FINAL REPORT
DATA MINING, SPRING 2017
UNIVERSITY OF UTAH

Ride Hailing Supply and Demand Forecasting using Didi-Tech Dataset

Kimberly Williamson, Gopal Menon

April 16, 2017

1 PROBLEM AND MOTIVATION

Didi Chuxing is the leading ride hailing company in China and processes over 11 million trips, plans over 9 billion routes and collects over 50TB of data per day. They organized a worldwide algorithm challenge in the year 2016 [**?**] for forecasting ride supply and demand. We used the 2016 Didi algorithm competition dataset to try and forecast taxi trip supply and demand for any given date, time, and location using regression models covered in the Data Mining 2017 Spring semester at the School of Computing of the University of Utah.

2 RIDE HAILING DATA

2.1 DATA FORMAT CONVERSION

In order to run the regression models, the categorical values in the Didi algorithm competition dataset needed to be converted into a regression friendly format. Depending on the type of categorical value, the new values are lists that consist of 0 values when the category is not present in an order and 1 or a count when the category is present in the order.

2.2 DATA FORMAT DETAILS

We have converted the data into a format that can be used by the regression models. The input to the regression consists of:

- An order key, comprised of the categorical values for start district, destination district and order time
- An order value consisting of
  - Traffic and points of interest for the start and destination districts
  - Weather at the time of the order
  - The value that needs to be predicted, which is the median order price or the number of orders

With the input data created, we have experimented with multiple regression models.

2.3 Data Size

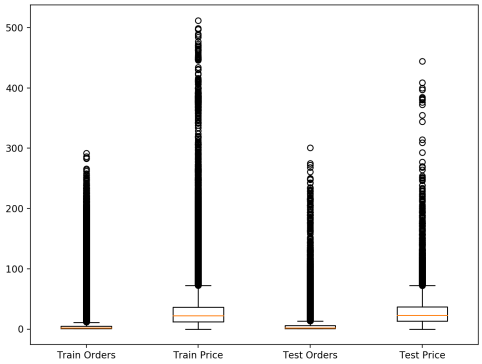

Figure 1: Box and whisker data plots.

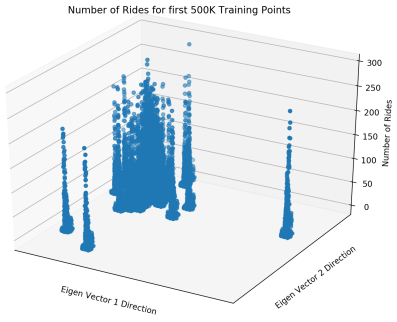

Figure 2: Number of rides scatter plot.

We have 1.3 million rows in our training data consisting of ride hailing orders. When we took into consideration, only those orders where the start and end districts have data for traffic and places of interest, the number of training rows were reduced to 921,000.

3 Key Idea

The start and end ride locations, driver id, passenger id and points of interest have all been anonymized in the ride hailing data. However we were hopeful that using the regression techniques, that would be able to extract patterns from the data, we could make accurate predictions. We were expecting some sort of correlation between the start and end locations, weather, time, points of interest and the values to be predicted - the number of rides and the price.

Although we did not expect the number of rides and the price to have a linear relationship with the ride features, we planned to start with linear regression and then move on to non-linear prediction models for better accuracy. The Python library scikit-Learn [**?**] was used for running regressions.

4 Prediction Accuracy

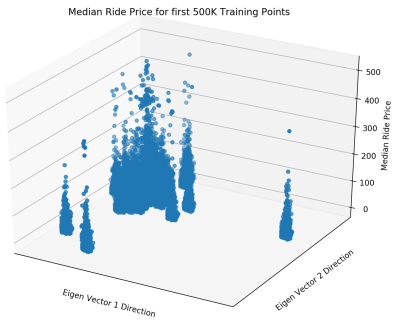

Figure 3: Order Price Scatter Plot.

The prediction accuracy results for number of rides is shown in table **??**. The regression was run without the data for traffic, weather and points of interest as results were more accurate without this data. The least error was for linear regression using stochastic gradient descent. The results from Guassian Kernel Regression were not very accurate either. The reason was probably that since the start and end locations had been anonymized, we did not have a way to compute a distance between two vectors. If we had this data, the kernel regression results would have been more accurate. We were hoping to get better results using non-linear regression, but the results were surprisingly worse than those for linear regression. We used the

Table 4.1: Regression Results for predicting number of rides

Regression Type	Mean Squared Error
Linear using Stochastic Gradient Descent	245.50
Gradient Boosting using features based on top 10 eigen vectors	285.97
Ridge Regression using features based on top 10 eigen vectors	293.83
Lasso Regression using features based on top 10 eigen vectors	293.83
Polynomial degree 2 regression using features based on top 10 eigen vectors	2.34×10^{53}
Polynomial degree 3 regression using features based on top 10 eigen vectors	3.53×10^{73}
Polynomial degree 4 regression using features based on top 10 eigen vectors	9.31×10^{93}
Gaussian Kernel Regression	376.90

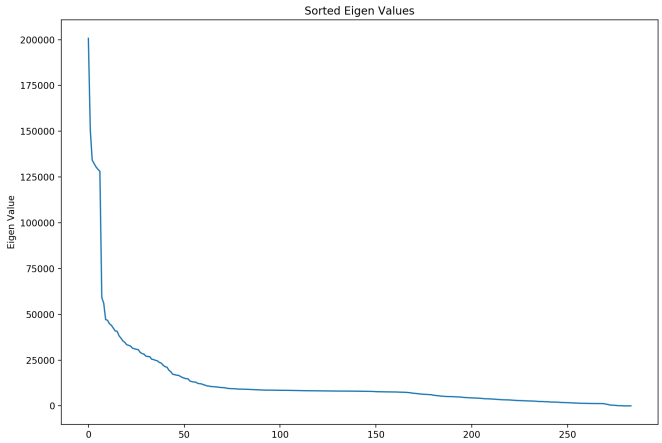

Figure 4: Sorted eigen values.

first ten eigen vectors for dimensionality reduction as we found that most of the information was encapsulated in these. This is shown in figure **??**.

5 Patterns in Ride Hailing Data

Since we were not getting good results, we decided to see what the data looks like and plotted box and whisker plots for the data. These plots are shown in figure **??** and seem to suggest that there are a lot of outliers. The outliers are usually computed using the inter-quartile distance between the first and third quartiles and using that figure to include elements that are up to 1.5 times the inter-quartile distance on either side. Elements outside these limits are marked as outliers. However it is doubtful that these are outliers since it is actual data and the reported number of outliers is large.

We next looked at clustering methods to determine and filter out outliers. After experiment-

Figure 5: Number of rides scatter plot for start district and week day.

ing with a couple of methods with no improvement in the mean squared error, we confirmed our previous theory that the identified outliers are most likely not true outliers.

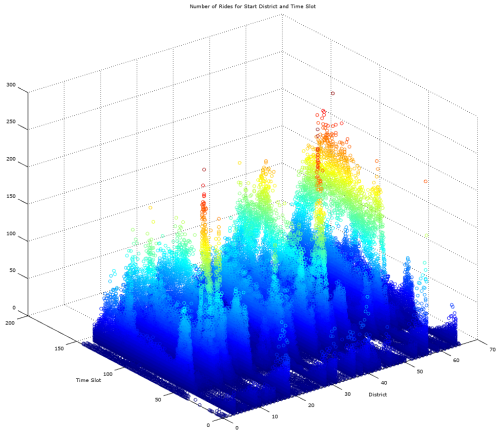

Figure 6: Number of rides scatter plot for start district and time slot.

Next we plotted the number of rides and the order price for each data point after it had been projected on the first two eigen vectors and multiplied by the corresponding eigen values. These are shown in figures **??** and **??**. These plots show that for the same coordinate along the first two eigen vectors, there are many values for the number of rides and price. This is an indication of the complexity of the data and the difficulty in using it to make predictions. The number of rides plot for the start district and day of week in figure **??** also shows a similar complexity that makes prediction difficult. Figure **??** is another example of data complexity.

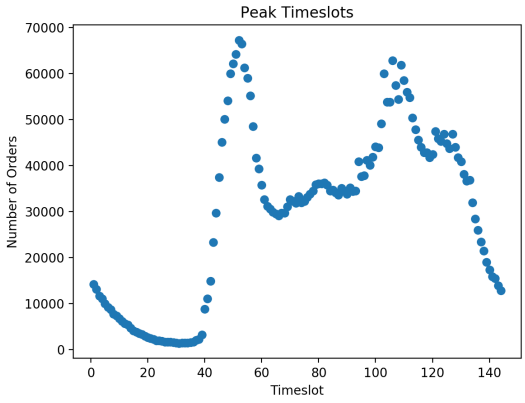

Figure 7: Number of rides scatter plot for a timeslot (Monday-Friday).

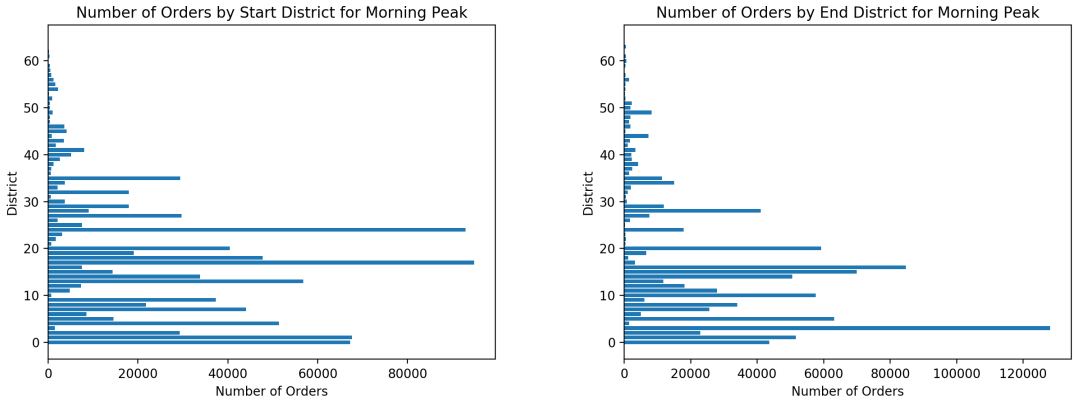

Figure 8: Number of rides scatter plot for a district during peak time period (Monday-Friday).

We also looked at patterns that could be attributed to the passengers taking taxis to and from work. Using a Monday through Friday work week, we first found the peak timeslots shown in figure **??**. Then using the peak timeslots, we plotted the number of orders per start district and the number of orders per end district for the first peak shown in figure **??**. While there does appear to be some pattern that could lead to deducing residential and commercial districts, the pattern is not strong enough to reduce the complexity of the dataset.
The dataset can be separated into different collections, the DateTime collection, the POI collection, the Weather collection, and the Traffic collection. Each collection in the dataset has been expanded to allow for the categorization of the variables. We ran dimension reduction on each collection separately to obtain a two-dimension subspace using Singular Value Decomposition(SVD). The objective of the SVD process was to visualize the collections to determine if the collection will fit to a linear line. These visualizations further proved that the dataset is not of a linear fit.

6 CONCLUSION

It looks like traditional regression methods do not work for complicated situations where we need to model human behavior. Deep learning techniques have been shown to give better results where traditional methods have failed or do not result in the desired level of accuracy. This is possibly a problem that is better suited to using deep learning.

REFERENCES

[1] "Algorithm Competition." *Algorithm Competition*. N.p., n.d. Web. 28 Jan. 2017.

[2] "Scikit-learn." *Scikit-learn: Machine Learning in Python - Scikit-learn 0.18.1 Documentation*. N.p., n.d. Web. 19 Mar. 2017.