

K-nearest neighbours optimization

Victor Kitov
v.v.kitov@yandex.ru

Yandex School of Data Analysis



Complexity of K-NN

- Complexity of training: no training needed!
- Complexity of prediction: $O(N)$
 - distance from x to all objects of training sample need to be calculated
- Variants to simplify search:
 - decrease the size of the training set
 - remove outliers (editing)
 - delete uninformative objects (condensing)
 - structurize feature space to accelerate search
 - KD-tree, ball-tree, LAESA.

Table of Contents

- 1 Reduction of training sample
- 2 Structuring of feature space

Margin

- Consider the training set: $(x_1, c_1), (x_2, c_2), \dots (x_N, c_N)$, where c_i - is the correct class for object x_i , and $\mathbf{C} = \{1, 2, \dots C\}$ - the set of possible classes.
- Define margin:

$$M(x_i, c_i) = g_{c_i}(x_i) - \max_{c \in \mathbf{C} \setminus \{c_i\}} g_c(x_i)$$

- margin is negative \Leftrightarrow object x_i was incorrectly classified
- the value of margin shows the preference of algorithm to assign x_i the correct class c_i compared to other classes

Removal of outliers (editing)

Main idea

Outliers are objects that deviate significantly from model's expectations. They can be defined as

$$\{x_i : M(x_i, c_i) < -\delta\}$$

for sufficiently large $\delta > 0$.

In editing we remove such observations.

Uninformative observations

Main idea

Uninformative observations do not contribute to class information when they are accounted for.

- Usually the majority of observations are uninformative for datasets of large size with simple class separating curve.
- Condensing - removal of uninformative observations.
- Editing should precede condensing, otherwise all outliers will be included.

Condensing algorithm

Listing 1: Removal of uninformative observations

```
for each class  $c = 1, 2, \dots, C$ : # add the most  
     $x(c) = \arg \max_{x_i: c_i=c} \{M(x_i, c_i)\}$  # representative example  
Initialize etalons:  $\Omega = \{x(c), c = 1, 2, \dots, C\}$   
  
repeat while accuracy significantly increases:  
     $x_i = \arg \min_{x_i \in TS \setminus \Omega} M(x, \Omega)$  # add object  
     $\Omega = \Omega \cup x_i$  # with smallest margin  
  
return  $\Omega$ 
```

Table of Contents

- 1 Reduction of training sample
- 2 Structuring of feature space

Structuring of feature space

- Hierarchical arrangement of space through a sequence of simple nested figures
 - KD-trees - nested rectangles
 - Ball-trees - nested balls
- Comments:
 - **K-NN stops being online**, because the space structure needs to be recalculated as new observations arrive
 - distance metric should **satisfy triangle inequality**:
$$\forall x_1, x_2, z : \rho(x_1, x_2) \leq \rho(x_1, z) + \rho(z, x_2)$$

KD-tree

- Tree construction uses the following recursive function:

```
build_node( $\Omega$ ):  
    if  $|\Omega| < n_{min}$ :  
        return node with assigned objects  $\Omega$   
    else:  
        find feature with maximal spread in  $\Omega$ :  
         $x^j = \arg \max_{x^i} \sigma(x^i)$   
        find median  $\Omega$ :  $\mu = \text{median}\{x^j\}$   
        return inner node with two child-nodes:  
        left child =  
            build_node( $x^j, < \mu, \{x_k \in \Omega : x_k^j < \mu\}$ )  
        right child =  
            build_node( $x^j, \geq \mu, \{x_k \in \Omega : x_k^j \geq \mu\}$ )
```

KD-tree: finding distance from x to rectangle

- Distance from $x = [x^1, \dots, x^D]^T$ to rectangle $\{(h_1, \dots, h_D) : h_d^{\min} \leq h_d \leq h_d^{\max}\}$ equals to $\rho(x, z)$, where z - is the closest to x point on the rectangle with the following coordinates:

$$z^d = \begin{cases} h_d^{\min} & x^d < h_d^{\min} \\ x^d & h_d^{\min} \leq x^d \leq h_d^{\max} \\ h_d^{\max} & x^d > h_d^{\max} \end{cases}$$

- Tree depth:
 - Best case: $\lceil \log_2 N \rceil$
 - Worst case: N

Ball trees

- Nested sequence of balls
- Nesting is not in geometrical sense. It means that parent ball contains all objects contained in its child balls
- Each object from the parent ball is associated with single child ball.
- Characteristics of each ball:
 - center c
 - objects, associated with ball z_1, z_2, \dots, z_K
 - radius $R = \max_i \|z_i - c\|$

Ball trees: recursive generation

- for parent ball $Ball(c, \Omega)$ (with center c and associated objects Ω):
 - select $c_1 = \arg \max_{z_i \in \Omega} \|z_i - c\|$
 - select $c_2 = \arg \max_{z_i \in \Omega} \|z_i - c_1\|$
 - divide Ω into two groups:

$$\Omega_1 = \{z_i : \|z_i - c_1\| < \|z_i - c_2\|\}$$

$$\Omega_2 = \{z_i : \|z_i - c_2\| < \|z_i - c_1\|\}$$

- set for $Ball(c, \Omega)$ two child balls $Ball(c_1, \Omega_1)$ and $Ball(c_2, \Omega_2)$.

Minimum distance from x to $B = \text{Ball}(c, R)$

From triangle inequality for every $z \in B$:

$$\rho(x, c) \leq \rho(x, z) + \rho(z, c)$$

It follows that

$$\rho(x, z) \geq \rho(x, c) - \rho(z, c) \geq \rho(x, c) - R$$

Comments

- For application of KD-tree и ball-tree distance metric $\rho(x, z)$ should satisfy axioms for distance.
- Algorithm can be extended to find K nearest neighbours instead of one (maintaining a queue).
- The larger is D , the less efficient is feature space structuring:
 - its purpose is to split objects into geometrically compact groups
 - and for large D almost all objects become equally distant from each other
 - for example in KD-tree closeness in one coordinate does not guarantee general closeness of objects
- For large D ball-trees are more efficient than KD-trees, because balls are more compact figures than rectangles and give more tight lower bounds to contained objects.