In [ ]:

```
# Logistic Regression + SGD
# implemented using numpy in Python

import numpy as np
import scipy
from scipy.special import expit
import matplotlib.pyplot as plt

def loadData(filename):
    X=[]
    count = 0

    text_file = open(filename, "r")
    lines = text_file.readlines()

    for line in lines:
        X.append([])
        words = line.split(",")
        for word in words:
            X[count].append(float(word))
        count += 1
    return np.asarray(X)

def dataNorm(X):
    X_norm = np.copy(X)
    X_norm = np.insert(X_norm, 0, 1, axis=1)
    for i in range(1, X_norm.shape[1]):
        X_norm[:,i] = (X_norm[:,i]-np.amin(X_norm[:,i]))/(np.amax(X_norm[:,i])-np.amin
    return np.asarray(X_norm)

def errCompute(X_norm, theta):

    # variable initialization
    x = X_norm[:,:-1]
    y = X_norm[:,-1]
    M = X_norm.shape[0]
    yHat = expit(np.dot(x,theta))

    result = (np.dot(y,np.log(yHat)) + np.dot((1-y),np.log(1-yHat))) / (-M)
    return result

def stochasticGD(X_norm, theta, alpha, num_iters):

    # variable initialization
    x = X_norm[:,:-1]
    print "x",x.shape
    y = np.reshape(X_norm[:,-1],(x.shape[0],1))
    print "y",y.shape
    # creating an array to record the error after each iteration
    errRecords = np.zeros((num_iters,1))

    # stochasticGD algorithm
    for idx in range(num_iters):
        #print "theta",theta.shape
        i = idx % x.shape[0]
        yHat = expit(np.dot(x,theta))
        for j in range(x.shape[1]):
            #print "yhat", yHat.shape

            theta[j] += alpha * (y[i] - yHat[i]) * x[i][j]
        errRecords[idx] = errCompute(X_norm, theta)

    # errCompute() should return 0.3151
    print "errCompute() = ", errCompute(X_norm, theta)

    # accuracy verification
    #yPredict = loadData('LogisticRegresion_data/predict.data') ### to change path he
    yHat = np.around(expit(np.dot(x,theta)))
    accuracy = (y == yHat).mean() * 100
    print "accuracy = ", accuracy, "%"


    # plot of error function against iteration number
    x_axis = [x for x in range(0,num_iters)]
    plt.plot(x_axis,list(errRecords))
    plt.ylabel('Error')
    plt.xlabel('Number of Iterations')
    plt.show()

    return theta

def LogRMain(filename):

    # data load
    X = loadData(filename)
    # normalization
    X_norm = dataNorm(X)
    # theta to be learnt
    theta = np.zeros((X_norm.shape[1]-1,1))

    # LogR, iteration here is the times of passing datasets
    theta = stochasticGD(X_norm, theta, 0.01, 1372*20)

    return theta
```