EE5907 Programming Assignment Q3

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)
```

Load Data

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

In [3]:
```
def binarize(array):
    """array: list of emails, each 57 features long"""
    return np.array([x>0 for x in array]).astype('uint8')
xtrainBin = binarize(xtrain)
xtestBin = binarize(xtest)
```

In [4]:
```
#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 [3]:
```
#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)
```

In [4]:
```
classMLE = np.average(ytrain)
```

In [5]:
```
def sigmoid(x):
    return 1/(1 + np.exp(-x))
```

In [6]:
```
def reg_params(x_array2D, y_array1D, w, lamb):
    """returns NLL, g, H"""
    N_features = x_array2D.shape[1] # number of features per mail: default is 57
    x_array2D = np.insert(x_array2D,0,1,axis=1) # modify feature list to include a 1 at t
he start

    mu = sigmoid(np.dot(x_array2D, w)) # array1D of mus, one for each email
    NLL = -np.sum([y*np.log(m) + (1-y)*np.log(1-m) for y,m in zip (y_array1D, mu)])
    g = np.dot(x_array2D.T,(mu-y_array1D))
    S = np.diag(mu*(1-mu))
    H = np.dot(x_array2D.T, np.dot(S, x_array2D))

    NLL_reg = NLL + 0.5*lamb*np.inner(w[1:],w[1:])
    g_reg = g + np.insert(lamb*w[1:],0,0) # first element of g is unregularised
    H_reg = H + np.diag(np.insert(lamb*np.ones(N_features),0,0)) # first diagonal element
is unregularised

#     print x_array2D
#     print mu

    return NLL_reg, g_reg, H_reg
```

In [9]:
```
NLL_reg, g_reg, H_reg = reg_params(xtrainZ,ytrain,np.ones(58),1)
```

```
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:7: RuntimeWarning: divide by
zero encountered in log
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:7: RuntimeWarning: invalid va
lue encountered in multiply
```

In [7]:
```
def omega(x_array2D, y_array1D, lamb):
    """
    this function implements Newton's method for logistic regression to find w which desc
ribes the dataset x_array2D, yarray1D
    """
    """initialisation"""
    N_features = x_array2D.shape[1]
    wInit = np.zeros(N_features+1)
    w = wInit
    converge = False
    counter = 0
    """iterator to find optimal w"""
    try:
        while converge==False:
            _, g, H = reg_params(x_array2D, y_array1D, w, lamb)
            counter += 1
            w_new = w - np.dot(np.linalg.inv(np.dot(H.T,H)),np.dot(H.T,g)) # numerically
more stable than inv(H).g
    #         w_new = w - np.dot(np.linalg.inv(H),g)
            error = np.inner(w_new-w,w_new-w)/np.inner(w,w)
            if  error < 0.001:
                """arbitrary error threshold set to 0.1%"""
                converge = True
            else:
                w = w_new
        return w_new
    except:
        print ('lambda = {}\n iteration = {}\n w_new-w = {:.5f}%\n, g = {}\n H = {}\n'.fo
rmat(lamb, counter, error*100, g, H))
```

In [8]:
```
def LogReg_ErrorRate(x_array2D, y_array1D, x_training_array2D, y_training_array1D, lamb):
    """
    uses omega to classify an array of emails x_array2D
    compares results with y_array1D to check for error rate
    """
    w = omega(x_training_array2D,y_training_array1D,lamb)
    category = sigmoid(w[0]+np.dot(x_array2D, w[1:]))>0.5
    category_error = np.logical_xor(category,y_array1D)
    error_rate =  np.sum(category_error)/len(y_array1D)
    return error_rate
```

In [9]:
```
lambdas = np.insert(np.arange(15,105,5),0,np.arange(1,11,1))
print lambdas
```

```
[  1   2   3   4   5   6   7   8   9  10  15  20  25  30  35  40  45  50
  55  60  65  70  75  80  85  90  95 100]
```

Binarised Data

In [27]:
```
TrainingBin_ErrorRates = map(lambda lamb: LogReg_ErrorRate(xtrainBin,ytrain,xtrainBin,ytrain,lamb), lambdas)
```

```
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:17: RuntimeWarning: divide by zero encountered in double_scalars
```

In [34]:
```
# np.savetxt('LogisticRegression_TrainingBinErrorRates',zip(lambdas,TrainingBin_ErrorRates),header='lambdas\tTrainingBin_ErrorRate')
```

In [16]:
```
# lambdas,TrainingBin_ErrorRates = np.loadtxt('LogisticRegression_TrainingBinErrorRates').T
```

In [20]:
```
plt.figure()
plt.title('error rates for binarised training data')
plt.plot(lambdas, np.array(TrainingBin_ErrorRates)*100)
plt.xlabel('lambda')
plt.ylabel('error rate (%)')
plt.show()
```

In [38]:
```
# Error rates for training data for lambda = 1,10,100
map(lambda lamb: LogReg_ErrorRate(xtrainBin,ytrain,xtrainBin,ytrain,lamb), [1,10,100])
```

```
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:17: RuntimeWarning: divide by zero encountered in double_scalars
```

Out[38]:
```
[0.064274061990212072, 0.066884176182707991, 0.093964110929853184]
```

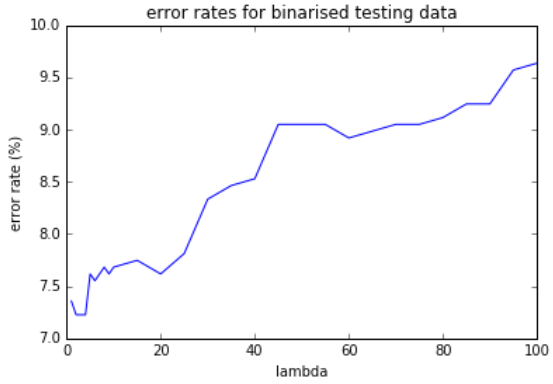
In [38]:
```
TestingBin_ErrorRates = map(lambda lamb: LogReg_ErrorRate(xtestBin,ytest,xtrainBin,ytrain,lamb), lambdas)
```

```
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:17: RuntimeWarning: divide by zero encountered in double_scalars
```

In [39]:
```
# np.savetxt('LogisticRegression_TestingBinErrorRates',zip(lambdas,TestingBin_ErrorRates),header='lambdas\tTestingBin_ErrorRate')
```

In [22]:
```
# lambdas,TestingBin_ErrorRates = np.loadtxt('LogisticRegression_TestingBinErrorRates').T
```

In [23]:
```
plt.figure()
plt.title('error rates for binarised testing data')
plt.plot(lambdas, np.array(TestingBin_ErrorRates)*100)
plt.xlabel('lambda')
plt.ylabel('error rate (%)')
plt.show()
```



In [39]:
```
# Error rates for testing data for lambda = 1,10,100
map(lambda lamb: LogReg_ErrorRate(xtestBin,ytest,xtrainBin,ytrain,lamb), [1,10,100])
```

```
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:17: RuntimeWarning: divide by
zero encountered in double_scalars
```

Out[39]:
```
[0.073567708333333329, 0.076822916666666671, 0.096354166666666671]
```

Z-normed Data

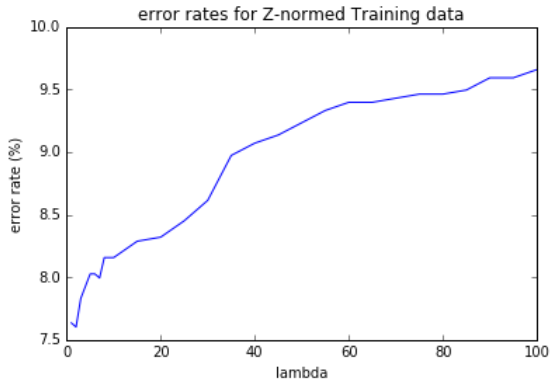
In [46]:
```
TrainingZ_ErrorRates = map(lambda lamb: LogReg_ErrorRate(xtrainZ,ytrain,xtrainZ,ytrain,la
mb), lambdas)
```

```
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:17: RuntimeWarning: divide by
zero encountered in double_scalars
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:7: RuntimeWarning: divide by
zero encountered in log
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:7: RuntimeWarning: invalid va
lue encountered in multiply
```

In [47]:
```
# np.savetxt('LogisticRegression_TrainingZErrorRates',zip(lambdas,TrainingZ_ErrorRates),h
eader='lambdas\tTrainingZ_ErrorRate')
```

In [24]:
```
# lambdas,TrainingZ_ErrorRates = np.loadtxt('LogisticRegression_TrainingZErrorRates').T
```

In [25]:
```
plt.figure()
plt.title('error rates for Z-normed Training data')
plt.plot(lambdas, np.array(TrainingZ_ErrorRates)*100)
plt.xlabel('lambda')
plt.ylabel('error rate (%)')
plt.show()
```



In [40]:
```
# Error rates for training data for lambda = 1,10,100
map(lambda lamb: LogReg_ErrorRate(xtrainZ,ytrain,xtrainZ,ytrain,lamb), [1,10,100])
```

```
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:17: RuntimeWarning: divide by
zero encountered in double_scalars
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:7: RuntimeWarning: divide by
zero encountered in log
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:7: RuntimeWarning: invalid va
lue encountered in multiply
```

Out[40]:
```
[0.076345840130505715, 0.081566068515497553, 0.096574225122349103]
```

In [43]:
```
TestingZ_ErrorRates = map(lambda lamb: LogReg_ErrorRate(xtestZ,ytest,xtrainZ,ytrain,lamb)
, lambdas)
```

```
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:17: RuntimeWarning: divide by
zero encountered in double_scalars
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:7: RuntimeWarning: divide by
zero encountered in log
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:7: RuntimeWarning: invalid va
lue encountered in multiply
```

In [44]:
```
# np.savetxt('LogisticRegression_TestingZErrorRates',zip(lambdas,TestingZ_ErrorRates),hea
der='lambdas\tTestingZ_ErrorRate')
```

In [26]:
```
# lambdas,TestingZ_ErrorRates = np.loadtxt('LogisticRegression_TestingZErrorRates').T
```

In [27]:
```
plt.figure()
plt.title('error rates for Z-normed testing data')
plt.plot(lambdas, np.array(TestingZ_ErrorRates)*100)
plt.xlabel('lambda')
plt.ylabel('error rate (%)')
plt.show()
```

In [41]:
```
# Error rates for testing data for lambda = 1,10,100
map(lambda lamb: LogReg_ErrorRate(xtestZ,ytest,xtrainZ,ytrain,lamb), [1,10,100])
```

```
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:17: RuntimeWarning: divide by
zero encountered in double_scalars
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:7: RuntimeWarning: divide by
zero encountered in log
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:7: RuntimeWarning: invalid va
lue encountered in multiply
```

Out[41]:
```
[0.068359375, 0.071614583333333329, 0.085286458333333329]
```

Log Data

In [31]:
```
# TrainingLog_ErrorRates = map(lambda lamb: LogReg_ErrorRate(xtrainLog,ytrain,xtrainLog,y
train,lamb), lambdas)
TrainingLog_ErrorRates = []
valid_lambdas = []
invalid_lambdas = []
for lamb in lambdas:
    try:
        TrainingLog_ErrorRates.append(LogReg_ErrorRate(xtrainLog,ytrain,xtrainLog,ytrain,
lamb))
        valid_lambdas.append(lamb)
    except:
        invalid_lambdas.append(lamb)
        pass
```

```
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:17: RuntimeWarning: divide by
zero encountered in double_scalars
```

In [32]:
```
# np.savetxt('LogisticRegression_TrainingLogErrorRates',zip(valid_lambdas,TrainingLog_Err
orRates),header='lambdas\tTrainingLog_ErrorRate')
```

In [33]:
```
# valid_lambdas,TrainingLog_ErrorRates = np.loadtxt('LogisticRegression_TrainingLogErrorR
ates').T
```

In [34]:
```
plt.figure()
plt.title('error rates for Log-normed Training data')
plt.plot(valid_lambdas, np.array(TrainingLog_ErrorRates)*100)
plt.xlabel('lambda')
plt.ylabel('error rate (%)')
plt.show()
```

In [39]:
```
# Error rates for training data for lambda = 1,10,100
map(lambda lamb: LogReg_ErrorRate(xtrainLog,ytrain,xtrainLog,ytrain,lamb), [1,10,100])
```

```
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:17: RuntimeWarning: divide by
zero encountered in double_scalars
```

Out[39]:
```
[0.055791190864600326, 0.056443719412724309, 0.064926590538336049]
```

In [13]:
```
# TestingLog_ErrorRates = map(lambda lamb: LogReg_ErrorRate(xtrainLog,ytrain,xtrainLog,yt
rain,lamb), lambdas)
TestingLog_ErrorRates = []
valid_lambdas = []
invalid_lambdas = []
for lamb in lambdas:
    try:
        TestingLog_ErrorRates.append(LogReg_ErrorRate(xtestLog,ytest,xtrainLog,ytrain,lam
b))
        valid_lambdas.append(lamb)
    except:
        invalid_lambdas.append(lamb)
        pass
```

```
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:17: RuntimeWarning: divide by
zero encountered in double_scalars
```

In [35]:
```
# np.savetxt('LogisticRegression_TestingLogErrorRates',zip(valid_lambdas,TestingLog_Error
Rates),header='lambdas\tTestingLog_ErrorRate')
```

In [36]:
```
# valid_lambdas,TestingLog_ErrorRates = np.loadtxt('LogisticRegression_TestingLogErrorRat
es').T
```

In [37]:
```
plt.figure()
plt.title('error rates for Log-normed Testing data')
plt.plot(valid_lambdas, np.array(TestingLog_ErrorRates)*100)
plt.xlabel('lambda')
plt.ylabel('error rate (%)')
plt.show()
```

In [38]:
```
# Error rates for testing data for lambda = 1,10,100
map(lambda lamb: LogReg_ErrorRate(xtestLog,ytest,xtrainLog,ytrain,lamb), [1,10,100])
```

```
/usr/local/lib/python2.7/dist-packages/ipykernel/__main__.py:17: RuntimeWarning: divide by
zero encountered in double_scalars
```

Out[38]:
```
[0.05078125, 0.053385416666666664, 0.065104166666666671]
```

In [40]:
```
import seaborn
```

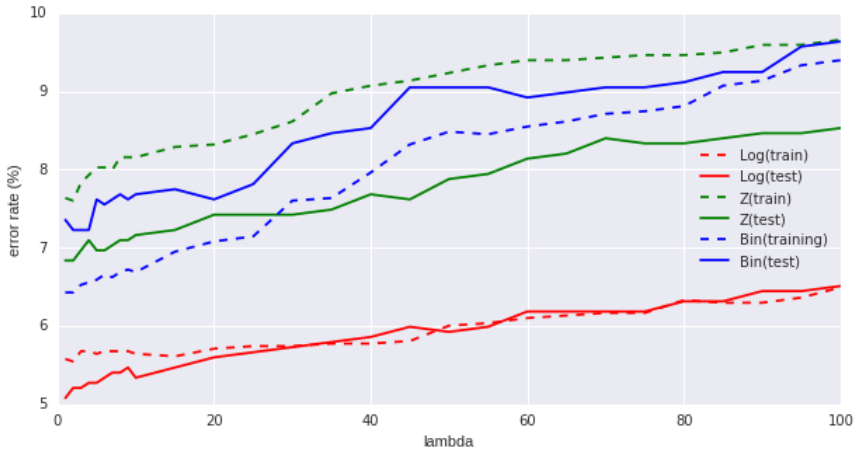
```
/usr/local/lib/python2.7/dist-packages/IPython/html.py:14: ShimWarning: The `IPython.html`
package has been deprecated since IPython 4.0. You should import from `notebook` instead.
`IPython.html.widgets` has moved to `ipywidgets`.
  "`IPython.html.widgets` has moved to `ipywidgets`.", ShimWarning)
```

In [52]:
```
plt.figure(figsize=(10,5))
plt.plot(valid_lambdas, np.array(TrainingLog_ErrorRates)*100, ls = '--', color = 'r', lab
el='Log(train)')
plt.plot(valid_lambdas, np.array(TestingLog_ErrorRates)*100, color = 'r', label='Log(test
)')

plt.plot(lambdas, np.array(TrainingZ_ErrorRates)*100, ls = '--', color = 'g', label='Z(tr
ain)')
plt.plot(lambdas, np.array(TestingZ_ErrorRates)*100, color = 'g', label='Z(test)')

plt.plot(lambdas, np.array(TrainingBin_ErrorRates)*100, ls = '--', color = 'b', label='Bi
n(training)')
plt.plot(lambdas, np.array(TestingBin_ErrorRates)*100, color = 'b', label='Bin(test)')

plt.xlabel('lambda')
plt.ylabel('error rate (%)')
plt.ylim(5,10)
plt.legend(loc='best')
plt.show()
```



In [ ]:
```
```