Report

Implementation

We modeled our network as a set of weights in an array (or multiple arrays for hidden layers).

Next, we created a method for type of network that takes in the weights along with a point, label, and the activation function(s) handle(s), and returns a prediction.

From there we implemented functions to perform back propagation for both types of networks, which takes as arguments the network's weights, a “minibatch” of points and labels, and function handles for the activation function(s), the loss function, and their derivatives. The program makes use of the prediction function described above to have the network classify a point. Then it uses the prediction and the true label (along with the function handles) to do back propagation (regardless of hidden layers) and compute the gradient of the loss with respect to each weight. It averages the gradients and return them.

We created 2 functions that would train each kind of network for a specified number of epochs. For each epoch they divide the data set randomly, and then iterate through it. For each “minibatch” of points, the functions compute the loss gradient with respect to the weights for each by “minibatch” by using the back propagation functions described above, and perform a gradient descent update with a specified step size function.

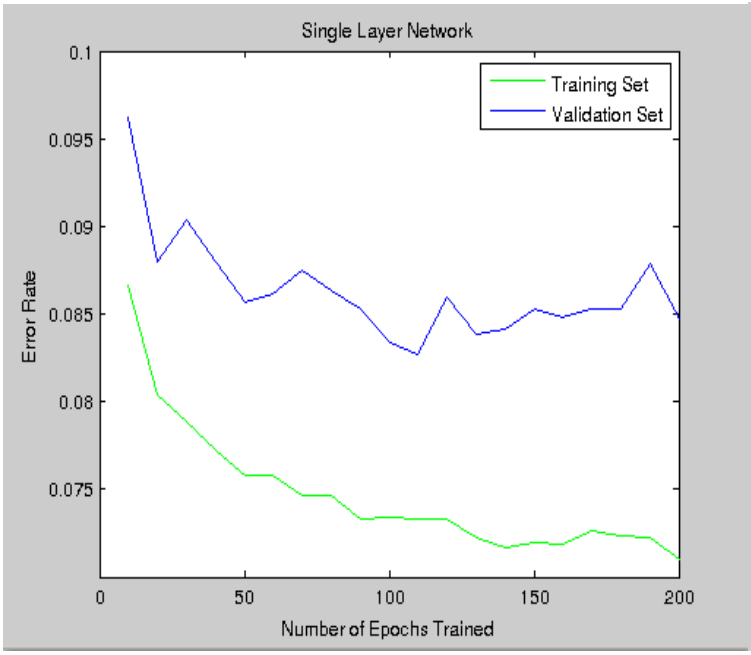
We created 2 functions to test networks. Taking in a network's weights, test points, and the proper function handles, these functions iterate through the points and use the prediction function above, recording both the accumulated loss and the average error.

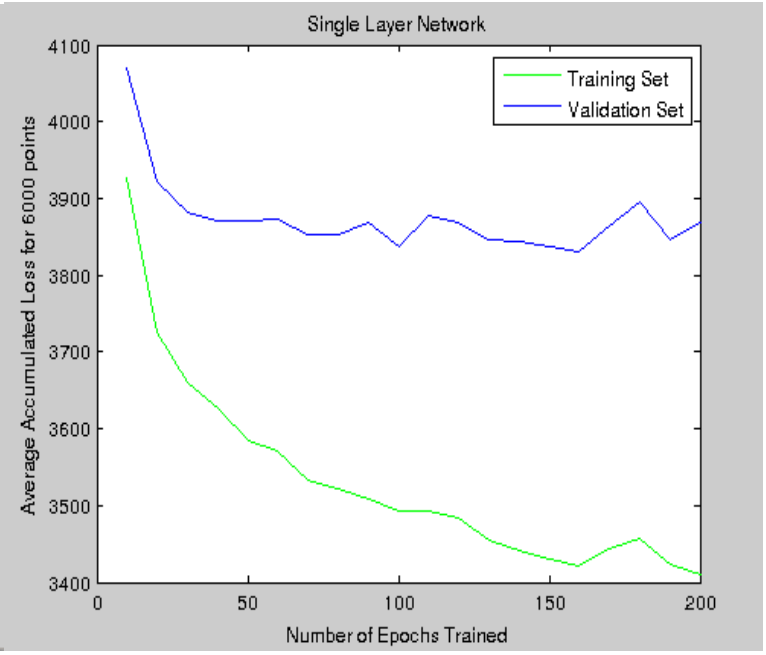
Lastly, for each network type we created a function that normalizes input data, initializes weights for a network, and trains it for the specified number of epochs, pausing at a specified frequency to test the network and write the results to a file. This is the only file that should need to be used.

Results

For each network, we used cross entropy to estimate loss.

This single layer network took about 4 hours to train on 60K data points. It reached an error rate of about 8.6% on the validation set, not too bad. The report frequency was 10 epochs.
Here are graphs of error rates and average sum of loss per 6000 points for the training and validation sets.





The network with 2 hidden layers took about 15 hours to train on 60K data points. It reached an error rate of about 2.2%!!!! on the validation set. Here are graphs of the error rates and average cumulative loss per 6000 points. The report frequency was 1 epoch.