CS457 Project 3

Nick Rohde

May 3, 2018

Contents

1 Introduction

This project revolved around how the accuracy of a Neural Network changes as the number of nodes in the hidden layer varies. Four simple mathematical functions were implemented and networks with 1-10 hidden neurons trained to discover which performs best. The functions used for this were:

$$f(x) = \frac{1}{x} \qquad 0 < x \leq 100$$

$$g(x) = log_{10}(x) \qquad 0 < x \leq 10$$

$$h(x) = e^{-x} \qquad 0 < x \leq 10$$

$$l(x) = sin(x) \qquad 0 \leq x \leq \frac{\pi}{2}$$

2 Experimentation

For the experiments, the neural networks were trained on different sizes of training, validation, and testing sets. An equally distributed set of N points in the function's domain was generated, and then randomly distributed among the three sets; the 5 sizes for experimentation were 20, 40, 60, 80, 100, which were divided roughly $60:20:20$ over the training, validation, and testing sets, respectively. These sets were sufficiently large to allow the network to form an understanding of the function, without over-training it; on top of this, due to the small domain, generating larger sets would've put points extremely close to each other.

The networks were trained with a back-propagation method that updated weights after the entire training set had been evaluated, training was stopped either after 100 iterations – though they all stopped after at most 3 iterations – or once the network stopped improving – this was defined as a less than 0.5% increase in accuracy of two consecutive training iterations (i.e. $error_{i-1} - error_i < 0.005 \implies stop$).

The experimentation data displayed below shows networks with 1-10 hidden neurons, networks with more neurons were tested, however, after 10 there was absolutely no increase in performance, thus, they were omitted to avoid redundancy.

3 Data

This section displays the numerical data from the experimentation, discussed in the Analysis section, as well as graphical representations (best and worst) of the tests run on the neural networks.

3.1 Reciprocal Function

Table 1: Testint Set Results

Test Set Size: Hidden Neurons	4	9	14	19	24
	Testing Error (%)				
1	1.969×10^{-5}	2.269×10^{-5}	5.813×10^{-5}	7.642×10^{-5}	9.335×10^{-5}
2	3.449×10^{-5}	3.220×10^{-4}	9.165×10^{-3}	3.944×10^{-3}	2.041×10^{-4}
3	5.131×10^{-4}	6.060×10^{-5}	1.951×10^{-4}	7.398×10^{-3}	3.248×10^{-4}
4	8.250×10^{-6}	1.048×10^{-4}	1.710×10^{-3}	1.424×10^{-2}	1.745×10^{-4}
5	2.432×10^{-3}	1.354×10^{-4}	1.187×10^{-2}	9.994×10^{-4}	4.318×10^{-4}
6	6.561×10^{-3}	1.328×10^{-3}	3.738×10^{-3}	1.473×10^{-4}	1.387×10^{-2}
7	8.628×10^{-4}	5.611×10^{-5}	7.416×10^{-4}	1.220×10^{-2}	1.528×10^{-2}
8	4.535×10^{-5}	2.885×10^{-3}	1.723×10^{-3}	1.194×10^{-2}	8.071×10^{-5}
9	8.835×10^{-5}	5.035×10^{-3}	1.460×10^{-3}	9.038×10^{-3}	9.236×10^{-3}
10	5.811×10^{-3}	2.232×10^{-4}	3.422×10^{-3}	4.022×10^{-3}	3.729×10^{-2}

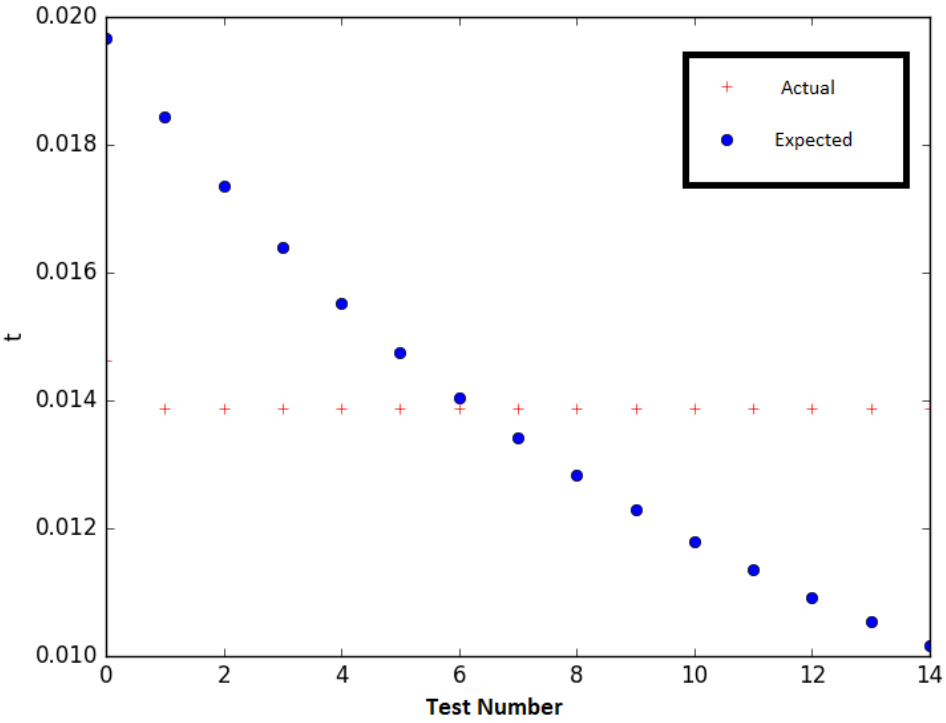

Figure 1: Worst Performance, 1 Neuron, Training with 14 Points.

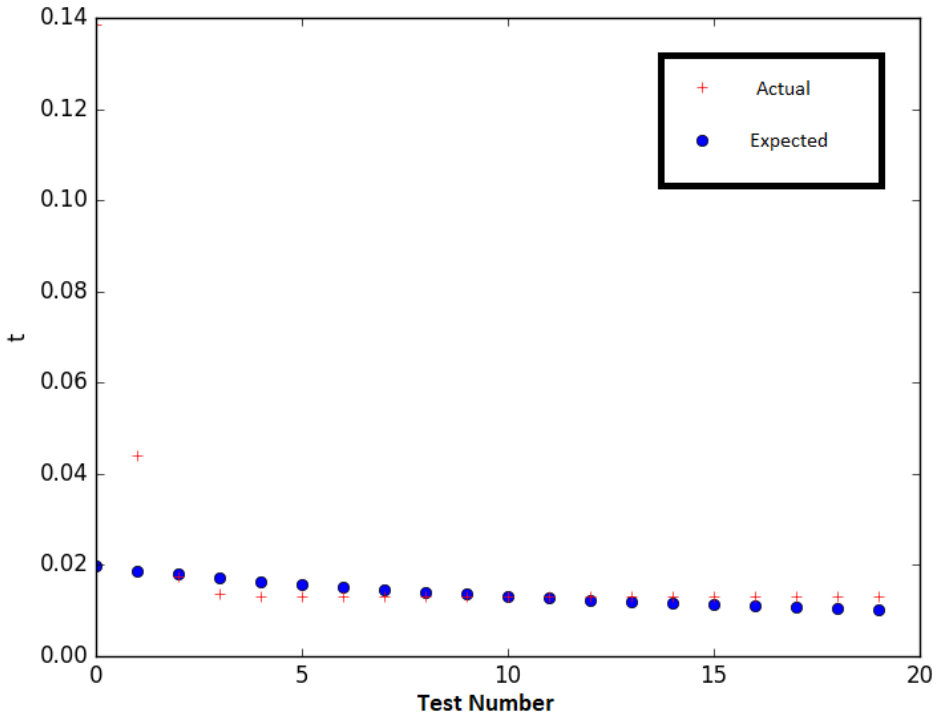

Figure 2: Best Performance, 3 Neurons, Training with 19 Points.

3.2 Common Log

Table 2: Testint Set Results

Test Set Size: Hidden Neurons	4	9	14	19	24
	Testing Error (%)				
1	8.880×10^{-3}	3.302×10^{-2}	4.209×10^{-1}	4.787×10^{-2}	1.458×10^{-1}
2	5.599×10^{-3}	5.048×10^{-5}	4.172×10^{-2}	9.343×10^{-2}	6.543×10^{-4}
3	9.566×10^{-2}	4.443×10^{-2}	1.382×10^{-1}	2.125×10^{-2}	1.996×10^{-1}
4	4.003×10^{-2}	2.879×10^{-1}	8.304×10^{-2}	1.463×10^{-1}	1.712×10^{-2}
5	1.222×10^{-1}	3.596×10^{-2}	1.585×10^{-1}	5.437×10^{-3}	8.538×10^{-1}
6	2.191×10^{-2}	4.096×10^{-3}	4.012×10^{-2}	5.932×10^{-2}	5.855×10^{-2}
7	2.216×10^{-2}	1.615×10^{-2}	5.316×10^{-2}	4.213×10^{-2}	3.900×10^{-2}
8	3.135×10^{-3}	5.961×10^{-2}	1.635×10^{-2}	2.911×10^{-2}	1.201×10^{-1}
9	1.701×10^{-1}	7.511×10^{-4}	5.361×10^{-2}	2.938×10^{-1}	5.063×10^{-2}
10	3.419×10^{-2}	9.448×10^{-2}	6.656×10^{-3}	1.928×10^{-1}	1.548×10^{-2}

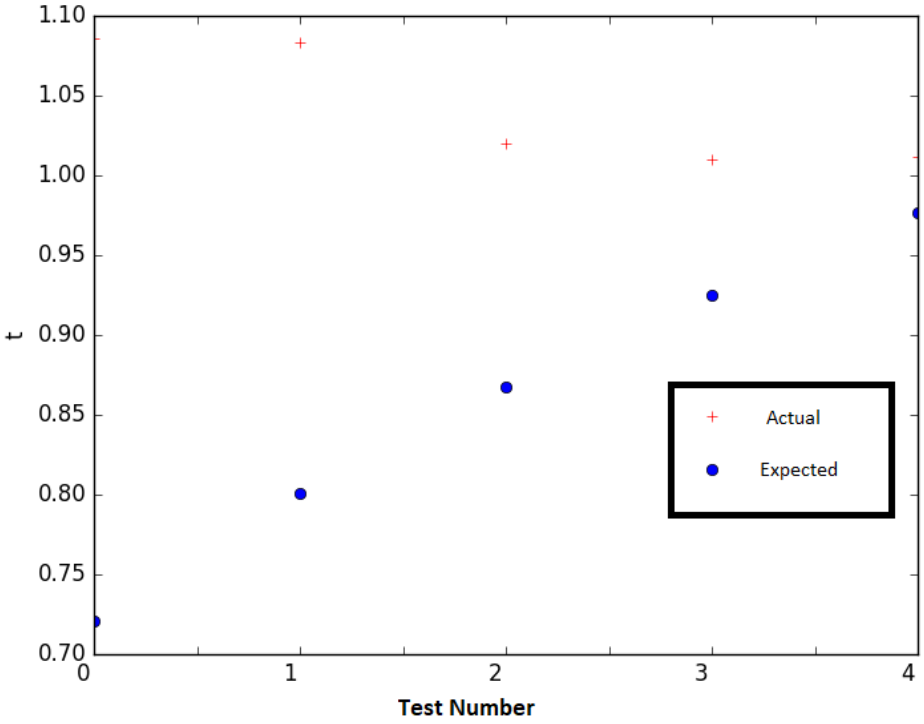

Figure 3: Worst Performance, 5 Neurons, Training with 4 Points.

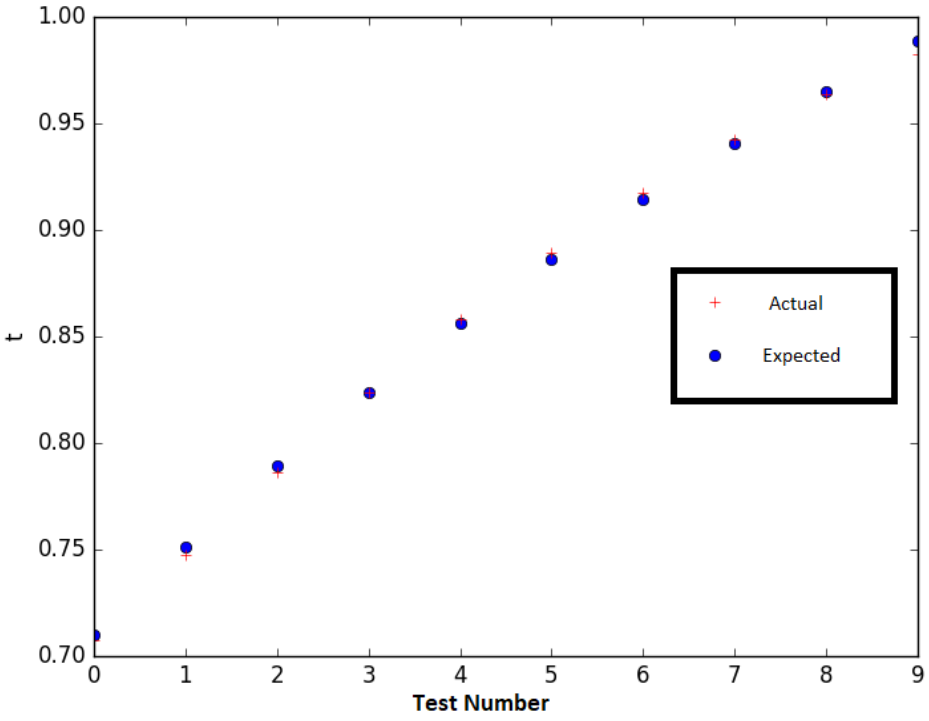

Figure 4: Best Performance, 2 Neurons, Training with 9 Points.

3.3 Exponential Function

Table 3: Testint Set Results

Test Set Size: Hidden Neurons	4	9	14	19	24
	Testing Error (%)				
1	6.400×10^{-5}	1.184×10^{-5}	1.530×10^{-5}	4.771×10^{-1}	1.774×10^{-1}
2	2.727×10^{-5}	8.093×10^{-6}	1.071×10^{-3}	3.801×10^{-3}	2.937×10^{-2}
3	1.216×10^{-3}	4.694×10^{-3}	1.215×10^{-4}	1.278×10^{-3}	2.794×10^{-2}
4	1.997×10^{-3}	1.299×10^{-3}	3.560×10^{-3}	2.630×10^{-2}	5.942×10^{-2}
5	5.863×10^{-5}	1.923×10^{-1}	1.318×10^{-2}	4.128×10^{-3}	1.079×10^{-2}
6	9.817×10^{-4}	5.125×10^{-3}	9.261×10^{-4}	1.643×10^{-4}	1.669×10^{-1}
7	6.256×10^{-3}	3.555×10^{-3}	2.979×10^{-3}	2.912×10^{-3}	2.881×10^{-3}
8	6.701×10^{-2}	8.567×10^{-3}	1.518×10^{-2}	5.481×10^{-2}	5.467×10^{-3}
9	5.976×10^{-5}	7.465×10^{-2}	3.529×10^{-2}	2.161×10^{-2}	1.197×10^{-2}
10	5.002×10^{-2}	6.116×10^{-2}	4.075×10^{-2}	1.582×10^{-2}	2.593×10^{-1}

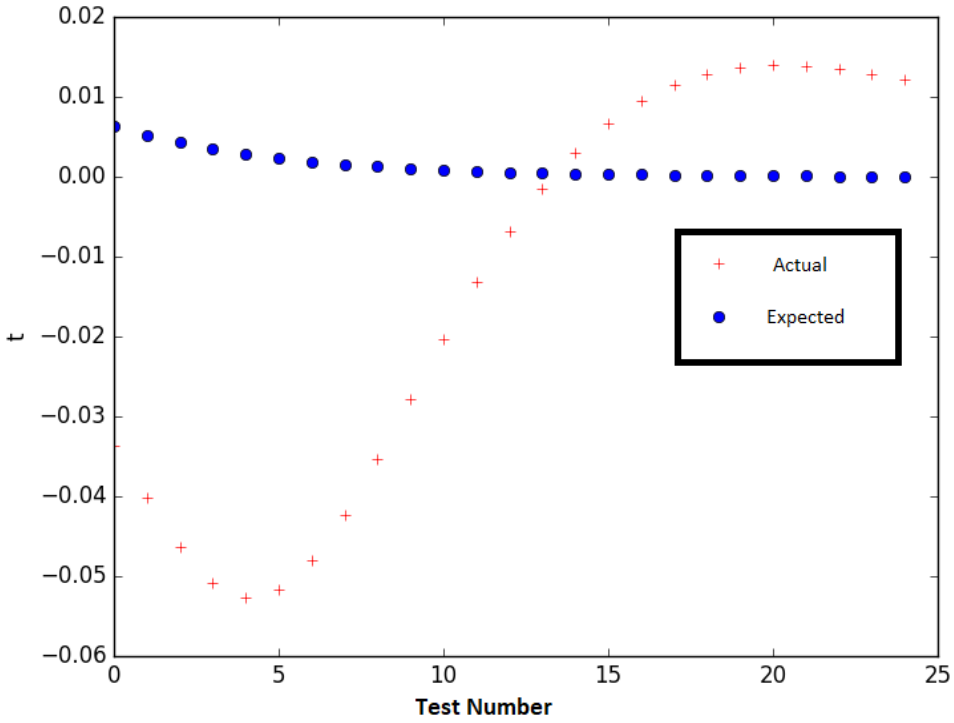

Figure 5: Worst Performance, 9 Neurons, Training with 24 Points.

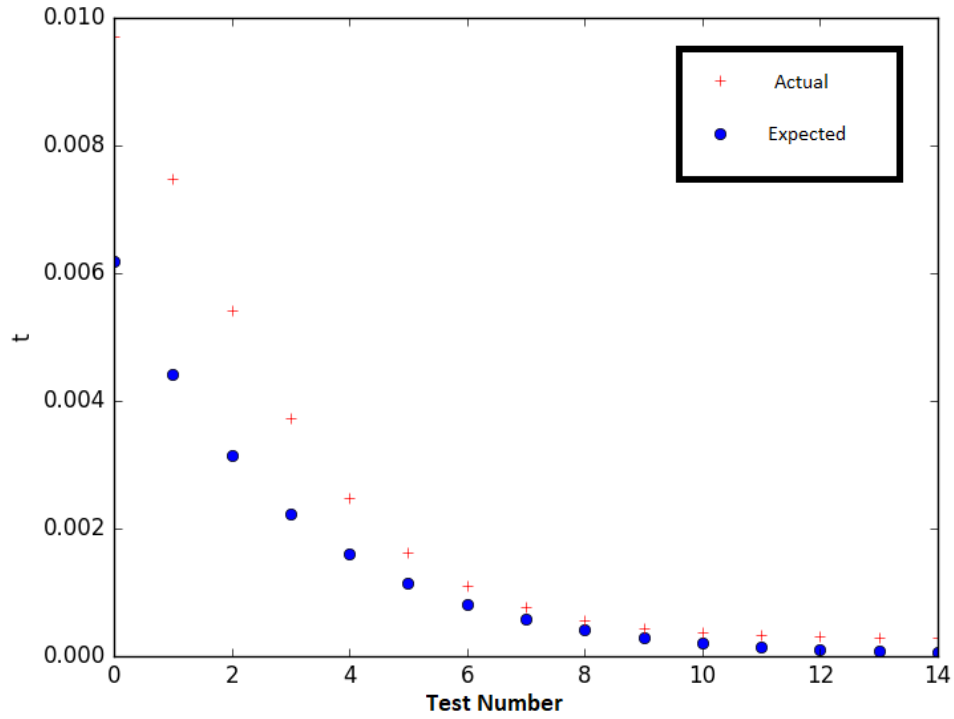

Figure 6: Best Performance, 1 Neuron, Training with 14 Points.

3.4 Sine Wave

Table 4: Testint Set Results

Test Set Size: Hidden Neurons	4	9	14	19	24
	Testing Error (%)				
1	1.785×10^{-2}	1.527×10^{0}	1.765×10^{-1}	8.301×10^{-1}	2.577×10^{0}
2	7.558×10^{-3}	4.831×10^{-3}	7.151×10^{0}	1.159×10^{0}	2.391×10^{-1}
3	4.898×10^{-1}	4.249×10^{-1}	6.471×10^{-2}	2.268×10^{0}	6.794×10^{-2}
4	3.182×10^{-1}	8.587×10^{-1}	2.628×10^{-1}	2.270×10^{-1}	4.345×10^{0}
5	4.226×10^{-2}	9.639×10^{-1}	5.250×10^{-2}	3.292×10^{-2}	6.997×10^{-1}
6	7.339×10^{-3}	4.093×10^{-1}	4.379×10^{-2}	3.230×10^{-1}	6.355×10^{-1}
7	1.442×10^{-1}	6.364×10^{-1}	1.250×10^{-1}	3.117×10^{-2}	5.872×10^{0}
8	5.624×10^{-1}	1.332×10^{-2}	1.152×10^{0}	1.672×10^{0}	3.276×10^{-1}
9	4.225×10^{-1}	3.088×10^{-2}	4.412×10^{-2}	7.927×10^{-1}	3.720×10^{0}
10	2.334×10^{-2}	2.072×10^{-2}	3.820×10^{-1}	5.174×10^{-3}	8.338×10^{-3}

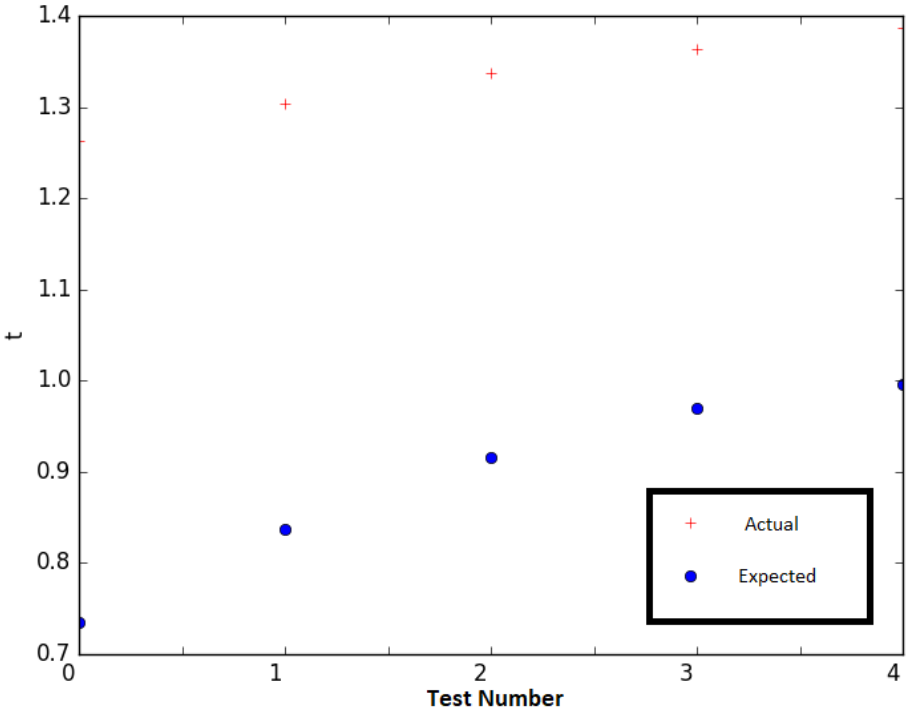

Figure 7: Worst Performance, 3 Neurons, Training with 4 Points.

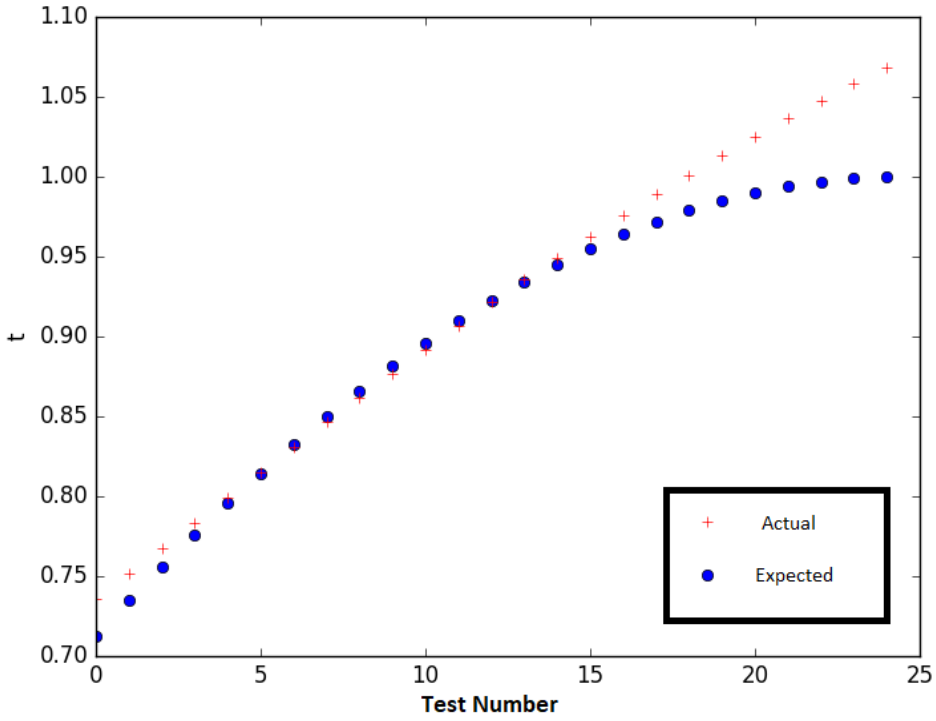

Figure 8: Best Performance, 10 Neurons, Training with 24 Points.

4 Analysis

In this section, we will discuss the Data from the previous section.

The first observation that can be made from the data is that it appears that the accuracy of the network depends more on the amount of training data it receives, as well as the spread of this data over the domain, rather than the number of neurons. As long as there are enough neurons – which appears to depend on the function being classified – to generalise the function, the data was what distinguished between a 'good' and 'bad' classifier. This is supported by the fact that all networks (regardless of function) trained with 6 data points were unable to properly generalise the function; though having more neurons did not hurt the networks trained on 6 points, the quality of the predictions that they made was terrible (almost always straight lines).

In the case of functions with a small domain, it is interesting to note that overfitting likely occurred in the larger sets of training data. The Common Log, Exponential Function, and Sine Wave all showed higher errors with the largest training set, and lower errors with the smaller ones; thus, we can assume that some amount of overfitting occurred in these cases. Though, it should be mentioned that the Sine Wave had its best prediction done by the networks with 10 neurons, and 80/100 data point sets.

4.1 Reciprocal Function

The reciprocal function, $f(x)$, was one of the easiest functions for the network to learn. With a single neuron, the network was not able to properly classify the function and instead only guessed straight lines somewhere in the range (shows in Figure 1). This continued with 2 neurons. However, with 3 neurons, the network was able to understand that the function was steeply decreasing at first, and then a straight line. Though the network was not able to predict all values perfectly, it managed to classify the majority of points well with 3 neurons (or more), this can be seen in Figure 2 which shows the network with 3 hidden neurons, trained and tested on 19 points.
Interestingly, adding more neurons did not significantly improve the performance of the network; 3 neurons was all that was needed in order to approximate $f(x)$, and more than that changed little about the quality of the predictions made by the network.

Thus, to predict the value of $f(x)$, it appears that having 3 neurons is sufficient in our domain and any more neurons only increases the time investment for training, but not the accuracy.

4.2 Common Log

The Common Log, $g(x)$, was by far the easiest function to learn for the network; on top of that, it also achieve the best result in terms of accuracy, by a long shot. With 2 neurons, and 9 test points the network achieved an error of less than 10^{-5}, the lowest error achieved by any other classifier, Figure 4 shows the plot of expected versus actual points, and here we can see how well the network predicted value of $g(x)$. All predicted values are almost a perfect match to the actual value, with little variance. Similarly to the reciprocal function, adding more neurons beyond the best performance only increased the total error, and did not provide a significant improvement in terms of prediction accuracy.

What is particularly interesting about this function is that adding extra neurons was directly correlated to increased error. The reciprocal function only stopped improving after 3 neurons and, at times, performed worse. With this function, on the other hand, the prediction only got worse after 2 neurons, and did not recover the accuracy it achieved with 2 neurons in any trial.
Hence, we can conclude that predicting $g(x)$ is best done with 2 Neurons, and a no more, as this clearly results in a worse classifier.

4.3 Exponential Function

The Exponential Function, $h(x)$, was by far the most interesting visually. The network for this function performed extremely well with only a single hidden neuron, which was quite unexpected since this function is fairly similar to the reciprocal function, which did not perform as well with a single neuron. Furthermore, much like the common log, it peaked at 1 neuron, and then decayed after that almost completely; close to every classifier that used more than 1 neuron made completely erroneousness predictions – the error only evened out because of the averaging

Once again, prediction of $h(x)$ is best done with fewer neurons than more, just this time the effect of more is much worse than before, and we can conclude that using 2 neurons is both sufficient for accurate predictions of $h(x)$ and optimal within our domain.

4.4 Sine Wave

The Sine Wave, $l(x)$, was the most challenging for the classifier to correctly predict. As can be seen in Table 4 the error was very high for all numbers of hidden neurons. One interesting part to note is that this was the only function where 10 neurons resulted in the best results – 5 neurons also did very well, but shifted the sine wave up by a small factor. In Figure 8 we can see the best result was still not perfect, however, extra neurons did not change this; the classifiers were unable to get the curvature of the wave correctly. To accurately predict sine, it is most likely necessary to add an additional hidden layer to the network, though this was not verified.

5 Implementation

Experiments were conducted with a modified version of 'sinewave.py' and 'mlp.py', both from the textbook (Machine Learning: An Algorithmic Perspective, Marsland, 2015, 2nd Edition, Chapter 4).