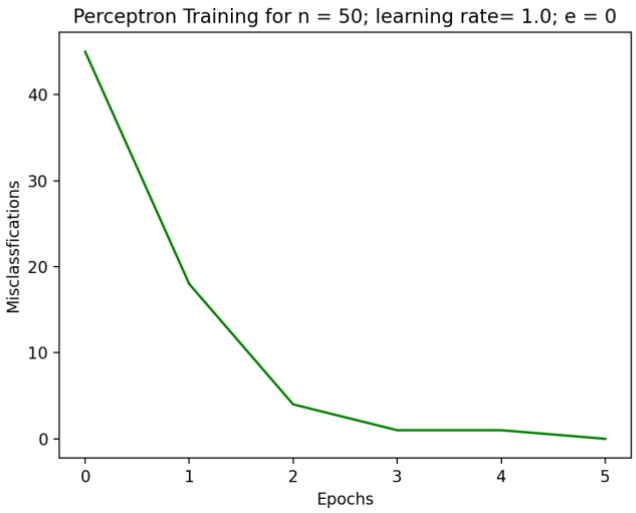
Name: Advait Pai
Email: apai21@uic.edu
UIN: 677368201

Homework 3 Report

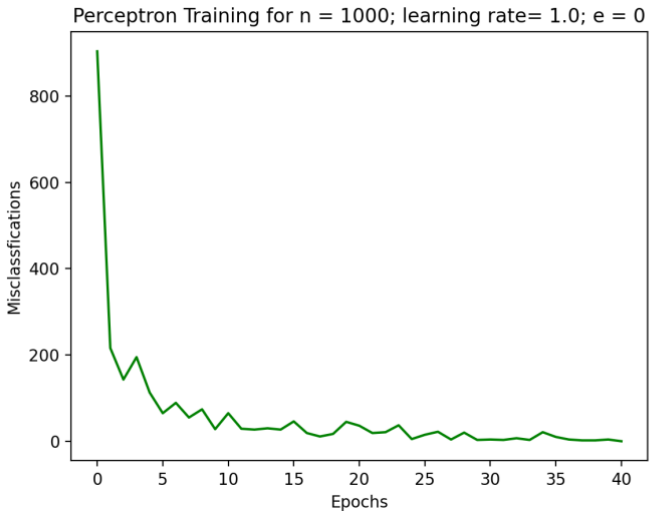
Perceptron Training Algorithm n = 50; learning rate = 1; e = 0



Misclassification for Test Set when n is 50 = 4476
Percentage of Misclassification = 44.76

Comments: At n = 50, the training converges at 5 epochs itself, which may make the weights seem correct but that is not the case. During testing, the percentage of misclassification is 44.76. For a small training set, since the training algorithm has not seen many samples, the percentage of misclassification is going to be high even though we get 0 errors during training. This model we have trained could be underfit as well as it does not have enough samples to generalise the weights over.

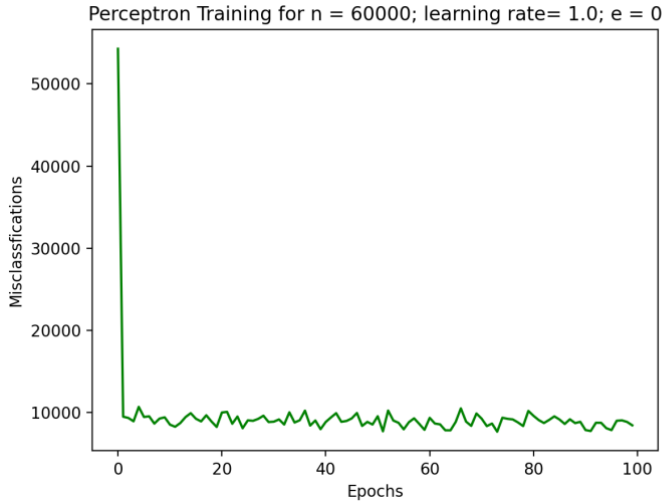
Perceptron Training Algorithm n = 1000; learning rate = 1; e = 0



Misclassification for Test Set when n is 1000 = 1699
Percentage of Misclassification = 16.99

Comments: At n = 1000, we see that the there is a significant reduction in the number of misclassifications in the Test Set, this could be due to increased generalisation of the weights during training on more samples.

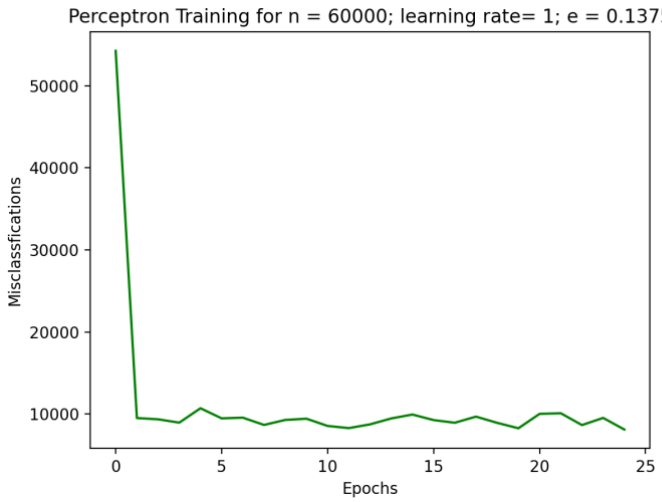
Perceptron Training Algorithm n = 60000; learning rate = 1; e = 0



Misclassification for Test Set when n is 60000 = 1609
Percentage of Misclassification = 16.09

Comments: In this case, since I have set a limit on the number of epochs = 100, and the problem does not converge, the weights of the final epoch (not global minimum) are chosen. When we see the graph, we can visually analyse that there is at least more than weight combination of weights which could lead to lesser number of misclassifications. We explore these options by setting the e threshold a little greater than 0 i.e. 0.1375.

Perceptron Training Algorithm n = 60000; learning rate = 1; e = 0.1375



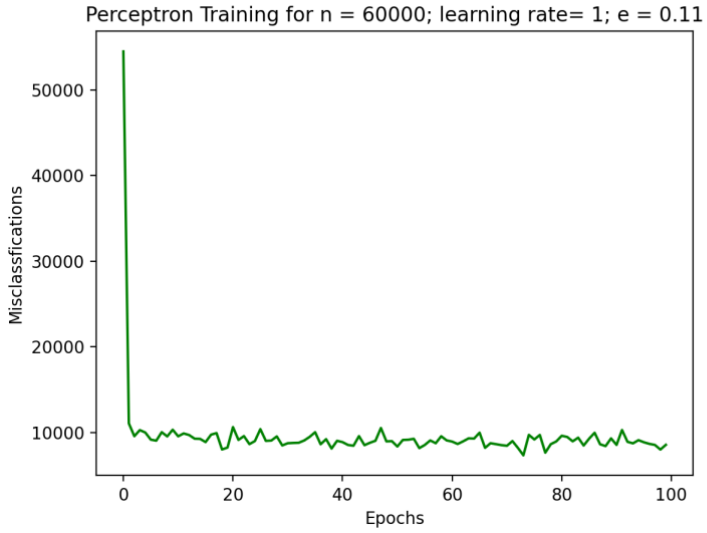
Misclassification for Test Set when n is 60000 = 1522
Percentage of Misclassification = 15.22

Comments: With the e changed to 0.1375, we notice that the misclassification percentage reduced to 15.22, which is lower than the previous output that we saw. The training algorithm stops at 24 epochs as well. Again, there may be better weights in the subsequent training steps, we ignore them due to the set limit.

We now change the weights using the following code:

```
np.random.seed(100)
W_new = np.random.uniform(-10,10,(10,784))
```

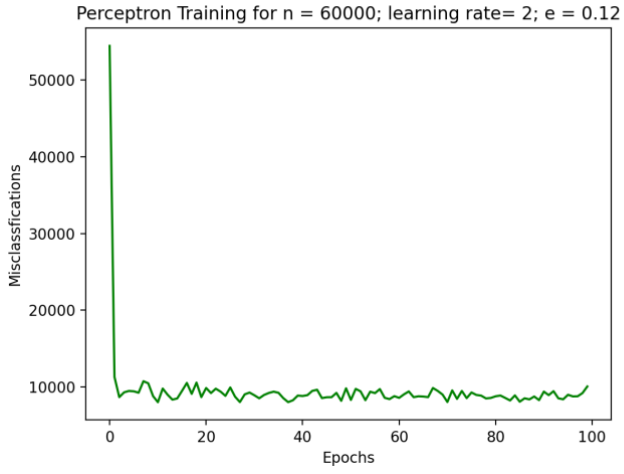
Perceptron Training Algorithm n = 60000; learning rate = 1; e = 0.11



Misclassification for Test Set when n is 60000 = 1464
Percentage of Misclassification = 14.64

Comments: With weights changed, we see that the there is a further reduction in the number of misclassifications but again since the epochs = 100, the algorithm has not converged. Due to this there could be more optimal weights that satisfy the problem.

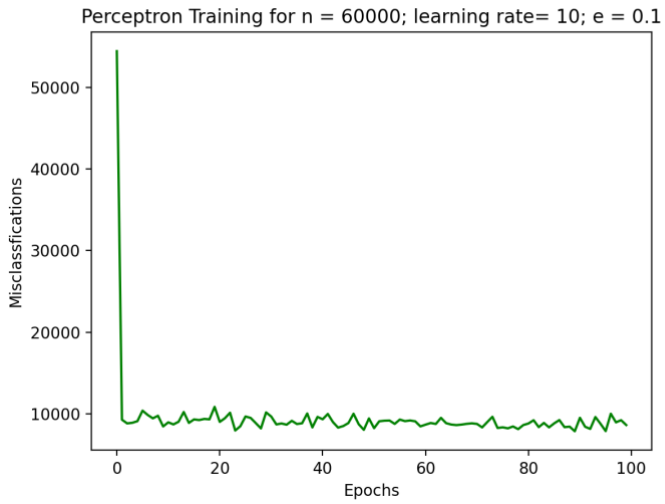
Perceptron Training Algorithm n = 60000; learning rate = 2; e = 0.12



Misclassification for Test Set when n is 60000 = 1352
Percentage of Misclassification = 13.52

Comments: Due to the number of epochs being 100, again the algorithm does not converge even with a slightly increased e threshold, and thus we could have had more optimal weights.

Perceptron Training Algorithm n = 60000; learning rate = 10; e = 0.1



Misclassification for Test Set when n is 60000 = 1448
Percentage of Misclassification = 14.48

Comments: Due to the number of epochs being 100, again the algorithm does not converge. Maybe trying with a higher e threshold may allow it to reach an optimal solution.

Closing Comments for new weights

After trying with different combinations for the new weights, we see that lowering the threshold does not allow convergence hence the new weights may not offer before performance as compared to the old weights.

Additionally, the algorithm implementation may itself be a limitation which causes the misclassification error percentage is around 13.75%. and we may not get a better solution than that.

Code

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

# Code to read input
def read_idx(filename):
    with open(filename, 'rb') as f:
        zero, data_type, dims = struct.unpack('>HBB', f.read(4))
        shape = tuple(struct.unpack('>I', f.read(4))[0] for d in range(dims))
        return np.fromstring(f.read(), dtype=np.uint8).reshape(shape)

train_X = read_idx('train-images-idx3-ubyte')
train_Y = read_idx('train-labels-idx1-ubyte')
test_X = read_idx('t10k-images-idx3-ubyte')
test_Y = read_idx('t10k-labels-idx1-ubyte')

# Helper Functions
def step_func(x1):
    for j in range(0,len(x1)):
        if x1[j] >= 0:
            x1[j] = 1
        else:
            x1[j] = 0
    return x1

def d_x(label):
    temp = np.zeros((10,1))
    temp[label]=1
    return temp

def get_label(v):
    return np.argmax(v)

# Perceptron Training Algorithm
def train_PTA(n,lr,e,W_init):
    epochs = 0
    errors = []
    flag = True
```

```
    while flag:
        error = 0
        for i in range(0,n):
            X = train_X[i].reshape(784,1)
            v = np.dot(W_init,X) # Induced Local Field
            label = get_label(v)
            if label != train_Y[i]:
                error+=1
        errors.append(error)
        epochs+=1
        for k in range(0,n): # Weight Update Stage
            X1 = train_X[k].reshape(784,1)
            v = np.dot(W_init,X1)
            W_init = W_init + lr * (d_x(train_Y[k]) - step_func(v)) * X1.T
        # print("Epoch No:",epochs-1,"Error:",error)
        if (errors[epochs-1]/n <= e) or (epochs>=100):
            flag = False
            break
    epochs_1 = [m for m in range(0,len(errors))]
    t = "Perceptron Training for n = "+str(n)+"; learning rate= "+str(lr)+"; e = "+str(e)
    plt.title(t)
    plt.xlabel("Epochs")
    plt.ylabel("Misclassfications")
    plt.plot(epochs_1, errors, color ="green")
    plt.show()
    return W_init

def test_PTA(W_calc,n):
    t_error = 0
    for i in range(0,len(test_X)):
        t_X = test_X[i].reshape(784,1)
        v = np.dot(W_calc,t_X) # Induced Local Field
        label = get_label(v)
        if label != test_Y[i]:
            t_error+=1
    print("Misclassification for Test Set when n is",n,"=",t_error)
    print("Percentage of Misclassification =",(t_error/10000*100))

np.random.seed(2702)
W = np.random.uniform(-1,1,(10,784))
```

```
test_PTA(train_PTA(50,1.0,0,W),50)
test_PTA(train_PTA(1000,1.0,0,W),1000)
test_PTA(train_PTA(60000,1.0,0,W),60000)

# Different E
test_PTA(train_PTA(60000,1,0.1375,W),60000)


np.random.seed(100)
W_new = np.random.uniform(-10,10,(10,784))
# 1st time
test_PTA(train_PTA(60000,1,0.11,W_new),60000)
# 2n time
test_PTA(train_PTA(60000,2,0.12,W_new),60000)
# 3rd time
test_PTA(train_PTA(60000,10,0.1,W_new),60000)
```