模式识别作业

18-自控一班刘静博

k-近邻算法笔记

一 KNN 算法的概述

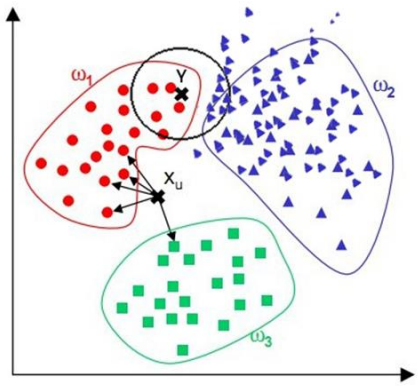
k 近邻算法简称 kNN 算法，由 Thomas 等人在 1967 年提出[1]。它基于以下思想：如果一个样本在特征空间中的 k 个最相邻的样本中的大多数是属于某一个类别，则该样本也属于这个类别，并具有这个类别上样本的特性。这就要求待分类的样本周围的 k 个邻居样本都已经准确分类，该样本所属的类别一般按照 k 邻居的属性进行投票，得票最高的属性就是待分类样本的属性。

kNN 算法我的总结就是：”近朱者赤近墨者黑”

图一

如图二所示，用 KNN 算法判别点 Xu 属于 w1，w2，w3 三块区域中的哪一块，选取距离 Xu 最近的 5 个点作为邻居集，如图所示其中 4 个邻居属于 w1，只有一个邻居属于 w2，所以我们选取 w1 作为该点所属的区域。

但是这种算法也有其弊端，举例说明。如图二所示，现在运用该算法判断 Y 点所属的区域。我们选取距离该点为 r 以内的点为它的邻居点集，如图所示黑色的圆内的点都属于我们所选的邻居点集。但是很明显其邻居点集中蓝色的点占绝大数，按照我们刚刚说过的判定原则 Y 点应该属于 w2 区域，但是很明显 Y 很靠近 w1 区域，也就是说我们判断失败了，失败的原因是数据分布的密集程度不一样，那有什么好的解决方法呢？不用说，先人的智慧都是无穷的。



图二

我们可以通过对每一个邻居设置权值来改变最终的结果，就像现实生活中，领导会听取一些意见，但是影响他决策的人往往就他身边的几个谋士，那又有一个问题出来了，如何设

置权重，其实很简单，正如上面举得听取意见的例子。当然是和领导走的近，领导的亲信他们的影响比较大。运用到算法中就是，谁和待分类目标最近，谁的权重最大，距离越远，其所占的权重越小，对分类的影响就越低，根据距离加上权重比如: 1/d (d: 距离)。按照这个思路，Y 就能被准确的分类。

二 算法的基本步骤

k 近邻算法没有求解模型参数的训练过程，参数 k 由人工指定，它在预测时才会计算待预测样本与训练样本的距离。

对于分类问题，给定个训练样本，（x_i, y_i），其中x_i为维特征向量，y_i为标签值，设定参数 k，假设类型数为 c，待分类样本的特征向量为 x。预测算法的流程为：

1. 读取训练集（已知属性的样本集）和测试集（需要待分类的样本集）
2. 选择参数 K（K 的值就是要选择的邻居的个数，一般为奇数，便于投票出结果，不会出现 1：1 的情况，建议选择 3，5，7，9）
3. 在训练样本集中找出离 x 最近的 K 个样本，假设这些样本的集合为 N。统计集合 N 中每一类样本的个数C_i,i=1-c。
4. 根据少数服从多数的投票法则(majority-voting)，让未知实例归类为 K 个最邻近样本中最多数的类别。最终的分类结果为 argmax(C_i)。

在这里 argmax(C_i)表示最大的C_i值对应的那个类。如果 k=1，则 k 近邻算法退化成最近邻算法。

三 距离的定义

根据前面的介绍我们知道，kNN 算法的实现很大程度上依赖于样本之间的距离值，因此需要定义距离的计算方式。关于距离的衡量方法，接下来介绍常用的几种距离定义，它们分别适用于不同特点的数据。一般选择作为距离的函数必须满足四个条件。

1.第一个条件是对称性，即 A 到 B 的距离和 B 到 A 的距离必须相等：

$$d(x_i, x_j)=d(x_j, x_i)$$

2.第二个条件是区分性，如果两点间的距离为 0，则两个点必须相同：

$$d(x_i, x_j)=0 \rightarrow x_i=x_j$$

3.第三个条件是三角不等式：

$$d(x_i, x_k)+ d(x_k, x_j)> d(x_i, x_j)$$

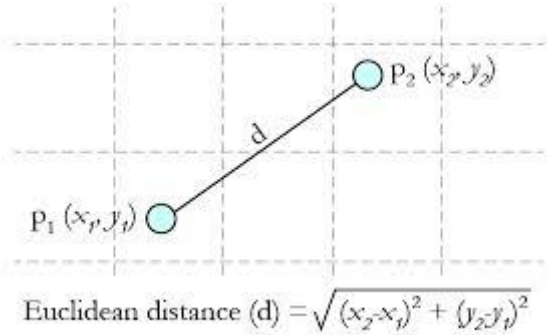
4. 第四个条是非负性，即距离不能是一个负数：

$$d(x_i, x_j) \neq 0$$

四．基本的距离函数

1.欧氏距离

欧氏距离是最常见的距离定义，它就是维欧氏空间中两点之间的距离，如图三所示。



$$d(x,y)=\sqrt{\sum_{i}^{n}(x_i-y_i)^2}$$

在使用欧氏距离时应该尽量将特征向量的每个分量归一化，以减少因为特征值的尺度范围不同所带来的干扰。否则数值小的特征分量会被数值大的特征分量淹没。例如，特征向量包含两个分量，分别为身高和肺活量，身高的范围是 150-200 厘米，肺活量为 2000-9000，如果不进行归一化，身高的差异对距离的贡献显然为被肺活量淹没。欧氏距离只是将特征向量看做空间中的点，并没有考虑这些样本特征向量的概率分布规律。

Mahalanobis 距离是一种概率意义上的距离，给定两个向量 X,Y 以及矩阵 S，它的距离定义如下：

$$d(x,y)=\sqrt{(x+y)^T S(x+y)}$$

要保证根号内的值非负，即矩阵S必须是半正定的。这种距离度量的是两个随机向量的相似度。当矩阵S 为阶单位矩阵I 时，Mahalanobis 距离退化为欧氏距离。矩阵可以通过计算训练样本集的协方差矩阵得到，也可以通过训练样本学习得到，优化某一目标函数。

五. 算法的优缺点

算法的优点：简单，易于理解，容易实现，通过对 K 的选择可具备丢噪音数据的健壮性。

算法的缺点：需要大量空间储存所有已知实例，算法复杂度高（需要比较所有已知实例与要分类的实例）当其样本分布不平衡时，比如其中一类样本过大（实例数量过多）占主导的时候，新的未知实例容易被归类为这个主导样本，因为这类样本实例的数量过大，但这个新的未知实例实际并木接近目标样本。

四 python 实战应用

1. 数据集介绍

Iris 数据集是常用的分类实验数据集，由 Fisher, 1936 收集整理。Iris 也称鸢尾花卉数据集，是一类多重变量分析的数据集。数据集包含 150 个数据集，分为 3 类，每类 50 个数据，每个数据包含 4 个属性。可通过花萼长度，花萼宽度，花瓣长度，花瓣宽度 4 个属性（数据单位是 cm）预测鸢尾花卉属于（山鸢尾，杂色鸢尾，维吉尼亚鸢尾）三个种类中的哪一类。

1.首先读取数据（数据集在文件中，请自行下载观看），将数据集的 5/12 分为测试集剩下的部分作为训练集。

```
train = []
tests = []
a = float(5/12)
with open('irisdata.txt', 'rt') as file:
    lines = csv.reader(file)
    datas = list(lines)
    # print(len(datas))测试
    for x in range(len(datas)-1):
        for y in range(4):
            datas[x][y] = float(datas[x][y])
        if random.random() < a:
            train.append(datas[x])
        else:
            tests.append(datas[x])
#print(tests,train)
```

2.我们需要定义一个欧式函数，因为求距离会一直用到。

```
def get_distance(neighbors1, neighbors2, leng):
    distance = 0
    #计算所有维度的差的平方和，此处传入的leng其实就是数据的维度
    #举例[3,4,5,8]的维度就是4
    for x in range(leng):
        distance += math.pow((neighbors1[x]-neighbors2[x]), 2)
    return math.sqrt(distance)
```

5. 下面开始写主程序，第一步根据距离选取 k 个邻居集。

```
for x in range(len(tests)):
    # traintrain[x]
    #获取某个待分类数据的k个邻居
    distances = []
    leng = len(tests[x]) - 1  # 测试集的维度
    for y in range(len(train)):  # 对训练集中的每一个数据计算它到测试元的距离
        # testone
        dist = get_distance(tests[x], train[y], leng)
        distances.append((train[y], dist))
        # distances.append(dist)
    distances.sort(key=operator.itemgetter(1))  # 对距离从小到大进行排序
    neighbors_t= []
    for x in range(k):
        neighbors_t.append(distances[x][0])
```

6. 根据邻居集的属性进行投票，选择票数最多的属性作为预测值。

```
    classVotes = {}
    for x in range(len(neighbors_t)):
        response = neighbors_t[x][-1]  # 提取邻居的属性
        if response in classVotes:  # 对k个邻居进行属性投票
            classVotes[response] += 1
        else:
            classVotes[response] = 1
    # print(classVotes)
    sortedVotes = sorted(classVotes.items(), key=operator.itemgetter(1), r
    # print(sortedVotes)
    result = sortedVotes[0][0]
    pre_results.append(result)
```

6 最后显示判断的准确率（该代码实现的平台的 pycharm 软件，运行这么多次，这次效果最好，一般在 93 到 96 之间）。（该处代码很简单不显示）

```
D:\anaconda\python.exe D:/python运营代码/python/机器学习/临近取样/模式识别作业.py
预测正确率 98.85057471264368
```

详细代码请看附录。4.

总结：knn 算法虽然在花蕊分类问题中取得了良好的效果，但是再将其其应用中依然出现了一些值得我们思考的问题。首先是是当训练样本数大、特征向量维数很高时计算复杂度高，原因是每次预测时要计算待预测样本和每一个训练样本的距离，而且要对距离进行排序找到最近的 k 个样本。另外一个就是 k 值的确定。在本应用中 k 的选择是随意的，能否结合其他算法进行自动确定优化 k 值，使得识别精度达到最好。