Fengnan Wang (fwang40)

My code

./hw8.py

#!/usr/local/bin/python

import numpy as np

import matplotlib.pyplot as plt

import random

import cvxopt

import cvxopt.solvers

import pylab as pl

x=[]

y=[]

C=[]

for i in range(0,100):

x.append(random.uniform(0,1))

y.append(random.uniform(0,1))

C.append([x[i],y[i]])

C=np.array(C)

#print 'C',C

"""

plt.plot(x,y,'ko')

plt.xlabel('x')

plt.ylabel('y')

plt.title('input patterns')

plt.show()

x\_origin = np.linspace(0, 1, 100)

y\_origin = np.linspace(0, 1, 100)

x\_origin, y\_origin = np.meshgrid(x\_origin,y\_origin)

line = plt.plot(x\_origin, np.sin(10\*x\_origin)/5.0 +0.3, 'k', linewidth=2)

F = (x\_origin - 0.5)\*\*2 + (y\_origin - 0.8)\*\*2-0.15\*\*2

plt.contour(x\_origin,y\_origin,F,[0])

plt.show()

"""

C0=[]

C1=[]

d=[0]\*100

def classifer(x,y):

for i in range(100):

if y[i] < np.sin(10\*x[i])/5.0 +0.3 or (x[i] - 0.5)\*\*2 + (y[i] - 0.8)\*\*2 < 0.15\*\*2:

d[i]=1.0

C1.append([x[i],y[i]])

X1=np.array(C1)

else:

d[i]=-1.0

C0.append([x[i],y[i]])

X0=np.array(C0)

return d,C0,C1,X1,X0

d,C0,C1,X1,X0=classifer(x,y)

d1=np.ones(len(C1))

d0=np.ones(len(C0))\*-1

d=np.array(d)

print 'C0',C0

print 'C1',C1

print 'd',d,type(d)

"""

plt.plot(\*zip(\*C0), marker='o', color='r', ls='')

plt.plot(\*zip(\*C1), marker='\*', color='k', ls='')

x\_origin = np.linspace(0, 1, 100)

y\_origin = np.linspace(0, 1, 100)

x\_origin, y\_origin = np.meshgrid(x\_origin,y\_origin)

line = plt.plot(x\_origin, np.sin(10\*x\_origin)/5.0 +0.3, 'k', linewidth=2)

F = (x\_origin - 0.5)\*\*2 + (y\_origin - 0.8)\*\*2-0.15\*\*2

plt.contour(x\_origin,y\_origin,F,[0])

plt.show()

"""

#linear\_kernel

def k1(x1,x2):

return np.dot(x1,x2)

#polynomial\_kernel

def k2(x1,x2,d=3):

return (1+np.dot(x1,x2))\*\*d

#gaussian\_kernel

def k3(x1,x2,sigma=0.17):

x1=np.array(x1)

x2=np.array(x2)

return np.exp(-np.linalg.norm(x1-x2)\*\*2/(2\*(sigma\*\*2)))

# k matrix

k=np.zeros((100,100))

#print 'C[2]',C[2],'C[4]',C[4]

#print np.dot(C[2],C[4])

for i in range(100):

for j in range(100):

k[i,j]=k3(C[i],C[j])

print 'k',k

print 'x',C

print 'd',d

m,n=C.shape

P=cvxopt.matrix(np.outer(d,d)\*k)

q=cvxopt.matrix(np.ones(m)\*-1)

A=cvxopt.matrix(d,(1,m))

print A,len(A)

b=cvxopt.matrix(0.0)

G=cvxopt.matrix(np.diag(np.ones(m)\*-1))

h=cvxopt.matrix(np.zeros(m))

solution=cvxopt.solvers.qp(P,q,G,h,A,b)

a=np.ravel(solution['x'])

print 'a',a,len(a)

def weight(a,K):

sv=a>1e-4

ind=np.arange(len(a))[sv]

a1=a[sv]

X\_sv=C[sv]

sv\_d=d[sv]

print "%d support vectors out of %d points" % (len(a1), m)

b=0

for i in range(len(a1)):

b += sv\_d[i]

b -= np.sum(a1\*sv\_d\*K[ind[i],sv])

b /= len(a1)

return a1,sv\_d,X\_sv,b

def project(x,a1,sv\_d,X\_sv,b):

# x equals to X\_train

y\_p =np.zeros(len(x))

for i in range(len(x)):

s=0

for a,sv\_d,sv in zip(a1,sv\_d,X\_sv):

s+=a\*sv\_d\*k3(x[i],sv)

y\_p[i] = s

return y\_p+b

a1,sv\_d,X\_sv,b = weight(a,k)

y\_p = project(C,a1,sv\_d,X\_sv,b)

pl.plot(X0[:,0],X0[:,1],'ro')

pl.plot(X1[:,0],X1[:,1],'b\*')

pl.scatter(X\_sv[:,0], X\_sv[:,1], s=100, c='k')

X1, X2 = np.meshgrid(np.linspace(0,1,100), np.linspace(0,1,100))

X = np.array([[x1, x2] for x1, x2 in zip(np.ravel(X1), np.ravel(X2))])

Z = project(X,a1,sv\_d,X\_sv,b).reshape(X1.shape)

pl.contour(-X1, -X2, -Z, [0.0], colors='k', linewidths=0.1)

pl.contour(-X1, -X2, -(Z + 1), [0.0], colors='b', linewidths=0.1)

pl.contour(-X1, -X2, -(Z – 1), [0.0], colors='r', linewidths=0.1)

pl.axis("tight")

pl.show()

pcost dcost gap pres dres

0: -2.6120e+01 -7.6343e+01 3e+02 1e+01 2e+00

1: -6.2178e+01 -1.1771e+02 1e+02 5e+00 1e+00

2: -1.5538e+02 -2.2250e+02 1e+02 4e+00 9e-01

3: -2.7706e+02 -3.7119e+02 1e+02 3e+00 6e-01

4: -3.4311e+02 -4.2798e+02 1e+02 2e+00 4e-01

5: -3.7154e+02 -3.9033e+02 2e+01 3e-01 5e-02

6: -3.7319e+02 -3.7728e+02 5e+00 5e-02 1e-02

7: -3.7374e+02 -3.7395e+02 2e-01 6e-04 1e-04

8: -3.7387e+02 -3.7388e+02 4e-03 6e-06 1e-06

9: -3.7387e+02 -3.7387e+02 7e-05 6e-08 1e-08

Optimal solution found.

a

[1.90571623e-08 5.10348355e-07 7.94454008e-08 1.88649778e-08 2.48678583e-08 1.97631827e+00 7.96577710e+01 2.61665669e-07

7.35168407e-08 2.89610175e-08 3.64258727e-08 1.15228464e+01

8.32252105e+01 1.28518723e+01 2.65503858e-08 4.78784872e+01

2.79736956e-07 1.24826341e+02 1.26264164e-07 6.12308695e+00

1.52021783e-08 1.49643021e-07 2.20483156e-08 4.83943949e-08

3.62109162e-05 2.06763404e-08 3.47317645e-08 4.78140445e-08

4.65738001e-08 5.81275312e+00 1.23520605e+00 6.06106852e-08

1.20543393e-07 1.87821778e-06 4.95689540e-07 5.80456713e-07

4.48604708e-08 2.76765093e-07 1.73990441e-07 4.73966905e-08

2.23623512e-07 4.66996029e-08 2.87915011e-08 7.67236519e-07

8.96076553e-08 3.11936595e-07 7.99819465e-08 1.47875885e-07

4.87263948e-08 2.34717811e-07 7.36014554e-08 1.32591788e+01

2.23946923e-07 2.64483578e-08 3.27471460e-08 9.77673883e-08

3.65658355e-08 1.51478918e-07 1.26066776e+02 2.02385066e-07

8.49500550e-08 1.19702890e-07 3.49067334e+00 6.60607520e-07

7.39532554e-07 3.03122750e-08 1.38972453e-07 3.04191338e-08

8.01985536e-08 1.37524767e+01 2.60847532e-07 2.10173467e-06

8.04138371e+00 3.22960906e-08 4.93885221e-08 1.25931693e+02

1.41029945e-06 4.55259653e-03 3.60929536e-08 1.15605194e-07

2.66674506e-08 2.97681614e-08 7.48939025e+01 1.38976581e-07

4.12622793e-08 2.10513683e-07 4.09852317e-08 6.18608479e-08

6.34572390e-08 5.39232987e-08 3.55573889e-08 3.56276143e-07

7.19886828e+00 3.21595261e-07 9.17235985e-08 4.23871852e-08

6.22311503e-08 7.04120167e-08 3.67091105e-08 4.81546313e-08]

20 support vectors out of 100 points

clf

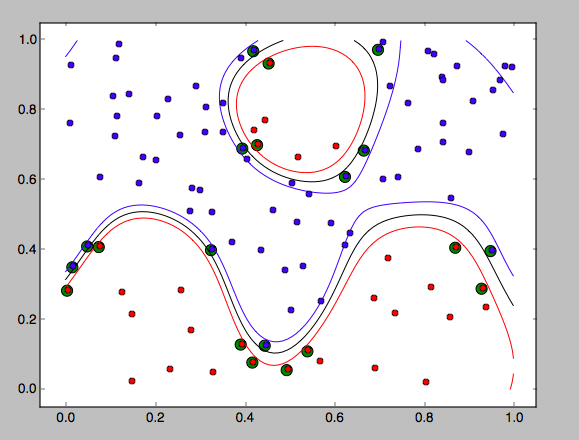
[ 3.16096863 3.42877158 3.70695122 ..., -0.58140702 -0.53579287

-0.49217292] (10000,) x1shape (100, 100)

I used contour function to implement my graph

And I tired different sigma to present my result.

Sigma=0.17



Sigma=0.175

