hw5.5

November 7, 2018

In [1]:
```
import numpy as np
import math
import matplotlib.pyplot as plt
```

In [2]:
```
# read train files
with open("./new_train3.txt") as f:
    train3data = f.readlines()
size_train3 = len(train3data)
train3 = np.zeros((size_train3,64))
for index,image in enumerate(train3data):
    seperate = image.strip("\n").split(" ")[:-1]
    image_list = np.array(list(map(int,seperate)))
    train3[index,:] = image_list
with open("./new_train5.txt") as f:
    train5data = f.readlines()
size_train5 = len(train5data)
train_y = np.zeros((size_train3,1))      #label 3 is zero
train5 = np.zeros((size_train5,64))
for index,image in enumerate(train5data):
    seperate = image.strip("\n").split(" ")[:-1]
    image_list = np.array(list(map(int,seperate)))
    train5[index,:] = image_list
train_x = np.vstack((train3,train5))
train_y = np.vstack((train_y, np.ones((size_train5,1))))    #label 5 is one

#read test files
with open("./new_test3.txt","r") as f:
    test3data = f.readlines()
with open("./new_test5.txt","r") as f:
    test5data = f.readlines()
test_x = []
test_y = []
for index,data in enumerate(test3data):
    test_x.append(np.array(data.strip("\n").split(" ")[:-1]).astype(int))
    test_y.append(0)
for index,data in enumerate(test5data):
    test_x.append(np.array(data.strip("\n").split(" ")[:-1]).astype(int))
    test_y.append(1)
```

```
        test_x = np.array(test_x)
        test_y = np.array(test_y)
```

In [20]:
```
def sigmoid(x):
    return 1.0 / (1.0 + np.exp(-x))

def predict(train,w):
    z = sigmoid(np.dot(train,w))
    predict_class = (z >= 0.5).astype(int)
    return predict_class

def compute_error(pred,ground):
    return float(np.sum(pred.flatten()!=ground))/pred.shape[0]*100

def copute_log_likelihood(train,label,ws):
    log_likelihood = 0
    for index in range(train.shape[0]):
        example = train[index,:].reshape(1,train[index,:].shape[0])
        log_likelihood += label[index] * np.log(sigmoid(np.sum(np.dot(example,ws))))
        + (1 - label[index]) * np.log(sigmoid(-np.sum(np.dot(example,ws))))
    return log_likelihood

def compute_acc(pred,label):
    return float(np.sum(pred.flatten()==label))/pred.shape[0] * 100

def trainData(train,label,iterations,eta,method="gradient_ascent"):
    ws = np.zeros((64,1))
    error_list = []
    log_like_list = []
    if method == "gradient_ascent":
        for iter in range(iterations):
            z = sigmoid(np.dot(train,ws))
            diff = label - z
            gradient = np.dot(train.T, diff)
            ws = ws + eta/train.shape[0] * gradient
            error_list.append(compute_error(predict(train,ws),label))
            log_like_list.append(copute_log_likelihood(train,label,ws))
    return ws,error_list,log_like_list


def testData(test,label,model):
    pred_label = predict(test,model)
    acc = compute_acc(pred_label,label)
    err = compute_error(pred_label,label)
    print("The accuracy of the model is {}%".format(acc))
    print("The error rate of the model is {}%".format(err))
```

In [8]:
```
#training process
```

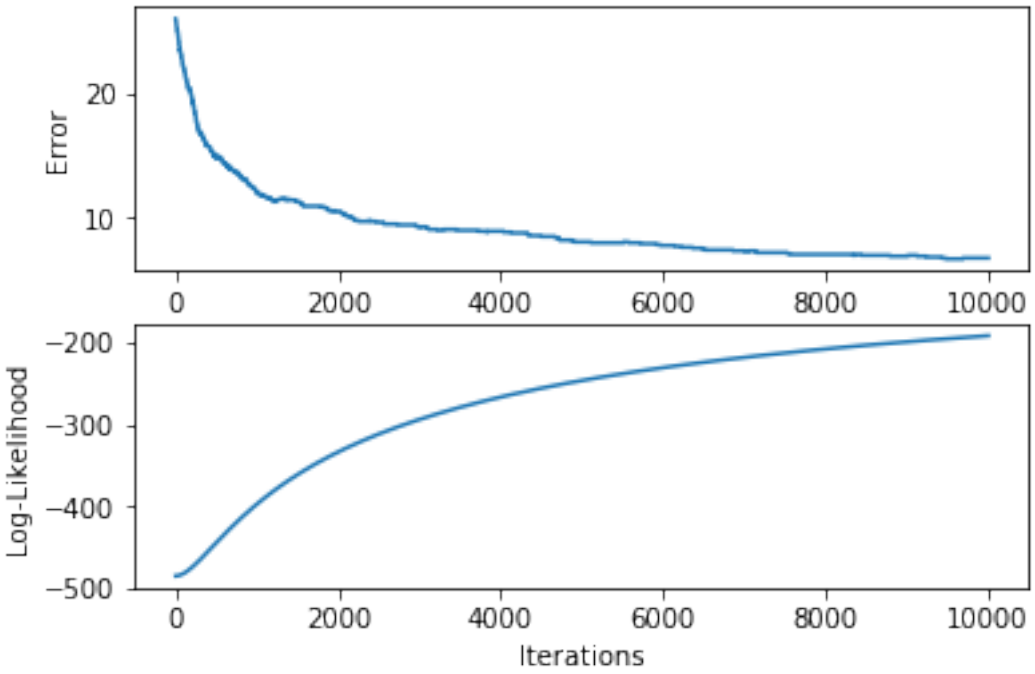
```
times = 10000
(model,errs,log_likes) = trainData(train_x,train_y,times,0.001)
```

In [21]:
```
#test process
testData(test_x,test_y,model)
```

```
The accuracy of the model is 92.875%
The error rate of the model is 7.124999999999999%
```

In [12]:
```
# training process
plt.figure()
plt.subplot(2,1,1)
plt.xlabel("Iterations")
plt.ylabel("Error")
plt.plot(range(times),errs)
plt.subplot(2,1,2)
plt.xlabel("Iterations")
plt.ylabel("Log-Likelihood")
plt.plot(range(times),log_likes)
```

Out[12]: [<matplotlib.lines.Line2D at 0x11a88f390>]



In [23]:
```
print(model.reshape(8,8))
```

```
[[-0.14978948 -0.23379796 -0.2620632  -0.26735281 -0.15650103  0.05819675
   0.26559247  0.44076956]
 [-0.05206076  0.01651187  0.06728831  0.11427564  0.18547774  0.15784308
  -0.16366097 -0.21322931]
 [ 0.23635434  0.41390298  0.52715968  0.34259459 -0.02491731 -0.50741746
  -0.80810187 -0.53627667]
 [ 0.2355871   0.34742695  0.32223668  0.00385604 -0.2056208  -0.24906787
  -0.22016208 -0.16484149]
 [ 0.10586976  0.08902139  0.03682623 -0.07382843 -0.09211044 -0.1196024
  -0.11450961 -0.24424491]
 [ 0.21715615  0.00884969  0.03549628  0.12870283  0.08384332  0.0158127
  -0.07514835 -0.30496598]
 [ 0.15881496  0.11698806  0.1166019   0.07327297 -0.00699652 -0.06135614
  -0.00213004 -0.23139402]
 [-0.08032515  0.0984083   0.11727974  0.07875866  0.01869178  0.03971351
  -0.08999777 -0.06336839]]
```

In [ ]: