Aprendizaje automático y minería de datos: Práctica 5

Jorge Rodríguez García y Gonzalo Sanz Lastra

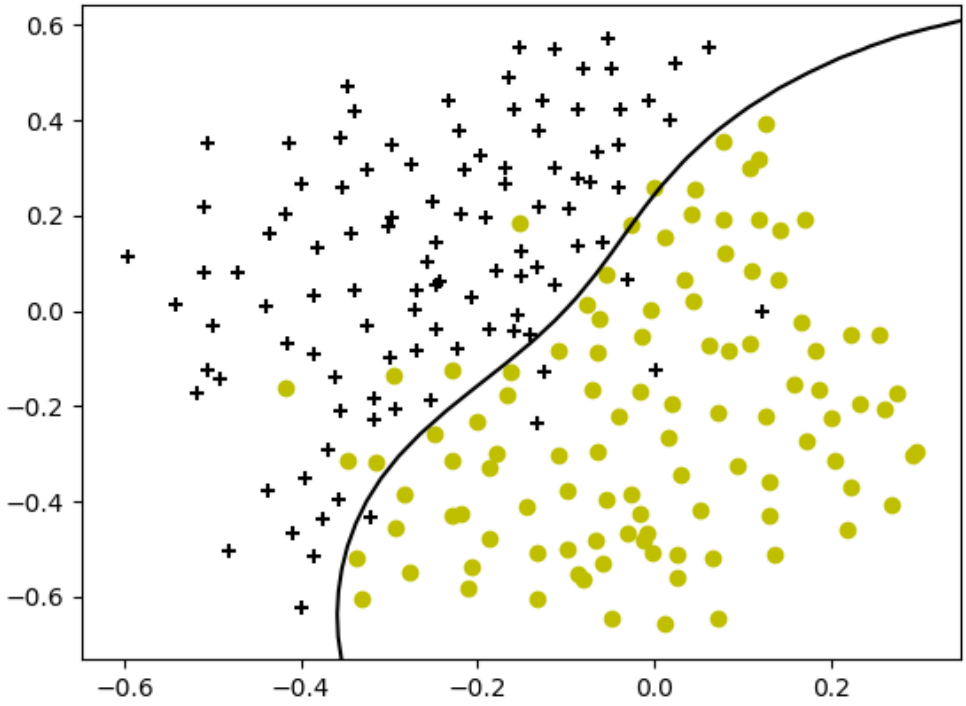
Código de la práctica:

Parte 1:

```
def load_mat(file_name):
    """carga el fichero mat especificado y lo devuelve en una matriz
data"""
    return loadmat(file_name)

def functionGraphic(X, Y, clf):
    # pintamos muestras de entrenamiento

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

    pos = np.where(Y==0)[0]
    plt.scatter(X[pos, 0], X[pos, 1], marker='o', c= 'y')


    ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    # create grid to evaluate model
    xx = np.linspace(xlim[0], xlim[1], 30)
    yy = np.linspace(ylim[0], ylim[1], 30)
    YY, XX = np.meshgrid(yy, xx)
    xy = np.vstack([XX.ravel(), YY.ravel()]).T
    Z = clf.decision_function(xy).reshape(XX.shape)

    # plot decision boundary and margins
    ax.contour(XX, YY, Z, colors='k', levels=[0]
    #, alpha=0.5, linestyles=['--', '-', '--']
    )

    # plot support vectors
    #ax.scatter(clf.support_vectors_[:, 0], clf.support_vectors_[:, 1],
s=100,
    #        linewidth=1, facecolors='none', edgecolors='k')

    plt.savefig('TrainingExamples.png')# plot the decision function
```

```
    plt.show()

def main():
    valores = load_mat("ex6data3.mat")

    X = valores['X']         # datos de entrenamiento
    y = valores['y']

    Xval = valores['Xval']         # datos de validacion
    yval = valores['yval']

    C = 0.01
    sigma = 0.01

    maxCorrects = 0
    bestC = C
    bestSigma = sigma

    # probamos la mejor combinacion de C y sigma que de el menor error
    for i in range(8):
        C = C * 3
        sigma = 0.1
        for j in range(8):
            sigma = sigma * 3
            clf = SVC(kernel='rbf', C=C, gamma= 1/(2*sigma**2))
            clf.fit(X, y)
            corrects = (yval[:, 0] == clf.predict(Xval)).sum()

            if maxCorrects < corrects:
                maxCorrects = corrects
                bestC = C
                bestSigma = sigma

    clf = SVC(kernel='rbf', C=bestC, gamma= 1/(2*bestSigma**2))
    clf.fit(X, y)

    functionGraphic(X, y, clf)

main()
```

Ajuste después de aplicar SVM con las mejores C y sigma:



Parte 2:

```
def emailToWordOcurrence(email, wordsDict):
    result = np.zeros(len(wordsDict))

    for i in range(len(email)):
        if email[i] in wordsDict:
            index = wordsDict.get(email[i]) - 1
            result[index] = 1
    return result

def dataManager(ini, fin, directoryName, yValue):
    X = np.empty((0, 1899)) # 60% de 500
    Y = np.empty((0, 1))

    for i in range(ini + 1, fin + 1):
        email_contents =
codecs.open('{0}/{1:04d}.txt'.format(directoryName, i), 'r',
        encoding='utf-8', errors='ignore').read()
        email = email2TokenList(email_contents)

        wordsDict = getVocabDict()

        wordOcurrence = emailToWordOcurrence(email, wordsDict)
        X = np.vstack((X, wordOcurrence))
```

```
        Y = np.vstack((Y, yValue))

    return X, Y

def main():
    Xspam, Yspam = dataManager(0, 300, "spam", 1)
    Xeasy, Yeasy = dataManager(0, 1531, "easy_ham", 0)
    Xhard, Yhard = dataManager(0, 150, "hard_ham", 0)

    X = np.vstack((Xspam, Xeasy))
    X = np.vstack((X, Xhard))
    Y = np.vstack((Yspam, Yeasy))
    Y = np.vstack((Y, Yhard))

    XspamVal, YspamVal = dataManager(300, 400, "spam", 1)
    XeasyVal, YeasyVal = dataManager(1531, 2041, "easy_ham", 0)
    XhardVal, YhardVal = dataManager(150, 200, "hard_ham", 0)

    Xval = np.vstack((XspamVal, XeasyVal))
    Xval = np.vstack((Xval, XhardVal))
    Yval = np.vstack((YspamVal, YeasyVal))
    Yval = np.vstack((Yval, YhardVal))

    XspamTest, YspamTest = dataManager(400, 500, "spam", 1)
    XeasyTest, YeasyTest = dataManager(2041, 2551, "easy_ham", 0)
    XhardTest, YhardTest = dataManager(200, 250, "hard_ham", 0)

    Xtest = np.vstack((XspamTest, XeasyTest))
    Xtest = np.vstack((Xtest, XhardTest))
    Ytest = np.vstack((YspamTest, YeasyTest))
    Ytest = np.vstack((Ytest, YhardTest))

    bestC = 21.87
    bestSigma = 8.1

    clf = SVC(kernel='rbf', C=bestC, gamma= 1/(2*bestSigma**2))
    clf.fit(X, Y)

    corrects = (Ytest[:, 0] == clf.predict(Xtest)).sum()

    print((corrects / Xtest.shape[0])*100)

main()
```