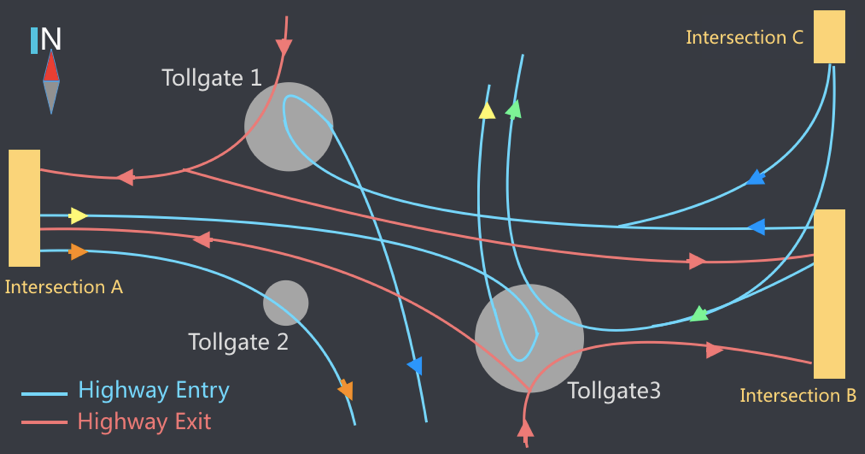
**Takeaway from KDD Cup 2017 Competition**

Hou Jue, June 5 2017

Hi there, this is a documentation of the journey we took as a team to participate in this year’s KDD Cup competition --- ideas, takeaways and thoughts. Please help to make it more complete by pointing out any part that needs supplement or clarification. Feel free to share your comments, discussions or questions with everyone by adding to this documentation or through our email chain.

1. **Introduction**

This competition first drew our attention because of the interesting nature of the problem --- traffic prediction. We are tasked with making predictions for 1) 20-minute average travel time from point A to point B on highway during peak hours; 2) 20-minute volume at highway tollgates during peak hours. The problem involves both time series patterns and other non-temporal factors, for example, weather.



The training data in the first round given include the 3-month highway travel time data from July 19 to Oct 17 in 2016, 1 month volume data from Sep 19 to Oct 17, weather data from July 19 to Oct 17 as well as the two-hour traffic data before the peak hours in the test period. The test period in the first round is the week from Oct 18 to Oct 24. In the final round, data in this period was added to the training data and test period was updated to the week from Oct 25 to Oct 31. Evaluation metric used in this competition is Mean Absolute Percentage Error (MAPE).

The problem is more complex compared to usual competitions because there are two tasks involved, namely travel time and volume, and 11 subtasks in total involving different routes and tollgates as shown on the map above. How to cross-utilize information in the different tasks and dataset and how to set up the modelling framework becomes very critical in this problem.

1. **Our Journey**

**2.1. Machine Learning Models**

The first modelling method that came to my mind was traditional machine learning, because of my familiarity of this method, its past success in this kind of tasks and the availability of a handy tool, DataRobot.

My first thought was to combine training data for all subtasks and time points in the ML model. This idea met objections from the team because 1) the patterns in different subtasks might be different; 2) the model learnt through all time points may not generalize to the peak hour. After discussion, we decided to go with the setup that separates the subtasks into individual models and includes peak hour intervals only. In hindsight, I think this is a correct starting point, but if we had more time to improve the models, it would be worthwhile to add in non-peak hours into modelling to capture the effect of preceding traffic conditions and historical patterns.

We generated the following features for the ML model: lag features in different previous intervals, lag features in aggregation form like median and variance, immediate preceding window features (features that come from the 2 hours before the peak hours), historical mean features in the same interval, day of week, holiday, weather (temperature, humidity, pressure, etc.) and traffic flow features calculated directly from the trajectories amongst many others. In total there were 80 + features in the initial rounds contributed by different team members.

The process of feature generation and performing version-control is very labor-intensive and error-prone. There were minor errors discovered in many versions due to file-merging issue and miscommunication among contributors. As a result, manual corrections and re-work were performed many times throughout the process, which is time and effort consuming. This is one part of the areas I think we can do better next time through better version control and communication mechanism.

**2.2. Issue and Mitigation in Machine Learning Model**

The training and testing of the ML models were done in DataRobot. The initial accuracy of the ML models is: 24% for time prediction and 25% for volume prediction. A particular problem that surfaced was that the performance on validation set is a lot better than that on hold-out set. A closer look at the feature importance from the model gave me a good insight on the reason behind that.

The feature importance shows an extremely high importance for most immediate recent intervals. For example, if the target interval is 9:00 to 9:20, the 8:40 to 9:00 interval has a very strong prediction power. This is intuitively correct but it is problematic because we don’t always have recent interval data available for all the peak hour intervals. If the model trained gives too much weight to the recent intervals, it is going to perform well on training and validation set where these information is available, but poorly on the test set.

This finding made me realize a mistake I made in making these lag features. I should treat the unknown information in test set as unknown information in training set as well, even if they are actually available in training set. For example, if we are predicting for 9:00 to 9:20, since we don’t know the information from 8:40 to 9:00 in test set, we should ‘pretend’ that we don’t know this in training set as well, to avoid misleading the model.

Correction of this mistake in 10+ features improved the model performance significantly (time prediction MAPE was improved from 24% to 19%) and bridged the gap between validation accuracy and test accuracy.

* 1. **Time-series Models**

After initial tryout with machine learning models, Zhiyan proposed using various time-series methods, with inspiration from this link: https://www.kaggle.com/c/walmart-recruiting-store-sales-forecasting/discussion/8125.

More specifically, he implemented SVD, stlf in combination with ets or auto.arima() on both tasks. For missing values in the training set, imputation was made using the na.kalman() function from imputeTS package. For the gaps in test set, imputation was made by iteratively adding predicted values for the gap period and adding back into the training set.

* 1. **Other methods**

In addition to machine learning and time-series prediction, Sijia also tested the idea of deep learning and k-nearest-neighbors (kNN). More specifically, LSTM and kNN were trained on the preceding 2-hour intervals before target intervals and used to predict for the target intervals.

1. **Results**

In terms of prediction result on test set given in the first round, machine learning method has a MAPE of 18.5% on time prediction and 17.2% on volume prediction. On the other hand, time-series method has a MAPE of 19.3% on time prediction and 15.1% on volume prediction. kNN has an MAPE of 16.5% on volume prediction. As a benchmark, the best models on the leaderboard for the first round have an MAPE of 17% for time prediction and 12% for volume prediction.

It is found that machine learning method has a higher accuracy in time prediction than time series method but time-series method has a higher accuracy in volume prediction than machine learning method. This is most likely due to the larger training sample size in time prediction task (around 3 months) than volume prediction task (only 1 month). As ML models need a relatively larger sample size to learn generalizable patterns and rules while time-series mainly rely on weekly, hourly and preceding interval data, their performance difference for the two tasks is understandable.

For the final test set, we submitted the following ensemble models:

* Time prediction: Ensemble of 70% ML + 30% time series
* Volume prediction: Ensemble of 80% time series + 20% ML

The above models achieved MAPE of 19.0% (1.6% from No. 1 on leaderboard) and 20.0% (8.6% from No. 1 on leaderboard) respectively.

1. **Key Takeaways**

It has been an exciting and fun 3 months, in which we have put our modeling knowledge, coding skills to test and gained a lot of valuable lessons. I would summarize the lessons learnt throughout the journey that may benefit our next competition participation into the below key points.

1. Different methods have different performances depending on the problem scenarios, as with ML and time series in this case, so be open-minded about techniques and models.
2. Good version control and collaboration method can save a lot of time and frustration.
3. Do logic check and proof read on your model or feature. If a feature turns out to be much more predictive than all the others, or if a model has much better result on validation than test, it may suggest a potential problem.
4. Treat competition as a ‘real’ project and set clear timelines. We would have benefited from regular progress checkpoints (eg. every week) to ensure all the ideas had enough time to be implemented.