**Machine Learning Coursework 1 Report**

**Group B members:**

|  |  |
| --- | --- |
| Name | Student ID |
| Koo Wai Kun | 025254 |
| Darren Cheng Sum Jun | 025299 |
| Ong Kok Ting | 025288 |
| Loh Jin Xian | 016763 |

**Table of content**

|  |
| --- |
| Code explanation for different problems |
| 1. Binary classification  1.1 Parameters used  1.2 Classification result for each fold  1.3 Average accuracy |
| 2. Multi-classification  2.1 Parameters used  2.2 Classification result for each fold  2.3 Average accuracy |
| 3. Regression  3.1 Parameters used  3.2 Regression result for each fold  3.3 Lowest root mean squared error |
| Actions taken to prevent overfitting |
| Conclusion |

**Code explanation for different problems**

1. **Binary Classification**

**1.1 Parameters used**



Figure 1a. Network building code

A feed-forward backpropagation network is used as it is suitable for supervised learning due to it requiring the derivative of the loss function, where a target value is known.

As for the training function, The Levenberg-Marquardt, or ‘*trainlm‘* is used. It is used due to it being one of the fastest backpropagation algorithm available. It is also designed to work with loss functions.

For the learning function, Gradient Descent, or ‘*learngd’* is used. It optimizes the error of the cost function, while backpropagation computes how the error changes as the weights in the neurons are changed.

The number of epochs is set at 150 to prevent overfitting or underfitting.

**1.2 Classification result for each fold**

To validate the network built, k-fold cross validation where k=10 is done.

The data is partitioned into 10 folds with the *cvpartition()* function.

|  |  |  |
| --- | --- | --- |
| Fold(k) | Performance | Accuracy (%) |
| 1 | 0.0747 | 86.67 |
| 2 | 0.1044 | 86.67 |
| 3 | 0.0747 | 93.33 |
| 4 | 0.0457 | 93.33 |
| 5 | 0.0311 | 93.33 |
| 6 | 0.1071 | 93.33 |
| 7 | 0.0727 | 93.33 |
| 8 | 0.2172 | 73.33 |
| 9 | 0.0977 | 93.33 |
| 10 | 0.1210 | 80 |

Table 1a. Classification results per cross-validation fold

**1.3 Average accuracy**

The average accuracy of the k-fold cross validation is **88.67%**

1. **Multi-Classification**

**2.1 Parameters used**

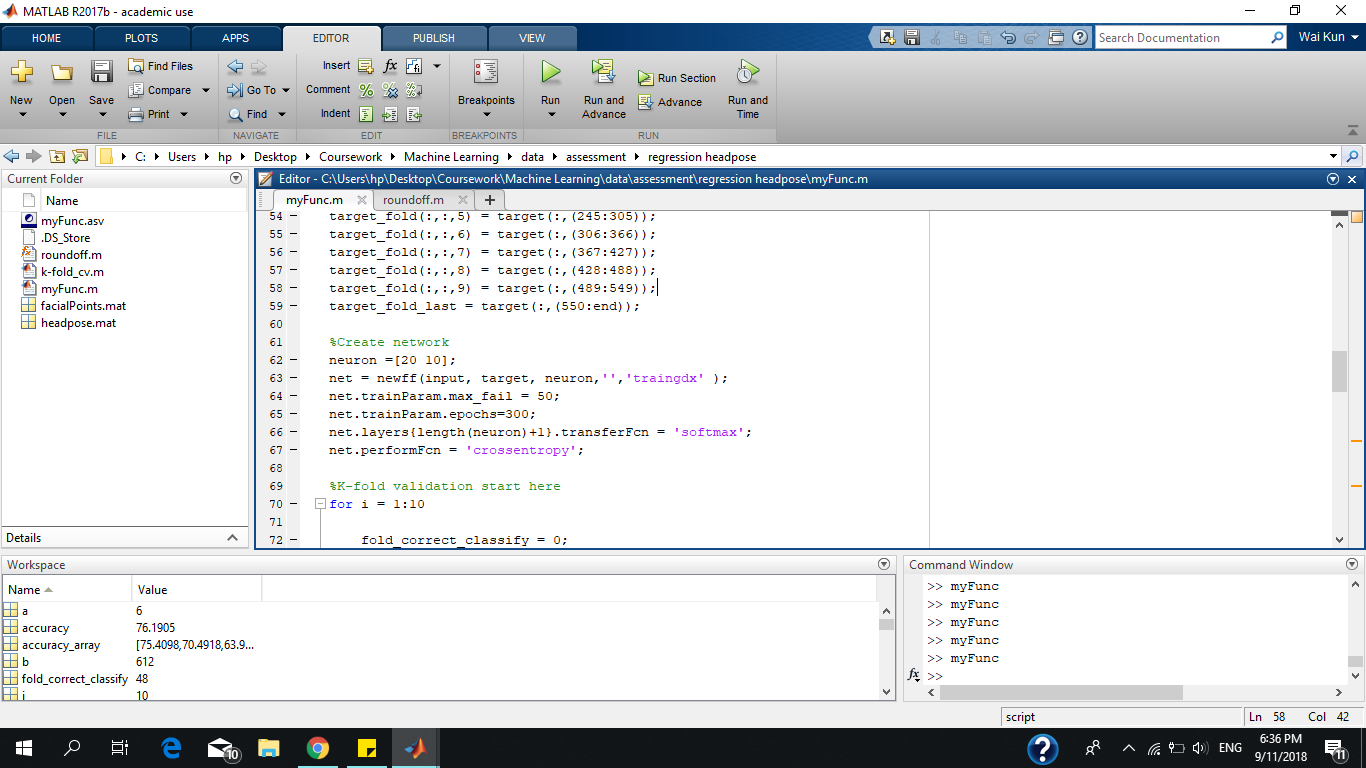


Figure 2a. Parameters setting for multi-classification problem

*‘traingdx’* is used because it yields better accuracy over other training functions.

|  |  |
| --- | --- |
| **Training function** | **Average accuracy** |
| traingdx | 74.5098 |
| trainrp | 72.0588 |
| trainscg | 71.8954 |

Table 2a. Accuracy obtained from different training functions

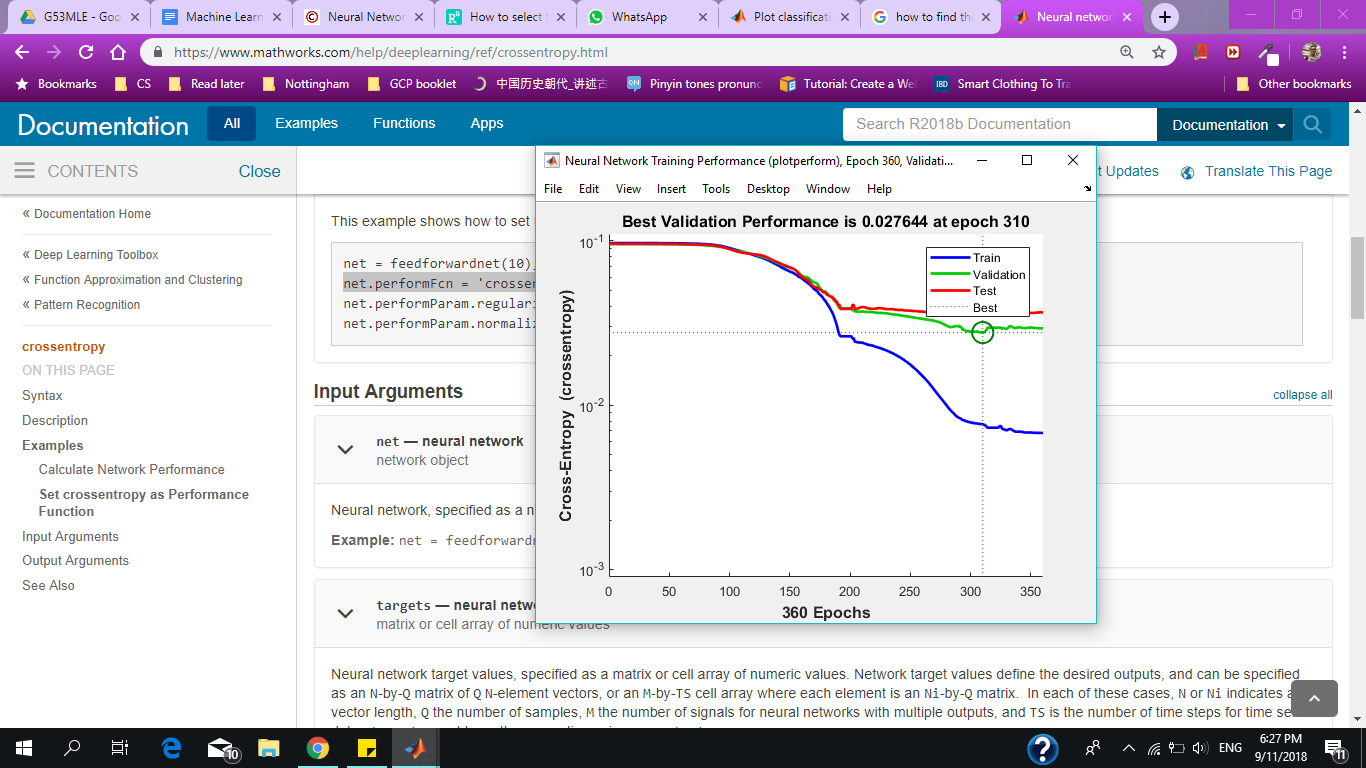
By keeping the *‘trainlm’* fixed, the learning function is changed to compare the result.

|  |  |
| --- | --- |
| **Learning function** | **Average accuracy** |
| learngdm (default) | 74.5098 |
| learngd | 66.1765 |

Table 2b. Accuracy obtained from different learning functions

The transfer function for the hidden layer is changed to ‘softmax’ because ‘softmax’ works better with multiclass classification problem while a sigmoid function is used for 2-class classification problem. Softmax classifier is a linear classifier that uses cross entropy loss function, thus we use ‘crossentropy’ as the performance function.

2 hidden layer is used with the topology of [20 10]. Additional layers or number of neurons will cause the training to overfit. Validation check is set to 50 to avoid stopping the training process too soon. Epoch size is set to 300 as Test error starts to increase and training error decreasing.

Figure 2b. Performance graph for 500 epochs 

**2.2 Classification result for each fold**

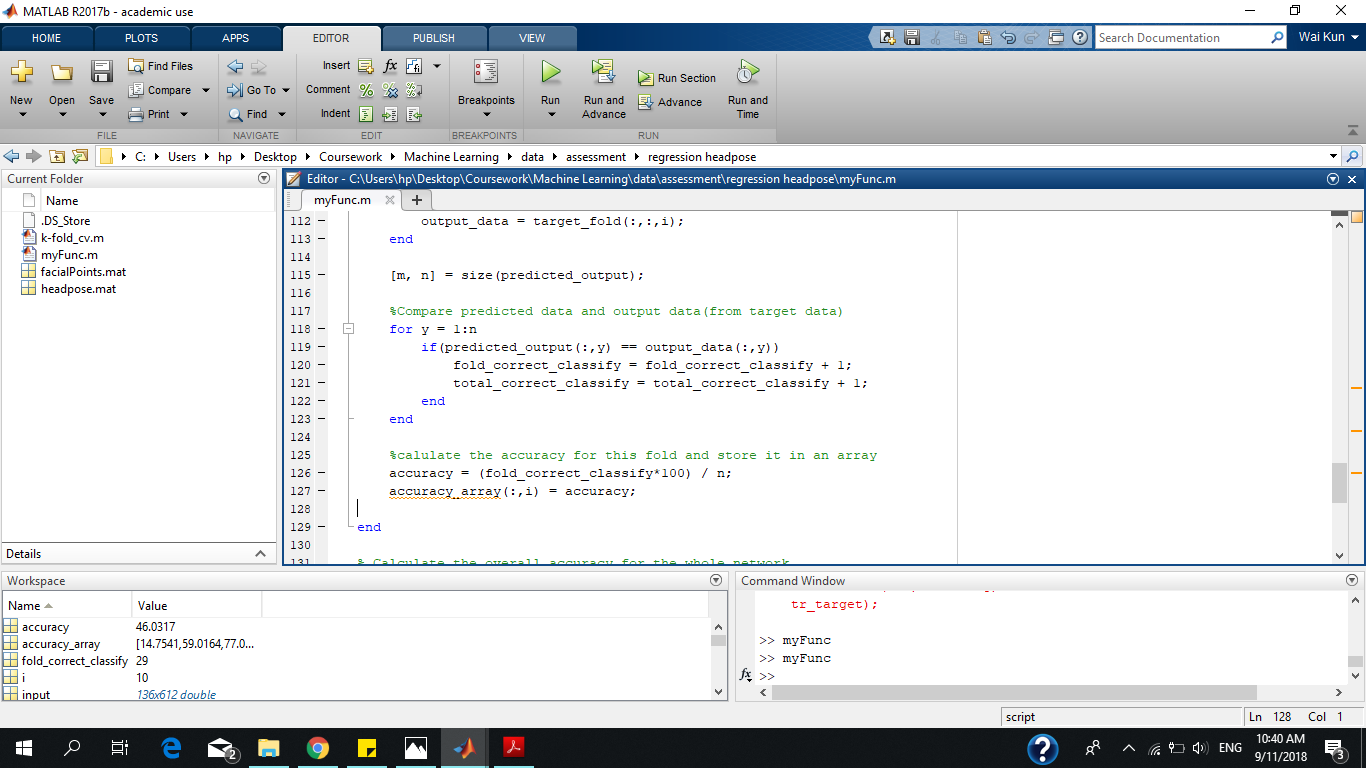


Figure 2c. Calculation for accuracy

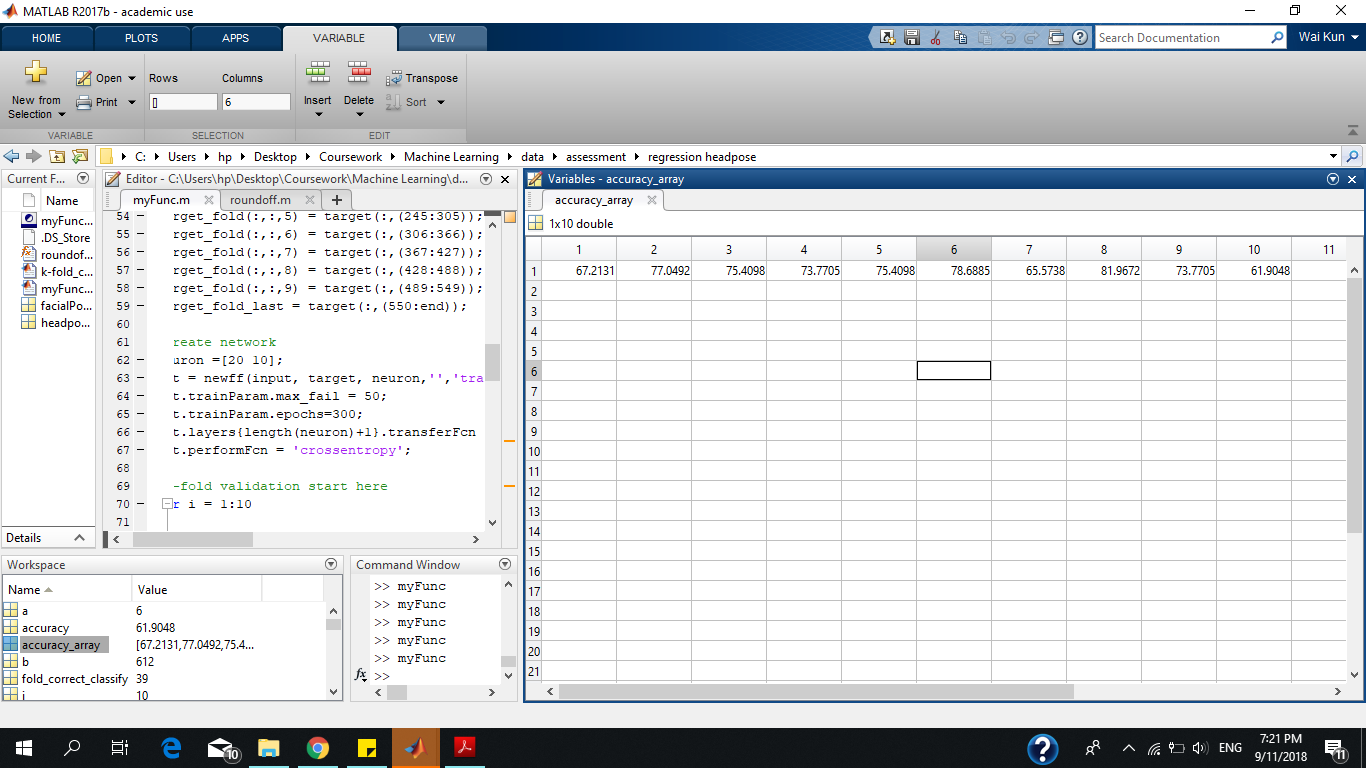


Figure 2d. Accuracy array table

‘acccuracy\_array’ stored the percentage of accuracy for all 10 folds, which can then be read from the workspace.

|  |  |
| --- | --- |
| **Fold(k)** | **Accuracy (%)** |
| 1 | 67.21 |
| 2 | 77.05 |
| 3 | 75.41 |
| 4 | 73.77 |
| 5 | 75.41 |
| 6 | 78.69 |
| 7 | 65.57 |
| 8 | 81.97 |
| 9 | 73.77 |
| 10 | 61.90 |

Table 2c. Average accuracy obtained from different training function

**2.3 Average accuracy**

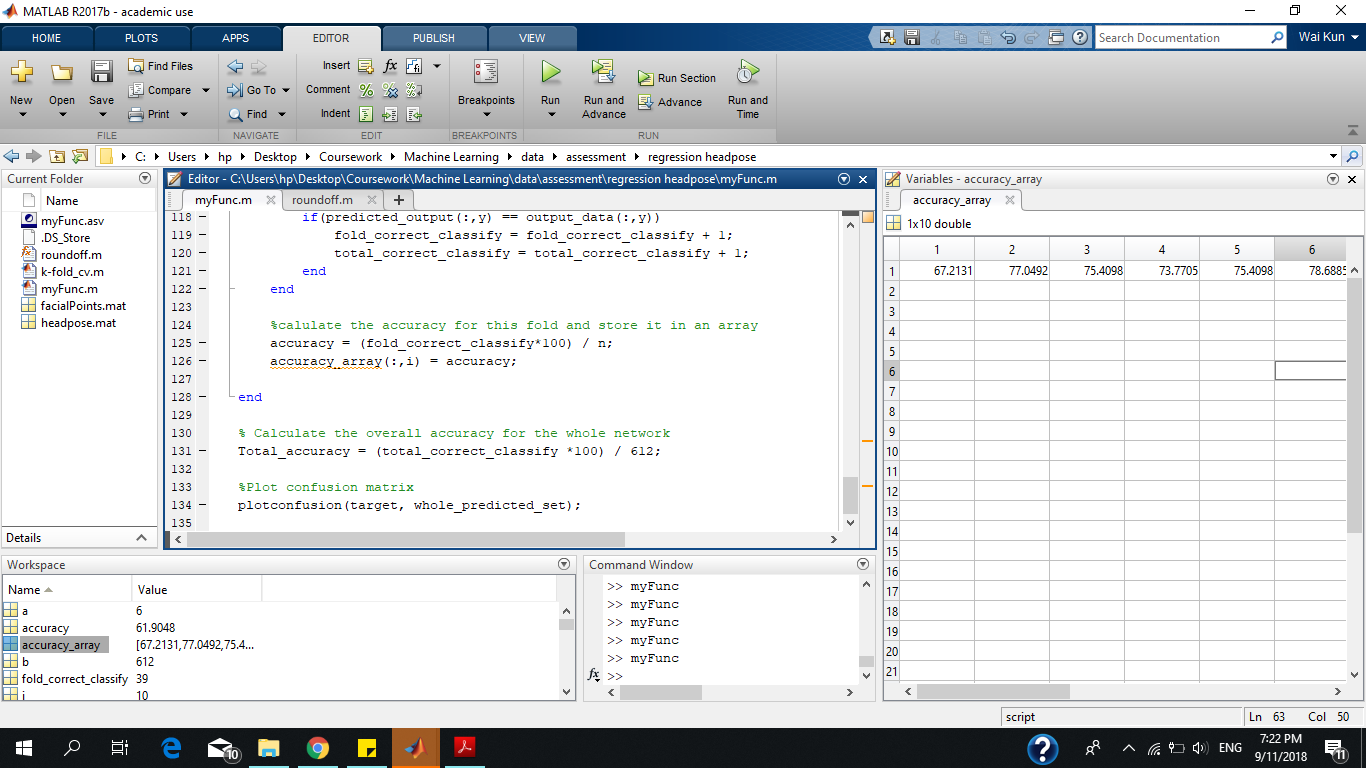
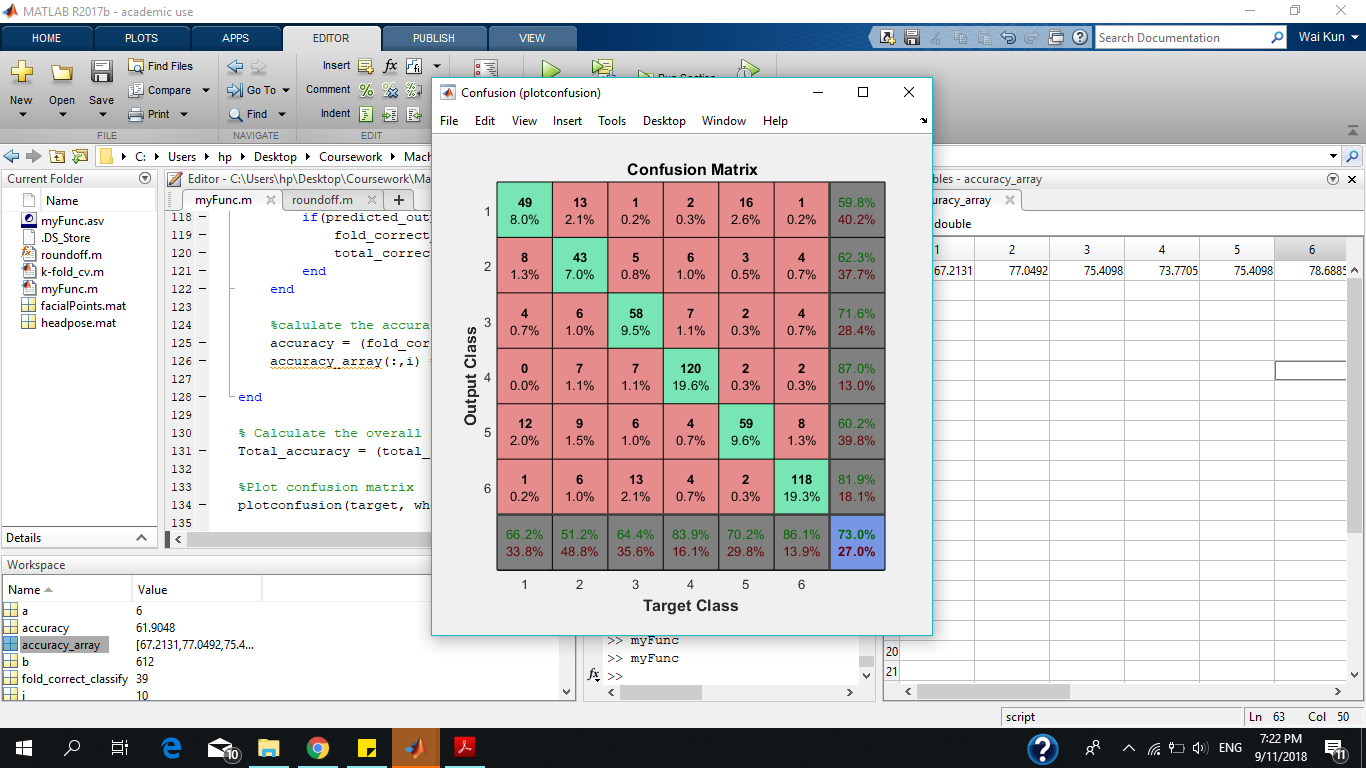


Figure 2e. Calculation for overall accuracy

Average accuracy = (Total number of correctly classified classes) x 100

Total number of data samples



At last, the confusion matrix is plotted. Overall accuracy is 73% and the error rate is 27%

1. **Regression**

**3.1 Parameters used**

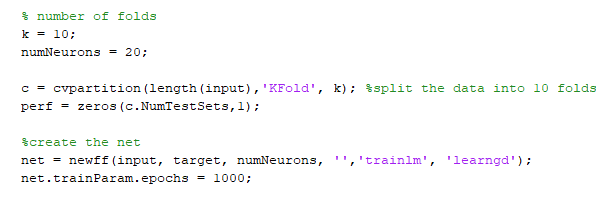


Figure 3a. Parameters setting for the regression problem

In neural network, most parameters are tuned based on trials and errors. After some testing, the decided number of neurons used for this regression model is 20. This particular number is chosen as it results in the lowest root mean squared error and the network can still be trained in a reasonable amount of time.

As for the training function, *‘trainlm’* is used because it performs better on function fitting (nonlinear regression) problems compared to other training functions and there’s enough memory to run it.

**3.2 Regression result for each fold**

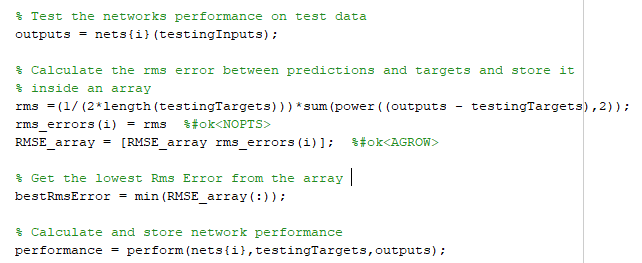
****

Figure 3b. Calculation for root mean squared error

To validate the network built, k-fold cross validation where k=10 is used.

The data is partitioned into 10 folds with the Matlab *cvpartition()* function.

|  |  |
| --- | --- |
| Fold(k) | Root mean squared error (RMSE) |
| 1 | 1.2997 |
| 2 | 0.9621 |
| 3 | 1.2611 |
| 4 | 1.1250 |
| 5 | 1.2920 |
| 6 | 1.0965 |
| 7 | 1.1069 |
| 8 | 1.2056 |
| 9 | 0.9351 |
| 10 | 1.1804 |

Table 3.2 Regression results per cross-validation fold

**3.3 Lowest root mean squared error**

After completing 10-fold cross validation for the regression model, the lowest root mean squared error obtained from using 20 neurons is **0.9351**.

**Actions taken to prevent overfitting**

To generalize the network built, during the training process, the data is split into a training set, validation set, and test set.

To prevent overfitting, the training is stopped when the lines start to diverge, where the training line continues descending. This shows that the network is overtrained, and is overfitting the data. The optimal number of epochs is obtained based on this factor.

The number of validation checks is set based on the training of the network. Increasing the number of validation checks, which will in turn increase the number of iterations, will result in overtraining, despite a lower error value.

**Conclusion**

This coursework has allowed us to get the hands-on experience on the implementation of Artificial Neural Network(ANN). We evaluate the performance of the neural network based on the different solutions, which are confusion matrix for a classification problem and root mean squared error (RMSE) for a regression problem.

One difficulty that we had faced is when parameters are tuned for the classification problem, most of the training functions yield the similar overall accuracy, which makes it hard to decide which training function to use. Regardless, the parameters and solutions suggested for the problems have generally improved the generalisation of the neural network and prevented the overfitting of training.

Overall, the result that we obtained from the neural networks are satisfactory and has achieved our aim of the coursework.