**CSC 4760 Homework 3**

**Instructions:**

1. Train a linear regression model to predict the attribute of acceleration based on weight and horsepower from the provided data set in ‘Homework3.mat’
2. Plot the samples and the plane representing the trained regression model together in one figure.
3. Upload the plot and your code in one file.

**Results:**A screenshot of a computer program

Description automatically generated

**Figure 1 – Lines 1-30:**

The section of code depicted in figure 1 describes the initial objectives for the lab, and also contains the code for importing the .mat file into python. Python does not natively support .mat files, which means you need to install scipy or some other converter. Scipy converts the .mat file into a dictionary, which is why the data values for Horsepower, Weight, and Acceleration are using the name for that column to create the data set.

A screenshot of a computer program

Description automatically generated

**Figure 2 – Code lines 33-60**

The section of code depicted in figure 2 cleans the data and converts it into pytorch tensors. The data is cleaned to get rid of any not a number result. The model that was presented as an example in class could handle multiple inputs, but only if they were combined as one array as a column\_stack so that it could be later passed to the training model.

A screenshot of a computer program

Description automatically generated

**Figure 3 – Code lines 68-94**

The code depicted in figure 3 defines the class for the linear regression model, this involves initializing the class and making sure that it can use torch for the linear regression. The model is then created as an instance of the class, and so is the type of loss function, and the rate of loss. It should be noted that the original loss rate tended to overcorrect and lead to infinity loss, as a result the loss rate was adjusted to a much lower number. The training inputs were then defined for the actual training of the model.

A screenshot of a computer program

Description automatically generated

**Figure 4 – Code lines 96-122**

Figure 4 depicts the actual training of the model. The model class is used to generate inputs based on the training data, the loss is then calculated based on the different between the output from the model and the training data. The weights are then adjusted. For utility in the comments I printed the loss each time – as for some loss rates it was trending towards infinity. The model was then used to generate training data – which was then not used because it caused issues in plotting, so instead the model was used directly for plotting – and it tended to work!

A computer screen shot of a code

Description automatically generated

Figure 5 – Code lines 127-158

Figure 5 depicts the code that actually plots the data. This is pretty simple, a 3d data plane is generated and the original data is populated into the data plane. However due to weird formatting, the predict function was not outputting correctly – instead the model was used to generate new data points based on flattened version of horsepower and acceleration – and then a plane was generated. This did not cause any issues, as the dictionary generated initial versions of those arrays was a list of lists where the lowest level of list only contained one value.

A graph of a graph with a gradient of red dots

Description automatically generated with medium confidence

Figure 6 – The Generated Plot