**David R Mohler**

**EE 5410: Neural Networks**

**Exam #1 Take Home**

**04-02-2018**

**Problem #1: Classification of MNIST Data set with Back Propagation**

The MNIST data set consists of a large number of handwritten digits described by 28x28 pixel images. For this first problem we designed a three-layer back propagation neural network which consists of the input layer, a single hidden layer, and the output layer. Due to the large amount of input data being used, 55,000 training patterns, I began with a reduced training set of the first 1000 patterns in order to debug the network and test a minimal proof of concept. A relatively high learning rate () was used for the network in order to encourage faster reductions in the mean square error (MSE) of classifications. When observing the MSE some artifacts appear where the curve is not monotonically decreasing. This is likely due to the chosen learning rate being as large as it is. Once the network had been tested, the entire data set was presented to the network for training. It should be noted that trials conducted on the complete training set were reduced in maximum allowable epochs due to the massive time consumption of the network training process. To classify the data, ten output classes are use which yield non-normalized probabilities that a given pattern belongs to a certain class. The maximum of these scores are taken as the classification of the pattern by the network. The TensorFlow implementation of the MNIST classifier from class was able to obtain an accuracy of approximately 85.94% over the course of 1000 epochs. By comparison, it can be seen from the table below that my implementation using 10 hidden neurons is able to classify the data with approximately 3.5% higher accuracy, however, due to the lack of the use of the GPU in my implementation, TensorFlow is significantly faster in its training process.

|  |  |  |  |
| --- | --- | --- | --- |
| **Number of Training Patterns** | **Number of Hidden Neurons** | **Learning Rate** | **Accuracy**  **%** |
| 1000 | 5 | 0.8 | 73.26 |
| 55000 | 5 | 0.8 | 85.8 |
| 55000 | 7 | 0.8 | 88.13 |
| 55000 | 10 | 0.8 | 89.43 |

**Problem #2: Multi-Layer Back Propagation Networks:**

For problem 2 we developed a five-layer network consisting of the input layer, three hidden layers and the output layer. For simplicity, I chose to implement the network using the much smaller data set used to demonstrate back propagation from class. This consisted of 120 training patterns and 30 testing patterns. The network was tested under a variety of different hyper parameters to observe the performance of the network in classification of the testing data. As is shown in the figure below, when operating with lower numbers of neurons in the hidden layers there is a greater likelihood of misclassification (in this instance), similarly in the presence of fewer neurons the network required a greater number of epochs to minimize the MSE. The network was generally tested at learning rates greater than to minimize training time, however, the network is still able to classify correctly given a lower learning rate given enough time to converge, as shown by the last row of the table below.

|  |  |  |  |
| --- | --- | --- | --- |
| **Hidden Layer Topology**  **(H1,H2,H3) Number of Neurons** | **Number of Epochs** | **Learning Rate** | **Accuracy**  **%** |
| 3,3,3 | 250 | 0.5 | 96.77 |
| 5,6,5 | 250 | 0.5 | 100 |
| 10,5,10 | 250 | 0.5 | 100 |
| 10,5,10 | 250 | 0.25 | 100 |
| 10,5,10 | 1500 | 0.1 | 100 |

**Extra Credit:** Please note that the extra credit for variable size network has been placed on the server but it has some limitations. **Please do not input any less than 2 hidden layers** as it was designed for displaying back prop in multi-layer networks that were larger than the ones previously shown. Also, note that above 3 layers the network appears have issues with memorization. i.e. the network drives the error to approximately 0.33 and no longer learns. This is likely due to the small data set, even though the order of the patterns is constantly shuffled, the sheer volume is not enough and allows the network to memorize and not learn; it should be possible to modify to use MNIST as a better example. However, I did not have enough remaining time to demonstrate this due to the large execution time of MNIST data without use of the GPU. The program is capable of building larger network topologies, however the data set used is not sufficient to demonstrate their capabilities.

* See Mohler\_Exam1Prob2Flex.py