CS 559 Neural Networks HW #3 Report

Author: Kai Bonsol

I. Description

Computer program uses the multicategory perceptron training algorithm to classify images of digits from the MNIST dataset.

The algorithm was implemented and ran several times on the training dataset with different parameters:

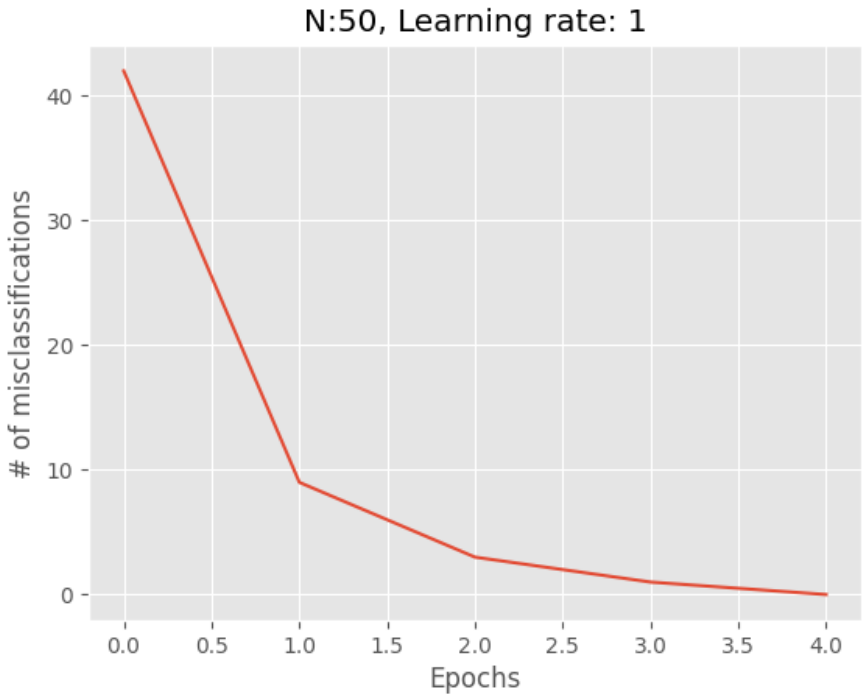
- N, the number of samples out of the dataset (max 60,000) to use
- η, the learning rate.
- ε, the threshold with which to accept convergence

The resulting weight matrix is then ran against the test set to evaluate the multicategory PTA model with the following parameters.

II. Results

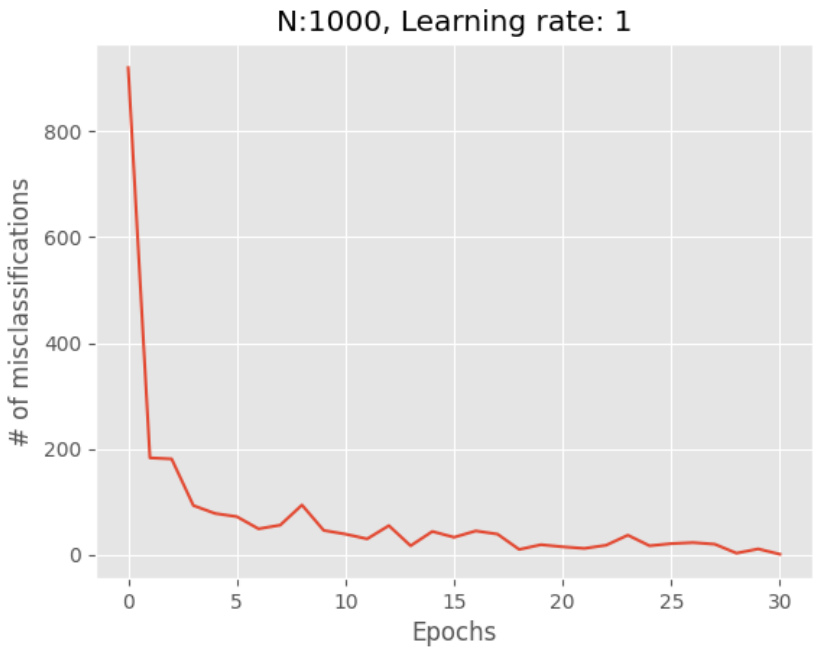
For N = 50, η = 1, and ε = 1e-3:

Graphing epochs to # of misclassifications we see convergence to 0 misclassifications.



Computing misclassifications on the test set resulted in an error percentage of 42.96%. The discrepancy in the training results and the test results is from overfitting the training dataset, since N is small relative to the entire dataset the neural network parameterized by the resulting weight vector is not a good model of the data.

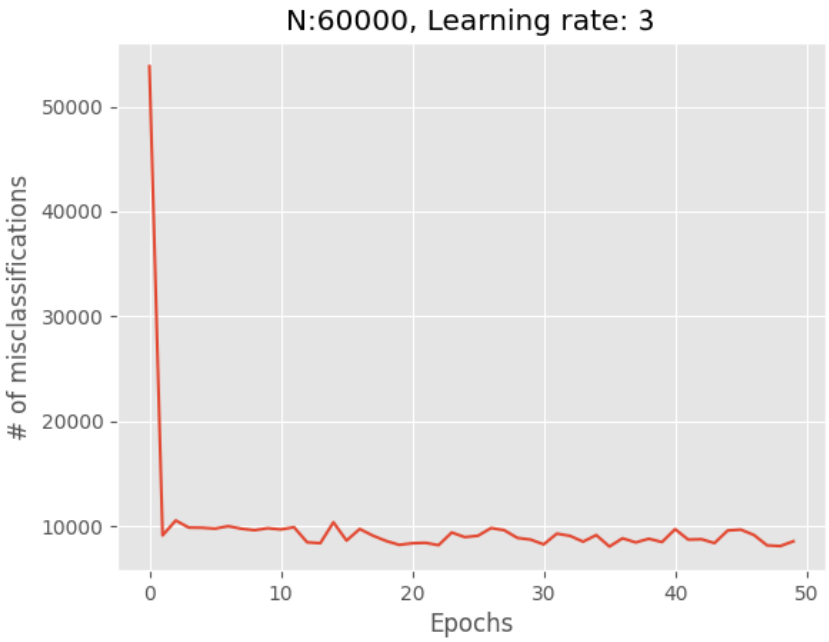
For N = 1000, η = 1, and ε = 1e-3:



Test error rate: 17.88%

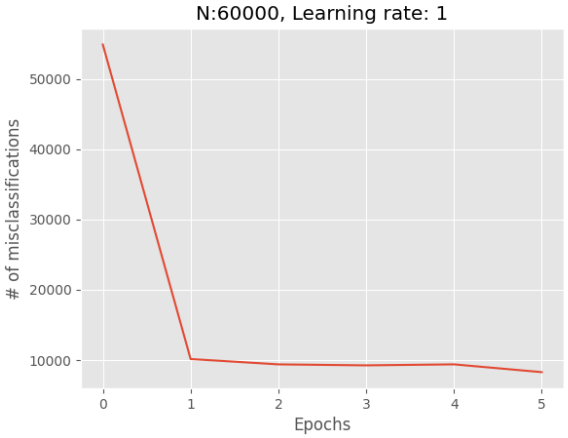
The discrepancy between 0 misclassifications training data and 17.88% error rate on the testset is from the same reason for the N = 50 case. N is small relative to the entire dataset and the neural network is overfit on the training data.

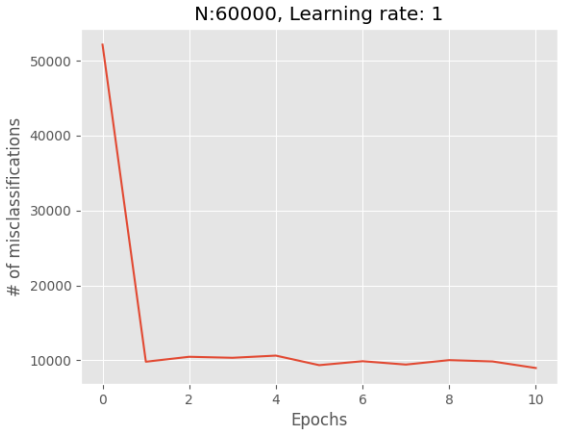
Next, I analyzed N = 60,000, η = 1, and ε = 0:

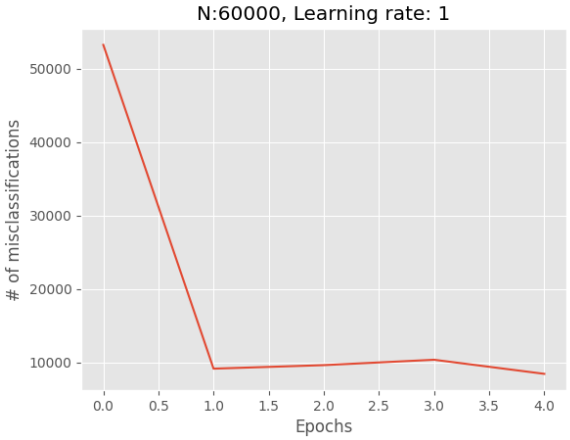


Here the multicategory PTA does not seem to converge after many epochs, hovering around approximately 9000, we can compute an appropriate ε to achieve convergence as 9,000/60,000 = 0.15.

With this new epsilon, we repeat the multicategory PTA several times with different initial weight matrices:







The misclassification rate computed on the test set for each (read left-right, top-bottom) was: 15.36%, 15.22%, and 16.78%; all much closer to the training misclassification rate, meaning the larger dataset partialled out the discrepancy caused by overfitting the data. The different initial weight matrices seem to vary in convergence time but not significantly, and all initial W's seem to misclassify every training example.

III. Code

```
#
# CS 559 Neural Networks HW3
# Author: Kai Bonsol
#

#
# Description:
# Program which uses multicategory PTA for digit classification.
#

import numpy as np
import matplotlib.pyplot as plt
import torch
from torchvision import datasets

plt.style.use('ggplot')

# componentwise_step
# params: list type
def componentwise_step(x):
    def step(y):
        if y >= 0:
            return 1
        return 0
    ret = []
    for xi in x:
        ret.append(step(xi))
    return np.asarray(ret)

# multicategory perceptron training algorithm
# params: trainset - from torchvision.datasets.MNIST
#         learning rate, threshold, n - sample size
def multicategory_PTA(trainset, learning_rate, threshold, n, limit=None):

    W = torch.rand(10, 784)
    epoch = 0
    errors = {}

    while True:
        errors[epoch] = 0
        # count misclassification errors
        for i in range(1, n):
            # calculate the induced local fields with
            # current training sample and weights:
            img, label = trainset[i]
            xi = np.array(list(img.getdata()))
            v = np.matmul(W, xi)

            # choose the output neuron with the largest induced local field
            # if the neuron output the incorrect digit, increase error for
this epoch.

            j = np.argmax(v)
            if j != label:
                errors[epoch] += 1
```

```
        epoch += 1
        # update the weights
        for i in range(1, n):
            img, label = trainset[i]

            # data, induced local field, desired output, output
            xi = np.array([list(img.getdata())])
            v = np.matmul(W, xi.T)
            dx = np.zeros((10,))
            dx[label] = 1
            fx = np.array([dx - componentwise_step(v)]).T

            learning_signal = np.matmul(fx,xi)
            W = W + learning_rate * learning_signal

        if errors[epoch-1]/n <= threshold:
            break
        if limit != None:
            if epoch >= limit:
                break
            print(str(epoch) + "/" + str(limit))
    return W, errors

def test_misclassification(testset, W, n):

    errors = 0
    for i in range(1, n):
        img, label = testset[i]
        xi = np.array(list(img.getdata()))
        v = np.matmul(W, xi)

        j = np.argmax(v)
        if j != label:
            errors += 1

    return errors

def graph_misclassification(errors, n , learning_rate, title):
    fig, ax = plt.subplots()
    ax.set_xlabel("Epochs")
    ax.set_ylabel("# of misclassifications")
    ax.set_title("N:" + str(n) + ", Learning rate: " + str(learning_rate))
    ax.plot(errors.keys(),  errors.values())
    fig.savefig("hw3_" + title + ".png")
    fig.show()

# (a, b)
mnist_trainset = datasets.MNIST(root='./data', train=True,
                                        download=False, transform=None)

mnist_testset = datasets.MNIST(root='./data', train=False,
                                        download=False, transform=None)


W, errors = multicategory_PTA(mnist_trainset, n=50, learning_rate=1,
threshold=1e-3)
```

```
test_errors = test_misclassification(mnist_testset, W, 10000)

graph_misclassification(errors, 50, 1, "N50L1")
print("Test misclassification rate: " + str(test_errors/10000))

# likely different because of small n, not representative enough of large
dataset.

W, errors = multicategory_PTA(mnist_trainset, n=1000, learning_rate=1,
threshold=1e-3)
test_errors = test_misclassification(mnist_testset, W, 10000)

graph_misclassification(errors, 1000, 1, "N1000L1")
print("Test misclassification rate: " + str(test_errors/10000))

# differences still from proportionally small n.

# (h)

W, errors = multicategory_PTA(mnist_trainset, n=60000, learning_rate=1,
threshold=0, limit=50)

graph_misclassification(errors, 60000, 3)

# (i)

W, errors = multicategory_PTA(mnist_trainset, n=60000, learning_rate=1,
threshold=0.15)
test_errors = test_misclassification(mnist_testset, W, 10000)

graph_misclassification(errors, 60000, 1, "i1")
print("Test misclassification rate: " + str(test_errors/10000))

W, errors = multicategory_PTA(mnist_trainset, n=60000, learning_rate=1,
threshold=0.15)
test_errors = test_misclassification(mnist_testset, W, 10000)

graph_misclassification(errors, 60000, 1, "i2")
print("Test misclassification rate: " + str(test_errors/10000))

W, errors = multicategory_PTA(mnist_trainset, n=60000, learning_rate=1,
threshold=0.15)
test_errors = test_misclassification(mnist_testset, W, 10000)

graph_misclassification(errors, 60000, 1, "i3")
print("Test misclassification rate: " + str(test_errors/10000))
```