Neural Networks Projects

Engine Function Fitting Problem

Tom Makkink

2018

**1 Introduction**

Function fitting is the process of training a neural network on a set of inputs, to map to a set of outputs. Once the network has been fitted to this data, it can be used to analyze the way different inputs affect the output, and to generate results based on inputs it was not trained on.

We would like to create a neural network that be used to estimate the torque and emissions of an engine based on fuel use and speed.

**2 The Data**

The data was acquired from the Matlab Neural Network Toolbox, and was loaded as follows:

[p,t]= engine\_dataset;

The pattern p matrix is a 2 x 1199 matrix, defining the fuel rate and speed of an engine under different conditions. The target t was a 2 x 1199 matrix of the two attributes: torque and nitrous oxide emissions.

**3 Method**

A feedforward neural network was created with the following attributes:

* Two hidden layers with 20 neurons in each.
* The transfer functions used in the hidden layer were tansig, and purelin in the outer layer.
* The data was divided into 60% training set, 20% testing set and 20% validation set.
* The Levenberg-Marquardt back-propagation algorithm (trainlm) was used in the training method.
* Gradient descent with moment was chosen as the learning function.
* The performance function was the mean square of errors (MSE).

The data was propagated through the network as follows:

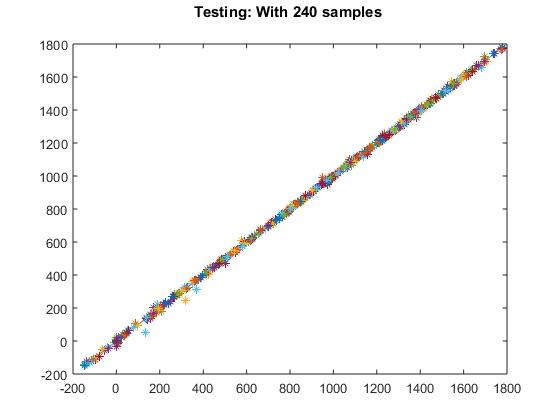
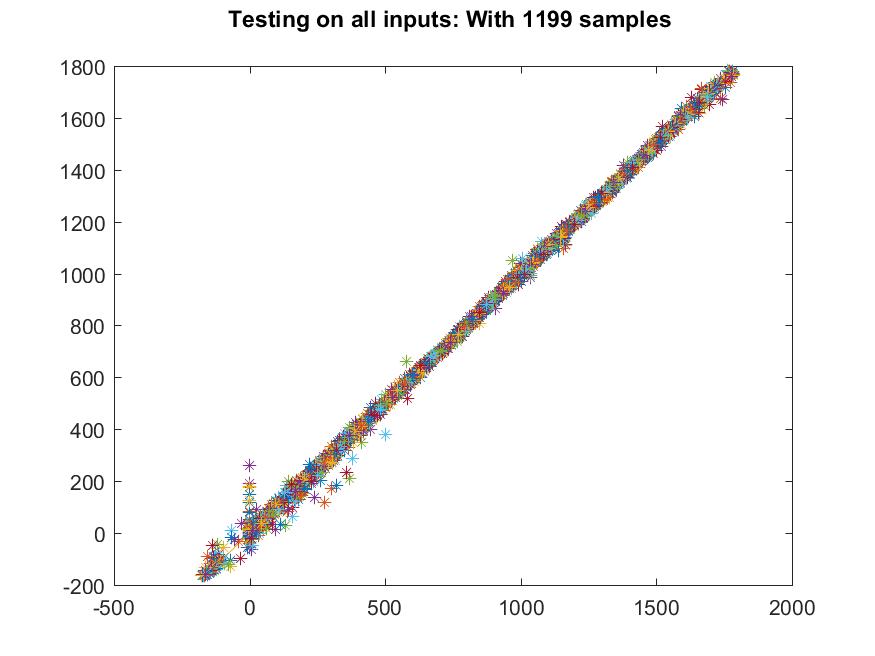
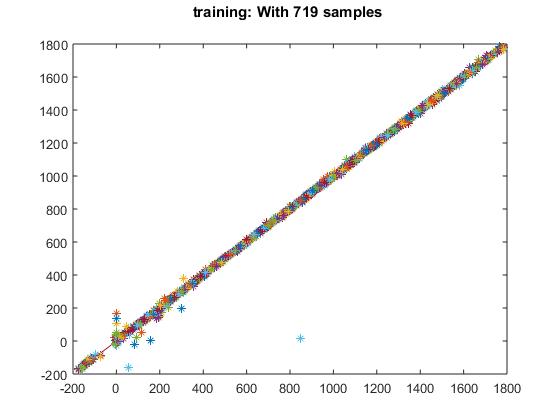
atrain = sim(engineNet, ptrain);;

atest = sim(engineNet, ptest);

a = sim(engineNet, p);

The R-squared statistic and correlation coefficient were then calculated for the training set, the test set, and on all the values, and produced the following results:

|  |  |  |  |
| --- | --- | --- | --- |
| Data Set | Correlation Coefficient | P value | R-squared statistic |
| Training | [1, 0.999505, 0.999505, 1] | [1, 0, 0, 1] | 0.999718, 0.997918 |
| Testing | [1, 0.999727, 0.999727, 1] | [1, 0, 0, 1] | 0.999815, 0.998891 |
| Testing (all inputs) | [1, 0.999603, 0.999603, 1] | [1, 0, 0, 1] | 0.999758, 0.998358 |



Another script was then created to explore the relationship between the inputs and outputs, called *engine\_sim.m.*

The data was loaded and the average, minimum and maximum found as follows:

[P, T] = engine\_dataset;

p = P';

t = T';

% Find Max

pMax = max(p);

% Find Min

pMin = min(p);

% Find Mean

pMean = mean(p);

The fuel rate was the fixed and the speed varied:

% Fix the fuel rate, vary the speed

varySpeed(1,:) = repmat(pMean(2), 1, 1200);

varySpeed(2,:) = linspace(pMin(1), pMax(1), 1200);

The speed was then fixed and the fuel rate varied:

% Fix the speed, vary the fuel rate

varyFuel(1,:) = linspace(pMin(2), pMax(2), 1200);

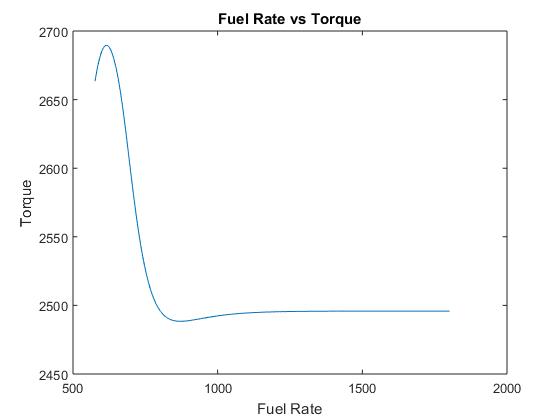
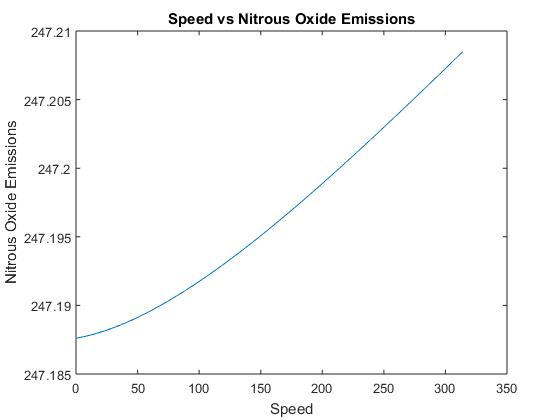
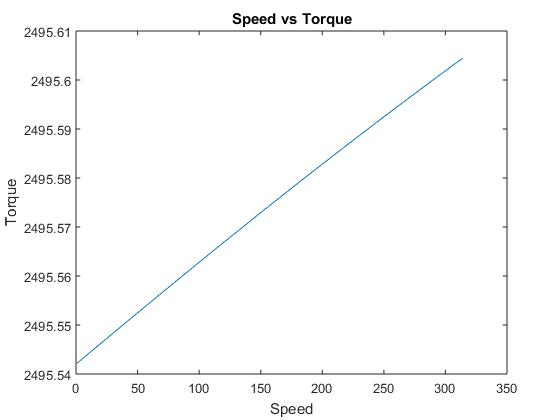
varyFuel(2,:) = repmat(pMean(1), 1, 1200);

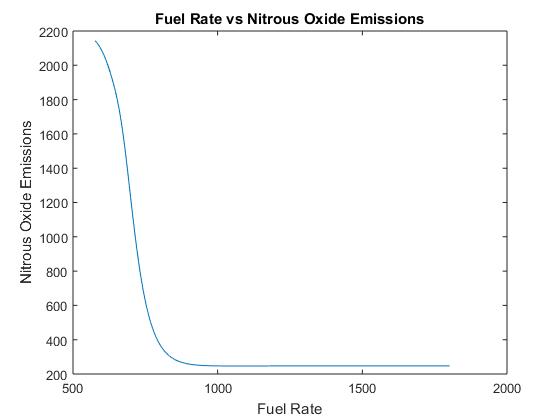
The two sets of data were then simulated on neural network *engineNet*, which was trained and saved in *engine\_net*.

speedNet = sim(engineNet, varySpeed);

fuelNet = sim(engineNet, varyFuel);

The relationship between speed and fuel rate compared to torque and nitrous oxide was then plotted, and produced the following graphs:





**4 Conclusion**

An R-squared statistic of 0.999758, 0.998358 when testing on all inputs is very good, and it is likely that given an input of speed and fuel rate, that the neural network could provide an accurate estimation of the nitrous oxide emissions and torque.

**4 References**

*https://www.mathworks.com/help/nnet/gs/neural-network-toolbox-sample-data-sets.html*