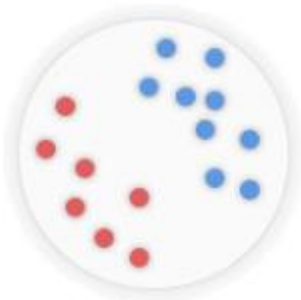
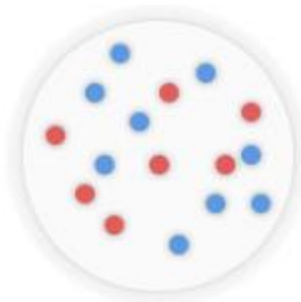
Метрические методы классификации

Гипотеза компактности

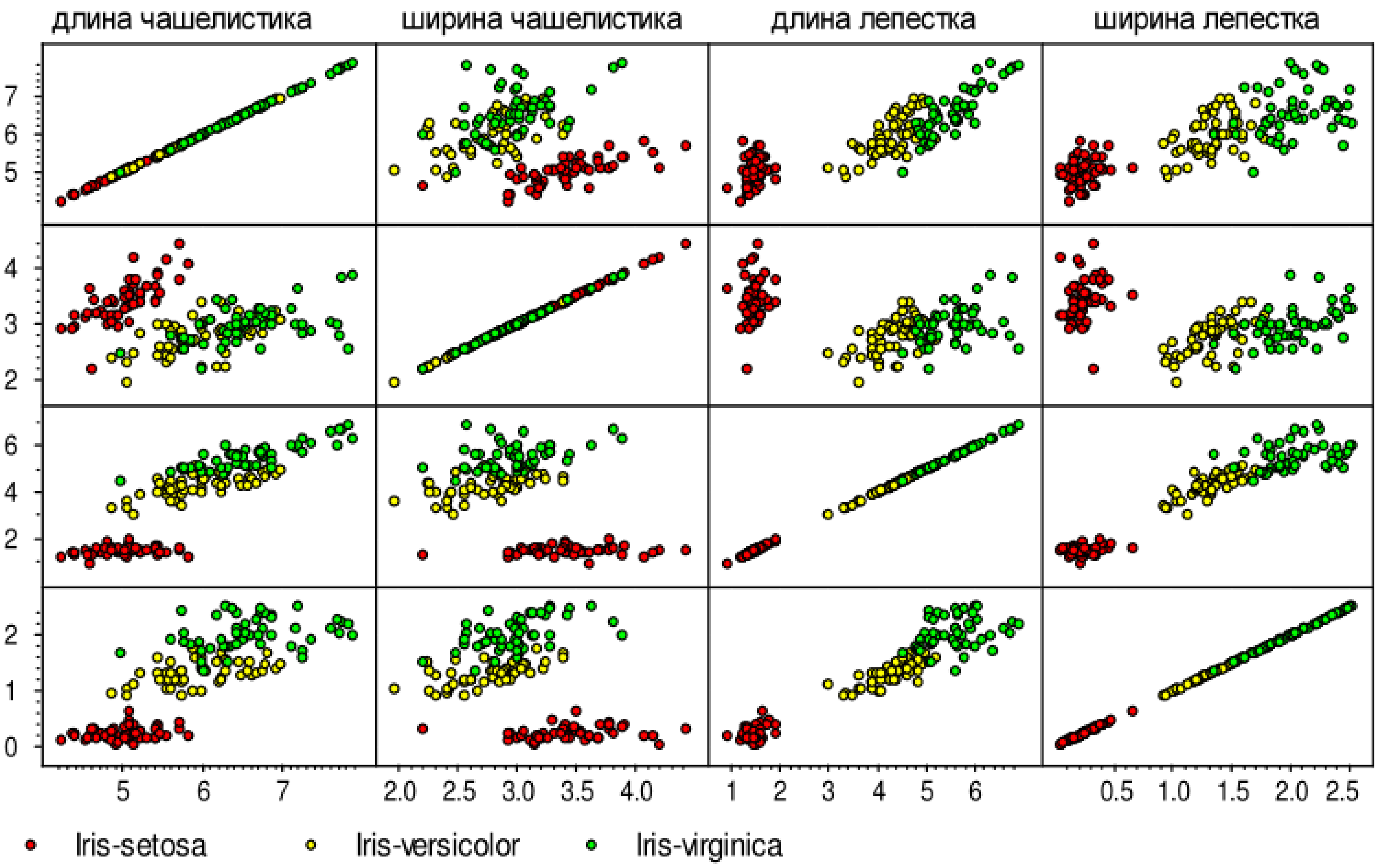


Выполнена



Не выполнена

Ирисы


длина чашелистика
ширина чашелистика
длина лепестка
ширина лепестка
7
6
5
4
3
2
6
4
2
2
1
0
5
6
7
2.0
2.5
3.0
3.5
4.0
1
2
3
4
5
6
0.5
1.0
1.5
2.0
2.5
Iris-setosa
Iris-versicolor
Iris-virginica

Формализация понятия «близость»

Евклидова метрика (теорема Пифагора)

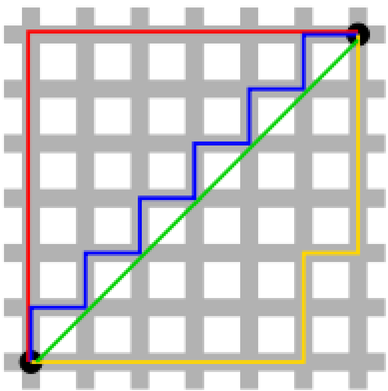
и обобщённая метрика Минковского

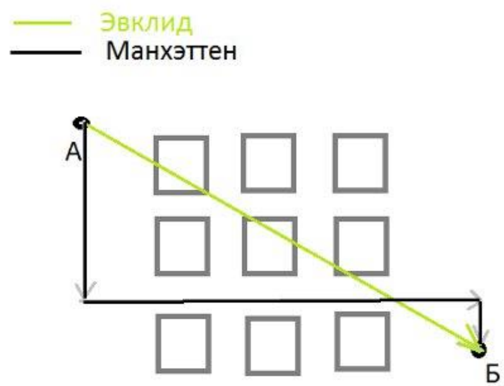
$$\rho(x, x_i) = \left(\sum_{j=1}^{n} \left| x^j - x_i^j \right|^2 \right)^{1/2}$$

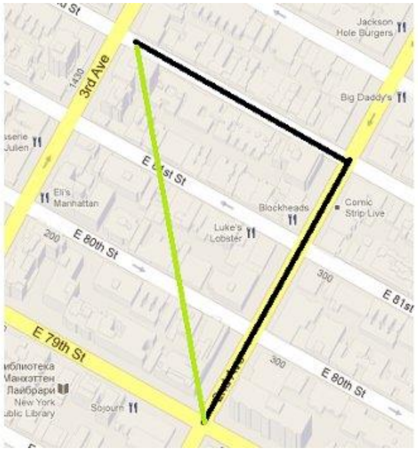
$$\rho(x, x_i) = \left(\sum_{j=1}^{n} w_j \left| x^j - x_i^j \right|^p \right)^{1/p}$$

- $\rho(x, y) \geq 0$
- $\rho(x, y) = \rho(y, x)$
- $\rho(x, y) \leq \rho(x, z) + \rho(z, y)$

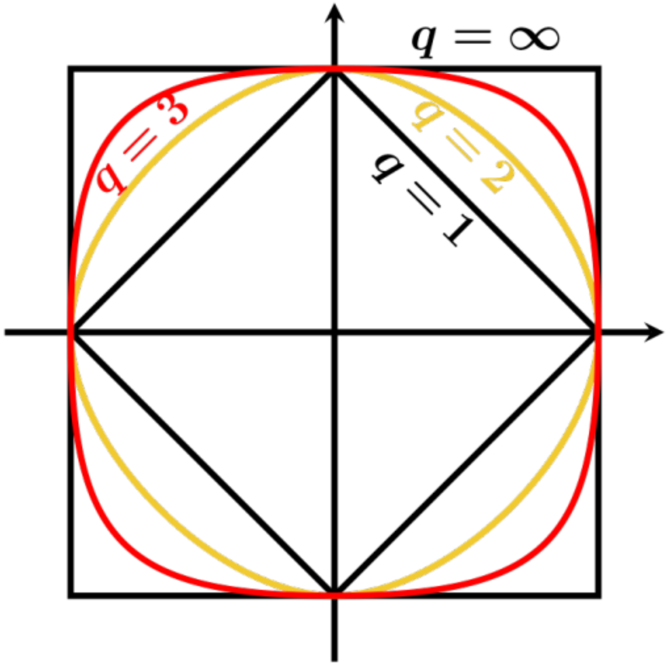
Метрика Евклида, метрика Манхэттена







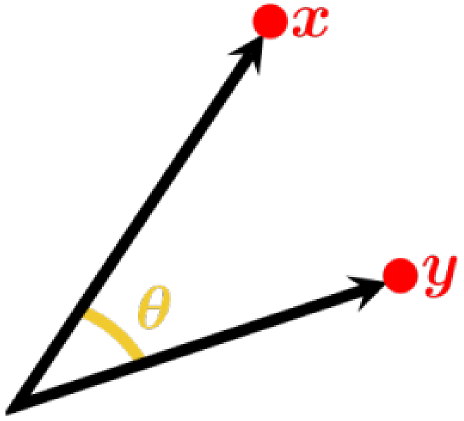
Анализ метрик



Косинусное расстояние

$$\text{similarity} = \cos(\theta) = \frac{x \cdot y}{\|x\| \cdot \|y\|} = \frac{\sum\limits_{i=1}^{n} x_i y_i}{\sqrt{\sum\limits_{i=1}^{n} x_i^2} \sqrt{\sum\limits_{i=1}^{n} y_i^2}}$$

Функция близости, а не расстояние



Коэффициент корреляции Пирсона

$$r = \frac{\sum_{i=1}^{n}((x_i - \bar{x})(y_i - \bar{y}))}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Другие метрики

- Скалярное произведение: $\sum x_i \cdot y_i$
- Коэффициент Дайса: $\frac{2\sum x_i \cdot y_i}{\sum x_i^2 + \sum y_i^2}$
- Косинусная мера: $\frac{\sum x_i \cdot y_i}{\sqrt{\sum x_i^2} \cdot \sqrt{\sum y_i^2}}$
- Коэффициент Жаккара: $\frac{\sum x_i \cdot y_i}{\sum x_i^2 + \sum y_i^2 - \sum x_i \cdot y_i}$

Обобщённый метрический классификатор

$$\rho\left(u, x_u^{(1)}\right) \leq \rho\left(u, x_u^{(2)}\right) \leq \cdots \leq \rho\left(u, x_u^{(l)}\right)$$

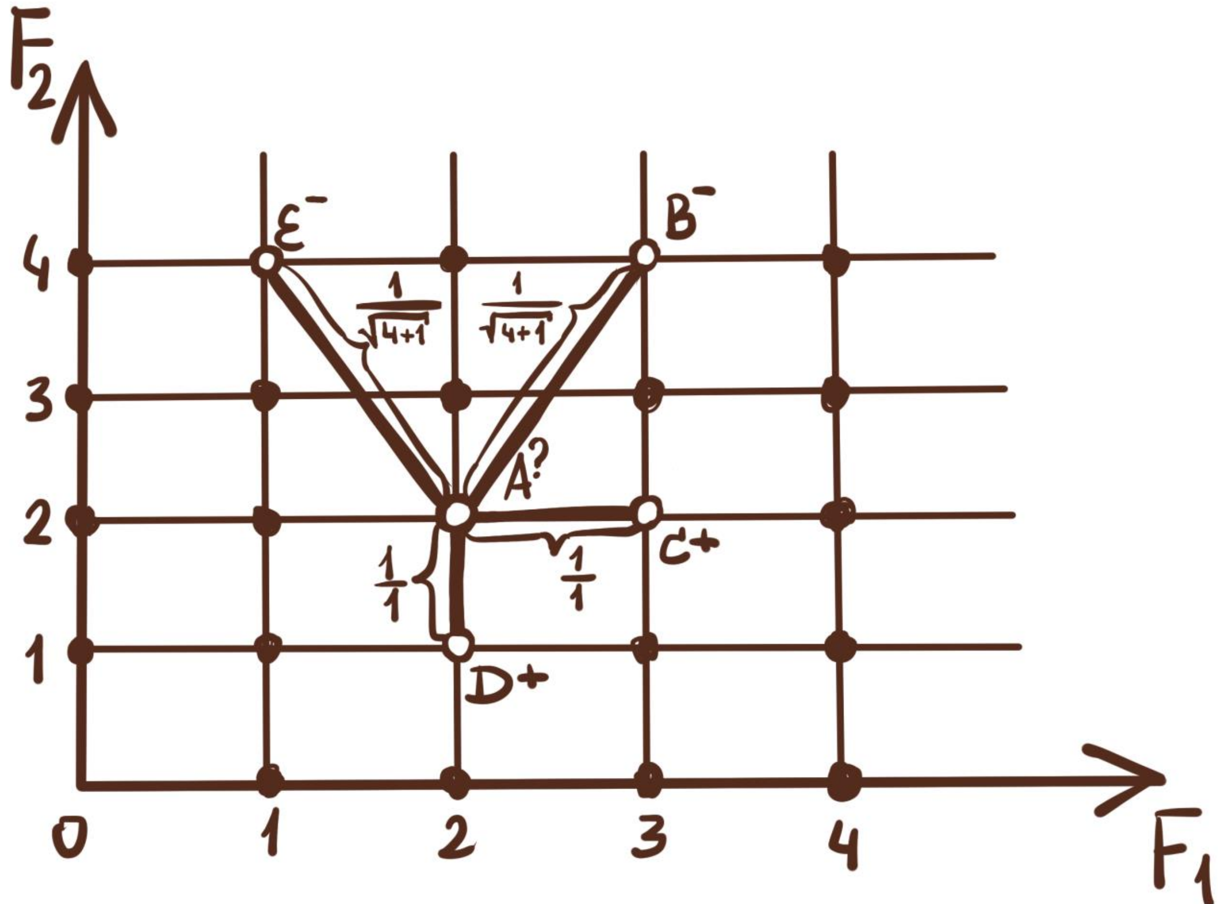
$$a(u; X^l) = arg\max_{y \in Y} \Gamma_l(u, X^l)$$

$$\Gamma_l(u, X^l) = \sum_{i=1}^{l} \left[y_u^{(i)} = y\right] w(i, u)$$

$y_u^{(i)}$ - класс (ответ) *i*-го соседа объекта *u;*

$x_u^{(i)}$ - *i*-ый сосед объекта *u.*

Пример работы



Метод *k* ближайших соседей

$w(i, u) = [i = 1]$; - метод ближайшего соседа;

$w(i, u) = [i \leq k]$; - метод *k* ближайших соседей.

Преимущества:

- Простота реализации;
- Параметр *k* можно оптимизировать.

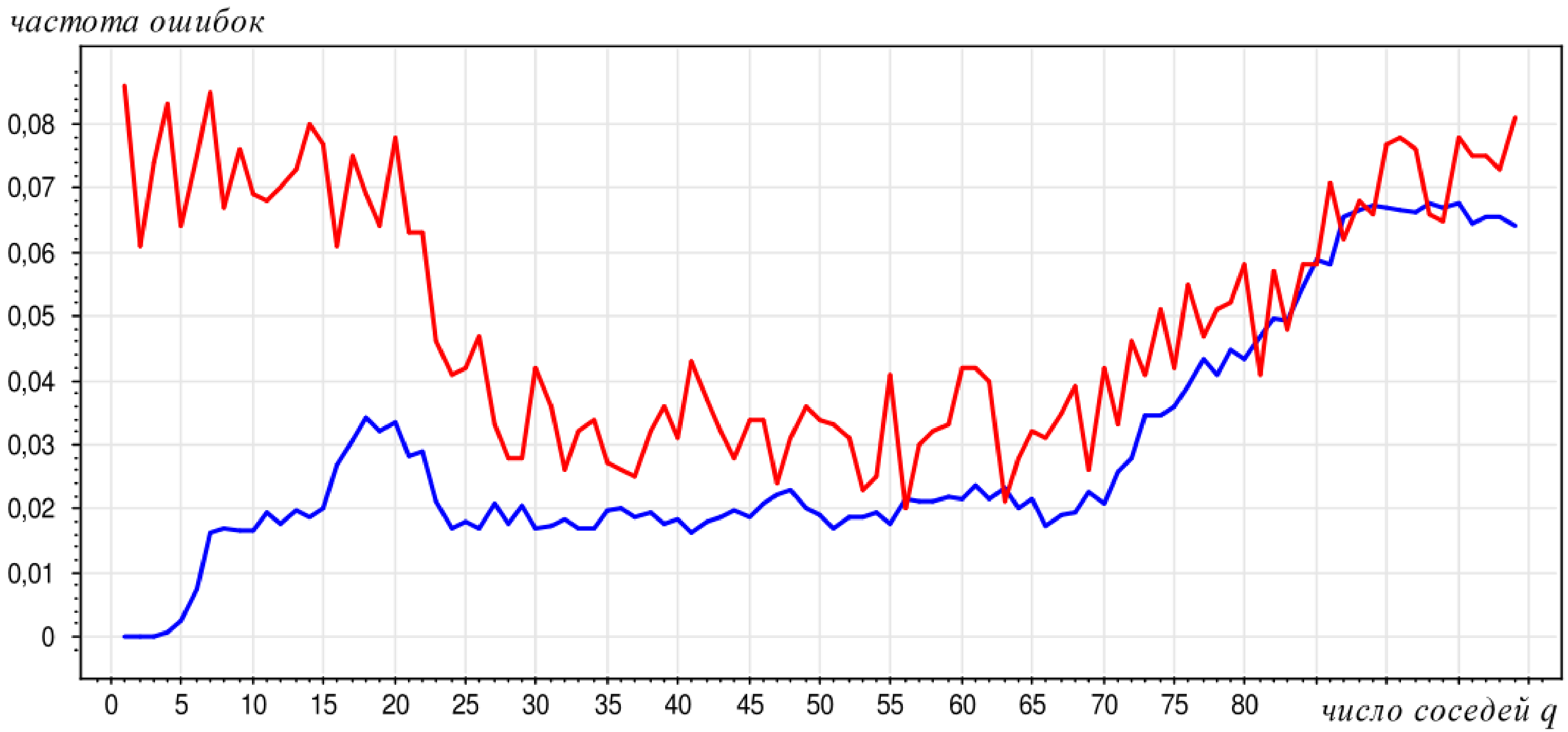
Недостатки:

- Неоднозначность классификации при $\Gamma_y(x) = \Gamma_s(x), y \neq s$;
- Не учитывается значение расстояния (учитывается только близость).

Leave one out

$$LOO\left(k, X^l\right) =$$
$$= \sum_{i=1}^{l} \left[a\left(x_i; X^l \setminus \{x_i\}, k\right) \neq y_i\right] \to \min_k$$

Пример. UCI: Iris



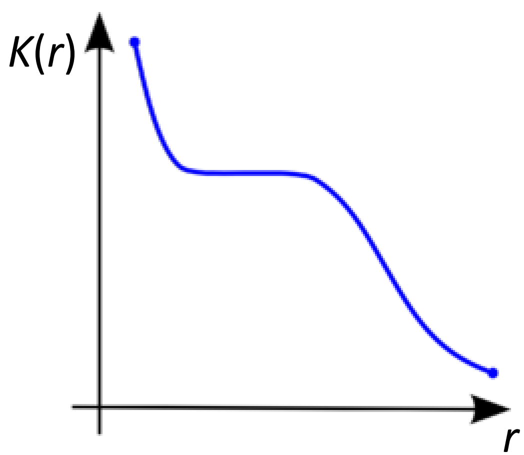
— смещённое число ошибок, когда объект учитывается как сосед самого себя

— несмещённое число ошибок LOO

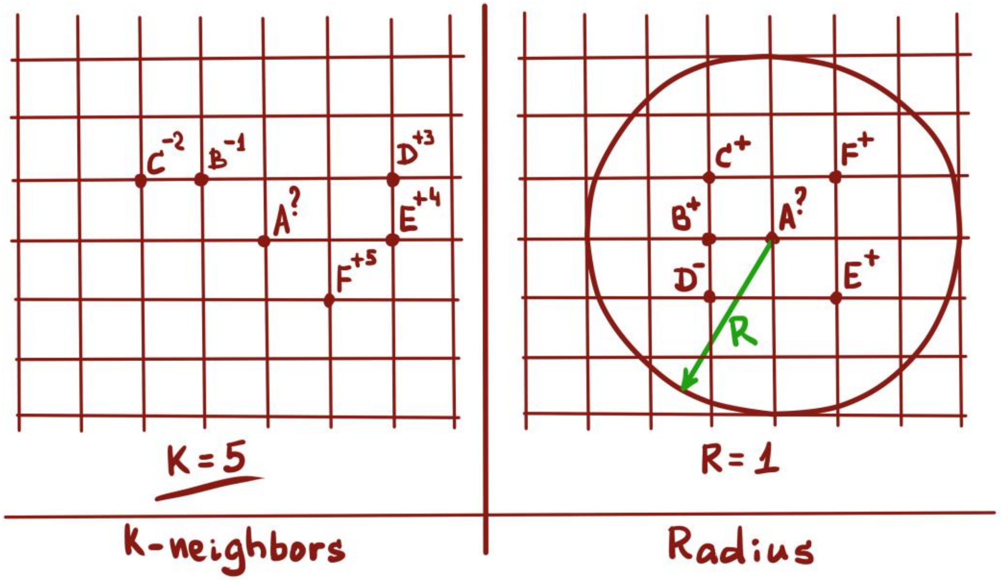
Метод окна Парзена

$w(i, x) = K\left(\frac{\rho(x, x^{(i)})}{h}\right)$, h – ширина окна;

$K(r)$ – ядро, невозрастающее

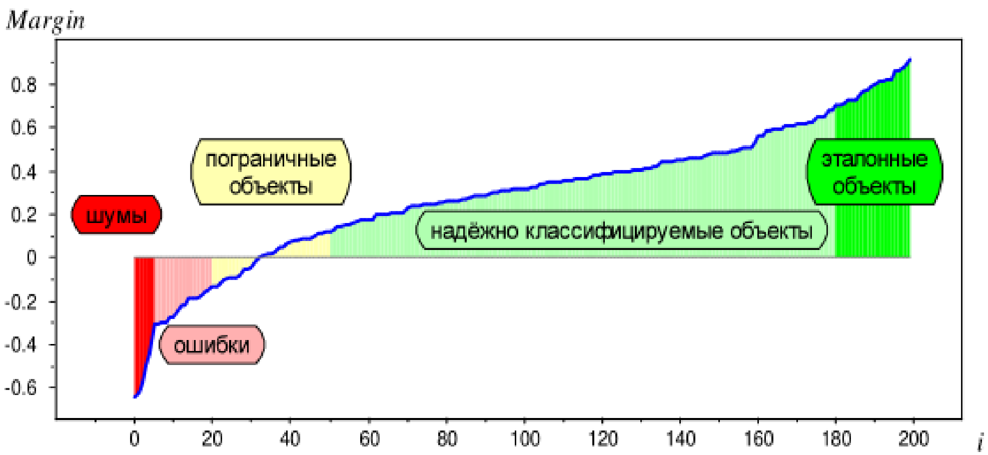


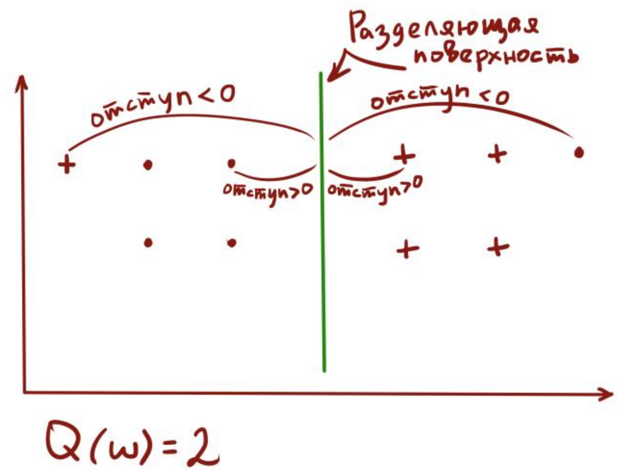
Сравнение методов



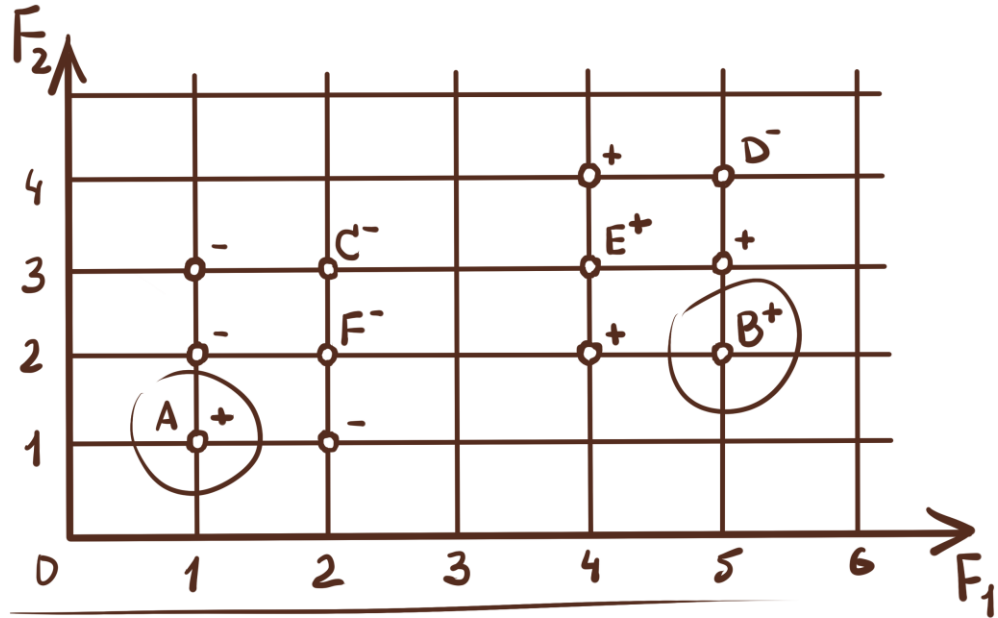
Отбор эталонных объектов

$$M(x_i) = \Gamma_{y_i}(x_i) - \max_{y \in Y \setminus y_i} \Gamma_y(x_i)$$





Пример оценки отступа



A^+: 4, 5, 5, 6, 6

A^-: 1, 1, 2, 2, 3, 7

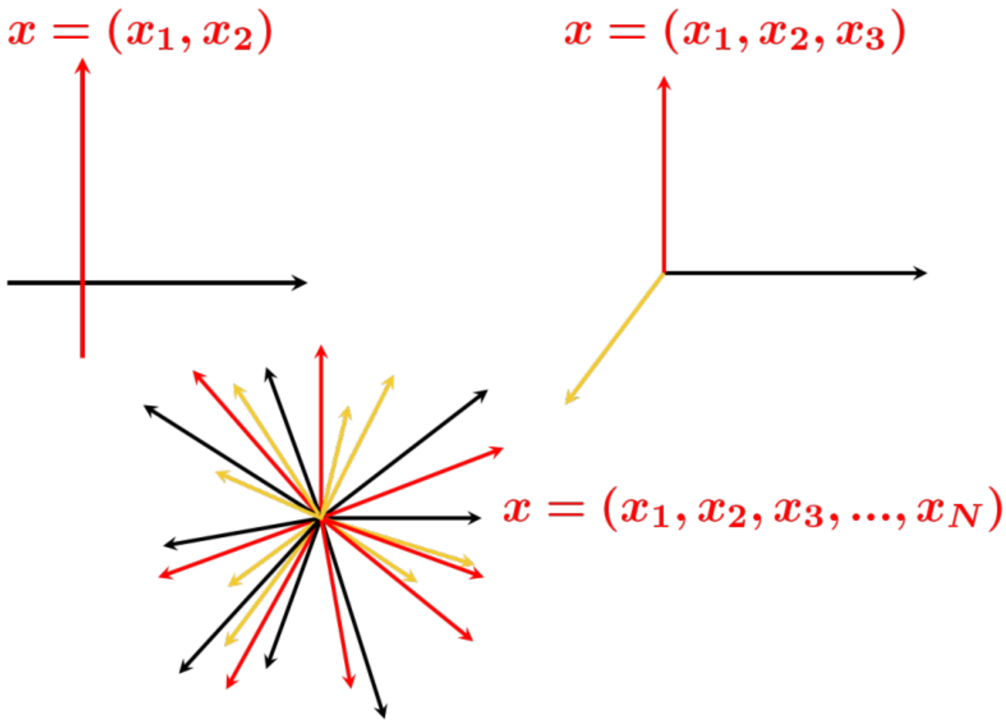
$$M(A) = \left(\frac{1}{4}+\frac{1}{5}+\frac{1}{5}+\frac{1}{6}+\frac{1}{6}\right)-\left(\frac{1}{1}-\frac{1}{1}-\frac{1}{2}-\frac{1}{2}-\frac{1}{3}-\frac{1}{7}\right) = -1,99$$

B^+: 1, 1, 2, 3, 5

B^-: 2, 3, 4, 4, 5, 6

$$M(B) = \left(\frac{1}{1}+\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\frac{1}{5}\right)-\left(\frac{1}{2}-\frac{1}{3}-\frac{1}{4}-\frac{1}{4}-\frac{1}{5}-\frac{1}{6}\right) = 1,33$$

Проклятие размерности

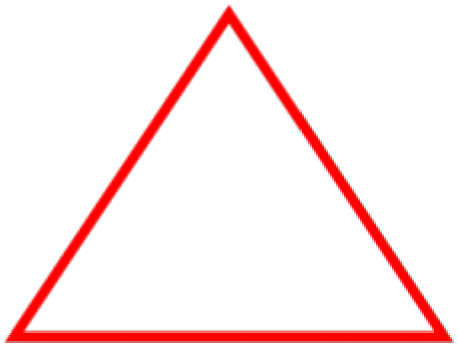


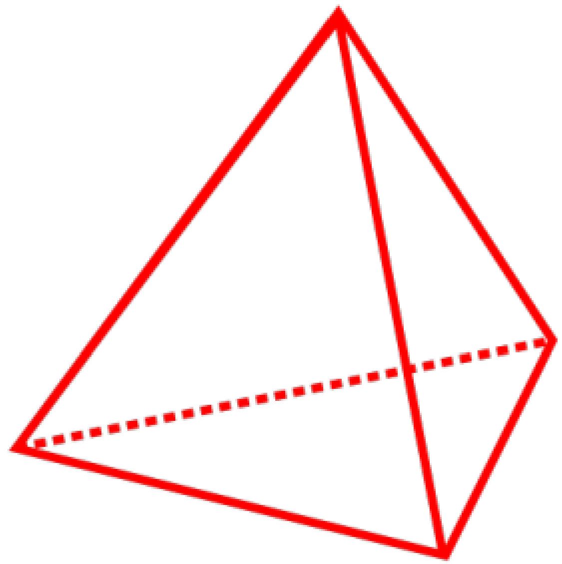
$$x_1 = (a_1, a_2, ..., a_N)$$

$$x_2 = (a_1 + \varepsilon, a_2 + \varepsilon, ..., a_N + \varepsilon)$$

$$x_3 = (a_1, a_2 + \Delta, a_3, ..., a_N)$$

Проклятие размерности



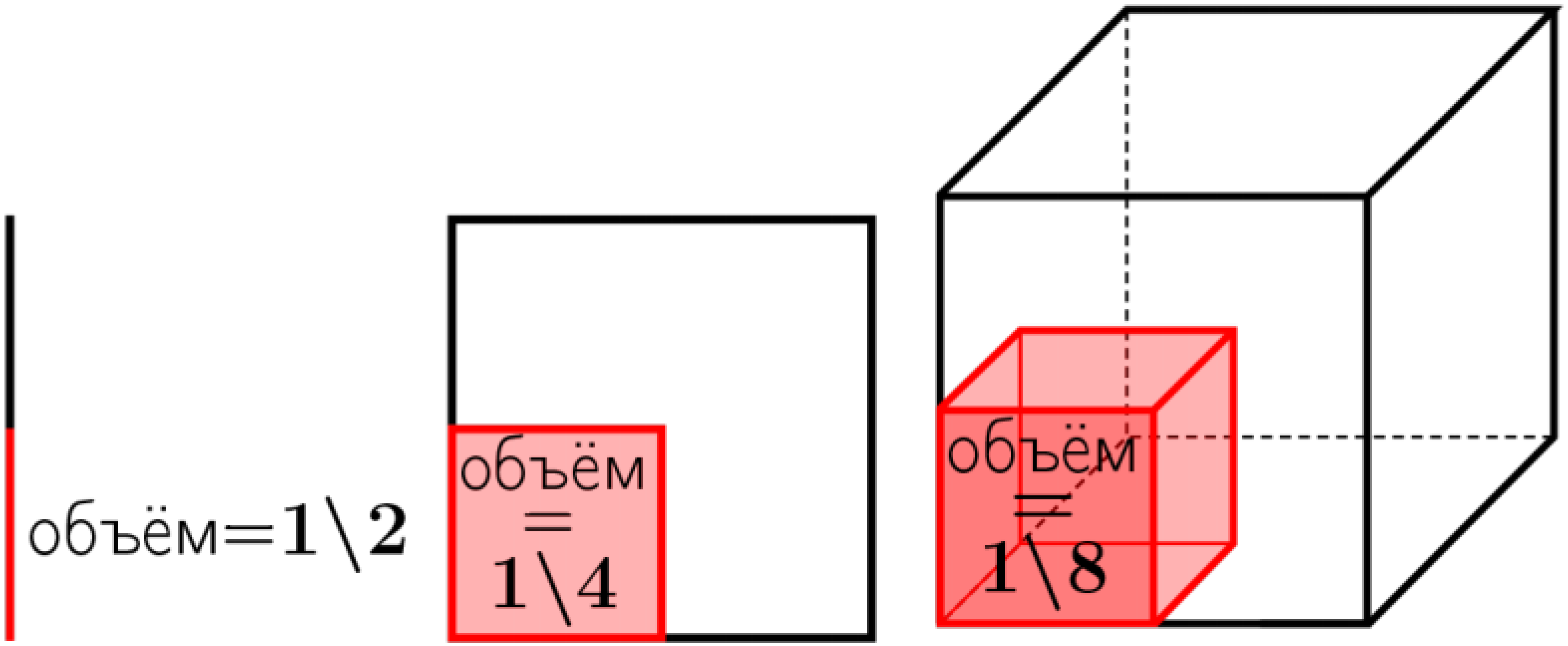


В N-мерном пространстве — до $N+1$ равноудалённой точки

Проклятие размерности

- Рассмотрим векторы размерности N из бинарных признаков:
$$X = (0, 0, 1, 0, 1, 1, \ldots, 1)$$
- Всевозможных комбинаций значений признаков — 2^N
- С ростом N экспоненциально увеличивается необходимое количество данных

Попадания в куб



KNeighborsClassifier (реализация *k* ближайших соседей)

sklearn.neighbors.KNeighborsClassifier

class `sklearn.neighbors.` **KNeighborsClassifier** (*n_neighbors=5, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=1, **kwargs*) [source]

Classifier implementing the k-nearest neighbors vote.

Read more in the User Guide.

Parameters:

n_neighbors : int, optional (default = 5)

Number of neighbors to use by default for `kneighbors` queries.

weights : str or callable, optional (default = 'uniform')

weight function used in prediction. Possible values:

- 'uniform' : uniform weights. All points in each neighborhood are weighted equally.
- 'distance' : weight points by the inverse of their distance. in this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.
- [callable] : a user-defined function which accepts an array of distances, and returns an array of the same shape containing the weights.

algorithm : {'auto', 'ball_tree', 'kd_tree', 'brute'}, optional

Algorithm used to compute the nearest neighbors:

- 'ball_tree' will use `BallTree`
- 'kd_tree' will use `KDTree`
- 'brute' will use a brute-force search.
- 'auto' will attempt to decide the most appropriate algorithm based on the values passed to `fit` method.

leaf_size : int, optional (default = 30)

Leaf size passed to BallTree or KDTree. This can affect the speed of the construction and query, as well as the memory required to store the tree. The optimal value depends on the nature of the problem.

p : integer, optional (default = 2)

Power parameter for the Minkowski metric. When p = 1, this is equivalent to using manhattan_distance (l1), and euclidean_distance (l2) for p = 2. For arbitrary p, minkowski_distance (l_p) is used.

metric : string or callable, default 'minkowski'

the distance metric to use for the tree. The default metric is minkowski, and with p=2 is equivalent to the standard Euclidean metric. See the documentation of the DistanceMetric class for a list of available metrics.

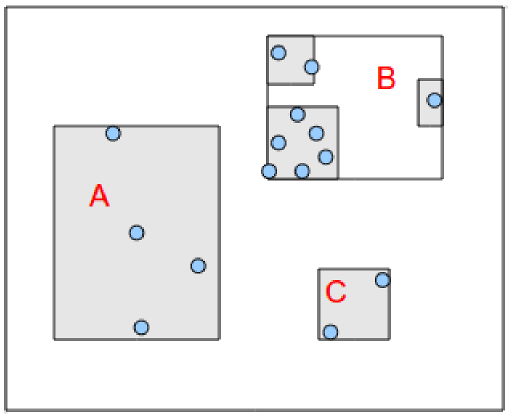
metric_params : dict, optional (default = None)

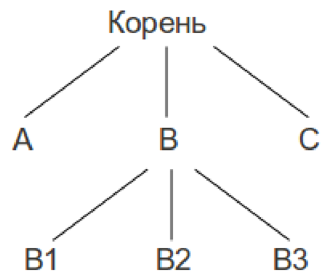
Additional keyword arguments for the metric function.

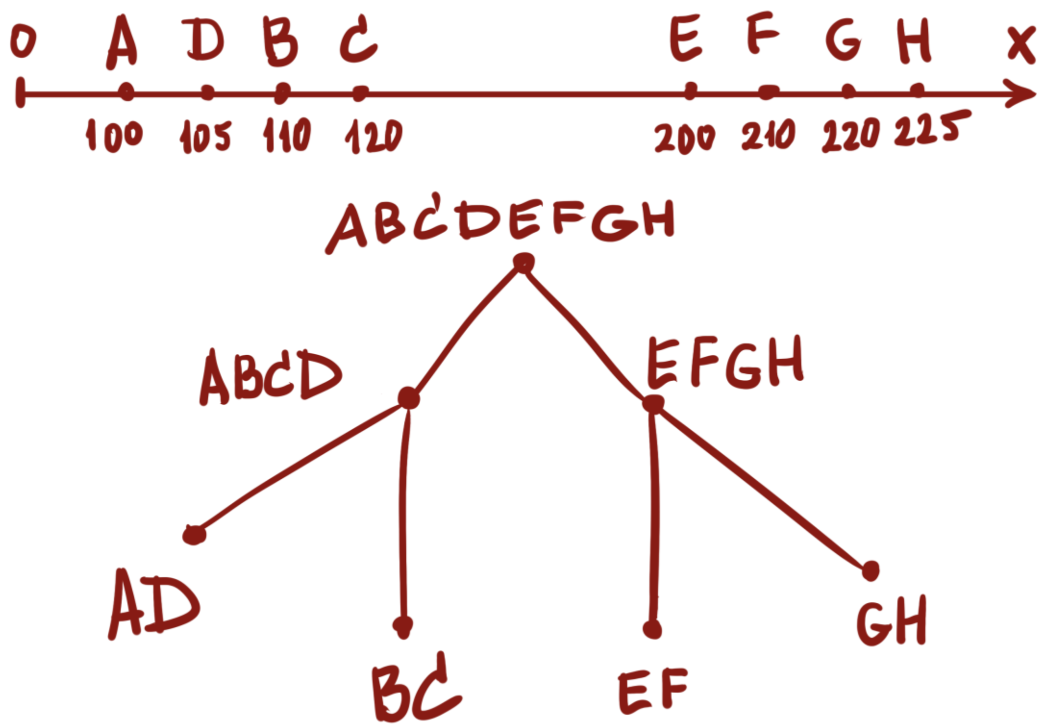
n_jobs : int, optional (default = 1)

The number of parallel jobs to run for neighbors search. If `-1`, then the number of jobs is set to the number of CPU cores. Doesn't affect `fit` method.

Дерево для поиска соседа







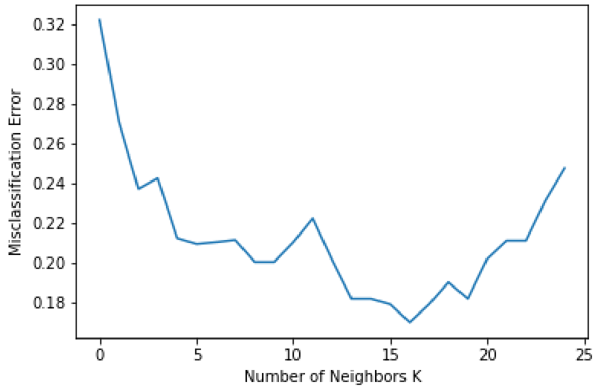
Пример 1

```
import numpy as np
import pylab as pl
import sklearn as sk
from sklearn import neighbors, datasets
from sklearn.cross_validation import train_test_split
from sklearn.model_selection import cross_val_score

# import some data to play with
iris = datasets.load_iris()
X = iris.data[:, :2] # we only take the first two features.
Y = iris.target

from sklearn.cross_validation import train_test_split

# we create an instance of Neighbours Classifier and fit the data.
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.33, random_state=42)

# instantiate learning model (k = 3)
knn = neighbors.KNeighborsClassifier(n_neighbors=3)

# fitting the model
knn.fit(X_train, y_train)

# predict the response
pred = knn.predict(X_test)

# evaluate accuracy
print("Accuracy:", sk.metrics.accuracy_score(y_test, pred))
print("Test labels:", y_test)
print("Predicted labels:", pred)
```

```
Accuracy: 0.76
Test labels: [1 0 2 1 1 0 1 2 1 1 2 0 0 0 0 1 2 1 1 2 0 2 0 2 2 2 2 2 0 0 0 0 1 0 0 2 1
 0 0 0 2 1 1 0 0 1 2 2 1 2]
Predicted labels: [1 0 2 1 1 0 1 2 1 2 2 0 0 0 0 2 2 1 1 1 0 1 0 1 2 2 1 2 0 0 0 0 2 0 0 1 2
 0 0 0 2 2 2 0 0 1 2 2 2 2]
```

Пример 2

```
import matplotlib.pyplot as plt


# creating odd list of K for KNN
myList = list(range(1,50))

# subsetting just the odd ones
neighbors_ = filter(lambda x: x % 2 != 0, myList)

# empty list that will hold cv scores
cv_scores = []

# perform 10-fold cross validation
for k in neighbors_:
    knn = neighbors.KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X_train, y_train, cv=10, scoring='accuracy')
    cv_scores.append(scores.mean())

# changing to misclassification error
MSE = [1 - x for x in cv_scores]

# plot misclassification error vs k
plt.plot(MSE)
plt.xlabel('Number of Neighbors K')
plt.ylabel('Misclassification Error')
plt.show()
```



RadiusNeighborsClassifier (реализация окна Парзена)

sklearn.neighbors.RadiusNeighborsClassifier

class `sklearn.neighbors.` **RadiusNeighborsClassifier** (*radius=1.0*, *weights='uniform'*, *algorithm='auto'*, *leaf_size=30*, *p=2*, *metric='minkowski'*, *outlier_label=None*, *metric_params=None*, ***kwargs*) [source]

Classifier implementing a vote among neighbors within a given radius

Read more in the User Guide.

Parameters:

radius : float, optional (default = 1.0)

Range of parameter space to use by default for `radius_neighbors` queries.

weights : str or callable

weight function used in prediction. Possible values:

- 'uniform' : uniform weights. All points in each neighborhood are weighted equally.
- 'distance' : weight points by the inverse of their distance. in this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.
- [callable] : a user-defined function which accepts an array of distances, and returns an array of the same shape containing the weights.

Uniform weights are used by default.

algorithm : {'auto', 'ball_tree', 'kd_tree', 'brute'}, optional

Algorithm used to compute the nearest neighbors:

- 'ball_tree' will use `BallTree`
- 'kd_tree' will use `KDTree`
- 'brute' will use a brute-force search.
- 'auto' will attempt to decide the most appropriate algorithm based on the values passed to `fit` method.

Note: fitting on sparse input will override the setting of this parameter, using brute force.

leaf_size : int, optional (default = 30)

Leaf size passed to BallTree or KDTree. This can affect the speed of the construction and query, as well as the memory required to store the tree. The optimal value depends on the nature of the problem.

p : integer, optional (default = 2)

Power parameter for the Minkowski metric. When p = 1, this is equivalent to using manhattan_distance (l1), and euclidean_distance (l2) for p = 2. For arbitrary p, minkowski_distance (l_p) is used.

metric : string or callable, default 'minkowski'

the distance metric to use for the tree. The default metric is minkowski, and with p=2 is equivalent to the standard Euclidean metric. See the documentation of the DistanceMetric class for a list of available metrics.

outlier_label : int, optional (default = None)

Label, which is given for outlier samples (samples with no neighbors on given radius). If set to None, ValueError is raised, when outlier is detected.

metric_params : dict, optional (default = None)

Additional keyword arguments for the metric function.

Ошибки классификации

- Ошибка первого рода
- Ошибка второго рода