R05921063 陳定楷 HW2 討論對象：F04942066劉楚彤、R04942143吳兆倫

1. **Logistic regression**
   pseudo code of training model class

```
def sigmoidFunc(x):
        return 1./(1+exp(-x))
sigma_b_sq = 0
sigma_w_sq = 0
w = np.ones((58,1))
bias = 0
for iter in range(10000):
        z = bias+train_in*w
        err = train_out-sigmoidFunc(z)
        pdv_w = ((-1)*err.T*train_in).T
        pdv_b = (-1)*err.sum()
        sigma_w_sq += pdv_w**2
        sigma_b_sq += pdv_b**2
        w = w-eta*pdv_w/(sigma_w_sq**0.5)
        bias = bias-eta*pdv_b/(sigma_b_sq**0.5)
test_z = bias+test_in*w
test_out = (sigmoidFunc(test_z)>0.5).astype(int)
```

   iteration = 10000，learning rate = 0.2，train_in和train_out為使用3-fold cross validation後的training data，test_in是testing data，test_out則是所求之答案。sigma_b_sq和sigam_w
   需的偏微分平方積。使用的features總共有58維，分別是data原先提供的57種attributes，再加上第44和第51種attribute(word frequency of ”project”和char_freq_[)的三次方和作為第58種attribute。因此w為一個58維的weighting vector。

2. **Method 2: Generative model**
   pseudo code of training model class

```
P_c0 = num_0/(num_0+num_1)
P_c1 = num_1/(num_0+num_1)
mean_0 = sum(trainIn_0)/num_0
mean_1 = sum(trainIn_1)/num_1
cov_0 = ((trainIn_0.T-mean_0)*(trainIn_0.T-mean_0).T)/num_0
cov_1 = ((trainIn_1.T-mean_1)*(trainIn_1.T-mean_1).T)/num_1
cov = P_c0*cov_0+P_c1*cov_1
detCov = np.linalg.det(cov)**0.5
invCov = np.linalg.inv(cov)

def P_xc0(x):
        z = -0.5*(x-mean_0).T*invCov*(x-mean_0)
        return (1/(2*np.pi)**(fNum/2))*(1/detCov)*exp(z)
def P_xc1(x):
        z = -0.5*(x-mean_1).T*invCov*(x-mean_1)
```

```
        return (1/(2*np.pi)**(fNum/2))*(1/detCov)*exp(z)
def P_c1x(x):
        if (P_xc0(x)*P_c0+P_xc1(x)*P_c1)==0:
                return 0        # Default class = 1
        else:
                return P_xc1(x)*P_c1/(P_xc0(x)*P_c0+P_xc1(x)*P_c1)

test_out = (P_c1x(test_in)>0.5).astype(int)
```

Class 0和class 1分別為"non-spam"和"spam"，num_0和num_1分別為兩個class在training data中的數量，mean_0和mean_1為每個attribute的mean，cov_0和cov_1則是兩個class的covarianc 我class 1與class 0使用相同的covariance cov，detCov和invCov分別是cov的determinant平方根以及inverse。trainIn和test_in分別是training data和testing data的input，使用的是原始57維的features，test_out為所求之答案。
兩種方法的比較在3.Discussion中說明。

3. **Discussion**
   本次使用了三種model與input features的組合

   (a) logistic regression+58-dimension features data(如1.所述)
   public score: 0.93000, private score: 0.91333

   (b) logistic regression+57-dimension features data(original data)
   public score: 0.92333, private score: 0.91667

   (c) generative model+57-dimension features data(如3.所述)
   public score: 0.87667, private score: 0.86000

   其中(c)在public/private leaderboard上正確率皆低於(a)和(b)，因此generative model較前兩者來得差，但training的速度比(a)(b)都來得快。
   (a)中兩種features的三次方積是測試過feature (attr1, attr2)到(attr56, attr57)的所有組合後，選出testing accurate最高者(attr44, attr51)。想法是：可能有某幾種字的組合出現時，是垃圾信件的機率就上升。雖然在testing accurate和public score上都比(b)高，但是private score的分數低於(b)，應該是有overfitting的狀況發生。