

# Predicting and Modeling Driver Behavior with Self Learning Algorithms

Jeremy Zhang

## **Abstract**

In high school, many students often obtain their driver's license. For many, it is an exciting but anxious experience to drive. With the assistance of technologies from car manufacturers, many mistakes made by teenagers on the road can be prevented to make the roads safer. However, these systems are easily blinded due to heavy precipitation, fog, and unclear lane marks. Humans constantly face driving situations where clear, uninterrupted sight lines of the road ahead are not present. Through predicting the movement of cars based on their direction, speed, and the space ahead, it allows cars to navigate even when weather or traffic conditions are not ideal.

Machine learning algorithms would enable a vehicle to "learn" the way a human would drive in the real world, allowing it to adapt to a wider range of scenarios that may not be covered by current advanced driver assistance systems. This study uses multiple regression and artificial neural network to predict driver behavior based on highway data from the Federal Highway Administration NGSIM database.

In this experiment, both algorithms had similar performance but different strengths and weaknesses. The artificial neural network was better at predicting closer to the actual mean of acceleration while multiple regression was best at predicting if a vehicle would speed up or slow down. In the future, research could be done to implement a system that would draw on the strengths of both algorithms. This algorithm would fully realize the goals of an autonomous advanced driver assistance system that has the capability to work in all weather and road conditions.

## **Acknowledgement of Major Assistance**

The author is grateful to the Federal Highway Administration NGSIM Program for data availability.

## Table of Contents

Abstract .....	2
Acknowledgement of Major Assistance .....	2
List of Figures .....	4
List of Tables .....	4
1. Introduction .....	5
1.1 Motivation and Purpose .....	5
1.2 Artificial Neural Network Backpropagation .....	6
1.3 Multiple Regression .....	6
2. Materials, Methods, and Procedures .....	7
2.1 Materials .....	7
2.1.1 Data Structure and Source .....	7
2.2 Methods and Procedures .....	8
2.2.1 Data Processing .....	8
2.2.2 Artificial Neural Network Implementation and Training .....	8
2.2.3 Multiple Regression Implementation and Training .....	11
2.2.4 Out-of-Sample Prediction .....	12
3. Results .....	12
3.1 Artificial Neural Network Training .....	12
3.2 Multiple Regression Training .....	13
3.3 Out-of-Sample Prediction Results .....	13
3.4 Statistical Test Results .....	14
3.5 Cumulative Absolute Errors .....	14
4. Discussion and Conclusions .....	15
5. Future Work and Future Applications .....	16
6. Bibliography .....	17

## List of Figures

Figure 1 Structure of a neuron .....	6
Figure 2 Major steps of an Artificial Neural Network.....	6
Figure 3 The study area and the area surrounding. ....	7
Figure 4 Artificial Neural Network Backpropagation Flowchart .....	8
Figure 5 Artificial Neural Network Implementation in Eclipse .....	9
Figure 6 Illustrated Backpropagation Weight Calculation .....	10
Figure 7 Multiple Regression Implementation in Microsoft Excel .....	11
Figure 8 Artificial Neural Network Training Results .....	12
Figure 9 Multiple Regression Training Results .....	13
Figure 10 Algorithm Out-of-sample Predictions .....	13
Figure 11 Cumulative Errors .....	14

## List of Tables

Table 1 Statistical Test Results .....	14
Table 2 Critical Value Table.....	15
Table 3 Summary of activation functions .....	16

# 1. Introduction

## 1.1 Motivation and Purpose

In high school, many students often obtain their driver's license. For many, it is an exciting but anxious experience to drive. Teenagers drivers differ because a wide variety of factors, such as their individual personality, perception of the environment, behavior tendencies, and their knowledge, skill, and experience with operating motor vehicles. (Shope and Bingham, 2008) Even when basic knowledge and skills have been trained, much practice and experience is needed to make teens safer on the road. Due to the young age of teenagers and the general lack of experience because of that young age, novice teen drivers are less able to recognize and detect risks than are more experienced drivers. (Brown and Groeger, 1988; McKnight and McKnight, 2003)

Car manufacturers have developed many advanced driver assistance systems (ADAS) to help mitigate the risk of accidents and make automobile travel safer and more efficient. Some of the technology packages that come with many vehicles include: forward collision warning and auto-braking, lane departure warning and prevention, and blind spot detection. Many ADAS use a wide variety of sensors including radars and cameras. These sensors are usually mounted at the front of the vehicle behind the front grille. These sensors constantly take pictures or send out ultra-high frequency waves to detect vehicles and objects. When an object is detected within the minimum threshold distance, the system will sound the alert or apply the brakes.

These systems are easily blinded due to heavy precipitation, fog, and unclear lane marks. Manufacturers have built in a safe-guard that prevents ADAS from operating when the sensors cannot see well enough. The driver is notified that the system is disabled and the system will resume automatically when the sensors clear up. When ADAS cannot see the road ahead clearly, they just simply disable themselves. Humans, however, constantly face driving situations where clear, uninterrupted sight lines of the road ahead are not present. Through predicting the movement of cars based on their direction, speed, and the space ahead, it allows cars to navigate even though weather or traffic conditions are not ideal.

Machine learning algorithms would allow for an ADAS implementation that would be adapting to the many different situations and scenarios that the driver is facing. These machine learning algorithms would also be able to make predictions based on any previous observations made. Machine algorithms fall into two general categories; linear and non-linear. In some cases, one type of algorithm may work better than another. In this study, the nonlinear algorithm chosen was an artificial neural network, and the linear algorithm was multiple regression. Being able to predict sudden changes in vehicle acceleration could save many lives and ultimately realize the true potential of many ADAS. In this study an artificial neural network and multiple regression were closely examined for their accuracy of vehicle acceleration predictions.

## 1.2 Artificial Neural Network Backpropagation

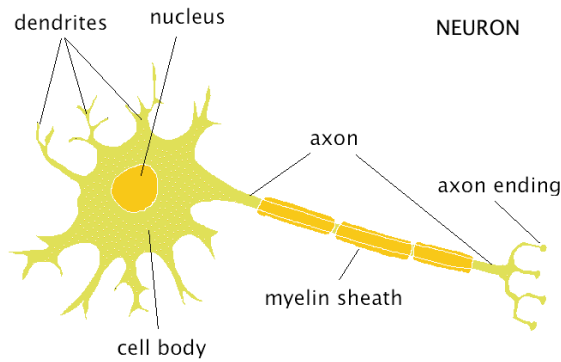


Figure 1 Structure of a neuron

The approach used to achieve the goal of minimizing error through tuning weights is called the backward propagation of errors, or more commonly, backpropagation. This approach is significantly more efficient and much faster than using a “brute force” method of trying all possible combinations of nodes. Instead, backpropagation starts with random weights

generated by the computer, and progresses through a cycle with four major steps to minimize errors and to get a better fit. These four steps are forward propagation, error calculation, backpropagation, and updating the weights to minimize the errors.

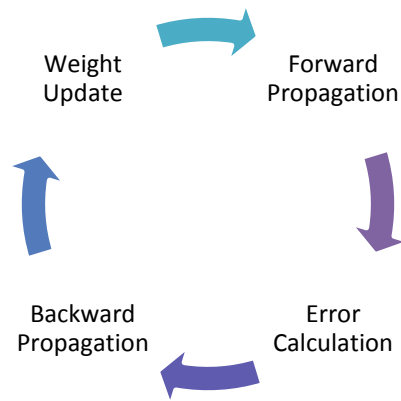


Figure 2 Major steps of an Artificial Neural Network

## 1.3 Multiple Regression

Multiple regression was chosen as the linear based algorithm in this study. The goal of multiple regression is to find a line of best fit for the given training examples. Multiple regression is an extension of simple linear regression. It is used to predict the value of a variable based on the value of two or more input variables. The variable predicted was called the dependent variable. The variables used to predict the value of the dependent variable were called the independent variables.

The derivation of least square in matrix notation started from  $y = Xb + \epsilon$ , which really is the same as:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1K} \\ x_{21} & x_{22} & \cdots & x_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{NK} \end{bmatrix} * \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_N \end{bmatrix}$$

The idea of the ordinary least squares estimator consists in choosing  $b\{i\}$  in such a way that, the sum of squared residuals:  $\sum_{i=1}^N e_i^2$  in the sample is as small as possible. By solving the expression, this gives  $b = (X'X)^{-1}X'y$ .

## 2. Materials, Methods, and Procedures

### 2.1 Materials

#### 2.1.1 Data Structure and Source

The data for this study was provided by the United States Federal Highway Administration Research and Data Exchange's Next Generation Simulation (NGSIM) database.



Figure 3 The study area and the area surrounding.

The data was collected from the south-bound direction of US 101 in Los Angeles, California on June 15<sup>th</sup>, 2005 from 7:50am to 8:05am using eight video cameras mounted on 10 Universal City Plaza, a thirty-five story building adjacent to the study area of the highway. These cameras were able to capture all vehicle positions that were traced at 0.1 second intervals, and the video was processed to provide detailed information on vehicle's type, size, absolute location, acceleration, velocity, distance between other vehicles, and the disposition of other vehicles around it.

Vehicle acceleration was provided in feet/second<sup>2</sup>, and instantaneous vehicle velocity was provided into feet/second. For the study, these numbers were converted to metric units (m/s<sup>2</sup> and m/s). Vehicle velocity and acceleration of the two cars preceding the study car were taken in as factors, along with the velocity of the study car as it is believed that the greatest factor of vehicle acceleration is the acceleration of the vehicles preceding it. The distance between each car and the time it takes for the car to travel that distance at its current velocity were also factors that could influence vehicle acceleration.

## 2.2 Methods and Procedures

### 2.2.1 Data Processing

The data for this study was first downloaded from the NGSIM database. The 15 minute segment of data from 7:50am to 8:05am consisted of about 2,800 cars and one million individual lines of data. For this study, cars 8, 21, and 25 were chosen because the cars followed each other, did not change lanes, and had approximately 1,700 individual data points per car.

Car 25 was chosen as the study vehicle, and cars 8 and 21 were the two cars directly in front of the vehicle for all 378 timestamps (each timestamp is 0.1s). The data for these cars were pasted into a Microsoft Excel spreadsheet with each row corresponding to one timeframe. The data was then split such that approximately 80% of the data was used for training and the remaining 20% was used for an out-of-sample prediction. The Excel spreadsheet was also saved as a text file (.txt) to make data input for the artificial neural network.

### 2.2.2 Artificial Neural Network Implementation and Training

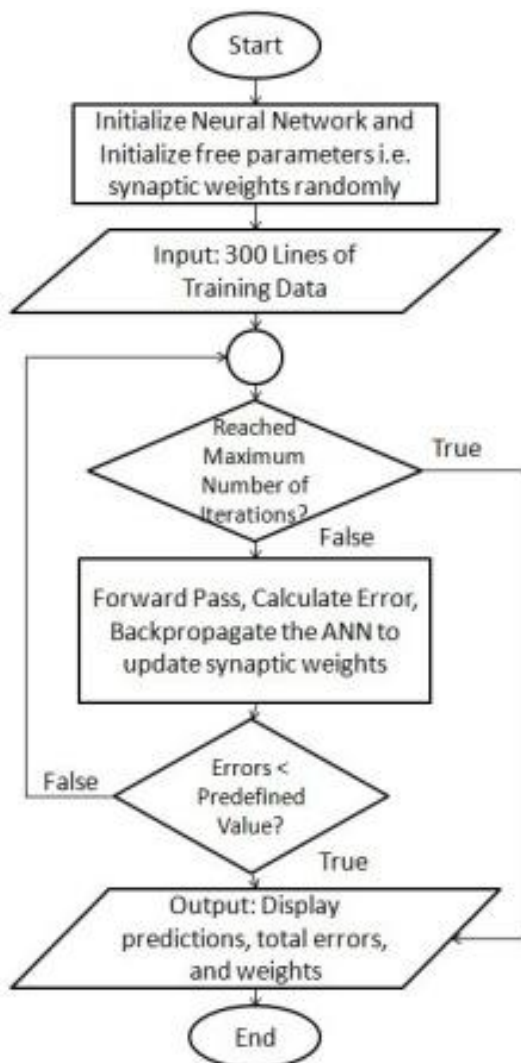


Figure 4 Artificial Neural Network Backpropagation Flowchart

The artificial neural network and the backpropagation algorithm used to tune the weights within the artificial neural network were programmed using Java. The integrated development environment (IDE) used was Eclipse Luna (version 4.4). The artificial neural network was divided into three classes; a backpropagation class, a neuron class, and a connections class.

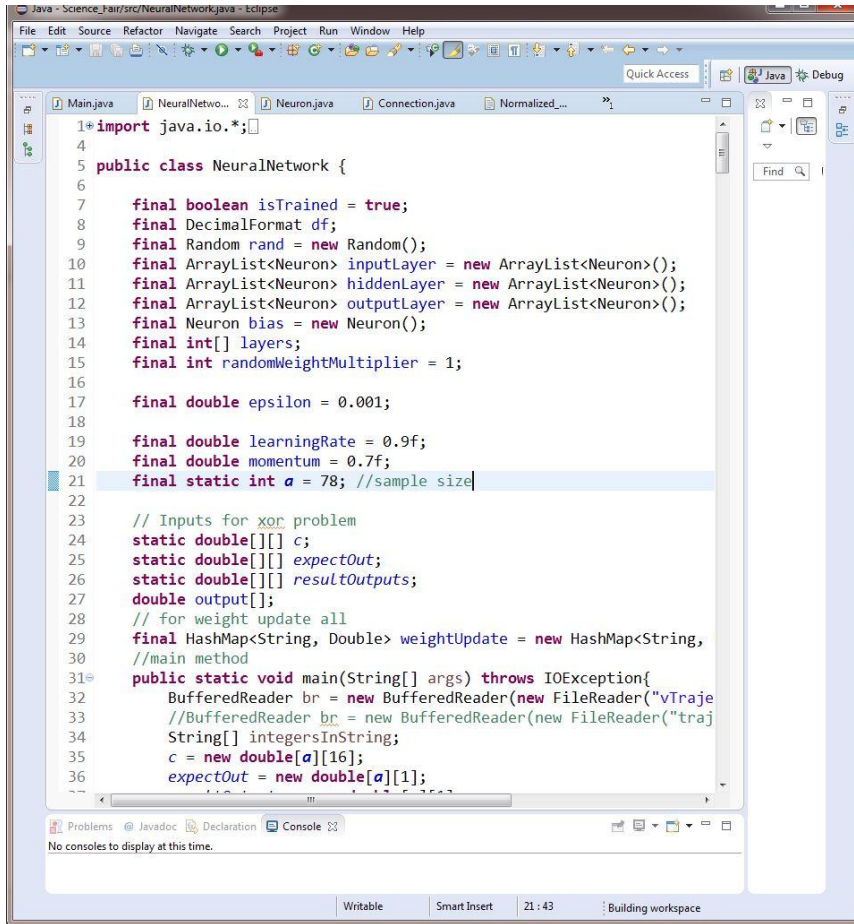
The neuron class was implemented so that neuron objects could be used. These neurons contained their net value and post-activation values.

The connections class was implemented so that each synapse connection would be an object. These connections contained a weight, and both endpoints of the connection

The class that initialized the artificial neural network and ran backpropagation (NeuralNetwork.java) also served as the main class. This class handled the inputs and outputs of the algorithm. The input was from a text file saved before (vTrajectory\_8\_21\_25.txt) that contained all the individual data points and parameters. The



outputs were from another text file (Normalized\_Values.txt) that contained all the outputs normalized between 0 and 1 to decrease the time it takes to train the artificial neural network using backpropagation. The program (Figure 5) outputted to System.out (the console) because of the relative ease to code and convince of immediately seeing training results rather than setting up files and configuring them. After the training process the program called the printAllWeights method to print all the weights.



```

1 import java.io.*;
2
3
4 public class NeuralNetwork {
5
6
7     final boolean isTrained = true;
8     final DecimalFormat df;
9     final Random rand = new Random();
10    final ArrayList<Neuron> inputLayer = new ArrayList<Neuron>();
11    final ArrayList<Neuron> hiddenLayer = new ArrayList<Neuron>();
12    final ArrayList<Neuron> outputLayer = new ArrayList<Neuron>();
13    final Neuron bias = new Neuron();
14    final int[] layers;
15    final int randomWeightMultiplier = 1;
16
17    final double epsilon = 0.001;
18
19    final double learningRate = 0.9f;
20    final double momentum = 0.7f;
21    final static int a = 78; //sample size
22
23    // Inputs for xor problem
24    static double[][] c;
25    static double[][] expectOut;
26    static double[][] resultOutputs;
27    double output[];
28    // for weight update all
29    final HashMap<String, Double> weightUpdate = new HashMap<String,
30    //main method
31    public static void main(String[] args) throws IOException{
32        BufferedReader br = new BufferedReader(new FileReader("vTraje
33        //BufferedReader br = new BufferedReader(new FileReader("traj
34        String[] integersInString;
35        c = new double[a][16];
36        expectOut = new double[a][1];
37    }

```

Figure 5 Artificial Neural Network Implementation in Eclipse

NeuralNetwork.java is also responsible for setting up the artificial neural network. The layers were stored as an ArrayList of neuron objects. The artificial neural network used in this study comprised of three layers; one input layer with sixteen nodes, one hidden layer with nine nodes, and one output layer with one node.

The main purpose of this class was to run the backpropagation algorithm. This algorithm works with four major steps; forward propagation, error calculation, backpropagation, and weight update. In

forward propagation, the artificial neural network

propagates forward to evaluate the error. The net value for each hidden node was found using this formula:

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + w_3 * i_3 + \dots + w_{16} * i_{16}$$

In this equation,  $w_1$  was the weighed connection between nodes  $i_1$  (Input layer, node 1) and node  $h_1$  (Hidden layer, node 1),  $w_2$  was the weighed connection between nodes  $i_2$  and node  $h_1$ . This naming pattern continues. Now that the net value for one of the hidden layer neurons was calculated, the process was repeated for all hidden layer neurons. Next, all the net values were squashed using a logistic function, and the following formula:

$$out_{h1} = \frac{1}{1 + e^{-net_{h1}}}$$

The activation was then performed on all hidden layer nodes. This feed-forward process was then repeated for the connections between the hidden layer and the output layer.

The error was calculated using the formula:

$$E_{total} = \sum (target - output)^2$$

This error was gradually reduced over time as the weights were adjusted.

Backpropagation was done by evaluating:  $\frac{\partial E_{total}}{\partial w_{145}}$

In other words, the computer needed to find how much a change in  $w_{145}$  affects the total error. However to figure out that, one must understand how changing  $w_{145}$  affects the net value of  $o_1$  (The output layer node), how changing the net value of  $o_1$  affects the post-activation value of  $o_1$ , and how changing the post-activation value of  $o_1$  affects the overall error. Expressed as a formula the equation is:

$$\frac{\partial net_{o1}}{\partial w_{145}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_{145}}$$

This was then further simplified down to the delta rule:

$$\frac{\partial E_{total}}{\partial w_{145}} = -(target_{o1} - out_{o1}) * out_{o1}(1 - out_{o1}) * out_{h1}$$

The delta rule was the equation that was implemented into the program code as evaluating the delta rule was faster and much more trivial to implement than implementing partial derivatives in Java.

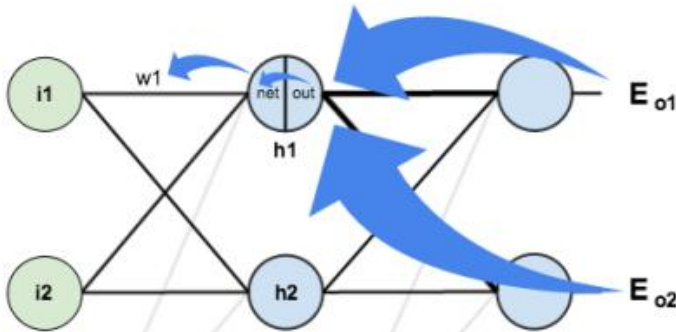


Figure 6 Illustrated Backpropagation Weight Calculation

After  $\frac{\partial E_{total}}{\partial w_{145}}$  was evaluated using the data rule, the weight can be updated. To decrease the overall error, the value obtained from the delta rule was multiplied by a learning rate (in this study, the learning rate was 0.7) and then was subtracted from the original weight.

The weight update formula for weight 145 is:

$$w_{145}^+ = w_{145} - \eta * \frac{\partial E_{total}}{\partial w_{145}}$$

This process of updating weights was repeated for all of the weights in the artificial neural network, and then the whole process of forward propagation, error calculation, backpropagation, and weight update was repeated for 100,000 times. This process was written as its own method. This was done so that each layer and node can call the method and use it.

### 2.2.3 Multiple Regression Implementation and Training

Multiple regression was implemented through the Microsoft Excel regression tool pack. Because the Analysis ToolPak was not a default feature of Microsoft Excel, it needed to be installed first. After Microsoft Excel was opened, the “File” tab in the ribbon was selected. Under that, “Options” was selected to open the Excel Options menu. The check box in front of "Analysis ToolPak" in the list of available add-ins was checked off, and the “OK” button was clicked to turn on the Analysis ToolPak.

After the Analysis ToolPak was installed, the Microsoft Excel spreadsheet created during data processing was loaded in. Each column corresponded to one parameter. Then, the top ribbon was switched to the “Data” tab and the “Data Analysis”

item was selected. This opened a small menu, and from that menu, “Regression” was selected. After selecting “Regression” and clicking on the “OK” button at the right, another menu appeared. In this menu, the location of the cells that contained the expected outputs was typed into the “Input Y Range” field within the Input section of the menu. The location of the cells that contained all 16 input parameters were selected and typed into the “Input X Range” field on the menu.

The “New Worksheet Ply” button was selected to place the results in a new sheet within the current workbook. After clicking on the "OK" button at the right of the menu, the computer processed the data and found a line of best fit.

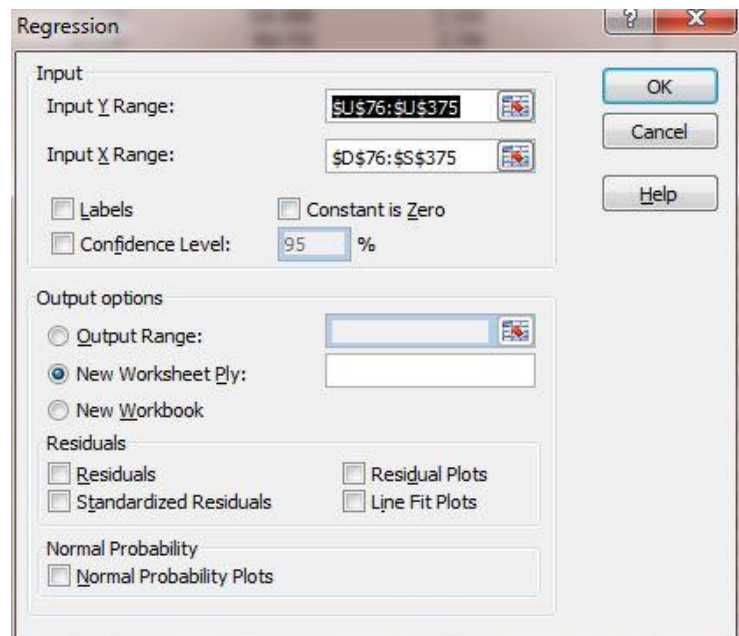


Figure 7 Multiple Regression Implementation in Microsoft Excel

### 2.2.4 Out-of-Sample Prediction

After training each algorithm with the training data, the remaining data set (approximately 20% of the data) was given to each algorithm for out-of-sample predictions. This phase of the study was done to evaluate the accuracy of the predictions generated by each algorithm.

For the artificial neural network, the previously trained weights were inserted back into the algorithm. Instead of running backpropagation 100,000 times, the program was set to propagate forward just once with the previously trained weights. This process was repeated for all 78 lines of data within the prediction set and the algorithm made a prediction for each line of data. This prediction was then de-normalized and compared to the actual values.

For multiple regression, each line of data was inputted. Each parameter was multiplied by its respective weight found during training and then added to a total sum. The intercept-value that was also found earlier in training was added to the total sum. This total sum was the prediction for that line of data. This process was repeated for all remaining observations, and the algorithm made a prediction for each observation. This number was then compared to the actual values.

## 3. Results

### 3.1 Artificial Neural Network Training

Figure 8 shows the actual outputs and prediction outputs made during training. Vehicle acceleration was on the y-axis and the timestamp was on the x-axis. The sum of squared errors for training was 13.09. The actual data set featured many spikes in vehicle acceleration ranging from  $3.5\text{m/s}^2$  to  $-3.5\text{m/s}^2$ . The neural network predictions only ranged from approximately  $1.5\text{m/s}^2$  to  $-1.5\text{m/s}^2$ .

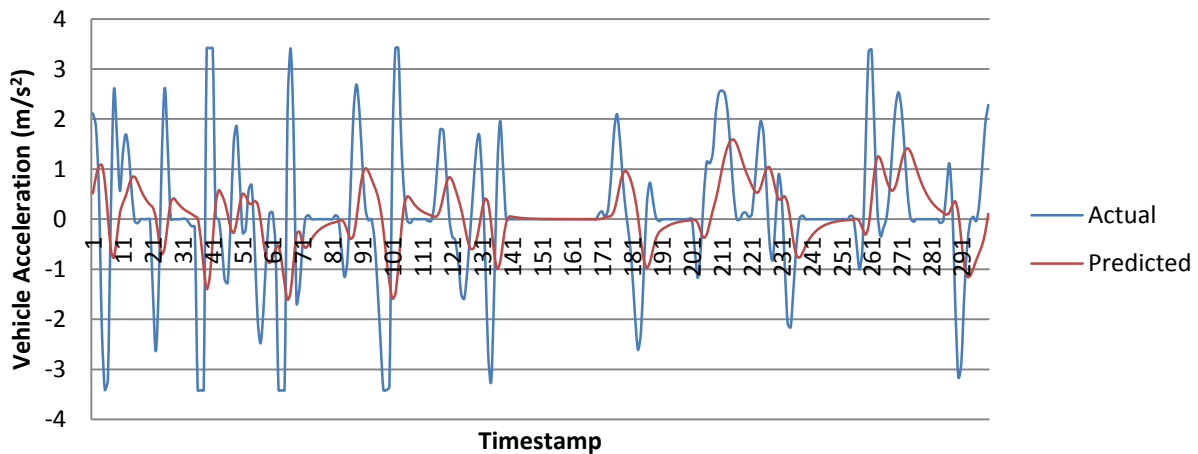


Figure 8 Artificial Neural Network Training Results

### 3.2 Multiple Regression Training

Figure 9 shows the actual outputs and prediction outputs made during training. Vehicle acceleration was on the y-axis and the timestamp was on the x-axis. The sum of squared errors was 19.71. The multiple regression predictions had a range from  $2.5\text{m/s}^2$  to  $-2.5\text{m/s}^2$ . Multiple regression also predicted many spikes in a period of no acceleration between timestamps 140 to 167, and also during timestamps 139 to 152.

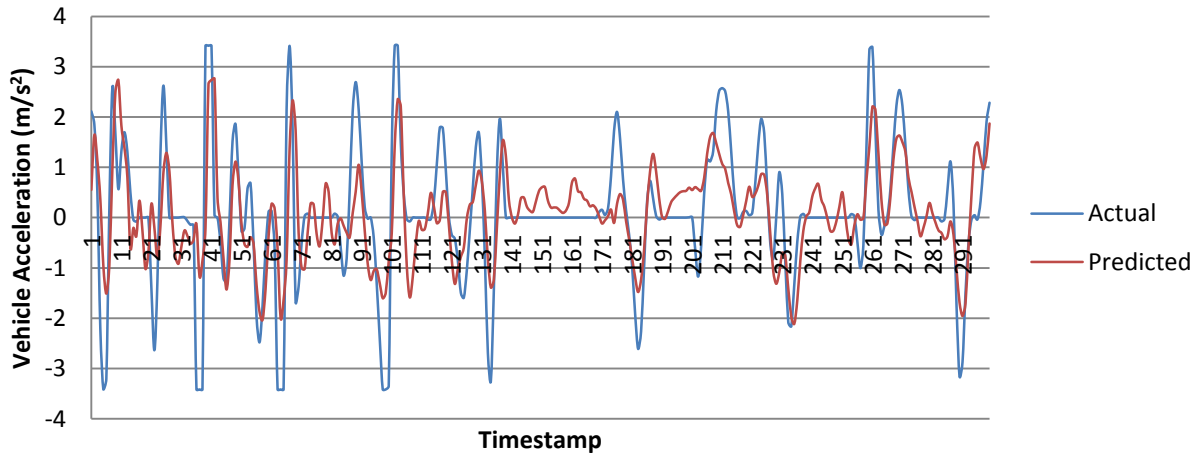


Figure 9 Multiple Regression Training Results

### 3.3 Out-of-Sample Prediction Results

Figure 10 shows the predictions made by both algorithms with the prediction set of data that comprised of approximately 20% of the total data. The data consisted of 78 timestamps, and the data ranged from  $3.5\text{m/s}^2$  to  $-3.5\text{m/s}^2$ . Multiple regression predictions ranged from  $2.5\text{m/s}^2$  to  $-5\text{m/s}^2$ . Artificial neural network predictions ranged from  $1.5\text{m/s}^2$  to  $-1\text{m/s}^2$ .

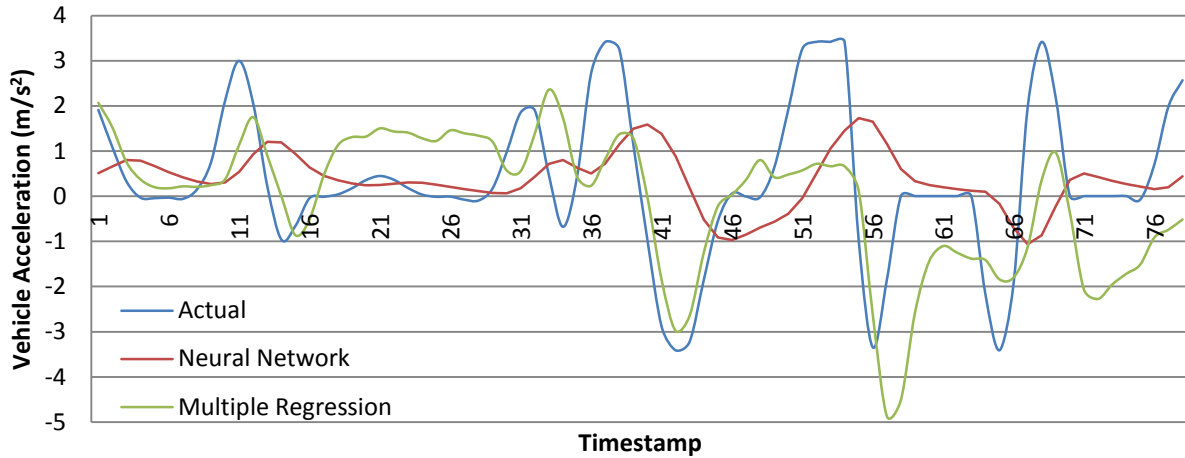


Figure 10 Algorithm Out-of-sample Predictions

### 3.4 Statistical Test Results

Based on out-of-sample prediction, statistical tests were performed to evaluate the performance of the algorithms. The mean of all three data sets (actual, artificial neural network predicted, and multiple regression predicted) were calculated. The standard deviation was calculated for all three sets. T scores were also calculated using the equation. Furthermore, a percentage of correct direction was calculated. If a vehicle's acceleration increases and the algorithm also predicts the vehicle's acceleration increasing, then it is counted as correct direction. The same principle applies for deceleration. This number was then calculated as a percentage. A table of these findings can be seen in Table 1.

All units $\text{m/s}^2$	Average	Standard deviation	Number of Observations	T score	% Correct Direction
Neural network	1.22	1.99	78	0.02	49%
M-Regression	-0.21	4.88	78	-1.73	64%
Actual Data	1.21	5.36	78		

Table 1 Statistical Test Results

### 3.5 Cumulative Absolute Errors

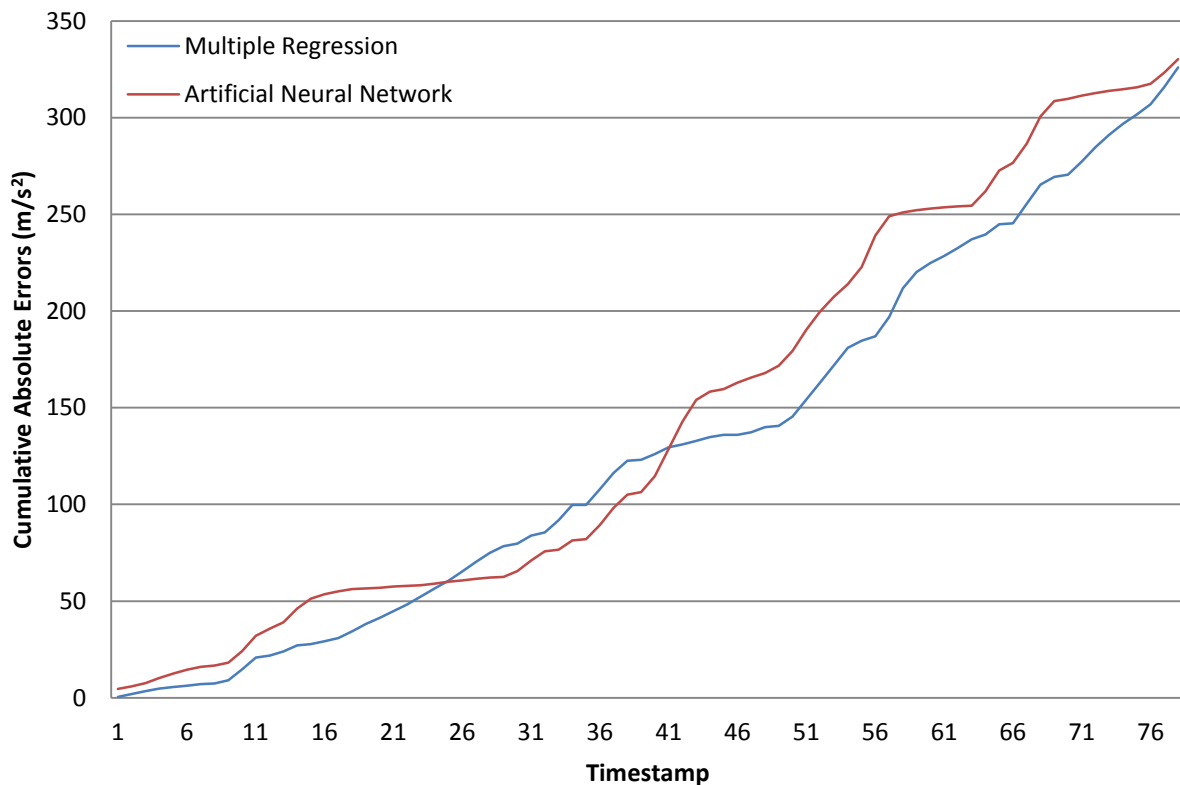


Figure 11 Cumulative Errors

After the predictions were made, the cumulative absolute errors were calculated for the prediction set. The cumulative absolute errors were found by first subtracting the predicted value from the actual value, and then taking the absolute value of that number. This number was then added to a total sum that began at zero. This process was repeated for all 78 timestamps. The artificial neural network started with a cumulative error of  $4.59\text{m/s}^2$  and gradually rose to  $330.26\text{m/s}^2$  as each timestamp's error was added. Multiple regression started with a cumulative error of  $0.53\text{m/s}^2$  and gradually rose to  $325.87\text{m/s}^2$ . (Figure 11)

#### 4. Discussion and Conclusions

Throughout this study, an artificial neural network and multiple regression were closely examined for their accuracy of vehicle acceleration predictions. It was originally hypothesized that the artificial neural network would perform better because human driver behavior is often influenced by a variety of factors, some more than others, and seems to not have any correlation between data points. Because a neural network is better at correlating non-linear data with seemingly no correlation, it was hypothesized that a neural network would more accurately predict vehicle acceleration. To accurately evaluate the performance of both algorithms, many statistical techniques and tests were utilized. The significance of the results of these tests are described in detail below.

The artificial neural network had an average of  $1.22\text{ m/s}^2$  and a standard deviation of  $1.99\text{ m/s}^2$ . Multiple regression had an average of  $-0.21\text{ m/s}^2$  and a standard deviation of  $4.88\text{ m/s}^2$ . This is compared to the validation set, which had an average of  $1.21\text{ m/s}^2$  and a standard deviation of  $5.36\text{ m/s}^2$ . To accurately compare predicted vehicle acceleration means to actual vehicle acceleration means, the t-test value was calculated. T-score for the neural network was 0.02 and -1.73 for multiple regression. For a sample size of 50 to 100 observations at a 90% confidence level, the critical values were 1.68 and -1.68 (Table 2). A null hypothesis and an alternate hypothesis were adopted. The null hypothesis ( $H_0$ ) being that the predicted means equals the actual means. The alternate hypothesis ( $H_1$ ) then stated that the predicted means do

$1 - \alpha$	.80	.90
$t : n-1=5$	1.48	2.02
$t : n-1=15$	1.34	1.75
$t : n-1=25$	1.32	1.71
$t : n-1=35$	1.31	1.69
$t : n-1=50$	1.30	1.68
$t : n-1=100$	1.29	1.66

Table 2 Critical Value Table

not equal the actual means. Because the artificial neural network had a t-score within the 90% confidence critical value of 1.68, it is said that there is not enough evidence to reject the null hypothesis. This meant that the predicted means were very close to the actual means. On the other hand, multiple regression had a t-score -1.73 that was outside of the 90% critical value of -1.68 ( $-1.73 < -1.68$ ). In this case,  $H_0$  was rejected for multiple regression. This meant that the predicted means most likely did not equal the actual means, which is an indicator that the artificial neural network performed better than multiple regression.



As for predicting if a vehicle accelerates or decelerates, the artificial neural network predicted correctly 49% of the time while multiple regression predicted correctly 64% of the time. This is an indicator that multiple regression performs better than an artificial neural network at predicting only acceleration direction.

Finding the cumulative absolute errors was another approach at evaluating algorithm performance. The artificial neural network's cumulative absolute errors amounted to 330.26m/s<sup>2</sup> while multiple regression's cumulative absolute errors amounted to 325.87m/s<sup>2</sup>. This meant that the artificial neural network performed only slightly better at minimizing errors. However, this difference is very small and it can be concluded that the artificial neural network and multiple regression had similar errors, thus having similar performance.

The shape of the algorithm out-of-sample predictions graph depicted in figure 10 were also evaluated and taken into consideration. From just the graph, it can be concluded that the artificial neural network has smoother and more human-like predictions, while multiple regression does a better job of predicting sudden and abnormal changes in vehicle acceleration.

It can be concluded from the many statistical tests and approaches, and some non-statistical methods, that both algorithms had similar performance but have different strengths and weaknesses. This causes the hypothesis that was originally formulated to only be partially correct. While the artificial neural network did perform well, multiple regression also performed at a comparable level with the artificial neural network.

## 5. Future Work and Future Applications

Throughout the study, many assumptions were made regarding artificial neural network hyper-parameters (number of layers, number of nodes, learning rate). Changing many of these hyper-parameters may have an effect on the accuracy of the predictions generated by the algorithm. The multiple regression implementation used in this study limited the amount of input factors to 16. Adding more input factors may also improve the accuracy of the predictions.

In the future, another algorithm that takes the strengths of both algorithms used in this study will yield better predictions. It has the potential to be very beneficial in saving lives and making road travel much faster

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer NN	

Table 3 Summary of activation functions



and more efficient. This algorithm would also fully realize the goals of an autonomous advanced driver assistance system that would have the capability to work in all weather and road conditions.

Experiments in the future can be done on using different types of activation functions and different machine learning algorithms such as Support Vector Machine, and Hidden Markov Models (Table 3) to predict vehicle acceleration.

## **6. Bibliography**

Brown, I. D., & Groeger, J. A. (1988). Risk perception and decision taking during the transition between novice and experienced driver status. *Ergonomics*, 31(4), 585-597.

Freedman, D. A. (2009). *Statistical Models: Theory and Practice*. Cambridge: Cambridge University Press.

LeCun, Y., Yoshua, B., & Hinton, G. (2015). Deep Learning. *Nature*, 521(7553), 436-444. doi:10.1038

McKnight, A. J., & McKnight, A. S. (2003). Young novice drivers: careless or clueless?. *Accident Analysis & Prevention*, 35(6), 921-925.

Rencher, A. C., & Christensen, W. F. (2012). *Methods of multivariate analysis*. Hoboken, NJ: Wiley.

Rojas, R. (1996). *Neural Networks - A Systematic Introduction*. Berlin: Springer-Verlag.

Shope, J. T., & Bingham, C. R. (2008). Teen driving: motor-vehicle crashes and factors that contribute. *American Journal of Preventive Medicine*, 35(3), S261-S271.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10), 1550-1560. doi:10.1109/5.58337