KNN(K-Nearest Neighbors)(K-近邻算法)

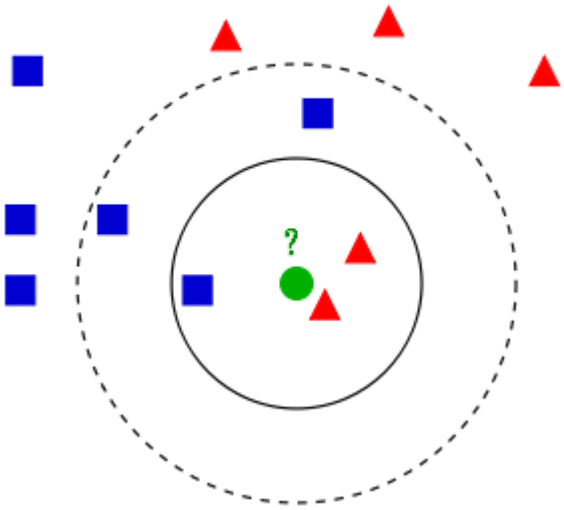
1.概念

KNN算法的核心思想是用距离最近的k个样本数据的分类来代表目标数据的分类。

或者说未标记样本的类别,由距离其最近的k个邻居投票决定.

其原理具体地讲,存在一个训练样本集,这个数据训练样本的数据集合中的每个样本都包含数据的特征值和分类值(即类别值),输入新的不含目标变量的数据,将该数据的特征与训练样本集中每一个样本进行比较,找到最相似的k个数据,这k个数据出席那次数最多的分类,即输入的具有特征值的数据的分类.

2.直观理解



例如,训练样本集中包含一系列数据,这个数据包括样本空间位置(特征)和分类信息(即目标变量,属于红色三角形还是蓝色正方形),要对中心的绿色数据的分类.运用KNN算法思想,距离最近的k个样本的分类来代表测试数据的分类,那么:

当k=3时,距离最近的3个样本在实线内,具有2个红色三角和1个蓝色正方形,因此将它归为红色三角.

当k=5时,距离最近的5个样本在虚线内,具有2个红色三角和3个蓝色正方形,因此将它归为蓝色正方形.

3.算法原理

假设X_test为未标记的数据样本,X_train为已标记类别的样本,算法原理伪代码如下:

- 遍历X_train中所有样本,计算每个样本与X_test的距离,并保存在Distance数组中
- 对Distance数组进行排序,取距离最近的k个点,记为X_knn
- 在X_knn中统计每个类别的个数
- 代表记得样本的类别,就是在X_knn中样本最多的类别

4.距离度量

> 闵可夫斯基的距离(Minkowski Distance)

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

> 常用的距离计算公式:

- 欧式距离(Euclidean Distance):

当p = 2时,

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

- 曼哈顿距离(Manhattan Distance):

当p = 1时,

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

不同的距离度量所确定的最近邻点是不同的.

5.时间复杂度

根据算法原理,每次需要预测一个点时,我们都血药计算训练数据集里每个点到这个点的距离,然后选出距离最近的k个点进行投票.

当数据集很大时,这个计算成本非常高,针对N个样本,D个特征的数据集,其算法的时间复杂度为 $O(DN^2)$.

6.特点

> 参数K的选择

- K越大:模型的**偏差**越大,对噪声越不敏感.这是因为造成了欠拟合.
- K越小:模型的**方差**越大.这是因为造成了过拟合.

> 优点

- 1.实现简单可以作为实验的Baseline.
- 2.对异常值和噪声有较高的容忍度

> 缺点

- 1.计算量大
- 2.对内存的需求也比较大

从算法的原理可以看出来,每次对一个未标记样本进行分类时,都需要全部计算一遍距离.

7.Scikit-Learn – KNN

用的是forge数据集,该数据集是一个二分类数据集,它有两个特征.

8.KNN实现

> - 1.线性扫描(即数组模拟)
> - 2.KD-Tree (时间复杂度$O(DN)log(N)$)