Forecasting Utilization in City Bike-Share Program

Christina Lee, David Wang, Adeline Wong

1 Introduction

In this project, we use a variety of machine learning models to predict the number of bikes in use in a given hour in a public city bike-share program. Bike-share programs allow riders to check out a bike at one location and deposit it at another – usually at an electronic bike station. These public bikes offer an alternative to providing maintenance and security for a personal bike, and fit transportation niches including the “last mile” between public transit termini and a commuter's final destination [1].

Bike-share systems have a relatively young history. Whereas bicycles were invented in the early 1800s and cars have been around since the late 1800s, bike-share systems only came into existence in 1965. Bike-share systems and computing actually have an intertwined history. Early bike-share systems suffered from theft of the bikes, so later systems evolved locks. As computing devices have grown smaller, it became possible to build electronic bike stations that allowed users to check bikes out with a card and enforce the bike's return with a hefty fine [1]. Such systems also enable tracking of bikes' movements, and modern bikes are even equipped with GPS, which can allow fine-grained tracking of bikes and rider habits.

We think this is a cool project because while the mechanics of bike-sharing are simple, the individual rider choice and the movements of bikes during a day allow room for a great deal of complexity. Machine learning is the tool of choice when working with problems that are either too complex for a human to comprehend or are infeasible to program by hand, and it seems an appropriate tool to apply to the prediction of bike demand over time. In addition, the relative newness of bike-share systems and their steadily growing popularity make this an interesting problem from a practical perspective. We hope that the results of this study will help bike-share program managers better design models to predict utilization, which could inform decisions such as when to perform maintenance and when and reallocate bikes within the system without affecting customer satisfaction with bike availability.

2 Dataset

The data used for this project comes via the Kaggle contest “Bike Sharing Demand” (Kaggle dataset from [3]) from Capital Bikeshare, based in the Washington, D.C., metro area. Capital Bikeshare began operations in September 2010 with 400 bicycles at 49 stations and by September 2012, had grown to 2,800 bikes at 288 stations [2].

The Kaggle data spans the two years from January 1, 2011 to December 31, 2012. For the purposes of the competition, in addition to training data, there was a standardized, unlabeled test set provided to all contestants. The training data covers the first 20 days of each month, and the test data covers of the remaining ten or eleven days. The test data consists 10,886 data points, one for each hour, with 12 features – given below. The test set consists of 6,493 data points with the same features.

Field	Description
datetime	hourly date and timestamp
season	1 – spring, 2 – summer, 3 – fall, 4 – winter
holiday	1 if the given day is a holiday, 0 otherwise
workingday	1 if the given day is neither a holiday nor a weekend, 0 otherwise
weather	1 – clear to partly cloudy, 2 – misty and/or cloudy, 3 – light rain/storm, 4 – heavy rain/snow/storm + fog
temp	temperature in Celsius
atemp	“feels like” temperature in Celsius
humidity	relative humidity
windspeed	wind speed

casual	number of bikes rented by non-registered users
registered	number of bikes rented by registered users
count	total number of bikes rented

Table 1. Description of data fields.

3 Features and Preprocessing

From the raw data, the following features were extracted: Raw data was used as-is for binary data (holiday, workingday) and numeric data (temp, atemp, humidity, windspeed). Categorical variables (season, hour of the day, weather) were divided into multiple binary features. For example, instead of keeping season as a feature on its own, there are three binary features: season==2, season==3, season==4. The season==1 feature is excluded because it is determined by season==2, season==3, and season==4, and dependent variables can have deleterious effects on machine learning models. The result is a total of 35 features. For this problem, the target variable is count; i.e. total number of bikes rented in a given hour. Thus, the task at hand is a regression problem.

4 Models

The following models were used to predict bike utilization per hour:

1. Poisson regression

We expected that utilization would depend on the weather buckets and hour buckets, so it makes sense to run a general linear model regression on the various buckets. The Poisson distribution is used for predicting the number of events over time, and therefore seems to be an appropriate model for this problem. Poisson distributions are in the exponential family, so this model is a standard GLM.

2. Neural network

The Poisson regression model assumes that the variables are more-or-less independent. Another way to view the problem is throw at the problem a black-box machine learning model that is able to capture complexity, such as interacting variables. Neural networks are good at recognizing these hidden patterns, hence our choice to train a neural network. After trying a variety of parameters, the final neural network consisted of four hidden layers (9 nodes in the first hidden layer, 3 in the second hidden layer, 3 in the third hidden layer, and 23 in the fourth hidden layer), trained using the `neuralnet` R package and learning rate of 0.001 with resilient backpropagation with weight backtracking.

The following are intermediate training and test results with increasingly complex models. The error function used was root mean square log error (RMSLE), which is explained in the following section.

Model	α=0.001, train	α=0.001, test
25	0.52980	0.76402

Model	α=0.001, train	α=0.001, test
30	0.64128	0.82307

Model	α=0.001, train	α=0.001, test
23-9-3-3	0.42730	0.54228

Model	α=0.5, train	α=0.5, test	α=0.3, train	α=0.3, test	α=0.01, train	α=0.001, test
9-9-9-9	0.44855	0.50273	0.47535	0.54223	0.45186	0.50879

Model	α=0.3, train	α=0.3, test	α=0.1, train	α=0.1, test	α=0.001, train	α=0.001, test
9-3-3-23	0.45375	0.49966	0.47132	0.51644	0.45102	0.48992

3. Markov model

One of the more obvious patterns in this problem is that the data is cyclic. During weekdays, there are the morning and evening rush hours, and on weekends, folks take joyrides and run errands in the afternoons. In contrast to making no judgments about the data, a third way of viewing the problem is to try to capture in the model some of the obvious cyclicness of user demand. To do so, we trained a set of Markov models, one for each (working day or non-working day, hour, current weather) combination. In addition, bikes can be in one of two states, either checked-in or checked-out. So the model captures state in four variables.

One of the strongest assumptions made by Markov models is that the transition matrix does not change over time. Since this is obviously not true over the course of a day – for example, users checking bikes out to go to work at 8 AM and tend to be checking in bikes after arriving at work at 10 AM – we model demand by bucketizing by hour and giving each hour its own transition matrix. In Figure 1 are graphs of average demand in the training set and predicted on the test set.

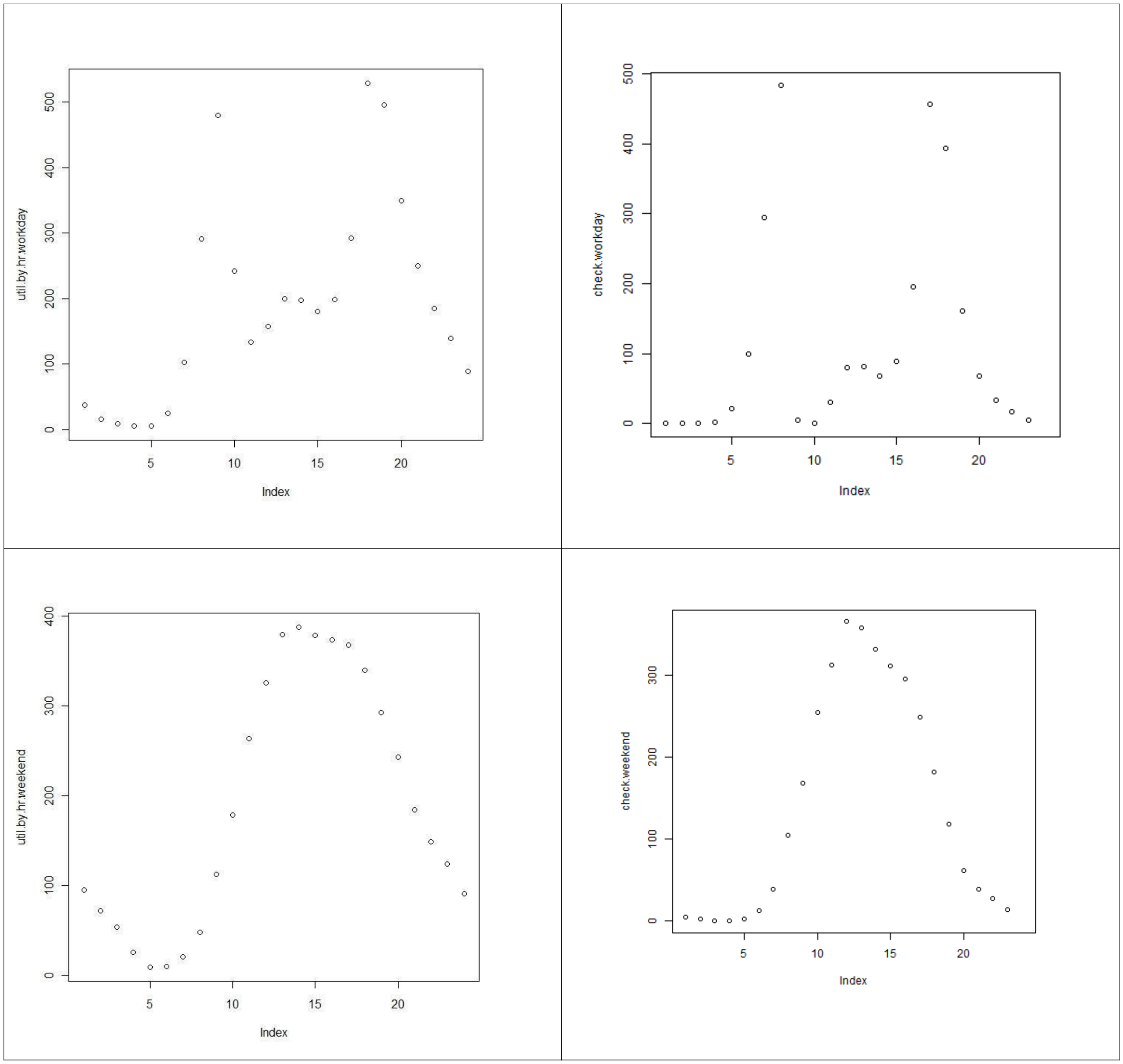

Figure 1. The top figures are working days, and the bottom figures are weekends and holidays. On the left are averages on the training data, while on the right are averages of predictions on the test set. The model was able to broadly capture demand fluctuations, including the weekday morning and evening rush hours.

5 Results

To evaluate our predictions, we used root mean squared logarithmic error:

$$\text{error} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(log(p_i+1) - log(a_i+1))^2}$$

where p_i is the value of prediction for the i-th datapoint, and a_i is the actual value. The metric is reasonable since the error value grows larger as the log value of the ratio of the predictions and the actual values grow larger. The chart below summarizes the errors for the various models we trained:

Model	Training error	Test error
Neural Network	0.451017	0.48992
Poisson Regression	0.676233	0.69899
Markov Model	0.73738	1.70463
Mean Value Benchmark*	1.569198	1.58456

*For reference purposes, predictions take the mean value of the training data.

In the Kaggle competition, we are rank 570 out of 1,580 teams entered. The top score (lowest reported error) is 0.24976.

6 Discussion

Qualitatively, we see that both neural network and Poisson regression perform much better than the mean value benchmark. Neural network outperformed Poisson regression quite significantly. This result was as expected. We used Poisson regression under the assumption that, given the nature of the problem (predicting number of users given different factors), it would lend itself well to Poisson regression. However, this means that if the distribution does not follow a nicely Poisson distribution, it would result in large errors. Neural networks, on the other hand, are a flexible model that are good at recognizing complex, non-linear patterns that are not easily observable by humans or other learning algorithms; it picked up on the patterns of the dataset and adapted itself.

The Markov model had mixed results. While, it had a training error in around the same ballpark as the neural network and Poisson regression, its test error was worse than even the mean value benchmark. It appears that the trained model overfit the training data. In an overfitting regime, it would be reasonable to reduce the number of variables. However, this model already used many fewer variables than both the neural network and Poisson regression, which used the full set of 35 variables. Variable reduction should probably come in the form of training fewer trainsition matrices – perhaps one matrix for group of two or three consecutive hours would be an improvement.

To better understand the performance of the models, keep in mind that this is a regression problem and we are trying to predict the number of bikes in use. The baseline error (i.e., we predict that the number of bikes checked out is always the average over all training data points) is 1.58456. An error of 0.69899 means a 59% reduction in error (2.4 times less error). The error of 0.48992 means a 67% reduction in error (a 3-fold decrease).

7 Conclusion

As expected, neural networks were able to capture a great deal of subtlety in the data. Surprisingly, the attempt to capture transitions over time via the Markov model had terrible performance.

8 Future

There seems to be evidence that casual users and registered users have different usage patterns[4]; it would be interesting to look at that. Another interesting problem would be to look at the interaction of different transportation methods and bike-share; for example, how does a metro strike affect the utilization of a bike-share program?

The feature space could also be expanded by using consecutive pairs of hours as data points, rather than single hours. Furthermore, the Kaggle dataset is from only one bike-share program; we would like to look at data from different bike-share systems as well. For instance, Bay Area Bike Share provides a similar dataset as the Capital Bikeshare one, but is enhanced with the start time and station and end time and station fo each ride [7]. JCDecaux has a rich data set across many countries, but data is only accessible in real time via an API (presumably the use-case is for visualizations of bikes currently in the system and their precise locations) [5].

Another idea is to apply a recurrent neural network, that is, a neural network that allows cycles of "neurons." This effectively gives the neural network state, which can be thought of as a short-term memory, which could allow them to take into account cyclical variables, such as daily demand and seasonal changes. While the upside is great, unfortunately, recurrent neural networks are a relatively new idea and training them has so far proved challenging [6]. The short term memory, in practice, tends to be short, so it is uncertain whether it would even be able to capture a the 24 x 7 states required to learn weekly variations in bike demand, let alone the year-long variation due to weather.

Finally, since the data comes from the early history of the bikeshare program, as it was ramping up, error might be reduced by fitting a line on maximum daily demand, and scaling predicted usage based on the demand capacity at that time in the program's history.

References

1. Bike sharing system. (n.d.). Retrieved December 11, 2014, from en.wikipedia.org/wiki/Bicycle_sharing_system.
2. Capital Bikeshare. (n.d.). Retrieved December 11, 2014, from en.wikipedia.org/wiki/Capital_Bikeshare.
3. Fanaee-T, H. & Gama, J. (2014) Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence* 2(2-3), 113-127.
4. Gebhard, K. & Noland, R. (2013) The Impact of Weather Conditions on Capital Bikeshare Trips. *TRB 2013 Annual Meeting*. Retrieved December 6, 2014, from https://phillymotu.files.wordpress.com/2013/04/the-impact-of-weather-conditions-on-capital-bikeshare-trips.pdf.
5. Getting started. (n.d.) Retrieved October 16, 2014, from https://developer.jcdecaux.com/#/opendata/vls?page=getstarted.
6. Hinton, G. *Lecture 7: Learning in recurrent networks.* Personal Collection of Geoffrey Hinton, University of Toronto, Toronto. Retrieved December 9, 2014, from http://www.cs.toronto.edu/~bonner/courses/2014s/csc321/lectures/lec7.pdf.
7. Open Data Callenge. (n.d.) Retrieved October 16, 2014, from http://www.bayareabikeshare.com/datachallenge.