Fengnan Wang fwang40

**My Code:**

**HW2.py**

#!/usr/bin/python

import os, struct

from array import array as pyarray

from numpy import append, array, int8, uint8, zeros

import numpy as np

from pylab import \*

import random

import matplotlib.pyplot as plt

def load\_mnist(dataset="training", digits=np.arange(10)):

if dataset == "training":

fname\_img = os.path.join('/Users/ou/Dropbox/course/559 neural networks/HW/hw2/train-images-idx3-ubyte')

fname\_lbl = os.path.join('/Users/ou/Dropbox/course/559 neural networks/HW/hw2/train-labels-idx1-ubyte')

elif dataset == "testing":

fname\_img = os.path.join('/Users/ou/Dropbox/course/559 neural networks/HW/hw2/t10k-images-idx3-ubyte')

fname\_lbl = os.path.join('/Users/ou/Dropbox/course/559 neural networks/HW/hw2/t10k-labels-idx1-ubyte')

else:

raise ValueError("dataset must be 'testing' or 'training'")

flbl = open(fname\_lbl, 'rb')

magic\_nr, size = struct.unpack(">II", flbl.read(8))

lbl = pyarray("b", flbl.read())

flbl.close()

fimg = open(fname\_img, 'rb')

magic\_nr, size, rows, cols = struct.unpack(">IIII", fimg.read(16))

img = pyarray("B", fimg.read())

fimg.close()

ind = [ k for k in range(size) if lbl[k] in digits ]

N = len(ind)

images = zeros((N, rows, cols), dtype=uint8)

labels = zeros((N, 1), dtype=int8)

for i in range(len(ind)):

images[i] = array(img[ ind[i]\*rows\*cols : (ind[i]+1)\*rows\*cols ]).reshape((rows, cols))

labels[i] = lbl[ind[i]]

return images, labels

#imshow(images.mean(axis=0), cmap=cm.gray)

#show()

#print images[1]

#images /= 255.0

def get\_weights(m,n):

list\_x=[0]\*784

list\_w=[[ 0 for i in range(10)] for j in range(784)]

for i in range(1,784):

for j in range(1,10):

list\_w[i][j]=random.uniform(m,n)

return list\_w

def get\_x(image):

list\_x=[]

for i in range(28):

for j in range(28):

list\_x.append(image[i][j])

return list\_x

###################### initalize all global variable ########################

vout=[]

x=[]

####################### all functions ###########################

def sample(condition):

images=[]

labels=[]

if condition=='training':

for n in range(10):

image,label =load\_mnist('training',digits=[n])#the label n

images.append(image)

labels.append(label)

elif condition=='testing':

for n in range(10):

image,label =load\_mnist('testing',digits=[n])#the label n

images.append(image)

labels.append(label)

else:

print'the condition is wrong'

return images,labels

def equals(i,j,w):

dout=[0]\*10

x=get\_x(images[i][j])

v = np.dot(x,w)

v = v.tolist()

max\_index=v.index(max(v))

#desired\_output

dout[i]=1

if max\_index==i:

return 1,x,v,dout

else:

return 0,x,v,dout

# current\_output

def current\_output(vout):

cout=[]

for i in range(len(vout)):

cout.append(1 if vout[i] >= 0 else 0)

return cout

def pta(u,dout,cout,x,w):

for i in range(784):

for j in range(10):

w[i][j]=w[i][j]+u\*(dout[j]-cout[j])\*x[i]

return w

def train(w,u,n,minmum):

error\_epoch=[]

epoch = 0

while True:

# step 1 count errors

error = 0

for j in range(n/10):

for i in range(10):

(m,x,v,d)=equals(i,j,w)

#print m

c=current\_output(v)

if (m==0):

error = error + 1

elif (m==1):

error = error + 0

error\_epoch.append(error)

print "@epoch", epoch, "# errors = ", error

# step 2 add epoch

epoch = epoch + 1

# step 3 update W

for j in range(n/10):

for i in range(10):

(m,x,v,d)=equals(i,j,w)

c=current\_output(v)

w=pta(u,d,c,x,w)

# step 4 check if continue

# 0 should work

if (error\_epoch[epoch-1])<= minmum\*n:

break

return w

def test(w,u,n):

# step 1 count errors

error = 0

for j in range(n/10):

for i in range(10):

(m,x,v,d)=equals(i,j,w)

#print m

c=current\_output(v)

if (m==0):

error = error + 1

elif (m==1):

error = error + 0

return error

print'##################### u=1 min=0 ################################'

print'############ Q(f) 50 d ##################'

images,labels=sample('training')

w=get\_weights(-1,1)

w\_f=train(w,1,50,0)

print'########### Q(f) 10000 e ################'

images,labels=sample('testing')

#IndexError: index 892 is out of bounds for axis 0 with size 892

error=test(w\_f,1,8920)+test(w\_f,1,1080)

print "# errors of 50 = ", error

print'########### Q(g) 1000 d #################'

images,labels=sample('training')

w=get\_weights(-1,1)

w\_g=train(w,1,1000,0)

print'########### Q(g) 10000 e ################'

images,labels=sample('testing')

#IndexError: index 892 is out of bounds for axis 0 with size 892

error=test(w\_g,1,8920)+test(w\_g,1,1080)

print "# errors of 1000 = ", error

print'########### Q(h) 60000 d ################'

images,labels=sample('training')

w=get\_weights(-1,1)

#IndexError: index 5421 is out of bounds for axis 0 with size 5421

w\_n=train(w,1,54210,0)

w\_h=train(w\_n,1,5790,0)

print'########### Q(h) 10000 e ################'

images,labels=sample('testing')

#IndexError: index 892 is out of bounds for axis 0 with size 892

error=test(w\_h,1,8920)+test(w\_h,1,1080)

print "# errors of 60000= ", error

print '########### Repeat the following two subitems three times ##########'

print'########### first time u=10 minimum=0.01 ################'

images,labels=sample('training')

w=get\_weights(-1,1)

#IndexError: index 5421 is out of bounds for axis 0 with size 5421

w\_n=train(w,10,54210,0.01)

w\_h=train(w\_n,10,5790,0.01)

images,labels=sample('testing')

#IndexError: index 892 is out of bounds for axis 0 with size 892

error=test(w\_h,10,8920)+test(w\_h,10,1080)

print "# errors of 60000= ", error

print'########### second time u=0.1 minimum=0.01 ################'

images,labels=sample('training')

w=get\_weights(-1,1)

#IndexError: index 5421 is out of bounds for axis 0 with size 5421

w\_n=train(w,0.1,54210,0.01)

w\_h=train(w\_n,0.1,5790,0.01)

images,labels=sample('testing')

#IndexError: index 892 is out of bounds for axis 0 with size 892

error=test(w\_h,0.1,8920)+test(w\_h,0.1,1080)

print "# errors of 60000= ", error

print'########### third time u=10 minimum=0.1 ################'

images,labels=sample('training')

w=get\_weights(-1,1)

#IndexError: index 5421 is out of bounds for axis 0 with size 5421

w\_n=train(w,10,54210,0.1)

w\_h=train(w\_n,10,5790,0.1)

images,labels=sample('testing')

#IndexError: index 892 is out of bounds for axis 0 with size 892

error=test(w\_h,10,8920)+test(w\_h,10,1080)

print "# errors of 60000= ", error

**Results:**

eryao:eryao ou$ ./hw2.py

**(f) Run Steps (d) and (e) for n=50, =1,**

##################### u=1 min=0 ################################

############ Q(f) 50 d ##################

@epoch 0 # errors = 41

@epoch 1 # errors = 6

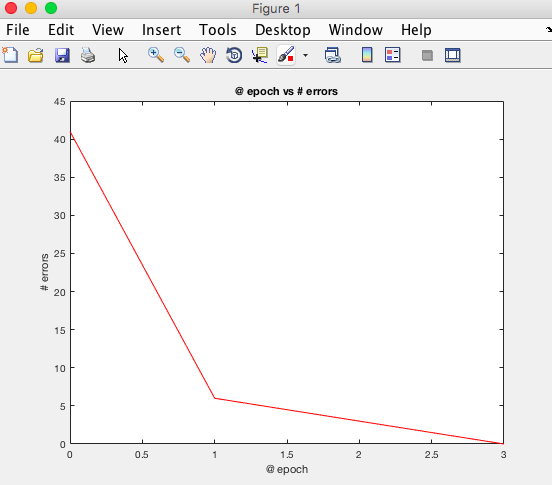
@epoch 2 # errors = 3

@epoch 3 # errors = 0

########### Q(f) 10000 e ################

**# errors of 50 = 4298/10000= 42.98%**

the percentages of errors obtained through the training and test samples are different ( epoch=0 82% vs 42.98% and finally 0% vs 42.98%), the weights obtained after training present a better result, which means the algorithm works.



**(g) Run Steps (d) and (e) for n=1000, =1,**

########### Q(g) 1000 d #################

@epoch 0 # errors = 911

@epoch 1 # errors = 127

@epoch 2 # errors = 140

@epoch 3 # errors = 120

@epoch 4 # errors = 69

@epoch 5 # errors = 59

@epoch 6 # errors = 62

@epoch 7 # errors = 59

@epoch 8 # errors = 33

@epoch 9 # errors = 54

@epoch 10 # errors = 48

@epoch 11 # errors = 30

@epoch 12 # errors = 41

@epoch 13 # errors = 16

@epoch 14 # errors = 46

@epoch 15 # errors = 22

@epoch 16 # errors = 15

@epoch 17 # errors = 19

@epoch 18 # errors = 26

@epoch 19 # errors = 9

@epoch 20 # errors = 14

@epoch 21 # errors = 14

@epoch 22 # errors = 

@epoch 23 # errors = 10

@epoch 24 # errors = 11

@epoch 25 # errors = 5

@epoch 26 # errors = 1

@epoch 27 # errors = 7

@epoch 28 # errors = 6

@epoch 29 # errors = 2

@epoch 30 # errors = 4

@epoch 31 # errors = 4

@epoch 32 # errors = 3

@epoch 33 # errors = 1

@epoch 34 # errors = 1

@epoch 35 # errors = 2

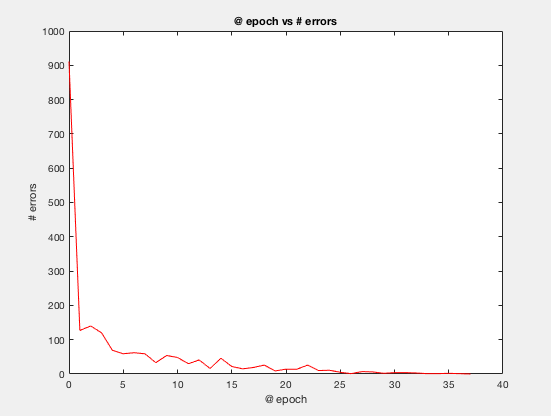
@epoch 36 # errors = 1

@epoch 37 # errors = 0

########### Q(g) 10000 e ################

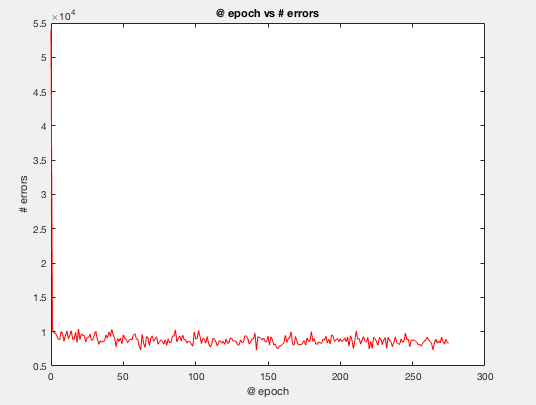
**# errors of 1000 = 1787/10000= 17.87%**

the percentages of errors obtained through the training and test samples are different (epoch=0 91.1% vs 17.87% and finally 0% vs 17.87%) and after comparing the weights obtained in 50 samples training and in 1000 samples, I found more samples could train a better and more idealized weight.



**(h) Run Steps (d) and (e) for n=60000, =1,**

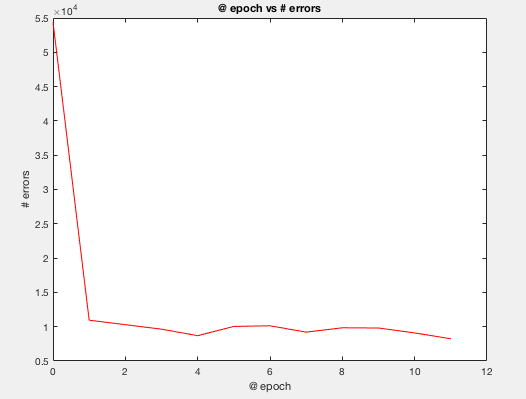
the number of epoch is too big; I will just show pictures below.



the algorithm does not converge because the training samples are not linearly separable. The error number begins to oscillate.

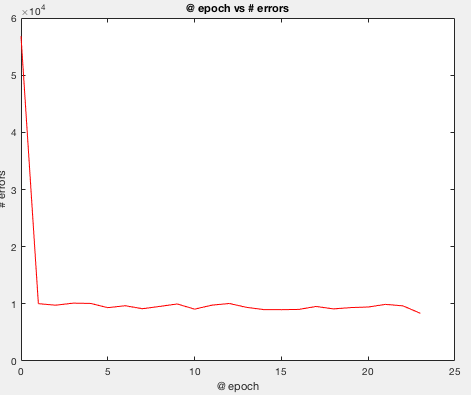
**(I)Repeat the following two sub items three times with different initial weights and comment on the results:**

**=1,**



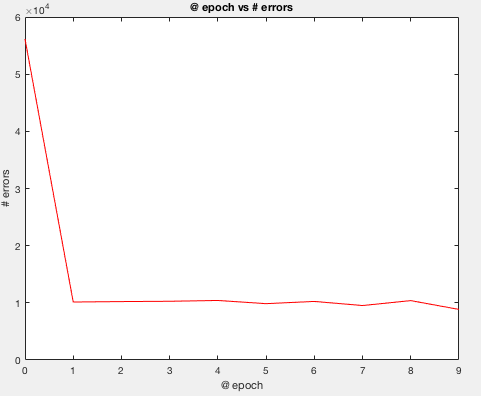
The percentage of errors number is 1630/10000 = 16.30%

**=10,**



The percentage of errors number is 1580/10000 = 15.80%

**=1,**



The percentage of errors number is 1918/10000 = 19.18%

After three experiences, the most obvious conclusion I can draw is that when becomes bigger, the epoch for convergence decrease significantly. And the final error percentage is nearly 10%. However, after training 1000 samples, we can also get the similar result. Moreover, no matter how much is, there is no obvious difference in final result, maybe epoch becomes bigger or maybe it will be smaller than prior one. Therefore, I need more experiences to verify these uncertain aspects.