K-nearest neighbours optimization

Victor Kitov
v.v.kitov@yandex.ru

Complexity of K-NN

- Complexity of training: no training needed!
- Complexity of prediction: $O(ND)$
    - distance from x to all objects of training sample need to be calculated

Complexity of K-NN

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

Table of Contents

Margin

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

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

  - margin is negative <=> 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

Editing: removal of outliers

Main idea

Filter outliers (objects that deviate significantly from model's expectations). These are points having margin below threshold

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

for some $\delta < 0$.

Several iterations of algorithm may be needed.

Condensing: removal of uninformative observations[1]

Main idea

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

Listing 1: Removal of uninformative observations

```
for each class c = 1, 2, ...C: # add the most
    x(c) = arg max_{x_i : c_i = c} {M(x_i, c_i)}     # representative example
Initialize etalons: Ω = {x(c), c = 1, 2, ...C}

repeat while accuracy significantly increases:
    x_i = arg min_{x_i ∈ TS\Ω} M(x, Ω) # add object
    Ω = Ω ∪ x_i                          # with smallest margin

return Ω
```

[1] Is it better to apply first editing then condensing or vice versa?

Table of Contents

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**: $\forall x_1, x_2, z: \ \rho(x_1, x_2) \le \rho(x_1, z) + \rho(z, x_2)$

KD-tree construction[2]

- Tree construction uses the following recursive function:

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

- Geometrically it is better to take mean instead of median, but tree may become unbalanced.

[2] Estimate complexity of KD-tree construction using median splits.

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 ← root node of TREE
while CURRENT_NODE is not leaf node:
   x^i ← discriminative feature of CURRENT_NODE
   μ ← threshold μ of current_node
   if x^i ≤ μ:
      CURRENT_NODE ← left child of CURRENT_NODE
   else:
      CURRENT_NODE ← right child of CURRENT_NODE

NN ← closest object to x from all objects associated
      with the leaf node.
NN_DIST ← 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 ← parent node of CURRENT_NODE
   for each NODE of PARENT_NODE except checked nodes:
      RECT_DIST ← distance from x to rectangle,
                  associated with NODE
      if RECT_DIST ≥ 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 ← closest object to x from all objects
                   associated with CURRENT_NODE.
      CURRENT_NN_DIST ← distance from x to CURRENT_NN.
      if CURRENT_NN_DIST < NN_DIST:
         NN ← CURRENT_NN
         NN_DIST ← CURRENT_NN_DIST
      return NN,NN_DIST
   else:
      for each NODE from children of CURRENT_NODE:
         DIST ← distnace from x to rectangle of CURRENT_NODE
         if NN_DIST ≥ 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 $x = [x^1, ...x^D]^T$ to rectangle $\{(h_1, ...h_D) : h_d^{min} \le h_d \le h_d^{max}\}$ equals to $\rho(x, z)$, where z - is the closest to x point on the rectangle with the following coordinates:

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

- 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 $z_1, z_2, ...z_K$
    - 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 $B = Ball(c, R)$

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

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

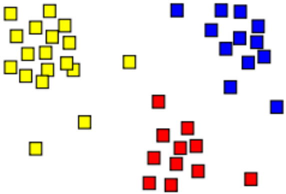
It follows that

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

Alternative optimization - clustering

- Clustering
  - cluster points into clusters
  - find cluster centers and radii
  - for new x find closest clusters and search NN only in them

Clustering:



Alternative optimization - LSH

- Idea of locality semantic hashing:
  - suppose we have hash function $h_\theta(x)$ such that $P(h_\theta(x) = h_\theta(z))$ approaches 1 when x and z become similar.
  - we sample L parameters $\theta_1, ...\theta_L$ and obtain L hash functions $H(x) = [h_{\theta_1}(x), ...h_{\theta_L}(x)]$
  - a bucket $H = [h_1, ...h_L]$ is a set $\{x : H(x) = H\}$
  - group training set $x_1, ...x_n$ into buckets.
  - for new x we find its bucket $H(x) = [h_{\theta_1}(x), ...h_{\theta_L}(x)]$ and search nearest neighbours only among similar buckets (having most of h_i the same).
  - we get nearest neighbours with probability (increasing with L).

Comments

- For 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 tighter lower bounds to contained objects.