Optimizing Interactive Systems with Data-Driven Objectives

Julia Kiseleva
University of Amsterdam

Optimizing Interactive Systems with Data-Driven Objectives

Ziming Li [*1] **Artem Grotov** [*1] **Julia Kiseleva** [1 2] **Maarten de Rijke** [1] **Harrie Oosterhuis** [1]

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 accurately. Conversely, 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. Then 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 GridWorld simulations.

1. Introduction

Interactive systems play an important role in assisting users in a wide range of tasks, they are characterized by doing so through repeated interactions with humans. For instance, if users are looking for information, interactive systems can assist them in the form of web search engines (Williams et al., 2016), dialogue systems (Li et al., 2016), or intelligent assistants (Kiseleva et al., 2016). Here users interact with systems following the request-response schema: first the user takes an action, which could be a question or a query, then the interactive system produces a reply, which could be an answer or a search engine result page. Such interactions can continue for several iterations until the user decides to stop when he is either satisfied or frustrated with his experience. Importantly, an interactive system and its users always have a shared goal: for users to have the best experience. Thus, both a system and its users are expected to behave accordingly, e.g., the user sends the query that he expects to lead him to the desired results and the interactive system provides the search results that are most helpful to the user. But despite their shared goal, only the user can observe their own experience, leaving interactive systems unable to directly optimize their behavior.

Currently, the optimization of interactive systems relies on assumptions about user needs and frustrations (Li et al., 2017). Commonly, an objective function is manually designed to reflect the quality of an interactive system in terms of user satisfaction. The drawback of this approach is that it is heavily based on domain knowledge, e.g., clicks on search results (Luo et al., 2015b) or the cross-entropy between generated replies and predefined answers (Li et al., 2016). Additionally, a handcrafted objective function is expensive to maintain and does not generalize over different domains. 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 stochastic environment as a user (Hofmann et al., 2013a; Li et al., 2016). However, user needs are inherently complex and depend on many different factors (Kosinski et al., 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 the optimal experience.

Conversely, we propose an approach that overcomes this discrepancy by simultaneously: inferring an objective function directly from data, namely user interactions with the system; and optimizing the system for this data-driven objective. Thereby in contrast to traditional perspectives, our data-driven objectives are learned from user behavior, instead of being hand-crafted. We suppose 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. We seek to answer the following **main research question**:

> *Can we optimize an interactive system for users through data-driven objectives?*

To answer this research question, we introduce a novel algorithm: Interactive System Optimizer (ISO). It provides a new principled approach by concurrently: inferring data-driven objectives from their interactions; and optimizing

[*]Equal contribution [1]Informatics Institute, University of Amsterdam, Amsterdam, The Netherlands [2]UserSat.com. Correspondence to: Ziming Li <z.li@uva.nl>, Artem Grotov <a.grotov@uva.nl>.

the interactive system accordingly. Thus, ISO does not depend on any domain knowledge.

In this paper, we start by formalizing the interaction process between a user and an interactive system as a Markov Decision Process (MDP) (Section 3). 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 4).
- A novel algorithm, ISO, that optimizes an interactive system through data-driven objectives (Section 5). Our experiments with a different types of simulated user behavior (Section 6) show ISO has higher performance by increasing the expected state value at least 89% and up to 136% (Section 7).

2. 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 & 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 et al., 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 et al., 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 & 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 et al., 2000).

Since the inception of IRL by Russell (1998), several different IRL algorithms have been proposed. Generally, these methods assume the environment can be modelled as an MDP. Many IRL methods (Ziebart et al., 2008; Lopes et al., 2009) model the reward functions as linear combinations of hand selected state features, these linear functions are then chosen so that they can explain the behavior of the agent. However, linear functions are rarely capable of explaining complex behavior in real environments. Thus, other work has introduced methods for non-linear reward functions, such as margin-based methods (Bagnell et al., 2007; Ratliff et al., 2009; Levine et al., 2010), that recover non-linear reward functions through feature construction while assuming the demonstrated behavior is optimal. For suboptimal behavior Levine et al. (2011) combine probabilistic reasoning about stochastic expert behavior with non-linear reward functions, outperforming prior methods in suboptimal settings. To avoid their reliance on handcrafted state features, recent methods have exploited the representational capacity of neural networks to approximate complex reward functions without meticulous feature engineering (Wulfmeier et al., 2015). Alternatively, Finn et al. (2016) explore how Inverse Optimal Control can learn behaviors and recover reward functions from demonstrations in high-dimensional robotics settings. Another branch of IRL uses evaluated suboptimal demonstrations where the agent's trajectories are scored by an expert. By changing the expert's role from a demonstrator to a judge, El Asri et al. (2013) learn reward functions from the scores even when the transition functions are unknown. Burchfiel et al. (2016) show that this IRL method is robust to labelling errors in scored trajectories; a disadvantage of these approaches is that obtaining scores is often very costly.

In this paper we propose ISO the critical difference with previous work is that interactive systems are optimized while the objectives are inferred from recovered user reward functions.

3. Modeling User-System Interactions

In this section we explain how we model user interactions (Section 3.1) and define different types of user behavior in interactive systems (Section 3.2).

3.1. Modeling user behavior in interactive systems

We assume that the agent is a user who interacts with the interactive system with the goal of maximizing his expected rewards. This process is modelled using a finite MDP: (S, A, τ, r, γ), in the following way:

1. S is a finite set of states that represent responses from the interactive system to the user.
2. A is a finite set of actions that the user can perform on the system to move between states.
3. τ is a transition probability table and $\tau(s, a, s')$ is the probability of transitioning from state s to state s' under action a at time t:

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

   The set of all possible τ is T.
4. $r(s, a, s')$ 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:

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

where R_t is reward at time t.
For similarity in exposition, we write rewards as $r(s)$ rather than $r(s, a, s')$ in our setting; the conversion is trivial (Ng et al., 2000).

5. $\gamma \in [0, 1]$ is a discount factor.

We write P to denote the set of interactive systems, i.e., triples of the form (S, A, τ). System designers have a control over the sets S, A and the transition probability table, τ, and τ 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 the state $S_t = s$:

$$\pi(a|s) = \mathbb{P}(A_t = a \mid S_t = s). \tag{3}$$

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 τ, π, and D_0, where D_0 is the initial distribution of states. A *trajectory* is a sequence of state-action pairs:

$$\zeta_i = S_0, A_0, S_1, A_1, \ldots, S_t, A_t, \ldots. \tag{4}$$

Next we introduce different ways of generating ζ_i.

3.2. Defining different types of user behaviors

There are two aspects to characterize the type of user behaviors that influence the shape of $\zeta_i \in H$:

1. **What are the underlying principles that govern how users make decisions while interacting?**
   (a) *Randomly.* Users have no prior information about an interactive system and behave randomly.
   (b) *Optimally.* Users know how to behave optimally in an interactive system to satisfy their needs.
   (c) *Suboptimally.* The behavior is suboptimal, which is better than random but not as good as optimal.
2. **Are users giving explicit feedback about the quality of an interactive system?**
   (a) *Yes.* Users provide us with feedback about the quality of the interactive system by labelling $\zeta_i \in H$.
   (b) *No.* Users do not give us any feedback and the $\zeta_i \in H$ are unlabelled.

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 which are used to optimize an interactive system.

4. Defining Data-driven Objectives

In this section we present a way to convert user needs to interactive system objectives (Section 4.1) and explain how these objectives can be estimated (Section 4.2).

4.1. Defining Interactive System Objectives

We define the *quality* of an interactive system as the expected quality of trajectories under optimal user policy. The quality of the i-th trajectory, ζ_i, is the discounted sum of the rewards of each state in the trajectory: $\sum_{t=0}^{\infty} \gamma^t R_{t+1}$. The expected quality of the i-th trajectory, ζ_i, is the value of its starting state, S_0, in interactive system under user policy π:

$$\upsilon_\pi(S_0) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \right], \tag{5}$$

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 user policy π and transition probability table τ. The quality of the interactive system under user policy π is:

$$\mathbb{E}_{S_0 \sim D_0}[\upsilon_\pi(S_0)], \tag{6}$$

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

$$\upsilon_*(S_0) = \max_{\pi \in \Pi} \upsilon_\pi(S_0), \tag{7}$$

where Π is the set of possible user policies. We formulate the problem of finding the optimal interactive system's transition probability table, denoted τ^*, in the following terms:

$$\tau^* = \arg\max_{\tau \in T} \mathbb{E}_{S_0 \sim D_0}[\upsilon_*(S_0)]. \tag{8}$$

Therefore, Eq. 8 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.

4.2. 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. 4, the complete history of interactions, $\hat{H}$, consists of trajectories $\hat{\zeta}_i \sim \hat{Z}$, which include the user reward R_t:

$$\hat{\zeta}_i = S_0, A_0, R_1, S_1, A_1, R_2 \ldots, R_t, S_t, A_t, \ldots. \tag{9}$$

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. 4. To address this challenge we apply IRL methods (Section 2), which are proposed to recover the rewards of different states, $r(s)$, for trajectories $\zeta_i \in H$.

Assumptions about user reward function:

- There is a state feature function, $\phi : S_t \rightarrow \mathbb{R}^k$, which can describe a state with a k-dimensional feature vector.

- There exist unknown true reward weights $\theta \in \mathbb{R}^k$ which linearly map the state features, $\phi : S_t$, to a reward value, $r(s) = \theta^T \phi(s)$, which represent the satisfaction of a user for this state.

We adopt two types of IRL method, as we have assumed two scenarios for user feedback (Section 3.2).

Unlabeled trajectories. We use Maximum Entropy Inverse Reinforcement Learning (MaxEnt-IRL) (Ziebart et al., 2008) if users give no feedback about their experience with an interactive system. The core idea of this method is that trajectories with equivalent rewards have equal probabilities to be selected and trajectories with higher rewards are exponentially more preferred, which can be formulated as:

$$\mathbb{P}(\zeta_i \mid \theta) = \frac{1}{\Omega(\theta)} e^{\theta^T \phi(\zeta_i)} = \frac{1}{\Omega(\theta)} e^{\sum_{t=0}^{|\zeta_i|-1} \theta^T \phi(S_t)}, \quad (10)$$

where $\Omega(\theta)$ is the partition function. MaxEnt-IRL maximizes the likelihood of the observed data under the maximum entropy (exponential family) distribution. Its task can be seen as a classification problem where each trajectory represents one class. MaxEnt-IRL employs gradient descent to update the reward weights θ.

Labeled trajectories. We employ Distance Minimization Inverse Reinforcement Learning (DM-IRL) (Burchfiel et al., 2016) for scenarios when users do give us feedback. For DM-IRL, it is not essential for the trajectories to be optimal because trajectories, $\zeta_i \in H$, are labeled. Therefore, 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:

$$\psi(\zeta_i) = \sum_{t=0}^{|\zeta_i|-1} \gamma^t \phi(S_t), \quad (11)$$

where γ is the discount factor. The score of a trajectory ζ_i is assumed to be:

$$\text{score}_{\zeta_i} = \theta^T \psi(\zeta_i). \quad (12)$$

Since the score for each trajectory is supplied, the task reduces to a normal regression problem.

Once we have recovered the reward function $r(s)$ we can proceed to the optimization objectives presented in Eq. 8.

5. Optimizing Interactive System with Data-driven Objectives

In this section, we aim to find the best interactive system for an optimally behaving user. This is equivalent to finding the optimal transition probability table τ^*, defined in Eq. 8.

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

$$\tau_\pi^* = \arg\max_{\tau \in T} \mathbb{E}_{S_0 \sim D_0}[\upsilon_\pi(S_0)]. \quad (13)$$

This problem is equivalent to finding the optimal policy in a new MDP$^+(S^+, A^+, \tau^+, r^+, \gamma^+)$, 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 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 probability $\tau^+(S_{t+1}^+|S_t^+, A_t^+)$ conditioned on the policy $\pi(A_{t+1}|S_{t+1})$ and transition probability $\tau(S_{t+1}|S_t, A_t)$ from the original MDP.

Therefore, finding the optimal τ_π^* from Eq. 13 is equivalent to finding the optimal π_*^+ for MDP$^+$ as follows:

$$\pi_*^+ = \arg\max_{\pi^+ \in \Pi^+} \mathbb{E}_{S_0^+ \sim D_0^+}[\upsilon_{\pi^+}(S_0^+)], \quad (14)$$

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_*

Algorithm 1 Interactive System Optimizer (ISO)

1: **Input:** Original system (S, A, τ), r, γ, D_0.
2: **Output:** Optimized system (S, A, τ^*)
3: Construct original MDP(S, A, τ, r, γ)
4: $\pi_*(a|s) = RL(S, A, \tau, r, \gamma)$
5: Transform MDP to MDP$^+(S^+, A^+, \tau^+, r^+, \gamma^+)$:
- $S_t^+ = (S_t, A_t)$
- $A_t^+ = S_{t+1}$
- $\tau^+(S_{t+1}^+|S_t^+, A_t^+) = \tau(S_{t+1}|S_t, A_t) \cdot \pi_*(A_{t+1}|S_{t+1})$
- $r(S_t^+)^+ = r(S_t)$
- $\gamma^+ = \gamma$

6: $D_0^+ \sim (S_0 \sim D_0, A_0 \sim \pi_*(a|S_0))$
7: $\pi^+(A_t^+|S_t^+) = \tau(S_{t+1}|S_t, A_t)$
8: $\pi_*^+(a^+|s^+) = RL(S^+, A^+, \tau^+, r^+, \gamma^+)$
9: $\tau^*(S_{t+1}|S_t, A_t) = \pi_*^+(A_t^+|S_t^+)$

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 3, so we have input: original system (S, A, τ), reward function r, discount factor γ and initial distribution of states D_0.
Line 2: ISO outputs the optimized interactive system (S, A, τ^*).

Line 3: ISO formulates the original system as $\text{MDP}(S, A, \tau, r, \gamma)$.
Line 4: ISO uses an appropriate RL algorithm to find the optimal user policy $\pi_*(a|s)$.
Line 5: ISO transforms the original $\text{MDP}(S, A, \tau, r, \gamma)$ into the new $\text{MDP}^+(S^+, A^+, \tau^+, r^+, \gamma^+)$. 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 $\pi^+(A_t^+|S_t^+) = \tau(S_{t+1}|A_t, S_t)$ means that finding the optimal π_*^+ according to Eq. 14 is equivalent to finding the optimal τ_π^* according to Eq. 13.
Line 8: We can use an appropriate RL algorithm to find $\pi_*^+(A_t^+|S_t^+)$.
Line 9: ISO extracts $\tau^*(S_{t+1}|S_t, A_t)$ from the optimal system policy $\pi_*^+(A_t^+|S_t^+)$. The extraction process is trivial: $\tau^*(S_{t+1}|S_t, A_t) = \pi^+(A_t^+|S_t^+)$. Therefore, ISO terminates by returning the *optimized* interactive system.

Once ISO has delivered the optimized system (S, A, τ^*), we expose it to users so they can interact with it. Hence, it is natural to assume that users adjust their policy towards τ^*. After enough iterations the user policy will converge to the optimal one. The iteration between optimizing the interactive system for the current policy and updating the user policy for the current interactive system continues until both converge. The proof of convergence closely follows proofs of convergence of RL algorithms (Sutton & Barto, 1998). Value iteration converges to unique global optima because the Bellman backup operator which it uses is a contraction operator. The updates to the interactive system are performed using value iteration in the transformed MDP which is a contraction. Then the user policy is also updated using value iteration – another contraction. A combination of contractions is a contraction, therefore ISO converges to a unique optimum.

In summary, we have presented the Interactive System Optimizer (ISO). It optimizes an interactive system using data-driven objectives. It works by transforming the original MDP, solving it and using its solution to yield the optimal transition probability table in the original MDP.

6. Experimental Setup

In this section we explain our experimental setup to test the performance of the Interactive System Optimizer (ISO).

Designing an interactive system. To design an interactive system we need a finite set of states S, a finite set of actions A and a transition probability table τ. Features of a state $\phi(s)$ are fixed. We use GridWorld as an example of an interactive system. In our setting, GridWorld is an $N \times N$ grid of states, where $N = 6$ ($|S| = 36$). It supports four possible actions per state ($|A| = 4$) that represent the four directions in which a user can move. In standard settings of GridWorld, from any state a user can only jump to neighboring ones, so the transition probability table, τ, are static. For our experimental setup, we design a more complex environment where a user can move between any two states and the transition probability is changeable. For an initial interactive system, D_0 is randomly sampled as well as τ. At each iteration ISO delivers τ^*, which substitutes the initial τ.

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 with reward 1 while others with 0. As we use one-hot features for each state, $r(s)$ is guaranteed to be linear. We use 25% because we see quantitatively same performance when the proportion of the rewarded states changes. We hypothesize that the complexity of the problem is proportional to the entropy in the reward function because it leads to higher entropy in the observed trajectories and higher variance in the reward estimate. We use value iteration method (Ziebart, 2010) to obtain the optimal user policy π^*. There are two main aspects of user behavior, as introduced in Section 3.2:

Types of user trajectory: Suboptimality in user behavior influences the quality of the recovered reward functions, which in turn can affect the performance of ISO as it relies on $r(s)$ to optimize an interactive system. To simulate optimal user behavior, we use π_* trained with the real reward function. To model suboptimal user behavior we use two user policies: (1) an optimal user policy π_*; and (2) an adversarial policy $(1-\pi_*)$. We included an adversarial policy instead of a random one because it is the hardest case as users behave opposite of what we expect. The final dataset H is a mix of trajectories generated by two policies. The noise factor (NF) $\in [0.0, 1.0]$[1] determines the proportion of trajectories in H generated by adversarial policy. Hence, user trajectories are modelled as follows: (1) adversarial (adv) when $NF = 1.0$; (2) optimal (opt) – $NF = 0.0$; and (3) suboptimal (sub) – $NF = 0.4$.

Types of user feedback: The generated history of user interactions H represents the case of *unlabelled trajectories*. To generate a dataset with *labelled trajectories* $\hat{H}$ we calculate the score using $r(s)$ as shown in Eq. 12.

At each iteration, we sample six datasets reflecting different types of histories of user interactions: $\hat{H}_{adv}$, $\hat{H}_{opt}$, $\hat{H}_{sub}$, H_{adv}, H_{opt}, H_{sub}, each of size $10,000$ and $|\zeta_i| \in [20, 30]$.

Evaluation process. To evaluate the performance of ISO, we report the expected state value under optimal policy (Eq. 6) for an *initial* interactive system and an *optimised* one, which we derive after 200 iterations. A higher expected state value means users are more satisfied while interacting with the interactive system. We randomly initialize 100 reward functions and report the overall perfor-

[1]For example, $NF = 0.1$ means that 10% of the trajectories are noisy and generated with the adversarial policy.

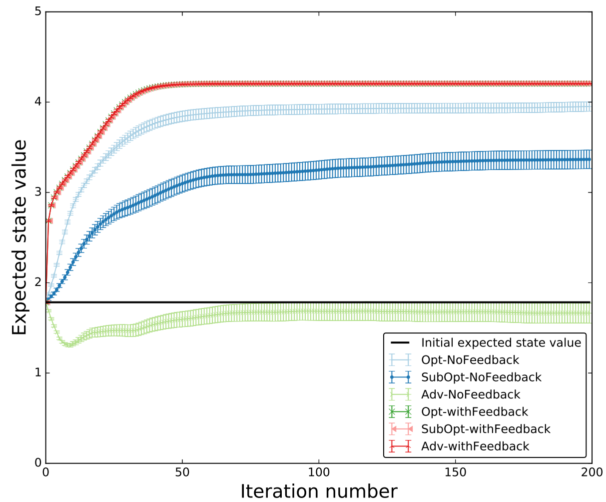

Figure 1. Performance of ISO. Expected state value over 100 random functions with standard error. Note that the curves for Opt-withFeedback, SubOpt-withFeedback and Adv-withFeedback have almost the same shape.

mance. Also, relative improvements are computed. We use a t-test to show statistical significance ($p < 0.01$) of derived relative improvements. We separately show the quality of the selected IRL methods for different types of user behavior.

In summary, we have described our experimental setup, which includes the design of an interactive system, user behavior simulation, and evaluation metrics. Next, we present and discuss our experimental results.

7. Results and Discussion

In this section, we present the experimental results and analyze the performance of ISO and its robustness.

Performance of ISO. Table 1 displays the expected state values of the *initial* and *optimized* interactive system and the relative improvement (Impr) that ISO achieves using labelled and unlabeled trajectories of the adversarial, optimal, and suboptimal user behavior. ISO manages to improve the interactive system in all cases but one – when there is no feedback and the user behavior is adversarial. As expected, when the user gives feedback about the quality of the trajectories, the task is simpler and ISO manages to get higher improvements than when the labels are not provided. While working with labeled trajectories, ISO is also completely insensitive to the optimality of the user behavior. 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.

Improving interactive systems with ISO. Figure 1 shows how the quality of the interactive system increases with each iteration of ISO. ISO converges quite fast – as we can see in Figure 1, after 50 iterations the expected state value begins to plateau. Most improvements happen in the first several iterations. Also, ISO improves consistently – each iteration is an improvement over the previous one. User trajectories range from optimal (Opt) to suboptimal (SubOpt) to adversarial (Adv) – as long as there is user feedback, ISO is able to improve the initial expected state value. As expected, with respect to adversarial user trajectories without feedback, ISO fails to optimize the interactive system and the expected state value decreases. Thus, ISO works with accurately labelled trajectories, but usually obtaining high-quality labels is intractable and expensive in a real interactive system because the real rewards are invisible.

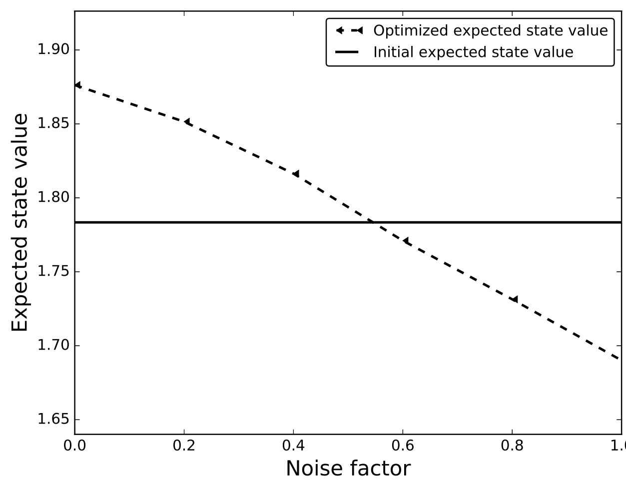

Figure 2. Performance of ISO without feedback under different noise factors after the first iteration.

Suboptimal trajectories in the absence of feedback. In Figure 2 we analyze further what happens after one iteration when the user trajectories are increasingly suboptimal and there is no feedback. The suboptimality of user behavior only matters in the absence of labels, so we only plot the performance of ISO without labels across different levels of noise in Figure 2. Recall that the noise factor is the proportion of trajectories generated by the adversarial policy compared to the optimal user policy. The tipping point for our algorithm is around 0.5 – when more than half of the trajectories come from the adversarial policy ISO starts to deteriorate. However, as long as the noise factor is below 0.5, ISO manages to optimize the interactive system already after the first iteration. With more iterations, the robustness of ISO gets even stronger. Thus, while ISO is able to deal with unlabelled trajectories, we have to assume that the majority of users behaves optimality; its performance degrades when this assumption is violated. With a noise factor of 0.4, ISO manages to get 89% improvement, with 136% being the maximal improvement (Table 1: suboptimal trajectories).

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

(2) Feedback \ (1) Trajectories	(a) Adversarial			(b) Optimal			(c) Suboptimal		
	Initial	Optimized	Impr	Initial	Optimized	Impr	Initial	Optimized	Impr
Explicit feedback	1.78	4.21	136%*	1.78	4.21	136%*	1.78	4.21	136%*
No feedback	1.78	1.66	−6%	1.78	3.95	122%*	1.78	3.36	89%*

Table 1. The performance of ISO, measured as relative improvement (*Impr*) in expected state value over the *Initial* interactive system of the *Optimized* version (after 200 iterations) for different types of user behavior (Section 3.2): (1) trajectory generation principles: (a) *Adversarial*, (b) *Optimal*, (c) *Suboptimal*; (2) with and without user feedback. * indicates statistically significant changes ($p < 0.01$) using a paired t-test.

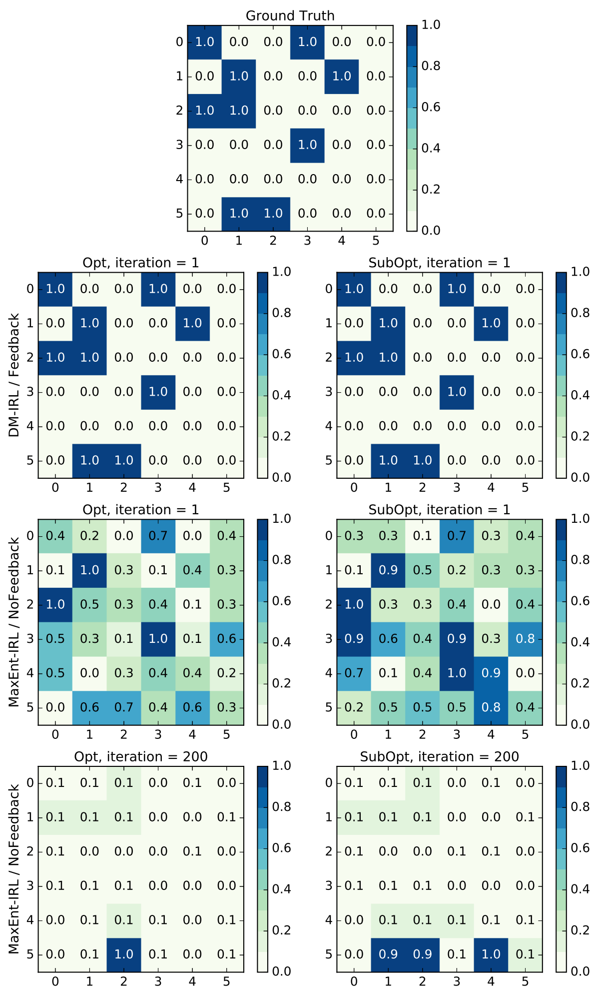

Figure 3. The quality of IRL methods (from top to bottom): *Line 1:* the true $r(s)$; *Line 2:* the recovered $r(s)$ by DM-IRL; the recovered $r(s)$ by MaxEnt-IRL *Line 3:* after the first iteration; and *Line 4:* after the 200-th iteration.

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, $r(s)$, which we analyze for the following types of user behavior: $\hat{H}_{opt}$, $\hat{H}_{sub}$, H_{opt}, H_{sub}.[2] For the case of labeled trajectories, we can see that values of $r(s)$ recovered by DM-IRL are identical to the ground truth in Figure 3 (2nd line).[3] For the case of unlabelled trajectories, the quality of MaxEnt-IRL is worse as shown in Figure 3 (3rd and 4th lines). However, MaxEnt-IRL can still give a general overview of $r(s)$ if user trajectories are optimal as presented in Figure 3 (3rd line, left). With each iteration of running ISO the shape of the sampled trajectories becomes more similar, which means most trajectories pass by the same states and the diversity of trajectories decreases. This makes it even more difficult to recover $r(s)$ so the MaxEnt-IRL quality deteriorates with the number of iterations. Hence, improving the performance of IRL methods is likely to significantly boost the performance of ISO.

In summary, we have presented the experimental results and analyzed the performance of ISO with and without labels and across different type of user trajectories. We can conclude that ISO works well in the presence of user feedback. In case of unlabelled trajectories, the performance of ISO depends on the optimality of user interactions.

8. 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 et al., 2012). Many applications of RL to optimizing interactive systems come from Information Retrieval (IR), recommender systems, and dialogue systems. Hofmann et al. (2011; 2013b) apply RL to optimize IR systems; they use RL for online learning to rank and use interleaving to infer user preferences (Hofmann et al., 2013a). Later work on RL in IR predefines reward functions as the number

[2]Due to space limitations we omit the adversarial trajectories as the least interesting case.

[3]We sampled 100 different reward functions to run ISO, but we report the quality of one $r(s)$ due to space limitation.

of satisfied clicks in session search (Luo et al., 2015a;b). Shani et al. (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 the hand crafted 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), clicks (Kutlu et al., 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 IRL. Ziebart et al. (2012) use IRL for predicting the desired target of a partial pointing motion in graphical user interfaces. Monfort et al. (2015) use IRL to predict human motion when interacting with environment. IRL has also been applied to dialogues to extract the reward function and model the user (Pietquin, 2013). While typically,IRL is used to model user behavior in order to make predictions about it. 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 in spirit 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.

Thus, the key distinctions between our work and previous studies are 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 to improve the user experience.

9. Conclusions and Future work

In this paper, 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 on 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 investigated the following main research question: *Can we optimize an interactive system for users through data-driven objectives?* As an answer 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. Users display one of three behaviors: random, suboptimal, optimal. Each of these behaviors reflects different levels of familiarity with the interactive system; i.e., an unexperienced user will display random behavior, whereas an experienced user will maximize their experience by displaying optimal behavior. User satisfaction is modelled by rewards received from certain interactions, and the user interaction history is represented by a set of trajectories. Thus if a user is not displaying random behavior, their trajectories will be somewhat indicative of their preferences. Optionally users can also give explicit feedback on the quality of the interactive system by labelling their trajectories.

Secondly, we infer the user needs from the observed interactions, in the form of a data-driven objective. Since the user goal is to find the optimal policy that maximizes his gain from the system, their interactions will indirectly indicate their satisfaction. Making use of this property, we use Inverse Reinforcement Learning (IRL) to recover the user reward function from the observed user behavior. We experiment with two IRL methods, one that works with explicit feedback, the other without. Importantly, these methods work without any domain knowledge, and are thus even applicable when prior knowledge is absent.

Thirdly, ISO optimizes the interactive system to match the inferred objective. The interactive system chooses how to respond to the user actions, and from the user perspective these responses are state transitions. However, the interactive system is in control of the transitions, thus these are the actions it chooses from. We optimize the system behavior by using a transformed MDP that represents the system perspective. Using the recovered reward signal the system changes its behavior, and thus how it responds to the user interactions. In response the user is expected to change his behavior as well, to adopt the new system policy. 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. In the end, both the behavior of the system and the user have been optimized according to the user satisfaction. Our experimental results show that ISO robustly improves the user experience across different types of user behavior.

In conclusion, we have proposed a new approach to infer objectives from user interactions that is using IRL methods. Furthermore, we have invented the principled algorithm ISO that simultaneously infers objectives from interactions, while optimizing a system for these inferred user preferences. 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 singular goal, this approach could be extended for 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, Pieter and Ng, Andrew Y. Apprenticeship learning via inverse reinforcement learning. In *ICML*, pp. 1–8. ACM, 2004.

Bagnell, JA, Chestnutt, Joel, Bradley, David M, and Ratliff, Nathan D. Boosting structured prediction for imitation learning. In *NIPS*, pp. 1153–1160, 2007.

Burchfiel, Benjamin, Tomasi, Carlo, and Parr, Ronald. Distance minimization for reward learning from scored trajectories. In *AAAI*, pp. 3330–3336. AAAI Press, 2016.

Dehghani, Mostafa, Zamani, Hamed, Severyn, Aliaksei, Kamps, Jaap, and Croft, W Bruce. Neural ranking models with weak supervision. In *SIGIR*, pp. 65–74. ACM, 2017.

Duan, Yan, Schulman, John, Chen, Xi, Bartlett, Peter L, Sutskever, Ilya, and Abbeel, Pieter. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

El Asri, Layla, Laroche, Romain, and Pietquin, Olivier. Reward shaping for statistical optimisation of dialogue management. In *SLSP*, pp. 93–101. Springer, 2013.

Finn, Chelsea, Levine, Sergey, and Abbeel, Pieter. Guided cost learning: Deep inverse optimal control via policy optimization. In *ICML*, pp. 49–58. ACM, 2016.

Hofmann, Katja, Whiteson, Shimon, and de Rijke, Maarten. Balancing exploration and exploitation in learning to rank online. In *ECIR*, pp. 251–263. Springer, 2011.

Hofmann, Katja, Schuth, Anne, Whiteson, Shimon, and de Rijke, Maarten. Reusing historical interaction data for faster online learning to rank for IR. In *WSDM*, pp. 183–192. ACM, 2013a.

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

Kiseleva, Julia, Williams, Kyle, Awadallah, Ahmed Hassan, Zitouni, Imed, Crook, Aidan, and Anastasakos, Tasos. Predicting user satisfaction with intelligent assistants. In *SIGIR*, pp. 45–54. ACM, 2016.

Kosinski, Michal, Stillwell, David, and Graepe, Thore. Private traits and attributes are predictable from digital records of human behavior. *PNAS*, 110:5802–5805, 2013.

Kutlu, Mucahid, Khetan, Vivek, and Lease, Matthew. Correlation and prediction of evaluation metrics in information retrieval. *arXiv preprint arXiv:1802.00323*, 2018.

Levine, Sergey, Popovic, Zoran, and Koltun, Vladlen. Feature construction for inverse reinforcement learning. In *NIPS*, pp. 1342–1350, 2010.

Levine, Sergey, Popovic, Zoran, and Koltun, Vladlen. Nonlinear inverse reinforcement learning with gaussian processes. In *NIPS*, pp. 19–27, 2011.

Levine, Sergey, Finn, Chelsea, Darrell, Trevor, and Abbeel, Pieter. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39): 1–40, 2016a.

Levine, Sergey, Pastor, Peter, Krizhevsky, Alex, Ibarz, Julian, and Quillen, Deirdre. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, pp. 173–184, 2016b.

Li, Jiwei, Monroe, Will, Ritter, Alan, Jurafsky, Dan, Galley, Michel, and Gao, Jianfeng. Deep reinforcement learning for dialogue generation. In *EMNLP*, pp. 1192–1202, 2016.

Li, Ziming, Kiseleva, Julia, de Rijke, Maarten, and Grotov, Artem. Towards learning reward functions from user interactions. In *ICTIR*, pp. 941–944. ACM, 2017.

Lopes, Manuel, Melo, Francisco, and Montesano, Luis. Active learning for reward estimation in inverse reinforcement learning. In *ECML/PKDD*, pp. 31–46. Springer, 2009.

Lowe, Ryan, Noseworthy, Michael, Serban, Iulian V, Angelard-Gontier, Nicolas, Bengio, Yoshua, and Pineau, Joelle. Towards an automatic turing test: Learning to evaluate dialogue responses. In *ACL*, pp. 1116–1126, 2017.

Luo, Jiyun, Dong, Xuchu, and Yang, Hui. Learning to reinforce search effectiveness. In *ICTIR*, pp. 271–280. ACM, 2015a.

Luo, Jiyun, Dong, Xuchu, and Yang, Hui. Session search by direct policy learning. In *ICTIR*, pp. 261–270. ACM, 2015b.

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

Mohri, Mehryar, Rostamizadeh, Afshin, and Talwalkar, Ameet. *Foundations of Machine Learning*. MIT Press, 2012.

Monfort, Mathew, Liu, Anqi, and Ziebart, Brian. Intent prediction and trajectory forecasting via predictive inverse linear-quadratic regulation. In *AAAI*, pp. 3672–3678. AAAI Press, 2015.

Ng, Andrew Y, Russell, Stuart J, et al. Algorithms for inverse reinforcement learning. In *ICML*, pp. 663–670. ACM, 2000.

Odijk, Daan, Meij, Edgar, Sijaranamual, Isaac, and de Rijke, Maarten. Dynamic query modeling for related content finding. In *SIGIR*, pp. 33–42. ACM, 2015.

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

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

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

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

Silver, David, Huang, Aja, Maddison, Chris J, Guez, Arthur, Sifre, Laurent, Van Den Driessche, George, Schrittwieser, Julian, Antonoglou, Ioannis, Panneershelvam, Veda, Lanctot, Marc, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

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

Wang, Jane X, Kurth-Nelson, Zeb, Tirumala, Dhruva, Soyer, Hubert, Leibo, Joel Z, Munos, Remi, Blundell, Charles, Kumaran, Dharshan, and Botvinick, Matt. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Wei, Honghao, Zhang, Fuzheng, Yuan, Nicholas Jing, Cao, Chuan, Fu, Hao, Xie, Xing, Rui, Yong, and Ma, Wei-Ying. Beyond the words: Predicting user personality from heterogeneous information. In *WSDM*, pp. 305–314. ACM, 2017.

Williams, Kyle, Kiseleva, Julia, Crook, Aidan C., Zitouni, Imed, Awadallah, Ahmed Hassan, and Khabsa, Madian. Detecting good abandonment in mobile search. In *WWW*, pp. 495–505, 2016.

Wulfmeier, Markus, Ondruska, Peter, and Posner, Ingmar. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*, 2015.

Zhu, Yuke, Mottaghi, Roozbeh, Kolve, Eric, Lim, Joseph J, Gupta, Abhinav, Fei-Fei, Li, and Farhadi, Ali. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, pp. 3357–3364. IEEE, 2017.

Ziebart, Brian, Dey, Anind, and Bagnell, J Andrew. Probabilistic pointing target prediction via inverse optimal control. In *IUI*, pp. 1–10. ACM, 2012.

Ziebart, Brian D. *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy*. PhD thesis, Carnegie Mellon University, 2010.

Ziebart, Brian D, Maas, Andrew L, Bagnell, J Andrew, and Dey, Anind K. Maximum entropy inverse reinforcement learning. In *AAAI*, pp. 1433–1438. AAAI Press, 2008.