竞赛对应文档

一、模型的选用

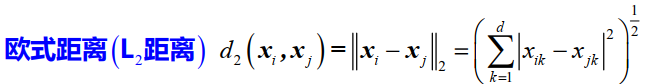
根据之前的学习，我们学过KNN，决策树，聚类，PCA特征降维，贝叶斯，逻辑回归等算法。通过对数据的观察发现，数据只有两个特征，所以不适合使用决策树、PCA特征降维的方法。聚类适用于无类别标签的数据，所以也不选用。本次竞赛我主要使用逻辑回归和KNN两种算法实现，在优化的过程中KNN算法得到的预测结果正确率更高。

二、KNN模型的实现

KNN模型是由距离最近的K个近邻决定测试数据的预测类型。

KNN实现分类问题的三要素：距离度量、超参数K值的确定、决策规则

1. 距离度量：使用欧式距离，公式如下



2. 超参数K值的选择：K=15，在无优化之前随意设定的K值为15，之后在优化的过程中，使用交叉验证法的到的K值进行的预测正确率没有K=15的正确率高，可能是由于数据中存在噪声点。

3. 决策规则：多数表决

三、核心代码分析

def knn(train\_set, train\_label, test\_set, k):

##利用欧式距离计算最近邻

#生成ndarray类型的数组，用于存放训练样本和测试样本之间的欧式距离

distance\_arr = np.empty(len(train\_set))

#遍历训练集中的每一个数据，计算训练集中每个数据与测试数据的欧氏距离

for i in range(len(train\_set)):

#计算训练数据与测试数据距离的平方和

dis = (train\_set[i][0] - test\_set[0]) \*\* 2 + (train\_set[i][1] - test\_set[1]) \*\* 2

#将距离的平方和开根号

dis = dis \*\* 0.5

#添加到欧式距离数组中

distance\_arr[i] = dis

#数组值从小到大的索引值排序，获得排序后原始数据下角标

index = distance\_arr.argsort()

#获得距离最小的前k个近邻的下角标

min\_index = index[:k]

##计算前k个近邻中每个类别的个数

#生成字典类型，用于存放类别出现的次数

count\_label = {}

#遍历前k个近邻的类别

for i in min\_index:

label = train\_label[i]

#若数组中存在该类别，则类别数加1

if label in count\_label:

count\_label[label] = count\_label[label] + 1

#否则将类别添加到字典中，初始值为1

else:

count\_label[label] = 1

#使用sorted函数对labelCount按照value值降序排序

count\_sorted=sorted(count\_label.items(), key=lambda d:d[1], reverse=True)

#返回标签出现最多的那个为预测类别

return count\_sorted[0][0]

四、问题、难点解决过程

* **训练集和测试集每个数据之间欧式距离的计算实现**

计算每一个测试数据与每一个训练数据之间的欧氏距离，无从下手怎么把两个庞大的数据集进行一个一个的运算。

解决方法：对每一个测试数据进行k近邻的预测，即在knn算法中实现对一条测试数据的类别预测。计算欧式距离的时候也只需计算一条测试数据对每一个训练数据的欧式距离。所以欧式距离的计算就只需遍历一遍测试集数据然后依照公式计算即可。

* **如何获取距离最小的前k个数据类别**

解决办法：将距离数组进行从小到大的排序，获得前k个数据

* **将距离数组排序后无法获得相应数据的类别**

解决办法：将距离数组进行排序的时候，保留原始数组的下标。可以使用numpy中的argsort（）函数获取数组从小到大排序后的原始数据下角标

* **List数组无法使用argsort（）函数**

解决办法：生成ndarray类型的数组，用于存放训练样本和测试样本之间的欧式距离。先用np.empty(len(train\_set))构造一个ndarray类型的数组，再将欧式距离的值计算出来后替换ndarray类型的数组中的值

* **如何计算k个近邻中每个类别出现的个数**

解决办法：使用字典类型，类别和个数组成键值对保存在字典中

* **如何获取类别标签出现最多的类别**

解决办法：将类别和个数组成的字典进行从大到小的排序，获取排序后的第一个键作为预测类别标签。

* **如何使字典按照value值进行降序排序**

解决办法：使用sorted函数，令函数中key的参数设置为value值这一列，使用lambda函数表示value值

* **K值的优化**

解决办法：使用交叉验证的方法确定k值，但是结果并没有自己设定的正确率高，可能是存在噪声点，所以删除了K值的优化。

* **是否对数据进行规范化预处理**

解决方法：对数据进行0均值，1方差的预处理后，预测数据的正确率并没有提高，反而下降，所以不使用数据的规范化预处理

* **保存成csv格式时计数从0开始，而标准需要从1开始**

解决办法：test\_predict.index += 1 使每一个结果的index加一