Table of Contents

Setting Up the Benchmark Neural Network & Datasets 1

Visual Representation of the Data 2

Experimentation 5

Experiment 1 – Selection of Data through Random Sampling 7

Experiment 2 – Selection of Dataset Size 8

Experiment 3 – Effects of the Number of Hidden Nodes 11

Experiment 4 – Effects of Changing the Learning Rate 15

Experiment 5 – Effects of the Number of Learning Cycles 18

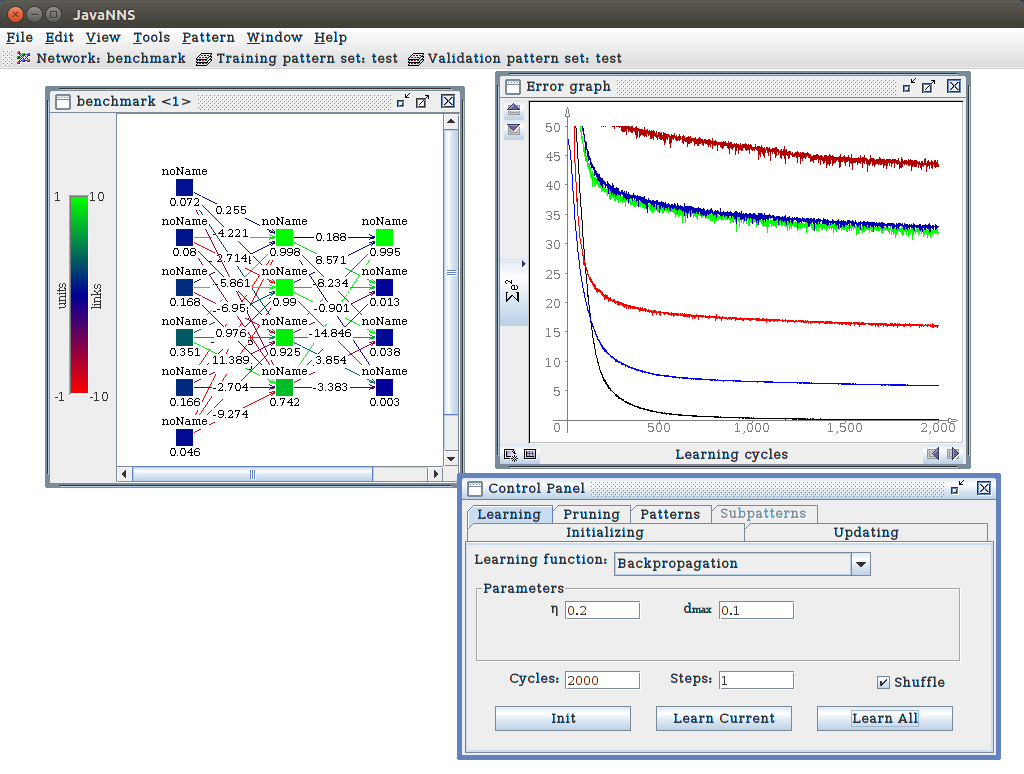
Investigation Into Imputation 22

## Setting Up the Benchmark Neural Network & Datasets

The first step to creating the neural network is formatting the data. The data in its original form is not optimised for use in a neural network, as the range of data is not appropriate. The data firstly needs normalising, taking into account the minimum and maximum values. As there is no obvious relationship between the parameters, each parameter will be normalised individually.

Secondly, the data needs to be divided into training and testing sets. For the sake of setting a benchmark, the initial size of the training set will be 200 samples, and the size of the test set will be 200 samples. For each of the sets 50 values will need to be taken from each of the classes. In order to ensure an even distribution of data, each of the classes will be randomly sorted using the 'sort –R' command in Linux.

To set up the neural network there will be 6 inputs nodes (as there are 6 parameters in the data), 4 hidden nodes (purely as a random starting value) and 4 output nodes (as there are four classes to identify). The learning rate  will be set to 0.2 and the number of cycles will be set to 2000, these are picked at random.



**Initial neural network Setup**

Number of hidden nodes: 4

Learning rate - : 0.2

dmax : 0.1

Number of cycles: 2000

Shuffle patterns: On

**Initial Data Samples**

200 training samples

200 testing samples

### Visual Representation of the Data

The graphs below compare each parameter across all of the classes.

#### Discussion of Graphs

By comparing the classes by parameter, we can see how each class truly differentiates. We can also, with some accuracy, predict how the neural network will perceive the data. A major issue with the data itself is outliers. These outliers may be a true representation of a parameter, however if these outliers share the same space as the same parameter in another class, it can cause confused results.

By surveying the classes in the graphs above, we can see that class 1 has the most tightly packed data points. The data points rarely deviate from the average range. However, classes 2, 3 and 4 have outlying points that deviate uncharacteristically from their main cluster group, often crossing over into other areas associated with other classes.

Ultimately, the data defines how well a neural network will perform. The data above is suitable, each parameter within a class has its own definable range. There are areas where certain parameters seem to totally merge with others, making it extremely difficult to identify. However, neural networks take into account all parameters within a class e.g. classes 2 and 3 are extremely muddled in some areas of parameter D, though in parameter F they can be easily separated. This is where neural networks excel, finding patterns in multidimensional space.

## Experimentation

This section focuses on modifying properties of the neural network, and explores the outcomes. The performance of a neural network relies heavily upon the datasets it uses to learn, but also the configuration of the network itself. A series of experiments will be implemented to explore the affects that these properties have, and whether they are beneficial or detrimental to the performance of the neural network.

The main variables that affect a neural network's performance have been identified as:

* **Data Samples**

Selection or randomness of data

Data sample range

Sample size

* **Neural Network Configuration**

Number of hidden layers

Learning rate

Learning cycles

Using the variables above, the experiments have been split into two categories - Neural network configuration & input data.

#### Initial Benchmark & Collecting Results

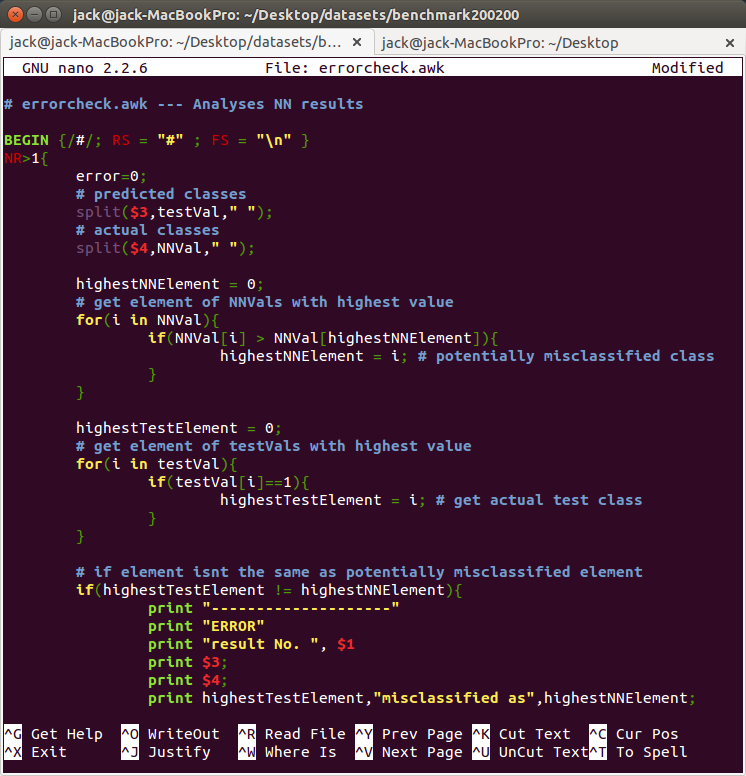
**Misclassification Matrix**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 50 | 0 | 0 |
| Class 3 | 1 | 2 | 47 | 0 |
| Class 4 | 1 | 0 | 1 | 48 |

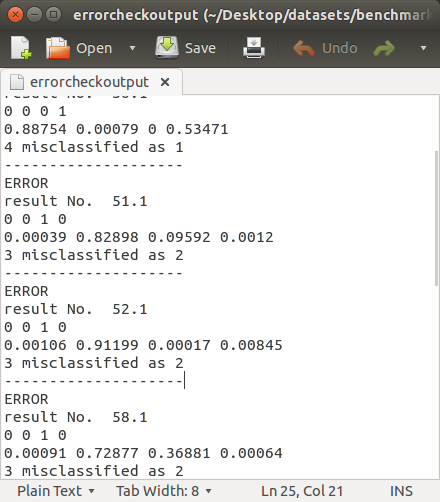
5 misclassifications

A misclassification matrix is a method of visually representing the accuracy of a neural network. The above matrix shows how the initial benchmark performed. Already, predictions made in previous sections are true.

In order to collect the results accurately and efficiently, the process needs to be automated. To do this a small .awk script has been developed to count the number of misclassifications found in the results file generated by JNNS.



The misclassifications that have been identified can be easily counted using search functionality found in most text processors.



### Experiment 1 – Selection of Data through Random Sampling

This experiment will focus on the effects of the actual selection of data to train the neural network. 4 sets of training data (datasets of 200 values) will be randomly picked from the entire dataset. Another 4 random sets of data will be used as testing sets. This will show whether the selection of data is important to the performance of the neural network.

Each dataset will consist of 200 randomly selected samples from the original dataset. This experiment will show the effects of the selection of data. If the results show considerably different levels of performance associated with each dataset then the data selection process is proven to be an important factor.

**Expected outcome:**

There will be little variance between the number of errors generated.

**Outcome:**

Dataset 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 45 | 4 | 1 |
| Class 3 | 1 | 4 | 45 | 0 |
| Class 4 | 1 | 0 | 1 | 48 |

12 misclassifications

Dataset 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 49 | 1 | 0 | 0 |
| Class 2 | 0 | 49 | 1 | 0 |
| Class 3 | 0 | 7 | 43 | 0 |
| Class 4 | 0 | 0 | 0 | 50 |

9 misclassifications

Dataset 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 1 | 42 | 7 | 0 |
| Class 3 | 0 | 3 | 47 | 0 |
| Class 4 | 1 | 0 | 1 | 48 |

13 misclassifications

Dataset 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 49 | 0 | 1 |
| Class 3 | 0 | 10 | 40 | 0 |
| Class 4 | 0 | 0 | 0 | 50 |

11 misclassifications

**Discussion of Outcome**

The tests above show just how important it is to trial different sets of data before committing to a certain neural network configuration. In order for a neural network to effectively classify sets of data, the training set itself needs to truly represent these characteristics.

### Experiment 2 – Selection of Dataset Size

This experiment will look at the effects of dataset size. There will be 6 sets of data of varying sizes. To measure the effectiveness of each dataset, the testing set will remain the same as the initial benchmark. Care has been taken to ensure that samples inside the training sets are not present in the test set.

Set sizes that will be used for experimentation:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Dataset 1 | Dataset 2 | Dataset 3 | Dataset 4 | Dataset 5 |
| Training set size | 64 | 100 | 300 | 500 | 600 |
| Testing set  size | 200 | 200 | 200 | 200 | 200 |

**Expected outcome:**

Datasets that are too small will result in the neural network being unable to learn the necessary patterns to be able distinguish classes from each other. This means the neural network will effectively become too specialised in a certain set of data. As with any dataset, there is the risk that it won’t represent characteristics of the real data, this is only intensified with smaller datasets.

It is doubted that the largest dataset in this experiment will have adverse affects on the performance of the neural network, as it still isn’t large enough.

**Outcome:**

Dataset 1: 64 samples (16 per class)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 47 | 3 | 0 |
| Class 3 | 0 | 9 | 41 | 0 |
| Class 4 | 0 | 0 | 1 | 49 |

13 misclassifications

Dataset 2: 100 samples (25 per class)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 50 | 0 | 0 |
| Class 3 | 1 | 7 | 42 | 0 |
| Class 4 | 0 | 0 | 0 | 50 |

8 misclassifications

Dataset 3: 300 samples (75 per class)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 50 | 0 | 0 |
| Class 3 | 1 | 7 | 42 | 0 |
| Class 4 | 1 | 0 | 0 | 49 |

9 misclassifications

Dataset 4: 500 samples (125 per class)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 50 | 0 | 0 |
| Class 3 | 1 | 6 | 43 | 0 |
| Class 4 | 0 | 0 | 0 | 50 |

7 misclassifications

Dataset 5: 600 samples (150 per class)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 49 | 1 | 0 |
| Class 3 | 0 | 6 | 44 | 0 |
| Class 4 | 0 | 0 | 0 | 50 |

7 misclassifications

**Discussion of Outcome**

The most notable outcome of the experiment above is the first test (14 misclassifications), where the training set size is insufficient to successfully classify the data. Once a sufficient amount of data has been provided for the training set there seems to be little improvement in terms of performance. One conclusion from this experiment could be that a neural network can perform fairly effectively with small training sets, assuming that it represents the characteristics of the dataset.

**Neural Network Configuration Experimentation**

### Experiment 3 – Effects of the Number of Hidden Nodes

This experiment studies the effects of the number of hidden nodes within the network. The datasets used in this experiment are the same as the initial benchmark datasets.

**Expected outcome:** It is difficult to approximate the correct number of hidden nodes. This is because the nodes are deeply involved with the problem itself. A typical outcome is that an insufficient amount of nodes will affect the accuracy of the neural network and a larger amount will cause noticeable overhead. Another possible issue is that the network will lose its ability to generalise if too many nodes are added.

**Outcome:**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Test 1 | Test 2 | Test 3 | Test 4 | Test 5 | Test 6 | Test 7 | Test 8 | Test 9 | Test 10 |
| No. of hidden nodes | 2 | 6 | 8 | 10 | 12 | 14 | 20 | 40 | 100 | 200 |

Test 1: 2 hidden nodes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 48 | 2 | 0 |
| Class 3 | 1 | 5 | 44 | 0 |
| Class 4 | 1 | 0 | 1 | 48 |

10 misclassifications

Test 2: 6 hidden nodes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 49 | 1 | 0 |
| Class 3 | 2 | 5 | 43 | 0 |
| Class 4 | 1 | 0 | 1 | 48 |

10 misclassifications

Test 3: 8 hidden nodes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 50 | 0 | 0 |
| Class 3 | 1 | 7 | 42 | 0 |
| Class 4 | 1 | 0 | 1 | 48 |

10 misclassifications

Test 4: 10 hidden nodes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 48 | 2 | 0 |
| Class 3 | 1 | 6 | 43 | 0 |
| Class 4 | 1 | 0 | 0 | 49 |

10 misclassifications

Test 5: 12 hidden nodes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 49 | 1 | 0 |
| Class 3 | 1 | 6 | 43 | 0 |
| Class 4 | 1 | 0 | 1 | 48 |

10 misclassifications

Test 6: 14 hidden nodes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 48 | 1 | 1 |
| Class 3 | 1 | 6 | 43 | 0 |
| Class 4 | 1 | 0 | 0 | 49 |

10 misclassifications

Test 7: 20 hidden nodes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 1 | 47 | 2 | 0 |
| Class 3 | 1 | 6 | 43 | 0 |
| Class 4 | 1 | 0 | 0 | 49 |

11 misclassifications

Test 8: 40 hidden nodes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 48 | 2 | 0 |
| Class 3 | 1 | 6 | 43 | 0 |
| Class 4 | 1 | 0 | 0 | 49 |

10 misclassifications

Test 9: 100 hidden nodes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 49 | 1 | 0 |
| Class 3 | 1 | 6 | 43 | 0 |
| Class 4 | 1 | 0 | 0 | 49 |

9 misclassifications

Test 10: 200 hidden nodes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 50 | 0 | 0 |
| Class 3 | 1 | 8 | 41 | 0 |
| Class 4 | 1 | 0 | 0 | 49 |

10 misclassifications

**Discussion of Outcome**

The experiment proved the unpredictability of hidden nodes. The number of hidden nodes is only applicable to a problem itself. The results above showed that the number of nodes had little effect on the accuracy of the neural network.

One unrecorded outcome was the effect the number of hidden nodes had on the time the neural network took to learn. The larger the number of hidden nodes the longer the network took to train. In systems that handle massively high dimensional data it is understandable why networks can take large amounts of time to train.

### Experiment 4 – Effects of Changing the Learning Rate

This experiment will study the effects caused by altering the learning rate. The learning rate defines the speed in which the network will learn.

**Expected outcome:** If the learning rate is too low, the network will be unable to learn the needed patterns.

**Outcome:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Test 1 | Test 2 | Test 3 | Test 4 | Test 5 | Test 6 | Test 7 |
|  | 0.0025 | 0.025 | 0.1 | 0.4 | 0.8 | 1.6 | 3.2 |

Test 1: 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 3 | 23 | 16 | 8 |
| Class 3 | 2 | 3 | 44 | 1 |
| Class 4 | 3 | 1 | 1 | 45 |

38 misclassifications

Test 2: 0.025

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 1 | 48 | 1 | 0 |
| Class 3 | 1 | 7 | 42 | 0 |
| Class 4 | 0 | 1 | 0 | 49 |

11 misclassifications

Test 3: 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 49 | 1 | 0 |
| Class 3 | 1 | 6 | 43 | 0 |
| Class 4 | 1 | 0 | 3 | 46 |

12 misclassifications

Test 4: 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 48 | 2 | 0 |
| Class 3 | 1 | 6 | 43 | 0 |
| Class 4 | 1 | 0 | 1 | 48 |

11 misclassifications

Test 5: 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 46 | 2 | 2 |
| Class 3 | 1 | 6 | 43 | 0 |
| Class 4 | 1 | 0 | 1 | 48 |

13 misclassifications

Test 6: 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 47 | 2 | 1 |
| Class 3 | 1 | 6 | 43 | 0 |
| Class 4 | 1 | 0 | 0 | 49 |

11 misclassifications

Test 7: 

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 48 | 2 | 0 |
| Class 3 | 1 | 6 | 43 | 0 |
| Class 4 | 1 | 0 | 1 | 48 |

11 misclassifications

**Discussion of Outcome**

The outcome achieved was expected, the lower the learning rate, the longer it took to learn the necessary pattern. In the case of test 1, it did not learn very much at all as it misclassified more than an eighth of the dataset.

### Experiment 5 – Effects of the Number of Learning Cycles

This experiment will study the effects of changing the number of learning cycles. The number of learning cycles defines how long the network will be trained.

**Expected outcome:**

If the network is not trained for long enough, the network will not be informed of the problem, and will not be able to classify data accurately. On the other hand, if there are too many learning cycles, the network will become over-trained, and technically learn the training set, not the relationships in the data, making it unable to generalise.

**Outcome:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Test 1 | Test 2 | Test 3 | Test 4 | Test 5 | Test 6 | Test 7 |
| No. of learning cycles | 50 | 250 | 500 | 1000 | 4000 | 8000 | 16000 |

Test 1: 50 learning cycles

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 3 | 43 | 1 | 3 |
| Class 3 | 2 | 18 | 29 | 1 |
| Class 4 | 1 | 0 | 0 | 49 |

29 misclassifications

Test 2: 250 learning cycles

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 50 | 0 | 0 |
| Class 3 | 1 | 7 | 42 | 0 |
| Class 4 | 1 | 0 | 0 | 49 |

9 misclassifications

Test 3: 500 learning cycles

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 50 | 0 | 0 |
| Class 3 | 1 | 7 | 42 | 0 |
| Class 4 | 0 | 1 | 1 | 48 |

10 misclassifications

Test 4: 1000 learning cycles

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 50 | 0 | 0 |
| Class 3 | 1 | 7 | 42 | 0 |
| Class 4 | 1 | 0 | 0 | 49 |

9 misclassifications

Test 5: 4000 learning cycles

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 46 | 2 | 2 |
| Class 3 | 2 | 5 | 43 | 0 |
| Class 4 | 1 | 0 | 1 | 48 |

13 misclassifications

Test 6: 8000 learning cycles

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 44 | 2 | 4 |
| Class 3 | 1 | 6 | 43 | 0 |
| Class 4 | 1 | 0 | 1 | 48 |

15 misclassifications

Test 7: 16000 learning cycles

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 2 | 45 | 2 | 1 |
| Class 3 | 2 | 4 | 44 | 0 |
| Class 4 | 1 | 0 | 1 | 48 |

13 misclassifications

**Discussion of Outcome**

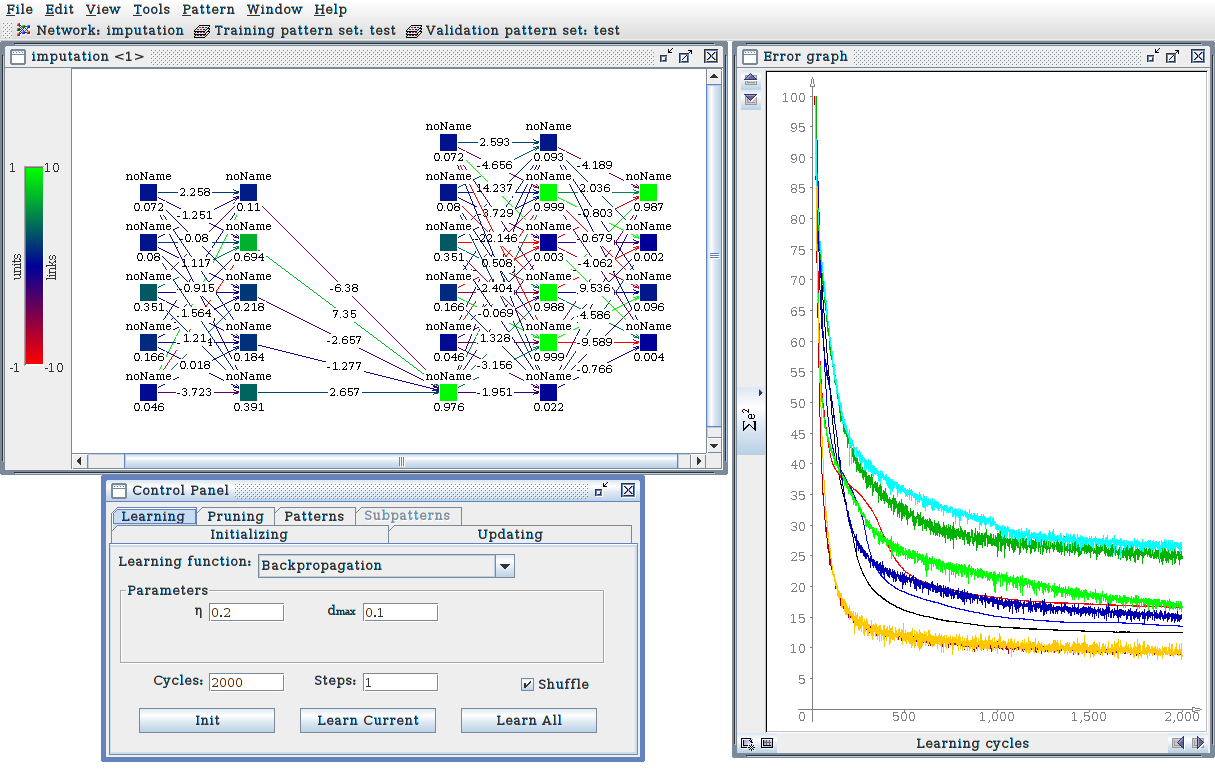
As expected, the experiment showed a gradual increase in misclassifications, as the number of learning cycles increased. However, the most severe results were created by test 1, where there were a huge number of misclassifications due to the small number of learning cycles. Test 2 showed that a neural network copes fairly well with a relatively small number of learning cycles. The number of cycles in test 2 were some of the best, and it used the second smallest set of cycles in the experiment.

## Investigation Into Imputation

This section explores the technique of imputation. This technique is primarily used to compensate for missing parameter data in a dataset. In a reality data collection doesn’t always go to plan, and certain parameters in the data are either corrupted, or missing all together.

Imputation is a technique designed to overcome this situation. The technique s originally derived from statistics, using the data that we already have to generate a value for the missing data. This technique is considered to provide a good guess of the missing values.

To configure an imputation network the network must be set up as usual, then a second block of inputs have to be created. These inputs take the same values as the original network, however it feeds its values into the original network. What this essentially does is learn a relationship based upon the known parameters. The network can then use this relationship as a way of substituting the missing parameter.

****

**Experimentation**

To test the effectiveness of the imputed network it will be compared to the benchmark network created at the beginning of the testing. The learning rate and cycles will also be the same as the benchmark. The datasets will also be the same.

A test will be run for each of the missing parameters.

Missing Parameter - A

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 50 | 0 | 0 |
| Class 3 | 1 | 7 | 42 | 0 |
| Class 4 | 1 | 0 | 1 | 48 |

10 misclassifications

Missing Parameter - B

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 49 | 1 | 0 |
| Class 3 | 1 | 6 | 43 | 0 |
| Class 4 | 1 | 0 | 2 | 47 |

11 misclassifications

Missing Parameter - C

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 1 | 47 | 2 | 0 |
| Class 3 | 2 | 5 | 38 | 5 |
| Class 4 | 1 | 1 | 11 | 37 |

27 misclassifications

Missing Parameter - D

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 4 | 46 | 0 | 0 |
| Class 3 | 0 | 8 | 42 | 0 |
| Class 4 | 2 | 0 | 0 | 48 |

14 misclassifications

Missing Parameter - E

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 49 | 1 | 0 |
| Class 3 | 1 | 7 | 42 | 0 |
| Class 4 | 1 | 0 | 0 | 49 |

10 misclassifications

Missing Parameter - F

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Predicted | | | |
|  |  | Class 1 | Class 2 | Class 3 | Class 4 |
| Actual Result | Class 1 | 50 | 0 | 0 | 0 |
| Class 2 | 0 | 48 | 0 | 2 |
| Class 3 | 1 | 7 | 41 | 1 |
| Class 4 | 0 | 3 | 0 | 47 |

14 misclassifications

**Outcome:**

The imputation seems to have worked to a certain degree, the network has successfully created a substitute value for each of the classes. However, there is a definite impact on accuracy of classification.

One notable outcome was when parameter C was imputed. This shows that imputation techniques are definitely more suited to certain datasets. By looking at the graphs at the beginning of the document, parameter C has a fairly chaotic placement of data points across all classes. This could explain the poor classification of class 4.

Imputation could be deemed successful if the impact on accuracy is acceptable. In reality the data may be so rare that there is no other option but to create an imputation network.

The imputation technique shown above could feasibly be expanded, so that it could tackle two missing parameters. However, the main issue with this is that the network could become unwieldy, with accuracy of classification possibly being affected even further.

# Critical Review of Neural Networks

**Introduction**

This essay is a critical review of neural networks. The essay will consider the practical applications of neural networks and examine why they are suitable. The essay will generally focus on neural networks that make use of back propagation, or multi layer perceptron networks.

1. A Neural Network Explained

A neural network essentially mimics the characteristics of the brain, most notably, the neurons. A brain holds a vast amount of neurons, all of which are interconnected to create a network. What neurons do is help the brain to recognise patterns, and provide the ability to generalise (roughly recognise the difference between objects). Its this generalisation that makes neural networks good for classification problems.

A back propagated artificial neural network simulates neural activity by making use of neurons (also known as nodes) and placing them into their own specific layers. When constructing an artificial neural network, there are similar characteristics to its biological counterpart.

An input layer simulates a stimulus - a collection of neurons that are suited to a type of class. The input layer has as many nodes that are needed to successfully take into account all of the parameters provided by the input class. Then there is the hidden layer. The hidden layer is arguably the most important of the layers. The nodes in the hidden layer are fed information by the input layer, at which point pre-defined weightings take an effect. Weights are calculated based upon the data that is used to train the network (more on this in the next section). Finally an output layer, consisting of as many nodes as there are classes to define, is fed yet more information. This is then used to signal the correct output node, and therefore signify which class was passed into the network at the input layer.

This is where neural networks excel - classifying data.

1. Setting up an Artificial Neural Network

This section explores the techniques behind setting up an MLP neural network and the potential pitfalls associated with the various steps.

* 1. Preparing the Dataset
* Potentially the most important step in creating a neural network is selecting the data. The neural network will need data not only to train it, but to also for testing. The datasets used must truly represent the testing data i.e. the datasets need to have the characteristics that are actually being classified. This seems like an obvious point, however it is fairly difficult to select a dataset that truly represents the entire classification problem, especially when the data is multidimensional. There is also the issue of anomalies in the data, where the training data has entries that are unusual for the class. It is up to the discretion of the individual, how to handle this data. There are many techniques to circumvent this issue, however the most productive is usually a matter of trial and error – training the network, testing it, retraining it with different data and comparing results from the two runs.
  1. Configuring the Network
* The network itself relies on many variables that affect its level of performance. The number of training cycles, learning rate, and number of hidden nodes are just a few of the issues that will impede performance. Again there is very little in the way of calculating an optimal solution e.g. the number of hidden nodes is very dependent on the problem itself, and is very hard to pin down – a network with 100 hidden nodes may misclassify just as many classes as a network with 20, however it may perform awfully with 50 hidden nodes.
* This section has shown that it is in fact, quite a lengthy process to set up a functioning neural network. Another issue to take into account is that even when the network is functioning its ability to classify data is only so accurate.

1. Practical Applications of NN

Neural networks have a widespread number of applications, most notably automated classification problems (e.g. identifying species). There are also other applications beyond simple classification issues that neural networks can solve. As explained by Taylor, J.G. (1993), neural networks can also be used for the following applications:

* 1. Time Series Prediction
* Neural networks have applications in finance. Neural networks can find patterns and non-linear relationships in market data. Neural networks can also make accurate forecasts of future financial trends, an invaluable tool for stockbrokers, or large businesses.
* However, is it wise to rely heavily on such a simple system. As shown above Neural networks are inaccurate to a certain degree. It is probably better to use these systems as a guide, rather than the absolute rule.
  1. Steering of autonomous vehicles
* There have been many advances in autonomous vehicle guidance over the last decade. According to Bose, N.K. and Liang (1996), initial attempts in vehicle guidance proved unsuccessful, due to their pure mathematical-based approaches. Since neural networks have been used for this application, performance has improved vastly. However, as stated in section 2.2, neural networks aren’t correct 100% of the time. Google’s self driving car is highly sophisticated, using many forms of machine learning. The system takes training data from real driving (Madrigal, 2014), even this doesn’t stop the system from making mistakes (Gibbs, 2016). This was shown recently when a self driving car collided with another vehicle.
* It is left to society to weigh the pros and cons of this technology. Driverless car technology could be one of the largest breakthroughs in the next decade, simply ignoring the power of neural networks, due to their safety would be unwise. It has been predicted that technology similar to Google’s self-diving car could potentially reduce fuel consumption, road accidents, and congestion (Spectrum.ieee.org, 2015). Fault detection and warning systems are imperative for this kind of system.
  1. Detection of Explosives
* An experiment in brazil was conducted to test the effectiveness of using a neural network to detect illicit items . The network was designed to be embedded in a system for use at an airport. According to Ferreira, Crispim and Silva (2009), the system performed effectively, successfully classifying 97% of the items. Drugs and explosives were included in the selection of items. If this system was adopted for use in airports it must be decided if the level of accuracy is suitable. According to the paper, a 3% chance of misclassifying the item puts the system on par with existing systems.

These are just a few of the applications of neural networks, but as the list progresses you can see the applications become applied to tasks that are highly involved with health and safety.

**Conclusion**

A neural network isn’t ideal for every problem. Areas where there needs to be a ‘real-time’ response and the ability to generalise, is where a neural network excels. In all of the applications discussed there has been the potential of noisy data. The financial forecasting issue is a prime example of this. In safety-critical environments, neural networks should be used as part of a chain of systems, where there is some form of contingency.