Rain Prediction in Kuala Lumpur Using Neural Network

WID3003 Neural Computing

Dr Woo Chaw Seng

WEA150022 Ong Jia Aun

Table of Contents

Introduction

Weather forecasting is the application of science and technology to predict the conditions of the atmosphere for a given location and time. It is extremely useful as to plan ahead of time for certain things. For example, if winter is coming earlier this year, then agricultural crops have to be harvest earlier as well. In Malaysia, weather forecasting can help with flooding-prone zones to prepare and brace for impending flood that might arise. With the advancement of machine learning, neural network in particular has been adopted in prediction and forecasting outcome. The basic components of a neural network are the input layer, hidden layers, and output layer. Each neuron in a layer is connected to all of the neurons in the next layer, making a mesh-like network. With an algorithm to adjust each neuron's connection to the other, we are able to train a model to predict a certain thing. In this assignment, we are looking at how to use a feedforward neural network in Matlab to do rain forecasting.

Problem scope

The historical weather data of Kuala Lumpur are taken from the website recommended by Dr Woo. The aim of this is to gather relevant input data for the training of the neural network. Some data pre-processing is required to normalize the data to help the neural network to learn better. For this assignment, we are only looking at the data at Kuala Lumpur and attempt to predict whether it will rain based on the given input factors.

Input

Network Architecture Design

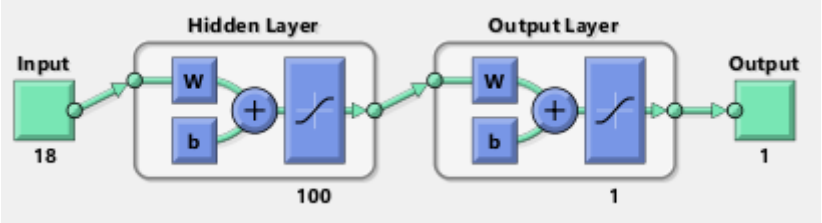

Figure 1 Neural network design

Hidden layer has 100 neurons.

Network Initialization

Matlab Neural Network Toolbox is used for quick deployment. The toolbox sets the best weight initialization for the neural network by default. The network uses TANSIG (tangent sigmoid function which map output to range of -1 and 1) as activation function by default and TRAINSCG (Scaled conjugate gradient backpropagation) as a network training function that updates weight and bias values according to the scaled conjugate gradient method. TRAINSCG is able to train any neural network as long as its weight, net input, and activation functions can have derivative functions. Backpropagation is used to calculate derivatives of performance perf with respect to the weight and bias variables X. Cross-Entropy is used as the metric to measure the performance of the model.

Dataset for Training and Testing

Dataset is gathered from the website wunderground.com. In order to be more efficient in gathering the data, I have used PHP to extract data from the HTML page. The data gathered are from the period Jan 1, 2015 to December 31, 2017. 75% of the data is used for training, 25% of the data is used for validation. Dataset from Jan 1, 2018 to March 31, 2018 is used for testing.

Data Normalization

For data normalization, I used MinMax Scaling. This ensures that the range of values are between 0 and 1.

The formula is $zi=xi-min(x)/max(x)-min(x)$,

where $x = (x1,...,xn)x=(x1,...,xn)$ and zi is now your ith normalized data.

The target output has been translated from string to numerical representation as well. All weather conditions other than rain and thunderstorm are considered to be 0 else 1. Some columns such as wind speed which has a lot of missing data has been removed entirely as it affects the performance of the neural network. Timestamp of data is removed as well as it does not affect the output data.

Process

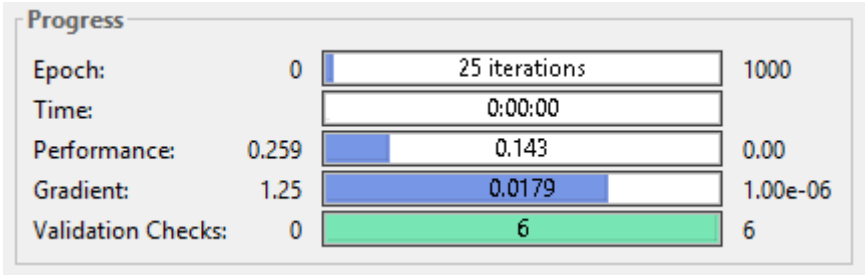

Figure 2 Progress of training the neural network

25 epochs to reach performance of 0.143 and gradient of 0.0179.

Convergence State

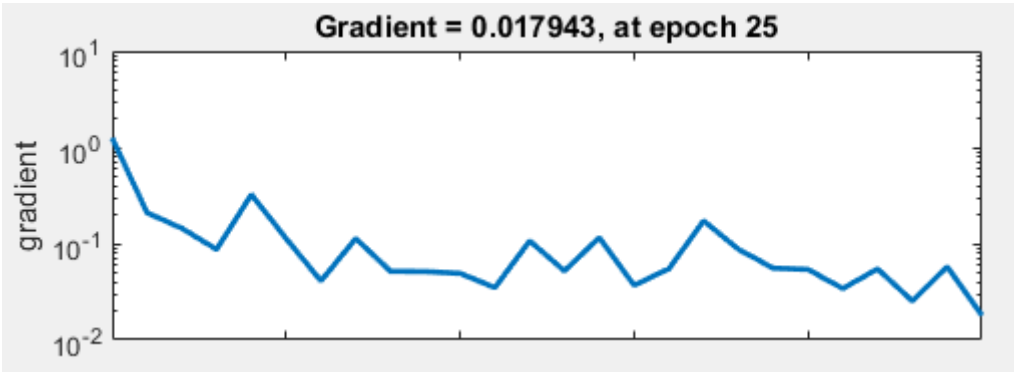

Figure 3 Converging state of the gradient descent

The gradient converged at epoch 25, with the global minimum being 0.017943.

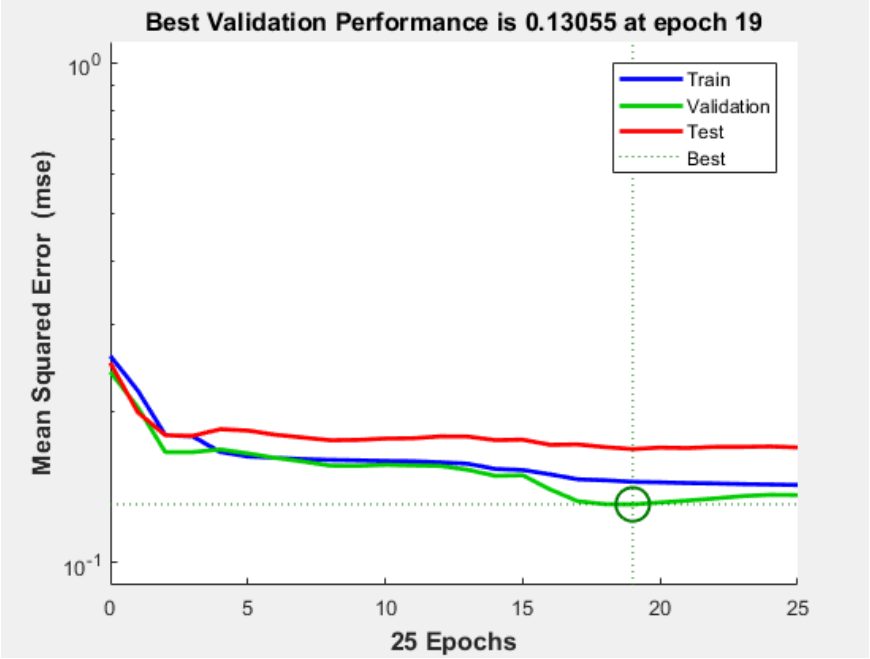

Figure 4 Training, validation and test loss

The network loss is the best at epoch 19 with a loss of 0.13055.

Hard Stop Condition

The hard stop condition for the training is when the validation checks reaches 6 times.

Output

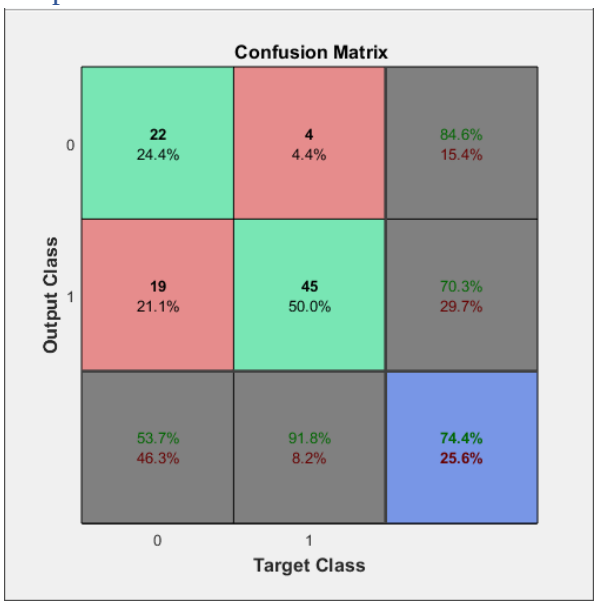

Figure 5 Result from unnormalized data

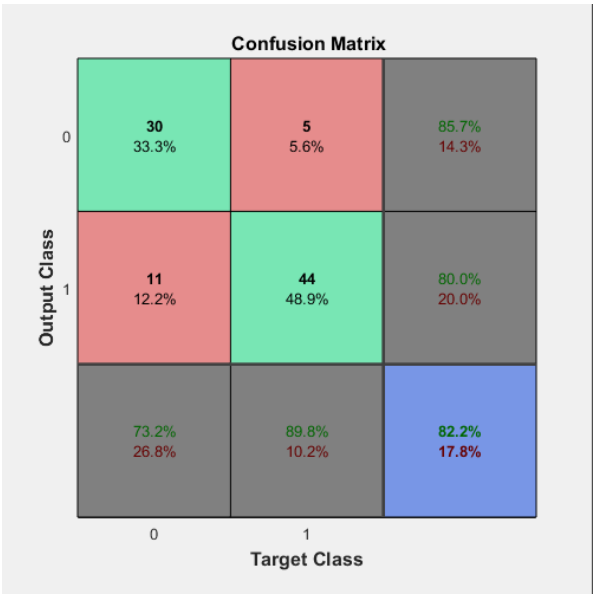

Figure 6 After normalizing data

The above confusion matrix shows the result of the model with the testing dataset (90 samples). By comparing the pre-normalized dataset and normalized dataset for training the neural network, it is obvious that there is nearly an 8% increase in the accuracy of the model. With more dataset, the model will be able to learn even better and achieve a higher accuracy than the current model of 82.2%.

Conclusion

Data cleaning and normalization is important for neural network to achieve good result. There is also a noticeable increment in accuracy as more dataset is feed into the model for training. There is still a space for improvement for this forecasting task, but it is a good start for us to learn about steps for making predictions using neural network utilizing Matlab.

References

[1] Weather Underground, historical data of weather. https://www.wunderground.com

[2] https://www.mathworks.com/help/nnet/ref/trainscg.html

[3] https://www.mathworks.com/help/nnet/ref/train.html

[4] https://en.wikipedia.org/wiki/Weather_forecasting

[5]https://www.mathworks.com/matlabcentral/answers/122989-how-to-get-best-test-error-accuracy-with-neural-networks-pattern-recognition