Generative Adversarial Reward Learning for Generalized Behavior Tendency Inference

Xiaocong Chen, Lina Yao, Senior Member, IEEE, Xianzhi Wang, Member, IEEE, Aixin Sun, Member, IEEE, and Quan Z. Sheng, Member, IEEE

Abstract—Recent advances in reinforcement learning have inspired increasing interest in learning user modeling adaptively through dynamic interactions, e.g., in reinforcement learning based recommender systems. In most reinforcement learning applications, reward functions provide the critical guideline for optimization. However, current reinforcement learning-based methods rely on manually-defined reward functions, which cannot adapt to dynamic, noisy environments. Moreover, they generally use task-specific reward functions that sacrifice generalization ability. We propose a generative inverse reinforcement learning for user behavioral preference modeling to address the above issues. Instead of using predefined reward functions, our model can automatically learn the rewards from user’s actions based on discriminative actor-critic network and Wasserstein GAN. Our model provides a general approach to characterizing and explaining underlying behavioral tendencies. Our experiments show our method outperforms state-of-the-art methods in several scenarios, namely traffic signal control, online recommender systems, and scanpath prediction.

Index Terms—Inverse reinforcement learning, behavioral tendency modeling, adversarial training, generative model

1 INTRODUCTION

Behavior modeling provides a footprint about user’s behaviors and preferences. It is a cornerstone of diverse downstream applications that support personalized services and predictive decision-making, such as human-robot interactions, recommender systems, and intelligent transportation systems. Recommender systems generally use user’s past activities to predict their future interest [1], [2], [3], and past studies integrate demographic information with user’s long-term interest on personalized tasks [4], [5], [6], [7]. In human-robot interaction, a robot learns from user behaviors to predict user’s activities and provide necessary support [8]. Multimodal probabilistic models [9] and teacher-student network [10] are often used to predict user’s intention for traffic prediction or object segmentation.

Traditional methods learn static behavioral tendencies via modeling user’s historical activities with items as a feature space [11] or a user-item matrix [12]. In contrast, reinforcement learning shows advantages in learning user’s preference or behavioral tendency through dynamic interactions between agent and the environment. It has attracted lots of research interests in recommendation systems [6], intention prediction [13], traffic control [14], and human-robot interaction domains [15]. Reinforcement learning covers several categories of methods, such as value-based methods, policy-based methods, and hybrid methods. All these methods use the accumulated reward during a long term to indicate user’s activities. The reward function is manually defined and requires extensive effort to contemplate potential factors [16], [17].

In general, user’s activities are noisy, occasionally contaminated by imperfect user behaviors, and thus may not always reveal user’s interest or intention. For example, in online shopping, a user may follow a clear logic to buy items and randomly add additional items because of promotions or discounts. This makes it difficult to define an accurate reward function because the noises also affect the fulfillment of task goals in reinforcement learning. Another challenge lies in the common practice of adding task-specific terms to the reward function to cope with different tasks.

Current studies usually require manually adjusting the reward function to model user’s profiles [2], [18], [19]. Manual adjustment tends to produce imperfect results because it is unrealistic to consider all reward function possibilities, not to mention designing reward functions for new tasks.

A better way to determine the reward function is to learn it automatically through dynamic agent-environment interactions. Inverse reinforcement learning recently emerged as an appealing solution, which learns reward function learning from demonstrations in a few scenarios [20].

It faces two challenges for user behavior modeling. First, it requires a repeated, computational expensive reinforcement learning process to apply a learned reward function [21]; second, given an expert policy, there could be countless
reward functions for choice, making the selection of reward function difficult and the optimization computationally expensive. The only recommendation model [22] that adopts improved inverse reinforcement learning simply skips the repeated reinforcement learning process. Thus, it is hard to converge due to the lack of sampling efficiency and training stability.

Furthermore, the model only works for recommender systems and lacks generalization ability.

Manually designed reward functions have less feasibility and generalizability in coping with such challenges. Although [22] employs inverse reinforcement learning to learn a reward from demonstration, this work still suffers the undefined problem due to the nature of the algorithm.

More specifically, when $D(s, a)$ is zero, $\log D(s, a)$ would be undefined, making the whole reward function undefined. As a response, we manipulate the function by adding an extra learnable term to mitigate such a problem. In addition, existing studies have not considered the absorbing state problem such that agents will stop learning once the absorbing states are reached—the agent will receive zero rewards in absorbing states, which lead to a suboptimal policy. [13] first uses inverse reinforcement learning for scanpath prediction. However, this work suffers the same problem as [22], i.e., the reward could be zero in absorbing states and cannot be generalized to other tasks. [22] integrates the actor-critic network and IRL based on GAN, which suffers training instability. Moreover, sample efficiency is another drawback for the on-policy actor-critic network. In this paper, we employ Wasserstein GAN [23] to improve model stability and importance sampling in the replay buffer to transfer it into off-policy learning with increased sample efficiency.

In this paper, we aim to construct user models directly from an array of various demonstrations efficiently and adaptively, based on a generalized inverse reinforcement learning method.

Learning from demonstrations not only avoids the need for inferring a reward function but also reduces computational complexity.

To this end, we propose a new model that employs a generative adversarial strategy to generate candidate reward functions and approximate the true reward. We use the new model as a general way of characterizing and explaining tendencies in user behaviors.

In summary, we make the following contributions:

- We propose a new inverse reinforcement learning-based method to capture user’s behavioral tendencies. To the best of our knowledge, this is the first work to formulate user’s behavioral tendency using inverse reinforcement learning.
- We design a novel stabilized sample-efficient discriminative actor-critic network with Wasserstein GAN to implement the proposed framework. Our framework is off-policy and can reduce interactions between system and environment to improve efficiency. Besides, we integrate a learnable term into our reward function to increase the capability of our method.
- Our extensive experiments demonstrate the generalization ability and feasibility of our approach in three different scenarios. We use visualization to show the explainability of our method.

2 PROBLEM FORMULATION AND PRELIMINARY

Behavioral tendency refers to user’s preferences at a certain timestamp and is usually hard to be evaluated directly. The common way to evaluate behavioral tendencies is to examine how well the actions taken out of the learned behavioral tendencies match the real actions taken by the user. It is similar to reinforcement learning’s decision-making process, where the agent figures out an optimal policy $\pi$ such that each action of it could achieve a good reward.

In this work, we define behavioral tendencies modeling as an optimal policy-finding problem. Given a set of users $U = \{u_0, u_1, \ldots, u_n\}$, a set of items $O = \{o_0, o_1, \ldots, o_m\}$ and user’s demographic information $D = \{d_0, d_1, \ldots, d_n\}$. We first define the Markov Decision Process (MDP) as a tuple $(S, A, P, R, \gamma)$, where $S$ is the state space (i.e., the combination of the subset of $O$, subset of $U$ and its corresponding $D$), $A$ is the action space, which includes all possible agent’s decisions, $R$ is a set of rewards received for each action $a \in A$, $P$ is a set of state transition probability, and $\gamma$ is the discount factor used to balance the future reward and the current reward. The policy can be defined as $\pi : S \rightarrow A$—given a state $s \in S$, $\pi$ will return an action $a \in A$ so as to maximize the reward. However, it is unrealistic to find a universal reward function for user behavioral tendency, which is highly task-dependent. Hence, we employ Inverse reinforcement learning (IRL) to learn a policy $\pi$ from the demonstration from expert policy $\pi_{exp}$, which always results in user’s true behavior. We formulate the IRL process using a uniform cost function $c(s, a)$ [20]:

$$\min_{\pi} \max_{c \in \mathbb{C}} \mathbb{E}_{\pi}[c(s, a)] - \mathbb{E}_{\pi}[c(s, a)]$$  \hspace{1cm} (1)

The cost function class $\mathbb{C}$ is restricted to convex sets defined by the linear combination of a few basis functions $\{f_1, f_2, \ldots, f_k\}$. Hence, given a state-action pair $(s, a)$, the corresponding feature vector can be represented as $f(s, a) = [f_1(s, a), f_2(s, a), \ldots, f_k(s, a)]$. $\mathbb{E}_{\pi}[c(s, a)]$ is defined as (on $\gamma$-discounted infinite horizon):

$$\mathbb{E}_{\pi}[c(s, a)] = \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t c(s_t, a_t) \right]$$  \hspace{1cm} (2)

According to Eq.(1), the cost function class $\mathbb{C}$ is convex sets, which have two different formats: linear format [24] and convex format [25], respectively:

$$\mathbb{C}_l = \left\{ \sum_i w_i f_i : \|w\|_2 \leq 1 \right\}$$  \hspace{1cm} (3)

$$\mathbb{C}_c = \left\{ \sum_i w_i f_i : \sum_i w_i = 1, \forall i \text{ s.t. } w_i \geq 0 \right\}$$  \hspace{1cm} (4)
The corresponding objective functions are as follows:

\[
\begin{align*}
\| \mathbb{E}_\pi[f(s, a)] - \mathbb{E}_{\pi_E}[f(s, a)] \|_2 & \quad (5) \\
\mathbb{E}_\pi[f_j(s, a)] - \mathbb{E}_{\pi_E}[f_j(s, a)] & \quad (6)
\end{align*}
\]

Eq. (5) is known as feature expectation matching [24], which aims to minimize the l2 distance between the state-action pairs that are generated by learned policy \( \pi \) and expert policy \( \pi_E \). Eq. (6) aims to minimize the function \( f_j \) such that the worst-case should achieve a higher value [26]. Since Eq. (1) suffers the feature ambiguity problem, we introduce \( \gamma \)-discounted causal entropy [27] (shown below) to relieve the problem:

\[
H(\pi) \triangleq \mathbb{E}_\pi[-\log \pi(a|s)] = \mathbb{E}_{s_t,a_t\sim \pi} \left[ -\sum_{t=0}^\infty \gamma^t \log \pi(a_t|s_t) \right] \quad (7)
\]

As such, Eq. (1) can be written by using the \( \gamma \)-discounted causal entropy as:

\[
\min_{\pi} -H(\pi) - \mathbb{E}_{\pi_E}[c(s, a)] + \max_{c \in C} \mathbb{E}_\pi[c(s, a)] \quad (8)
\]

Suppose II is the policy set. We define the loss function \( c(s, a) \) to ensure the expert policy receives the lowest cost while all the other learned policies get higher costs. Referring to Eq. (8), the maximum causal entropy inverse reinforcement learning [28] works as follows:

\[
\max_{c \in C} \left( \min_{\pi \in II} -H(\pi) + \mathbb{E}_\pi[c(s, a)] \right) - \mathbb{E}_{\pi_E}[c(s, a)] \quad (9)
\]

Then, the policy set II can be obtained via policy generation. Policy generation is the problem of matching two occupancy measures and can be solved by training a Generative Adversarial Network (GAN) [29]. The occupancy measure \( \rho \) for policy \( \pi \) can be defined as:

\[
\rho_\pi(s, a) = \pi(s|a) \sum_{t=0}^\infty \gamma^t P(s_t = s|\pi) \quad (10)
\]

We adopts GAIL [21] and make an analogy from the occupancy matching to distribution matching to bridge inverse reinforcement learning and GAN. A GA regularizer is designed to restrict the entropy function:

\[
\psi_{GA}(c(s, a)) = \begin{cases} 
\mathbb{E}_{\pi_E}[-c(s, a) - \log(1 - \exp(c(s, a)))] & c < 0 \\
\infty & c \geq 0
\end{cases} \quad (11)
\]

The GA regularizer enables us to measure the difference between the \( \pi \) and \( \pi_E \) directly without the reward function:

\[
\psi_{GA}(\rho_\pi - \rho_{\pi_E}) = \max_{D \in \{0,1\}^{S \times A}} \mathbb{E}_\pi[\log D(s, a)] + \mathbb{E}_{\pi_E}[\log (1 - D(s, a))] \quad (12)
\]

The loss function from the discriminator \( D \) is defined as \( c(s, a) \) in Eq.(9); it uses negative log loss (commonly used for binary classification) to distinguish the policies \( \pi \) and \( \pi_E \) via state-action pairs. The optimal of Eq.(12) is equivalence to the Jensen-Shannon divergence [30]:

\[
D_{JS}(\rho_\pi, \rho_{\pi_E}) = D_{KL}(\rho_\pi || (\rho_\pi + \rho_{\pi_E})/2) + D_{KL}(\rho_{\pi_E} || (\rho_\pi + \rho_{\pi_E})/2) \quad (13)
\]

Finally, we rewrite inverse reinforcement learning by substituting the GA regularizer into Eq.(8):

\[
\min_{\pi} -\lambda H(\pi) + \psi_{GA}(\rho_\pi - \rho_{\pi_E}) \quad (14)
\]

where \( \lambda \) is a factor with \( \lambda \geq 0 \). Eq.(14) has the same goal as the GAN, i.e., finding the squared metric between distributions. Eq.(14) can be further extended into the following, which serves as the objective function for GAIL:

\[
\min_{\pi} -\lambda H(\pi) + \psi_{GA}(\rho_\pi - \rho_{\pi_E}) \equiv \min_{\pi} \max_{D} \mathcal{L}_D \\
\mathcal{L}_D = \mathbb{E}_{\pi}[\log D(s, a)] + \mathbb{E}_{\pi_E}[\log (1 - D(s, a))] - H(\pi) \quad (15)
\]

We summarized all the notations used in this paper in Table 1.

### 3 Methodology

The overall structure of our proposed method (shown in Fig. 1) consists of three components: policy and reward learning, stabilized sample efficient discriminative actor-critic network, and its optimization. Policy and reward learning aims to solve the reward bias and the absorbing state problem by introducing a learnable reward function and environment feedback. The stabilized actor-critic network aims to improve the training stability and sample efficiency for the existing methods. Optimization refers to the method to optimize the policy and the algorithms to train the overall approach.

#### 3.1 Policy and Reward Learning

We consider behavioral tendencies inference as an agent policy learning problem and an agent policy as the
abstraction of user’s behavioral tendencies. Policy learning aims to make the learned policy $\pi$ and expert policy $\pi_E$.

We define the occupancy measure $r$ in Eq.(10) and solve policy learning as an occupancy measure based distribution matching problem [24]. To this end, we define a reward function below to determine the performance in existing methods:

$$r(s, a) = \log \frac{D(s, a)}{1 + D(s, a)}$$

[31] design a dynamic robust disentangled reward function for the approximation by introducing the future state $s'$. 

$$r'(s, a) = \log \frac{D(s, a, s')}{{1 + D(s, a, s')}}$$

The reward function defined in Eq.(16) is not robust for dynamic environments. Although Eq.(17) improves it by assigning positive and negative rewards for each time step to empower the agent to fit into different scenarios, both Eqs.(16) and (17) have the absorbing state problem, i.e., the agent will receive no reward at the end of each episode, leading to sub-optimal policies [32]. Specifically, instead of exploring more policies, the reward function $r(s, a)$ will assign a negative reward bias for the discriminator to distinguish samples from the generated policies and expert policies at the beginning of the learning process. Since the agent aims to avoid the negative penalty, the zero reward may lead to early stops.

Moreover, the above two reward functions are more suitable for survival or exploration tasks rather than the goal of this study. For survival tasks, the reward used on GAIL is $\log D(s, a)$, which is always negative because $D(s, a) (\in [0, 1])$ encourages the agent to end current episode to stop more negative rewards. For exploration tasks, the reward function $-\log (1 - D(s, a))$ is always positive and may result in the agent looping in the environment to collect more rewards. Moreover, because $D(s, a)$ could be zero which will make the whole term undefined.

Hence, we add a bias term to the reward function $r(s, a)$, as defined by either Eq.(16) or Eq.(17) to overcome the reward bias and the undefined problem. In addition, we introduce a new reward given by environment $r_e$ for reward shaping. Finally, we have the following:

$$r_n(s, a) = \lambda_i \left( r(s, a) + \sum_{t=T+1}^{\infty} y^{t-T} r(s_{n, t}) \right) + r_e$$

where $r(s_{n, t})$ is a learnable reward function, which is trainable during the training process. We also add a dimension to indicate whether the current state is an absorbing state or not (denoted by 1 or 0, respectively). Besides, we simply sample the reward from the replay buffer, considering the bias term is unstable in practice.

### 3.2 Stabilized Sample Efficient Discriminative Actor-Critic Network

The stabilized sample efficient discriminative actor-critic network aims to enable the agent to learn the policy efficiently. We take a variant of the actor-critic network, advantage actor-critic network [33] as the backbone of our approach. In this network, the actor uses policy gradient
and the critic’s feedback to update the policy, and the critic uses Q-learning to evaluate the policy and provides feedback [34].

Given the state space at timestamp $t$, the environment determines a state $s_t$, which contains user’s recent interest and demographic information embedded, via the actor-network [35, 36]. The actor-network feeds the state $s_t$ to a network that has four fully-connected layers with ReLU as the activation function. The final layer of the network outputs a policy function $\pi$, which is parameterized by $\theta$. Then, the critic network takes two inputs: the trajectory $(s_t, a_t)$, and the current policy $\pi_\theta$ from the actor-network. We concatenate the state-action pair $(s_t, a_t)$ and feed it into a network with four fully-connected layers (with ReLU as the activation function) and a softmax layer. The output of the critic-network is a value $V(s_t, a_t) \in \mathbb{R}$ to be used for optimization (to be introduced later).

The discriminator $D$ is the key component of our approach. To build an end-to-end model that better approximates the expert policy $\pi_E$, we parameterize the policy with $\pi_\theta$ and clip the discriminator’s output so that $D : S \times \mathcal{A} \rightarrow (0,1)$ with weight $w$. The loss function of $D$ is denoted by $\mathcal{L}_D$. Besides, we use Adam [37] to optimize weight $w$ (the optimization for $\theta$ will be introduced later).

We consider the discriminator $D$ as a local cost function to guide the policy update. During the minimization of the loss function $\mathcal{L}_D$, i.e., finding a point $(\pi, D)$ for it, the policy will move toward expert-like regions (divided by $D$) in the latent space.

Like many other networks, Actor-critic network also suffers the sample inefficiency problem [38], i.e., the agent has to conduct sampling from the expert policy distribution, given the significant number of agent-environment interactions needed to learn the expert policy during the training process. In this regard, we use an off-policy reinforcement learning algorithm (instead of on-policy reinforcement learning algorithms) to reduce interactions with the environment. In particular, we introduce a replay buffer $\mathcal{R}$ to store previous state-action pairs; when training the discriminator, we sample the transition from the replay buffer $\mathcal{R}$ in off-policy learning (instead of sampling trajectories from a policy directly). We thereby define the loss function as follows:

$$\mathcal{L}_D = \mathbb{E}_R[\log D(s,a)] + \mathbb{E}_{\pi_E}[\log (1-D(s,a))] - \lambda H(\pi) \tag{19}$$

Eq.(19) matches the occupancy measures between the expert and the distribution induced by $\mathcal{R}$. Instead of comparing the latest trained policy $\pi$ and expert policy $\pi_E$, it comprises a mixture of all policy distributions that appeared during training. Considering off-policy learning has different expectation from on-policy learning, we use importance sampling on the replay buffer to balance it.

$$\mathcal{L}_D = \mathbb{E}_R \left[ \frac{\rho_{\pi_E}}{\rho_\pi} \log D(s,a) \right] + \mathbb{E}_{\pi_E}[\log (1-D(s,a))] - \lambda H(\pi) \tag{20}$$

Considering GAN has the training instability problem [39], we employ the Wasserstein GAN [23] to improve the discriminator’s performance. While a normal GAN minimizes JS-Divergence cannot measure the distance between two distributions, Wasserstein GANs uses the EM-distance and Kantorovich-Rubinstein duality to resolve the problem [40].

$$\mathbb{E}_{\pi}[\log D(s,a)] - \mathbb{E}_{\pi_E}[\log (D(s,a))] + \mathbb{E}_{\pi_E}[\|\nabla D(s,a)\| - 1]^2 \tag{21}$$

We further use gradient penalty to improve the training for Wasserstein GANs [23], given the gradient penalty can improve training stability for JS-Divergence-based GANs [41]. We thereby obtain the final loss function as follows:

$$\mathcal{L}_D = \mathbb{E}_R \left[ \frac{\partial \pi_\theta}{\partial \mathcal{R}}(s,a) \log D(s,a) \right] + \mathbb{E}_{\pi_E}[\log (1-D(s,a))] - \lambda H(\pi) + \mathbb{E}_{\pi_E}[\|\nabla D(s,a)\| - 1]^2 \tag{22}$$

### 3.3 Optimization

We conduct a joint training process on the policy network (i.e., the actor-critic network) and the discriminator. We parameterize the policy network with policy parameter $\theta$ and update it using trust region policy optimization (TRPO) [42] based on the discriminator. TRPO introduces a trust region by restricting the agent’s step size to ensure a new policy is better than the old one. We formulate the TRPO problem as follows:

$$\max_\theta \frac{1}{T} \sum_{t=0}^{T} \mathbb{E}_{a_t|s_t} \left[ \frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)} A_t \right]$$

subject to $D_{kld}^\text{old}(\pi_{old}, \pi_\theta) \leq \eta \tag{23}$

where $A_t$ is the advantage function calculated by Generalized Advantage Estimation (GAE) [43]. GAE is described as follows:

$$A_t = \sum_{\ell=0}^{\infty} (y_{\lambda_{\ell}})^{\ell} r_{t+\ell}$$

where $\delta_{t+\ell} = -V(s_t) + \sum_{\ell=0}^{\infty} (y_{\lambda_{\ell}})^{\ell} r_{t+\ell} \tag{24}$

where $r_{t+\ell}$ is the test reward for $\ell$-step’s at timestamp $t$, as defined on Eq.(18). Considering the high computation load of updating TRPO via optimizing Eq.(23), we update the policy using a simpler optimization method called Proximal Policy Optimization (PPO) [44], which has an objective function below:

$$\min_{\pi \sim \pi_{dd}} \sum_{t=0}^{T} \min \left( \frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)}, 1 - \epsilon, 1 + \epsilon \right) A_t,$$

clip \left( \frac{\pi_\theta(a_t|s_t)}{\pi_{old}(a_t|s_t)}, 1 - \epsilon, 1 + \epsilon \right) A_t \right) \tag{25}$$

where $\epsilon$ is the clipping parameter representing the maximum percentage of change that can be made by each update.

The overall training procedure is illustrated in Algorithm 1, which involves the training of both the discriminator and the actor-critic network. For the discriminator, we use Adma as the optimizer to find the gradient for Eq.(22) for weight $w$ at step $i$: 

\[
E_\pi [\nabla_w \log (D_w(s,a))] + E_{\pi_0} [\nabla_w \log (1 - D_w(s,a))] \\
+ E_{\pi_0} [\|\nabla_D (D(s,a))\| - 1]^2 \]  
(26)

Algorithm 1. Training Algorithm for Our Model

Input: Expert replay buffer \(R_E\), Initialize Policy Replay Buffer \(R\), Initialize policy parameter \(\theta_0\), clipping parameter \(\varepsilon\)

1: **Function Absorbing(\(r\)) is**
2: \[\text{if } s_t \text{ is a absorbing state then}\]
3: \[\{s_t, a_t, r_t, s_t'\} \leftarrow \{s_t, a_t, r_t, s_t\};\]
4: \[\tau \leftarrow \tau \cup \{s_t, a_t, r_t, s_t\};\]
5: **end**
6: return \(\tau;\)
7: **end**
8: for \(\tau \{s_t, a_t, r_t, s_t'\}_{t=1}^T \in R_E\) do
9: \(\tau \leftarrow \text{Absorbing}(\tau);\)
10: **end**
11: \(R \leftarrow \emptyset;\)
12: for \(i = 1, 2, \ldots\) do
13: Sampling trajectories \(\tau \{s_t, a_t, r_t, s_t'\}_{t=1}^T \sim \pi_0;\)
14: \(R \leftarrow R \cup \text{Absorbing}(\tau);\)
15: for \(j = 1, \ldots, |\tau|\) do
16: \(s_t, a_t, r_t, s_t'_{i=1}^T \sim R;\)
17: for \(k = 1, \ldots, B\) do
18: \(r = \log (D_{w_1}(s_t, a_t)) - \log (1 - D_{w_1}(s_t, a_t));\)
19: Compute the reshape reward \(r'\) by Eq.(18);\)
20: \(s_t, a_t, r', s_t'_{i=1}^T \sim (s_t, a_t, r', s_t'_{i=1}^T);\)
21: **end**
22: for \(k = 0, 1, \ldots\) do
23: Get the trajectories \((s, a)\) on policy \(\pi_\theta = \pi(\theta_k);\)
24: Estimate advantage \(A_t\) using Eq.(24);\)
25: Compute the Policy Update ;\)
26: \(\theta_{k+1} = \arg \max \theta \text{Eq.(25)};\)
27: By taking \(K\) step of minibatch SGD (via Adma);\)
28: **end**
29: \(\theta_t \leftarrow \theta_K;\)
30: **end**
31: **end**

Algorithm 2. PPO Update

Input: Initialize policy parameter \(\theta_0\), clipping parameter \(\varepsilon\)

1: for \(k = 0, 1, \ldots\) do
2: Get the trajectories \((s, a)\) on policy \(\pi_\theta = \pi(\theta_k);\)
3: Estimate advantage \(A_t\) using Eq.(24);\)
4: Compute the Policy Update ;\)
5: **end**
6: \(\theta_t \leftarrow \theta_K;\)

4 Experiments

We evaluate the proposed framework and demonstrate its generalization capability by conducting experiments in three different environments: Traffic Control, Recommendation System, and Scanpath Prediction.

Our model is implemented in Pytorch [45]. All experiments are conducted on a server with 6 NVIDIA TITAN X Pascal GPUs, 2 NVIDIA TITAN RTX with 768 GB memory.

4.1 Urban Mobility Management

In the traffic control scenario, the agent is required to control cars to conduct a certain task. The objective is to minimize the total waiting time in the trip.

4.1.1 Simulation of Urban Mobility

Traffic signal control is critical to effective mobility management in modern cities. To apply our model to this context, we use the Simulation of Urban MOBility (SUMO) [46] library, a microscopic, space-continuous, and time-discrete traffic flow simulation tool, to test the method’s performance. The agent controls traffic signals, and a car may take three actions facing traffic lights: go straight, turn left, or turn right, depending on user’s preference. We design a simple two-way road network that contains eight traffic lights for testing. We employ an open-sourced library sumo-rl\(^1\) to enable our agent can interact with the simulation environment (including receiving the reward) directly. The number of cars available in the environment is unlimited; the environment keeps generating cars until this simulation step ends or the road reaches its full capacity.

4.1.2 Expert Policy Acquisition

Since there is no official expert policy available for our customized road network, we use the same strategy as introduced by [47] to collect a set of imperfect expert policies from a pre-trained policy network. This policy network is built upon an actor-critic network, which is trained by using Deep Deterministic Policy Gradients (DDPG) [48]. Expert policies are stored via state-action pairs, which concatenate observed states and expert actions.

4.1.3 Baseline Methods

We evaluate our model against several traditional reinforcement learning methods in this scenario.

- Q-Learning: An off-policy reinforcement learning method that finds the best action given the current state.
- Deep Q-learning: A deep Q-learning method that employs the neural network to extract features.
- SARSA: State–action–reward–state–action (SARSA) is an improved Q-learning method commonly used for traffic signal control.
- Advantage Actor-Critic Network (A2C) [33]: An asynchronous method built on an actor-critic network for deep reinforcement learning.
- GIRL-RS [22]: An inverse reinforcement learning based method for recommendation. We modify the structure by removing the recommender to handle the traffic control task.

\(^1\) https://github.com/LucasAlegre/sumo-rl
Experiments are conducted in exactly the same environment to ensure a fair comparison. All the baseline methods are implemented by using PyTorch and are publicly available. The reward provided by environment for each simulation step can be defined as:

\[ r = \sum_{n=0}^{N\text{ts}} s \cdot N_{cp} \]  

where \( N_{ts} \) is the number of traffic signals available in the environment, \( s \) is the average car speed in this simulation step, and \( N_{cp} \) is the number of cars passed this traffic signal at the end of this simulation step. The evaluation metric is the total waiting time defined below:

\[ t = \sum_{i=0}^{1000} \sum_{c=0}^{N} t_{ic} \]  

where \( t_{ic} \) is the time that the car \( c \) waits at traffic light \( i \), and 1,000 is the duration for one simulation step. If car \( c \) does not meet traffic light \( i \), we set \( t_{ic} = 0 \).

4.1.4 Hyper-Parameters Setting and Results

DDPG parameters for the pre-trained model include \( \gamma = 0.95 \), \( \tau = 0.001 \), the size of the hidden layer 128, the size of the reply buffer 1,000, and the number of episode 20,000. Parameters for Ornstein-Uhlenbeck Noise include the scale 0.1, \( \mu = 0 \), \( \theta = 0.15 \), \( \sigma = 0.2 \). For our method, we set the number of time steps to 20,000, the hidden size of the advantage actor-critic network to 256, the hidden size for discriminator to 128, the learning rate to 0.003, factor \( \lambda \) to \( 10^{-3} \), mini batch size to 5, and the epoch of PPO to 4. For the generalized advantage estimation, we set the discount factor \( \gamma \) to 0.995, \( \lambda_0 = 0.97 \), and \( \epsilon = 0.2 \). We also set \( \lambda_1 = 1 \) for reward shaping and \( \lambda = 1 \) for \( H(\pi) \). The results in Fig. 2(a) show our method generally outperforms all baseline methods.

4.2 Recommendation System

In the recommendation scenario, the agent aims to interact with a dynamic environment to mine user’s interests and make recommendations to users.

4.2.1 VirtualITB

We use an open-source online recommendation platform, VirtualITB [49], to test the performance of the proposed methods in a recommendation system. VirtualITB is a dynamic environment built on OpenAI Gym\(^3\) to test our method’s feasibility on recommendation tasks. VirtualITB employs a customized agent to interact with it and achieves the corresponding rewards. It can also generate several customers with different preferences during the agent-environment interaction. In VirtualITB, each customer has 11 static attributes encoded into an 88-dimensional space with binary values as the demographic information. The customers have multiple dynamic interests, which are encoded into a 3-dimensional space and may change over the interaction process. Each item has several attributes (e.g., price and sales volume), which are encoded into a 27-dimensional space. We use CTR as the evaluation metric because the gym environment can only provide rewards as feedback. CTR is defined as follows:

\[ CTR = \frac{r_{episode}}{10 \cdot N_{step}} \]  

where \( r_{episode} \) is the reward that the agent receives at each episode. At each episode, the agent may take \( N_{step} \) steps and receive a maximum reward of 10 per step.

4.2.2 Baseline Methods

We evaluate our model against five state-of-the-art methods covering methods based on deep Q-learning, policy gradient, and actor-critic networks.

- IRecGAN [50]: An online recommendation method that employs reinforcement learning and GAN.
- PGCR [51]: A policy-Gradient-based method for contextual recommendation.
- GAUM [52]: A deep Q-learning based method that employs GAN and cascade Q-learning for recommendation.
- KGRL [35]: An Actor-Critic-based method for interactive recommendation, a variant of online recommendation.

Fig. 2. Comparison results. From left to right, the subfigures represent the results in (a) Traffic Control, (b) Recommendation System, and (c) Scanpath Prediction. Our methods generally outperform baseline methods.

---

2. https://github.com/hill-a/stable-baselines
3. https://gym.openai.com/
Note that GAUM and PGCR are not designed for online recommendation, and KGRL requires a knowledge graph—which is unavailable to the gym environment—as the side information. Hence, we only keep the network structure of those networks when testing them on the VirtualTB platform.

4.2.3 Hyper-Parameters Setting and Results
The hyper-parameters are set in a similar way as in the traffic signals control. We set the number of episodes to 200,000 for both the pre-trained policy network and our method. To ease comparison, we configure each iteration to contain 100 episodes. The results in Fig. 2(b) show our method outperform all state-of-the-art methods. KGRL’s poor performance may be partially caused by its reliance on knowledge graph, which is unavailable in our experiments.

4.3 Scanpath Prediction
Scanpath prediction is a type of goal-directed human intention prediction problem [13]. Take the last task in Fig. 1 for example. Given a few objects, a user may first look at item 1, then follows the item numbers annotated in the figure, and finally reaches item 8. The task aims to predict user’s intention (i.e., item 8), given the start item (i.e., item 1).

4.3.1 Experimental Setup
We follow the same experimental setup as [13] and conduct all experiments on a public COCO-18 Search dataset.4. We replace the fully-connected layer in the actor-network with CNN to achieve the best performance of our method on images. The critic-network has a new structure with two CNN layers followed by two fully-connected layers. The discriminator contains all CNN layers with a softmax layer as output. We also resize the input image from the original size of 1680 × 1050 into 320 × 512 and construct the state by using the contextual beliefs calculated from a Panoptic-FPN with a backbone network (ResNet-50-FPN) pretrained on COCO2017. We use probability mismatch and scanpath ratio as the main evaluation metrics. Probability mismatch is defined as the sum of the absolute differences between the human and model cumulative probability of target fixation [13]; Scanpath ratio is calculated as the ratio of euclidean distance between the initial fixation location and the center of the target to the summed euclidean distances between fixations to the target.

4.3.2 Baseline Methods
We compare our method with several baseline methods, including simple CNN based methods, behavior-cloning based methods, and inverse reinforcement-learning-based method. Experiments are conducted under the same conditions to ensure fairness.

- Detector: A CNN-based architecture to predict the location of a target item.
- Fixation heuristics [13]: A method similar to Detector but using the fixation to predict the location of a target item.
- BC-CNN [53]: A behavior-cloning method that uses CNN as the basic layer structure.
- BC-LSTM [54]: A behavior-cloning method that uses LSTM as the basic layer structure.

Fig. 3. Results of ablation study for three selected environments.
- IRL [13]: A state-of-the-art inverse reinforcement-learning-based method for scanpath prediction.
- GIRL-RS[22]: A generative inverse reinforcement learning method for recommendation. We remove the recommender and replace it by the prediction components as [13] did.

### 4.3.3 Performance Comparison

The hyper-parameters settings are the same as those used for the recommendation task. We also use the same evaluation metrics as used in [13] to evaluate the performance: cumulative probability, probability mismatch, and scanpath ratio. The results in Fig. 2(c) show the cumulative probability of the gaze landing on the target after the first six fixations. We report the probability mismatch and scanpath ratio in Table 2.

### 4.4 Evaluation on Explainability

Explainability plays a crucial role in understanding the decision-making process. By visualizing the learned reward map, we show in this experiment that our model can provide a certain level of interpretability. We evaluate the explainability for our model in the scanpath prediction scenario. Fig. 4 shows that the reward maps recovered by the our model depend heavily on the category of the search target. In the first image, the highest reward is assigned to the piazza when drinking beers. Similarly, the searching of road signal on the road, the stop signal get almost all of the reward while the car get only a few.

### 4.5 Ablation Study

We test using three different optimization strategies (DDPG, Adaptive KL Penalty Coefficient and Twin Delayed DDPG) to update the policy parameter $\theta$. The Adaptive KL Penalty Coefficient is defined as:

$$L(\theta) = \mathbb{E}_t \left[ \frac{\pi_\theta(a_t|s_t)}{\pi_\theta(\alpha|s_t)} A_t - \beta KL[\pi_\theta(\cdot|s_t), \pi_\theta(\cdot|s_t)] \right]$$

where the $\beta$ will be adjust dynamically by the following:

$$\left\{ \begin{array}{ll}
\beta \leftarrow \beta/2 & d < d_{\text{target}} * 1.5 \\
\beta \leftarrow \beta * 2 & d > d_{\text{target}} * 1.5 \\
\end{array} \right.$$  

where $d = \mathbb{E}_t[KL[\pi_{\text{old}}(\cdot|s_t), \pi_\theta(\cdot|s_t)]]$

We empirically choose coefficient 1.5 and 2 and select total waiting time, CTR, and cumulative probability as the evaluation metrics to compare the three optimization strategies for traffic signal control, recommendation system, and scanpath prediction, respectively. The results (shown in Fig. 3) show our optimization method achieve a similar result as TD3 on all the three tasks but is better than TD3 on the recommendation task. Hence, we conduct a further step about the parameter selection about PPO and GAE, which can be found in Table 3.

Moreover, we investigate the effect of the proposed stabilized approach and sample efficiency. The results are shown in Fig. 3, where Our-E represents our method without the stabilized training and Our-R represents our method without the approach we use to increase the sample efficiency. Figs. 3d, 3e, 3f shows the stabilizing training method is more important than sample efficiency in complex tasks, such as traffic control and recommendation. The possible reason could be that the scanpath prediction task has limited candidate items, which reduce the effect of removing the stabilizing method (i.e., Ours-E) on the final performance. The replay buffer would be a strong bonus for that as agent is able to access the past experience to improve the performance. To validate this claim, we conduct experiments in a more complex task by increasing the fixations from 6 to 10 and report the results in Fig. 5a. We also conduct the experiment in a new dataset, Microwave-Clock Search (MCS) [56], to further investigate the above claim. Different from the COCO-Search-18 dataset, MCS has a more complex environment but simpler tasks, where users only search for microwave, clock, or both. The results in Figs. 5a and b show Ours-E performs better than Ours-E at an early stage but is later outperformed by Ours-R as the number of fixations increases. Such results can support our claim that the Ours-E may perform better than Ours-R when the task is simple and candidate items are limited.

### 5 RELATED WORK

User behavior tendency modeling has been an active topic in research, and most previous efforts have been focusing...
on feature engineering rather than an end-to-end learning structure. Kim et al. [7] considers long-term interest as a reasonable representation of general interest and acknowledges its importance for personalization services. On this basis, Liu et al. [57] propose a framework that considers both long-term and short-term interest for user behavior modeling. Rather than establishing static models, Chung et al. [58] models long-term and short-term user profile scores to model user behaviors incrementally. Recently, Song et al. [59] propose to jointly model long-term and short-term user interest for recommendation based on deep learning methods. Pi et al. [60] further propose a MIMN model for sequential user behavior modeling. Despite good performance on their respective tasks, all the above methods are task-specific and lack generalization ability.

Reinforcement learning is widely used for user behavior modeling in recommendation systems. Zheng et al. [2] adopt deep Q-learning to build up user profiles during the interaction process in a music recommendation system. Zou et al. [61] improve the Q-learning structure to stabilize the reward function and make the recommendation robust. [35], [62], [63] apply reinforcement learning for extracting user’s interest from a knowledge graph. Liu et al. [36] embed user’s information into a latent space and conduct recommendation via deep reinforcement learning. Different from those mentioned works, Pan et al. [51] applies the policy gradient directly to optimize the recommendation policy. Chen et al. [52] integrates GAN into the reinforcement learning framework so as to enrich the latent space with user’s side information to improve the recommendation accuracy. Shang et al. [64] consider the environment co-founder factors and propose a multi-agent based reinforcement learning method for recommendation. All the above studies require defining accurate reward functions, which are hard to obtain in the real world.

Inverse reinforcement learning emerges where reward functions cannot be defined [20]. Lee et al. [65] firstly use inverse reinforcement learning to learn user’s behavior styles. However, general inverse reinforcement learning is computationally expensive. Ho et al. [21] propose a generative reinforcement learning approach to improve efficiency. Fu et al. [31] further extend the idea to a general form to obtain a more stable reward function. Kostrikov et al. [32] find a generative method may suffer instability in training, which can be relieved by using EM-distance instead of JS-divergence. Yang et al. [13] first introduce the inverse reinforcement learning into the scanpath prediction and demonstrate the superior performance. IRL demonstrates a huge potential and is widely used in robot learning as it can empower the agent to learn from demonstration in different environments and tasks without dramatic exploration about the environment or being familiar with the tasks. Chen et al. [22] expand this idea into recommender systems and show the feasibility of IRL in recommendation tasks.

## Conclusion and Future Work

In this paper, we propose a novel method based on an advantage actor-critic network with inverse reinforcement learning to overcome the adverse impact caused by inaccurate reward functions for user behavior modeling. In particular, we use the Wasserstein GAN instead of GAN to increase training stability and a replay buffer for off-policy learning to increase sample efficiency. A comparison with several state-of-the-art methods in three different scenarios (namely traffic signal control, recommendation system, and scanpath prediction) demonstrate our method’s feasibility and superior performance to baseline methods. This work poses a promising direction towards applying inverse reinforcement learning to real life. It demonstrates the feasibility of generalized behavior modeling in several scenarios, such as recommender systems and traffic light control, which could be of importance in smart cities and related applications.

Although experience replay can boost sample efficiency by switching the sampling process from the environment to replay buffer, it may not be ideal as some tasks (e.g., recommendation) may have giant state and action spaces. Moreover, not every experience is useful even if it comes from demonstration because expert demonstrations are randomly sampled from replay buffer and may be orthogonal with the current state, which leads to adverse actions. Possible solutions include state-aware experience replay methods and prioritized

### Table 3
CTR for Different Parameter Settings for GAE and PPO

| GAE: $\lambda_g$ | 0.94 | 0.95 | 0.96 | 0.97 | 0.98 | 0.99 |
|-----------------|------|------|------|------|------|------|
| PPO: $\epsilon$ | 0.05 | 0.630 ± 0.063 | 0.632 ± 0.064 | 0.633 ± 0.062 | 0.630 ± 0.059 | 0.626 ± 0.060 | 0.629 ± 0.059 |
| 0.10            | 0.632 ± 0.062 | 0.635 ± 0.060 | 0.636 ± 0.061 | 0.636 ± 0.058 | 0.634 ± 0.061 | 0.633 ± 0.060 |
| 0.15            | 0.633 ± 0.060 | 0.635 ± 0.061 | 0.639 ± 0.061 | 0.640 ± 0.057 | 0.639 ± 0.059 | 0.638 ± 0.061 |
| 0.20            | 0.634 ± 0.060 | 0.636 ± 0.060 | 0.641 ± 0.063 | 0.643 ± 0.061 | 0.643 ± 0.063 | 0.641 ± 0.058 |
| 0.25            | 0.631 ± 0.061 | 0.635 ± 0.059 | 0.636 ± 0.060 | 0.637 ± 0.060 | 0.636 ± 0.061 | 0.634 ± 0.059 |
| 0.30            | 0.630 ± 0.059 | 0.631 ± 0.061 | 0.632 ± 0.060 | 0.630 ± 0.059 | 0.630 ± 0.058 | 0.629 ± 0.050 |

### Figures

- (a) Increase the fixations to 10 in COCO-Search-18
- (b) Results in MCS

Fig. 5. Further ablation study for Scanpath prediction.
experience replay based methods [66]. Another point for potential improvement is with Wasserstein GAN, as the Lipschitz constraint is hard to enforce and may lead to a model convergence issue.

References

[1] J. Hong, E.-H. Suh, J. Kim, and S. Kim, “Context-aware system for proactive personalized service based on context history,” Expert Syst. Appl., vol. 36, no. 4, pp. 7448–7457, 2009.

[2] G. Zheng et al., “DRN: A deep reinforcement learning framework for news recommendation,” in Proc. World Wide Web Conf., 2018, pp. 167–176.

[3] S. Zhang, L. Yao, A. Sun, and Y. Tay, “Deep learning based recommender system: A survey and new perspectives,” ACM Comput. Surv., vol. 52, no. 1, pp. 1–38, 2019.

[4] L. Hu, L. Cao, S. Wang, G. Xu, J. Cao, and Z. Gu, “Diversifying personalized recommendation with user-session context,” in Proc. Int. Joint Conf. Artif. Intell., 2017, pp. 1858–1864.

[5] X. Xu, P. Dong, Y. Meng, and X. Wang, “Exploring in interaction personalized music recommendation: A reinforcement learning approach,” in Proc. Adv. Neural Inform. Process. Syst., 2011, pp. 101–108.

[6] S. Seko, T. Yagi, M. Motegi, and S. Muto, “Group recommendation using feature space representing behavioral tendency and power balance among members,” in Proc. 5th ACM Recomender Syst., 2011, pp. 217–224.

[7] Y. Shi, M. Larson, and A. Hanjalic, “Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges,” ACM Comput. Surv., vol. 47, no. 1, pp. 1–45, 2014.

[8] Z. Yang, L. Huang, Y. Chen, Z. Wei, S. Ahn, G. Zelinsky, D. Samaras, and M. Hoai, “Predicting goal-directed human attention using inverse reinforcement learning,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 193–202.

[9] A. L. Bazzan, “Opportunities for multiagent systems and multiagent reinforcement learning in traffic control,” Auton. Agents Multi-Agent Syst., vol. 18, no. 3-4, 2009, Art. no. 342.

[10] H. Liu, Y. Zhang, W. Si, X. Xie, Y. Zhu, and S.-C. Zhu, “Interactive robot knowledge patching using augmented reality,” in Proc. IEEE Int. Conf. Robot. Automat., 2018, pp. 1947–1954.

[11] X. Chen, L. Yao, J. McAuley, G. Zhou, and X. Wang, “A survey of deep reinforcement learning for recommender systems: A systematic review and future directions,” 2021, arXiv:2109.03540.

[12] K. Arulkumaran, M. D. Deisenroth, M. Brundage, and A. A. Bharath, “Deep reinforcement learning: A brief survey,” IEEE Signal Process. Mag., vol. 34, no. 6, pp. 26–38, Nov. 2017.

[13] S.-Y. Chen, Y. Yu, Q. Da, J. Tan, H.-K. Huang, and H.-H. Tang, “Stabilizing reinforcement learning in dynamic environment with approximate Q-learning,” in Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2018, pp. 1187–1196.

[14] H. Chen et al., “Large-scale interactive recommendation with tree-structured policy gradient,” in Proc. AAAI Conf. Artif. Intell., 2019, pp. 3312–3320.

[15] A. Y. Ng et al., “Algorithms for inverse reinforcement learning,” in Proc. Int. Conf. Mach. Learn., 2000, Art. no. 2.

[16] J. Ho and S. Ermon, “Generative adversarial imitation learning,” in Proc. Adv. Neural Inform. Process. Syst., 2016, pp. 4565–4572.

[17] X. Chen, L. Yao, A. Sun, X. Wang, X. Xu, and L. Zhu, “Generative inverse deep reinforcement learning for online recommendation,” 2020, arXiv:2011.02248.

[18] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, “Improved training of wasserstein GANs,” in Proc. Adv. Neural Inform. Process. Syst., 2017, pp. 5767–5777.

[19] P. Abbeel and A. Y. Ng, “Apprenticeship learning via inverse reinforcement learning,” in Proc. 21st Int. Conf. Mach. Learn., 2004, Art. no. 1.

[20] U. Syed, M. Bowling, and R. E. Schapire, “Apprenticeship learning using linear programming,” in Proc. 25th Int. Conf. Mach. Learn., 2008, pp. 1078–1085.

[21] U. Syed and R. E. Schapire, “A game-theoretic approach to apprenticeship learning,” in Proc. Adv. Neural Inform. Process. Syst., 2008, pp. 1449–1456.

[22] M. Bloem and N. Bambos, “Infinite time horizon maximum causal entropy inverse reinforcement learning,” in Proc. 53rd IEEE Conf. Decis. Control, 2014, pp. 4911–4916.

[23] B. D. Ziebart, J. A. Bagnell, and A. K. Dey, “Modeling interaction via the principle of maximum causal entropy,” in Proc. Int. Conf. Mach. Learn., 2010, pp. 1255–1262.

[24] I. Goodfellow et al., “Generative adversarial nets,” in Proc. Adv. Neural Inform. Process. Syst., 2014, pp. 2672–2680.

[25] X. Nguyen et al., “On surrogate loss functions and f-divergences,” Ann. Statist., vol. 37, no. 2, pp. 876–904, 2009.

[26] J. Fu, K. Luo, and S. Levine, “Learning robust rewards with adversarial inverse reinforcement learning,” in Proc. Int. Conf. Learn. Representations, 2018. [Online]. Available: https://openreview.net/forum?id=rKHywL-A-

[27] I. Kostrikov, K. K. Agrawal, D. Dwibedi, S. Levine, and J. Tompson, “Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning,” in Proc. Int. Conf. Learn. Representations, 2019. [Online]. Available: https://openreview.net/forum?id=HkXh][]

[28] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, “Asynchronous methods for deep reinforcement learning,” in Proc. Int. Conf. Mach. Learn., 2016, pp. 1928–1937.

[29] V. R. Konda and J. N. Tsitsiklis, “Actor-critic algorithms,” in Proc. Adv. Neural Inform. Process. Syst., 2000, pp. 1008–1014.

[30] H. Chen, C. Huang, L. Yao, X. Wang, W. Liu, and W. Zhang, “Knowledge-guided deep reinforcement learning for interactive recommendation,” 2020, arXiv:2004.08068.

[31] F. Liu, H. Guo, X. Li, R. Tang, Y. Ye, and X. He, “End-to-end deep reinforcement learning based recommendation with supervised embedding,” in Proc. 13th Int. Conf. Web Search Data Mining, 2020, pp. 384–392.

[32] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014, arXiv:1412.6980.

[33] Z. Wang et al., “Sample efficient actor-critic with experience replay,” 2016, arXiv:1611.01224.

[34] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein gan,” 2017, arXiv:1701.07875.

[35] C. Villani, Optimal Transport: Old and New, vol. 338. Berlin, Germany: Springer, 2008, pp. 101–108.

[36] F. Liu, H. Guo, X. Li, R. Tang, Y. Ye, and X. He, “End-to-end deep reinforcement learning based recommendation with supervised embedding,” in Proc. 13th Int. Conf. Web Search Data Mining, 2020, pp. 384–392.

[37] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in Proc. Int. Conf. Mach. Learn., 2015, pp. 1889–1897.

[38] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, “High-dimensional continuous control using generalized advantage estimation,” 2015, arXiv:1506.02438.

[39] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” 2017, arXiv:1707.06347.

[40] A. Paszke et al., “Pytorch: An imperative style, high-performance deep learning library,” in Proc. Adv. Neural Inform. Process. Syst., 2018, pp. 700–709.

[41] X. Bai, J. Guan, and H. Wang, “A model-based reinforcement learning with adversarial training for online recommendation,” in Proc. Adv. Neural Inform. Process. Syst., 2019, pp. 4902–4909.
CHEN ET AL.: GENERATIVE ADVERSARIAL REWARD LEARNING FOR GENERALIZED BEHAVIOR TENDENCY INFERENCE

Xiaocong Chen is currently working toward the PhD degree with the School of Computer Science and Engineering, The University of New South Wales (UNSW), Australia. His research interests are deep learning application and recommender system.

Lina Yao (Senior Member, IEEE) received the master's and PhD degrees from the School of Computer Science, University of Adelaide, in 2014. She is currently an associate professor with the School of Computer Science and Engineering, The University of New South Wales (UNSW), Australia. Her research interest lies in machine learning and data mining, and applications to Internet of Things, information filtering and recommending, human activity recognition and brain computer interface. She is a member of the ACM.

Xianzhi Wang (Member, IEEE) received the PhD degree from the Harbin Institute of Technology. He is a lecturer with the School of Computer Science, University of Technology Sydney, Australia. His research interests include Internet of Things, artificial intelligence, information fusion, and recommender systems. He was a member ACM.

Aixin Sun (Member, IEEE) received the BASc (First Class Honours) and PhD degrees in computer engineering from the School of Computer Science and Engineering, Nanyang Technological University, Singapore, in 2001 and 2004, respectively. He is currently an associate professor with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. His research areas include text mining, social computing, multimedia, and digital libraries.

Quan Z. Sheng (Member, IEEE) received the PhD degree in computer science from the University of New South Wales (UNSW). He is currently a full professor and the head of the Department of Computing, Macquarie University. He has more than 390 publications as edited books and proceedings, refereed book chapters, and refereed technical papers in journals and conferences. He is a member of ACM.

For more information on this or any other computing topic, please visit our Digital Library at www.computer.org/csdl.