Sharif University of Technology
Electrical Engineering School

Computational Intelligence CHW1

REGRESSION, MULTI LAYER PERCEPTRON

Armin Panjehpour
arminpp1379@gmail.com

Supervisor(s): Dr. Hajipour
Sharif University, Tehran, Iran

26/04/2022

Part.1 - Function Fitting

Here we use linear regression, logistic regression and MLP for function fitting. Our function:

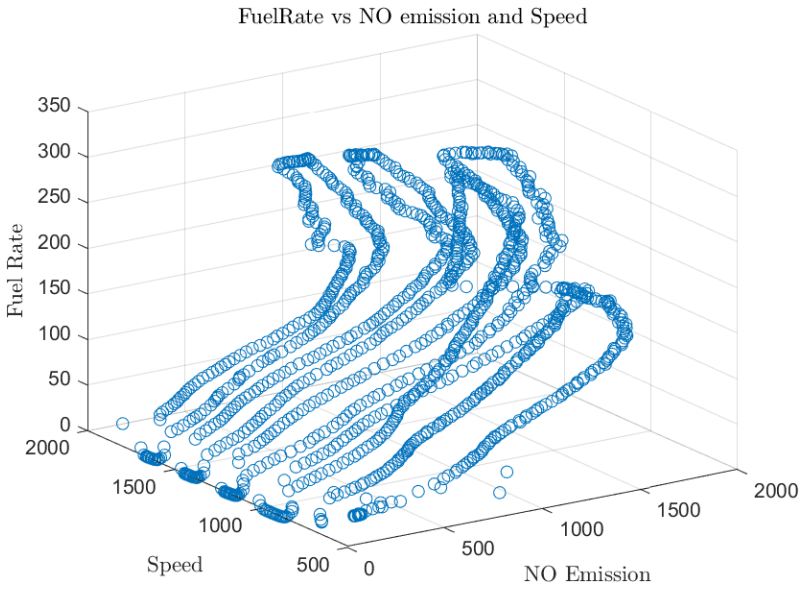

Figure 1: z = f(x,y)

Then we have to separate our data to two train and test datasets. We take first 700 samples as the train data and the rest as the test dataset. So we have a 3×700 matrix for training and a 3×499 matrix for test.

Part.1.a - Linear Regression At first, we create the predictor matrix X which contains Speed and NO Emission train values, plus an added column of ones. Then after creating the responses vector, y, we give y and X to the "regress" function in Matlab and get the coefficients estimates for the fitted plane. Here's the training data in a 3D plot and the fitted plane:

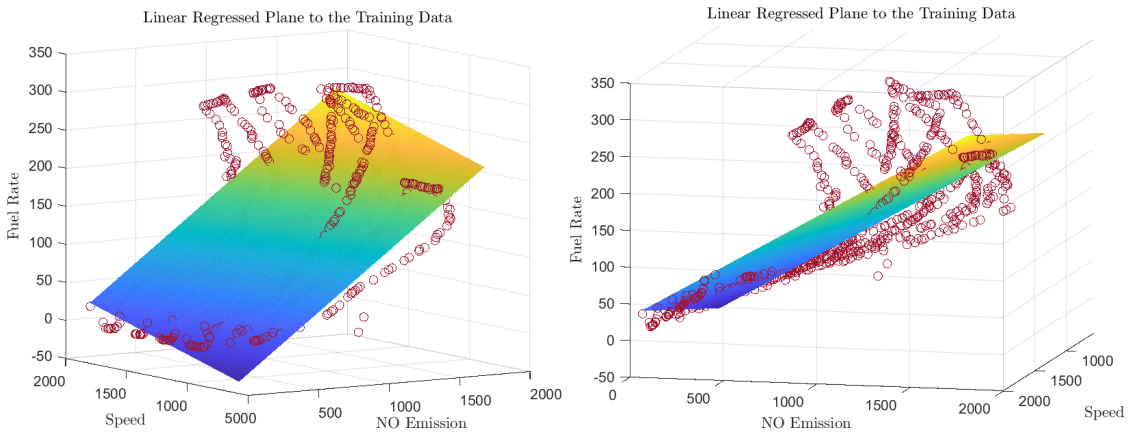

Figure 2: fitted plane using linear regression to the train data

Now we have to check, how good our model describes our data. So, we calculate MSE on both train and test data:

$$, \quad y_i: Real\ Value \qquad \hat{y}_i: Estimated\ ValueMSE = \frac{1}{N}\Sigma_{i=1}^{N}(y_i - \hat{y}_i)^2$$

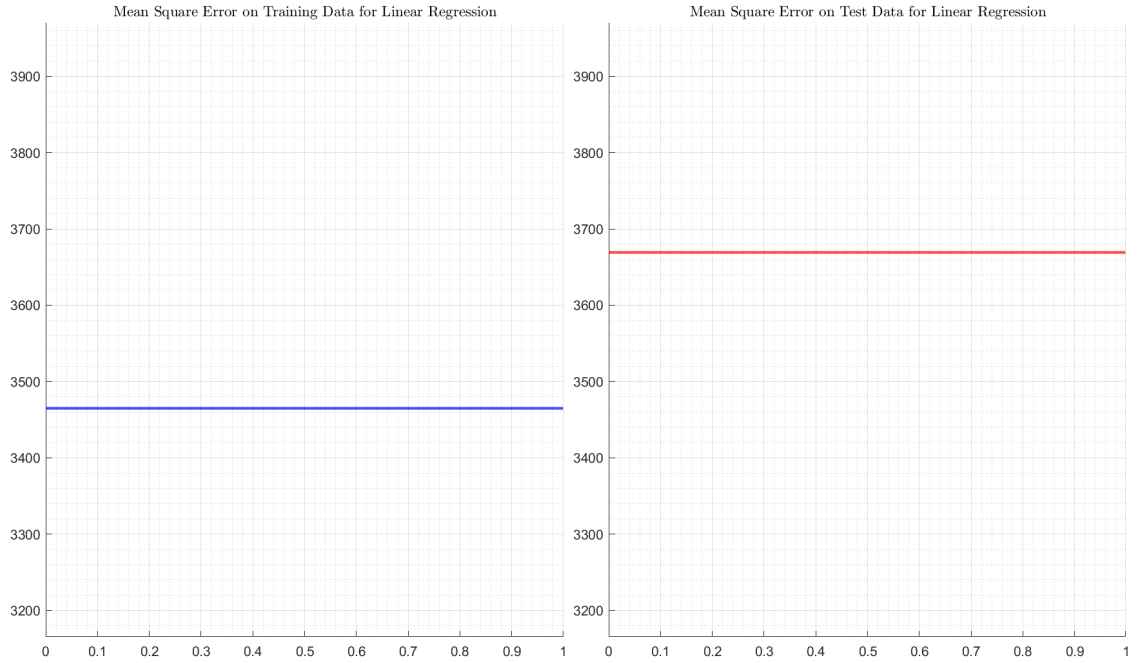

Figure 3: MSE for train and test data

As you can see, the errors for both train and test dataset are close but high. It is obvious that our data is not described very good with the fitted plane, thus we have such a big MSE.

Part.1.b - Logistic Regression At first, we convert the logistic regression to a linear regression as you can see below:

$$y = \frac{Y}{1+e^{a+bx}} \quad \Leftrightarrow \quad \frac{1}{y} = \frac{1+e^{a+bx}}{Y} \quad \Leftrightarrow \quad \frac{Y-y}{y} = e^{a+bx} \qquad \ln\left(\frac{Y-y}{y}\right) = a+bx$$

Figure 4: logistic regression to linear regression

Y is selected as the maximum value of the data + 1. Then we do linear regression like the previous part and here's the result:

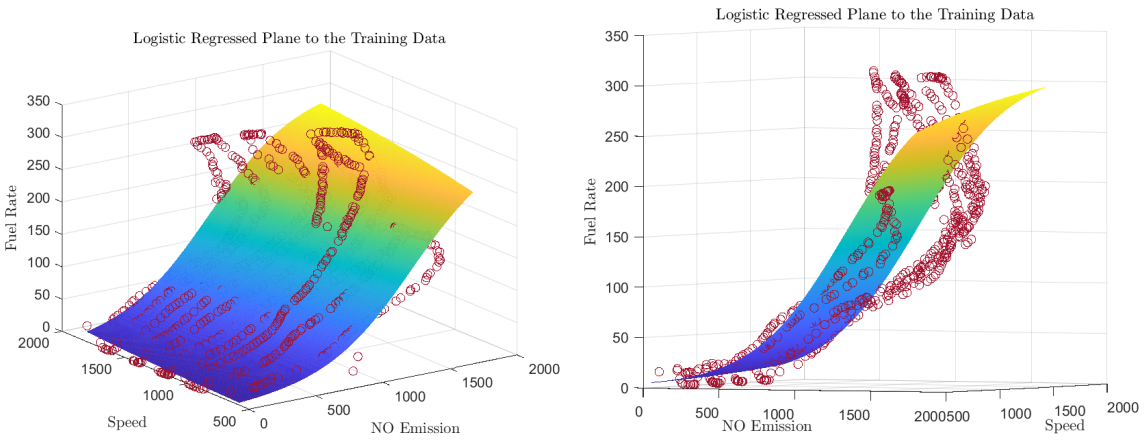

Figure 5: logistic regression to the train data

Now we check the model's performance by calculating the MSE for both train and test data:

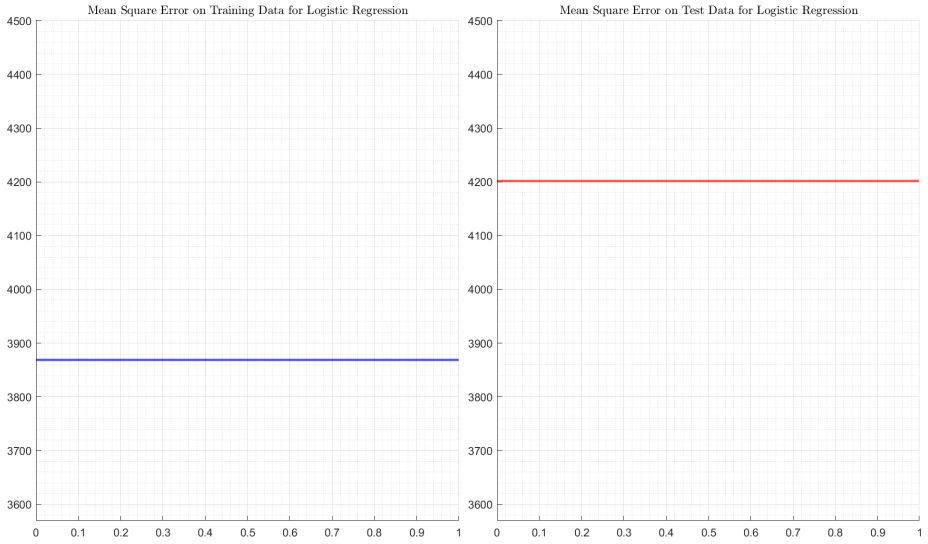

Figure 6: MSE for train and test data

As you can see, MSE is increased in logistic regression comparing to the linear regression!

Part.1.c - Multi Layer Perceptron Using "fitnet" in Matlab, we design a 2 layer perceptron. At first, we normalize our train and test data by train data mean and std. Then we choose the number of neurons of the hidden layer, by some trail and error, we take 7 neurons. Also we take 20 percentage of the train data as validation data:

```
% structure of the network
hiddenLayerNeuronNum = 7;
net = fitnet(hiddenLayerNeuronNum,'trainlm');
net.divideParam.trainRatio = 0.8;
net.divideParam.valRatio = 0.2;
net.divideParam.testRatio = 0;
```

At the end, we re-scale the estimated outputs with the train data mean and std and calculate MSE. Here‘s the result:

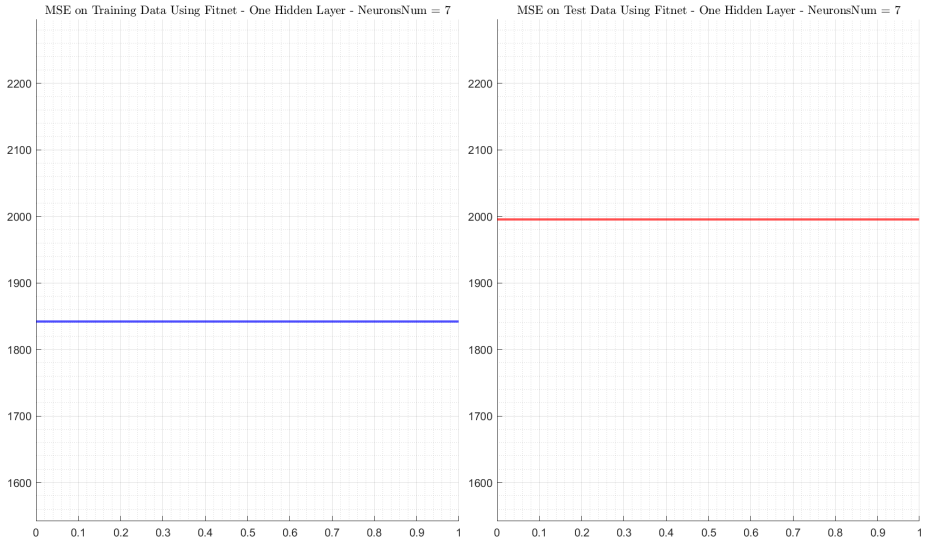

Figure 7: MSE for train and test data

As you can see, the MSE is much more less than the two kinds of regression we did in previous parts.

Part.1.d - Summary Here we plot MSE of all models:

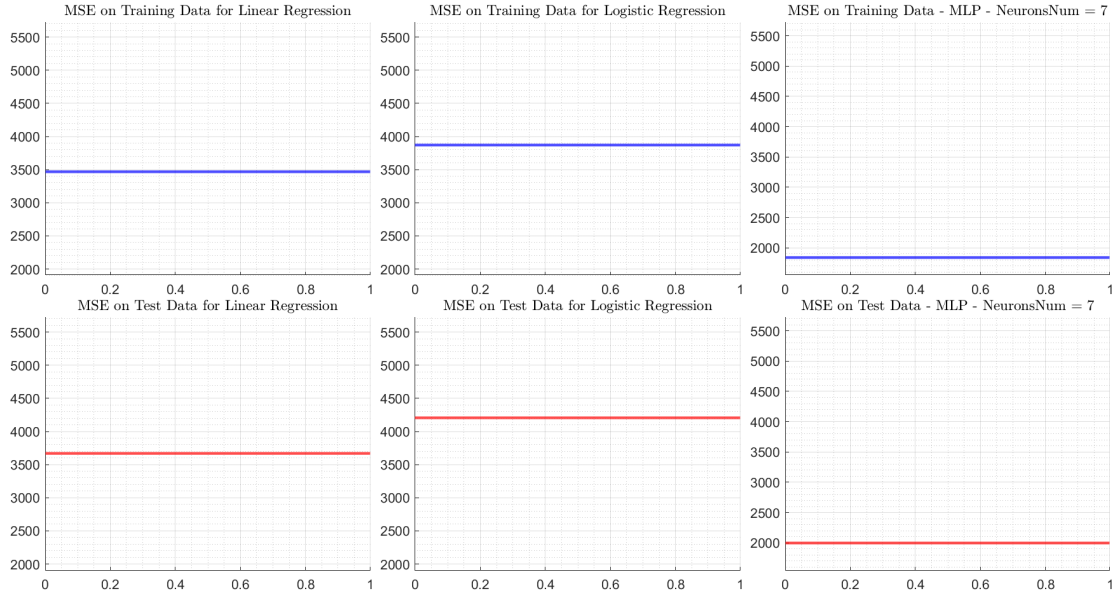

Figure 8: MSE for train and test data of all models

As you can see, MLP has better performance on the data since it is much more complicated than the linear and logistic regression. Since our data is not linear, a plane fitted by linear regression doesn't have that good performance on it. The same for the logistic regression too, the data is not well described by a logistic function so they don't perform well.