EE5907 Programming Assignment Q2

Lee Jianwei A0018867

In [1]:
```
from __future__ import division
import scipy.io
from scipy.stats import norm
import numpy as np
import matplotlib.pyplot as plt

from multiprocessing import Pool
np.set_printoptions(precision=2, suppress=True)
```

Data Processing

Load

In [2]:
```
d = scipy.io.loadmat('spamData.mat')
ytest = d['ytest'].flatten()
ytrain = d['ytrain'].flatten()
xtest = d['Xtest']
xtrain = d['Xtrain']
```

In [3]:
```
#z-normalise features
def znorm1D(array1D):
    m = np.mean(array1D)
    s = np.std(array1D)
    return np.array((array1D-m)/s)
def znorm2D(array2D):
    """znorm along columns of 2D array"""
    znormed_columns = np.array([znorm1D(c) for c in array2D.T]) # each column in array2D
is a row (called c) in array2D.T
    return znormed_columns.T
xtrainZ = znorm2D(xtrain)
xtestZ = znorm2D(xtest)
```

In [113]:
```
# test znorm works:
f = 35 # feature number 10
print np.mean(xtrainZ[:,f]), np.std(xtrainZ[:,f])
```

```
-8.11386484555e-18 1.0
```

In [6]:
```
#log-transform features
def log2D(array):
    """array: list of emails, each 57 features long"""
    return np.array([np.log(x+0.1) for x in array]) #x is a 57 element array
xtrainLog_ = log2D(xtrain)
xtestLog_ = log2D(xtest)
xtrainLog = xtrainLog_
xtestLog = xtestLog_
```

In [7]:
```
xtrainLog_
```

Out[7]:
```
array([[-2.3 , -2.3 , -2.3 , ...,  0.43,  1.81,  3.5 ],
       [-2.3 , -2.3 , -2.3 , ...,  0.1 ,  0.1 ,  1.63],
       [-1.71, -1.71, -0.15, ...,  2.03,  6.51,  7.25],
       ...,
       [-2.3 , -1.61, -0.92, ...,  2.11,  5.65,  7.54],
       [-2.3 , -2.3 , -2.3 , ...,  1.36,  2.41,  4.11],
       [-2.3 , -2.3 , -2.3 , ...,  2.31,  5.53,  6.67]])
```

Z-norm

MLE from training Data

In [12]:
```
classMLE = np.average(ytrain)
```

In [ ]:
```
%matplotlib notebook
i = 36
plt.figure()
plt.hist(xtrainZ[:,i][ytrain==1],100,
#          range=(np.min(xtrainZ[ytrain==1][:,i]),np.max(xtrainZ[ytrain==1][:,i]))
        );
plt.axvline(xtrainZ_mean_spam[i])
# plt.semilogx()
plt.show()
```

In [192]:
```
# training data feature MLE
xtrainZ_mean_spam = np.mean(xtrainZ[ytrain==1],axis=0)
xtrainZ_sigma_spam = np.std(xtrainZ[ytrain==1],axis=0)
xtrainZ_mean_notspam = np.mean(xtrainZ[ytrain==0],axis=0)
xtrainZ_sigma_notspam = np.std(xtrainZ[ytrain==0],axis=0)
```

In [214]:
```
print xtrainZ.shape, len(xtrainZ_mean_spam)
```

```
(3065, 57) 57
```

In [194]:
```
def pdf(x,mean,sigma):
    z = (x-mean)/sigma
    return 1/np.sqrt(2*3.142)/sigma*np.exp(-z**2/2)
```

In [195]:
```
 def classifyZ(email):
    """
    BASED ON MLE
    classifies each email according to a gaussian likelihood,
    parameterised by the plug-in estimate, which is the training data feature MLE
    :params email: 57-element Z-transformed array
    """
    N_features = len(email)

    SecondTerm_Spam = 0
    SecondTerm_notSpam = 0

    for j in np.arange(N_features):

#         SecondTerm_Spam += np.log(pdf(email[j],
#                                       xtrainZ_mean_spam[j],
#                                       xtrainZ_sigma_spam[j]))

#         SecondTerm_notSpam += np.log(pdf(email[j],
#                                          xtrainZ_mean_notspam[j],
#                                          xtrainZ_sigma_notspam[j]))
        """Log Likelihood"""
        SecondTerm_Spam += -np.log(xtrainZ_sigma_spam[j])-((email[j]-xtrainZ_mean_spam[j]
)/(xtrainZ_sigma_spam[j]))**2/2
        SecondTerm_notSpam += -np.log(xtrainZ_sigma_notspam[j])-((email[j]-xtrainZ_mean_n
otspam[j])/(xtrainZ_sigma_notspam[j]))**2/2

    LogProb_Spam = np.log(classMLE) + SecondTerm_Spam

    LogProb_notSpam = np.log(1-classMLE) + SecondTerm_notSpam

    class_ = (LogProb_Spam > LogProb_notSpam).astype('uint8')
    return class_
```

In [196]:
```
def error_rate_Z(emails,categories):
    N_errors = 0
    count = 0
    for idx,mail in enumerate(emails):
        try:
            N_errors += np.logical_xor(classifyZ(mail), categories[idx])
            count += 1
        except:
            print idx
    return N_errors/len(emails)
```

In [198]:
```
classifyZ(xtrainZ[0]), ytrain[0]
```

Out[198]:
```
(1, 1)
```

In [199]:
```
error_rates_training_Z = error_rate_Z(xtrainZ, ytrain)
```

In [200]:
```
error_rates_training_Z
```

Out[200]:
```
0.18858075040783034
```

In [201]:
```
error_rates_test_Z = error_rate_Z(xtestZ, ytest)
```

In [202]:
```
error_rates_test_Z
```

Out[202]:
```
0.16276041666666666
```

Log-Transform

In [8]:
```
# training data feature MLE
xtrainLog_mean_spam = np.mean(xtrainLog[ytrain==1],axis=0)
xtrainLog_sigma_spam = np.std(xtrainLog[ytrain==1],axis=0)
xtrainLog_mean_notspam = np.mean(xtrainLog[ytrain==0],axis=0)
xtrainLog_sigma_notspam = np.std(xtrainLog[ytrain==0],axis=0)
```

In [17]:
```
 def classifyLog(email):
    """
    BASED ON MLE
    classifies each email according to a gaussian likelihood,
    parameterised by the plug-in estimate, which is the training data feature MLE
    :params email: 57-element log array
    """
    N_features = len(email)

    SecondTerm_Spam = 0
    SecondTerm_notSpam = 0

    for j in np.arange(N_features):

#         SecondTerm_Spam += np.log(norm.pdf(email[j],
#                                    xtrainLog_mean_spam[j],
#                                    xtrainLog_sigma_spam[j]))

#         SecondTerm_notSpam += np.log(norm.pdf(email[j],
#                                       xtrainLog_mean_notspam[j],
#                                       xtrainLog_sigma_notspam[j]))
        """Log Likelihood"""
        SecondTerm_Spam += -np.log(xtrainLog_sigma_spam[j])-((email[j]-xtrainLog_mean_spa
m[j])/(xtrainLog_sigma_spam[j]))**2/2
        SecondTerm_notSpam += -np.log(xtrainLog_sigma_notspam[j])-((email[j]-xtrainLog_me
an_notspam[j])/(xtrainLog_sigma_notspam[j]))**2/2

    LogProb_Spam = np.log(classMLE) + SecondTerm_Spam

    LogProb_notSpam = np.log(1-classMLE) + SecondTerm_notSpam

    class_ = (LogProb_Spam > LogProb_notSpam).astype('uint8')
    return class_
```

In [10]:
```
def error_rate_Log(emails,categories):
    N_errors = 0

    for idx,mail in enumerate(emails):
            N_errors += np.logical_xor(classifyLog(mail), categories[idx])

    return N_errors/len(emails)
```

In [13]:
```
error_rates_training_Log = error_rate_Log(xtrainLog, ytrain)
```

In [14]:
```
error_rates_training_Log
```

Out[14]:
```
0.17259380097879282
```

In [15]:
```
error_rates_test_Log = error_rate_Log(xtestLog, ytest)
```

In [16]:
```
error_rates_test_Log
```

Out[16]:
```
0.15950520833333334
```