«Машинное обучение»

Метрические методы

Александр Дьяконов

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

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

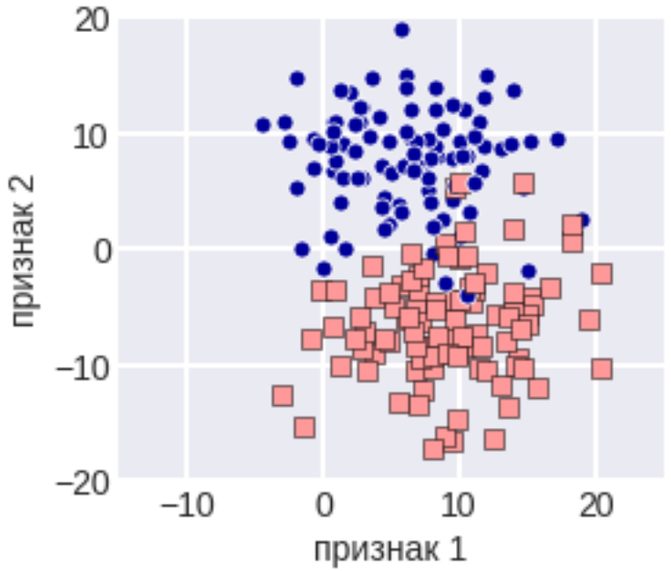
Примеры:

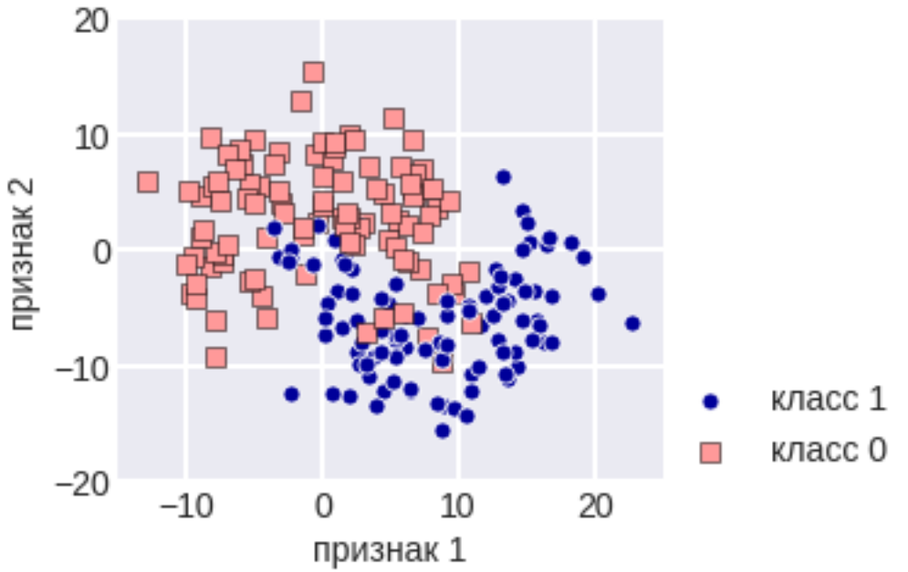
- Nearest centroids algorithm / Distance from Means
- kNN (Nearest Neighbor)

ещё называют:

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

Модельные задача классификации





на них будем показывать работу алгоритмов

Ближайший центроид (Nearest centroid algorithm)

Задача классификации на непересекающиеся классы с вещественными признаками:

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

центроиды:

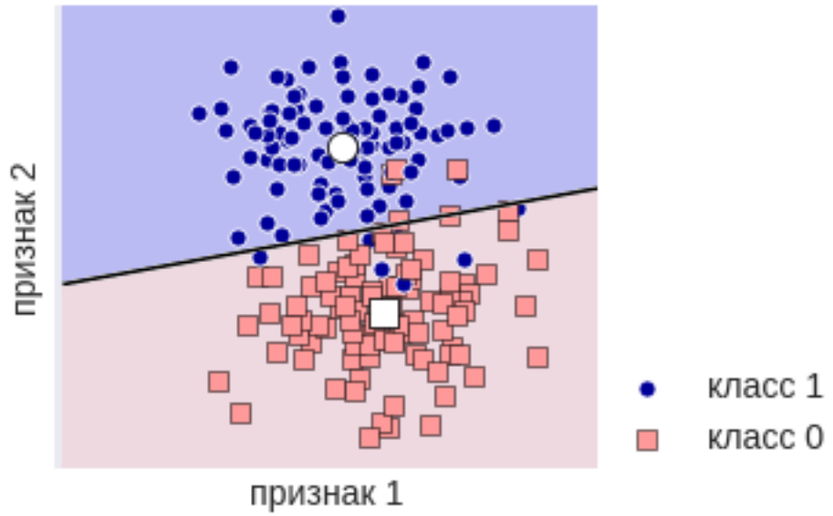
$$c_j = \frac{1}{|\{i : y_i = j\}|} \sum_{i: y_i = j} x_i$$

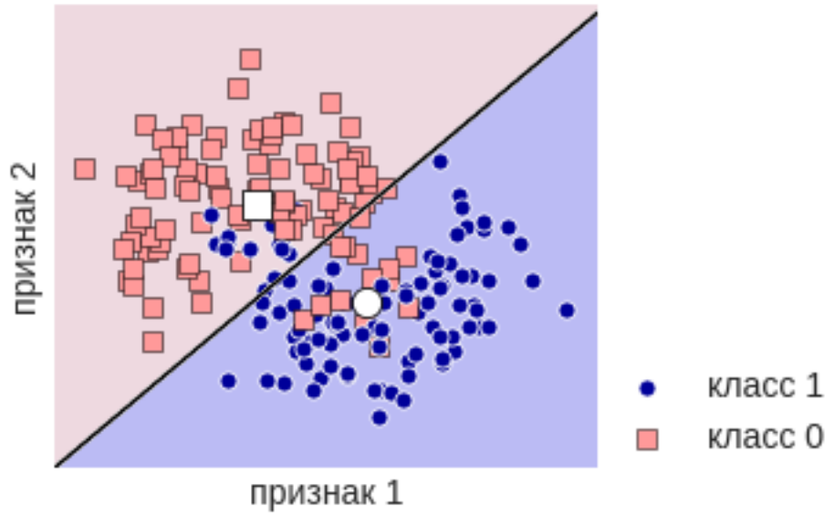
классификация:

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

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 × 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)
```

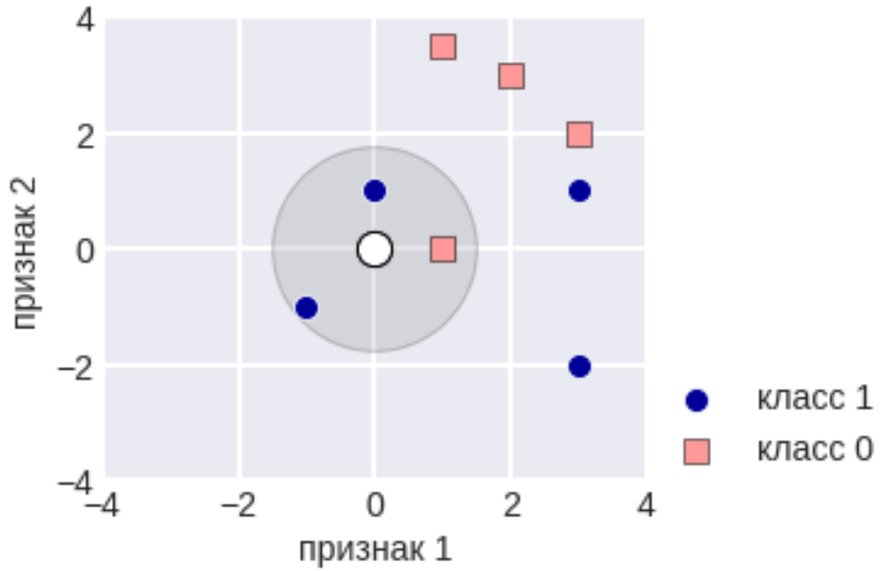
есть параметр

```
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)

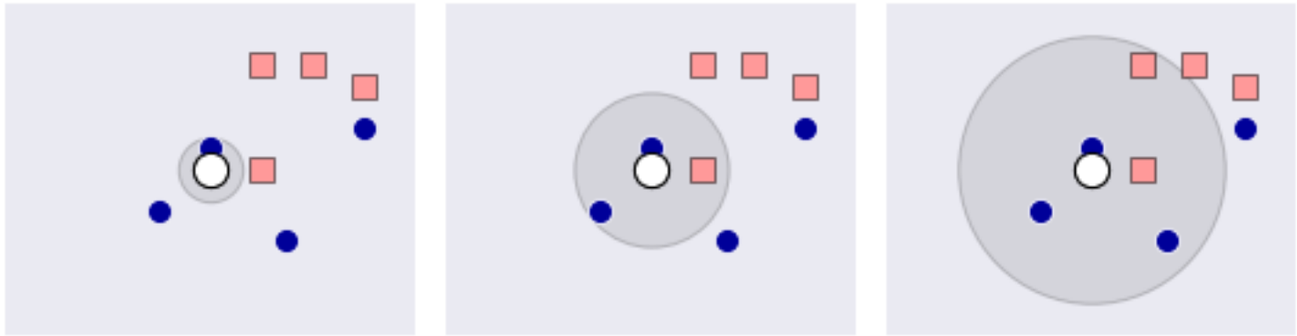
Окрестность

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

$$\rho(x, x_1) \le \ldots \le \rho(x, x_m)$$

В методе k ближайших соседей (kNN = k nearest neighbours)
окрестность выбирается

$N(x) = \{x_1, \ldots, x_k\}$ **– k ближайших соседей:**



Окрестность

Есть также «Fixed-Radius Near Neighbor»

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

про него не будем подробно

Метод k ближайших соседей (kNN)

Гиперпараметр k можно выбрать на скользящем контроле дальше

Ещё гиперпараметры (потом):

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

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)

Термины

Нетерпеливый алгоритм (Eager learner)	**как только есть обучение – получает значения параметров (учит модель)**
Ленивый алгоритм (Lazy learner)	**не использует обучающую выборку до классификации**

Обоснование 1NN

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

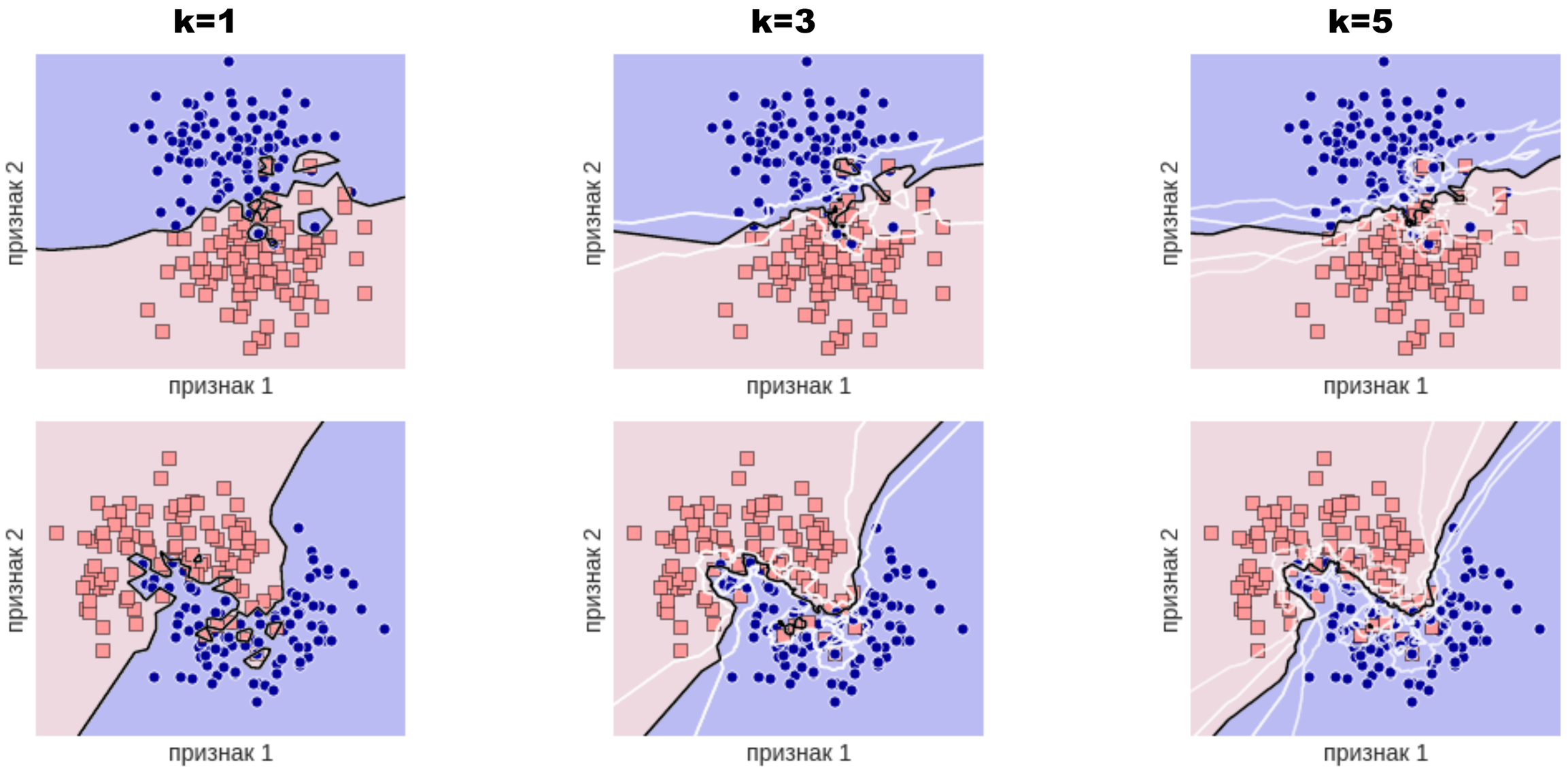
Пусть в некоторой области вероятность встретить объект класса 0 (без огр-я общ-ти) – $p \le 0.5$ – (это же и вероятность ошибки оптимального алгоритма), тогда

	объект	
сосед	**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

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

Решение модельной задачи при разном числе соседей



Решение модельной задачи при разном числе соседей

Как увидим дальше, k отвечает за «сложность модели»

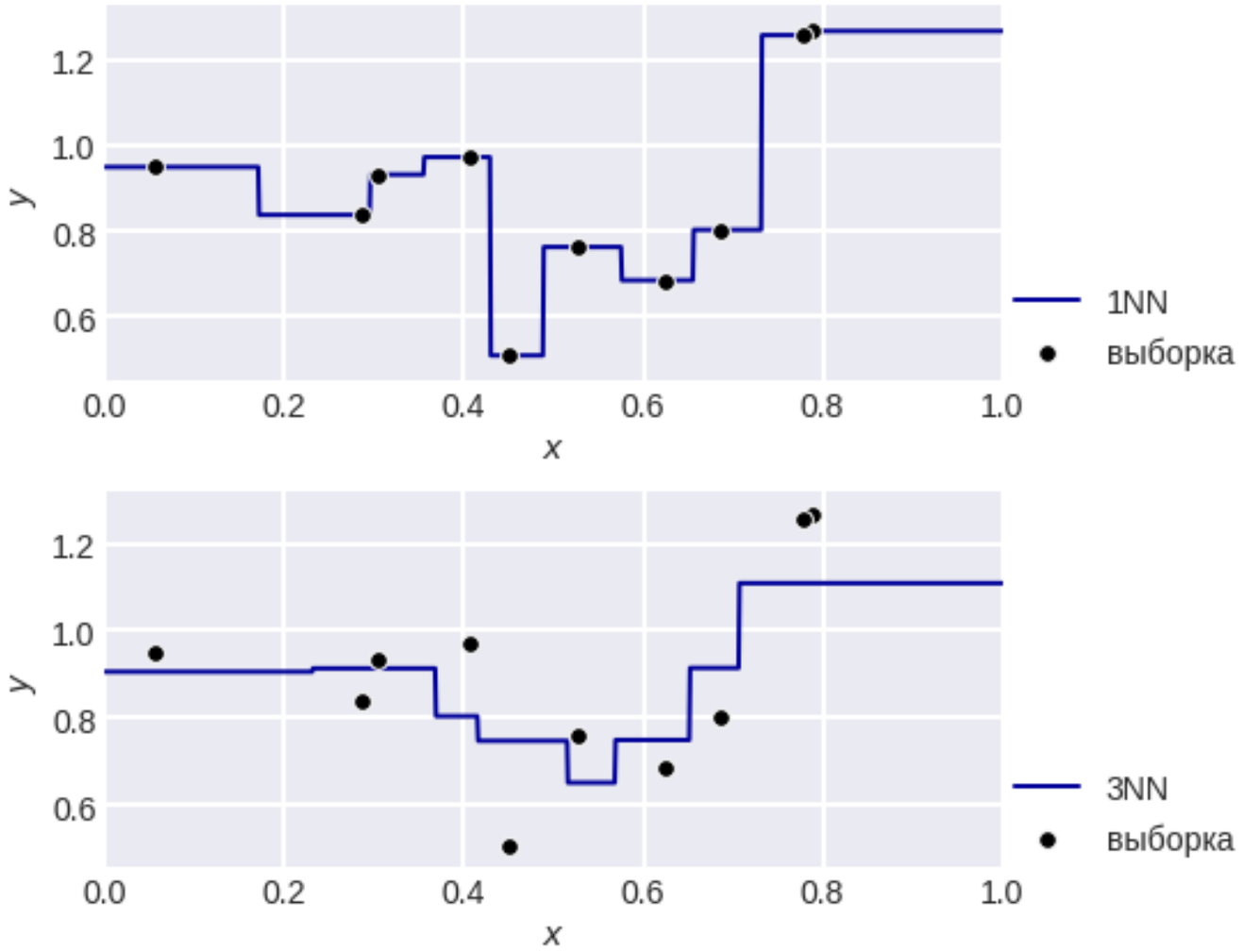
Минутка кода

```
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

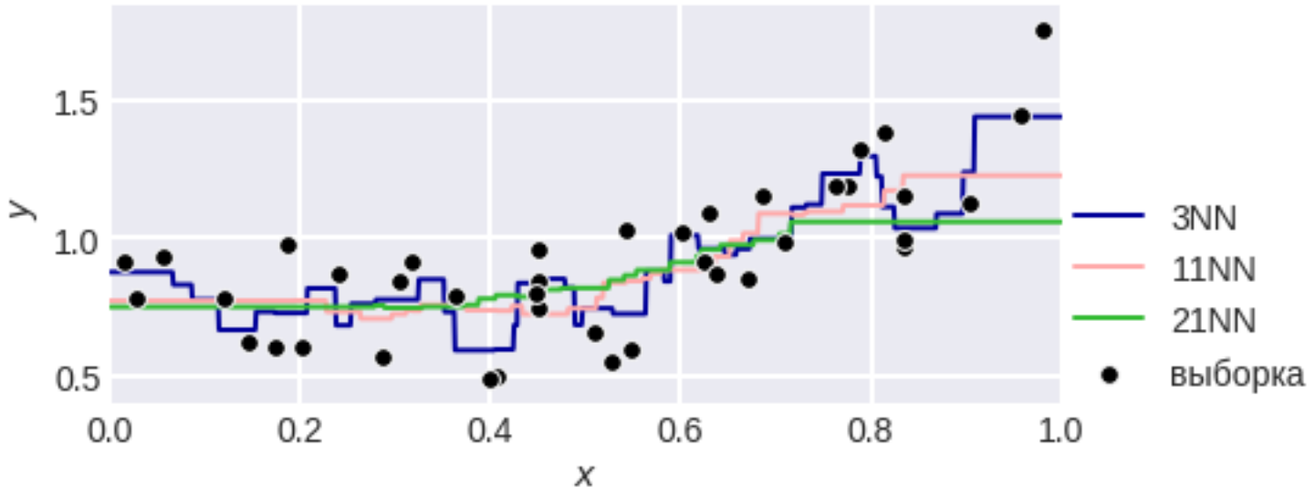
Метод ближайшего соседа

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 $k \to +\infty$ **и** $k / n \to 0$**, то**

$$\mathbf{E}\,|a(x) - f(x)| \xrightarrow[n \to +\infty]{} 0$$

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...

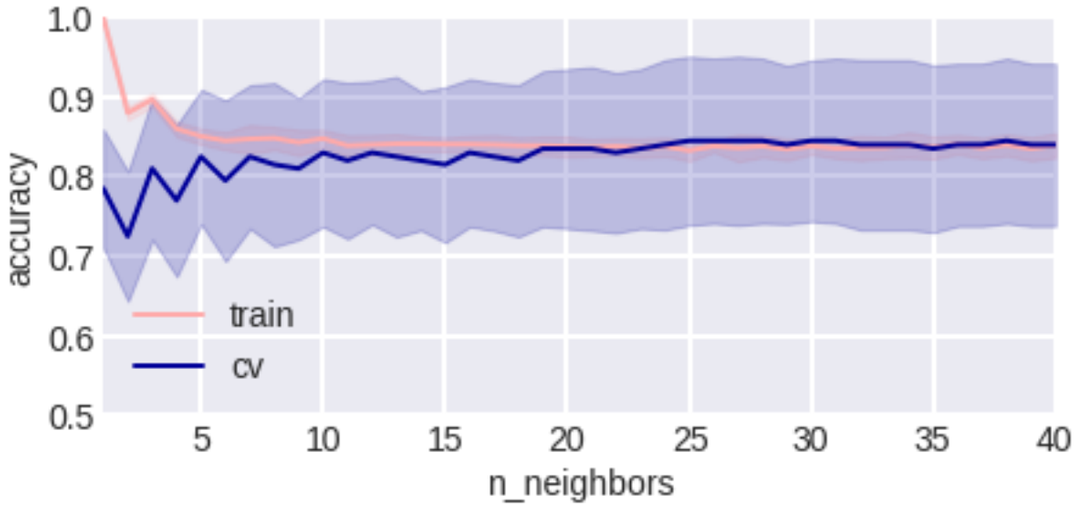
G. H. Chen, D. Shah «Explaining the Success of Nearest Neighbor Methods in Prediction» // Publisher: Now Foundations and Trends https://devavrat.mit.edu/wp-content/uploads/2018/03/nn_survey.pdf

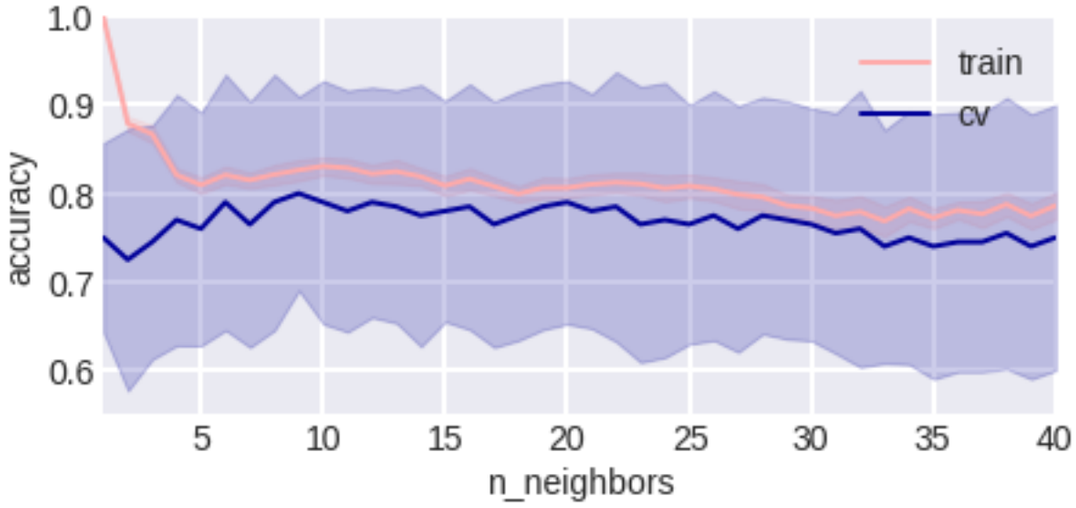
Подбор гиперпараметров специальными методами контроля

```
# 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)
```



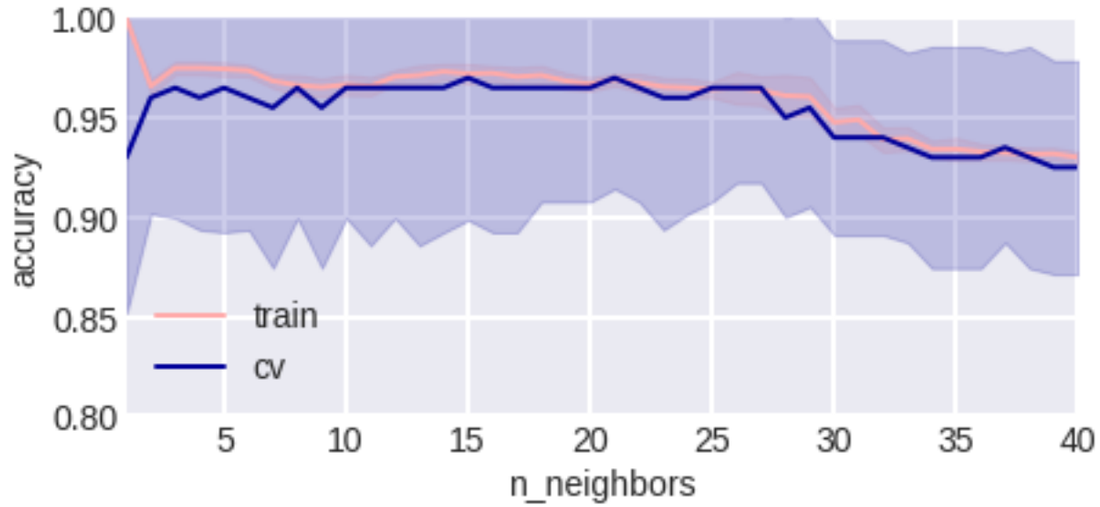


Как был получен второй график?

Подбор гиперпараметров специальными методами контроля

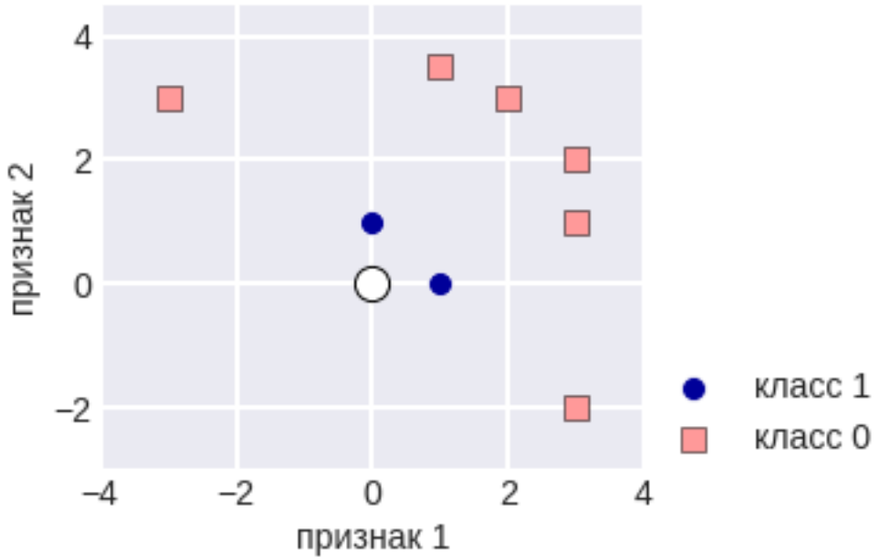
Как был получен второй график?

Сделали дисбаланс классов –
стало невыгодно делать большие k



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

Проблема классического kNN



близкие соседи должны быть важнее

Весовые обобщения kNN

классика:

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

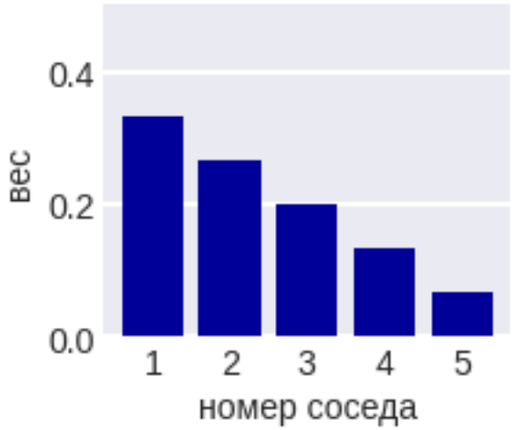
обобщение:

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

разные весовые схемы:

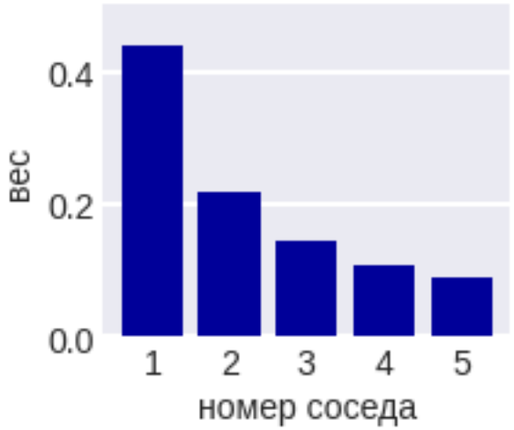
$$w_1 \geq w_2 \geq \ldots \geq w_k > 0$$

Весовые схемы



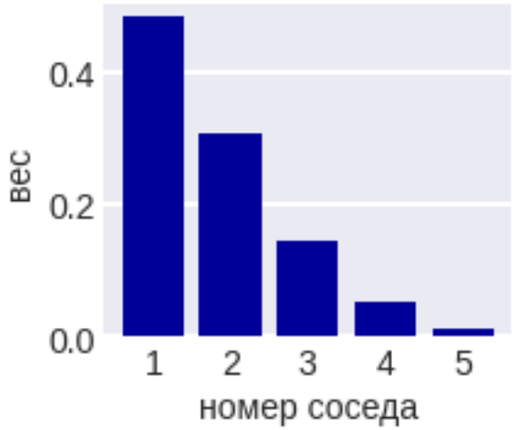
$$w_t = (k - t + 1)^{\delta}$$

$$k^{\delta} \geq (k-1)^{\delta} \geq \ldots \geq 1^{\delta} > 0$$



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

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



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

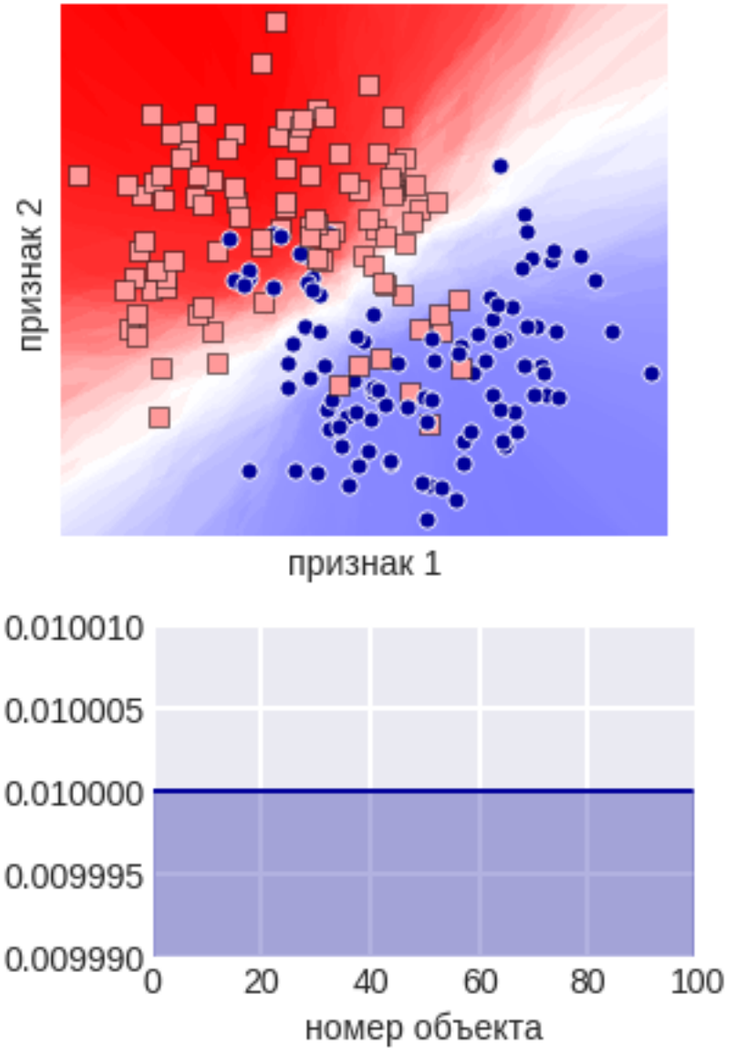
Весовые схемы

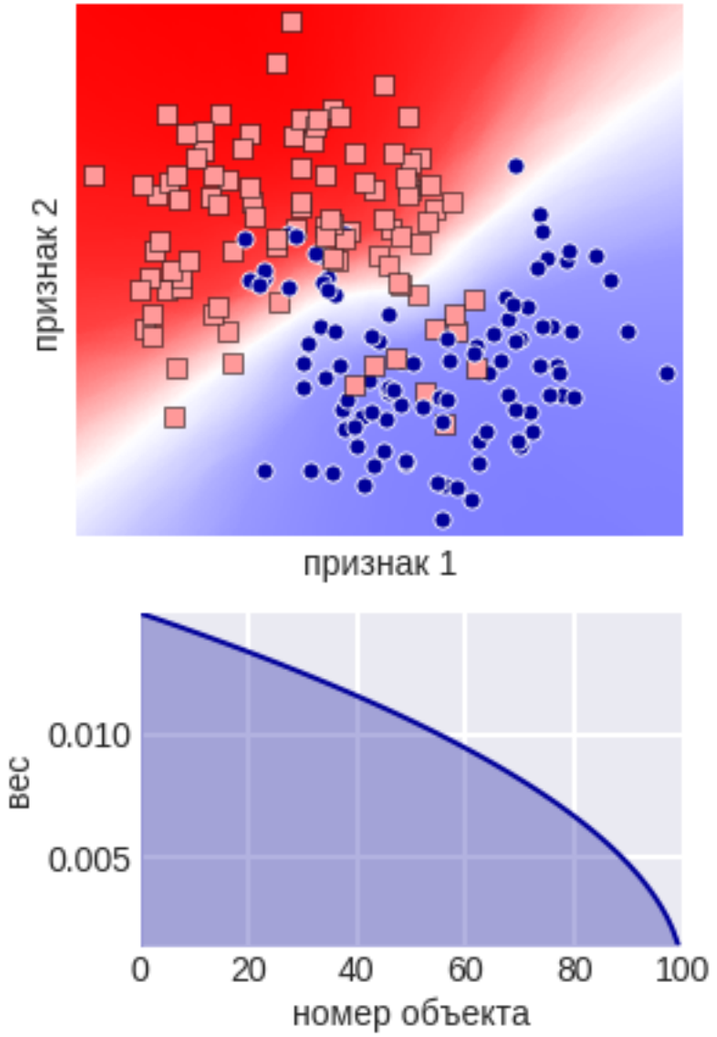
Последний способ хорош только на картинках…

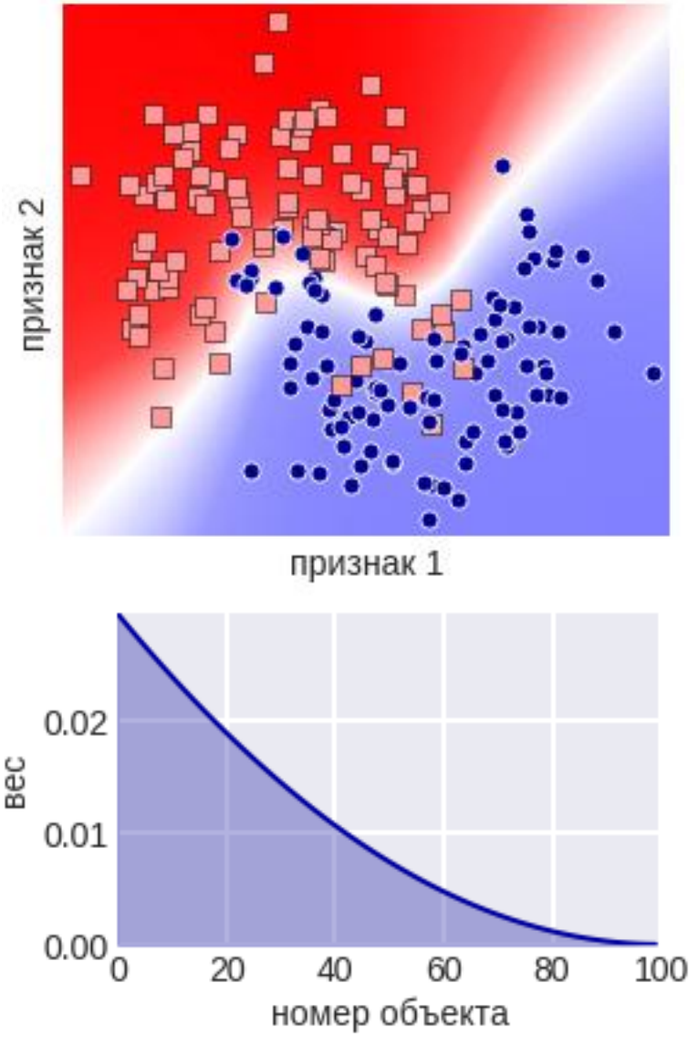
часто веса лучше отнормировать, чтобы сумма = 1

главное преимущество –
богатое пространство вероятности в задачах классификации!
можем различать степени принадлежности у разных объектов
потом будем подробно разбирать

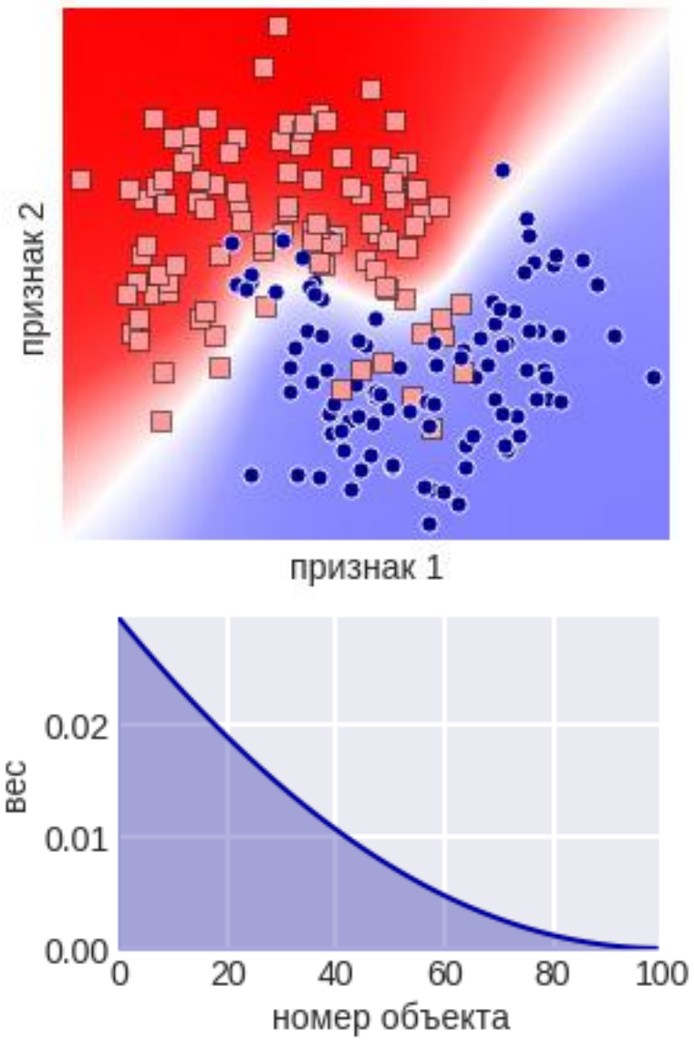
Весовые обобщения kNN

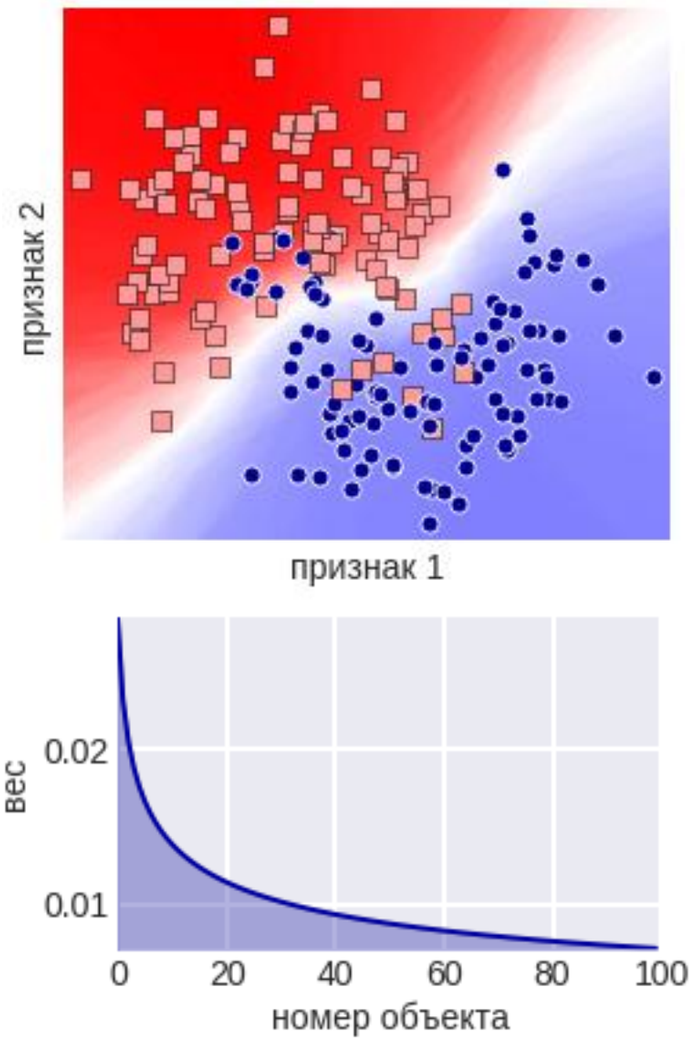


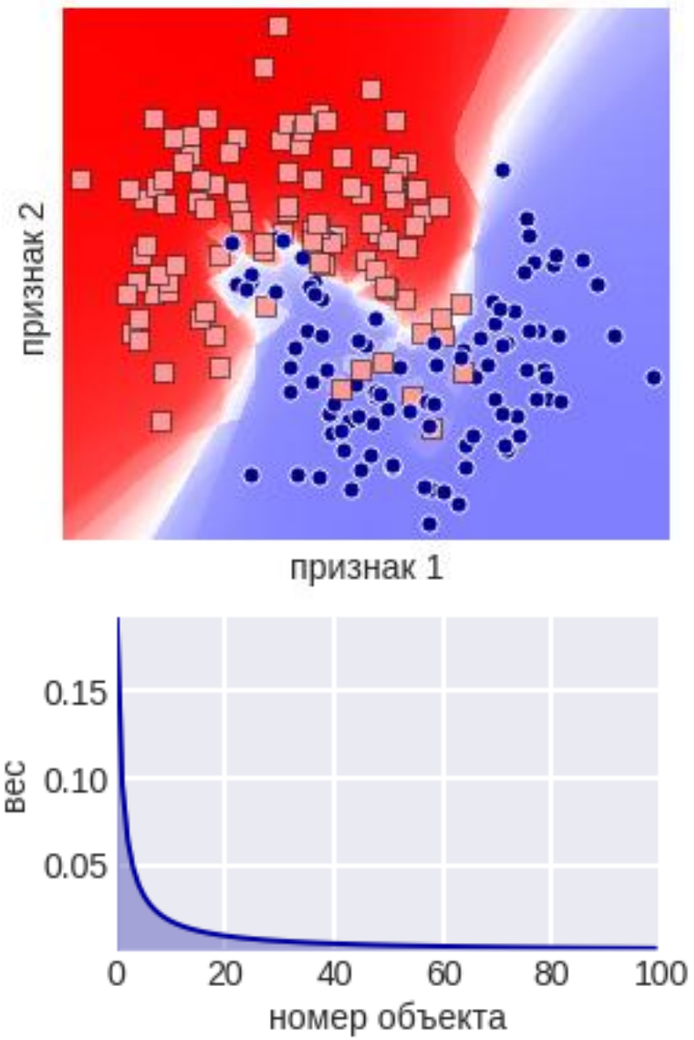




Весовые обобщения kNN



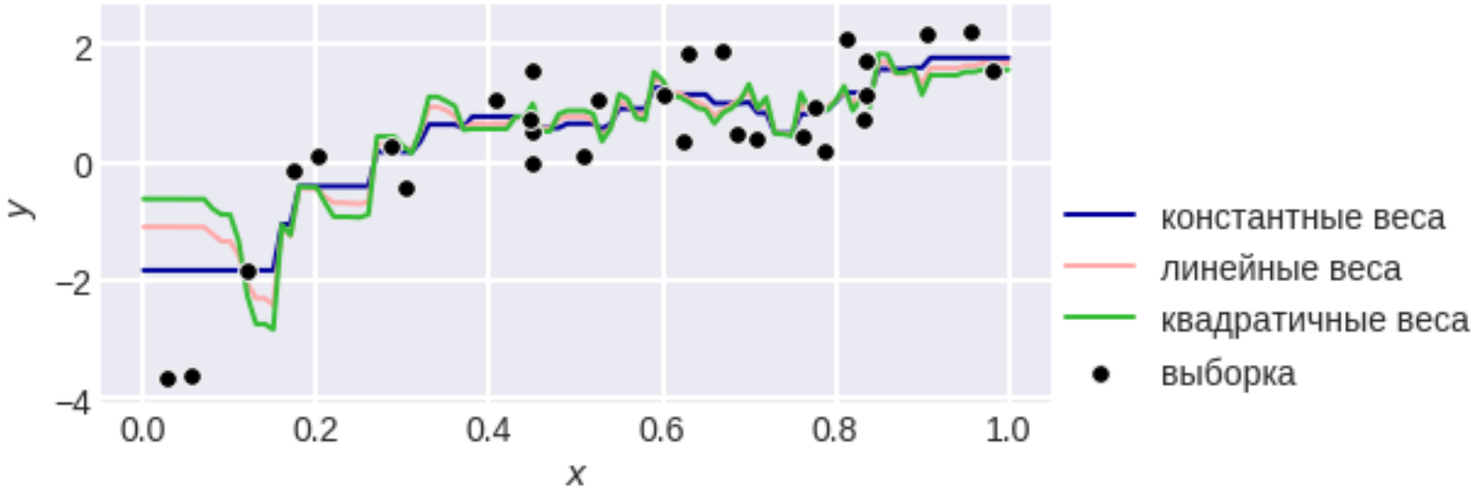




Весовые обобщения в регрессии

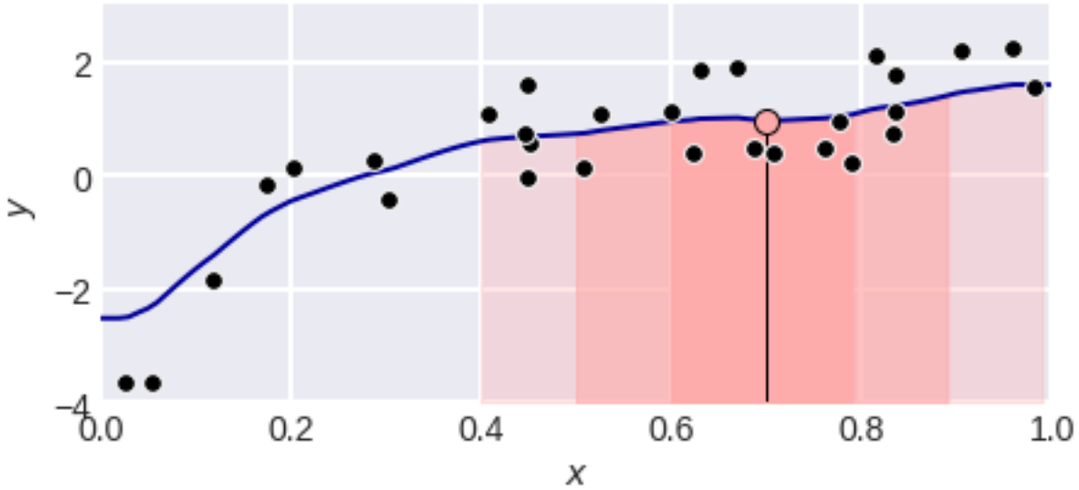
$$\frac{\sum_{t=1}^{k} w_t y(x_t)}{\sum_{t=1}^{k} w_t}$$

пример для 5NN



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

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



ответ – взвешенное усреднение целевых значений

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

Регрессия Надарая-Ватсона (Nadaraya-Watson regression)

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

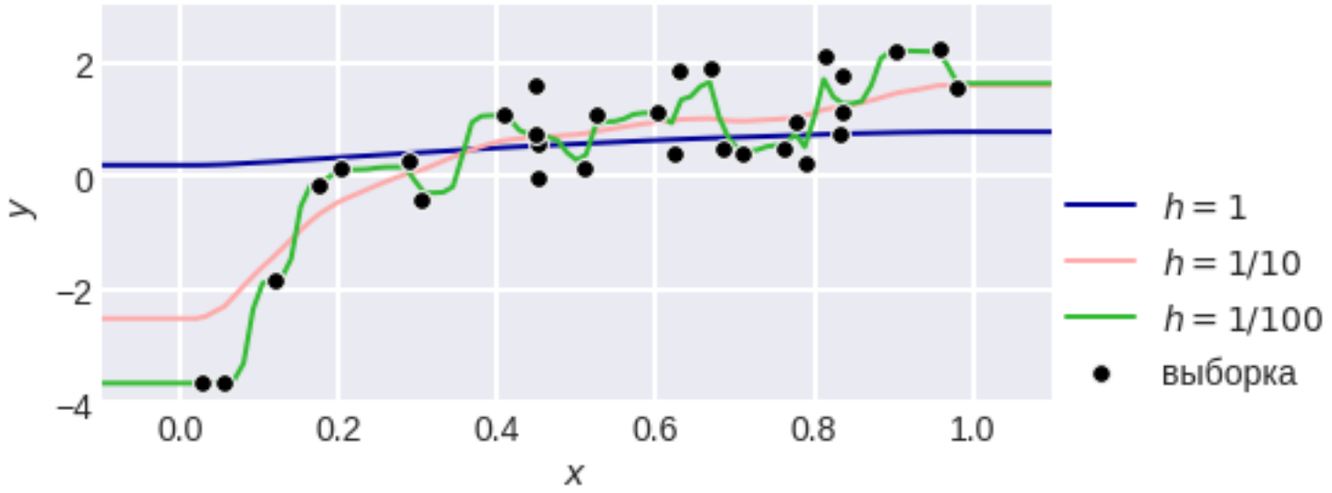
Смысл весов – чем ближе объект обучения, тем скорее ответ похож на его метку

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

Ядро с шириной h.

Здесь также как выше… (про функции ядра)

Регрессия Надарая-Ватсона (Nadaraya-Watson regression)



пример регрессии при разных значениях ширины ядра

Регрессия Надарая-Ватсона (Nadaraya-Watson regression)

Смысл:

ответ алгоритма – решение оптимизационной задачи

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

Свойства:

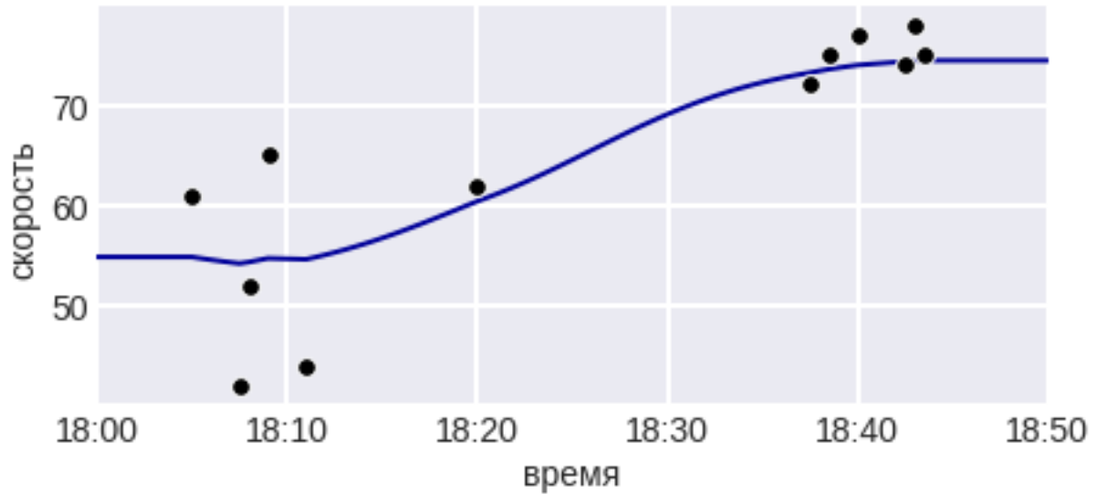
+ хорошее решение задачи сглаживания

– не решает задачи экстраполяции

Приложения регрессии Надарая-Ватсона

1. Сглаживание сигналов

2. «Многомерные» усреднения



Метрики

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

1. $\rho(x,z) \geq 0$
2. $\rho(x,z) = 0 \Leftrightarrow x = z$ **(без – полумерика/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**
- **Махаланобиса**

- **Canberra distance**
- **Хэмминга**
- **косинусное**
- **расстояние Жаккара**

- **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 – Чебышёва (L_∞)

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

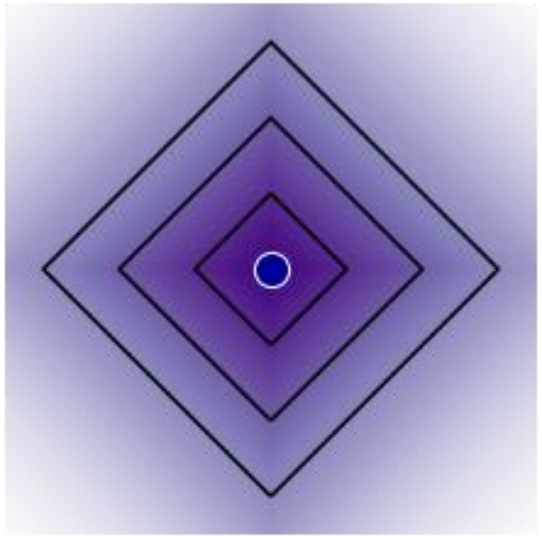
Частный случай – Манхэттенская (L_1)

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

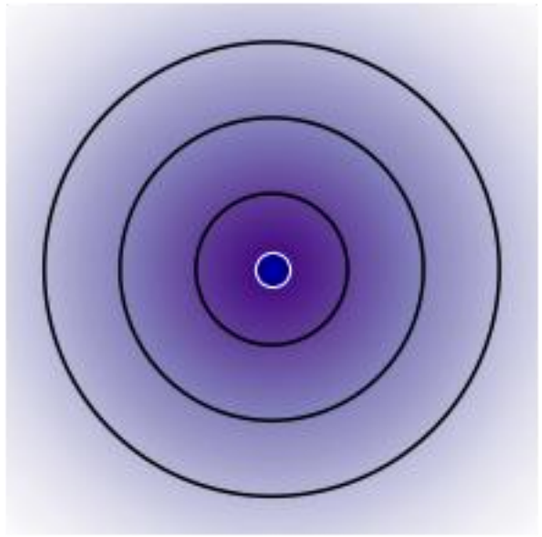
здесь $x = (x_1, \ldots, x_n), \quad z = (z_1, \ldots, z_n)$

Различные метрики

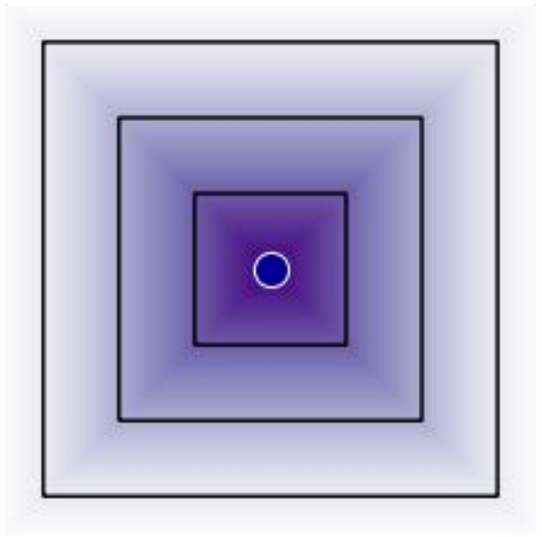
$$\left(| x_1 - z_1 |^p + | x_2 - z_2 |^p \right)^{1/p}$$



L_1



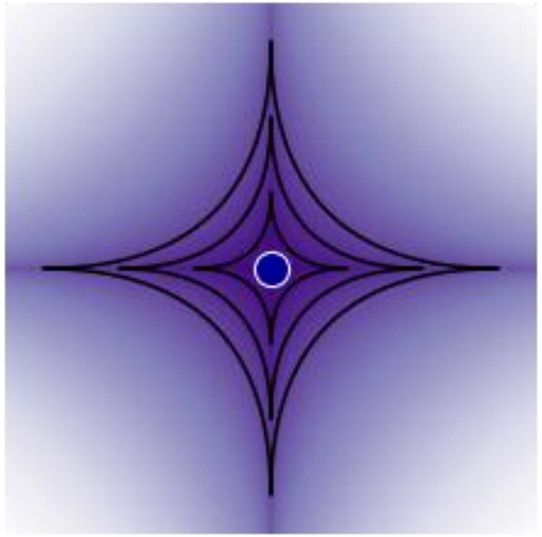
L_2



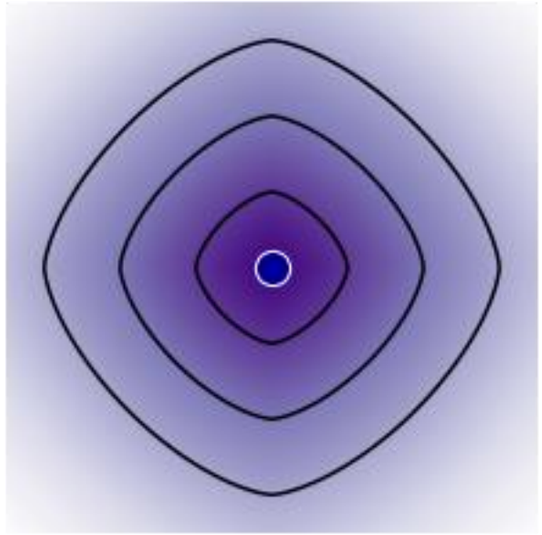
L_∞

Различные метрики

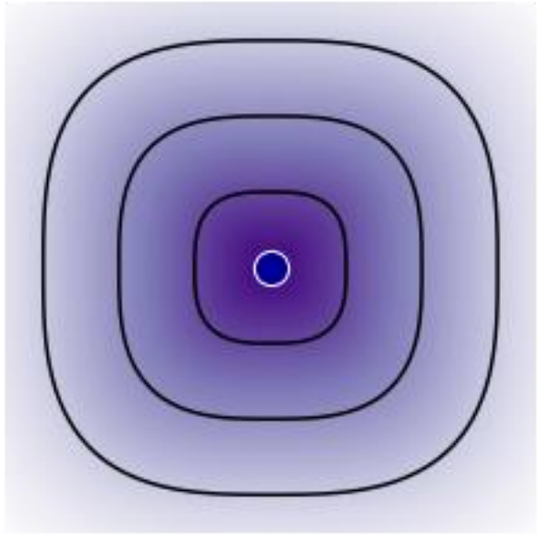
$$\left(| x_1 - z_1 |^p + | x_2 - z_2 |^p \right)^{1/p}$$



$L_{0.5}$



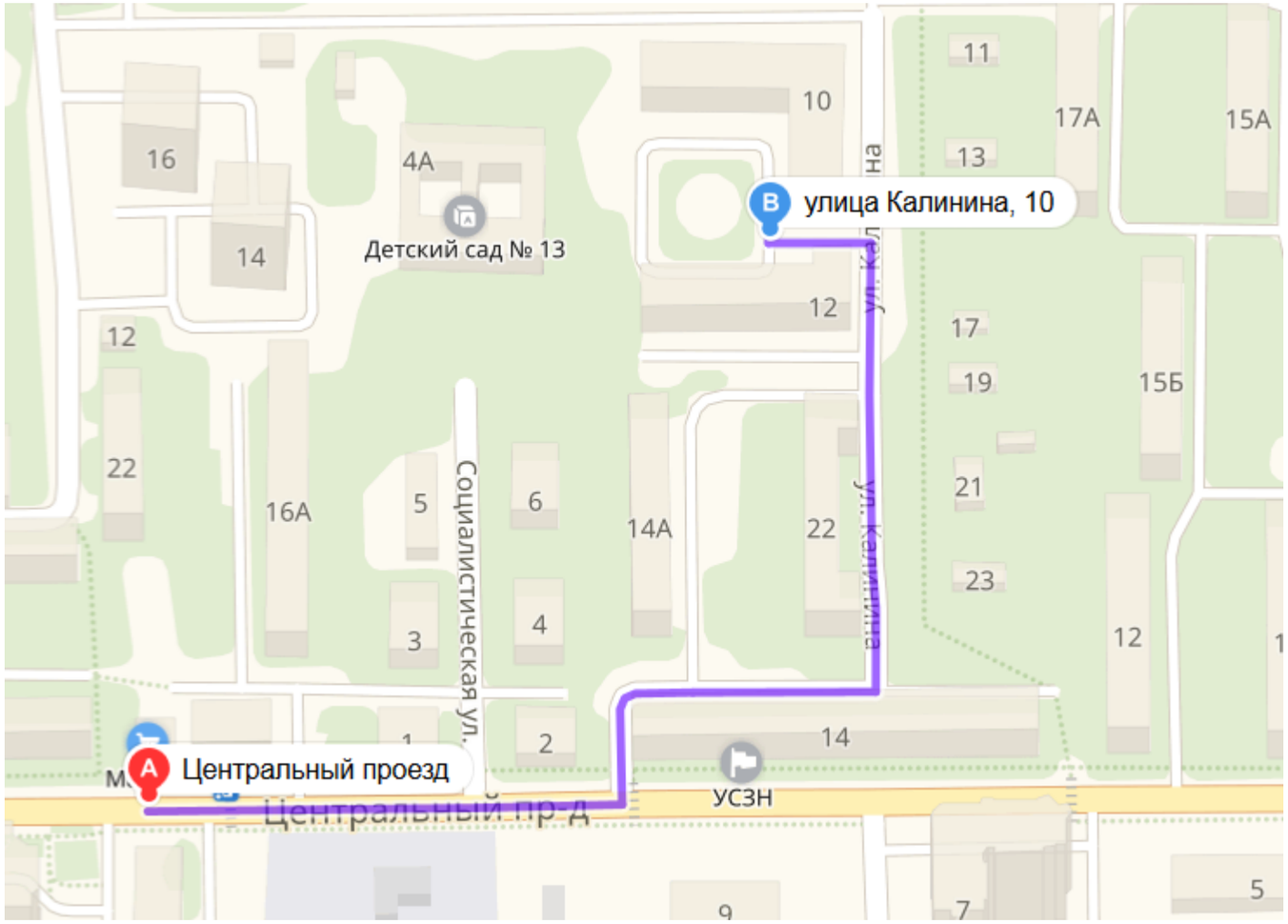
$L_{1.5}$



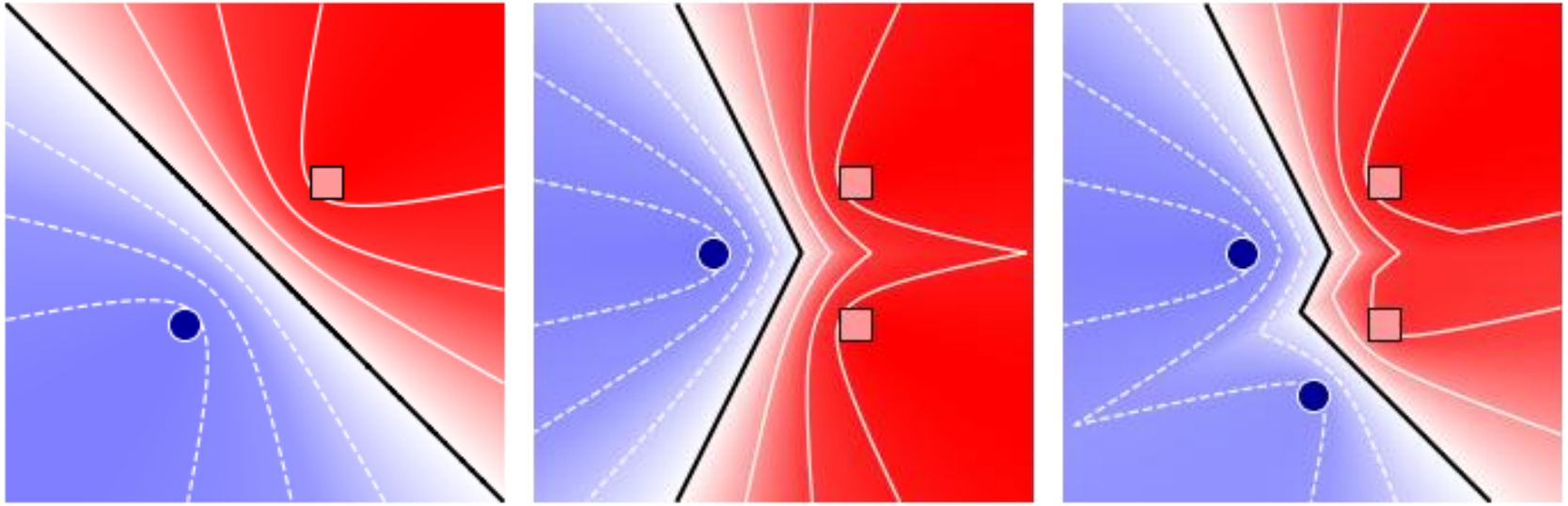
L_3

что такое L_0?

Различные метрики



Разделяющие поверхности L_2



Разделяющие поверхности L_1

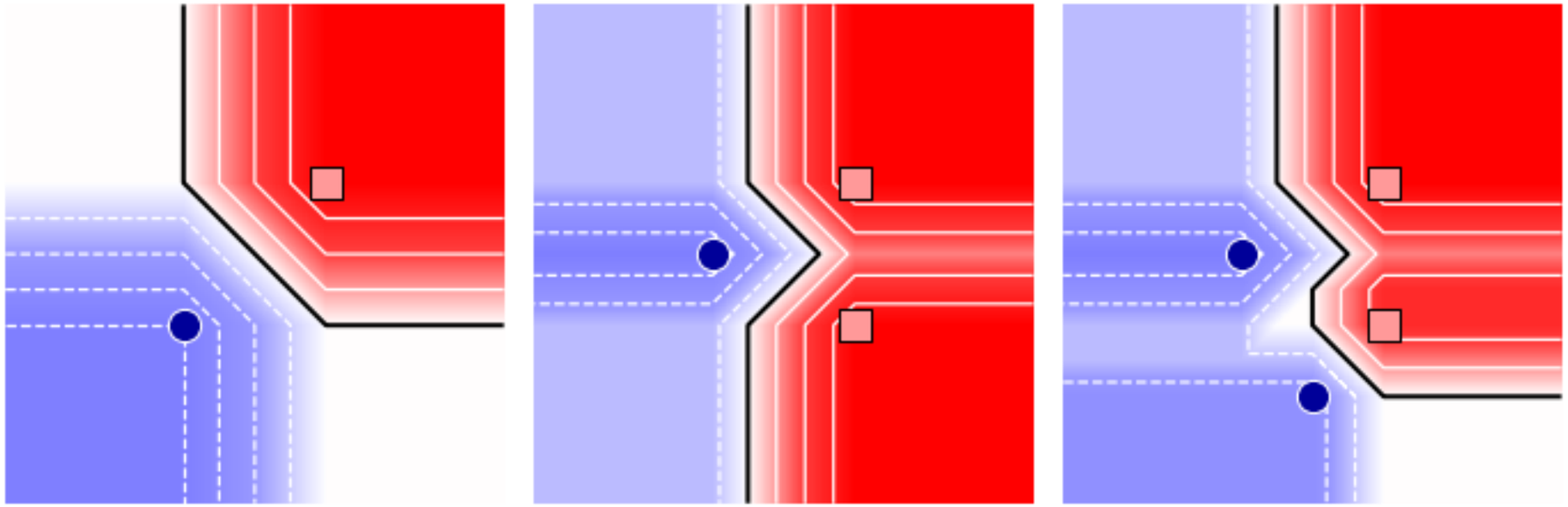
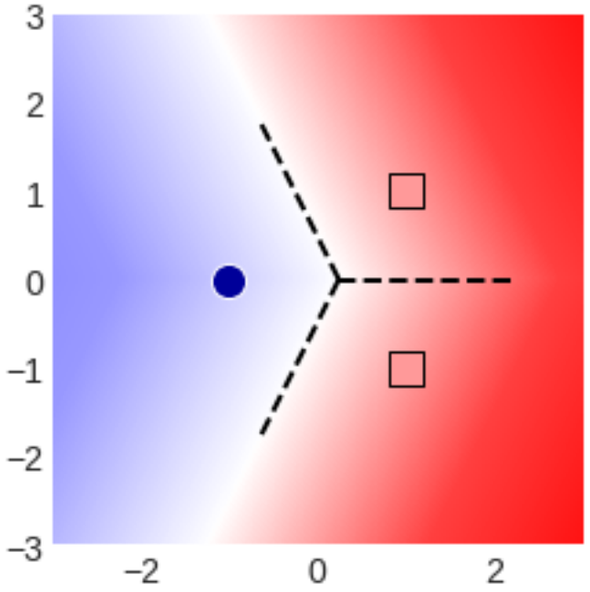
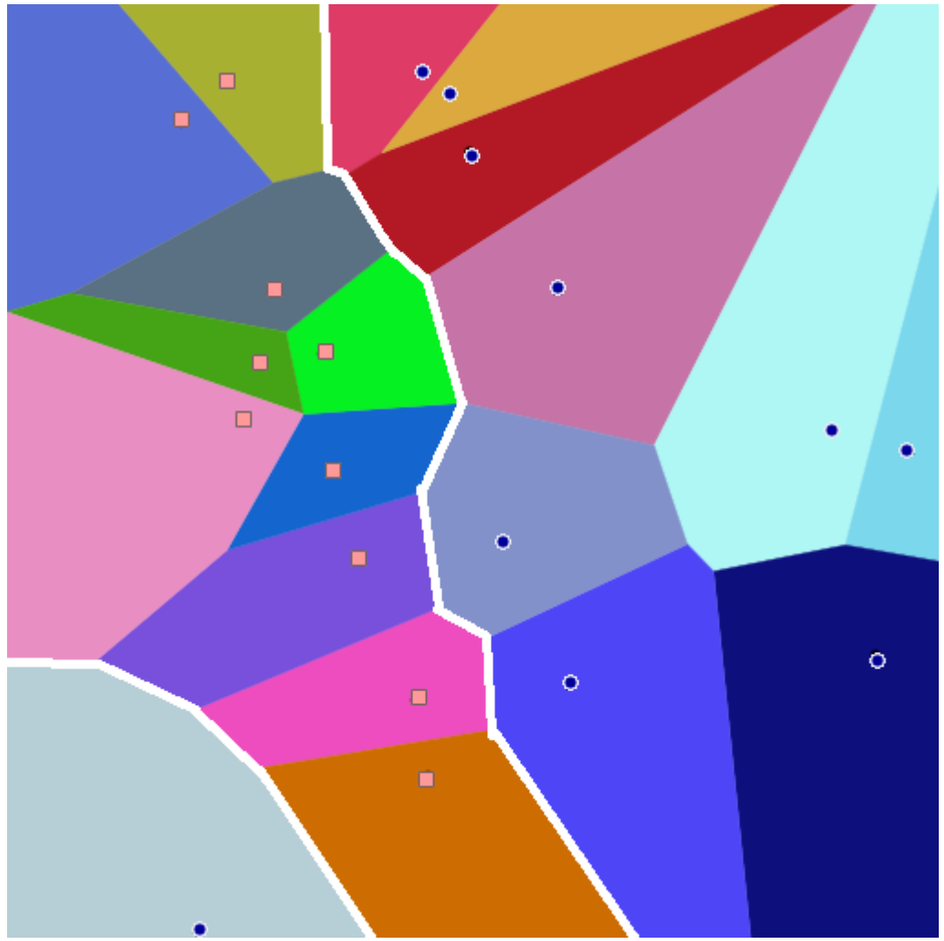


Диаграмма Вороного (Voronoi diagram)

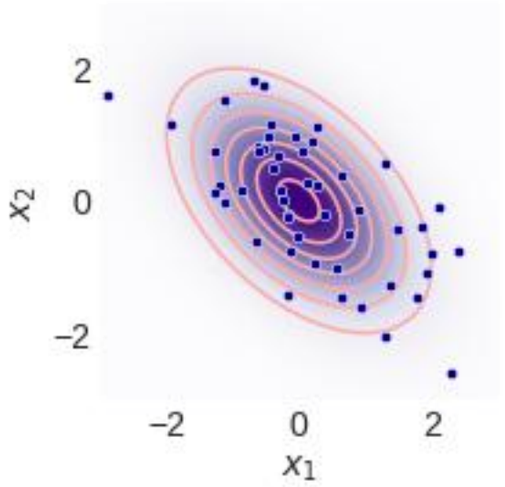




https://en.wikipedia.org/wiki/Voronoi_diagram

Расстояние Махаланобиса (Mahalanobis distance)

– евклидово расстояние после преобразования $x \to \varphi(x) = \Sigma^{-1/2}(x - \mu)$

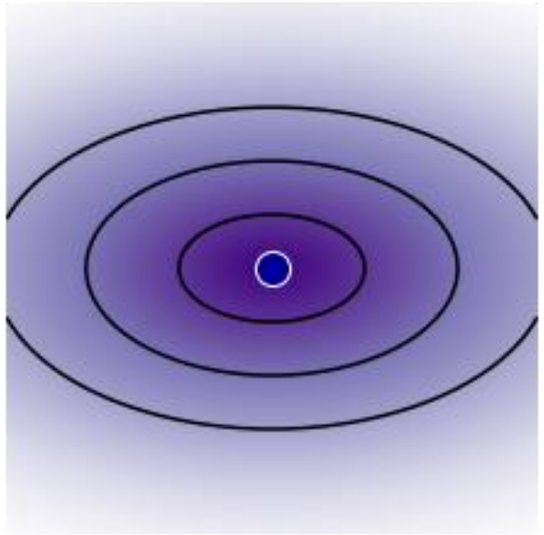


стандартизует нормальные данные

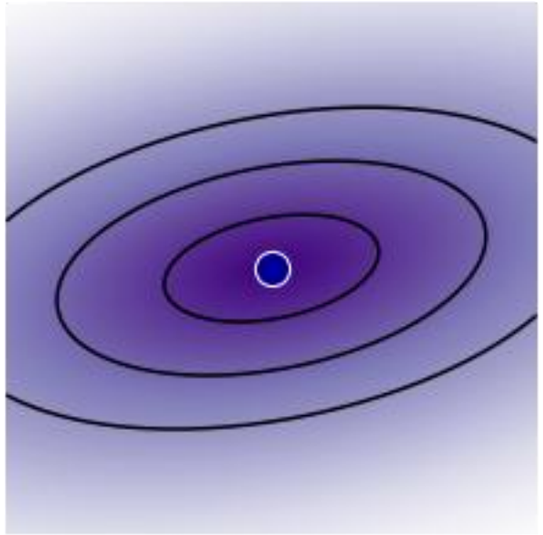
$$\text{norm}(\mu, \Sigma) \to \text{norm}(0, I)$$

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

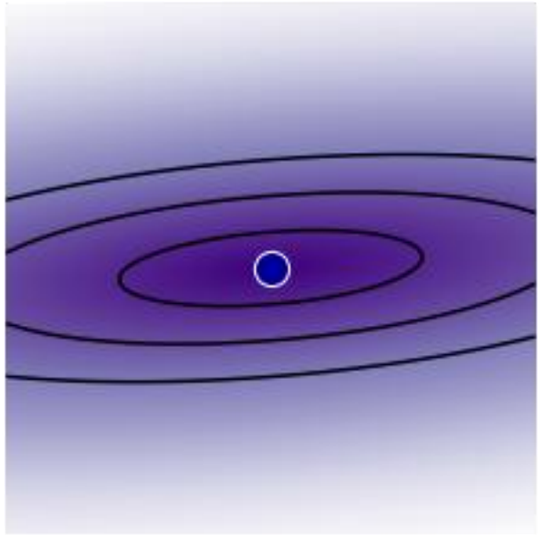
Расстояние Махаланобиса



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



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



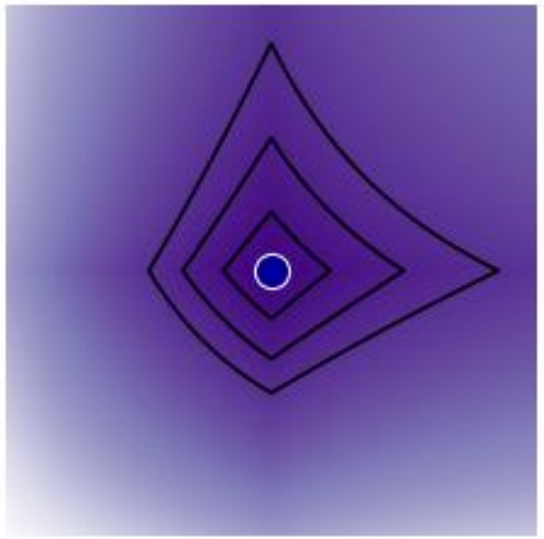
$$\Sigma = \begin{bmatrix} 4 & 0.3 \\ 0.3 & 0.25 \end{bmatrix}$$

Расстояния

Canberra distance

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

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



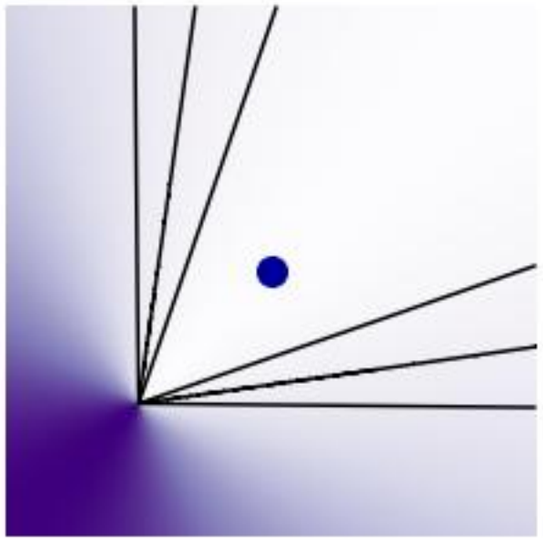
Хэмминга

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

Функции сходства

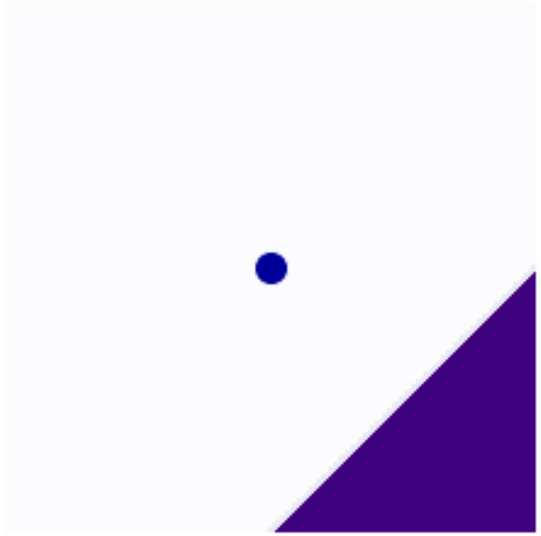
Косинусная мера сходство (не расстояние)

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



Коэффициенты корреляции

$$\mathrm{cor}(x,z) = \frac{(x-\overline{x})^{\mathrm{T}}(z-\overline{z})}{\sqrt{\| x-\overline{x} \|_2^2 \cdot \| z-\overline{z} \|_2^2}}$$



если работать с нормированными (+ центрированными) 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:

$$1-\frac{x^{\mathrm{T}}z}{x^{\mathrm{T}}x+z^{\mathrm{T}}z-x^{\mathrm{T}}z}=\frac{x^{\mathrm{T}}x+z^{\mathrm{T}}z-2x^{\mathrm{T}}z}{x^{\mathrm{T}}x+z^{\mathrm{T}}z-x^{\mathrm{T}}z}=\frac{\|x-z\|_2^2}{\|x-z\|_2^2+x^{\mathrm{T}}z}$$

тут есть много разных вариантов…

Проблема выбора метрики

- **зависимость от масштаба**

  нормировка признаков

  однородные признаки

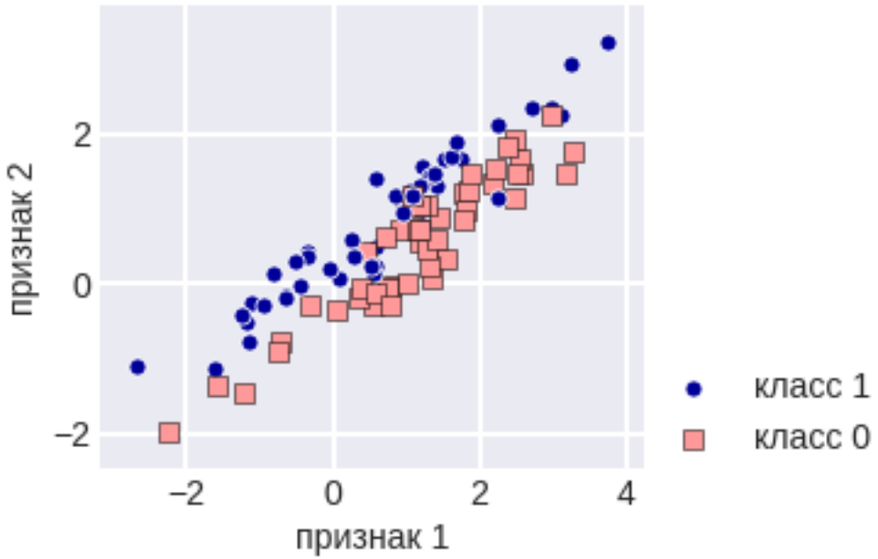
  смесь метрик

- **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

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

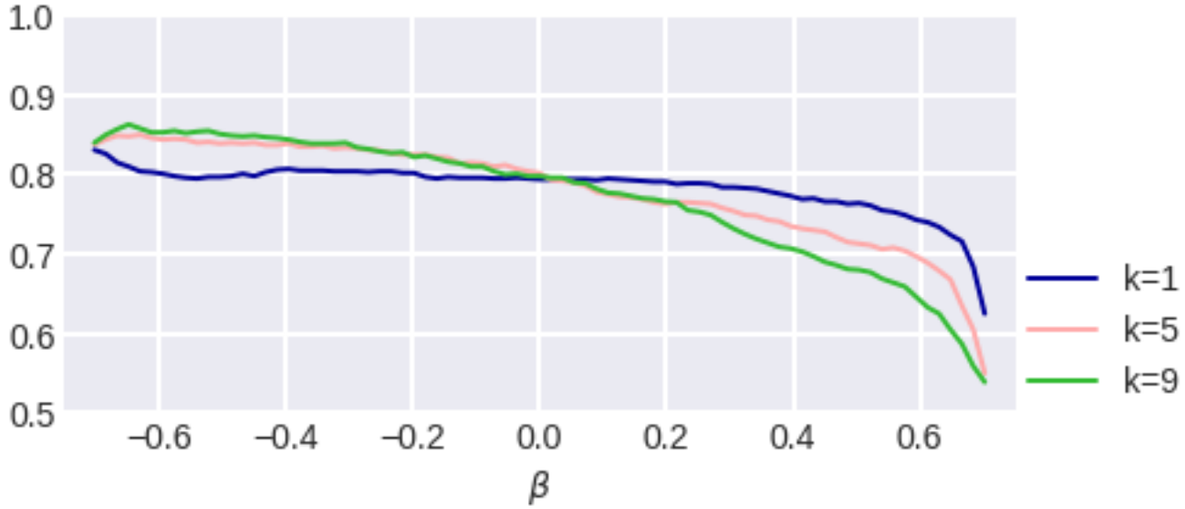
$$\rho(x,z) = \sqrt{(x-z)^{\mathrm{T}}\Sigma^{-1}(x-z)}$$

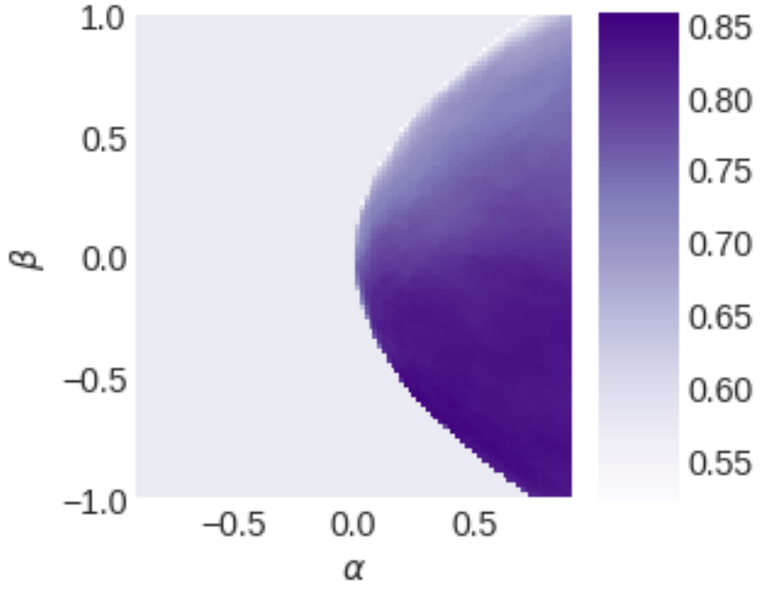
Пусть Σ^{-1} – матрица параметров…

Оптимизировать

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

Обучение метрики: Metric Learning






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

Q = np.zeros([101, 101])
LS1, LS2 = np.linspace(-0.9, 0.9, 101), np.linspace(-1, 1, 101)
for i, b in enumerate(LS1):
    for j, a in enumerate(LS2):
        if a - b*b < 0.00001:
            q = np.nan
        else:
            A = np.linalg.inv(np.array([[a, b], [b, 1]]))
            model = KNeighborsClassifier(n_neighbors=5,  metric='mahalanobis',
                                         metric_params={'V': A})
            model.fit(X, y)
            a = model.predict(X2)
            q = accuracy_score(y2, a)
        Q[i, j] = q
```

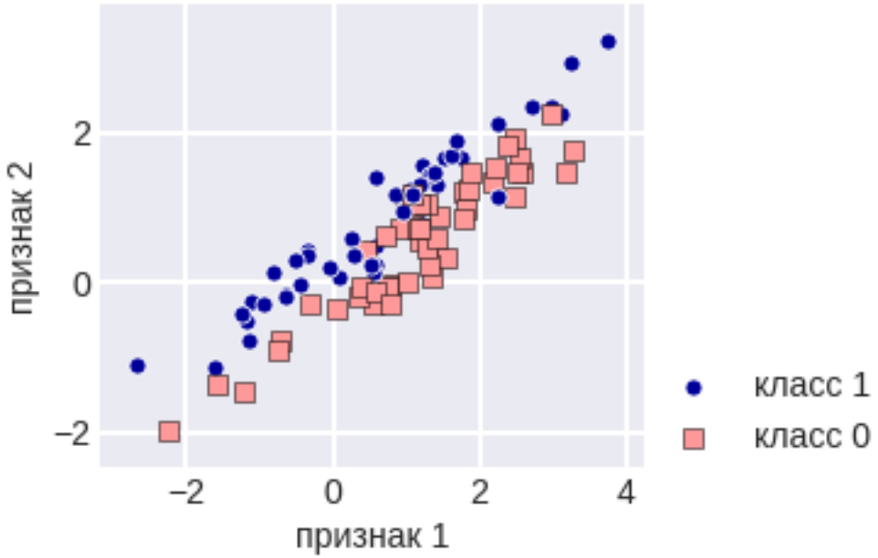
$$\Sigma = \begin{bmatrix} \alpha & \beta \\ \beta & 1 \end{bmatrix}$$

здесь можно перебрать параметры, а вообще – методы оптимизации

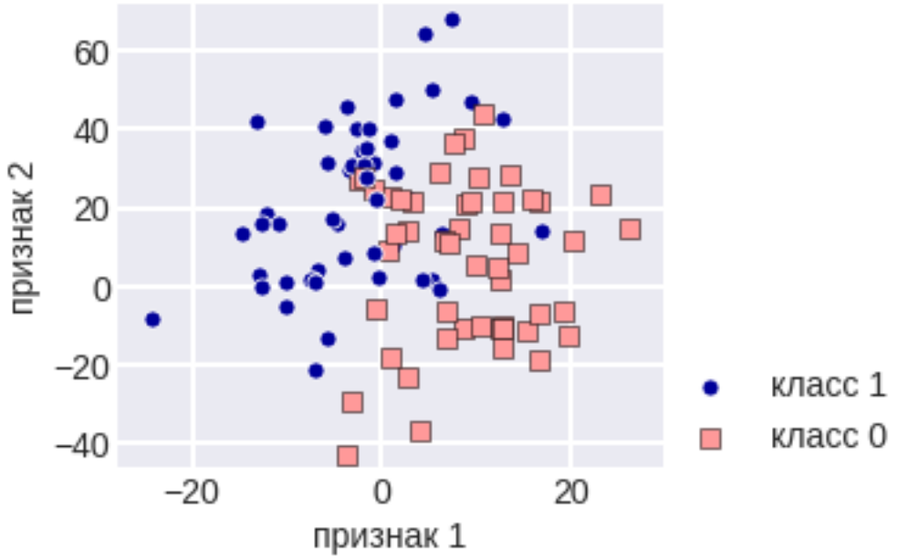
почему такой рис?

Обучение метрики: Metric Learning

Neighborhood Components Analysis – 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

Neighborhood Components Analysis – что под капотом:

ищем преобразование $x \to \varphi(x) = L(x - \mu)$

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

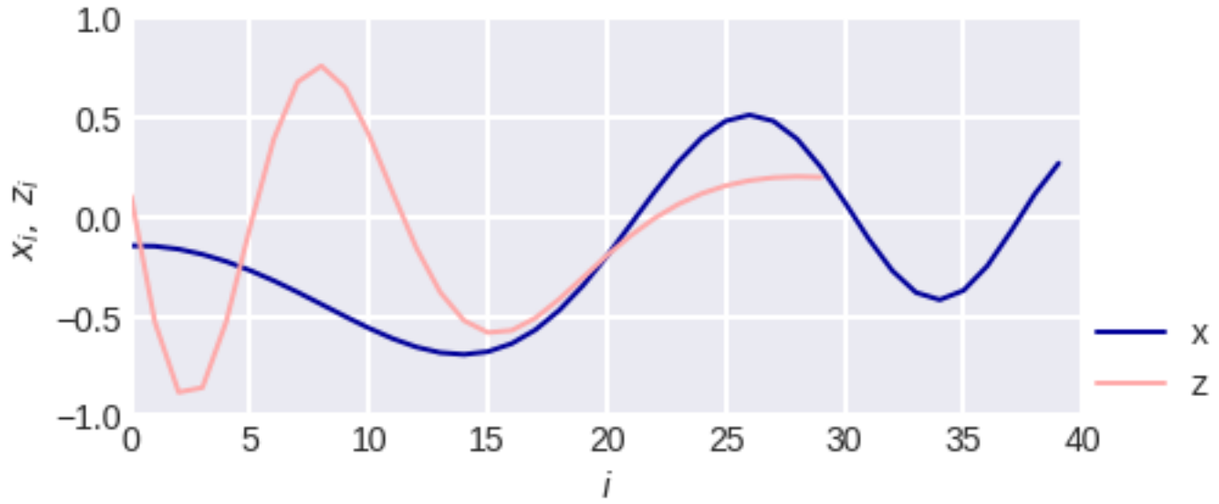
если $x_i \in K_s$, **то** $p_i = \sum_{x_j \in K_s} p_{ij}$

(сумма вероятностей по объектам этого же класса)

задача оптимизации

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

Метрики на временных рядах



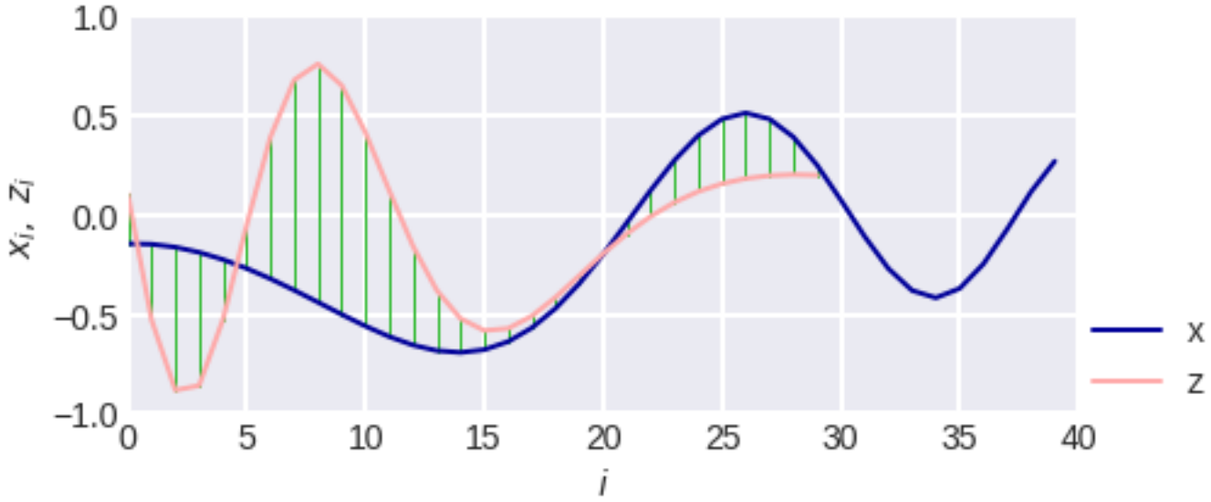
как ввести расстояние?

Ряды могут быть разной длины…

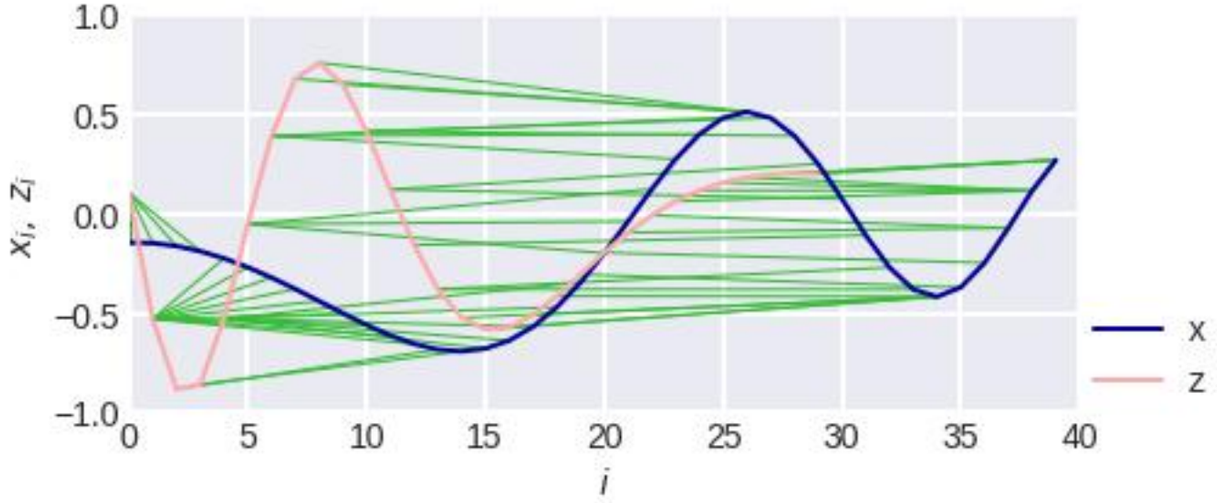
Как оценить похожесть формы?

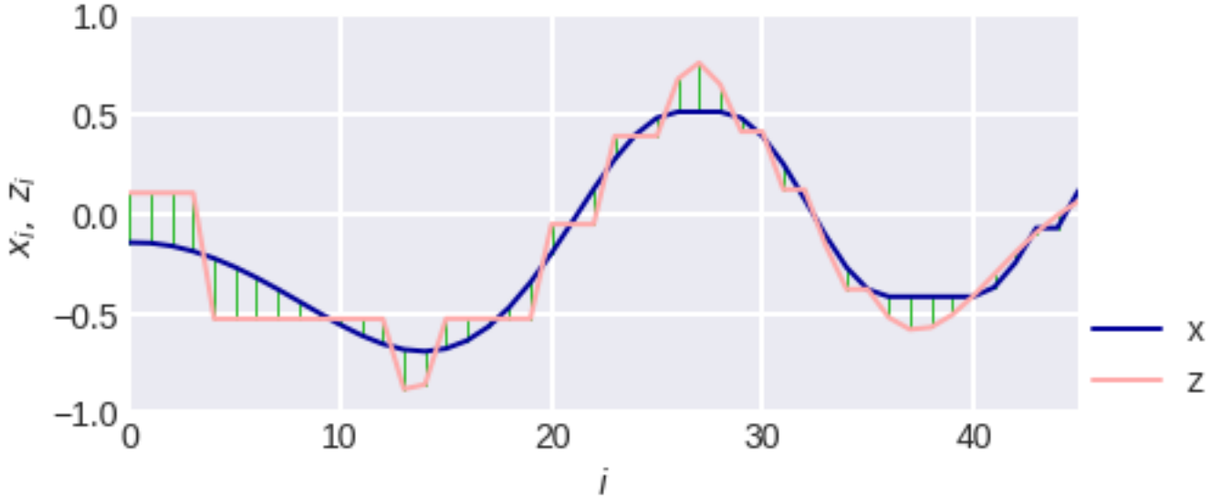
Метрики на временных рядах

Евклидово расстояние



DTW = Dynamic time warping





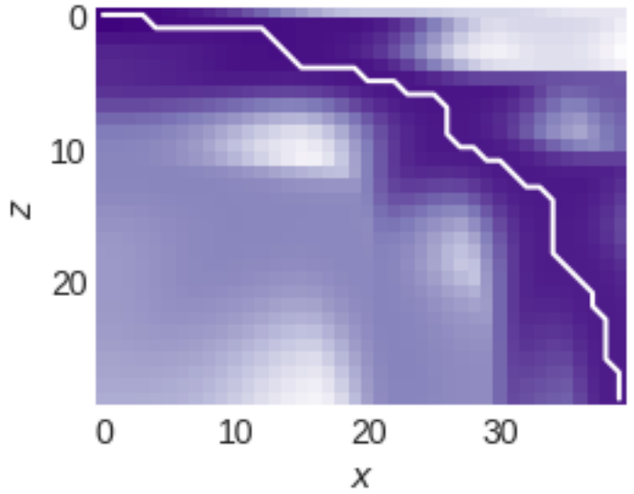
Метрики на временных рядах

матрица расстояний между точками

$$\left[\begin{array}{c|cccc} & x_1 & x_2 & \cdots & x_m \\ \hline z_1 & |x_1 - z_1| & |x_2 - z_1| & \cdots & |x_m - z_1| \\ z_2 & |x_1 - z_2| & |x_2 - z_2| & \cdots & |x_m - z_2| \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ z_n & |x_1 - z_n| & |x_2 - z_n| & \cdots & |x_m - z_n| \end{array}\right]$$

Надо найти соответствие…

$$1 = g(1) \le g(2) \le \ldots \le g(k) = m$$
$$\{g(1), g(2), \ldots, g(k)\} = \{1, 2, \ldots, m\}$$
$$1 = h(1) \le h(2) \le \ldots \le h(k) = n$$
$$\{h(1), h(2), \ldots, h(k)\} = \{1, 2, \ldots, n\}$$
$$\sum_{i=1}^{k} |x_{g(i)} - z_{h(i)}| \to \min$$



Метрики на временных рядах

Пусть есть векторы (временные ряды): $x = (x_1, \ldots, x_m),\ z = (z_1, \ldots, z_n)$

срез – $x[:i] = (x_1, \ldots, x_i)$

рекурсивное определение:

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

начальные условия рекурсивного определения:

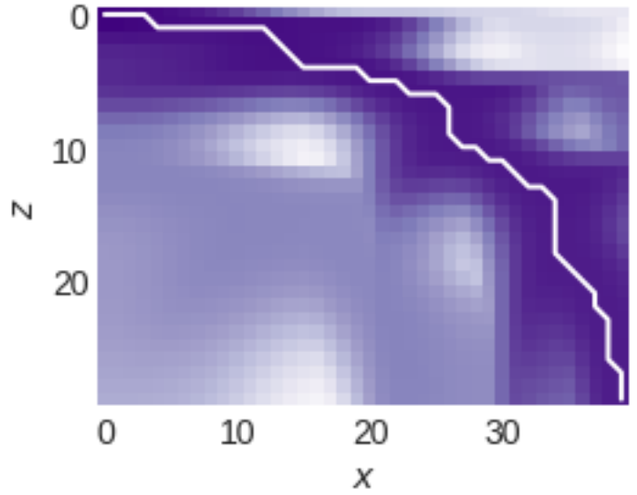
$$\mathrm{DTW}(x[:0], z[:0]) = 0$$
$$\mathrm{DTW}(x[:i], z[:0]) = \mathrm{DTW}(x[:0], z[:i]) = \infty \quad \text{при} \quad i > 0$$

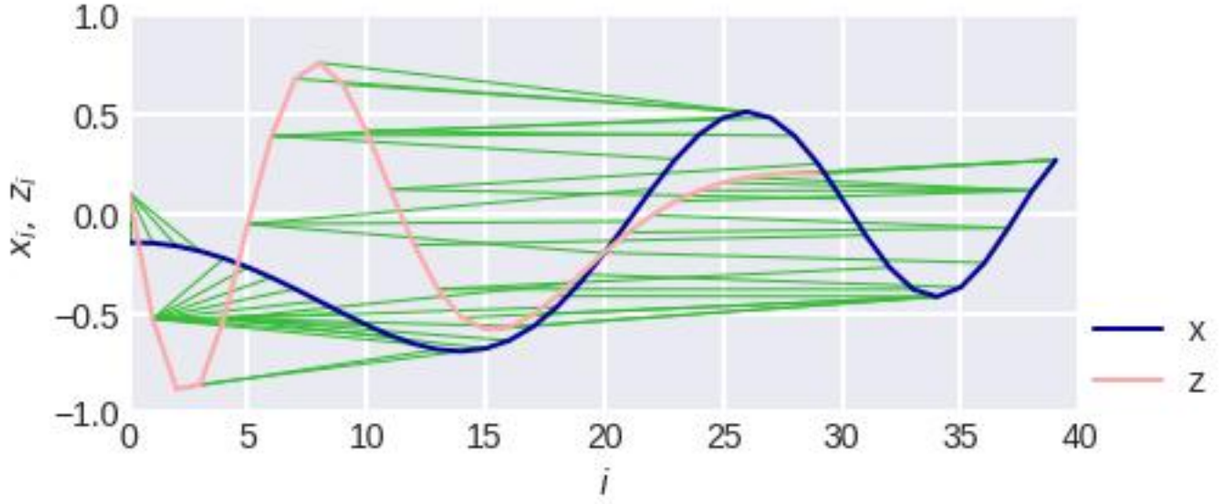
здесь нет нормировки… хотя может быть

Метрики на временных рядах

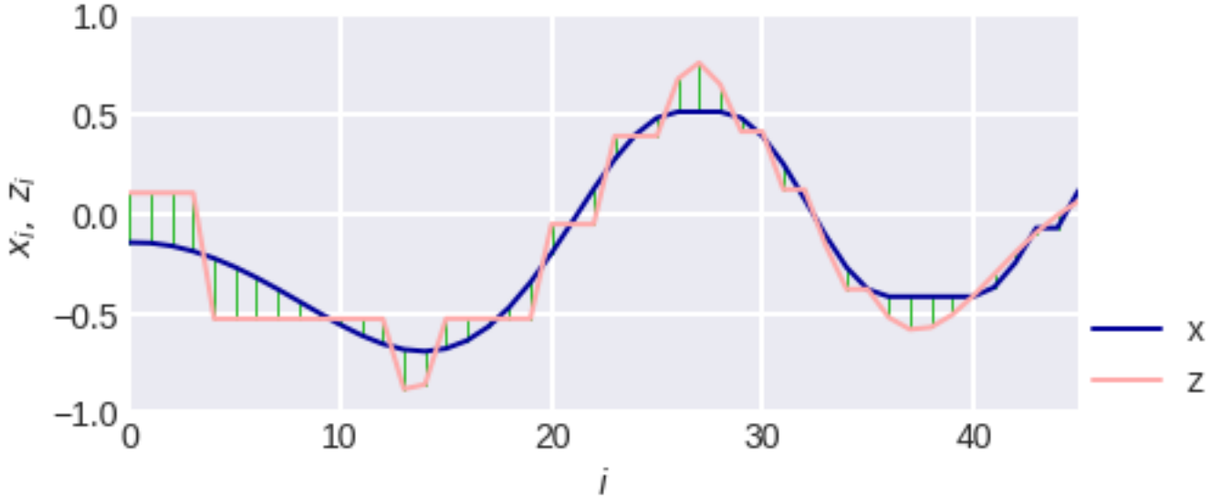
Для адекватности и быстроты часто

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





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 их одной строки можно получить другую

использование: исправление опечаток

Расстояние Левенштейна

определение аналогично DTW

(можно для каждой операции – свой штраф)

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

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

Приложения метрического подхода: нечёткий матчинг таблиц

↔

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

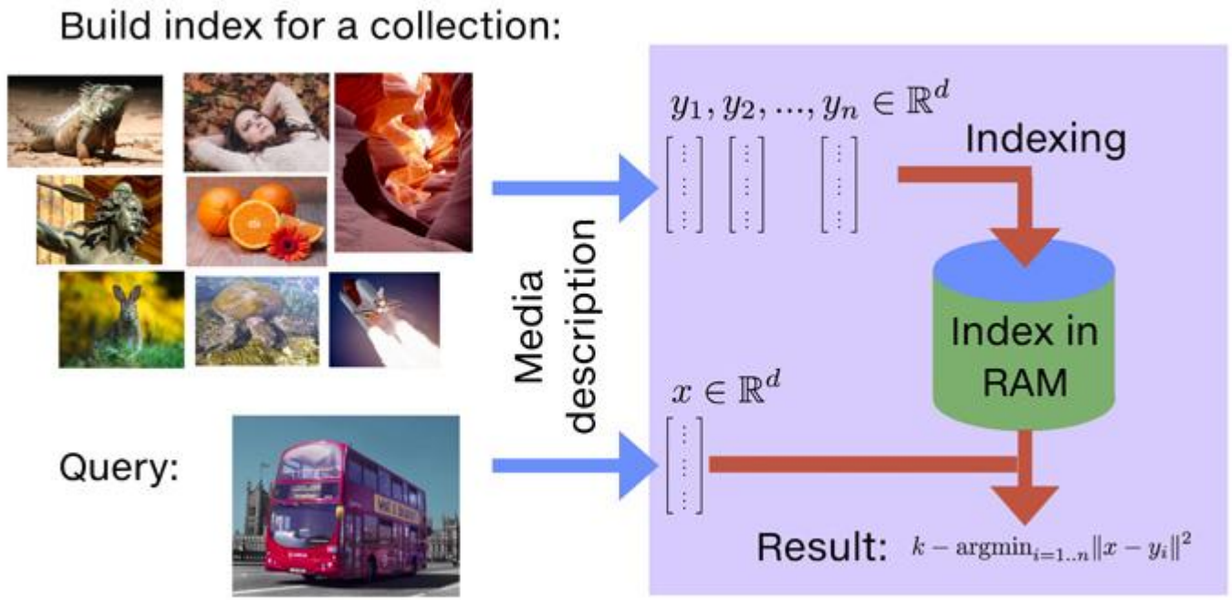
101, 'Lecture', 'eng', 'biology', '2008-12-04', '2009-02-12', 'Implementing a common framework on business', 'Professor Rudolf Smith', ...

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

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.

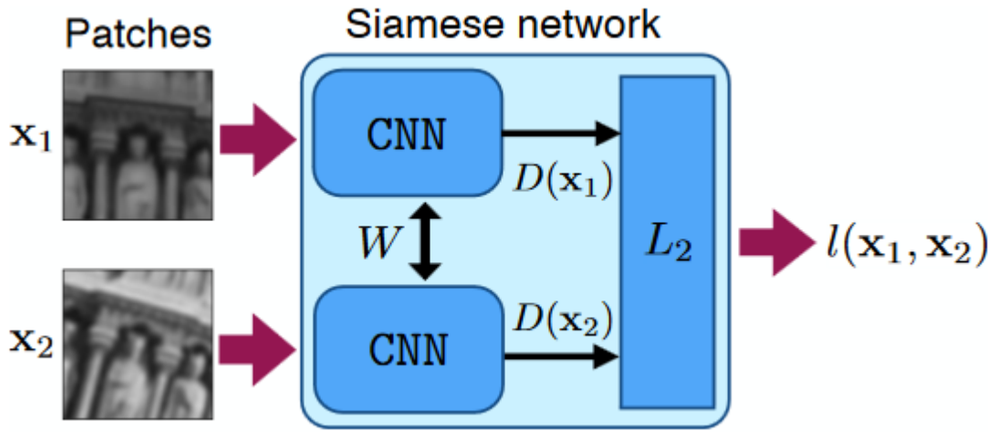
Приложения метрического подхода: поиск похожих объектов



звука, изображения, видео, …

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

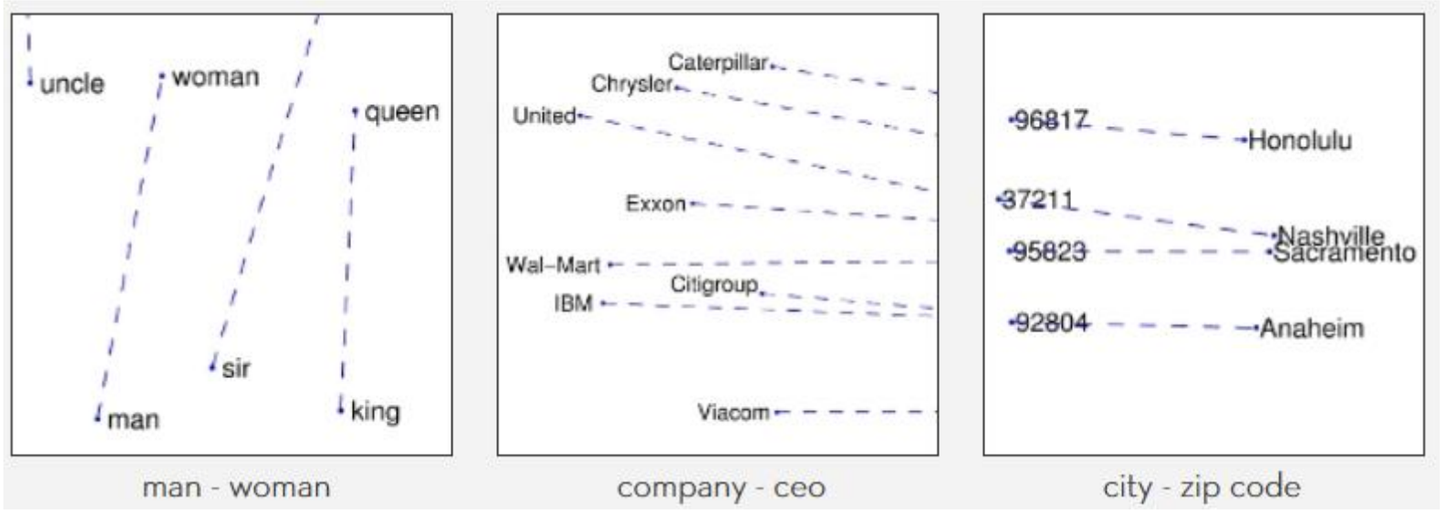
Приложения метрического подхода: Сиамские сети



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/

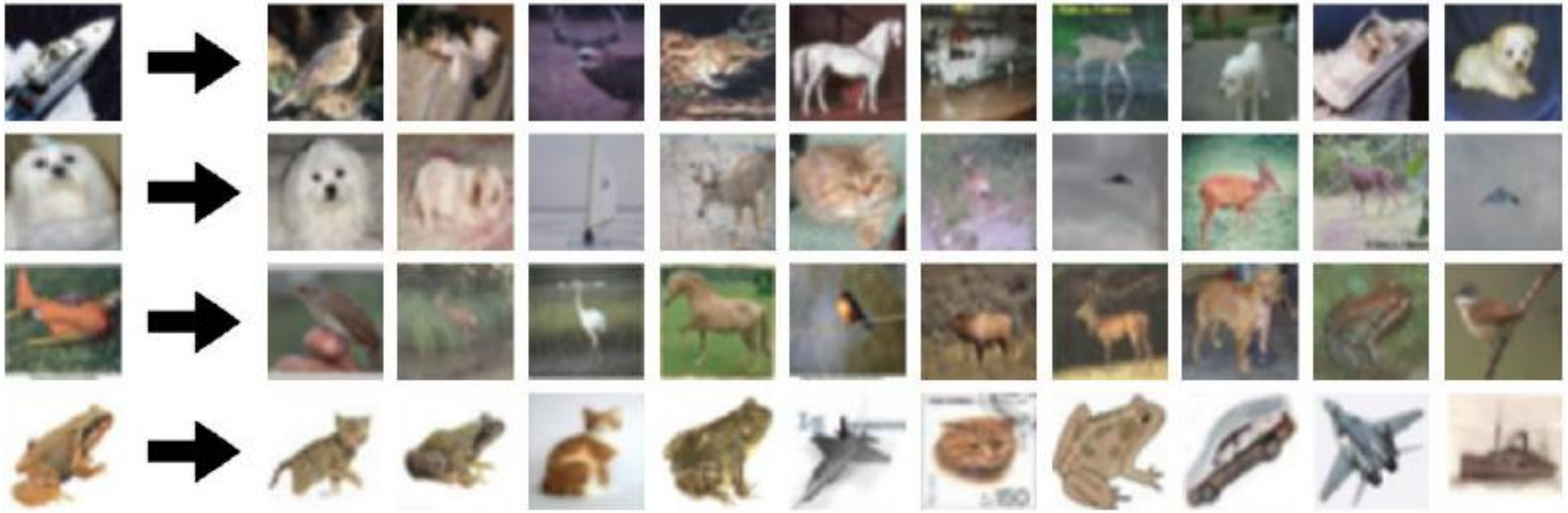
Приложения метрического подхода: SIFT image descriptors

TEXMEX

датасеты для приближённого поиска соседей

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

Приложения метрического подхода: простота интерпретаций

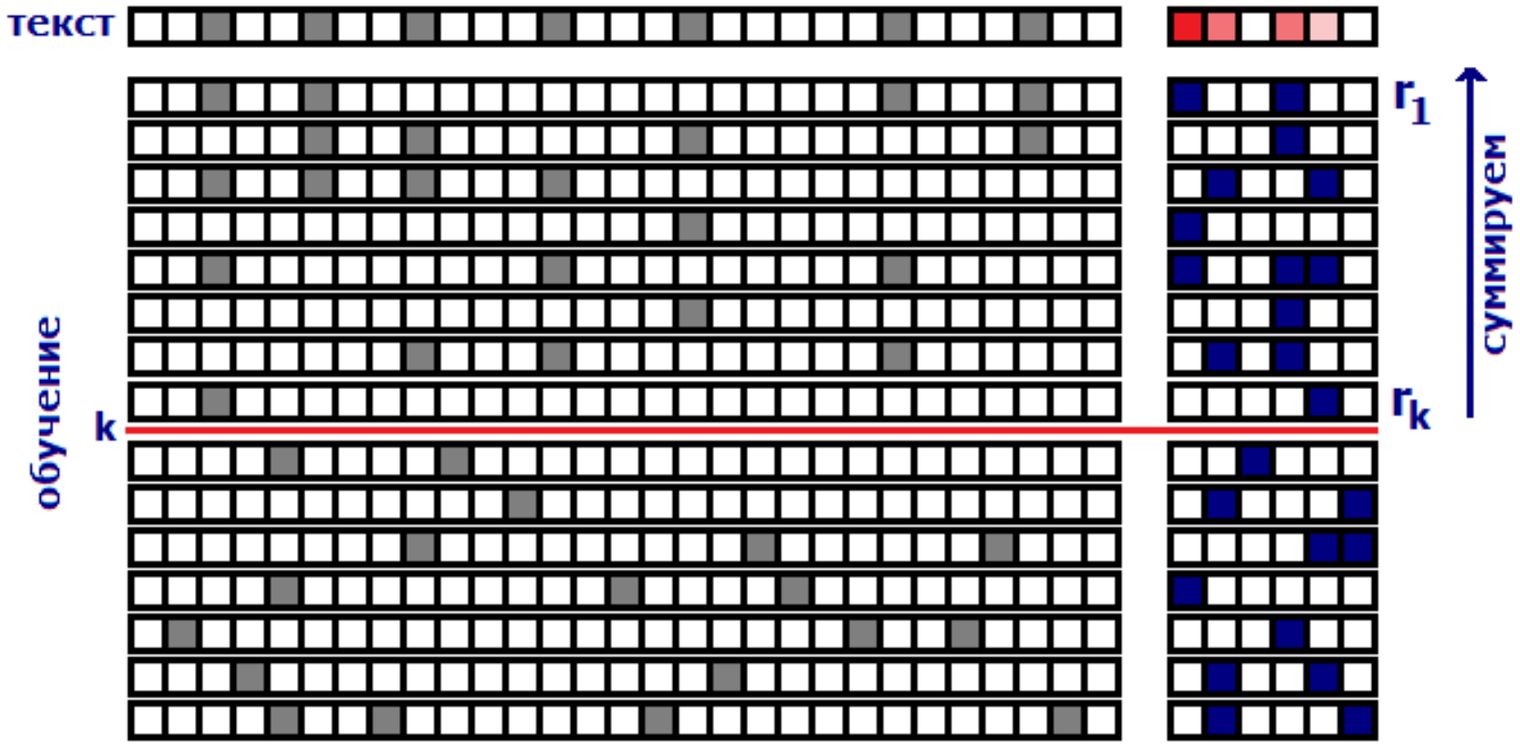


http://cs231n.stanford.edu/2017/syllabus.html

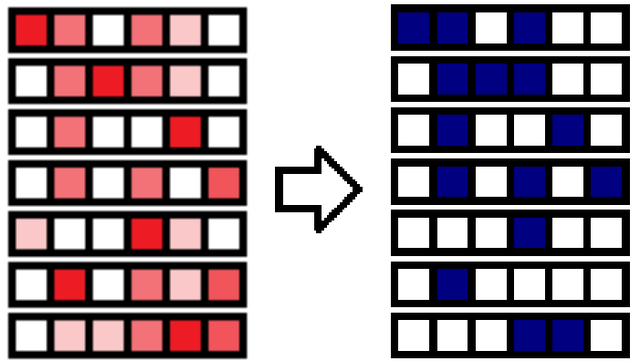
Приложения метрического подхода: классификация текстов

задача «Large Scale Hierarchical Text Classification»

Взвешенный kNN



Итог – матрица оценок



https://www.kaggle.com/c/lshtc

Приложения метрического подхода: классификация текстов

задача «Large Scale Hierarchical Text Classification»

вычисление центроидов

Приложения метрического подхода: прогнозирование рядов

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

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

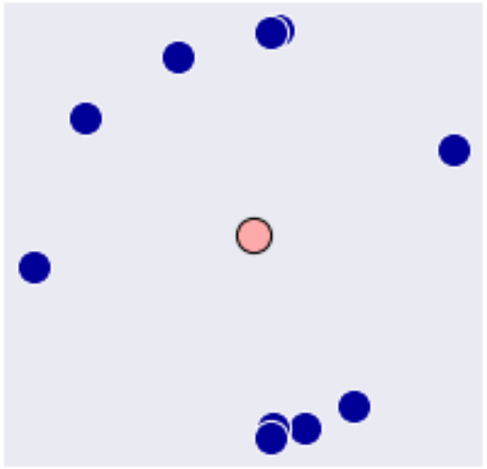
$$A(x_1,\ldots,x_l) = (a_1x_1 + a_2,\ldots,a_1x_l + a_2)$$

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

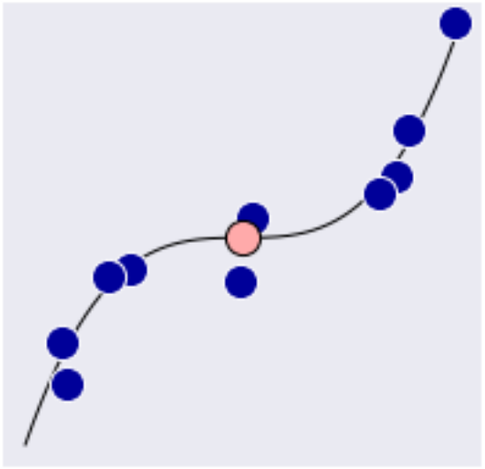
http://www.neural-forecasting-competition.com/NN5/results.htm

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 признаковых 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!!!

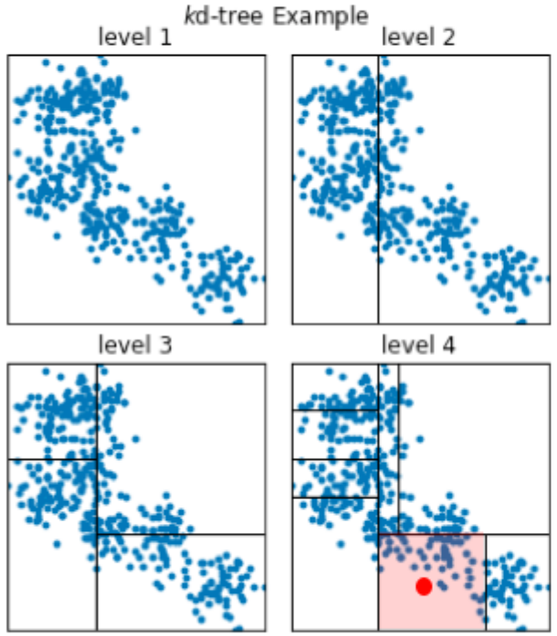
Эффективность

быстрые точные и приближённые методы поиска соседей

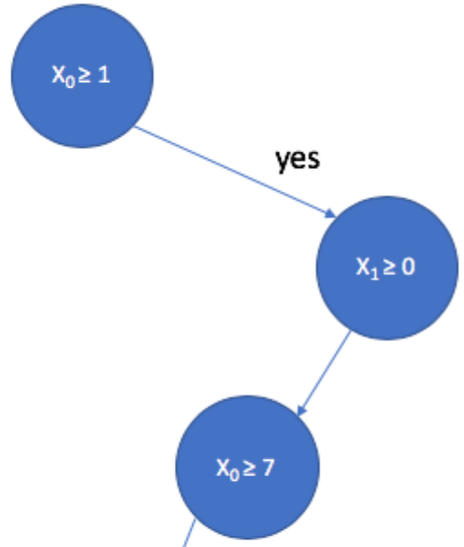
Аппроксимация ~ с большой вероятностью находим точку, которая не сильно дальше, чем ближайшая

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

(3, -1)
(6, -2)
...
(4, -1)

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

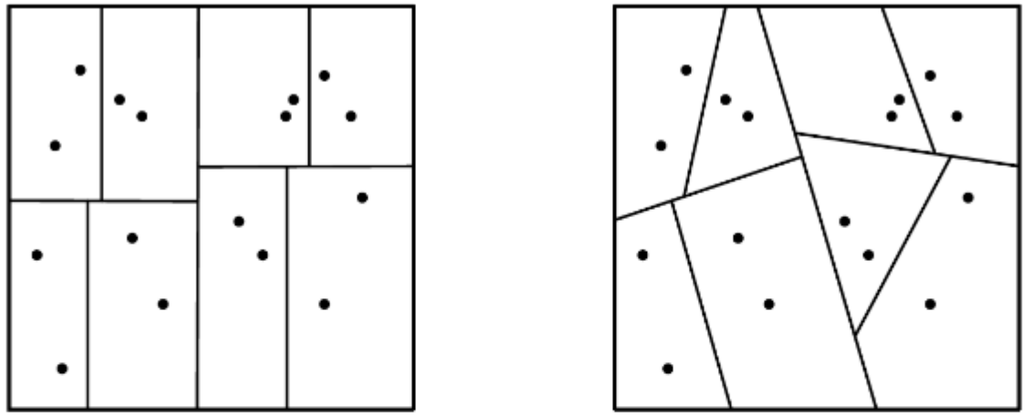


Figure 6.3: A comparison between how a k-d tree and a randomized partition tree divide up the feature space for two dimensional data (figure source: Dasgupta and Sinha 2015). Left: a k-d tree uses axis-aligned splits. Right: a randomized partition tree uses splits with directionality that is randomized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

Для приближённого поиска можно перебирать только объекты в листе

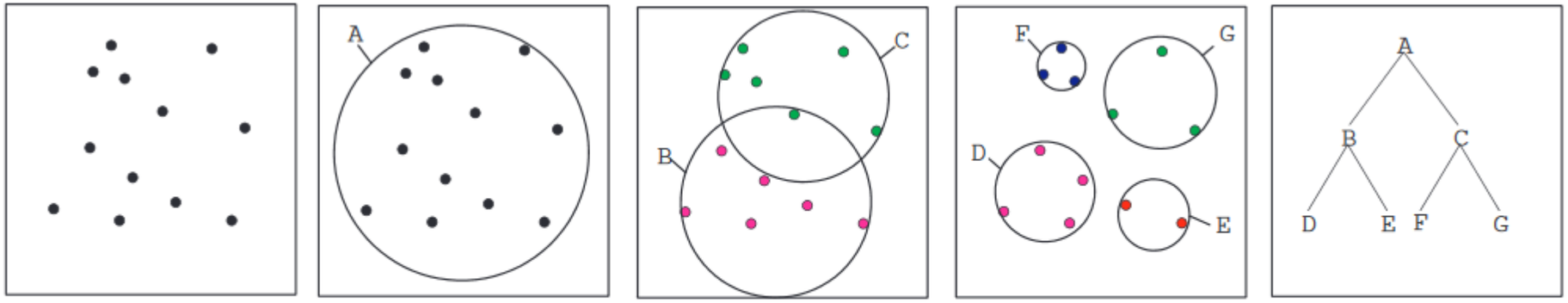
Можно построить несколько деревьев…

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



1a. A dataset | 1b. Root node | 1c. The 2 children | 1d. The 4 grandchildren | 1e. The internal tree structure

Figure 1: An example of ball-tree structure

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

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

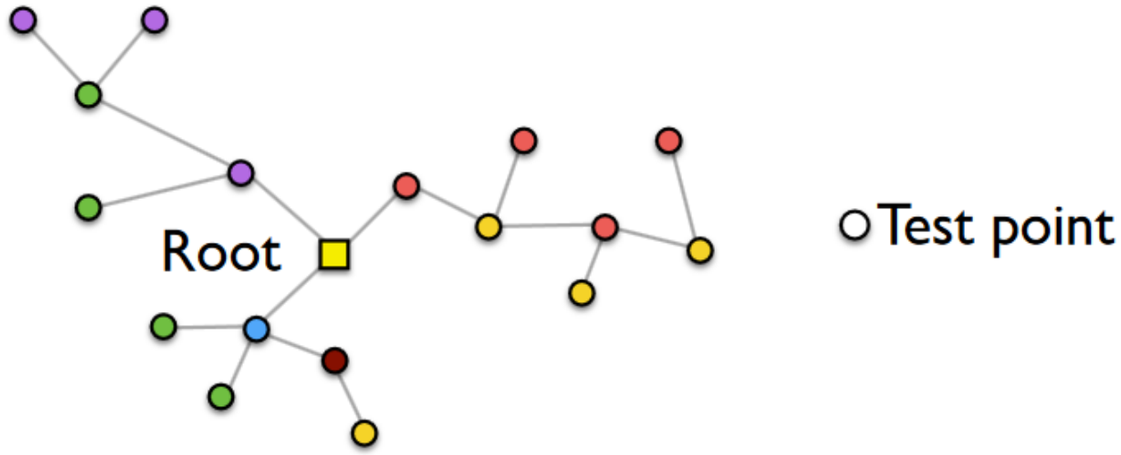
LSH ~ hash-функция: близкие объекты имеют похожие хэши

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

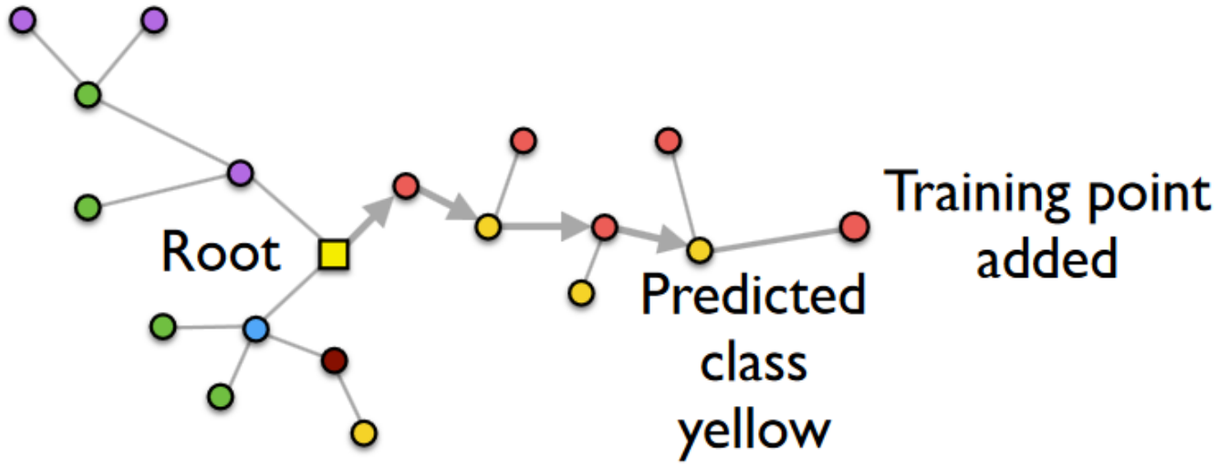


Режим теста

- **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), то нашли «соседа»**

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

Эффективность: Boundary Trees



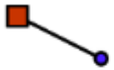
Режим обучения

- **Есть текущее 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) и метка текущей вершины отлична от метки точки – добавляем вершину (как потомок текущей вершины)**

Эффективность: Boundary Trees


Boundary Tree

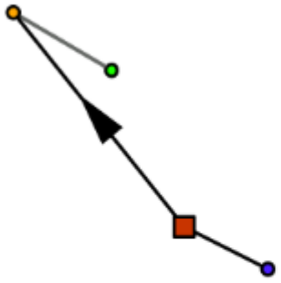

Querying BT everywhere


Boundary Tree

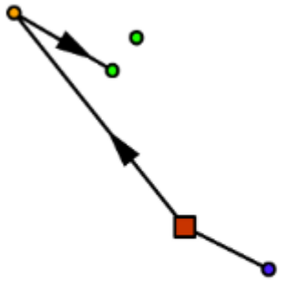

Querying BT everywhere


Boundary Tree

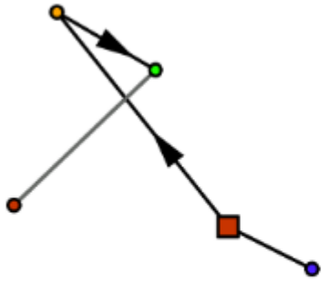

Querying BT everywhere


Boundary Tree


Querying BT everywhere

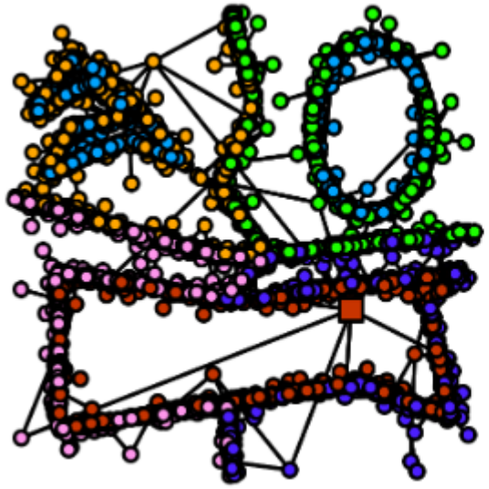

Boundary Tree


Querying BT everywhere


Boundary Tree

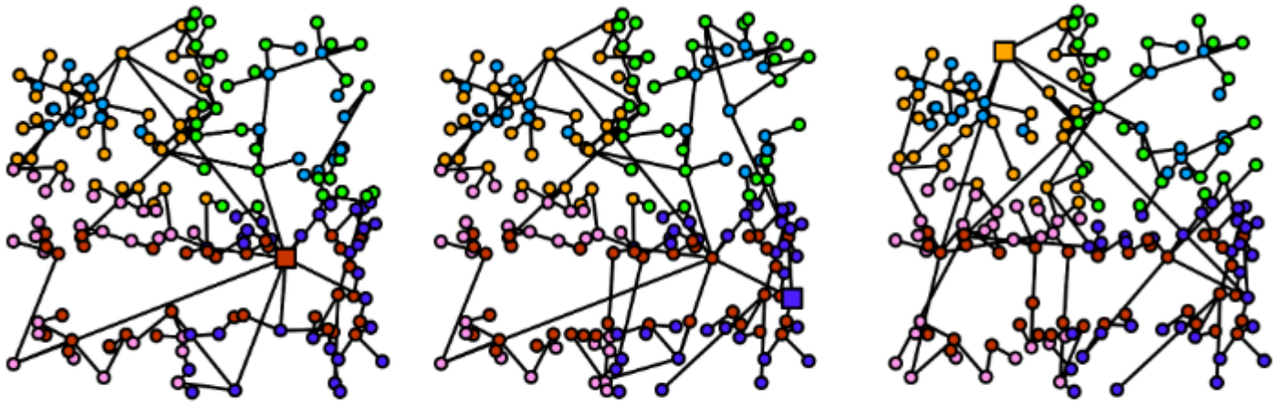

Querying BT everywhere

Эффективность: Boundary Trees



Boundary Tree

Querying BT everywhere



Shown 10,000 points

Модельная задача:

m = 100 000

в дереве |V| = 2 220

Лес:

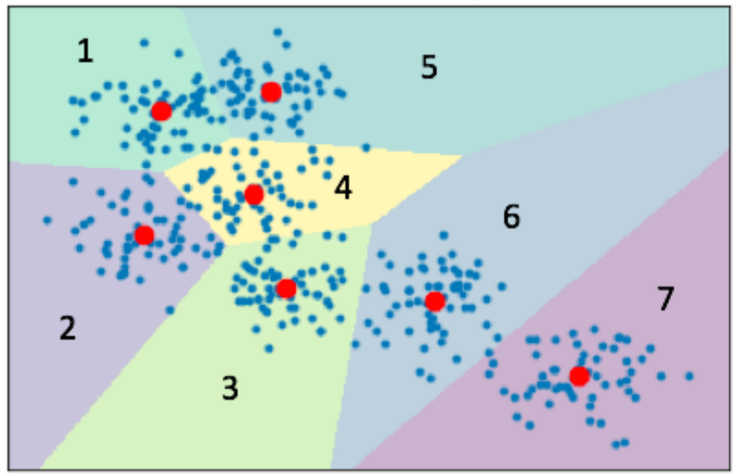
Строим с разным обходом выборки

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	...

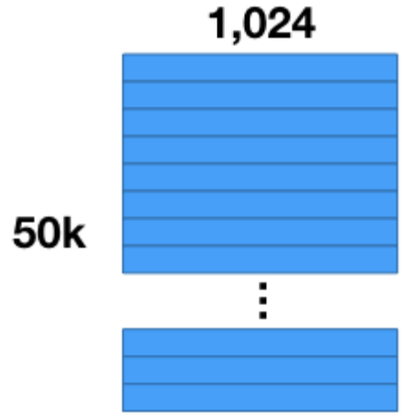
Assuming each vector has an identifier, we can maintain a list of vectors for each Voronoi cell. This type of data structure is known as an **inverted index**.

предварительная кластеризация

инвертированный индекс!

Эффективность: квантование «Product quantization»

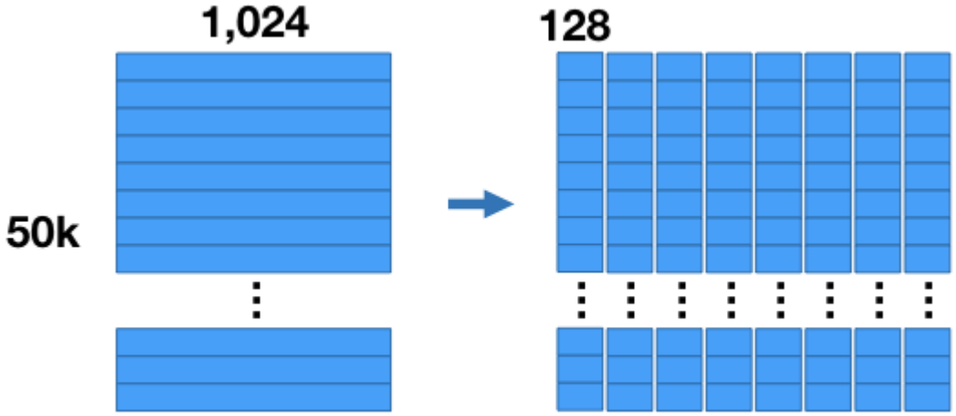
по-прежнему будем сравнивать с каждым из m объектов,
но упростим сравнение



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

Сейчас перейдём ещё к меньшей размерности

но это не снижение размерности (новое пр-во дискретное)

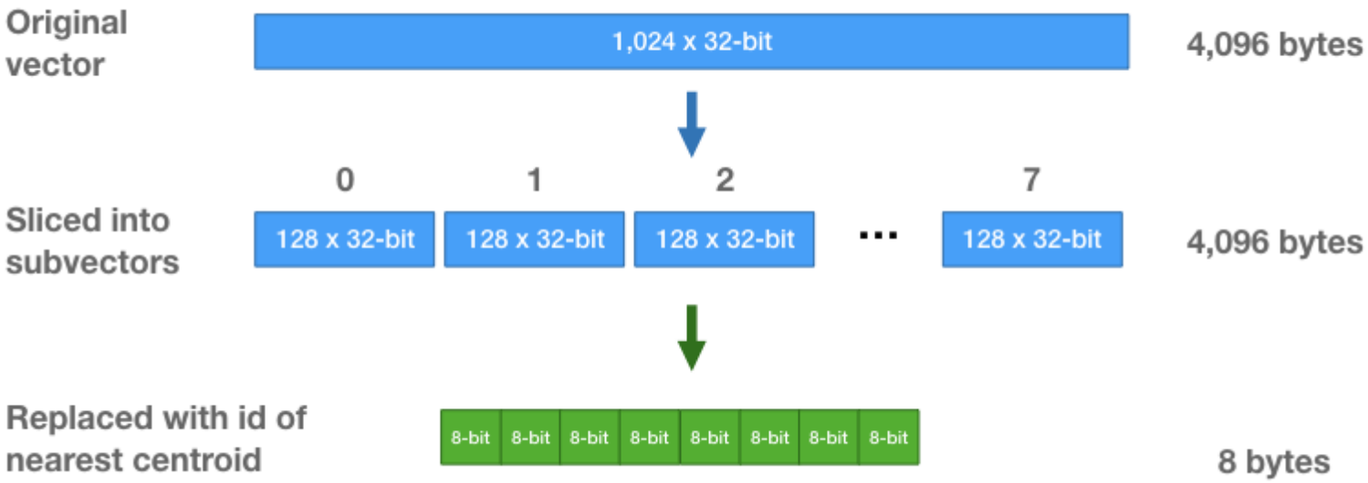


Разделим пространство на подпростанства (декартовым образом) 128*8

В каждом найдём k=256 центроидов

Будем хранить номера центроидов

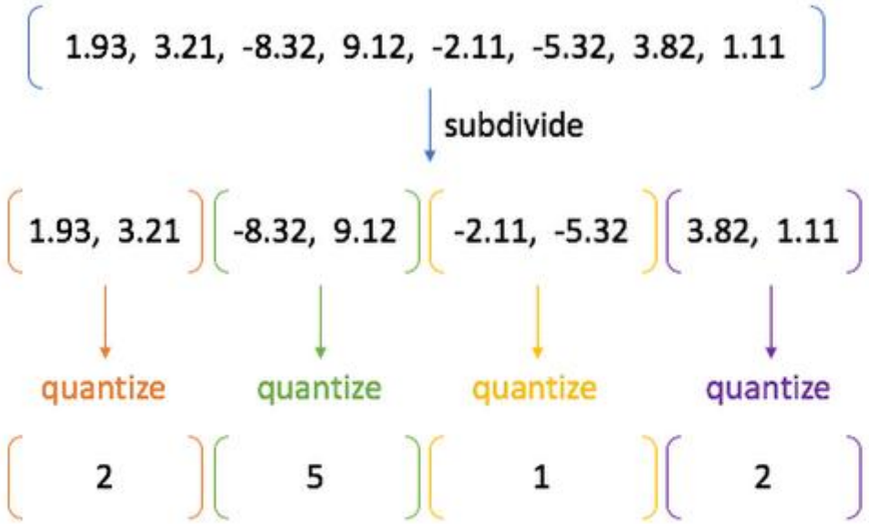
Эффективность: квантование «Product quantization»



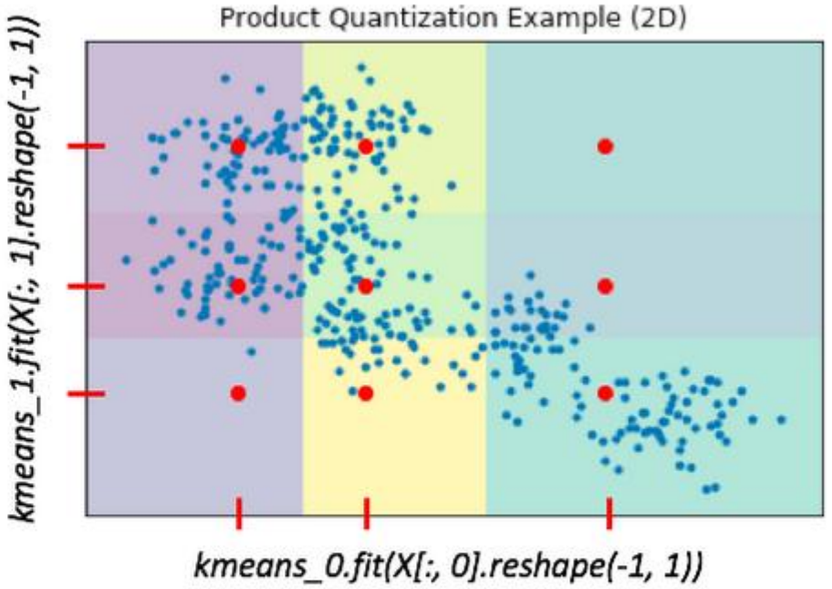
Для хранения id центроида – 8 бит

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

Эффективность: Product quantization



8D example



$X = (x_0, x_1)$

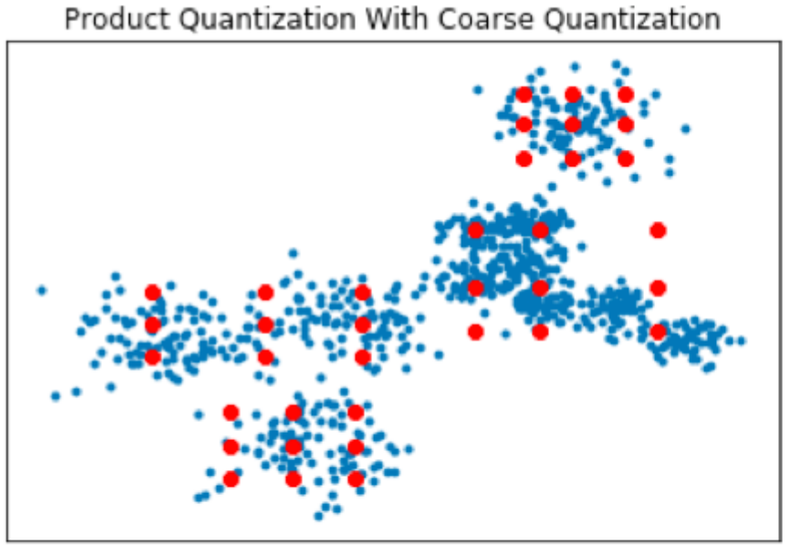
2D example (easier to visualize)

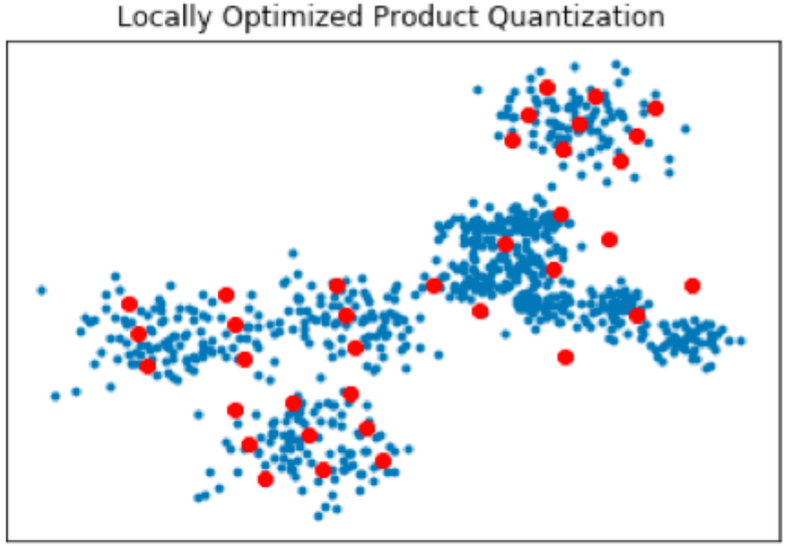
квантование по отдельным координатам (или подпространствам)

см. также

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

Эффективность: Locally optimized product quantization





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

Эффективные методы поиска ближайших соседей

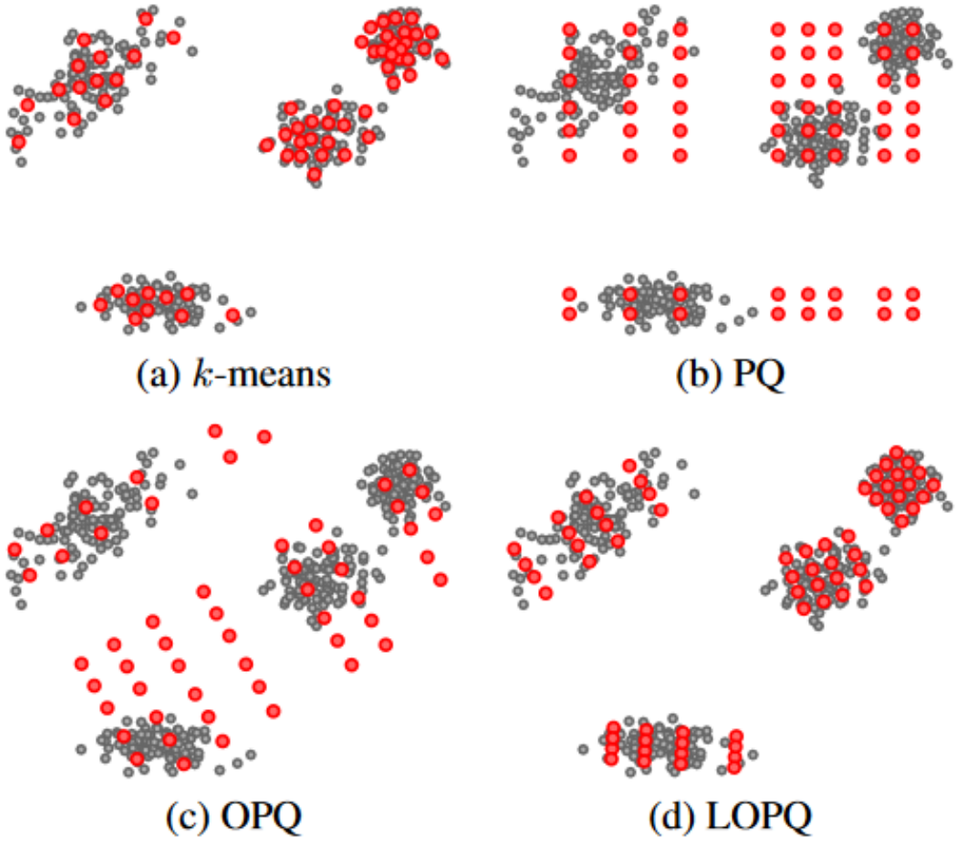
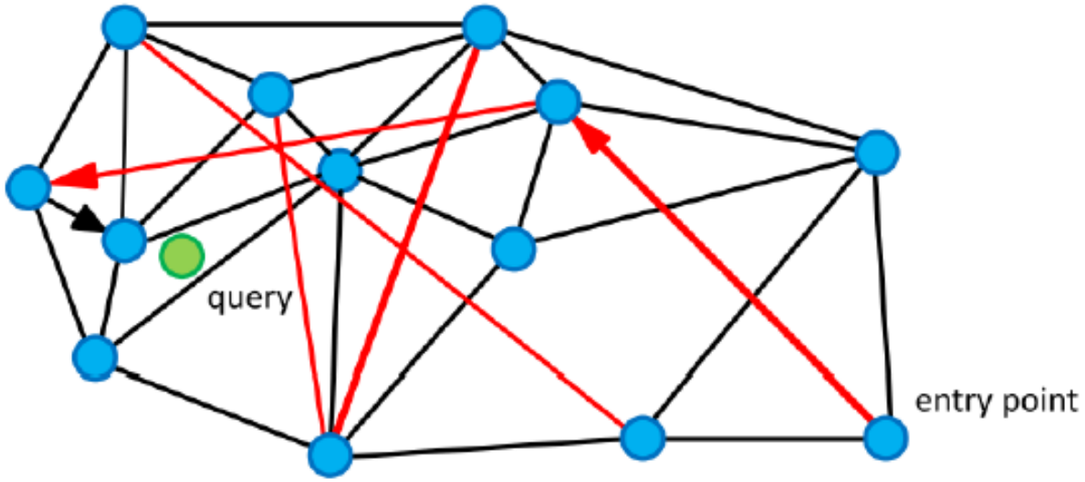


Figure 1. Four quantizers of 64 centroids (•) each, trained on a random set of 2D points (•), following a mixture distribution. (c) and (d) also reorder dimensions, which is not shown in 2D.

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

Эффективность: Navigable Small World (NSW)

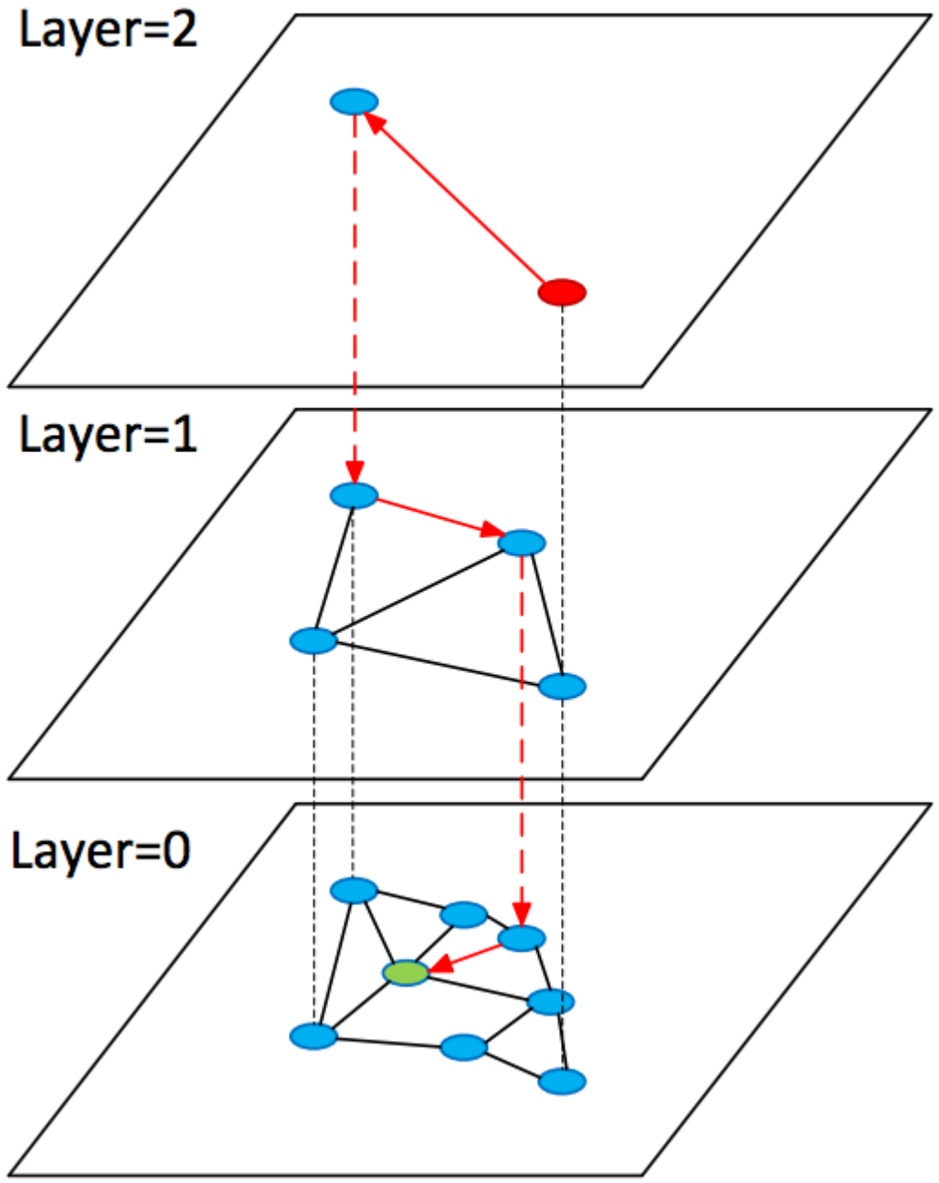
главное – найти быстро, но не факт, что найдём самого ближайшего



Одна из идей «small world graph»:
объект связан с k соседями, плюс t длинных связей
(в оригинале – графа Делоне, т.к. этого достаточно для поиска БС)

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

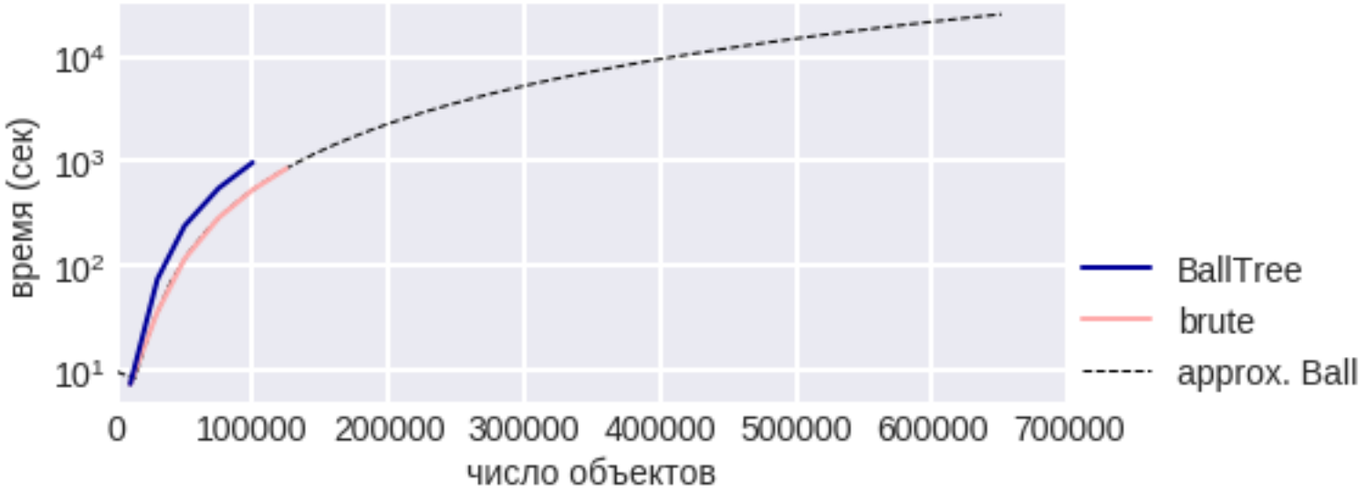
Approximate Nearest Neighbor Search for Sparse Data in Python
https://github.com/facebookresearch/pysparnn

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

Hnswlib – fast approximate nearest neighbor search
https://github.com/nmslib/hnswlib

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 признаковых 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