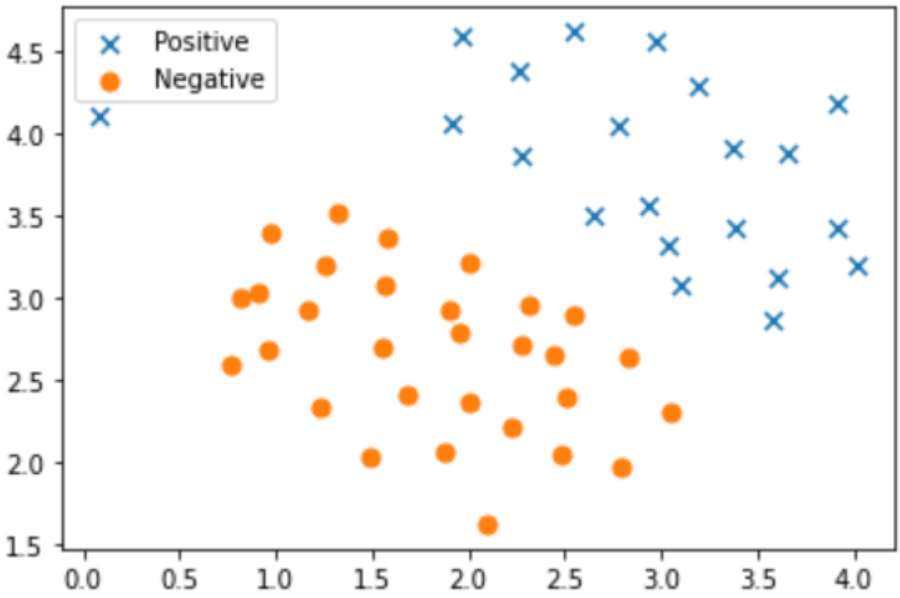
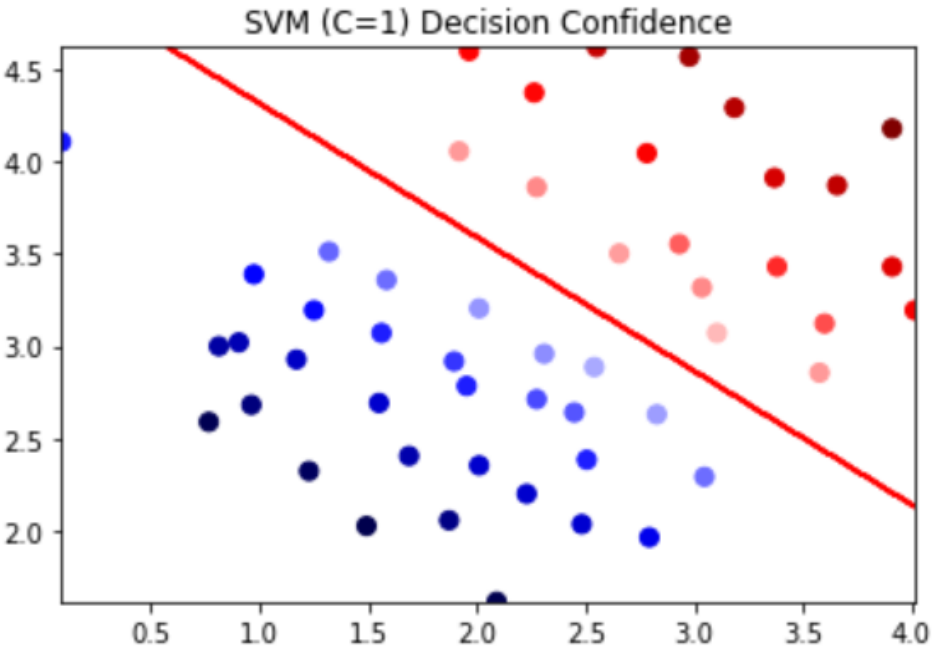
hw6

1. linear decision boundary : data overview



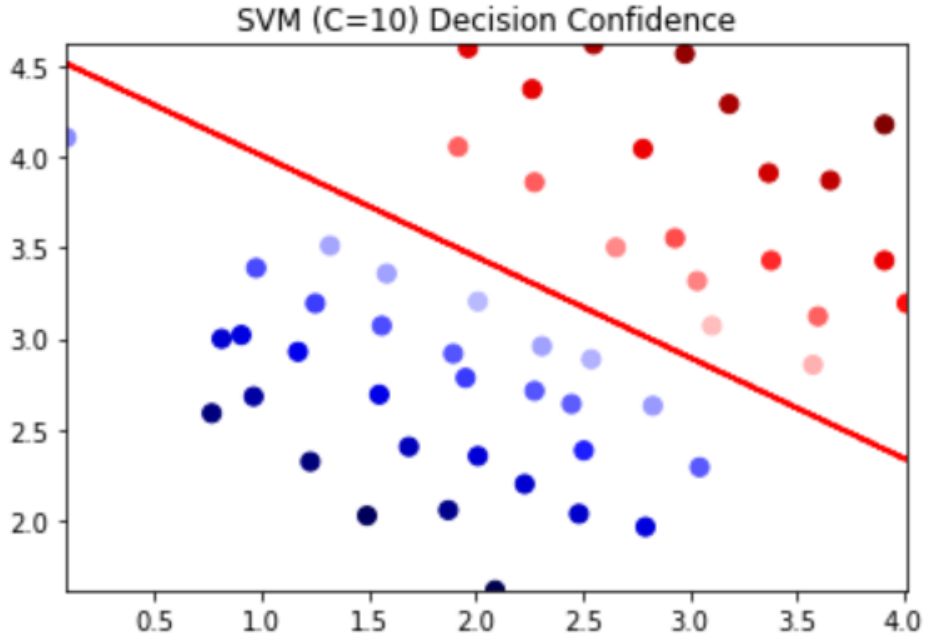
2. C's influence on result

C = 1



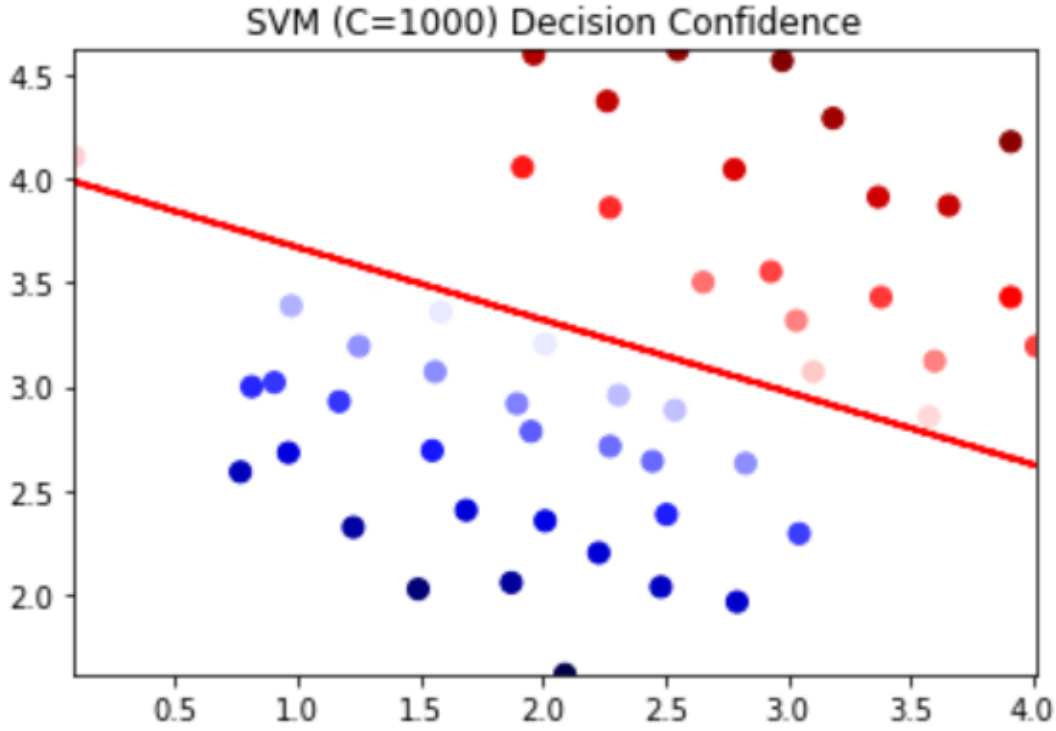
这是 C = 1 时的决策边界，可以看出效果还算比较好

C = 10



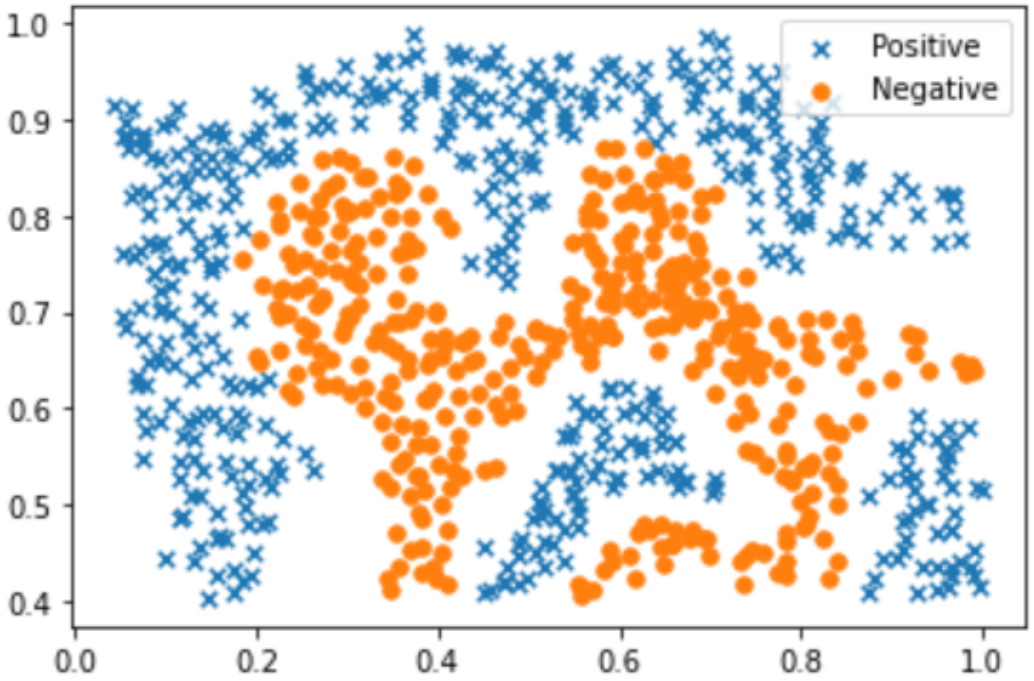
C = 10 时决策边界开始有点倾向平缓了，有过拟合的趋势

C = 1000



C = 1000， 这时已经出现比较严重的过拟合了

3. non linear decision boundary : data overview

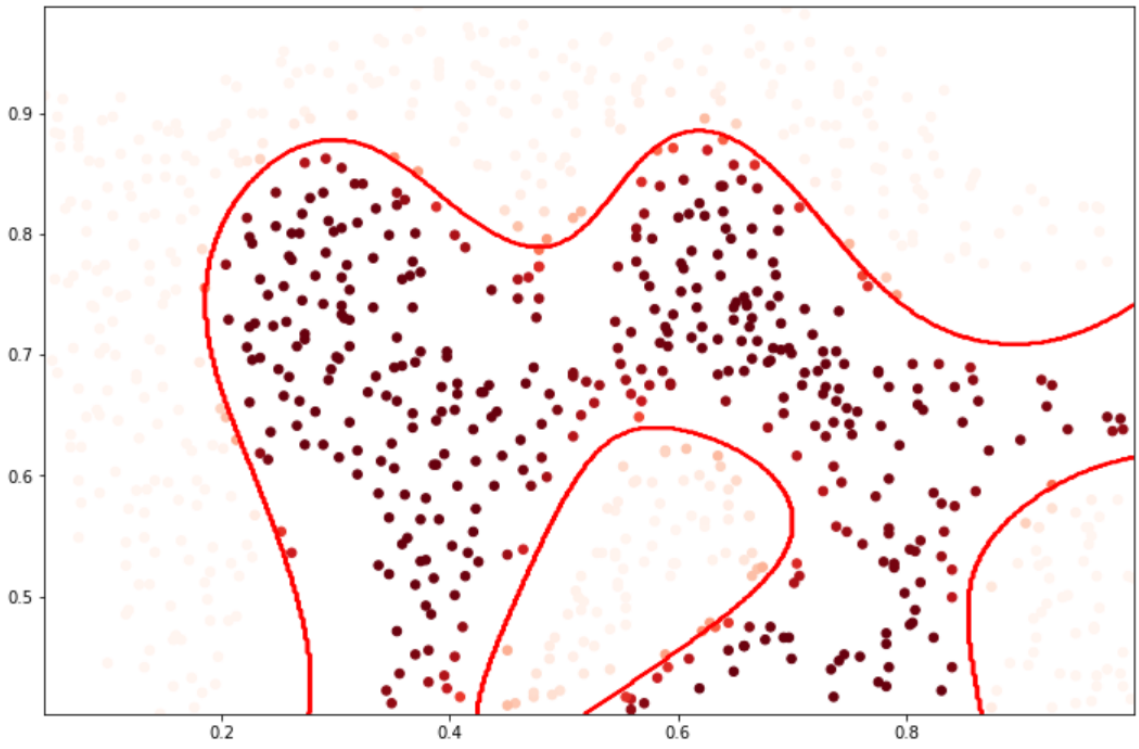


4. about parameters

```
svc3 = svm.SVC(C=100, gamma=10, kernel='rbf', probability=True)
# 查阅svm.SVC相关资料，回答此处的gamma参数和rbf核函数的关系
# gamma代表：γ = 1 / (2σ2)：γ 越大高斯分布越窄，样本分布越集中；γ 越小高斯分布越宽，样本分布越密集；
clfs3=svc3.fit(data[['X1', 'X2']], data['y'])
svc3.score(data[['X1', 'X2']], data['y'])
```

```
0.9698725376593279
```

5. decision boundary



6. grid search

```
for C_val in C_values:
    for gamma_val in gamma_values:
        #此处补充完善3行代码，完成网格搜索
        svc = svm.SVC(C=C_val, gamma=gamma_val, max_iter=1000)
        svc.fit(X, y)
        score = svc.score(Xval, yval)
        #补充结束
```

```
Out[136]: (0.965, {'C': 0.3, 'gamma': 100})
```

7. spam email : data overview

```
X = spam_train['X']
Xtest = spam_test['Xtest']
y = spam_train['y'].ravel()
ytest = spam_test['ytest'].ravel()

# 观察X, y, Xtest, ytest 形状
X.shape, y.shape, Xtest.shape, ytest.shape
```

```
((4000, 1899), (4000,), (1000, 1899), (1000,))
```

8. train and test

```
#这里使用svm.SVC的默认参数来进行训练，请回答C和gamma的默认值是多少
svc = svm.SVC(C=2, gamma='scale')
svc.fit(X, y)
print('Training accuracy = {0}%'.format(np.round(svc.score(X, y) * 100, 2)))
# C = 1,  auto= 1 / n_features = 1 / 1899
```

```
Training accuracy = 99.68%
```

```
print('Test accuracy = {0}%'.format(np.round(svc.score(Xtest, ytest) * 100, 2)))
```

```
Test accuracy = 99.1%
```

通过调整 C 和 gamma 使测试集上的准确率提高到了 99.1 %