PURVANG LAPSIWALA

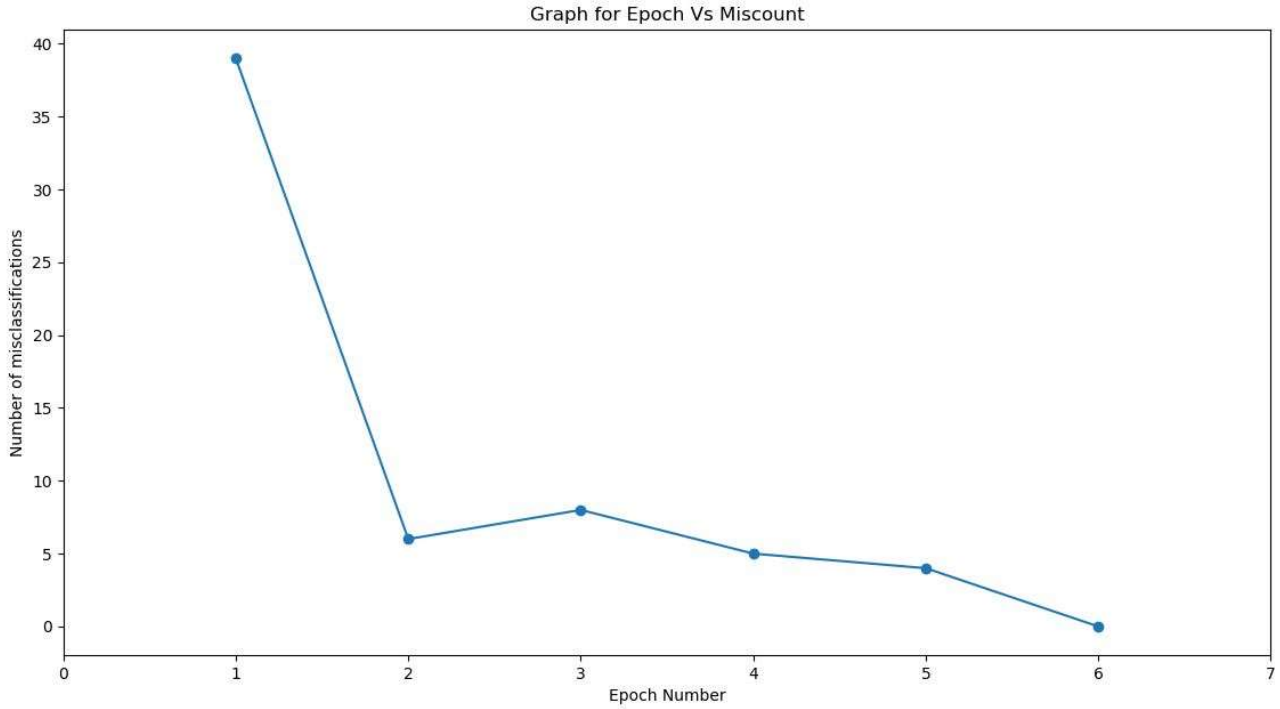
UIN = 662689378

Answer: 4

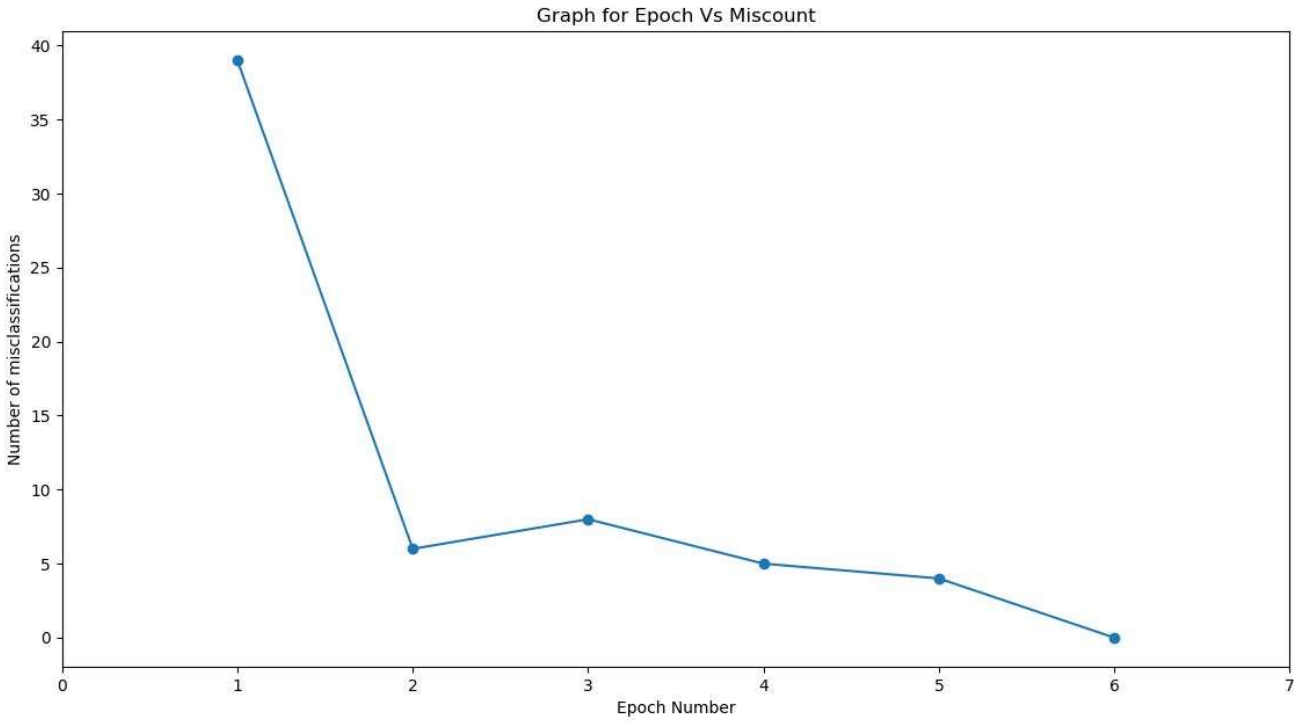
(f)

Taking 50 number of elements (Images) for training , learning rate 1 and epsilon equal to 0, Perceptron Training Algorithm took 6 epoch to converge and final error and error percentage both becomes 0 after 6^{th} epoch.

When testing algorithm is done with 50 elements and same weight obtained from above PTA with learning rate 1 and epsilon 0, error percentage or percentage of misclassified samples obtained is 38%.



Now, Testing Algorithm is done for 10000 elements and for the same weighs obtained from above and result for percentage of misclassification was 43.7%

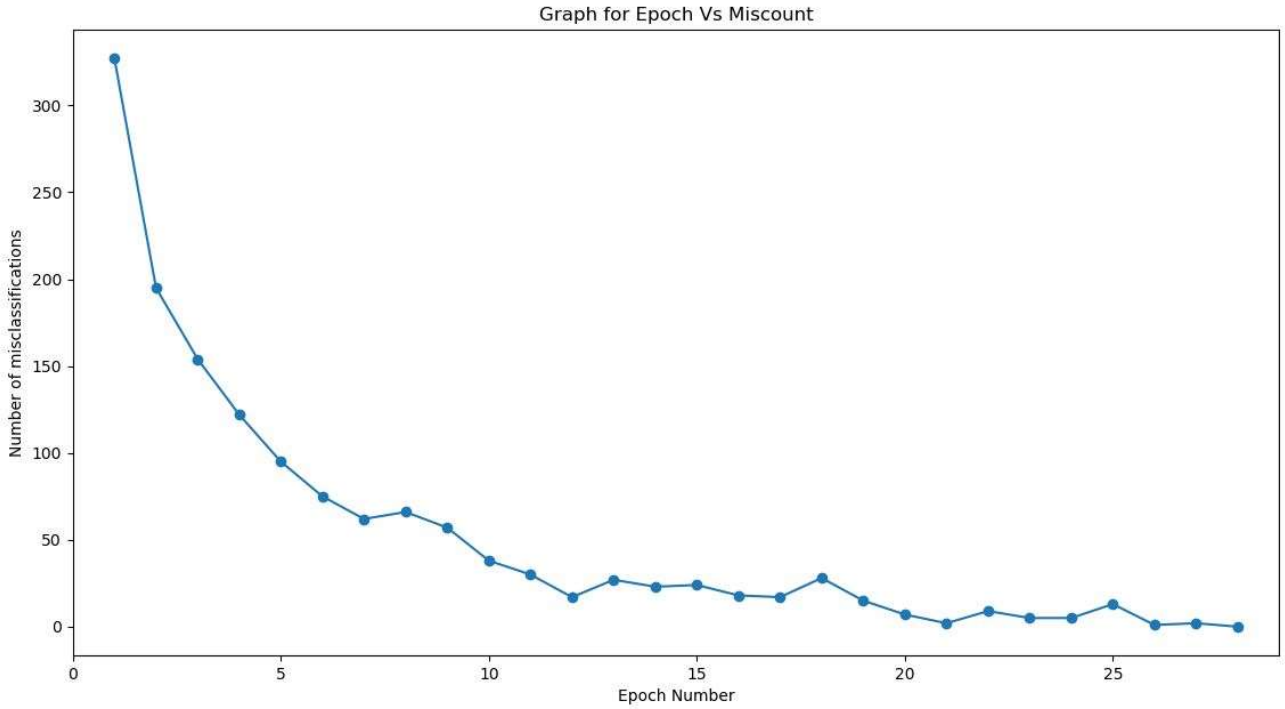


From above Two results, it is clear that as we are testing for more images from the same network that was trained, then number of test samples do make effect on error. Here in training sample we got error of zero but in testing we are having huge error rate that shows that our network is over trained for given training set. It happened because of training on very few elements. So when network exposed to more elements, it failed to give desired output. Though error percentage difference is very small, but as number of elements to be tested increase, error rate is also increases because of more variety in input.

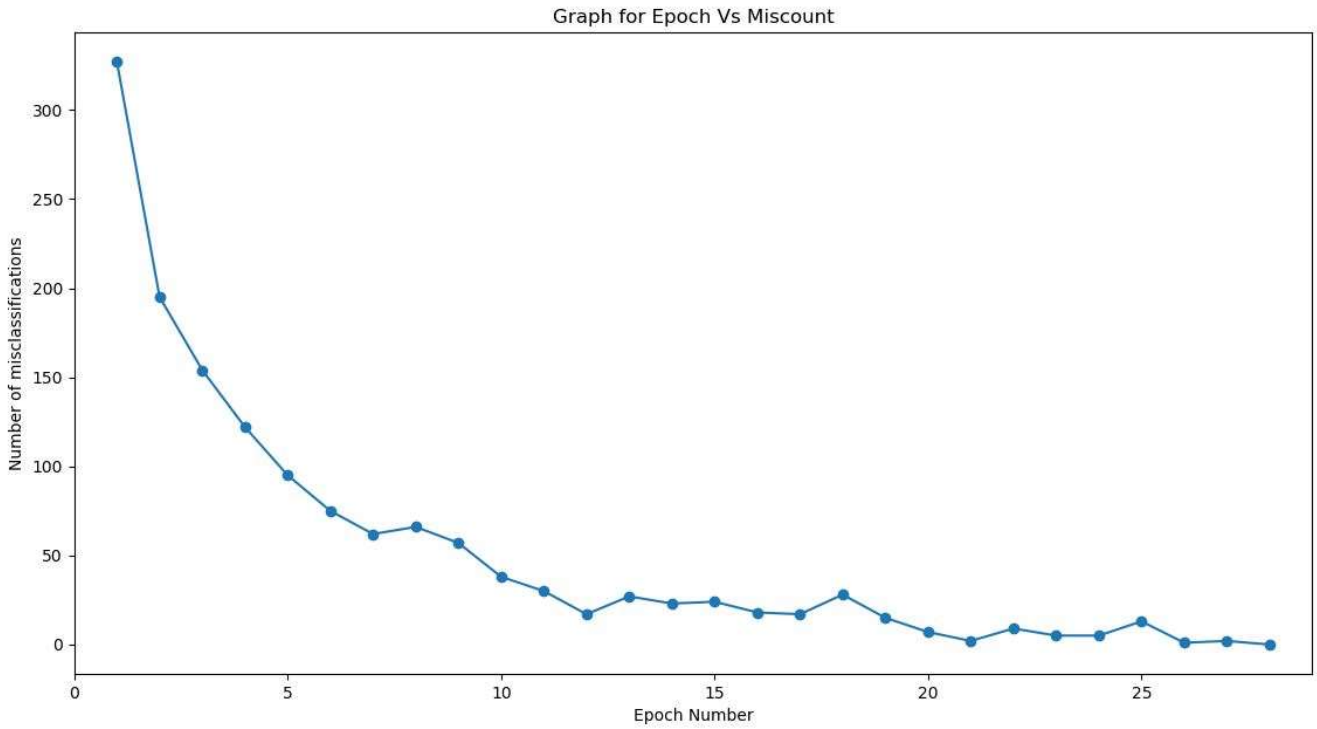
(g)

Now taking 1000 number of elements (Images) for training, learning rate 1 and epsilon equal to 0, Perceptron Training Algorithm took 28 epoch to converge and terminates with 0 error and 0 error percentage.

When testing algorithm is done with 50 elements and same weight obtained from above PTA, error percentage or percentage of misclassified samples obtained is 18.4%



Now, Testing Algorithm is done for 10000 labels and for the same weighs obtained from above and result for percentage of misclassification was 16.4% with error and percentage of error is 0 and number of epochs to converge the algorithm is 28.

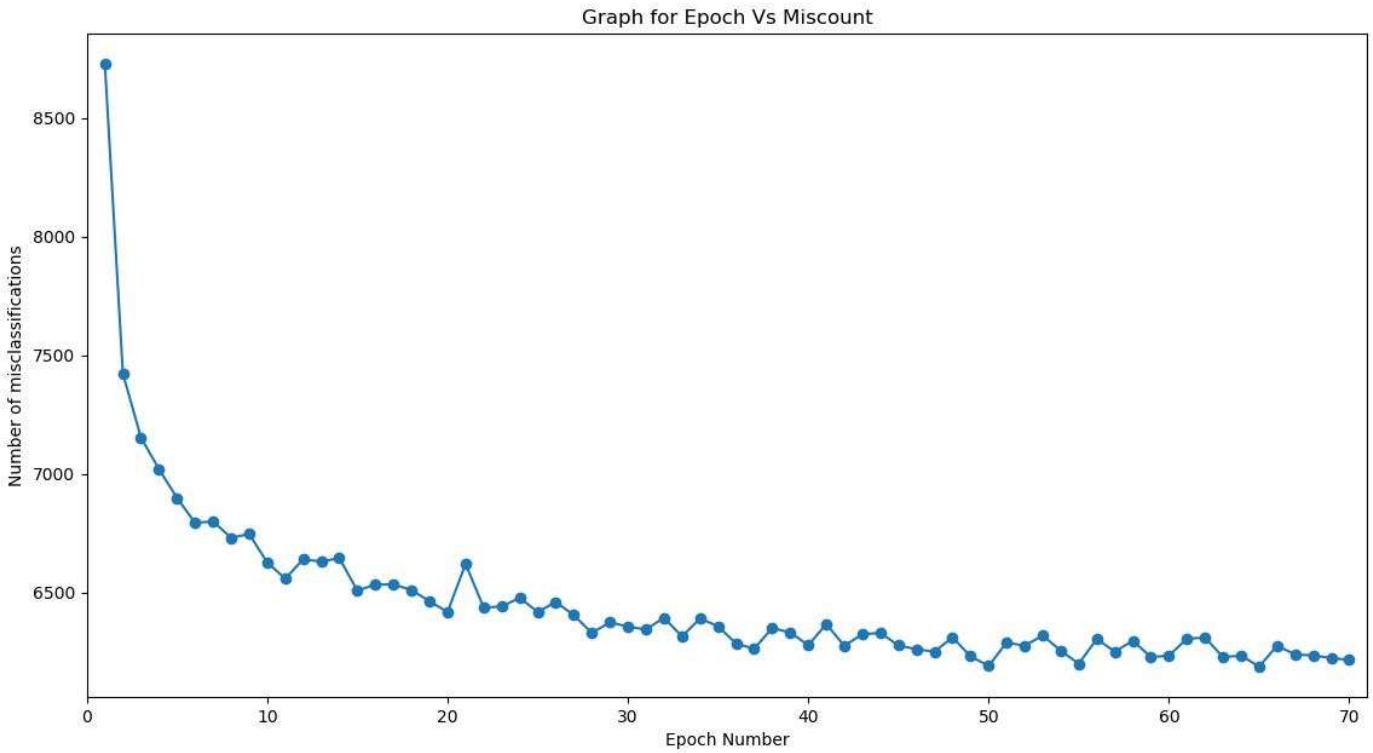


From above results, it can be clearly seen that as number of elements are increased for training of network and network has verity of inputs upon which it has to train, algorithm more converges and we get less number of misclassification with increment of test elements as compared to previous case and also error of misclassification reduces.

(h)

For Training of network was done with 60000 elements and testing the same network with 10000 elements and epsilon equal to 0 along with learning rate 1, Number of epoch taken by the network to converge and to make ratio of number of misclassifications and total number of elements below epsilon is 70 and error percentage is 10.39 and final error number is 6235.

And after performing testing operation, error percentage that obtained is 12.70%

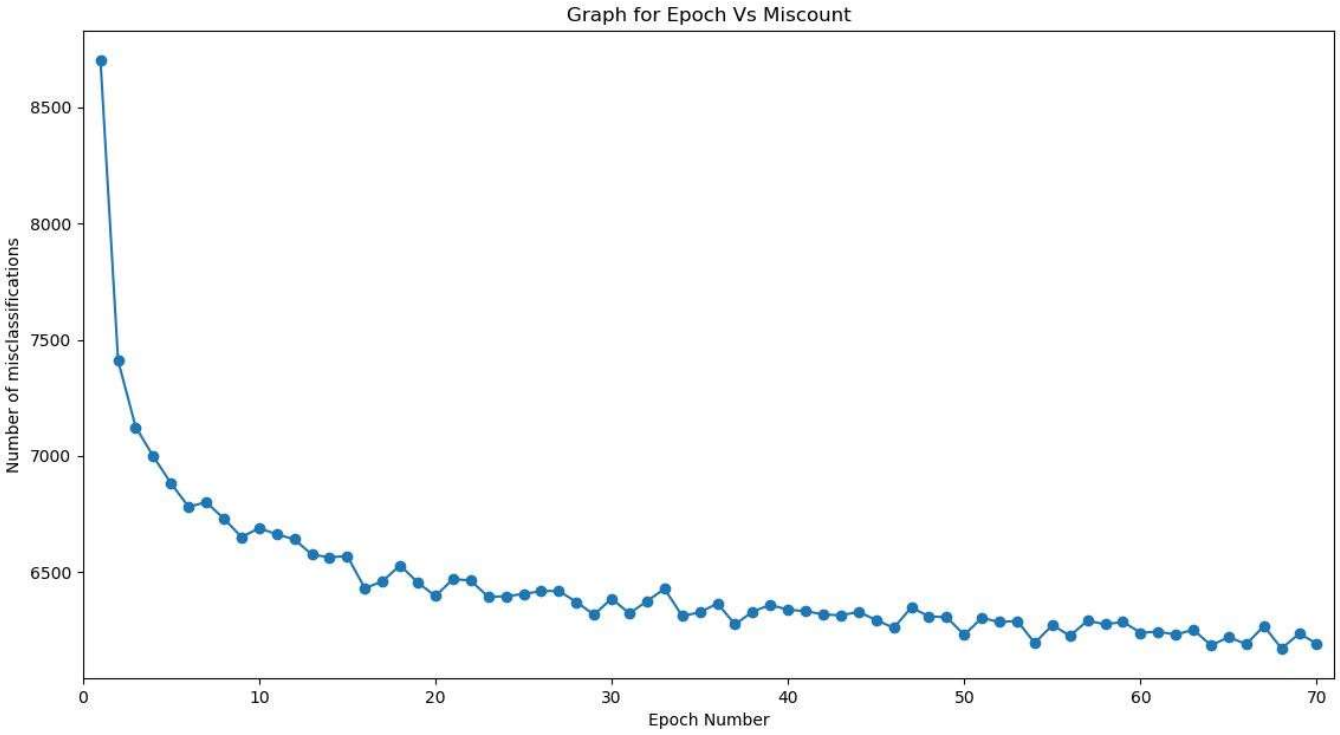


So we can clearly see from above result that with taking all parameters same, algorithm is trained with 60000 elements. So algorithm converged more properly as it has wide variety of inputs to train on. Therefore weights we got are more optimal to predict desired output. And the percentage of error that achieved is also less than previous experiments.

I(1)

Now Instead of taking epsilon equal to 0, first taking epsilon equal to 0.10 and total number of epoch taken to satisfy condition is 70. Error percentage during training phase is around 10.32 with 6193 total misclassifications.

If testing is done for 10000 elements with the same weights obtained above, error percentage is 11.18%.

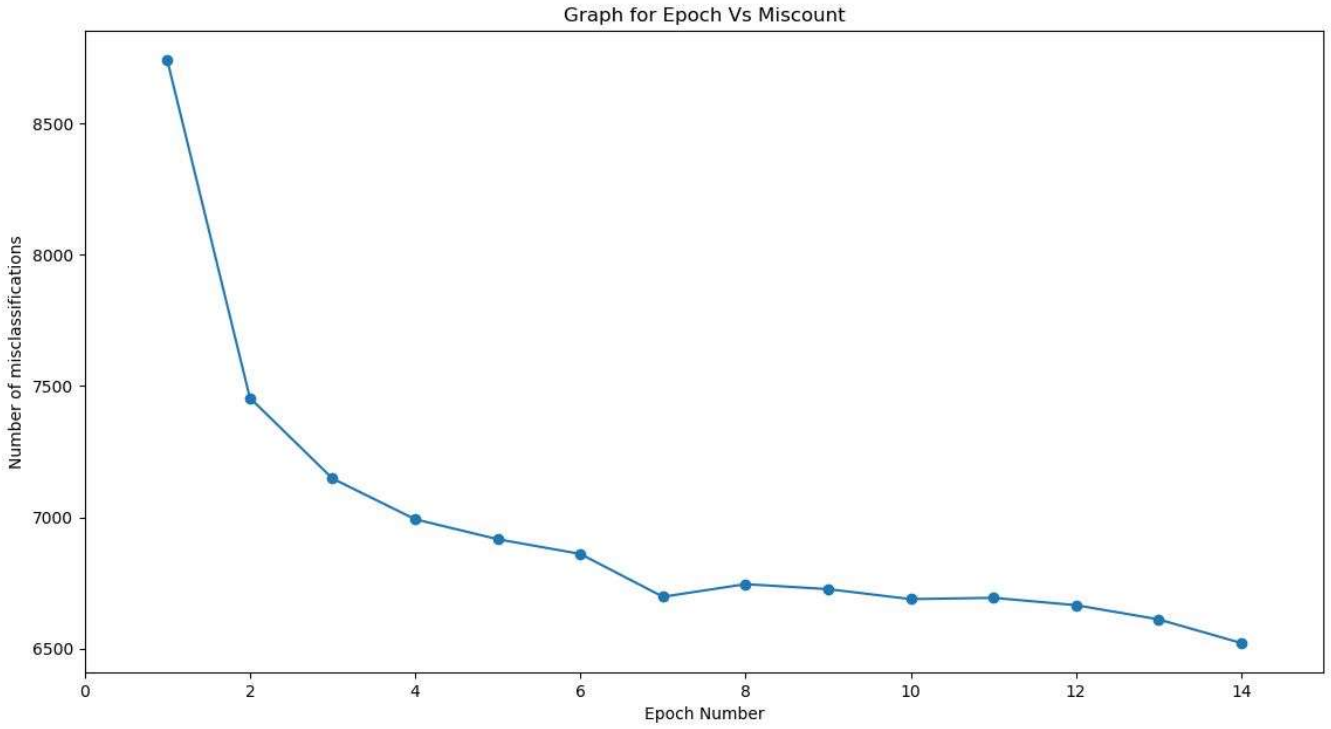


i(2)

Now taking epsilon equal to 0.11.and initializing new weights, total number of epoch taken to satisfy condition is 14. Error percentage during training phase is around 10.87 with 6522 total misclassifications.

If testing is done for 10000 elements with the same weights obtained above, error percentage is 15.18%.

It is clear that as we increase the ratio for error and number of misclassification, number of epoch taken to satisfy the condition drastically reduce. So by doing experiments it is noticed about the optimum ratio is around 0.10 and if ratio is increases the number of epoch drastically reduces. Also it can be seen that number of misclassifications are more means more error because algorithm converges too fast without proper training.

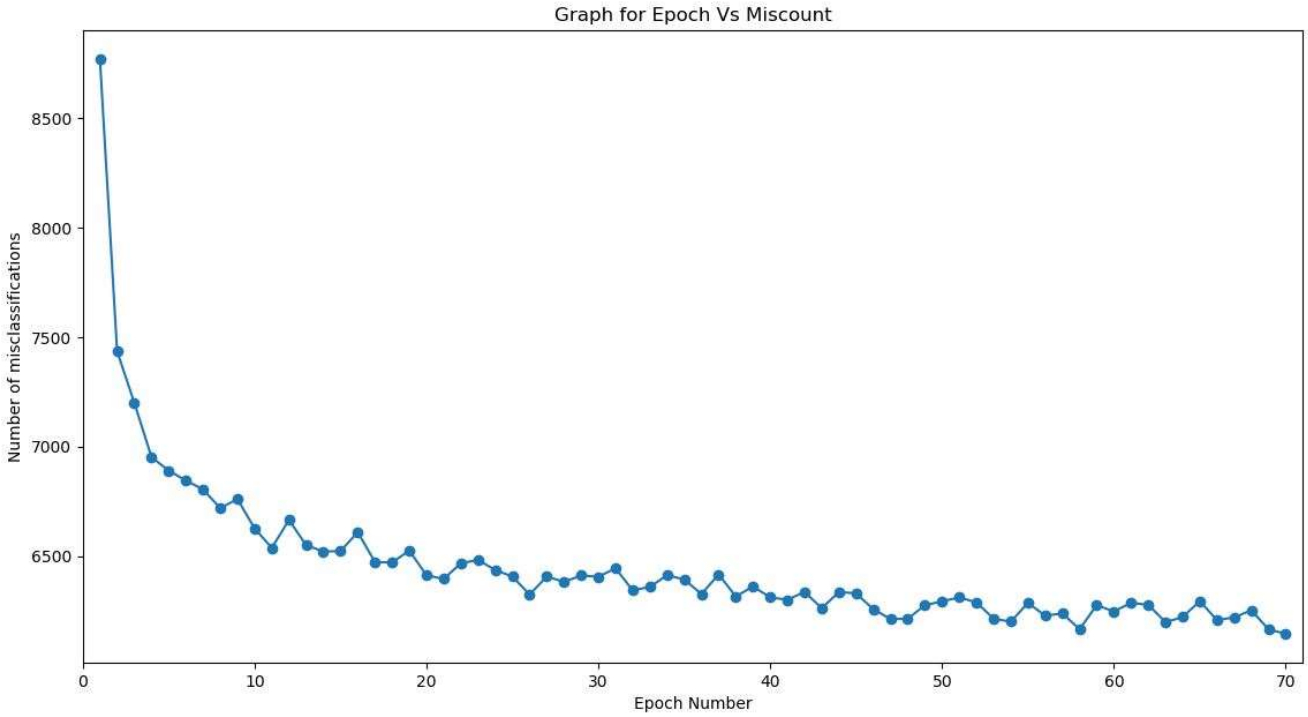


i(3)

Now taking epsilon equal to 0.9.and initializing new weights, total number of epoch taken to satisfy condition is 70. Error percentage during training phase is around 10.24 with 6146 total misclassifications.

If testing is done for 10000 elements with the same weights obtained above, error percentage is 12.18%.

So epsilon value makes huge impact on algorithm convergence as from result it can be seen that value is 0.10 where algorithm gives less number of misclassifications but as we increases epsilon, algorithm gives more number of misclassifications for same parameters.



SOURCE CODE:

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

#Weight Function for generating weights
def weightfunc():
    i_weight = np.zeros((10,784))
    for i in range(0,10):
        x = (1-(-1))*random.sample(784) - 1
        i_weight[i] = x
    weight_matrix=np.matrix(np.array(i_weight))
    return weight_matrix
```

```
#Function to train images for Multicategory TrainPTA
def TrainPTA(N1,eta,epsilon,weight_matrix):
    # Data read from the Label files
    label_data= open('train-labels-idx1-ubyte','rb')
    struct.unpack('>I',label_data.read(4))[0]
    struct.unpack('>I',label_data.read(4))[0]
    desired=[]
    for j in range(N1):
        desired.append(struct.unpack('>B',label_data.read(1))[0])
    flag=1
    error=0
    epoch=0
    epoch_arr=[]
    error_arr=[]
    while(flag==1):
        error=0
        img_data= open('train-images-idx3-ubyte','rb')
        img_data.read(4)
        img_data.read(4)
        img_data.read(4)
        img_data.read(4)
        for k in range(N1):
            x1=[]
            for l in range(784):
                x1.append(struct.unpack('>B',img_data.read(1))[0])
            matrix_inputs=np.matrix(np.array(x1))
            matrix_inputs_transpose=np.transpose(matrix_inputs)
            dot=np.dot(weight_matrix,matrix_inputs_transpose)
            actval=np.argmax(dot)
```

```
            new_mat=np.zeros((10,1))
            if(desired[k]!=actval):
                new_mat[actval]=-1
                new_mat[desired[k]]=1
                error=error+1
                weight_matrix=weight_matrix + np.dot((eta*new_mat),matrix_inputs)
        epoch+=1
        error_arr.append(error)
        epoch_arr.append(epoch)
        if(epoch==70): break
        print("Epoch =  ",epoch," Error = ",error)
        if((error/N1)>epsilon):
            flag=1
        else:
            flag=0
    error_percentage = (error/N1)*100
    print("Error Percentage = ",error_percentage,"%")
    print("Epoch=",epoch)
    print("Error=",error)
    plt.title("Graph for Epoch Vs Miscount")
    plt.xlim(0,epoch+1)
    plt.plot(epoch_arr,error_arr,'o-')
    plt.xlabel('Epoch Number')
    plt.ylabel('Number of misclassifications')
    plt.show()
    return(weight_matrix)

#Function to test images for Multicategory TrainPTA
def TestPTA(N2,weight_matrix):
    error_percentage=0
```

```
    label_data= open('t10k-labels-idx1-ubyte','rb')
    struct.unpack('>I',label_data.read(4))[0]#Reading first two bytes
    struct.unpack('>I',label_data.read(4))[0]
    desired=[]
    for j in range(N2):
        desired.append(struct.unpack('>B',label_data.read(1))[0])
    error=0
    img_data= open('t10k-images-idx3-ubyte','rb')
    img_data.read(4)
    img_data.read(4)
    img_data.read(4)
    img_data.read(4)
    for k in range(N2):
        x1=[]
        for l in range(0,784):
            x1.append(struct.unpack('>B',img_data.read(1))[0])
        matrix_inputs=np.matrix(np.array(x1))
        matrix_inputs_transpose=np.transpose(matrix_inputs)
        dot=np.dot(weight_matrix,matrix_inputs_transpose)
        actval=np.argmax(dot)
        new_mat=np.zeros((10,1))
        if(desired[k]!=actval):
            new_mat[actval]=-1
            new_mat[desired[k]]=1
            error=error+1
    error_percentage = (error/N2)*100
    print("Error Percentage = ",error_percentage,"%")

print("Executing all possible combinations")
print("Program Starts")
```

```
weight_glob=weightfunc()

print("Part f(1)")
weight_2=TrainPTA(50,1,0,weight_glob)
TestPTA(50,weight_2)

print("Part f(2)")
weight_3=TrainPTA(50,1,0,weight_glob)
TestPTA(10000,weight_3)

print("Part g(1)")
weight_4=TrainPTA(1000,1,0,weight_glob)
TestPTA(1000,weight_4)

print("Part g(2)")
weight_5=TrainPTA(1000,1,0,weight_glob)
TestPTA(10000,weight_5)

print("Part h")
weight_1=TrainPTA(60000,1,0.106,weight_glob)
TestPTA(10000,weight_1)

print("Part i(1) Set")
weight_glob1=weightfunc()
weight_6=TrainPTA(60000,1,0.10,weight_glob1)
TestPTA(10000,weight_6)

print("Part i(2) Set")
weight_glob2=weightfunc()
weight_7=TrainPTA(60000,1,0.11,weight_glob2)
```

```
TestPTA(10000,weight_7)


print("Part i(3) Set")
weight_glob3=weightfunc()
weight_8=TrainPTA(60000,1,0.09,weight_glob3)
TestPTA(10000,weight_8)
```