最大後驗機率分類器

HW2

7109056202 資工碩一 陳彥儒

壹、 目的

本次的作業就是要設計一個最大後驗機率分類器。資料集沿用作業一我們自己生的Data來處理。

貳、 方法

我們計算測試資料在兩個類別的後驗機率，並將其分類給後驗機率最大的類別。根據貝氏定理我們的後驗機率會正比於likelihood * priori probability。假設我們的資料符合高斯分布，則Likelihood function = $\frac{1}{\sqrt{(2\pi)^k|\Sigma|}} exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))$ ，

$\mu = mean\ vector$, $\Sigma = covariance\ matrix$

參、 實作

I. Load Data

一開始將我們HW1生的Data載入進來，那train_data為class 1前5000筆+class 2前5000筆，test_data為class1後

5000+class 2後5000筆。兩個的shape = (10000,50)。以下為程式碼截圖:

```
def open_data():
    train_data = np.load('./data/train_data.npy')
    test_data  = np.load('./data/test_data.npy')
    print(train_data.shape)
    return train_data,test_data
```

II. Label

因為訓練資料和測試資料都是前5000筆Class 1、後5000筆Class 2 。所以我就Class 1 = 0，Class 2 = 1，假設第一筆資料是類別1的，那他的Label表示為[[0]]。以下是程式碼截圖:

```
def label_data():
    class_1 = [[0]]*5000
    class_2 = [[1]]*5000

    y_train =  np.concatenate((class_1,class_2),axis=0)
    y_test = np.concatenate((class_1,class_2),axis=0)
    #print(y_train)
    return y_train,y_test
```

III. Concatenate data & Label

我們要為data標籤化，所以將每一筆資料後面再加上Label。以下是程式碼截圖:

```
def concatenate_Data(x_train,x_test,y_train,y_test):
    new_train_data = np.concatenate((x_train,y_train),axis = 1)
    new_test_data  = np.concatenate((x_test,y_test),axis = 1)
   # print(new_train_data)
```

IV. Data split

那因為接下來需要分開對Class 1 、Class 2做運算，所以就把資料分開來。以下為程式碼截圖:

```
def data_split(data):
    #split data
    cls1_data = np.array([[]])
    cls2_data = np.array([[]])

    cls1_data = data[:5000,:]
    cls2_data =data[5000:10000,:]
    return cls1_data,cls2_data
```

V. Mean Vector

我們把class1、class2的train data都拿進去估計mean，直接使用numpy來算mean，shape會是50*1。可以看到我們class1的mean vector，都很接近0，如下圖:

```
temp1
[ 0.00638598  0.00578116  0.01131401  0.02065771  0.01668635  0.01033172
  0.01267386  0.01055866  0.00638299 -0.00167799  0.00232512 -0.00268395
 -0.01125215 -0.01243746 -0.00330168 -0.0026206  -0.00618119 -0.00768246
 -0.00905678 -0.00509056  0.00216494  0.00518474  0.00149553 -0.00284358
 -0.00616032 -0.00759416 -0.01572396 -0.00981194 -0.02066199 -0.01869505
 -0.01788179 -0.00977457 -0.01173746 -0.01128181 -0.00515916  0.00491337
  0.00045964 -0.00267182 -0.01067502 -0.01180578 -0.00386507 -0.00360074
 -0.00401612 -0.00266449 -0.00965314 -0.01119891 -0.01031794 -0.01085126
 -0.0108587  -0.00396657]
```

class2的mean vector，都很接近0.5，如下圖:

```
temp2
[0.48665767 0.49005531 0.49669313 0.50480275 0.50244968 0.48865449
 0.49046361 0.49271137 0.47777411 0.48639462 0.49699584 0.48599383
 0.4826773  0.49207542 0.50094544 0.49953837 0.48818478 0.49460285
 0.49636128 0.48721397 0.49132533 0.4894121  0.48825357 0.4852703
 0.48740651 0.49249792 0.49858709 0.5150609  0.50503082 0.50285288
 0.510495   0.51336237 0.50786845 0.49692569 0.49854775 0.49887216
 0.49750132 0.50342532 0.49747963 0.4949959  0.49569088 0.49192914
 0.49928572 0.49157704 0.49255321 0.49735694 0.49195518 0.49910813
 0.49114828 0.48713028]
```

程式碼截圖如下:

```
def mean(cls1_test_data,cls2_test_data):
    temp1  = np.array([[]])
    temp2  = np.array([[]])
    temp1 = np.mean(cls1_test_data, axis=0)
    temp1 = temp1[:50]
    temp2 = np.mean(cls2_test_data, axis=0)
    temp2 = temp2[:50]
    print('\ntemp1')
    print(temp1)
    print('\ntemp2')
    print(temp2)
    return temp1 , temp2
```

VI. Covariance Matrix

我們把class1、class2的train data都拿進去估計Covariance Matrix，直接使用numpy來算Covariance Matrix，shape會是50*50。以下為程式碼截圖:

```
def cov(cls1_test_data,cls2_test_data):
    print('\n Conv1')
    print(np.cov(cls1_test_data[:,:50].T))
    return np.cov(cls1_test_data[:,:50].T),np.cov(cls2_test_data[:,:50].T)
```

VII. Likelihood Function

這次分類器就是利用Likelihood Function來做，公式為: $\frac{1}{\sqrt{(2\pi)^k|\Sigma|}} exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))$，以下為程式碼片段:

```python
def PDF(x,mu,cov):
    diff_x_mu = np.array([[]])

    for i in range(50):
        if i == 0 :
            diff_x_mu = np.append(diff_x_mu,[[ float(x[i][0])-mu[i] ]],axis=1)
        else :
            diff_x_mu = np.append(diff_x_mu,[[ float(x[i][0])-mu[i] ]],axis=0)
    #print(diff_x_mu.shape)

    covinv = np.linalg.inv(cov)
    temp = np.dot(diff_x_mu.T,covinv)
    temp2 = np.dot(temp,diff_x_mu)
    tempexp = math.exp(temp2*(-1/2))
    demo = math.pow((math.pi)*2,50)
    covdet = np.linalg.det(cov)
    demo*= covdet
    demo = math.pow(demo,(1/2))
    p = tempexp/demo
    return p
```

VIII.Predict Model

將我們的test data丟進去做預測，以下為實作步驟:

a.首先，類別1先判斷，如果預測正確的就是類別1的likelihood要大於類別2，再用變數correct1去累計。

b.再來換判斷類別2，如果預測正確的就是類別2的likelihood要大於類別1，再用變數correct2去累計。

c.最後再將correct1和correct2相加後再除10000來看準確率，如下圖:

```
 [ 1.2026699 ]]
class1 test correct :  4785
class2 test correct :  4783
Accuracy of prediction :  95.68 %

In [3]:
```

可以看到準確率來到95.68%，非常的高。我們再來看把train data 丟進去回測發現準確率是96.44%，看起來這樣的結果挺合理的。如右圖:

```
class1 test correct :  4829
class2 test correct :  4815
Accuracy of prediction :  96.44 %
```

肆、心得

這次的作業讓我們知道分類器裡面，貝式分類器是做最好的，也透過這次機會讓我知道likelihood function如何實作，謝謝老師的教導以及助教辛苦的批改，謝謝您。

伍、完整程式碼

```python
import pandas as pd
import math
import numpy as np
from sklearn.naive_bayes import GaussianNB
from sklearn import metrics

train_data = np.zeros((10000,50))
test_data = np.zeros((10000,50))

new_train_data = np.zeros((10000,50))
new_test_data = np.zeros((10000,50))

def open_data():
    train_data = np.load('./data/train_data.npy')
    test_data  = np.load('./data/test_data.npy')
    print(train_data.shape)
    return train_data,test_data

def label_data():
    class_1 = [[0]]*5000
    class_2 = [[1]]*5000

    y_train =  np.concatenate((class_1,class_2),axis=0)
    y_test = np.concatenate((class_1,class_2),axis=0)
    #print(y_train)
    return y_train,y_test

def concatenate_Data(x_train,x_test,y_train,y_test):
    new_train_data = np.concatenate((x_train,y_train),axis = 1)
    new_test_data  = np.concatenate((x_test,y_test),axis = 1)
   # print(new_train_data)


    return new_train_data,new_test_data

def mean(cls1_test_data,cls2_test_data):
    temp1  = np.array([[]])
    temp2  = np.array([[]])
    temp1 = np.mean(cls1_test_data, axis=0)
    temp1 = temp1[:50]
    temp2 = np.mean(cls2_test_data, axis=0)
    temp2 = temp2[:50]
    print('\ntemp1')
    print(temp1)
    print('\ntemp2')
    print(temp2)
    return temp1 , temp2

def cov(cls1_test_data,cls2_test_data):
    print('\n Conv1')
    print(np.cov(cls1_test_data[:,:50].T))
    return np.cov(cls1_test_data[:,:50].T),np.cov(cls2_test_data[:,:50].T)

```

```python
def data_split(data):
    #split data
    cls1_data = np.array([[]])
    cls2_data = np.array([[]])

    cls1_data = data[:5000,:]
    cls2_data =data[5000:10000,:]
    return cls1_data,cls2_data

def PDF(x,mu,cov):
     diff_x_mu = np.array([[]])

     for i in range(50):
         if i == 0 :
             diff_x_mu = np.append(diff_x_mu,[[ float(x[i][0])-mu[i] ]],axis=1)
         else :
             diff_x_mu = np.append(diff_x_mu,[[ float(x[i][0])-mu[i] ]],axis=0)
     #print(diff_x_mu.shape)

     covinv = np.linalg.inv(cov)
     temp = np.dot(diff_x_mu.T,covinv)
     temp2 = np.dot(temp,diff_x_mu)
     tempexp = math.exp(temp2*(-1/2))
     demo = math.pow((math.pi)*2,50)
     covdet = np.linalg.det(cov)
     demo*= covdet
     demo = math.pow(demo,(1/2))
     p = tempexp/demo
     return p
def test_data_predict(cls1_test_data,cls2_test_data,mean1,mean2,cov1,cov2):
    cls1_test_data = cls1_test_data[:,:50]
    cls2_test_data = cls2_test_data[:,:50]
    #print(cls1_test_data)
    cls1_test_data = cls1_test_data.T
    cls2_test_data = cls2_test_data.T
    #print(cls1_test_data.shape)
    #test data, label is class 1
    correct1 = 0
    correct2 = 0
    for j in range(5000):
        x = np.array([[]])
        for i in range(50):
            if i==0 :
                x = np.append(x,[[cls1_test_data[i,j]]],axis = 1)
            else :
                x = np.append(x,[[cls1_test_data[i,j]]],axis = 0)

        if j == 0 :
            print(x)
        P_x_1 = PDF(x,mean1,cov1)
        P_x_2 = PDF(x,mean2,cov2)
        if P_x_1 >=P_x_2 :
            correct1 +=1

    #test data, label is class 2

    for j in range(5000):
        x = np.array([[]])
        for i in range(50):
            if i==0 :
                x = np.append(x,[[cls2_test_data[i,j]]],axis = 1)
            else :
                x = np.append(x,[[cls2_test_data[i,j]]],axis = 0)


        P_x_1 = PDF(x,mean1,cov1)
        P_x_2 = PDF(x,mean2,cov2)
        if P_x_1 <=P_x_2 :
            correct2 +=1


    print("class1 test correct : ",correct1)
    print("class2 test correct : ",correct2)
    print("Accuracy of prediction : ", round(((correct1+correct2)/10000)*100,2),"%" )
```

```python
if __name__ =="__main__":

    #open data
    x_train,x_test = open_data()
    #label
    y_train,y_test = label_data()

    #concatenate data & label
    new_train_data,new_test_data = concatenate_Data(x_train,x_test,y_train,y_test)
    #data split
    cls1_test_data,cls2_test_data = data_split(new_test_data)

    cls1_train_data,cls2_train_data = data_split(new_train_data)

    #mean vector 50*1
    mean1,mean2 = mean(cls1_train_data,cls2_train_data)

    #covariance matrix 50*50
    cov1,cov2 = cov(cls1_train_data,cls2_train_data)


    #test data predict
    test_data_predict(cls1_test_data,cls2_test_data,mean1,mean2,cov1,cov2)
    #train data predict
    test_data_predict(cls1_train_data,cls2_train_data,mean1,mean2,cov1,cov2)
```