Práctica 4 AA

Support Vector Machines

By Marcos Castro Cacho & Inés Prieto Centeno

May 20, 2020

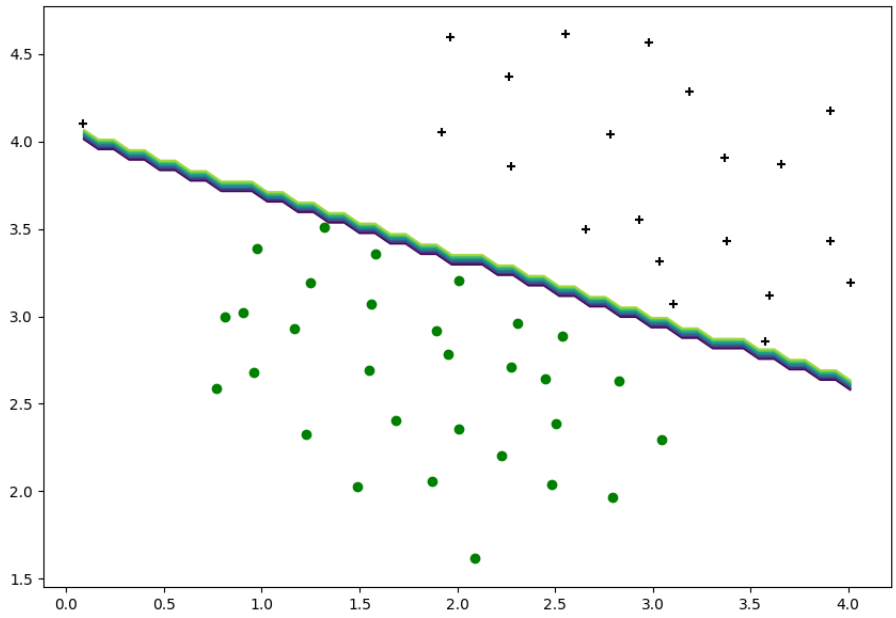
Chapter 1

Kernel Lineal

En esta primera parte de la practica nos familiarizamos con el concepto de kernel lineal. Con los datos provistos utilizamos el siguiente código para separar los datos.

```
from sklearn.svm import SVC
import matplotlib.pyplot as plt
from scipy.io import loadmat
import numpy as np

def paint(X, y, svm):
    pos = (y == 1).ravel()
    pos2 = (y == 0).ravel()

    plt.scatter(X[pos, 0], X[pos, 1], marker='+', c='k')
    plt.scatter(X[pos2,0],X[pos2,1], color='green')

    x1_min, x1_max = X[:, 0].min(), X[:, 0].max()
    x2_min, x2_max = X[:, 1].min(), X[:, 1].max()

    x1,x2 = np.meshgrid(np.linspace(x1_min, x1_max,len(X)), np.linspace(x2_min, x2_
    yp = svm.predict(np.array([x1.ravel(),x2.ravel()]).T).reshape(x1.shape)

    plt.contour(x1,x2, yp)
    plt.show()

def main():
    data = loadmat('ex6data1.mat')
    y = data['y']
    X = data['X']
    svm = SVC(kernel='linear', C=100)
    svm.fit(X,y.ravel())
    paint(X,y,svm)

main()
```

Tras ejecutar el código se nos presenta el siguiente resultado


4.5
4.0
3.5
3.0
2.5
2.0
1.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

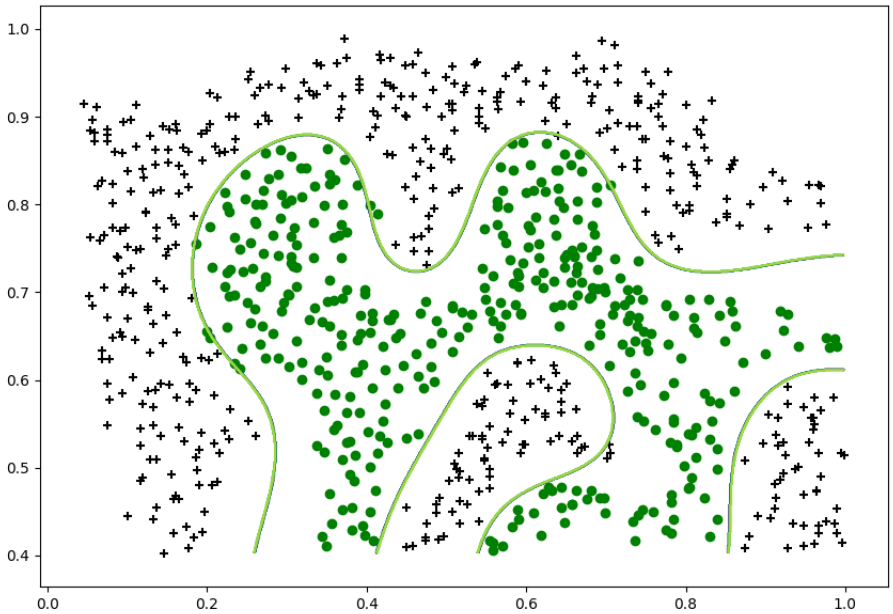
Chapter 2

Kernel gaussiano

En esta caso presentamos prácticamente el mismo código, salvo que en este caso estamos llamando a un kernel gaussiano en vez de a uno lineal. la nueva funcion main es la siguiente(paint no cambia)

```
def main():
    data = loadmat('ex6data2.mat')
    y = data['y']
    X = data['X']
    sigma = 0.1
    svm = SVC(kernel='rbf', C=1, gamma=1/(2*sigma**2))
    svm.fit(X,y.ravel())
    paint(X,y,svm)
```

Tras ejecutar el código con el nuevo sed de datos se nos presenta el siguiente resultado



Chapter 3

Detección de spam

En esta parte utilizaremos el tercer set de datos provisto, con objetivo de serán el spam en los correos. Primero implementamos las funciones addX y addY para recoger todos los datos de entrada.

```
def addX(directorio, X, Xval ,Xtest, numFicheros, dicVoc):
    for i in range(numFicheros):
        email_contents = codecs.open('{0}/{1:04d}.txt'.format(directorio,i+1), 'r',
        tokens  = email2TokenList(email_contents)
        arrayPalabras = np.zeros(1900)
        for palabra in tokens:
            if palabra in dicVoc:
                arrayPalabras[dicVoc[palabra]] = 1
        if(i + 1 <= int(numFicheros*0.7)):
            X = np.vstack((X,arrayPalabras))
        elif (i + 1 <= int(numFicheros*0.9)):
            Xval = np.vstack((Xval,arrayPalabras))
        else:
            Xtest = np.vstack((Xtest,arrayPalabras))
    return X,Xval,Xtest

def addY(numFicherosSpam, numFicherosNoSpam):
    y_ones = np.ones(int(numFicherosSpam))
    y_zeros = np.zeros(int(numFicherosNoSpam))
    return np.append(y_ones, y_zeros)
```

Después de tener todos los datos en el programa, los procesamos de la siguiente manera.

```
def main():
    np.set_printoptions(threshold=sys.maxsize)
    directorio = "spam"
    dicVoc = getVocabDict()
    num_spam = 500
    num_easy_ham = 2501
    num_hard_ham = 250
    X = np.empty((0,1900))
    Xval = np.empty((0,1900))
```

```
Xtest = np.empty((0,1900))
X, Xval, Xtest = addX("spam", X,Xval,Xtest, num_spam, dicVoc)
X, Xval, Xtest = addX("easy_ham", X,Xval,Xtest, num_easy_ham, dicVoc)
X, Xval, Xtest = addX("hard_ham", X,Xval,Xtest, num_hard_ham, dicVoc)

y = addY(num_spam*0.7, (num_easy_ham+num_hard_ham)*0.7)
yval = addY(num_spam*0.2,(num_easy_ham+num_hard_ham)*0.2)
ytest = addY(num_spam*0.1,(num_easy_ham+num_hard_ham)*0.1)

print(X e Y a adido perfectamente)

array = np.array([0.01,0.03,0.1,0.3,1,3,10,30])
minimo = np.inf
c,g = 0,0
for i in array:
    print("C: ", i)
    for j in array:
        print("sigma: ", j)
        sigma = j
        svm = SVC(kernel='rbf', C=i, gamma=1/(2*sigma**2))
        svm.fit(X,y.ravel())
        pred = svm.predict(Xval)
        err = np.sum(pred != yval.ravel())/np.size(yval.ravel(), 0)
        if(minimo > err):
            minimo = err
            c = i
            g = j
svm = SVC(kernel='rbf', C=c, gamma=1/(2*g**2))
svm.fit(X,y.ravel())
print(svm.predict(Xtest))
err = np.sum(pred != ytest.ravel())/np.size(ytest.ravel(), 0)
print(err)
```

Iteramos en todos los valores de Alpha y C y después de un rato obtenemos el siguiente valor de