

脉冲星检测——基于 knn 方法

任务一：基于固定比例的训练和测试样本集，以及固定特征，设计分类方法，检测脉冲星检测的准确率。

首先，拿到的数据由 8 个 feature 和 1 个 lable 构成，跟之前简单数据分类中数据的 2 个坐标和 1 个 class，本质上是一样的，所以可以直接用之前的 knn 方法来做，使用 Python 语言。

为了使用 numpy 库来处理数据，用 excel 将 csv 文件以 lable 降序排序，此时，所有的脉冲星数据在前面，非脉冲星数据在后面，再将 csv 文件转换成 txt 文件：1010.txt。现在就可以直接用 numpy 库读取数据了：

```
data=np.loadtxt("D:\\1010.txt")
```

现在需要将数据分割成训练集和测试集。在网上找了一下分割数据的方法，发现 SPXY 法还可以，它是由 KS 方法优化后得到的，相当于在数据的维度空间中，找到距离最远的两个向量，然后再在两个向量间均匀分割数据。这样，已知的数据就在维度空间中被均匀分割了，分割成的训练集和测试集是两个相似的集合。[1]但是由于时间和技术限制，暂时写不出 SPXY 代码，就只好用 sklearn 库中的 train_test_split 来随机分割数据，这里的训练集和测试集之比是 9:1，代码如下：

```
train,test=train_test_split(data,test_size=0.1,random_state=42)
```

后面就是使用 knn 方法判断 test 集合中的每条数据的 label 了。这里一个需要注意的地方是，脉冲星样本占比比较小，所以对 test 中的每个数据，它最近的不同 label 的 neighbor 的个数需要除以训练集中该 label 的样本占比的值，这样就不会受到样本不同 label 数据的比例的影响了。

Python 代码如下：（该 Python 文件为：mltask1.py）

```
import numpy as np
from sklearn.cross_validation import train_test_split
data=np.loadtxt("D:\\1010.txt") # use np to get data, having changed rsv file
to txt file
train,test=train_test_split(data,test_size=0.1,random_state=42)
print('length of train data:%d'%len(train))
print('length of test data:%d'%len(test))
lentrain=len(train)
lentest=len(test)
N0=0 # negative data of train data
N1=0 # positive data of train data
for i in range(lentrain):
    if train[i][8]==0:
        N0=N0+1
    else:
        N1=N1+1
print('negative data of train data:%d'%(N0))
print('positive data of train data:%d'%(N1))
k=100
length=[[0 for x in range(2)] for y in range(lentrain)]
testresult=[0 for y in range(lentest)]
```

```

TP=0
FP=0
TN=0
FN=0
for j in range(lentest): # test all test data
    if j*1.25>lentest: # print the schedule
        print('going on:80%')
    elif j*2>lentest:
        print('going on:50%')
    elif j*3.33>lentest:
        print('going on:30%')
    elif j*10>lentest:
        print('going on:10%')
    for i in range(lentrain): # compute all lengths between train data and
current test data
        length[i][0]=(train[i][0]-test[j][0])**2+\
            (train[i][1]-test[j][1])**2+\
            (train[i][2]-test[j][2])**2+ \
            (train[i][3] - test[j][3]) ** 2 + \
            (train[i][4] - test[j][4]) ** 2 + \
            (train[i][5] - test[j][5]) ** 2 + \
            (train[i][6] - test[j][6]) ** 2 + \
            (train[i][7] - test[j][7]) ** 2
        length[i][1]=train[i][8]
length=sorted(length)
n0=0
n1=0
for i in range(lentrain): # find the k nearest neighbor
    if length[i][1]==0:
        n0=n0+1
    if length[i][1]==1:
        n1=n1+1
    if int((n0*(N0+N1))/N0)>=k: # adjust n0 according to N0 and N1
        testresult[j]=0
        if test[j][8]==1:
            TN=TN+1
        else:
            FN=FN+1
        break
    if int((n1*(N0+N1))/N1)>=k: # adjust n1 according to N0 and N1
        testresult[j]=1
        if test[j][8]==0:
            FP=FP+1
        else:

```

```

        TP=TP+1
    break
print('FN:%d' %FN)  # print the result
print('TP:%d' %TP)
print('TN:%d' %TN)
print('FP:%d' %FP)
P=TP/(TP+FP)
R=TP/(TP+TN)
F=(P*R*2)/(P+R)
print('F:%f' %F)

```

运行结果如下：

```

length of train data:16108
length of test data:1790
negative data of train data:14622
positive data of train data:1486

```

```

FN:1567
TP:137
TN:16
FP:70

```

再根据公式 $P=TP/(TP+FP)$ $R=TP/(TP+TN)$ $F=(P*R*2)/(P+R)$
 计算出结果 $F=0.761111$

任务二：训练一个有效分类系统，检测训练集的正负样本比例对分类系统的影响。如，正负比例 1:1,1:2,..., 1 : N 等，比较分类系统的有效性。

这个任务仍然使用 knn 方法，在分割训练集和测试集时，脉冲星数据在前，非脉冲星数据在后，这两部分数据的前 0.9 部分作为训练集，这两部分数据的后 0.1 部分作为测试集。其中，由于脉冲星数据较少，训练集中的脉冲星数据全部用于训练，训练集中的非脉冲星数据取训练集中的脉冲星数据的 N 倍；测试集中的脉冲星数据全部用于测试，测试集中的非脉冲星数据取测试集中的脉冲星数据的 N 倍。这样，训练集和测试集的数据不会交叉，而且训练集和测试集中的脉冲星数据比例和脉冲星数据比例相同。

分割训练集和测试集的代码如下：

```

data=np.loadtxt("D:\\1010.txt") # use np to get data, having changed rsv file
to txt file
data=data.tolist()
lendata=len(data)
posdata=0
negdata=0
for i in range(lendata): # find out the negative data and positive data of

```

```

data
    if data[i][8]==0:
        negdata=negdata+1
    if data[i][8]==1:
        posdata=posdata+1
postrain_num=int(0.9*posdata) # train data : test data = 9:1
N=1
train=data[0:postrain_num]+data[posdata:posdata+postrain_num*N] # extract
train data from data
# extract test data from data
test=data[postrain_num:posdata]+data[posdata+int(0.9*negdata):posdata+int(0.9*n
egdata)+(posdata-postrain_num)*N]

```

后面的对 test 中每个数据计算距离和寻找 k nearest neighbor 过程是一样的。

Python 代码如下：(该 Python 文件为：mltask2.py)

```

import numpy as np
data=np.loadtxt("D:\\1010.txt") # use np to get data, having changed rsv file
to txt file
data=data.tolist()
lendata=len(data)
posdata=0
negdata=0
for i in range(lendata): # find out the negative data and positive data of
data
    if data[i][8]==0:
        negdata=negdata+1
    if data[i][8]==1:
        posdata=posdata+1
postrain_num=int(0.9*posdata) # train data : test data = 9:1
N=6
train=data[0:postrain_num]+data[posdata:posdata+postrain_num*N] # extract
train data from data
# extract test data from data
test=data[postrain_num:posdata]+data[posdata+int(0.9*negdata):posdata+int(0.9*n
egdata)+(posdata-postrain_num)*N]
print('length of train data:%d'%len(train))
print('length of test data:%d'%len(test))
lentrain=len(train)
lentest=len(test)
N0=0 # negative data of train data
N1=0 # positive data of train data
for i in range(lentrain):
    if train[i][8]==0:
        N0=N0+1

```

```

    else:
        N1=N1+1
print('negative data of train data:%d'%(N0))
print('positive data of train data:%d'%(N1))
k=100
length=[[0 for x in range(2)] for y in range(lentrain)]
testresult=[0 for y in range(lentest)]
TP=0
FP=0
TN=0
FN=0
for j in range(lentest): # test all test data
    if j*1.25>lentest: # print the schedule
        print('going on:80%')
    elif j*2>lentest:
        print('going on:50%')
    elif j*3.33>lentest:
        print('going on:30%')
    elif j*10>lentest:
        print('going on:10%')
    for i in range(lentrain): # compute all lengths between train data and
current test data
        length[i][0]=(train[i][0]-test[j][0])**2+\
            (train[i][1]-test[j][1])**2+\
            (train[i][2]-test[j][2])**2+ \
            (train[i][3] - test[j][3]) ** 2 + \
            (train[i][4] - test[j][4]) ** 2 + \
            (train[i][5] - test[j][5]) ** 2 + \
            (train[i][6] - test[j][6]) ** 2 + \
            (train[i][7] - test[j][7]) ** 2
        length[i][1]=train[i][8]
length=sorted(length)
n0=0
n1=0
for i in range(lentrain): # find the k nearest neighbor
    if length[i][1]==0:
        n0=n0+1
    if length[i][1]==1:
        n1=n1+1
    if int((n0*(N0+N1))/(N0))>=k: # adjust n0 according to N0 and N1
        testresult[j]=0
        if test[j][8]==1:
            TN=TN+1
        else:

```






```

        FN=FN+1
    break
if int((n1*(N0+N1))/N1)>=k: # adjust n1 according to N0 and N1
    testresult[j]=1
    if test[j][8]==0:
        FP=FP+1
    else:
        TP=TP+1
    break
print('FN:%d' %FN) # print the result
print('TP:%d' %TP)
print('TN:%d' %TN)
print('FP:%d' %FP)
P=TP/(TP+FP)
R=TP/(TP+TN)
F=(P*R*2)/(P+R)
print('F:%f' %F)






```

运行结果为：






N=1:

	FN:153
	TP:148
	TN:16
	FP:11
	F:0.916409






N=2:

	FN:309
	TP:148
	TN:16
	FP:19
	F:0.894260






N=3:

	FN:470
	TP:148
	TN:16
	FP:22
	F:0.886228






N=4:

	FN:632
	TP:148
	TN:16
	FP:24
	F:0.880952

N=5:

	FN:792
	TP:146
	TN:18
	FP:28
	F:0.863905

N=6:

	FN:953
	TP:145
	TN:19
	FP:31
	F:0.852941

由此可见，当 N 变大时，F 值变小，说明正负样本比例接近 1 时更好。

参考文献：

[1] 展晓日,朱向荣,史新元等.SPXY 样本划分法及蒙特卡罗交叉验证结合近红外光谱用于橘叶中橙皮苷的含量测定[J].光谱学与光谱分析,2009,29(4):964-968.