«Машинное обучение»

Метрические методы

Александр Дьяконов

План

Метрические алгоритмы

Ближайший центроид

Метод k ближайших соседей (kNN)

Весовые обобщения kNN

Регрессия Надарая-Ватсона

Метрики

Приложения метрического подхода

Метрические алгоритмы

«distance-based» – анализируются расстояния

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

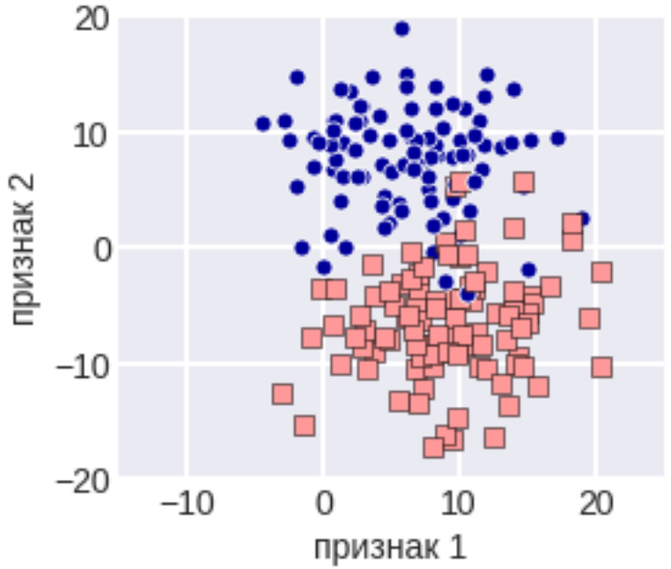
Примеры:

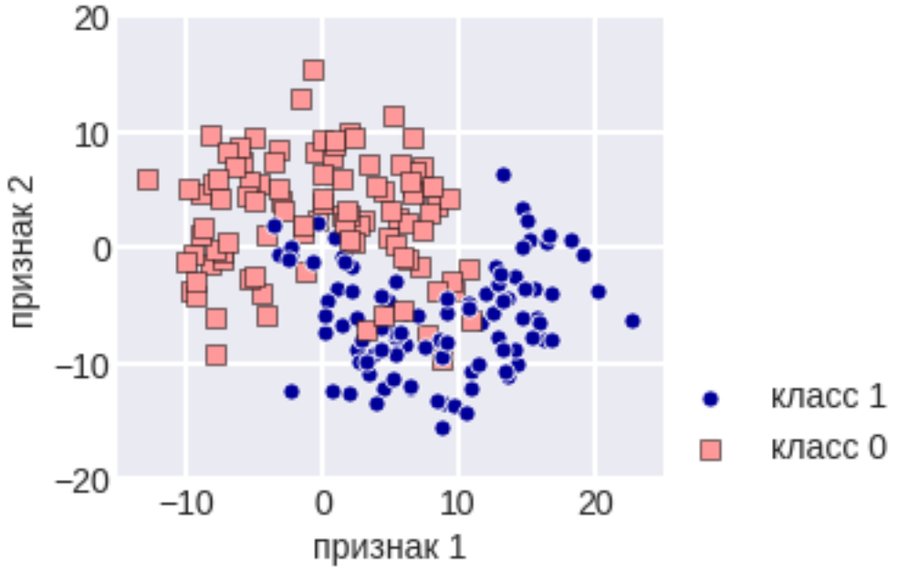
- Nearest centroids algorithm / Distance from Means
- kNN (Nearest Neighbor)

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:

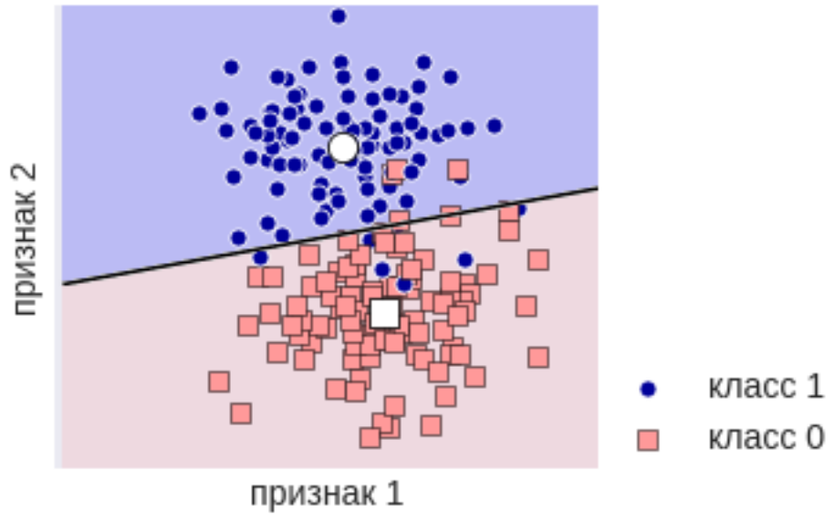
$$c_j = \frac{1}{|\{i : y_i = j\}|} \sum_{i: y_i = j} x_i$$

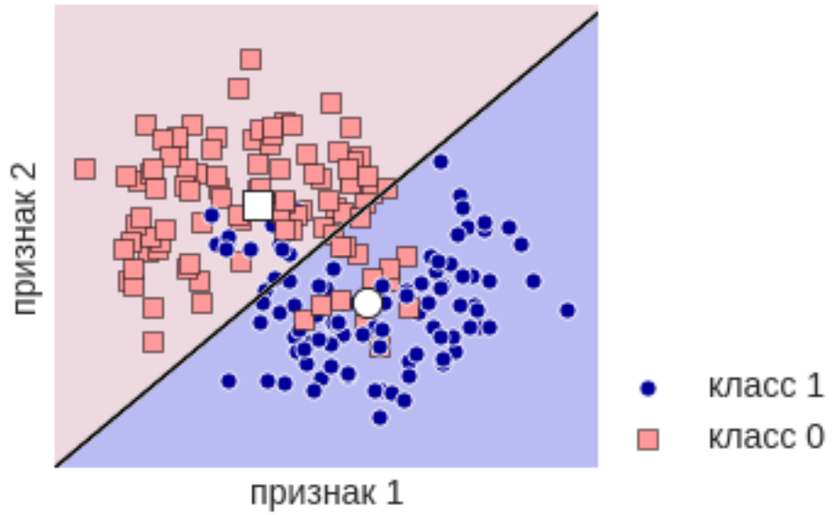
классификация:

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

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»

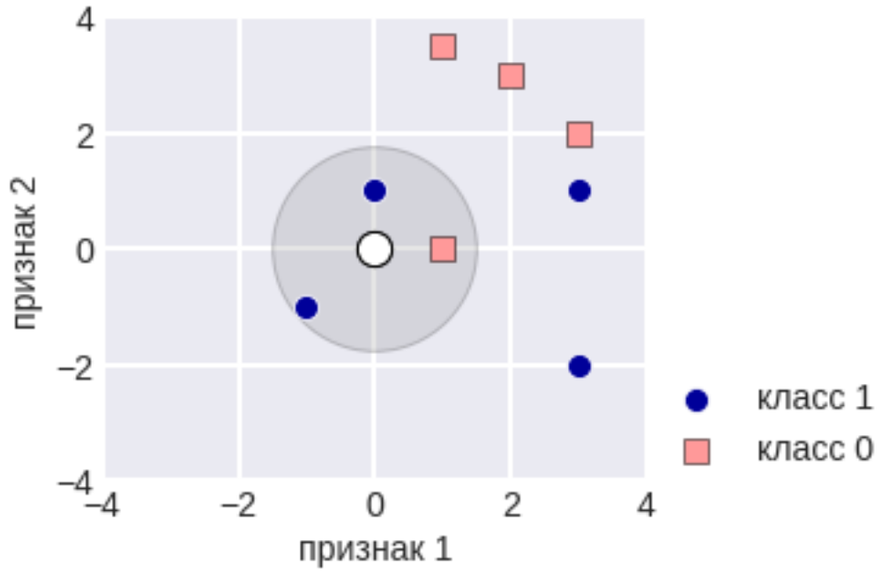
Минутка кода: ближайший центроид (Nearest centroids algorithm)

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

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)

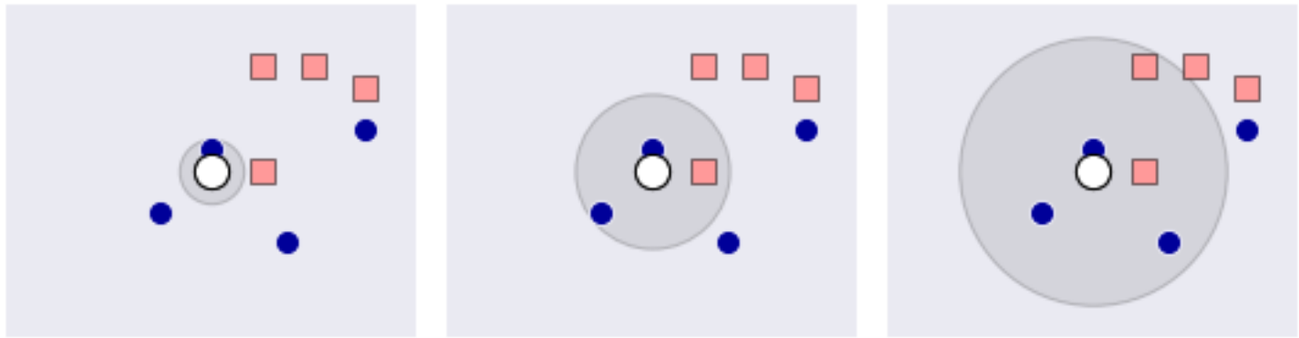
Окрестность

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: метрика (параметры метрики), ядро (+ параметры ядра) и т.п.

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

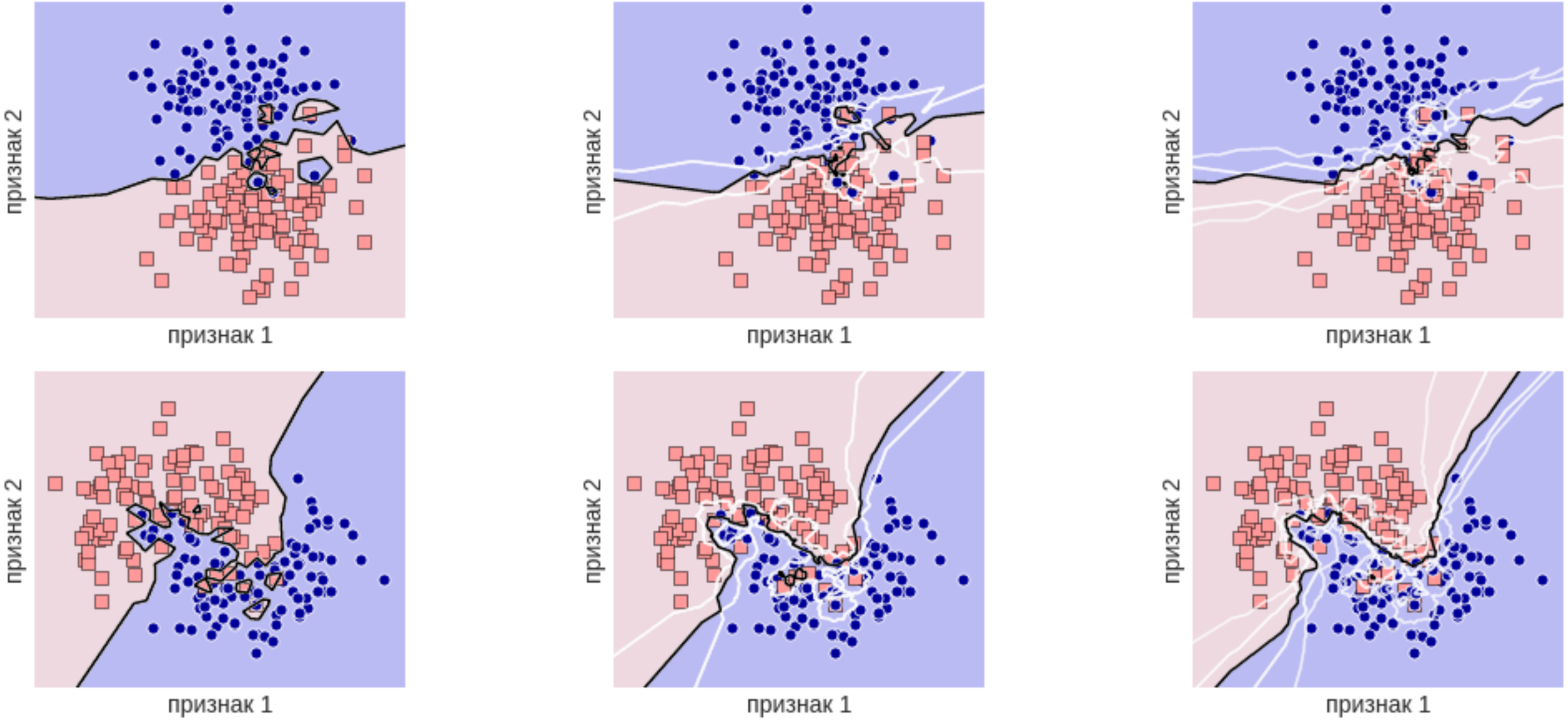
Термины

Нетерпеливый алгоритм (Eager learner)	Обучение – получение значений параметров После обучения данные не нужны
Ленивый алгоритм (Lazy learner)	Не использует обучающую выборку до классификации Формально нет обучения – храним всю выборку

Решение модельной задачи при разном числе соседей

Как увидим дальше, k отвечает за «сложность модели»

k=1 **k=3** **k=5**

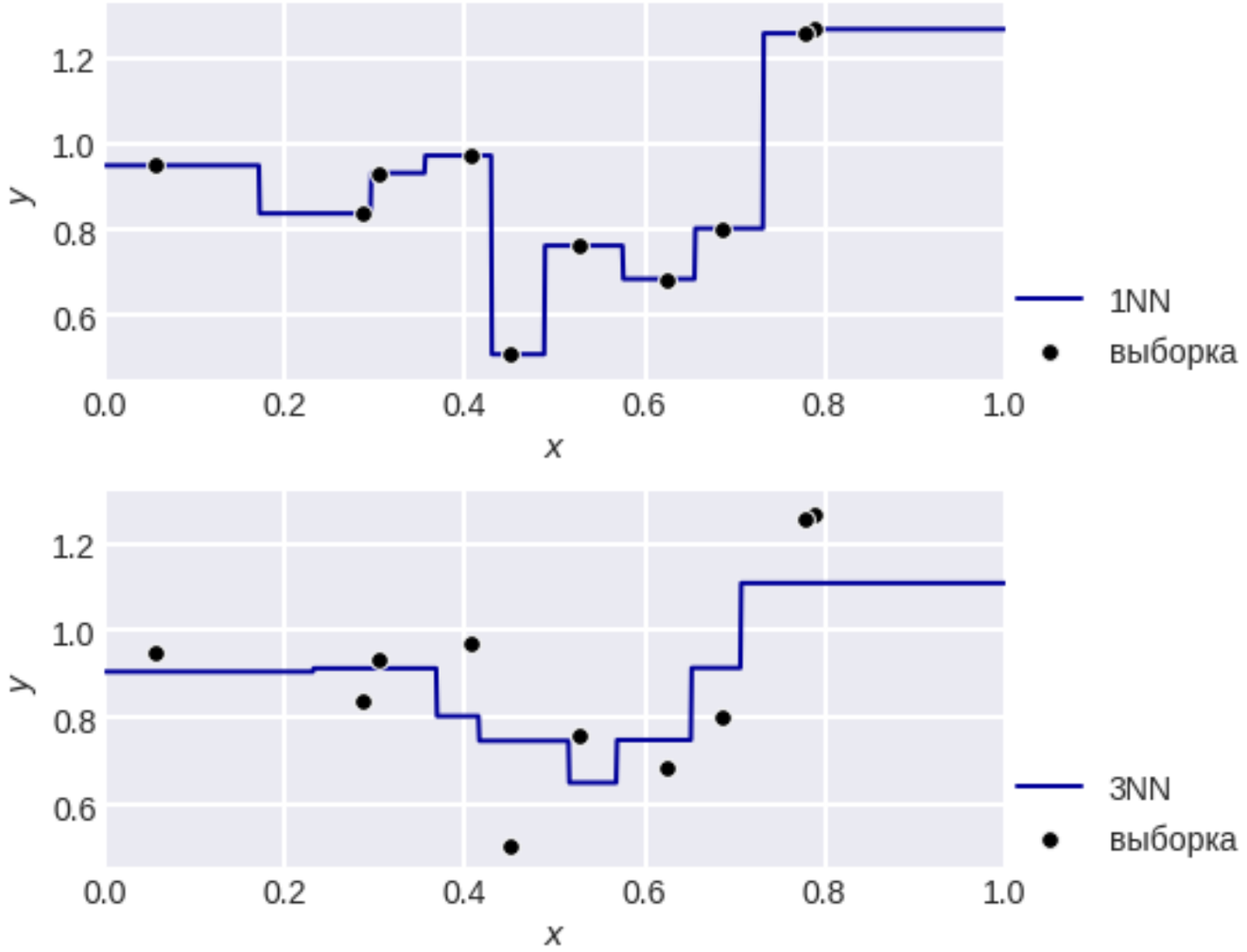


Минутка кода

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=5, # число соседей
    radius=1.0,           # ограничение пространства
    algorithm='auto',     # алгоритм для определения БС ball_tree,
                          # kd_tree, brute
    leaf_size=30,         # параметр для BallTree / KDTree
    weights='uniform',    # веса («distance», функция)
    metric='minkowski',   # метрика (функция или строка),
                          # scipy.spatial.distance
    p=2,                  # параметр для minkowski
    metric_params=None)   # дополнительные параметры для метрики
model.fit(X, y)
a = model.predict(X2)
p = model.predict_proba(X2)[:, 1]
```

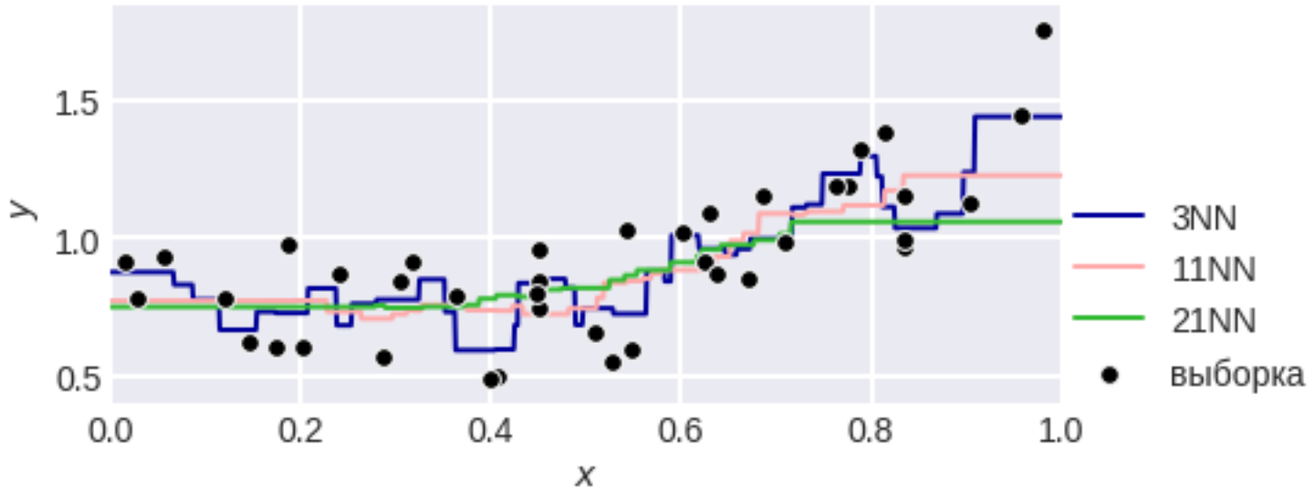
Метод ближайшего соседа в регрессии

обобщается на регрессию



Метод ближайшего соседа в регрессии

обобщается на регрессию



Минутка кода

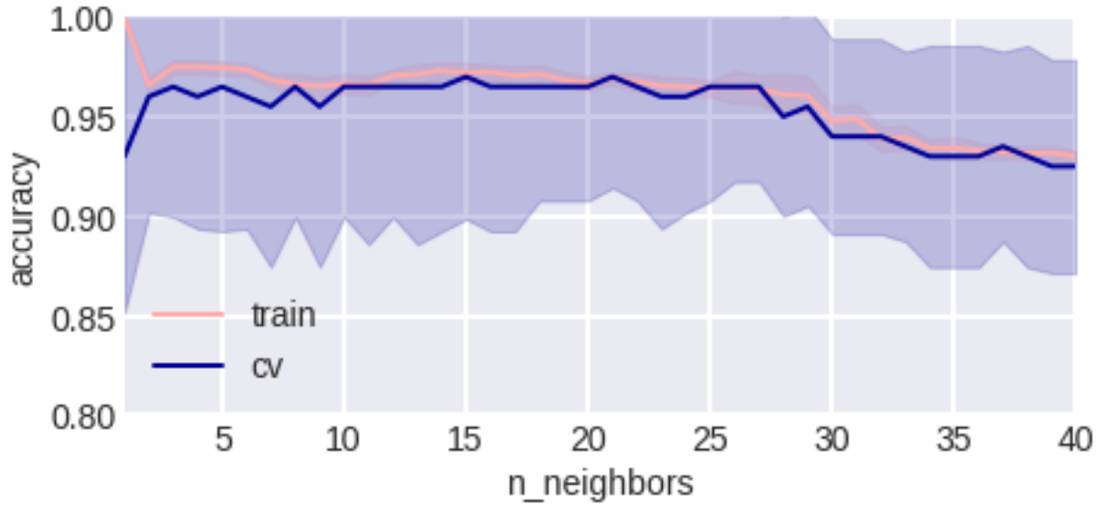
```
from sklearn.neighbors import KNeighborsRegressor
model = KNeighborsRegressor(n_neighbors=3) # kNN-регрессия
model.fit(x_train, y_train) # обучение
a = model.predict(x_test) # ответ
```

Подбор гиперпараметров специальными методами контроля

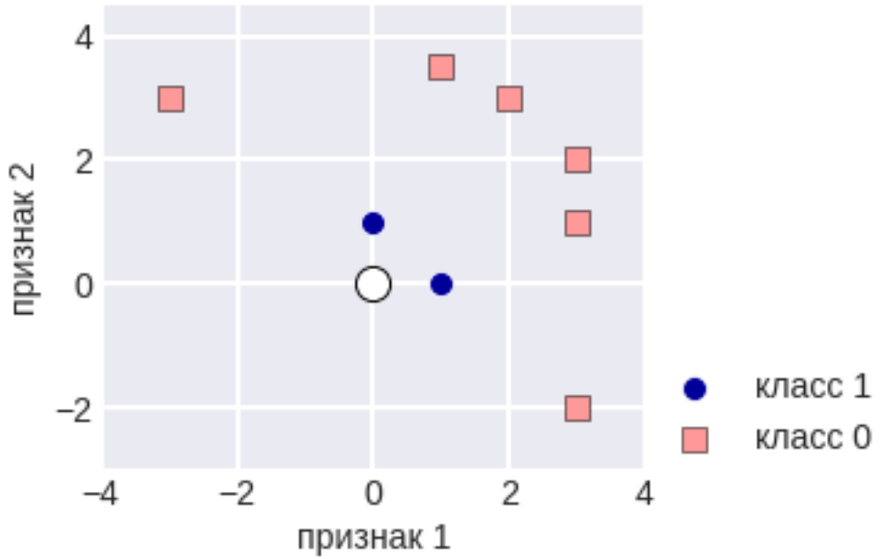
```
# cv-контроль
from  sklearn.model_selection import KFold
cv = KFold(n_splits=10, shuffle=True,
random_state=2)
# модель
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=5)
# параметр
param_name = "n_neighbors"
# его значения
pars = np.arange(1, 41)

# 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)
```



Проблема классического kNN



близкие соседи должны быть важнее

Весовые обобщения kNN

классика:

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

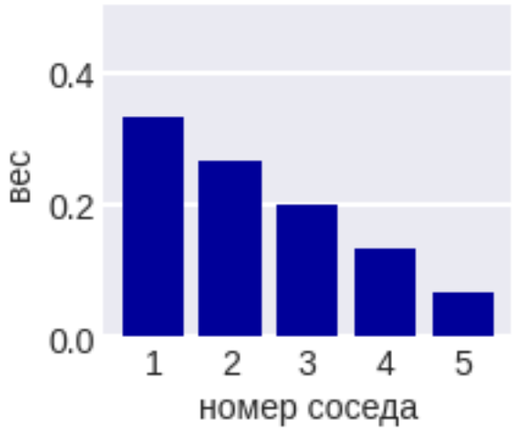
обобщение:

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

разные весовые схемы:

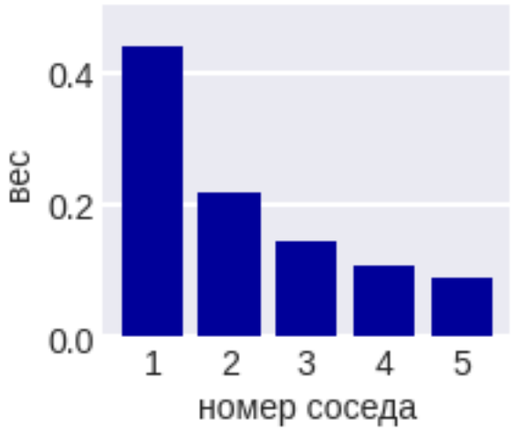
$$w_1 \geq w_2 \geq \ldots \geq w_k > 0$$

Весовые схемы



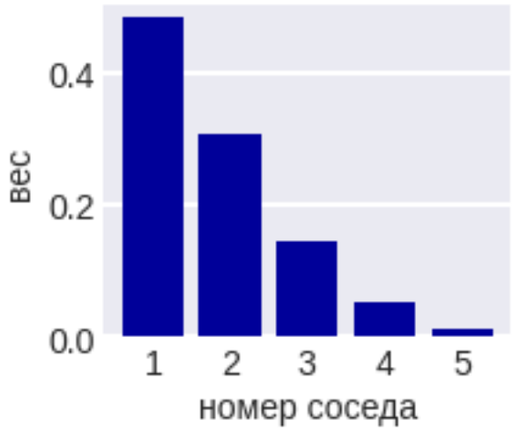
$$w_t = (k - t + 1)^{\delta}$$

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



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

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



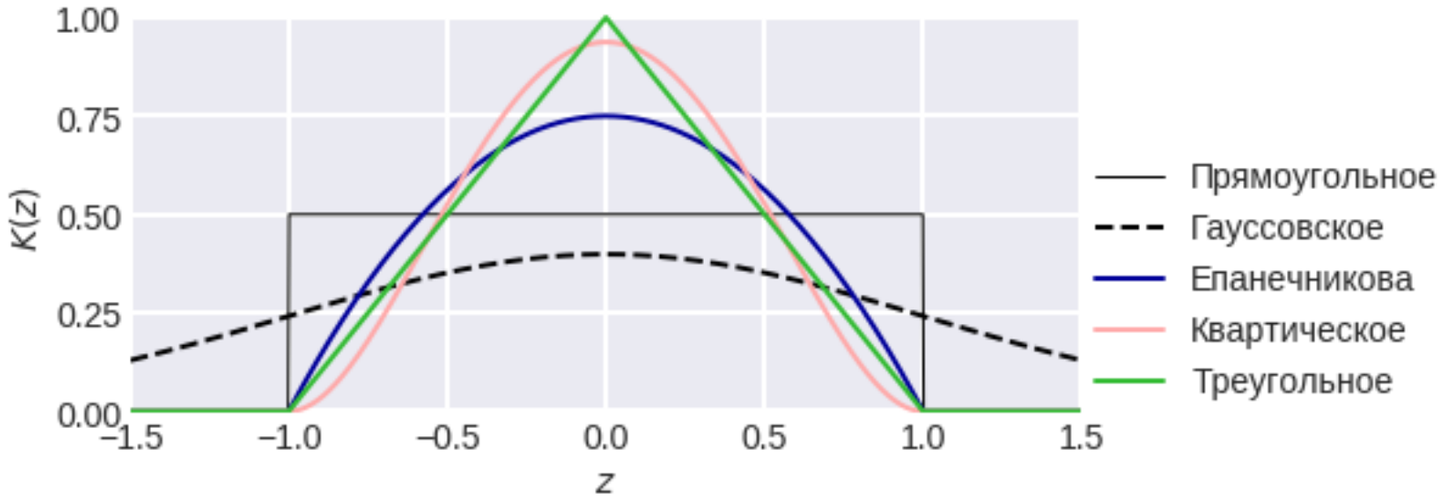
$$w_t = K\left(\frac{\rho(x, x_t)}{h(x)}\right)$$

часто веса лучше отнормировать, чтобы сумма = 1

главное преимущество –

богатое пространство вероятности в задачах классификации!

Что такое ядро



Прямоугольное / tophat

$$K(z) = \frac{1}{2} I[|z| \le 1]$$

Треугольное / linear

$$K(z) = \max(\min(1-z, 1+z), 0)$$

Гауссовское / gaussian

$$K(z) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z^T z}{2}\right)$$

Квартическое

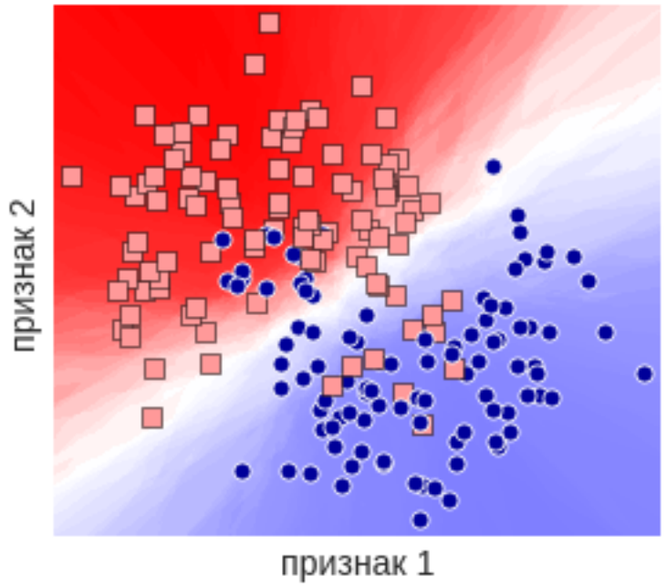
$$K(z) = \frac{15}{16}(1-z^2)^2 I[|z| \le 1]$$

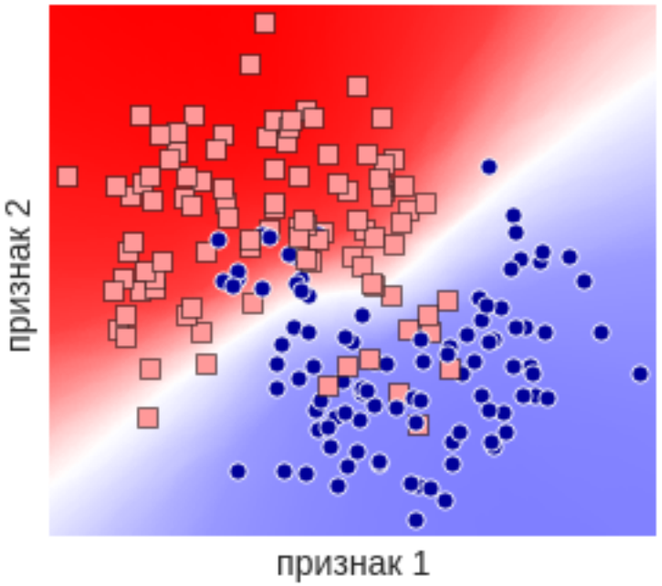
Епанечникова / epanechnikov

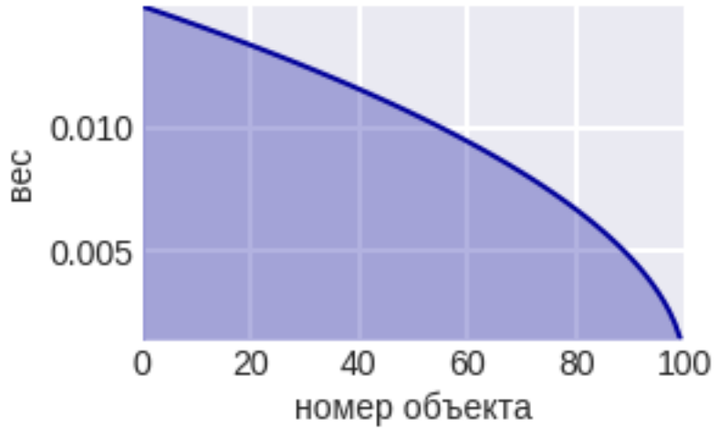
$$K(z) = \frac{3}{4}(1-z^2) I[|z| \le 1]$$

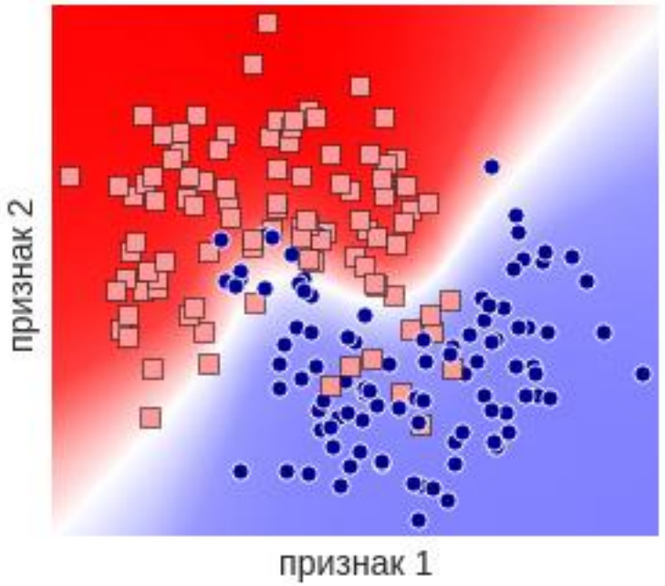
https://scikit-learn.org/stable/auto_examples/neighbors/plot_kde_1d.html

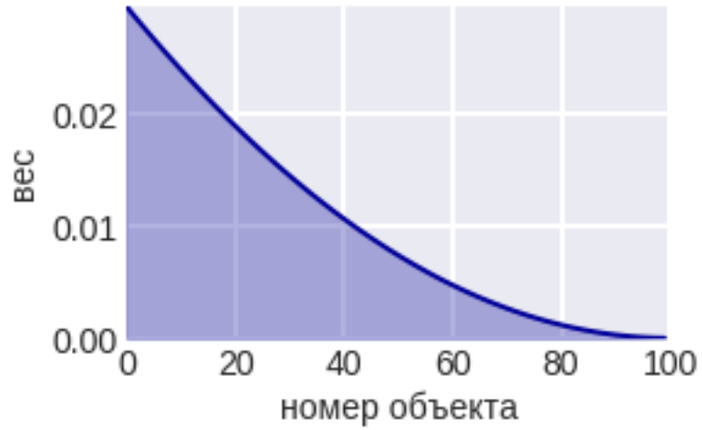
Весовые обобщения kNN



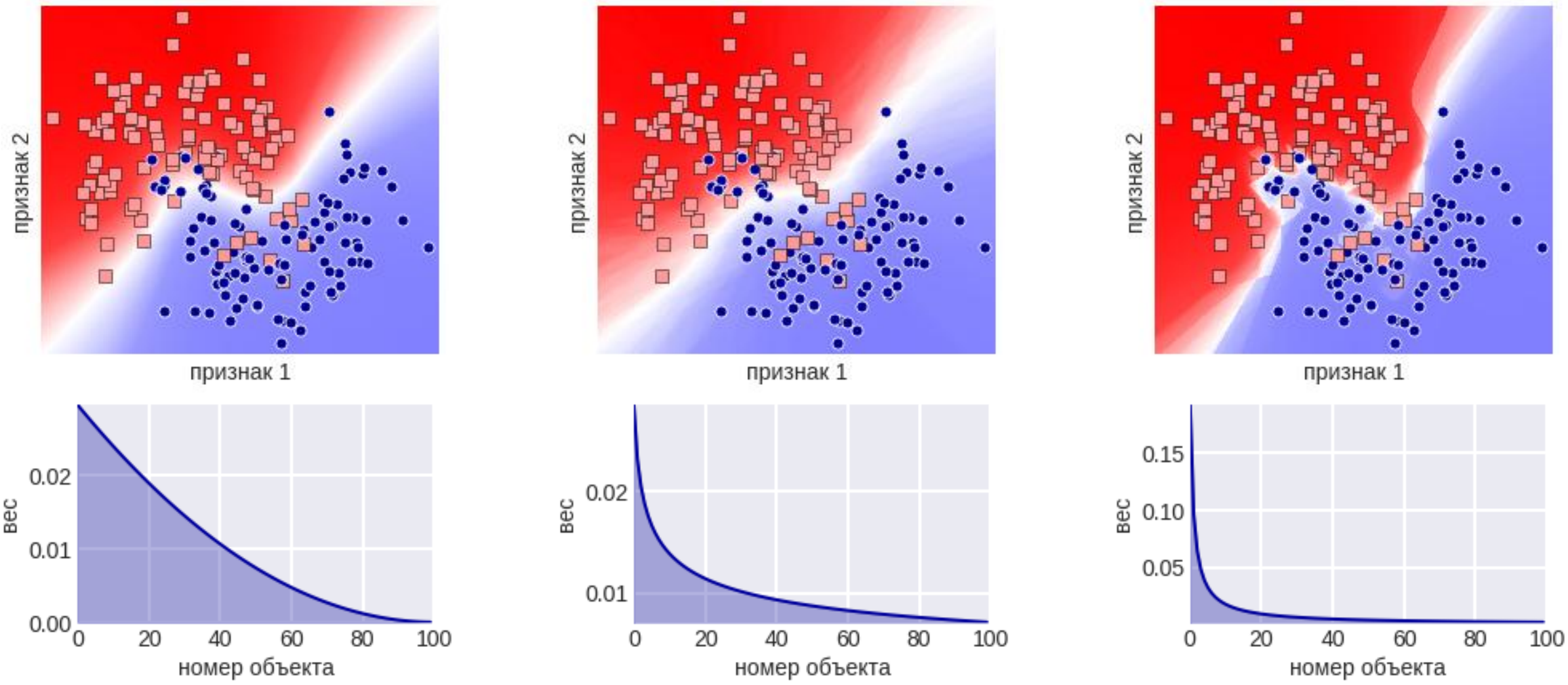








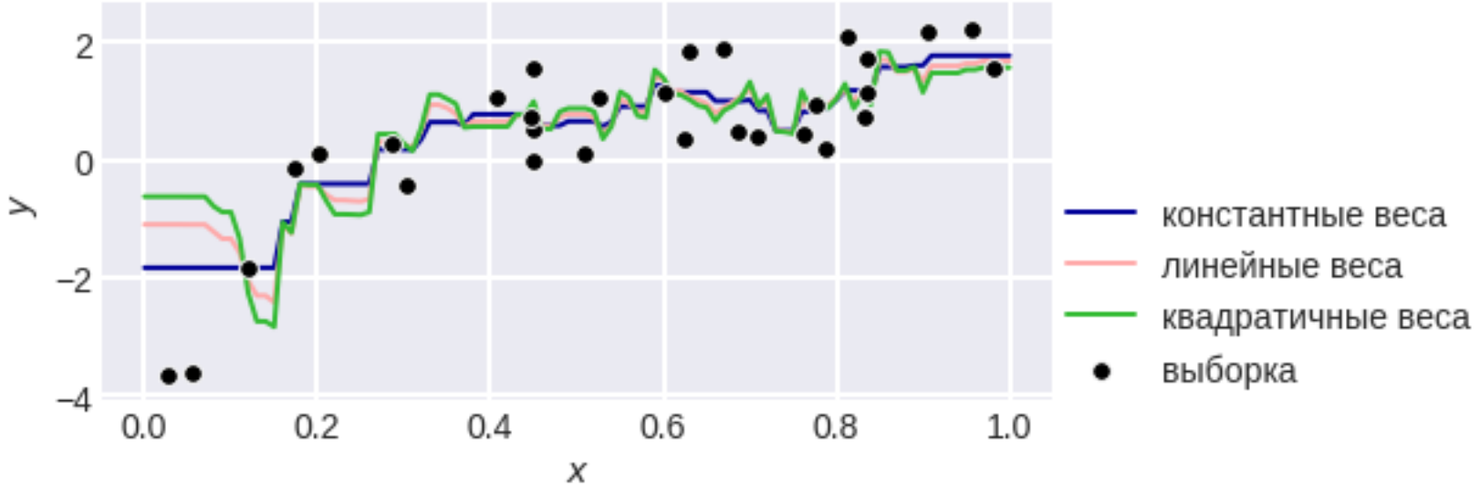
Весовые обобщения kNN



Весовые обобщения в регрессии

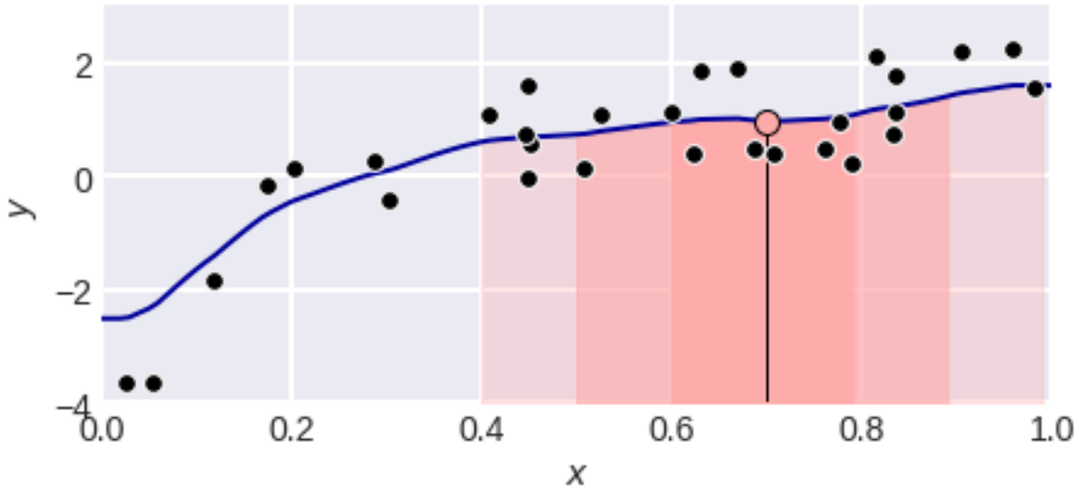
$$\frac{\sum_{t=1}^{k} w_t y(x_t)}{\sum_{t=1}^{k} w_t}$$

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

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

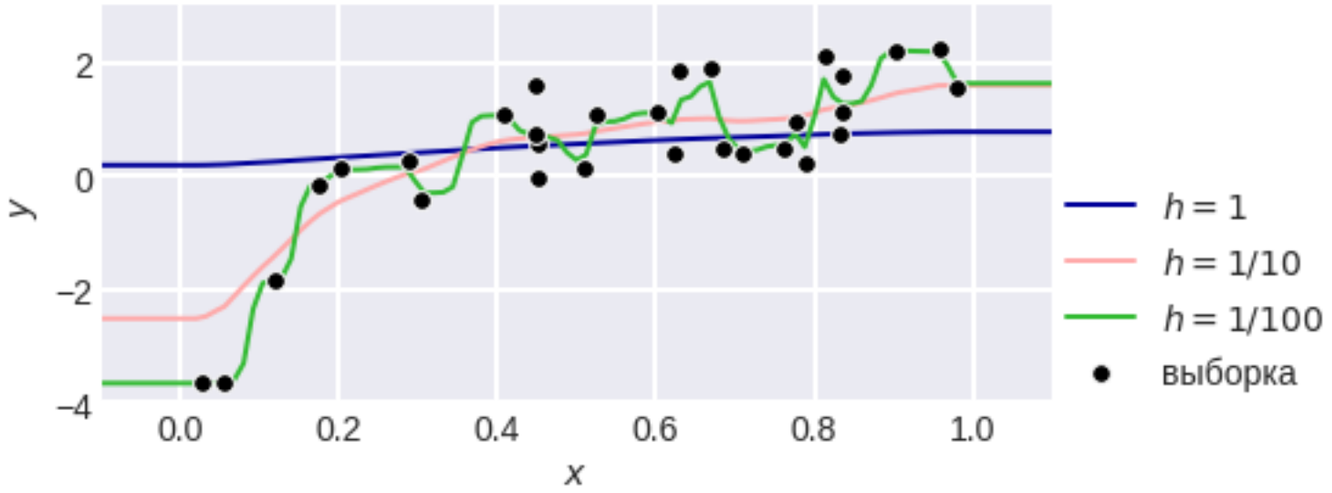
Смысл весов – чем ближе объект обучения, тем скорее ответ похож на его метку

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

Ядро с шириной h.

Здесь также как выше… (про функции ядра)

Регрессия Надарая-Ватсона



пример регрессии при разных значениях ширины ядра

Регрессия Надарая-Ватсона

Смысл:

ответ алгоритма – решение оптимизационной задачи

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

Свойства:

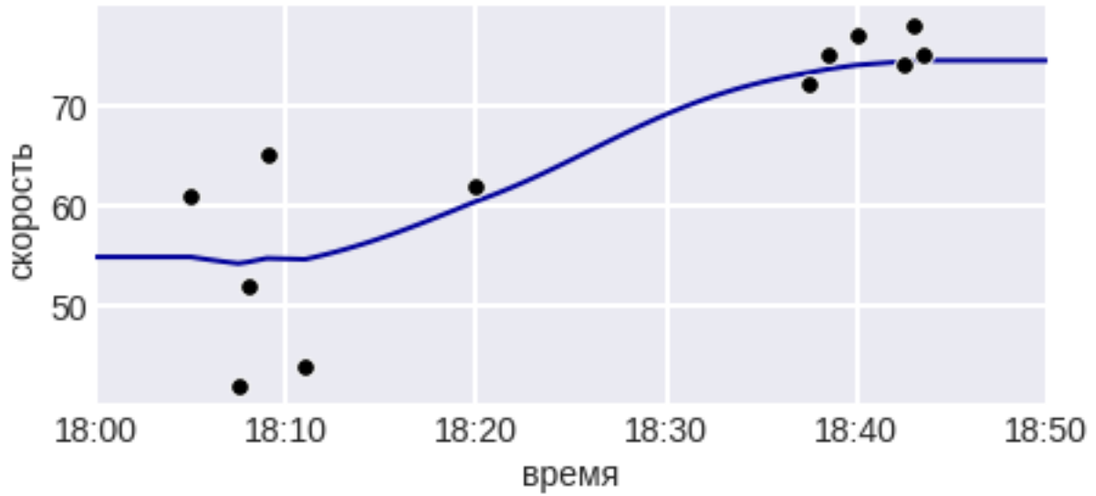
+ хорошее решение задачи сглаживания

– 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_∞)

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

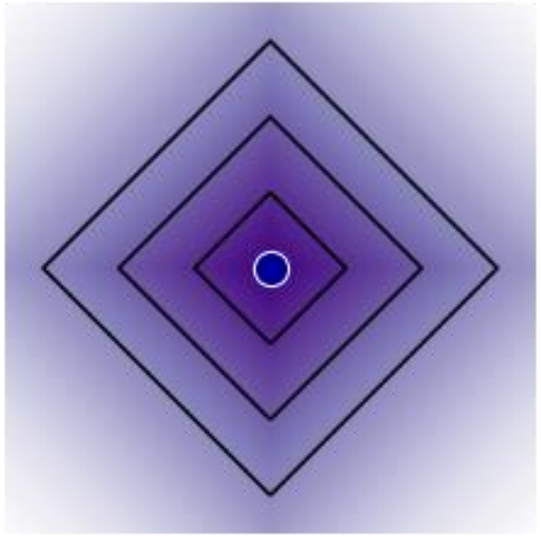
Частный случай – Манхэттенская (L_1)

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

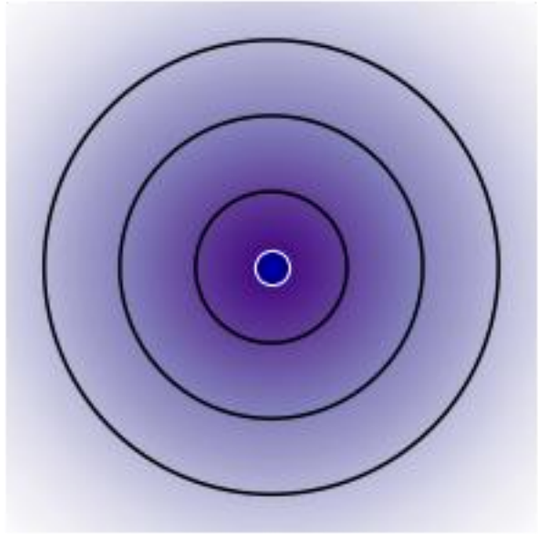
здесь $x = (x_1, \ldots, x_n), \quad z = (z_1, \ldots, z_n)$

Различные метрики

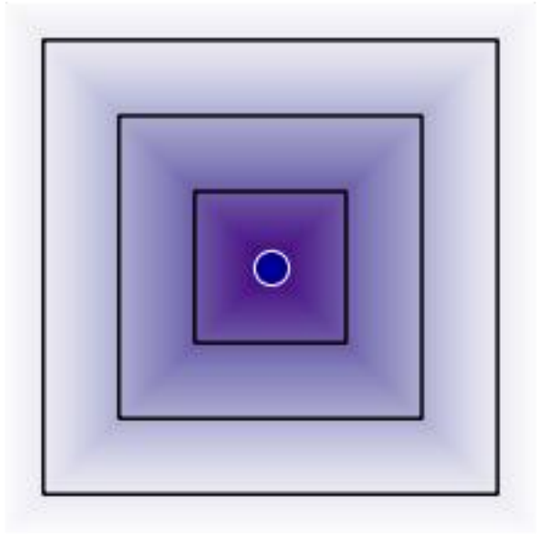
$$\left(| x_1 - z_1 |^p + | x_2 - z_2 |^p \right)^{1/p}$$



L_1

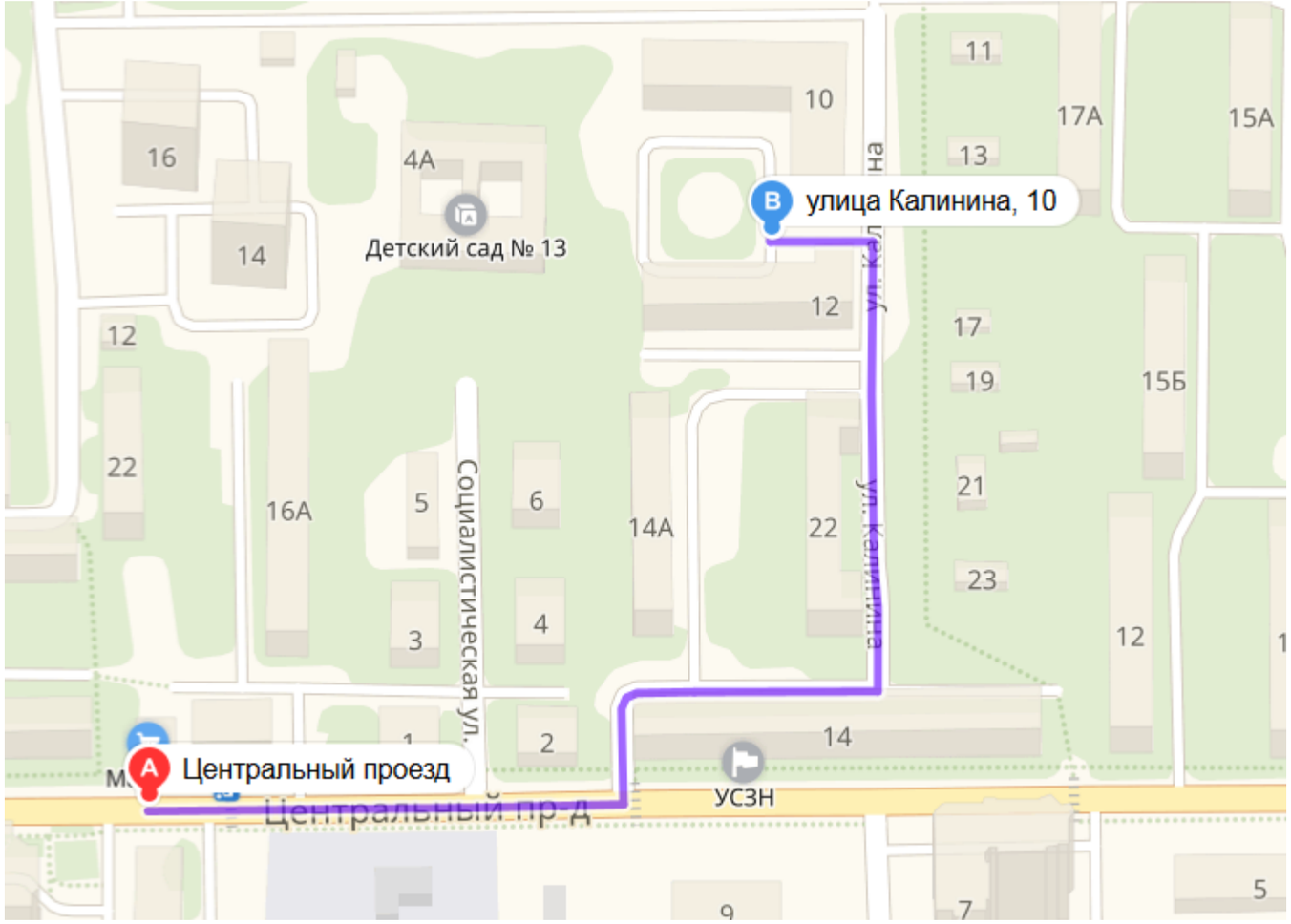


L_2



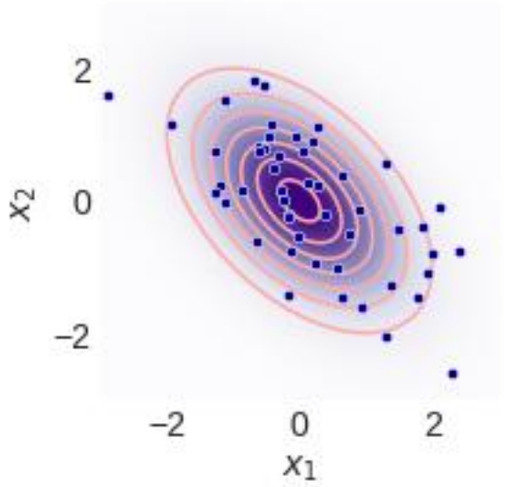
L_∞

Различные метрики



Расстояние Махаланобиса (Mahalanobis distance)

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

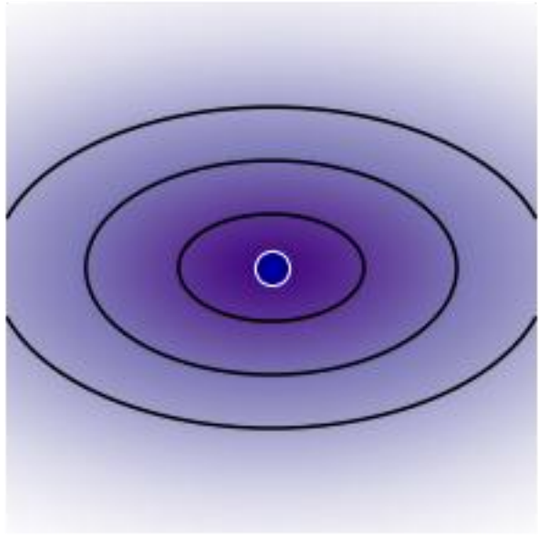


стандартизует нормальные данные

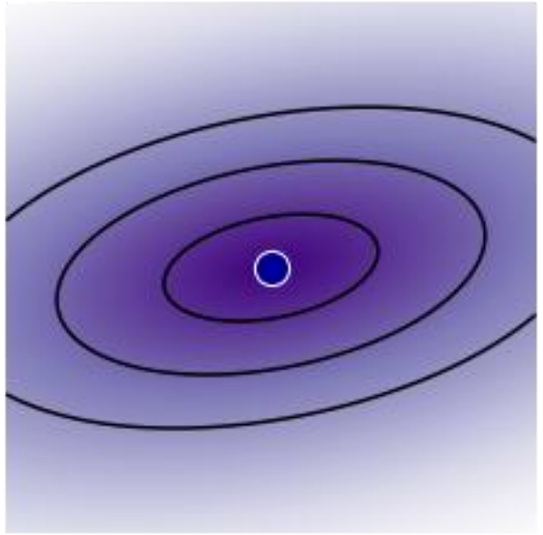
$$\text{norm}(\mu, \Sigma) \to \text{norm}(0, I)$$

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

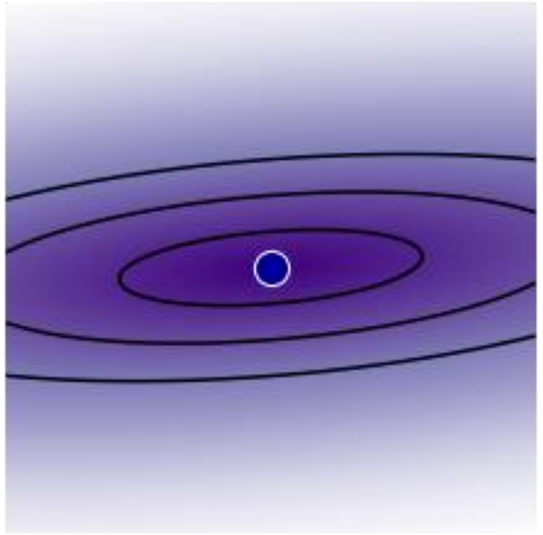
Расстояние Махаланобиса



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



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



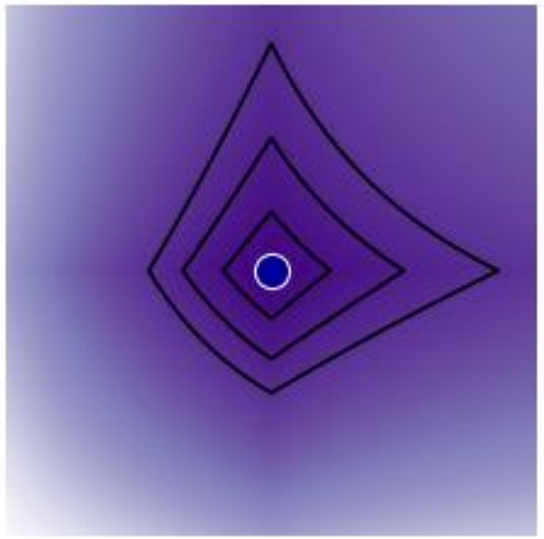
$$\Sigma = \begin{bmatrix} 4 & 0.3 \\ 0.3 & 0.25 \end{bmatrix}$$

Различные метрики

Canberra distance

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

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



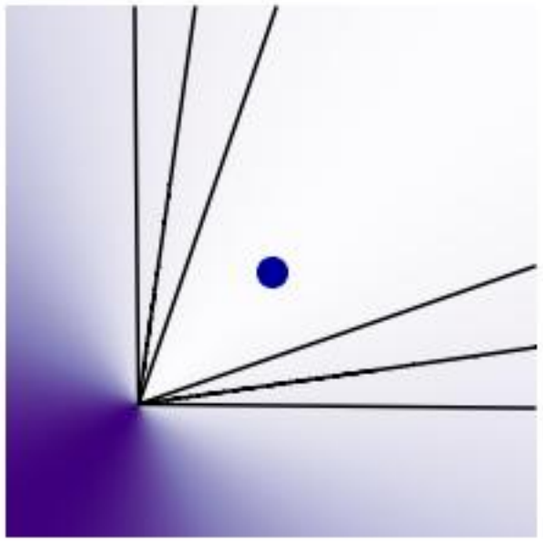
Хэмминга

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

Функции сходства

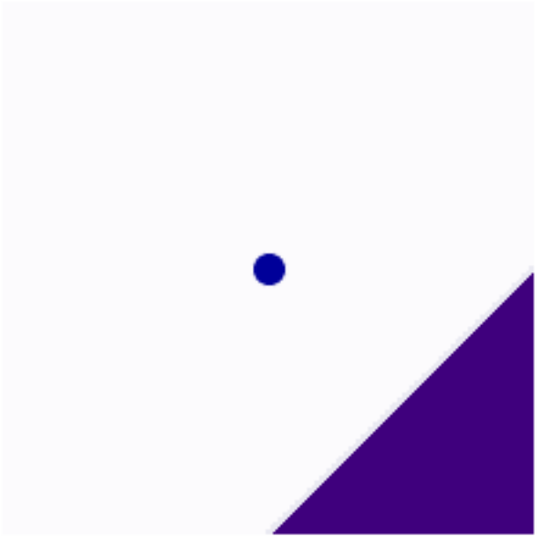
Косинусная мера сходство (не расстояние)

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



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

Различные метрики

Расстояние дЖаккарда (на множествах)

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

тут есть много разных вариантов…

Проблема выбора метрики

- **зависимость от масштаба**

  нормировка признаков

  однородные признаки

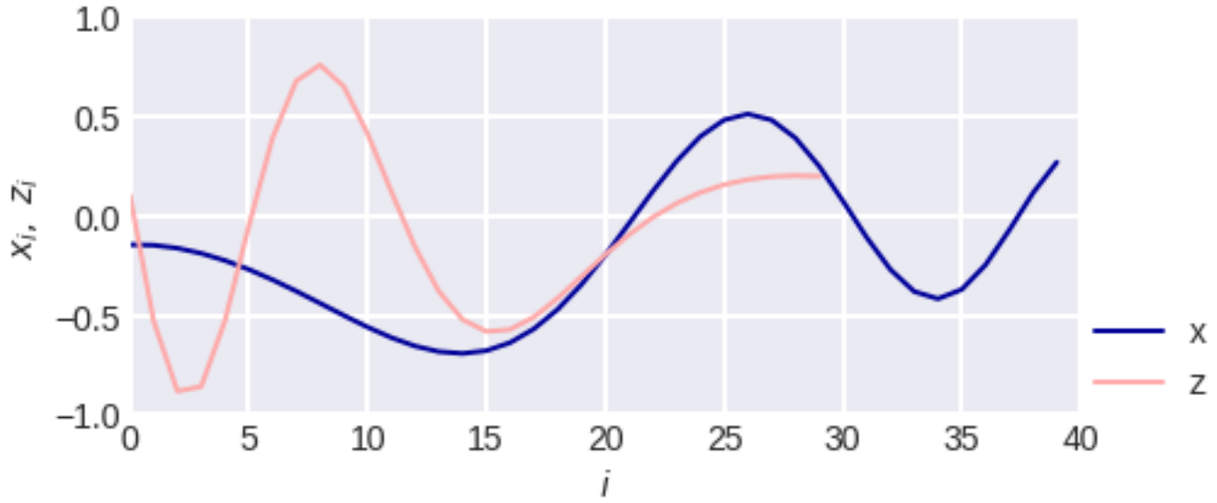
  смесь метрик

- **можно выбирать не метрику, а близость**

  пример: косинусная мера сходства

- **часто выбор функции расстояния, как ни странно, довольно прост…**

Метрики на временных рядах



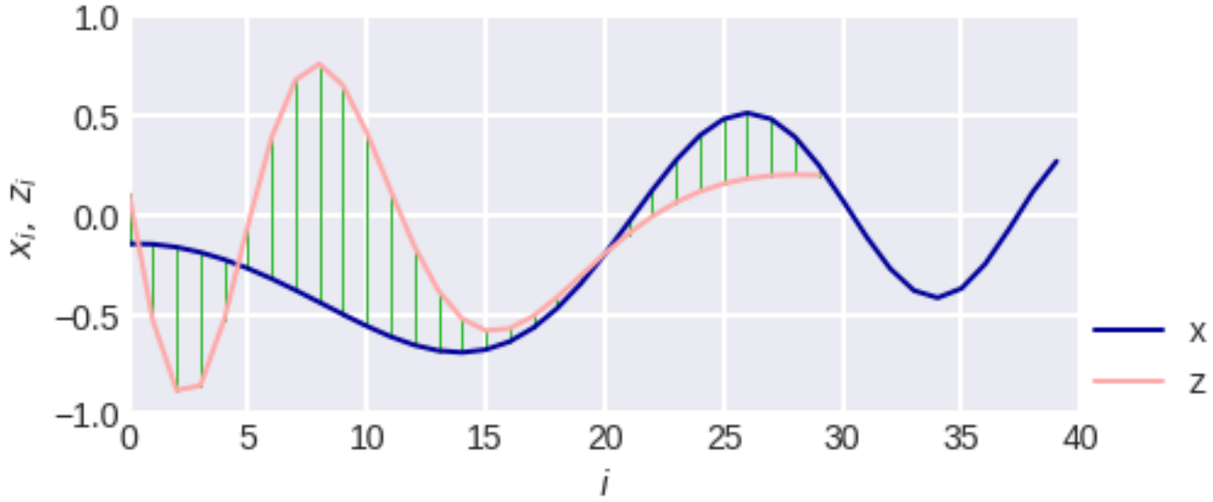
как ввести расстояние?

Ряды могут быть разной длины…

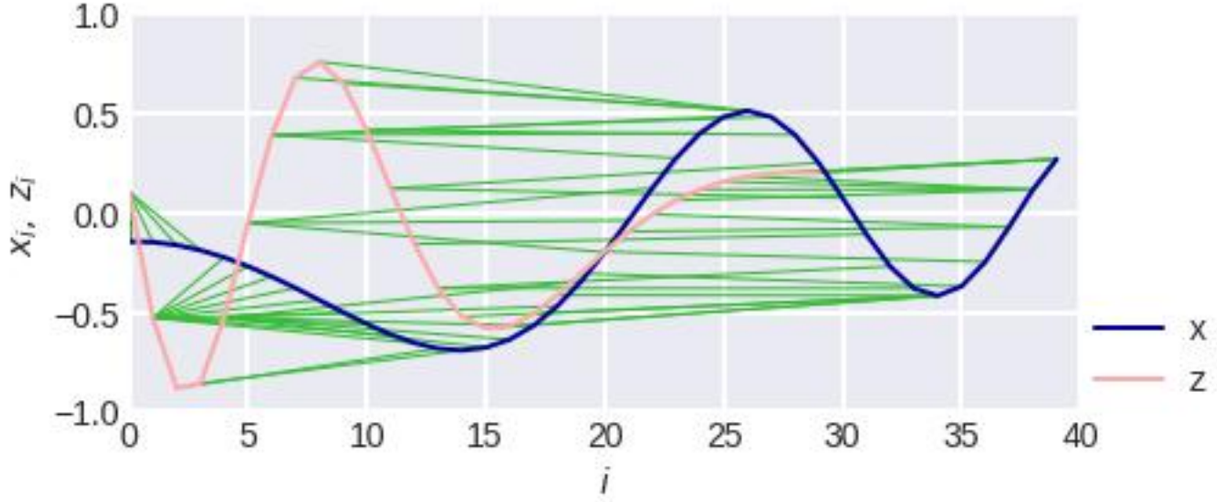
Как оценить похожесть формы?

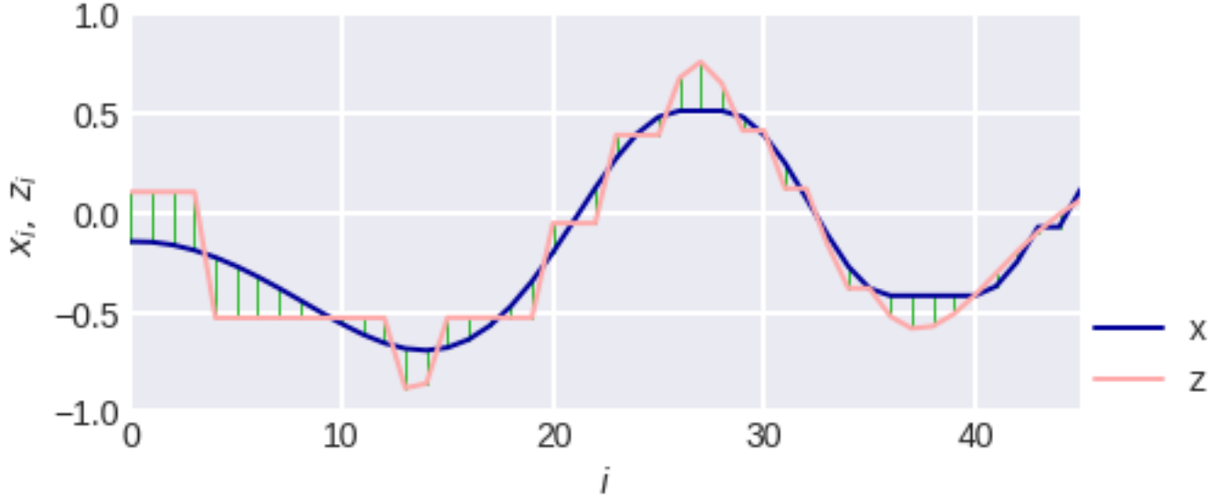
Метрики на временных рядах

Евклидово расстояние



DTW = Dynamic time warping





Расстояние Левенштейна

```
затравка2
поправка2 (exchange=3)
поправка_22 (insert=2)
поправка 22 (dist=5)
```

«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

Приложения метрического подхода: Ленкор

«VideoLectures.Net Recommender System Challenge»
(ECML/PKDD Discovery Challenge 2011)
– написать рекомендательную систему в режиме холодного старта

Описание лекции

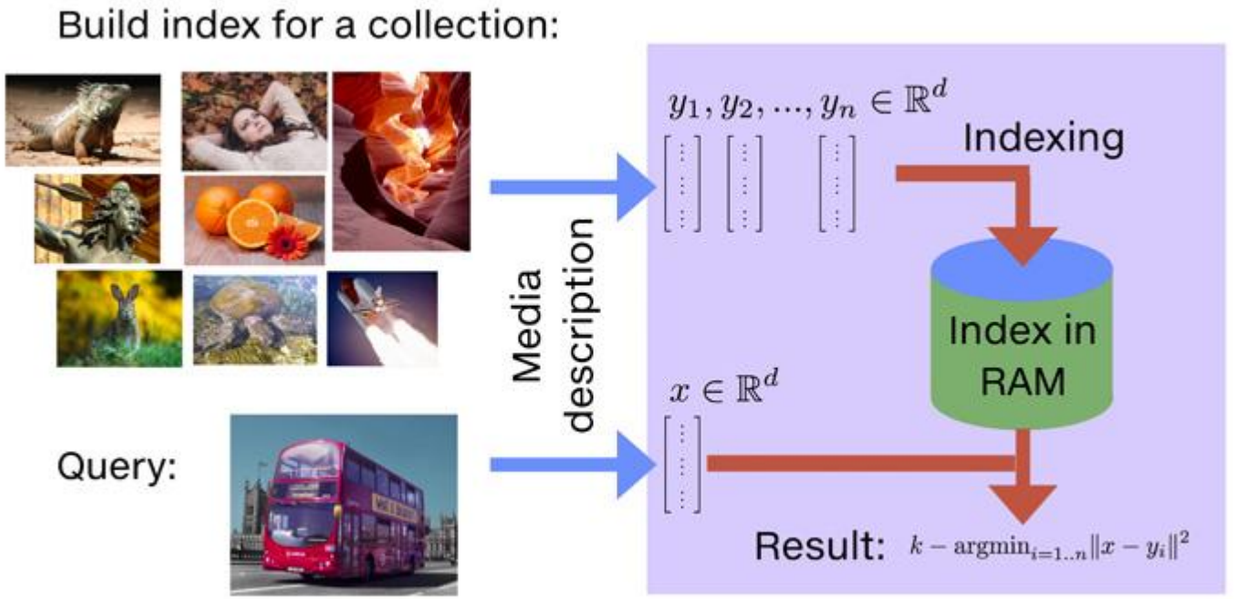
101, 'Lecture', 'eng', 'biology', '2008-12-04', '2009-02-12', 'Implementing a common framework on business', 'Professor Rudolf Smith', …

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

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.

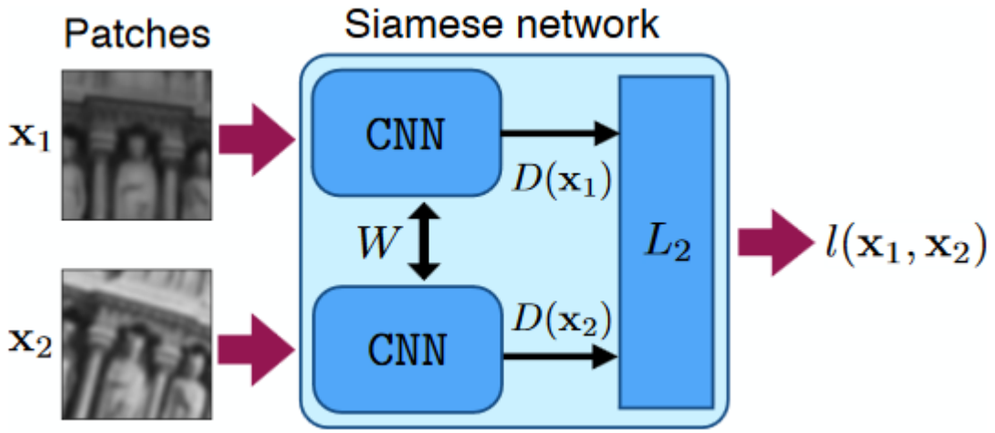
Приложения метрического подхода: поиск похожих объектов



звука, изображения, видео, …

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

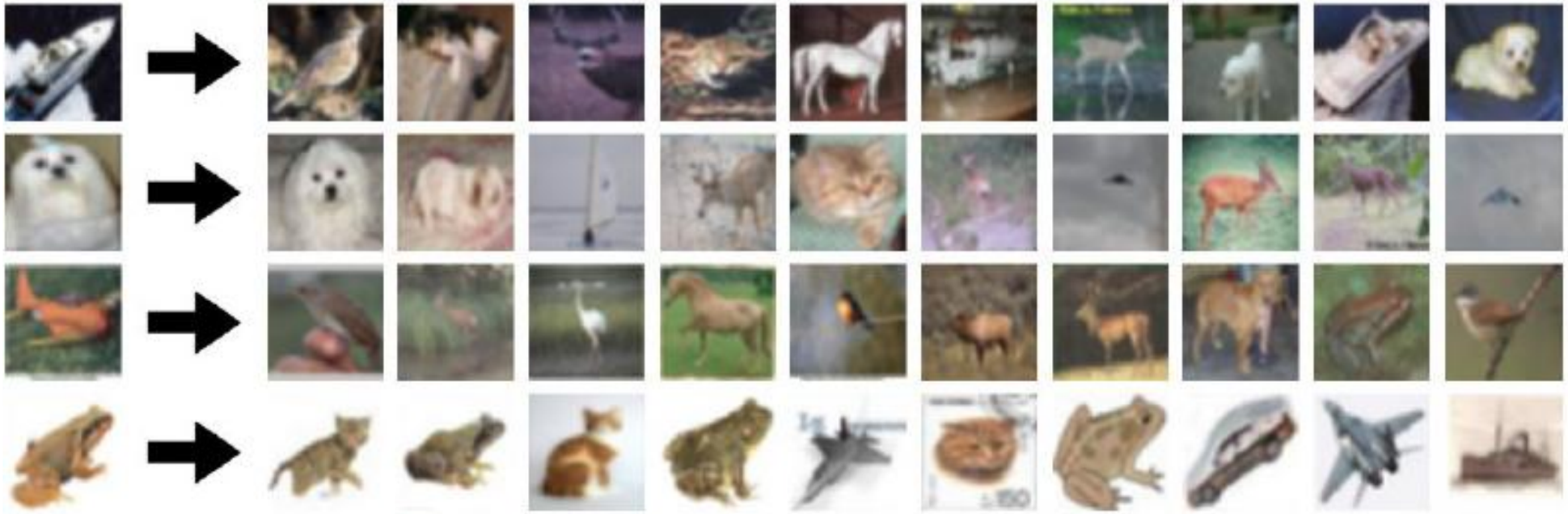
Приложения метрического подхода: Сиамские сети



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: простота интерпретаций

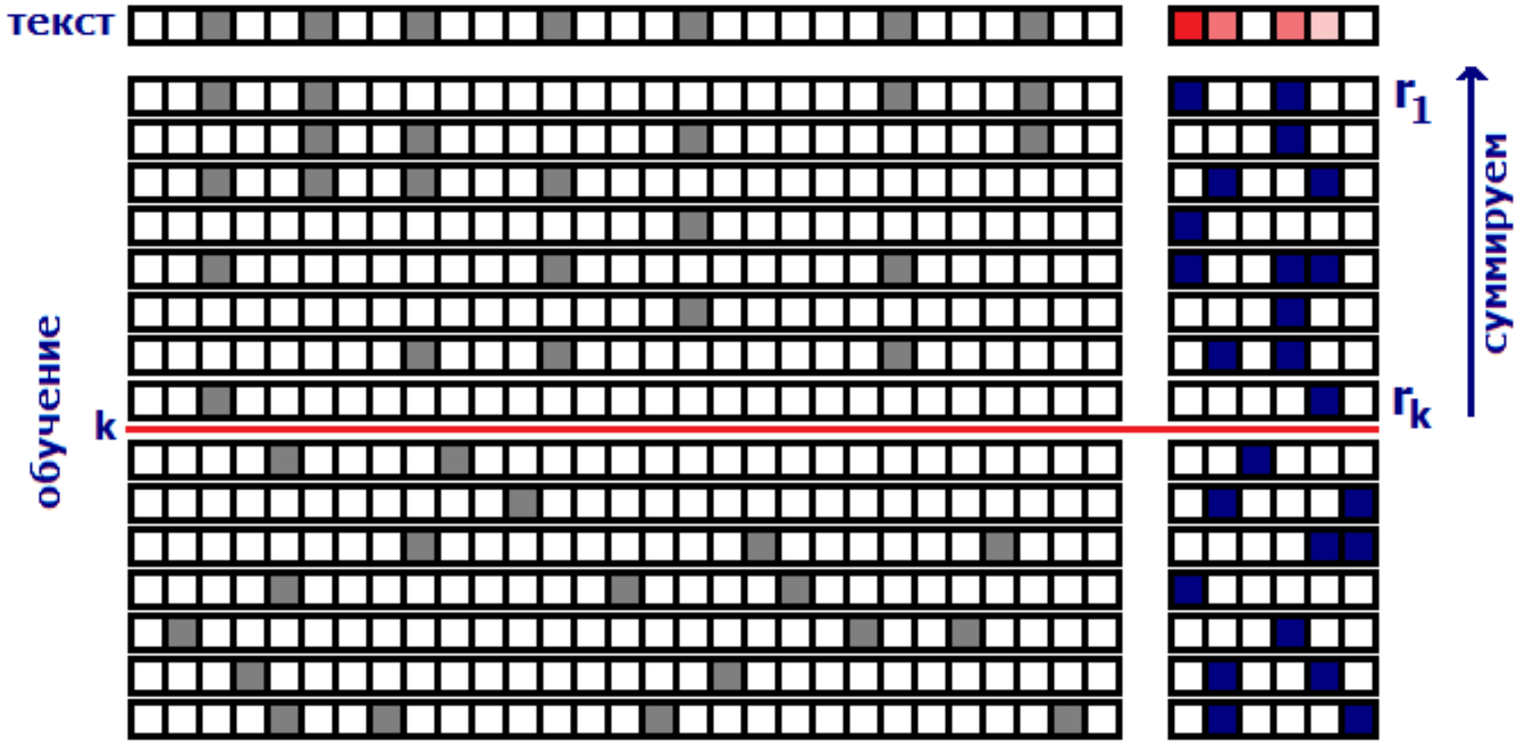


http://cs231n.stanford.edu/2017/syllabus.html

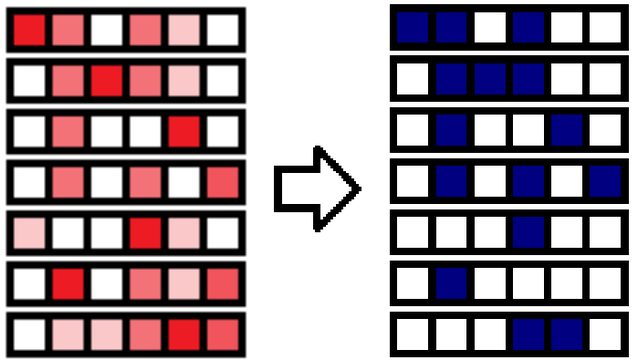
Приложения метрического подхода: классификация текстов

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

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

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

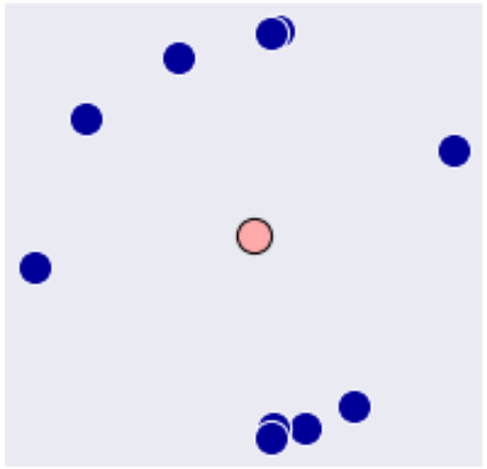
$$A(x_1, \ldots, x_l) = (a_1 x_1 + a_2, \ldots, a_1 x_l + a_2)$$

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

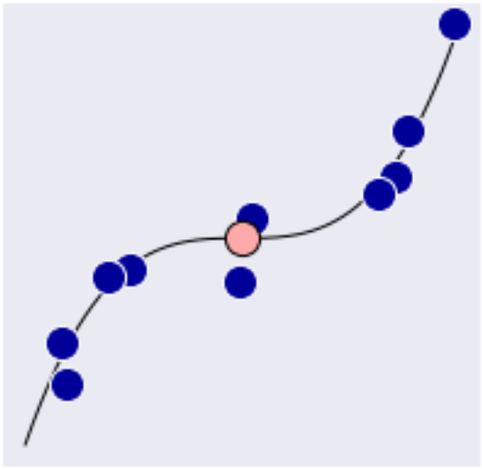
http://www.neural-forecasting-competition.com/NN5/results.htm

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)

Ещё интересные функции

`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

Метрические алгоритмы: итог

+ не требуется признаковых описаний
(достаточно уметь измерять расстояния / близости)
+ легкая реализация
+ интерпретируемость
+ нет обучения
+ мало гиперпараметров (хотя… есть метрика)
+ можно учитывать контекст (с помощью метрики)

– медленная классификация (если kNN, зависит от объёма обучения)
– требуется хранение всей обучающей выборки
– требуется подбор метрики (нормировки признаков)

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

Итог

весовые схемы

очень мощное оружие

метрики

много… есть специализированные

могут быть параметризованы

Кстати, есть эффективные способы обнаружения соседства

Ссылки

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