Taxi Trip Time Prediction

1st place solution write up

2015-Jul-09

Introduction

This is the write up of the 1st place solution of the Taxi Trip Time Prediction competition on Kaggle. The goal of the competition was to build a predictive framework that is able to predict the total travelling time of 320 taxi rides based on their (initial) partial trajectories. The framework can then be used to improve the efficiency of electronic taxis dispatching systems in Porto.

My approach

One of the first observations I made was that the position and also the remaining driving time depends very strongly on the cut-off position of the trip. For certain positions it was very clear where the taxi trip is heading to (for example T15 in Fig1), and making a precise prediction is feasible. For other trips only some parts of the city could be excluded. But looking at all trips in the training set it is possible to identify these parts. This can be done by collecting all trips which were close to the cut-off position of the test trip. Figure 1 shows the start (green) and end (red) position of the collected trips for four trips in the test set. The blue dot shows the cut-off position of the corresponding test trip. We can see that for the trips T25, T50, and T59 the latitude coordinate has a strong influence on the distribution of the end points and thus also for the remaining driving time (see Fig. 1 on the right).

So I decided to train an individual model for every trip in the test dataset and to fall back to the more general ones, if the size of training data set for these models is too low. The following models were trained:

- Base model: Based on a dataset, where the features were extracted from all the tracks in the training set, and longer tracks were sampled more frequently than shorter ones.
- General expert models for short trips (e.g. 1, 2 or 3 positions are known).
- Expert models for each test trip (e.g. trained on tracks which cross the test trip at the cut-off position).

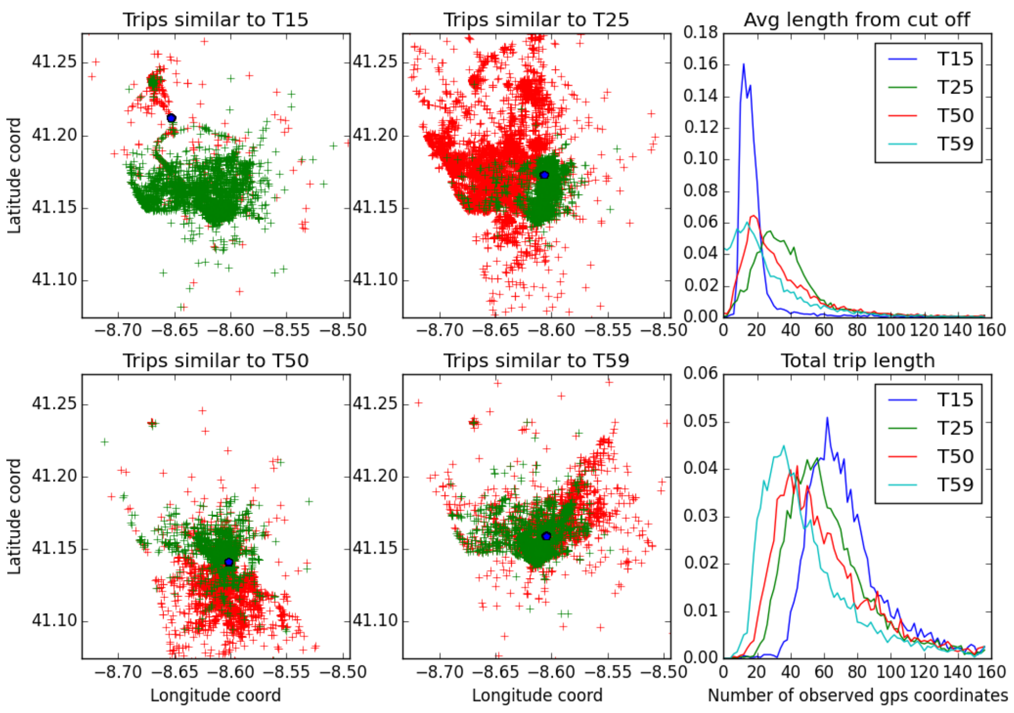

Figure 1: Start (green) and end (red) positions of all tracks in the training set, where the track was very close to the cut-off position (blue) of the test trip. On the left side the histogram of the total trip length and the remaining trip points for the test trip on the left is shown.

Pre-processing and feature engineering

The training set contained a lot of very short trips as can be seen in the left plot of Figure 2, which shows a histogram of the total trip length. The high fraction of trips less than 4 does not follow the general type of the distribution. I therefore excluded them from the analysis. Another type of error I observed was misread GPS coordinates, and I excluded them by cutting of very long distance trips. The threshold was set to 99 percent trip length for the end position predictions, and 99.9 percent for the travel time prediction, respectively.

The remaining trips where then used for the generation of the model specific training sets. Only the training set of the base model contained all trips. For this dataset every trip was randomly cut-off in between. Because the test set was collected at specific time points, longer trips are more likely included. Fig. 2 shows that by increasing the sampling frequency for longer tracks linearly with trip length, the frequency of short trips can be reduced. The resulting distribution is closer to the one of the test set (black dotted line).

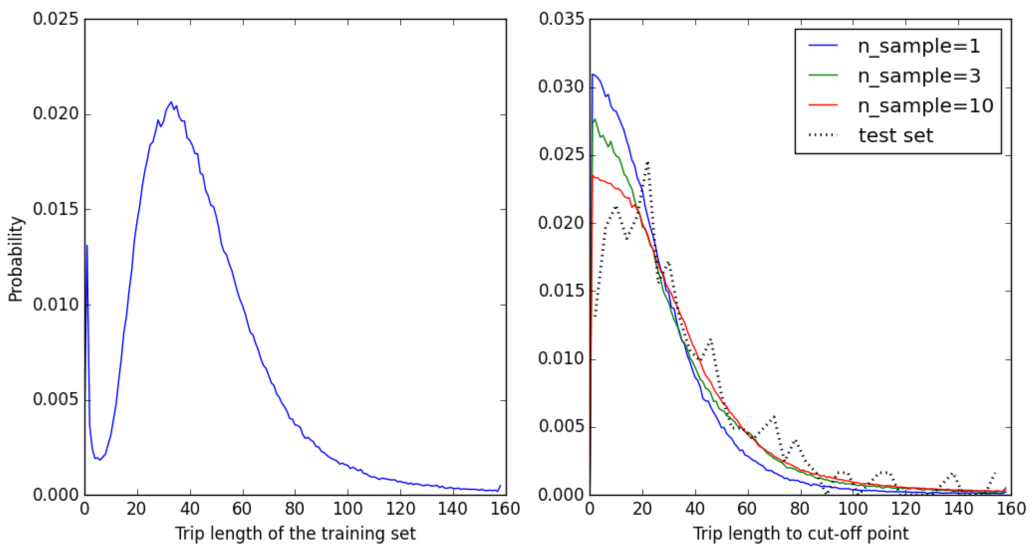

Figure 2: Left plot shows the distribution of trip lengths. Right plot shows the trip length of the training set after sampling. More frequent sampling of longer trips decreases the number of short trips in the training set and the resulting distribution more similar to the test set distribution (black dotted line).

The training set contained the following features:

- working day (Mon-Sun)
- the hour the trip started
- trip length
- latitude coordinate start point
- longitude coordinate start point
- latitude coordinate cut-off point
- longitude coordinate cut-off point
- distance from start point to city center
- heading towards the city center (from the start point)
- distance from city center to cut-off point
- heading from the city center (towards the cut-off point)
- net distance to cut-off point
- median velocity of the car along the track
- velocity of the car at the cut-off point
- heading of the car at the cut-off point

Interestingly, most of the meta data seemed to have little to no predictive power, so in the end I only used the time stamp.

Training

All models are trained using a 5 fold cross-validation technique. I used RandomForestRegressor (RFR) and GrandientBoostingRegressor (GBR) from sklearn with default settings except the number of trees, which was set to 200.

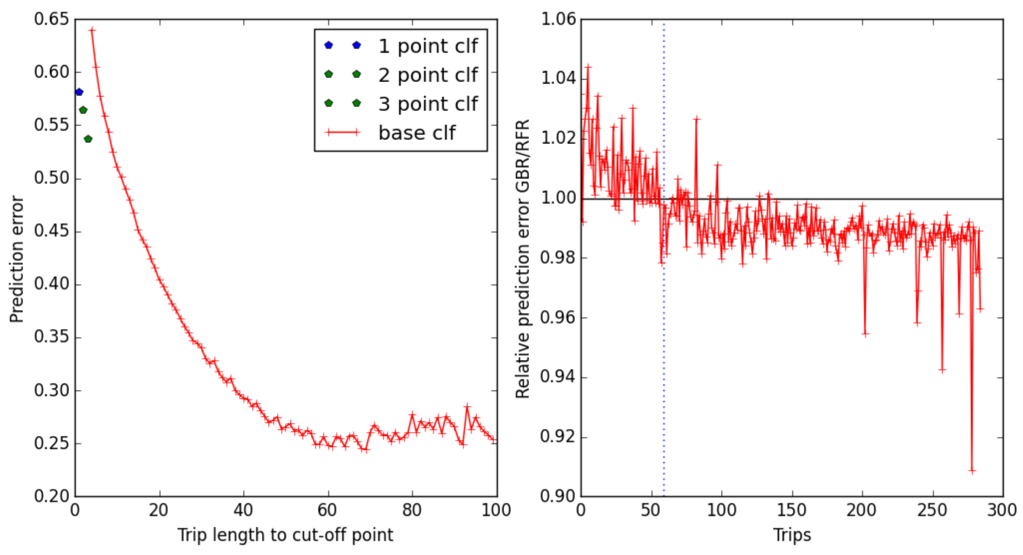

Figure 3: Left plot shows the cross validated prediction error of the base model against trip length. The predictions of the short trip experts (blue/green) are considerably lower. The right plot shows a comparison between RandomForestRegressor and the GradientBoostingRegressor against increasing training set size. For training set size above 4000 (blue line) the GBR is slightly better than the RFR.

Figure 3 shows on the left the CV prediction error of the RandomForestRegressor plotted against the trip length of the base model (red). The error decreases fast to values below 0.3 because of the logarithmic transformation of the predicted travel times. So it was crucial in this competition to optimize the prediction error especially for short trips. The trained experts for short trips (one (blue), two or three (green) positions are known) clearly outperform the base model.

The expert models for the cut-off position were trained using both RandomForestRegressor and GradientboostingRegressor. The training set size varied from a few trips up to 100000. Models were trained only for those test trips where the training set size was above 1000. Interestingly, with increasing training set size the CV prediction error of the GradientBoostingRegressor was slightly better than the one of the RandomForestRegressor as the right plot in Fig. 3 shows.

Model averaging

In total I trained over 300 models, so I had to select which models to use in the final submission. Since you are allowed to select two final submissions for evaluation, I decide to use the expert models based on the cut-off position in only one of it.

Submission 1 was a blend of the base model with the expert models for short trips. Fig 3 shows that the CV error of the short trip expert model based on 3 positions is lower than the base model up to the trip length of 9. Therefore I decided to use also the short trip models for trips up to a length to 10 and 15, respectively. I calculated the average of all 4 models with equal weight and tested the submission on the public leaderboard. The blended submission with a threshold of 15 performed a little better and I used this value for the final submission. I skipped a more thorough analysis via a solid CV because of time reasons.

Submission 2 was based on submission 1. In addition the predictions were replaced by the ones of the trip expert models for all trips with sufficient training set size. I tested two thresholds (1000 and 2000) on the public leaderboard, and used 1000 for submission 2 because of the slightly better score. Again, I skipped a more thorough analysis via a solid CV because of time reasons.

Interestingly, submission 1 scored better in the private leaderboard than submission 2 (0.5092 vs. 0.5045), whereas for the public leaderboard it was the other way around (0.5253 vs. 0.5354).

Software and hardware

I used python with its popular scientific modules, in particular numpy for feature processing, pandas for data handling and sklearn for model training. Preprocessing and training of the bigger models was quit memory intensive. I used a 12-core server with 24GByte of ram, which helped a lot to reduce the training times.

Additional comments and observations

Here are a few things I tried, with varying levels of success:

- I tried to identify some key points in the map (e.g. crossings of main routes) and to extract some extra features based on the last visited key point of the car. But it only gave very little improvement.
- I also tried neural networks early on but the results were not promising at that time. So I stopped using them for this competition, although I think that recurrent neural networks fit optimal for this task.
- The CV prediction error of the models was lower than the one I achieved at the public leaderboard at the time prediction task whereas at the position prediction task it was much worse.

Conclusion

I had a lot of fun working on this problem and I learned a lot. The forum and the starter scripts were very helpful. Many thanks. I would also like to thank the Software Competence Center Hagenberg for letting me use their computational infrastructure for this competition.