BUS ARRIVAL TIME PREDICTION WITH REGRESSION MODEL

May 17, 2017

Student Name: Jun Wang
Student ID: 1147872
Research Advisor: Roni Khardon
Tufts University
Department of Computer Sciences

Contents

ABSTRACT

This project is mainly focused on how to predict the bus arrival time from the available online resource and compare the performance between different regression models. The main part of this project can be divided into two stages: 1. preprocessing and sampling the historical data and the GTFS data from the online database[1][2]. 2. train and predict with different regression models with the preprocessed dataset and compare their performance from different aspects. In the first stage, several critical tables were generated and stored with the information like the stop sequence, travel duration. Most of the abnormal value from the historical data has been removed and the quality of the dataset is highly reliable. In the second stage, baseline algorithm and several different regression models like linear regression, support vector machine regression, neural network regression and the Gaussian Process regression were utilized. Through comparing their performance in different aspects, it is concluded that the regression model has much better performance comparing to the baseline algorithm, and the Gaussian Process Regression in the single-shape model has the best performance comparing to other models.

INTRODUCTION

The increasing population in cities requires more and more intelligent scheduling in public transportation system in modern society. Thus, as part of the intelligent transportation system, prediction on bus arrival time with highly reliable accuracy also plays a key role[7]. Considering the large scale of data for the public transportation system, it attracts more and more researchers to utilize the Machine Learning Technique to solve this problem. For now, the techniques for predicting bus arrival time can be classified into several different types[8]: 1. Model based on historical data like average speed, daily pattern, etc.; 2. Statistical model like regression models; 3. Kalman Filtering Model; 4. Machine Learning techniques like neural network.

In this project, two different types of the model were utilized: baseline model and the regression model. The baseline model is similar to the model based on historical data in previous paragraph and the regression model includes the linear regression, Support Vector Machine(SVM) regression, Neural Network (NN) regression and Gaussian Process (GP) regression.

In predicting bus arrival time, many stochastic factors might affect the result like the traffic condition, weather, passenger numbers, etc[7]. However, in many situations, the only available data might only be the historical data of the vehicle motions in the public transportation system. For example, it is not always easily to obtain the number of passengers in each vehicle at any given time. Fortunately, with the enough historical data, the pattern of the vehicles' motion in each route could still be found.

The basic aim of this project is to provide some reasonable way to preprocessing the historical data and obtaining some reasonable features. Then, through the feature selection, model selection and the comparison between models, some reasonable features would be obtained.

DATA COLLECTION

Basic Concepts

Since in this project, we investigate the bus system specifically, which might includes specific terminologies with highly frequent usage, it is necessary to list all of them here in case any reader is confused with those names during the reading.

- Trip ID: Each Trip ID represents a specific trip at a given time. Usually the stop sequence and the time schedule for a specific Trip ID should be exactly the same every day.
- Route ID: Each Route ID represents a specific route in the Bus System.
- Shape ID: Each Shape ID represents a specific stop sequence. In other words, all the trips should have a specific Shape ID. If their Shape IDs are the same, their stop sequences should be exactly the same as well. However, though each route with a specific direction should have only one stop sequence(Shape ID), it is possible that a route has multiple different Shape ID and these stop sequences only share the same initial station and the final station.
- Vehicle ID: Each Vehicle ID represents a specific ID of a bus. In most of the situation, the driver won't change the bus in a specific trip, so the Vehicle ID of a specific trip on the given date should be exactly the same.
- Direction ID: The Direction ID can only be 0 or 1 and they represent two opposite directions in a specific route.
- Stop ID: Each Stop ID represents a specific stop in the bus system.
- dist_next_stop: The distance along the route between the current location of the bus at a given time to the next stop in the stop sequence
- dist_along_route: The distance along the route between the initial station and the given location or stop
- segment: An interval between two neighboring stops according to the stop sequence
- segment_start: The start station in the segment.
- segment_end: The end station in the segment.

- travel_duration: The time for a bus traveling from segment_start to segment_end
- single_shape model: The dataset is split into different subsets according to the Shape ID then train and test the models with each subset respectively.
- multiple_shape model: The whole dataset is used to train and test the models directly without considering the Shape ID.

Obtain Dataset

In order to train the model for predicting the bus arrival time at any given time, it is necessary to use the historical data as the dataset for training and testing. However, for a specific historical data[2], each record only indicates the location of a specific bus at a given time. Thus, it is necessary to have the extra information including the route id, direction id, stop sequence, etc. All these extra information are can be found by checking the General Transit Feed Specification (GTFS) data[1]. Besides that, since it is reasonable that the weather might affect the operation of the bus system, it is necessary to obtain the dataset for the weather[3] with the same date of the historical data.

Thus, in order to train and predict the bus arrival time, there are three different necessary files:

- Historical data
- GTFS data
- Weather Information

Historical data

Historical data can be downloaded from NYC Transit Data Archive[2]. It is a list of historical records and each record includes the following critical information:

- timestamp: It is the time for that corresponding record, which includes the date, hour, minute and the second
- vehicle_id: It is the Vehicle ID for that record
- longitude: The longitude of the location for the vehicle at the given time in the record

- latitude: The latitude of the location for the vehicle at the given time in the record
- service_date: The schedule of the date in the record. In most of the situation, service_date is the same to the date in timestamp. However, in the midnight, the service_date might be the same to the date of previous day.
- trip_id: The Trip ID for that record
- block_assigned: Sometimes the database cannot provide the location of the specific bus at the given time, instead, it will provide the location of another vehicle with the same Trip ID at the given time. If the record is replaced by another vehicle, the value of block_assigned is 0, otherwise the value of block_assigned is 1.
- next_stop_id: The Stop ID for the vehicle which is approaching at the given time in the record.
- dist_along_route: The dist_along_route of the next stop
- dist_from_stop: The distance between the current location of the vehicle and the next stop

GTFS data

GTFS data describes the public transportation schedule and the corresponding geographic information together. For example, it provides the latitude and the longitude for each stop in the dataset and it also records the relationship between all the trips and the routes. Among the complete GTFS dataset, there are several important files which are necessary for this project:

- trips.txt: Each record represents a specific trip in the dataset. In that record, it provides the Route ID, Shape ID, Direction ID for that corresponding trip.
- stop_times.txt: Each record represents a stop in a specific trip. In that record, it provides the Trip ID, the Stop ID, the sequence number of that stop in the trip, the scheduled arrival time and the departure time, etc.

All the necessary GTFS data can be found on the transitfeeds[4].

Weather Information

The weather can be downloaded throught the API interface of Wunderground[3]. Through this API, we can get the information of the specific date for snow, rain, fog, etc.

Preprocess Dataset

To obtain the necessary data for analysis, it is necessary to clean and extract the data from the original ones. The data quality for the GTFS and the weather data is good, but considering the precision and the bugs in the historical data, we need to pay more attention when extracting the data from the historical data. All the necessary data files are listed here:

- weather: This file includes the weather(rainy, snowy, or sunny) for a specific date. But the most important column is named "weather", which is either 0, 1, or 2. The corresponding meaning for 0, 1, and 2 is sunny, rainy, and snowy.
- route_stop_dist: This file includes the distance between each stop and the initial station in each stop sequence(share same Shape ID).
- history: This file is the filtered historical data by removing some abnormal value and add some more extra information.
- segment: This file includes the travel duration for all the segments in different dates in the history data.
- api_data: This file provides the necessary information for sampling from the dataset for training and testing.
- dataset: This file provides all the samples with all necessary features for different regression models

The detail for preprocessing above files is following.

weather

Weather data is downloaded from the website wunderground[3] with the given api token. By providing the date list as the input, the corresponding weather data will be downloaded. The weather.csv file can indicate the existence of the rain and snow on each date.

route_stop_dist

Though in each record of the historical data, it provides the dist_along_route for the corresponding next stop, it is very inefficient when checking the whole dataset for a specific stop in a stop sequence. Thus, it is necessary to extract the distance for all stops in all different stop sequence. The process for generating route_stop_dist is below:

1. Since each Shape ID represents a specific stop sequence, generate the list of Shape ID from the trips.txt file in GTFS.
2. Generate the stop sequence and the corresponding dist_along_route for each stops for all Shape ID.
   (a) Get any Trip ID for a specific Shape ID from trips.txt file.
   (b) Get the corresponding stop sequence for that Trip ID from the stop_times.txt file.
   (c) For each stop in that stop sequence, check the corresponding dist_along_route if the Stop ID can be found in the column "next_stop_id" of the historical data with the same Trip ID. That dist_along_route should be exactly the same for all records with the same next_stop_id and trip_id in the historical data and that value is dist_along_route for that stop.

Thus in the table of route_stop_dist, it includes at least the Shape ID, Stop ID, and the dist_along_route in each record.

history

Though all the historical data can be downloaded from the online dataset[2], the quality of the dataset is not reliable and the efficiency would be slow. Thus, it is necessary to generate the highly reliable historical data without any unnecessary information.

Thus the whole process includes the following steps:

1. Generate the historical data from the dataset with only given date list which are sorted.
2. Generate the list of Shape ID from route_stop_dist table.
3. Generate the list of Trip ID according to the Shape ID and the trips.txt file in GTFS.
4. Filter the historical data to remove all the records whose trip id is not in the list of Trip ID. It is necessary to filter the historical data, because the historical data actually

involves several districts in New York City, while this project only focused on one district of the New York City.

5. Remove all the records whose value of block_assigned is not 1, because it means that the corresponding record is actually replaced by another vehicle with the same Trip ID and this replacement would cause error in data analysis.

6. Remove all the records whose dist_along_route is not a number. In some records, the value of the dist_along_route is '
N', which means missing value, and these missing values should be remove.

7. Remove all the records whose dist_along_route is equal to 0, because the GPS device sometimes records the location of the vehicle even when it hasn't start yet.

segment

This table records all the travel duration for each segment in each trips every day. Since in the historical data, the location of a vehicle is recorded around every 1.5 minutes, it is necessary to calculate the estimated arrival time for some stops according to the time duration and the distance between the records. Then the corresponding estimated travel duration can be calculated as well. However, because of the precision of the historical data, some of the stops might be skipped in the historical data, which requires some other ways to estimated the corresponding travel duration for these segments.

The whole process is below:

1. Calculate the travel duration between the segments according to the historical data.
   (a) Split the history table according to the service_date and the Trip ID.
   (b) Add the travel duration of each segments for each subset of the history table and concatenate these data to form the raw dataset for segment table.
      i. Generate the estimated arrival time for each subset of the history table
         A. Start from the first record of the table to find a record whose next_stop_id is different with that of the next record.
         B. Calculate the time duration between these two neighboring record.
         C. Calculate the distance ratio: $ratio = \frac{dist\_bus\_stop}{dist\_bus\_bus}$. $dist\_bus\_stop$ represents the distance between the vehicle and the next stop and $dist\_bus\_bus$ represents the distance between the location of the vehicle in these two neighboring records.

D. Use the ratio and the time duration to estimate the arrival time for the vehicle arriving to the next stop.

E. Traverse through the whole table and get all the estimated arrival time for the corresponding stops.

ii. According to the estimated arrival time list, calculate the travel duration between each neighboring element in the list, which is the raw data for segment table.

2. Check each segment and compare the stop sequence to find all the skipped stops in these segments. Estimate the travel duration for these skipped segments.

   (a) Split the raw data of th segment table according to the service_date and the Trip ID.

   (b) For each subset of the raw data, add all the skipped segments.

      i. Find the corresponding stop sequence from stop_times.txt file for each subset according to the Trip ID.

      ii. Compare the stop sequence with the subset.

      iii. If there are skipped stops within the segment of raw data, calculate the average travel duration by simply dividing the total travel duration of the segment with the number of child segments within it.

api_data

The table api_data is used for sampling, so each record of api_data only provides some information which could be found from the api interface for the public transportation system. For example, in NYC transportation system, api interface can provide the following information in each request:

- Vehicle ID
- Trip ID
- Route ID
- Stop ID
- current time and date
- dist_along_route of the vehicle at the given time

It is reasonable that each record of the api_data table also provides the same information. The process to generate an api_table is below:

1. Provide a list of timestamp.
2. Split the history table according to the Shape ID (represent the stop sequence).
3. For each subset of the history table, generate the corresponding subset of the api_data
   (a) According to the Shape ID, obtain the corresponding stop sequence from the route_stop_dist table and generate 3 different stops by random. These 3 stops will be used as the target stop.
   (b) Split the subset of the history table by service_date and the trip_id.
   (c) For each part of the corresponding subset, calculate the dist_along_route of the vehicle at the given time in the time list. If the dist_along_route indicates that the vehicle has passed the target stop or the vehicle hasn't departed from the initial station yet, skip this part.

In this project, time list is [12:00, 12:05, 12:10, 12:15, 12:20, 12:25, 12:30, 18:00, 18:05, 18:10, 18:15, 18:20, 18:25, 18:30].

Among this time list, the timestamp around 12:00PM represents the non-rush hour while the timestamp around 18:00PM represent the rush hour.

MODEL

In this project, several different algorithms are implemented: Baseline Algorithm, Linear Regression, Support Vector Machine Regression(SVR), Neural Network Regression (NNR), and the Gaussian Process Regression (GPR).

Baseline Algorithm

The key idea of baseline algorithm is to use the average travel duration to predict the arrival time according to the information provided in api_data table. The baseline algorithm is a simple model based on the historical data which use the travel duration in different segments as the pattern to describe the difference between the segments and the trips.

The whole process is below:

1. Generate a record from the api_data table
2. According to the Shape ID in the record filter the segment tabel and generate the table with average travel duration of each segment from filtered result.
3. According to the dist_along_route in the record of api_data and the route_stop_dist table, find which segment the vehicle is within.
4. Find the segment list that the vehicle will pass through to arrive to the target stop.
5. Use the average travel duration to calculate the predicted arrival time for that record in api_data table.

Regression Model

The difference between the regression model and the model based on historical data is that the regression model is built with mathematical function and shows better performance even under the unstable traffic conditions[8].

For linear regression model, the ordinary least squares linear regression model is applied and the package in sklearn was used. The ordinary least squares model trains a linear model with the coefficient $w = (w_1, ..., w_p)$ such that the residual sum of the squares between the predicted response and the observed response is minimized[5].

$$\arg\min_{w} |Xw - y|^2$$

Support Vector Machine(SVM) Regression is similar to that of the Support Vector Machine (SVM) Classification[11]. The principle of SVM Regression is still to obtain the coefficient W such that maximizes the margin while minimizes the error[12]. The SVM Regression was implemented with the package in sklearn.

The Neural Network(NN) regression used here is a multi-layer perceptron(MLP) which trains the model by using backpropagation. The square error is used as the loss function and predict with the continuous output[13]. The NN Regression was implemented with the package in sklearn.

Guassian Process(GP) is a generic supervised learning technique for solving the classification and regression problems[14]. The GP Regression was implemented with the package in GPy[15]. In this project, considering the similarity between different trips, Multitask-Learning (MTL) Gaussian Process was also implemented with GPy[15]. In MTL Gaussian Process, each task is corresponding to a specific stop sequence(Shape ID).

IMPLEMENTATION

The process of implementation can be divided into three different parts: 1. determined candidate features and outputs; 2. do feature selection and model selecion; 3. train the model with complete dataset and the selected models and features.

Features and Outputs

Features

According to previous research[6][7][8], there are many different factors related to the bus arrival time. For example, the factors like traffic congestion, weather condition, boarding passenger might all affect the predicting result. Here, considering the available dataset and the efficiency of algorithm, several features are selected as below:

- weather: The value for weather feature can only be 0, 1, or 2 to represent sunny, rainy or snowy.
- rush_hour: The value for rush_hour feature can only be 0 or 1 to represent non-rush hour or rush hour(The rush hour is claimed as any time between 17:00 - 20:00).
- baseline_predicted_time: The value of baseline_predicted_time is the predicted arrival time from the current location of the vehicle to the target stop with the baseline algorithm. The reason to use this baseline algorithm is that the only available predicted algorithm with only historical dataset.
- ratio_current_trip: The value of ratio_current_trip is the ratio between the actual arrival time and the predicted arrival time through the baseline from the initial station to the current location in the api_data table within the exactly the same trip.
- ratio_prev_trip: The value for the ratio_prev_trip is the ratio between the actual arrival time and the predicted arrival time through the baseline from the current location to the target stop within the latest previous trip which has the same Shape ID
- prev_arrival_time: The value for the prev_arrival_time is the predicted arrival time from the current location to the target stop by summing up the travel duration for the corresponding segments which are the latest according to the segment table.

It is also noticeable that since the value of the features vary a lot, it is necessary to apply the normalization for the inputs before training and predicting with the dataset.

Outputs

There are two possible outputs for consideration: 1. the predicted arrival time; 2. the ratio between the actual arrival time and the predicted arrival time by baseline algorithm.

The first output is straightforward, but the performance is not ideal because the numeric scale between the input features and the output is too significant. The second output needs more calculation to obtain the predicted arrival time with different algorithm, but the performance is much better because the numeric scale is more closed to the input features. Thus, the ratio between the actual arrival time and the predicted arrival time by baseline algorithm is used as the output values.

Feature Selection

An incomplete backward elimination was utilized for feature selection. Start from the all the features, then remove each feature and run the model with the dataset, finally then remove any two features and run the model with the dataset. According to the performance of the prediction result with different features combination, final features were be selected(See Table 1).

Model Selection

In this project, model selection represents selecting the ideal kernel function for GP, activation function and solver function for NN.

Here, different activation functions and the solver functions within the neural network regression (NNR) model and different kernel functions within the Gaussian Process Regression(GPR) model are tested. According to the performance of the prediction result, corresponding models were determined.

Learning Curve

The learning curve is an important performance of models in Machine Learning, because it can describe how the performance increase with the increase of the sample size. According to the learning curve, some model with poor performance in small number of samples might show promising performance when the dataset size is relatively larger.

In this project, there are 10666 samples in the dataset with 15 different Shape ID and these samples were divided into 10 different subsets after shuffling the order of the samples. Each subset includes all the samples in the previous subset such that with the numbers of samples in the subsets are increasing and the last subset was exactly the same as the complete dataset. Train the models with these 10 different datasets and the learning curve could be obtained.

Overall Comparison

Overall comparison involves two parts: 1. comparison of performance after 5-fold cross validation for the dataset; 2. comparison of the performance after splitting the dataset according to the actual arrival time.

RESULT AND DISCUSSION

Since mean squared error(MSE)[10] is utilized to assess the performance in previous research[9], it is reasonable to utilize them here as well.

$$MSE = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2$$

However, considering the distance between the location of the vehicle and the target stop, the MSE value might increase with the increase of the distance. Thus, use the ratio as bellow to calculate the MSE value could avoid that.

$$ratio = \frac{actual\_arrival\_time}{predicted\_arrival\_time}$$

In the implementation for learning curve and the overall comparison, every regression model except the MTL Gaussian Process are divided into two different models according to whether they train and predict the model with single-shape model or multiple-shape model.

Feature Selection

The performance of all different models with different feature combinations are listed in the table(See Table 1).

From the table, it is obvious that:

- The performance of the baseline is not affected. The reason is that the baseline algorithm only based on the average travel duration of the segments.
- The average performance of the Gaussian Process is the best.
- The average performance with complete features is good enough(The smallest MSE value is 0.093 and the MSE value for the complete features is 0.094), thus there is no need to remove any features in this project.

Model Selection

In this project, the model selection mainly focused on the Neural Network and the Gaussian Process.

Table 1: MSE of ratio with different features and models

removed features	baseline	linear_regression	SVM	NN	GP	Average
None	0.229	0.057	0.072	0.056	0.056	0.094
rush_hour	0.229	0.059	0.070	0.058	0.058	0.095
current_trip	0.229	0.057	0.070	0.056	0.056	0.094
pre_arrival_time	0.229	0.165	0.164	0.127	0.119	0.161
baseline_result	0.229	0.135	0.165	0.142	0.131	0.160
weather	0.229	0.057	0.070	0.054	0.057	0.093
prev_trip	0.229	0.057	0.070	0.056	0.056	0.094
weather, rush_hour	0.229	0.059	0.069	0.058	0.058	0.095
weather, baseline_result	0.229	0.136	0.165	0.142	0.133	0.161
weather, current_trip	0.229	0.057	0.069	0.061	0.056	0.094
weather, prev_trip	0.229	0.057	0.069	0.055	0.056	0.093
weather, prev_arrival_time	0.229	0.125	0.164	0.125	0.118	0.152
rush_hour, baseline_result	0.229	0.142	0.165	0.142	0.140	0.164
rush_hour, current_trip	0.229	0.059	0.069	0.061	0.058	0.095
rush_hour, prev_trip	0.229	0.059	0.069	0.060	0.058	0.095
rush_hour, prev_arrival_time	0.229	0.135	0.164	0.142	0.132	0.160
baseline_result, current_trip	0.229	0.135	0.165	0.114	0.130	0.155
baseline_result, prev_trip	0.229	0.135	0.165	0.142	0.132	0.161
baseline_result, prev_arrival_time	0.229	0.112	0.136	0.110	0.111	0.140
current_trip, prev_trip	0.229	0.062	0.069	0.061	0.061	0.096
current_trip, prev_arrival_time	0.229	0.128	0.164	0.142	0.120	0.157
prev_trip, prev_arrival_time	0.229	0.128	0.164	0.142	0.118	0.156
Average	0.229	0.096	0.116	0.096	0.092	

Table 2: Model Selection: Activation function for Neural Network

activation function	relu	tanh	logistic	identity	baseline
MSE	0.0627	0.0804	0.0811	0.0801	0.2294

Table 3: Model Selection: Solver function for Neural Network

Solver Function	adam	sgd	lbfgs	baseline
MSE	0.0601	0.0947	0.0546	0.229

Neural Nework

The performance of the neural network with different activation function is listed in table below(See Table 2):

In the Table 2, relu, tanh, logistic and identity represents the following functions:

- relu: $f(x) = max(0, x)$
- tanh: $f(x) = \tanh x$
- logistic: $f(x) = \frac{1}{1+\exp -x}$
- identity: $f(x) = x$

The performance of the neural network with different solver function is listed in table below(See Table 3)

In the Table 3, adam, sgd, and lbfgs represents the following solver function:

- adam: a sgd-based optimizer proposed by Kingma, Diederik, and Jimmy Ba
- sgd: stochastic gradient descent
- lbfgs: an optimizer of quasi-Newton method

From the comparison of between different activation function and the solver functions(See Table 2 and Table 3), it is reasonable to select the relu and lbfgs as the activation function and the solver function respectively.

Gaussian Process

The model selection for Gaussian Process is focused on selecting kernel functions(See Table 4). In that table, RBF, Matern52, Matern32, Linear are all different kernel functions. ARD and NoARD represents whether the flag of the ARD for the kernel function is true or false. In the

Table 4: Model Selection: Kernel Functions for Gaussian Process

Kernel function	MSE of ratio
GP_Matern52_NoARD	0.055566466
GP_RBF_NoARD	0.055813246
GP_Matern32_NoARD	0.056079204
GP_Linear_NoARD	0.127708616
GP_Linear_ARD	0.126393555
GP_RBF_ARD	0.056764756
GP_Matern52_ARD	0.056438826
GP_Matern32_ARD	0.055222999
baseline	0.229350586

kernel function, when ARD is true, it means that length scale parameter in each direction is different. According to the table, the performance of the kernel function Matern32_ARD has the best performance, so this kernel function Matern with ARD will be used in the future application.

Learning Curve

The learning curve of the MSE with the increase of the dataset size is indicated in the figure(See Figure 1) and the detailed data can be found in the Table 5

From the learning curve and the table, it is indicated that:

- With the increase of the dataset size, the MSE of ratio is decreasing in almost all the models and will remain stable generally for all.
- The performance of the baseline algorithm is much worse comparing to all the other models
- Comparing the data in the table, it is obvious that the performance of all the model in single_shape model is better than the multiple_shape model and the MTL Gaussan Process.
- The performance of the linear regression may affected by the dataset significantly according to the Table 5. For example, when the sample_size is 5814, the MSE value for linear_regression in single_shape model is 5806, which is highly abnormal while the other models with the exactly same dataset is very normal. Thus the ordinary linear regression might not work well for unstable dataset.

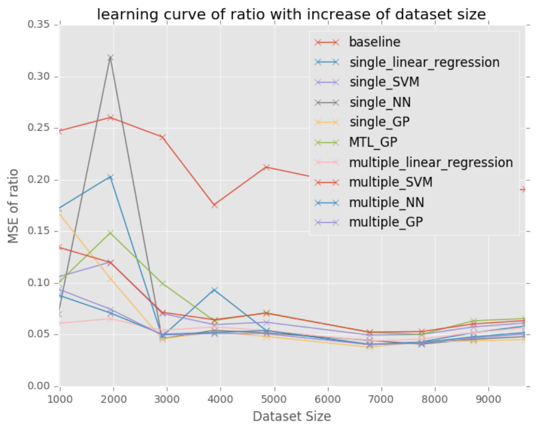

Figure 1: The learning curve of ratio with increase of dataset size.

Table 5: The MSE of ratio with different dataset size and model

sample_size	baseline	single _linear _regression	single _SVM	single _NN	single _GP	MTL _GP	multiple _linear _regression	multiple _SVM	multiple _NN	multiple _GP
969	0.247	0.172	0.1059	0.0701	0.1681	0.1001	0.0609	0.1346	0.0883	0.0941
1938	0.2602	0.2029	0.1202	0.3188	0.1045	0.1484	0.0655	0.1199	0.0711	0.0747
2907	0.2414	0.0482	0.0706	0.0462	0.0459	0.0998	0.0542	0.0717	0.0504	0.0497
3876	0.1757	0.0934	0.0597	0.0541	0.0526	0.0635	0.0574	0.0645	0.052	0.051
4845	0.2122	0.0543	0.062	0.0517	0.0483	0.0712	0.0542	0.0708	0.0539	0.0509
5814	0.1921	5806.131	0.0584	0.0489	0.0473	0.057	0.052	0.0612	0.0484	0.0474
6783	0.1912	0.0402	0.0495	0.0443	0.0378	0.0522	0.0437	0.0524	0.0407	0.0403
7752	0.2023	0.043	0.0502	0.0409	0.043	0.0503	0.0458	0.053	0.0424	0.0418
8721	0.1871	0.0519	0.0575	0.0455	0.044	0.0634	0.052	0.0604	0.0479	0.0467
9690	0.1907	0.0584	0.0614	0.0481	0.0452	0.0656	0.0568	0.0636	0.052	0.0506
10659	0.2098	8.7233	0.0549	0.0435	0.0418	0.0658	0.0505	0.0578	0.0478	0.0451

Table 6: The overall performance after 5 fold cross validation

fold_number	baseline	single _linear _regression	single _SVM	single _NN	single _GP	MTL _GP	multiple _linear _regression	multiple _SVM	multiple _NN	multiple _GP
0	0.2167	0.3781	0.0546	0.0471	0.0398	0.0684	0.0513	0.0606	0.0491	0.0467
1	0.2487	0.0452	0.0632	0.0487	0.0450	0.0776	0.0510	0.0668	0.0513	0.0500
2	0.2077	0.0558	0.0663	0.0496	0.0519	0.0729	0.0588	0.0698	0.0542	0.0537
3	0.2326	0.0450	0.0788	0.0507	0.0460	0.0754	0.0580	0.0735	0.0568	0.0572
4	0.1663	0.0893	0.0520	0.1726	0.0414	0.0615	0.0497	0.0524	0.0465	0.0441
Average	0.2144	0.1227	0.0630	0.0737	0.0448	0.0712	0.0537	0.0647	0.0516	0.0503

Overall Comparison

Finally, a 5-fold cross-validation within all different models was implemented(See Table 6). According to this table, several conclusions can be drawn:

- The performance of non-MTL Gaussian Process is better than other models.
- In multiple-shape model, the performance of linear model and the NN is better than that of the SVM and GP. However, in the single-shape model, the performance of SVM and GP is better than that of the linear model and NN.
- Comparing to other regression models, the linear model is more unstable and may vary significantly according to the dataset.
- The performance of baseline model is much worse than any other regression models.

Besides the cross validation, in order to investigate the effects of the distance between the current location and the target stop, the whole dataset is also divided into multiple groups according to the value of the actual arrival time: [0 1000, 1000 2000, 2000 3000, 3000 4000, 4000 5000]. The result is indicated here(See Figure 2 and Table 7).

From the figure and the table, it is concluded that:

- The performance of the baseline is much worse comparing to regression models.
- When the actual arrival time is small, the MSE value of different regression model is closed. However, with the increase of the actual arrival time, the MSE value varies significantly in different models.
- With the increase of the actual arrival time, the performance of the most regression models becomes better and better.
- With the increase of the actual arrival time, the performance of the SVM is become worse and worse.

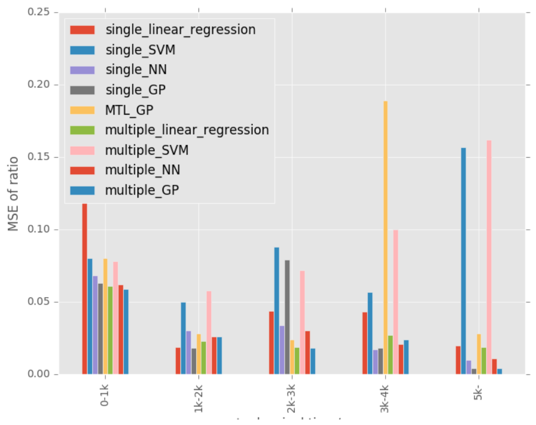

Figure 2: The MSE value with different actual arrival time(the MSE value of baseline not shown in this figure).

Table 7: The MSE value with different actual arrival time

actual _arrival _time/sec	baseline	single _linear _regression	single _SVM	single _NN	single _GP	MTL _GP	multiple _linear _regression	multiple _SVM	multiple _NN	multiple _GP
0-1k	0.106	0.2	0.08	0.068	0.063	0.08	0.061	0.078	0.062	0.059
1k-2k	0.173	0.019	0.05	0.03	0.018	0.028	0.023	0.058	0.026	0.026
2k-3k	0.604	0.044	0.088	0.034	0.079	0.024	0.019	0.072	0.03	0.018
3k-4k	1.119	0.043	0.057	0.017	0.018	0.189	0.027	0.1	0.021	0.024
5k-	2.216	0.02	0.157	0.01	0.004	0.028	0.019	0.162	0.011	0.004
Average	0.8436	0.0652	0.0864	0.0318	0.0364	0.0698	0.0298	0.094	0.03	0.0262

- With the increase of the actual arrival time, the performance of the Gaussian Process (including the MTL GP and non-MTL GP) is unstable but the overall trend is becoming better.

CONCLUSION

In this project, a basic training and prediction process with the historical dataset and the scheduled dataset(GTFS) was set up and the performance of different learning algorithms also proved the reliability of this process.

Through this project, it is concluded that:

1. Including the baseline algorithm, the performance of these models indicated that this process is effective and promising to investigate more in the future.

2. Through the feature selection, it is concluded that all the features should be included in the dataset when training and testing.

3. Through the learning curve, it indicates that the performance of the regression model is much better than that of the baseline algorithm.

4. The overall performance of the single-shape model is better than that of the multiple-shape model, which indicates that the Shape ID or Trip ID is actually a significant factor when predicting the bus arrival time.

5. The performance of the non-MTL Gaussian Process is better than that of the MTL Gaussian Process and other regression models.

6. According to the distance between the current location and the target stop, the performance of different models vary a lot. Though most of the model shows better performance when predicting with the long distance, some models like the Gaussian Process may vary a lot.

7. The linear regression model is highly sensitive to the dataset and the might not work well under situation with unstable factors.

Bibliography

[1] Google.
`https://developers.google.com/transit/gtfs/`

[2] NYC Transit Data Archive.
`http://data.mytransit.nyc.s3.amazonaws.com/README.HTML`

[3] Weather Underground.
`https://www.wunderground.com/weather/api/d/docs?d=data/history&MR=1`

[4] transitfeeds.
`http://transitfeeds.com/`

[5] Scikit-Learn.
`http://scikit-learn.org/stable/modules/linear_model.html#ordinary-least-squares`

[6] Bin, Y. *Bus Arrival Time Prediction Using Support Vector Machines.* Journal of Intelligent Transportation Systems, 10(4):151-158, 2006.

[7] Chien, S. *Dynamic bus arrival time prediction with artificial neural networks.* Journal of Transportation Engineering, 128(5):429-438, 2002.

[8] Altinkaya, M. *Urban Bus Arrival Time Prediction: A Review of Computational Models.* International Journal of Recent Technology and Engineering, 2(4):2277-3878, 2013.

[9] Pan, Jian. *A Self-learning algorithm for predicting bus arrival time based on historical data model.* Cloud Computing and Intelligent Systems (CCIS), 2012 IEEE 2nd International Conference on. Vol. 3. IEEE, 2012.

[10] Scikit-Learn.
`http://scikit-learn.org/stable/modules/model_evaluation.html#mean-squared-error`

[11] Scikit-Learn.
http://scikit-learn.org/stable/modules/svm.html#svr

[12] Saed Sayad.
http://www.saedsayad.com/support_vector_machine_reg.htm

[13] Scikit-Learn
http://scikit-learn.org/stable/modules/neural_networks_supervised.html#regression

[14] Scikit-Learn.
http://scikit-learn.org/stable/modules/gaussian_process.html#gaussian-process-regr

[15] GPy. *GPy: A Gaussian process framework in python.*
http://github.com/SheffieldML/GPy, 2012.