**第一次竞赛**

**过程一：刚开始使用逻辑回归，经过几次提交后，使用学习率0.008，迭代步数500，得到最大的准确率0.81428；**

**过程二：然后转而使用k近邻，记过近邻数k值的几次选择，大致得到k=60时准确率最高为0.84761；**

**过程三：对文件HTRU\_2\_train.csv中的数据进行处理，排除极端值的影响，还是基于k近邻算法，得到准确率为0.85238。**

1. trainData, testData, trainClass = initData()

调用initData函数，获取脉冲星的训练数据集，和测试数据集：

# 读取两个文件中的数据并返回经处理后的数据

def initData():

# 首先调用processData函数对训练文件HTRU\_2\_train.csv的数据进行处理

data1 = processData()

# 读取测试文件HTRU\_2\_test.csv中的数据

data2 = np.loadtxt('HTRU\_2\_test.csv', delimiter=',')

trainSet = data1

testSet = data2

# trainData存放训练集前两列（即不包括类别）

trainData = trainSet[:, 0:-1]

# testData存放测试集所有数据（即两列）

testData= testSet[:, :]

# trainClass存放训练集最后一列的类别信息（即0或1）

trainClass = trainSet[:, -1]

return trainData, testData, trainClass

其中，trainData是训练样本集**HTRU\_2\_train.csv**的前两列的属性数据，testData表示的**HTRU\_2\_test.csv**的所有数据（即前两列的属性数据），trainClass代表**HTRU\_2\_train.csv**的最后一列的类别数据（即0或1）。

2. 其中data1 = processData()的processData函数对HTRU\_2\_train.csv文件中的数据进行处理，删除测试集中指定索引序列对应的行：

def processData():

data1 = np.loadtxt('HTRU\_2\_train.csv', delimiter=',')

# 第一列属性

a = data1[:, 0]

b = list(a)

# 调用findDeleteIndex函数找到data1数据中符合要求可以进行删除的索引

index = findDeleteIndex(b)

# 删除index中的索引返回给trainSet

data2 = np.delete(data1, index, axis=0)

return data2

3. 其中index = findDeleteIndex(b)的findDeleteIndex函数返回列表参数b中符合要求的元素对应索引所组成的列表：

def findDeleteIndex(b):

num = 0

arr = []

# Q1表示在b列表数据中从小到大占25%位置的元素

Q1 = np.percentile(b, 25)

# Q3表示在b列表数据中从小到大占75%位置的元素

Q3 = np.percentile(b, 75)

# 第75%和第25%的元素中间相差的数字

IQR = Q3 - Q1

step = 0.3 \* IQR

# 遍历b列表中的所有元素

for nu in b:

# 若b列表中的元素比Q1-step小或比Q3+step大，arr就记录该元素的 # 索引号

if (nu < Q1 - step) | (nu > Q3 + step):

arr.append(num)

num = num + 1

return arr

4. for i in testData:

testSet = [i[0], i[1]]

# 采用k近邻算法，其中的参数k为16，表示选择最近邻的数目为16

ret = knn(testSet, trainData, trainClass, 16)

testPredList.append(ret)

其中n(testSet,trainData,trainClass, 16)，knn函数有4个输入参数，用于分类的输入向量是inX，输入的训练样本集为dataSet，标签向量为labels，标签向量的元素数目和矩阵dataSet的行数相同，最后的参数k表示用于选择最近邻居的数目，在此代码中近邻数K的其值为16，即考虑距离每一个测试样本testData，最近邻的16个训练样本trainData的数据。

5. # 使用k近邻算法，返回测试集的类别

def knn(inX, dataSet,labels, k):

# 获取标签向量的元素数目（即dataSet的行数）

dataSetSize = dataSet.shape[0]

# 使用欧式距离公式计算inX到各dataSet的距离

diffMat = np.tile(inX, (dataSetSize, 1)) - dataSet

sqDiffMat = diffMat\*\*2

sqDistances = sqDiffMat.sum(axis=1)

distances = sqDistances\*\*0.5

sortedDistIndices = distances .argsort()

classCount = {}

# 选择距离最小的k个点

for i in range(k):

voteIlabel = labels[sortedDistIndices[i]]

classCount[voteIlabel] = classCount.get(voteIlabel, 0) + 1

# 对数据按照从小到大的次排序，确定前k个最小元素所在的主要分类

# 将classCount字典分解为元组列表，然后使用itemgetter(1)按照第二个元素的从大到小次序

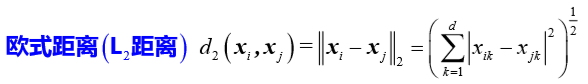
# 对元组进行排序，最后返回发生频率最高的元素标签

sortedClassCount = sorted(classCount.items(),

key = operator.itemgetter(1), reverse = True)

return sortedClassCount[0][0]

这里的距离采用欧式距离，公式为：



计算完所有点之间的距离后，可以对数据按照从小到大的次序排序，然后确定前k个距离最小元素所在的主要分类，输入k总是正整数，最后，将classCount字典分解为元组列表，然后使用程序第二行导入运算符模块的itemgetter方法，按照第二个元素的次序对元组进行排序（逆序排序），最后返回发生频率最高的元素标签。

6. testPredList = np.array(testPredList)将得到的测试数据集testData的对应类别转化成ndarray类型，从而使用astype方法将testPredList中的数据转化成整数0或1，testPredList = testPredList.astype(np.int)。

7. data = pd.DataFrame(testPredList, index=range(1, 701))将testPredList转化成DataFrame类型，并保存成csv文件data.to\_csv(testPredict.csv')，然后打开将testPredict.csv文件，将第一行改为“id”，“y”。