目录

插图

机器学习面试笔记- $\mathcal{K}$ 近邻算法

ShenHengheng

1 简介

$\mathcal{K}$ 近邻算法算法是基本的分类与回归算法,$\mathcal{K}$ 近邻算法的基本思路是这样的, 假定给定一个训练数据集, 数据集中的实例类别是已定的, 分类是, 对新的实例, 根据其最近的 $\mathcal{K}$ 个训练实例的类别, 通过**多数表决**的方法确定新的实例的类别. 因此 $\mathcal{K}$ 近邻算法不具有显式的学习过程, k 近邻算法实际上将训练数据的特征空间进行划分来作为其分类的" 模型", k 近邻算法的三个要素分别是: k **值的选择, 距离测度的选择和分类决策规则**,k 近邻算法由 1968 年由 Cover 提出.

本章主要由以下几部分组成: 首先讲解 k 近邻算法的基本概念与定义, 然后接下来讲解 k 近邻模型和三要素, 最后讲解 k 近邻算法的一个实现-kd 树.

2 k 近邻算法

算法: (k 近邻算法)

输入: 训练数据集

$$T = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), ..., (x_N, y_N)\}$$

其中输入空间 (特征空间) 是 $\mathcal{X} \subseteq \mathbf{R}^n$, 输出空间是 $\mathcal{Y} = \{c_1, c_2, c_3, ..., c_k\}$, 输入 $x \in \mathcal{X}$ 表示输入实例的特征向量, 对应于输入空间的点, 输出 $y \in \mathcal{Y}$ 表示实例的类别.

输出: 实例 x 的类别 y

算法描述:

(1) 根据给定的距离度量, 在训练集 T 中找出与 x 的最邻近的 k 个点, 涵盖这 k 个点的领域记作 $N_k(x)$.

(2) 在 $N_k(x)$ 中根据分类决策规则确定 x 的类别 y:

$$y = \arg\max_{c_j} \sum_{x_i \in N_k(x)} I(y_i = c_j), \quad i = 1, 2, 3, ..., N, j = 1, 2, 3, .., k. \tag{1}$$

其中 I 为指示函数, 即当 $y_i = c_j$ 时 I 为 1 否则为 0.

k 近邻算法的特殊情况: 当 k 为 1 时, 表示最近邻算法. 对于输入的实例 x, 最近邻算法使用 x 的最近邻的点的类别作为 x 的类别.

3 k 近邻模型

k 近邻模型实际对应特征空间的划分, k 近邻模型由三要素组成 - k **值的选择, 距离测度的选择和分类决策规则**.

3.1 模型

k 近邻模型中, 一旦 k **值的选择, 距离测度的选择和分类决策规则**确定, 那么对于任一输入的实例点 x 的类别也将唯一确定. 相当于将特征空间划分为很多自空间, 确定自空间所有输入实例点的类别.

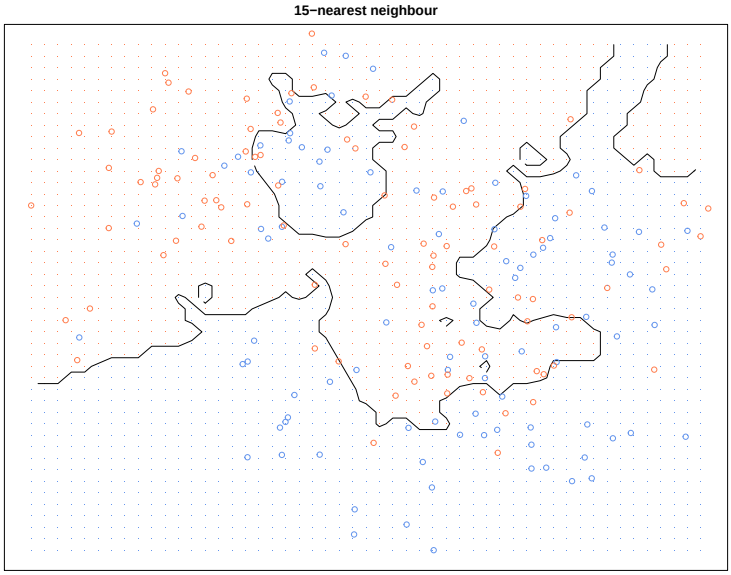


图 1: 15-近邻模型对应的特征空间的划分

3.2 距离度量

特征空间两点的距离是两个实例点相似度的反应, 一般情况下,k 近邻模型选择欧式距离作为距离度量, 但也可以其他距离, 比如更一般的 L_p 距离, Minkoski 距离, Manhattan 距离, Chebyshev 和 Hamming 距离.

设特征空间 $\mathcal{X}$ 是 n 维的实数向量空间 $\mathbf{R}^n$, 那么 $x_i, x_j \in \mathcal{X}, x_i = (x_{i1}, x_{i2}, ..., x_{in})$, 那么 x_i, x_j 的欧式距离定义如下:

$$d(x_i, x_j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + ... + (x_{in} - x_{jn})^2} \tag{2}$$

更一般地, 特征空间 $\mathcal{X}$ 是 n 维的实数向量空间 $\mathbf{R}^n$, x_i, x_j 的距离定义如下:

$$d_j(x_i, x_j) = ((\sum_{l=1}^{p} |x_i^{(l)} - x_j^{(l)}|^p)^{\frac{1}{p}} \tag{3}$$

当 $p = 2$ 时, 为欧式距离, 当 $p = 1$ 时, 为曼哈顿距离.

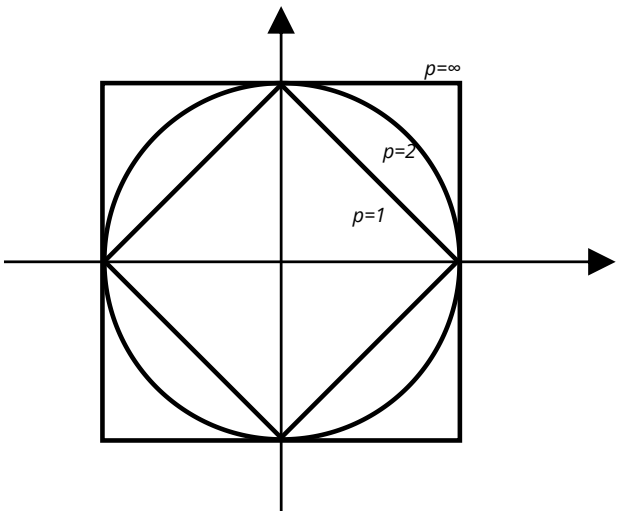


图 2: L_p 距离

3.3 k 值的选择

k 值的选择对于 k 近邻模型有很大影响. 如果 k 值选择的比较小, 那么将会造成过拟合, 近似误差减小, 只有与输入实例最邻近的训练实例才会对结果有作用. 估计误差增大, k 值减小也意味着模型变得复杂. 如果 k 值选择的比较大, 意味着模型的估计误差将会减小, 但是将会造成近似误差增大, 这时与输入实例较远的实例也会对预测起作用, 使得预测发生错误, k 值的减小将会造成模型变得简单.

若 $k = N$, 那么无论输入实例是什么, 将会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, 忽略了大量有用的信息.

在实际中, k 往往取较小值, 往往使用交叉验证的方式进行选取.

3.4 分类决策规则

分类决策规则为: 多数表决, 由输入的邻近 k 个训练样本的类别的多数表决决定输入实例的类别.

多数表决: 如果分类的损失函数为 0-1 损失函数, 分类函数为:

$$f : \mathbf{R}^n \to \{c_1, c_2, c_3, ..., c_k\}$$

那么误分类的概率为:

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

对给定的实例 $x \in \mathcal{X}$, 其最近的 k 个实例点构成的集合 $N_k(x)$, 如果涵盖 $N_k(x)$ 区域类别为 c_i, 那么误分类率为

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

要使损失函数最小, 即经验风险最小, 就要使 $\sum_{x_i \in N_k(x)} I(y_i = c_i)$ 最大, 所以多数表决规则等价于经验风险最小化.

4 k 近邻算法的一种实现 - kd 树

kd 树是用于存储 k 维空间的有限点集的数据结构。最近，S.Omohundro 在一项关于提高神经网络学习速度的可能技术的调查中推荐了它 [Omohundro，1987]。

实现 k 近邻时, 主要考虑的问题是如何快速的对训练数据进行快速 k 近邻搜索, 这与特征空间的维数大以及训练数据容量大时非常重要.

k 近邻最简单的实现方法就是线性扫描, 这时需要计算输入实例与每一个训练实例的距离, 当训练集很大时, 这个方法非常耗时, 这种方法是不可行的.

为了提高 k 近邻搜索的效率, 可以考虑 kd 树这种特殊的数据结构来存储训练数据, 以减少计算距离的次数.

4.1 构造 kd 树

算法: (构造平衡 kd 树)

输入: k 维空间数据集 $T = x_1, x_2, x_3, ..., x_N$,

其中 $x_i = (x_i^{(1)}, x_i^{(2)}, x_i^{(3)}, ..., x_i^{(N)}), i = 1, 2, 3, ..., N$,

输出: kd 树

算法:

(1) 开始: 构造跟节点, 根节点对应于包含 T 的 k 维空间的超矩形区域.

选择 $x^{(1)}$ 作为坐标轴, 以 T 的所有实例的 $x^{(1)}$ 坐标的**中位数**为切分店, 将根节点对应的超矩形区域切分为两个子区域, 切分由通过切分店并与坐标轴垂直的超平面实现.

由根节点生成深度为 1 的左右两个字节点, 左子节点对应坐标 $x^{(1)}$ 小于切分点的子区域, 右节点对应于坐标 $x^{(1)}$ 大于切分点的子区域.

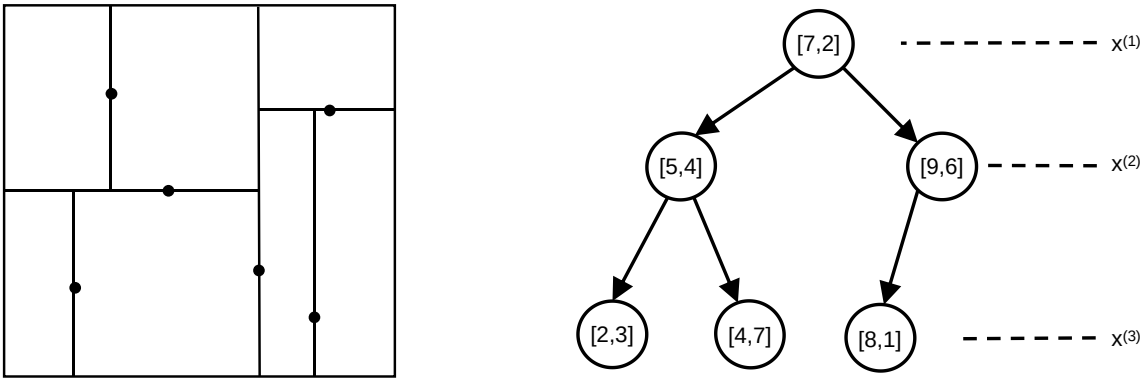
将落在切分超平面的点保存在根节点中.

(2) 重复: 对深度为 j 的节点, 选择 $x(l)$ 为切分的坐标轴, $l = j \mod k + 1$ 以该节点的区域中的所有实例的 $x^{(l)}$ 坐标的**中位数**为切分店, 将该节点对应的超矩形区域切分为两个子区域, 切分由通过切分店并与坐标轴垂直的超平面实现.

由该节点生成深度为 $j + 1$ 的左右两个字节点, 左子节点对应坐标 $x^{(l)}$ 小于切分点的子区域, 右节点对应于坐标 $x^{(l)}$ 大于切分点的子区域.

将落在切分超平面的点保存在该节点中

(3) 直到两个子区域没有实例存在时, 停止, 从而形成 kd 树的区域划分.



(a) 特征空间划分

(b) kd 树

图 3: 构造 kd 树

4.2 搜索 kd 树