# kNN

k-Nearest Neighbor——k最近邻分类算法

1. 基础算法：
2. 计算未知样本和每个训练样本的距离dist
3. 统计k-最近邻样本中每个类标号出现的次数
4. 选择出现频率最大的类标号作为未知样本的类标号
5. 用于回归：

通过找出一个样本的k个最近邻居，将这些邻居的属性的平均值赋给该样本，就可以得到该样本的属性。更有用的方法是将不同距离的邻居对该样本产生的影响给予不同的权值(weight)

1. 优点：
2. 对于类域的交叉或重叠较多的待分样本集来说，kNN方法较其他方法更为适合，因为由于kNN方法主要靠周围有限的邻近样本，而不是靠判别类域的方法来确定所属类别的
3. 当样本不是一起获得而是随着时间一个个得到时，可以通过保持一个优先队列对未知样本进行分类
4. 缺点
5. 计算量大，每次对每一个待分类的文本都要计算它到全体已知样本的距离，才能求得它的K个最近邻点

**改进：**在原样本中选择数量比原样本集少的多的样本数量对未知样本进行分类

* 算法：设原始样本集为original，新样本集为output

①初始化output为空集

②从original中选择一个样本，并使用output样本中的样本对其进行kNN，

若分类错误则将该样本加入output，否则认为这个样本冗余，不做任何处理

③重复②直至遍历完original所有样本

1. 当样本不平衡时，如一个类的样本容量很大，而其他类样本容量很小时，有可能导致当输入一个新样本时，该样本的K个邻居中大容量类的样本占多数

**改进：**加权决定样本的类标号

,

其中

对于文本分类，一般选取为两文档的余弦夹角

1. k值选取很重要，k值太小会放大噪音数据的干扰，降低分类精度，k值太大而待分类样本属于数据量较少的类，则在选择k个近邻时会将大量不相似的数据也包含进来，导致分类效果降低

**改进：**通常用交叉检验的方式，每次测试完毕后计算误差率，然后重新设定不同的k值进行训练，最后取误差率最小的k 值；k一般要求低于训练样本数的平方根

1. 变量数越多，欧式距离的区分能力就越差

**改进1：**先对变量进行标准化：

标准化后的值= (标准化前的值－分量的均值) /分量的标准差

**改进2：**选用别的距离如曼哈顿距离

1. 例子：

|  |  |  |  |
| --- | --- | --- | --- |
| 电影名称 | 打斗次数 | 接吻次数 | 电影类型 |
| California Man | 3 | 104 | Romance |
| He’s Not Really into Dudes | 2 | 100 | Romance |
| Beautiful Woman | 1 | 81 | Romance |
| Kevin Longblade | 101 | 10 | Action |
| Robo Slayer 3000 | 99 | 5 | Action |
| Amped II | 98 | 2 | Action |
| 未知 | 18 | 90 | Unknown |

问未知电影的电影类型是Romance还是Action？

R代码：（knn参数train，test要求每行为一个数据，level要求与数据个数一致）

install.packages("class")

library(class)

train=data.frame(fight=c(3,2,1,101,99,98),kiss=c(104,100,81,10,5,2))

test=c(18,90)

level=c(rep("Romance",3),rep("Action",3))

k=3

print(knn(train,test,level,k))