Learning Data-Driven Objectives to Optimize Interactive Systems

Ziming Li
University of Amsterdam
The Netherlands
z.li@uva.nl

Julia Kiseleva
Microsoft Research AI
USA
jukisele@microsoft.com

Alekh Agarwal
Microsoft Research AI
USA
alekha@microsoft.com

Maarten de Rijke
University of Amsterdam
The Netherlands
M.deRijke@uva.nl

Abstract

Effective optimization is essential for interactive systems to provide a satisfactory user experience. However, it is often challenging to find an objective to optimize for. Generally, such objectives are manually crafted and rarely capture complex user needs in an accurate manner. We propose an approach that infers the objective directly from observed user interactions. These inferences can be made regardless of prior knowledge and across different types of user behavior. We introduce Interactive System Optimizer (ISO), a novel algorithm that uses these inferred objectives for optimization. Our main contribution is a new general principled approach to optimizing interactive systems using data-driven objectives. We demonstrate the high effectiveness of ISO over several simulations.

1 Introduction

Interactive systems play an important role in assisting users in a wide range of tasks (Argelaguet et al., 2016; Kiseleva et al., 2016; Li et al., 2016; Williams et al., 2016). They are characterized by repeated interactions with humans which follow the request-response schema, where the user takes an action first, and then the interactive system produces a reply. Such interactions can continue for several iterations until the user decides to stop when they are either satisfied or frustrated with their experience. Importantly, an interactive system and its users always have a shared goal: for users to have the best experience. But despite their shared goal, only the user can observe their own experience, leaving interactive systems unable to directly optimize their behavior.

Currently, optimizing interactive systems relies on assumptions about user needs and frustrations (Li et al., 2017). Commonly, an objective function is manually designed for a particular task to reflect the quality of an interactive system in terms of user satisfaction (Kelly, 2009). The drawbacks of this approach are that a handcrafted objective function is heavily based on domain knowledge, that it is expensive to maintain, and that it does not generalize over different tasks. Moreover, it is impossible to design such functions when there is a lack of domain knowledge. Given an objective function, optimization can be done following the Reinforcement Learning (RL) paradigm; previous work does this by considering an interactive system as the agent and the underlying stochastic environment induced by the user (Hofmann et al., 2013; Li et al., 2016). However, user needs are inherently complex and depend on many different factors (Kosinski, Stillwell, and Graepe, 2013; Wei et al., 2017). Consequently, manually crafted objective functions rarely correspond to the actual user experience. Therefore, even an interactive system that maximizes such an objective

function is not expected to provide an optimal experience. In contrast with physically constrained environments (Silver et al., 2016; Levine et al., 2016a,b; Finn et al., 2017), the agent is supposed to adapt to a fixed environment that can not be changed. But users have difficulties in comprehending dramatic changes in an interactive system, so any changes should be done gradually to let users adapt to a newly optimized interactive system.

We propose a general perspective on how to overcome this discrepancy by simultaneously (1) inferring an objective function directly from data, namely user interactions with the system, and (2) iteratively, and step by step, optimizing the system for this data-driven objective. In contrast to more traditional perspectives on optimizing interactive systems, our data-driven objectives are learned from user behavior, instead of being hand-crafted. We hypothesize that by incorporating a data-driven objective, interactive systems can be optimized to an objective closer to the actual user satisfaction, unlike previous methods that optimize for assumed user preferences. To this respect, we introduce a novel algorithm: Interactive System Optimizer (ISO). It provides a principled approach by concurrently inferring data-driven objectives from user interactions and optimizing the interactive system accordingly. Thus, ISO does not depend on any domain knowledge.

Below, we start by formalizing the interaction process between a user and an interactive system as a Markov Decision Process (MDP) (Section 4). Then we make the following contributions:

- The first method that infers data-driven objectives solely from user interactions, that accurately reflect the users' needs without using any domain knowledge (Section 5).
- A novel algorithm, ISO, that optimizes an interactive system through data-driven objectives (Section 6). Since users will change their behavior following updates of the interactive system, we devise experiments to simulate interactions in the real world. Our experiments with different types of simulated user behavior (Section 7) show that ISO increases the expected state value significantly.

2   Related Work

Relevant work for this paper comes in two broad strands: how to optimize interactive systems and what reward signal can be used for optimization.

Optimizing interactive systems. Interactive systems can be optimized by direct and indirect optimization. Direct optimization aims at maximizing the user satisfaction directly; in contrast, indirect optimization solves a related problem while hoping that its solution also maximizes user satisfaction (Dehghani et al., 2017). Direct optimization can be performed using supervised learning or RL (Mohri, Rostamizadeh, and Talwalkar, 2012). Many applications of RL to optimizing interactive systems come from Information Retrieval (IR), recommender systems, and dialogue systems. Hofmann, Whiteson, and de Rijke (2011, 2013) apply RL to optimize IR systems; they use RL for online learning to rank and use interleaving to infer user preferences (Hofmann et al., 2013). Shani, Heckerman, and Brafman (2005) describe an early MDP-based recommender system and report on its live deployment. Li et al. (2016) apply RL to optimize dialogue systems; in particular, they optimize handcrafted reward signals such as ease of answering, information flow, and semantic coherence.

Rewards for interactive systems. When applying RL to the problem of optimizing interactive systems, we need to have rewards for at least some state-action pairs. Previous work typically handcrafts those, using, e.g., NDCG (Odijk et al., 2015) or clicks (Kutlu, Khetan, and Lease, 2018) before the optimization or the evaluation of the algorithm. Instead of handcrafting rewards, we recover them from observed interactions between the user and the interactive system using Inverse Reinforcement Learning (IRL). Ziebart, Dey, and Bagnell (2012) use IRL for predicting the desired target of a partial pointing motion in graphical user interfaces. Monfort, Liu, and Ziebart (2015) use IRL to predict human motion when interacting with the environment. IRL has also been applied to dialogues to extract the reward function and model the user (Pietquin, 2013). IRL is used to model user behavior in order to make predictions about it. But we use IRL as a way to recover the rewards from user behavior instead of handcrafting them and optimize an interactive system using these recovered rewards. The closest work to ours is by (Lowe et al., 2017), who learn a function to evaluate dialogue responses. However, the authors stop at evaluation and do not actually optimize the interactive system.

The key difference between our work and previous studies is that we first use recovered rewards from observed user interactions to reflect user needs and define interactive system objectives. Subsequently the interactive system can be optimized according to the defined data-driven objectives so as to improve the user experience.

3 Background

Reinforcement Learning (RL) and Inverse Reinforcement Learning (IRL) are the fundamental techniques used in the framework we propose in this paper.

In RL an agent learns to alter its behavior through trial-and-error interactions with its environment (Sutton and Barto, 1998). The goal of the agent is to learn a policy that maximizes the expected return. RL algorithms have successfully been applied to areas ranging from traditional games to robotics (Mnih and others, 2015; Silver et al., 2016; Levine et al., 2016a,b; Duan et al., 2016; Wang et al., 2016; Zhu et al., 2017).

The task of IRL is to extract a reward function given observed, optimal (or suboptimal) behavior of an agent over time (Ng and Russell, 2000). The main motivation behind IRL is that designing an appropriate reward function for most RL problems is non-trivial; this includes animal and human behavior (Abbeel and Ng, 2004), where the reward function is generally assumed to be fixed and can only be ascertained through empirical investigation. Thus inferring the reward function from historical behavior generated by an agent's policy can be an effective approach. Another motivation comes from imitation learning, where the aim is to teach an agent to behave like an *expert* agent. Instead of directly learning the agent's policy, other work first recovers the expert's reward function and then uses it to generate a policy that maximizes the expected accrued reward (Ng and Russell, 2000). Since the inception of IRL (Russell, 1998), several IRL algorithms have been proposed, including maximum margin approaches (Abbeel and Ng, 2004; Ratliff, Silver, and Bagnell, 2009), and probabilistic approaches such as (Ziebart et al., 2008) and (Boularias, Kober, and Peters, 2011). In the last few years, a number of adversarial IRL methods (Finn et al., 2016; Finn, Levine, and Abbeel, 2016; Ho and Ermon, 2016; Fu, Luo, and Levine, 2017) have been proposed because of its ability to adapt training samples to improve learning efficiency.

4 Modeling User-System Interactions

We assume that the agent is a user who interacts with the environment, an interactive system, with the goal of maximizing their expected rewards. As a running example we can consider a user who is interacting with a search engine. This process is modeled using a finite MDP (S, A, T, r, γ), in the following way:

- S is a set of states that represent responses from the interactive system to the user. S is finite as there are limited predefined number of responses that interactive system can return.
- A is a finite set of actions that the user can perform on the system to move between states. In case of search engine, a user can run a query, click on the returned results, reformulate a query etc.
- T is a transition distribution and $T(s, a, s')$ is the probability of transitioning from state s to state s' under action a at time t:

$$T(s' \mid s, a) = \mathbb{P}(S_{t+1} = s' \mid S_t = s, A_t = a). \tag{1}$$

  For search engines, being at the start page (which is s) a user is making an action a, e.g. running a query, and the engine redirects him to a result page (which is s').
- $r(s, a, s')$ is the expected immediate reward after transitioning from s to s' by taking action a. We compute the expected rewards for (state, action, next state) triples as:

$$r(s, a, s') = \mathbb{E}[R_t \mid S_t = s, A_t = a, S_{t+1} = s'], \tag{2}$$

  where R_t is reward at time t. In case of search engine, a user is getting a reward for finding a desired information. However, the rewards are not observed. For simplicity in exposition, we write rewards as $r(s)$ rather than $r(s, a, s')$ in our setting; the extension is trivial (Ng and Russell, 2000).
- $\gamma \in [0, 1]$ is a discount factor.

We write $\mathcal{P}$ to denote the set of interactive systems, i.e., triples of the form (S, A, T). System designers have control over the sets S, A, and the transition distribution, T, and T can be changed to optimize an interactive system.

The *user behavior strategy* is represented by a policy, which is a mapping, $\pi \in \Pi$, from states, $s \in S$, and actions, $a \in A$, to $\pi(a|s)$, which is the probability of performing action $A_t = a$ by the user when in state $S_t = s$. The observed history of interactions between the user and the interactive system, H, is represented as a set of trajectories, $\{\zeta_i\}_{i=1}^n$, drawn from a distribution Z, which is brought about by T, π, and D_0, where D_0 is the initial distribution of states. To simplify the problem, we assume that user behavior is homogeneous, i.e. one user generate H. A *trajectory* is a sequence of state-action pairs:

$$\zeta_i = S_0, A_0, S_1, A_1, \ldots, S_t, A_t, \ldots. \tag{3}$$

In this paper we act under assumption that the user is an optimal agent in terms of maximizing his reward under the system dynamics, and the system wants to improve the user experience over time by creating progressively easier MDPs to solve for the user. However, a interactive system cannot transition from all initial to s_0 in one step, due to design constraints. For example, if a user is searching for holidays destination, the system cannot redirect him to the final stage of booking a hotel because he needs to go through the necessary step, e.g. paying for it.

To summarize, we have described the basic principles of modeling interactions between users and an interactive system. Next, we detail how to define data-driven objectives that are used to optimize an interactive system.

5 Defining Data-driven Objectives

Defining Interactive System Objectives. We define the *quality* of an interactive system as the expected state value under an optimal user policy. The value of a state S_0 under a policy π is given as:

$$v_\pi(S_0) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \right], \tag{4}$$

where the expectation $\mathbb{E}_\pi[\cdot]$ is taken with respect to sequences of states $S_0, S_1, \ldots, S_t, \ldots$ drawn from the policy π and transition distribution T. The quality of the interactive system under user policy π is:

$$\mathbb{E}_{S_0 \sim D_0}[v_\pi(S_0)], \tag{5}$$

where D_0 is the initial distribution of states. In the proposed setting, the user goal is to find the best policy such that $\mathbb{E}_{S_0 \sim D_0}[v_\pi(S_0)]$ is maximized. $v_*(S_0)$ defines the maximum possible value of $v_\pi(S_0)$ as follows:

$$v_*(S_0) = \max_{\pi \in \Pi} v_\pi(S_0), \tag{6}$$

where Π is the set of possible user policies. We formulate the problem of finding the optimal interactive system's transition distribution, denoted T^*, in the following terms:

$$T^* = \arg\max_{T \in T} \mathbb{E}_{S_0 \sim D_0}[v_*(S_0)]. \tag{7}$$

Therefore, Eq. 7 represents the objective that we use to optimize an interactive system in order to improve the user experience. To estimate these interactive system objectives, we first need to recover R_t, which we will discuss next.

Recovering user rewards. We assume that continued user interactions with the system indicate a certain level of user satisfaction, which can be reflected by experienced rewards. In contrast with $\zeta_i \in H$ presented in Eq. 3, the complete history of interactions, $\hat{H}$, consists of trajectories $\hat{\zeta}_i \sim \hat{Z}$, which include the user reward R_t:

$$\hat{\zeta}_i = S_0, A_0, R_1, S_1, A_1, R_2 \ldots, R_t, S_t, A_t, \ldots. \tag{8}$$

The problem is that the true reward function is hidden and we need to recover it from the collected incomplete user trajectories, H, shown in Eq. 3. To address this challenge we apply IRL methods (Section 3), which are proposed to recover the rewards of different states, $r(s)$, for $\zeta_i \in H$.

Our assumptions about the user reward function is: given state feature functions $\phi : S_t \to \mathbb{R}^k$ that describes S_t as a k-dimensional feature vector, the true reward function $r(s)$ is a linear combination of the state features $\phi(s)$, which can be given as $r(s) = \theta^T \phi(s)$.

To uncover the reward weights θ, we use Maximum Entropy Inverse Reinforcement Learning (MaxEnt-IRL) (Ziebart et al., 2008), where the core idea is that trajectories with equivalent rewards have equal probability to be selected and trajectories with higher rewards are exponentially more preferred, which can be formulated as: $\mathbb{P}(\zeta_i \mid \theta) = \frac{1}{\Omega(\theta)} e^{\theta^T \phi(\zeta_i)} = \frac{1}{\Omega(\theta)} e^{\sum_{t=0}^{|\zeta_i|-1} \theta^T \phi(S_t)}$, where $\Omega(\theta)$ is the partition function. MaxEnt-IRL maximizes the likelihood of the observed data under the maximum entropy (exponential family) distribution. Once we have recovered the reward function $r(s)$ we can proceed to the optimization objectives presented in Eq. 7.

Oracle rewards. For comparison, we employ Distance Minimization Inverse Reinforcement Learning (DM-IRL) (El Asri, Laroche, and Pietquin, 2013; Burchfiel, Tomasi, and Parr, 2016) to have a perfectly recovered reward weight θ. DM-IRL directly attempts to regress the user's actual reward function that explains the given labels. DM-IRL uses discounted accrued features to represent the trajectory: $\psi(\zeta_i) = \sum_{t=0}^{|\zeta_i|-1} \gamma^t \phi(S_t)$, where γ is the discount factor. The score of a trajectory ζ_i is $\text{score}_{\zeta_i} = \theta^T \psi(\zeta_i)$. Since the exact score for each trajectory is supplied, the recovered rewards with DM-IRL are exactly the ground truth of reward functions, which can be regarded as oracle rewards.

6 Optimizing Interactive Systems

Algorithm 1 Interactive System Optimizer (ISO)

1: **Input:** Original system (S, A, T), r, γ, D_0.
2: **Output:** Optimized system (S, A, T^*)
3: Construct original MDP(S, A, T, r, γ)
4: $\pi_*(a|s) = RL(S, A, T, r, \gamma)$
5: Construct syst. MDP$^+(S^+, A^+, T^+, r^+, \gamma^+)$:
   - $S_t^+ = (S_t, A_t)$
   - $A_t^+ = S_{t+1}$
   - $T^+(S_{t+1}^+|S_t^+, A_t^+) = \pi_*(A_{t+1}|S_{t+1})$
   - $r(S_t^+)^+ = r(S_t)$
   - $\gamma^+ = \gamma$
6: $D_0^+ \sim (S_0 \sim D_0, A_0 \sim \pi_*(a|S_0))$
7: $\pi^+(A_t^+|S_t^+) = T(S_{t+1}|S_t, A_t)$
8: $\pi_*^+(a^+|s^+) = RL(S^+, A^+, T^+, r^+, \gamma^+)$
9: $T^*(S_{t+1}|S_t, A_t) = \pi_*^+(A_t^+|S_t^+)$

We start by explaining how to maximize the quality of an interactive system for a user behaving according to a fixed stationary policy π:

$$T_\pi^* = \arg\max_{T \in \mathcal{T}} \mathbb{E}_{S_0 \sim D_0}[v_\pi(S_0)]. \tag{9}$$

This problem is equivalent to finding the optimal policy in a new MDP$^+(S^+, A^+, T^+, r^+, \gamma^+)$, where the agent is an interactive system and the stochastic environment is a user. In MDP$^+$, the state S_t^+ is represented by a combination of the state S_t the user is in and the action A_t the user takes at time step t from the original MDP; the action A_t^+ is the original state S_{t+1}. The interactive system observes the current state S_t^+ and picks an action A_t^+ under the interactive system policy $\pi^+(A_t^+|S_t^+)$. Then the user returns the next state S_{t+1}^+ according to the transition distribution $T^+(S_{t+1}^+|S_t^+, A_t^+)$ which is inferred from the policy model $\pi(A_{t+1}|S_{t+1})$. Therefore, finding the optimal T_π^* from Eq. 9 is equivalent to finding the optimal π_*^+ for MDP$^+$ as follows:

$$\pi_*^+ = \arg\max_{\pi^+ \in \Pi^+} \mathbb{E}_{S_0^+ \sim D_0^+}[v_{\pi^+}(S_0^+)], \tag{10}$$

which can be done using an appropriate RL method such as Q-learning or Policy Gradient. D_0^+ is the initial distribution of states in MDP$^+$. After we have demonstrated how to optimize the interactive system for a given stationary policy, we return to the original problem of optimizing the interactive system for an optimal policy π_*.

We propose a procedure ISO that is presented in Algorithm 1 and has the following main steps:

Line 1: We assume that we have an estimate of the reward function $r(s)$ using one of the IRL methods described in Section 4, so we have as input: the original system (S, A, T), the reward function r, the discount factor γ, and the initial distribution of states D_0.
Line 2: ISO outputs the optimized interactive system (S, A, T^*).
Line 3: ISO formulates the original system as MDP(S, A, T, r, γ).
Line 4: ISO uses an appropriate RL algorithm to find the current policy $\pi_*(a|s)$ given the reward function r. *Line 5:* ISO transforms the original MDP(S, A, T, r, γ) into the new MDP$^+(S^+, A^+, T^+, r^+, \gamma^+)$. In our setting, S_t^+ has the same reward value as S_t. The discount factor γ^+ remains the same.
Line 6: ISO transforms D_0 to D_0^+ to match the distribution of first state-action pairs in the original MDP.
Line 7: The equivalence $\pi^+(A_t^+|S_t^+) = T(S_{t+1}|A_t, S_t)$ means that finding the optimal π_*^+ according to Eq. 10 is equivalent to finding the optimal T_π^* according to Eq. 9. Therefore, the transition distribution can be regarded as a policy network or a policy table from MDP's perspective depending on the policy learning method.
Line 8: We can use an appropriate RL algorithm to find $\pi_*^+(A_t^+|S_t^+)$.
Line 9: ISO extracts $T^*(S_{t+1}|S_t, A_t)$ from the optimal system policy $\pi_*^+(A_t^+|S_t^+)$. The extraction process is trivial: $T^*(S_{t+1}|S_t, A_t) = \pi^+(A_t^+|S_t^+)$. Therefore, ISO terminates by returning the *optimized* interactive system.

Once ISO has delivered the optimized system (S, A, T^*), we expose it to users so they can interact with it. We assume that users adjust their policy to T^*. After enough iterations the user policy will converge to the optimal one. Iterations between optimizing the interactive system for the current policy and updating the user policy for the current interactive system continue until both converge.

7 Experimental Setup

Designing an interactive system. For experimental evaluation we simulate an arbitrary interactive system where we need a finite set of states S, a finite set of actions A and a transition distribution T. We use simulations as we require to have a changeable environment which is provided to users for interactions. ISO is designed to improve an interactive system gradually (not in go). Features of a state $\phi(s)$ are fixed. For our experimental setup, we simulate the interactive system where $|S| = 64$ and $|A| = 4$. We work with a complex environment where a user can transition between any two states if these two states are connected. The connections between two states are predefined and fixed, but the transition distribution is changeable. In another word, for the same system in different runs, the connectivity graph of this systems is fixed and will not be changed once it is sampled at the very beginning. We use a hyper-parameter, the connection factor cf, to define the number of possible next states after the user has taken one specific action at the current state. For an initial interactive system, D_0 is randomly sampled as well as T. At each iteration ISO delivers T^*, which substitutes the initial T.

Modeling user behavior. To model user behavior we require a true reward function $r(s)$, and an optimal user policy π_*. We utilize a linear reward function $r(s)$ by randomly assigning 25% of the states reward 1, while all others receive 0. As we use one-hot features for each state, $r(s)$ is guaranteed to be linear. We use a soft value iteration method (Ziebart, 2010) to obtain the optimal user policy π^*.

The quality of the recovered reward functions user behavior is influenced by how trajectories are created, which in turn can affect the performance of ISO as it relies on $r(s)$ to optimize an interactive system. We experiment with the following types of user trajectories:

Optimal: Users know how to behave optimally in an interactive system to satisfy their needs. To simulate it, we use π_* trained with the real reward function.
SubOptimal: Not all users know the system well, which means that the demonstrated behavior is a mixture of optimal and random. We propose two different methods to get suboptimal behavior. The degree of optimality of user behavior is controlled by (1) the proportion of random behavior[1] (2) or the user action noise, which are collectively called the noise factor (NF) $\in [0.0, 1.0]$.

---

[1]To model suboptimal user behavior we use two user policies: (1) an optimal user policy π_*; and (2) an adversarial policy $(1 - \pi_*)$. We include an adversarial policy instead of a random one because it is the hardest case as users behave opposite of what we expect.

Mix of Behaviors (MB): The final dataset H is a mix of trajectories generated by the optimal policy and the adversarial policy[2].

Noise in Behavior (NB): In this case, the trajectories in H are generated from the optimal policy but we add noise to the user actions to get suboptimal behavior [3].

The generated history of user interactions H represents the case of *unlabelled trajectories* which will be fed to MaxEnt-IRL. To generate a dataset with *labelled trajectories* $\hat{H}$ we calculate the score using $r(s)$ as shown in Section 5. $\hat{H}$ are the input to DM-IRL.

At each iteration, we sample the following datasets reflecting different types of history of user interactions: $\hat{H}$, $H_{Optimal}$, $H_{SubOptimal-0.2-MB}$, $H_{SubOptimal-0.6-MB}$, $H_{SubOptimal-0.2-NB}$, $H_{SubOptimal-0.6-NB}$ each of size $15,000$ and $|\zeta_i| \in [30, 40]$.

Evaluation process. To evaluate the performance of ISO, we report the expected state value under optimal policy (Eq. 5) for an *initial* interactive system and an *optimized* one, which we derive after around 100 iterations. A higher expected state value means users are more satisfied while interacting with the interactive system. We initialize 40 different initial interactive systems by randomly sampling reward functions and transition distribution, and report the overall performance over these 40 systems. The true reward functions and the connectivity graphs of these sampled systems are fixed in the whole optimizing process.

8 Results and Discussion

Improving interactive systems with ISO. Figure 1 shows how the quality of the interactive system increases with each iteration of ISO in terms of different connection factors. As expected, when the user gives feedback about the quality of the trajectories (IRL-labelled), the task is simpler and ISO manages to get high improvements with the oracle rewards. However, the picture changes when we hide the labels from the trajectories. Without labels, ISO relies on the optimality of user behavior to recover the reward function. As the optimality decreases, so does the behavior of ISO, and the performance decays. With the oracle rewards from DM-IRL, ISO converges quite fast – as we can see in Figure 1(b) and Figure 1(c), after 20 iterations the expected state value begins to plateau. Most improvements happen in the first several iterations. Thus, ISO works with accurately labeled trajectories, but usually obtaining high-quality labels is intractable and expensive in a real interactive system because the real rewards are invisible. We report it as the oracle performance in our experiment.

With respect to unlabelled trajectories ISO is able to improve the initial expected state value. In Figure 1(a), the influence of the noise factor and types of trajectories (MB or NB) is clear. However, in Figure 1(c) where there are fewer connections between two states, only the convergence speeds of different curves are different but they all converge to the same state value eventually. ISO manages to optimize the interactive system even though the user trajectories are quite noisy.

More remarkable, the convergence speed and final converged values are different depending on the connection factors. As we can see, it is more difficult to get high performance when there are more connections between different states in the predefined systems. More connections mean that more possible trajectories could be taken and it is intractable for MaxEnt-IRL to learn a reward function from this kind of situation. By contrast, in Figure 1(c), each state-action pair can only have two possible next states and the final average state value is much higher than the system in Figure 1(a).

Impact of ISO components. The performance of ISO depends on its two components: (1) RL methods used to optimize the user policy π for the original MDP and system policy π^+ for transformed MDP$^+$; and (2) IRL methods – to recover the true reward function. The dependence on RL methods is obvious – the end result will only be as good as the quality of the final optimization, so an appropriate method should be used. The performance of ISO can be influenced by the quality of the recovered reward functions, $r(s)$. For the case of labeled trajectories, the values of $r(s)$ recovered by DM-IRL are identical to the ground truth since a regression model is used and we have the exact label for each user trajectory. For the case of unlabelled trajectories, the quality of the recovered reward function is

---

[2]E.g., $NF = 0.2$ means that 20% of the trajectories are generated with the adversarial policy.

[3]E.g., $NF = 0.2$ means the probability is 20% that the user will not choose the action with the highest probability in the optimal policy.
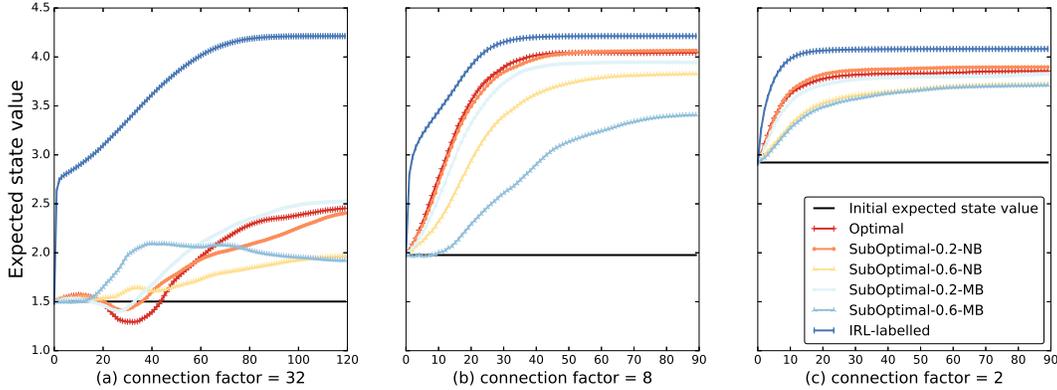
Figure 1: Performance of ISO over 40 randomly sampled systems. The x-axis is the number of iterations of ISO and the y-axis is the expected state value.

worse than DM-IRL. MaxEnt-IRL can only give a general overview of $r(s)$ if the user trajectories are optimal. If there are not enough constraints on the connections between states, with each iteration of running ISO, the shape of the sampled trajectories becomes more similar, which means that most trajectories pass by the same states and the diversity of trajectories decreases. We found that this makes it even more difficult to recover $r(s)$ and the MaxEnt-IRL quality deteriorates with the number of iterations, which results in lower performance in Figure 1(a). Since we are using one of the most common IRL methods in this work, more advanced IRL methods could be adopted to achieve higher performance. Hence, improving the performance of IRL methods is likely to significantly boost the performance of ISO.

9 Conclusions and Future work

We have recognized that previous work on interactive systems has relied on numerous assumptions about user preferences. As a result, interactive systems have been optimized for manually designed objectives that do not align with the true user preferences and cannot be generalized across different domains. To overcome this discrepancy, we have proposed a novel algorithm: the Interactive System Optimizer (ISO), that both infers the user objective from their interactions, and optimizes the interactive system according to this inferred objective.

Firstly, we model user interactions using MDP, where the agent is the user, and the stochastic environment is the interactive system. User satisfaction is modeled by rewards received from interactions, and the user interaction history is represented by a set of trajectories. Secondly, we infer the user needs from the observed interactions, in the form of a data-driven objective. Importantly, our method works without any domain knowledge, and is thus even applicable when prior knowledge is absent. Thirdly, ISO iterates between optimizing the interactive system for the current inferred objective; and letting the user adapt to the new system behavior. This process repeats until both the user and system policies converge. Our experimental results show that ISO robustly improves the user experience across different types of user behavior.

Since optimizing an interactive system based on data-driven objectives is novel, many promising directions for future work are possible. For instance, while ISO performs well for users with a single goal, this approach could be extended to settings with multiple goals. Similarly, extensions considering more personalized goals could benefit the overall user experience. Finally, investigating the scalability and real world applicability of ISO could open many research possibilities.

References

Abbeel, P., and Ng, A. Y. 2004. Apprenticeship learning via inverse reinforcement learning. In *ICML*, 1–8. ACM.

Argelaguet, F.; Hoyet, L.; Trico, M.; and Lécuyer, A. 2016. The role of interaction in virtual embodiment: Effects of the virtual hand representation. In *VR*, 3–10. IEEE.

Boularias, A.; Kober, J.; and Peters, J. 2011. Relative entropy inverse reinforcement learning. In *AISTATS*, 182–189.

Burchfiel, B.; Tomasi, C.; and Parr, R. 2016. Distance minimization for reward learning from scored trajectories. In *AAAI*, 3330–3336. AAAI Press.

Dehghani, M.; Zamani, H.; Severyn, A.; Kamps, J.; and Croft, W. B. 2017. Neural ranking models with weak supervision. In *SIGIR*, 65–74. ACM.

Duan, Y.; Schulman, J.; Chen, X.; Bartlett, P. L.; Sutskever, I.; and Abbeel, P. 2016. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*.

El Asri, L.; Laroche, R.; and Pietquin, O. 2013. Reward shaping for statistical optimisation of dialogue management. In *SLSP*, 93–101. Springer.

Finn, C.; Christiano, P.; Abbeel, P.; and Levine, S. 2016. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*.

Finn, C.; Yu, T.; Zhang, T.; Abbeel, P.; and Levine, S. 2017. One-shot visual imitation learning via meta-learning. In *CoRL*, 357–368.

Finn, C.; Levine, S.; and Abbeel, P. 2016. Guided cost learning: Deep inverse optimal control via policy optimization. In *ICML*, 49–58. ACM.

Fu, J.; Luo, K.; and Levine, S. 2017. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*.

Ho, J., and Ermon, S. 2016. Generative adversarial imitation learning. In *Advances in neural information processing systems*, 4565–4573.

Hofmann, K.; Schuth, A.; Whiteson, S.; and de Rijke, M. 2013. Reusing historical interaction data for faster online learning to rank for IR. In *WSDM*, 183–192. ACM.

Hofmann, K.; Whiteson, S.; and de Rijke, M. 2011. Balancing exploration and exploitation in learning to rank online. In *ECIR*, 251–263. Springer.

Hofmann, K.; Whiteson, S.; and de Rijke, M. 2013. Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval. *Information Retrieval Journal* 16(1):63–90.

Kelly, D. 2009. Methods for evaluating interactive information retrieval systems with users. *Foundations and Trends in Information Retrieval* 3(1–2):1–224.

Kiseleva, J.; Williams, K.; Awadallah, A. H.; Zitouni, I.; Crook, A.; and Anastasakos, T. 2016. Predicting user satisfaction with intelligent assistants. In *SIGIR*, 45–54. ACM.

Kosinski, M.; Stillwell, D.; and Graepe, T. 2013. Private traits and attributes are predictable from digital records of human behavior. *PNAS* 110:5802–5805.

Kutlu, M.; Khetan, V.; and Lease, M. 2018. Correlation and prediction of evaluation metrics in information retrieval. *arXiv preprint arXiv:1802.00323*.

Levine, S.; Finn, C.; Darrell, T.; and Abbeel, P. 2016a. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research* 17(39):1–40.

Levine, S.; Pastor, P.; Krizhevsky, A.; Ibarz, J.; and Quillen, D. 2016b. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research* 173–184.

Li, J.; Monroe, W.; Ritter, A.; Jurafsky, D.; Galley, M.; and Gao, J. 2016. Deep reinforcement learning for dialogue generation. In *EMNLP*, 1192–1202.

Li, Z.; Kiseleva, J.; de Rijke, M.; and Grotov, A. 2017. Towards learning reward functions from user interactions. In *ICTIR*, 941–944. ACM.

Lowe, R.; Noseworthy, M.; Serban, I. V.; Angelard-Gontier, N.; Bengio, Y.; and Pineau, J. 2017. Towards an automatic turing test: Learning to evaluate dialogue responses. In *ACL*, 1116–1126.

Mnih, V., et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Mohri, M.; Rostamizadeh, A.; and Talwalkar, A. 2012. *Foundations of Machine Learning*. MIT Press.

Monfort, M.; Liu, A.; and Ziebart, B. 2015. Intent prediction and trajectory forecasting via predictive inverse linear-quadratic regulation. In *AAAI*, 3672–3678. AAAI Press.

Ng, A. Y., and Russell, S. J. 2000. Algorithms for inverse reinforcement learning. In *ICML*, 663–670. ACM.

Odijk, D.; Meij, E.; Sijaranamual, I.; and de Rijke, M. 2015. Dynamic query modeling for related content finding. In *SIGIR*, 33–42. ACM.

Pietquin, O. 2013. Inverse reinforcement learning for interactive systems. In *Workshop on Machine Learning for Interactive Systems*, 71–75. ACM.

Ratliff, N. D.; Silver, D.; and Bagnell, J. A. 2009. Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots* 27(1):25–53.

Russell, S. 1998. Learning agents for uncertain environments. In *COLT*, 101–103. ACM.

Shani, G.; Heckerman, D.; and Brafman, R. I. 2005. An MDP-based recommender system. *Journal of Machine Learning Research* 6(Sep):1265–1295.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529(7587):484–489.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement learning: An introduction*. MIT press Cambridge.

Wang, J. X.; Kurth-Nelson, Z.; Tirumala, D.; Soyer, H.; Leibo, J. Z.; Munos, R.; Blundell, C.; Kumaran, D.; and Botvinick, M. 2016. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*.

Wei, H.; Zhang, F.; Yuan, N. J.; Cao, C.; Fu, H.; Xie, X.; Rui, Y.; and Ma, W.-Y. 2017. Beyond the words: Predicting user personality from heterogeneous information. In *WSDM*, 305–314. ACM.

Williams, K.; Kiseleva, J.; Crook, A. C.; Zitouni, I.; Awadallah, A. H.; and Khabsa, M. 2016. Detecting good abandonment in mobile search. In *WWW*, 495–505.

Zhu, Y.; Mottaghi, R.; Kolve, E.; Lim, J. J.; Gupta, A.; Fei-Fei, L.; and Farhadi, A. 2017. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, 3357–3364. IEEE.

Ziebart, B. D.; Maas, A. L.; Bagnell, J. A.; and Dey, A. K. 2008. Maximum entropy inverse reinforcement learning. In *AAAI*, 1433–1438. AAAI Press.

Ziebart, B.; Dey, A.; and Bagnell, J. A. 2012. Probabilistic pointing target prediction via inverse optimal control. In *IUI*, 1–10. ACM.

Ziebart, B. D. 2010. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. Ph.D. Dissertation, Carnegie Mellon University.