University of Chinese Academy of Sciences

CS091M4042H Assignment 1

Pattern Recognition and Machine Learning

熊兴旺*
Sno: 2018E8013261007
中国科学院计算技术研究所

November 7, 2018

*email: xingw.xiong@gmail.com; xiongxingwang@ict.ac.cn

1 第二章作业

1.1 Problem

作业及编程（编程可选）

- 设以下模式类别具有正态概率密度函数：

  ω_1：$\{(0\ 0)^T, (2\ 0)^T, (2\ 2)^T, (0\ 2)^T\}$

  ω_2：$\{(4\ 4)^T, (6\ 4)^T, (6\ 6)^T, (4\ 6)^T\}$

  （1）设$P(\omega_1)= P(\omega_2)=1/2$，求这两类模式之间的贝叶斯判别界面的方程式。

  （2）绘出判别界面。

- 编写两类正态分布模式的贝叶斯分类程序。（可选例题或上述作业题为分类模式）

1.2 Solution

模式的均值向量 m_i 和协方差矩阵 C_i 可用下式估计：

$$m_i = \frac{1}{N_i}\sum_{j=1}^{N_i} x_{ij}, \qquad i = 1,2 \tag{1.1a}$$

$$C_i = \frac{1}{N_i}\sum_{j=1}^{N_i} (x_{ij} - m_i)(x_{ij} - m_i)^T, \qquad i = 1,2 \tag{1.1b}$$

其中 N_i 为类别 ω_i 中模式的数目，x_{ij} 代表在第 i 个类别中的第 j 个模式。

由上式可求出各类的均值和协方差矩阵：

$$m_1 = \frac{1}{4}(4,4) = (1,1) \tag{1.2a}$$

$$m_2 = \frac{1}{4}(20,20) = (5,5) \tag{1.2b}$$

$$C_1 = C_2 = \frac{1}{4}\begin{pmatrix} 4 & 0 \\ 0 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{1.2c}$$

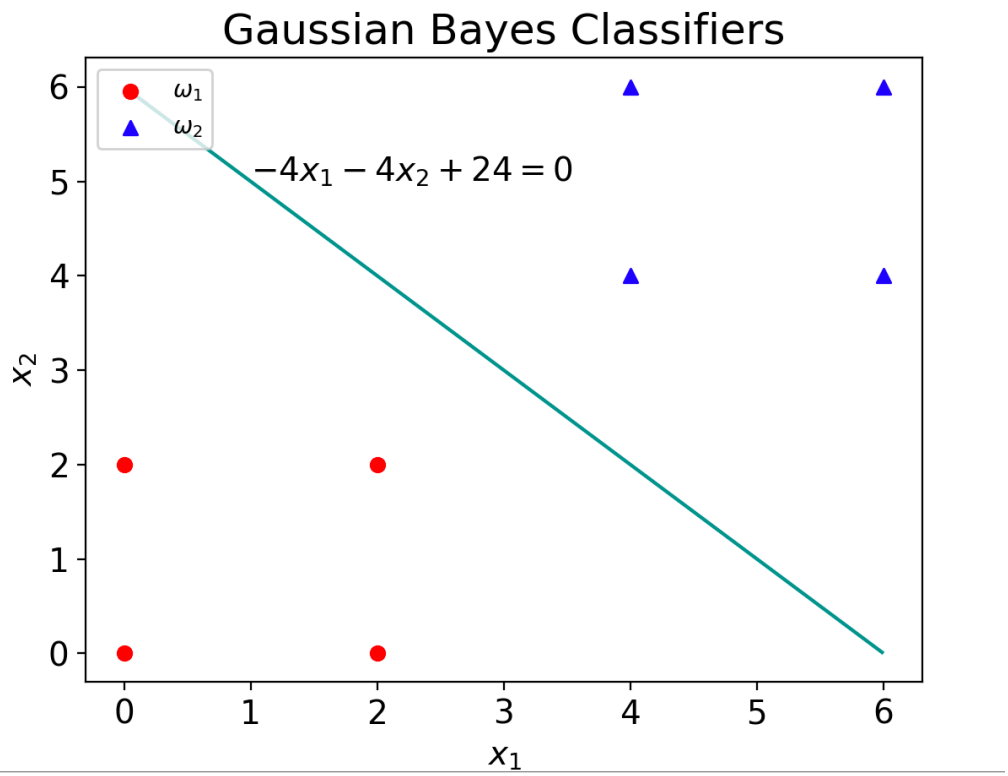
$$C = C_1^{-1} = C_2^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{1.2d}$$

设 $P(\omega_1) = P(\omega_2) = \frac{1}{2}$，因 $C_1 = C_2$，则判别界面为：

$$d_1(x) - d_2(x) = (m_1 - m_2)^T C^{-1} x - \frac{1}{2} m_1^T C^{-1} m_1 + \frac{1}{2} m_2^T C^{-1} m_2 \tag{1.3a}$$

$$= -4x_1 - 4x_2 + 24 = 0 \tag{1.3b}$$

画出分类判别界面：



1.3 Code Implement

下面是两类正态分布模式的贝叶斯分类代码：

```
1 import math
```

```
import numpy as np
import matplotlib.pyplot as plt

class BayesClassfifier:
    def __init__(self):
        pass

    def train(self, d1, d2, p1, p2):
        self.p1=p1
        self.p2=p2
        self.w1=np.mat(d1).T
        self.w2=np.mat(d2).T

        # Mean Vector
        self.m1=np.array([row.mean() for row in self.w1]).reshape(self.w1.shape[0], 1)
        self.m2=np.array([row.mean() for row in self.w2]).reshape(self.w2.shape[0], 1)
        t1=self.w1-self.m1
        t2=self.w2-self.m2

        # CoVariance
        self.c1=1.0/self.w1.shape[1]*t1.dot(t1.T)
        self.c2=1.0/self.w2.shape[1]*t2.dot(t2.T)

        self.c1_inv = self.c1.I
        self.c2_inv = self.c2.I
        self.c1_det = np.linalg.det(self.c1)
        self.c2_det = np.linalg.det(self.c2)

    # discriminant function
    def calc(self, x):
        d1 = math.log(self.p1)-1/2*math.log(self.c1_det) \
                -1/2*(x-self.m1).T*self.c1_inv*(x-self.m1)
        d2 = math.log(self.p2)-1/2*math.log(self.c2_det) \
                -1/2*(x-self.m2).T*self.c2_inv*(x-self.m2)
        return (d1-d2).mean()

    '''
        predict(x)==True :   x in \omega_1
        predict(x)==False:   x in \omega_2
    '''
    def predict(x):
        return True if calc(x) >=0 else False

    def draw(self):
        plt.title("Gaussian Bayes Classifiers",fontsize=18)
        plt.xlabel("$x_1$",fontsize=14)
        plt.ylabel("$x_2$",fontsize=14)
        plt.tick_params(axis='both',which='major',labelsize=14)
```

```
50
51          plt.scatter(self.w1[0].tolist(), self.w1[1].tolist(),\
52                  s=35, c='r', marker='o', label="$\omega_1$")
53          plt.scatter(self.w2[0].tolist(), self.w2[1].tolist(),\
54                  s=35, c='b',marker='^', label="$\omega_2$")
55          # 绘制等高线
56          lb = min(self.w1.min(), self.w2.min())
57          ub = max(self.w1.max(), self.w2.max())
58          x1=np.linspace(lb, ub, 100)
59          x2=np.linspace(lb, ub, 100)
60          X1,X2=np.meshgrid(x1, x2)
61          assert(len(X1) == len(X2))
62          Y=[]
63          for i in range(len(X1)):
64              assert(len(X1[i]) == len(X2[i]))
65              Y.append([0]*len(X1[i]))
66              for j in range(len(X1[i])):
67                  Y[i][j]=self.calc(np.mat([X1[i][j], X2[i][j]]).T)
68
69          plt.contour(X1, X2, Y, 0)
70          plt.text(1, 5, r'$-4x_1-4x_2+24=0$', fontsize=14)
71          plt.legend()
72          plt.show()
73
74
75  d1=[(0,0), (2,0), (2,2), (0,2)]
76  d2=[(4,4), (6,4), (6,6), (4,6)]
77  p1=p2=0.5
78
79  bayes = BayesClassfifier()
80  bayes.train(d1,d2,p1,p2)
81  bayes.draw();
```