CS249 Course Project Summary: Taxi Trajectory Prediction (ECML/PKDD 15)

Yunsheng Bai, Tzu-Wei Chuang, Junheng Hao, and Xue Sun

University of California, Los Angeles

Abstract. The recent widespread adoption of the electronic dispatch systems has allowed many new opportunities in the taxi market. While the system makes it easy to see where a taxi has been, it is still hard to know where the taxi is heading to. Therefore the goal of this project is to predict the final destination of a taxi in terms of the latitude and the longitude to improve the efficiency of the taxi electronic dispatch systems. We introduced a similarity based model as well as a Neural Network approach that showed competitive results as the first place of the Kaggle Competition.

Keywords: Neural Network, Prediction, Pre-engineering.

1 Introduction

The motivation behind this challenge is that customers are using an unicast-based one to one messages or calling in to a dispatch system to schedule a taxi. While the implementation of electronic dispatch systems can be useful, that is having mobile data terminals installed in vehicles to provide information on GPS localization and taximeter state to see where the taxi has been, it is still hard to know where the taxi is heading because taxi drivers using this system does not indicate the final destination of the ride. Therefore, it is hard for the dispatch system to choose which taxi to dispatch for which customer or which pickup location, so taxi trajectory prediction for the final destination of a taxi trip can be extremely helpful.

The goal of this project is to predict the taxi trip destination in Porto, Portugal based on some given initial partial trajectories. The predicted output should contain three columns of information, the unique trip TRIP ID, LATITUDE and LONGITUDE which is the destination in WGS84 coordinates. We introduced a similarity-based model that correlates initial starting point to the final destination. We also implemented a neural network approach that embeds all trip meta data and trajectory and prefix with clustering methods. Both methods showed competitive results as the first place of the Kaggle Competition. More specifically, our retrained NN model achieved the cost as low as 1.84 which beats the reported best score (1.87) and similarity-based model successfully got a cost score of 2.27, which also took a top-20 place in the contest.

The paper is organized as follows. In Section 2 we introduce related work on location prediction. Section 3 discusses key aspects of our approaches as well

as the implementation details of the baseline model and neural network model. Data preparation, performance evaluation method and our results as well as threats to validity are presented and analyzed in Section IV. Finally, Section V concludes the paper and briefly presents future work.

2 Related Work

The first place winners of the competition describe their approach in [1]. It is a neural network based approach with only one fully connected layer followed by a softmax and a centroid layer. One novelty idea is how they compute the final destination: rather than simply use two output neurons (one for the latitude and and one for the longitude), they propose the weighted combination of a set of predefined destination points as the final destination, where the weights are the outputs of the softmax layer. These predefined destination points represent the prior knowledge of the distribution of the data. Other novel ideas include the use of embeddings to encode the original data, and the selection of a fixed number of trajectory points as input to the hidden layer, etc. The beauty about this approach is that it requires less feature engineering and its modular architecture allows it to be plugged easily. Its end-to-end training also implies that it has a higher accuracy. However, that also means it requires a lot of hyperparameter tuning, a longer training time, and more architectures to consider. Since it is not a tree-based approach, it is also less interpretable.

BlueTaxi [2] in IBM Research (Ireland) put forward their ideas on destination and trip time prediction from partial trajectories, which also serves as one of the best solutions in this contest. An approach based on trip matching and ensemble learning with proper feature extraction was presented by their team. The predictive modelling module is defined as follows. For destination prediction, they choose Random Forest (RF) due to easy access of the contribution of each feature on the final prediction; for trip time prediction, a model ensemble is trained with individual members of Gradient Boosted Regression Trees (GBRT), Random Forest Regressor (RF), and Extremely Randomized Trees regressor (ERT). In their tree-based model, features such as kernel regression features, travel time and Haversine distance to nearest neighbors, time gap between the cut-off and starting timestamps are included to get a more robust result. Besides, the main method proposed by the first place of the competition, trip time prediction by Thomas Hoch[3] also adopts Random Forest Regressor (RFR) and Gradient Boosting Regressor (GBR) with effective feature engineering. Moreover, for sequential data such as taxi trajectory, pattern matching and similarity-metric based model are always one option for prediction.

Based on the survey above, two major approaches, that is, tree based learning model and neural network based model, are typically achieved optimal solutions for this problem.

3 Implementation

3.1 Strategy Analysis

Preprocessing is good in terms of selecting features that are more useful. We believe that understanding this feature extraction step can help us better apply the algorithms to yield the best result. Therefore in this section, we discuss the insights that we have gained during the strategy analysis stage.

(1) Some trips tend to stop for a long time in middle of the trip due to traffic. We believe that counting the number of waiting instances can be meaningful because if a taxi is staying at the same place for a decent amount of time, its previous trajectories may not even be that useful therefore we can consider this waiting point the new start point. To count the number of waiting instances, it is basically the same as counting the number of coordinates that are the same because each coordinate is measured every 15s in the input file. Since the original data set contains 1.7 million entries, we used a tenth of it, that is 170000 data points. The result is Counter (2: 111348, 1: 29742, 3: 26986, 4: 1915, 0: 9), so an average waiting of (2)*15 seconds = 30 seconds. We can therefore conclude that 65.49% trips waited for 30 seconds, 15.87% trips waited for 45 seconds, and 1.13% trips waited for 60 seconds.

(2) Our visualization result also indicated that previous part of a journey can sometimes become a burden when predicting the destination.. This confirms to our common knowledge for people are always getting closer to their destination in almost any journey.

(3) The graph below explain for why we choose to associate taxi stands to the starting point of the trip to better predict the destination. For example, almost 3600 trips started from taxi stand 9 has the same destinations shown in Figure.1. below, therefore, high destination counts provide a some confidence that people from one taxi stand tend to go to the same destination. From the heatmap in the right side figure.2., we can intuitively and clearly see that most places with high-frequency visits are located along the taxi stands rather than distributed evenly.

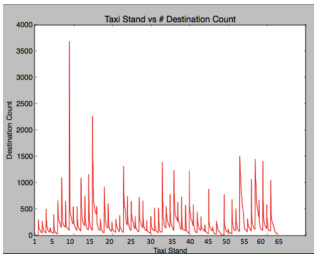

Fig. 1. Max count for same destination from different taxi stand

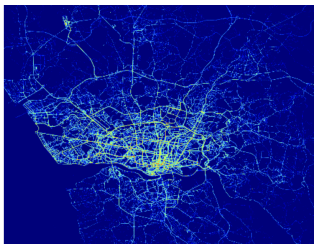

Fig. 2. Taxi stand id with Destination count and trip heatmap on entire provided dataset

(4) Set out time influences the destination to certain degree. From our visualization result, we find a number of popular destinations for trip from 11PM to 2PM local time on weekends. We believe people are probably looking for dining place around noon time and thus can share same destination.

To get a better understand of uber movement, we also plot the journey in the testing set across all country:

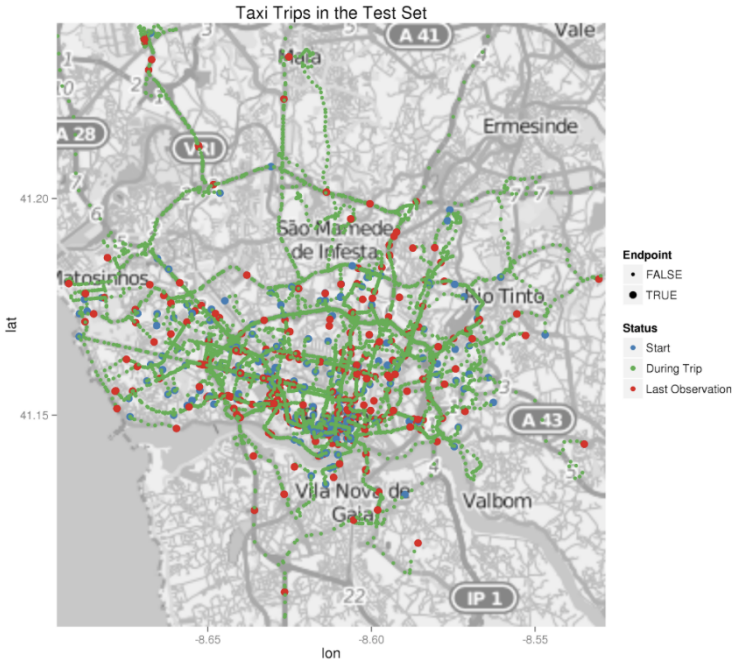

Fig. 3. Neural network architecture for the winner group

3.2 Data Preprocessing

Enlightened by the result mentioned above, we conduct following data preprocessing on our training set:

(1) To better bin the locations, we first divide the area covered by the training set. The final grid solution we choose is 100m * 100m.

(2) In one training example, if the taxi does not move more than one meter within 45 seconds. We suppose the driver has forgot to cooperate the system and has actually start another journey with different customers. Thus, we ignore the previous trajectory.

(3) For testing set, we cut the journey and only keep the last 90 seconds trajectory.

(4) We also calculate the nearest taxi stand for every trip and include it as another important feature for our training model.

3.3 Baseline Similarity Based Model

As our baseline model, we adopt a very intuitive customized nearest neighbor approach for this task. First for every testing case c, we define a training sample is related to c if it:

(1) Shares the same nearest taxi stand to trip starting point with c

(2) Sets out from within $-t \ +t$ hours from the starting time of c. (Here we consider time within day as the only difference between trips in time perspective.)

We aggregate this trips into a related training set for c and denote it as $Train_c$. When picking the top-K neighboring samples, we only consider the ones in $Train_c$. To measure the similarity between training sample t and test case c with the same trajectory length, we use a customized matching function to calculate the pairwise distance:

$$dis(t, c) = dis(t_1, c_1) + \lambda dis(t_2, c_2) + ... + \lambda^{n-1} dis(t_n, c_n)$$

As the above formula shows, we use a slide window to match the training trajectory and the partial testing trajectory. Then we pick out the smallest distance from all windows and use it as the final distance between the two trajectories. Here we adopt an exponential weighting strategy with $\lambda > 1$ to given place higher importance on the latter part of the journey.

Now that we can calculate the distance between test case c and every training sample in $Train_c$. We pick out the top-K training samples that have smallest trajectory distance and use their geology median coordinates as the final prediction for c.

We tune the our hyper-parameters $K, t$and λ on the validation set to generate the final prediction for the testing data.

3.4 High-Level Framework for Neural Network

The winning method of ECML/PKDD 15: Taxi Trajectory Prediction (I) use Neural Network (NN), MLP more specifically and many other variations. As we know, a neural network can artificially obtain features from data and generate exciting results in many complex data-driven tasks including image recognition. In this section, we focus on the frameworks of NN-based and its implementation on training those models together with explanations on proposed models and training procedure.

3.4.1 Best model architecture Shown in the Fig.4, the components of the model with the best performance are input layer (metadata and trajectory prefix data respectively), hidden layer, softmax and clustering alignment, which are

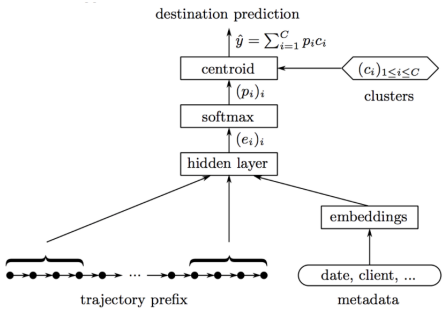

Fig. 4. Neural network architecture of the winner group

from the insights from winning methods in [1] and a data-driven model particularly designed for taxi trajectory dataset in Kaggle.

Input Layer: The MLP model provides separated ways to encode different data types. More specifically, the K-pair($2K$) trajectory data from both start and end points are directly fed into the hidden layer. In our experiment, Ks value is set as 5, which is considered as a hyperparameter. Meanwhile, other information such as metadata including date, client, trip type (see Section 4 for more detail), those features belongs categorical variables other than numeric ones, so it is better to embed those original features into 10-dimension vectors before serving as the input of hidden layers. Note that the embeddings of metadata are directly concatenated with $2K$ pairs of trajectory data.

Hidden Layer: The hidden layer here is conventional MLP structure with softmax layer (with input value of e_i to output value p_i) connected to clustering and output layer, which is explained in the next paragraph.

Destination Clustering and Output Layer: Destination prediction is indeed a regression problem with the output of two scalar values. However, it is not easy to predict the exact numeric results of latitude and longitude along with the limited provided input. The final part of the model adopted a strategy as destination clustering. A predefined destination set (a few thousand cluster centroids c_i from all destination in training data) is associated with the output scalar value from the softmax layer. (Note that those selected centroids are generated by mean-shift clustering method.) The final prediction is the weighted p_i average of those predefined destination points c_i:

$$\hat{y} = \sum_{i=1}^{C} p_i c_i, \;\; p_i = \frac{exp(e_i)}{\sum_{j=1}^{C} exp(e_j)}$$

3.4.2 Alternative NN-based models There are many variational models for this prediction task based on neural network design, which provides insightful alternative models in spite of their inferior performances. For example, to encode the trajectory prefix, which has the format of sequential data, a widely-used

model is Recurrent Neural Networks (RNNs) with lots of variations of GRU, LSTM and bidirectional LSTM. In Figure 5, RNN-based model architectures are presented. Due to limited time and the long time for training process, we could not cover the test for RNN models.

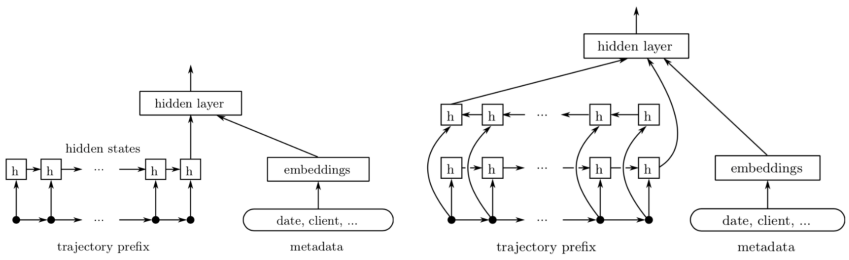

Fig. 5. Alternative RNN based model

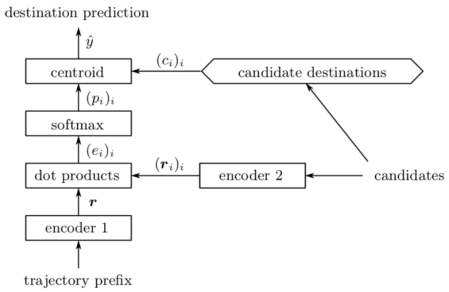

Fig. 6. Alternative Alternative Memory Network Model

A novel model framework we tested is Memory Networks (shown in Figure 6). In the model of memory network, two encoders are employed to encode the trajectory prefix and entire trajectories in a random batch of training candidates. The encoders can be models we mentioned before including MLP, LSTM and so forth. The final prediction is based on similarity of two representations. The top layer for clustering and destination output is the same as the winning model in Section 3.4.1. More information can be found in [6].

3.4.3 Implementation and training process The implementation is based on ¡https://github.com/adbrebs/taxi¿, and the training process is performed on a single NVIDIA TESLA V100 GPU with 16GB memory. The entire training duration for MLP, MLP (without embedding) and Memory Network are approximately 24 hours (over 7,000,000 iteration), 5 hours (over 3,100,000 iterations)

and 28 hours (over 9,000,000 iterations). For hyper-parameters of neural network, we adopted the parameter settings from the winning methods, that is, SGD with a fixed learning rate of 0.01 and a momentum of 0.9, together with a batch size of 200.

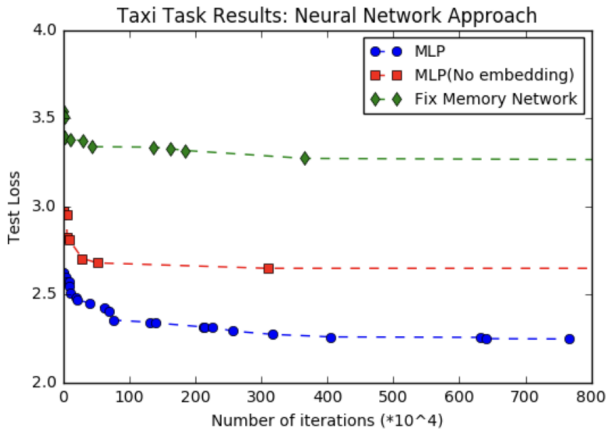

Fig. 7. Alternative Alternative Memory Network Model

The figure 7 shows some sample training processes of NN-based models. A high-level conclusion is that, model architecture mostly determines the final performance and the training process can slightly improve the model performance but only be effective with a proper model base. From our observation, MLP, which may be the simplest neural network, outperformed other more complicated, time-consuming models such as memory network, and other reported LSTM and Bi-LSTM. The reason can partly be explained as the entire dataset are not large enough for tuning a typical RNN sequential data encoder in complex neural networks or even memory network.

4 Experiment setup

Data description Dataset related to the contest is provided by Kaggle and no external data resources are allowed to use in model. The dataset accurately describes a complete year (from 01/07/2013 to 30/06/2014) of the trajectories for all the 442 taxis running in the city of Porto, in Portugal. These taxis operate through a taxi dispatch central, using mobile data terminals installed in the vehicles. The entire dataset mainly contain three files, which is explained in detail in the following part of this section.

The basic features in the training and testing data set include : TRIP ID, CALL TYPE, ORIGIN CALL, ORIGIN STAND, TAXI ID, START TIMESTAMP, DAY TYPE, MISSING DATA TAG, and (PARTIAL) TRAJECTORY

POLYLINE information. The competition also provide a file containing all meta data for 63 different taxi stands.

5 Evaluation

5.1 Performance Metrics

In terms of the scoring function, since the goal of this project is to predict the destination of taxi trips in Porto, Portugal based on their initial partial trajectories, and the output should be the final trips destination in WGS84 coordinates, i.e. latitude and longitude, the benchmark is simply the difference in distance between the predicted location and the actual destination. In this project, we will summarize every part of the approach in terms of the actual score because the loss function will evolve down the chain, and therefore it will be very hard to weigh each layer individually. The scoring function we used is the Mean Haver-sine Distance formula, which gives the distance between two points on a sphere based on their latitude and longitude and is described below:

$$\alpha = sin^2(\frac{\phi_2 - \phi_1}{2}) + cos(\phi_1)cos(\phi_2)sin^2(\frac{\phi_2 - \phi_1}{2})$$

$$d = 2 * r * atan(\sqrt{\frac{\alpha}{1-\alpha}})$$

Where ϕ is the latitude and λ is the longitude, d is the distance between two points, and r is the spheres radius which is the Earths radius or 6371 kilometers. We submitted our code to the Kaggle challenge board and obtained the score which will be discussed in detail in the performance comparison section.

5.2 Performance Comparison

In this section, we present our results for both similarity based model and neural network based model. As we mentioned in Section 1 partly, Table 1 has shown the full final cost score of all the models we have implemented.

Model	Custom Test	Kaggle Public	Kaggle Private
Similarity-based model	2.634	2.654	2.278
Memory Network (Basic setting)	3.262	2.978	3.167
MLP, clustering, no embedding (our training)	2.646	2.791	2.407
MLP, clustering with embedding (our training)	**2.246**	**2.605**	**1.844**
Winning Method (MLP, reported)	2.819	2.398	1.879
Winning Method (Memory network, reported)	2.879	2.779	2.209

Table 1. Different model performance

The results are encouraging that our training MLP model with embedding and centroid clustering even slightly **outperformed reported winning**

method by the cost score of 1.84. Meanwhile, our proposed similarity-based model achieved a score of 2.27 on private board, which took the 23rd place on the leader board. This model can successfully be applied as a basic core part for trajectory analysis and destination prediction.

Baseline experiment code and data visualization code can be found on the project github link here: https://github.com/vivi51123/CS249-Current-Topics-in-Data-Structure. The website features the dataset as well as the documentation on how to run and test the code. As the website evolves, more documentation and features on the algorithm will be available.

6 Conclusion

In conclusion, though this is a closed competition, we still achieve a lot by discussing and learning from other teams on Kaggle. We also constantly update and fix our model for better performance. We find it extremely helpful to do data visualization and certain analysis before settling down main approach. And we spend a lot of time learning about different neural network design as well as doing the experiment. Our final performance on this course project is satisfying.

References

1. De Brbisson, Alexandre, et al. "Artificial neural networks applied to taxi destination prediction." arXiv preprint arXiv:1508.00021(2015)
2. Hoang Thanh Lam, et al. (Blue) Taxi Destination and Trip Time Prediction from Partial Trajectories arXiv preprint arXiv:1509.05257(2015)
3. ECML/PKDD 15 Taxi Trip Time Prediction (II): 1st Place Solution write-up
4. Kaggle contest of ECML/PKDD 15: Taxi Trajectory Prediction (I), Discussion board: https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i/discussion
5. Kaggle Kernel: Visualization of taxi trip end points https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i/kernels
6. Weston, Jason, Sumit Chopra, and Antoine Bordes. "Memory networks." arXiv preprint arXiv:1410.3916 (2014).