Big-Query -Geotab Intersection Congestion

Xiaohan Yin
Net ID: xyin13
School of Information Sciences
MS in Information Management
xyin13@illinois.edu

Yu Zhu
Net ID: yuzhu5
School of Information Sciences
MS in Information Management
yuzhu5@illinois.edu

1. Introduction

When stuck at an intersection, especially in rush hour traffic, there are usually a great number of vehicles waiting in line and only given mere seconds to pass through. Therefore, if we are able to establish useful models to predict the waiting time and distance to the stop, it may help a lot to reduce the quantity of commuters at the traffic light, and thus reducing the rate of congestions.

2. Data Preprocessing

2.1 Getting to Know the Data

To begin with, we built data structure table to get insights from the datasets. It turned out the file of training sets has 857409 rows and 28 columns, including int and float that belong to numerical data, and object that belongs to categorical data. Besides, there is no text data or image data included.

2.2 Data Explanation

In order to explain the data in a reasonable way, we plan to do data preprocessing first to deal with missing data and noisy data. What's more, It's also necessary to eliminate the effects of outliers to estimate the results better. Finally, we also plan to use methods like normalization and hierarchy generation to transform the data, which will be beneficial to compare and gain consistent conclusions.

Then, we are going to import Numpy and Pandas packages to visualize the data. For example, the heat map to test the correlation level among variables; the bar chart to show the waiting time and distance respectively in each city; the boxplot to reflect the distribution attributes of waiting time and distance at different time, and the line chart to gain the top 50 intersections with longest waiting time or distance, etc.

2.3 Data Cleaning and Discovery

Firstly, we explored the data structure by analyzing the data types, calculating the null values and unique values for each vector. We obtained the general description of the data shown as follow:

	index	data types	null value numbers	unique value numbers
0	RowId	int64	0	857409
1	IntersectionId	int64	0	2539
2	Latitude	float64	0	4505
3	Longitude	float64	0	4541
4	EntryStreetName	object	8189	1707
5	ExitStreetName	object	5534	1693
6	EntryHeading	object	0	8
7	ExitHeading	object	0	8
8	Hour	int64	0	24
9	Weekend	int64	0	2
10	Month	int64	0	9

Table 1 Data Structure of data (part)

In this process, we find that only the data involved with waiting time and distance are numerical data. And for this kind of data, through visualized their distribution, we found that they are not normal distribution. Thus, we scaled the numerical data into -1 to 1.

Next, we divided the train data into two parts: train dataset and test dataset to establish and train the model.

Also, we explored the relationship between some vectors and waiting time.

We created a heat map for all vectors to explore their correlation.

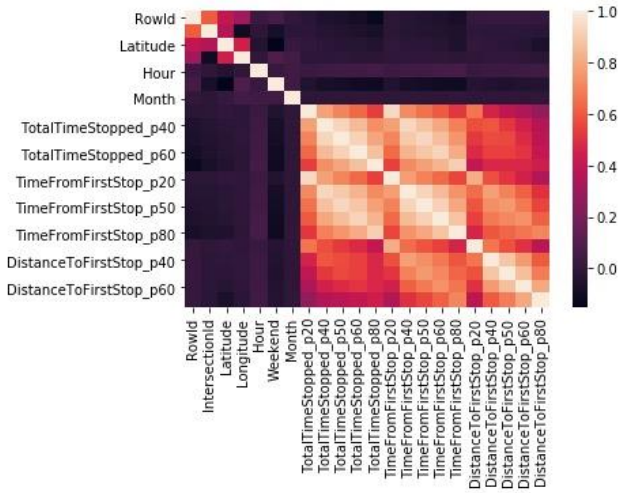

Figure 1 Heat Map for each vectors

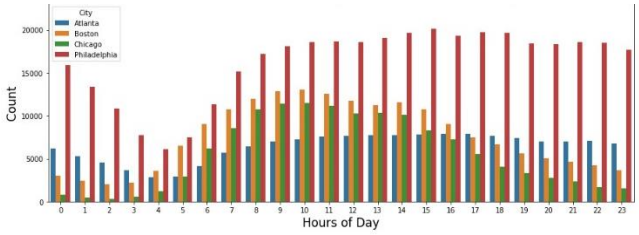

Figure 2 Hour Count Distribution by City

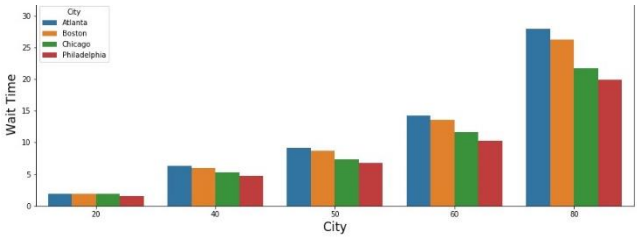

Figure 3 Wait Time Distribution by City

Then, we used EntryHeading and ExitHeading to calculate the turn angle. We found that the turn angle had great impact on the traffic congestion.

We divide the train data into two parts: train dataset and test dataset to consistently train the BP-Neural Network Model.

We created the dummy vectors for the categorical data such as Hour, Month, Weekend, City and Angle.

3. Baselines

3.1 BP-Neural Network

The BP neural network is a gradient descent method to minimize the mean error of the network output. Shown as the following figure, there is an input layer, one or more hidden layers and one output layer.

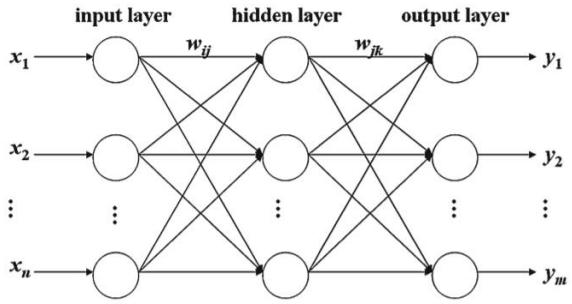

Figure 4 BP-Neural Network Model

We can get the output of j[th] neuron of the hidden layer:

$$b_j = f_1(\sum_{i=1}^{n} w_{ij}x_i - \theta_j)\ (i = 1,2,\dots,n; j = 1,2,\dots,s)$$

Then the output y_k of the output layer is:

$$y_k = f_2(\sum_{j=1}^{s} w_{jk}b_j - \theta_k)\ (j = 1,2,\dots,s; k = 1,2,\dots,m)$$

The error function defined by the network actual output is:

$$e = \sum_{k=1}^{m} (t_k - y_k)^2$$

3.2 LightGBM

Light GBM is a gradient boosting framework that uses tree based learning algorithm. It grows trees horizontally, which will reduce much loss than other level-wise algorithms. Also, it can handle huge amount of data and supports GPU learning with high speed and lower memory.

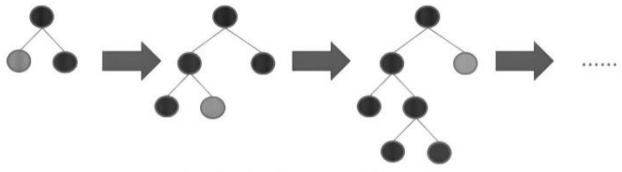

Figure 5 LightBGM

3.3 Random Forest Model

Random forests are designed for regression and classification that by constructing multiple decision trees at training time and outputting classes or mean prediction of the individual trees.

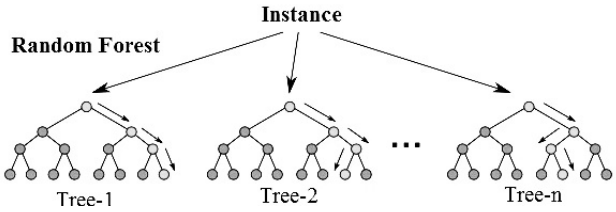

Figure 6 Random Forest Model

The training algorithm applies the technique of bagging. It selects a random sample recursively for B times to fit trees to these samples.

Then after training, averaging the predictions from individual trees to predict for unseen samples:

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x')$$

And the standard deviation of the predictions from individual trees is:

$$\sigma = \sqrt{\sum_{b=1}^{B} \frac{(f_b(x') - \hat{f})^2}{B-1}}$$

4. Model Establishment

4.1 BP-Neural Network

Our running environment is Pycharm 3.7. We primary used Pybrain package in Python 3 to build our neural network model.

We began to established the BP-neural network model as following steps:

1. We initialized the supervised dataset with SupervisedDataSet () method.
2. We initialized the neural network by identifying the input layer, hidden layer and output layer with buildNetwork () method. We defined four hidden layers in our model.
3. Then, we loaded the train dataset into the supervised dataset.
4. We used Back Propagation algorithm, trained the data until convergence with BackpropTrainer () method. Totally, we trained 30 times.
5. We compared the predicted data obtained by the model to the data from test dataset. Then, we used inverse_transform () method to transform the normalized data to its original form.
6. We calculated Root Mean Squared Error (RMSE) to evaluate the accuracy of the model.

4.2 LightGBM

In Pycharm 3.7, we used lightgbm package in Python 3 to establish our LightGBM model.

We set a series of parameters to directly utilize LGB training model. We set the parameter 'objective' to 'regression' to determine the specific problem that the model worked on. We used the parameter 'num_leaves' to determine the number of leaves, and used the parameter 'num_trees' to determine the number of trees. In our model, we set 'rmse' as the evaluation metric.

After setting the parameters, we began to establish the LightGBM model as the following steps:

1. Firstly, we used lgb.train() to train a model with a list of parameters. We initialed the number of round is 100.
2. We used save_model () method to save the training model.Then, we used Booster() method to initial the model we saved. We can load the model to add more features and adjust the parameters.
3. After establishing the model, we used lgb.cv () method to conduct a cross validation with 5-fold.
4. We set the parameter 'valid_sets' to initial evaluation dataset. Then, we used 'early_stopping_rounds' to determine the range of early stopping. In this way, we would have a better performance of model.
5. Then, we used predict () method to obtain the result based on the model.
6. Finally, we chose Root Mean Squared Error (RMSE) as the evaluation metric to evaluate the accuracy of the prediction.

4.3 Random Forest

We primary used RandomForestRegressor package from sklearn.ensemble in PyCharm 3.7 to conduct the model.

With the help of the package, the process is convenient. After data preprocessing, we divided the train dataset into two parts. We used first part as our training set, and the second part as our validation set.

Firstly, we used RandomForestRegressor () method to initial the model. Then, used the training set and validation set to train and fit the model with the method model.fit (). After that, we used model.predict () method to predict the result of test dataset. Finally, RMSE was used as a evaluation metric to determine the accuracy of the model.

5. Result Analysis

1) BP-Neural Network

We divided 70% train dataset as training set, and 30% train dataset as validation set. Since the storage requirement of this model is too large, we selected first 8000 rows of data to train the model. The RMSE is shown as follows:

Target	TotalWaitTime_P20	Total WaitTime_P50	Time From First_p20	Time From First_p50	DisTo First_p20	DisTo First_p50
RMSE	3.01257387	12.2587657 1	5.27244797	19.0281099 2	14.3664673 8	34.0277539 5

Table 2 RMSE of BP-Neural Network

2) Random Forest

Since the restriction of the input consistence, the training set and validation set should have the equal length. We divided 50% train dataset as training set, and 50% train dataset as validation set. The result is shown as follows:

Target	Total WaitTime_P20	Total WaitTime_P50	Time From First_p20	Time From First_p50	DisTo First_p20	DisTo First_p50
RMSE	1.283 61676 34924 125	3.472 88584 71489 56	2.075 93408 48617 3	4.654 52420 76126 58	3.398 71622 59417 563	6.987 56793 77783 335

Table 3 RMSE of Random Forest

3) LightGBM

We divided 70% train dataset as training set, and 30% train dataset as validation set. In this case, we set the number of leaves as 31, and the number of trees as 100.

Target	Total WaitTime_P20	Total WaitTime_P50	Time From First_p20	Time From First_p50	DisTo First_p20	DisTo First_p50
RMSE	6.951 16937 04519 51	14.93 29534 91111 226	11.40 33969 73821 947	22.85 84117 07279 398	27.22 30119 17872 025	70.71 35692 03068 39

Table 4 RMSE of LightGBM

Based on our overall consideration, LightGBM had the best performance. Thus, we used LightGBM model to predict the target values.

The parts of predictions of LightGBM is shown as follows:

RowId	Target
1920335	1.81516279
1920336	0.58286949
1920337	0.62488686
1920338	0.62488686
1920339	0.59789359
1920340	0.44291079
1920341	0.53060117
1920342	0.53060117
1920343	2.16526165
1920344	3.9463203

Table 5 Prediction for TotalTimeStopped_p20 (Part)

6. Further Analysis

6.1 Ablation Study

In our model, early Stopping is used to avoid overfitting. We removed this part from our current model to test the significance of it.

1) Without Early Stopping

Target	Total WaitTime_P20	Total WaitTime_P50	Time From First_p20	Time From First_p50	DisTo First_p20	DisTo First_p50
RMSE	6.95169670260941	14.9329534911112 26	11.4032986757846 2	22.8580275764783 28	27.2230119171723 4	70.7154648049726

Table 6 RMSE Without Early Stopping

From the table, we can know that there was few difference between the prediction with early stopping and the prediction without early stopping. This means there was no overfitting in our model. However, it is still importance to set the early stopping to prevent from the overfitting case.

6.2 Parameter Study

1) num_leaves = 10, num_trees = 30

Target	Total WaitTime_P20	Total WaitTime_P50	Time From First_p20	Time From First_p50	DisTo First_p20	DisTo First_p50
RMSE	6.96325379484154 1	15.0124584519412 17	11.4261771270638 18	22.9802289772823 7	27.2517536598725 7	70.8529396741393 1

Table 7 RMSE with num_leaves = 10, num_trees = 30

2) num_leaves = 100, num_trees = 300

Target	Total WaitTime_P20	Total WaitTime_P50	Time From First_p20	Time From First_p50	DisTo First_p20	DisTo First_p50
RMSE	6.95208522149593 7	14.9351343488448 2	11.4061389660347 51	22.8593228648297 67	27.2271359668850 37	70.7184601521891 8

Table 8 RMSE with num_leaves = 100, num_trees = 300

From the tables, we can know that with the number of leaves and trees become larger, the RMSE becomes smaller. However, the running time also becomes larger. Therefore, we selected 31 and 100 as the number leaves and the number trees in our model.

7. Discussion and Future Scope

During the process of establishing and comparing several models for intersection congestion, we found the following problems and considered possible solutions:

Firstly, it is obvious that values of RMSE for distinct variables are greatly different from each other, so we probably did not extract all the related features, which led to evident differences.

Then, because of the limitation of algorithms, our values of RMSE for those three models are from different dimensions: nueral netwok with part dataset, random forests with 50% training dataset and 50% test dataset, and LightGBM with 70% training dataset and 30% test dataset. Therefore, we may need to continue to modify the model to compare their performance in a unified way.

Finally, in our LightGBM model, we can only predict one variable for one time, which means it would be huge amount of work load when we need to predict multiple variables. So maybe we can improve our algorithm to make it more convenient to predict and evaluate the model.

Reference

[1] Ding, Shifei, Chunyang Su, and Junzhao Yu. "An optimizing BP neural network algorithm based on genetic algorithm." Artificial intelligence review 36.2 (2011): 153-162.

[2] PyBrain Documentation. http://pybrain.org/docs/index.html

[3] LightGBM Documentation. https://lightgbm.readthedocs.io/en/latest/

[4] Ke G, Meng Q, Finley T, et al. Lightgbm: A highly efficient gradient boosting decision tree[C]//Advances in Neural Information Processing Systems. 2017: 3146-3154.

[5] Louppe G. Understanding random forests: From theory to practice[J]. arXiv preprint arXiv:1407.7502, 2014.

[6] Github. https://github.com/apachecn/lightgbm-doc-zh

Contribution

Xiaohan Yin (50%): Prepressed the data, Established the model, Wrote the paper

Yu Zhu (50%): Prepressed the data, Established the model, Wrote the paper