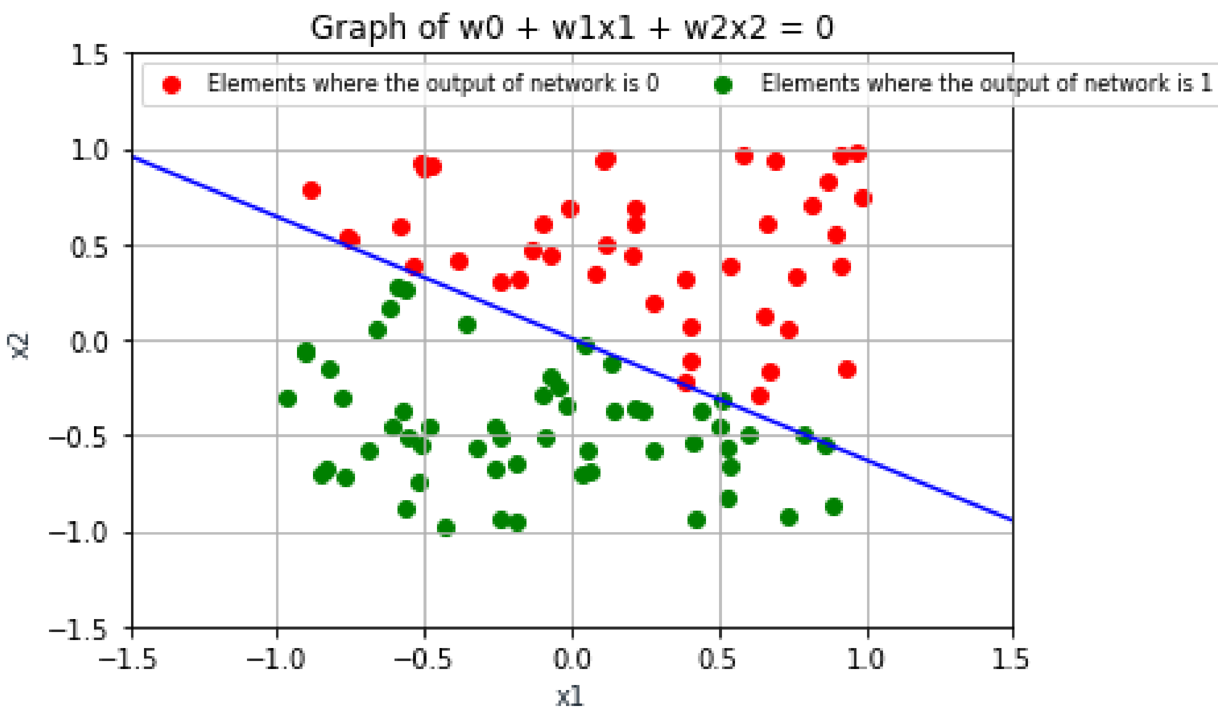
Q1) Write a computer program that runs the perceptron training algorithm with the step activation function u(·). Implement the following steps and report your results

With 100 elements:

The overall composition of the dataset is:



Here W0, W1, W2 = [-0.009476335089441956, -0.2991156742653023, 0.8507104983388833] which are the perfect Weights For classification

Random Weights W0_,W1_,W2_ = [-0.808725,-0.08573708, 0.930979] are selected and fed into the neural network

Number of misclassifications are 49 when using the above random weight

After one epoch

Weights become [[-0.87802575]

[-0.2029027]

[-1.14059471]]

And Number of misclassifications reduce to 38

Next,

Epoch Number: 2

Updated weights: [[-0.87802575]

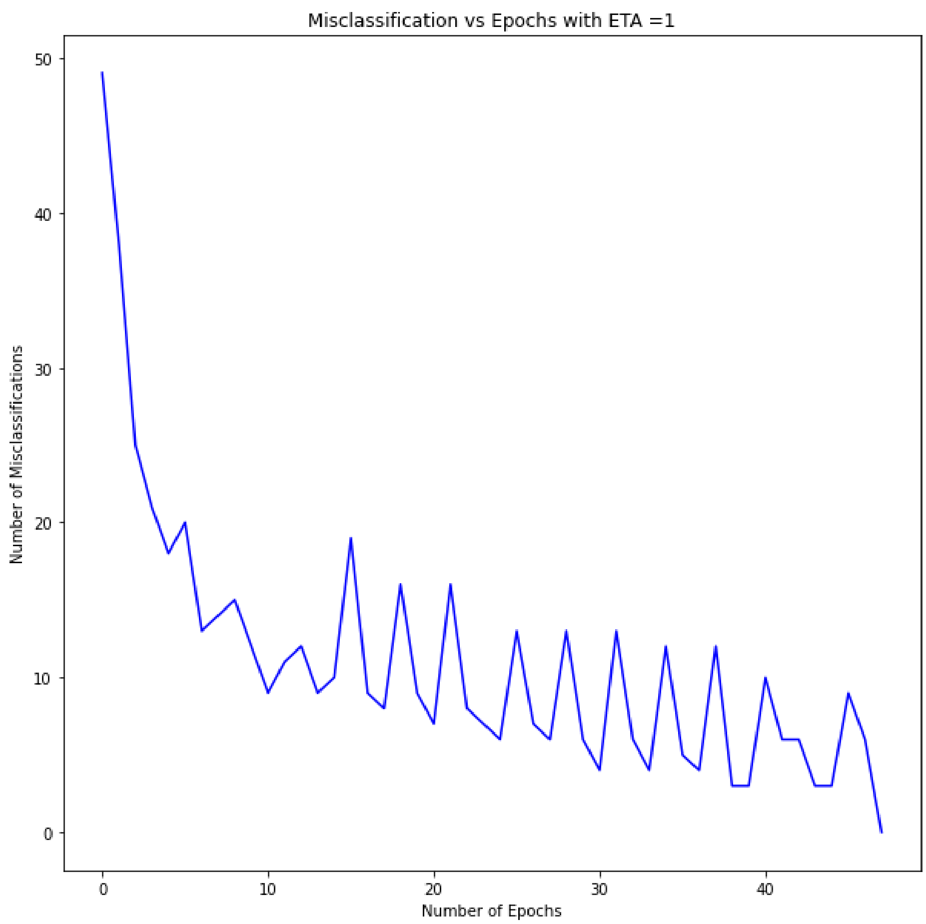
[0.88141992]

[-1.73969219]]

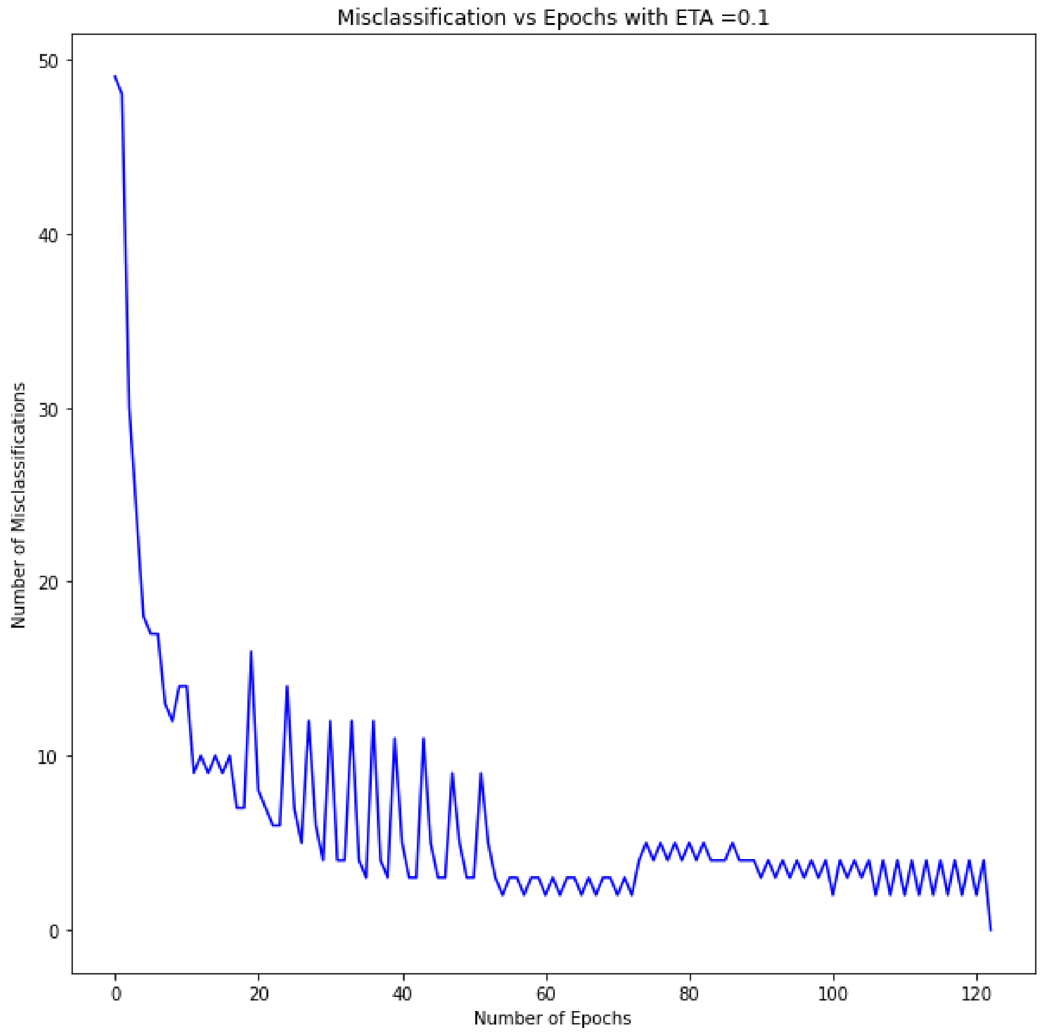
Number of missclassifications: 25

And so on until convergence

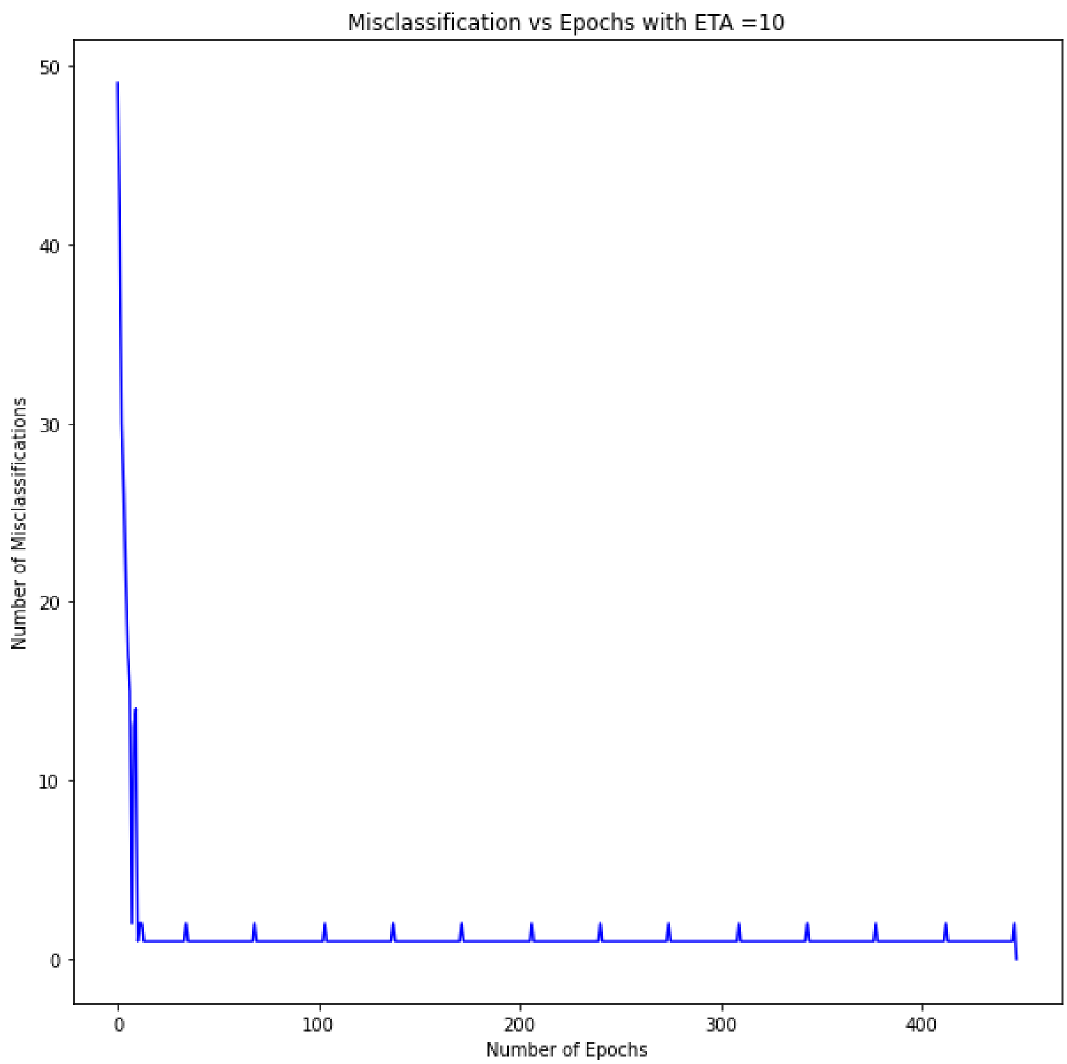
The final weights with ETA = 1 are [[0.12197425] [3.36640913] [-8.78704387]] on epoch 47 comparing to the optimal weights [-0.009476335089441956, -0.2991156742653023, 0.8507104983388833] they are very different but since the perceptron converges they successfully classify the elements



Now same is repeated by changing ETA to 0.1 and then to ETA = 10



Eta = 0.1 causes the network to take various tiny steps to convergence, hence the epochs needed exceed 120. This eta is very low in value and the network needs higher value to make it converge faster.



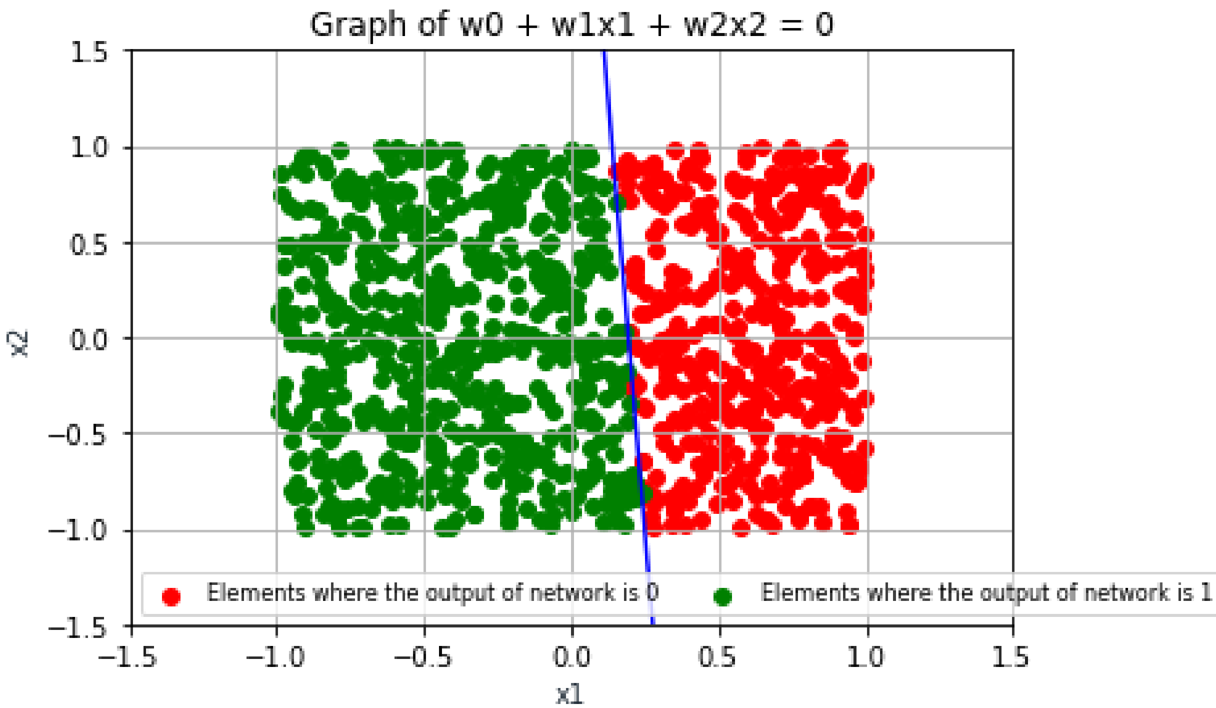
With Eta=10, this network shoots from the local minimum to another point which is beyond the minimum. This Eta is too high as the network will keep on shooting back and forth until it reaches convergence hence the number of epochs needed is well beyond 450

Comment on whether we would get the exact same results (in terms of the effects of η on training performance) if we had started with different w0, w1, w2, S, w0_ 0 , w0_ 1 , w0_ 2 .

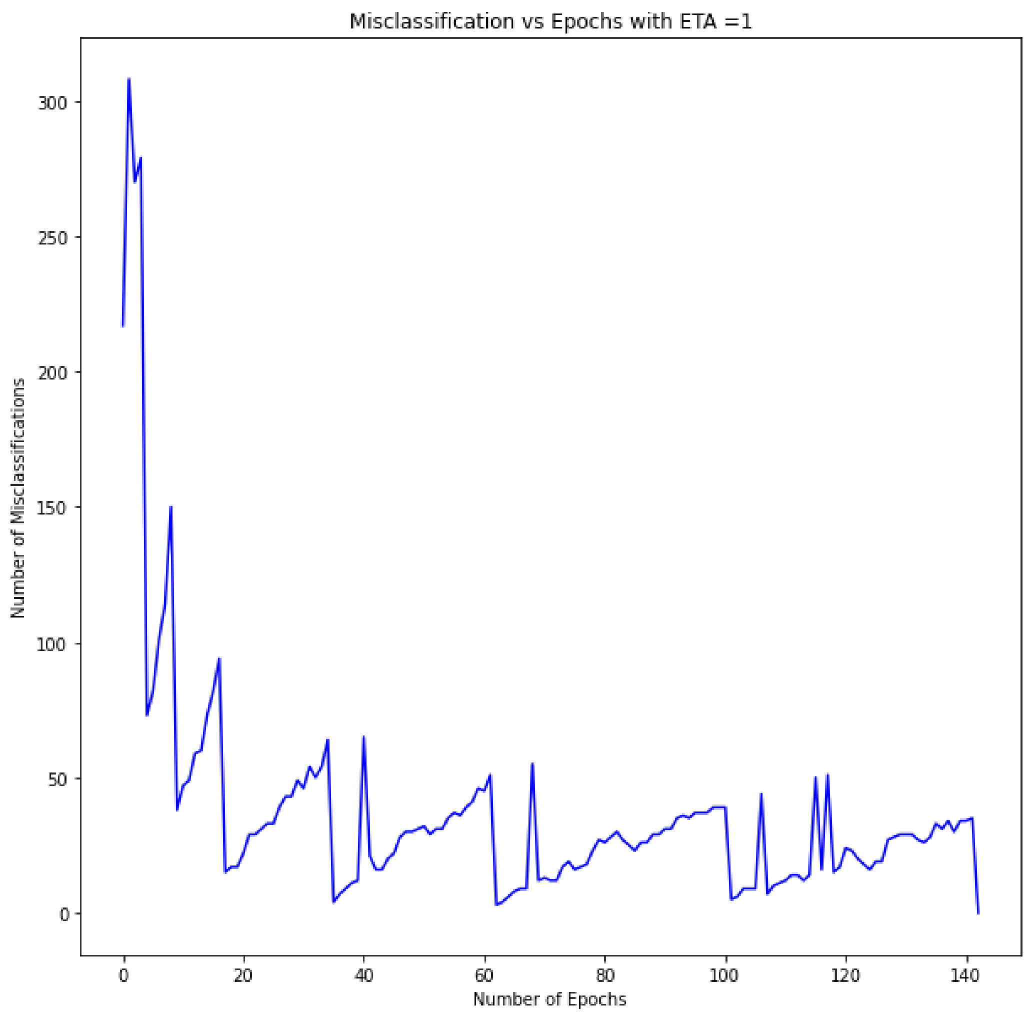
Depending on the values of weights, an ideal Eta (eta with the least number of time taken to convergence) could be anything, so for some problems 0.1 is more ideal than 1 and for some maybe 10 is more ideal.

Do the same experiments with n = 1000 samples

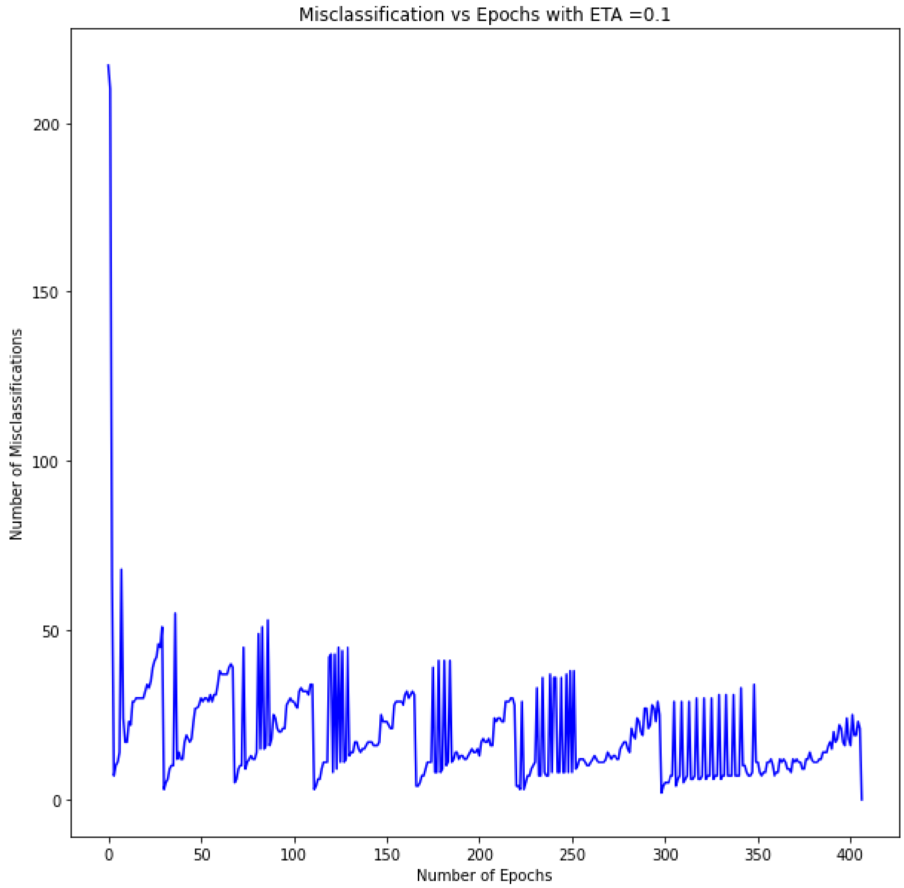
With n = 1000



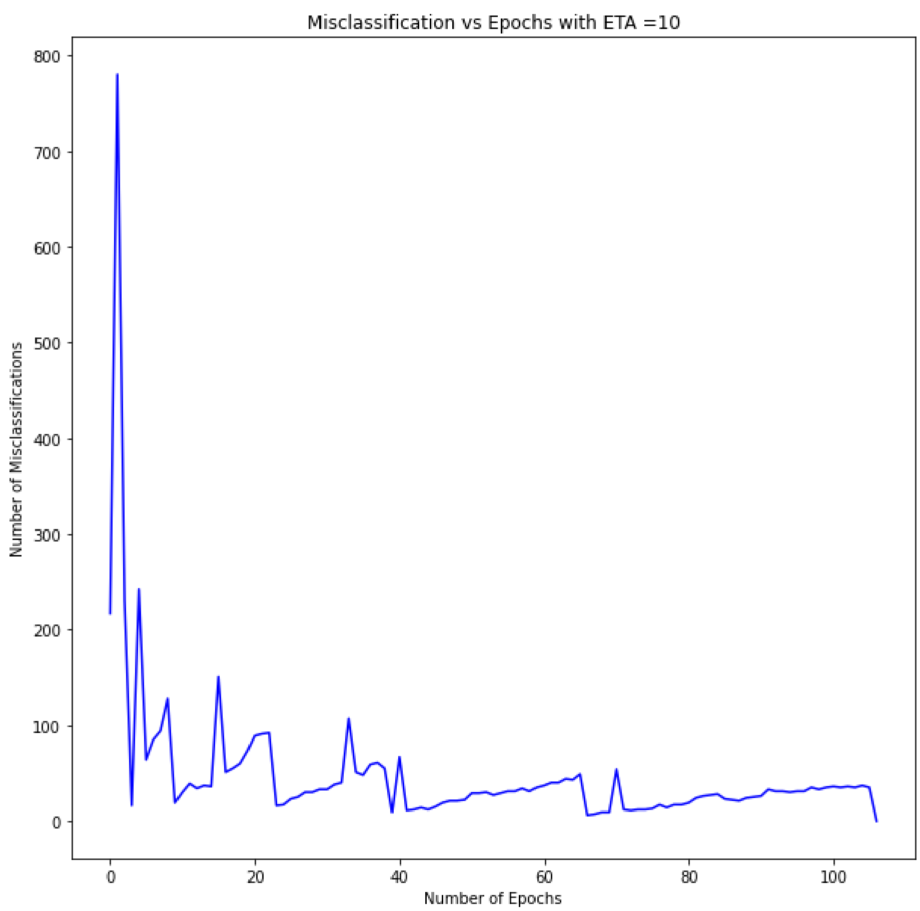
Graph of w0 + w1x1 + w2x2 = 0
x2
x1
Elements where the output of network is 0
Elements where the output of network is 1



Misclassification vs Epochs with ETA =1
Number of Misclassifications
Number of Epochs



Misclassification vs Epochs with ETA =0.1
Number of Misclassifications
Number of Epochs
0
50
100
150
200
250
300
350
400



Misclassification vs Epochs with ETA =10
Number of Misclassifications
Number of Epochs
0
100
200
300
400
500
600
700
800
20
40
60
80

With 1000 samples, the number of epochs required for convergence increase drastically

Hence, the amount of time required to process is much higher than when the number of elements are 100.

Here ideal ETA is closer to 10.

Code:

```
import numpy as np
import matplotlib.pyplot as plt
w0 = np.random.uniform(-0.25,0.25)
w1 = np.random.uniform(-1,1)
w2 = np.random.uniform(-1,1)
n = int(input("Enter value of N (ie number of elements)"))
xi = np.random.uniform(-1,1,size = (n,2))
b = np.ones((n,1))
xi_ = np.hstack((b , xi))

w = [w0 , w1 , w2]
print(w, "Perfect Weights For classification")
S0 = np.array([])
S1 = np.array([])
s0 = []
s1= []
for i in range(n):
  q = np.dot(w, xi_[i])

  if q < 0:
    S0 = np.vstack(xi_[i])
    s0.append(S0)

  else:
    S1 = np.vstack(xi_[i])
    s1.append(S1)

dataset = s0 + s1
s0 = np.asarray(s0)
s1 = np.asarray(s1)
dataset = np.asarray(dataset)

def plot (w,s0,s1):
  x1 = np.linspace(-5,5,n)
  x2 = (-w[0]-w[1]*x1)/w2
  plt.plot(x1, x2, '-b', label='perfect classification line')
  plt.title('Graph of w0 + w1x1 + w2x2 = 0')
  plt.xlabel('x1', color='#1C2833')
  plt.ylabel('x2', color='#1C2833')
  plt.legend(loc='upper left')
```

```
  S0_x = []
  S0_y = []
  S1_x= []
  S1_y= []
  for i in range(len(s0)):
    S0_x.append(s0[i][1])
    S0_y.append(s0[i][2])
  for i in range(len(s1)):
    S1_x.append(s1[i][1])
    S1_y.append(s1[i][2])
  scat1 = plt.scatter(S0_x,S0_y,color = 'red', label='Elements where the output of ne
twork is 0')
  scat2 = plt.scatter(S1_x,S1_y,color = 'green', label = 'Elements where the output o
f network is 1')
  plt.legend((scat1, scat2),
           ('Elements where the output of network is 0', 'Elements where the output o
f network is 1'),
           scatterpoints=1,
           loc='best',
           ncol=3,
           fontsize=8)
  plt.ylim([-1.5,1.5])
  plt.xlim([-1.5,1.5])
  plt.grid()
  plt.show()



w0_ = np.random.uniform(-1,1)
w1_ = np.random.uniform(-1,1)
w2_ = np.random.uniform(-1,1)
w_ = [w0_ , w1_ , w2_]
print(w_, 'randomly selected weights')
##notepad

lab1 = np.ones(len(s0),dtype = int)
lab2 = np.zeros(len(s1),dtype = int)

label = np.append(lab1,lab2)

def miscal(dataset, w_):
  miscal_ = 0
  for i in range(len(dataset)):
    y = w_[0] + (dataset[i][1]*w_[1]) + (dataset[i][2]*w_[2])
    if (y[0] >= 0):
      y[0] = int(1)
    else:
      y[0] = int(0)
    if (y[0] != label[i].astype(int)):
      miscal_ = miscal_ +1
```

```python
  return miscal_


def PTA(w_,xi):
  epoch = 0
  omegas = []
  miss = []
  while (miscal(dataset, w_) != 0):
        miss.append(miscal(dataset, w_))
        print ('Number of missclassifications: ', miss[epoch])
        epoch = epoch + 1
        print ('Epoch Number: ', epoch)

        for i in range(len(dataset)):
            y = w_[0] + (dataset[i][1]*w_[1]) + (dataset[i][2]*w_[2]) ##dot product
            if y[0] >= 0:
              y = 1
            else:
              y = 0

            difference = label[i]-y
            updated_input =[1]+dataset[i][0:2]

            if difference != 0:
              ip1 = dataset[i][0]*eta*difference
              ip2 = dataset[i][1]*eta*difference
              ip3 = dataset[i][2]*eta*difference
              a = w_[0]+ip1
              b = w_[1]+ip2
              c = w_[2]+ip3
              w_ = [a, b, c]
              w_ = np.asarray(w_)
        #print ('Updated weights: ', w_)
        omegas.append(w_)
  final_misclassification = miscal(dataset,w_)
  print ('Final weights: ', w_)
  return omegas, miss


plot(w,s0,s1)

eta = 1
PTA(w_,dataset)
print ('Initial weight: ' , w_)
omegas=[]
omegas, miss = PTA(w_,xi_)
n_epochs = range(len(omegas)+1)
fig, ax = plt.subplots(figsize=(10,10))
ax.plot(n_epochs, miss+[0], c = 'blue')
```

```
plt.title('Misclassification vs Epochs with ETA =' + str(eta))
plt.ylabel('Number of Misclassifications')
plt.xlabel('Number of Epochs')
plt.show()


eta = 0.1
PTA(w_,dataset)
print ('Initial weight: ' , w_)
omegas=[]
omegas, miss = PTA(w_,xi_)
n_epochs = range(len(omegas)+1)
fig, ax = plt.subplots(figsize=(10,10))
ax.plot(n_epochs, miss+[0], c = 'blue')

plt.title('Misclassification vs Epochs with ETA =' + str(eta))
plt.ylabel('Number of Misclassifications')
plt.xlabel('Number of Epochs')
plt.show()


eta = 10
PTA(w_,dataset)
print ('Initial weight: ' , w_)
omegas=[]
omegas, miss = PTA(w_,xi_)
n_epochs = range(len(omegas)+1)
fig, ax = plt.subplots(figsize=(10,10))
ax.plot(n_epochs, miss+[0], c = 'blue')

plt.title('Misclassification vs Epochs with ETA =' + str(eta))
plt.ylabel('Number of Misclassifications')
plt.xlabel('Number of Epochs')
plt.show()
```