A Joint Inverse Reinforcement Learning and Deep Learning Model for Drivers' Behavioral Prediction

Guojun Wu, Yanhua Li*
{gwu,yli15}@wpi.edu
Worcester Polytechnic Institute
Worcester, USA

Shikai Luo, Ge Song, Qichao Wang, Jing He, Jieping Ye, Xiaohu Qie and Hongtu Zhu†
{luoshikai,songge,davidwangqichao}@didiglobal.com
{hejing,yejieping,tiger.qie,zhuhongtu}@didiglobal.com
DiDi
Beijing, China

ABSTRACT

Users' behavioral predictions are crucially important for many domains including major e-commerce companies, ride-hailing platforms, social networking, and education. The success of such prediction strongly depends on the development of representation learning that can effectively model the dynamic evolution of user's behavior. This paper aims to develop a joint framework of combining inverse reinforcement learning (IRL) with deep learning (DL) regression model, called IRL-DL, to predict drivers' future behavior in ride-hailing platforms. Specifically, we formulate the dynamic evolution of each driver as a sequential decision-making problem and then employ IRL as representation learning to learn the preference vector of each driver. Then, we integrate drivers' preference vector with their static features (e.g., age, gender) and other attributes to build a regression model (e.g., LTSM-neural network) to predict drivers' future behavior. We use an extensive driver data set obtained from a ride-sharing platform to verify the effectiveness and efficiency of our IRL-DL framework, and results show that our IRL-DL framework can achieve consistent and remarkable improvements over models without drivers' preference vectors.

KEYWORDS

Drivers' Behavioral Prediction, Inverse Reinforcement Learning, User Modelling

ACM Reference Format:
Guojun Wu, Yanhua Li and Shikai Luo, Ge Song, Qichao Wang, Jing He, Jieping Ye, Xiaohu Qie and Hongtu Zhu. 2020. A Joint Inverse Reinforcement Learning and Deep Learning Model for Drivers' Behavioral Prediction. In *The 29th ACM International Conference on Information and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3340531.3412682

1 INTRODUCTION

Accurately predicting users' behavior based on their history and short-term features plays a critical role in many domains, such as major e-commerce companies, ride-hailing platform, social networking, and education. As an illustration, we consider an important problem of predicting drivers' behavior, such as working hours or income per day, in some ride-hailing platforms, such as Uber and Lyft. We call such a problem as **Drivers' Behavioral Prediction**. Solving such driver prediction problem usually requires the integration of drivers' history with some 'static' features including personal characteristics (e.g., age or gender), non-platform environment variables (e.g., government policy), and platform policies (e.g., dispatching policy) to predict drivers' future behaviors. Improving the accuracy of such prediction may allow the ride-hailing platform to achieve a healthy equilibrium between dynamic supply and demand systems.

However, most existing predictive models focus on the use of static features and some summary statistics of users' history, such as Recency, Frequency, and Monetary (RFM) value [3]. For instance, RFM analysis as a marketing technique has been widely used to analyze customer's behavior including how recently a customer has purchased (recency), how often the customer purchases (frequency), and how much the customer spends (monetary). It is beneficial to improve customer segmentation by dividing customers into various groups for future personalized services and for identifying customers who are more likely to respond to promotions [9]. However, those summary statistics may have minimal predictive power in some problems, such as driver's behavioral prediction. For instance, as shown in Table 2, the inclusion of RFM gains little improvement in prediction accuracy.

A significant challenge associated with moving beyond the simple summary statistics of users' history is how to use representation learning to learn an effective embedding vector of the evolution of users' properties. In many cases, such evolution can be very complicated due to its considerable heterogeneity across users and/or time intervals. For instance, in the driver's behavioral prediction problem, we consider a working cycle model such that each driver makes a sequence of decisions ordered by time, such as being idle, taking orders, and logging off every day. Even within each driver, such a sequence pattern may vary dramatically across days. Moreover, since the working cycle model is a sequential decision problem, it may be solved by using reinforcement learning (RL) [17]. Different from normal sequential problem, which can be solved using RNN or LSTM, the original idea of RL is to find an optimal policy mapping from states to actions to maximize user's long-term rewards, rather than learning his/her embedding vector. Instead, we consider Inverse Reinforcement Learning (IRL) algorithm for

CIKM '20, October 19–23, 2020, Virtual Event, Ireland

ACM ISBN 978-1-4503-6859-9/20/10…$15.00
https://doi.org/10.1145/3340531.3412682

learning the user's embedding vector based on his/her preference vector.

The IRL problems arise naturally when one is interested in predicting future behavior of agents based on the observations of his/her past behavior. The key idea of IRL is to find a reward function such that the distribution of state and action sequences under a (near-)optimal policy with respect to the reward function matches the demonstrated trajectories from an agent [13, 18]. Various estimation methods for IRL including apprenticeship learning [1], maximum entropy IRL [18], Bayesian IRL [14], relative entropy IRL [5], and maximum causal entropy IRL [4], have been developed in the literature. For instance, a broadly used solution to IRL problem [5] proposes a model-free method to find the policy, while neural-network-based reward function [7] can represent more complex expert behaviors. Moreover, IRL can also be viewed as a special case of Generative Adversarial Networks (GANs) [6, 8].

Motivated by solving the drivers' behavioral prediction problem, we develop a joint IRL-DL framework of integrating drivers' history and static features to predict drivers' future behavior. We use RL to formulate each driver's working cycle in a day sorted by time as a sequential decision-making process. The working cycle of each driver consists of different decisions when the driver maximizes his/her total inherent rewards weighted by income and other aspects. Then, we use IRL to learn each driver's decision-making preference vector. Finally, we combine each driver's preference vector with other attributes to build a deep learning regression model (e.g., LTSM-neural network) for prediction. Our contributions can be summarized as follows.

- We are the first to propose a general prediction framework of combining drivers' static features and decision-making preference vector together for prediction.
- By modeling a driver's daily working cycle process as a sequential decision-making problem, we use IRL to learn the driver's preference vector.
- We use a large-scale real-world data set obtained from a ride-sharing platform to show that the use of a preference vector can achieve up to 13% improvement than baseline models in terms of the prediction accuracy on different tasks.

The rest of this paper is organized as follows. We introduce the driver's behavioral prediction problem and provide a brief overview of data and the architecture of IRL-DL in Section 2. The data-preprocessing part is elaborated in Section 3. Section 4 is to model the driver's working cycle as a Markov Decision Problem, and in Section 5, we show how to learn a driver's preference vector. Details of experiments and evaluations are given in Section 6, followed by a brief discussion of related works in Section 7. We conclude our paper in Section 8.

2 OVERVIEW

In this section, we introduce the drivers' behavioral prediction problem, provide a description of the data set we use, and outline the IRL-DL framework.

2.1 Drivers' Behavioral Prediction Problem

Definition 1 (Working Cycle). *Each driver in a ride-hailing platform can decide when to start to work and when to stop working every day. We define a complete working cycle of every driver at a given day as the time interval from the starting time, denoted as $Time_{in}$, to the ending time, denoted as $Time_{off}$.*

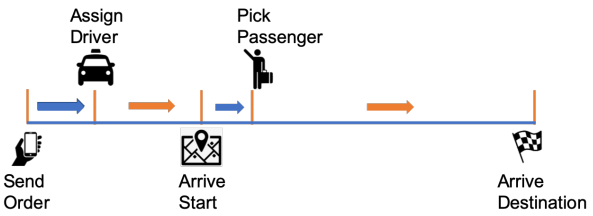

Figure 1: Illustration of an order being processed.

We introduce an *inter-cycle trajectory* as a working static sequence of multiple working cycles. The inter-cycle trajectory is used to model the long-term behaviors of a driver.

Definition 2 (Inter-cycle trajectory). *The inter-cycle trajectory of a driver is an ordered sequence, denoted as $Tr_{inter} = (W_1, \ldots, W_N)$, where N is the total number of working cycles and $W_i = \{t_{start,i}, t_{end,i}, f_{W_i}\}$ represents the $i-th$ working cycle, in which $t_{start,i}$ is the start time of W_i, $t_{end,i}$ is the end time of W_i, and f_{W_i} is the feature vector of W_i for $i = 1, \ldots, N$.*

Furthermore, we introduce an *intra-cycle trajectory* consisting of a sequence of all orders that each driver finishes within the same working cycle. We use intra-cycle trajectories to characterize the short-term behaviors of a driver.

Definition 3 (Intra-cycle trajectory). *The intra-cycle trajectory of a driver's $i-th$ working cycle is an ordered sequence of*

$$Tr_{i,intra} = (O_{i,1}, O_{i,2}, \ldots, O_{i,N_i}),$$

where N_i is the total number of orders in the $i-th$ working cycle, and $O_{i,k} = \{t_{k;start,i}, t_{k;end,i}, f_{O_{i,k}}\}$ represents an order with $t_{k;start,i}$ being the start time of order $O_{i,k}$, $t_{k;end,i}$ being the end time of $O_{i,k}$, and $f_{O_{i,k}}$ being the feature vector of $O_{i,k}$ for $k = 1, \ldots, N_i$.

We use X to denote the vector of all features other than intra-cycle and inter-cycle trajectories. We now formally define the drivers' behavioral prediction problem.

Drivers' Behavioral Prediction Problem Definition Given a driver's other features in X, intra-cycle trajectory TR_{intra}, and inter-cycle trajectory TR_{inter}, we want to predict his/her future behavior (e.g., total online time, income, or finished orders), denoted as G, in the next period T.

2.2 Data Description

We use four data types in the data set obtained from a ride-hailing platform including (i) order data; (ii) finance data; (iii) customer service worksheet data; and (iv) driver log data. For consistency, all these data types are aligned with the same time period starting from March 2018 to July 2018.

Order Data. Figure 1 shows a typical order in a ride-hailing platform consisting of three key stages including pick-up stage(driving to the origin), waiting stage(waiting for the passenger), and serving stage(sending the passenger to the destination).

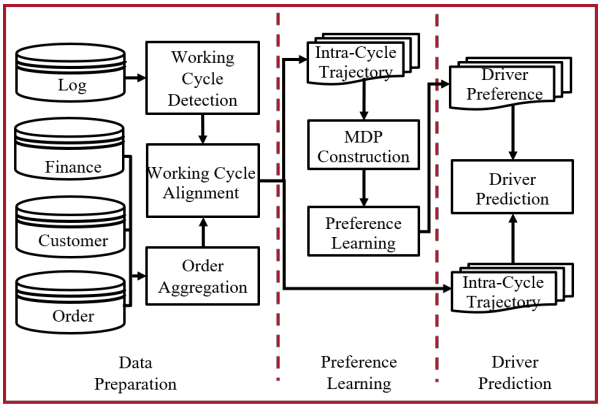

Figure 2: The IRL-DL framework for the driver prediction problem

Finance Data. We consider various charging variables of each order including customer's payment, driver's cut, driver's incentives, time rate, distance rate, and sometimes long-distance rate.

Customer Service Worksheet Data. We extract some variables from customer service worksheet, such as the number of complaints of each driver received from passengers since drivers with more complaints are more likely to leave the platform.

Driver Log Data. We extract the inter- and intra-cycle trajectories of each driver.

2.3 IRL-DL Framework

Figure 2 provides an overview of our proposed IRL-DL framework consisting of three main components including data preprocessing, preference learning, and driver prediction.

- **Data Preprocessing.** We first extract drivers' working cycles from their log data. Then, we align order data, finance data, and customer service data together at the order level. Subsequently, we generate the intra-cycle trajectories, inter-cycle trajectories, and additional features of all drivers.
- **Preference Learning.** For each driver, we model the whole working cycle as a Markov Decision Process (MDP) and use IRL to learn a reward function that a driver uses to make various decisions based on intra-cycle trajectories. The reward function represents the preference function that each driver would decide to keep working or log off based on current and future expected rewards.
- **Driver Prediction.** We integrate the preference vector of each driver with all other features to build a predictive model to predict a driver's status.

3 DATA PREPROCESSING

In this section, we introduce three key preprocessing steps to generate various features and trajectories.

3.1 Working Cycle Detection

Actions of each driver can be viewed as a sequence of actions ordered by time. Table 1 illustrates an example of one driver's action log.

Table 1: Example of Driver Log

Driver	Time	Action
0001	2018-03-20 08:00:35	Log in
0001	2018-03-20 08:03:04	Assigned Order
0001	2018-03-20 08:45:13	Finish Order
...	...	...
0001	2018-03-20 18:54:29	Log off

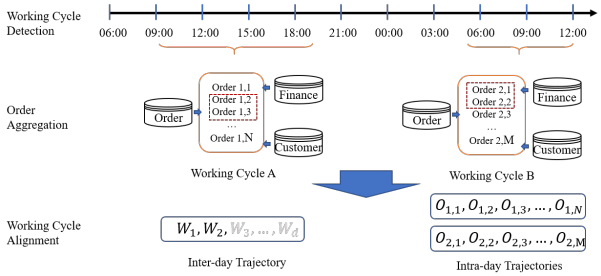

Figure 3: An illustration of data preparation process. The process contains 3 parts. The upper time axis indicates two consecutive days. There are two valid log-in/log-off pairs. The first one is from 9am to 19pm in the first day and the second pair is from 5am to 12am in the second day. Then, we generate features for each order and aggregate them by working cycle. The outputs of this process contain both inter-day and intra-day trajectories. For intra-day trajectories, we have 2 trajectories since we have two working cycles.

The working cycle should start with a log-in action and end with the following log-off action. We show an example in Figure 3. As we have stated, we split those working cycles based on the driver's log-in/log-off actions.

3.2 Order Aggregation

In the order aggregation process, we mainly have two tasks. The first task is to aggregate features of the same order sitting in different data sources. The second task is to aggregate ExpressPool orders based on which trip they belong to. ExpressPool is a carpool product, whose details will be given later.

Features of one order are in different data sources. Features, such as pick-up distance, driving distance, and other trip-related features are in the ordered dataset. Financial features also include subsidies of drivers in the financial dataset, and one order may have multiple records for different subsidies, such as passenger's coupon and driver's incentive. Other features including customer service features like rating and complaints are in a separate data source, and we align them using order IDs. For some complaints without a particular order ID, we use driver ID and passenger ID to identify, which order those complaints belong to.

The second task is to process ExpressPool orders. Each passenger would pay fixed fees based on the ExpressPool pricing model. As for the driver, he/she would still get his/her share based on the distance and time he/she drove. In this case, the income of the driver was calculated based on the travel distance and travel time from picking up the first passenger to dropping off the last passenger. We then

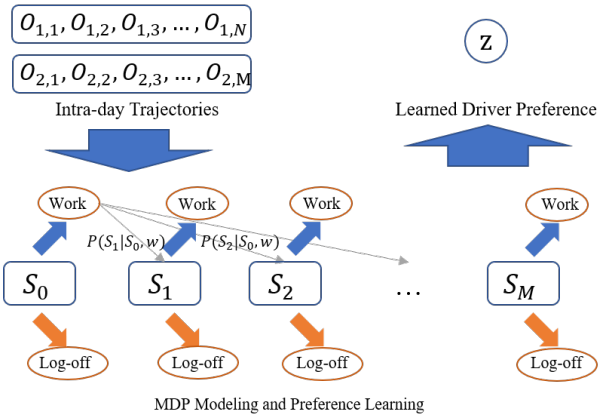

Figure 4: An illustration of MDP model for driver working problems: $s_0, \ldots, s_M$ are states and each state has two actions including work and log-off. We use intra-day trajectories to build MDP and learn the preference of each driver using IRL. The output of this process is a preference vector Z for each driver.

should aggregate ExpressPool orders to their corresponding trips. Some features, such as subsidy and income can be added together. For example, in Working Cycle A of Figure 3, orders 2 and 3 are ExpressPool Orders, so we have to aggregate their features together.

3.3 Working Cycle Alignment

Finally, we can then use working cycles and aggregated orders to generate both intra-cycle and inter-cycle trajectories. For intra-cycle trajectories, we fit aggregated order data into different working cycles of each driver and then sort all orders by using its assignment time. For inter-cycle trajectories, we can calculate features for each working cycle. The features that we used in intra-cycle and inter-cycle trajectories include *Working Status* such as **Total Finished Orders** and **Last Order Feature** like the total income of last order. Just like the example in Figure 3, for the two working cycles, we can generate one intra-day trajectory for each working cycle and one inter-day trajectory for all cycles.

4 MDP FOR WORKING CYCLE MODELING

We explain how to use MDP to model the decision-making processes of a driver's working cycle.

We regard each driver as an independent "agent" since each driver has his/her own preference. Each driver uses individual preference to evaluate various decision-making features associated with the current state and each possible action, such as income, working hours, and personal experience [10]. The reward function indicates the driver's preference for different features, and each driver makes his/her own decision to maximize a total inherent reward during the whole working cycle other than only maximizing his/her real income. For example, if a driver could have an unpleasant day, then he/she might decide to stop working. Thus, for long term reward maximization problem, it is natural to use MDP to model driver's decision-making process. Below, we explain how to build an MDP model based on the preprocessed data in Section 3.

Agent: We use each driver as a single agent and then collect a certain amount of intra-cycle trajectories, e.g., 3-month data. However, for some part-time drivers, we may not have enough trajectories to learn a robust preference function, so we collect more trajectories until there are at least N intra-cycle trajectories. We set N to be 30, and thus we have at least 30 trajectories for each driver. However, for simplicity, we only draw two trajectories in Figure 4.

Action set A**:** Let T_m be a predefined time interval associated with S_m for $m = 0, 1, \ldots, M$. As illustrated in Figure 4, a driver always has two actions during the idle stage including continuing to work and logging off, forming an action set A in our problem. For logging off, we omit temporal log-offs. Naturally, one intra-cycle trajectory would have one log-off action in the end and multiple working actions.

State set S**:** As shown in Figure 4, we divide each day into fixed-length time slots, for example, 10 minutes per time slot. Thus, we have $1,440$ states in total and $M = 1440$. However, since drivers usually have their own pattern and they can work at most 12 hours per working cycle, the state space is not very large.

Transition probability function $P : S \times A \times S \to [0, 1]$. There are two actions in A for each driver. If a driver would decide to log off, then the driver could finish the trajectory. If the driver would continue to work, then it could lead to different result states. Specifically, if the platform does not assign any order to the driver, then it would result in an idle state. Otherwise, the driver transfers to the state represented by the end time of an order. Thus, the transition probability contains two parts including the probability to be assigned an order, denoted as $P_o(s)$, and the distribution of driving time, denoted as $P_d(s'|s, a)$. By combining those two probabilities together, we have $P(s'|s, a) = \{1 - P_o(s)\} + P_o(s)P_d(s'|s, a)$. For example, in Figure 4, if the driver logs in at state S_0 and does not get any order assigned to him or her, then the driver will transit to state S_1. Otherwise, the driver will transit to S_2 or other states based on how it takes the driver to finish the order. However, for some complicated dispatching strategy, $P(s'|s, a)$ may be different across subjects, which makes it very difficult to estimate $P_o(s)$ and $P_d(s'|s, a)$. Therefore, we employ a model-free method to learn the reward function elaborated in Section 5.

Reward R**:** When drivers make decisions on whether to log off, they consider various decision-making features, such as working hours, income, and experience. We will give these decision-making features in the following subsection for more details.

In MDP, $R : S \times A \to \mathbb{R}$ captures the unique personal preference of an agent, which maps decision-making features (at a state s while taking action a) to reward value. These decision-making features include working hours, income, and experience, among others. Such reward function $R(s, a)$ can be inversely learned from intra-cycle trajectory data. In our problem, we assume $R(s, a_{off}) = 0$, indicating that the immediate reward is 0 when a driver decides to log off.

5 IRL-DL

In this section, we introduce IRL, describe how to learn driver's preference based on the MDP of driver's working cycle, and integrate driver's preference with other attributes to predict driver's future behavior.

5.1 Inverse Reinforcement Learning

Inverse Reinforcement Learning (IRL) has been widely used to learn a reward function $R(s,a)$ of an MDP in the past decade. The IRL is to find a reward function $R(s,a)$ such that the distribution of action and state sequences under a (near-)optimal policy with respect to $R(s,a)$ matches with the demonstrated trajectories observed from an agent [13, 18]. A broadly used solution to IRL problem [5] proposes a model-free method to find the policy, which best represents demonstrated behaviors with the highest entropy, subject to the constraint of matching feature expectations to the distribution of demonstrated trajectories. The reward function $R(s,a)$ is assumed to be a linear function of observed feature $f(s,a)$ at state-action pair (s,a), that is $R(s,a) = \theta^T \cdot f(s,a)$, where θ is a preference vector. Thus, the reward of a trajectory is given by $R(\zeta) = \theta^T f(\zeta)$, where ζ represents a trajectory from MDP and $f(\zeta) = \sum_{(s,a)\in\zeta} f(s,a)$. Then, if some trajectories are collected, we have $\sum_{\zeta\in TR} P(\zeta) f(\zeta) = \widetilde{f}$, where TR is the set of all possible trajectories, $P(\zeta)$ is the probability of ζ being generated by the MDP and $\widetilde{f}$ is an empirical feature expectation based on collected trajectories. We can then calculate the preference vector θ by solving a maximum entropy problem and have $P(\zeta) = e^{\theta^T f(\zeta)} / \sum_{\zeta\in TR} e^{\theta^T f(\zeta)}$. We may calculate the maximum likelihood estimate of θ by using standard gradient decent method with the gradient given by $\widetilde{f} - \sum_{\zeta\in\widetilde{TR}} P(\zeta) f(\zeta)$, where $\widetilde{TR}$ is the set of observed drivers' trajectories.

5.2 Driver's Preference Learning

In our problem, $\widetilde{f}$ and $f(\zeta)$ can be easily calculated from observed data. However, it remains an open question how to calculate the probability of trajectory being generated by the MDP constructed by using method mentioned in Section 4, namely $P(\zeta)$. From the definition of $P(\zeta)$, we know that,$P(\zeta) = P_0(s_0) \sum_{t=1}^{T} \pi(s_t,a_t) P(s_{t+1}|s_t,a_t)$, where $P_0(s_0)$ is the initial start distribution, $\pi(s_t,a_t)$ is the probability from policy, and $P(s_{t+1}|s_t,a_t)$ is the transition probability, in which s_{t+1} is the next state in ζ. It follows that the transition probability contains two parts including $P_o(s_t)$ and $P_d(s_{t+1}|s_t,a_t)$.

However, it is difficult to estimate the two parts of the transition probability by using observed data. The travel time distribution $P_d(s_{t+1}|s_t,a_t)$ is long-tailed and quite wide due to the presence of occasionally long orders, such as orders to airport or even sometimes to another city. It brings difficulties that wide distribution would increase computational complexity and the long-tailed part of $P_d(s_{t+1}|s_t,a_t)$ would not be robust to noise. Moreover, the probability of getting an order $P_o(s_t)$ is more difficult to estimate due to dispatching strategy and environment. Thus, model-based methods based on transition probability do not work well for our problem.

We employ Relative Entropy Inverse Reinforcement Learning (REIRL) to learn the preference vector. The REIRL is a model-free method and uses importance sampling to estimate $F = \sum_{\zeta\in\widetilde{TR}} P(\zeta) f(\zeta)$ as follows:

$$F = \sum_{\zeta\in\widetilde{TR}} \frac{U(\zeta)\phi(\zeta)^{-1} e^{\theta^T f(\zeta)}}{\sum_{\zeta'\in\widetilde{TR}} U(\zeta')\phi(\zeta')^{-1} e^{\theta^T f(\zeta')}} f(\zeta), \tag{1}$$

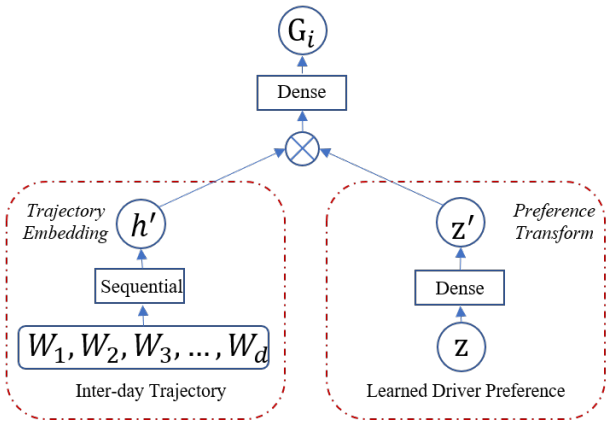

Figure 5: Illustration of the network structure of IRL-DL. The left component takes inter-day trajectory as input to feed into a sequential layer and the output is r h'. The right part takes driver preference Z as input and the output is denoted as z'.

where $U(\zeta)$ is a uniform distribution and $\phi(\zeta)$ is the trajectory distribution from a sample policy ϕ, where $\phi(\zeta) = \Pi_{(s,a)\in\zeta}\phi(s,a)$. The brief derivation and the proof of (1) can be found in [5].

The gradient $\widetilde{f} - \sum_{\zeta\in\widetilde{TR}} P(\zeta) f(\zeta)$ can be estimated by

$$\begin{aligned}\nabla_\theta L(\theta) =& \widetilde{f} - \sum_{\zeta\in\widetilde{TR}} P(\zeta) f(\zeta) \\ =& \widetilde{f} - \sum_{\zeta\in\widetilde{TR}} \frac{U(\zeta)\phi(\zeta)^{-1} e^{\theta^T f(\zeta)}}{\sum_{\zeta'\in\widetilde{TR}} U(\zeta')\phi(\zeta')^{-1} e^{\theta^T f(\zeta')}} f(\zeta).\end{aligned} \tag{2}$$

Applying the IRL algorithm, we can learn the preference vector θ.

To make preferences across models of different drivers comparable, we use *z-score* [11] normalization. Higher/lower z-score on the k-th feature, denoted as z_{ik}, means that driver i has strong preference on the k-th feature. The use of z-score in our problem has two major advantages. First, we can compare different drivers' preferences via their z-scores. Second, the elements of z_i would most likely lie between $[-3,3]$, but those of θ_i could be any real value. This shows the advantage of training neural networks. The z-score can be calculated as $z_i = \text{diag}(H_i^{-\frac{1}{2}}) \odot \theta_i$, where i indicates the i-th driver, H_i^{-1} is the inverse of Hessian Matrix of θ_i, diag$(\cdot)$ means diagonal elements of a matrix, and we use $\odot$ to denote element-wise multiplication. Furthermore, we have

$$\begin{aligned}H_i =& \frac{\partial^2 L(\theta_i)}{\partial^2 \theta_i} \\ =& \left[\sum_{\zeta} P(\zeta) f_i(\zeta) \left(f_j(\zeta) - \sum_{\zeta'} P(\zeta') f_j(\zeta')\right)\right],\end{aligned} \tag{3}$$

where $f_i(\zeta)$ and $f_j(\zeta)$ means the i-th and j-th elements of $f(\zeta)$, respectively. And $P(\zeta)$ can be estimated using importance sampling. After acquiring the z-score z_i of each driver, we can then build up a model to predict drivers' future behavior.

Algorithm 1 Driver Prediction

1: **INPUT:** Driver ID D_i;
2: **OUTPUT:** Some predicted status of D_i, denoted as G_i;
3: Collect intra-cycle trajectory set TR_{intra} of D_i;
4: Build Markov Decision Process using TR_{intra};
5: Learn the preference vector θ_i via Relative Entropy Inverse Reinforcement Learning;
6: Extract inter-cycle trajectory set TR_{inter} for driver D_i's historical behaviors;
7: Use trained PhasedLSTM model to predict G_i based on the input of θ_i and TR_{inter} using the LatentCross Framework in Figure 5;
8: Return G_i

5.3 Regression Models for Drivers' Behavioral Prediction

We elaborate more on regression models and features that we use to predict drivers' behavior, such as total online time, income and finished orders. Figure 5 illustrates the general LatentCross [2] framework of the neural network for doing driver's prediction. It is the Driver Prediction part in Figure 2. The first component is the trajectory embedding model, which in our case we use PhasedLSTM [12] as an example, but any model dealing with sequential data can be plugged in the sequential layer. As shown in Figures 2 and 5, this part takes Inter-day trajectories, which are the output of data preparation process demonstrated in Figure 3, as input and produces an trajectory embedding. The right component of Figure 2 is to take the driver preference that we learn in the preference learning phase as input. Then, we apply several dense layers to add non-linearity. As stated above, the input is z_i rather than the raw preference vector θ_i. Then, we use the element-wise product layer to combine trajectory embedding and driver's preference, which is the same as LatentCross, in order to capture two-way relationship between the hidden state and each context feature [2]. Finally, a dense layer is added to produce final prediction.

Algorithm 1 shows how to predict a driver's behavior for a trained prediction model. First, we collect intra-cycle trajectories from multiple data sources that we described in Section 2. Then, we construct a working-cycle MDP based on the observed intra-cycle trajectories and we can use MDP and IRL to learn driver's preference vector. Finally, we combine the inter-cycle trajectory, driver's preference vector, and other attributes to predict driver's behavior. Furthermore, for a new driver, without historical behavior data, we return the mean prediction of all drivers as predictive value.

6 EVALUATION

In this section, we use extensive experiments to verify the effectiveness and efficiency of our proposed IRL-DL for drivers' income.

6.1 Experiments Setting

We extracted data from 58, 160 drivers starting from March, 2018 to July, 2018 in a city. Data from March to May were used to train, and data in June and July were used as test data. First, we collected drivers' two-month intra-cycle trajectories to learn drivers' preference and then used 30-day long inter-cycle trajectories and drivers' preference to build prediction models, in which the target value is driver's total income in the next 30 days. To predict income, we use driver's total income in the 30 days of July as the ground truth. Then, intra-cycle trajectories generated in June were used to learn driver's individual preference and the prediction was made based on driver's preference and 30-day long inter-cycle trajectories. Since the accurate prediction of when a driver would log off indicates that the learned preference vector may indicate how drivers make his/her log-off decision, we used AUC score to evaluate the accuracy of log-off action prediction accuracy for preference learning.

As for driver's income prediction, Mean Absolute Error (MAE) is used as the evaluation metric. For information safety, we normalize the actual response by dividing by their average. We compare our model with three sets of baseline models.

- **Shallow Models.** We compare our method with two types of models. The first one is the BG-NBD model, in which we denote drivers' working days as "purchase" and their earnings as "monetary value" and build a RFM model to do drivers' behavioral prediction. Moreover, instead of using LSTM and neural network, we generate statistical features in various time intervals and feed those features into a shallow model, such as XGBoost, to predict each driver's income.
- **Models Without Driver Preference.** In those models, we omit driver's preference and only use driver's historical behaviors to predict driver's income. We also use RFM features extracted from the BG-NBD model to train those models.
- **Trajectory Embedding Component.** We also compare multiple sequential models including RNN, LSTM, and Phased-LSTM. Moreover, any other sequential model can be added into our IRL-DL framework since the goal of this paper is that the inclusion of additional decision-making preference can be combined with any trajectory embedding models in order to achieve better prediction accuracy.

6.2 Preference Learning

We evaluate the effectiveness of our preference learning model. Figure 6 shows the distribution of AUC scores with the y-axis being the number of driver. It indicates that for most drivers, preference vector learned can accurately predict whether a driver would log-off. However, for a few number of drivers, we do not have accurate prediction due to the lack of sufficient records. See Figure 7 for the relationship between AUC score and the size of records. Each point in Figure 7 indicates one individual driver. We also verify that the preference vector we learn is correlated with driver's future behavior. Specifically, we use the K-Means algorithm to split drivers into five different clusters based on their preferences. Figure 8 shows the distribution of future online time in the next 30 days for each cluster, revealing that different clusters correspond to different online time distributions. It may further indicate that drivers' preferences define their working patterns, contributing to their future performance. In summary, driver's preference is accurate and of great potential value for the drivers' behavioral prediction problem.

6.3 Drivers' Behavioral Prediction

We conduct experiments to show that our proposed framework can increase the prediction performance in a variety of tasks.

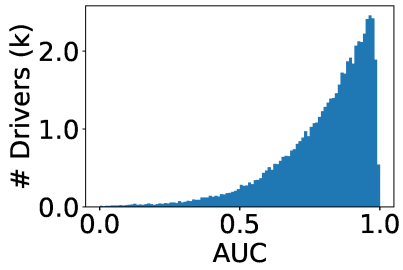

Figure 6: Number of drivers vs AUC score.

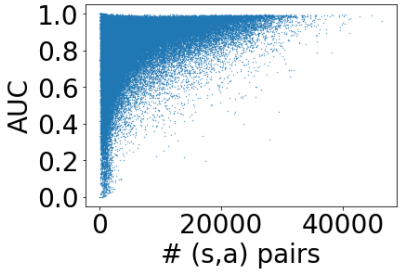

Figure 7: AUC score vs number of (s, a) pairs collected

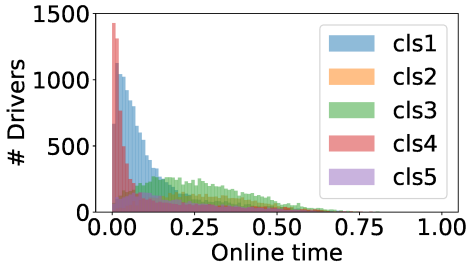

Figure 8: Online time distribution of different group of driver

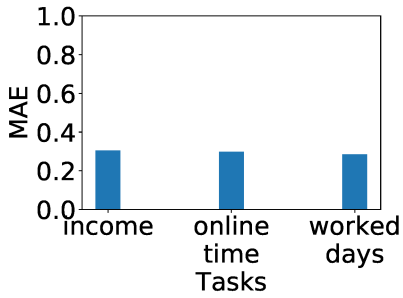

Figure 9: MAE on different tasks.

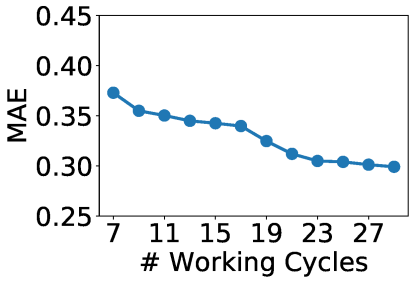

Figure 10: MAE vs amount of data.

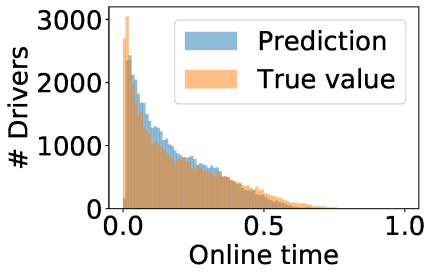

Figure 11: Prediction distribution of Task A.

Table 2: Online time prediction error (MAE)

	w/o RFM w/o Pref.	w RFM w/o Pref.	w/o RFM w Pref.	w RFM w Pref.
BG-NBD	N/A	0.5953	N/A	N/A
XGB	0.4532	0.4166	0.3898	0.3890
RNN-30	0.3883	0.3703	0.3258	0.3257
RNN-60	0.3681	0.3642	0.3146	0.3140
LSTM-30	0.3574	0.3510	0.3105	0.3103
LSTM-60	0.3430	0.3413	0.2957	0.2956
PLSTM-30	0.3502	0.3419	0.3004	0.2993
PLSTM-60	0.3483	0.3371	0.2933	0.2930

Algorithm Efficiency. We show the computational effectiveness of our proposed framework. Table 2 shows 29 different models that we used. The first one is the BG-NBD model, which has the poorest performance. The second four models are based on XGBoost. We also consider RNN, LSTM and PhasedLSTM models with different numbers of output size in the trajectory embedding layer as well as the output dimension of preference layer.

We have the following observations. First, the use of driver's preference can reduce prediction error for more than 10%. Second, deep models, such as RNN and LSTM, outperform the XGBoost model. Third, the benefit of increasing network complexity from 30 to 60 is not significant enough. Therefore, without loss generality, we fix output size at 30. Fourth, the RNN model without driver preference can outperform XGBoost even with the use of driver preference. Fifth, among all models, PhasedLSTM with the use of driver preference can achieve the best performance since PhasedLSTM can capture the long-term effect of temporal sequences. Therefore, PhasedLSTM would be used as a default option in the following evaluation. Also, for shallow models, RFM features can slightly improve prediction accuracy solely. Also, for deep models like LSTM and PhasedLSTM, the contribution of using recency, frequency and monetary values is pretty marginal, since the RFM related pattern can be learned by sequential models.

Different Tasks. We consider additional tasks including predicting total online time, driver's average income per day, and driver's worked days in the next month. The results in Figure 9 indicate that our model performs well in all tasks. Therefore, preference vector may represent driver's behavior in different aspects.

Amount of Data. We investigate the amount of data that we need. In this experiment, we choose drivers' total online time in the following 30 days as target response. The result in Figure 10 shows the prediction error (MAE) versus the number of working cycles that we used to train the model. The results indicate that MAE is essentially the same even with more than 23 working cycles. We can also reduce the randomness of data by including more observations. This randomness may come from two resources. The first one is the environment randomness and the second one is that drivers may exhibit certain randomness.

Prediction Illustration. We choose one task to illustrate the prediction result obtained from the proposed framework. Figure 11 presents the actual distribution of driver's online time and the predictive distribution. We have the following observations. First, the two distributions are very similar to each other. Specifically, the KL-divergence of those two distributions is equal to 0.1503. However, if we aggregate all drivers together, then their behaviors would be more predictable and the prediction performance is better. Second, the true distribution of A is more long-tailed compared with the prediction distribution of A. It indicates that our prediction

model may make a few conservative predictions since the long-tailed part is often caused by rare cases.

7 RELATED WORK

In this section, we summarize the literature works in two related areas to our study: 1) inverse reinforcement learning, and 2) user choice modeling. Learning reward function $R(s, a)$ of an MDP is a problem which has been broadly studied in the past decade, which is called Inverse Reinforcement Learning. The inverse reinforcement learning problem (IRL) is to find a reward function $R(s, a)$, such that the distribution of action and state sequences under a (near-)optimal policy with respect to $R(s, a)$ matches the demonstrated trajectories from an agent [13, 18]. A broadly used solution to IRL problem [5] proposes a model-free method to find the policy. Besides, neural-network-based reward function [7] is considered, which can represents more complex expert behaviors. And the Inverse Reinforcement Learning can be viewed as a special application of GAN [6, 8]. Also, IRL is used to model human's sequential decision-making process and is used to predict human behaviors in urban transit system [16]. Inspiring by these work, we also use Inverse Reinforcement Learning algorithm to extract driver's preference vector. User choice modeling has been extensively studied in the literature with applications, which investigate how users make decisions in various application scenarios. For examples, in [15], they use random utility maximization and random regret minimization to analyze users' choice on park-and-ride lots. In [18], the authors propose a probabilistic approach to discover reward function for which a near-optimal policy closely mimics observed behaviors. However, differing from these works, we employ data-driven approaches to study the unique decision-making process of urban public transit passengers.

8 CONCLUSION

In this paper, we introduce a novel drivers' behavioral prediction framework that can make accurate prediction of driver's future behavior. In this framework, we use driver's historical intra-cycle behaviors to learn driver's preference on his/her decision-making process and then combine the learned preference with inter-cycle features to predict driver's behavior in the future. Our extensive evaluation results based real-world data sets demonstrate that our framework can achieve the prediction accuracy of $MAE = 0.29$, which is on average 13% lower than existing state-of-the-art approaches without using driver's preference. We believe that the idea can benefit domains where both micro (i.e., intra-cycle) and macro (i.e., inter-cycle) decision making process are involved. For example, e-commerce platforms, like Amazon, Alibaba and etc, may apply this methodology to improve the prediction of customers' life time value. This paper gives a simple framework to capture customers' intrinsic preference from the decision making process (search, click, add to cart, ..., search, etc) of each visit, and to combine the learned preference with features aggregated (LSTM, etc) from recent visits.

9 DEPLOYMENT

The driver behavioral prediction algorithm has already been deployed in the production system, used in various applications. For example, if a driver is predicted to be less effective in the next 30 days (less finished orders), we can define a user-specific incentive strategy to increase his willingness to work, based on certain root cause analysis on this driver's recent behavior. We also rely on the prediction system to help evaluate a new pricing strategy, when randomized controlled experiments are restricted by law in the scenario. We apply the new pricing strategy on all the drivers for a couple of days, and the difference between the new pricing strategy versus the old is simply the mean of the observed metrics minus that of the predicted metrics.

The driver behavioral prediction system protects the privacy rights of drivers from three aspects. 1) We avoid using any personally identifiable feature, like social security number, in any of our algorithm. Only features like age or gender that may be attributable to more than one individual are included in the static features of our prediction algorithm. 2) The data are only available to a small group of people who are working on this project, and any one sharing drivers' personal information will break the law. 3) The behavior prediction system only returns the prediction results and related diagnostic information to related users of the system.

REFERENCES

[1] Pieter Abbeel and Andrew Y Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *ICML*. 1.

[2] Alex Beutel, Paul Covington, Sagar Jain, Can Xu, Li Jia, Vince Gatto, and Ed H. Chi. 2018. Latent Cross: Making Use of Context in Recurrent Recommender Systems. In *Eleventh Acm International Conference*.

[3] Derya Birant. 2011. Data mining using RFM analysis. In *Knowledge-oriented applications in data mining*. IntechOpen.

[4] Michael Bloem and Nicholas Bambos. 2014. Infinite time horizon maximum causal entropy inverse reinforcement learning. In *53rd IEEE Conference on Decision and Control*. IEEE, 4911–4916.

[5] Abdeslam Boularias, Jens Kober, and Jan Peters. 2011. Relative entropy inverse reinforcement learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. 182–189.

[6] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. 2016. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852* (2016).

[7] Chelsea Finn, Sergey Levine, and Pieter Abbeel. 2016. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*. 49–58.

[8] Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*. 4565–4573.

[9] Su-Yeon Kim, Tae-Soo Jung, Eui-Ho Suh, and Hyun-Seok Hwang. 2006. Customer segmentation and strategy development based on customer lifetime value: A case study. *Expert systems with applications* 31, 1 (2006), 101–107.

[10] Ugo Lachapelle, Larry Frank, Brian E Saelens, James F Sallis, and Terry L Conway. 2011. Commuting by public transit and physical activity: where you live, where you work, and how you get there. *Journal of Physical Activity and Health* (2011).

[11] Erich L Lehmann and George Casella. 2006. *Theory of point estimation*. Springer Science & Business Media.

[12] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. 2016. Phased lstm: Accelerating recurrent network training for long or event-based sequences. In *Advances in neural information processing systems*. 3882–3890.

[13] Andrew Y Ng, Stuart J Russell, et al. 2000. Algorithms for inverse reinforcement learning.. In *ICML*.

[14] Deepak Ramachandran and Eyal Amir. 2007. Bayesian Inverse Reinforcement Learning.. In *29th International Joint Conferences on Artificial Intelligence*, Vol. 7. 2586–2591.

[15] Bibhuti Sharma, Mark Hickman, and Neema Nassir. 2017. Park-and-ride lot choice model using random utility maximization and random regret minimization. *Transportation* (2017).

[16] Pengfei Wang, Yanjie Fu, Guannan Liu, Wenqing Hu, and Charu Aggarwal. 2017. Human mobility synchronization and trip purpose detection with mixture of hawkes processes. In *KDD*.

[17] Jiangchuan Zheng and Lionel M Ni. 2014. Modeling heterogeneous routing decisions in trajectories for driving experience learning. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM, 951–961.

[18] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. 2008. Maximum Entropy Inverse Reinforcement Learning.. In *AAAI*.