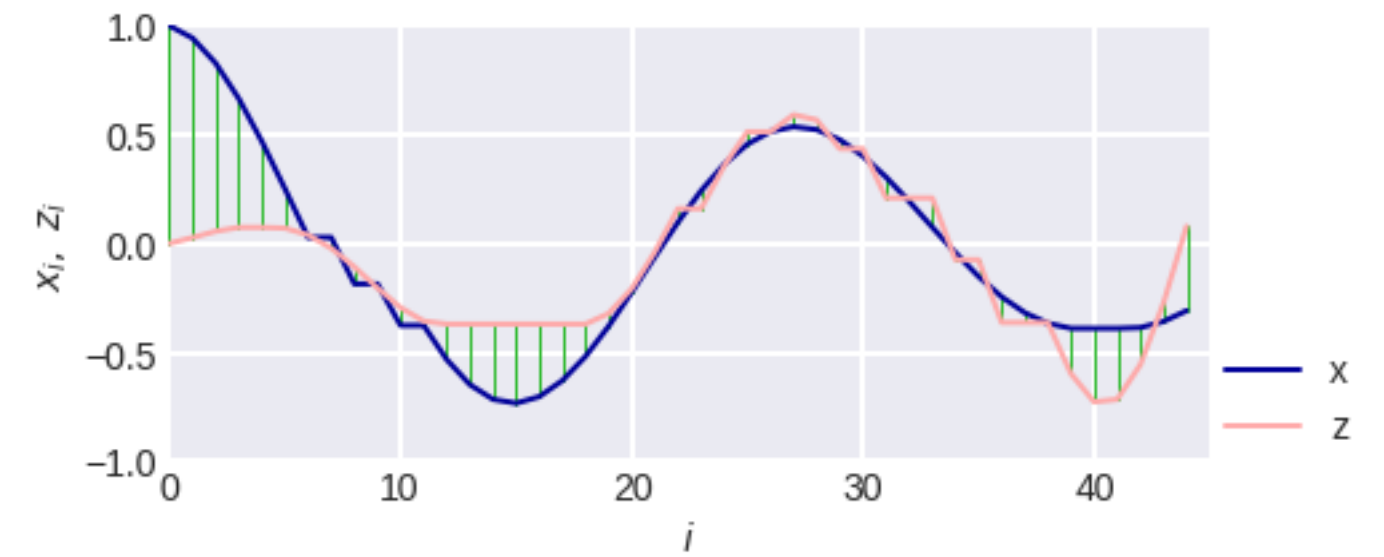
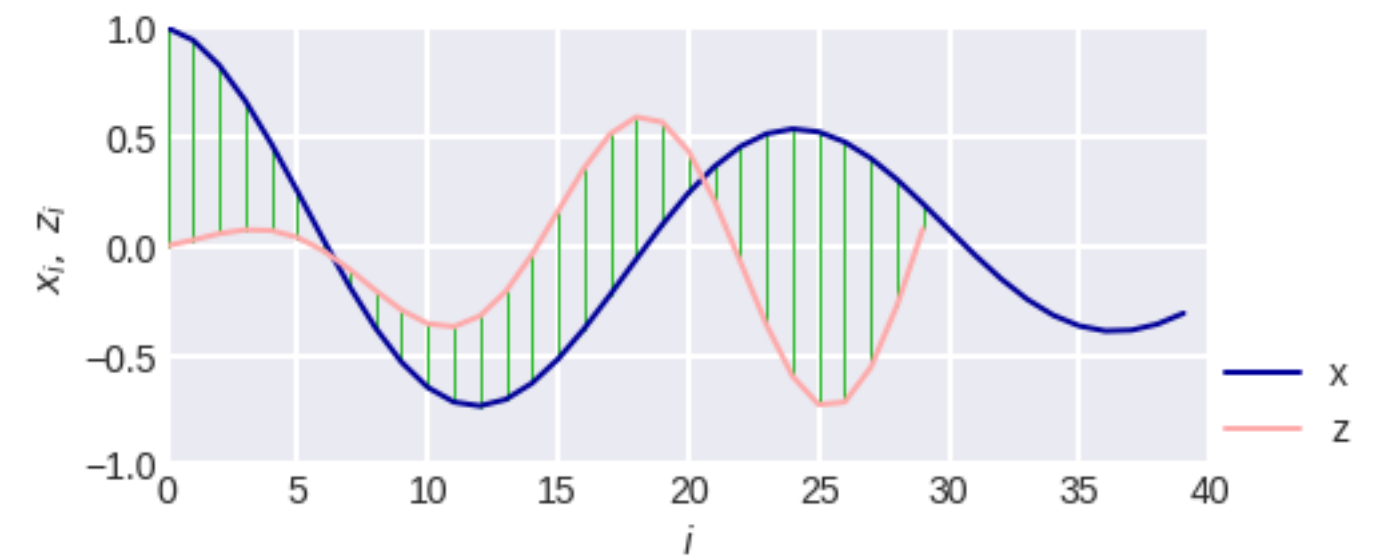
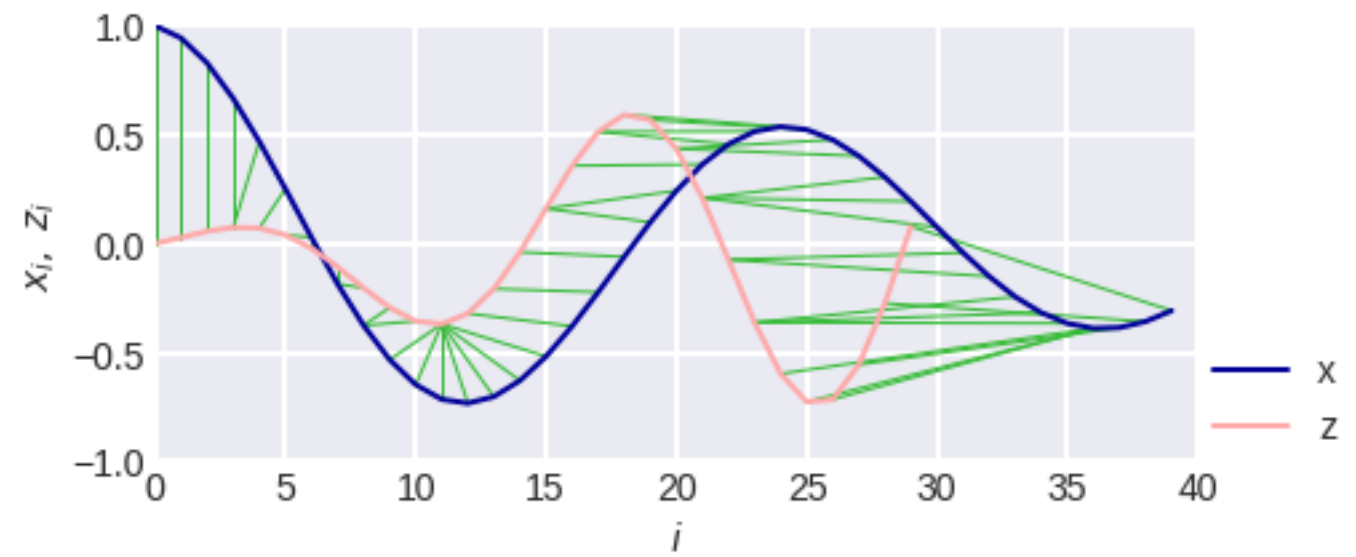
как ввести расстояние?

Ряды могут быть разной длины...
Как оценить похожесть формы?

Метрики на временных рядах

Евклидово расстояние

DTW = Dynamic time warping



Метрики на временных рядах

Пусть есть векторы (временные ряды)

$$x = (x_1, \dots, x_m)$$

$$z = (z_1, \dots, z_n)$$

срез –

$$x[:i] = (x_1, \dots, x_i)$$

рекурсивное определение –

$$\text{DTW}(x[:i], z[:j]) = \rho(x_i, z_j) + \min \begin{cases} \text{DTW}(x[:i-1], z[:j-1]) \\ \text{DTW}(x[:i-1], z[:j]) \\ \text{DTW}(x[:i], z[:j-1]) \end{cases}$$

начальные условия рекурсивного определения

$$\text{DTW}(x[:0], z[:0]) = 0$$

$$\text{DTW}(x[:i], z[:0]) = \text{DTW}(x[:0], z[:i]) = \infty$$

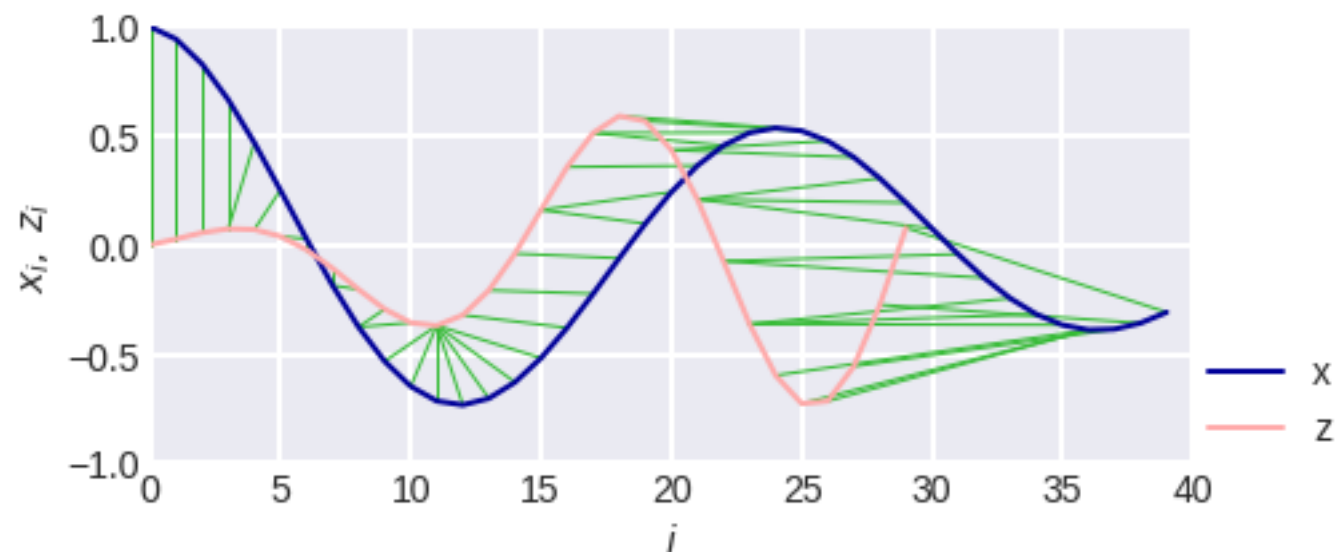
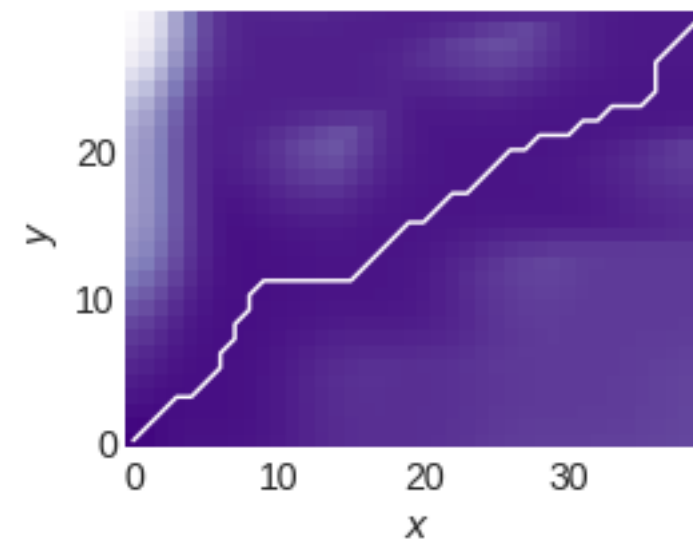
при $i > 0$

здесь нет нормировки... хотя может быть

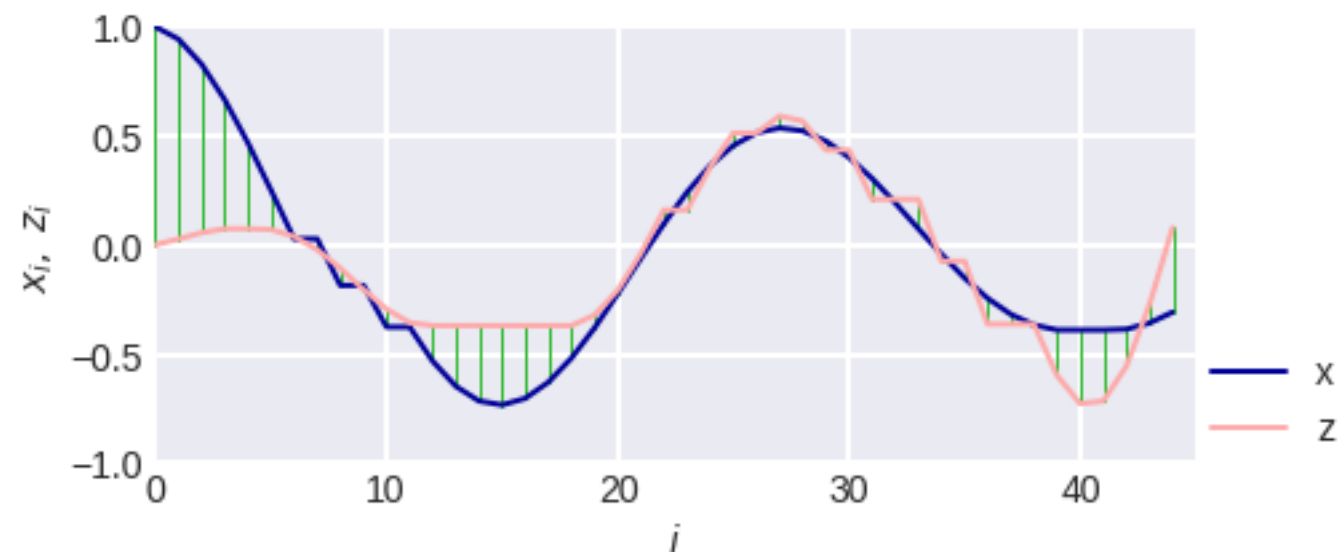
Метрики на временных рядах

Для адекватности и быстроты часто

$$|i - j| > r \Rightarrow \text{DTW}(x[:i], z[:j]) = +\infty$$



оптимальные соответствия точек



если разжать ряды соотв. образом

Расстояние Левенштейна

«Edit distance»

Расстояние между строками

Вводим элементарные операции правки:

- вставить букву
- удалить букву
- заменить букву

**расстояние – минимальное число операций,
с помощью которых из одной строки можно получить другую**
использование: исправление опечаток

Расстояние Левенштейна

определение аналогично DTW
(можно для каждой операции – свой штраф)

$$D(x[:i], z[:j]) = \min \begin{cases} \text{sub}(x_i, z_j) + D(x[:i-1], z[:j-1]) \\ \text{add}(x_i) + D(x[:i-1], z[:j]) \\ \text{add}(z_j) + D(x[:i], z[:j-1]) \end{cases}$$

$$\text{sub}(x_i, z_j) = \begin{cases} 0, & x_i = z_j, \\ 1, & x_i \neq z_j, \end{cases}$$

Приложения метрического подхода: нечёткий матчинг таблиц

пример АМ

матчинг таблиц разных аудиторий для рекламного агенства

Приложения метрического подхода: Ленкор

«VideoLectures.Net Recommender System Challenge» (ECML/PKDD Discovery Challenge 2011)

– написать рекомендательную систему в режиме холодного старта

Описание лекции

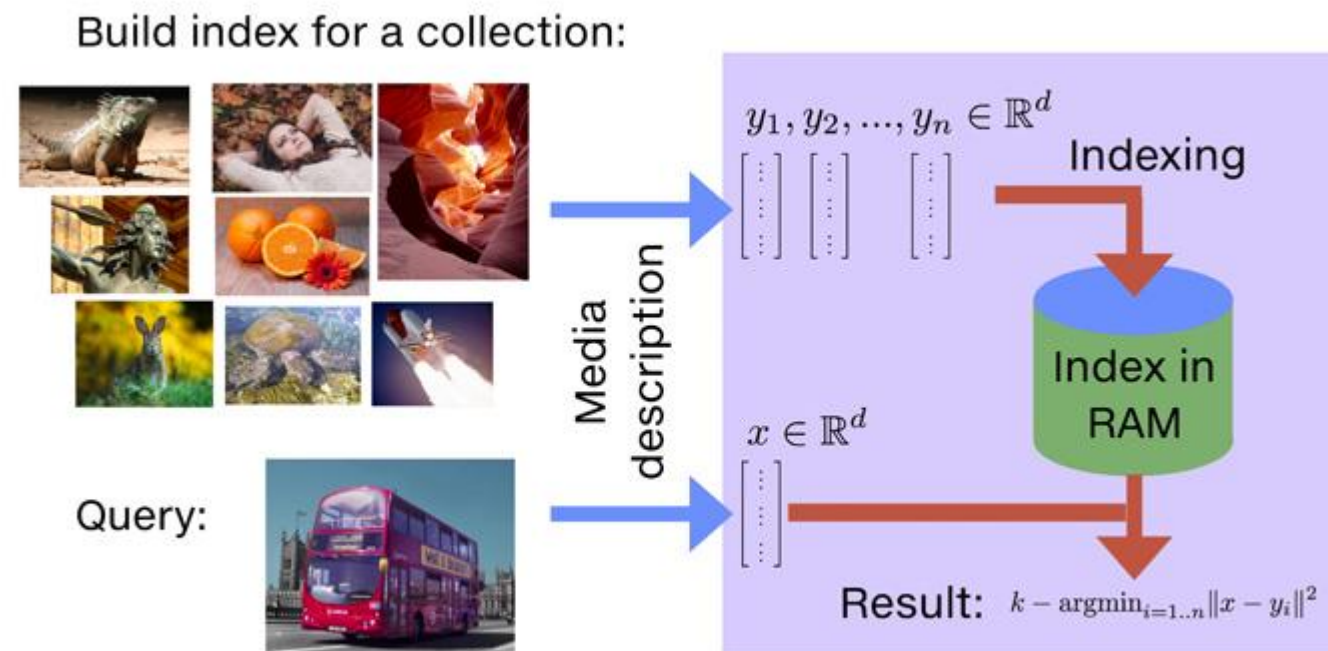
101, 'Lecture', 'eng', 'biology', '2008-12-04', '2009-02-12', 'Implementing a common framework on business', 'Professor Rudolf Smith', ...

$$\begin{aligned} \rho(\text{Lecture}_1, \text{Lecture}_2) = \\ = c_1 \cdot \rho_1(\text{Author}_1, \text{Author}_2) + c_2 \cdot \rho_2(\text{Title}_1, \text{Title}_2) + \dots + c_r \cdot \rho_r(\text{Subject}_1, \text{Subject}_2) \end{aligned}$$

**метрики можно параметризовать и настраивать параметры
«хитрый весовой учёт близости» – см. совместные просмотры**

Дьяконов А.Г. Алгоритмы для рекомендательной системы: технология LENKOR // Бизнес-Информатика, 2012, №1(19), С. 32–39.

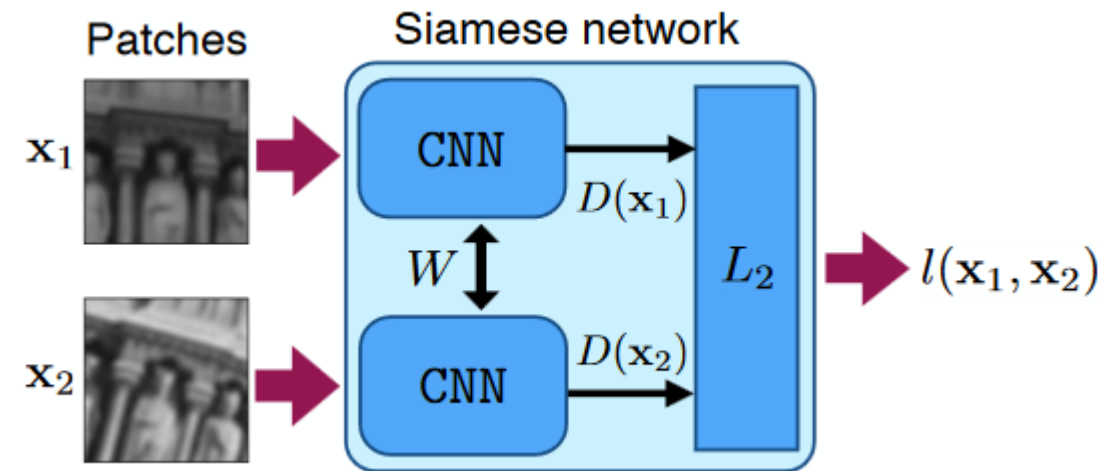
Приложения метрического подхода: поиск похожих объектов



звука, изображения, видео, ...

<https://code.fb.com/data-infrastructure/faiss-a-library-for-efficient-similarity-search/>

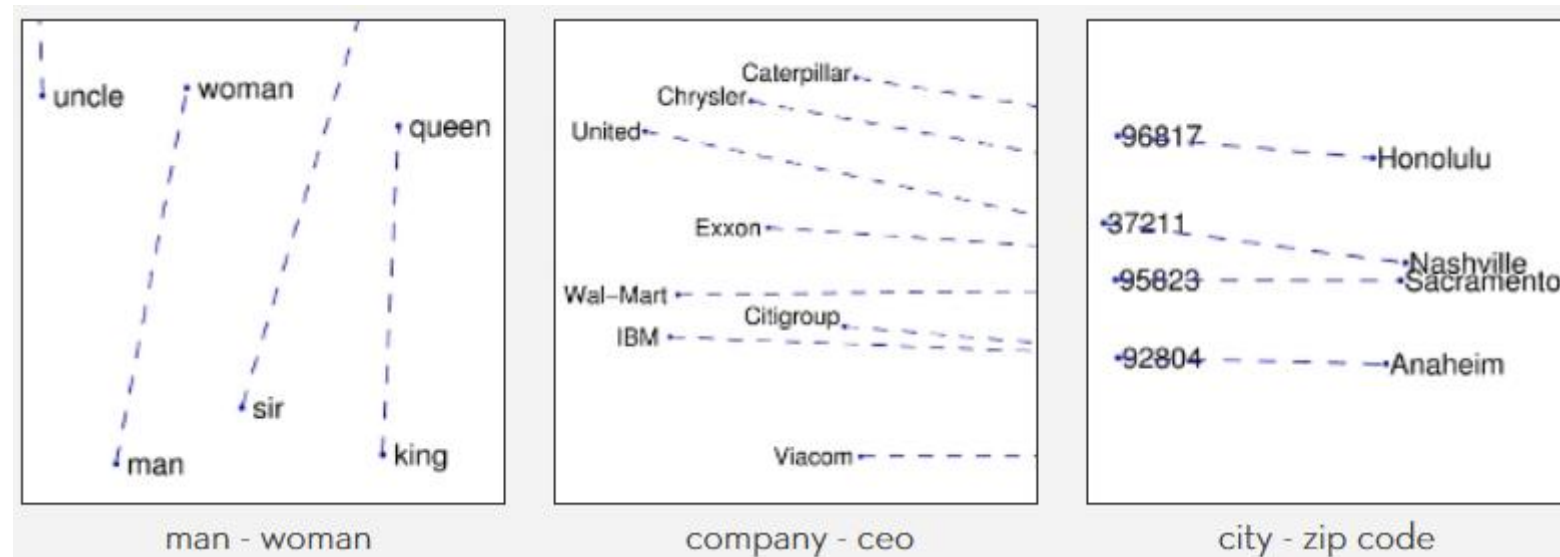
Приложения метрического подхода: Сиамские сети



Discriminative Learning of Deep Convolutional Feature Point Descriptors

[Simo-Serra et al., 2015 <http://icwww.epfl.ch/~trulls/pdf/iccv-2015-deepdesc.pdf>]

Приложения метрического подхода: GLOVE word vectors



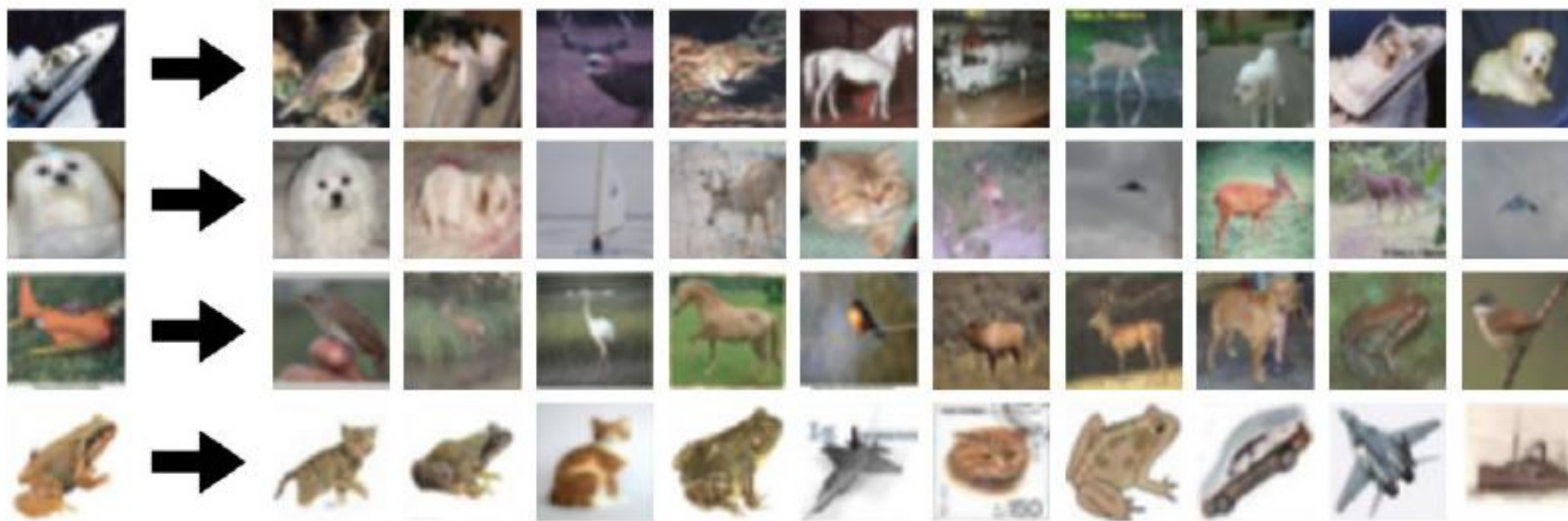
<https://nlp.stanford.edu/projects/glove/>

Приложения метрического подхода: SIFT image descriptors

TEXMEX

<http://corpus-texmex.irisa.fr/>

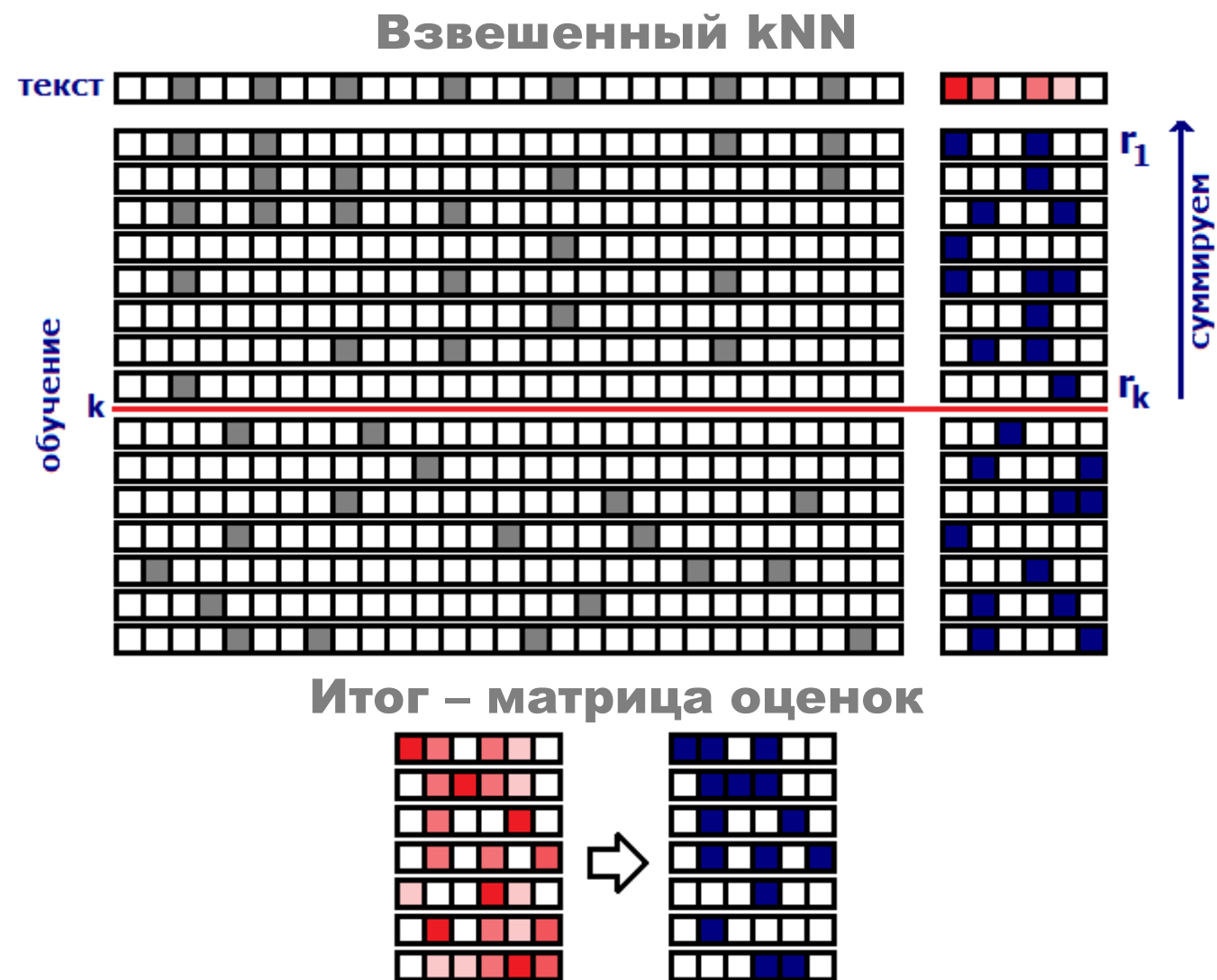
Приложения метрического подхода: простота интерпретаций



<http://cs231n.stanford.edu/2017/syllabus.html>

Приложения метрического подхода: классификация текстов

задача «Large Scale Hierarchical Text Classification»



<https://www.kaggle.com/c/lshhc>

Приложения метрического подхода: классификация текстов
задача «Large Scale Hierarchical Text Classification»

вычисление центроидов

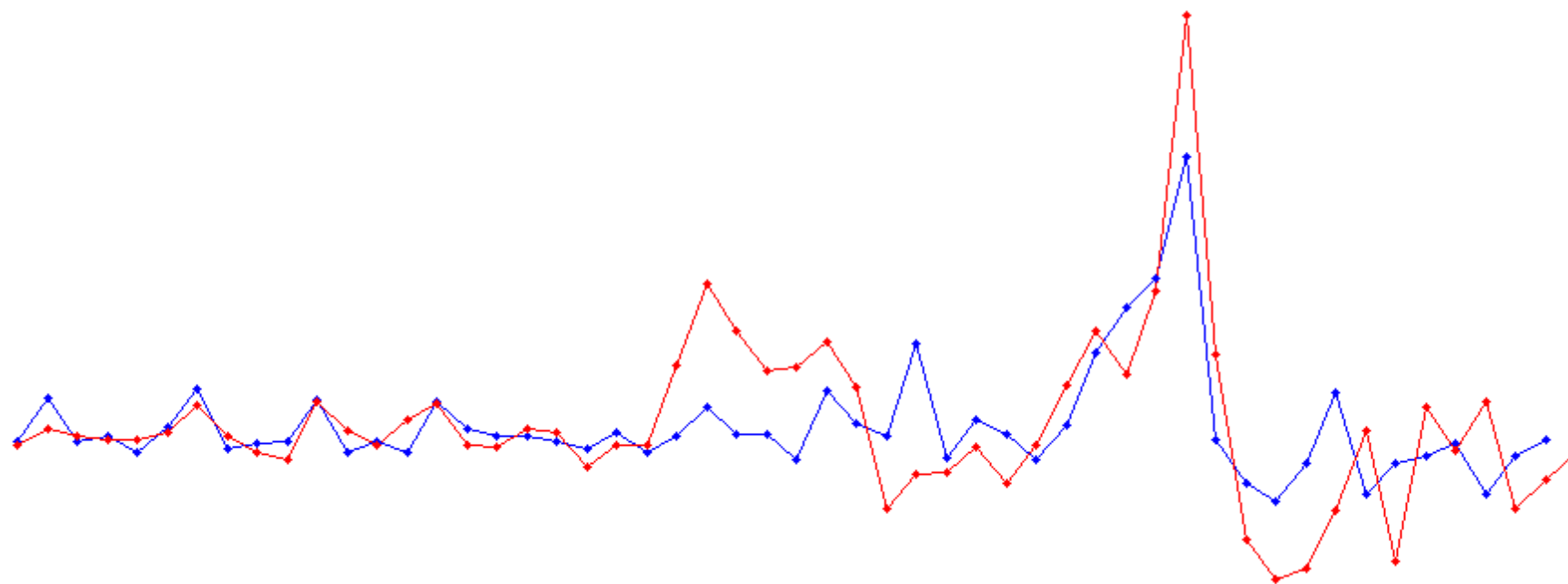
Приложения метрического подхода: прогнозирование рядов

$$\tilde{f} = (f_1, \dots, f_n) \rightarrow \tilde{g} = (f_1, \dots, f_n, f_{n+1}, \dots, f_{n+t})$$

$$\|A(f_{k-l+1}, \dots, f_k) - (f_{n-l+1}, \dots, f_n)\| \rightarrow \min_{k, \tilde{a}}$$

$$A(x_1, \dots, x_l) = (a_1 x_1 + a_2, \dots, a_l x_l + a_2)$$

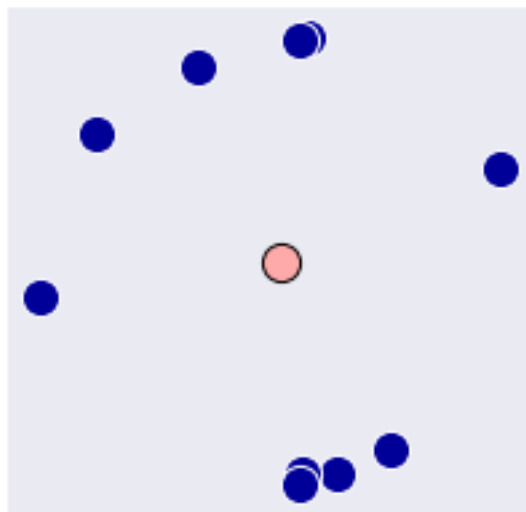
$$\sum_k c_k A(f_{k+1}, \dots, f_{k+l}), \sum_k c_k = 1$$



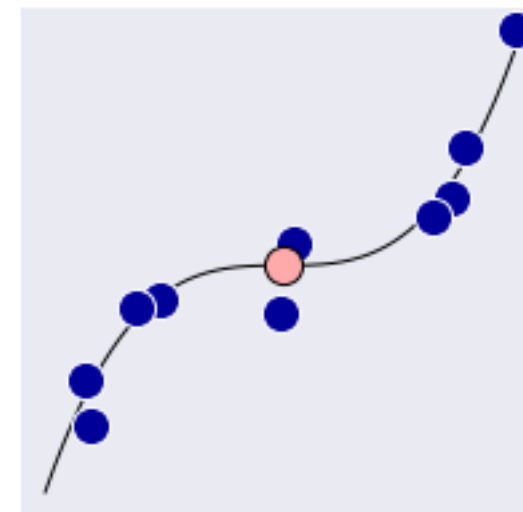
<http://www.neural-forecasting-competition.com/NN5/results.htm>

Проблема проклятия размерности

**в пространствах большой размерности все
объекты примерно на одном расстоянии**



но, к счастью, на реальных данных...



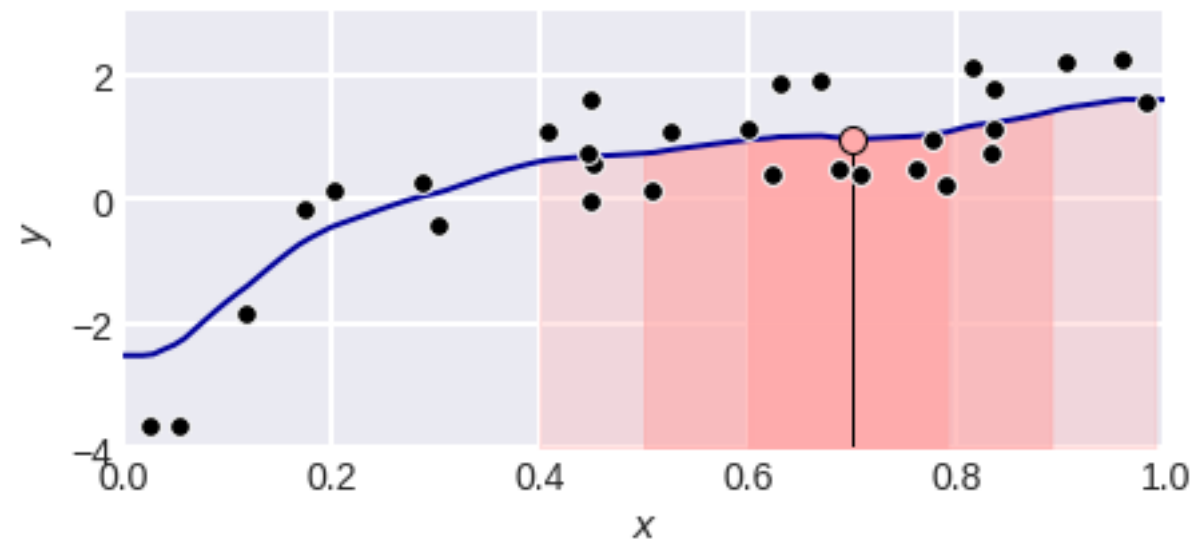
**есть внутренняя (intrinsic) размерность,
все объекты лежат около низкоразмерного многообразия
(low-dimensional manifold)**

Метрические алгоритмы

- + не требуется признаков описаний**
(достаточно уметь измерять расстояния / близости)
- + легко реализуемы**
- + интерпретируемость**
- + нет обучения**
- + мало гиперпараметров (хотя... есть метрика)**
- + можно учитывать контекст (с помощью метрики)**
- медленная классификация (зависит от объёма обучения)**
- требуется хранение всей обучающей выборки**
- требует подбора метрики (нормировки признаков)**

**Считается, что в пространствах гигантских размерностей стандартные метрики неадекватны (проклятие размерности),
но в реальности расположение объектов неслучайно – есть геометрия!!!**

Регрессия Надарая-Ватсона (Nadaraya-Watson regression, 1964)



ответ – взвешенное усреднение целевых значений

$$a(x) = \frac{w_1(x)y_1 + \dots + w_m(x)y_m}{w_1(x) + \dots + w_m(x)}$$

Регрессия Надарая-Ватсона (Nadaraya-Watson regression)

$$a(x) = \frac{w_1(x)y_1 + \dots + w_m(x)y_m}{w_1(x) + \dots + w_m(x)}$$

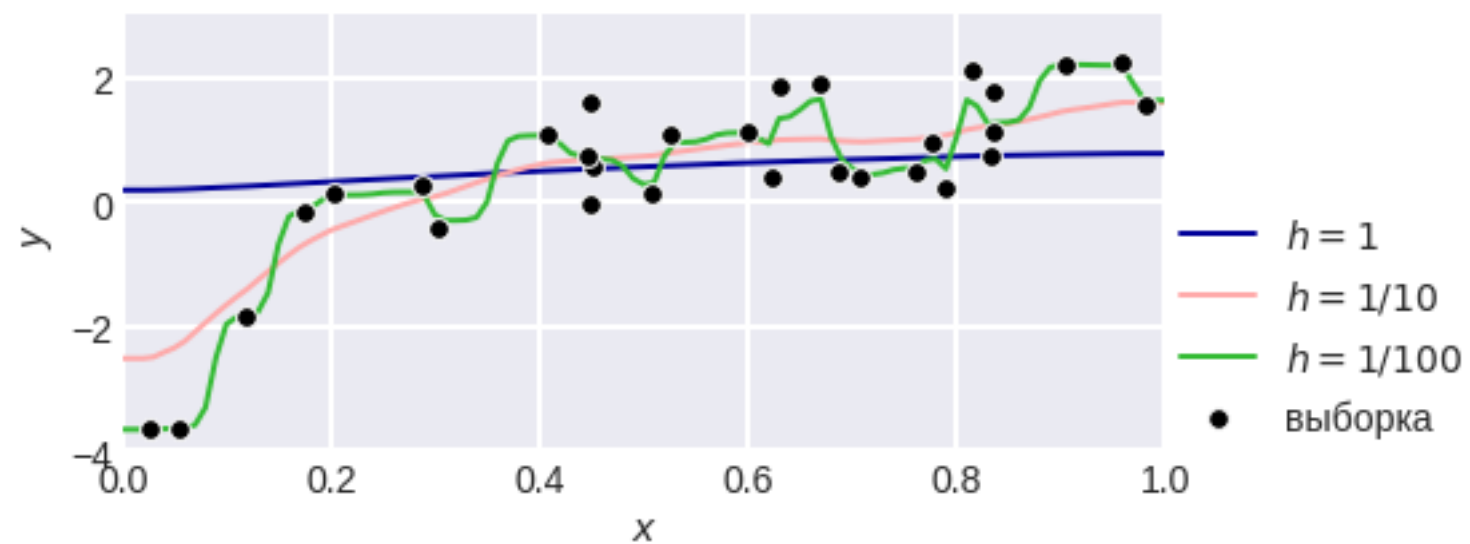
**Смысл весов – чем ближе объект обучения,
тем скорее ответ похож на его метку**

$$w_i(x) = K \left(\frac{\rho(x, x_i)}{h} \right)$$

Ядро с шириной h .

Здесь также как выше... (про функции ядра)

Регрессия Надарая-Ватсона (Nadaraya-Watson regression)



Регрессия Надарая-Ватсона (Nadaraya-Watson regression)

Смысл:

ответ алгоритма – решение оптимизационной задачи

$$\sum_{i=1}^m w_i(x)(a - y(x_i))^2 \rightarrow \min_a$$

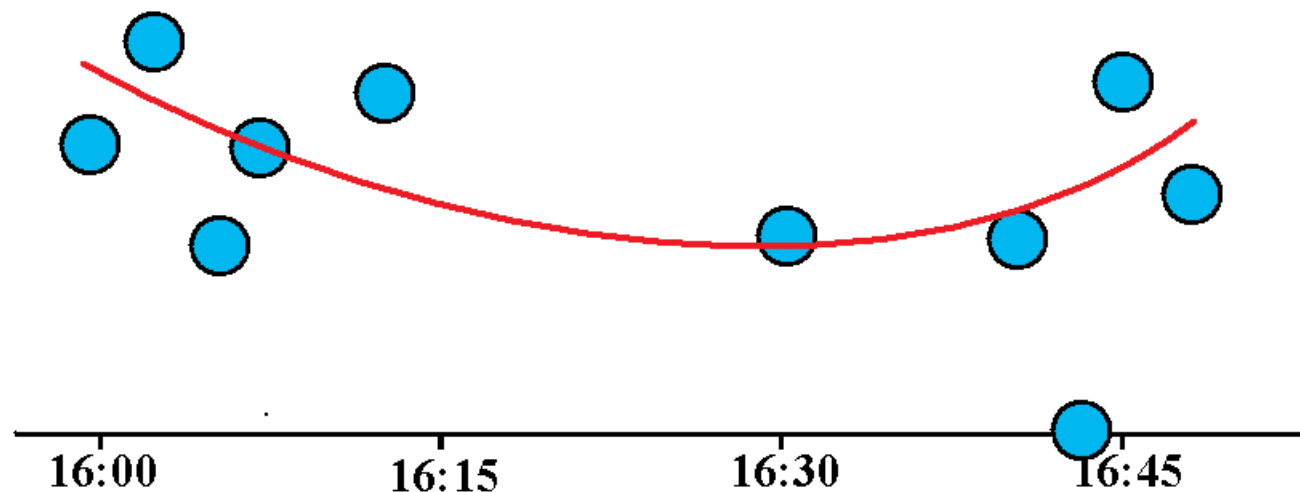
+ хорошее решение задачи сглаживания

– не решает задачи экстраполяции

Приложения регрессии Надарая-Ватсона

1. Сглаживание сигналов

2. «Многомерные» усреднения



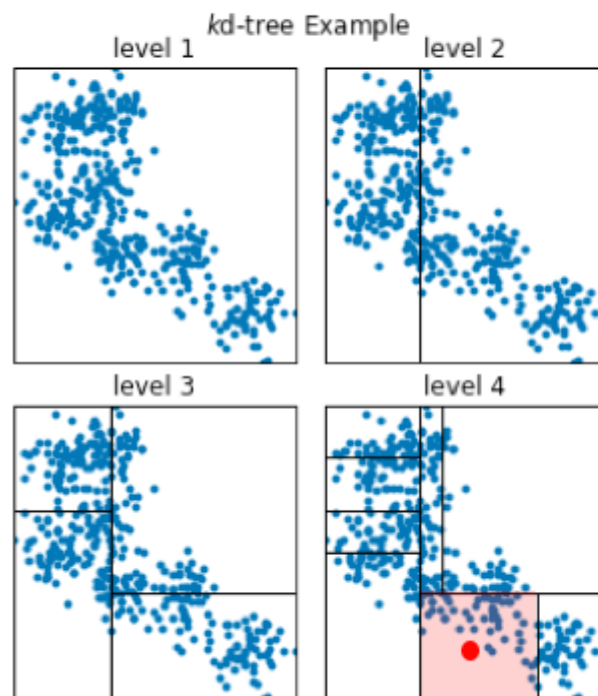
Эффективность

быстрые точные и приближённые методы поиска соседей

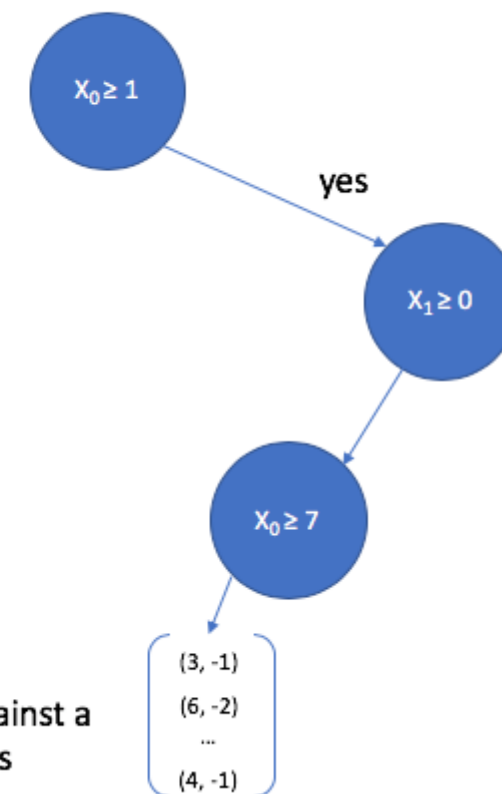
**Аппроксимация ~ с большой вероятностью находим точку,
которая не сильно дальше, чем ближайшая**

Эффективность: точное нахождение 1NN (Exact Nearest Neighbors) kD-деревья / k-d tree (Bentley, 1979) для евклидова пространства

Query: (4, -2)



Now we only have to
calculate distances against a
small **subset** of vectors



строим дерево

для точки – находим лист области пространства, которой она принадлежит
ищем ближайшую в листе (для точности надо смотреть соседние листья)

<https://www.jeremyjordan.me/scaling-nearest-neighbors-search-with-approximate-methods/>

Эффективность: точное нахождение 1NN (Exact Nearest Neighbors) ■

kD-деревья / k-d tree (Bentley, 1979) для евклидова пространства

Как строить дерево? Например, так:
ищем признак с максимальной дисперсией,
разбиваем по его медиане

Когда ищем ближайший
поднимаемся по дереву,
оценивая расстояние до области
(если оно меньше до текущего ближайшего – ищем там)

Эффективность: Random Projection Trees

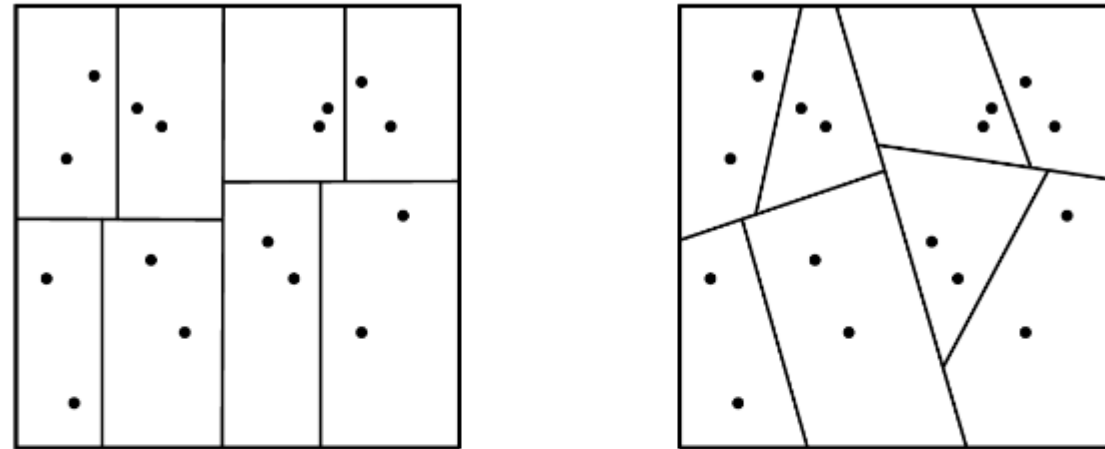


Figure 6.3: A comparison between how a k-d tree and a randomized partition tree divide up the feature space for two dimensional data (figure source: Dasgupta and Sinha 2015). Left: a k-d tree uses axis-aligned splits. Right: a randomized partition tree uses splits with directionality that is randomized (such as being drawn randomly from a unit ball).

Для приближённого поиска можно перебирать только объекты в листе
Можно построить несколько деревьев...

G. H. Chen, D. Shah Explaining the Success of Nearest Neighbor Methods in Prediction // Publisher: Now Foundations and Trends

Ещё название «Annoy»

<https://erikbern.com/2015/10/01/nearest-neighbors-and-vector-models-part-2-how-to-search-in-high-dimensional-spaces.html>

Эффективность: Ball Tree

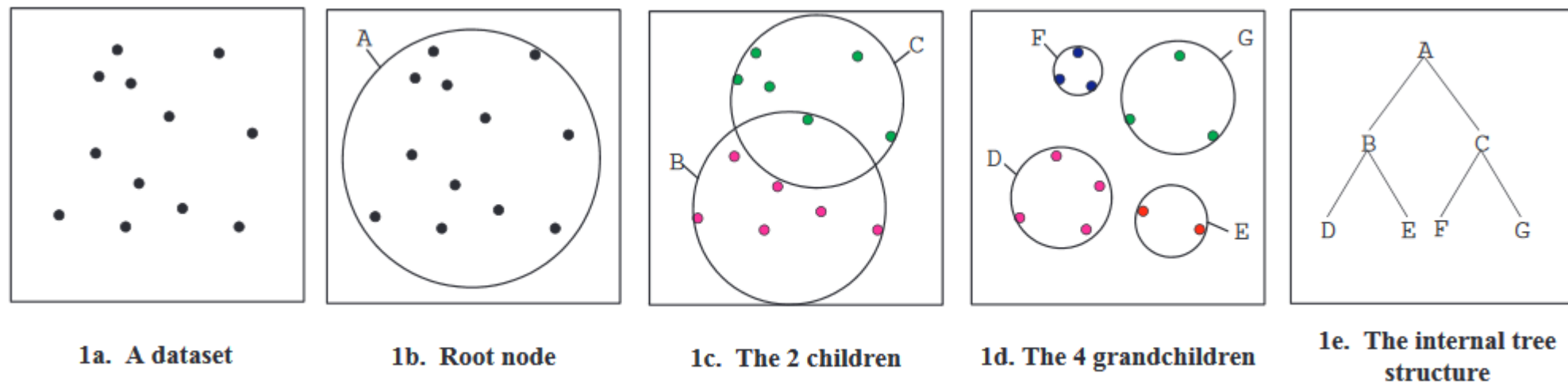


Figure 1: An example of ball-tree structure

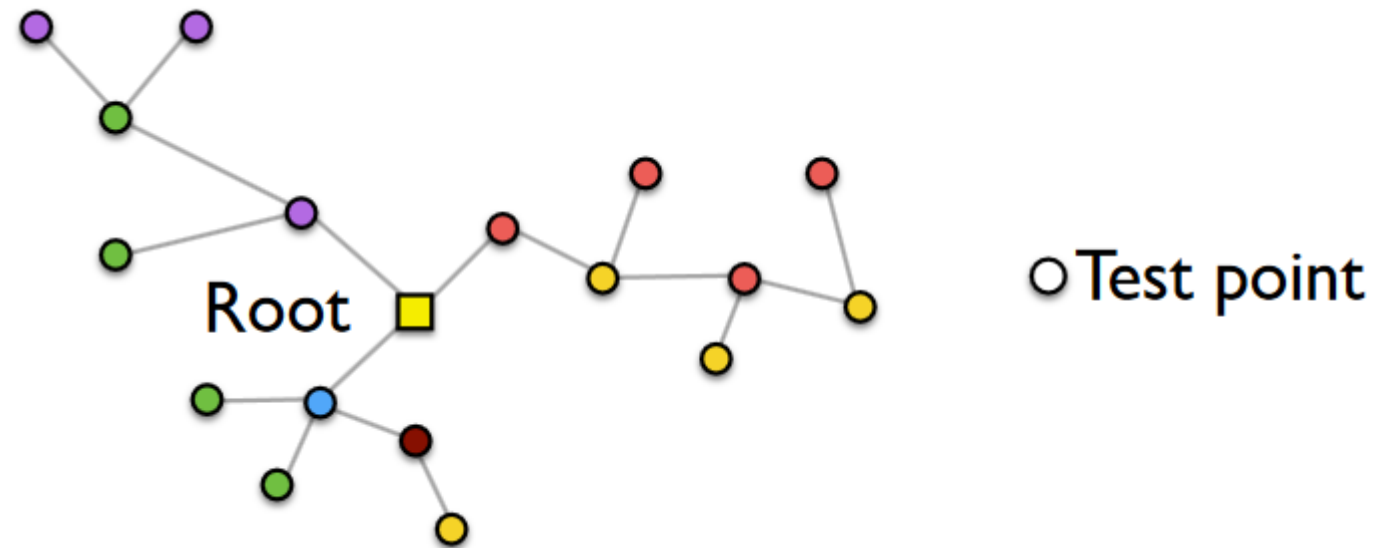
<http://people.ee.duke.edu/~lcarin/liu06a.pdf>

Эффективность: Approximate Nearest Neighbors – Locality-Sensitive Hashing (LSH)

**LSH ~ hash-функция: близкие объекты имеют похожие хэши
хэши короткие \Rightarrow легко сравнивать**

смотрим на все объекты с таким же hash-значением
если их нет, то на все (или используем другую hash-функцию)

Эффективность: Boundary Trees



Пусть есть построенное дерево и точка

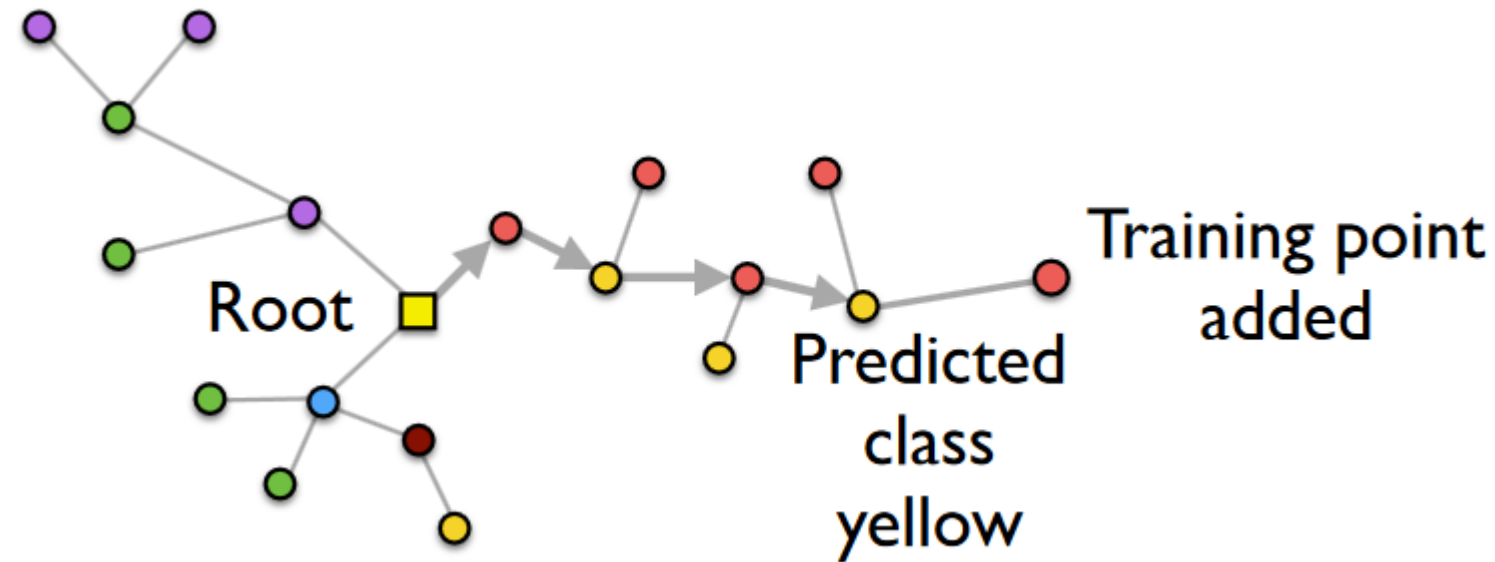
Спускаемся по дереву: смотрим расстояние до текущей вершины и её потомков

Если до какого-то потомка ближе – спускаемся (переходим в него)

Если нет (или в листе) – добавляем вершину / или нашли «соседа»

Mathy, C., N. Derbinsky, J. Bento, J. Rosenthal, and J. Yedidia (2015) «The Boundary Forest Algorithm for online supervised and unsupervised learning» In: AAAI Conference on Artificial Intelligence.

Эффективность: Boundary Trees



Пусть есть построенное дерево и точка

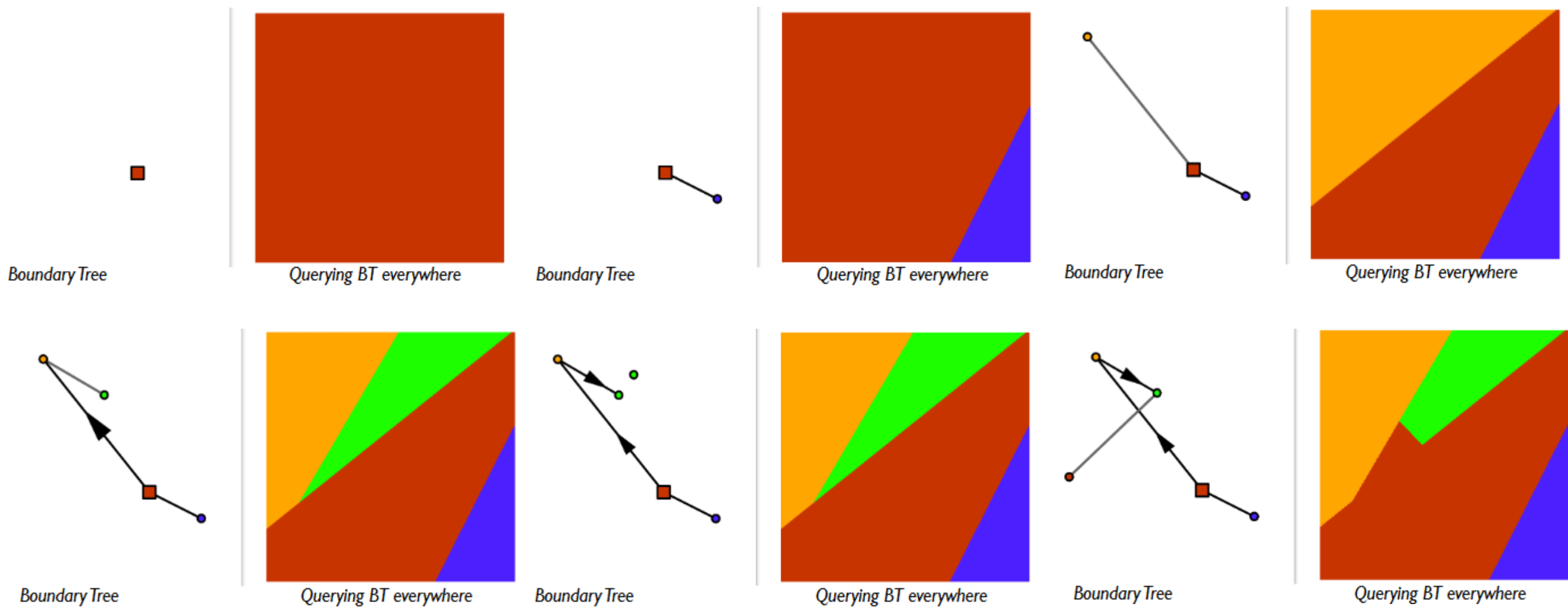
Спускаемся по дереву: смотрим расстояние до текущей вершины и её потомков

Если до какого-то потомка ближе – спускаемся (переходим в него)

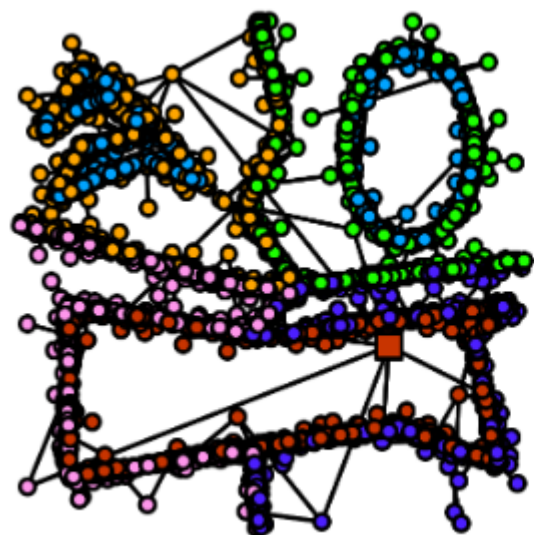
Если нет (или в листе) – добавляем вершину / или нашли «соседа»

При обучении можно добавлять вершину только,
если предсказание по текущему дереву на ней неверное

Эффективность: Boundary Trees



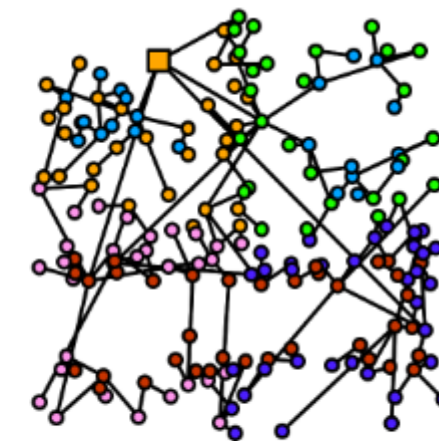
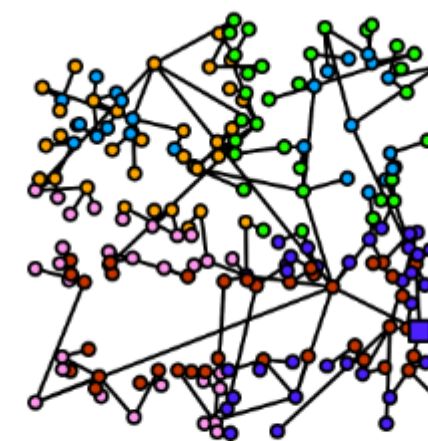
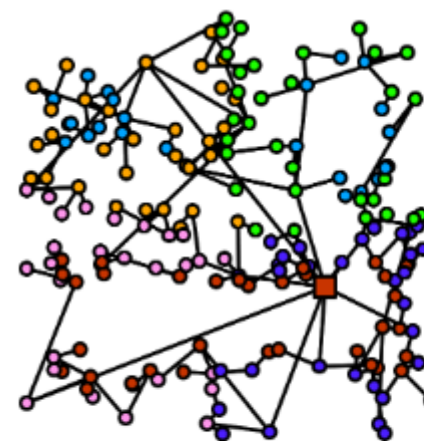
Эффективность: Boundary Trees



Boundary Tree



Querying BT everywhere



Shown 10,000 points

Модельная задача:

$m = 100\,000$

в дереве $|V| = 2\,220$

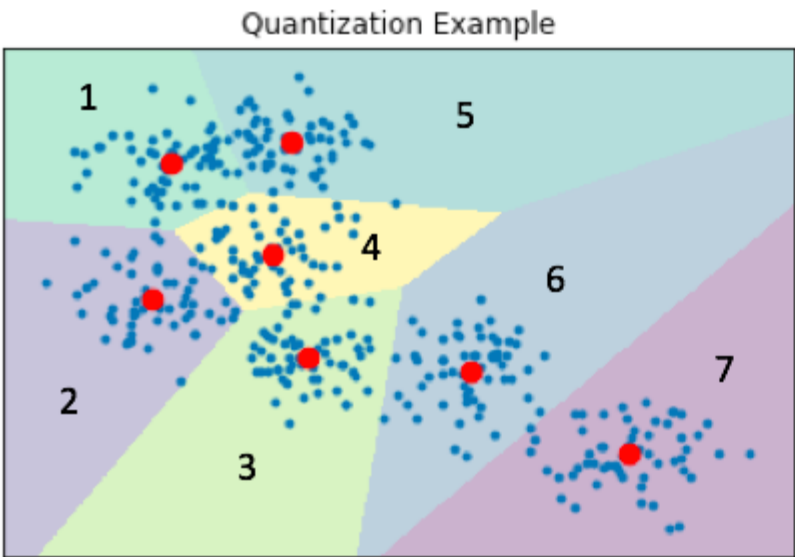
Лес:
Строим с разным обходом выборки

Есть также обобщения на регрессию.

<https://nn2017.mit.edu/wp-content/uploads/sites/5/2017/12/BoundaryForestNIPS2017.pdf>

Эффективные методы поиска ближайших соседей

Quantization



inverted index

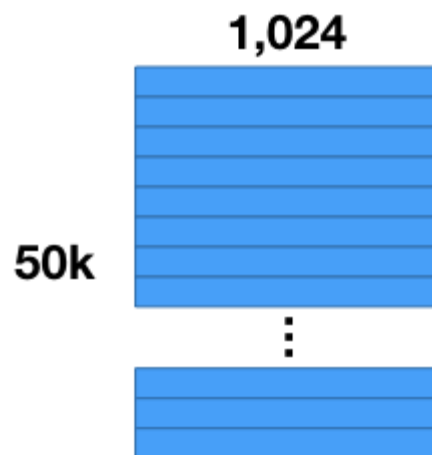
centroid	(ids)				
1	1	2	9	12	...
2	6	13	16	31	...
3	3	2	22	29	...
4	...				
5					
6					
7	4	5	7	14	...

Assuming each vector has an identifier, we can maintain a list of vectors for each Voronoi cell. This type of data structure is known as an **inverted index**.

**предварительная кластеризация
инвертированный индекс!**

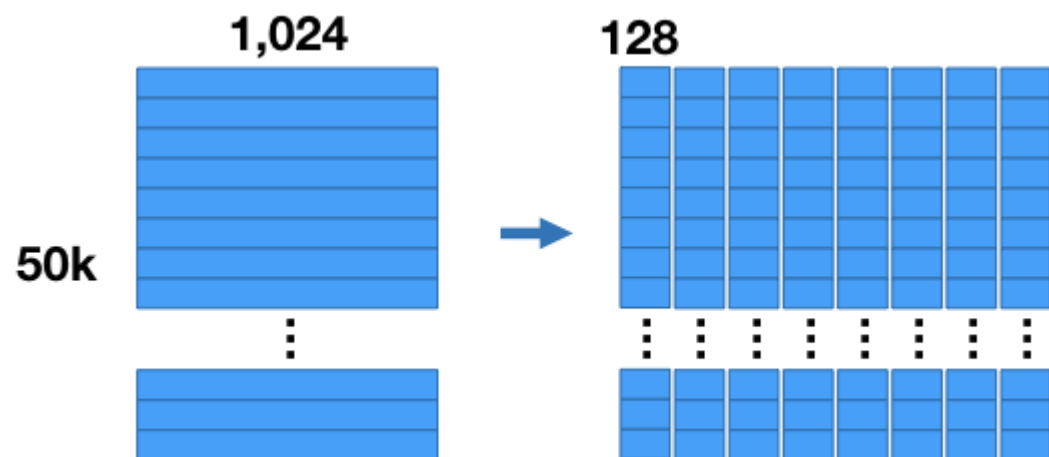
Эффективность: квантование «Product quantization»

по-прежнему будем сравнивать с каждым из m объектов,
но упростим сравнение



Пример: изображение \rightarrow вектор (пусть уже сделано с помощью НС)

Сейчас перейдём ещё к меньшей размерности
но это не снижение размерности (новое пр-во дискретное)

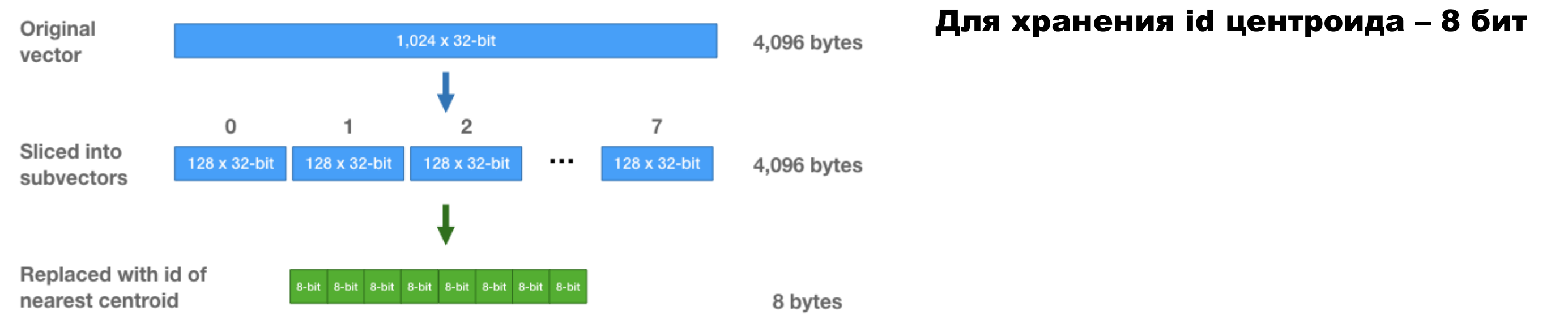


Разделим пространство на подпространства (декартовым образом) 128×8

В каждом найдём $k=256$ центроидов

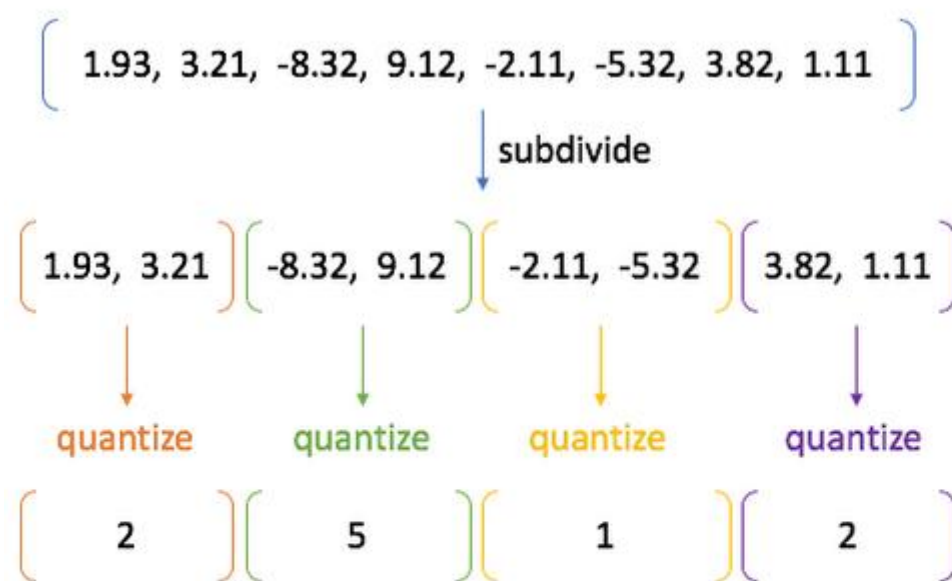
Будем хранить номера центроидов

Эффективность: квантование «Product quantization»

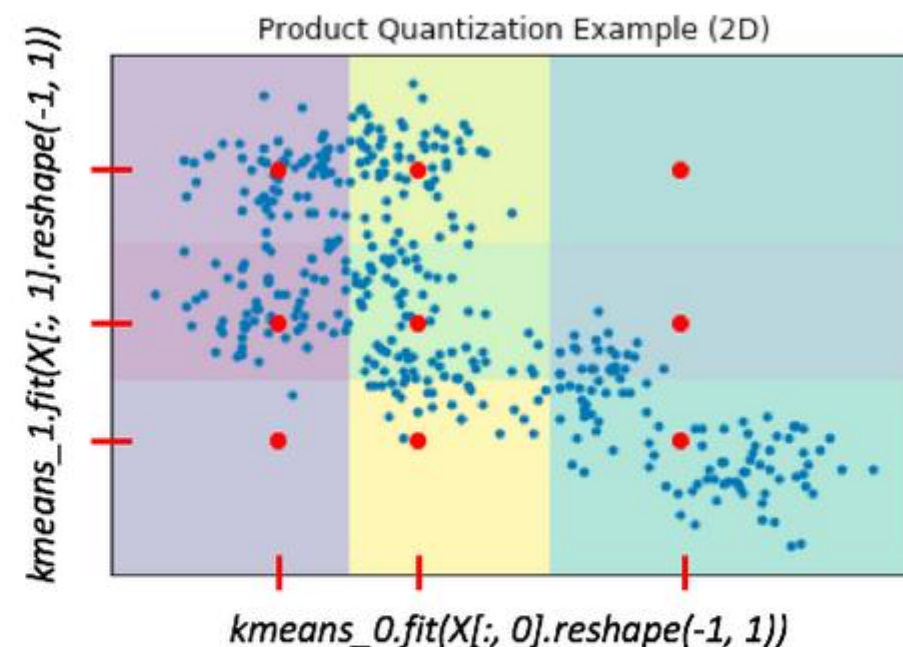


<https://mccormickml.com/2017/10/13/product-quantizer-tutorial-part-1/>

Эффективность: Product quantization



8D example



$$X = (x_0, x_1)$$

2D example (easier to visualize)

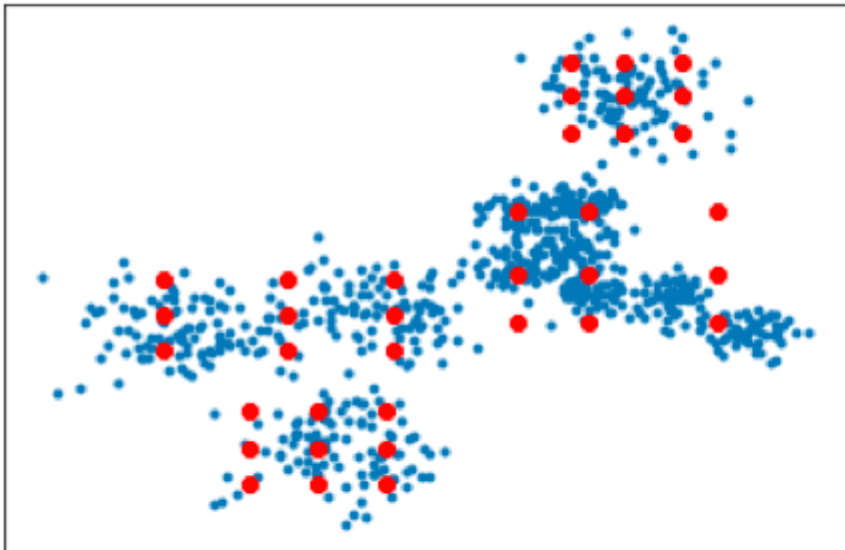
квантование по отдельным координатам (или подпространствам)

см. также

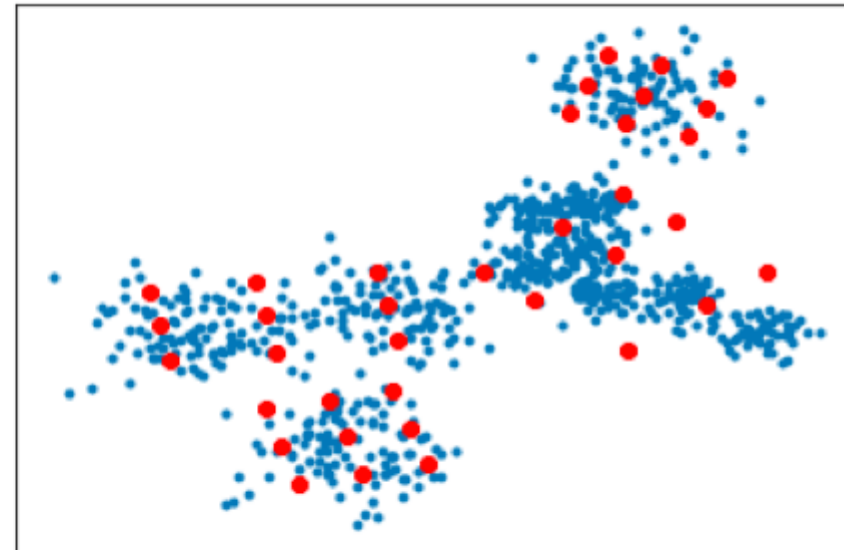
<https://cache-ash04.cdn.yandex.net/download.yandex.ru/company/cvpr2012.pdf>

Эффективность: Locally optimized product quantization

Product Quantization With Coarse Quantization



Locally Optimized Product Quantization



Locally optimize → PCA align the product centroids in each coarse cell

Эффективные методы поиска ближайших соседей

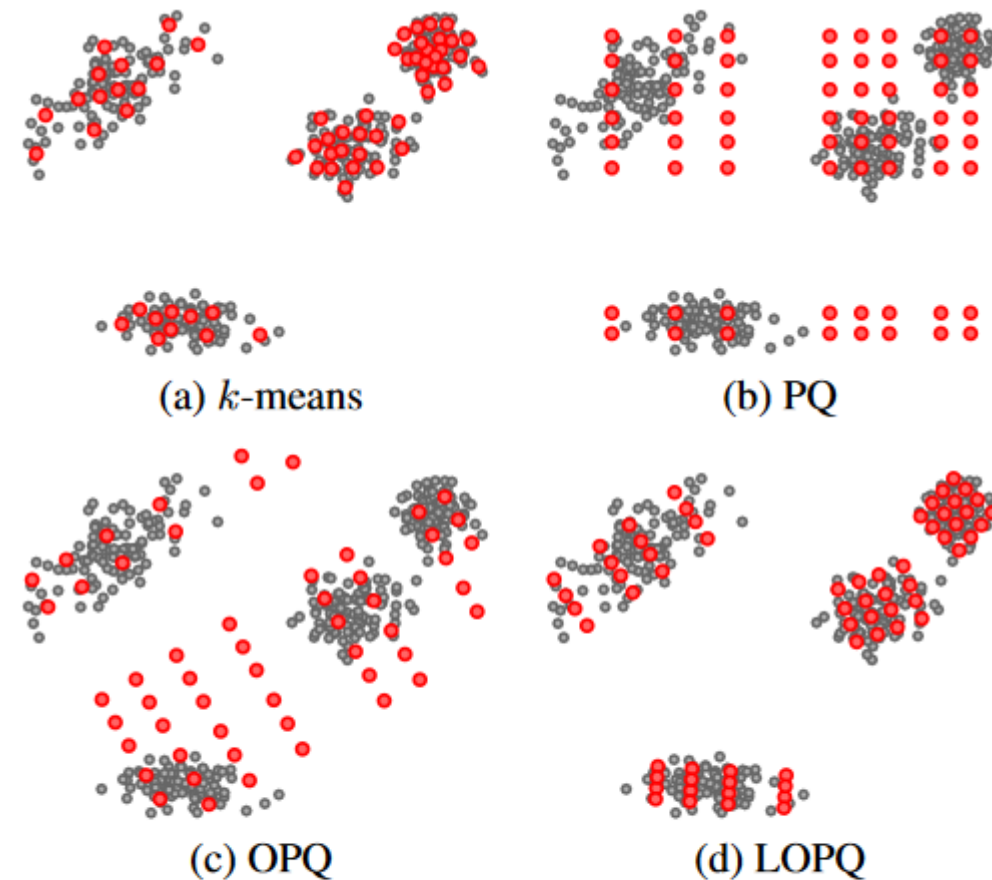
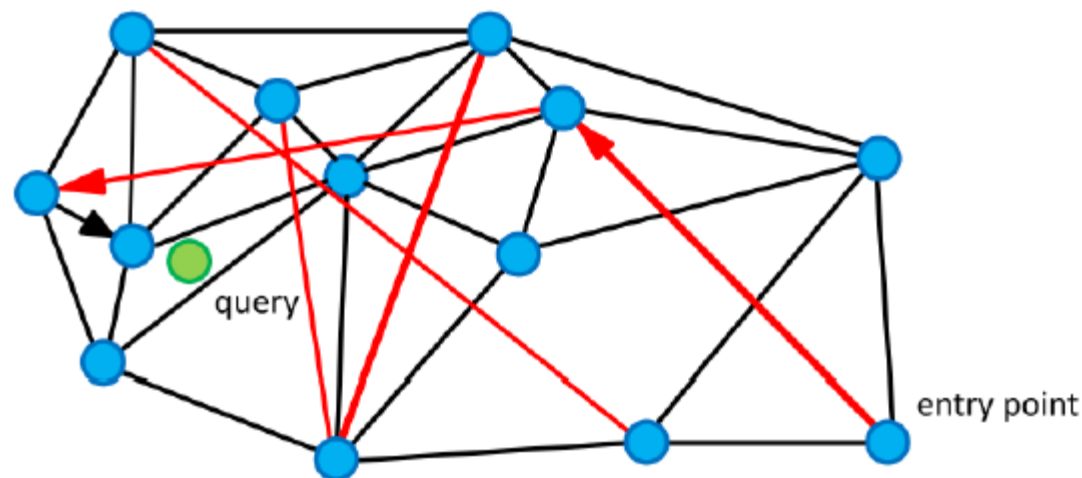


Figure 1. Four quantizers of 64 centroids (•) each, trained on a random set of 2D points (•), following a mixture distribution. (c) and (d) also reorder dimensions, which is not shown in 2D.

<http://image.ntua.gr/iva/files/lopq.pdf>

Эффективность: Navigable Small World (NSW)

главное – найти быстро, но не факт, что найдём самого ближайшего

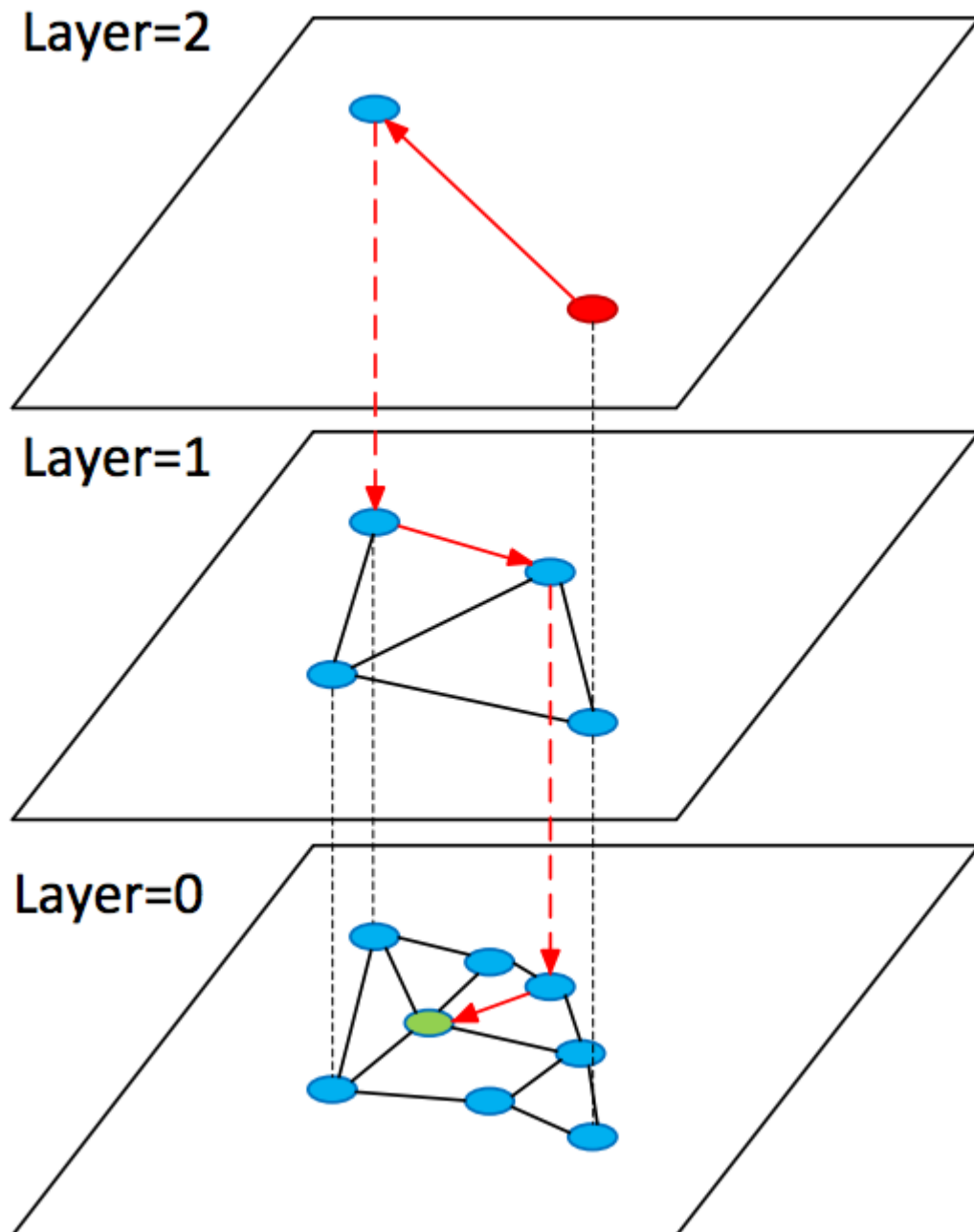


**Одна из идей «small world graph»:
объект связан с k соседями**

**(в оригинале – графа Делоне, т.к. этого достаточно для поиска БС),
плюс t длинных связей**

**При поиске БС жадно идём по графу,
перебираем соседей текущей вершины,
считаем до них расстояния**

Эффективность: Hierarchical Navigable Small World (HNSW)



**Иерархические вложения:
слои – прореживание выборки**

выполняем поиск послойно

**+ простой и эффективный
+ эффективен по памяти**

<https://habr.com/ru/company/mailru/blog/338360/>

Библиотеки

FALCONN - FAst Lookups of Cosine and Other Nearest Neighbors

<https://github.com/FALCONN-LIB/FALCONN>

Approximate Nearest Neighbor Search for Sparse Data in Python

<https://github.com/facebookresearch/pysparnn>

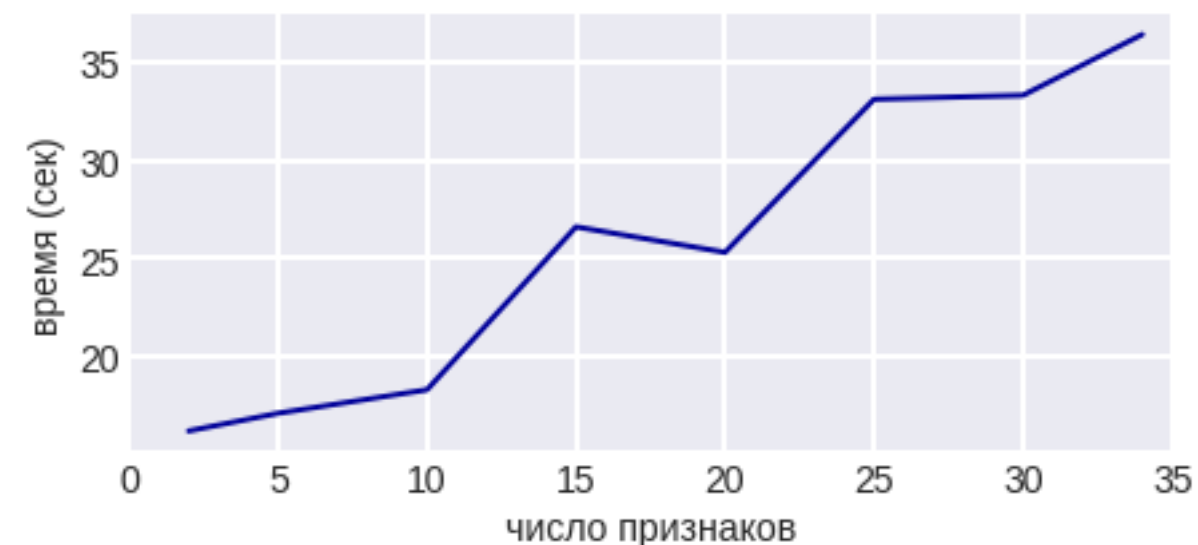
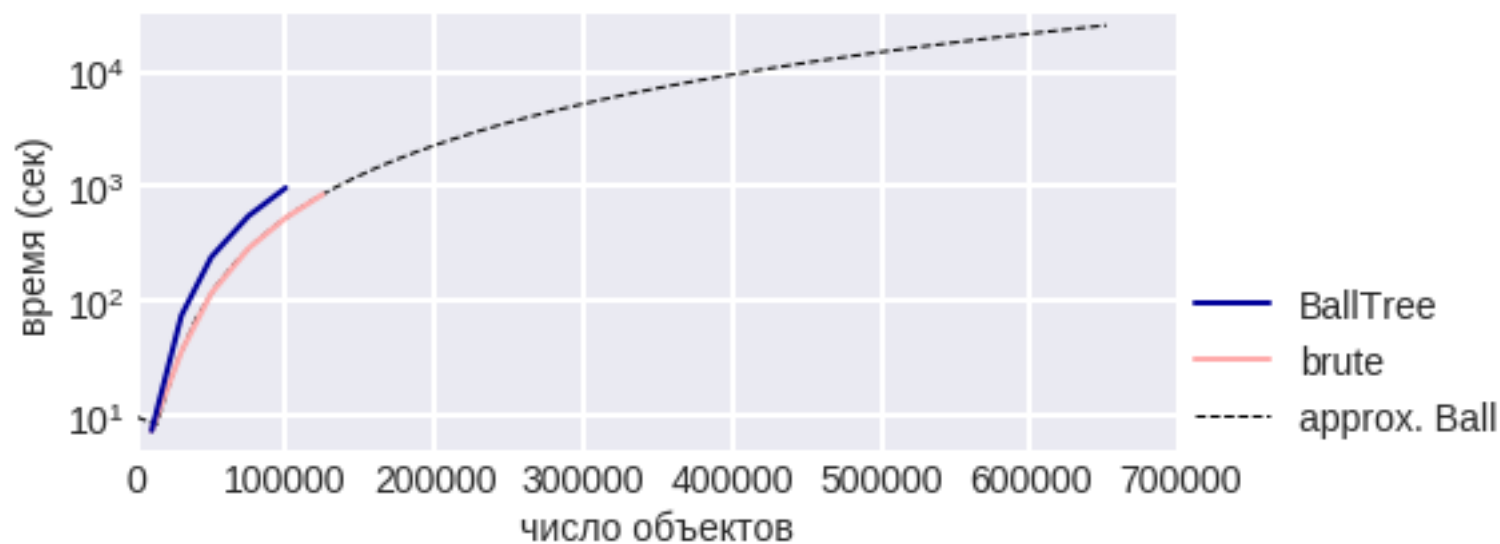
Faiss: A library for efficient similarity search

<https://engineering.fb.com/data-infrastructure/faiss-a-library-for-efficient-similarity-search/>

Hubless Nearest Neighbor Search for Bilingual Lexicon Induction. ACL 2019

<https://github.com/baidu-research/HNN#HNN>

Эксперименты с метрикой Хэмминга



В R^n ищем 2 ближайших соседей.

С метрикой Хэмминга не поддерживается KDtree.

Слева – экстраполировали зависимость (шкала логарифмическая).

■ ещё по размерности пространства – вроде при $n > 10$ можно обычный перебор использовать

Реализация `sklearn.neighbors.NearestNeighbors`

`n_neighbors` – **число соседей (5)**

`radius` – **ограничение пространства (1.0)**

`algorithm` – **алгоритм для определения БС (auto, ball_tree, kd_tree, brute)**

`leaf_size` – **параметр для BallTree / KDTree**

`metric` – **метрика (функция или строка:), см. `scipy.spatial.distance`**

`scikit-learn`: [cityblock, cosine, euclidean, l1, l2, manhattan]

`scipy.spatial.distance`: [braycurtis, canberra, chebyshev, correlation, dice, hamming, jaccard, kulsinski, mahalanobis, minkowski, rogerstanimoto, russellrao, seuclidean, sokalmichener, sokalsneath, sqeuclidean, yule]

`p` – **параметр для minkowski (2)**

`metric_params` – **дополнительные параметры для метрики**

`n_jobs` – ...

Реализация KNeighborsClassifier/ KNeighborsRegressor

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
```

`n_neighbors` – число соседей

`weights` – веса («uniform», «distance», функция)

`algorithm` – алгоритм для эффективного нахождения соседей
(«auto», «ball_tree», «kd_tree», «brute»)

`leaf_size` – для BallTree / KDTree

`p` – параметр для метрики Минковского

`metric` – метрика («minkowski»)

`metric_params` – параметры для метрики

`n_jobs` – число процессов для нахождения соседей

```
from sklearn.neighbors import KNeighborsRegressor
```

`weights` – весовая схема для объектов (uniform, distance, функция)

Реализация

```
from sklearn.neighbors.nearest_centroid import NearestCentroid
```

– ближайший центроид

```
sklearn.neighbors.kneighbors_graph
```

– граф соседей

```
sklearn.neighbors.RadiusNeighborsClassifier
```

```
sklearn.neighbors.RadiusNeighborsRegressor
```

– алгоритмы с соседством по радиусу

```
sklearn.neighbors.NeighborhoodComponentsAnalysis
```

– обучение метрики

```
sklearn.neighbors.KernelDensity
```

– KDE-оценка плотности

Итог

Простые ленивые методы

можно использовать и без признаков описаний
но задача свелась к выбору метрики
метод недооценёны!

Ближайший центроид

примитивный, но иногда эффективный
и быстрый метод

kNN

- не эффективен на больших данных (время, память)
- не более чем в 2 раза хуже оптимального алгоритма
 - нет процедуры обучения
 - выбросы / дисбаланс классов

Итог

весовые схемы
очень мощное оружие

метрики
много... есть специализированные
могут быть параметризованы

Эффективные способы обнаружения соседства
есть много современных способов

Код

https://github.com/Dyakonov/ml_hacks/blob/master/dj_IML_kNN.ipynb

Теория kNN

https://github.com/mlss-2019/slides/tree/master/learning_theory