HW 3

Ruijie Song

Mar.4.2021

Ex1.

ECE 595 HW 3

1. a) $tr[A] = \sum_{i=1}^{d} [A]_{i,i}$. $\quad A \in \mathbb{R}^{d \times d}$

$tr(xx^T A) = \cancel{tr(x(A^T x)^T) = x^T A^T x = x^T A x}$

$\cancel{= tr(x(Ax^T)) = tr}$

$= tr(x(A^T x)^T) = x^T A^T x = x^T A x$

b) $\Sigma \in \mathbb{R}^{d \times d}$

$$\sum_{n=1}^{n} (x_n - \mu)^T \Sigma^{-1} (x_n - \mu) = tr\left[\Sigma^{-1} \sum_{n=1}^{N} (x_n - \mu)(x_n - \mu)^T\right].$$

$$\therefore \; p(D|\Sigma) = \frac{1}{(2\pi)^{Nd/2}} |\Sigma^{-1}|^{N/2} \exp\left\{-\frac{1}{2} tr\left[\Sigma^{-1} \sum_{n=1}^{N} (x_n - \mu)(x_n - \mu)^T\right]\right\}$$

c) $\underset{\sim}{\tilde{\Sigma}} = \frac{1}{N} \sum_{n=1}^{N} (x_n - \mu)(x_n - \mu)^T$, $\quad A = \Sigma^{-1} \tilde{\Sigma}$, $\lambda_1, \cdots \lambda_d$ eigenvalue of A.

$tr[A] = \sum_{i=1}^{d} \lambda_i$

$$p(D|\Sigma) = \frac{1}{(2\pi)^{Nd/2}} |\Sigma^{-1}|^{N/2} \exp\left\{-\frac{N}{2} tr[A]\right\}$$

$$= \frac{1}{(2\pi)^{Nd/2}} |\Sigma^{-1}|^{N/2} \exp\left(-\frac{N}{2} \sum_{i=1}^{d} \lambda_i\right)$$

$A = \Sigma^{-1} \tilde{\Sigma} \qquad \Sigma^{-1} = \tilde{\Sigma}^{-1} A. \qquad (\Sigma^{-1})^{N/2} = (\tilde{\Sigma}^{-1} A)^{N/2}$

$$p(D|\Sigma) = \frac{1}{(2\pi)^{Nd/2} |\tilde{\Sigma}|^{N/2}} \left(\prod_{i=1}^{d} \lambda_i\right)^{N/2} \exp\left\{-\frac{N}{2} \sum_{i=1}^{d} \lambda_i\right\}$$

d) $\log P(D|\Sigma) = -\frac{Nd}{2}\log(2\pi) - \frac{N}{2}\log(|\tilde{\Sigma}|) + \frac{N}{2}\sum_{i=1}^{d}\log(\lambda_i)$

$+ \left(-\frac{N}{2}\sum_{i=1}^{d}\lambda_i\right)$

$\arg\max_{\lambda} \log(P(D|\Sigma)) = \arg\min_{\lambda}(-P(D|\Sigma))$

$= \arg\min_{\lambda}\left(-\frac{N}{2}\sum_{i=1}^{d}\log(\lambda_i) + \frac{N}{2}\sum_{i=1}^{d}\lambda_i\right)$

$\frac{\partial P(D|\Sigma)}{\partial \lambda_i} = -\frac{N}{2}\sum_{i=1}^{d}\frac{1}{\lambda_i} + \frac{N}{2}d = 0$

$\lambda_i = 1$

e) since $A = I$, $\tilde{\Sigma} = A \cdot \Sigma = \hat{\Sigma}_{ML}$

$\cancel{\frac{d}{d\Sigma}\log(P(D|\Sigma)) = \frac{d}{d\Sigma}\left(\frac{-Nd}{2}\log 2\pi - \frac{N}{2}\log(|\tilde{\Sigma}|) + \frac{N}{2}\sum_{i=1}^{d}\log(\lambda_i)\right.}$

$\cancel{\left. - \frac{N}{2}\sum_{i=1}^{d}\lambda_i\right)}$

f) $\frac{d}{d\Sigma}\log(P(D|\Sigma)) = \frac{d}{d\Sigma}\left(\frac{N}{2}\log|\Sigma| + \frac{N}{2}\cancel{\log(2\pi)}d\right.$

$\left. + \sum_{n=1}^{N}\left\{\frac{1}{2}(x_n-\mu)^T\Sigma^{-1}(x_n-\mu)\right\}\right)$

$= \frac{N}{2}\cancel{\frac{1}{\Sigma}} + \sum_{n=1}^{N}(-1)\frac{1}{2}(x_n-\mu)^T(x_n-\mu)\cancel{\Sigma^{-2}}\,\Sigma^{-1} = 0$

$\cancel{\frac{N}{2}}\ \cancel{\frac{1}{2}}\sum_{n=1}^{N}(x_n-\mu)^T(x_n-\mu)\cancel{\Sigma} = \cancel{0}\ \frac{N}{\cancel{2}}\Sigma$

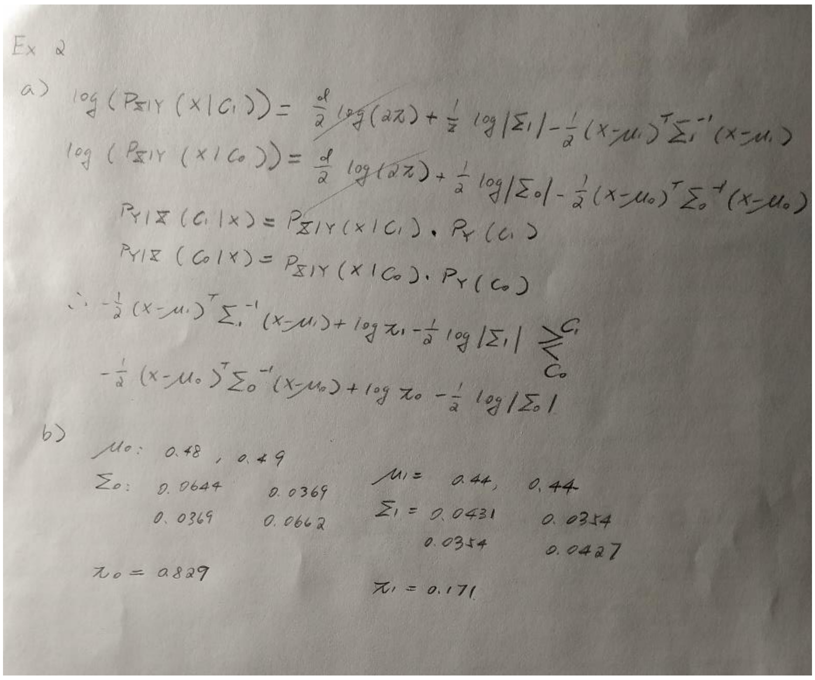
$\therefore \Sigma = \frac{1}{N}\sum_{n=1}^{N}(x_n-\mu)(x_n-\mu)^T$

g) $\hat{\mu} = \frac{1}{N}\sum_{n=1}^{N}x_n$,

$\hat{\Sigma}_{\cancel{unbiased}\ unbias} = \frac{1}{N-1}\sum_{n=1}^{N}(x_n-\hat{\mu})(x_n-\hat{\mu})^T$

Ex2.

a & b



Ex 2

a) $\log(P_{X|Y}(x|C_1)) = \cancel{\frac{d}{2}\log(2\pi)} + \frac{1}{2}\log|\Sigma_1| - \frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1)$

$\log(P_{X|Y}(x|C_0)) = \cancel{\frac{d}{2}\log(2\pi)} + \frac{1}{2}\log|\Sigma_0| - \frac{1}{2}(x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0)$

$P_{Y|X}(C_1|x) = P_{X|Y}(x|C_1) \cdot P_Y(C_1)$

$P_{Y|X}(C_0|x) = P_{X|Y}(x|C_0) \cdot P_Y(C_0)$

$\therefore -\frac{1}{2}(x-\mu_1)^T\Sigma_1^{-1}(x-\mu_1) + \log\pi_1 - \frac{1}{2}\log|\Sigma_1| \underset{C_0}{\overset{C_1}{\gtrless}}$

$-\frac{1}{2}(x-\mu_0)^T\Sigma_0^{-1}(x-\mu_0) + \log\pi_0 - \frac{1}{2}\log|\Sigma_0|$

b) μ_0: 0.48 , 0.49

Σ_0: 0.0644 0.0369
0.0369 0.0662

$\mu_1 =$ 0.44, 0.44

$\Sigma_1 =$ 0.0431 0.0354
0.0354 0.0427

$\pi_0 = 0.829$

$\pi_1 = 0.171$

c.

d.

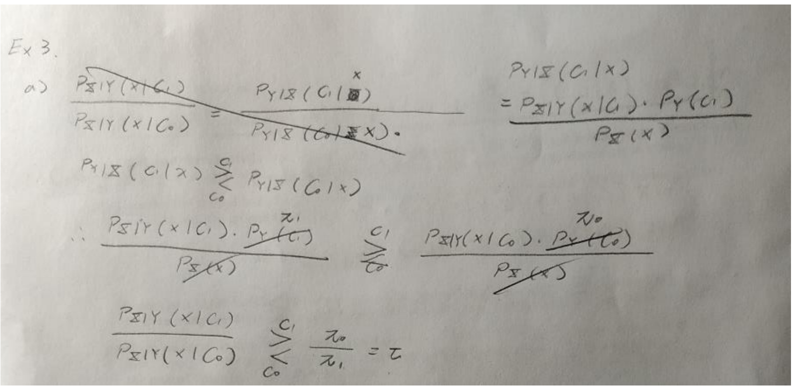
MAE = 0.08762355033904315

e.

No, it does not perform well. Maybe blurred background of this plot makes the prediction inaccurate.

Ex3.

a.



Ex 3.

a) $\cancel{\dfrac{P_{X|Y}(x|C_1)}{P_{X|Y}(x|C_0)} = \dfrac{P_{Y|X}(C_1|x)}{P_{Y|X}(C_0|x)}}$

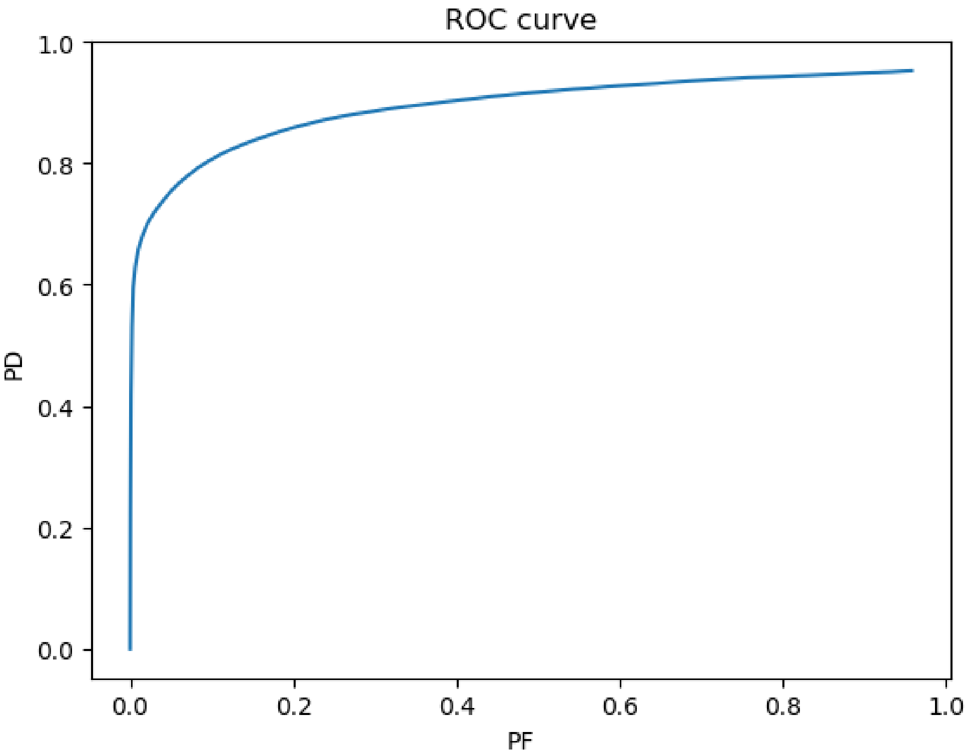
$$P_{Y|X}(C_1|x) = \frac{P_{X|Y}(x|C_1)\cdot P_Y(C_1)}{P_X(x)}$$

$$P_{Y|X}(C_1|x) \underset{C_0}{\overset{C_1}{\gtrless}} P_{Y|X}(C_0|x)$$

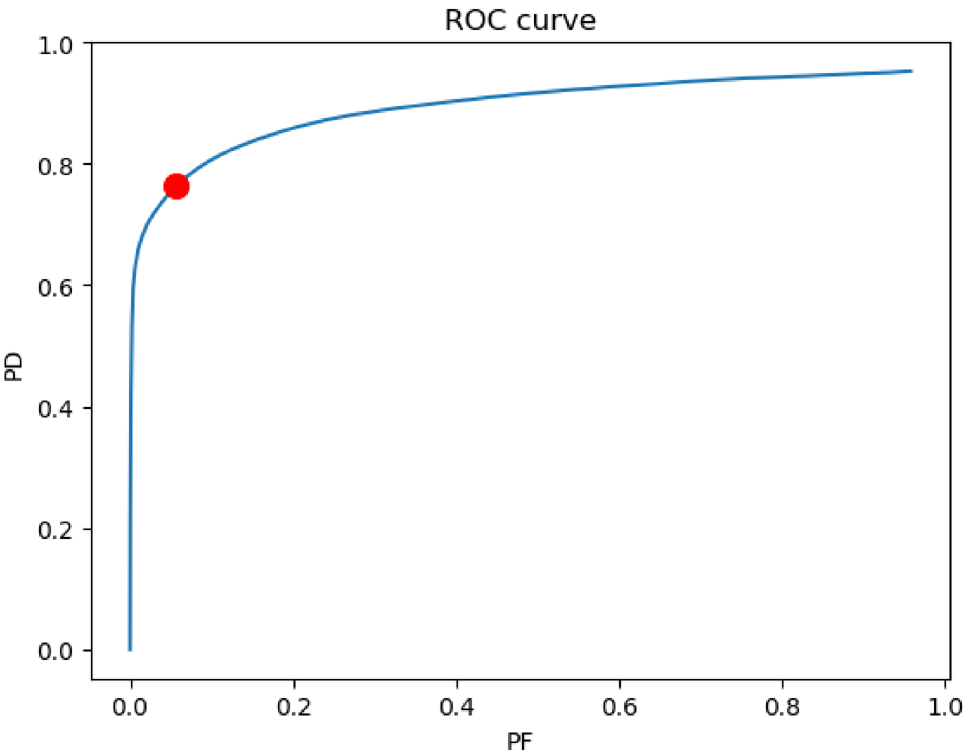
$$\therefore \frac{P_{X|Y}(x|C_1)\cdot \overset{\pi_1}{\cancel{P_Y(C_1)}}}{\cancel{P_X(x)}} \underset{C_0}{\overset{C_1}{\gtrless}} \frac{P_{X|Y}(x|C_0)\cdot \overset{\pi_0}{\cancel{P_Y(C_0)}}}{\cancel{P_X(x)}}$$

$$\frac{P_{X|Y}(x|C_1)}{P_{X|Y}(x|C_0)} \underset{C_0}{\overset{C_1}{\gtrless}} \frac{\pi_0}{\pi_1} = \tau$$

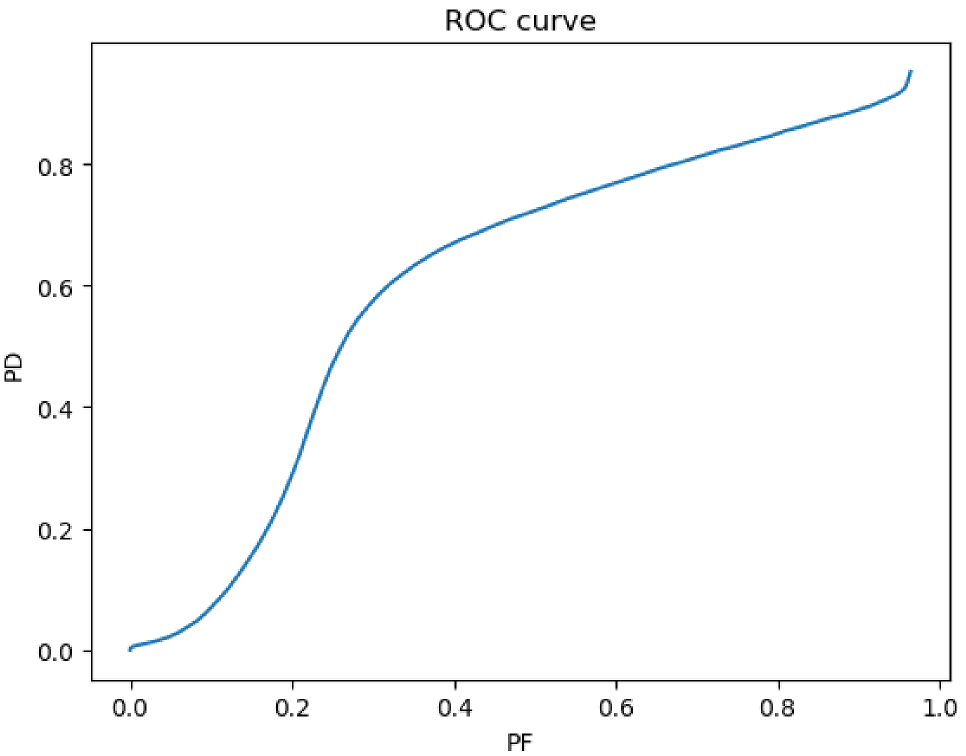
b.



c.


ROC curve
1.0
0.8
0.6
0.4
0.2
0.0
PD
0.0
0.2
0.4
0.6
0.8
1.0
PF

d.


ROC curve
0.8
0.6
0.4
0.2
0.0
PD
0.0
0.2
0.4
0.6
0.8
1.0
PF

```python
# -*- coding: utf-8 -*-
"""
Created on Thu Mar  4 23:17:38 2021

@author: 11327
"""

import numpy as np
import matplotlib.pyplot as plt
import cv2
from PIL import Image
import cvxpy as cvx

train_cat = np.matrix(np.loadtxt('train_cat.txt', delimiter = ','))
train_grass = np.matrix(np.loadtxt('train_grass.txt', delimiter = ','))

Y = plt.imread('cat_grass.jpg') / 255
# Y = cv2.imread('rabbit.jpg',0) / 255

# EX2
# 2
# i
row0,col0 = np.shape(train_grass)
miu0 = np.zeros(row0)
for i in range(row0):
    miu0[i] = np.mean(train_grass[i])

row1,col1 = np.shape(train_cat)
miu1 = np.zeros(row1)
for i in range(row1):
    miu1[i] = np.mean(train_cat[i])

# ii
Sigma0 = np.cov(train_grass)
Sigma1 = np.cov(train_cat)

# iii
K0 = col0
K1 = col1
pi0 = K0 / (K1+K0)
pi1 = K1 / (K1+K0)

# c
M,N = Y.shape
prediction = np.zeros((M,N))
'''
for i in range(M-8):
    for j in range(N-8):
        block = Y[i:i+8, j:j+8] # This is a 8x8 block
        block = np.reshape(block,(64,1))
        C1 = (-1/2)*np.dot(np.dot((block-miu1).T,np.linalg.inv(Sigma1)),(block-miu1))+np.log(pi1
        C0 = (-1/2)*np.dot(np.dot((block-miu0).T,np.linalg.inv(Sigma0)),(block-miu0))+np.log(pi0
        if C1[0,0] > C0[0,0]:
            prediction[i,j] = 1
        elif C1[0,0] < C0[0,0]:
            prediction[i,j] = 0

prediction = prediction*255
im = Image.fromarray(prediction)
im = im.convert('L')
im.save('outfile.png')
```

```
# d
prediction = prediction / 255

rowp,colp = prediction.shape
truth = plt.imread('truth.png')
MAE = 0
for i in range(rowp-8):
    for j in range(colp-8):
        MAE = MAE + np.abs(prediction[i,j]-truth[i,j])

MAE = 1/(rowp*colp) * MAE
'''

# Ex 3
# b
truth = plt.imread('truth.png')
# Number of true positive in ground truth
TP_total = np.count_nonzero(truth)
# Number of true negative in ground truth
TN_total = len(truth.ravel()) - TP_total
'''
d = 64
abs_Sigma1 = np.linalg.det(Sigma1)
abs_Sigma0 = np.linalg.det(Sigma0)
inv_Sigma1 = np.linalg.inv(Sigma1)
inv_Sigma0 = np.linalg.inv(Sigma0)
const = np.power((2*np.pi),d)
div = np.zeros((M,N))

for i in range(M-8):
    for j in range(N-8):
        block = Y[i:i+8, j:j+8] # This is a 8x8 block
        block = np.reshape(block,(64,1))
        c1 = 1/(np.sqrt(const*abs_Sigma1)) * np.exp(-0.5*np.dot(np.dot((block-miu1).T,inv_Sigma1
        c0 = 1/(np.sqrt(const*abs_Sigma0)) * np.exp(-0.5*np.dot(np.dot((block-miu0).T,inv_Sigma0
        div[i,j] = np.log(c1[0,0]/c0[0,0])

n_tau = 101
PF = np.zeros(n_tau)
PD = np.zeros(n_tau)
tauset = np.linspace(-200,100,n_tau)
for k in range(n_tau):
    tau = (tauset[k])
    prediction = np.zeros(Y.shape)
    true_positive = 0; true_negative = 0
    false_positive = 0; false_negative = 0
    prediction = div > tau
    for i in range(M - 8):
        for j in range(N - 8):
            if (prediction[i][j]==1) and (truth[i][j]>=0.5): true_positive  += 1
            if (prediction[i][j]==1) and (truth[i][j]<=0.5): false_positive += 1
            if (prediction[i][j]==0) and (truth[i][j]>=0.5): false_negative += 1
            if (prediction[i][j]==0) and (truth[i][j]<=0.5): true_negative  += 1

  # Calculate Pd and Pf

    PD[k] = true_positive/TP_total
    PF[k] = false_positive/TN_total

plt.figure()
```

```
plt.plot(PF,PD)
plt.xlabel('PF')
plt.ylabel('PD')
plt.title('ROC curve')

tau_op = pi0 / pi1
tau = tau_op
prediction = np.zeros(Y.shape)
true_positive = 0; true_negative = 0
false_positive = 0; false_negative = 0
prediction = div > tau
for i in range(M - 8):
    for j in range(N - 8):
        if (prediction[i][j]==1) and (truth[i][j]>=0.5): true_positive  += 1
        if (prediction[i][j]==1) and (truth[i][j]<=0.5): false_positive += 1
        if (prediction[i][j]==0) and (truth[i][j]>=0.5): false_negative += 1
        if (prediction[i][j]==0) and (truth[i][j]<=0.5): true_negative  += 1

PD_op = true_positive/TP_total
PF_op = false_positive/TN_total

plt.plot(PF_op,PD_op, 'ro', markersize=10)
plt.imshow()
'''

# d
d = 64
A = np.vstack([train_cat.T,train_grass.T])
b = np.vstack([np.ones((K1,1)),-1*np.ones((K0,1))])
theta = cvx.Variable((d,1))
objective = cvx.Minimize(cvx.sum_squares(A@theta - b))
prob = cvx.Problem(objective)
prob.solve()
sol = theta.value

pro = np.zeros((M,N))
for i in range(M-8):
    for j in range(N-8):
        block = Y[i:i+8, j:j+8] # This is a 8x8 block
        block = np.reshape(block,(64,1))
        pro[i,j] = (np.dot(sol.T,block))

n_tau = 100
PF = np.zeros(n_tau)
PD = np.zeros(n_tau)
tauset = np.linspace(-1.2,-0.1,n_tau)
for k in range(n_tau):
    tau = (tauset[k])
    prediction = np.zeros(Y.shape)
    true_positive = 0; true_negative = 0
    false_positive = 0; false_negative = 0
    prediction = pro > tau
    for i in range(M - 8):
        for j in range(N - 8):
            if (prediction[i][j]==1) and (truth[i][j]>=0.5): true_positive  += 1
            if (prediction[i][j]==1) and (truth[i][j]<=0.5): false_positive += 1
            if (prediction[i][j]==0) and (truth[i][j]>=0.5): false_negative += 1
            if (prediction[i][j]==0) and (truth[i][j]<=0.5): true_negative  += 1

  # Calculate Pd and Pf
```

```
    PD[k] = true_positive/TP_total
    PF[k] = false_positive/TN_total

plt.figure()
plt.plot(PF,PD)
plt.xlabel('PF')
plt.ylabel('PD')
plt.title('ROC curve')
```