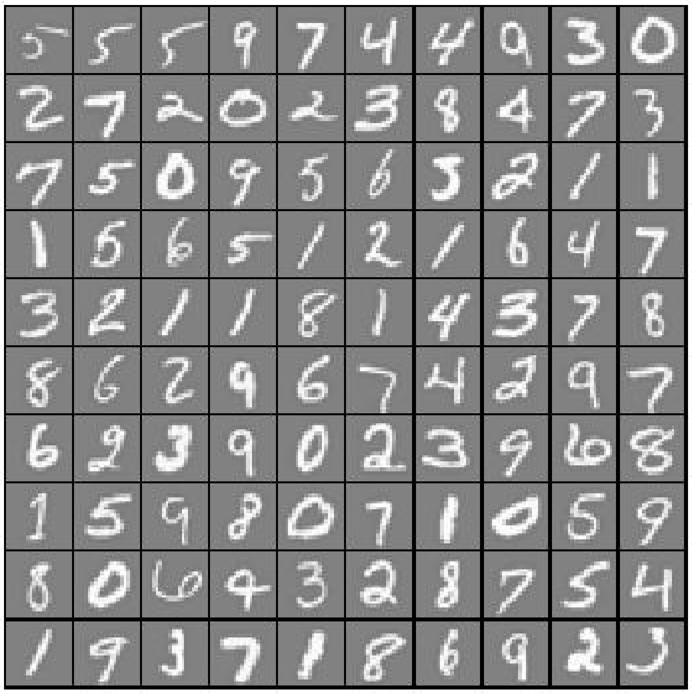
Nate Zimmerman
Elise Russell
11/30/15

Neural Networks and Fuzzy Systems: Project 5 Report

Project Overview

We followed the instructions given with the sample code to edit the scripts in the Neural Networks Part 1 folder and the scripts in the Neural Networks Part 2 folder. The Neural Networks Part 1 scripts are concerned with the implementation of logistic regression and cost function, while the Neural Networks Part 2 scripts are concerned with the implementation and training of an actual network using forward propagation and backpropagation on the given data.

The given data consisted of 5000 samples of handwritten numerals, in 20x20 pixel grayscale images, and each is assigned a class from 0 - 9 corresponding to the numeral represented in the picture. Below is an example of some of the inputs which were to be identified:



Neural Networks Part 1

For this section of the assignment, there were two stages. In the first, we had to modify the following scripts:

- lrCostFunction.m, to implement the logistic regression cost function
- oneVsAll.m, to train a set of one-vs-all logistic regression classifiers on the given data

- predictOneVsAll.m, to use a set of one-vs-all logistic regression classifiers to predict the classes of a given data set

In the second stage, we had to modify the following script:

- predict.m, to use a trained neural network represented as a pair of Theta matrices to generate predictions of the classes of a given data set using forward propagation.

Since a previous assignment required us to implement logistic regression, we were readily able to port the sections of code that we needed and adapt them to these scripts. Additionally, the forward-propagation algorithm turned out to be simple to implement in a vectorized manner; the additions we made consist of five lines of code successively multiplying matrices together and applying the sigmoid function.

Results of Neural Networks Part 1

The logistic regression master script for the first stage of this part of the homework was called ex3.m. When run, it first showed a grid of some of the data images in order to visualize the data, then it trained a set of logistic regression classifiers corresponding to each class. It then used predictOneVsAll.m to get the accuracy of the set of classifiers on the training data. Our implementation accuracy was 94.48%.

The forward-propagation master script for the second stage of this part of the homework was called ex3_nn.m. When run, it first loaded the already-trained theta parameters, then calculated the training set accuracy of the forward-propagation algorithm using predict.m. This implementation used a pre-calculated neural network with pre-loaded, fixed theta weights. The implementation accuracy was 97.52%. It then displayed sample images and the predictions that the system assigned them, giving the user a visual idea of the network's performance.

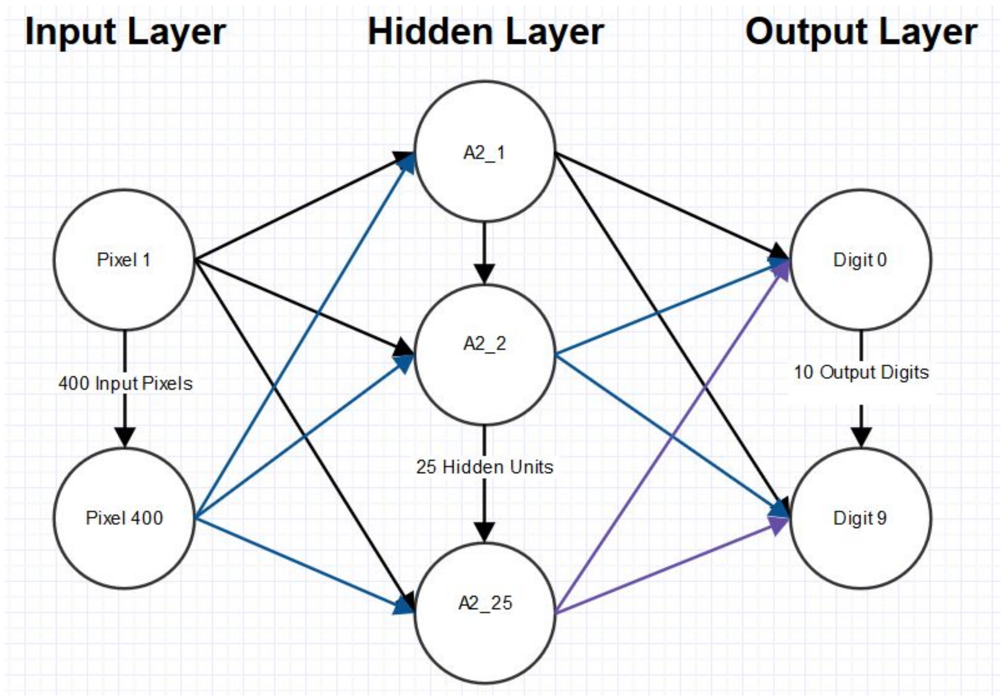
Neural Networks Part 2

For this section of the assignment, we had to modify the following scripts:

- sigmoidGradient.m, to implement a derivative of the sigmoid function at the given vector of inputs
- randInitialWeights.m, to initialize a network's theta matrices with random weights
- nnCostFunction.m, to implement forward-propagation and backpropagation using the given data and theta matrices, then calculate the overall cost function and the gradient matrices for the network on that data.

The first two scripts were relatively easy to implement, since they are straightforward demonstrations of simple concepts. The nnCostFunction script contained the meat of the project, and we implemented it in two stages. First, we figured out how to implement forward-propagation and backpropagation, while calculating the necessary values, on each training sample separately in a **for** loop. Then, we vectorized this implementation and discarded the **for** loop so that the script was cleaner and would run faster.

Below is an example diagram of the neural network:



As mentioned previously, there were 400 input pixels to the neural network. There were 25 hidden units which had sigmoid activation functions. Finally, the neural network had 10 logistical outputs ranging from numerals 0 to 9 to be identified by the network.

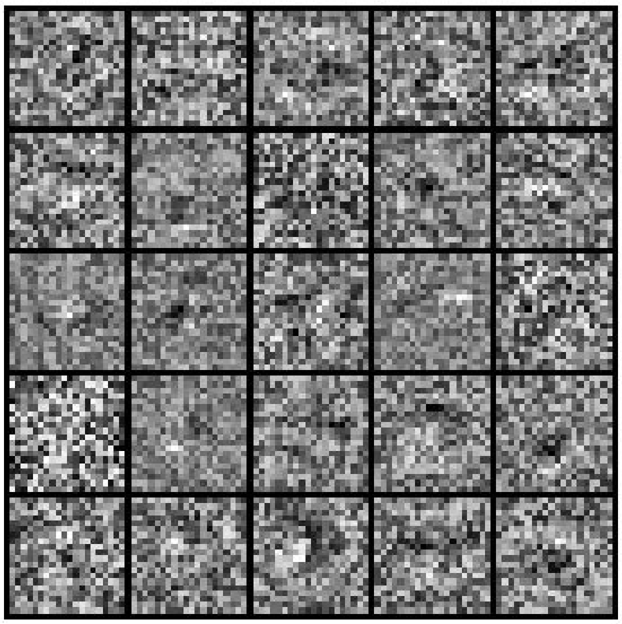
Results of Neural Networks Part 2

The master script for this section of the assignment was called ex4.m. It first runs through several checks to determine the accuracy of various parts of the implementation, with and without regularization. Our implementation passed these checks as follows:

- Feedforward Using Neural Network… Cost at pre-loaded parameters (should be about 0.287629): 0.287629.
- Checking Cost Function (w/ Regularization) … Cost at pre-loaded parameters (should be about 0.383770): 0.383770.
- Evaluating sigmoid gradient at [1 -0.5 0 0.5 1]: [3.586161 2.755252 2.500000 2.755252 3.586161]
- Checking backpropagation… relative difference (should be less than 1e-9): 2.32978e-11.
- Checking backpropagation (w/ Regularization)... relative difference (should be less than 1e-9): 2.26112e-11.

- Cost at fixed debugging parameters (w/ lambda = 10) (should be about 0.576051): 0.576051.

The script then trains the neural network using our implementation over 50 iterations. This ends with a cost usually around 6.490774e-01. It then shows a visualization of the hidden layer. Below is the visualization of the hidden layer of the neural network generated by our implementation.



Last, the script calculates the network's accuracy over the training set. Our network had a 95.84% accuracy. Note that this accuracy could have been adjusted (and possibly improved) by running more iterations or tweaking of the regularization parameter. We used “1” for our lambda regularization parameter.

Conclusion:
Overall this homework project was an adequate introduction to logistical regression as well as elementary neural networks. Given the information obtained by completing this assignment, implementation of neural networks on other data sources should be possible as well as easier to implement.

References:
https://www.coursera.org/learn/machine-learning