CS 559: Neural Networks

Homework 3

Author:
Gaetano Coppoletta
Email:
gcoppo2@uic.edu

September 21, 2022

1 Introduction

This exercise consists in the usage of the multicategory perceptron training algorithm written in python for digit classification. The data set used for training and testing the neural network can be found at this link. We will use up to 60000 images from the training set and 10000 images from the test set. Each image is 28x28, so the neural network will have 784 nodes in the input layer and ten nodes in the output layer. The bias is ignored. We have to find 784 x 10 = 7840 weights such that the network outputs $[1,0,0,0,0,0,0,0,0,0]^T$ if the input image corresponds to 0, $[0,1,0,0,0,0,0,0,0,0]^T$ if the input image corresponds to 1 etc. Now we discuss the result obtained.

2 f

We run the algorithm with 50 images taken from the training set, η =1 and ϵ=0. The result is listed below.

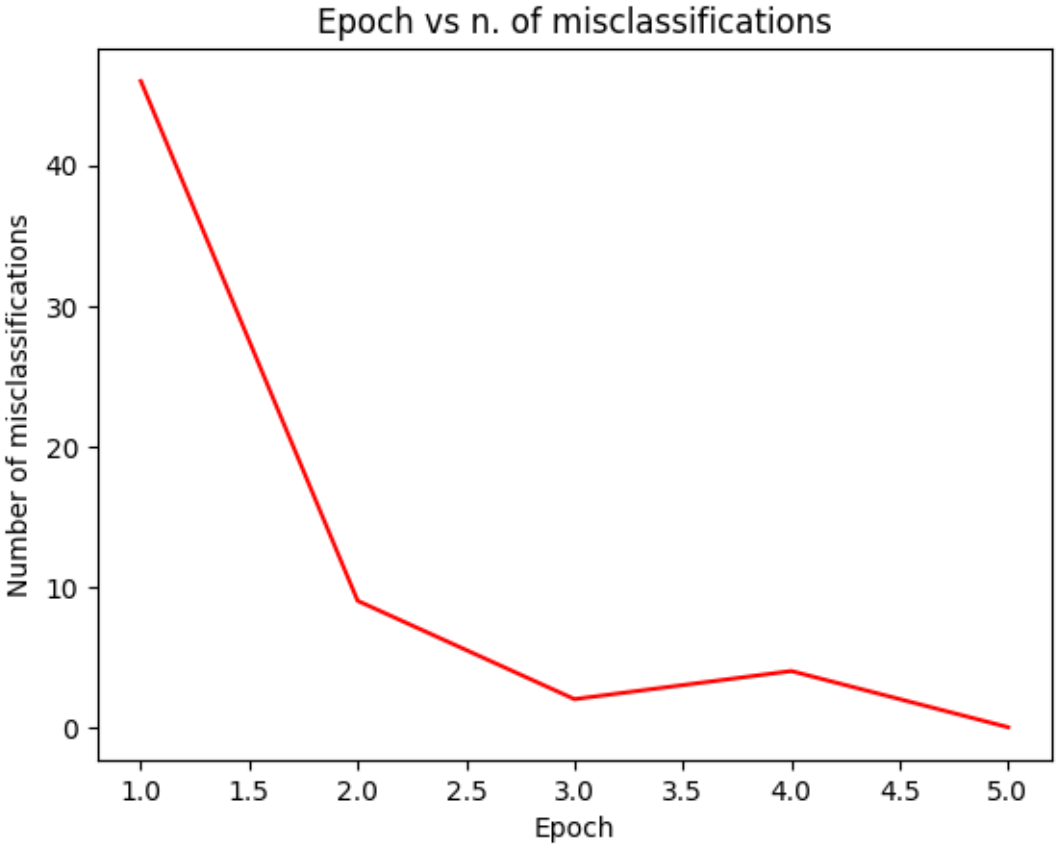

Figure 1: Epoch vs n. of misclassification with n=50

As we can see from the chart shown in 1 the algorithm converges to 0 in 5 epochs during the training phase, we obtain an error of 0.0%. When we test the neural network obtained with the test set we obtain an error of the 45.94%. This error is much bigger than the 0.0% obtained during the training phase. The difference is caused by the small number of images on which the NN is trained, only 50. The bigger the number of images on which the NN is trained the more the NN seems to be precise on the test set. This phenomenon is known as overfitting, which is a modeling error in statistics that occurs when a function is too closely aligned to a limited set of data points. As we can see from next section, a bigger set of training images will lead the algorithm to a minor percentage of errors.

3 g

We run the algorithm with 1000 images taken from the training set, η=1 and ϵ=0. The result is listed below.

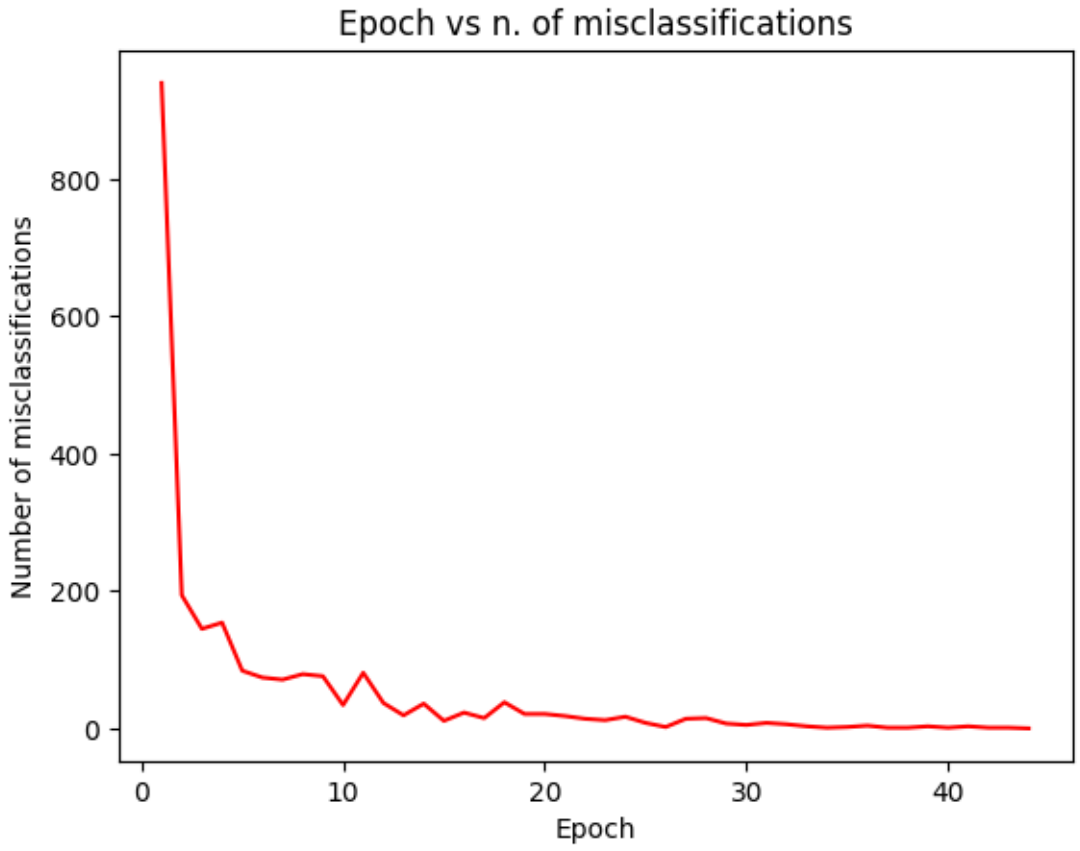

Figure 2: Epoch vs n. of misclassification with n=1000

As we can see from the chart shown in 2 the algorithm converges to 0 in 44 epochs during the

training phase, so we have an error of 0.0% During the test phase, instead, we obtain an error of the 17.65%. The error is higher than the one obtained in the training phase. Comparing the error obtained with 1000 images with the one obtained with 50 images we can observe that the second one is much bigger than the first one (almost three times bigger). This is due to the fact that the NN trained with 1000 images is more precise than the one trained with only 50 images, so in the one with 1000 images has less overfitting.

4 h

We run the algorithm with 60000 images taken from the training set, η=1 and ϵ=0. In order to stop the algorithm at a certain point we train the NN for 100 epochs. This is done because with a large number of images the NN could not converge to 0. The result is listed below.

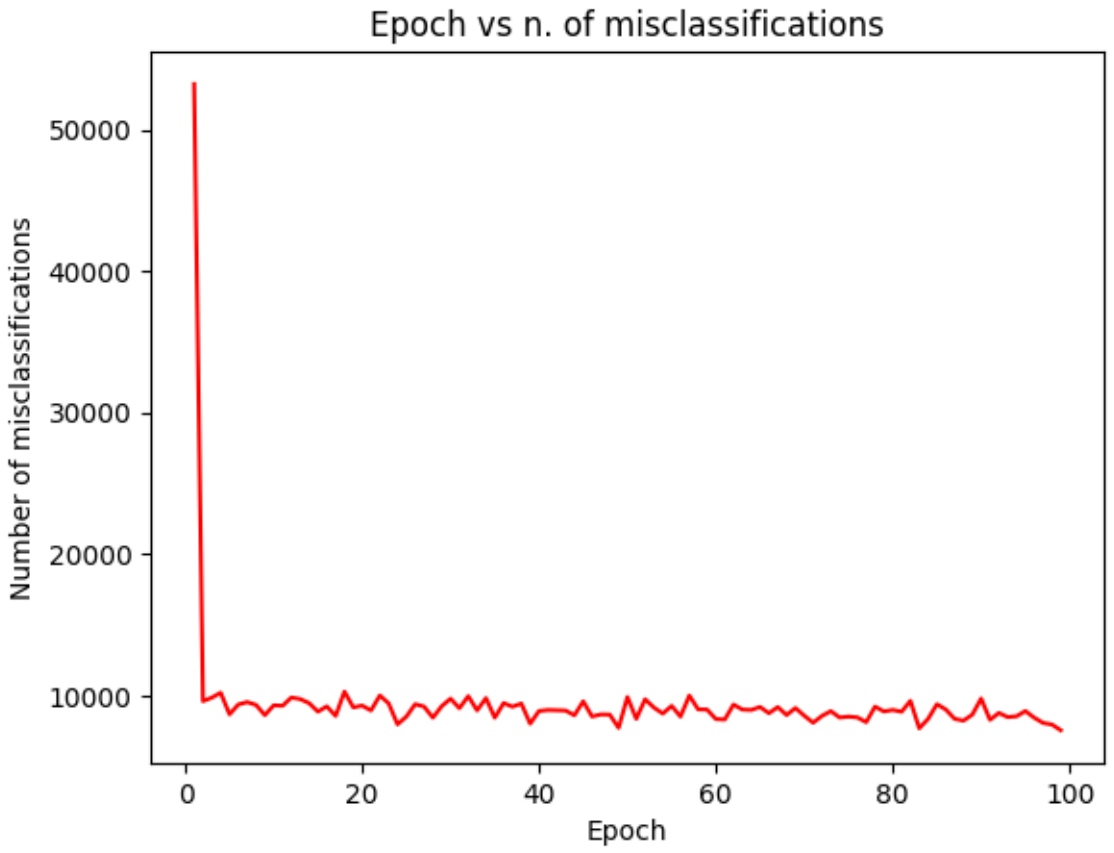

Figure 3: Epoch vs n. of misclassification with n=60000

As we can see from the chart shown in 3 the algorithm does not converge to 0, it is stopped after 100 epochs to observe the results. The error in this case is not 0.0% as in the previous case, but

since the algorithm does not converge to 0 we have an error of 12.58%. On the test set, instead, we observe an error of the 15.24%. Since the NN is trained on a lot of images, it obtains better results in the test set. This happens because a big training set allows the NN to become more general even if it does not converge, so we obtain better result on the test set.

5 i

We want to pick one ϵ different from 0 so that the algorithm will eventually terminate. We will use an ϵ=0.14 and η=0.4. The number of images of the training set is 60000. Now we want to run the algorithm with different initial weight, so we choose the weights between [-1,1] for the first run, between [-1/2,1/2] for the second run and between [-1/10,1/10] for the third run . The results are shown below.

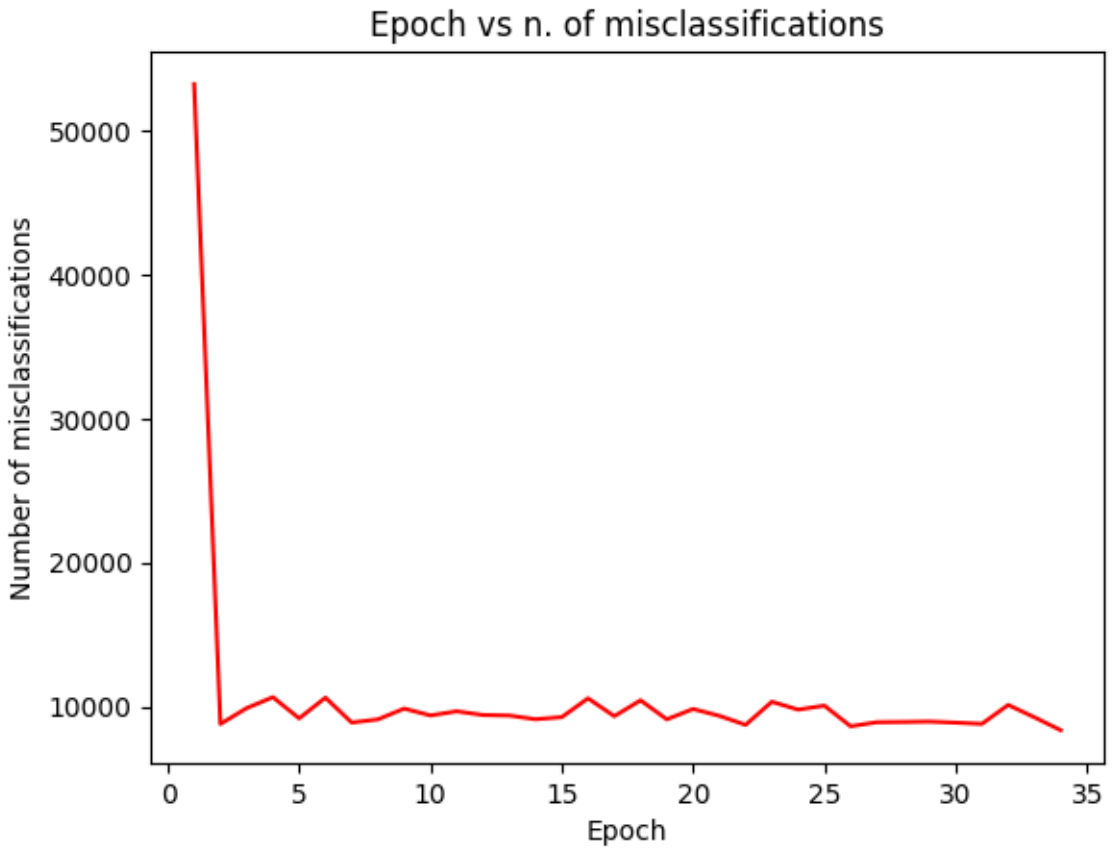

Figure 4: Weights choosen between [-1,1]

The algorithm with weights choosen between [-1,1] gives an error during the training phase of the 13.91%. During the test phase we have an error of the 16.25%.

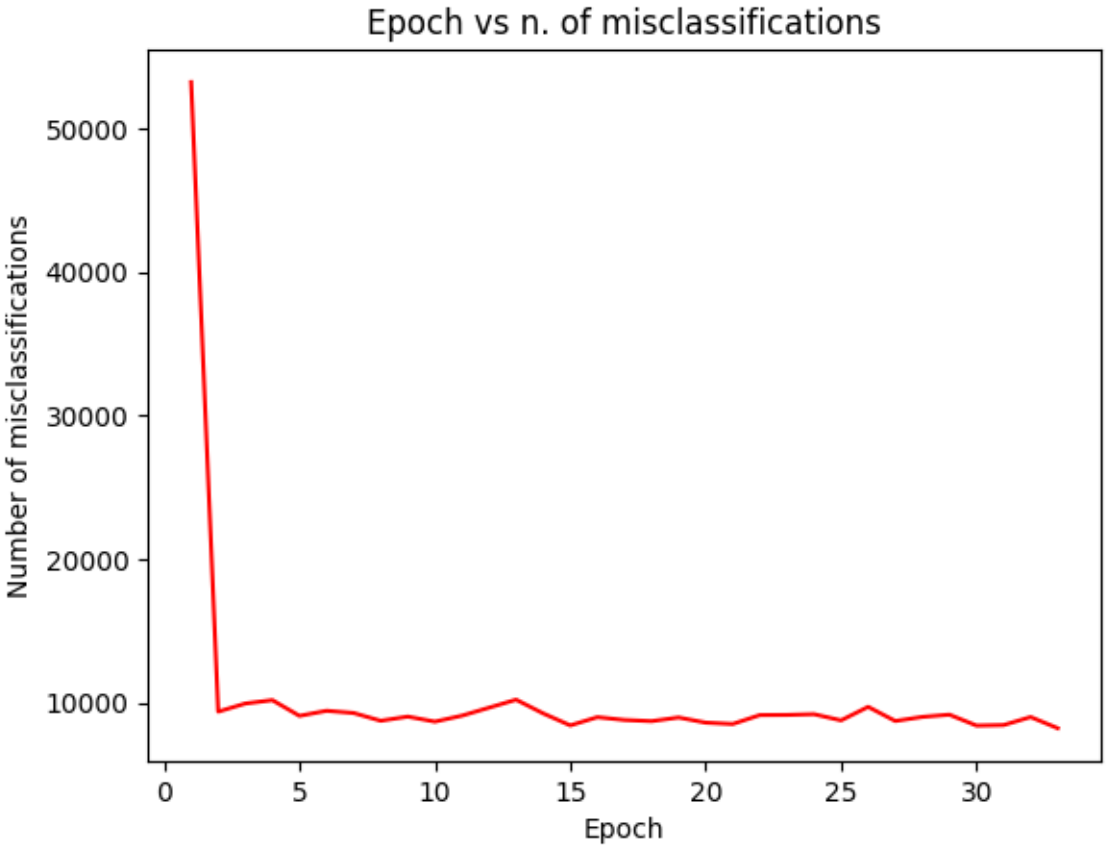

Figure 5: Weights choosen between [-1/2,1/2]

The algorithm with weights choosen between [-1/2,1/2] gives an error during the training phase of the 13.70%. During the test phase we have an error of the 16.84%.

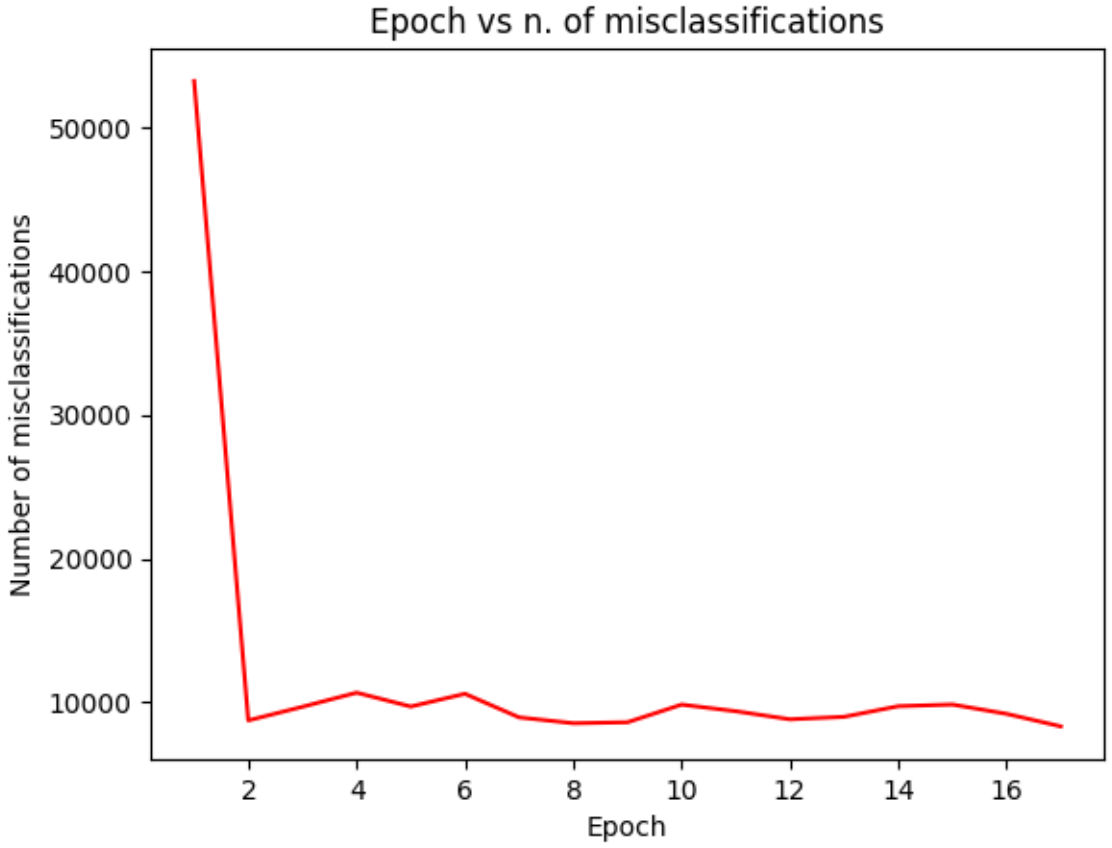

Figure 6: Weights choosen between [-1/10,1/10]

The algorithm with weights choosen between [-1/10,1/10] gives an error during the training phase of the 13.88%. During the test phase we have an error of the 15.32%.

Observing the three charts we notice that the number of epochs needed to terminate the algorihm is different in each case. The number of epochs is different because it depends on the initial weights, if they are "far" from one of the optimal solution it will takes more epochs to reach one of those solutions. One thing that is similar for all the three cases is the error on both the training and the test sets. This happens because of the fixed value that we assign to ϵ, which is equal to 0.4.

6 Code

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 import gzip
4 import random
```

```
5
6  def stepFunction(input):
7      output = np.zeros((len(input),1),dtype=float)
8      for i in range(0,len(input)):
9          if(input[i]>=0):
10             output[i]=1
11         else:
12             output[i]=0
13     return output
14
15
16 if __name__ == '__main__':
17     np.random.seed(1)
18     f = gzip.open('../data/train-images-idx3-ubyte.gz','r')
19
20     image_size = 28
21
22     #decide n <=60000
23     n = 1000
24
25     f.read(16)
26     buf = f.read(image_size * image_size * n)
27     data = np.frombuffer(buf, dtype=np.uint8).astype(np.float32)
28     data = data.reshape(n, image_size, image_size, 1)
29
30     images=np.zeros((784,n),dtype=float)
31     for i in range(0,n):
32         images[:,i]= np.asarray(data[i]).reshape(-1)
33
34     f = gzip.open('../data/train-labels-idx1-ubyte.gz','r')
35     f.read(8)
36     labels=np.zeros(n)
37     for i in range(0,n):
38         buf = f.read(1)
39         labels[i]=np.frombuffer(buf, dtype=np.uint8)
40
41
42     #parameters
43     eta=1
44     #eta=0.4 #for (i)
45     epsilon=0
46     #epsilon=0.1 #for (i)
47
48     #initialize random weights
49     W = np.random.uniform(-1,1,(10, 784))
50     #W = np.random.uniform(-1/2,1/2,(10, 784))
51     #W = np.random.uniform(-1/10,1/10,(10, 784))
52     epoch=1
53     em = np.zeros((99999,2),dtype=float)
54     #| epoch | misclassification |
55
56     #TRAINING
57     for i in range(0,n):
```

```
58          v=np.dot(W, images[:,i]) #v=Wxi
59          j=np.argmax(v)
60          if(j!=labels[i]):
61              em[epoch,1]=em[epoch,1]+1
62
63
64      for z in range(0,n):
65          label_bin=np.zeros((10,1))
66          label_bin[int(labels[z])]=1
67          Wxi=np.dot(W,images[:,z]) #Wx_i
68          W=W+eta*(np.dot((label_bin-stepFunction(Wxi)),(images[:,z]).reshape
                (1,784)))
69
70      em[epoch,0]=epoch
71      epoch=epoch+1
72
73      while(em[epoch-1,1]/n > epsilon):  #(and epoch!=100) in the while for point
            h
74          em[epoch,0]=epoch
75          for i in range(0,n):
76              v=np.dot(W, images[:,i]) #v=Wxi
77              j=np.argmax(v)
78              if(j!=labels[i]):
79                  em[epoch,1]=em[epoch,1]+1
80          epoch=epoch+1
81          #print(epoch)
82          for z in range(0,n):
83              label_bin=np.zeros((10,1))
84              label_bin[int(labels[z])]=1
85              Wxi=np.dot(W,images[:,z]) #Wx_i
86              #data_t=data[z]
87              W=W+eta*(np.dot((label_bin-stepFunction(Wxi)),(images[:,z]).reshape
                    (1,784)))
88
89
90      #plot
91      plt.title("Epoch vs n. of misclassifications")
92      plt.xlabel("Epoch")
93      plt.ylabel("Number of misclassifications")
94      plt.plot(em[1:epoch,0],em[1:epoch,1],'r')
95      plt.show()
96
97      print("Training: we have",em[epoch-1,1]/n *100,"%","error\n")
98
99      #TESTING
100     num_test_images=10000
101     f = gzip.open('../data/t10k-images-idx3-ubyte.gz','r')
102     f.read(16)
103     buf = f.read(image_size * image_size * num_test_images)
104     test_data = np.frombuffer(buf, dtype=np.uint8).astype(np.float32)
105     test_data = test_data.reshape(num_test_images, image_size, image_size, 1)
106     test_images = np.zeros((784,num_test_images),dtype=float)
107     for i in range(0,num_test_images):
```

```
108         test_images[:,i]=np.asarray(test_data[i]).reshape(-1)
109
110     f = gzip.open('../data/t10k-labels-idx1-ubyte.gz','r')
111     f.read(8)
112     test_labels=np.zeros(num_test_images)
113     for i in range(0,num_test_images):
114         buf = f.read(1)
115         test_labels[i] = np.frombuffer(buf, dtype=np.uint8).astype(np.int64)
116
117     test_errors=0
118
119     for i in range(0,num_test_images):
120             v_prime=np.dot(W,test_images[:,i]) #v'=Wx'_i
121             j=np.argmax(v_prime)
122             if(j!=test_labels[i]):
123                 test_errors=test_errors+1
124
125     print("Test: we have",test_errors/num_test_images *100,"%","error\n")
```