**Handwritten Digital Recognition using Shall ANN**

Hevander Da Costa

University of North Georgia

**Abstract - Artificial neural networks are computing systems inspired by biological neural networks. Neural networks are composed of simple elements operating in parallel. Parallel processing helps to process given data more efficiently than linear processing. They are used along with many other machine learning algorithms to perform “learning tasks” after receiving examples as input. One of the learning tasks is pattern recognition. Through an automated process the neural network is able to classify the received data by analyzing regularities within or between data tests. In the case of this project, the neural network will be used to recognize handwritten examples of numerical values one through to as the desired target values of numbers one through nine.**

1. Introduction

In this project the neural network will be involved in the process of supervised learning. Supervised learning is the data mining task of inferring a function from labeled training data. The training data consist of a set of training examples. Training data in this project comes from the MNIST Database of 60000 examples of handwritten digits. A shallow neural network is used in this project containing ten hidden layers. Each layer contains a set number of set number of nodes to process the examples data in parallel.

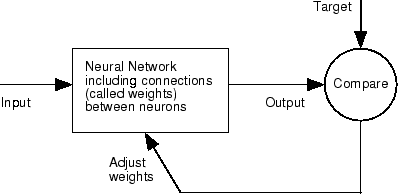


Figure. 1.

The examples are used to train the neural network. Training is used to assign specific weights and biases to the neural network. After which testing data is implemented to. The labels, training and test, dictate to the neural network what the outputs should be.

1. Considerations

The neural shallow neural network used in this experimental project is produced with a stock MATLAB pattern recognition neural network function “patternnet”. Many of the other techniques of data processing in this project were a composite of stock neural network related functions and examples provided by “MathWorks”. Different training functions were used in this experiment which produced varying degrees of accuracy. Even with testing interactions using the same training functions degrees of accuracy can vary. The training functions used were also stock MATLAB functions. Use of stock functions and composite methods of data processing were used to save time, as creating a neural network from the ground up would be a longer process. The number of nodes in the neural network also had to be limited because of computer hardware limitations. Adding more nodes leads to increased runtime. The marginal improvements in accuracy did not justify the longer runtimes and taxing of the hardware.

1. Experimentation and Results

Initially the project involved using code to trim the examples. Theoretically trimming the examples would have decreased run times as the neural network would have to deal with fewer extra pixel values, leaving mostly the pixel values related to the digits. However, the written examples have different sizes and gray scale gradations. This would have complicated the code as the code would have to account for the varying image sizes. For simplicity the complete image was used as a twenty-eight by twenty-eight array. Being that this is a shallow neural network the number of hidden nodes stayed at ten. The number of nodes and the training functions were adjusted. A code block for a for-loop to manage the implementation of different training function per run was initially added. However, using the for loop resulted in an error -("trainFcn" cannot be set to non-existing function "t".). Within the code a for-loop manages the multiple iterations of the neural network with a linearly increase count of nodes. The neural network starts with a base count of ten nodes. It then start an incrementation of node count per run of twenty nodes up to one hundred nodes. It was observed that fewer nodes led to more iterations of a possible thousand using MATLAB’s ***nntraintool*** GUI. More iteration did not seem to strongly correlate with a higher degree of accuracy. For this experiment, it was empirically determined that node count correlated with higher degree of accuracy than iteration count.

Training Functions used in this experiment were: ***trainscg, traingda, traingd, trainlm***. Surprisingly, all these functions the same metrics for three measurement: Best Validation Performance, Error Histogram with 20 Bins, and Hidden Neurons vs. Accuracy.

figure 1


Figure. 2. Number of hidden neurons vs. accuracy.

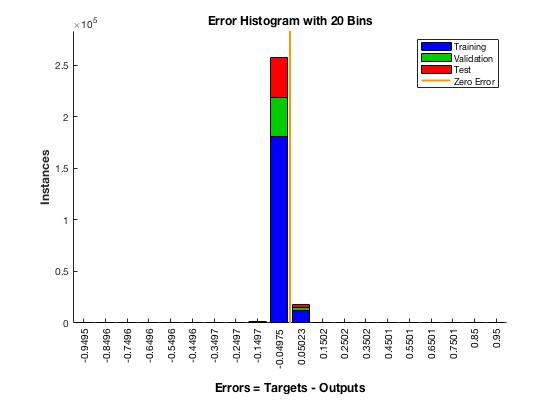
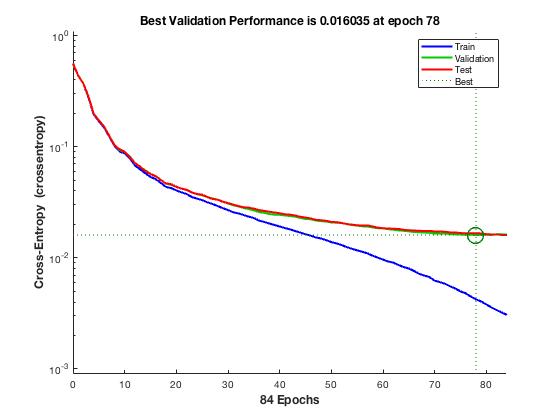


Figure. 3. Error Histogram with 20 Bins

Figure. 3. Best Validation Performance is 0.016035 at epoch 78

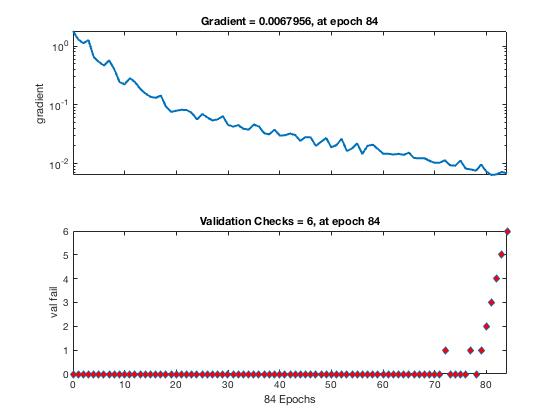


Figure. 5. Gradient = 0.0067956, at epoch 84

Figure. 6. Validation Checks = 6, at epoch 84

The functions varied in two measurements Receiver Operating Characteristic and Confusion as. Examples of the slight variance are shown in figures 6&7, 8&9.

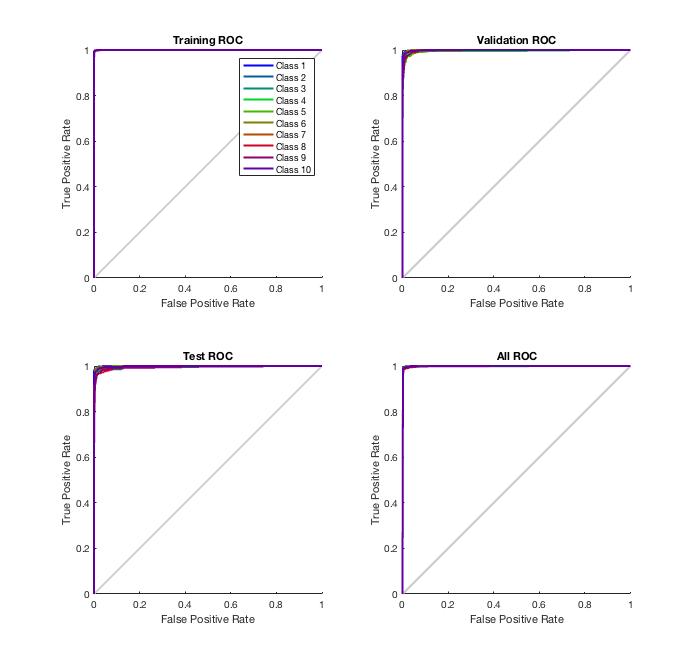


Figure. 7. Graphs of ROC for Training, Validation, Test, All. Function trainscg

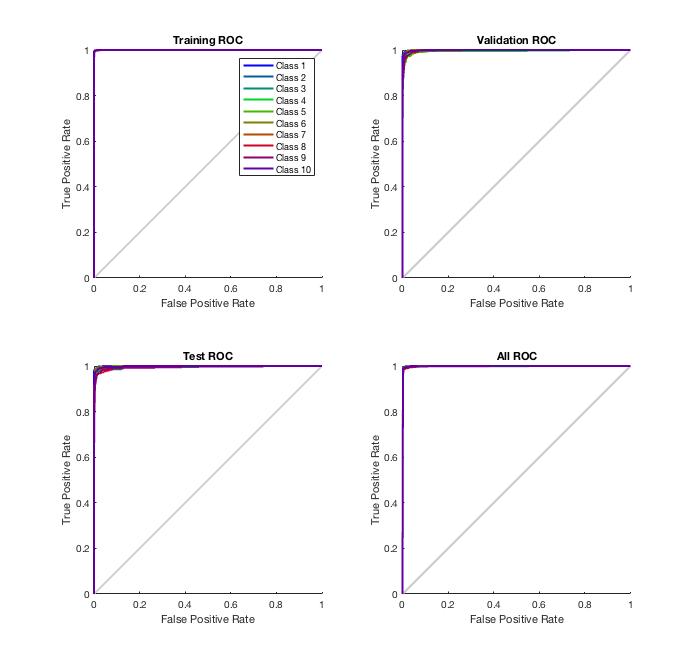


Figure. 8. Graphs of ROC for Training, Validation, Test, All. Function trainlm

The ***receiver operating characteristic*** is a metric used to check the quality of classifiers. For each class of a classifier, roc applies threshold values across the interval [0,1] to outputs. For each threshold, two values are calculated, the True Positive Ratio (TPR) and the False Positive Ratio (FPR). For a particular class *i*, TPR is the number of outputs whose actual and predicted class is class *i*, divided by the number of outputs whose predicted class is class *i*. FPR is the number of outputs whose actual class is not class *i*, but predicted class is class *i*, divided by the number of outputs whose predicted class is not class *i*.

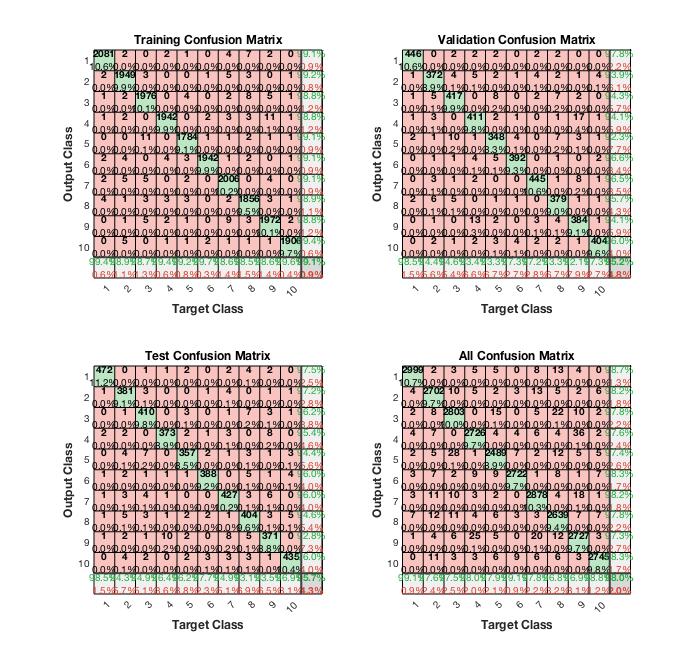


Figure. 9. Confusion Matric for Training, Validation, Test, All. Function trainscg

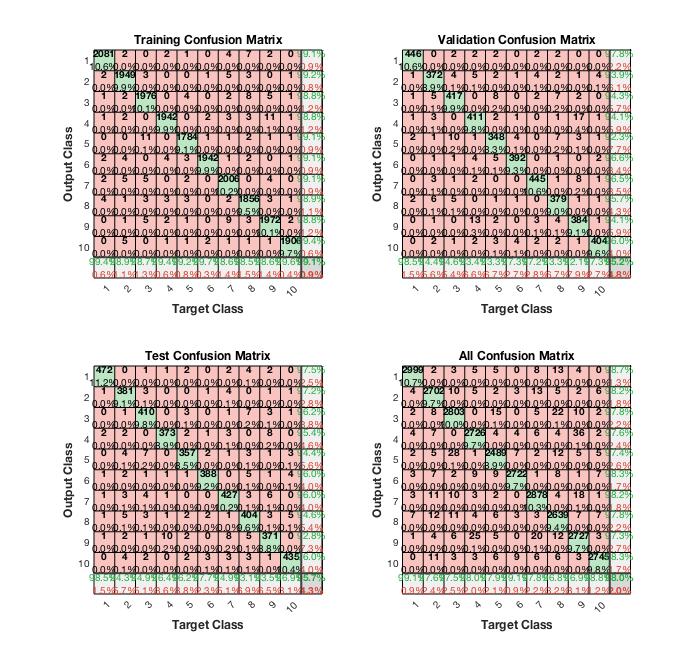


Figure. 10. Confusion Matric for Training, Validation, Test, All. Function trainlm

A ***confusion matrix***, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically for supervised learning .Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class or vice versa.

1. Conclusion

The four functions used in this experiment. This is indicative of the robustness of the functions. With six decimal places functions: trainscg, traingda, traingd, trainlm, had the same value for “Number of hidden Nodes vs. Accuracy”. Moreover, Error Histogram for with 20 Bins, and Gradient, and Validation Checks were the same. Being that these are built in MATLAB training functions a high degree of accuracy is expected. All of the had an over ninety-fiver percent accuracy for sixty to one hundred hidden neurons. However, it was observed that eighty neurons have the best accuracy. Expectations of the performance were that more neurons would linearly correlate with more accuracy. This was not the case is experimentation. The curve observed in “Number of Hidden nodes vs Accuracy” best fits a curve of natural log of x – ln(x). There is a gradual decrease in the positive rate of accuracy until eight neurons. Surprisingly as the neural network computes past eighty neural a negative rate of accuracy is observed. Further research found this to be the case with shallow neural networks – “Shallow learning refers to machine learning methods that plateau at a certain level of performance when you add more examples and training data to the network.”.

1. References
2. Select a Web Site. (n.d.). Retrieved November 12, 2018, from https://www.mathworks.com/help/deeplearning/gs/shallow-networks-for-pattern-recognition-clustering-and-time-series.html
3. Confusion. (n.d.). Retrieved November 12, 2018, from https://www.mathworks.com/help/deeplearning/ref/roc.html

MathWorks

1. Artificial Neural Networks for Beginners. (n.d.). Retrieved November 12, 2018, from https://blogs.mathworks.com/loren/2015/08/04/artificial-neural-networks-for-beginners/

MathWorks

1. THE MNIST DATABASE. (n.d.). Retrieved November 12, 2018, from http://yann.lecun.com/exdb/mnist/

Yann LeCun, Courant Institute, NYU Corinna Cortes, Google Labs, New York Christopher J.C. Burges, Microsoft Research, Redmond

1. Traingda. (n.d.). Retrieved November 12, 2018, from https://www.mathworks.com/help/deeplearning/ref/traingd.html

MathWorks

1. Competlayer. (n.d.). Retrieved November 12, 2018, from https://www.mathworks.com/help/deeplearning/ref/patternnet.html ,MathWorks