

Práctica 6: Support vector machiines

Aprendizaje Automático y Big Data

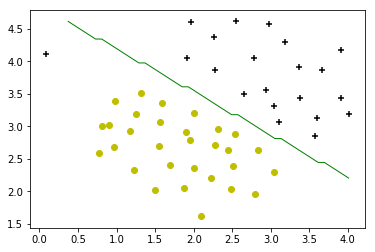


13 de diciembre de 2018

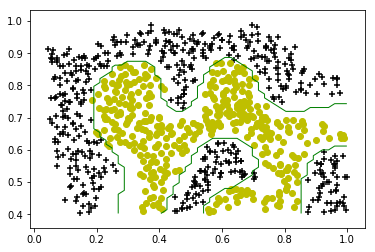
Felix Villar Y víctor ramos

Universidad Complutense de Madrid

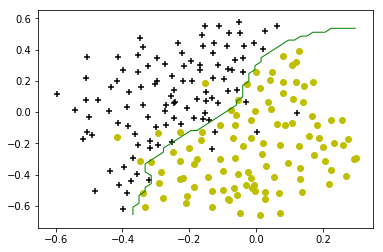
1. ***Support Vector Machines***
   1. **Kernel lineal**

****

* 1. **Kernel Gaussiano**

****

* 1. **Elección de los parámetros C y sigma**



1. ***Detección de spam***

***Resultado del código:***

C=0.01.Porcentaje=96.66666666666667

C=0.03.Porcentaje=96.26262626262626

C=0.1.Porcentaje=96.76767676767678

C=0.3.Porcentaje=96.66666666666667

C=1.0.Porcentaje=96.16161616161617

C=3.0.Porcentaje=94.64646464646465

C=10.0.Porcentaje=85.35353535353535

C=30.0.Porcentaje=85.35353535353535

Mejor solución lineal: C = 0.1. % = 96.76767676767678

C=0.01, sigma=0.01 .Porcentaje=84.84848484848484

C=0.01, sigma=0.03 .Porcentaje=84.84848484848484

C=0.01, sigma=0.1 .Porcentaje=84.84848484848484

C=0.01, sigma=0.3 .Porcentaje=84.84848484848484

C=0.01, sigma=1.0 .Porcentaje=84.84848484848484

C=0.01, sigma=3.0 .Porcentaje=84.84848484848484

C=0.01, sigma=10.0 .Porcentaje=84.84848484848484

C=0.01, sigma=30.0 .Porcentaje=84.84848484848484

C=0.03, sigma=0.01 .Porcentaje=84.84848484848484

C=0.03, sigma=0.03 .Porcentaje=84.84848484848484

C=0.03, sigma=0.1 .Porcentaje=84.84848484848484

C=0.03, sigma=0.3 .Porcentaje=84.84848484848484

C=0.03, sigma=1.0 .Porcentaje=84.84848484848484

C=0.03, sigma=3.0 .Porcentaje=84.84848484848484

C=0.03, sigma=10.0 .Porcentaje=84.84848484848484

C=0.03, sigma=30.0 .Porcentaje=84.84848484848484

C=0.1, sigma=0.01 .Porcentaje=84.84848484848484

C=0.1, sigma=0.03 .Porcentaje=84.84848484848484

C=0.1, sigma=0.1 .Porcentaje=84.84848484848484

C=0.1, sigma=0.3 .Porcentaje=84.84848484848484

C=0.1, sigma=1.0 .Porcentaje=84.84848484848484

C=0.1, sigma=3.0 .Porcentaje=85.45454545454545

C=0.1, sigma=10.0 .Porcentaje=85.55555555555556

C=0.1, sigma=30.0 .Porcentaje=84.84848484848484

C=0.3, sigma=0.01 .Porcentaje=85.15151515151516

C=0.3, sigma=0.03 .Porcentaje=85.15151515151516

C=0.3, sigma=0.1 .Porcentaje=85.15151515151516

C=0.3, sigma=0.3 .Porcentaje=85.15151515151516

C=0.3, sigma=1.0 .Porcentaje=85.65656565656565

C=0.3, sigma=3.0 .Porcentaje=86.86868686868688

C=0.3, sigma=10.0 .Porcentaje=94.94949494949495

C=0.3, sigma=30.0 .Porcentaje=84.84848484848484

C=1.0, sigma=0.01 .Porcentaje=86.46464646464646

C=1.0, sigma=0.03 .Porcentaje=86.46464646464646

C=1.0, sigma=0.1 .Porcentaje=86.46464646464646

C=1.0, sigma=0.3 .Porcentaje=86.46464646464646

C=1.0, sigma=1.0 .Porcentaje=86.86868686868688

C=1.0, sigma=3.0 .Porcentaje=88.88888888888889

C=1.0, sigma=10.0 .Porcentaje=97.07070707070707

C=1.0, sigma=30.0 .Porcentaje=91.81818181818183

C=3.0, sigma=0.01 .Porcentaje=86.46464646464646

C=3.0, sigma=0.03 .Porcentaje=86.46464646464646

C=3.0, sigma=0.1 .Porcentaje=86.46464646464646

C=3.0, sigma=0.3 .Porcentaje=86.46464646464646

C=3.0, sigma=1.0 .Porcentaje=87.17171717171716

C=3.0, sigma=3.0 .Porcentaje=90.0

C=3.0, sigma=10.0 .Porcentaje=96.86868686868686

C=3.0, sigma=30.0 .Porcentaje=95.75757575757575

C=10.0, sigma=0.01 .Porcentaje=86.46464646464646

C=10.0, sigma=0.03 .Porcentaje=86.46464646464646

C=10.0, sigma=0.1 .Porcentaje=86.46464646464646

C=10.0, sigma=0.3 .Porcentaje=86.46464646464646

C=10.0, sigma=1.0 .Porcentaje=87.17171717171716

C=10.0, sigma=3.0 .Porcentaje=87.97979797979798

C=10.0, sigma=10.0 .Porcentaje=96.56565656565657

C=10.0, sigma=30.0 .Porcentaje=96.76767676767678

C=30.0, sigma=0.01 .Porcentaje=86.46464646464646

C=30.0, sigma=0.03 .Porcentaje=86.46464646464646

C=30.0, sigma=0.1 .Porcentaje=86.46464646464646

C=30.0, sigma=0.3 .Porcentaje=86.46464646464646

C=30.0, sigma=1.0 .Porcentaje=87.17171717171716

C=30.0, sigma=3.0 .Porcentaje=85.25252525252526

C=30.0, sigma=10.0 .Porcentaje=96.96969696969697

C=30.0, sigma=30.0 .Porcentaje=96.56565656565657

Mejor solución gaussiana: C = 1.0, Sigma = 10.0. % = 97.07070707070707

1. ***Código de la práctica entera***

import numpy as np

from sklearn.svm import SVC

import scipy.io

import matplotlib.pyplot as plt

from process\_email import email2TokenList

from get\_vocab\_dict import getVocabDict

import codecs

def pintar(X,y,svm):

neg = np.where(y==0)

pos = np.where(y==1)

plt.figure()

x1\_min,x1\_max = X[:,0].min(), X[:,0].max()

x2\_min,x2\_max = X[:,1].min(), X[:,1].max()

xx1,xx2= np.meshgrid(np.linspace(x1\_min,x1\_max),np.linspace(x2\_min,x2\_max))

Z = svm.predict(np.c\_[xx1.ravel(), xx2.ravel()])

Z = Z.reshape(xx1.shape)

plt.scatter(X[pos,0],X[pos,1],marker ='+',c='k')

plt.scatter(X[neg,0],X[neg,1],marker ='o',c='y')

plt.contour(xx1,xx2,Z,[0.5],linewidths=1,colors='g')

def primerapartado():

data = scipy.io.loadmat('ex6data1.mat')

y = data['y']

X = data['X']

y = np.reshape(y,(51))

svm = SVC( kernel='linear', C=1.0)

svm.fit(X,y)

pintar(X,y,svm)

def segundoapartado():

data = scipy.io.loadmat('ex6data2.mat')

y = data['y']

X = data['X']

y = np.reshape(y,y.shape[0])

svm = SVC( kernel='rbf', C=1.0, gamma = 1/(2\*0.1\*\*2))

svm.fit(X,y)

pintar(X,y,svm)

def tercerapartado():

data = scipy.io.loadmat('ex6data3.mat')

y = data['y']

X = data['X']

yval = data['yval']

Xval = data['Xval']

y = np.reshape(y,y.shape[0])

a = np.array([0.01,0.03,0.1,0.3,1,3,10,30])

maxi = 0

Csol = 0

sigmasol= 0

for i in range(0,a.shape[0]):

for j in range(0,a.shape[0]):

svm = SVC( kernel='rbf', C=a[i], gamma = 1/(2\*a[j]\*\*2))

svm.fit(X,y)

w = svm.predict(Xval)

t = (w==yval[:,0])

p = (np.count\_nonzero(t)/yval.shape[0])\*100

text = 'C='+repr(a[i])+',sigma='+repr(a[j])+' .Porcentaje='+repr(p)

if(p>maxi):

Csol = a[i]

sigmasol = a[j]

maxi = p

print(text)

text = 'Mejor solucion: C = '+ repr(Csol)+', Sigma = '+repr(sigmasol)+ ' . % = ' +repr(maxi)

print(text)

svm = SVC(kernel='rbf', C=Csol, gamma = 1/(2\*sigmasol\*\*2))

svm.fit(X,y)

pintar(X,y,svm)

def cargar(directorio,numcorreos,vocdic,eSpam):

X = np.empty((numcorreos, 1899))

if eSpam:

y = np.ones((numcorreos,1))

else:

y = np.zeros((numcorreos,1))

frozenvoc = frozenset(vocdic)

for i in range(1,numcorreos):

email\_contents = codecs.open( '{0}/{1:04d}.txt'.format(directorio,i),'r',encoding='utf-8', errors='ignore').read()

email = email2TokenList(email\_contents)

for j in email:

if j in frozenvoc:

X[i,(vocdic.get(j)-1)] = 1

return X,y

def email():

dic = getVocabDict()

val = np.array([0.01,0.03,0.1,0.3,1,3,10,30])

spamX,spamy = cargar('spam',500,dic,1)

easyX,easyy = cargar('easy\_ham',2551,dic,0)

hardX,hardy = cargar('hard\_ham',250,dic,0)

Xent = np.vstack((spamX[:350],easyX[:1786],hardX[:175]))

yent = np.vstack((spamy[:350],easyy[:1786],hardy[:175]))

Xval = np.vstack((spamX[350:],easyX[1786:],hardX[175:]))

yval = np.vstack((spamy[350:],easyy[1786:],hardy[175:]))

maxilin = 0

Csollin = 0

for i in range(0,val.shape[0]):

svm = SVC( kernel='linear', C=val[i])

svm.fit(Xent,yent)

w = svm.predict(Xval)

t = (w==yval[:,0])

p = (np.count\_nonzero(t)/yval.shape[0])\*100

text = 'C='+repr(val[i])+'.Porcentaje='+repr(p)

if(p>maxilin):

Csollin = val[i]

maxilin = p

print(text)

text = 'Mejor solucion lineal: C = '+ repr(Csollin)+ ' . % = ' +repr(maxilin)

print(text)

maxigaus = 0

Csolgaus = 0

sigmasolgaus= 0

for i in range(0,val.shape[0]):

for j in range(0,val.shape[0]):

svm = SVC( kernel='rbf', C=val[i], gamma = 1/(2\*val[j]\*\*2))

svm.fit(Xent,yent)

w = svm.predict(Xval)

t = (w==yval[:,0])

p = (np.count\_nonzero(t)/yval.shape[0])\*100

text = 'C='+repr(val[i])+',sigma='+repr(val[j])+' .Porcentaje='+repr(p)

if(p>maxigaus):

Csolgaus = val[i]

sigmasolgaus = val[j]

maxigaus = p

print(text)

text = 'Mejor solucion gaussiana: C = '+ repr(Csolgaus)+', Sigma = '+repr(sigmasolgaus)+ ' . % = ' +repr(maxigaus)

print(text)

def main():

#primerapartado()

#segundoapartado()

#tercerapartado()

email()

main()