## Section A

### Part 1 - Neural Network Classifier

Optimisations investigated initially included splitting the data set into a training set and a testing set. By separating the data so it's not testing on the data that it's training on, giving a more accurate representation of real world learning. Different ratios of training were investigated to determine the optimal amount of training. Furthermore, the number of iterations of learning and the learning rate were experimented with.

The performance of the system was more strongly impacted by the number of iterations than any other factor. While modifying the other factors made minor impacts, reducing the number of iterations from 100,000 to 10,000 resulted in a significant drop in correct readings, as seen in Table 1.1.

|  |  |  |
| --- | --- | --- |
| **# Samples** | **Train/Test ratio (%)** | **Results** |
| 10,000 | 100/100 | Total: 1496/4606, 32.48% |
| 100,000 | 100/100 | Total: 3572/4606, 77.55% |

**Table 1.1 - Number of samples at 0.01 Learning rate**

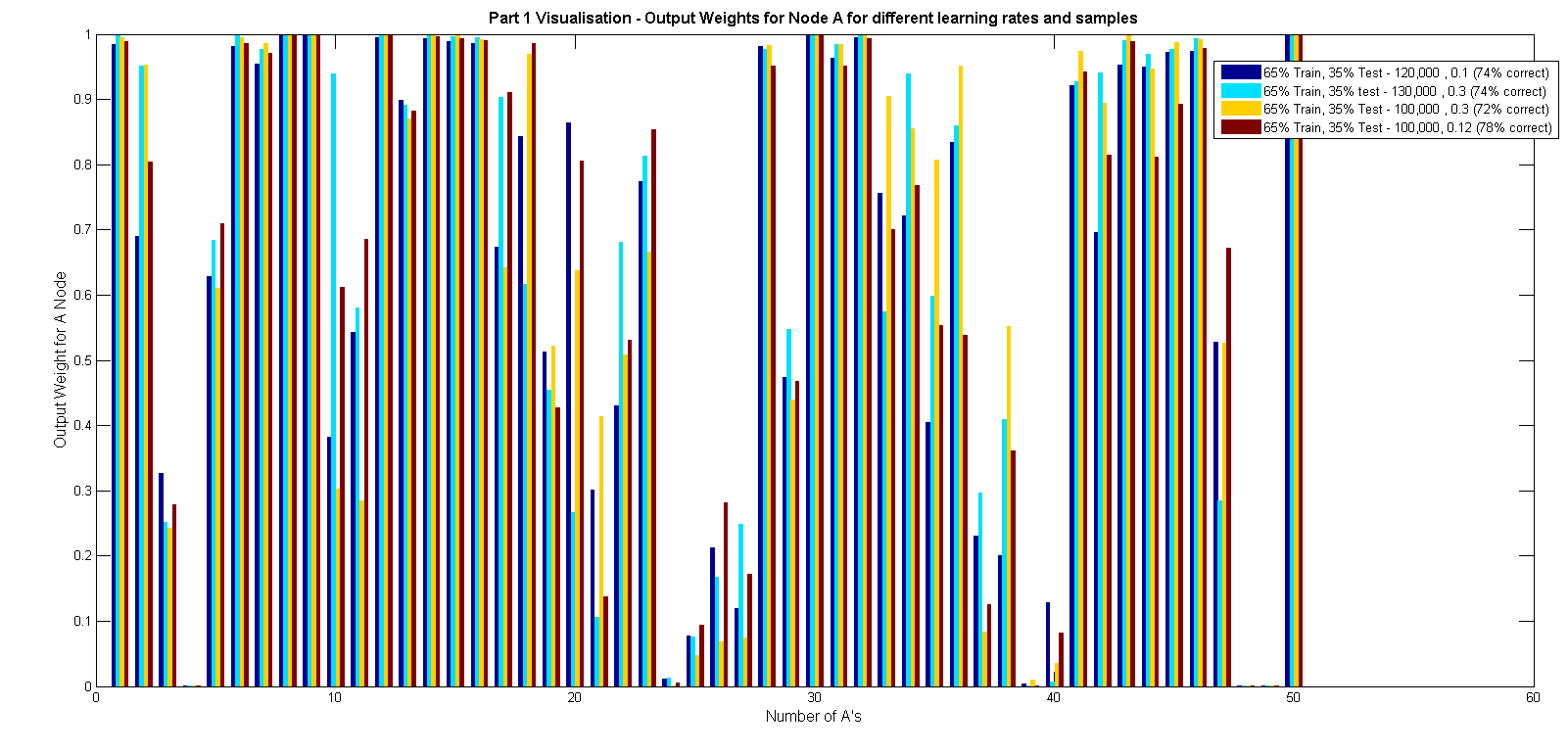
Optimal Learning rates were also investigated, and it was found that a higher learning rate is not the best choice. Once the learning rate reaches a threshold value, the number of correct identifications begins to decrease again. Through experimentation, the threshold was determined to be approximately 0.12. Different values tested are recorded in Table 1.2. The table also demonstrates how a significant increase in number of samples does not increase the percentage of correct identifications significantly enough to offset the training time increase.

|  |  |  |  |
| --- | --- | --- | --- |
| **# Samples** | **Learning Rate** | **Train/Test ratio (%)** | **Results** |
| 100,000 | 0.1 | 100/100 | Total: 4043/4606, 87.78% |
| 100,000 | 0.3 | 50/50 | Total: 1964/2303, 85.28% |
| 100,000 | 0.5 | 100/100 | Total: 3878/4606, 84.19% |
| 100,000 | 0.12 | 100/100 | Total: 4165/4606, 90.43% |
| 1,000,000 | 0.12 | 100/100 | Total: 4254/4606, 92.36% |
| 130,000 | 0.12 | 100/100 | Total: 4078/4606, 88.54% |
| 120,000 | 0.12 | 100/100 | Total: 4053/4606, 87.99% |
| 90,000 | 0.12 | 100/100 | Total: 4100/4606, 89.01% |

**Table 1.2 - Modifying the learning rate (0.01 can be seen in Table 1.1)**

Since using all the data to train, then testing on all the same data is a cheap and non-realistic method for real world applications, different ratios were tried and evaluated. Example ratios include 70/30, 50/50, 40/60, 60/40, 65/35.

After experimentation, it was decided that the following values are the optimal values:  
Number of samples: 100,000.  
Learning rate: 0.12.  
Train/Test ratio (%): 65/35.



**Figure 1.3 - Visualisation for output node A**

### Part 2 - ID3 Decision Tree

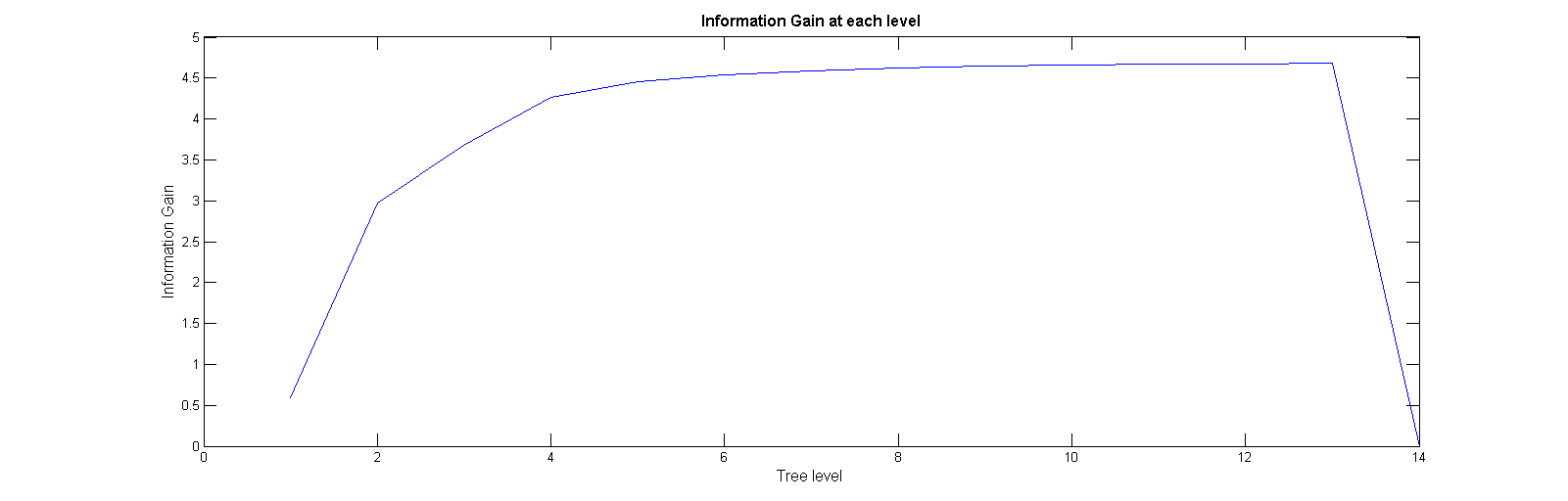
Optimisations investigated include using an ideal proportion threshold, number of samples and dividing the data set into training and testing data. Most changes to the proportion threshold and number of samples gave minimal change in performance. Through experimentation it was found that the best training to test ratio was 30% training/70% testing. Ratios that favoured training resulted in poorer performance, suggesting that the system was being overtrained. The results from using different ratios are documented in Table 2.1.

|  |  |  |  |
| --- | --- | --- | --- |
| **Ratio Test/Train (%)** | **Proportion Threshold** | **# Samples Threshold** | **Result** |
| 60/40 | 0.7 | 5 | 368/1842, 19.98% |
| 50/50 | 0.7 | 5 | 540/2303, 23.45% |
| 50/50 | 0.5 | 5 | 541/2303, 23.49% |
| 40/60 | 0.5 | 5 | 664/2763, 24.03% |
| 40/60 | 0.3 | 5 | 658/2763, 23.81% |
| 35/65 | 0.5 | 5 | 759/2993, 25.36% |
| 30/70 | 0.5 | 5 | 893/3224, 27.70% |
| 20/80 | 0.5 | 5 | 853/3684, 23.15% |

**Table 2.1 - Training with different ratios for the ID3 Decision Tree Classifier**

As can be seen in Table 2.1, the optimal training ratio was 30/70. Further optimisation came from experimenting with the proportion threshold and number of samples threshold, whilst using the optimal ratio. These values did not significantly increase performance however, with only a .3% increase being obtained.

The optimal values for the ID3 Decision tree are as follows:  
Ratio - 30/70  
Proportion Threshold - 0.4  
Samples Threshold - 2



**Figure 2.2 - Visualisation for information gain at each left child of the sub tree**

### Part 3 - Advanced Classifier

It was decided that the best option was to extend the Neural Network Classifier. The classifier itself is very similar to the provided Neural Network Classifier, however it stores an instance of a Multi-Layer Neural Network (MLNN), rather than a Single-Layer NN (SLNN). The implemented MLNN contains a single hidden layer, which takes the output of the original SLNN and applies extra weights and biases to calculate the output value. Information for backpropagation was sourced from the web, that described how to perform backpropagation in a MLNN with hidden layers. The error at each hidden unit is given by:  
  
**Figure 3.1 - Error at hidden unit 'k'**  
where hk(E) is the observed value at hidden node k, wki is the weight from the hidden node k to an output node i,  is the error at the output node.[1] The weights are adjusted similarly to the SLNN, but with an added layer of weights. See Figure 3.2 for the code.

Learning in the MLNN is achieved by calculating the errors at the output nodes and each hidden unit, and adjusting the weights accordingly. The visualisation provided shows how some of the weights change with each iteration of learning.

Figure 3.2 is a sample of the code that calculates the new weights at each step of the training.

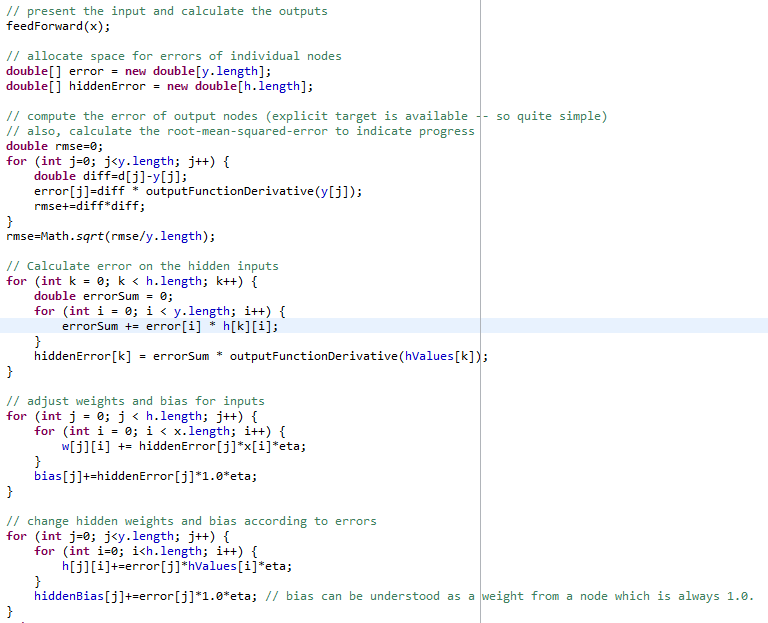
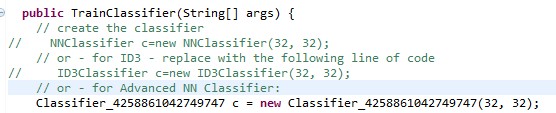
  
**Figure 3.2 - Training in the MLNN**

Figure 3.2 clearly demonstrates the formula in Figure 3.1 being applied, in order to calculate all the error values at the output and hidden nodes. The code is similar to examples given in the textbook by Jeff Heaton.[2]The code can be run exactly the same as the other NN Classifier. Figure 3.3 is a snippet of the code that creates the advanced classifier in TrainClassifier:  
  
**Figure 3.3 - The TrainClassifier class implementing the advanced classifier**

From there it is possible to split up the data however is most convenient and call the train method as with the NN Classifier.

The performance of the MLNN is marginally better than the performance of the SLNN and significantly better than that of the ID3 decision tree. Optimisations investigated include the learning rate, number of iterations of learning, the best ratio of training/testing and the inclusion of pre-processing. The MLNN only implements a single layer of hidden units. Due to time constraints, further layers were not investigated for optimality.

Without pre-processing, training time was generally under 5 seconds, with a correct guess rate around 65% for most iterations of learning/learning rates. The MLNN will achieve a much higher percentage of correct guesses than any other classifier at lower iterations of learning. Testing showed that with a 60/40 ratio and 0.4 learning rate, it took only 25000 iterations to achieve over 60% correctness. With 40000 iterations giving 65% correctness.

Table 3.4 documents optimal ranges for the MLNN to operate within. Any values outside these ranges will result in decreased performance. Furthermore, with pre-processing the MLNN will achieve at least 10% more accuracy during the testing phase. The pre-processing comes with a cost to training speed, however the increased correctness percentage justifies this downfall.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Ratio Test/Train (%)** | **Learning Rate** | **# Samples** | **Pre-processing** | **Results** |
| 90/10 | 0.15 | 40,000 | No | 232/460, 50.43% |
| 80/20 | 0.15 | 40,000 | No | 540/921, 58.63% |
| 60/40 | 0.15 | 40,000 | No | 1169/1842, 63.46% |
| 50/50 | 0.3 | 40,000 | No | 1493/2303, 64.83% |
| 40/60 | 0.5 | 50,000 | No | 1743/2763, 63.08% |
| 20/80 | 0.4 | 30,000 | No | 2225/3684, 60.40% |
| 10/90 | 0.4 | 40,000 | No | 2275/4145, 54.89% |
| 50/50 | 0.4 | 40,000 | Yes | 1753/2303, 76.12% |
| 40/60 | 0.15 | 50,000 | Yes | 2051/2763, 74.23% |
| **60/40** | **0.4** | **40,000** | **Yes** | **1416/1842, 76.87%** |

**Table 3.4 - Comparison of different input values for MLNN**

Table 3.4 shows great performance across the board over the different train/test ratios regardless of the number of samples and learning rate used. The optimal values are bolded in Table 3.4.

**Figure 3.5 - Visualisation for changing weights in the MLNN**

### Part 4 - Comparison of classifiers

The performance of each classifier was heavily dependent on the input values, however once an optimal value for each classifier had been determined, it was easy to compare the performance of the three. The ID3 classifier performed poorly in every aspect. It was the slowest classifier, and without pre-processing performed half as good as the other classifiers at best.

See the appendices for figures comparing the performance of each classifier with respect to different attributes and using different input values. It is clear that the best performing classifiers are the MLNN and the SLNN, with the MLNN having slightly better performance at different times.

## References

[1] Simon Colton, 2004, http://www.doc.ic.ac.uk/~sgc/teaching/pre2012/v231/lecture13.html

[2] Introduction to Neural Networks with Java, 1st Edition, Jeff Heaton, 2007