

# K-nearest neighbours optimization

Victor Kitov  
v.v.kitov@yandex.ru

**Skoltech**

Skolkovo Institute of Science and Technology

November-December 2015.

# Parzen window method

- Parzen window method:

$$\hat{f}(x) = \arg \max_{y \in Y} \sum_{n=1}^N \mathbb{I}[y_n = y] K \left( \frac{\rho(x, x_n)}{h(x)} \right)$$

- Selection of  $h(x)$ :
  - $h(x) = \text{const}$
  - $h(x) = \rho(x, z_K)$ , where  $z_K$  -  $K$ -th nearest neighbour.
    - better for unequal distribution of objects
    - for  $K(z) = \mathbb{I}[z \leq 1]$  and  $h(x) = \rho(x, z_K)$  leads to  $K$ -NN.

# 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  $\delta > 0$ .

### Listing 1: Removal of outliers

```
for each  $x_i, c_i$  in the training set  $TS$ :  
    calculate  $M(x_i, c_i)$ 
```

```
return filtered training set:  
     $\{(x_i, c_i) : M(x_i, c_i) \geq -\delta\}$ 
```

Several iterations of algorithm may be needed.

# Removal of uninformative observations (condensing)

## 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.
- Editing should precede condensing, otherwise all outliers will be included.

### Listing 2: 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:
     $x_i = \arg \min_{x_i \in T \setminus \Omega} M(x, \Omega)$  # 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  $\Omega$ :  
         $x^j = \arg \max_{x^j} \sigma(x^j)$   
        find median  $\Omega$ :  $\mu = \text{median}\{x^j\}$   
        return inner node with two child-nodes:  
        left child =  
            build_node( $x^j, < \mu, \{x_i \in \Omega : x_i^j < \mu\}$ )  
        right child =  
            build_node( $x^j, \geq \mu, \{x_i \in \Omega : x_i^j \geq \mu\}$ )
```

# KD-tree

- complexity  $O(DN \log_2 N)$ , because on level  $h$  with  $k$  inner nodes:
  - $|\Omega_1| = n_1, \dots, |\Omega_k| = n_k$ , so total number of objects  
 $n_1 + n_2 + \dots + n_k \leq N$
  - need to calculate  $\sigma$  for  $D$  features - complexity  
 $D(n_1 + \dots + n_k) \leq DN$
- procedure is repeated for  $h = 1, 2, \dots, H$ ,  $H \leq \lceil \log_2 N \rceil$ , because each time the number of objects in a node is divided in half (so the tree is balanced)
- geometrically it is better to take mean instead of median, but tree may become unbalanced.

## Nearest neighbour search using KD-tree

Step 1: For object of interest  $x$  find the leaf of tree, to which it belongs, then find initial estimate of nearest neighbour:

```
CURRENT_NODE  $\leftarrow$  root node of TREE
while CURRENT_NODE is not leaf node:
     $x^i \leftarrow$  discriminative feature of CURRENT_NODE
     $\mu \leftarrow$  threshold  $\mu$  of current_node
    if  $x^i \leq \mu$ :
        CURRENT_NODE  $\leftarrow$  left child of CURRENT_NODE
    else:
        CURRENT_NODE  $\leftarrow$  right child of CURRENT_NODE

NN  $\leftarrow$  closest object to  $x$  from all objects associated
    with the leaf node.
NN_DIST  $\leftarrow$  distance from  $x$  to NN.
```

## Nearest neighbour search using KD-tree

### Step 2: Ascending search

```
mark CURRENT_NODE as checked
while not all nodes of TREE checked:
    PARENT_NODE  $\leftarrow$  parent node of CURRENT_NODE
    for each NODE of PARENT_NODE except checked nodes:
        RECT_DIST  $\leftarrow$  distance from  $x$  to rectangle,
                        associated with NODE
        if RECT_DIST  $\geq$  NN_DIST:
            mark NODE and all its descendants as checked
        else:
            NN, NN_DIST = check_tree(NODE)
```

# Nearest neighbour search using KD-tree

Utility function, making descending search:

```
function check_tree(CURRENT_NODE,  $x$ , NN, NN_DIST):  
    if CURRENT_NODE is leaf node:  
        CURRENT_NN  $\leftarrow$  closest object to  $x$  from all objects  
                        associated with CURRENT_NODE.  
        CURRENT_NN_DIST  $\leftarrow$  distance from  $x$  to CURRENT_NN.  
        if CURRENT_NN_DIST < NN_DIST:  
            NN  $\leftarrow$  CURRENT_NN  
            NN_DIST  $\leftarrow$  CURRENT_NN_DIST  
        return NN, NN_DIST  
    else:  
        for each NODE from children of CURRENT_NODE:  
            DIST  $\leftarrow$  distance from  $x$  to rectangle of CURRENT_NODE  
            if NN_DIST  $\geq$  DIST:  
                mark NODE and all its descendants as checked  
            else:  
                NN, NN_DIST = check_tree(NODE,  $x$ , NN, NN_DIST)
```

## 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  $\Omega$ ):
  - 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  $\Omega$  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 triangle inequality.
- 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.