**第一次竞赛**

**给定两个关于脉冲星数据的文件HTRU\_2\_train.csv和HTRU\_2\_test.csv。基于KNN算法，实现脉冲星的分类。**

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：

def initData():

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

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

trainSet = data1

testSet = data2

trainData = trainSet[:, 0:-1]

testData= testSet[:, :]

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. for i in testData:

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

ret=knn(testSet,trainData,trainClass, 60)

testPredList.append(ret)

其中n(testSet,trainData,trainClass, 60)，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60，即考虑距离每一个测试样本testData，最近邻的60个训练样本trainData的数据。

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

dataSetSize = dataSet.shape[0]

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

sqDiffMat = diffMat\*\*2

sqDistances = sqDiffMat.sum(axis=1)

distances = sqDistances\*\*0.5

sortedDistIndices = distances .argsort()

classCount = {}

for i in range(k):

voteIlabel = labels[sortedDistIndices[i]]

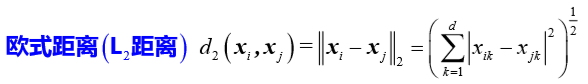
classCount[voteIlabel] = classCount.get(voteIlabel, 0) + 1

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。

4. 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)。

5. 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('data6.csv')。