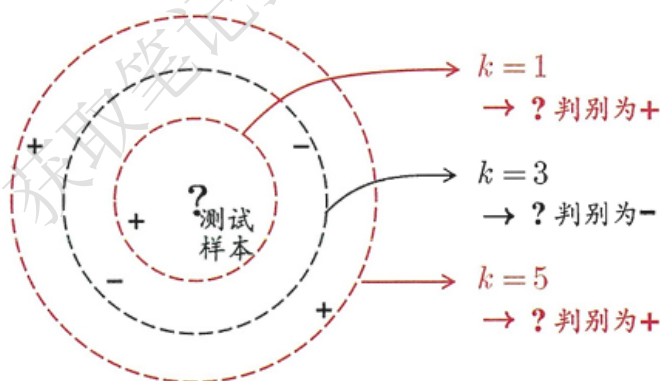


Lecture 10: KNN

KNN 算法基本概念

概念: KNN 算法基本概念

- 概念:
 - k 近邻 (k-Nearest Neighbor, 简称 KNN) 学习是一种常用的监督学习方法;
 - 给定测试样本, 基于某种距离度量、找出训练集中与其最靠近的 k 个训练样本, 然后基于这 k 个“邻居”的信息来进行预测;
 - 在分类任务中可使用“投票法”, 即将这 k 个样本中出现最多的类别标记作为预测结果;
 - 在回归任务中可使用“平均法”, 即将这 k 个样本的实值输出标记的平均值作为预测结果;
 - 还可基于距离远近、进行加权平均或加权投票, 距离越近的样本权重越大;
- 概念: 懒惰学习
 - KNN 算法是懒惰学习的代表, 这类学习技术仅仅是再训练阶段将样本保存起来, 训练时间开销为 0, 待收到测试样本后、再进行处理;
 - 相反的, 哪些在训练阶段就对样本就行学习、处理的方法, 称之为急切学习 (eager learning);
- 概念: KNN 模型
 - k 近邻法中, 当训练集、距离度量 (如欧氏距离)、k 值及分类决策规则 (如多数表决) 确定后, 对于任何一个新的输入实例, 它所属的类唯一地确定;
 - 这相当于根据上述要素, 将特征空间划分为一些子空间, 确定子空间里的每个点所属的类。这一事实从最近邻算法中可以看得很清楚;
- 示例:
 - k 近邻分类器示意图, 虚线显示出等距线: 测试样本在 $k=1$ 或 $k=5$ 时被判别为正例, $k=3$ 时被判别为反例;



- 特点:
 - 最近邻分类器虽简单, 但它的泛化错误率不超过贝叶斯最优分类器的错误率的两倍;

KNN 算法描述

概念: KNN 算法描述

- 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 ;

- 过程:

- (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

- (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 :

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

概念: 距离度量

- 概念:

- 特征空间中两个实例点的距离是两个实例点相似程度的反映;
- k 近邻模型的特征空间一般是 n 维实数向量空间 $\mathbb{R}^n$;
- 使用的距离是欧氏距离, 但也可以是其他距离, 如更一般的 L_p 距离或 Minkowski 距离 (Minkowski distance);

- 定义:

- 设特征空间 $\mathcal{X}$ 是 n 维实数向量空间 $\mathbb{R}^n$, $x_i, x_j \in \mathcal{X}$, $x_i = (x_i^{(1)}, x_i^{(2)}, \dots, x_i^{(n)})^T$
- $x_j = (x_j^{(1)}, x_j^{(2)}, \dots, x_j^{(n)})^T$, x_i, x_j 的 L_p 距离定义为:

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

- 概念: 不同的 P 取值

- 这里 $p \geq 1$ 。当 $p = 2$ 时, 称为欧氏距离 (Euclidean distance), 即:

- $$L_2(x_i, x_j) = \left(\sum_{l=1}^n |x_i^{(l)} - x_j^{(l)}|^2 \right)^{\frac{1}{2}}$$

- 当 $p = 1$ 时, 称为曼哈顿距离 (Manhattan distance), 即

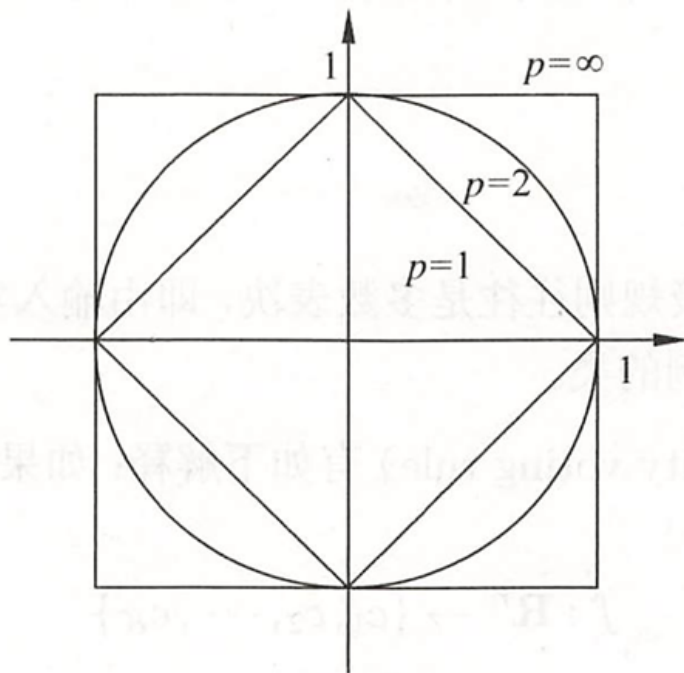
- $$L_1(x_i, x_j) = \sum_{l=1}^n |x_i^{(l)} - x_j^{(l)}|$$

- 当 $p = \infty$ 时, 它是各个坐标距离的最大值, 即

- $$L_\infty(x_i, x_j) = \max_l |x_i^{(l)} - x_j^{(l)}|$$

- 图示:

- 二维空间中 p 取不同值时, 与原点的距离为 1 ($L=1$) 的点的图形:



K 值的选择

概念： K 值的选择

- 概念：
 - 如果选择较小的 k 值，就相当于用较小的邻域中的训练实例进行预测，“学习”的近似误差 (approximation error) 会减小，只有与输入实例较近的 (相似的) 训练实例才会对预测结果起作用；
 - 但缺点是“学习”的估计误差 (estimation error) 会增大，预测结果会对近邻的实例点非常敏感；
 - 如果邻近的实例点恰巧是噪声，预测就会出错；
 - 如果选择较大的 k 值，就相当于用较大邻域中的训练实例进行预测；
 - 其优点是可以减少学习的估计误差，但缺点是学习的近似误差会增大；
 - 这时与输入实例较远的 (不相似的) 训练实例也会对预测起作用，使预测发生错误；
 - 如果 $k=N$ ，那么无论输入实例是什么，都将简单地预测它属于在训练实例中最多的类；
 - 这时，模型过于简单，完全忽略训练实例中的大量有用信息，是不可取的；
- 总结：
 - (1) k 值的减小就意味着整体模型变得复杂，容易发生过拟合；
 - (2) k 值的增大就意味着整体的模型变得简单；

分类决策规则

概念： 分类决策规则

- 概念：
 - k 近邻法中的分类决策规则往往是多数表决，即由输入实例的 k 个邻近的训练实例中的多数类决定输入实例的类；
- 分析：
 - 假设分类的损失函数是 0-1 损失函数，则分类函数为：

- $f: \mathbf{R}^n \rightarrow \{c_1, c_2, \dots, c_K\}$

- 误分类的概率:

- $P(Y \neq f(X)) = 1 - P(Y = f(X))$

- 概念: 误分类率

- 对给定的实例 $x \in \mathcal{X}$, 其最近邻的 k 个训练实例点构成集合 $N_k(x)$;
- 如果涵盖 $N_k(x)$ 的区域的类别是 c_j , 那么误分类率是:

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

KNN 的实现: kd 树

构造 kd 树

概念: 构造 kd 树

- 概念:

- kd 树是一种对 k 维空间中的实例点进行存储、以便对其进行快速检索的树形数据结构;
- kd 树是二叉树, 表示对 k 维空间的一个划分 (partition);
- 构造 kd 树相当于不断地用垂直于坐标轴的超平面将 k 维空间切分, 构成一系列的 k 维超矩形区域;
- kd 树的每个结点对应于一个 k 维超矩形区域;

- 实现: kd 树的构造

- (1) 构造根结点;
 - 构造根结点, 使根结点对应于 k 维空间中包含**所有实例点的超矩形区域**;
- (2) 通过下面的递归方法, 不断地对 k 维空间进行切分, 生成子结点;
 - 在超矩形区域 (结点) 上选择一个坐标轴和在此坐标轴上的一个切分点, 确定一个超平面, 这个超平面通过**选定的切分点并垂直于选定的坐标轴**, 将当前超矩形区域切分为**左右两个子区域** (子结点);
 - 这时, 实例被分到两个子区域;
- (3) 子区域内没有实例时终止
 - 持续上面递归过程, 这个过程直到子区域内没有实例时终止 (终止时的结点为叶结点);
 - 在此过程中, 将实例保存在相应的结点上;

- 示例: kd 树的构造

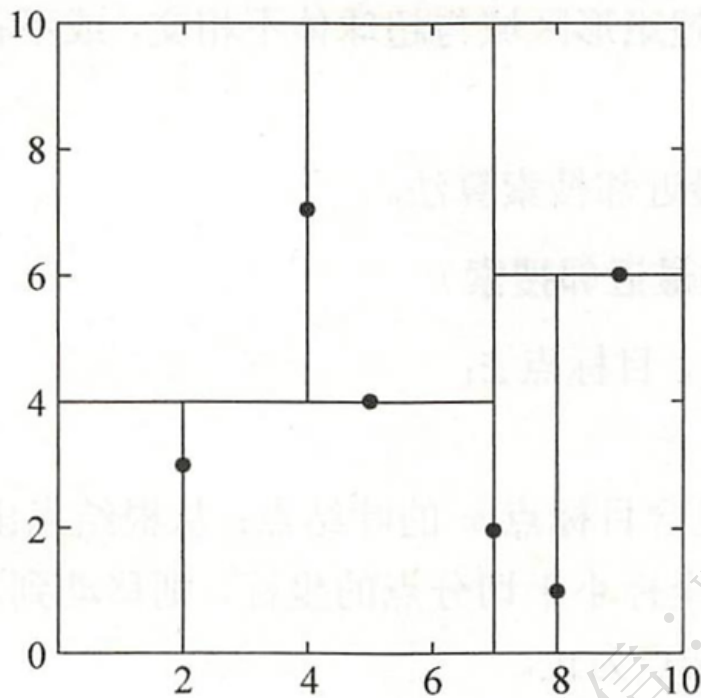
- 数据集:

- $$T = \{(2, 3)^T, (5, 4)^T, (9, 6)^T, (4, 7)^T, (8, 1)^T, (7, 2)^T\}$$

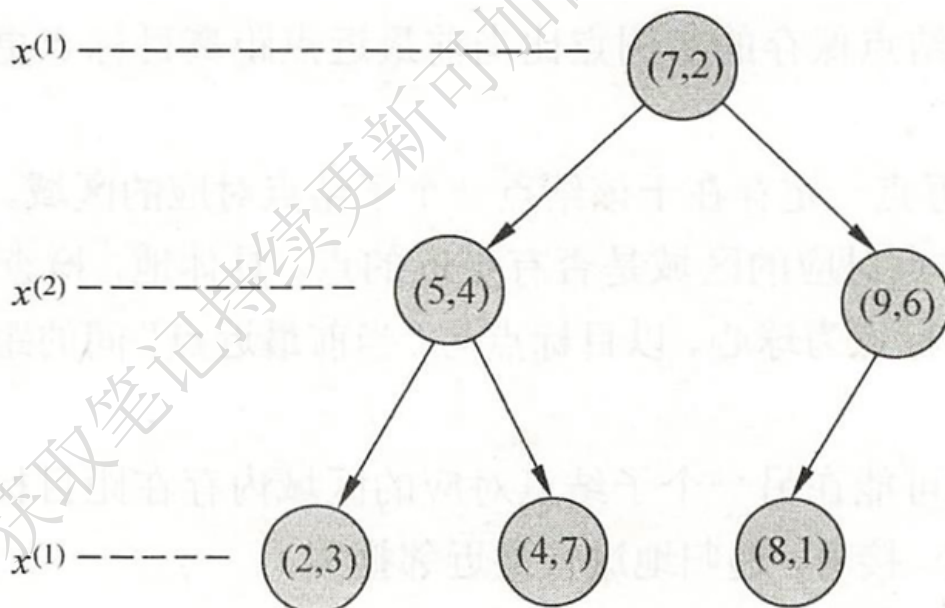
- 构造:

- (1) 根结点对应包含数据集 T 的矩形, 选择 $x^{(1)}$ 轴, 6 个数据点的 $x^{(1)}$ 坐标的中位数是 7, 以平面 $x^{(1)} = 7$ 将空间分为左、右两个子矩形 (子结点);

- (2) 左矩形以 $x^{(2)} = 4$ 分为两个子矩形，右矩形以 $x^{(2)} = 6$ 分为两个子矩形，如此递归；
- (3) 最后得到如图所示的特征空间划分 kd 树；
- 划分：



- **kd** 树：



搜索 kd 树

概念：搜索 kd 树

- 概念：
 - 给定一个目标点，搜索其最近邻。首先找到包含目标点的叶结点，然后从该叶结点出发，依次回退到父结点；
 - 不断查找与目标点最邻近的结点，当确定不可能存在更近的结点时终止；
 - 这样搜索就被限制在空间的局部区域上，效率大为提高；

- 包含目标点的叶结点对应包含目标点的最小超矩形区域。以此叶结点的实例点作为当前最近点。目标点的最近邻**一定在以目标点为中心并通过当前最近点的超球体的内部**；
- 然后返回当前结点的父结点，如果父结点的另一子结点的超矩形区域与超球体相交，那么在相交的区域内寻找与目标点更近的实例点；
- 如果存在这样的点，将此点作为新的当前最近点。算法转到更上一级的父结点，继续上述过程；
- 如果父结点的另一子结点的超矩形区域与超球体不相交，或不存在比当前最近点更近的点，则停止搜索；

• 算法：

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。

• 示例：

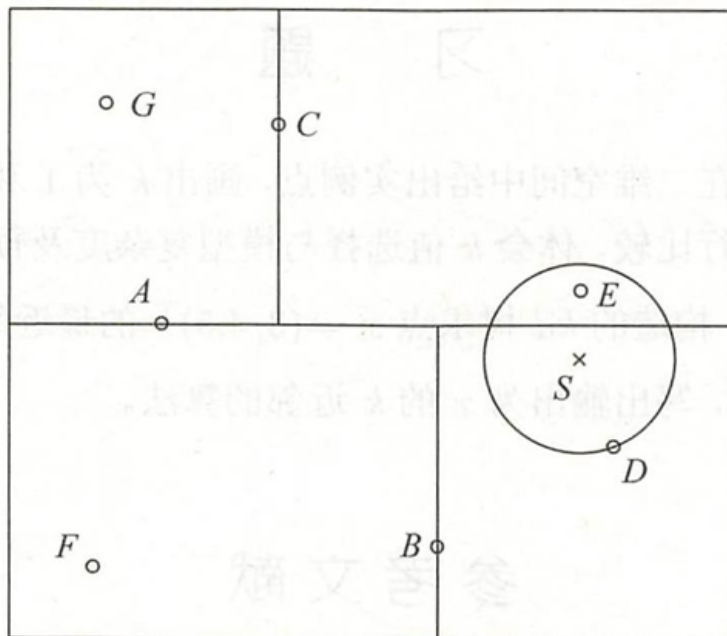
• 假设：

- 根节点为 A，其子节点分别为 B 和 C；

• 分析：

- 首先在 kd 树中找到包含点 S 的叶结点 D (图中的右下区域)，以点 D 作为近似最近邻。真正最近邻一定在以点 S 为中心通过点 D 的圆的内部。然后返回结点 D 的父结点 B，在结点 B 的另一子结点 F 的区域内搜索最近邻。结点 F 的区域与圆不相交，不可能有最近邻点。继续返回上一级父结点 A，在结点 A 的另一子结点 C 的区域内搜索最近邻。结点 C 的区域与圆相交；该区域在圆内的实例点有点 E，点 E 比点 D 更近，成为新的最近邻近点。最后得到点是点 S 的最近邻；

• 图示：



获取笔记持续更新可加微信: WandD2277