IT701: Topics in Deep Learning

Lab Assignment 3

Prepared by:

Rajat Kumar (201811024)

Guided by:

M V Joshi

Ankur Patil

1. Use data in files "ex2data1-logistic.xls" and "ex2data2-logistic.xls" to perform logistic regression for each these data sets. Use 90% data points each set for training the regressor and remaining 10% for testing the accuracy of classification.

Code:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import numpy as np

#loading datasets as dataframes
df1=pd.read_csv('ex2data1-logistic.csv')
df2=pd.read_csv('ex2data2-logistic.csv')

#seperating columns from dataset
x1=df1.iloc[:,0:2]
y1=df1.iloc[:,-1]

z1=df2.iloc[:,0]
z2=df2.iloc[:,1]

#adding x1^2,x1*x2,x2^2 as columns as we want non-linaer decision boundary
frames=[z1,z2,z1*z1,z1*z1,z2*z2]
x2=pd.concat(frames,axis=1)
x2.coulumns=range(x2.shape[1])
y2=df2['y']

#split train data=90% and test=10%
x1_train,x1_test,y1_train,y1_test=train_test_split(x1,y1,test_size=0.1)
x2_train,x2_test,y2_train,y2_test=train_test_split(x2,y2,test_size=0.1)

#normalising train data
```

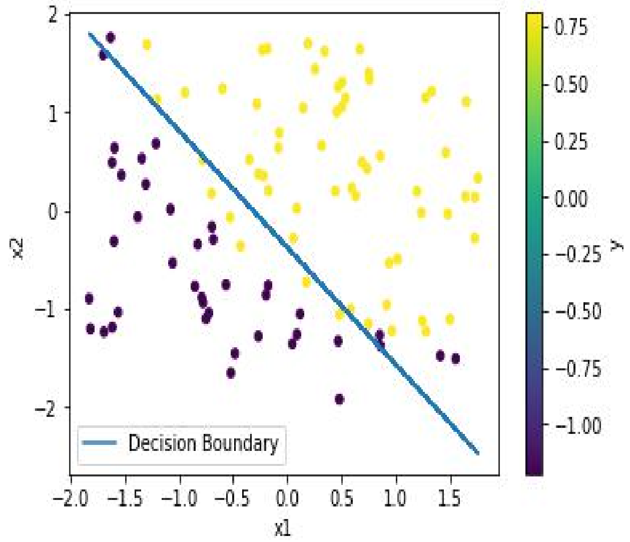
```
x1_train_norm=(x1_train-x1_train.mean())/x1_test.std()
x2_train_norm=(x2_train-x2_train.mean())/x2_test.std()

#adding 1st column as all 1's for x0
x1_train_norm=np.c_[np.ones(x1_train_norm.shape[0]),x1_train_norm]
x2_train_norm=np.c_[np.ones(x2_train_norm.shape[0]),x2_train_norm]

#function defined to find thetas
def find_thetas(alpha,itr,x_norm,y,theta):
    for i in range(itr):
        e=x_norm.dot(theta)
        sig=1./(1+np.exp(-e))
        theta-=(alpha/len(x_norm))*x_norm.T.dot(sig-y)
    return theta



alpha=0.01
iteration=1000
#thetas initalized to zero
theta1=np.zeros(3)
theta2=np.zeros(6)

#using find_theta function to compute values of parameters
theta1=find_thetas(alpha,iteration,x1_train_norm,y1_train,theta1)
theta2=find_thetas(alpha,iteration,x2_train_norm,y2_train,theta2)

#original data normalized
df1_norm=(df1-df1.mean())/df1.std()
df2_norm=(df2-df2.mean())/df2.std()
#finding points x2 to plot a decsion boundary
line_1=(-theta1[0]-(theta1[1]*df1_norm.iloc[:,0:1]))/theta1[2]
```

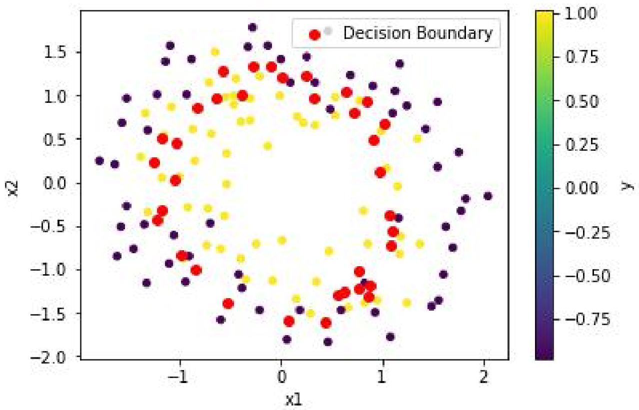
```
#ploting datapoints of first file and plotiing decision boundary
ax=df1_norm.plot(kind='scatter',x='x1',y='x2',c='y',colormap='viridis',sharex=False)
ax.plot(df1_norm['x1'],line_1,linestyle='solid',label='Decision Boundary')
ax.legend()

#Normalizing test data to compute accuracy
x1_test_norm=(x1_test-x1_test.mean())/x1_test.std()
x1_test_norm=np.c_[np.ones(x1_test_norm.shape[0]),x1_test_norm]

x2_test_norm=(x2_test-x2_test.mean())/x2_test.std()
x2_test_norm=np.c_[np.ones(x2_test_norm.shape[0]),x2_test_norm]



#function to compute accuracy of datasets
def accuracy(x_test_norm,theta,y_test):
    z=x_test_norm.dot(theta)
    sigmoid=1./(1+np.exp(-z))

    diff=[]
    for i in range(len(sigmoid)):
        if sigmoid[i]>0.5:
            diff.append(1)
        else:
            diff.append(0)
    d=diff-y_test
    qq=d.nonzero()
    acc=1.-(len(qq[0])/len(y2_test))
    return acc
acc1=accuracy(x1_test_norm,theta1,y1_test)
acc2=accuracy(x2_test_norm,theta2,y2_test)
print('Accuracy of dataset 1:',acc1)
print('Accuracy of dataset 2:',acc2)
```

```
#plot of second dataset
bx=df2_norm.plot(kind='scatter',x='x1',y='x2',c='y',colormap='viridis',sharex=False)

#computing values of htheta for values of theta
zz=x2_train_norm.dot(theta2)
sig=1./(1+np.exp(-zz))

X1=[]
X2=[]
#here we find points whose values of htheta-0.5 is less than 0.1 will be points of decision boundary
for i in range(len(sig)):
    if np.abs(sig[i]-0.5)<0.1:
        X1.append(x2_train_norm[i][1])
        X2.append(x2_train_norm[i][2])

#decision boundary for 2nd dataset
bx.scatter(X1,X2,color='r',label="Decision Boundary")
bx.legend()
```

Output:

```
Accuracy of dataset 1: 0.9166666666666666
Accuracy of dataset 2: 0.8333333333333334
```





Conclusion:

From above two datasets we can observe for first dataset we need linear boundary to separate both datapoints, where as for second dataset we need non-linear boundary which is represented by ellipse. For second we need to add terms of square to compute its decision boundary.

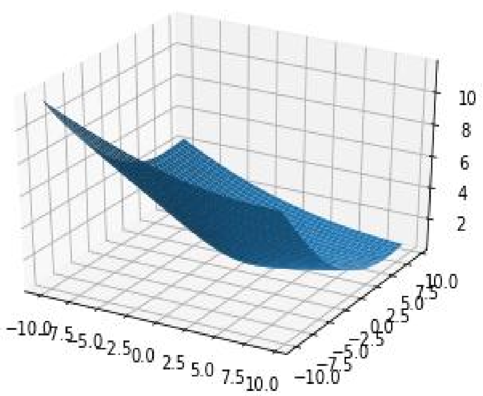
2. For testing the convexity / non-convexity of the cost function, consider one example from the first dataset. Now plot the cost function by varying the values of parameters (\theta) for logistic regression cost. Note that the hypothesis to be used is the sigmoid function in both the cases.

Code:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import numpy as np
import math
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
#loading datasets as dataframes
df1=pd.read_csv('ex2data1-logistic.csv')

x1=df1.iloc[:,0:2]
y1=df1.iloc[:,-1]

#split train data=90% and test=10%
x1_train,x1_test,y1_train,y1_test=train_test_split(x1,y1,test_size=0.1)

#normalising train data
x1_train_norm=(x1_train-x1_train.mean())/x1_test.std()

#adding 1st column as all 1's for x0
x1_train_norm=np.c_[np.ones(x1_train_norm.shape[0]),x1_train_norm]

list1=y1_train.values
theta0 = 1;
theta1 = np.arange(-10,10,0.5);
theta2 = np.arange(-10,10,0.5);
```

```
X,Y = np.meshgrid(theta1,theta2);
error = np.zeros([len(theta1),len(theta2)]);
for i in range(len(theta1)):
    for j in range(len(theta2)):
        temp = 0;
        for k in range(len(list1)):
            hthetax = 1 + np.exp(-
np.matmul(x1_train_norm[k,:],np.array([[theta0],[theta1[i]],[theta2[j]]])));
            hthetax = 1/hthetax;
            temp = temp + (math.log(hthetax)*list1[k] + math.log(1-hthetax)*(1 - list1[k]));
        error[i,j] = -temp/len(x1_train);
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d');
ax.plot_surface(X, Y, error)
```

Output:



Conclusion: From output we can observe that it is convex as it has point of minima.