COSC 528: Introduction to Machine Learning
Project-1: Multivariate Linear Regression
Rayhan Hossain

Introduction:

Multiple regression is a function of multiple inputs consisting of discrete or numeric values, that has one output that is a class or continuous value. In this project, our main objective is to build a linear and polynomial regression model to predict the mpg of cars. As a constraint, we had no privilege to use the py tools for finding the linear regression, rather we had to develop it from scratch.

Data:

We were given the auto-mpg dataset with our project folder which has a total of nine columns and 398 instances. This info was collected from vehicles from the 1970's and 1980's. There are 8 numeric classes with a mixture of discrete and continuous values.

Data Preprocessing:

Among the total nine columns, the last one is the name of the make which doesn't have a great effect on our result. That's why we need not consider this column. Three of them are multi-valued discrete, and five of them are continuous values. The attributes and their types are depicted below:

- mpg: continuous - to be predicted
- cylinders: multi-valued discrete
- displacement: continuous
- horsepower: continuous
- weight: continuous
- acceleration: continuous
- model year: multi-valued discrete
- origin: multi-valued discrete
- car name: string (unique for each instance)

We need to predict the value of mpg (miles per gallon) based upon other variables. So, we can say mpg is dependent and others columns are independent.
For the horsepower attribute, there are six missing values. People handle this missing values in different ways. Two well-known approaches are mean imputation and imputation by regression

possible. Some people prefer to simply remove them. At first, I tried to ignore the missing values then mean imputation.

Methodology:

For this section, I copied the figure and equations directly from our textbook.

Except the two attributes (mpg and names) I represent the data in a matrix form like the following one:

$$\mathbf{X} = \begin{bmatrix} X_1^1 & X_2^1 & \cdots & X_d^1 \\ X_1^2 & X_2^2 & \cdots & X_d^2 \\ \vdots & & & \\ X_1^N & X_2^N & \cdots & X_d^N \end{bmatrix}$$

Here is column is a single attribute and each row is a single experiment data sample for a unique car.

For multiple regression, the output is a vector r which is mpg for this work. We have a weighted sum of attributes $x_1, x_2 ... x_d$ and some noise. Here is a multivariate linear model:

$$r^t = g(\mathbf{x}^t | w_0, w_1, \ldots, w_d) + \epsilon = w_0 + w_1 x_1^t + w_2 x_2^t + \cdots + w_d x_d^t + \epsilon$$

Normally, we assume error to be normal with mean 0 and constant variance. To maximize the likelihood we need to minimize the sum of squared errors:

$$E(w_0, w_1, \ldots, w_d | \mathcal{X}) = \frac{1}{2} \sum_t (r^t - w_0 - w_1 x_1^t - w_2 x_2^t - \cdots - w_d x_d^t)^2$$

Doing the derivation with respect to all parameters we get the normal equations:

$$\begin{aligned}
\sum_t r^t &= N w_0 + w_1 \sum_t x_1^t + w_2 \sum_t x_2^t + \cdots + w_d \sum_t x_d^t \\
\sum_t x_1^t r^t &= w_0 \sum_t x_1^t + w_1 \sum_t (x_1^t)^2 + w_2 \sum_t x_1^t x_2^t + \cdots + w_d \sum_t x_1^t x_d^t \\
\sum_t x_2^t r^t &= w_0 \sum_t x_2^t + w_1 \sum_t x_1^t x_2^t + w_2 \sum_t (x_2^t)^2 + \cdots + w_d \sum_t x_2^t x_d^t \\
&\vdots \\
\sum_t x_d^t r^t &= w_0 \sum_t x_d^t + w_1 \sum_t x_d^t x_1^t + w_2 \sum_t x_d^t x_2^t + \cdots + w_d \sum_t (x_d^t)^2
\end{aligned}$$

Now we need to add a column to the front of our matrix with all ones since in our model w_0 term is the same as w_0. We can define a vector of all weights, and an output vector which is our milage values. So we get:

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^1 & x_2^1 & \cdots & x_d^1 \\ 1 & x_1^2 & x_2^2 & \cdots & x_d^2 \\ \vdots & & & & \\ 1 & x_1^N & x_2^N & \cdots & x_d^N \end{bmatrix}, \boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d, \end{bmatrix}, \boldsymbol{r} = \begin{bmatrix} r^1 \\ r^2 \\ \vdots \\ r^N \end{bmatrix}$$

Finally, the normal equation can be written as:

$$\mathbf{X^T X w = X^T r}$$
$$\mathbf{w = (X^T X)^{-1} X^T r}$$

And, our final goal is to compute and analyze the weight **w**.

Implementation:

At first, I started to implement using Java as I am more confident in it. Later I found that it was getting tough for me to do the basic data processing and matrix operations. Then I changed my mind and switched to Python. Pythons numpy gives a lot of functions for doing matrix operations in the easiest way.

Basically, I have two main files- one the project which does the linear regression, and another one is the data file.

Training and Validation:

Randomly selected 20% of the total dataset is used as the training dataset, and the rest 80% is used for testing and validation purpose. Mean and standard deviation of every attribute in the training are computed and used to standardize (i.e., z-normalize) the training set.

Result:

In the first run, I skipped the dataset with missing attributes, and I didn't standardize the parameters. So I received the w values very low, and these were really tough to

compare. Then when I standardized the dataset to create a understandable and comparable result, it was pretty good. The results are given in a table below:

Attribute	**Without Standardization**	**With Standardization**
mpg	-1.805867e+01	23.45731655
cylinders	-4.1825489e-01	-0.71059527
displacement	1.8887016e-02	1.96696134
horsepower	-1.138519e-02	-0.43435854
weight	-6.718658e-03	-5.68269540
acceleration	1.0262086e-01	0.28265403
Model year	7.5675505r-01	2.79477515
origin	1.4175156e+0	1.1355083

Conclusion:

From the above data, we can see that Cylinders and Horsepower have some negative effect on gas mileage. The great factor with the negative effect on mpg is the weight which is also expected from the raw data analysis. We can also see some with positive values meaning they have less effect on mileage.

Remarks:

The main challenge I faced was developing with the Python. When I switched from Java to Python it took me a long to find and understand the proper Python syntax. There were a lot of easier functions for plotting and visualizing the data which I missed. I hope I can come up with a good visualizing report next time. I also discussed with three of our students- Daniel, Povlin, and Paula to get a clear understanding of the algorithms and some basic help with Python.