Chap03-kNN

I. 算法（k-nearest neighbor, k-NN）

输入：训练数据集 $T=\{(x_1,y_1)\cdots(x_N,y_N)\}$

$x_i\in X\subseteq R^n$，$y_i\in Y=\{c_1,c_2,\cdots c_K\}$

输出：x 所属的类 y

(1) 根据给定的距离度量，在训练集 T 中找出与 x 最近邻的 k 个点，涵盖这 k 个点的 x 的邻域记为 $N_k(x)$

(2) 在 $N_k(x)$ 中根据分类决策规则（如多数表决）决定 x 的类别 y：

$$y=\arg\max_{c_j}\sum_{x_i\in N_k(x)} I(y_i=c_j),\quad i=1,\cdots N;\ j=1,2,\cdots K$$

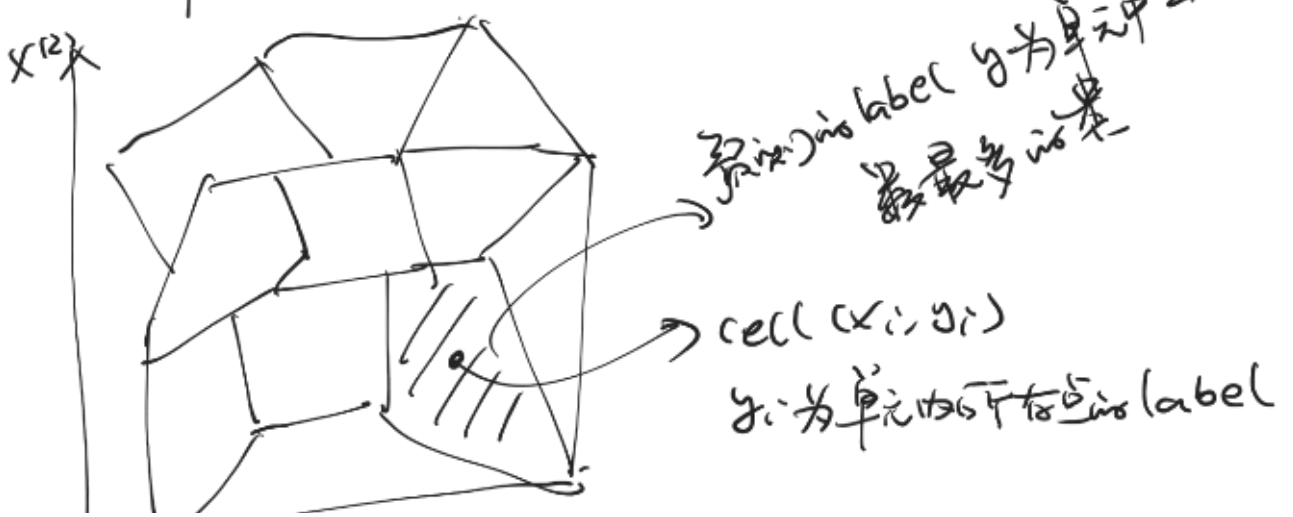
$$I(y_i=c_j)=\begin{cases}1 & y_i=c_j\\ 0 & \text{else}\end{cases}$$

$k=1$ 时，最近邻算法

KNN 没有显式的学习过程？

II. K-NN 模型

1. 模型



2. 距离度量 L_p-norms or Minkowski-distance

$x_i, x_j \in R^n$

$$L_p(x_i, x_j) = \left(\sum_{l=1}^{n} |x_i^{(l)} - x_j^{(l)}|^p \right)^{1/p}$$

$$\begin{cases} L_2 = \left(\sum_{l=1}^{n} (x_i^{(l)} - x_j^{(l)})^2 \right)^{1/2} \\ L_1 = \sum_{l=1}^{n} |x_i^{(l)} - x_j^{(l)}| \\ L_\infty = \max_l |x_i^{(l)} - x_j^{(l)}| \end{cases}$$

由不同的距离度量所确定的最近邻点不同.

3. K值的选择

K ↓ → 训练 误差 ↓ → 预测误差 ↑

K ↑ → ~ --- ↑ → --- ↓

K <=> 模型复杂度; K值减小 => 容易 over-fitting

K值增加 => 模型更简单

在应用中, K值一般取一个比较小的数值, 然后通过交叉验证法来选取最优的K值.

4. 分类的决策规则)

多数表决：$N_k(x)$中的多数类 $\Rightarrow$ x的类 y

⑵ Loss function

$N_k(x)$中的实例被误分类的概率为

$$p = \frac{1}{k}\sum_{x_i \in N_k(x)} I(y_i \neq c_j)$$

to Min

$$= \frac{1}{k}\sum_{x_i \in N_k(x)} [1 - I(y_i = c_j)]$$

argmax

$$= 1 - \frac{1}{k}\sum_{x_i \in N_k(x)} I(y_i = c_j)$$

Ⅲ. kd tree

kNN关键是如何进行k近邻搜索.

Linear Scan $= N + N-1 + \cdots + 1 \propto O(N^2)$

⑴ 构造平衡的 kd 树 伪代码

Input: k-维空间数据集 $T = \{x_1, x_2, \cdots x_N\}$

$x_i = (x_i^{(1)}, x_i^{(2)}, \cdots x_i^{(k)})$, $i = 1, \cdots N$

Output: kd-Tree

```
SPLIT(T, Node):
    if T.shape[0] = 1
        Node->val = T
        return
    j = Node -> depth
    l = j%(mod k) + 1
```

use partition to split X into
(L , $X^{(k)}_{median}$, R) → O(n)

Node -> Val = $X^{(k)}_{median}$
if T.shape[0] = 2 use larger one as median
    Node -> left = LNode
    LNode -> Val = L
    LNode -> depth = d+1
    return

Node -> left = LNode
Node -> right = RNode
LNode -> depth = d+1
RNode -> depth = d+1
SPLIT (L , LNode)
SPLIT (R , RNode)

BUILD-KD-TREE (T , root) =
    SPLIT (T , root) O(nlgn)

2. KD 树的最近邻搜索：

https://zhuanlan.zhihu.com/p/23966698

Find-Bottom (X, root) :
    temp = root , dim = X.size

```
while temp not None:
    p = temp
    key = temp->depth (mod dim) + 1
    if X[key] < temp->val[key]
        temp = temp->left
    else temp = temp->right
return p
```

```
k_list = [ ], search_list = [ ]
```

```
def nearest_search ( lowest_node, k):
    current_node = lowest_node
    if current_node not in search_list[ ]:
        search_list.append(current_node)
    if k_list.size < k:
        k_list.append(current_node)
    elif D(X, current_node->val) < Max(k_list):
        replace (k_list, current_node)

    while current_node not root:
        temp = current_node->parent
        if temp not in search_list:
            search_list.append(temp)
            if k_list.size < k:
                k_list.append(temp)
            elif D(X,... ,< M ...,
```

```
            replace(k_list, temp)
    key = temp->depth(mod dim) + 1
    sd = abs(temp->val(key) - X[key])
    if sd < max(k_list) or
        k_list < k :
        if current_node = temp->left
          and temp->right not None:
        low = Find-Bottom(X, temp.r)
        elif current_node = temp->right
         and temp->left not None:
         low = F...(x, temp.left)
         k-nearest(X, low, k)

    else: current = current->parent

else = current = current->parent
```

上次修改：02:19