# Multilayer Perceptron

Aaron Barnett
acba242@uky.edu

Abstract

This paper covers the classification of the handwritten digits zero and one by a multilayer perceptron. The data used is a subset from the MNIST handwritten digit dataset.

Beginning with implementation, the project has one perceptron class. The perceptron class is initialized with four matrices of weights: two for the layer weights and two for the bias weights. Weights are chosen uniformly from negative one to positive one. The perceptron class also has an alpha learning rate value and an epochs value in the initialization function. The class implements four functions: train, forward prop, activate layer, and accuracy. Train performs back propagation over output and hidden layers and updates weights accordingly over all examples passed in. Forward prop runs the network given one example. Activate layer uses sigmoid to return activation values for all elements in a given layer. Accuracy returns the percentage of correctly identified examples in a given dataset. The perceptron has three layers: an input layer with 784 nodes, a hidden layer with two nodes, and an output layer with one node. Output is rounded to the nearest integer for prediction. Numpy was the only library used for matrix operations.

## Accuracy

The perceptron was trained on 12,665 examples and tested on 2,115. An alpha learning rate of 0.5 was used for training over one epoch. For the given test set, the perceptron was able to reach an accuracy of 99.8%.

## Shortcomings

If given more time, I would implement a separate class for layers so that architectures of different sizes can be tested quickly. As the project stands, layers are hardcoded and all functions are written for the implemented architecture only. I would generalize these functions to allow for testing of different problems. I would also move the weight update that takes place in the train function so that deltas are calculated for an entire dataset and weights are only updated once. I think this would speed up the network training immensely as currently all weights are looped through for every example.