Multi-layer Feed Forward Neural Networks

Graham Burgsma

***Abstract*—A neural network is a complicated set of neurons, connections and many other parts. When working together these parts can learn and predict data otherwise difficult to categorize. The accuracy of a neural network depends on many different learning parameters such as learning rate, momentum rate, activation function and cross validation. The accuracy also depends on the network architecture with parameters such as the number of hidden layers and hidden neurons per layer. When all these values are working in unison, a neural network can have near perfect accuracy. This paper looks at investigating each of the learning parameters and network architecture individually to form the best combination possible.**

Keywords*—*Artificial Neural Network, Backpropagation, Feed Forward, Iris Plant Data, Breast Cancer Data.

I. Introduction

The purpose of this paper is to discover how learning parameters and neural architecture affect feed forward neural networks. Activation function, learning rate and momentum rate will be adjusted and compared and shown the difference they have on the network. The neural network itself will be tested by varying the number of hidden neurons and hidden layers in the network architecture.

Data being used for testing is the iris plant data set and Wisconsin breast cancer data set. Both are very common sets of data used for testing machine learning algorithms. These data sets are good for machine learning as each of the classifications in each are separable and so a neural network can learn and predict from the data. They are also good data sets because they are real data which makes it more applicable to see how machine learning algorithms work with such data. Also, these data sets are fairly simple and have been used many times that they have become a benchmark for many machine learning algorithms. For these reasons these two data sets have been chosen for the purpose of this paper.

The iris data set has four numeric attributes and a classification. There are three classifications for this data set: Iris Setosa, Iris Versicolour and Iris Virginica. The set contains 150 instances, 50 per classification. The four numerical attributes in this data set are predictable which means a machine learning algorithm should be able to learn and then accurately classify each of the instances. The Wisconsin breast cancer data set is a much larger data set than the iris data set. There are nine numeric attributes, each with the domain of 1-10. This data set has two classifications: Benign and Malignant. There are 699 instances in this data set, however 16 of them a missing attribute value thus were excluded. There is a 65.5% distribution for Benign and a 34.5% distribution for Malignant.

II. Neural Networks

An artificial neural network (ANN) in machine learning is a network that resembles biological neural networks, such as in the human brain. ANNs model the brain in two ways, the first is that knowledge acquired is done through a learning process. Secondly, ANNs use a network of connected neurons with synaptic weights, that adjust with learning, to store information and patterns. An ANN is good at taking a large number of inputs to produce a fairly accurate output. ANNs are very good at pattern recognition, some common uses are hand writing interpretation and image analysis such as finger print recognition or image compression. ANNs can be used for both supervised and unsupervised learning. With supervised learning the outcome is most often looking for a classification where unsupervised learning usually works to organize or cluster input.

A. Feed Forward Network

Feed Forward Neural Networks are the most simple form of an ANN in which the connections between neurons only move in one direction, they do not move backwards or have any cycles. Connections start from the inputs and are connected to one or more hidden layers which are then connected to the output. It is common for feed forward networks to use backpropagation for learning.

B. Backpropagation

Backpropagation calculates error in an neural network and is a popular method used for supervised learning networks. Backpropagation works in a three step process: forward propagation, backward propagation and weight update. The forward propagation or forward pass moves the training input linearly through the network in order to produce the output of the network. The output is then compared to the teacher value (in supervised learning) to produce an error signal. In backward propagation the error is moved backward through the network to each of the neurons. In the final step, weight update, the weights of the network are updated which is the key process in the network learning.

```
for each example in the training set do
    Forward Pass
    Calculate error (Teacher – Output)
    Backward pass
    Update the weights
end
```

C. Activation Function

The activation function is an important aspect of neural networks. Different activation functions have different uses, for the data sets used the logistic and hyperbolic tangent function were used. These equations can be seen in Figure 1 and Figure 2.

$$f(x) = \frac{1}{1+e^{-x}}$$

Fig. 1: Logistic Equation

$$f(x) = \frac{2}{1+e^{-2x}} - 1$$

Fig. 2: TanH Equation

III. Analysis

A. T-Test

The t-test evaluates two sets of data and determines if they are statistically different from each other. A t-test has two values that come from it, the t-value and the p-value. The t-value in a t-test represents the significance of different between the two sets of data. The large the t-value, the more significant the different is between the sets. The p-value in a t-test represents the probability that the two sets come from the same group. P-value is a percentage, so a p-value of 0.6 would mean there is a 60% chance the two sets come from the same group. The t-test is useful in determining if there is a statistical difference in two different sets or for a before and after test.

B. ANOVA

There are multiple types of an ANOVA test. For only one factor, a One Way ANOVA test is used to compare the mean from three or more different samples using an F distribution. A One Way ANOVA test using only two samples is equivalent to a T-Test. Similar to a T-Test it is used as a measure of variance between data sets.

IV. Results

Each test will be run against a control to compare and see the differences the changes make. The parameters for the control test are shown in Table I and the graph is shown in Figure 3. The program uses a seed value for generating random value and shuffling the data, this way the control test will always have the exact same result so comparisons are fair.

Data Set	Breast Cancer
Activation Function	Logistic
Epochs	250
Cross-Validation Folds	2 (Holdout)
Learning Rate	0.2
Momentum Rate	0.0
Hidden Layers	1
Hidden Neurons	15
Highest Training Result	0.90789
Highest Testing Result	0.87537
Average Training Result	0.89077
Average Testing Result	0.85073
Standard Deviation Training	0.01021
Standard Deviation Testing:	0.00680

TABLE I: Control Parameters and Results

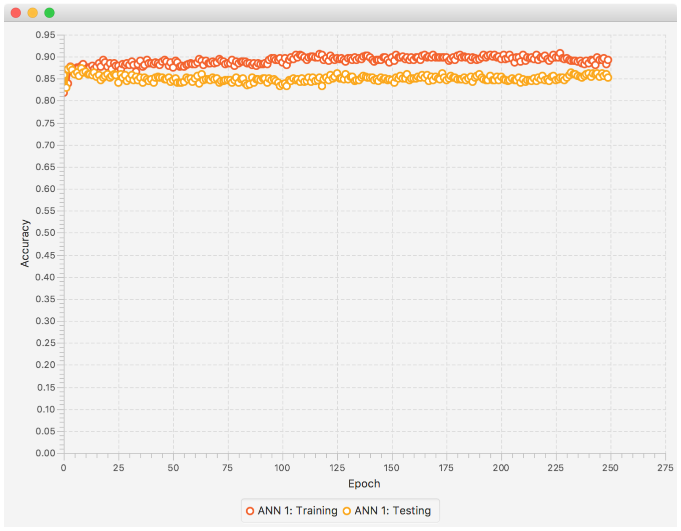

Fig. 3: Control Results

A. Hidden Neurons

Hidden neurons in a neural network are what increase the number of connections. Increasing the number of connections increases the weights in the network and therefore increase how well the network can learn and predict. For this test hidden neurons were increased to 50 from the original 15. As seen in figure 4, there is an increase in accuracy with increasing the number of hidden neurons. A paired t-test on the testing set of each runs gives -50.8990. From this it is evident there is a large difference between these runs, the negative shows that this run gave an increase over the control run. A one way ANOVA test gives 2590.7123 which is redundant in this kind of test as a one way ANOVA test with only two samples is the same as a t-test. The relationship between a t-test and an ANOVA test is $ANOVA = t\text{-}test^2$.

Highest Training Result	0.96053
Highest Testing Result	0.93255
Average Training Result	0.93246
Average Testing Result	0.88748
Standard Deviation Training	0.01399
Standard Deviation Testing:	0.00917

TABLE II: Increase Hidden Neurons Results

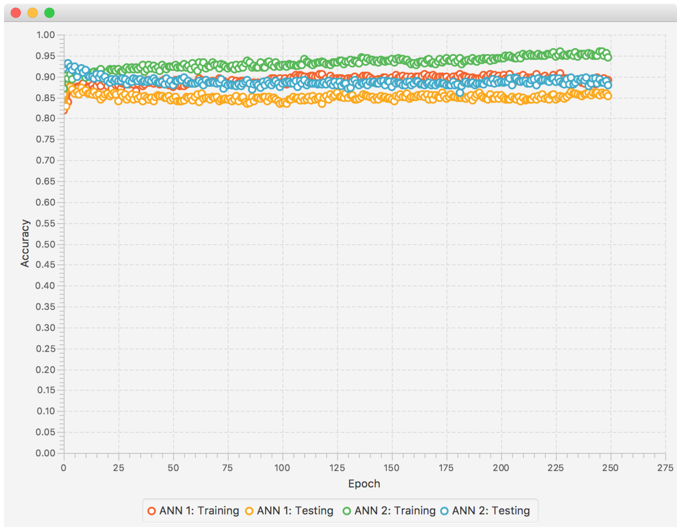

Fig. 4: 50 Hidden Neurons

B. Hidden Layers

In most cases increasing the number of hidden layers is not necessary, in fact no hidden layers are needed if the data is linearly separable. For this test it was chosen to increase the hidden layers to 4. With such a big increase it was theorized to have a noticeable improvement on the accuracy. From Figure 5 it is evident there is a slight increase in the accuracy over the number of epochs. Comparing the highest and average results to the control set, there is a slight increase in all categories. A t-test of the control and experimental sets gave -5.4118 which shows a light increase but isn't very significant.

Highest Training Result	0.94591
Highest Testing Result	0.90323
Average Training Result	0.91249
Average Testing Result	0.86057
Standard Deviation Training	0.04205
Standard Deviation Testing:	0.02794

TABLE III: Increase Hidden Layers Results

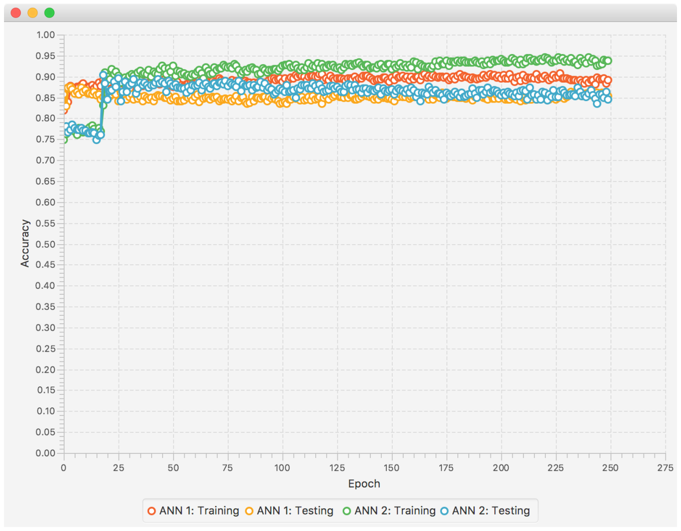

Fig. 5: 4 Hidden Layers

C. Learning Rate

Learning rate controls how fast a neural network can update its weights and make changes. Most often a small learning rate is better to avoid making abrupt changes in the network. For this test two runs were compared to the control, one with a learning rate of 0.8 and one with 0.005. From Figure 6 ANN 2 has a learning rate of 0.005 and ANN 3 has a learning rate of 0.8. From the graph alone it is clear a small learning rate can improve a neural network and a large learning rate can result in poor results. T-test results for the run with learning rate of 0.005 was -44.4552 which is a substantial improvement over the control run. The run with learning rate of 0.8 had a t-test result of 63.2975 which proves there is a significant different between the control and experimental run. From these findings it can be concluded that a small learning rate has an improvement on the accuracy of a neural network.

Highest Training Result	0.96491
Highest Testing Result	0.94428
Average Training Result	0.94576
Average Testing Result	0.93028
Standard Deviation Training	0.03061
Standard Deviation Testing:	0.02746

TABLE IV: Learning Rate 0.005 Results

Highest Training Result	0.81140
Highest Testing Result	0.81085
Average Training Result	0.69477
Average Testing Result	0.68822
Standard Deviation Training	0.03389
Standard Deviation Testing:	0.04002

TABLE V: Learning Rate 0.8 Results

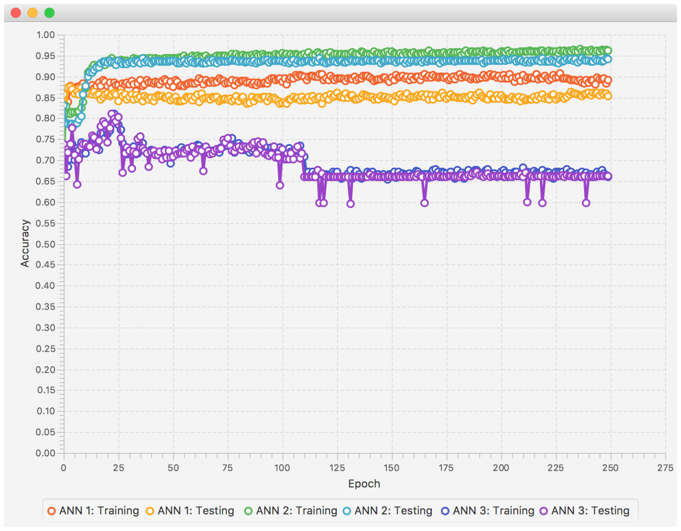

Fig. 6: Learning Rate 0.005 and 0.8

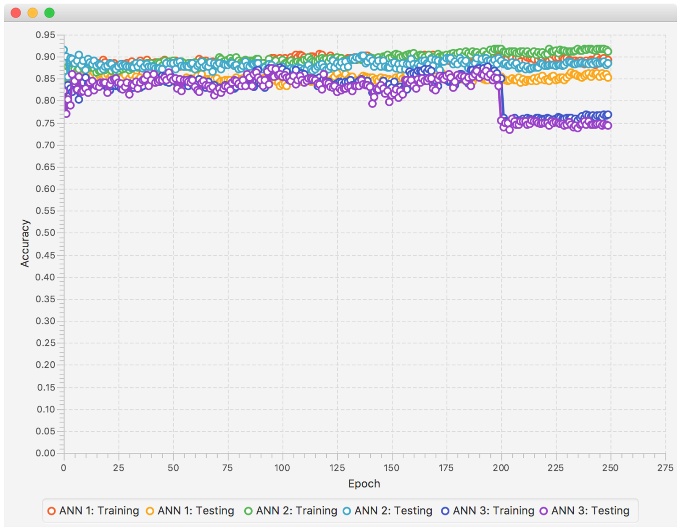

Fig. 7: Momentum Rate 0.1 and 0.5

D. Momentum Rate

Momentum is important in a neural network although not necessary. There are situations in which the neural network may get stuck in a local minimum, momentum helps push it out of these local minimums. T-test results confirm the results found in Figure 7. The t-test result for 0.1 momentum is -46.5493 which proves it improved over the control run. From the graph it may be seen to show that 0.5 momentum has a large negative effect on the run because there is a significant drop at 200 epoch. However, there is only a light change from the control test as the t-test returned 11.7290. Stopping at 200 epoch with 0.5 momentum, all readings are very close to the control run. From these results it can be concluded a small amount of momentum can slightly improve results and larger momentum has little effect but can produce inconsistent results. It should be noted that momentum has more effect with different data sets and with different parameters.

Highest Training Result	0.91813
Highest Testing Result	0.91496
Average Training Result	0.89447
Average Testing Result	0.88190
Standard Deviation Training	0.01426
Standard Deviation Testing:	0.00811

TABLE VI: Momentum Rate 0.1 Results

Highest Training Result	0.87719
Highest Testing Result	0.87390
Average Training Result	0.82734
Average Testing Result	0.82110
Standard Deviation Training	0.03588
Standard Deviation Testing:	0.03937

TABLE VII: Momentum Rate 0.5 Results

E. Activation Function

Two activation functions were chosen for the implementation of this neural network: logistic and hyperbolic tangent. Figure 8 shows three different runs. ANN 1 is the control run and uses the logistic activation function. ANN 2 and 3 both use the TanH activation function but with different learning parameters. ANN 2 had identical learning parameters to the control function just using TanH activation function. This produced very poor results and the learning flat-lined near 60% accuracy. It seemed use of the Tanh activation function caused the learning to get stuck in a local minimum. To counter this the learning rate was reduced from 0.2 to 0.02 so the learning would take smaller increments. This combination worked very well and produced one of the best results so far. See Figure 8 for graph representation of runs.

T-test results were conclusive with the graph and other results. The run with only TanH activation function had a very significant difference from the control of 235.5276. The run with reduced learning rate had a large change of -175.3389 from the control run. Along with the t-test, the average and highest results from this run were very good as seen in Table IX.

Highest Training Result	0.64766
Highest Testing Result	0.66276
Average Training Result	0.59563
Average Testing Result	0.58114
Standard Deviation Training	0.00953
Standard Deviation Testing:	0.01677

TABLE VIII: TanH Activation Function

Highest Training Result	0.98099
Highest Testing Result	0.97067
Average Training Result	0.97118
Average Testing Result	0.95957
Standard Deviation Training	0.00835
Standard Deviation Testing:	0.00708

TABLE IX: TanH Activation Function — 0.02 Learning Rate

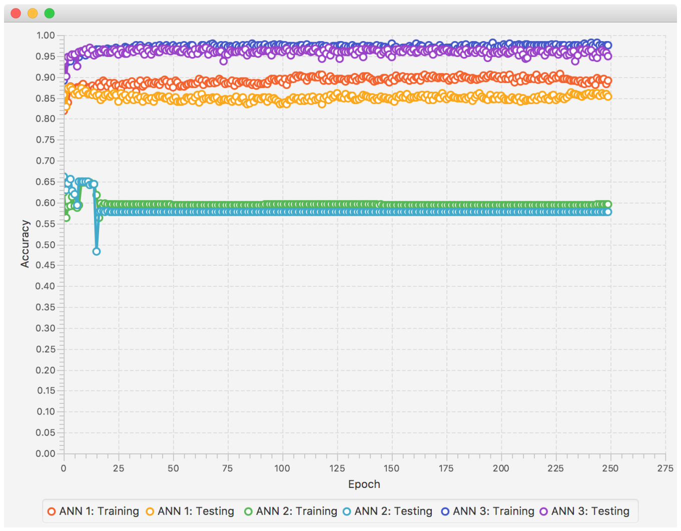

Fig. 8: TanH Activation Function

F. *Cross Validation*

The control for these runs is using the holdout method or simply cross validation with $k = 2$ folds. A run with $k = 5$ and $k = 10$ were tested and both flat-lined around 73% accuracy. After some experimenting it was determined that reducing the number of epochs allowed the cross validation to improve on the accuracy. Reducing the epochs down to 100 the run with $k = 5$ improved over the control run and had a t-test score of -40.7864. $k = 10$ however decreased in accuracy with a t-test score of 30.1834. Reducing the number of epochs further, to 50, improved both the $k = 5$ and $k = 10$ runs with t-test scores of -25.6672 and -39.0617 respectively. It is unclear why reducing the number of epochs would allow for cross validation to improve on the network accuracy, but it is an interesting finding.

G. *Best Parameters*

The best learning and neural architecture parameters were found by applying the best techniques found when testing each parameter separately. These parameters can be found along with the results in Table X and Figure 9. A t-test with the control test and the experimental best parameters gave a result of -234.5273 which is a very high amount of change between the sets. The average for the testing set was 96% accuracy which is an impressive score for this data set.

Epochs	250
Cross-Validation Folds	5
Learning Rate	0.002
Momentum Rate	0.1
Activation Function	TANH
Hidden Layers	1
Hidden Neurons	30
Highest Training Result	0.98135
Highest Testing Result	0.97353
Average Training Result	0.97689
Average Testing Result	0.96379
Standard Deviation Training	0.00603
Standard Deviation Testing:	0.00344

TABLE X: Best Results

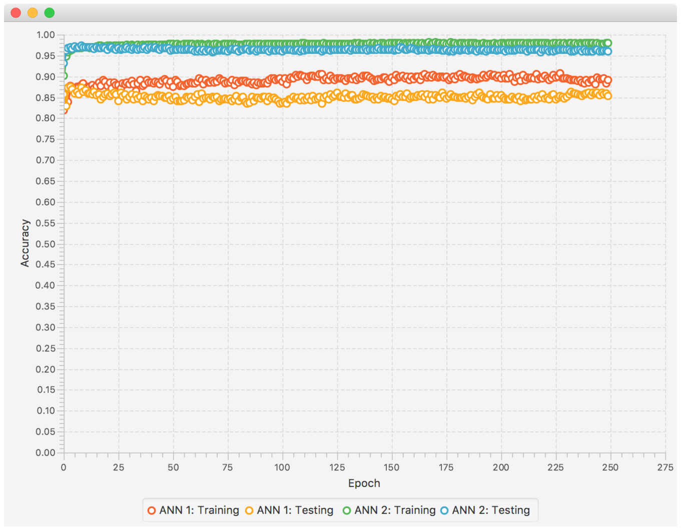

Fig. 9: Best Results

V. Conclusion

In conclusion, neural networks are powerful machine learning algorithms. There are many factors that affect the accuracy of a neural network. It has been discovered that in general a high number of hidden neurons, around 30-50 with only one hidden layer gives good results. Also, small learning rate near 0.005 and small momentum near 0.1 work well to keep the neural network learning. Activation function is mostly dependant on the data set being used, however for the data sets used in this experiment the hyperbolic tangent function performed better than the logistic function. Cross validation helps slightly improve accuracy but seems more useful in increasing the consistency of runs as a higher k-fold averages k number of runs. Each element of a neural network needs to work together with the rest of it for the system to work well. When it is working well, artificial neural networks can achieve high accuracy and see patterns in data humans may have not detected.