Exercise 3 logistic regression

資工三 A　109502009　吳尚明

1.report the accuracy and precision and recall for both the training and the testing data.

When epochs = 10000, alpha = 0.01

```
Logreg train accuracy: 0.976422
Logreg train precision: 0.938830
Logreg train recall: 0.810563
Logreg test accuracy: 0.978322
Logreg test precision: 0.938838
Logreg test recall: 0.799479
```

```
[[  0.        ]
 [ -9.60770821]
 [  0.44053858]
 [ 16.9074481 ]
 [  0.80578103]
 [ -5.81422069]
 [  2.75235616]
 [-11.04834491]
 [  3.6384396 ]]
```

When epochs = 10000, alpha = 0.001

```
Logreg train accuracy: 0.975081
Logreg train precision: 0.937838
Logreg train recall: 0.796785
Logreg test accuracy: 0.977763
Logreg test precision: 0.943838
Logreg test recall: 0.787760
```

```
[[  0.        ]
 [ -5.73659956]
 [ -2.99633919]
 [ 33.41568481]
 [-22.4210492 ]
 [ -7.61668477]
 [ -0.18734192]
 [-28.09504679]
 [ 22.69815388]]
```

以上是 alpha 和 epochs 變動時出現的兩個較好的結果，我會選擇 alpha=0.01，雖然 test precision 較低，但 test accuracy 較高。

2.A brief discussion of the results

我原本使用非常慢的運算方法

```
for j in range(0,9):
    for  i in range(0,8949):
        add+=(predict_prob(X[i], theta)-y[i])*X[i][j]
    theta[j]=theta[j]-add*alpha/8949
    add=0
```

這個方法計算的次數不能太高，因此結果也不夠準確，但後改成矩陣相乘的方法效果十分顯著。

將 theta 的公式利用矩陣相乘的方法大大提升計算速度，為我的 epochs 爭取更高的次數，以達到較好的結果:

```
for r in range(0,epochs):
    theta=theta-(alpha)*numpy.dot(X.transpose(),(predict_prob(X,theta)-y))
```

```
Logreg train accuracy: 0.976422
Logreg train precision: 0.938830
Logreg train recall: 0.810563
Logreg test accuracy: 0.978322
Logreg test precision: 0.938838
Logreg test recall: 0.799479
```

先比較 precision 和 recall ，precision=True Positive/Total Predicted Positive，recall=True Positive/Total Actual Positive，雖然 precision 和 recall 都是越高越好，但從結果可以看到，這兩項數值常常是不能兼顧的，從 precision 高 recall 低的狀況可以看出我們假陽性率是低的，但假陰性率就稍稍沒那麼低。

表現最為出色的是 accuracy，代表我們在所有情況中正確判斷真假的比例是相當高的。

以下是我的 ROC 圖形:可以看到曲線下的面積(AUC)是很高的