In the [previous post](https://blog.alpha-analysis.com/2019/09/r-neural-network.html) We fitted a neural network to the cars\_19 dataset using the [neuralnet](https://cran.r-project.org/web/packages/neuralnet/index.html) package.  In this post We are going to use TensorFlow to fit a [deep neural network](https://www.tensorflow.org/api_docs/python/tf/estimator/DNNRegressor) using the same data.

The main difference between the neuralnet package and TensorFlow is TensorFlow uses the [adagrad](https://en.wikipedia.org/wiki/Stochastic_gradient_descent#AdaGrad) optimizer by default whereas neuralnet uses [rprop+](https://en.wikipedia.org/wiki/Rprop)  Adagrad is a modified stochastic gradient descent optimizer with a per-parameter [learning rate.](https://en.wikipedia.org/wiki/Learning_rate)

The data which is all 2019 vehicles which are non pure electric (1253 vehicles) are summarized in previous posts below.

str(cars\_19)  
'data.frame': 1253 obs. of 12 variables:  
 $ fuel\_economy\_combined: int 21 28 21 26 28 11 15 18 17 15 ...  
 $ eng\_disp : num 3.5 1.8 4 2 2 8 6.2 6.2 6.2 6.2 ...  
 $ num\_cyl : int 6 4 8 4 4 16 8 8 8 8 ...  
 $ transmission : Factor w/ 7 levels "A","AM","AMS",..: 3 2 6 3 6 3 6 6 6 5 ...  
 $ num\_gears : int 9 6 8 7 8 7 8 8 8 7 ...  
 $ air\_aspired\_method : Factor w/ 5 levels "Naturally Aspirated",..: 4 4 4 4 4 4 3 1 3 3 ...  
 $ regen\_brake : Factor w/ 3 levels "","Electrical Regen Brake",..: 2 1 1 1 1 1 1 1 1 1 ...  
 $ batt\_capacity\_ah : num 4.25 0 0 0 0 0 0 0 0 0 ...  
 $ drive : Factor w/ 5 levels "2-Wheel Drive, Front",..: 4 2 2 4 2 4 2 2 2 2 ...  
 $ fuel\_type : Factor w/ 5 levels "Diesel, ultra low sulfur (15 ppm, maximum)",..: 4 3 3 5 3 4 4 4 4 4 ...  
 $ cyl\_deactivate : Factor w/ 2 levels "N","Y": 1 1 1 1 1 2 1 2 2 1 ...  
 $ variable\_valve : Factor w/ 2 levels "N","Y": 2 2 2 2 2 2 2 2 2 2 ...

To prepare the data to fit the neural network, TensorFlow requires categorical variables to be converted into a dense representation by using the **column\_embedding()** function.

cols <- feature\_columns(  
 column\_numeric(colnames(cars\_19[c(2, 3, 5, 8)])),  
 column\_embedding(column\_categorical\_with\_identity("transmission", num\_buckets = 7),dimension = 1),  
 column\_embedding(column\_categorical\_with\_identity("air\_aspired\_method", num\_buckets = 5),dimension=1),  
 column\_embedding(column\_categorical\_with\_identity("regen\_brake", num\_buckets = 3),dimension=1),  
 column\_embedding(column\_categorical\_with\_identity("drive", num\_buckets = 5),dimension=1),  
 column\_embedding(column\_categorical\_with\_identity("fuel\_type", num\_buckets = 5),dimension=1),  
 column\_embedding(column\_categorical\_with\_identity("cyl\_deactivate", num\_buckets = 2),dimension=1),  
 column\_embedding(column\_categorical\_with\_identity("variable\_valve", num\_buckets = 2),dimension=1)  
)

Similar to the neural network I fitted using **neuralnet()**, I am going to use two hidden layers with seven and three neurons respectively.

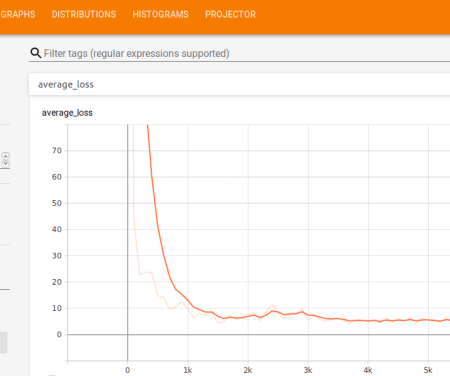
Train, evaluate, and predict:

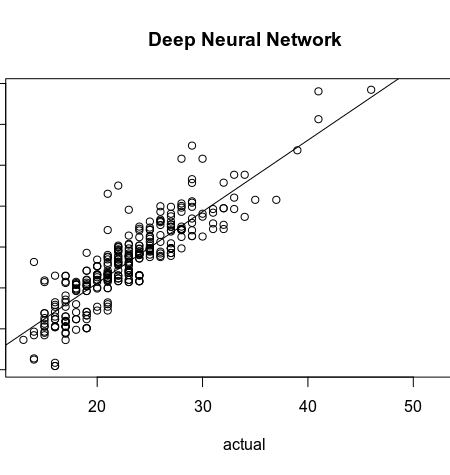
#Create a deep neural network (DNN) estimator.  
model <- dnn\_regressor(hidden\_units=c(7,3),feature\_columns = cols)  
  
set.seed(123)  
indices <- sample(1:nrow(cars\_19), size = 0.75 \* nrow(cars\_19))  
train <- cars\_19[indices, ]  
test <- cars\_19[-indices, ]  
  
#train model  
model %>% train(cars\_19\_input\_fn(train, num\_epochs = 1000))  
  
#evaluate model  
model %>% evaluate(cars\_19\_input\_fn(test))  
  
#predict  
yhat <- model %>% predict(cars\_19\_input\_fn(test))  
  
yhat <- unlist(yhat)  
y <- test$fuel\_economy\_combined

postResample(yhat, y)  
RMSE Rsquared MAE   
1.9640173 0.8700275 1.4838347

The results are similar to the other models and **neuralnet()**.

I am going to look at the error rate in [TensorBoard](https://www.tensorflow.org/tensorboard) which is a visualization tool.  TensorBoard is great for visualizing TensorFlow graphs and for plotting quantitative metrics about the execution of the graph.  Below is the mean squared error at each iteration.  It stabilizes fairly quickly.  Next post I will get into TensorBoard in a lot more depth.

[](https://i2.wp.com/s3-us-west-1.amazonaws.com/alpha-analysis.com/Pictures/TensorFlow_Neural_Network/neural_network_tensorBoard.png?ssl=1)

[](https://i1.wp.com/s3-us-west-1.amazonaws.com/alpha-analysis.com/Pictures/TensorFlow_Neural_Network/tensorflow_dnn.png?ssl=1)