homework 1

Zuyao Chen 201728008629002

- (1) $p(\boldsymbol{x}|w_i) \sim N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), i = 1, 2.$
  we have $N = 4$ instances for both categories , the mean and covariance matrix can be estimated by maximum likelihood

$$\hat{\boldsymbol{\mu}}_i = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x}_i^{(n)}, i = 1, 2;$$
$$\hat{\boldsymbol{\Sigma}}_i = \frac{1}{N}\sum_{n=1}^{N} (\boldsymbol{x}_i^{(n)} - \hat{\boldsymbol{\mu}}_i)^T(\boldsymbol{x}_i^{(n)} - \hat{\boldsymbol{\mu}}_i), i = 1, 2$$

  thus,

$$\hat{\boldsymbol{\mu}}_1 = (1, 1)^T, \hat{\boldsymbol{\Sigma}}_1 = \boldsymbol{I}, \hat{\boldsymbol{\mu}}_2 = (5, 5)^T, \hat{\boldsymbol{\Sigma}}_2 = \boldsymbol{I}$$

  according to the Bayesian rule,

$$\text{if } p(\boldsymbol{x}|w_1)P(w_1) > p(\boldsymbol{x}|w_2)P(w_2), \text{ then } \boldsymbol{x} \in w_1;$$
$$\text{if } p(\boldsymbol{x}|w_1)P(w_1) < p(\boldsymbol{x}|w_2)P(w_2), \text{ then } \boldsymbol{x} \in w_2$$

  let $f(\boldsymbol{x}) = \frac{p(\boldsymbol{x}|w_1)P(w_1)}{p(\boldsymbol{x}|w_2)P(w_2)} = 1$, since $P(w_1) = P(w_2) = 0.5$, then

$$\ln f(\boldsymbol{x}) = \ln p(\boldsymbol{x}|w_1) + \ln P(w_1) - \ln p(\boldsymbol{x}|w_2) - \ln P(w_2) = 0$$

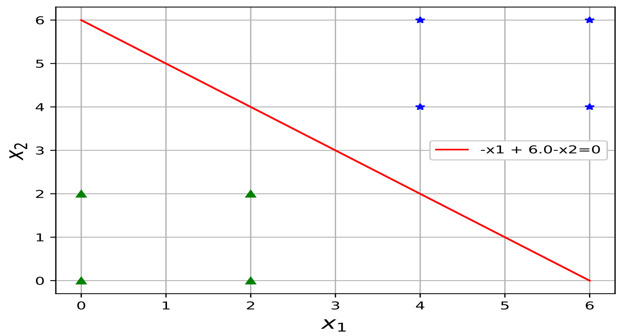
$$-\frac{1}{2}(\boldsymbol{x} - \hat{\mu}_1)^T \hat{\boldsymbol{\Sigma}}_1^{-1}(\boldsymbol{x} - \hat{\mu}_1) - \frac{1}{2}\ln\left|\hat{\boldsymbol{\Sigma}}_1\right| + \frac{1}{2}(\boldsymbol{x} - \hat{\mu}_2)^T \hat{\boldsymbol{\Sigma}}_2^{-1}(\boldsymbol{x} - \hat{\mu}_2) + \frac{1}{2}\ln\left|\hat{\boldsymbol{\Sigma}}_2\right| = 0 \tag{1}$$

  let $\boldsymbol{x} = (x_1, x_2)^T$, formula(1) equals to

$$(x_1 - 1, x_2 - 1)(x_1 - 1, x_2 - 1)^T - (x_1 - 5, x_2 - 5)(x_1 - 5, x_2 - 5)^T = 0$$
$$x_1 + x_2 - 6 = 0 \tag{2}$$

  formula(2) is the equation of decision boundary.

  (2) the graph of formula(1) is showed below.



- the program implementation(using Python) of the question above:

```
#!/usr/bin/env python

import numpy as np
import matplotlib.pyplot as plt
from sympy import *

#raw data
X1 = np.array([0,2,2,0])
Y1 = np.array([0,0,2,2])
X2 = np.array([4,6,6,4])
Y2 = np.array([4,4,6,6])
#prior prob.
P_w1 = 0.5
P_w2 = 0.5

W1 = np.array([X1,Y1]) #row vectors,shape is (2,N)
W2 = np.array([X2,Y2])
N = W1.shape[1]

#estimate the \mu and \Sigma
mu1 = np.mean(W1,axis = 1) #row
mu2 = np.mean(W2,axis = 1)
#method 1
#Sigma1 = np.cov(W1)
#Sigma2 = np.cov(W2)
#method 2
temp = W1.T - mu1
Sigma1 = 1.0/(N-1)*(temp.T).dot(temp) # 1/(N-1)(X-mu)'(X-mu)
temp = W2.T - mu2
Sigma2 = 1.0/(N-1)*(temp.T).dot(temp)

#define the equation of decision boundary
def func(X,mu,Sigma,prob):
  temp = X - mu
  inv = np.linalg.inv(Sigma)
  out = temp.T.dot(inv).dot(temp) \
    - 0.5*np.log(np.linalg.det(Sigma)) + np.log(prob)
  return out
def d(X):
  return func(X,mu1,Sigma1,P_w1) - func(X,mu2,Sigma2,P_w2)
#simplify
x1 = symbols('x1')
x2 = symbols('x2')
```

```
X = np.array([x1,x2])
result = simplify(d(X)) #that is the decision boundary
print 'd=',result
x2 = solve(d(X),X[1])
print 'x2=',x2
#plot
def f(x):
  out = []
  for i in range(len(x)):
    X[0] = x[i]
    out.append(solve(d(X),X[1]))
  return out

plt.plot(X1,Y1,'g^',X2,Y2,'b*')
L = max(W1.max(),W2.max())
x = np.linspace(0,L,10)
y = f(x)
plt.plot(x,y,label=str(x2[0])+'-x2=0',color='red')
plt.grid()
plt.legend(loc='center right')
plt.xlabel(r'$x_1$',fontsize = 16)
plt.ylabel(r'$x_2$',fontsize = 16)
plt.savefig('plot1.eps')
plt.show()
```