Predicting the Duration of a Bike Rental

Jessica Lauzon, Jayant Mukhopadhaya
Dept. of Aeronautics and Astronautics
{jlauzon, jmukho}@stanford.edu

Abstract

By predicting trip duration based on a variety of factors, we hope to inform the future installation and expansion of bike sharing programs. Towards that effort, we applied machine learning algorithms on a dataset from an SF Bike Sharing Project. We show preliminary results from a linear regression model and a deep neural network model. The linear regression model predicted bike trip durations within 10 seconds, while the neural net was constructed as a classification problem, and performed with around 40% accuracy. We then discuss how each model could be further improved.

I. Introduction

Cycling is a cheap and environmentally friendly mode of transportation. Bike sharing projects aim to provide this convenience by renting bikes in various, easily accessible locations. These locations have limited numbers of bikes and limited numbers of docking stations. Using data on bike sharing usage in SF we hope to inform the future installation and expansion of the bike sharing programs. The dataset used was found on Kaggle.com from an SF Bike Share Project, which contains two years of bike-sharing and weather data ending in the year 2015 [1]. This dataset is not the only bike sharing project on Kaggle. There is a dataset from 2014 that was uploaded with the specific goal of predicting the number of bikes rented during each hour [2]. Another group of researchers aimed to tackle the problem of balancing bike sharing systems - a balanced system has enough available bikes and empty slots at each station every time a user visits [3]. There is clearly a need for developing optimized bike sharing systems for urban areas where cycling is becoming popular.

With the dataset from Kaggle, machine learning algorithms were applied in order to predict the duration of a bike rental. First, a linear regression model was used for prediction. Before implementation, the given data needed preprocessing in order to be useful. Section II covers the assumptions used in order to create usable feature sets from the raw data. Next, a neural network was used to predict the duration. The methodology of the two models are covered in Section III, and the results are shown in Section IV. This project was completed as part of the Fall 2017 course CS229: Machine Learning at Stanford University.

II. Dataset and Features

The first step was to remove outliers from the dataset. Since the bike share charges extra money for rentals that take longer than 1 hour, we decided to limit our dataset to bike rentals that were between 5 min to 60 min. About 95% of the dataset lies in this range. To ensure a linear relationship between the features and the quantity of interest, many features needed to be preprocessed. This meant classifying some features into "buckets," combining features such as station locations into usable information, and so on. The features used were:

Start Station: Using the Latitude and Longitude coordinates of each station, the mean Euclidean distance from all other stations was used as a station identifier.

Start Time: Start time data was classified into five time buckets based on our own knowledge of travel times: night, early morning, rush hour, midday, and evening.

Day of the Week: There didn't seem to be much variation in trips within the week. The big difference was seen between weekday trips and weekend trips. This led us to create a binary feature: 0 for a weekday and 1 for a weekend.

Weather event: The dataset specified the overall weather for the day as: sunny (0), foggy (1), rainy (2), rainy and foggy (3), and thunderstorming (4). Numbers were chosen based on what would reduce someone's likelihood of travelling by bike.

Temperature: The temperatures didn't relate linearly to trip duration. The ideal bicycle riding temperature was thought to be 60-80 F. Temperatures higher and lower would deter cyclists. Since there were no days with a mean temp > 80F we used a binary classifier to sort the mean temperature into two categories, between 60-80F and below 60F.

Other weather parameters: There were no assumptions with features like visibility, precipitation, cloud cover, and wind. Mean values for the corresponding day was used for each of these.

III. Methods

III.a Linear Regression Methodology

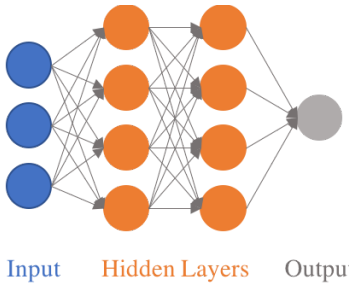
The preprocessing steps resulted in the projection of the data into a lower dimensional space. This resulted in a large number of identical examples that needed to be eliminated from the dataset. The closed form solution of linear regression was used to calculate the weights:

$$\theta = (X^T X)^{-1} X^T \vec{y}$$

The dataset was reduced by about 80% when uniqueness was enforced. We still have about 60,000 examples which we believe to be a sufficient dataset size. But this reduction in usable data, coupled with the linear regression results, spurred us to try implementing a neural network to improve our model.

III.b Neural Network Methodology

Given the nonlinear relationship between the inputs (features) and outputs (duration), using a neural network to predict duration seemed like a good choice. The neural network was implemented in both TensorFlow and MATLAB. While the results in the next section are from the MATLAB implementation, it is important to know the evolution of the neural network from TensorFlow to MATLAB for the final implementation. In general, a neural network consists of an input layer, which contains the features, hidden layers that each contain a certain number of nodes, and an output layer that contains the predictions. The training takes place on the weights, which are contained in the hidden layers. The transition from one layer to the next is done through an activation function. The activation functions, number of hidden layers and number of nodes are parts of the architecture of a neural network and can be tuned to achieve better performance. An example of a neural network with two hidden layers is shown below.



The TensorFlow neural network structure started as a regression problem, but quickly changed to a classification problem for debugging purposes. The change required the "bucketing" of the duration values into eleven 5-minute intervals so the model predicts a bucket numbered from 0 to 10. The TensorFlow model had an architecture of 4 hidden layers, with hidden units of 500, 400, 300, and 100, respectively. The activation function was a rectified linear unit (ReLU) for each layer except for the output layer which was softmax, and the loss was calculated using cross entropy.

In MATLAB, a simple architecture of 1 hidden layer with 300 hidden units was implemented. This model was also structured as a classification problem. The output activation function was softmax, and the loss was calculated using cross entropy.

For both models, the feature data was split into three proportions: 80% for training, 10% for a development set, and 10% for a test set.

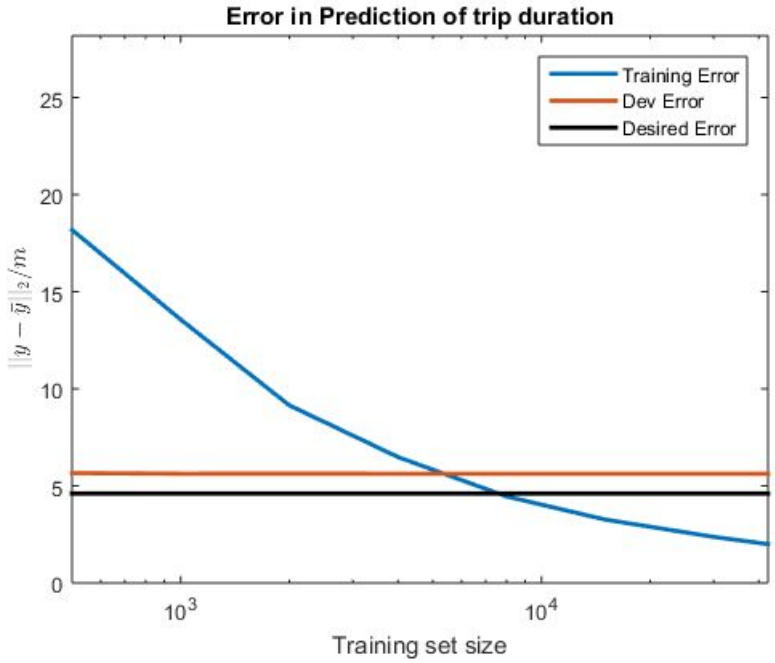
IV. Results

IV.a Linear Regression Results

The aim was to reduce our test error below 1% of the average trip duration, which in this case is about 5 seconds. The error is calculated as the L2 norm of the difference between the true duration, and the predicted duration.

$$e = \frac{\|y - \bar{y}\|_2}{m}$$

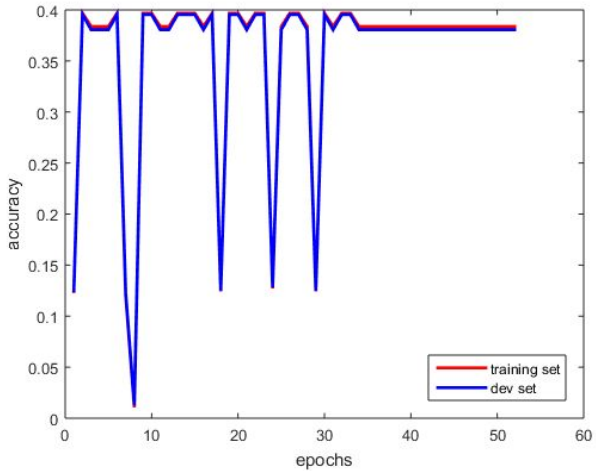
To perform some error analysis and to evaluate the model, the regression was performed using various training set sizes. In the plot below, we see the training set error reduce with larger training sets. The dev error did not seem to improve with training set size. This pointed to overfitting of data to the training set. In an attempt to reduce overfitting, regularization was implemented but this did not improve dev set performance. The final test error was 8 seconds, which is higher than the desired error. This was one of the reasons we implemented a neural network to attempt this prediction.

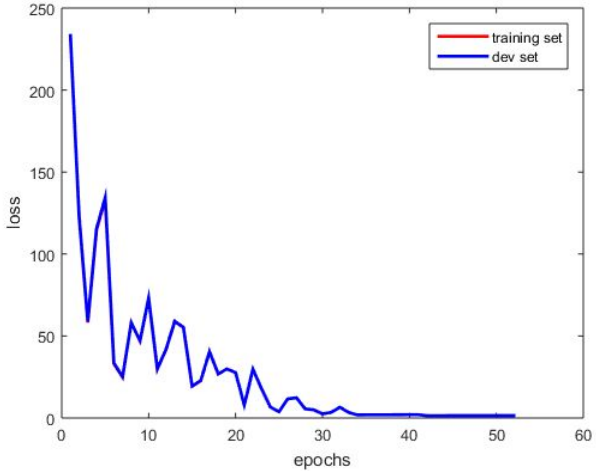


IV.b Neural Network Results

The TensorFlow neural network model achieved very limited success. While the cross entropy was consistently decreasing, the accuracy stayed about constant around 40%. Increasing the size of the neural network by increasing the number of hidden units showed little improvement, although the size was limited by the amount of computing power available. This was one of the main reasons that motivated the switch from using TensorFlow to MATLAB.

However, similar results were found by using MATLAB. The results were showing that the cross entropy loss was decreasing, but the accuracy seemed to have an upper bound. While working in MATLAB, it was found that the neural network was predicting the same duration interval for every training example. This seemed like the model was underfitting, so several steps were taken to combat this. First, a large percentage of the training data had a single truth value (about 50%), so undersampling of the over-represented class was tested. To undersample, we deleted rows of data for which the truth value was of the over-represented class until the percentage of that class out of all the data was about 35%. Secondly, we tried to increase the number of hidden units in order to capture more details of the dataset. Lastly, we increased the number of epochs. Unfortunately, the neural network model continued to suffer from underfitting. The plots below show the results of the final neural network model.





V. Conclusion

The results for both models show that further improvement is needed. In general, the dataset should be further explored to find trends within the data. For example, the vast majority of the bike rentals were under an hour, which makes sense because the Bike Share Project charges fees after an hour. Using these trends to our advantage requires understanding of the data and may improve model performance.

For linear regression, the model seemed to work well for the training set but the high development error and test error indicates a need for fine-tuning. The linearization of parameters should be revisited and perhaps more features should be included in the future. Weighted linear regression is another avenue that can be explored.

The neural network model was subject to underfitting, and steps were taken to combat this. Despite these steps, underfitting was not resolved. Further exploration of those strategies may be a good starting point for future work.

VI. Contributions

Most components of the project were worked on simultaneously by both group members. The final implementations of the linear regression and neural network in MATLAB were written by Jayant Mukhopadhaya while the TensorFlow neural network was created by Jessica Lauzon. The poster and final write-up were contributed to in equal parts by both group members.

VII. References

[1] https://www.kaggle.com/benhammer/sf-bay-area-bike-share. December 13, 2017. Note: Data was also taken from here.

[2] https://www.kaggle.com/c/bike-sharing-demand/data. December 13, 2017.

[3] Luca Di Gaspero, Andrea Rendl, and Tommaso Urli. “A Hybrid ACO+CP for Balancing Bicycle Sharing Systems” Hybrid Metaheuristics: 8th International Workshop, Italy, 2013.