Taxi Driver Trajectory Classification using Deep Neural Networks

Jannik Haas
WPI Data Science
Worcester Polytechnic University
100 Institute Road, Worcester MA, 01609
jbhaas@wpi.edu

1 INTRODUCTION

Sequences or trajectories are part of our everyday life and in big cities, taxi trajectory make up a large part of that. Being able to analyze these trajectories to distinguish one driver from another or other tasks can be done using different Neural Network structures. In this paper I look at records from five taxi drivers over several months and build a model that is able to classify the driver based on a single days' trajectory. These types of classification problems are very common and help to show the power of different Neural Networks.

2 PROPOSAL

In this dataset, each record is a single location of a taxi driver at a specific time of the day with the following features:

Plate	Longitude	Latitude	Time	Status
The plate of the taxi driver (in our case 0-5)	The longitude of the location	The latitude of the location	The date and time of the stamp	0 - empty/seeking passenger 1 - carrying passenger

I propose using a simple fully connected deep neural network which takes in features extracted from each driver's trajectory and learns to predict the plate of the driver that drove said trajectory. I will talk about several different attempted network structures and discuss the performances of them.

3 METHODOLOGY

3.1 DATA PROCESSING

To process the data I first grouped each trajectory by driver and day so that I had one complete trajectory for each driver. Driver behavior is suspected to be very different when seeking vs carrying passengers so I split up each days' trajectory into all seeking sub trajectories and all carrying sub trajectories. Then features were extracted from each of these sub trajectories to get the average features for each driver for each day. The latitude and longitude coordinates were also split up into 1km x 1km labeled grids such that a trajectory could be classified as a sequence of grids. To avoid any missing valued I dropped any records with zero trips which were a total of five. After extracting the features I also standardized the data before training the network.

3.2 FEATURE GENERATION

To be able to characterize a taxi driver's behavior both while carrying passengers and while looking for passengers I extracted the same features for both the carrying trajectories and the seeking trajectories.

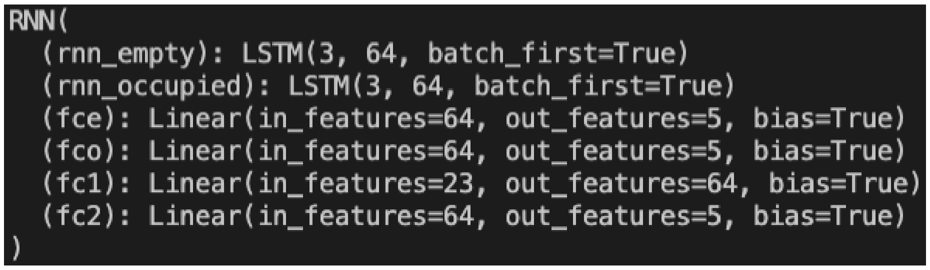
Extracted features:

- Average time seeking
- Average time carrying
- Average speed seeking
- Average speed carrying
- Average distance seeking
- Average distance carrying
- Start time
- End time
- Number of trips carrying
- Top two most frequently occupied grids carrying
- Top two most frequently occupied grids seeking

3.3 NETWORK STRUCTURE

LSTM

The first network I attempted used the following structure:



```
RNN(
  (rnn_empty): LSTM(3, 64, batch_first=True)
  (rnn_occupied): LSTM(3, 64, batch_first=True)
  (fce): Linear(in_features=64, out_features=5, bias=True)
  (fco): Linear(in_features=64, out_features=5, bias=True)
  (fc1): Linear(in_features=23, out_features=64, bias=True)
  (fc2): Linear(in_features=64, out_features=5, bias=True)
)
```

In this model each LSTM layer took in either the occupied or seeking trajectories respectively. The 3 input features for the LSTM layers were the latitude, longitude, and time. The last output of the LSTM layers were then put into a fully connected layer which output 5 learned features for both seeking and carrying trajectories. These 10 features were then concatenated with the extracted features and fed into a fully connected network. The goal here was to use the power off RNN to extract significant features from the complete carrying and seeking trajectories and then use them in combination with the extracted features to train a fully connected network. However with this model the training was very slow and the loss dropped quickly without increasing the accuracy as you can see in Figure 1.

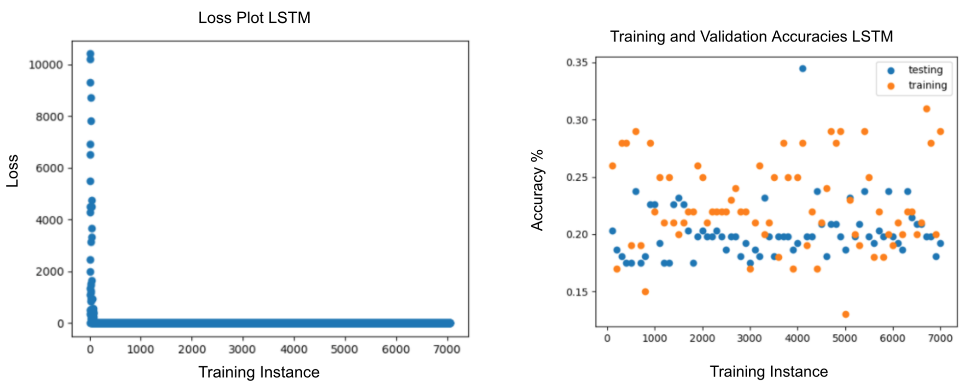
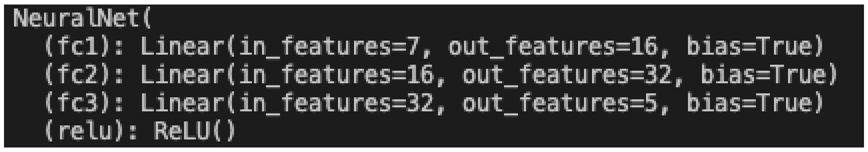


Figure 1: Loss Plot and Training and Validation Accuracies plot LSTM

Fully Connected Neural Net

After these results I decided to simplify the model by removing the LSTM layers and simply feeding the extracted features into a fully connected neural net with the following structure:



```
NeuralNet(
  (fc1): Linear(in_features=7, out_features=16, bias=True)
  (fc2): Linear(in_features=16, out_features=32, bias=True)
  (fc3): Linear(in_features=32, out_features=5, bias=True)
  (relu): ReLU()
```

The reason the input features are only 7 is because I extracted more features over time, but the best performing model was actually this one with 7 features. The model uses a relu activation function after the first two layers.

3.4 TRAINING AND VALIDATION PROCESS

For the training of the network I used online training where I update the model weights after each training sample rather than after a batch. Cross entropy loss was used as the loss function with the Adam optimizer. I experimented with different optimizers and activation functions, but Adam performed the best. I let the training run for a max of 10000 epochs, however the model always converged far before that and I used the training and validation accuracies plot to determine when to stop the training.

4 RESULTS

4.1 TRAINING AND VALIDATION RESULTS

Below are the loss plot and the training and validation accuracies plot for the fully connected neural net.

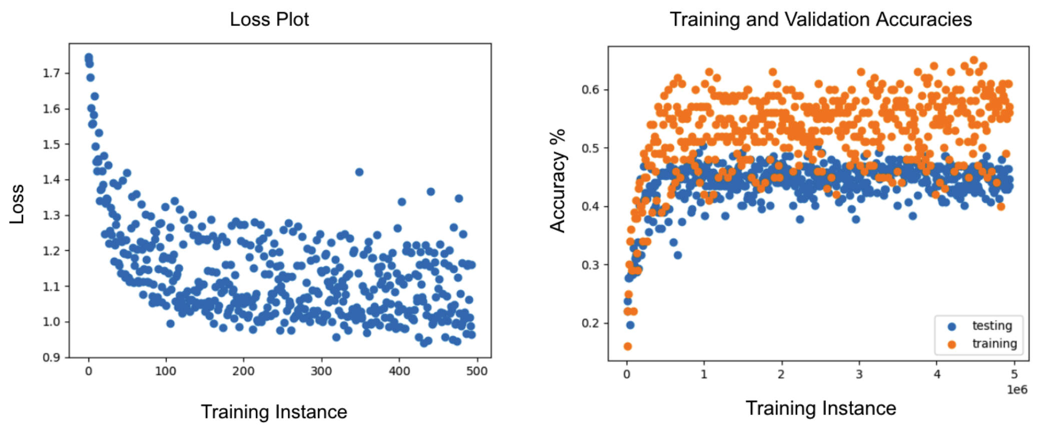


Figure 2: Loss Plot and Training and Validation Accuracies plot Fully Connected NN

As you can see from Figure 2 the training and validation accuracies converge around 1000 training instances, however the loss decreased slightly further. This is where early stopping was used to choose one of the checkpointed models rather than the final model. We can also see that the model does not suffer from overfitting.

4.2 MODEL COMPARISONS

The fully connected neural net trained much faster than the LSTM network mostly due to its simplicity and it also performed much better on both training and testing accuracy. I suspect this is due to the length of the trajectories that were used as inputs for the LSTMs. I only used the whole days seeking and carrying trajectories concatenated into one respectively. In the future I would like to experiment with using slices or single seeking and carrying trajectories as inputs to the LSTMs instead. For the fully connected linear model I experimented with different sizes and number of layers, however the simple 3 layer model performed best while most of the more complex models ended up overfitting to the training data. For these I also tried dropout which improved the results slightly but didn't overcome the overfitting problem.

4.3 HYPERPARAMETER TUNING

The learning rate was set to 1.5e^{-4} to start out with and I experimented with lower and larger learning rates from 1.5e^{-2} to 1.5e^{-6} with the best performances coming from the models trained with a learning rate of 1.5e^{-4}. I also tested different probabilities for the dropout layers when applicable, however I saw no improvement in overcoming the overfitting problem.

5 CONCLUSION

Overall, a lot of different model structures and training methods were attempted for this classification problem, but in the end the simplest model actually ended up outperforming the rest of the more complex model. This was partly due to overfitting on some of the more complex models and potentially just the more complex models not using the right activation functions etc. In the future I would like to experiment with more different feature extraction techniques for this data as well as revisit the LSTM model since I believe that could prove to be a very well performing model.

REFERENCES

“PyTorch Documentation¶.” *PyTorch Documentation - PyTorch 1.8.0 Documentation*, pytorch.org/docs/stable/index.html.

“User Guide¶.” *Scikit*, scikit-learn.org/stable/user_guide.html.

Brownlee, Jason. “How to Configure the Number of Layers and Nodes in a Neural Network.” *Machine Learning Mastery*, 6 Aug. 2019, machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/.