```
import struct
import numpy.random as random
import numpy as np
import matplotlib.pyplot as plt

# weights generated randomly which are used for step (f) and (g)
W = np.zeros((10,784))
for i in range(10):
    temp = (1-(-1))*random.sample(784) - 1
    W[i] = temp
mat_Wg=np.matrix(np.array(W))

# weights generated randomly which are used for step (i) first run
W = np.zeros((10,784))
for i in range(10):
    temp = (1-(-1))*random.sample(784) - 1
    W[i] = temp
mat_Wi=np.matrix(np.array(W))

# weights generated randomly which are used for step (i) second run
W = np.zeros((10,784))
for i in range(10):
    temp = (1-(-1))*random.sample(784) - 1
    W[i] = temp
mat_Wii=np.matrix(np.array(W))

# weights generated randomly which are used for step (i) third run
W = np.zeros((10,784))
for i in range(10):
    temp = (1-(-1))*random.sample(784) - 1
    W[i] = temp
mat_Wiii=np.matrix(np.array(W))

# function for multi-category Perceptron Training Algorithm for training images
def MCPTA_training(n,eta,epsilon,mat_W):
# importing training labels file
    train_lbl = open('train-labels.idx1-ubyte', 'rb')
    for i in range(2):
        train_lbl.read(4)
    des=[]
    for i in range(n):
        des.append(struct.unpack('>B', train_lbl.read(1))[0])

    flag=1
    epoch=0
    epoch_list=[]
    error_list=[]

# do epoch iterations until misclassifications converges below set threshold (i.e. epsilon)
    while(flag==1):
        error_epoch=0
# importing training images file
        train_img = open('train-images.idx3-ubyte', 'rb')
        for i in range(4):
            train_img.read(4)
# 1 grey scale image = 28X28 pixels
        for x in range(n):
            xi=[]
            for p in range(784):
                xi.append(struct.unpack('>B', train_img.read(1))[0])
            mat_xi=np.matrix(np.array(xi))

            mat_xi_T=np.transpose(mat_xi)
            mul = np.dot(mat_W,mat_xi_T)
            act = np.argmax(mul)
# updating weights in case of misclassification
            diff_mat = np.zeros((10,1))
            if des[x]!=act :
                diff_mat[act]=-1
                diff_mat[des[x]]=1
                error_epoch = error_epoch+1
```

```
                mat_W = mat_W + np.dot((eta*diff_mat),mat_xi)

        epoch = epoch +1
        if(epoch==100): break
        epoch_list.append(epoch)
        error_list.append(error_epoch)
        print("Epoch:",epoch,"Error: ",error_epoch)
        if (error_epoch/n)>epsilon:
            flag=1
        else:
            flag=0
    print("For ",n," training images, epoch list: ",epoch_list)
    print("For ",n," training images, error list: ",error_list)
 # plotting graph for number of epochs v/s number of misclassifications
    plt.title("Epoch V/S Miss")
    plt.xlim(0,epoch+1)
    plt.plot(epoch_list,error_list,'o-')
    plt.xlabel('Epoch')
    plt.ylabel('Misclassification')
    plt.show()
    return mat_W

# function for multi-category Perceptron Training Algorithm for test images

def MCPTA_test(n,mat_W1):
# importing test labels file
    test_lbl = open('t10k-labels.idx1-ubyte', 'rb')
    for i in range(2):
        test_lbl.read(4)
    des=[]
    for i in range(n):  ###60000
        des.append(struct.unpack('>B', test_lbl.read(1))[0])

    test_error=0
# importing test images file
    test_img = open('t10k-images.idx3-ubyte', 'rb')
    for i in range(4):
        test_img.read(4)
    for x in range(n):
        xi=[]
        for p in range(784):
            xi.append(struct.unpack('>B', test_img.read(1))[0])
        mat_xi=np.matrix(np.array(xi))

        mat_xi_T=np.transpose(mat_xi)
        mul = np.dot(mat_W1,mat_xi_T)
        act = np.argmax(mul)
# Calculating total misclassifications for test image
        diff_mat = np.zeros((10,1))
        if des[x]!=act :
            diff_mat[act]=-1
            diff_mat[des[x]]=1
            test_error = test_error + 1

    print("Total errors for ",n," test images: ",test_error)
    print("Percentage misclassification for ",n," test images: ",(test_error/n)*100)
    return

print("\r\n-----------------Multi-category PTA for 50 training images
--------------\r\n")
mat_W1= MCPTA_training(50,1,0,mat_Wg)
print("\r\n-----------------Multi-category PTA for 50 test images
--------------\r\n")
MCPTA_test(50,mat_W1)
print("\r\n-----------------Error Comparison of whole Test with 50 test images
--------------\r\n")
MCPTA_test(10000,mat_W1)

print("\r\n-----------------Multi-category PTA for 1000 training images
--------------\r\n")
mat_W1= MCPTA_training(1000,1,0,mat_Wg)
print("\r\n-----------------Multi-category PTA for 1000 test images
```

```
    ---------------\r\n")
140 MCPTA_test(1000,mat_W1)
141 print("\r\n-----------------Error Comparison of whole Test with 1000 test images
    ---------------\r\n")
142 MCPTA_test(10000,mat_W1)
143
144 print("\r\n<<<<<<<<<<<<<<<<<<<< step (h)- Terminating after 100 epochs as graphs
    don't converge >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>\r\n")
145 MCPTA_training(60000,1,0,mat_Wg)
146
147 print("\r\n-----------------Multi-category PTA for 60000 training images with
    different weights (say w1) ---------------\r\n")
148 mat_W1= MCPTA_training(60000,1,0.107,mat_Wi)
149 print("\r\n-----------------Multi-category PTA for 10000 test images with different
    weights (say w1) ---------------\r\n")
150 MCPTA_test(10000,mat_W1)
151
152 print("\r\n-----------------Multi-category PTA for 60000 training images with
    different weights (say w2) ---------------\r\n")
153 mat_W1= MCPTA_training(60000,1,0.107,mat_Wii)
154 print("\r\n-----------------Multi-category PTA for 10000 test images with different
    weights (say w2) ---------------\r\n")
155 MCPTA_test(10000,mat_W1)
156
157 print("\r\n-----------------Multi-category PTA for 60000 training images with
    different weights (say w3) ---------------\r\n")
158 mat_W1= MCPTA_training(60000,1,0.107,mat_Wiii)
159 print("\r\n-----------------Multi-category PTA for 10000 test images with different
    weights (say w3) ---------------\r\n")
160 MCPTA_test(10000,mat_W1)
```