Speed Prediction from Taxi Trajectory Data

Chunheng Jiang, Sean He, Jianxi Gao, and Yu Wei

Rensselaer Polytechnic Institute, {jiangc4,hex6, gaoj8,weiy4}@rpi.edu

December 10, 2018

Problem

We are given a trajectory data of DiDi Express and DiDi Premier drivers within the Second Ring Road of Xi'An City. All track points are bound to physical roads with resolution about 2-4 seconds. The problem is to predict the average speed of all vehicles running on a road segment (either north- or south-bound) at specific timestamp.

Data Preprocessing

Time Alignment

Each day contains millions records, and each record contains five columns: driver id, order id, timestamp, longitude, and latitude. Both drivers and orders are encrypted and anonymized with long strings. To reduce memory usage, we re-encode these strings with integers. It shrinks the file size for at least 50%. All timestamp entries, either in unix convenience or human readable format are aligned relative to 00:00:00, and replaced with time offset, i.e. time difference between current and the reference in terms of seconds.

Trajectory Truncation and Speed Estimation

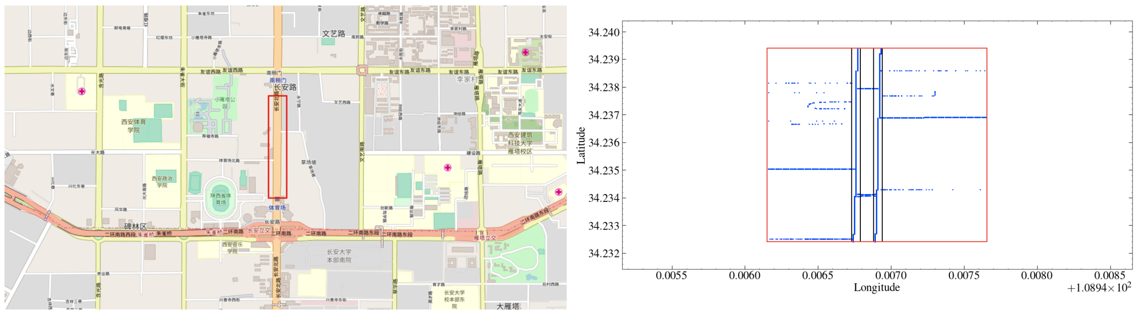

Fig. 1: Road section to be predict

The road section in question is shown in red rectangular in Fig. 1, and we manually locate the ranges of both north-bounding and south-bounding road sections in terms of longitudes and latitudes. We extract all trips that go through south-or-north bounding corridor. Then, from the truncated trajectory, we calculate the vehicle speed with the travel distance calculated from location information and the travel time. The speeds contain noisy data since the travel time of two sequential points is very short, approximately 2-4 seconds. The noise can be ignored in our computation, since the extracted data is sufficiently large. Another aspect is the time slots. The time offsets are split into evenly distributed cells, e.g. north-bounding time slots is a simple sequence $5, 305, 605, \ldots$, while south-bounding time slots are $4, 304, 604, \ldots$, with a gap of 5 minutes (300 seconds). Each estimated speed is allocated to its nearest time slot, e.g. the speed $k = 39$ of a truncated trip $a \rightarrow b$ during $t_1 = 505$ and $t_2 = 508$, since $t_2 - t_1 \ll 300$, it's nature to assume that the average driving speed at the nearest time slot $t = 605$ tends to be close to what at either t_1 or t_2.

Methods and Results

Speed prediction is a classical regression problem. Partially based on real life facts, we assume that the future traffic speed is predictable with the historical data. The dataset is built as follows. Based on the evenly distributed time slots, we extract all speed estimations in each time slot, and the average value is assigned to the corresponding time slot. We apply the same strategy to process the trajectory data of each day. Finally, we create a dataset of size $n \times d$, with $n = 284$ instance for north-bounding and $n = 279$ for south-bounding traffic flow, and $d = 59$ equals to the number of effective days (Note: Oct 30 and Oct 31, 2016 are redundant copies and thrown away). The number of instances is determined by the number of time slots, which are split following the same criterion used in the observation records.

Table 1: Optimal hyper-parameters of models. Training rmse is the average rmse of 10 fold cross validation over the training set, which accounts for 80% of the entire dataset, and testing rmse is over the remaining 20% instances. The rmse corresponds to the entire dataset without the missing entries of the learned model.

South/North	Random Forest			train rmse	test rmse	rmse
	n_estimators	max_depth				
South	200	4		2.358	6.016	4.140
North	100	4		4.195	9.915	7.036
S/N	Gradient Boosting			train rmse	test rmse	rmse
	n_estimators	max_depth	learning_rate			
South	60	3	0.184	0.084	6.750	2.950
North	240	2	0.109	0.548	9.409	4.646
S/N	SVR (RBF kernel)			train rmse	test rmse	rmse
	gamma	C	epsilon			
South	0.101	546.870	0.120	0.119	6.283	3.122
North	0.101	935.922	0.419	0.762	10.250	4.565

Three models: gradient boosting, random forest and support vector regressor[1] are trained with the z–score normalized dataset, where dataset is split into training and testing set following 80:20 rule. Each has its own pros and cons. The former two are ensemble models, whose advantages include ease to implement and efficient to train, but may be overfitting, which can be counteracted with cross-validation. SVM models are hard to tune and expensive to train [1,2]. But, they have a good generalization ability even with small data set.

[1] https://scikit-learn.org/

Also, each has many hyper-parameters to tune. Therefore, we employed the hyper-optimization method [3] implemented in hyperopt[2]. The optimal hyper-parameters for each model are reported in **Table** 1. With the selected model, predictions are made over the missing time periods and Fig. 2 shows the snapshots of the predictions. More detailed information can be found in the attached Jupyter Notebook.

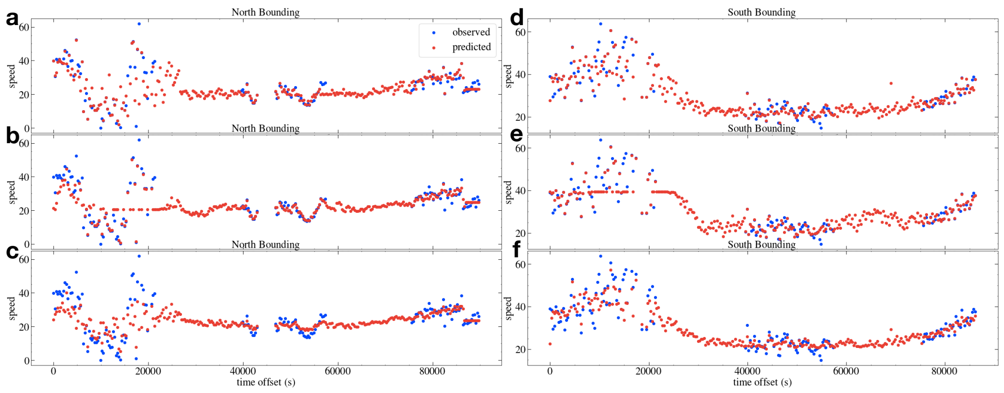

Fig. 2: **Traffic speed prediction. a-c,** Speed prediction on northbound, and **d-f,** on southbound corridor. **a,d** gradient boosting, **b,e** support vector, and **c,f** random forest regressor.

Discussions

The trajectory date is sparse, especially for the given road sections. All our computations is simplified, and the created dataset does not consider other factors that exercise some extend impacts on the traffic speed, locally or globally [4], e.g. the drivers' behaviors (the size of unique drivers is found in million, many drivers may be given multiple encoding strings), the weather (snowing slows down the entire system), surrounding business (determines the demand and supply), and the entering and exiting ramps (merge/branch the traffic flow, and form bottlenecks, may cause traffic collisions). More advanced techniques can be applied, including HMM [2], MCMC [4] and deep neural networks [5].

References

1. Bishop, C. M. *Pattern Recognition and Machine Learning* (Springer, 2006).

2. Friedman, J., Hastie, T. & Tibshirani, R. *The elements of statistical learning*, vol. 1 (Springer series in statistics New York, NY, USA:, 2001).

3. Snoek, J., Larochelle, H. & Adams, R. P. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, 2951–2959 (2012).

4. Woodard, D. *et al.* Predicting travel time reliability using mobile phone gps data. *Transportation Research Part C: Emerging Technologies* **75**, 30–44 (2017).

5. Goodfellow, I., Bengio, Y., Courville, A. & Bengio, Y. *Deep learning*, vol. 1 (MIT press Cambridge, 2016).

[2]https://github.com/hyperopt/