

算法 3.1 (k 近邻法)

输入：训练数据集

$$T = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}$$

其中, $x_i \in \mathcal{X} \subseteq \mathbb{R}^n$ 为实例的特征向量, $y_i \in \mathcal{Y} = \{c_1, c_2, \dots, c_K\}$ 为实例的类别, $i = 1, 2, \dots, N$; 实例特征向量 x ;

输出：实例 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 最邻近的 k 个点, 涵盖这 k 个点的 x 的邻域记作 $N_k(x)$;

(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 ;

$$y = \arg \max_{c_j \in \mathcal{Y}_k(x)} I(y_i = c_j), \quad i = 1, 2, \dots, N; \quad j = 1, 2, \dots, K \quad (3.1)$$

式 (3.1) 中, I 为指示函数, 即当 $y_i = c_j$ 时 I 为 1, 否则 I 为 0。

k 近邻法的特殊情况是 $k=1$ 的情形, 称为最近邻算法。对于输入的实例点 (特征向量) x , 最近邻法将训练数据集中与 x 最近邻点的类作为 x 的类。

算法 3.3 (用 kd 树的最近邻搜索)

输入：已构造的 kd 树; 目标点 x ;

输出： x 的最近邻。

(1) 在 kd 树中找出包含目标点 x 的叶结点：从根结点出发，递归地向下访问 kd 树。若目标点 x 当前维的坐标小于切分点的坐标，则移动到左子结点，否则移动到右子结点。直到子结点为叶结点为止。

(2) 以此叶结点为“当前最近点”。

(3) 递归地向上回退，在每个结点进行以下操作：

(a) 如果该结点保存的实例点比当前最近点距离目标点更近，则以该实例点为“当前最近点”。

(b) 当前最近点一定存在于该结点一个子结点对应的区域。检查该子结点的父结点的另一子结点对应的区域是否有更近的点。具体地，检查另一子结点对应的区域是否与以目标点为球心、以目标点与“当前最近点”间的距离为半径的超球体相交。

如果相交，可能在另一个子结点对应的区域内存在距目标点更近的点，移动到另一个子结点。接着，递归地进行最近邻搜索；

如果不相交，向上回退。

(4) 当回退到根结点时，搜索结束。最后的“当前最近点”即为 x 的最近邻点。

如果实例点是随机分布的，kd 树搜索的平均计算复杂度是 $O(\log N)$ ，这里 N 是训练实例数。kd 树更适用于训练实例数远大于空间维数时的 k 近邻搜索。当空间维数接近训练实例数时，它的效率会迅速下降，几乎接近线性扫描。

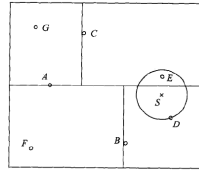


图 3.5 通过 kd 树搜索最近邻

算法：构建 k-树 (createKdTree)

输入：数据集 Data-set 和其所在的空间 Range

输出：Kd，类型为 k-d tree

1. If Data-set 为空，则返回空的 k-d tree

2. 调用节点生成程序：

(1) 确定 split 轴：对于所有描述子数据 (特征向量)，统计它们在每个维上的数据方差。假设每条数据记录为 64 维，可计算 64 个方差。挑选出最大值，对应的维就是 split 轴的值。数据方差大表明沿该坐标轴方向上的数据分散得比较开，在这个方向上进行数据分割有较好的分辨率；

(2) 确定 Node-data 轴：数据集 Data-set 按其第 split 轴的值排序。位于正中间的那个数据点被选为 Node-data。此时新的 Data-set' = Data-set \ Node-data (除去其中 Node-data 这一点)。

3. dataleft = {d 属于 Data-set' && d[split] ≤ Node-data[split]}

Left_Range = (Range && dataleft)

dataright = {d 属于 Data-set' && d[split] > Node-data[split]}

Right_Range = (Range && dataright)

4. left = 由 (dataleft, Left_Range) 建立的 k-d tree，即递归调用 createKdTree (dataleft, Left_Range) 并设置 left 的 parent 轴为 Kd;

right = 由 (dataright, Right_Range) 建立的 k-d tree，即调用 createKdTree (dataleft, Left_Range) 并设置 right 的 parent 轴为 Kd。

K近邻法(k-nearest neighbor,k-NN)

k近邻法

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

k近邻法定义：给定一个训练数据集，对输入的实例，在训练数据集中找到与实例最邻近

伪代码

算法

输入：Kd， //k-d tree类型

target //查询数据点

输出：nearest, //最近邻数据点

dist //最近邻数据点和查询点间的距离

1. If Kd为NULL，则返回dist为infinite并返回

2. //进行二叉查找，生成搜索路径

Kd_point = Kd.Kd //Kd_point中保存k-d tree根节点地址

nearest = Kd_point -> Node-data //初始化最近邻点

while (Kd_point)

push (Kd_point) 到search_path中; //search_path是一个堆栈结构，存储搜索路径节点指针

*** If Dist (nearest, target) > Dist (Kd_point -> Node-data, target)

nearest = Kd_point -> Node-data; //更新最近邻点

Max_dist = Dist (Kd_point, target); //更新最近邻点与查询点间的距离 ***

s = Kd_point -> split //确定待分裂的方向

If target[s] <= Kd_point -> Node-data[s] //进行二叉查找

Kd_point = Kd_point -> left

else

Kd_point = Kd_point -> right

nearest = search_path中最后一个叶子节点; //注意：二叉搜索时不计算选择搜索路径中的最近邻点，这部分已被注释

Max_dist = Dist (nearest, target); //直接取最后叶子节点作为回溯前的初始最近邻点

3. //回溯查找

while (search_path != NULL)

back_point = 从search_path取出一个节点指针; //从search_path堆栈弹栈

s = back_point -> split //确定分裂方向

If Dist (target[s], back_point -> Node-data[s]) < Max_dist //判断还需进入的子空间

If target[s] <= back_point -> Node-data[s]

Kd_point = back_point -> right; //如果target位于左子空间，就应进入右子空间

else

Kd_point = back_point -> left; //如果target位于右子空间，就应进入左子空间

将Kd_point压入search_path堆栈;

If Dist (nearest, target) > Dist (Kd_point -> Node-data, target)

nearest = Kd_point -> Node-data; //更新最近邻点

Min_dist = Dist (Kd_point -> Node-data, target); //更新最近邻点与查询点间的距离

伪代码