HW2

1. Logistic regression function

```
def sigmod(z):
    z=np.matrix(z)
    return 1/(1+np.exp(-z))

def crossentropy(X,y,theta,londa):
    temp = sigmod(X*theta)
    y=np.matrix(y)
    m=y.shape[0]
    X=np.matrix(X)
    tempy = np.ones(shape=(y.shape))-y
    tempx = np.ones(shape=(temp.shape))-temp
    reg = londa*(np.sum(np.square(theta[1:,:])))/2
    return -(((y.T*np.log(temp))+(tempy).T*np.log(tempx))+reg)/m

 def gradientdescent(X, y, theta, alpha, num_iters, londa):
    import numpy as np
    X = np.matrix(X)
    y = np.matrix(y)
    theta = np.matrix(theta)
    m = X.shape[0]
    adagrad = np.zeros(theta.shape)
    for i in range(num_iters):
        temp = X.T * (sigmod(X * theta) - y)
        temp[1:] += londa * theta[1:]
        temp = (alpha/m) * temp
        adagrad += np.square(temp)
        temp = temp/np.sqrt(adagrad)
        theta -= temp
    return theta, adagrad
```

2. The another method

另一種方法我採用 Probability Generative model，使用的是 Gaussian Distribution，要先求得 training data 之每個 features X 於 class1 or class0 的 Means 和 Covariance，並將 class1 與 class2 的 Covariance 依照其 data 數之比例做加權平均，讓 class1 和 class2 共用同一個 Covariance，之後將每筆 test data 之 features 代入公式(如下) ，即可求得 P⟨$C1|X$⟩作為預測每筆 test data 參考，若機率為>=0.5 則為 class1，否則為 class2。

$$\Sigma_1 = \Sigma_2 = \Sigma$$

$$a = (\mu^1 - \mu^2)^T \Sigma^{-1} x - \frac{1}{2}(\mu^1)^T (\Sigma^1)^{-1} \mu^1 + \frac{1}{2}(\mu^2)^T (\Sigma^2)^{-1} \mu^2 + ln\frac{N_1}{N_2}$$

$$P(C_1|x) = \sigma(a)$$

以下為程式碼片段：

```
from numpy import linalg
cov1=np.cov(trainData1.T)
cov0=np.cov(trainData0.T)
cov=np.matrix(cov0*(n0/m)+cov1*(n1/m))
mu = np.zeros(shape=(n,2))
for i in range(n):
    mu[i,0] = np.mean(trainData0[:,i])
    mu[i,1] = np.mean(trainData1[:,i])
pinv_cov = linalg.pinv(cov)
w = np.matrix(mu[:,0]-mu[:,1]).T * pinv_cov
b = (-mu[:,0].T * pinv_cov*mu[:,0] + mu[:,1].T * pinv_cov*mu[:,1])/2 +
math.log(n0/n1)
```

3. Discussion

於 public test data set 的 accuracy，logistic regression 比 probability generative model 還要好，當我做 regression 時，先將 training data 做 feature scaling，而執行 gradient descent 時，初始 weight 全部設為 0，learning rate 設為 0.05，iteration 設為 5000 次，並且沒加入 regularization 時，score 為 0.93，而 generative model 的 score 則為 0.87667。由此可見，單從 training data 的機率分布做預測是不夠精確的。

另外，我將 logistic regression 加入 regularization 做些不同權重(lomda)對於 accuracy 的影響，並將 training data set 的前 2/3 作為 training set，而後 1/3 作為 validation set，如下圖所示。藍色線為使用 training set，紅色線為使用 validation set。