Машинное обучение и анализ данных

Метрические методы

Дьяконов А.Г.

Московский государственный университет
имени М.В. Ломоносова (Москва, Россия)

Метрические алгоритмы

«distance-based» – анализируются расстояния

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

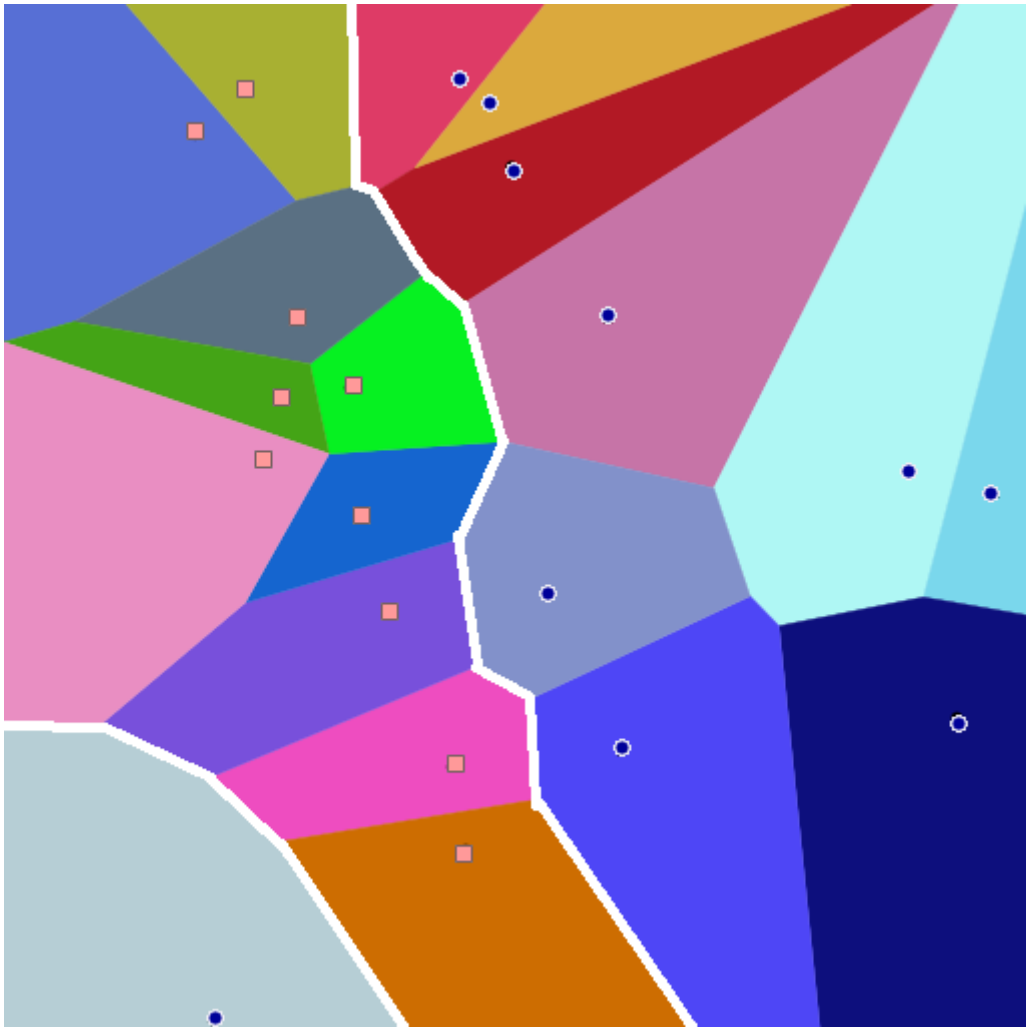
- **Distance from Means**
- **kNN (Nearest Neighbor)**

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

называют:

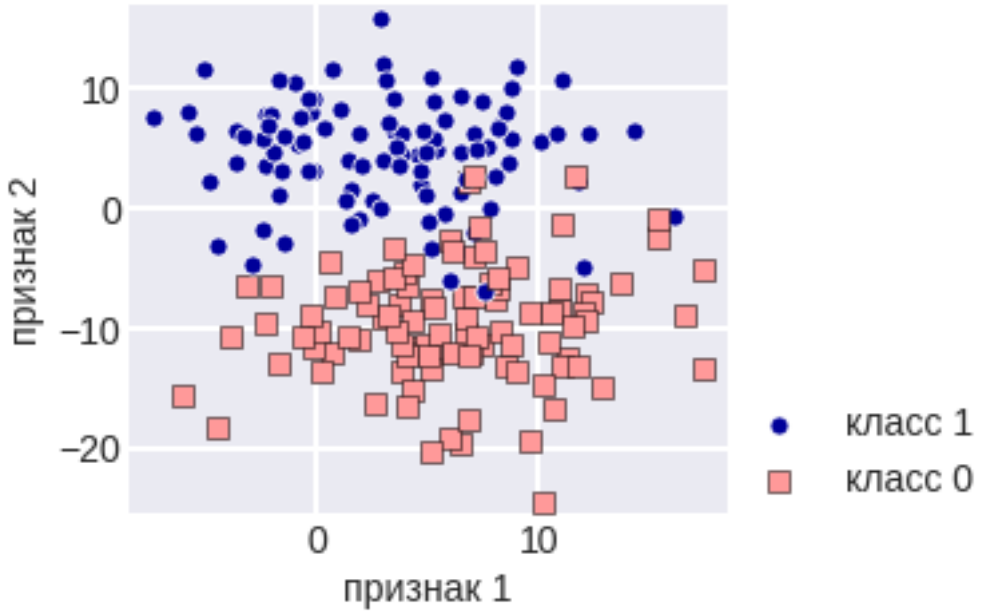
- **«memory-based»**
- **«instance-based»**
- **«non-parametric»**

Диаграмма Вороного (Voronoi diagram)



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

Модельная задача классификации



Ближайший центроид (Nearest centroid algorithm)

Задача классификации на непересекающиеся классы с вещественными признаками:

$$Y = \{1, 2, \ldots, l\}$$

$$x_i \in \mathbb{R}^n$$

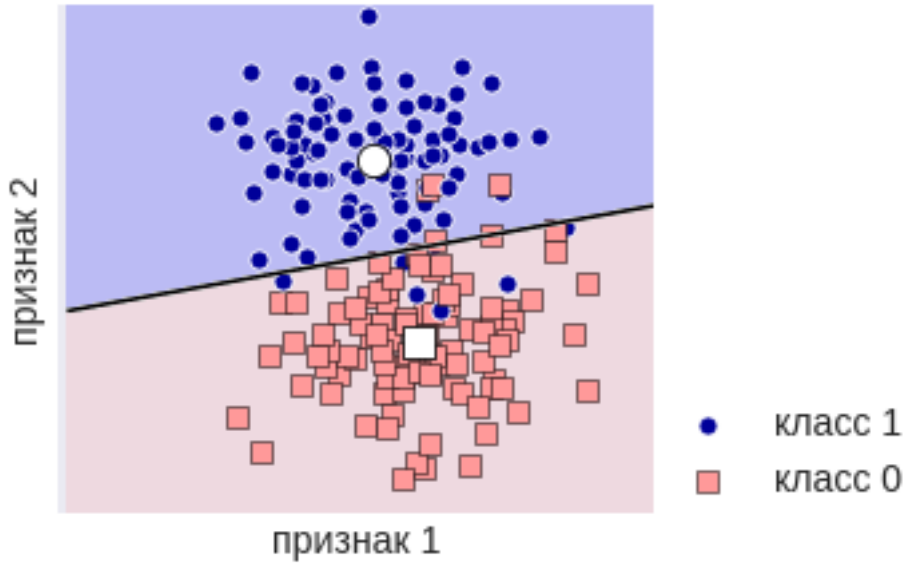
центроиды:

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

классификация:

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

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 цетроида

– очень простой алгоритм (интуитивно подходит в задачах, где объекты разных классов распределены «колоколообразно»)

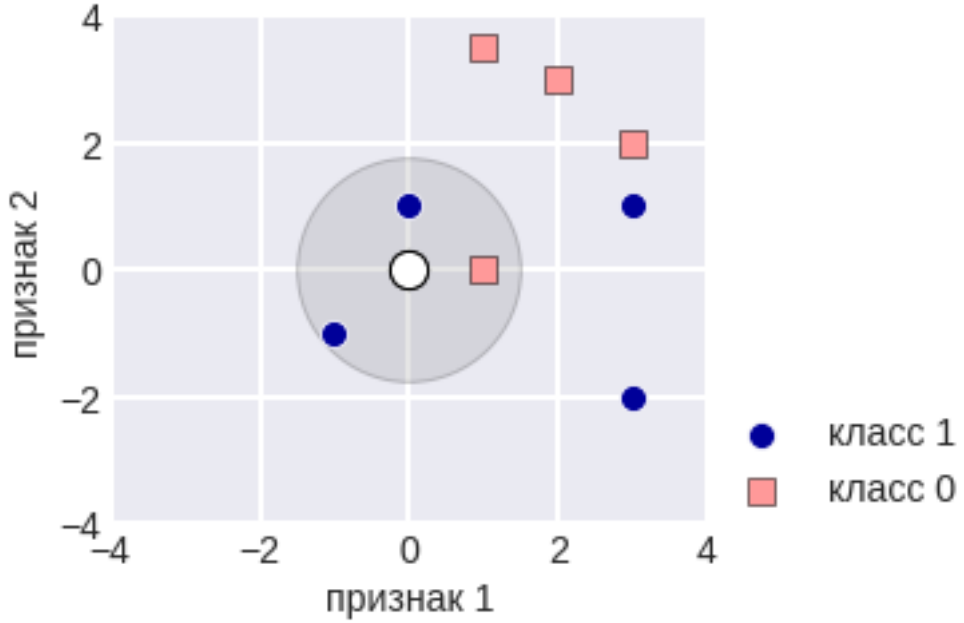
Подход, основанный на близости

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

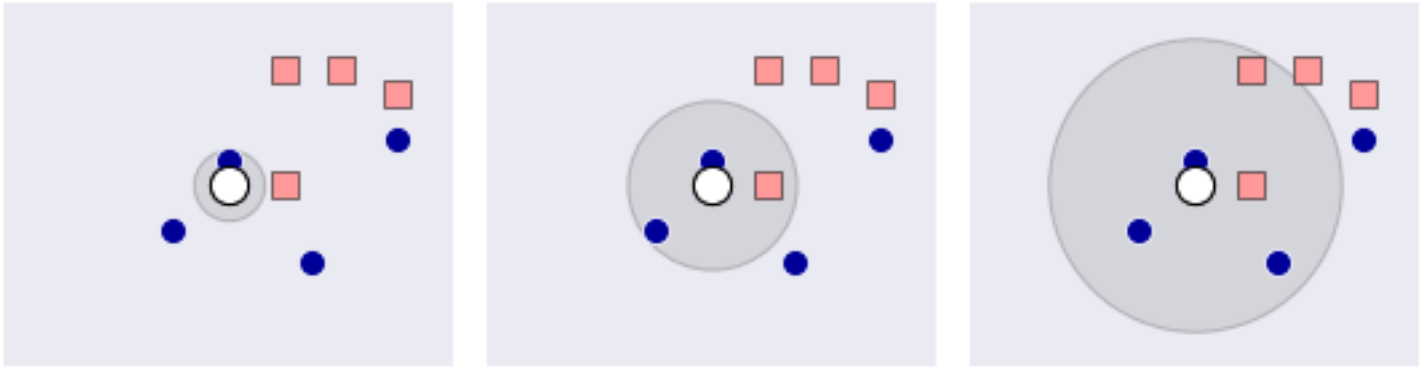
Если X – метрическое пространство с метрикой ρ,

пусть нумерация объектов такая, что

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

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

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



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 p – вероятность встретить объект класса (это же и вероятность ошибки оптимального алгоритма), тогда

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

вероятность ошибочной классификации

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

Термины

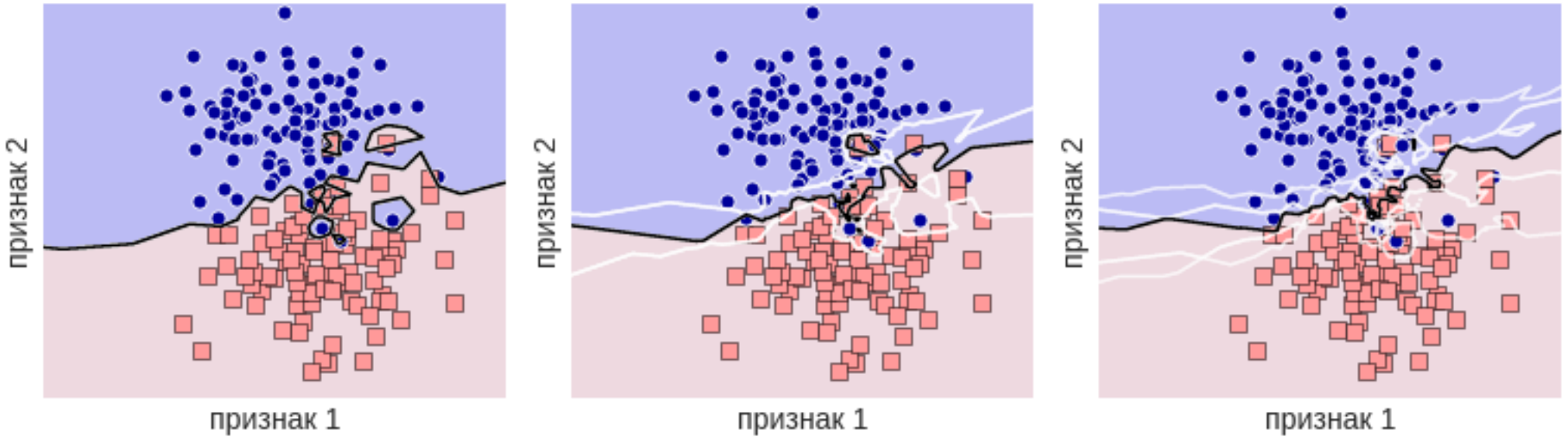
Eager learner

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

Lazy learner

не использует обучающую выборку до классификации

Решение модельной задачи при разном числе соседей

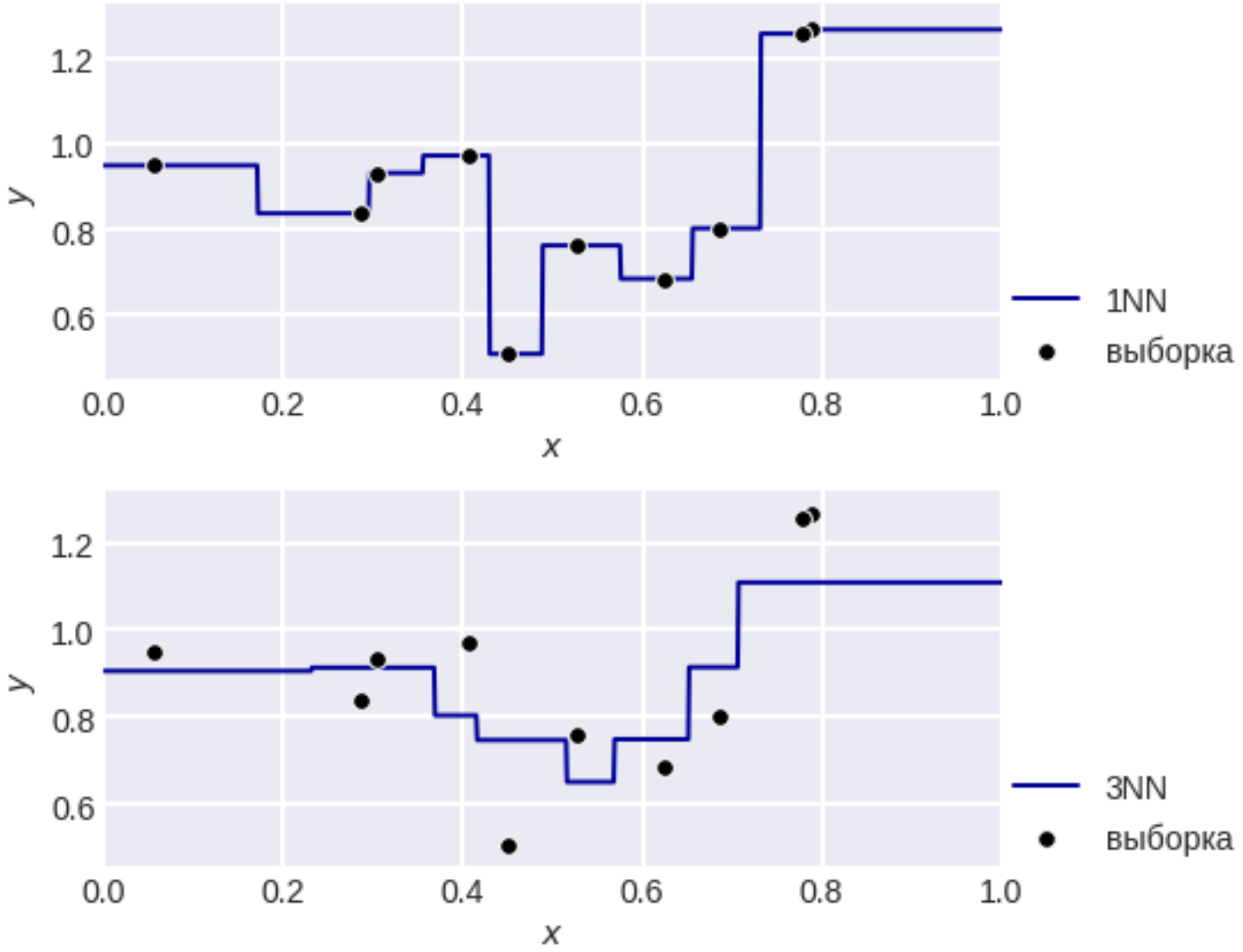


число соседей = 1, 3 и 5

Как увидим дальше, k отвечает за «сложность модели»

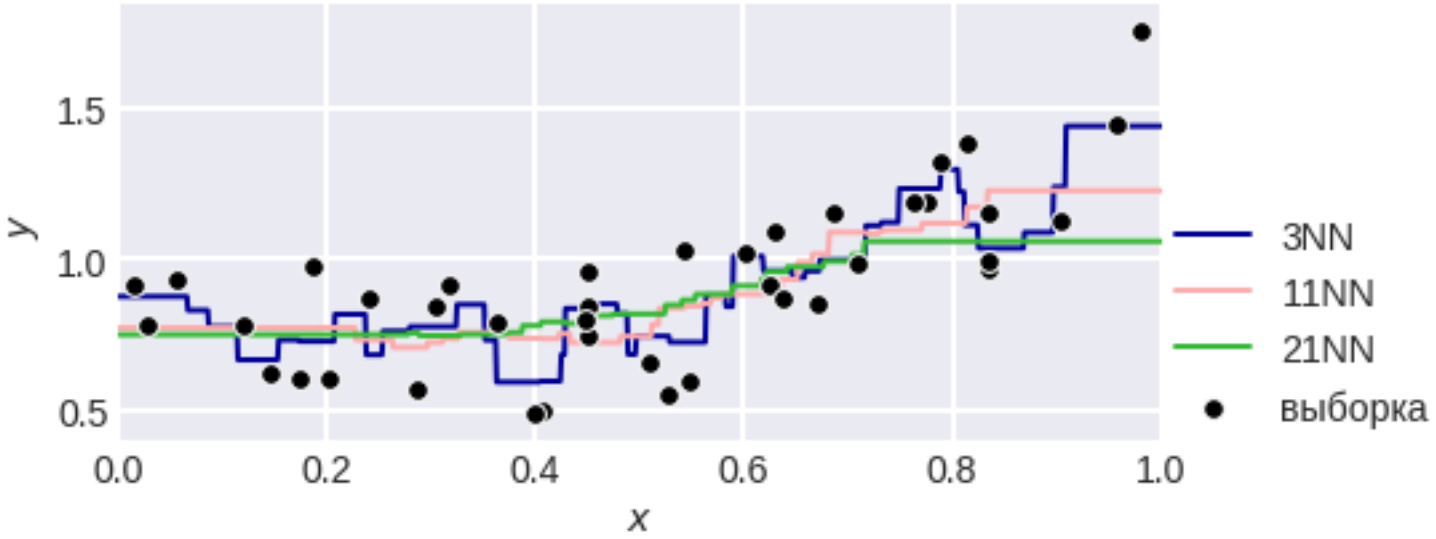
Метод ближайшего соседа

обобщается на регрессию

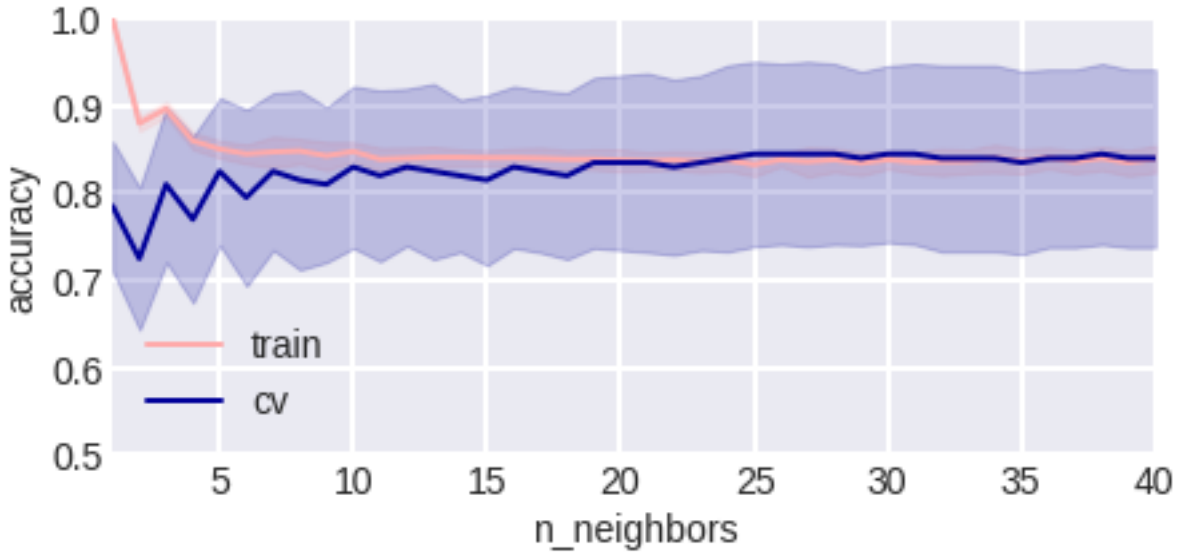


Метод ближайшего соседа

обобщается на регрессию



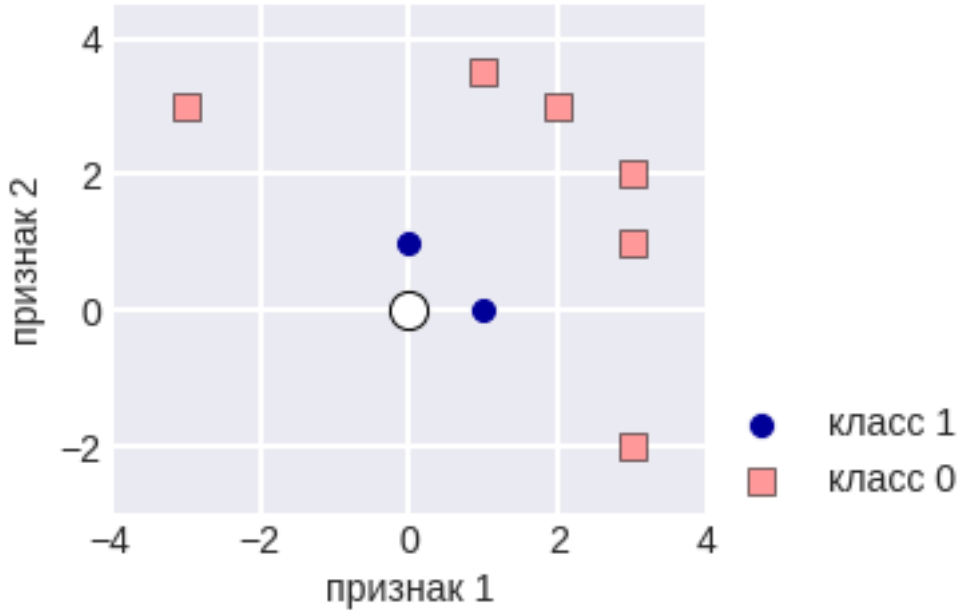
Подбор гиперпараметров специальными методами контроля



будет дальше...

```
# 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



близкие соседи должны быть важнее

Весовые обобщения kNN

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

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

Разные весовые схемы:

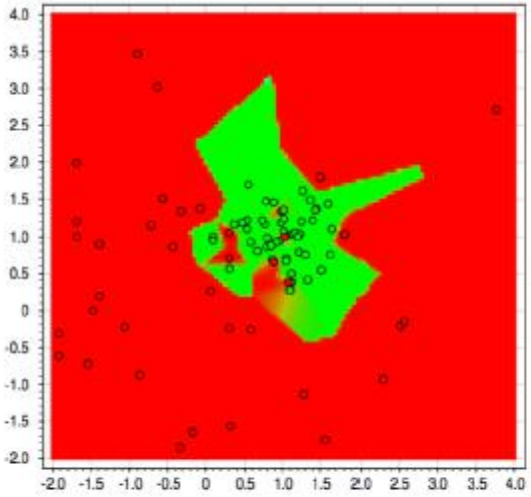
$$w_t = (k - t + 1)^{\delta}$$

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

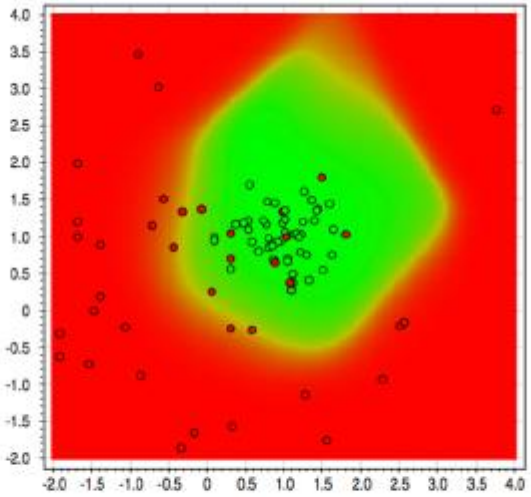
$$w_t = K\left(\frac{\rho(x, x_t)}{h(x)}\right)$$

Весовые Обобщения kNN

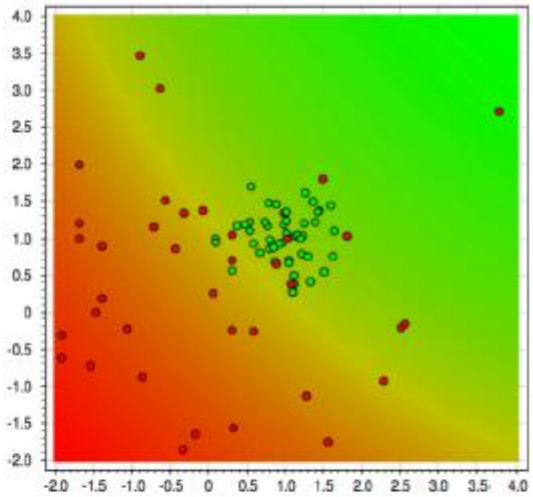
Последний способ хорош только на картинках...



$h = 0.05$



$h = 0.5$

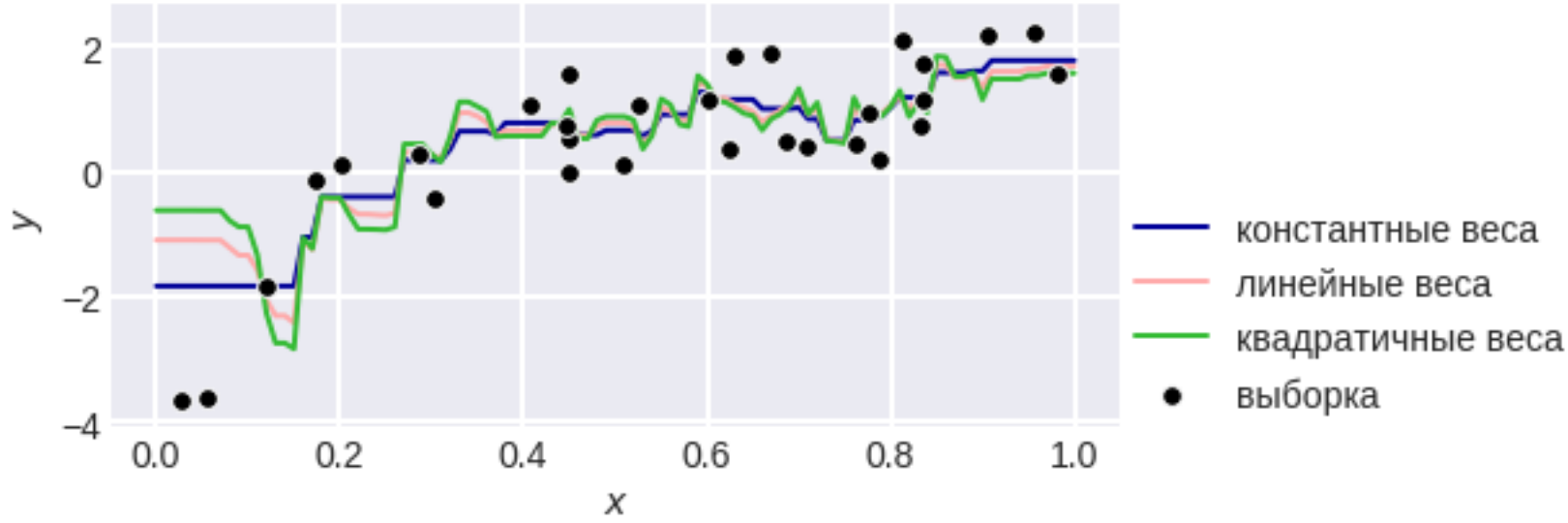


$h = 5$

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 $\rho(x,z): X \times X \to \mathbb{R}$

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**
- **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 (Lp)

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

Предельный случай – Чебышева (L∞)

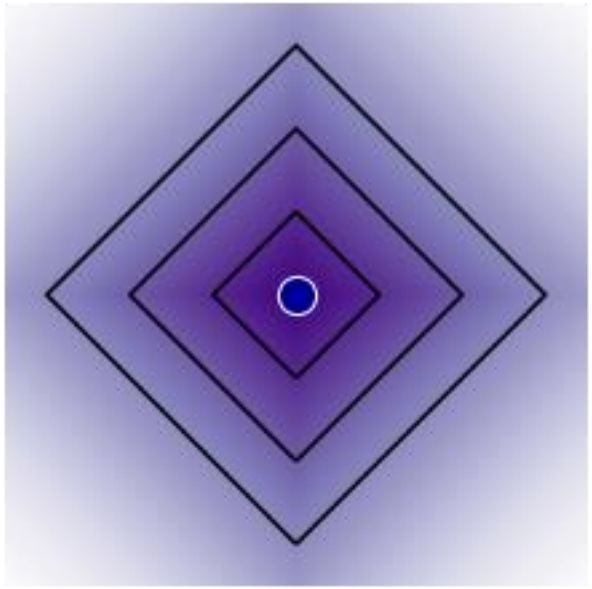
$$\left(\sum_{i=1}^{n}|\, x_i - z_i\,|^{\infty}\right)^{1/\infty} \sim \max_i |\, x_i - z_i\,|$$

Частный случай – Манхэттенская (L1)

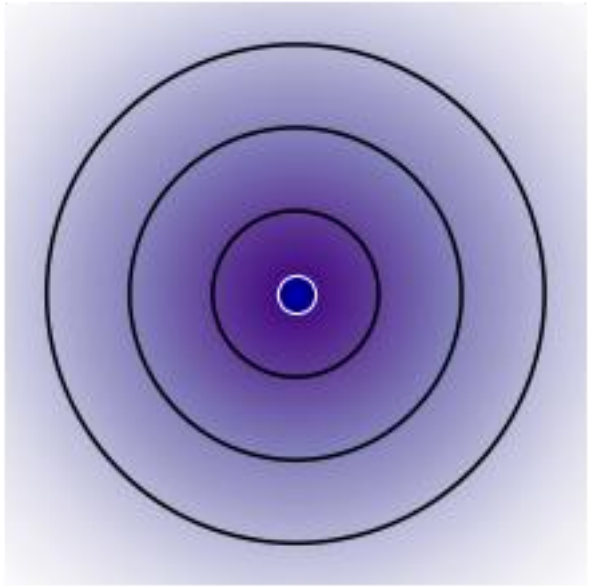
$$\sum_{i=1}^{n}|\, x_i - z_i\,|$$

Различные метрики

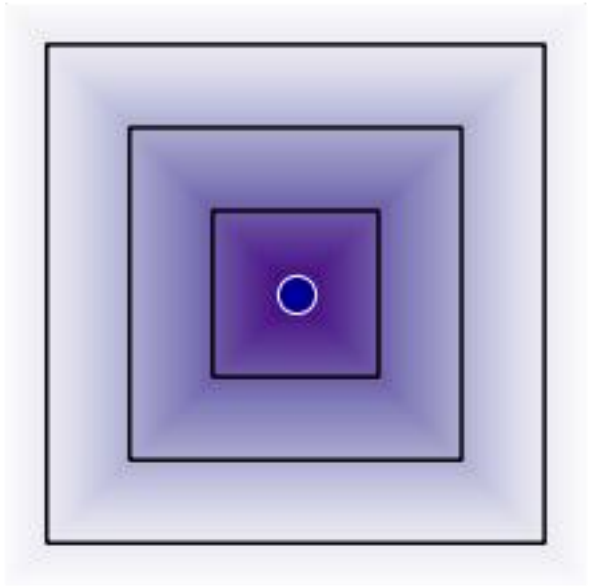
$$\left(| x_1 - z_1 |^p + | x_2 - z_2 |^p \right)^{1/p}$$



L_1



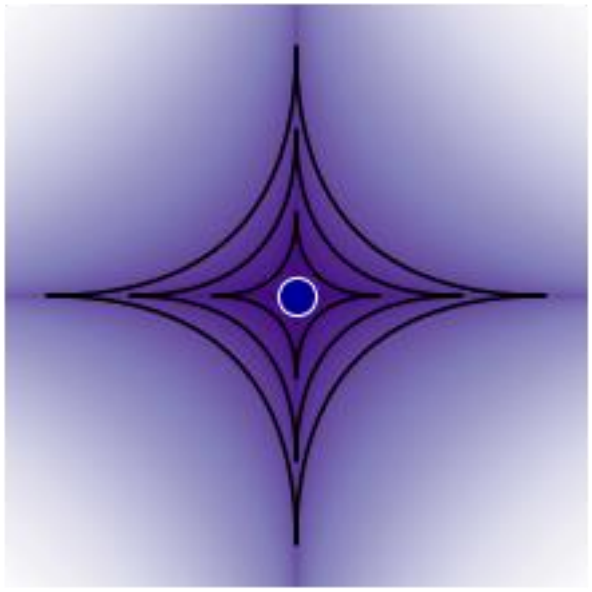
L_2



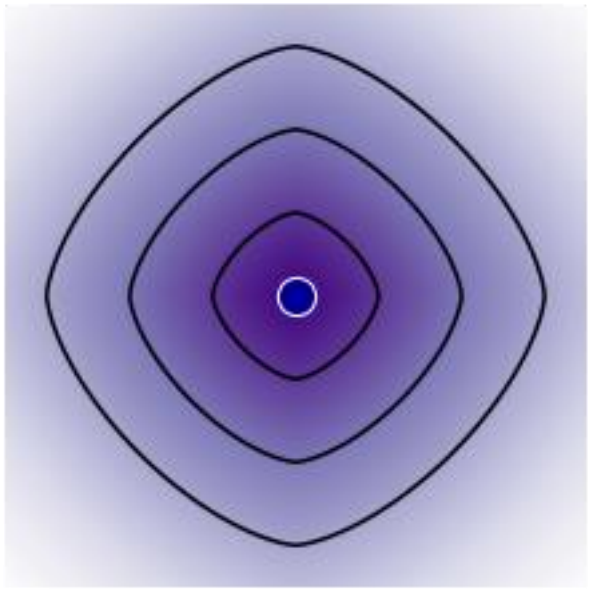
L_∞

Различные метрики

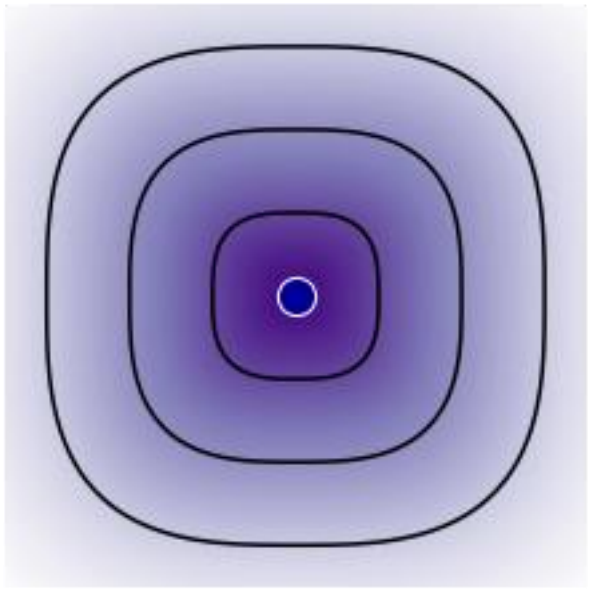
$$\left(| x_1 - z_1 |^p + | x_2 - z_2 |^p \right)^{1/p}$$



$L_{0.5}$

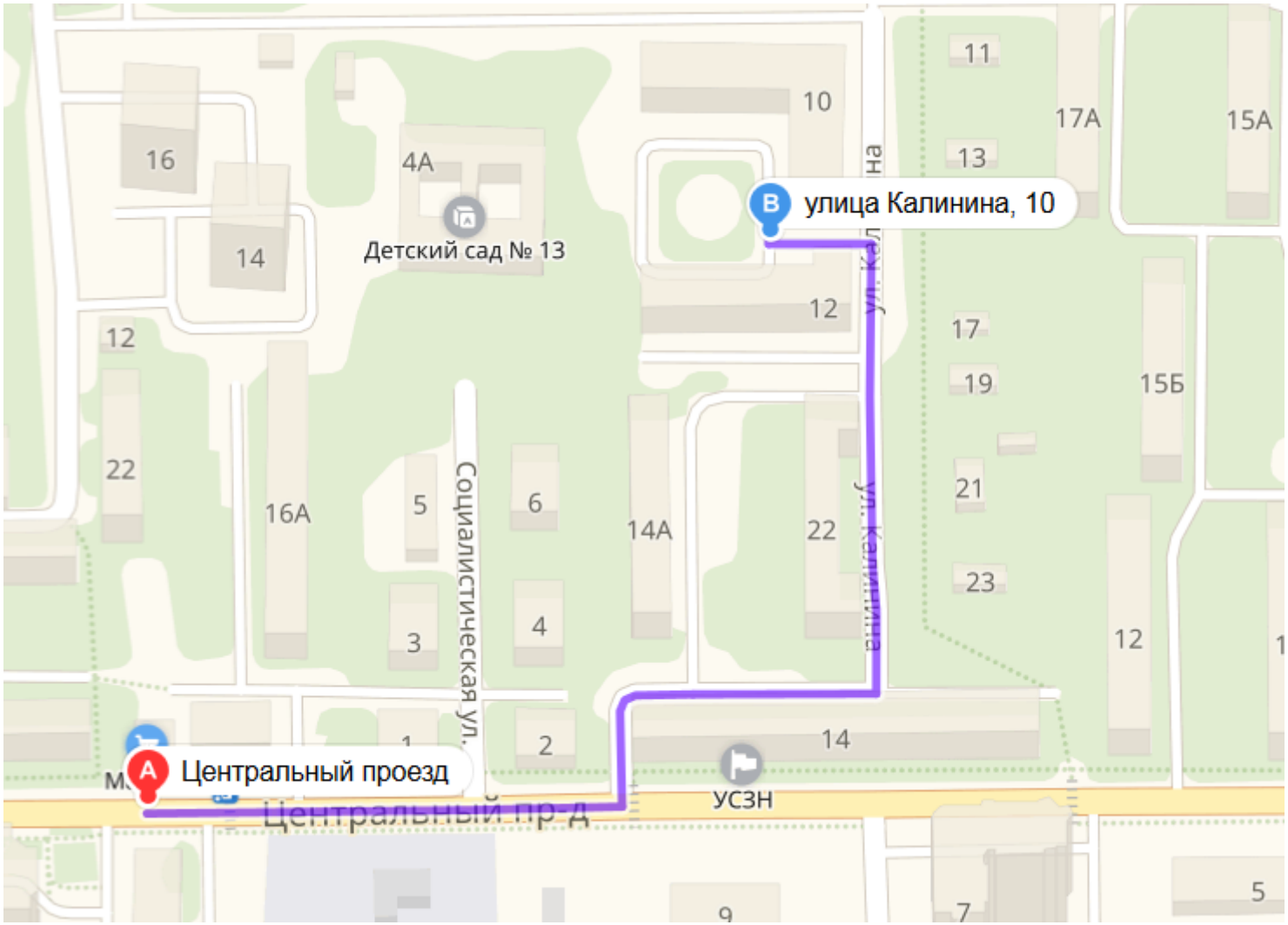


$L_{1.5}$

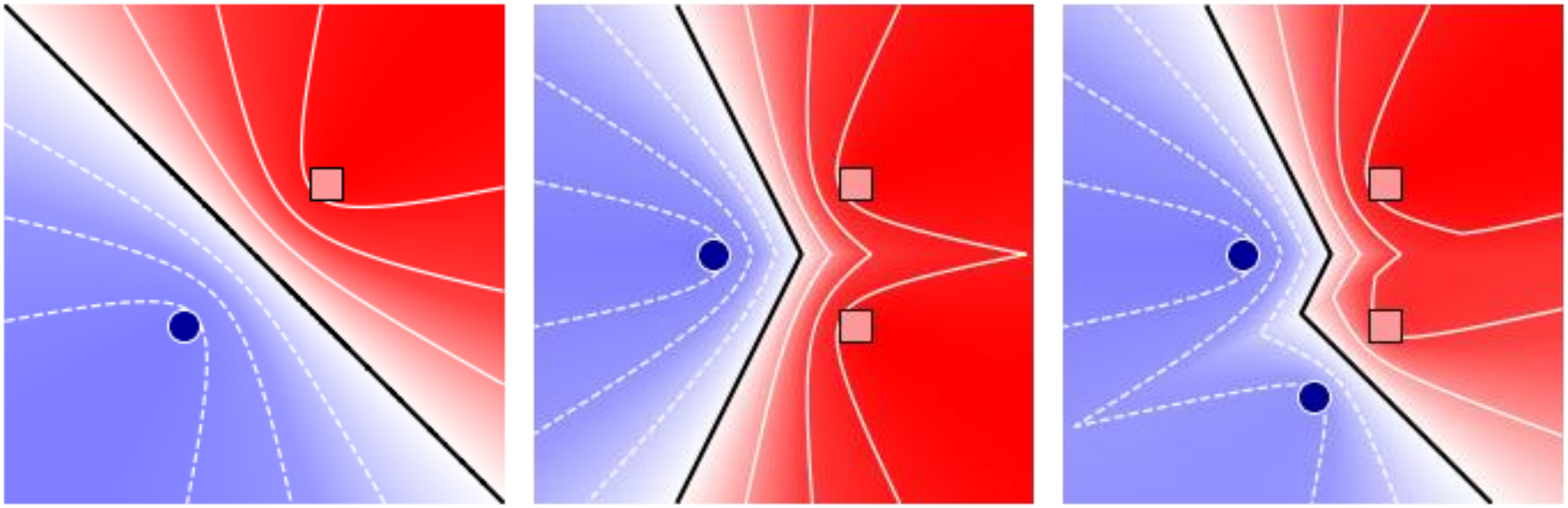


L_3

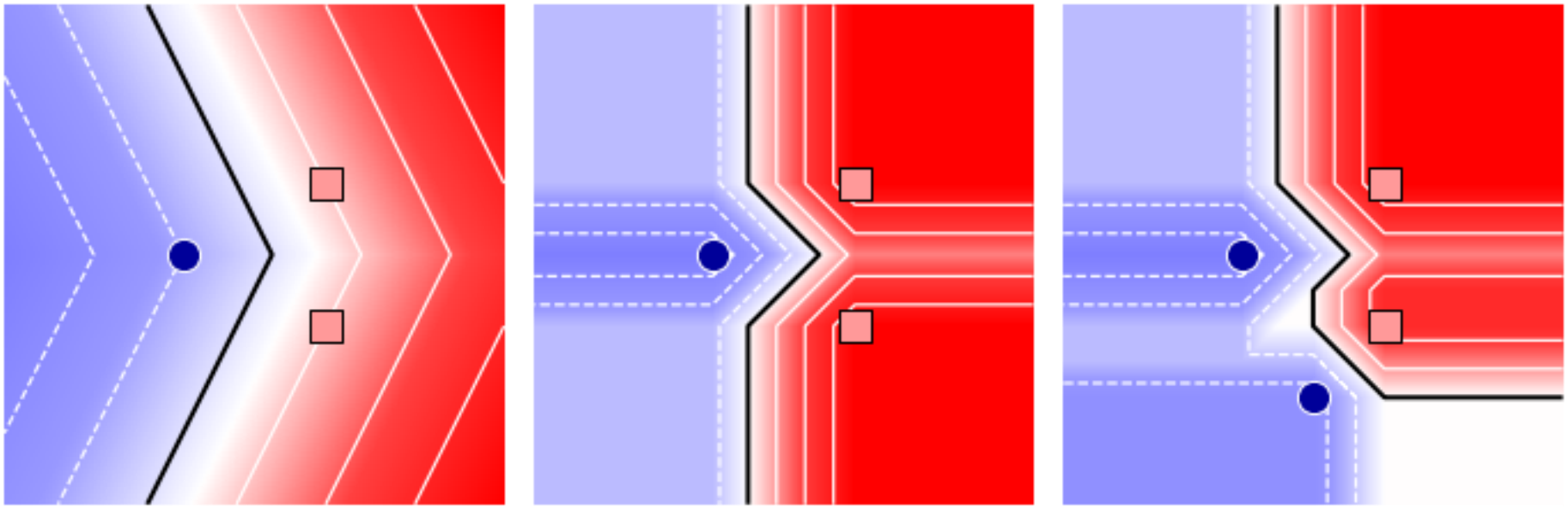
Различные метрики



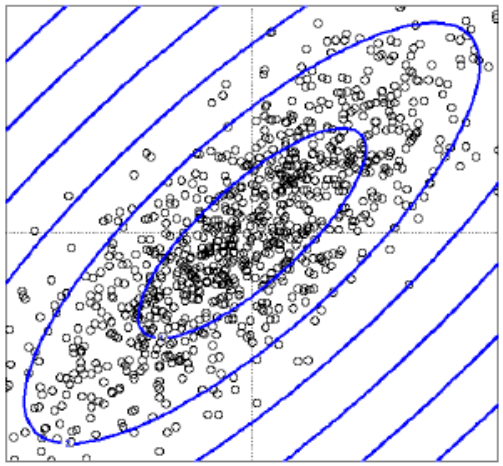
Разделяющие поверхности L_2



Разделяющие поверхности L_1



Расстояние Махалонобиса
(Mahalonobis distance)



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

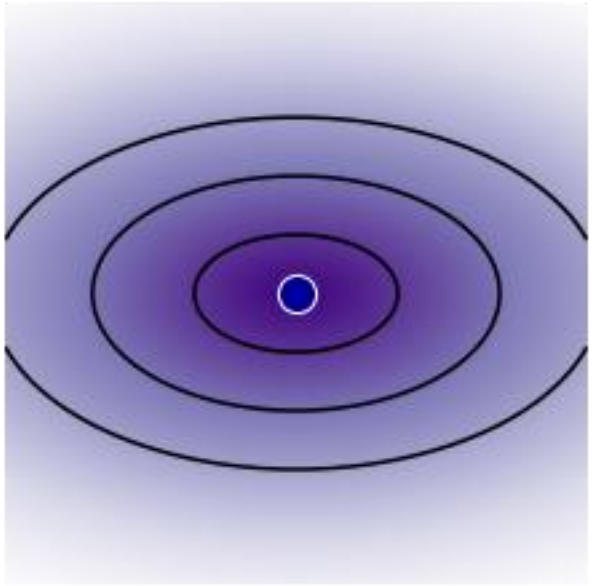
стандартизует нормальные данные

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

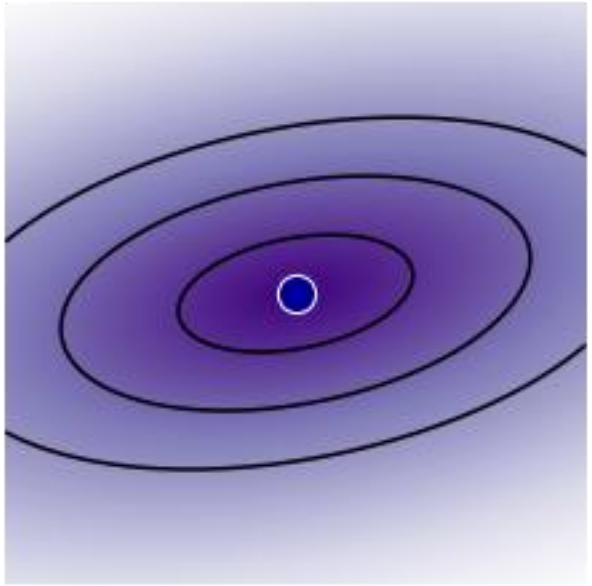
$$\rho(x,z) = \rho_{L_2}(\varphi(x), \varphi(z)) = \sqrt{(\varphi(x) - \varphi(z))^{\text{T}}(\varphi(x) - \varphi(z))} =$$

$$= \sqrt{(x - \mu)^{\text{T}} \Sigma^{-1} (x - \mu)}$$

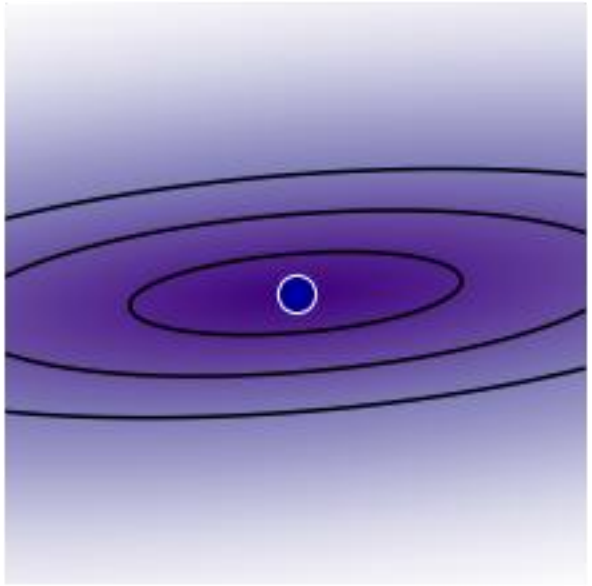
Расстояние Махалонобиса



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



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



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

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

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

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

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

тут есть много разных вариантов…

Проблема выбора метрики

- **зависимость от масштаба**

нормировка признаков

однородные признаки

смесь метрик

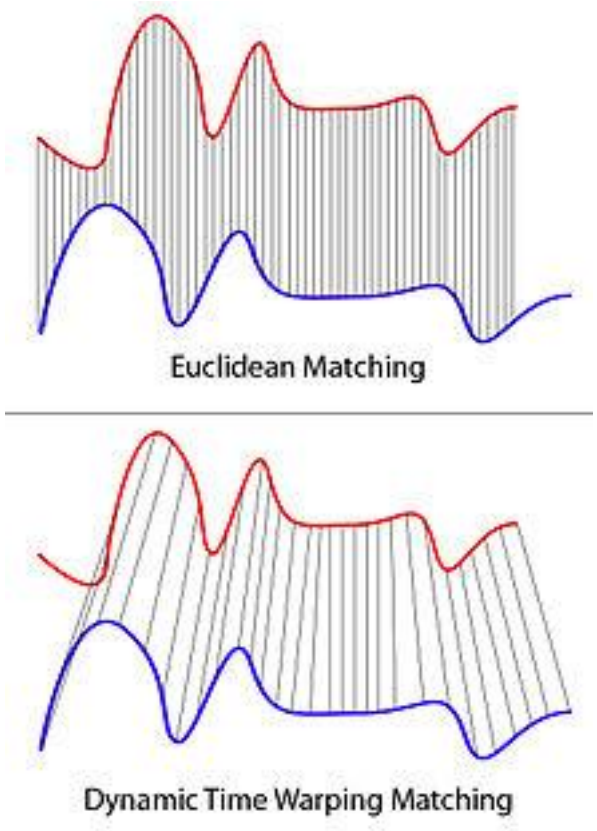
- **можно выбирать не метрику, а близость**

пример: косинусная мера сходства

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

Метрики на временных рядах

Евклидово расстояние
DTW = Dynamic time warping



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

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

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

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

при $i > 0$

Метрики на временных рядах

Для адекватности и быстроты часто

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

красивая картинка

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

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', ...

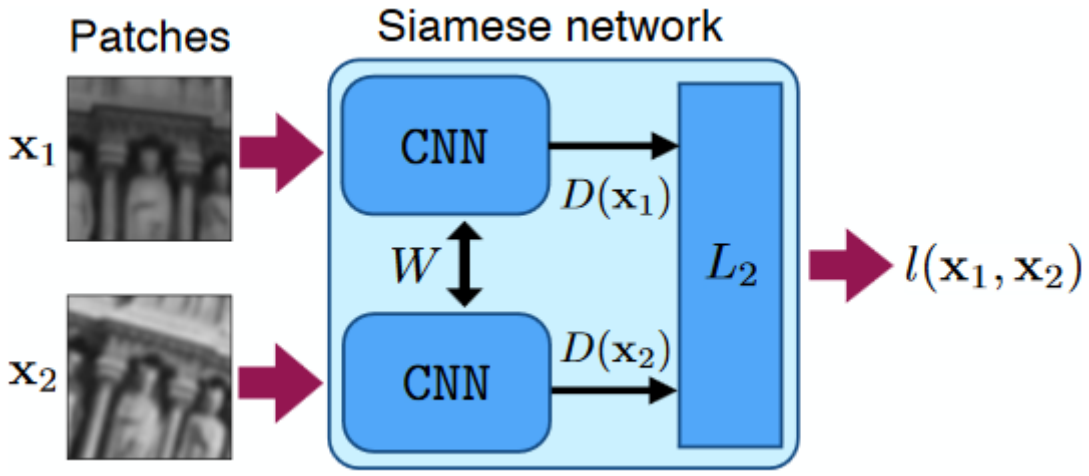
$$\rho(\text{Lecture}_1,\text{Lecture}_2)=c_1\cdot\rho_1(\text{Author}_1,\text{Author}_2)+c_2\cdot\rho_2(\text{Title}_1,\text{Title}_2)+\ldots$$
$$+c_n\cdot\rho_n(\text{Subject}_1,\text{Subject}_2)$$

метрики можно параметризовать и настраивать параметры

«хитрый весовой учёт близости» – см. совместные просмотры

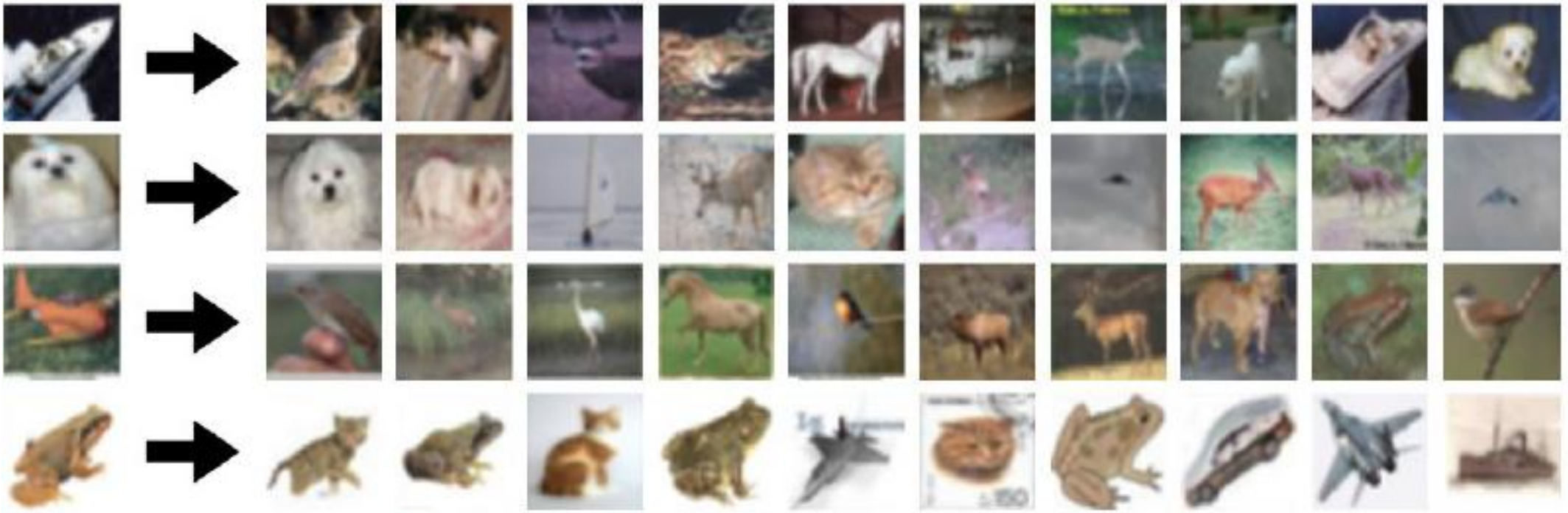
Дьяконов А.Г. Алгоритмы для рекомендательной системы: технология LENKOR // Бизнес-Информатика, 2012, №1(19), С. 32–39.

Приложения метрического подхода: 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

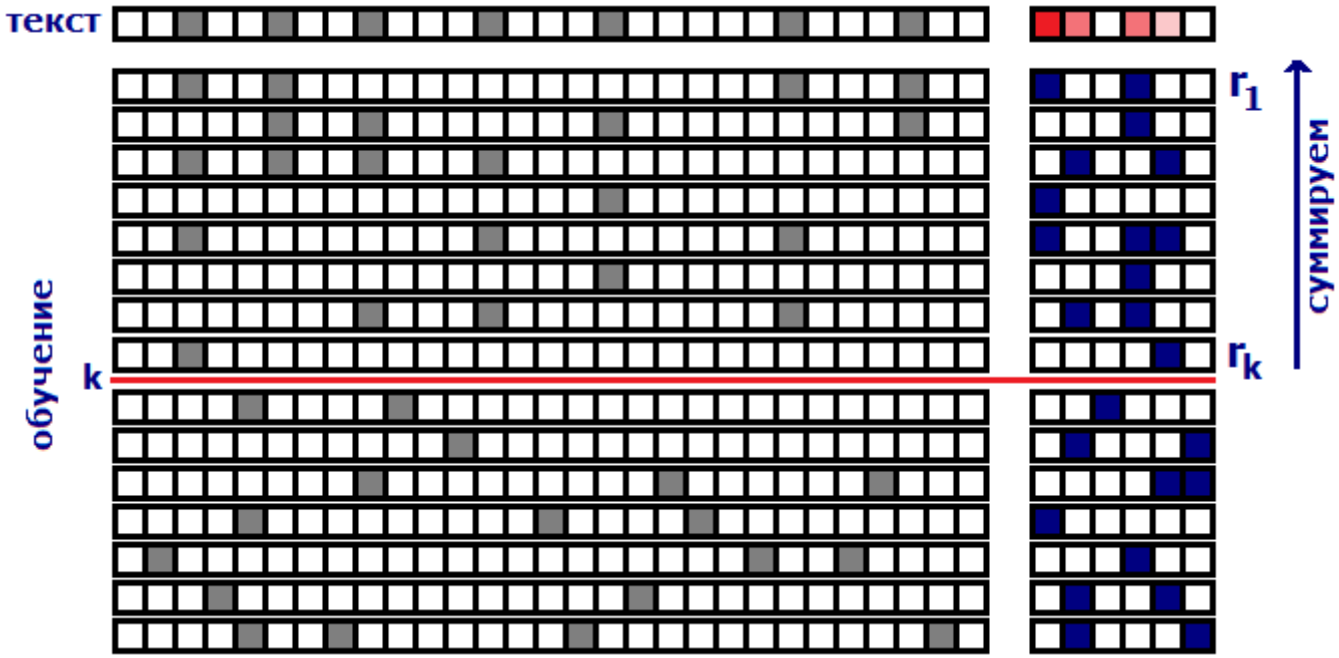


http://cs231n.stanford.edu/2017/syllabus.html

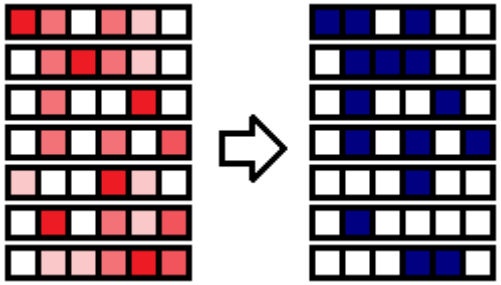
Приложения метрического подхода: классификация текстов

задача «Large Scale Hierarchical Text Classification»

Взвешенный kNN



Итог – матрица оценок



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

Эффективные методы поиска ближайших соседей *

nested rectangles: KD tree

nested balls: ball tree

Метрические алгоритмы

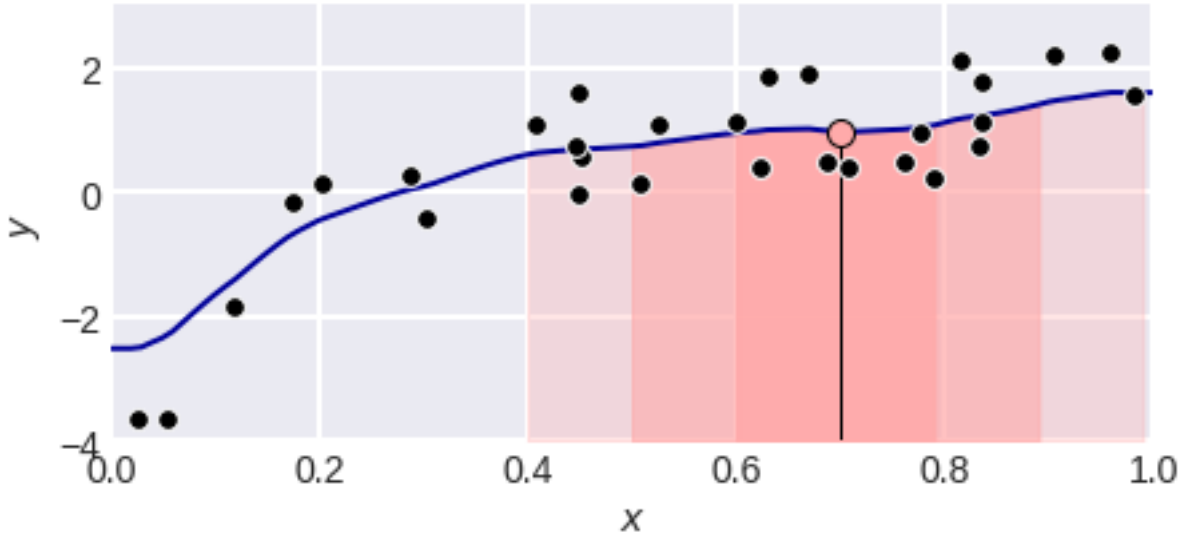
+ не требуется признаковых описаний
(достаточно уметь измерять расстояния / близости)
+ легко реализуемы
+ интерпретируемость
+ нет обучения

– 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)



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)

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

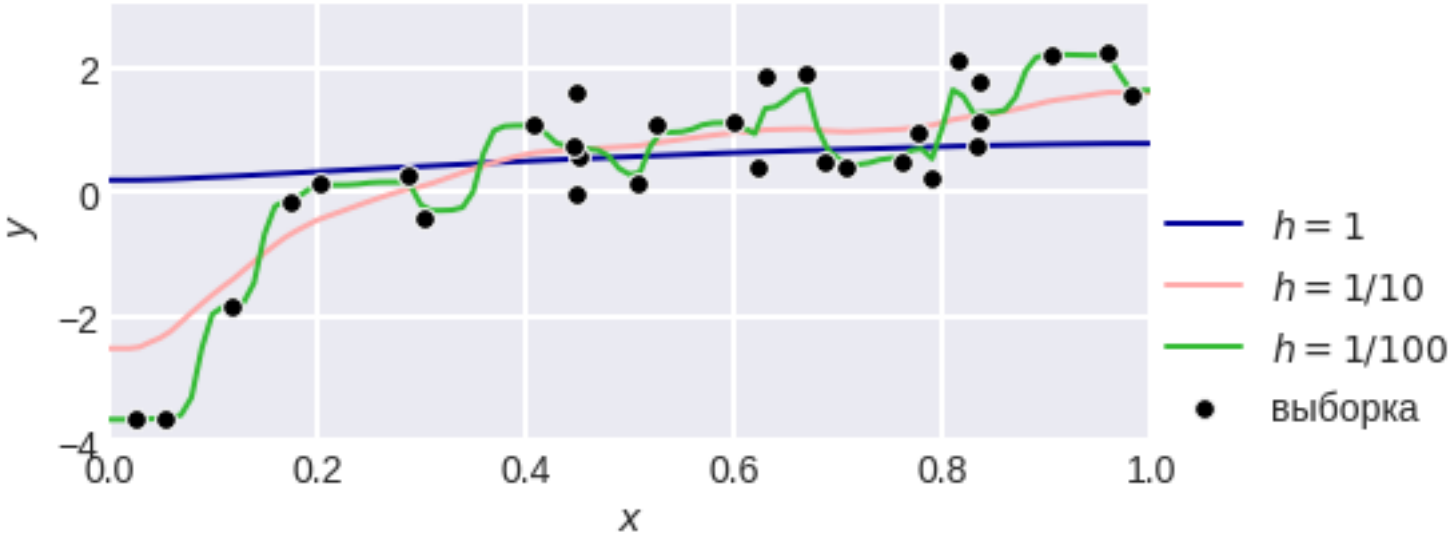
Смысл весов – чем ближе объект обучения, тем скорее ответ похож на его метку

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

Ядро с шириной h.

Здесь также как выше… (про функции ядра)

Регрессия Надарая-Ватсона (Nadaraya-Watson regression)



Регрессия Надарая-Ватсона (Nadaraya-Watson regression)

Смысл:

ответ алгоритма – решение оптимизационной задачи

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

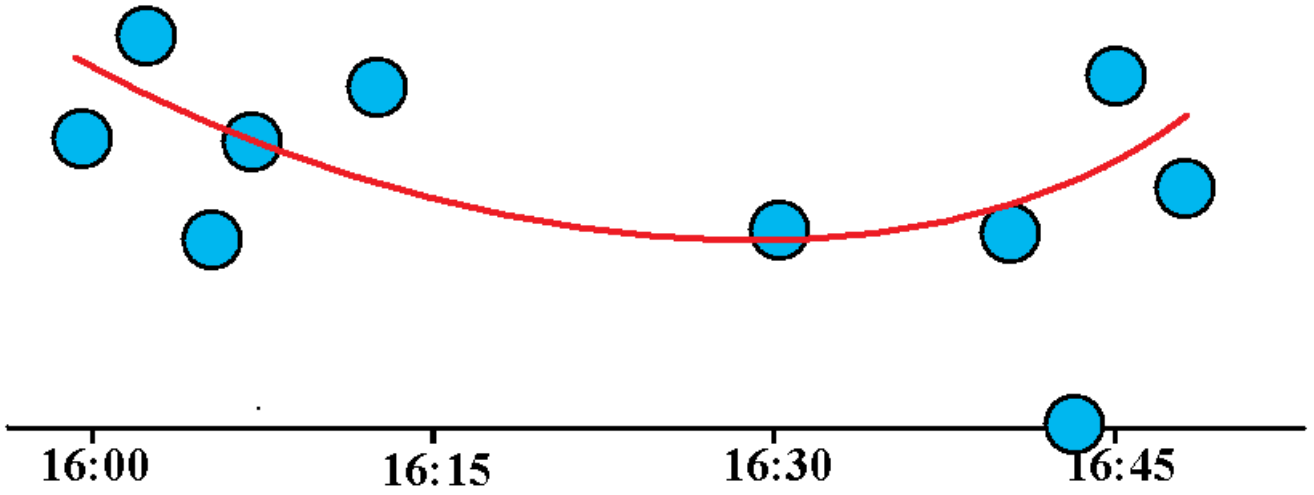
+ хорошее решение задачи сглаживания

– не решает задачи экстраполяции

Приложения регрессии Надарая-Ватсона

1. Сглаживание сигналов

2. «Многомерные» усреднения



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.KernelDensity
```

– KDE-оценка плотности

Код

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