Edward Venator

# EECS 484 Assignment 4a: Feedforward Neural Networks with Error Back Propagation

## Basic Methodology

I began by defining the derivative equations for the weight matrices Wji and Wkj.my derivative computation function is implemented as a nested for loop. It would run faster if I had implemented it using matrix operations, but since this code is not meant to be used in a production environment, I decided that the for-loop implementation was adequate.

My equations for the derivatives of the weight matrices are as follows:

Since both layers have a logistic sigmoid activation function, both have the same derivative:

Once this function was written, I made some minor tweaks to the code and collected data.

I then moved on to learning the MATLAB neural network toolbox. First, I updated the code to use the new feedforwardnet() constructor. I noticed that by default, Matlab reserves some of the training data for testing and validation. This causes problems with our training data because it is the minimal definition of our function. By setting net.divideFcn to ‘dividetrain,’ I forced Matlab to use all of the data for training. After making this change, I found that Matlab’s results were very similar to the results of the hand-written code, although the Matlab toolbox trained much faster. When the training function was set to ‘traingd’ (gradient descent), the results became even more similar, but the training took longer.

## Findings and Conclusions

### Varying Interneuron Numbers

### Varying Learning Rate

### The Matlab Neural Net Toolbox