6 SUPPORT VECTOR MACHINES (SVM)

June 19, 2019

In [1]:
```
#José Antonio Garrido Sualdea
```

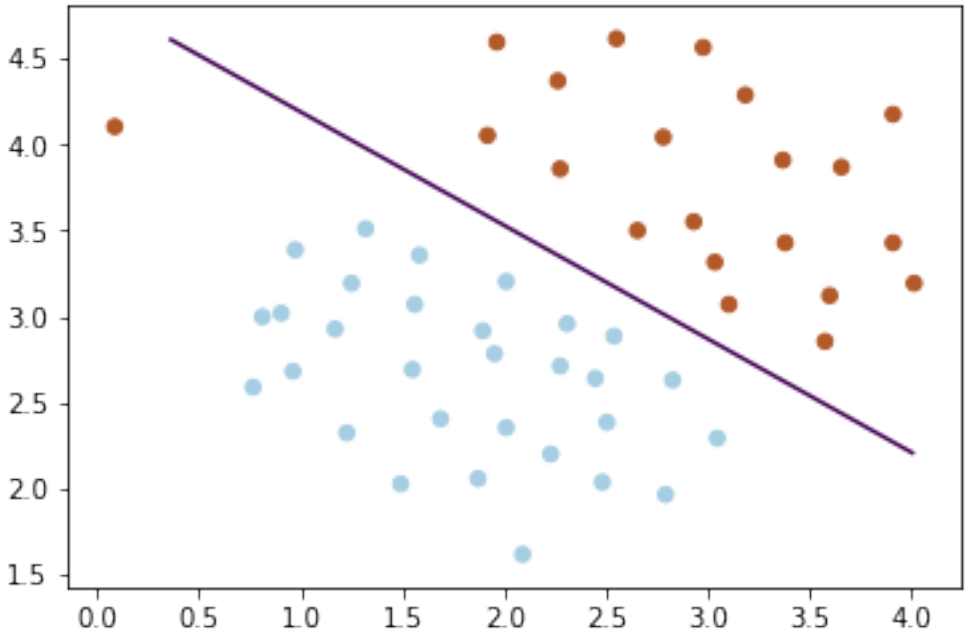
In [2]:
```
##############################
#SUPPORT VECTOR MACHINES(VMS)
##############################
```

In [3]:
```
from sklearn.svm import SVC
from scipy.io import loadmat
import matplotlib.pyplot as plt
import numpy as np
```

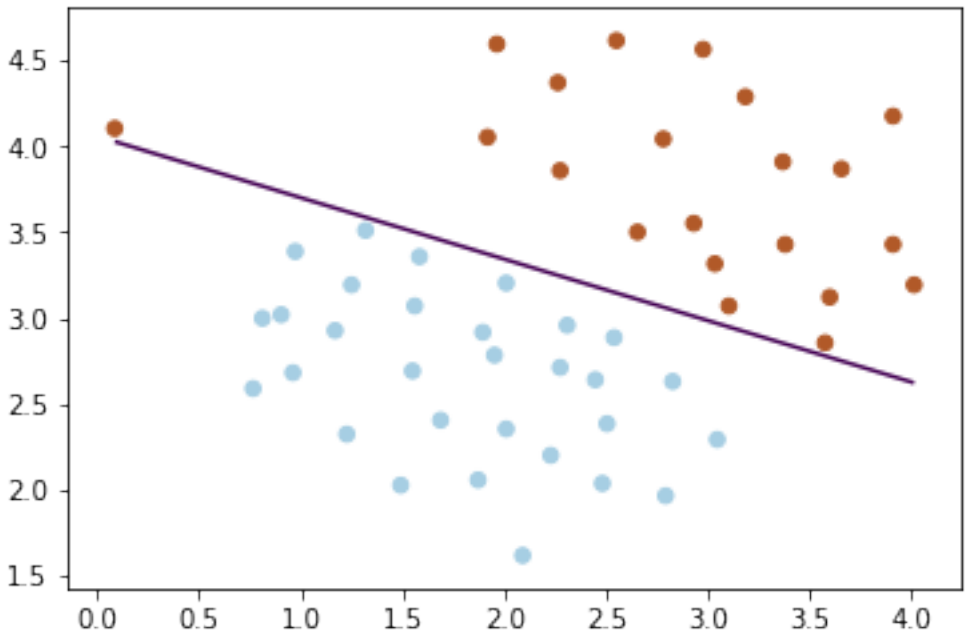
In [4]:
```
def plot_model(model, X, y):
    x_min = X[:, 0].min()
    x_max = X[:, 0].max()
    y_min = X[:, 1].min()
    y_max = X[:, 1].max()
    XX, YY = np.mgrid[x_min:x_max:200j, y_min:y_max:200j]
    Z = model.decision_function(np.c_[XX.ravel(), YY.ravel()])
    Z = Z.reshape(XX.shape)
    plt.contour(XX, YY, Z, levels=[0])
    plt.scatter(X[:,0], X[:,1], c=y, s=30, cmap = plt.cm.Paired)
```

In [5]:
```
data_1 = loadmat ('ex6data1.mat')
X_1 = data_1['X']
y_1 = np.ndarray.flatten(data_1['y'])
```

In [6]:
```
svm_1_1 = SVC(kernel='linear', C=1)
svm_1_1.fit(X_1, y_1)
plot_model(svm_1_1, X_1, y_1)
```

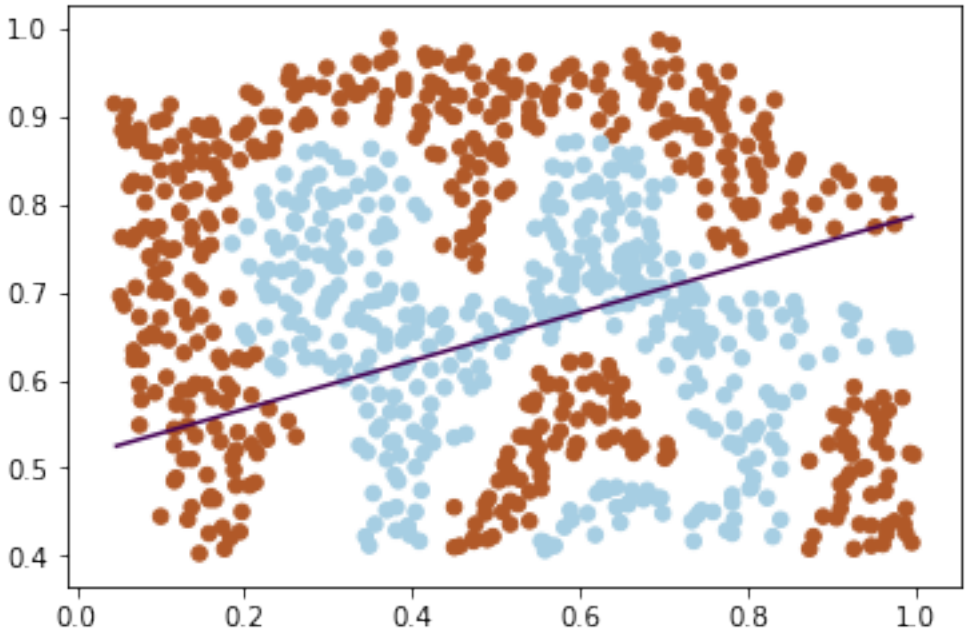


In [7]:
```
svm_1_2 = SVC(kernel='linear', C=100)
svm_1_2.fit(X_1, y_1)
plot_model(svm_1_2, X_1, y_1)
```

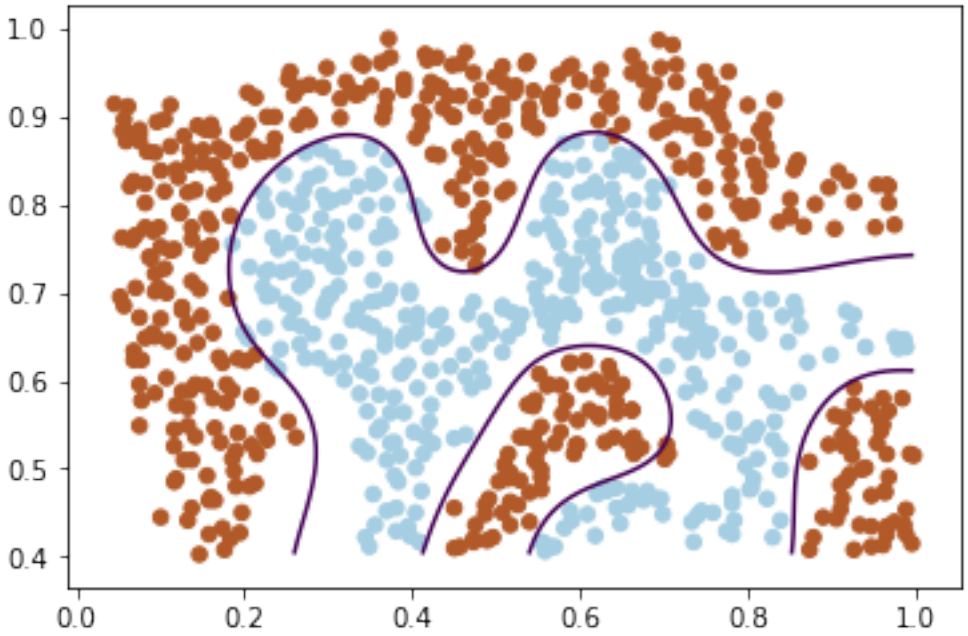


In [8]:
```
data_2 = loadmat ('ex6data2.mat')
X_2 = data_2['X']
y_2 = np.ndarray.flatten(data_2['y'])
```

In [9]:
```
svm_2_1 = SVC(kernel='linear', C=1)
svm_2_1.fit(X_2, y_2)
plot_model(svm_2_1, X_2, y_2)
```



In [10]:
```
C = 1
sigma = 0.1
svm_2_1 = SVC(kernel='rbf', C=C, gamma=1/(2*sigma**2))
svm_2_1.fit(X_2, y_2)
plot_model(svm_2_1, X_2, y_2)
```

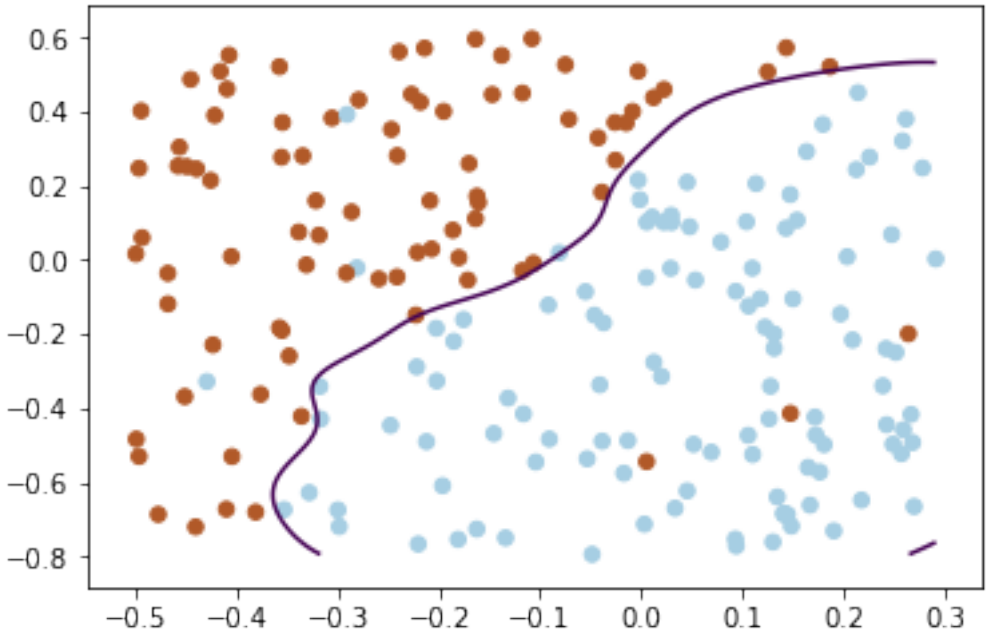



```
In [11]: #X, y de entrenamiento; Xval, yval de validación
         data_3 = loadmat ('ex6data3.mat')
         X_3 = data_3['X']
         y_3 = np.ndarray.flatten(data_3['y'])
         Xval_3 = data_3['Xval']
         yval_3 = np.ndarray.flatten(data_3['yval'])

In [12]: max_score = 0
         best_C = None
         best_sigma = None
         best_model3 = None
         for C in [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30]:
             for sigma in [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30]:
                 svm_3 = SVC(kernel='rbf', C=C, gamma=1/(2*sigma**2))
                 svm_3.fit(X_3, y_3)
                 score = svm_3.score(Xval_3, yval_3)
                 if score > max_score:
                     max_score = score
                     best_C = C
                     best_sigma = sigma
                     best_model3 = svm_3
         print('score: {}, C: {}, sigma: {}'.format(max_score, best_C, best_sigma))

score: 0.965, C: 1, sigma: 0.1
```

In [13]:
```
plot_model(best_model3, Xval_3, yval_3)
```



In [14]:
```
import codecs
from collections import Counter
from get_vocab_dict import getVocabDict
from process_email import email2TokenList
vocab =  getVocabDict()
```

In [15]:
```
def get_features(file_name):
    #contenido de un email en crudo (con etiq html...)
    email_contents = codecs.open(
        file_name, 'r', encoding='utf 8', errors='ignore'
        ).read()
    #palabras limpias y preprocesadas de un correo
    email = email2TokenList(email_contents)
    #diccionario que nos dice cada palabra y cuántas veces aparece cada una
    words = Counter(email)
    #vector tipo(vacio) dónde vamos a guardar la representación
    #del email vectorizada
    x = [0]*1899

    #relleno el vector
    for (word, count) in words.items():
        if word in vocab:
            i = vocab[word]
            x[i - 1] = count
```

```
            return x
```

In [16]:
```
#cargamos los datos marcándolos como spam o no en y
import os
X = []
y = []
spam_folder = 'spam/'
for file_name in os.listdir(spam_folder):
    X.append(get_features(spam_folder + file_name))
    y.append(1)
hard_folder = 'hard_ham/'
for file_name in os.listdir(hard_folder):
    X.append(get_features(hard_folder + file_name))
    y.append(0)
easy_folder = 'easy_ham/'
for file_name in os.listdir(easy_folder):
    X.append(get_features(easy_folder + file_name))
    y.append(0)
```

In [17]:
```
#separamos los datos aleatoriamente en datos de entrenamiento
#y datos de validación
from sklearn.model_selection import train_test_split
X_train, X_val, y_train, y_val = train_test_split(
    X, y, test_size=0.33, random_state=0
    )
```

In [18]:
```
max_score = 0
best_C = None
best_sigma = None
best_model = None
for C in [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30]:
    for sigma in [0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30]:
        svm_4 = SVC(kernel='rbf', C=C, gamma=1/(2*sigma**2))
        svm_4.fit(X_train, y_train)
        score = svm_4.score(X_val, y_val)
        if score > max_score:
            max_score = score
            best_C = C
            best_sigma = sigma
            best_model = svm_4
print('score: {}, C: {}, sigma: {}'.format(max_score, best_C, best_sigma))
```

```
score: 0.9678899082568807, C: 30, sigma: 30
```

In [19]:
```
best_model.predict([get_features('spam/0001.txt')])
```

Out[19]:
```
array([1])
```