Machine Learning for Image Classification Milestone 1

Sonia Sharapova
sharapova@uchicago.edu

Performances:

Python implementation:

In the Python implementation, the accuracies reached an 87.75% accuracy on the test data. This model included all of the necessary computations for the forward pass, error calculations, gradient calculations, and backpropagation. The MNIST loading code was taken from Git Hub (cited in code), and the model was implemented by me. This was used as a reference for the structure of the network.

Training:
The training of the dataset consisted of a forward pass through the network, calculating loss and accuracy, and backpropagation for gradient computation to update the weights.

C implementation:

I build my model in C, utilizing matrix-matrix multiplication. The structure of the Neural Network used a tracking of previous (bwd) and next (fwd) layers to point to the next layers in the system (or previous for backtracking).

In this implementation, the accuracies reached 0.8112, or 81.12%, over 10 epochs when checked across the validation set.

In the C implementation, I unfortunately was unable to implement the proper calculation of the biases as I had some bug issues I did not resolve in time.

Command line arguments:
- nl: number of layers in the neural network (excluding the first and last layers)
- nh: size on hidden layer
- ne: number of training epochs
- nb: number of training samples per batch
- alpha: learning rate

In my execution:
- nl: 2
- nh: 10
- ne: 10
- nb: 3000
- alpha: 0.3 -> learning rate.

Success rates:

The accuracy was around 57% when the hidden layers were of size 10, but reached 0.8112, or 81%, over the 10 epochs when the dimensions on the hidden layers were increased to 28x28 (the same size as the input layer).

The average running time per epoch was 28.92 seconds, and the total execution time was 232.94 seconds.

Per epoch, the validation step took 0.067 seconds.

The learning rate was set to 0.3 and the model performed worse when the learning rate was smaller (such as 0.1).

The number of images per batch was the training size divided by the number of samples per batch.