Capstone Project

Rishabh Goyal

Machine Learning Engineer Nanodegree

April 6th, 2018

Definition

Project Overview

The project is related to automotive domain. Now a days, Self Driving Car is gaining so much popularity and many companies are proposing their ways of doing it the correct way. Predicting the driver's behavior comes handy when solving such problem. In this activity, the car mimics the driver's behavior and take actions accordingly.

A recent trend in a gear shift schedule design is to focus on better interpretation of the driver's intention and estimation of road environment. Traditional gear shift schedules are typically designed as a static map on the plane of the driver command (throttle/pedal position) and vehicle speed. This project attempts to predict the gear shifts using Machine Learning by interpreting the driver's intention and the environment.

Problem Statement

The goal is to predict the gear position of the car driven in different circumstances, the task involved are following:

1. Download and preprocess the Car Trip Data Log[1] publicly available dataset on Kaggle.
2. Converting the time series to supervised learning problem.
3. Train a classifier that can determine the next gear position.
4. Validate the classifier using 0.5% of the data.
5. Test the classifier using 98.5% of the data.

1 The dataset is available at https://www.kaggle.com/vitorrf/cartripsdatamining

Evaluation Metrics

The evaluation metric for the model will be number of errors between original gear positions and predicted one. The model will be evaluated on per class micro averaged precision and individual class precision,

1. Precision (Class 1) = TP1/ (TP1 + FP1)
2. Micro Averaged Precision = (TP1 + TP2) / (TP1 + TP2 + FP1 + FP2)

For the purposes of this project, there will be no partial credit given for identifying nearer gear position. If the gear position is not equal to the original one, it will be treated as error.

Analysis

Data Exploration

The Car Trip Data Log Dataset contains data acquired from a driver's driving a vehicle through various conditions. The collected data will be used in an attempt to predict driver's behavior in order to improve gearbox control.

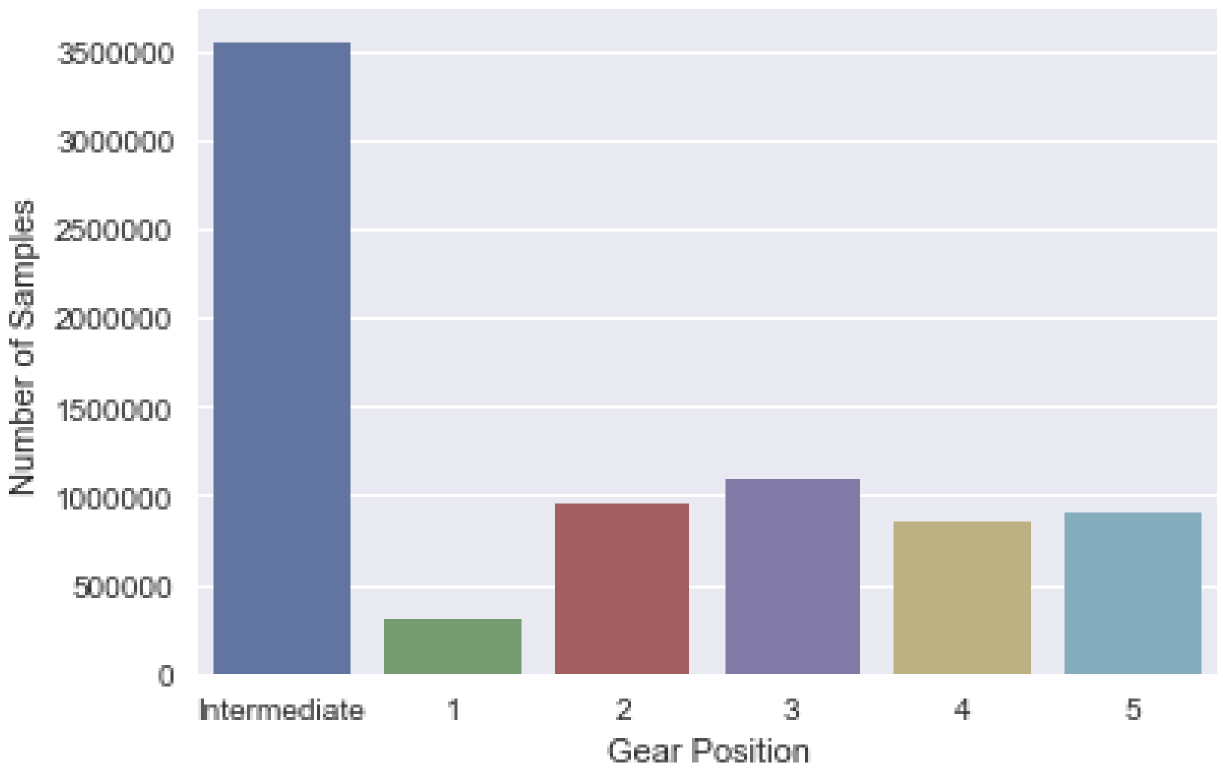
Below are the features available in this dataset. Out of which Shift number is the target variable for this project. Out of these 17 features, from 2nd to 8th we have vehicle parameters and from 9th to 17th we have driver informed parameters.

Input Variables - Vehicle Speed (in m/s) , Engine Load (% of max power) , Total Acceleration (m/s^2) , Engine RPM , Pitch , Lateral Acceleration (m/s^2) , Passenger count (0- 5) , Car's Load (0-10) ,Air Conditioning Status (0-4) ,Window Opening (0-10) , Radio Volume (0-10) , Rain intensity (0-10) , Visibility (0-10) , Driver's wellbeing (0-10)

Target Variable - Shift number (0, 1, 2, 3, 4 and 5)

Exploratory Visualization

The number of samples of different shift position are shown in the graph below. We have enough number of samples to train our model and then do the evaluation of it.

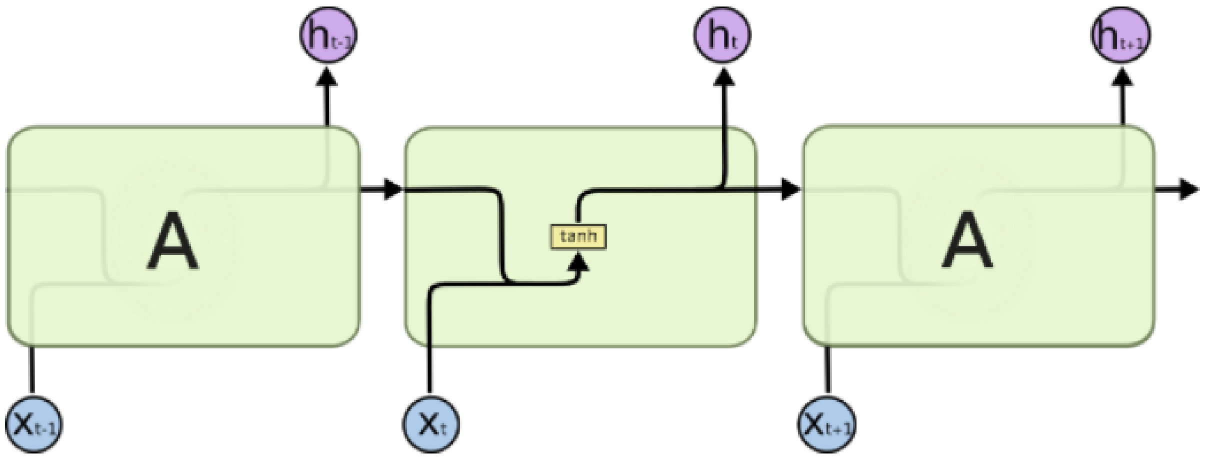


Algorithms and Techniques

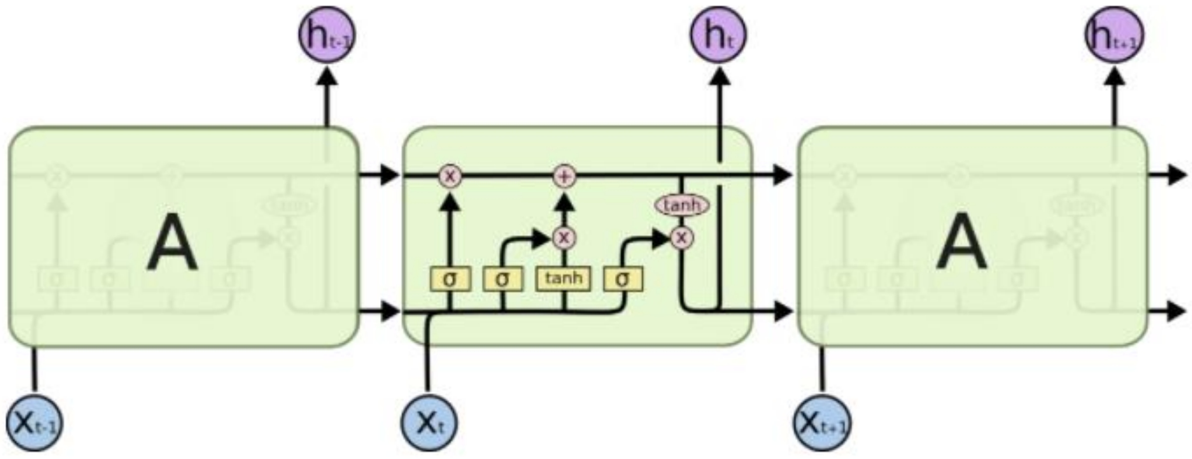
Traditional neural networks doesn't have memory element. For example, imagine you want to classify what kind of event is happening at every point in a movie. It's unclear how a traditional neural network could use its reasoning about previous events in the film to inform later ones.

Recurrent neural networks address this issue. They are networks with loops in them, allowing information to persist. Long Short Term Memory networks – usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997), and were refined and popularized by many people in following work.1 They work tremendously well on a large variety of problems, and are now widely used.

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer. This is shown in the following diagram.



LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.



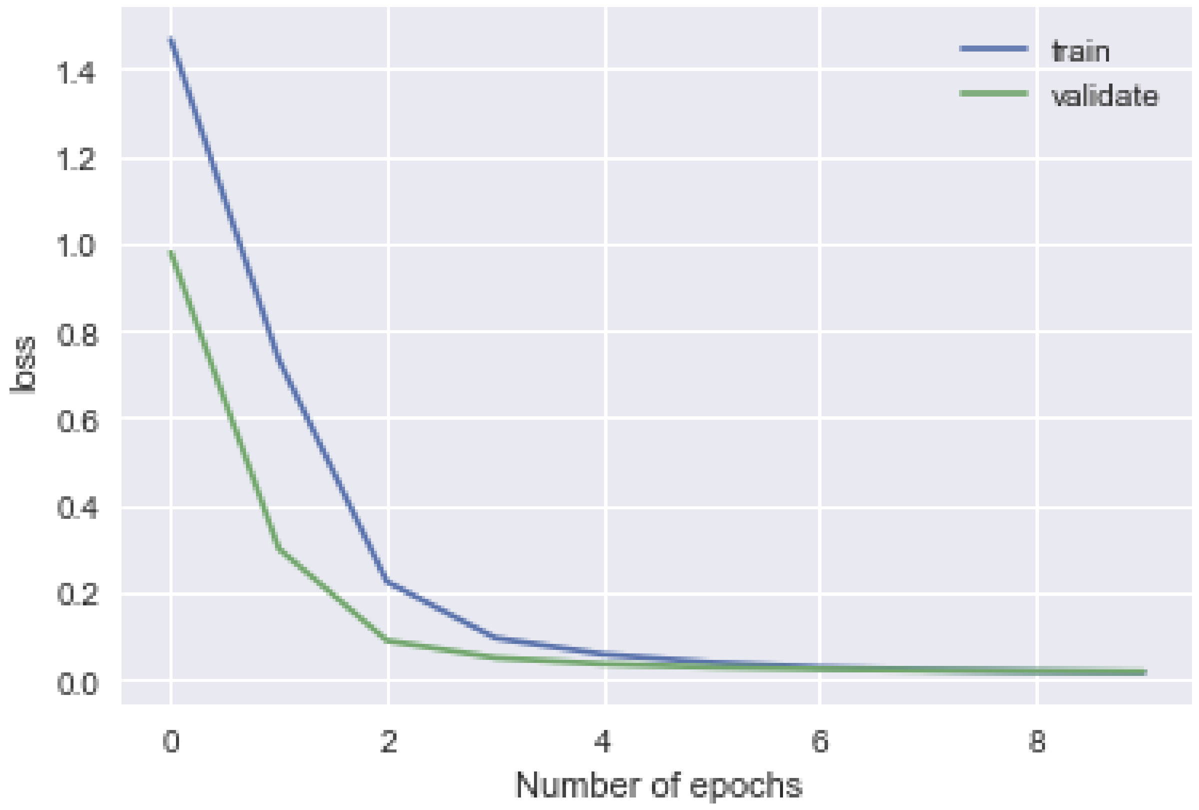
The classifier here is created using a Long Short Term Memory network of 50 units followed by a neural network of 6 nodes with sigmoid activation function, categorical cross entropy loss and adaptive moment estimation optimizer. The algorithm outputs the gear position. The output is captured as a two dimension array that has 6 rows, each representing existence of the a particular gear position i.e. if gear is at 0th location, then 1st column would be 1 and if gear is at 1st location, then 2nd column would be 1 so on and so forth.

Reference taken from blogpost by Christopher Olah http://colah.github.io/posts/2015-08-Understanding-LSTMs/

The following parameters can be tuned to optimize the classifier:

1. Training Parameters
    a. Training length (number of epochs)
    b. Batch size (how many samples to feed at once during a single training step)
    c. Learning rate (how fast to learn, this can be dynamic)
    d. Momentum
2. Neural network architecture
    a. Number of layers
    b. Layer types (LSTM, fully connected)
    c. Layer parameters
3. Preprocessing parameters

During training, both the training and the validation sets are loaded into RAM. After that, test is done of the preloaded test set. The result of the training process is shown as follows.



Benchmark

A Benchmarking model could not be found for this activity hence, I kept the benchmark to predict the Gear Position accurately above 85% of the time.

Methodology

Data Preprocessing

The preprocessing done in the "Preprocess.py" file which consists of following steps:

1. Outliers are identified and removed.
2. Target variable which is a categorical variable is converted to one-hot encoded vector of 6 columns i.e. ShiftNumber_0, ShiftNumber_1, ShiftNumber_2, ShiftNumber_3, ShiftNumber_4 and ShiftNumber_5
3. Dropped the time axis.
4. Converted the values to float
5. Using MinMaxScaler provided by sklearn, scaled the values with minimum as 0 and maximum as 1.
6. Converted the data from series to supervised[2] kind using shift() operation provided by pandas.
7. Dropped the columns which we don't want to predict.

Implementation

The implementation process can be split into two main stages:

1. The classifier training stage
2. The inference stage

2 The code for series to supervised is taken from https://machinelearningmastery.com/multivariate-time-series-forecasting-lstms-keras/

During the first stage, the classifier was trained on the preprocessed training data. This was done in a Jupyter notebook (titled “Create Classifier”), and can be further divided into the following steps:

1. Load both the training and validation data into memory, preprocessing them as described in the previous section.
2. Splitting train and validation data based on % of training samples to be used for training and validation. The function train_test_split() provided by sklearn cannot be used as we don’t need random split for a time series.
3. Splitting the inputs and outputs from the data. Here we have 6 output variables discussed in the earlier section.
4. Reshaping the input to be 3D [samples, timsteps, features].
5. Define the network architecture and training parameters.
6. Train the network, logging the validation/training loss and the validation accuracy.
7. Plot the logged values.
8. If the accuracy is not high enough, return to step 3

The following complications were faced

1. The amount of data to be used for training purpose. I overcame with this issue by providing different amount of data for training, then test it on rest of the data. When I found that the precision was quite accurate for the test data, I chose that combination.

Refinement

1. Firstly I tried this problem as a regression problem and found that the results were not that accurate. Then I changed the problem to a classification problem and found it to be a better solution.
2. After noticing the amount of time it takes to train this model. Also, the amount of data I had for the problem was huge. So, only small amount of data was used to train it, so that the training can be done in small span of time.

Results

Model Evaluation and Validation

During development, a validation set was used to evaluate the model.

The model generalizes well on the test set. We can trust the model as the amount of data used for test is 98.5% of the complete dataset. The test data contains 7532285 Data points which is not a small number. Out of these only 10,499 samples were misclassified.

The final architecture and hyper parameters were chosen because they performed the best among the tried combinations.

1) The number of LSTM units used are 50
2) The number of output units of Dense Layer are 6
3) The training runs for 0.1 % samples as there was huge amount of samples, this much data was enough to train the model to obtain the desired accuracy.
4) The number of validation samples used are 0.5% of the overall data.
5) The training is done for 10 epochs on a batch of 256.

Justification

Using the model which is trained. I was able to obtain micro averaged precision of 0.9986. I also calculated the individual precision for each class which is shown below:

ShiftNumber0 – Precision - 0.99849

ShiftNumber1 – Precision – 0.99525

ShiftNumber2 – Precision – 0.99826

ShiftNumber3 – Precision – 0.99889

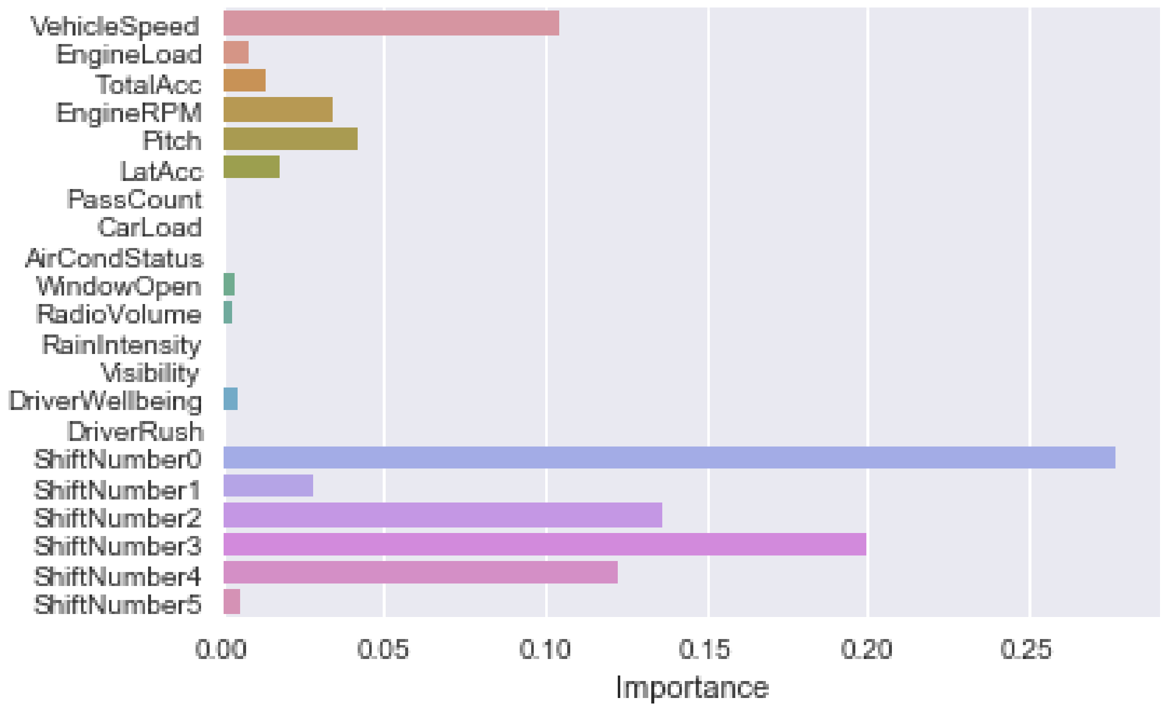
ShiftNumber4 – Precision – 0.99911

ShiftNumber5 – Precision – 0.99964

Conclusion

Free-Form Visualization

To find out the feature importance, I trained a random forest classifier on the data and found out the feature importance. A graph of the feature importance is shown below. As we can observe vehicle speed, engine load, total acceleration, engine rpm, pitch, lateral acceleration are the vehicle features which plays major importance in predicting the gear position. The passenger related features plays very less roll in finding the gear position which seems to be ok. The major roll is played by the last gear positions which is very evident.



Reflection

The process used for this project can be summarized using the following steps:

1. An initial problem and relevant, public dataset were found.
2. The data was downloaded and preprocessed

3. A benchmark was created for the classifier
4. The classifier was trained using the data (multiple times, until a good set of parameters were found)
5. The inference is drawn from the test data and metrics was obtained.

Things which I learnt while doing this project are as follows:

1. Using Keras for creating deep learning models
2. Feature engineering and cleaning the data.
3. Finding the best parameters to tune the model.

The biggest challenge was to work on a new data which I have not worked before or no one else has worked before. Generally we see problems where many people would have done some or the other experiments. For this problem I couldn't find any reference material which will help me understanding the data better. I overcome the problem by doing exploratory data analysis on the data, removing outliers, creating correlation plots etc.

Improvement

1. To achieve best results for optimal gear shift, data from a car which is driven by trained chauffeur and then train our model on that data.
2. Currently the data being taken was from 1 driver. Similarly data from multiple drivers can be used to tune the model at its best.