Driver Identification based on Route and Driving Time using Trajectory Data

Xintao Yan
Civil and Environmental Engineering, University of Michigan

Xingmin Wang
Civil and Environmental Engineering, University of Michigan

Tian Xie
Urban and Regional Planning, University of Michigan

In current research field, there are many researches on identifying drivers using mobility data, including CAN bus data and trajectory data. Many classification models they presenting were trained with data from these data sources focusing on the drivers' driving behavior. However, it is hard to quantify or describe the drivers' behavior because it is highly affected by the environment including the weather and traffic condition. In this paper, we explore the possibility of constructing driver identification model using the route and driving time information which uses the trajectory data as the only input. We use dataset from Safety Pilot Model Deployment (SPMD) and matched the original trajectory to the existing road network using Hidden Markov Model (HMM). To form the training data, a reformatted matrix is created to reflect the distance traveled in each network link and the travel time in a day. Both Nearest Subspace (NS) and Random Forest (RF) models are implemented in this paper. Analysis about the impact of driver number and individual driver data quality on model accuracy were performed. In conclusion, our driver identification model can achieve a satisfying performance using only trajectory data and individual driver's predicting accuracy which uses the route and the driving time as the main features.

I. Introduction

With the rapid development of smartphones and communication technologies, floating car data (FCD) are becoming more and more easy to obtain. Compared with traditional detectors, FCD provides a more substantial amount of data at a lower cost. In recent years, FCD has been widely used for many applications, such as queue length estimation, traffic state estimation, traffic volume estimation and risk evaluation [1–6].

Each probe vehicle works as a moving sensor on the road and its trajectory records driver's driving behavior and daily travel information. These precious data provide great convenience for researchers, but it also brings a lot of privacy concerns which are whether driver's identification information can remain anonymous. If the driver can be re-identified based on their historical trajectory data, then the driver's privacy will be compromised. More importantly, it will cause a widespread panic among those who are sharing their mobility data. Also, to flag counterfeit data, driver identification can help transportation researchers or organizations who rely on trajectory data to perform analysis detect outliers in their dataset.

Therefore, in recent years, many researchers focused on how to identify driver based on mobility data. Lots of researchers leverage CAN bus signals to develop identification models. That is because CAN bus signal records how drivers react to daily traffic situation which makes it a good representation of driving behavior. Wakita et al. [7] and Miyajima et al. [8] developed two types of models and made a comparison between them. The first type is based on a physical model (Helly model and optimal velocity model) and the second type is Gaussian Mixture Model (GMM). The results show that GMM has a better performance and achieve 78% accuracy among 274 drivers in the field test. Enev et al. [9]) used random forest method to reach 99% accuracy with 5 sensors signals and 87% accuracy among 15 test drivers with only the brake pedal sensor and route information. Hallac et al. [10] build a classifier according to a single turn scenario in 12 most frequently situations such as rural, urban, etc. The average accuracy is 50.1% for identifying between five drivers. There are also researchers use virtual simulators in order to mimic vehicle CAN bus data. Zhang et al. [11] used Hidden Markov Model (HMM) and GMM model based on data collected from drivers in a simulation

environment. Their model reaches 73% accuracy.

However, there are some deficiencies with the methods mentioned above. The most important limitation is that these data are all collected in a limited and controlled test environment. As a result, driving behavior of drivers are collected in a fixed traffic state such as traffic volume, time-of-day, weather, etc. In the real world, the trajectory data of different drivers may come from different traffic situations. Therefore, using real-world trajectory data may not return a satisfied classification accuracy. Another limitation is that CAN bus data are usually very hard to obtain which also restrict the implementation of those methods. What's more, using data collected from simulation environment would also not a good idea because drivers will not react the same as in the real world in virtual simulator due to the lack of real-world traffic patterns and weather conditions, etc.

The goal of the research is to explore whether we can do driver re-identification based on the daily route and driving time pattern of the driver only using the trajectory data i.e., longitude, latitude and timestamp of vehicles. This requires only trajectory information of each driver, so the model proposed in this paper has a good generalization ability and solid use prospects.

The rest of the paper is organized as followed. Section 2 will first introduce the data we used and the data preprocessing method. Then, Section 3 will thoroughly discuss the Nearest subspace and Random Forest approach we implement for driver re-identification. Result analysis and comparison of these two methods will be discussed in Section 4. Finally, Section 5 gives the concluding remarks and future works.

II. Dataset Preparation

A. Data Source

The dataset used in this paper is from the Safety Pilot Model Deployment (SPMD) project lead by the University of Michigan Transportation Research Institute (UMTRI) [12–14]. SPMD has as many as 2, 800 vehicles equipped with devices for V2V and V2I communication system. There are four types of vehicle equipment configurations in SPMD vehicles: Integrated Safety Device (ISD), Aftermarket Safety Device (ASD), Retrofit Safety Device (RSD) and Vehicle Awareness Device (VAD). Among the 300 vehicles equipped with ASD, 98 are equipped with data acquisition system (DAS), which is used to record data including forward object information, position information, lane tracking information and remote vehicle BSM and classification. There are many channels in the SPMD dataset in the dataset manual. In the DataWsu File of the DAS1 dataset, both the GPS trajectories and the heading are included, which will be our main data source.

There are totally 11 drivers' trajectory data for 30 days in the data source and each trajectory includes the latitude, longitude, timestamp and driver id with a sampled period of 0.1s. Besides the trajectory data, the map information is also required to map the trajectory data to the road network. This paper used the road network information provided by OpenStreetMap. In this road network, a road was defined as a link composed of a series of nodes with known latitude and longitude, and there are also some details of the road such as whether the road is a one way. The following figure 1 gives a brief view of the original data source. The blue lines, which are composed of a series of trajectory points, are the trajectory data provided by the SPMD while the light grey lines are the road network. In this figure, although it seems that the trajectory points are right on the road network, they are not matched yet.

B. Map-matching

The main features to identify the driver in this paper are the route and driving time. Therefore, the trajectories should be matched to the road network so that we can know which road each trajectory belongs to. This paper utilized the Hidden Markov Model to match the trajectory data proposed by Newson et al. [15].

The key problem in map matching is the tradeoff between the roads suggested by the location data and the feasibility of the path. For example, one simple method to determine the link that the GPS point belongs to is to find the nearest link among all the candidate links. However, this method ignore the relation between the GPS points and the feasibility of the path. The model proposed by [15] has two parts to quantify the probability that which link that one GPS point might belong to.

As shown in the figure 2(a), z_t and z_{t+1} are the GPS points at time stamp t and $t+1$, r_1, r_2 and r_3 are different links. The first part is related to the distance between the links and the trajectory points. The GPS points are assumed as a Gaussian distribution so that the probability can be written as:

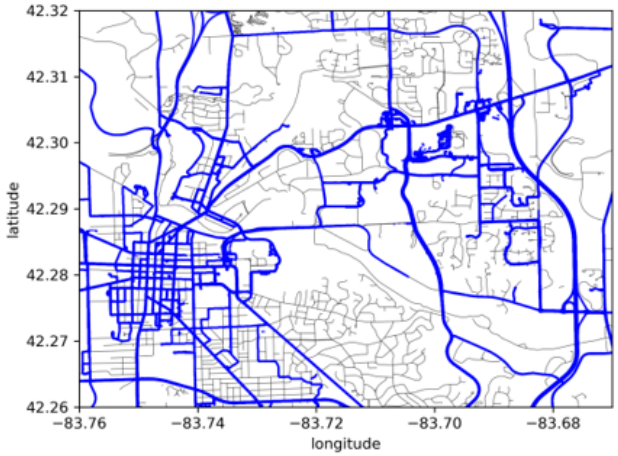
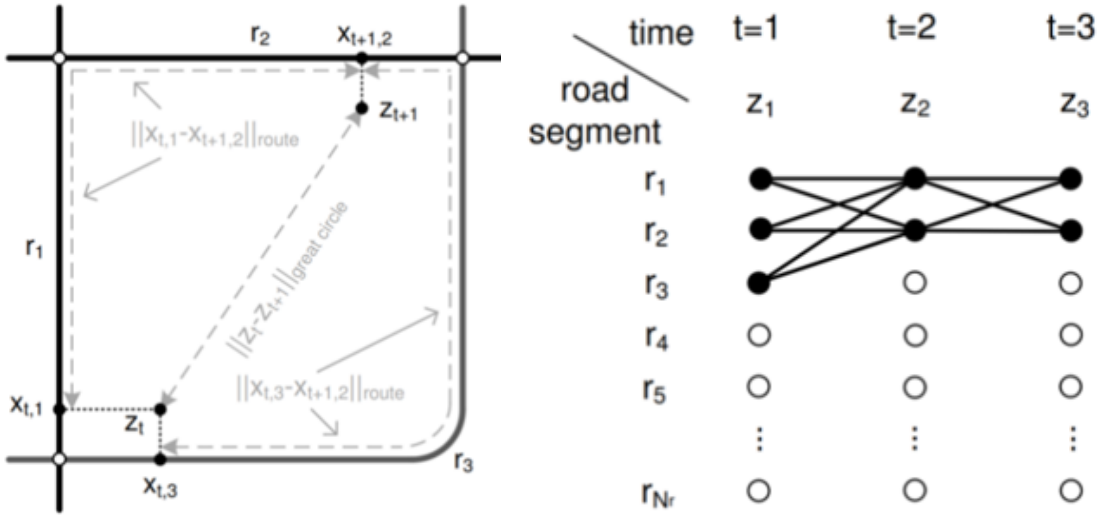


Fig. 1 Original data source: trajectories from Safety Pilot Model Deployment and road network from OpenStreetMap.



(a) Illustration (b) Markov chain

Fig. 2 Illustration of Hidden Markov chain map matching model

$$p(z_t|r_i) = \frac{1}{\sqrt{2\pi}\sigma_z} e^{-0.5\cdot(\frac{||z_t - x_{t,i}||}{\sigma_z})^2} \tag{1}$$

The other part to evaluate the probability is the feasibility of the path, the paper defined the transition probability as:

$$(d_t) = \frac{1}{\beta} e^{-\frac{d_t}{\beta}} \tag{2}$$

where the d_t equals to

$$d_t = |||z_t - z_{t+1}||_{greatcircle} - ||x_{t,i^*} - x_{t+1,j^*}||_{route}| \tag{3}$$

which means that the shorter the route, the greater the probability the route is the true route. So far the model has defined the probability of each state and also the transition probability from one state to another shown as figure 2(b). Then the map-matching problem can be converted to a dynamic programming problem.

Figure 3 gives the results of the map-matching algorithm. The blue line is the trajectory and the grey lines are the road network. The red points show the result after matching the trajectory data to the road network and the sampled period chosen here is 3 seconds. The green points are also the candidates after projecting the trajectory points to the links but they are not selected after the dynamic programming. The figure 3 shows more details about the dynamic programming, the x axis is the index of the time stamp while the y axis is the index of the candidate links. Because if the distance between the candidate links will be discarded if the distances between the links and the trajectory points exceed a threshold (about 20 m), most of the time there is only one candidate link. The number of candidate links will increase when there is a intersection and at this time the algorithm will automatically select the right link considering both the distance between the links and the trajectory points and the feasibility of the path.

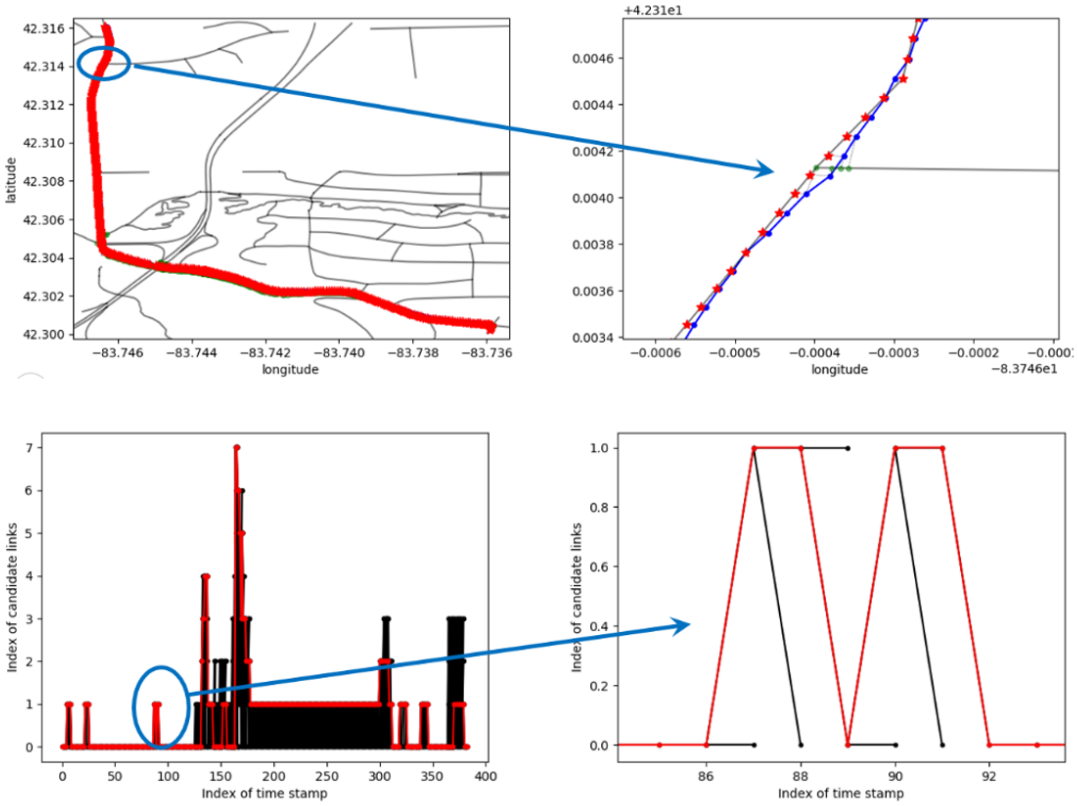


Fig. 3 Result example of map-matching and calculation details of Dynamic Programming

Since the dataset do not have a ground truth of the map-matching, it is hard to quantify the accuracy of the map-matching algorithm. To test the map-matching algorithm, we simply plotted the map-matching results as the figure 4 and manually evaluated the algorithm. It turned out to be quite accurate except for the absence of the road network.

C. Dataset Reformat

After matching the trajectory data to the road network, we can exactly know what the distance of each driver drives on each link. This paper utilize this route information to identify different drivers since different drivers have different regular demand for daily driving. Besides the route, the driving time is also used as a feature to distinguish different drivers. Figure 5(a) gives the driving time heatmap of a specific driver. The axis x is the time in a day (24 hours) while the axis y is index of the day in a month.

A shown in figure 5(b), there are totally 11 drivers' 30 days' trajectories. Each day's trajectory of each driver was considered as a sample data denoted as a column vector x. The column vector x is composed of the distance that the driver travel on each link and the driving time (unit:h) from 00:00 to 23:59. In the latter numerical experiment, several days' data was used as the test data while the left-over data was used as the training data.

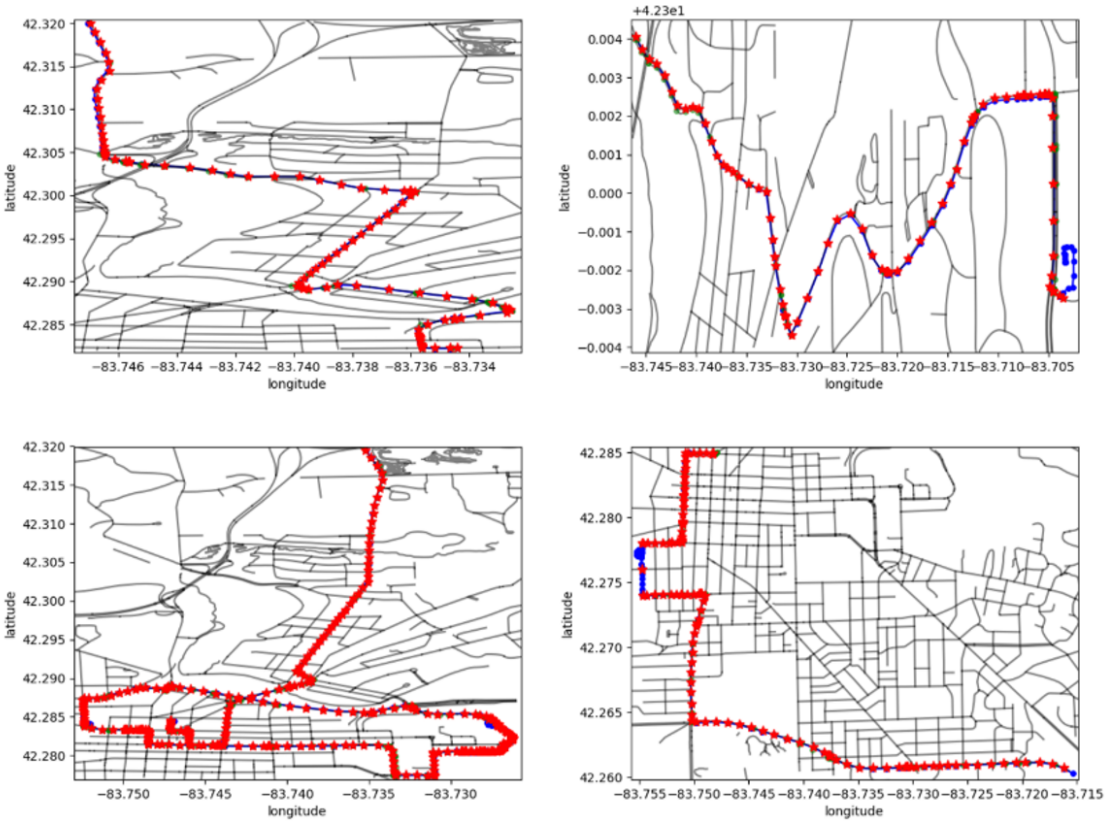
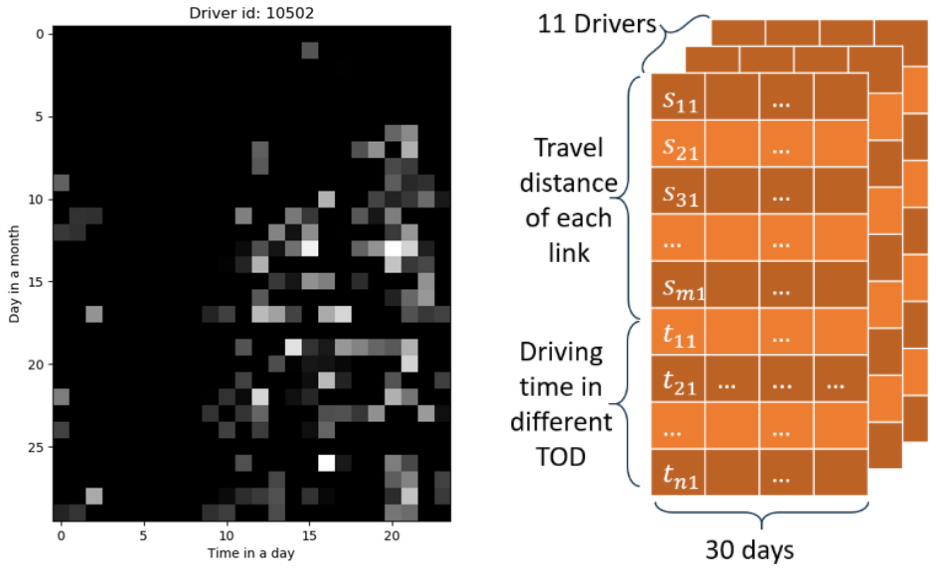


Fig. 4 More examples of the results of map-matching



(a) Driving time heat map

(b) Data format

Fig. 5 Caption place holder

III. Methodology and Numerical Results

A. Method 1: Nearest subspace classification

The nearest subspace (NS) classification algorithm is an efficient method for multi-classification problem. It has been widely used in machine learning region and shows good performance dealing with face recognition [16, 17] and handwritten digit classification [18], etc. In recent years, the nearest subspace has also been proved effective for classification problems with missing or noisy input data [19]. The training set data can be grouped according to its class (label). Each point has d dimension features and there are N data points in class i. Then the training set of class i can be represented as $D_i \in \mathbb{R}^{d*N}$. The key idea of NS is that each class will form a subspace and an unknown vector will be classified to which subspace it is closest to. The first step is to construct subspace of all classes. An efficient way to calculate orthogonal bases representing the subspace is based on matrix Singular Value Decomposition (SVD). For any matrix $A \in \mathbb{R}^{m\times n}$ can be written as the product

$$A = U\Sigma V^H = \sum_{i=1}^{r} \sigma_i u_i v_i^H \tag{4}$$

Where $U \in \mathbb{R}^{m*m}$ and $V \in \mathbb{R}^{n*n}$ are orthogonal matrices and u_i, v_i are left and right singular vectors, respectively. Σ is an $m \times n$ diagonal matrix with non-negative entries, ordered in the following way:

$$\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r \geq 0 \tag{5}$$

For any given vector x, it can be projected to $\mathcal{R}(A)$ by

$$A \cdot x = \sum_{i=1}^{r} \sigma_i u_i v_i^H x = \sum_{i=1}^{r} (\sigma_i)(v_i^H x) u_i \tag{6}$$

Therefore, $U = [\ u_1 \quad u_2 \quad ... \quad u_r\ ]$ is the projection matrix of $\mathcal{R}(A)$. However, in many applications, the input data is often high dimensional and contain noise. Therefore, using all left singular vectors to construct subspace will not return the best classification performance because it captures and reserves all the noise [5]. To resolve this problem, we need to select first k left singular vectors as the orthogonal bases of the subspace. The selection of dimension k can based on the singular value spectrum of all the training data. The number of singular values appear to separate from the continuous portion of the spectrum would be the initial selection of k. Then using validation, we can further modify the choice of k. Choosing a low-dimensional subspace instead of all left singular vectors is equivalent to feature selection. And the process of feature selection done by SVD is essentially the same as PCA method. According to the aforementioned process, we can obtain $U^i \in \mathbb{R}^{d*k}$ as a matrix of orthogonal basis vectors representing k dimension subspace of data in class i.

Then we can define the orthogonal projection matrix onto $\mathcal{R}(U^i)$ as $\mathcal{P}^i$:

$$\mathcal{P}^i = U^i \cdot (U^i)^H \tag{7}$$

Let $\mathcal{S}$ define the set of all classes. The predicted label for vector can be decided by the Euclidean distance to each subspace. It can be formulated as the following optimization problem:

$$label = \arg\min_{j\in\mathcal{S}} \left\|(I - \mathcal{P}^i)x\right\|_2^2 \tag{8}$$

Based on the mentioned process, we apply NS algorithm in our problem. The singular value spectrum of our dataset are shown as 6. The spectrum shows that there are approximately 15-20 singular values separate from the continuous portion. Therefore, the selection of subspace dimension should be 15-20.

However, due to the lack of data, some of the drivers just have very limited data then we can only set the dimension k as close to 15 as possible. To guarantee that k no less than 10, we filter out 3 drivers who have less than 10 driving data in this month. Then for the rest of 8 drivers, we set k equals to 10 and randomly choose 1 data per driver to constitute the test set. After doing 1000 times experiments, the mean accuracy of classification error is 71.04%. Noted that this result is achieved by very low dimension of subspace. According to the mechanism of NS algorithm, the accuracy will increase if we have more data and can set k higher. The following figure 7 shows the classification correction heatmap. The x-axis is the true label and the y-axis is the predicted label. Therefore, the entry (i, j) represents the percentage of

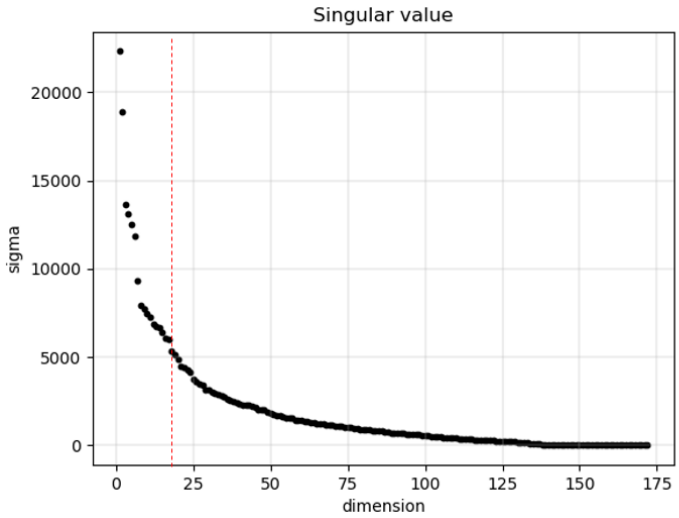


Fig. 6 Singular values scatters

test data belonging to the class i identified as the class j. So the elements on the diagonal indicate the classification accuracy of each class in 1000 experiments.

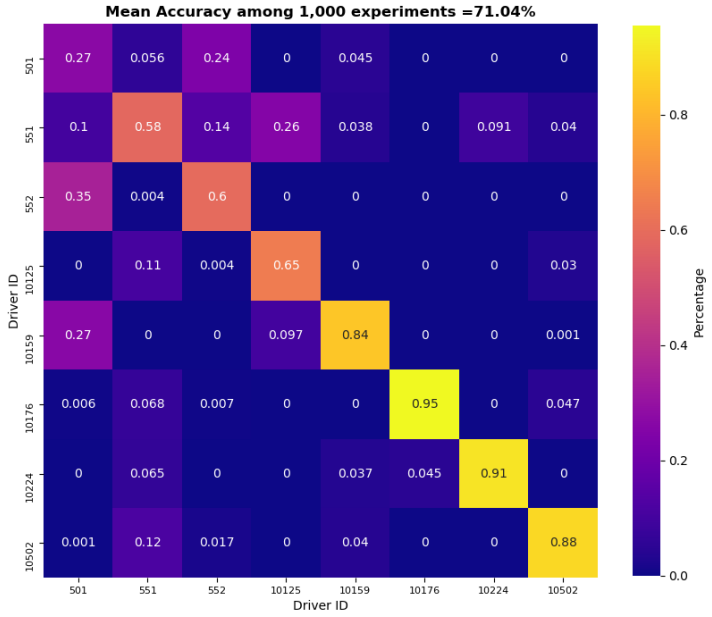


Fig. 7 Identification matrix using Nearest Space method

From the result, we can find that last four drivers have very high classification accuracy. But for the first driver, the model is not working well. One possible reason is that last four drivers might have very clear routing and driving period pattern. Another possible reason is that the driver 501 has only 10 training data, which is too limited compared to other drivers, resulting in insufficient representation of the driver's daily mode.

B. Method 2: Random forest classification

Aside from the nearest subspace classification, we also use random forest classification to solve this multiclass classification problem. Random Forest is an ensemble learning method which learns through constructing a number of decision trees and outputting the class which gets the highest probabilistic prediction from individual trees. Random forest algorithm was created by Ho (1995) [20], and Breiman (2001) [21] proposed an extended random forest classification by introducing "bagging" idea and random selection of features. Because the dataset we are trying to learn has high dimensionality and the decision tree is robust to the inclusion of irrelevant features, for this classification problem, we choose to use the Breiman's random forest classification algorithm.

The random forest model use bagging, also called bootstrap aggregating, technique. Given the training set $X = X_1, X_2, \cdots, X_n$ and training label $Y = Y_1, Y_2, \cdots, Y_n$, for each tree out of total L trees, bagging selects a random sample with replacement of the training set $N_l^{'} \leq N$ and train tree with $N_l^{'}$. Using bagging allow us to acquire better model performance by decreasing the variance of the model, without increasing the bias.

In an individual decision tree, the random forest model also implements feature bagging in determining split in each node. The feature bagging, also called as Random subspace method try to reduce the correlation between trees in an ensemble by training them on random samples of features instead of the entire feature set. For each tree, given a random sample from bagging $N_l^{'}$ with dimension D, at each candidate split in the learning process using Gini index, the tree will use a random subset of the features $D^{'}$ to determine the optimal splitting. Also, to further reduce variance in each decision tree and improve generalization performance, we constrain the maximum depth and minimum sample size required for each split. Together with the number of trees in the ensemble, we have used grid search cross-validation to find the best hyperparameter value.

There are three important parameters for our random forest classifier: $N_{estimators}$, max_depth, and $min_samples_split$. Is is unknown that which combination of these three hyperparameters is best for solving this classification problem. So, we did a grid search cross-validation to find the best combination of these three parameters. Before we started grid searching, we first divide the training set into 10 subsets of equal size and this is also called 10-fold cross-validation. Sequentially one of 10 subsets is used for testing the classifier trained by the rest of 9 subsets. And the accuracy of cross-validation is the percentage of data being correctly classified. After dividing the training set into 10 folds, then we start doing "grid-search" on different hyperparameter pairs. For $N_{estimators}$, we use a list of value range from 5 to 50; for max_depth, we use a list of value range from 15 to 40; and for $min_samples_split$, we use a list of value range from 2 to 11. After grid-search all possible pairs on 10-folds we divided, the best parameters set found is that

$$N_{estimators} = 35, max_{depth} = 21, min_{samples_split} = 2 \tag{9}$$

After we acquire the best hyperparameter set from grid-search cross-validation, the random forest classifier is ready to be trained. To show the overall performance of the classifier, we have train the random forest model for 1000 times and generate correction heatmap 8. The mean accuracy among 1000 experiment is 78.69%. And as we can see in the correction heatmap, driver 501 has not been classified very well compared to other drivers with an average accuracy of 0.48.

C. Comparison Between Two Methods

Compared with NS algorithm, Random Forest has a higher average accuracy. For specific driver, Random Forest performs much better in the first four (501, 10176, 10224, and 10502) and especially for the first driver(501). But NS outperformed the RF algorithm in the last four drivers (552, 551, 10125, and 10159). Take a closer look at recall performance, the first two drivers(501 and 10176) have a lower true positive rate. And in the RF algorithm, drivers 6 and 7 (551 and 10125) have underperformed in terms of recall performance.

IV. Results Analysis

Intuitively, for multi-classification problems, the classification accuracy will decrease as the classified categories increase. In our problem, when there are more drivers to be classified, the performance of the our models will decrease. Therefore, we would like to explore how the classification accuracy change with the number of drivers. For certain number of drivers, we randomly pick equivalent number from the dataset and do 100 times experiments to get the average accuracy. The result for NS algorithm and Random Forest are shown in the following figure 9. The blue line is the average accuracy and the shaded area is the 95% confidence interval. From the results we can find that generally classification accuracy decrease with the number of classes in both methods. The performance of these two methods are

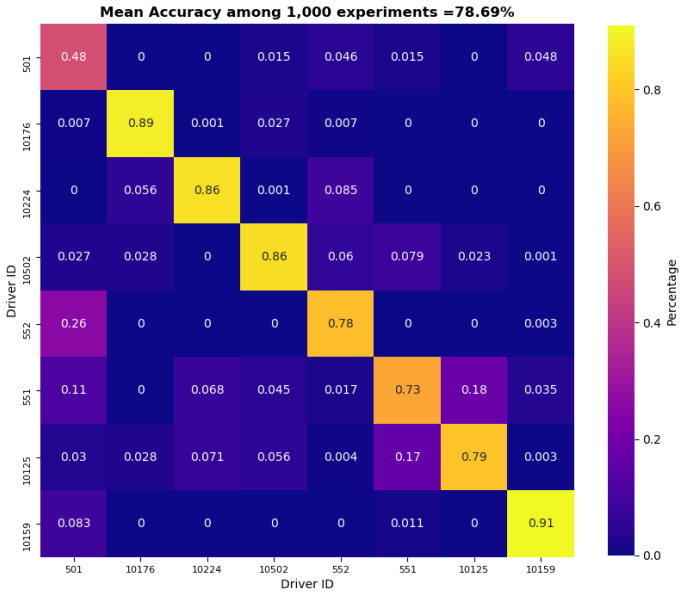
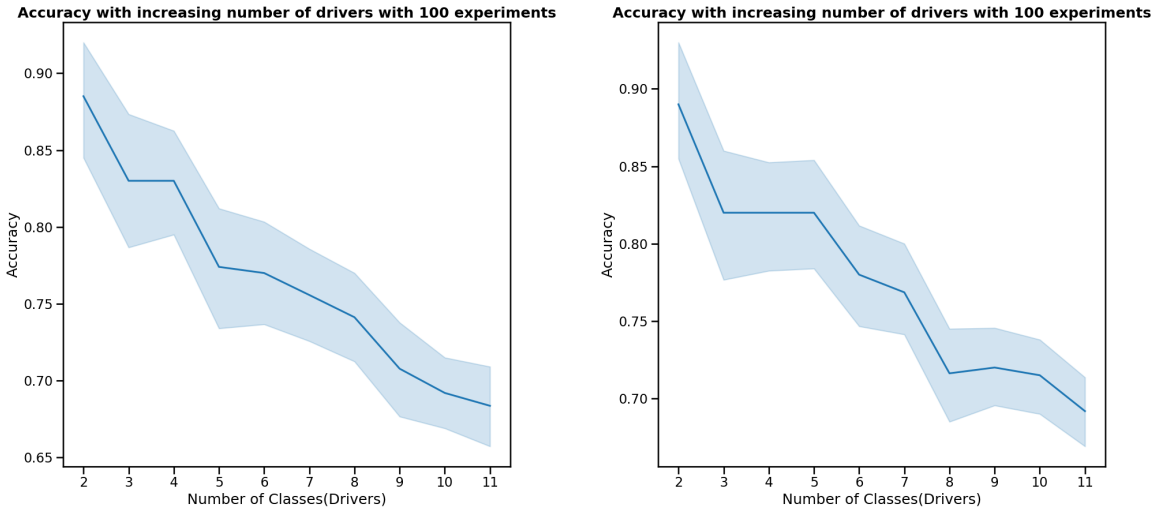


Fig. 8 Identification matrix using Random Forest



(a) Nearest Space (b) Random Forest

Fig. 9 Relationship between the average identification rate and the number of drivers

pretty close, where they all achieve around 89% with 2 drivers and 75% with 7 drivers. When there are 11 drivers, Random Forest performs slightly better with around 3% higher than NS classification accuracy. However, note that the random forest complexity is significantly higher than the NS algorithm, so the results are satisfied for both of the methods.

On the other hand, this paper utilized the daily route and driving time to identify the different drivers, which means that the driver with more regular daily driving pattern (route and time) and more data is more likely to be correctly identified.

The figure 10 gives the driving time heatmap of two different drivers. The first drivers almost drives start from the

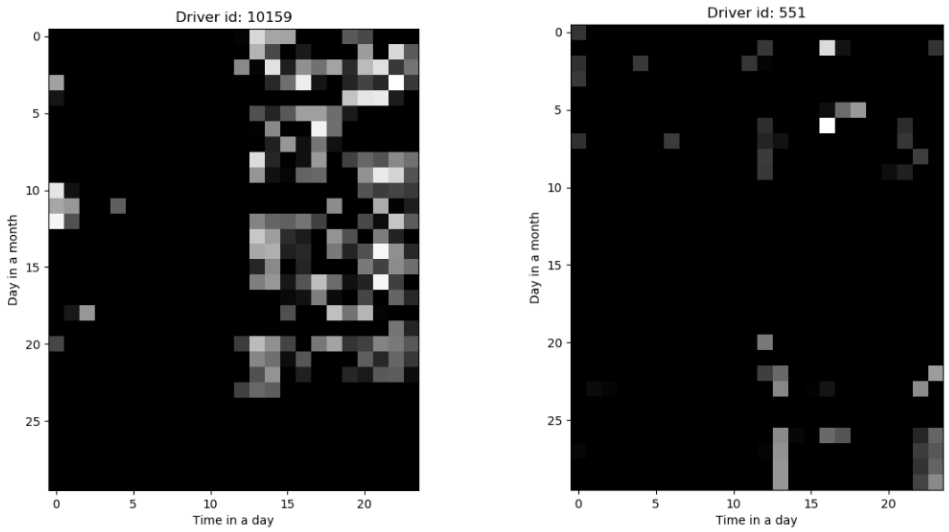


Fig. 10 Driving time comparison between two drivers

12:00 until night every day (perhaps a Uber driver) while there is no notable pattern for the second driver. As a result, the first driver has a high identification rate of 91% using random forest while the second driver is 73%. Another reason is that the first driver has more data to train than the second driver. Both the number of the data and the driving route choice behavior will exert influence on the drivers' identification rate.

V. Conclusion

In this paper, we discover the need and the challenge for driver identification from literature review. This paper offered a detailed trajectory data processing and machine learning solution to solve driver identification problem. It begin only with the longitude, latitude, and timestamp of vehicle trajectory data. After being matched to the real world road network using carefully designed Hidden Markov Model, the map-matched trajectory data has higher accuracy and can reflect the actual precise travel route in each trip. Focusing on the unique pattern in terms of time in a day and route choice for each driver, we reformatted the trajectory data into the travel distance in each network link and the average driving time in a day. The data quality issue was also examined and we keep 8 out of 11 total amount of drivers which have enough trips data for training models. Then we use both Nearest Subspace Algorithm and Random Forest Algorithm to train a machine learning model. From the result of both two classification methods, the Random Forest has a higher average accuracy (91%) compared to the Nearest Subspace (73%). After realizing the number of driver may impact the overall accuracy performance of the models, we examine the influence of various drivers number to the overall accuracy. Acknowledging that some drivers are predicted with lower accuracy than others, to find out the reason behind this, we take a closer look two drivers who have different identification rates. To conclude, our method achieve satisfying performance in 8 drivers classification. However, it is necessary to see how well our proposed method perform in a more large scale situation. We leave this for future research.

References

[1] Seo, T., and Kusakabe, T., "Probe vehicle-based traffic state estimation method with spacing information and conservation law," *Transportation Research Part C: Emerging Technologies*, Vol. 59, 2015, pp. 391–403.

[2] Zheng, J., and Liu, H. X., "Estimating traffic volumes for signalized intersections using connected vehicle data," *Transportation Research Part C: Emerging Technologies*, Vol. 79, 2017, pp. 347–362.

[3] Comert, G., and Cetin, M., "Queue length estimation from probe vehicle location and the impacts of sample size," *European Journal of Operational Research*, Vol. 197, No. 1, 2009, pp. 196–202.

[4] Hofleitner, A., Herring, R., Abbeel, P., and Bayen, A., "Learning the dynamics of arterial traffic from probe data using a dynamic Bayesian network," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 13, No. 4, 2012, pp. 1679–1693.

[5] Irschik, D., and Stork, W., "Road surface classification for extended floating car data," *Vehicular Electronics and Safety (ICVES), 2014 IEEE International Conference on*, IEEE, 2014, pp. 78–83.

[6] Herrera, J. C., Work, D. B., Herring, R., Ban, X. J., Jacobson, Q., and Bayen, A. M., "Evaluation of traffic data obtained via GPS-enabled mobile phones: The Mobile Century field experiment," *Transportation Research Part C: Emerging Technologies*, Vol. 18, No. 4, 2010, pp. 568–583.

[7] Wakita, T., Ozawa, K., Miyajima, C., Igarashi, K., Itou, K., Takeda, K., and Itakura, F., "Driver identification using driving behavior signals," *IEICE TRANSACTIONS on Information and Systems*, Vol. 89, No. 3, 2006, pp. 1188–1194.

[8] Miyajima, C., Nishiwaki, Y., Ozawa, K., Wakita, T., Itou, K., Takeda, K., and Itakura, F., "Driver modeling based on driving behavior and its evaluation in driver identification," *Proceedings of the IEEE*, Vol. 95, No. 2, 2007, pp. 427–437.

[9] Enev, M., Takakuwa, A., Koscher, K., and Kohno, T., "Automobile driver fingerprinting," *Proceedings on Privacy Enhancing Technologies*, Vol. 2016, No. 1, 2016, pp. 34–50.

[10] Hallac, D., Sharang, A., Stahlmann, R., Lamprecht, A., Huber, M., Roehder, M., Leskovec, J., et al., "Driver identification using automobile sensor data from a single turn," *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, IEEE, 2016, pp. 953–958.

[11] Zhang, X., Zhao, X., and Rong, J., "A study of individual characteristics of driving behavior based on hidden markov model," *Sensors & Transducers*, Vol. 167, No. 3, 2014, p. 194.

[12] Booz, H., Allen, "Safety pilot model deployment - sample data environment," 2015.

[13] Bezzina, D., and Sayer, J., "Safety pilot model deployment: Test conductor team report," *Report No. DOT HS*, Vol. 812, 2014, p. 171.

[14] Huang, X., Zhao, D., and Peng, H., "Empirical study of dsrc performance based on safety pilot model deployment data," *parameters*, Vol. 12, 2017, p. 14.

[15] Newson, P., and Krumm, J., "Hidden Markov map matching through noise and sparseness," *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, ACM, 2009, pp. 336–343.

[16] Lee, K.-C., Ho, J., and Kriegman, D. J., "Acquiring linear subspaces for face recognition under variable lighting," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, , No. 5, 2005, pp. 684–698.

[17] Naseem, I., Togneri, R., and Bennamoun, M., "Linear regression for face recognition," *IEEE transactions on pattern analysis and machine intelligence*, Vol. 32, No. 11, 2010, pp. 2106–2112.

[18] Savas, B., and Eldén, L., "Handwritten digit classification using higher order singular value decomposition," *Pattern recognition*, Vol. 40, No. 3, 2007, pp. 993–1003.

[19] Asendorf, N., and Nadakuditi, R. R., "The performance of a matched subspace detector that uses subspaces estimated from finite, noisy, training data," *IEEE Transactions on Signal Processing*, Vol. 61, No. 8, 2013, pp. 1972–1985.

[20] Ho, T. K., "Random decision forests," *Document analysis and recognition, 1995., proceedings of the third international conference on*, Vol. 1, IEEE, 1995, pp. 278–282.

[21] Breiman, L., "Random forests," *Machine learning*, Vol. 45, No. 1, 2001, pp. 5–32.