NYC Bike Trips Analysis and Prediction

Joseph Baker (jib2126)
Pedro Pérez Sánchez (pp2550)
December 2016

1 Introduction

Citi Bike is a bicycle sharing service in NYC with rental stations throughout Manhattan, Brooklyn, and a limited part of Queens. Many NYC residents use these bikes for commuting and for leisure. They provide apps that show the real-time status of their stations. This status includes information about how many bikes are available as well as how many docks are open. Notably absent though, is any form of forecasting. *Citi Bike* has made the trip data public and there are numerous interesting patterns found in this data set. A cursory look shows many cyclic patterns present, which is not surprising given people's daily, weekly, and seasonal schedules. In this paper we present simplification of the 656 stations by way of Gaussian mixture modeling and show a proof of concept prediction model that uses the cluster data to predict the number of trips to/from each station. We made all our code freely available on Github [1].

2 Data Exploration

Trips Data Set

The *Citi Bike* data set [2] is over 40 million records starting on July 1, 2013 at midnight, and continuing to the present. Each record contains the information about a single bike trip:

- Station id and time for both the beginning and end of the trip.
- Information about the start and end stations such as the stations' names and locations (latitude and longitude).
- Information about the renter. For subscribers this includes their gender and birth year.

Certainly many interesting questions can be asked regarding this data, for this study we focus on the start and stop station ids and times.

Data Manipulation

In order to have smoother data for our models and with the purpose of reducing the amount of used memory, we decided to group trips' start/end times into 30-minute 'buckets'. We represent the data set using two 656×57024 matrices S and E, where $S_{i,j}$ and $E_{i,j}$ respectively represent the count of trips that started and ended at station i at any time between minutes $t_0 + 30j$ and $t_0 + 30(j+1)$, where t_0 is July 1, 2013 at midnight. This representation allows us to randomly access the bucket corresponding to any time t, in minutes, as:

$$bucket(t) = \left\lfloor \frac{t - t_0}{30} \right\rfloor$$

For training our models, we used a combined 'flow' matrix F, where $F_{i,j}$ represents the net gain or loss of bikes at station i due to rental activity in interval j:

$$F = E - S$$

Data Exploration

Before looking at the data, we had the following assumptions:

1. There exists a yearly pattern showing more trips during the summer and less trips during the winter.
2. There exist strong weekly trends based on a typical 9-5 commute schedule on weekdays.

In order to verify or disprove our assumptions, we decided to visualize our data using the following plots:

- Figure 1 highlights the seasonal trend of the rentals and verifies our first assumption. Unsurprisingly, there is high usage during summer, decreasing usage during fall, low usage during winter and increasing usage during spring.
- Figure 2 shows the average weekly behavior for some stations, proving our assumption that some stations follow a weekly commute trend.

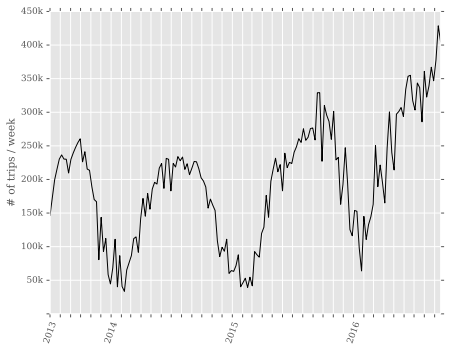
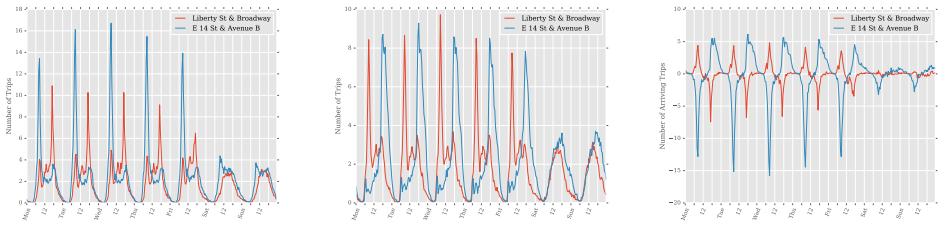


Figure 1: Total trips per week from 2013 - 2016

By looking at the 'flow' weekly pattern shown in Figure 2c, we noticed a clear difference between some stations with a positive flow in the morning and negative in the evening (business stations), and vice versa (residential stations). This motivated the idea of clustering stations using a mixture model.

3 Clustering Model

For clustering, each station is encoded as a 336-dimensional vector, which represents the average weekly flow for that station, split in 336 intervals of 30 minutes each. Given that all stations of the same type tend to have similar values at each component of the vector, a Gaussian Mixture Model (GMM) was used to fit the data. The graphical model for the GMM is presented in Figure 3 [3]. In this model, each interval is treated as an independent dimension, which loses information about reality, since the usage at 9:30 is probably highly correlated with the usage



(a) Number of trips started at station over week

(b) Number of trips stopped at station over week

(c) Net change in bikes at station over week

Figure 2: Average weekly behavior for selected stations

at 9:00. However, this naive modeling assumption was shown to work reasonably well in order to divide stations into clusters.

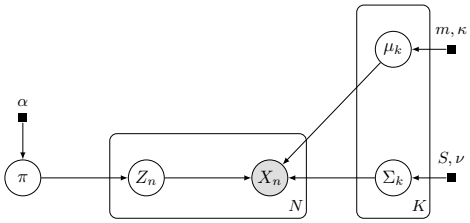


Figure 3: Graphical model for GMM.[1]

For learning the assigned cluster for each station, a collapsed Gibbs sampling algorithm was used. It was decided to use $K = 3$ as the number of clusters, given that 2 clusters ended up having business and residential stations in the same cluster, and a larger number of clusters resulted in clusters with very few stations.

After having the cluster assignment for all stations, we decided to show two plots to verify they match with our expectations.

Figure 4 is a visualization of the 336-dimensional mean vector for each of the three components:

- One cluster strongly resembled a business or other destination (e.g. university) of a commute cycle: each workday had a large influx of trips in the morning and a large outflux of trips in the evening. Weekends saw a similar pattern at a dramatically reduced volume.
- The second cluster was the reciprocal of the first, corresponding to residential stations. These stations have a large exodus of bike trips in the morning and in the evening they have a large volume of inbound trips. Again the weekends are marked by the same pattern at a reduced scale.
- The third type of station is one that is fairly balanced. Some stations just don't have a large amount of throughput so the values for each 30 min interval are small. Others do have usage, but the usage is pretty balanced so when considering the flow, it remains close to 0. We were surprised that a large portion of the station network actually falls into this third category. We expected many more stations to be in the residential category, especially outside of Manhattan.

Figure 5 shows the location of each station, color-coded depending on its assigned cluster, following the same color scheme as Figure 4. We can see that many of the business stations are located downtown and towards the center of Manhattan, while a majority of residential stations are located on the sides of the island, such as the Lower East Side and in the West Village. This distribution of the types of stations in NYC gives us some interesting insight into the various neighborhoods and districts in the city.

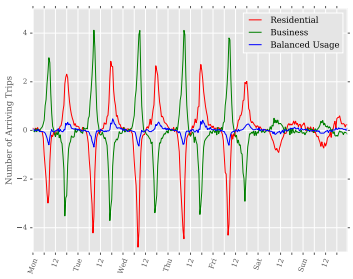


Figure 4: Cluster means

[1] See Appendix A for details about variables and distributions

Figure 5: Station map with cluster assignments

Posterior Predictive Checks

As we discussed in class, posterior predictive checks are a useful tool for confirming that a model makes sense for the given data. GMMs have a straightforward generative process, and we used this with the posterior to generate a set of points that we then could compare with our original data. As we show in Figure 6 the model correctly captures the morning and afternoon commute peaks although it does seem to have a higher variance around the baseline than the original data did. This suggests that while it could potentially be improved, it does capture the main structure of the data.

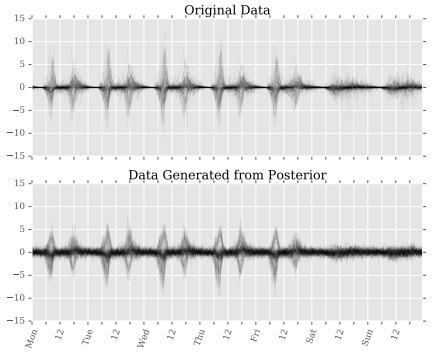


Figure 6: Posterior predictive check

4 Prediction Model

In this section, we show that knowing the cluster assigned to each station can better inform predictive claims about its behavior, such as predicting how many less bikes a station will have a half hour or hour past what we have data for. We compare the performance of a prediction model based on the clusters obtained in Section 3 with other, more naive, prediction models.

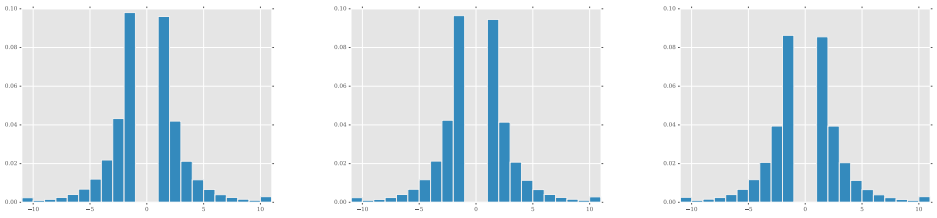
Naive Average Model

Our first naive predictor simply uses the average week as a guide for the behavior of the week under question. For instance, if you ask the predictor what the net change in bikes due to trips will be for next Thursday at 7:30 PM at a certain station, this predictor simply looks at the 7:30 PM value for that station's average week and reports it as the prediction. Obviously this isn't using any probabilistic modeling and if the actual usage is different from the average it will miscalculate the value. Another shortcoming of this model is that, for it to perform at all, it needs to store the average weekly behavior for each station. With 656 stations and a week consisting of 336 data points, we need to store 220,416 data points, and that grows by 336 with each new station added to the system. Another immediate drawback is that the predictions for new stations are often going to be wrong until enough time has passed for the average of that station to be a meaningful representation. Despite these drawbacks, because it is trivial to calculate the average weekly behavior for a given station (as we demonstrated in Section 2) this model is useful for a simple exploration of predictions and to become familiar with the means by which we determine a model's accuracy.

To analyze the performance of this simple model we trained the model on data up to June 1st, 2015. Then, we picked 5,000 random date points over the next year to query the model. On each of these particular dates we ask the model what the flow will be for the stations and we then pick 80% of the active stations to use in the test. Active stations are simply stations that have had some activity by the date in question. This process gave us a list of around 1.8 million test points that we could compare with the actual trip flow values in the data, about 8% of the total. For the naive average model, 62% of the predictions were within 1 'flow unit' of the actual value and a further 19% were between 1 and 2 units (Figure 7a). This indicates that while simple, this model actually does fairly well at predicting the net flow of a station.

Average Model with Seasonal Component

When we were exploring the data we noticed very strong seasonal cycles in the trip volume. This isn't surprising as many more people are willing to ride bikes in the summer than in the



(a) Naive avg: 61.69% $\pm$ 1 (b) Seasonal avg: 62.33% $\pm$ 1 (c) Cluster: 64.86% $\pm$ 1

Figure 7: Error breakdown for various predictors (note that all values less than -10 or greater than 10 are grouped in the outer buckets)

cold, snowy winter. The next obvious extension to our Naive Average Model is to introduce some multiplicative component that modifies the prediction based on what time of year the prediction is for. We grouped the total volume of start trips into weekly intervals and fit a Gaussian Process [4] with a combination kernel to the data. The kernel consisted of a radial basis function component for the long term trend of increasing popularity of the *Citi Bike* service, an exp-sine-squared component for tracking the seasonal cycles, and finally a white noise component to account for the inherent noise in the data. With this fitted model we had a function that would allow us to model and predict the seasonal fluctuations and by comparing that to the average baseline we now had our multiplicative factor (Figure 8).

We performed the same testing procedure as we did for the Naive Average model and we found that overall the performance of this enhanced model was almost exactly the same (Figure 7b). This was very surprising to us as the seasonal affect on the total trip volume was undeniable. We confirmed that there are seasonal patterns in the flow for the stations, but it appears that the impact of the seasons is more nuanced that a simple multiplicative factor.

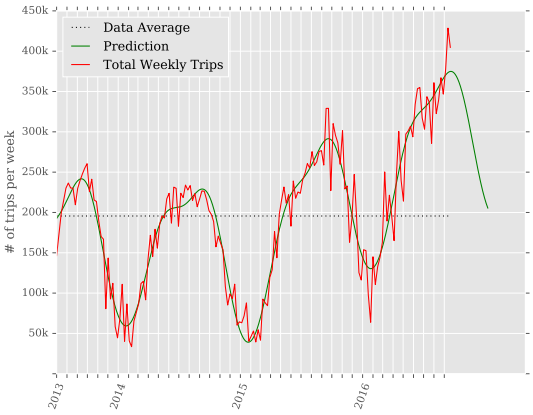


Figure 8: Total trip volume with gaussian process trend fitted

Even if this model had performed better, it still falls short on the model storage issue as well as not being able to generalize to new stations quickly. In fact, if a station is added at a time period where the system usage is far above or below average, the seasonal factor will actually exaggerate the predicted behavior at that station, as that station's average will only be based on the seasonal period while the algorithm assumes that a station's average is in line with network's average.

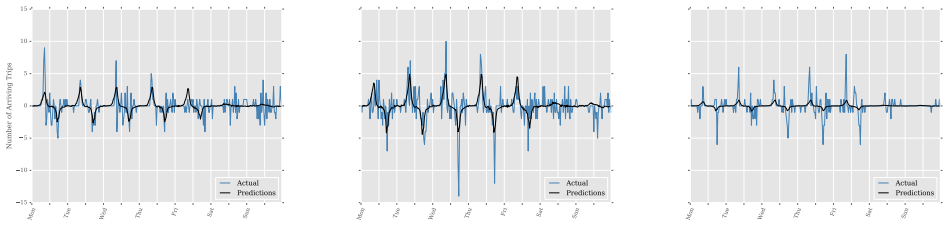
Clustering Model with Seasonal Component

An alternative to using each station's average when constructing predictions is using the mean for the cluster that the station belongs to. Since the stations within a cluster have similar behaviors, we aren't sacrificing much predictive power and in return we only have to store the 3 cluster definitions rather than 656. Furthermore, new stations can be added to the system and they just need to have enough data for them to be matched with a cluster, something that only should take a full week. Realistically this makes sense. If *Citi Bike* opens a new station in a residential area and, after a week, we realize that it follows the residential commute pattern, we have a reasonable basis for our further predictions because we know how the residential commute pattern plays out in other stations throughout the year.

Again we used the same testing procedure and were pleasantly surprised that the cluster based predictive model works better than the naive average model (Figure 7c). We assumed that the cluster wouldn't have as much information for a particular station as the station's average, but it appears as though the regularization caused by using the cluster mean actually increases the accuracy of the model for unseen examples. This model is around 85% accurate to within 2 trips of the actual value for a 30 minute interval. This level of accuracy is quite reasonable for a rough estimate of whether the 10 bikes left at a station will still be there in 30 minutes.

Model Reflection

While trying to diagnose why our simple multiplicative approach for seasonal correction did not improve the accuracy as much as expected, we considered samples from a single station across 3 seasons (Figure 9). It appears that, although the average behavior for a station fits nicely into a commute based waveform, a single sample of that behavior is quite choppy. Much of the error volume could be due to the inherent variance of this metric and a crude scaling based on season may simply bypass the deeper problem. When looking at aggregations of this data, much of the data seems to behave and fit into easily recognizable patterns. But when one considers the data at a much smaller scale, it has quite a bit of noise in it. Building a model that is able to work reliably on this small noisy scale is challenging.



Spring Summer Winter

Figure 9: 3 weeks taken from a station in each season along with predicted value for those weeks

5 Future Models

The models we presented are simple models for prediction and within our analysis we limited ourselves to past data and only predicting the net change in bikes at a station. One can easily imagine other extensions of these models that take temperature or precipitation as a factor in addition to the season or one that considers many more clusters to try and find differences among the large clusters that we described. Another useful idea would be to make use of the real-time API that *Citi Bike* provides to determine the current exact number of bikes at a station and combine that with our model which predicts what the net change in bikes at a station due to trips is for the next half hour or hour. This could inform riders as to whether there will be bikes or available docks at a station before they get there, which was our original goal for this project.

We feel that our initial models show that the data provided is valuable and sufficient for answering these questions and probabilistic modeling can provide insights into the behavior of the users of *Citi Bike*.

6 Conclusion

The *Citi Bike* trip data has interesting and valuable information hidden within it. With a bit of data cleaning and probabilistic modeling, we could uncover these insights. The models that we've learned in class have allowed us to transform the data into valuable descriptive and predictive models.

In this paper we have shown how the data can be clustered to discover behavior trends in neighborhoods where the stations are located. We also showed that by combining these trends with

overall usage patterns we can construct multiple predictors for the behavior of the immediate future. Predictive models like this can be extremely valuable for users of the *Citi Bike* service as well as *Citi Bike* itself. Knowing when and where the trips will occur directly informs the strategy for manually moving bikes from one station to another to keep demand satisfied. It also tells them which neighborhoods would benefit from more stations or larger stations. Finally, city planners can also make use of these results as they deal with encouraging bike riding and by determining where bike lanes would be most valuable.

References

[1] J. Baker and P. P. Sanchez. (2016) Code for neighborhood analysis for citi bike trip data. [Online]. Available: https://github.com/pedropobla/gms_project

[2] N. Y. C. Citibike. (2016) Citi bike system data. [Online]. Available: https://www.citibikenyc.com/system-data

[3] D. Blei, "Foundations of graphical models class notes. bayesian mixture models and the gibbs sampler," 2016.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. [Online]. Available: http://scikit-learn.org/stable/modules/gaussian_process.html

[5] P. Resnik and E. Hardisty, "Gibbs sampling for the uninitiated," DTIC Document, Tech. Rep., 2010.

[6] H. Kamper, A. Jansen, S. King, and S. Goldwater, "Unsupervised lexical clustering of speech segments using fixed-dimensional acoustic embeddings," in *Spoken Language Technology Workshop (SLT), 2014 IEEE*. IEEE, 2014, pp. 100–105.

A Appendix

Graphical model that was used for clustering stations:

- π ~ Dir(α)
- μ_k ~ Norm(m, κ)
- each diagonal element (i, i) of Σ_k ~ Scaled-Inv-Chi^2(S, v)
- Z_n ~ Cat(π)
- X_n ~ Norm(μ_{Z_n}, Σ_{Z_n})