Predicting Bus Arrivals Using One Bus Away Real-Time Data

Catherine M. Baker
Department of Computer Science
University of Washington
Seattle, WA 98195
cmbaker@cs.washington.edu

Alexander C. Nied
Department of Computer Science
University of Washington
Seattle, WA 98195
anied@cs.washington.edu

Abstract

Providing real-time bus arrival times can enhance the experience of riders using public transit, but errors in the predictions can cause a decrease in ridership. Using data collected from a local transit agency, we created models to predict arrival times using K-Nearest Neighbors and Kernel Regression and seven sets of features. The best feature set had a RMSE of 34 seconds, a large improvement over the baseline of always predicting on-time arrivals, which had an RMSE of 99 seconds.

1 Introduction

The Puget Sound region of One Bus Away covers King County Metro, Intercity Transit, Community Transit, and a handful of other transportation agencies in the area. Last year, King County Metro alone provided 115.4 million passenger trips [1]. One Bus Away seeks to improve the experience of riders in the area by providing real-time arrival times of buses [2]. This has the potential to improve the rider experience, but errors can temper this improvement. There was a significant effect on ridership by those who had experienced an errors in the predictions (9% took the bus less often) [3]. Currently, the application is generally able to provide good accuracy of the arrival of the bus, but there are still cases where further work could be done to improve the predictions of the arrival times.

2 Related Work

Estimating bus arrival times is an established problem in the transportation industry. Kalman Filters [4], Support Vector Machines [5], and K-Nearest Neighbors [6, 7] have been used to create models to estimate bus arrival times. Mathieu Sinn's work used *K*-Nearest Neighbors and Kernel Regression to predict bus arrival time using GPS measurements [7]. Sinn's group modified their original data, GPS coordinates at timestamps, into bins of trajectories by interpolating the data in 100 meter intervals.

3 Data

Our data is based on the real-time feed that is provided by the metro agencies. Each bus sends a burst of information every one to three minutes. These bursts contain a wide variety of information about the bus including: latitude, longitude, trip id, vehicle id, schedule deviation, current date and time, distance along the trip, and distance along the route amongst other information. This information can be used to isolate features for our models.

For our model validation, we selected a single route from the Intercity Transit provider in Thurston County which provides service to the state's capitol in Olympia. Route 13 is the most common route for this provider, running 54 trips a day in the outbound direction. Each trip of this route had an average of approximately 22 unique data bursts. We also used Route 12 in Olympia to evaluate the final models. As the timing of trips and number of trips may change for each service period, we used the service period that ran from June 9, 2013, to September 29, 2013 and only used the weekday trips.

3.1 Data Preprocessing

In an analysis of our data, we observed inconsistencies within the data. Although the bus generally travels along the route over time, sometimes it sends updates that suggest it travelled in reverse, or that a bus had travelled 6 km when it's coordinates are less than 1 km from its origin. This appeared to be an error with the distance along the trip metric provided in the information bursts. When we plotted the latitude and longitude against the distance along trip (figure 1), we could see that the distance would jump around within the trip. Therefore, we chose to clean this data to get an updated distance along the trip. In the offline data provided by the service providers, there is information about the route which gives us the true distance along the trips for set latitudes and longitudes. We interpolated the latitude and longitudes for every ten meters of the trip. From there, we used one nearest neighbor for each latitude and longitude from the bursts to get the new distance along the trip.

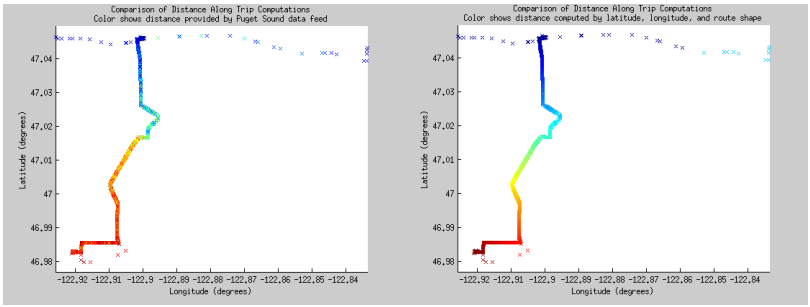

Figure 1: The two graphs show latitude and longitude plotted with distance along the trip being represented by the color of the point. The dark blue represents the beginning of the trip and the red the end of the trip. The left graph is the initial distance along the trip metric which is noisy and the right graph is the new distance along the trip which is much smoother.

The relationship between the new distances and old distances is perplexing and we have not determined why the old distances were as incorrect as they were. Intriguingly, the old distances appear to form clusters of inaccuracy. These clusters may be useful for seeing why not in isolate, but with the new distance metric, the models are able to better predict bus delays. Figure 2 shows this relationship.

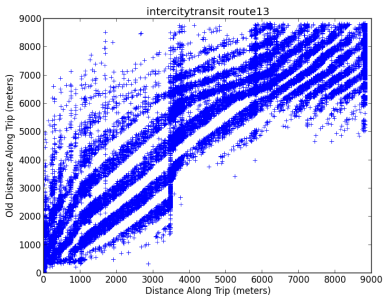

Figure 2: The relationship between the new distance along trip compared to the old distance along trip for the same Intercity Transit Route 13.

Other errors that the data is known to have include multiple updates for a single point and the pole problem. The pole problem occurs when the bus finishes the route just shy of the true end and then sits for a break. In this case, the updates assume that it has not finished the route and therefore continue to add delay to the end of that trip and the beginning of its next trip. The bus may also send updates while it is returning to the bus depot.

4 Feature Sets

Our features can be broken into four categories: Spatial, Temporal, Route-Based and Delay, as described in Table 1. The Spatial group contains measurements for the location of the bus at the update, including the refined distance (new) derived from the latitude, longitude, and route shape as described above.

The first temporal feature is the epoch time as reported in each update. This feature is recorded as the number of milliseconds since January 1, 1970. In case specific features of time are specifically relevant to predict bus arrival times, the day of the week, the days since the route schedule was reset, and the time of day are derived from the epoch time. These three additional temporal features are referred to in the feature sets as time derivatives and are used in these partial feature sets in place of epoch time.

The route based group uses the trip identification number for the bus as well as a derived trajectory. The trajectories represent the schedule deviation over the bus route. Given all the data bursts of a trip, the deviation was interpolated for every 100 meters. Duplicate updates at the same location are removed, selecting the latest one, with the exception of choosing the first update at the end of the trip in order to avoid the pole problem. These trajectories do not preserve temporal data, just distance by delay, so they are used in separate models.

The last category, Delay, records how much the bus has deviated from its schedule at each update. Our model predicts this feature and therefore, it is used both as the comparison for our error calculation as well as in the calculation of the predicted deviation. The delay is reported in 60 second intervals, with a positive delay indicating that the bus is behind schedule and a negative delay indicating that it is ahead of schedule.

Feature	Description
Spatial	
Old Distance	The distance along the trip given in the information burst
New Distance	The distance along the trip that we get from the data pre-processing
Latitude	The latitude of the bus at the update.
Longitude	The longitude of the bus at the update.
Temporal	
Epoch Time	The number of seconds since January 1, 1970.
Day of Week	Monday = 1, Tuesday = 2, ..., Friday = 5
Days	The number of days since the service period started.
Time of Day	The number of seconds since midnight of that day
Route Based	
Trip ID	The ID of each trip (i.e. the 6:15 am trip has ID = 1, the 6:30 am trip has ID = 2, etc.)
Trajectory	The interpolated deviation at every 100m
Delay	
Schedule Deviation	The self-reported offset of bus progress to scheduled arrival time

Table 1: This table describes each of the features that we used in our models.

From these features, we created seven sets of features to use to predict deviation. Below is a description of what features were included in each feature set. For all of these sets, every feature is normalized.

Feature Set	Features
New Distance Only	New Distance
Epoch Time Only	Epoch Time
New Distance and Epoch Time	New Distance Epoch Time
New Distance and Time Derivatives	New Distance, Day of Week Days, Time of Day
Old Distance and Time Derivatives	Old Distance, Day of Week Days, Time of Day
Full 9 Feature Set	All except trajectories and deviation
Trajectories	Trajectory

Table 2: This table shows which features are included in which feature sets.

5 Models

We used the k-nearest neighbors and kernel regression algorithms to estimate the number of seconds a bus has deviated from its schedule. We choose these algorithms for their computational complexity and their ability to predict without relying on linear projections. A linear regression on the features would be insufficient because of the variable nature of schedule deviations.

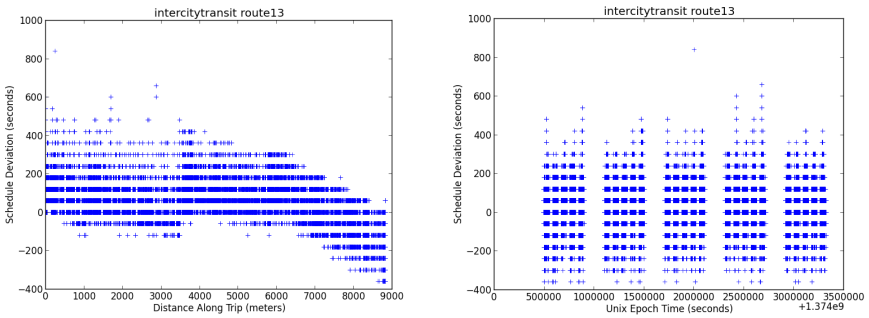

Figure 3: The left figure show the new distance along trip against deviation and the right figure shows the epoch time against the deviation (the gaps in the epoch time represent the weekends). From both figures you can see that both of the features have high variability of y within each value of x.

On the other hand, k-nearest neighbors (kNN) and kernel regression predict the schedule deviations by historical data more similar to the time at hand. Thereby, instead of using just the last update for the bus, for instance, recent updates of the bus the last time it was close by to a stop can give a more detailed image of the prediction. Both of these models were written in python from scratch only using the numpy libraries for data structures and methods.

In our implementation of k-nearest neighbors, we find the distance to each of the points in our training dataset from our test point and average the the deviation for each of the k closest points. We ran kNN using each possible value of k and select the k that provides the lowest validation error.

Kernel regression weights each point in the training dataset relative to our test point and then takes the weighted average of all the deviations. We implemented kernel regression using the following kernel:

$$kernel(x, query) = e^{\frac{-distance(x,query)^2}{\rho^2}}$$

In order to determine the best rho (kernel width), we ran kernel regression with each possible rho and selected the rho which provided the lowest validation error.

The amount each feature contributes to the distance between two points plays a large part in the accuracy of both kNN and kernel regression. In order to determine the best weights for each feature, we used stochastic gradient descent refine them. Stochastic gradient descent iterates through each data point in the validation set, predicting the desired feature and changing the weights of input features each time by the amount the feature contributed to the error. Stochastic descent requires an eta value, the amount the weights are changed by each iteration. For our data of 63k points, we used an eta of 0.00001. For each value of k or kernel width that we tested, we would run stochastic gradient descent to determine the weights. Having different weights for each model parameter ensures that we did not select a poor k due to suboptimal weights for that selection of k or kernel width.

Once we had determined the best model, we did some more in-depth analysis of the predictions. We did this by evaluating the predictions for different prediction periods. A prediction period constrains the data that can be used by our models to mimic looking at the predictions however many minutes before the buses arrival. For instance, if the bus arrived at 10:30 am, our 30 minute prediction period would only look at the data from before 10:00 am. We tested our best model with prediction periods of 60, 30, 15, 10, 5, 2, and 1 minutes before arrival.

5 Results

In order to begin evaluating our models, we first evaluated which feature sets were best at predicting the schedule deviation. We ran k-nearest neighbors at multiple values of k and kernel regression at multiple kernel widths. For each of these runs, we did stochastic gradient descent with five passes over the data to determine the best feature weights for the distance function. Below are the root mean squared error (RMSE) values we had for each model (table 3).

Validation Model RMSE	***k*-Nearest Neighbors (over k)**							**Kernel Regression (over kernel width)**					
Feature Set	1	2	5	10	15	30	60	0.125	0.25	0.5	1	2	4
New Distance Only	113	100	94	88	86	84	83	83	84	86	94	103	109
Epoch Time Only	137	134	125	124	124	125	117	112	112	112	111	111	111
New Distance and Epoch Time	106	103	92	88	86	86	84	84	84	85	87	95	103
New Distance and Time Derivatives	103	99	87	92	93	84	83	96	82	79	80	85	95
Old Distance and Time Derivatives	125	109	109	132	110	121	111	131	125	110	106	107	109
Full 9 Feature Set	24	25	23	22	24	24	29	25	24.2	24.4	29	46	70
Trajectories	67	60	54	58	55	56	58	70	65	64	48	53	63

Table 3: The table shows the root mean squared error on our validation set for each of our test cases. We can see that our models using less features or just the old distance along the trip and not the new distance along the trip tend to do less well. The best feature sets and parameters are highlighted for each model.

For kNN, we can see that our best feature set is the Full 9 Feature Set. The validation RMSE is lower for all values of k. Similarly, we have lower validation RMSE for all values of rho. In both cases, there is a large jump (~30 seconds) in validation RMSE to the next best model, the trajectories.

We were also interested in the effect of our pre-processing the data to get the new distance along the trip had. It is clear from the fact that the New Distance and Time Derivatives does 26 seconds better than Old Distance and Time Derivatives at their best values for kNN and 27 seconds better at their best values for kernel regression that the cleaning of distance along the trip did have an effect on predictions.

Once we knew that the Full 9 Feature Set was the best of the feature sets and kNN was the best model for that feature set, we did a more in-depth analysis of its results, particularly at different prediction periods. We wanted to see if there was a difference in accuracy as the data approached the buses arrival. We used the best k for the Full 9 Feature Set, k=10, to run the tests on. Interestingly, there was not a large difference in the RMSE at different prediction periods as shown in table 4.

	Error		
Prediction Period	Mean	Stdev	RMS
1 minutes	17	26	31
2	16	27	31
5	14	27	31
10	17	30	35
15	15	28	32
30	17	31	35
60	16	30	34

Table 4: This table shows the kNN error for different prediction periods when the feature set is Full 9 Feature Set and k=10.

In order to better evaluate our model, we compared our results to a baseline of having no real-time information where you would always predict that the bus will always be on time. We tested our models using a different route, Route 12 of intercity transit, to check for overfitting to our specific training route. Results were highly favorable for our analysis. The mean error for predicting on time arrivals was 70 seconds off, while it was only 14 seconds and 16 seconds off for our best

kNN and kernel regression. There was still variability in the estimates that make the root mean square error larger for each model. Figure 4 shows the results of this test.

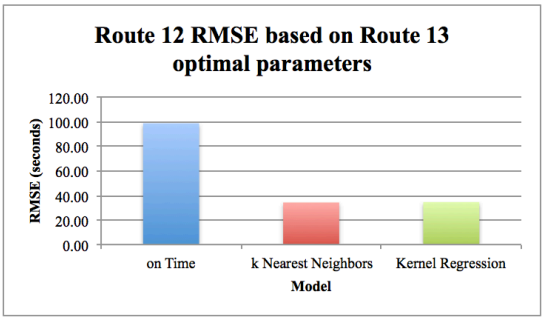

Figure 4. This shows the difference in RMSE for the baseline (rmse = 99.01) , the best k-nearest neighbors model (k=10, rmse = 34.34), and the best kernel regression model (kernel width = 0.25, rmse = 34.68)

The most meaningful features are the spatial features as shown by figure 5. The weights shown are for the normalized features, so larger values indicate features that are more important. Thereby, the spatial locality of a bus stop update is more significant than how long ago the update took place. Even though old distance is weighted higher than new distance, models with just the old distance did not perform as well as the models with just the new distance.

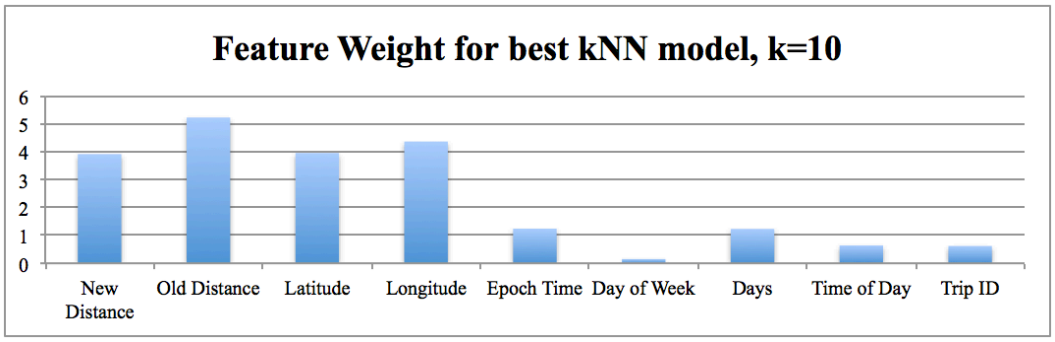

Figure 5. Coefficients for the difference of each feature between each point. This particular set shows the weights produced by stochastic gradient descent for the model with the best accuracy, kNN with 10 neighbors.

5 Future Work

Further applications of our research would be to expand our algorithms to the rest of the service areas including King Country Metro. The data feeds for King Country Metro are generally noisier than Intercity Transit's. Also, it would be interesting to compare the weights generated for each prediction period and see how they vary and if there is an optimal set of weights that will work for all prediction periods.

Additionally, it would be pertinent to integrate our model with the One Bus Away application. The data feed processing would have to be done server side, but nearest neighbors queries or single kernel regression samples would be quick.

6 Conclusion

Through our work, we explored the features of the Puget Sound real-time feeds. There were many features that were noisy, but when we used all the noisy features in combination with the refined distance metric, we were able to get the best results. Spatial features performed well in the model and we would recommend weighing spatial features highly in models predicting bus arrival times. The performance of a bus along the same place in a route is more dependable than how the bus was delayed earlier in the route. Furthermore, bus predictions were relatively stable regardless of how far out the bus was, 1 minute out or 30 minutes out.

In conclusion, we were able to decrease root mean square error by approximately 65 seconds from the baseline of predicting on-time arrivals. Both k-nearest neighbors and kernel regression performed well, with neither algorithm clearly doing better.

References

[1] King Country Metro website: http://metro.kingcounty.gov/am/metro.html

[2] Ferris, B., Watkins, K., & Borning, A. (2010) OneBusAway: Results from Providing Real-Time Arrival Information for Public Transit, *CHI 2010*, Atlanta, GA, USA

[3] Aaron Gooz, Kari Edison, and Alan Borning (2012) Benefits of Real-Time Transit Information and the Impacts of Data Accuracy on the Rider Experience. *TRB 2013 Annual Meeting*

[4] Cathey, F.W., & Dailey, D.J. (2003) A prescription for transit arrival/departure prediction using automatic vehicle location data, *Transportation Research Part C 11* (2003), pp. 241-264

[5] Bin, Y., Zhongzhen, Y. & Baozhen, Y. (2006) Bus Arrival Time Prediction Using Support Vector Machines, *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, 10:4, pp. 151-158

[6] Dalia Tiesyte, and Christian S. Jensen (2008) Similarity-based Prediction of Travel Times for Vehicles Traveling on Known Routes. ACM GIS 2008, Irvine, CA, USA

[7] Mathieu Sinn, Jin Won Yoon, Francesco Calabrese and Eric Bouillet (2012) Predicting arrival times of buses using real-time GPS measurements. International IEEE Conference on Intelligent Transportation Systems 2010, Anchorage, AK, USA