20100093
Assignment 2 report:

My average accuracy on the test data is around 92 percent.
Initially I declared two matrices of 784 x 30 and 30 x 10 and randomly generated values between -1 and 1. These matrices served as weights from input to hidden and hidden to output layer weights respectively.
I forward propagated these values to get an output layer matrix of 1 x 10.
Passing this through sigmoid gave me the output matrix of my network, in other words this output layer is a prediction of the number my network thinks it's been fed. After forward propagation, we calculate the errors in each layer. We move backwards from output to input. Firstly, I calculated the error for the weights from hidden to output. The formula was calculated by taking derivative of the cost function with respect to weights. We calculate the difference between the output vector and target vector and use it to compute errors in the layers.
The input vector is normalized to avoid overflow in the sigmoid function. Input vector is assigned values from 0 to 1 by dividing by 255.
After errors are calculated, it is multiplied by the learning rate and added into the old weights. The same process is extended to weights from input to hidden; the old weights are updated. This is repeated for the size of the training data.
I calculated the error using the cross entropy function and IF the error is smaller than 0.0000001, the loop is broken, to avoid over-training of the network. The entire network is trained for 2 epochs.
The average weight value lies in the range of $-0.5<w<0.5$
The trained matrices are written to file by using the savetxt function in numpy.
Execution time is calculated by the time library.
Sys is used to parse command line arguments.
The graphs show that 0.03 is the optimal learning rate as the data reaches a higher accuracy much faster.