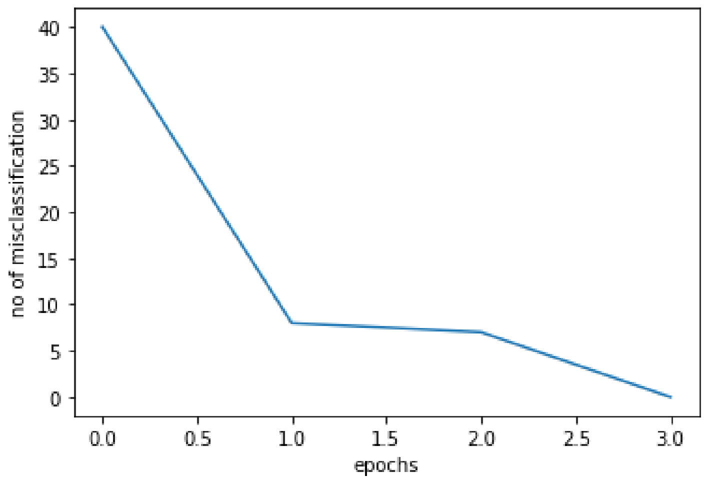
Initially,

n = 50, η = 1, and some very small e (here e = 0)



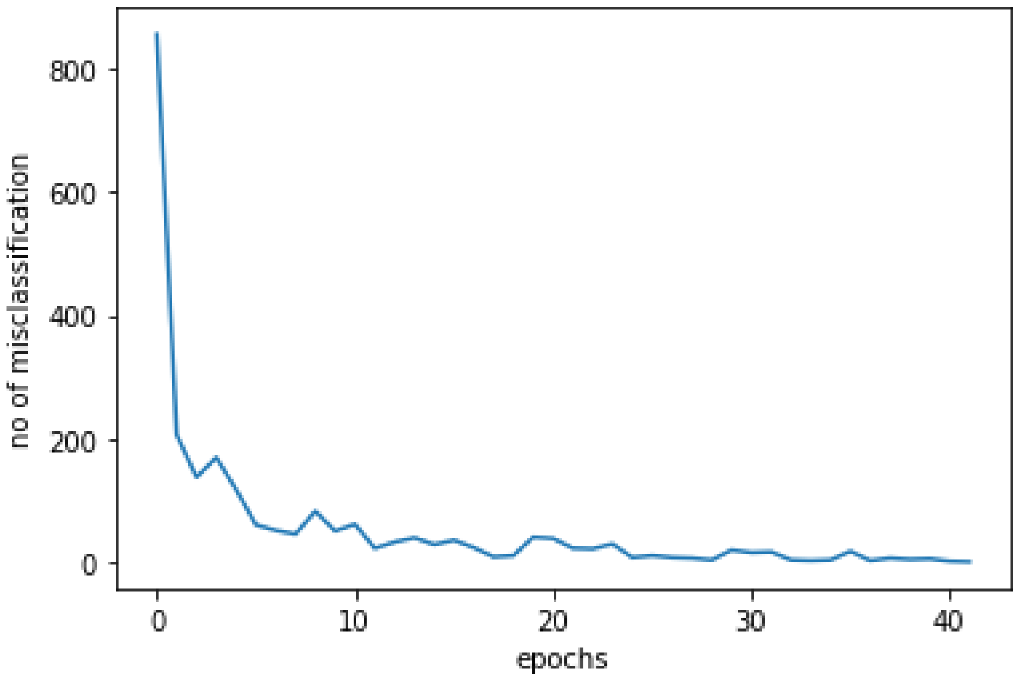
testing error = 4607

error % = 46.07

2) f) After training for 5 epochs, the network understands how to process the inputs better, which accounts for the lower percentage error in test data compared to the initial error in training data.

In the end, the number of errors for the training set reaches 0, however, since the number n is small, the error rate for the test set is high. This is due to the fact that the network simply doesn't have enough information to come to a conclusive outcome in other words, the network underfits.

for n = 1000, η = 1, and eta = 0



testing error = 1731

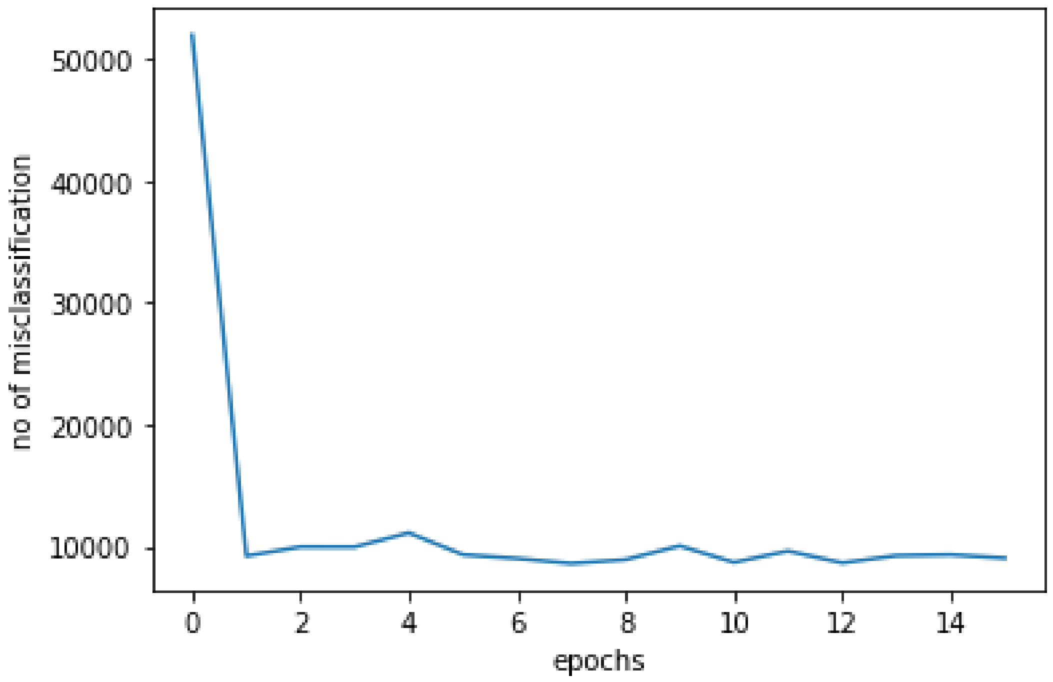
error % = 17.31

2) g) Our percentage of errors at the end of the 33-epoch training process is 17.31%, which is lower than the percentage of errors we obtained initially in the training data, since the network now knows how to handle inputs better.

Although the number of errors for the training set is ultimately zero, n=1000 is still not enough data for the error rate to be 0.

Our network has been trained on a lot more data and has been trained over a lot more epochs compared to (e), so it has a better updated weight and a lower error rate.

for n = 60000, η = 1, and eta = 0



testing error = 1744

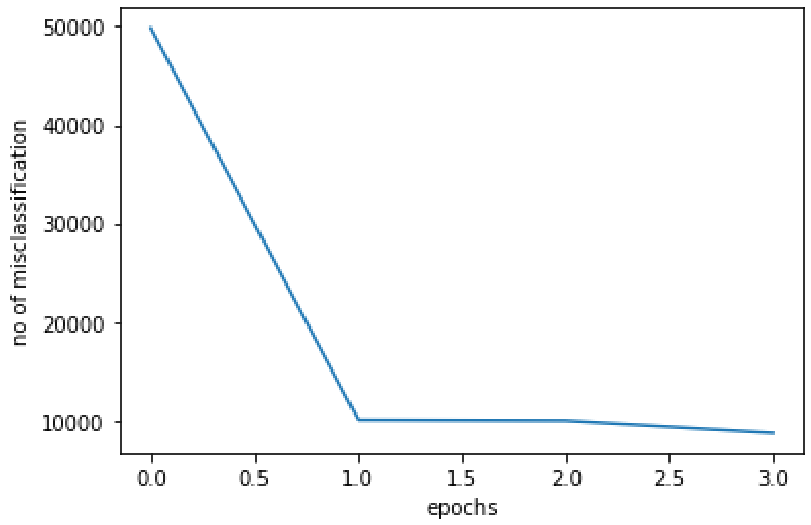
error % = 17.44

2) h) The algorithm was not converging (and running very slow) so I had to limit the Epoch number to 15. After epoch 1, the algorithm kept roughly the same number of misclassifications till epoch 15. Also the percentage of misclassified data is 17.44% for testing which is similar to n = 1000

2) i) after observing the value of updated epsilon, the threshold value I have chosen is 0.15 because it converges faster and the results are approximately the same when taking a smaller value.

__ first iteration

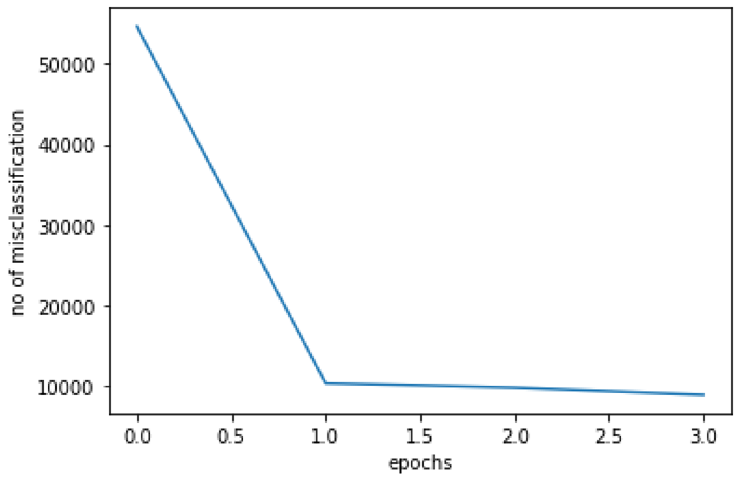
for n = 60000, η = 1, and eta = 0.15



testing error = 1522

error % = 15.22

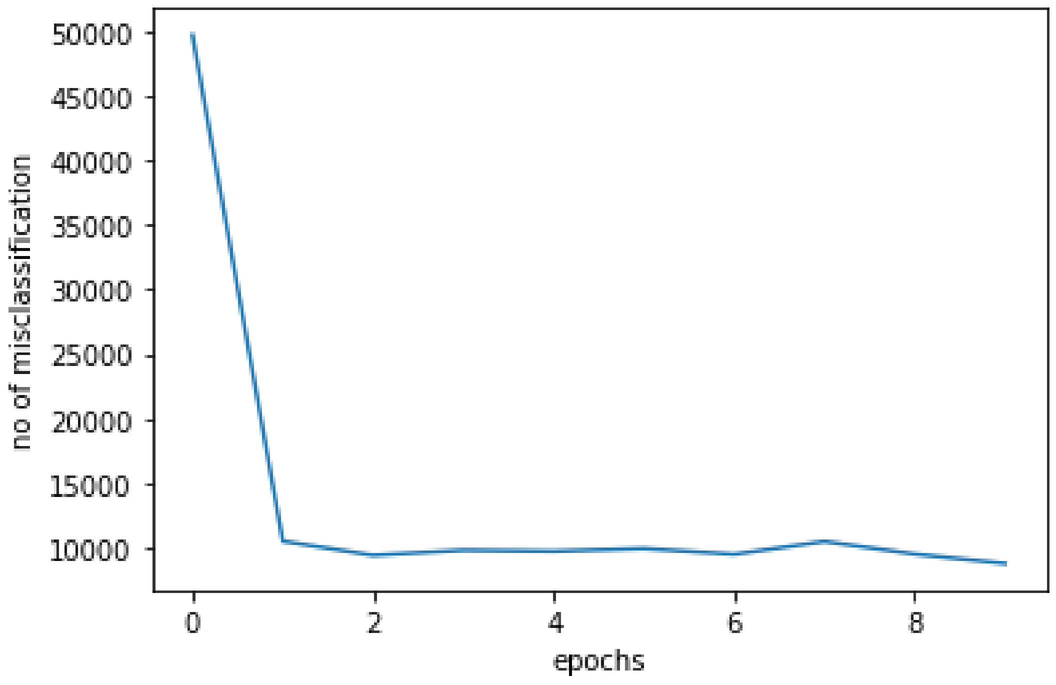
__second iteration



testing error = 1488

error % = 14.88

_____third iteration



testing error = 1560

error % = 15.6

conclusion: We can see that the graph for each attempt looks the same. The percentage of errors varies from 14-15% which is still comparable to the n = 1000 results.

CODE:

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

def Labels(name):
  f = gzip.open('/' + name)
  f.seek(4)
  images = struct.unpack('>I', f.read(4))[0]
  f.seek(8)
  labels = np.array(struct.unpack('>' + 'B' * images, f.read(images)))
  return (labels)

def Images(name):
  f = gzip.open('/'+name)
  f.seek(4)
  images = struct.unpack('>I', f.read(4))[0]
  c = struct.unpack('>I', f.read(4))[0]
  r = struct.unpack('>I', f.read(4))[0]

  start = f.seek(16)
  m = r * c
  pix_mat = []
  for i in range(images):
    f.seek(start + (i * m))
    pixel = np.array(struct.unpack('>' + 'B' * m, f.read(m)))
    pix_mat.append(pixel)
  return (np.array(pix_mat), images, r, c)

def test_train(ix,type):
  if (type == 'train'):
      numImages = TR_im
      labels  = TR_LB
  elif (type == 'test'):
      numImages = TS_IM
      labels = TS_LB
  if (ix <= numImages):
      return (labels[ix])

def Input(ix,type):
  d = np.zeros(10)
  label = test_train(ix,type)
  d[label] = 1
  return (d)

def PTA(W,n,eta, e):
  epoch = 0
  error = []
  pix_mat = trainingImgs
  labels = TR_LB
  while True:
      mis = 0
      for ix in range(n):
        v = W @ pix_mat[ix]
        if (np.argmax(v) != labels[ix]):
            mis += 1
      error.append(mis)
      epoch = epoch + 1
      for ix in range(n):
        W = W + eta*(Input(ix,type='train') - step_(W @ pix_mat[ix])).
reshape(-1, 1) @ (
            pix_mat[ix].reshape(-1, 1).T)
      print(error[epoch - 1] / n)
      if ((error[epoch - 1] / n <= e ) or (epoch>15)):
        break
  return ( W ,error,epoch)

def testing( W, n):
  error = 0
  testImages =testImgs
  for ix in range(n):
      v = W @ testImages[ix]
      if (np.argmax(v) != TS_LB[ix]):
        error += 1
  return error

def step_(X):
  if (type(X) == np.ndarray or type(X) == list):
      for ix, x in enumerate(X):
        if (x >= 0):
            X[ix] = 1
        else:
            X[ix] = 0
  elif (type(X) == int):
        if (x >= 0):
            X = 1
        else:
            X = 0
  return (X)

def sign_(X):
  if (type(X) == np.ndarray or type(X) == list):
      for ix, x in enumeeta(X):
        if (x < 0):
            X[ix] = -1
        elif (x == 0):
            X[ix] = 0
        elif (x > 0 ):
            X[ix] = 1
  elif (type(X) == int):

    if (x < 0):
        X = -1
    elif (x == 0):
        X = 0
    elif (x > 0 ):
        X = 1
  return X
```

```
def plot(epoch,misList):
  plt.plot(np.array(range(epoch)),misList)
  plt.xlabel('epochs')
  plt.ylabel('no of misclassification')
  plt.show()

m = r * c
W = np.empty((0, m))
for j in range(10):
    w = np.array([random.uniform(-1, 1) for i in range(m)])
    W = np.vstack([w, W])
trainingImgs,TR_im,r,c = Images('train-images-idx3-ubyte')
TR_LB = Labels('train-labels-idx1-ubyte')
testImgs,TS_IM,_,_ = Images('t10k-images-idx3-ubyte')
TS_LB = Labels('t10k-labels-idx1-ubyte')



W_upd ,epoch_erros ,epoch = PTA(W,n=60000,eta=1,e=0.15)
plot(epoch,epoch_erros)
error = testing(W_upd, TS_IM)
print("testing error = ",error)
print("error % = ", error/100)
```