Neural Networks
Assignment 2

Nitin Surya
nsurya3
660604663

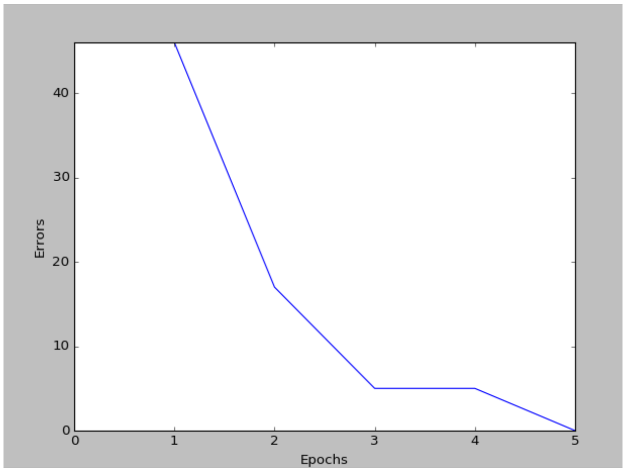
Program:

```
import os, struct # to unpack binary data
from array import array as pyarray # for reading data into array
import numpy as np # for numeric computations
import matplotlib.pyplot as plt # plot graphs

# Function read labels from file and
#   return the desired labels of all the input images
#   converts each label e.g. '5' -> [0 0 0 0 1 0 0...]
# Partial code taken from : http://g.sweyla.com/blog/2012/mnist-numpy/
def getLabelsArray(fname, size_img, output_dim):
    flbl = open(fname, 'rb')
    magic_nr, size_label = struct.unpack(">II", flbl.read(8))
    lbl = pyarray("b", flbl.read()) # signed integers
    flbl.close()

    desired_label = np.zeros((size_img, output_dim), dtype=np.int)
    for i in range(size_label):
        desired_label[i][lbl[i]] = 1

    return desired_label

# Function to read the image details from the file and
#   return image in [size, 1D-image-data] shape
# Partial code taken from : http://g.sweyla.com/blog/2012/mnist-numpy/
def getImagesArray(fname):
    fimg = open(fname, 'rb')
    magic_nr, size_img, rows, cols = struct.unpack(">IIII", fimg.read(16))
    img = pyarray("B", fimg.read()) # reading unsigned integers
    img = np.asarray(img).reshape(size_img, rows*cols)
    fimg.close()
    return img, size_img

# Returns after applying step function on V
def stepFunction(V):
    for i in range (0,10):
        value = V[i][0]
        V[i][0] = 1 if value >= 0 else 0
    return V

# Returns a matrix of size 10X1 with 1 at given index
def vectorWithOneOne(maxIndex):
    V = np.random.rand(10,1)
    V.fill(0)
    V[maxIndex][0]=1
    return V

# Returns index of maximum value in the output matrix
def findMaxIndex(V):
    max_val, max_i = V[0][0], 0
    for i in range (0,10):
        if (V[i][0] > max_val):
            max_val, max_i = V[i][0], i
    return max_i

# Function to update weights and returns the new w
def updateWeights(train_img_count, img, desired_label, w, n):
    for i in range(0, train_img_count):
        input_val = img[i].reshape(img.shape[1], 1)
        output = desired_label[i].reshape(desired_label.shape[1], 1)
        w = w + n * (output - stepFunction(w.dot(input_val))) * np.transpose(input_val)
    return w

# Returns the dot product of the img and label at index
def getDotProduct(img, desired_label, index, w):
    input_val = img[index].reshape(img.shape[1], 1)
    output = desired_label[index].reshape(desired_label.shape[1],1)
    vals = w.dot(input_val)
    return input_val, output, vals

# Function to train a multicategor perceptron
# which  creates a weights set randomly and
# saves it to the global variable 'w'
def MulticategoryPerceptronTrainer(train_img_count, n, e, img, debug=False):
    w = np.random.rand(desired_label.shape[1], img.shape[1])
    ep, errors, error_epochs, continue_loop = 0, [], [], True

    # Actual training section
    while(continue_loop and ep < 20):
        if debug:
            print("epoch :: ", ep)
        errors.append(0)

        # Counting the errors
        for index in range(0, train_img_count):
            if debug and index % 5000 == 0:
                print(index, " elements done")
            output, V = getDotProduct(img, desired_label, index, w)[1:]

            maxI = findMaxIndex(V)
            V = vectorWithOneOne(maxI)
            if (not(output == V).all()):
                errors[ep] = errors[ep]+1

        ep = ep + 1
        error_epochs.append(ep)
        w = updateWeights(train_img_count, img, desired_label, w, n)

        # checking if the minimum error ratio is reached or not
        prog_error = errors[ep - 1]/train_img_count
        if (prog_error < e):
            continue_loop = False
            print(ep, " Error rate :: ", prog_error)
        else:
            print(ep, " Error rate :: ", prog_error, " should be below e: ", e)

    if(continue_loop):
        print("Program didn't terminate, so stopped after 20 epochs")
    print("Epochs to convergence :: ", ep)
    plot_graph(error_epochs, errors)

    return w


def plot_graph(error_epochs, errors):
    # Plotting the graph
    plt.plot(error_epochs, errors, 'b')
    plt.xlabel("Epochs")
    plt.ylabel("Errors")
    # plt.axis([0, ep, 0, max(errors)])
    plt.show()

# Function to get the missclassifications on the test set
def tests(test_img, test_desired_label, w):
    test_img_size = test_img.shape[0]
    errors = 0
    print("Testing : ")
    for index in range(0, test_img_size):
        if index%10000 == 0:
            print(".")
        output, V = getDotProduct(test_img, test_desired_label, index, w)[1:]

        maxIndex = findMaxIndex(V)
        V = vectorWithOneOne(maxIndex)
        if (not(output == V).all()):
            errors = errors + 1
    print ("Missclassifications: ", errors)

# Train Data
img, size_img = getImagesArray("train-images.idx3-ubyte")
desired_label = getLabelsArray("train-labels.idx1-ubyte", size_img, 10)

# Test Data
testImg, test_size_img  = getImagesArray("t10k-images.idx3-ubyte")
test_desired_label = getLabelsArray("t10k-labels.idx1-ubyte", test_size_img, 10)

# actual training and testing
train_img_count, n, er = 50, 1, 0.01
w = MulticategoryPerceptronTrainer(train_img_count, n, er, img)
tests(testImg, test_desired_label, w)

train_img_count, n, er = 1000, 1, 0.01
w = MulticategoryPerceptronTrainer(train_img_count, n, er, img)
tests(testImg, test_desired_label, w)

train_img_count, n, er = 60000, 1, 0
w = MulticategoryPerceptronTrainer(train_img_count, n, er, img)
tests(testImg, test_desired_label, w)

# Final 3 time testing
for i in range(3):
    train_img_count, n, er = 60000, 0.5, 0.16 # more than min observed
    print("Training started.. Learning rate: ", n)
    w = MulticategoryPerceptronTrainer(train_img_count, n, er, img)
    tests(testImg, test_desired_label, w)
```

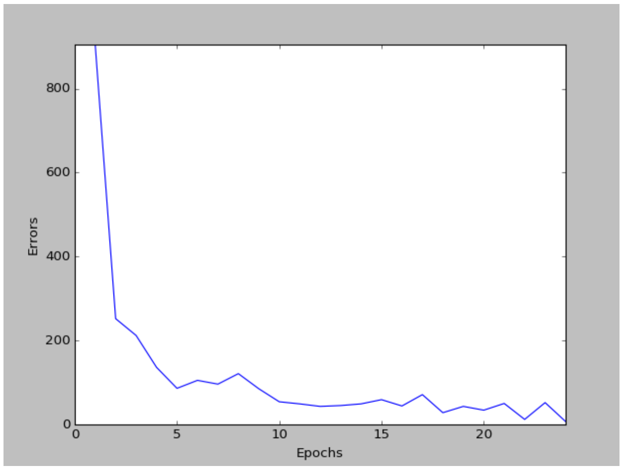
Console Output:

```
Epoch  1  Error rate ::  0.92  Below e:  0.01
Epoch  2  Error rate ::  0.34  Below e:  0.01
Epoch  3  Error rate ::  0.1  Below e:  0.01
Epoch  4  Error rate ::  0.1  Below e:  0.01
Epoch  5  Error rate ::  0.0
Epochs to convergence ::  5
Testing :
.
Missclassifications:  5280
Epoch  1  Error rate ::  0.905  Below e:  0.01
Epoch  2  Error rate ::  0.252  Below e:  0.01
Epoch  3  Error rate ::  0.212  Below e:  0.01
Epoch  4  Error rate ::  0.136  Below e:  0.01
Epoch  5  Error rate ::  0.086  Below e:  0.01
Epoch  6  Error rate ::  0.105  Below e:  0.01
Epoch  7  Error rate ::  0.096  Below e:  0.01
Epoch  8  Error rate ::  0.121  Below e:  0.01
Epoch  9  Error rate ::  0.085  Below e:  0.01
Epoch  10  Error rate ::  0.054  Below e:  0.01
Epoch  11  Error rate ::  0.049  Below e:  0.01
Epoch  12  Error rate ::  0.043  Below e:  0.01
Epoch  13  Error rate ::  0.045  Below e:  0.01
Epoch  14  Error rate ::  0.049  Below e:  0.01
Epoch  15  Error rate ::  0.059  Below e:  0.01
Epoch  16  Error rate ::  0.044  Below e:  0.01
Epoch  17  Error rate ::  0.071  Below e:  0.01
Epoch  18  Error rate ::  0.028  Below e:  0.01
Epoch  19  Error rate ::  0.043  Below e:  0.01
Epoch  20  Error rate ::  0.034  Below e:  0.01
Epoch  21  Error rate ::  0.05  Below e:  0.01
Epoch  22  Error rate ::  0.012  Below e:  0.01
Epoch  23  Error rate ::  0.052  Below e:  0.01
Epoch  24  Error rate ::  0.007
Epochs to convergence ::  24
Testing :
.
Missclassifications:  2166
Epoch  1  Error rate ::  0.9106833333333333  Below e:  0
Epoch  2  Error rate ::  0.22763333333333333  Below e:  0
Epoch  3  Error rate ::  0.20841666666666667  Below e:  0
Epoch  4  Error rate ::  0.21191666666666667  Below e:  0
Epoch  5  Error rate ::  0.2342  Below e:  0
Epoch  6  Error rate ::  0.20801666666666666  Below e:  0
Epoch  7  Error rate ::  0.21411666666666668  Below e:  0
Epoch  8  Error rate ::  0.2148  Below e:  0
Epoch  9  Error rate ::  0.21166666666666667  Below e:  0
Epoch  10  Error rate ::  0.22218333333333334  Below e:  0
Epoch  11  Error rate ::  0.21876666666666666  Below e:  0
Epoch  12  Error rate ::  0.21101666666666666  Below e:  0
Epoch  13  Error rate ::  0.22891666666666666  Below e:  0
Epoch  14  Error rate ::  0.22156666666666666  Below e:  0
Epoch  15  Error rate ::  0.21763333333333335  Below e:  0
Epoch  16  Error rate ::  0.20935  Below e:  0
Epoch  17  Error rate ::  0.19905  Below e:  0
Epoch  18  Error rate ::  0.18128333333333332  Below e:  0
Epoch  19  Error rate ::  0.20531666666666668  Below e:  0
Epoch  20  Error rate ::  0.2112  Below e:  0
Epoch  21  Error rate ::  0.20651666666666665  Below e:  0
Epoch  22  Error rate ::  0.21006666666666668  Below e:  0
Epoch  23  Error rate ::  0.2278  Below e:  0
Epoch  24  Error rate ::  0.2083  Below e:  0
Epoch  25  Error rate ::  0.21366666666666667  Below e:  0
Epoch  26  Error rate ::  0.20608333333333334  Below e:  0
Epoch  27  Error rate ::  0.19205  Below e:  0
Epoch  28  Error rate ::  0.20195  Below e:  0
Epoch  29  Error rate ::  0.21963333333333335  Below e:  0
Epoch  30  Error rate ::  0.21075  Below e:  0
Program didn't terminate, so stopped after 30 epochs
Epochs to convergence ::  30
Testing :
.
Missclassifications:  2433

Training started.. Learning rate:  0.5
1  Error rate ::  0.8947833333333334  should be below e:  0.16
2  Error rate ::  0.17465  should be below e:  0.16
3  Error rate ::  0.16228333333333333  should be below e:  0.16
4  Error rate ::  0.16671666666666668  should be below e:  0.16
5  Error rate ::  0.14976666666666666
Epochs to convergence ::  5
Testing :
.
Missclassifications:  1612
Training started.. Learning rate:  0.5
1  Error rate ::  0.8586  should be below e:  0.16
2  Error rate ::  0.17701666666666666  should be below e:  0.16
3  Error rate ::  0.15586666666666665
Epochs to convergence ::  3
Testing :
.
Missclassifications:  1659
Training started.. Learning rate:  0.5
1  Error rate ::  0.8612333333333333  should be below e:  0.16
2  Error rate ::  0.17886666666666667  should be below e:  0.16
3  Error rate ::  0.15343333333333334
Epochs to convergence ::  3
Testing :
.
Missclassifications:  1755
```

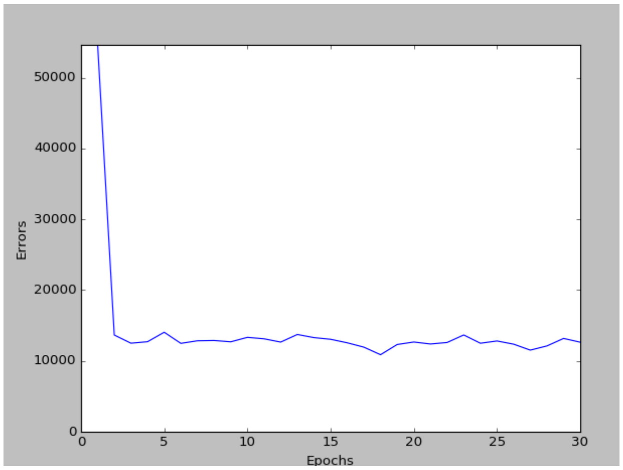
Plots:



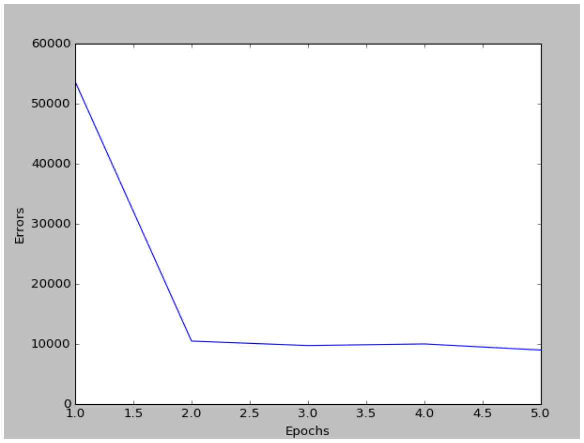
Train images size: 50
Learning rate: 1
Error rate: 0.01

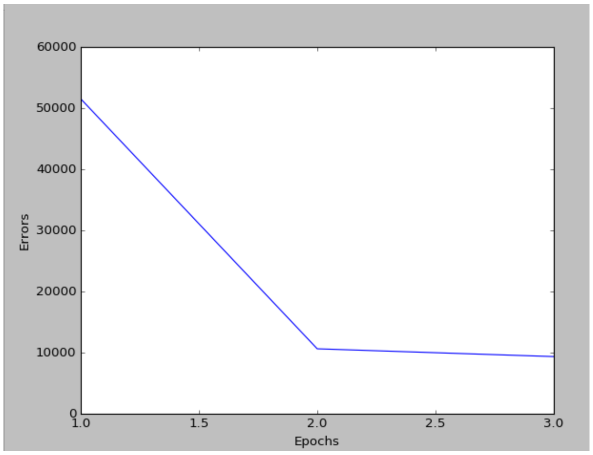


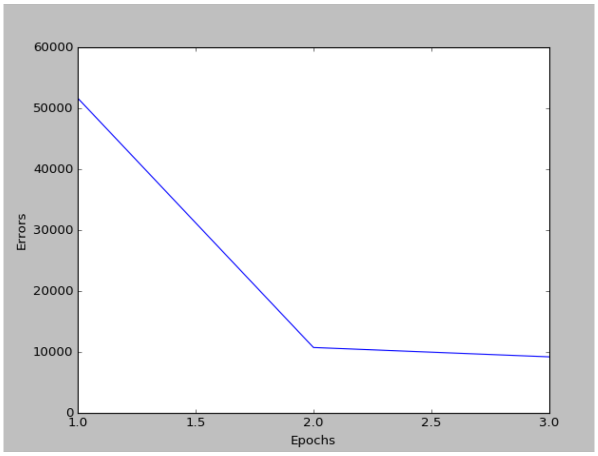
Train images size: 1000
Learning rate: 1
Error rate: 0.01



Train images size: 60000
Learning rate:
Error rate: 0


60000
50000
40000
30000
20000
10000
0
Errors
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Epochs


60000
50000
40000
30000
20000
10000
0
Errors
1.0
1.5
2.0
2.5
3.0
Epochs


60000
50000
40000
30000
20000
10000
0
Errors
1.0
1.5
2.0
2.5
3.0
Epochs