TRANSFOR19 Transportation prediction competition

Transportation Research Board Annual Meeting 2019

Melvin Wong, PhD Student (melvin.wong@ryerson.ca)

Laboratory of Innovations in Transport, Dept. Of Civil Engineering, Ryerson University

Github project page: https://github.com/mwong009/transp-trb-ft/

Problem definition

From a set of GPS coordinates of location of drivers and passengers using the DiDi ride hailing app, can we predict the traffic speed along a road segment? **Challenge**: The GPS traces along the road segment is removed during the forecasting time. Speed has to be discriminated between northbound trips and southbound trips.

Summary

- Full Python implementation of machine learning using open source Theano ML libraries.
- 32x32 grid segmented GPS traces, assuming that as traffic density
- 20 layer supervised **ResNet** based neural network for modelling derived speed (N/S) of traffic
- Used 2016-11-24 to validate our model. Best validation RMSE: 4.32
- Explanation and iPython Notebook here: Part 1: data processing, Part 2: data preparation, Part 3: model training

Novelty

- Fast and efficient machine learning libraries, GPU accelerated tensor based computation allows for fast processing of very large datasets
- Large capacity and number of parameters used serves as better predictors for a regression model
- Captures variances in noisy data

Outcomes

- We used the zonal density of the traffic system across the city to predict traffic speed in the study area
- A single model can be used for estimating Northbound and Southbound speed simultaneously

---

Required tools and preliminary steps

The procedure is run entirely on a Python 3.7 platform running Ubuntu 18.04. Theano Machine Learning Python libraries was used to train the model. CUDA 10.0 GPU tensor libraries was used to accelerate the learning process (up to 13x from an i7 CPU). GPU acceleration is optional, but is extremely efficient. Several python packages are required. To install, use `pip` on the `requirements.txt` file. Installation guide is on the iPython Notebook.

Data Analysis

The general outline of the procedure is follows:

- Import the GPS datafiles from the DiDi database
- Process the GPS datafiles and calculate the speed and density of the study zone
- Segment the study zone into 1024 zones
- Construct a working datatable by grouping observations into 5 min intervals
- Divide the datatable into 3 parts for Training, Validation and Testing

Import and extract GPS data

The layout of the project library are as follows

```
/root_folder
 |- /october
 |    |- 20161001
 |    |- ...
 |- /november
 |    |- 20161001
 |    |- ...
 |- /december
 |    |- 20161001
 |- /transptrb # ML modules
 |- /output # prediction outputs
 |- *.ipynb # jupyter notebook files
 |- README.md
```

Each month subfolder contains the `*.tar.gz` dataset downloaded from the DiDi website (except december). `transptrb` is the python module containing the class structure and definitions for the machine learning. Iterate through the entire root folder and create a list of filenames that contain the `*.csv` files. To speed up reading of the `*.csv` files inside a compressed folder, we use a stream based reading method `io.BytesIO` in Python.

Segmentation of the data into zones

From the datasets, we can obtain the boundaries of the study area by finding the minmax of the coordinate range. The range are: latitude (34.2053, 34.280221) longitude (108.91189, 108.99861). Using these ranges we divide the area into 32x32 segments. Each segment is grouped by 5 minute interval and unique Driver_ID-Order_ID pair.

Example data processing output:

```
60: Extraction of file <TarInfo 'xian/gps_20161129' at 0x7f024755be58> time elapsed: 66.85 minutes
Date range processed: 2016-11-30 00:00:00+08:00 to 2016-12-01 00:00:00+08:00
61: Extraction of file <TarInfo 'xian/gps_20161130' at 0x7f024755bf20> time elapsed: 67.93 minutes
Date range processed: 2016-12-01 00:00:00+08:00 to 2016-12-02 00:00:00+08:00
62: Extraction of file <TarInfo 'gps_20161201' at 0x7f02149dd048> time elapsed: 68.92 minutes
```

To find trips that are taken northbound or southbound, we use the latitude delta between each observation. If the observations have a majority positive latitude delta, then all observations for that particular Driver_ID-Order_ID pair are northbound trips and vice versa for southbound trips. We removed time periods between 12am and 6am because there were too little trips in that time frame.

Preparing train, validation and test datasets

We create 3 exclusive datasets sliced by timestamps: For training dataset, we used October, November and December data (EXCEPT November 24 6am-10:55am, 4pm to 8:55pm). For validation we used November 24 6am-10:55am, 4pm to 8:55pm to be as close as possible December 1st. Data variables are normalized to 0 mean and unit variance e.g.

```
train_x = ((train_x - datatable.mean(axis=0))/datatable.std(axis=0))
```

Machine learning

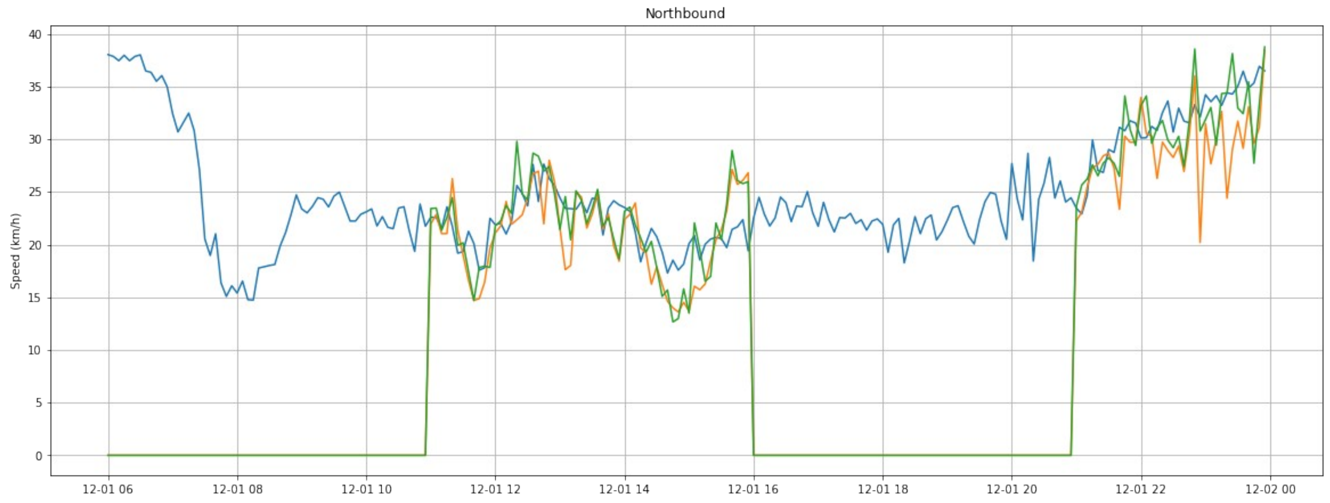
The code for the machine learning algorithm can be found in the iPython Notebook file `training.ipynb`. We used the following hyperparameters for model training:

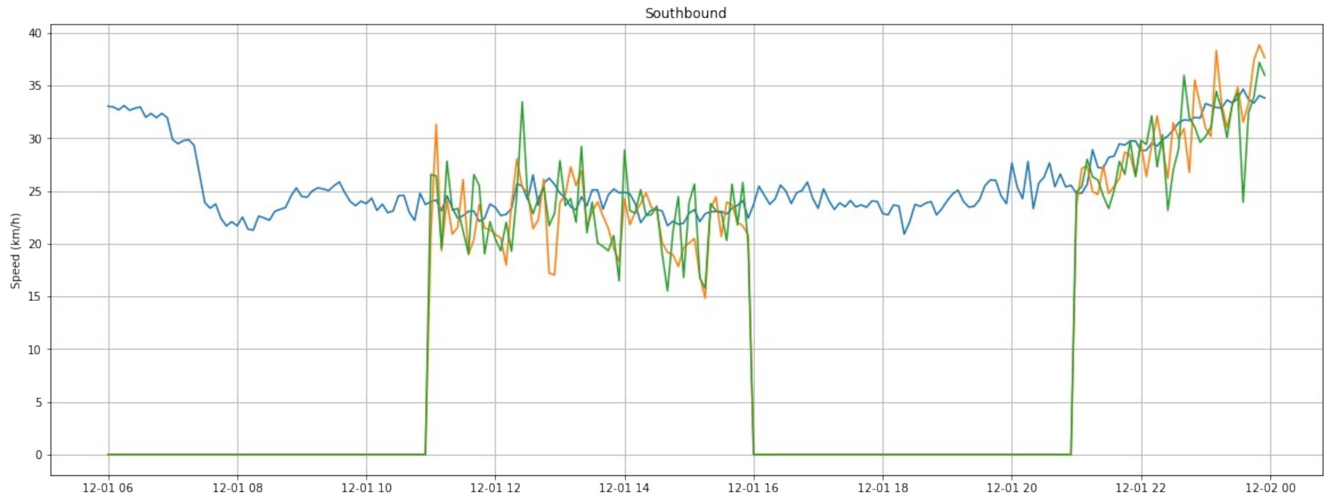
- Batch size: 4
- Model: Single Multi-layer regression model with residual connections (ResNet)

- Layers: 20 (19 x 64 hidden units + 1 x 128 hidden units)
- Activation: Self-normalizing exponential linear rectifier
- Optimizer: Nadam (learning rate=2e-4) with early stopping

Sample output

Self evaluation: predicted speed is blue, given speed is orange and processed speed is green.





Results and discussions

The prediction outputs are available in the output folder. Only the target time periods are predicted – 6am to 11am and 4pm to 9pm.

Our machine learning tool predicts traffic speed accurately with 4.32 RMSE. Note that this is only the performance on our validation dataset and may not reflect the actual test set. One full training process takes about 3 to 4 minutes (16 epochs).