New York City Taxi Fare Prediction Capstone

Thomas Patino

Introduction

Neural Networks are at the forefront of innovation. Companies, the government and individuals are looking into implementing deep learning in order to obtain data-driven solutions. This capstone aims to predict the fare amount (inclusive of taxes and tolls) for a taxi ride in New York City.

The taxi industry was overtaken by ridesharing companies by the likes of Uber and Lyft. In order to gain an advantage, a company must be able to predict the prices of their competition and undercut them by offering more affordable rides. Governments are also interesting in being able to predict taxi fares because it allows them to identify communities who do not have easy access to transportation. Lastly, consumers are also interesting in knowing which transportation method can get them to where they need in the most inexpensive way. For these reasons, predicting taxi fare is a pertinent, real-world problem that can be used to apply neural networks techniques learnt during the Masters in Data Science program at GWU.

Description of Data

The data comes from a Kaggle competition sponsored by Google and Tensorflow. The dataset consists of a million taxi cab rides taken in the past four years. The data consists of the date and time the rider was picked up, the location (in latitude and longitude) of the pick-up and drop off points, the number of passengers and the fare amount.

	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
0	2009-06-15 17:26:21.0000001	4.5	2009-06-15 17:26:21 UTC	-73.844311	40.721319	-73.841610	40.712278	1
1	2010-01-05 16:52:16.0000002	16.9	2010-01-05 16:52:16 UTC	-74.016048	40.711303	-73.979268	40.782004	1
2	2011-08-18 00:35:00.00000049	5.7	2011-08-18 00:35:00 UTC	-73.982738	40.761270	-73.991242	40.750562	2
3	2012-04-21 04:30:42.0000001	7.7	2012-04-21 04:30:42 UTC	-73.987130	40.733143	-73.991567	40.758092	1
4	2010-03-09 07:51:00.000000135	5.3	2010-03-09 07:51:00 UTC	-73.968095	40.768008	-73.956655	40.783762	1
5	2011-01-06 09:50:45.0000002	12.1	2011-01-06 09:50:45 UTC	-74.000964	40.731630	-73.972892	40.758233	1

Description of Deep Learning Network and Training Algorithm

This is a regression problem. The goal to predict an outcome variable that is a continuous value (fare amount in dollars). For this study Keras will be used, with Tensorflow as the backend, to create the predicting modeling. This Kaggle competition encourages participants to use Tensorflow (the sponsors), allowing us to directly compare this model and model submitted by others. This is a feed forward, multi-layer perceptron with back propagation.

This study uses the Keras sequential model, which is a linear stack of layers. The model starts off by specifying the input shape. We use the input_shape argument to automatically extract the size given the dataset. The first input layer will have 18 nodes because it is the same size of the input variables. The structure of the neural network is composed of an input layer, three hidden layers, and one output layer. The activation function of the hidden layers is Rectified Linear Unit (ReLU). An activation function checks the output value calculated by a neuron and decides whether the connecting neuron should be fired or not. ReLu activates when it is an output of more than zero and does not activate when the value is zero or less. ReLU was used because the fare amount cannot be a negative value.

Adam optimizer was used because it adds momentum and some noise to the weights to keep it from getting stuck in local minimum. Researchers at Stanford University also recommend using Adam optimizer for regression problems using Tensorflow.

The model starts with randomly initialized weights and biases. The training dataset is fed forward through the neural network and finds the predicted fare amount. Since the weights and biases are randomly initialize, the fare amount prediction will not be good estimate. After the estimate is predicted, we calculate the errors against the actual value. After calculating the error, we calculate the gradient of the error with respect to the weights and use the gradient descent to update the weights and bias. By adjusting the weights and bias to reduce the error, the network is learning.

Mini-batch was used, splitting the training sample into smaller packages to calculate the error and update coefficients. Mini-batch was used because it is not as computationally intensive as updating the error after each sample (stochastically). We also do not use batch because we cannot load the entire training sample into memory to calculate the error after all training examples have been evaluated. Batch normalization was used to transform the values to have a mean close to zero and a standard deviation close to one. This speeds up learning and reduces overfitting due to the regularization effect.

The evaluation metric for the competition is mean-squared error (MSE). If the model results in a large MSE, it means that there was a large average error. MSE was chosen because, according to Kaggle, one can directly judge the performance of a model in unseen data. Therefore, MSE was chosen as the loss function. The model cycles through the entire training dataset with one forward pass and one backward pass, which is called an epoch. 200 epochs were used to train the model, which is at the point where the loss function plateaued.

Experimental Setup

The process starts by cleaning and preprocessing the dataset. We removed all entries with incomplete information. We also removed any entries with a negative fare amount. The pickup

date and time were given in an impractical format '*2009-06-15 17:26:21 UTC'*; therefore the year, month, day, and hour was extracted. We also identified the date of the week in which the ride took place, which allows us to classify rides that took place on the weekend and during rush hour. Rides with coordinates outside of New York City were deleted.

Since the dataset contains coordinates, we can calculate the distance between the pickup and drop off locations. The Haversine distance was used to determine the distance between two points given their longitude and latitudes on a sphere. Haversine distance however, only calculates the distance of a straight line between two points; which is unrealistic because New York City streets are laid in a grid. To fix this problem, the Manhattan distance was also used to calculate distance. The Manhattan distance takes the sum of the horizontal and vertical distances between points on a grid, which is a more realistic measure of the distance travelled by cabs. Calculating distances is very computationally expensive, which is why even though the original dataset contained millions of taxi cab rides, the data had to be reduced to 40,000 records in order to be able to calculate and process distances.

New York City has three major airports: La Guardia, JFK and Newark Airport. Airport pickups and drop offs make up a considerate proportion of all cab rides. Airport rides also, in some cases, have a flat fare fee and are therefore different from distance- and time-based taxi cabs. The distance between the three major airports and the pickup and drop-off points was calculated. All pickups and drop-offs that took place within a .5-kilometer radius of an airport were labeled as an airport pick up or drop off.

After cleaning up, pre-processing and adding new features, the dataset looks as such:

	fare_amount	passenger_count	year	month	day	hour	rush_hour	weekend	straightdistance	manhattandistance	jfkpickup	jfkdropoff
0	4.5	1	2009	6	15	17	0	0	1.031	1.233	9.538	10.504
1	16.9	1	2010	1	5	16	0	0	8.450	10.958	23.071	21.523
2	5.7	2	2011	8	18	0	0	0	1.390	1.907	21.691	21.807
3	7.7	1	2012	4	21	4	0	1	2.799	3.148	22.192	20.370
4	5.3	1	2010	3	9	7	1	0	1.999	2.715	21.850	21.331

Results

The mean-squared error of the model is 13.7. The prediction performs average compared to the 1,488 entries in the public leaderboard of Kaggle. This model ranks in the 55th percentile of the leaderboard.

Summary and Conclusions

Predicting the fare of a taxi ride given the date and pick up and drop off locations can be a difficult task without the proper data. A huge part of how the fare is calculated comes from travel time. The fare in New York City goes up 50 cents for every minute that the taxi is stuck in traffic traveling at less than 12 miles per hour. The dataset provided does not have travel time information and, consequently, two rides with close pick up and drop off locations can have very different fare totals.

When creating new features to add to the dataset, one must keep in mind the usefulness/predictive power of the feature versus the resources that must be spent to obtain the information. Computing distance is very computationally expensive. Calculating the Haversine, Manhattan and airport distance is one of the most resource-intensive part of the project. In the future, it must be considered whether or not having different types of distances adds pertinent information or if we must forgo them to decrease computation computing time and resources.

The model learned to lower the mean squared error at the expense of usefulness of the prediction. At one point the predicted the same optimal output regardless of the input data. While

the metrics used to judge the performance of the model are a good guiding principle, we must also keep in mind how the model learns and if its predictions have practical applications in the real world. To remedy this problem, smaller batch sizes and more epochs were used.

The model could have been improved with more nodes and more layers, but it comes at the expense of understandability. A deeper or wider neural network could have further decreased the loss, but it would have required a larger dataset and more resources. When we add more layers, we increase the number of trainable parameters, and it can become an excessive model that wastes a lot of computation capital for the data given. We must strike a balance between a model that is stable, size-appropriate that fits within the established goal and a model that is large, with too many parameters but marginally better predictor of the outcome variable.

References

Linear and Logistic Regression in Tensorflow, Stanford University

https://web.stanford.edu/class/cs20si/2017/lectures/notes_03.pdf

Tensorflow Regression Example Problem:

https://www.tensorflow.org/tutorials/keras/basic_regression

Plotting pick up and drop off locations:

https://www.kaggle.com/breemen/nyc-taxi-fare-data-exploration

Keras Sequential Model Documentation:

https://keras.io/getting-started/sequential-model-guide/