Technical Report: *University of Edinburgh, 2013.*

Latent-Variable MDP Models for Adapting the Interaction Environment of Diverse Users

Subramanian Ramamoorthy
M. M. Hassan Mahmud
Benjamin Rosman
School Of Informatics
University of Edinburgh

Pushmeet Kohli
Machine Learning and Perception
Microsoft Research Cambridge

Abstract

Interactive interfaces are a common feature of many systems ranging from field robotics to video games. In many applications, these interfaces are targeted at a diverse and heterogeneous set of potential users and correspondingly there is substantial variety in what types of interfaces can be made available (e.g., KEYBOARD vs. joysticks) and how they may be configured (e.g., sensitivity levels). In this paper we are interested in solving the problem of personalizing an interface such that it adapts to present the user with a variation that is optimal with respect to their traits, such as skill level. We pose this problem by modelling the user as a parametrized Markov Decision Process (MDP), wherein the transition dynamics within a task depend on the latent skill level of the user. This notion of adapting at the level of action sets, picking optimally from a potentially very large space of action sets, is novel and provides a natural solution to the interface personalization problem. Our solution to this problem involves a latent variable formulation wherein there are hidden personality traits, such as a user skill-level, which are implicitly associated with a specific (optimal) choice of action set. We present an algorithm that iteratively eliminates potential action sets to quickly arrive at the action set optimally suited to a particular user. We evaluate this procedure in an experiment involving a simulated remote navigation domain, demonstrating that the combination of latent variable user models and type elimination outperforms baselines that do not model the diversity of user actions in this way.

1 Introduction

Interactive interfaces are a common feature of many systems ranging from field robotics to video games. Consider the user of a joystick based interface who wishes to perform a task such as navigating a robot in a hostile and dynamic environment. Users have varying degrees of skill and comfort with different types and settings of interfaces. With the emergence of gesture-based natural user interfaces, the range of choices is further expanded, involving numerous personality traits and idiosyncratic attributes that determine how nimble users might be in performing parts of tasks.

To make things concrete, consider the following example. A user is presented with two types of interfaces - keyboard or joystick - to perform the task of remote navigation of a robot in a cluttered environment. Each interface may be configured according to parameters such as sensitivity, e.g., is every twitch mirrored on the robot's path or is the interface somewhat more 'filtered'. We envision scenarios where the combinatorial possibilities of interface settings is very large, in the hundreds or thousands. In order to facilitate optimal performance by the user, we [1] would like to be able to adapt the interface options we present to the user. In particular, this adaptation must be quick in two ways. On the one hand, it is infeasible to expect the user to explore all possibilities. Equally, the interface should be adapted within a few episodes, within the level of patience of the average human user.

The objective of this paper is to present a novel model and algorithm to solve the problem of interface personalization/adaptation. We present a online learning algorithm that adapts the interface actions exposed to the user by observing the past interactions made by the user.

1.1 Related work

The notion of learning to adapt to user behaviour is garnering increasing attention from researchers in different domains. A common theme is that the user is in some sense not fully understood and the objective of the learning agent is to learn to fill in the gaps.

In one formulation, the learning agent may be assisting a user whose goal is unknown. For instance, [Fern and Tadepalli, 2010] present the hidden-goal MDP to model problems such as that faced by a doorman who must anticipate when and where to go in order to help someone. In general, this is a computationally hard problem and fairly strong assumptions are required to achieve tractable results, such as that the only unknown is the goal state of the user. Indeed, this version of the problem has a longer history in the literature on plan recognition. The notion of hidden goal is but one

[1] This paper is written from the perspective of a learning agent who is adapting the interface to aid a human user. So, when we use an unqualified 'we' in the remainder of the paper, it refers to this 'learning agent' that we sometimes abbreviate to 'learner'.

instance of the way unknowns are treated in a larger literature on partially-observable and mixed-observable Markov Decision Processes [Satinder Singh *et al.*, 2002] [Ong *et al.*, 2010].

One relevant example is the way such models are applied in dialogue systems [Satinder Singh *et al.*, 2002]. In a typical dialogue application, an interface makes an utterance, e.g., a greeting, to which the user responds, the system interprets the user response and continues to the next utterance and so on. In all of this, the user state is only noisily observed and indeed many essential variables such as what the user really wants done are hidden or partially observable. Similarly, in robotics, there is increasing interest in the notion of intention-aware motion planning [Bandyopadhyay *et al.*, 2012] [Sisbot *et al.*, 2007], to accommodate the motion of, e.g., pedestrians who may wander in erratic ways.

We note that our problem is somewhat different in that we are not so much trying to predict what the user might do in the future as we are trying to infer latent 'personality traits' of the user in order to shape their environment to enable them to do better in their chosen task. In a sense, our work bears a similarity to work on personalization such as in recommender systems and other web applications, where it is common to model user behaviour as a Markov Chain - jumping from item to item in a fairly stable transition pattern - and adapt recommendations with respect to its anticipated evolution [Jannach *et al.*, 2011]. In our applications, we need more expressive models to describe the user. Some recent work in the recommendation domain, such as [Chickering and Paek, 2007], [Shani *et al.*, 2005], is based on adapting parameters of an MDP or influence diagram - learning uncertain parameters in the transition model through a bandit-style procedure.

1.2 Overview

In this paper, we propose a novel model that captures aspects of our motivating problems that are sometimes omitted in related work such as mentioned above. Firstly, we seek to shape the interaction environment of the user - specifically, we want to modify the 'action sets' they make decisions using - as opposed to predicting and intervening on more detailed choices they may make with respect to trajectories. In practical terms, we want to help the user pick the best interface settings to play a game, enabling them to achieve their maximal performance. In our remote navigation example, this requires us to identify the latent skill level of the user (e.g., are they able to use a highly sensitive interface well, or should their responses be filtered substantially). We do this by observing the realized performance on the task, using an initial action set, based on which we improve our estimate of the user's skill level and continue to adapt the interface. This idea of adapting the *action set* in an MDP-based user model is a novel contribution of this paper.

Secondly, we seek to address the diversity of the user base and the need for the interface to perform well for any user drawn from such a diverse population - a robustness requirement. We do this by formulating a latent-variable model based on hidden personality traits such as the skill level. The latent variables imply a restriction on the types of transitions that need to be described in our user model. Implicitly, our model also captures the intuition that there are *behavioural types* of users - such as in the way members of particular demographic categories behave similarly when given different types of interfaces. This is translated into a very specific way in which we adapt action sets - we begin with a coarse choice that is tuned to a population of users and a more fine adaptive choice that is subsequently tuned to the individual.

Thirdly, we aim at quick adaptation, which is partly enabled by our formulation of the problem as one of eliminating behavioural types within a latent variable model. For instance, a child walking up to an XBox game does not want to spend several minutes executing complex calibration routines. They want to get started now, perhaps accommodating a few episodes of setup while the system improves its performance from a good baseline to a much better personalized optimum.

Our proposed procedure encapsulates two different types of choice - an offline computation of the robust solution to a stochastic optimization problem involving a parameterized MDP user model and a distribution over behavioural types in a population, and an online adaptive procedure that further adapts to a specific individual. This adaptation is based on the use of a novel hypothesis test using the realized performance of the user with action sets used so far.

Our domain is an instantiation of the motivating example in the first paragraph - we model the remote navigation task using different settings of the interface, with varying sensitivity and error rate for the user. We first synthesize a corpus of diverse user models who differ in skill level. Then, we demonstrate that each population of such users is well served a robust population-optimal choice of action set. Then, we show that our learning algorithm achieves further improvement, quickly. We present comparisons against baselines such as a state of the art online learning algorithm, EXP-3, which is much more erratic and slower to converge in contrast to our algorithm.

2 Preliminaries

We use $\triangleq$ for definitions, Pr to denote probability and $\mathbb{E}$ for expectation. Given a sequence of observations of random variable X, the empirical estimate of $\mathbb{E}[X]$ is the average of all the observed values. Similarly, for a distribution D, its empirical estimate is the empirical probability constructed from $x_{1:t}$ observations drawn from D. We denote this distribution by $P_{x_{1:t}}$. The rest of this section is devoted to defining Markov decision processes and the value functions of their policies.

For an introduction to reinforcement learning using MDPs, see [Puterman, 1994] [Sutton and Barto, 1998]. A finite MDP $\mathcal{M}$ is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, T, R, \gamma)$ where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions and $\mathcal{R} = [l, u] \subset \mathbb{R}$ is the set of rewards. $T(s'|s, a)$ is a the state transition distribution for $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$ while and $R(s, a)$, the reward function, is a random variable taking values in $\mathcal{R}$. Finally, $\gamma \in [0, 1)$ is the discount rate.

A (stationary) policy π for $\mathcal{M}$ is a map $\pi : \mathcal{S} \to \mathcal{A}$. The Q

function $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ of a policy π is defined by:

$$Q^\pi(s,a) = \mathbb{E}[R(s,a)] + \gamma \sum_{s'} T(s'|s,a) Q^\pi(s', \pi(s'))$$

The value function for π is defined as $V^\pi(s) = Q^\pi(s, \pi(s))$. An optimal policy π^* is defined as $\pi^* = \arg\max_\pi V^\pi$ – the Q function is given by

$$Q^*(s,a) = \mathbb{E}[R(s,a)] + \gamma \sum_{s'} T(s'|s,a) \max_a Q^*(s', a)$$

For the optimal policy the value functions is denoted by V^*. The goal of the agent is to estimate Q^* and then choose the action $\arg\max_a Q^*(s,a)$ at state s. In the sequel, we assume, without loss of generality, a fixed starting state s° for a MDP and define $V^\pi \triangleq V^\pi(s^\circ)$.

3 A Model of User Interaction

In this section we present our model of the user interaction problem described in the introduction. We will model the user operating the interface as an agent choosing actions in an MDP using an optimal policy. The user skill level/type will determine the transition function of the MDP – this models the fact that, for instance, a less skilled user will have a limited number of capabilities. Given an MDP, the user adapts to play using the corresponding optimal policy. In addition, unlike in standard applications of MDPs, we will allow the action set to be changed in the middle of an episode, which will correspond to the interface adapting its sensitivity to the user skill. The goal of the learner will be to choose action sets so as to maximize the future expected discounted reward of the agent. Note that, this goal is also quite different from standard applications of MDPs, where the goal is to find the optimal action. Here, our search space is a type-space, rather than the policy space, and the value of an type is the value of its optimal policy given the type.

More precisely, we are concerned with a class of MDPs M parametrized by a parameter τ:

$$\mathsf{M} \triangleq \{((\mathcal{S}, \mathcal{A}, \mathcal{R}, T(\cdot;\tau), R, \gamma) | \tau \in \mathsf{Sk}\} \quad (1)$$

The MDPs differ only in the transition function $T(s'|s,a;\tau)$, which now depends on τ. The parameter τ represents the type (more specifically, skill) of the agent, where Sk is the set of all types. For instance, in the example mentioned in Section 1, the user controlling a joystick corresponds to a MDP. Furthermore, a poorly skilled user will have poor control over the joystick which means that he will only be able to make coarse moves (regardless of the sensitivity of the joystick). Whereas, a highly skilled user will be able to make much more precise moves if given an appropriately sensitive joystick. This implies that the MDP corresponding to the task for each user type has different transition functions (for example, transitions with high and low entropy for low and high skilled users respectively).

We further assume that the action space $\mathcal{A}$ is large, but is partitioned, where AP is the set of all the cells of the partition. That is, if $\alpha, \alpha' \in \mathsf{AP}$, then α and α' are disjoint; and $\bigcup_{\alpha \in \mathsf{AP}} \alpha = \mathcal{A}$. We now only allow policies that take values in a single $\alpha \in \mathsf{AP}$ (we say the policy is restricted to α). Continuing with the joystick example, different action sets correspond to changing the sensitivity of the joystick. With a higher sensitivity joystick, the user now has the option (i.e. actions) to execute finer grained moves. Of course, whether she will be able to execute them depends on her skill level (i.e. the transition function ultimately also depends on the user type).

The value function of a policy π now depends on τ and is denoted by V^π_τ. For a given α, we denote:

$$V^\alpha_\tau \triangleq \max_\pi V^\pi_\tau \quad (2)$$

and denote by $\pi^*_\alpha \triangleq \arg\max_\pi V^\pi_\tau$, where the max is over πs restricted to α, .

Our learning problem can now be defined as follows. The agent solves a sequence of MDPs from the set M in collaboration with the learner – that is the goal of both the agent and the learner is to maximize the future expected discounted reward of the agent. At the beginning of each phase of the problem, nature chooses the type τ according to a distribution $K_1(\tau)$ and in response, the learner is required to choose the action set α. The agent can now only use policies restricted to α. The learner cannot observe the true τ but knows the value of T and R for each possible value of τ. The agent knows its own type and also knows the value of T and R. Once α is given, the agent learns and acts according to the policy π^*_α.

The role of the learner is as follows. At any point in time, the learner can step in and change the action set from α to a new action set α'. In response the agent starts acting according to $\pi^*_{\alpha'}$ instead. The goal of the learner is to choose the action set $\arg\max_\alpha V^\alpha_\tau$. In terms of joystick example, this means that at the beginning some user of unknown skill (τ) comes in to use the interface. She operates the interface to the best of her ability given the sensitivity. Given her actions, the learner tries to estimate the skill level and based on that either increases or decreases the sensitivity.

Given the above setup, the a-priori optimal action-set is defined to be

$$\alpha_* \triangleq \arg\max_\alpha \mathbb{E}_{K_1(\tau)}[V^\alpha_\tau] \quad (3)$$

In the next section, our goal will be to derive an *a-posteriori optimal* action-set selection policy, that is, a policy that that adaptively and optimally chooses the action-set at every step and performs better than α_*. The exact but computationally intractable a-posteriori optimal policy is defined in (5) and its approximation is defined in (6).

4 Determining Agent Type

In the following we present our approach to determining the agent type through interaction. In section 4.1, we show, in an idealized setting, how to optimally choose the action set given that we know that the type lies in a particular set. Then, in 4.2 we show that this problem is in fact intractable in general and present a heuristic solution to the problem. Finally, in Section 4.3 we present our algorithm that we use in the experiments. The algorithm uses a combination of the heuristic and statistical tests to adaptively choose action-sets.

4.1 The Idealized Learning Problem

In this section we discuss an idealized protocol for the learner interacting with nature for the problem of action-set identification that we discussed in the previous setting. This protocol allows us to develop an idealized solution in the next subsection which then helps us to construct a algorithm for action-set selection.

We assume that nature chooses the type τ but the learner is not told about it. In response the learner chooses an initial action set α and the agent learns and acts according to π^*_α. These actions give the learner information about the type of the agent and is used by it to choose the next action set. We assume that the learner knows the set of reward distribution R and transition distributions T but not the τ distribution K_1. Hence, the goal of the learner is to use the observed actions of the agent to estimate the transition function for the latent variable τ and compare that against its knowledge of T to determine the type of the agent so that it can choose the best possible action-set for the type.

The learning protocol for solving this problem is as follows. At each step t the learner has a set B_t that is its current guess of the set where the true type lies in, with $B_0 = \{\mathsf{Sk}\}$. It proposes an action set α_t and in response obtains the reward $V^{\alpha_t}_\tau$ due to the user operating in the MDP. In return nature gives the set $\mathsf{N}(\alpha_t, B_t, \tau)$ of types that are eliminated by the action set. Here, given any set $B \subset \mathsf{Sk}$, we denote

$$\mathsf{N}(\alpha, B, \tau) \triangleq \{\tau' | \tau' \in B, \\ \forall s, a, T(s'|s, a; \tau') = T(s'|s, a; \tau)\}$$

That is, $\mathsf{N}(\alpha, B, \tau)$ is the set of types in B that have the same transition distribution as τ when action set α is used. In other words, we are not able to distinguish τ from other types if the agent uses action-set α [2].

At the next step $t+1$, $\mathsf{N}(\alpha_t, B_t, \tau)$ is used as B_{t+1} and the above procedure repeats. The protocol is formally given as Protocol 1. Our goal is now to develop an action-set selection policy Π that maximizes the reward obtained by the learner in this protocol. We do so in the next subsection.

Protocol 1 Type Detection Game

1: **Initialize:** Set $B_0 = \mathsf{Sk}, t = 0$
2: **while** $true$ **do**
3: Learner chooses action set α_t
4: Learner obtains reward $v_t = V^{\alpha_t}_\tau$.
5: Nature returns set $B_{t+1} \triangleq N(\alpha_t, B_t, \tau)$.
6: $t \rightarrow t + 1$.
7: **end while**

4.2 A Latent Variable MDP Based Solution

Given the above, we can restate the action-set selection problem itself as a latent-variable meta-MDP. For this MDP, the meta-action (m-action) space is AP and the meta-state (m-state) space is 2^{Sk}. The transition and reward are both determined by the latent variable τ which is selected at the beginning of each episode according to $K_1(\tau)$. Starting from the initial state $B_0 \triangleq \mathsf{Sk}$ when the m-action α_t is taken at m-state B_{t-1}, the m-MDP transitions to state $\mathsf{N}(\alpha_t, B_{t-1}, \tau)$ deterministically and obtained reward $v(\tau, \alpha_t) = V^{\alpha_t}_\tau$. An action-set selection m-policy Π is a mapping $2^{\mathsf{Sk}} \rightarrow \mathsf{AP}$ with value

$$V^\Pi = \mathbb{E}_{K_1}[\sum_{t=0}^{\infty} \eta^t v(\tau, \alpha_t)] \quad (4)$$

where $\alpha_t = \Pi(B_t)$, $B_t = \mathsf{N}(\alpha_{t-1}, B_{t-1}, \tau)$, and $B_0 = \mathsf{Sk}$. Here, $\eta \in [0, 1)$ is a discount factor. The optimal action set selection policy is

$$\Pi^* = \arg\max_\Pi V^\Pi \quad (5)$$

Unfortunately, computing Π^* is in general intractable. In particular, a latent variable MDP is a type of POMDP [Choi *et al.*, 1999] and so what we have here is a POMDP model learning problem with state/observation space that is exponentially large in the parameter of interest (due to m-state space being 2^{Sk}). The POMDP model learning problem is very hard. State of the art algorithms do not scale beyond small maze problems with 30-40 states (see for instance [Doshi-Velez, 2009]), whereas we have a space exponentially large in the parameter of interest. Indeed, this exact version of the problem has been shown to be intractable [Sabbadin *et al.*, 2007].

Given the above, instead of Π^*, we use a heuristic policy to choose our action-set given that we know the type to belong to a particular set B. This policy is defined to be:

$$\hat{\Pi}(B) \triangleq \arg\max_\alpha \mathsf{val}(\alpha, B) \quad (6)$$

where the function val is defined as follows:

$$\mathsf{val}(\alpha, B) \triangleq \begin{cases} 1)\ \min_{\tau \in B} V^\alpha_\tau & \text{if } \mathsf{N}(\alpha, B, \tau) = B \\ 2)\ \min_\tau [V^\alpha_\tau + & \frac{\max_{\alpha'} \sum_{\tau' \in \mathsf{N}(\alpha, B, \tau)} V^{\alpha'}_\tau}{\mathsf{N}(\alpha, B, \tau)} \\ 3)\ -\infty & \text{if } \mathsf{N}(\alpha, B, \tau) = \emptyset \end{cases}$$

The interpretation of val is as follows. Cases 1 and 2 combined give an inductive definition for the situation where $\tau \in B$. The base case (case 1) says that value of an action set α when the action-set B cannot be reduced further is the worst case value of α given $\tau \in B$. A special instance of this is when $B = \{\tau\}$ and so $\min_{\tau \in B} V^\alpha_\tau = V^\alpha_\tau$.

The inductive step in case 2 defines val when B can be reduced in a mini-max fashion. Here, the first term is the *exploration cost* which gives the worst-case reward of the agent acting with the action set α. This is the reward obtained when the agent acts and the learner collects information to eliminate types from τ. The second term gives the optimistic, *exploitation reward* and measures the potential reward obtained by eliminating types using α. This is simply the average value of the best action-set on the remaining set of types.

Finally, case 3 addresses the pathological case where $\tau \notin B$ and so the value is $-\infty$.

[2] Note that, in practice, this step of determining which types are eliminated is a noisy process because we need to estimate the transition function of the current MDP from observations because the type τ, which determines T, is unknown to us. See next section.

4.3 Type Detection Algorithm

In this section we present our algorithm for detecting the type of the agent while interacting with it. This is given as algorithm 2 and this is a practical approximation of Protocol 1. The approximation comes from two places. First, the learner only gets a sampled version of v_t at each step rather than the actual v_t. And additionally, the set B_t constructed at each step is noisy, in that we are only sure upto a certain probability β that B_t contains the true type τ.

The basic setup is that the agent type τ determines the transition distribution $T(\cdot|s,a;\tau)$. The algorithm then observes the agent act and collects the observed next-state samples at each state-action pair. The algorithm also maintains a collection B_t of candidate types the true type might belong to, and applies a statistical test f on the collected samples to reject elements of B_t as the true type. For a type $\tau \in B_t$, the test f computes the probability that τ *would not* have generated the observed sample. If this probability is above a user defined parameter $\beta \in [0,1)$, then the type τ is rejected as the true type with probability/confidence level β. We now present a novel Hoeffding bound based test and then we present our algorithm for eliminating types. The advantage of our new test over existing tests is that we get precise sample complexity bounds.

The Hoeffding Bound Based Test

In the sequel, $s_{1:k}$ will be a the set of next states observed at a particular state-action pair s,a that has been visited k times. The type we want to reject at confidene level β to be τ. The Hoeffding test statistic is given by:

$$f_H[s_{1:k}, T(\cdot|s,a;\tau)] = 1 - \exp[2K - k(\|Pr_{s_{1:k}} - T(\cdot|s,a;\tau)\|_1)^2] \quad (7)$$

with $K = \ln(2^{|\mathcal{S}|}-2)$ and $Pr_{s_{1:k}}$ is the empirical distribution of $s_{1:k}$ (see Section 2). f_H is a valid test in the following sense (proof in the appendix):

Lemma 1. *$f_H[s_{1:k}, T(\cdot|s,a;\tau)]$ is the probability that after k steps τ* will not *have generated a sample $\bar{s}_{1:k}$ such that*

$$\|Pr_{\bar{s}_{1:k}} - T(\cdot|s,a;\tau)\|_1 > \|Pr_{s_{1:k}} - T(\cdot|s,a;\tau)\|_1 \quad (8)$$

So if $f_H[s_{1:k}, T(\cdot|s,a;\tau)] > \beta$, with probability β, τ would have generated a sample less than $\|Pr_{s_{1:k}} - T(\cdot|s,a;\tau)\|_1$ and we can reject τ with probability β. When we have n different types τ_i to test, by the union bound, we need that:

$$\sum_i (1 - f_H[s_{1:k}, T(\cdot|s,a;\tau_i)]) \leq (1-\beta) \Rightarrow \sum_i f_H[s_{1:k}, T(\cdot|s,a;\tau_i)] \geq n - 1 + \beta \quad (9)$$

for the guarantee above to hold for all the n types simultaneously.

The Algorithm

Our learning algorithm Online-Interact is given as Algorithm 2. The algorithm runs until a termination condition $term$ is satisfied. Within the loop, the learner deals with that particular task and tries to determine the type of the user. Specifically, at each time step, it allows the user to take an action and updates its estimate of the empirical distribution at that pair (lines 5–6). The algorithm then tries to use one of the two statistical tests defined in the previous section to (lines 7–8) eliminate as many types as possible from the current set B_t of plausible types (with $B_0 = \{\mathsf{Sk}\}$). Then it chooses the next action set according to $\hat{\Pi}$ (defined in (6)) on the remaining set.

Algorithm 2 Online-Interact($\mathsf{M}, f, \beta, term, \hat{\Pi}$)

1: **Input:** M the set of MDPs, test f and constant β for f, task termination condition $term$, and heuristic action-set selection policy $\hat{\Pi}$.
2: **Initialize:** $B_0 = \mathsf{Sk}$; $\hat{T}(s'|s,a)$ to be uniform over $\mathcal{S}$ for each $a \in \mathcal{A}$ and $s \in \mathcal{S}$; $\hat{R}(|s,a)$ to be 0 for each s,a.
3: Let s_0 be the initial state and $t \leftarrow 1$.
4: **while** $term = false$ **do**
5: Observe agent action a_t, reward r_t and next state s_t.
6: Let $s_{1:k}$ and $r_{1:k}$ be the set of next states and rewards observed after taking a at state s where $s = s_{t-1}, a = a_t$ (inclusive of the current step).
7: Compute

$$B'' \triangleq \arg\max\{|B'| : B' \subset B_t, \sum_{\tau \in B'} f(s_{1:k}, T(\cdot|s,a;\tau)) > |B'| - 1 + \beta\}$$

8: Set $B_{t+1} \leftarrow B_t - B''$;
9: **if** $B_{t+1} = \emptyset$ **then** set $B_{t+1} \leftarrow \mathsf{Sk}$ [**In case the type changes or our test fails**].
10: Set $\alpha_{t+1} \leftarrow \hat{\Pi}(B_{t+1})$.
11: $t \leftarrow t+1$.
12: **end while**

5 Experiments

We consider a user interaction scenario where the user controls a point mass around a 20×20 cell 2D world to take it from fixed start to a fixed goal location. The user is scored based on the time taken, and receives a reward of -1 for every step taken, and 100 for reaching the goal. Every user has a skill level $\tau \in [0,1]$ determining how adept they are at using the system. The action sets presented to the user are different controllers, which vary in sensitivity $\alpha \in (0,1]$. Hence each user has a skill level that is optimal for it. At each time step, the user selects a cardinal direction and then proceeds in the chosen direction with probability $p \propto \frac{\tau}{1+|\tau-\alpha|}$ normalised such that $p \in [0.25, 1]$. With probability $1-p$ the movement is in one of the remaining three cardinal directions uniformly at random. The intuition is that the more skilled a user, the more likely the user is to succeed in moving in the intended direction. Furthermore, each input device may have been designed to be optimal for a particular skill level, and so deviation from the intended user on a device impedes performance.

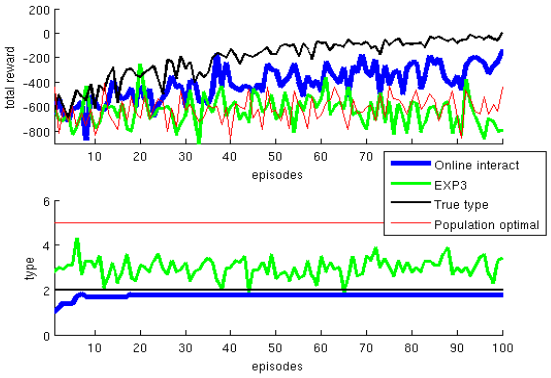

Figure 1: Results for true type 2.

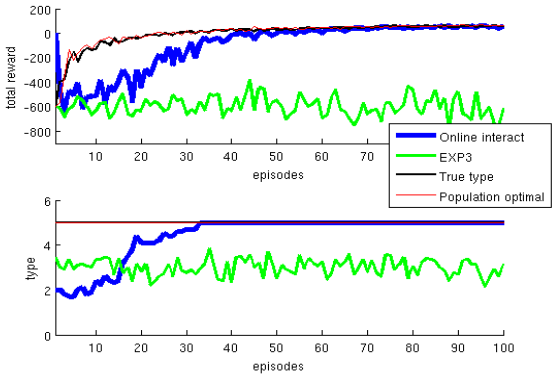

Figure 2: Results for true type 5.

In our experiments, we consider a discrete case where both τ and α can take values from $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. The effect of this is that learning for a user of the lowest skill level is difficult, regardless of the controller. We assume K_1 is uniform, and during the offline phase, we draw one user from K_1 and allow it to learn on each of the five controllers and estimate the transition functions $T(s'|s, a; \alpha, \tau)$. This forms our prior knowledge. During the online phase a new user with an unknown τ is randomly drawn and we proceed as in Online-Interact. Each user continuously learns to act optimally in the domain using Q-learning [Sutton and Barto, 1998]. However, whenever the action set is changed, the user starts learning again from scratch, as the previous policy would not accurately reflect the new conditions.

All experiments were run for 100 episodes, each until reaching the goal or timing out after 1000 steps. All results show averages of 10 runs. Shown are comparisons between Online-Interact (Algorithm 2), EXP-3 online bandits algorithm [Auer *et al.*, 2002], the population optimal policy given by Equation (5), and the true type. Due to lack of space we only show results for type 2 and 5 (Figure 1 and 2), but describe the results for all 5. Optimal performance is given by the true type. The population optimal was computed at $\alpha = 0.9$ as the action set giving maximal returns for unknown user skill, assuming users are uniformly distributed. The user is at a disadvantage when presented with action sets by EXP-3, as it is slow to converge, and every change of action sets forces the user to relearn from scratch. Online-Interact typically converges within 10 episodes. Our results show that our adaptive method outperforms EXP-3 in all cases. It outperforms the population optimal in most cases, and other than the optimal action set for that particular type, which is to be expected. Unfortunately, it also fails to identify the true type 1 and 4. This implies that our type detection method needs to be further refined.

6 Conclusion

In this paper we introduced a novel framework for handling the problem of identifying user types in adaptive user-interaction problems. In this problem, the goal is to determine the type of the user so that we can present him with an interface that is most suitable for her to accomplish her task. We showed that this problem can be formulated as the problem of *action-set selection* in a latent variable MDP. To the best of our knowledge, this problem formulation has not been used by other works before, and allows us to model more complex user behaviour than in prior work in adaptive user-interaction. We derived an optimal but computationally intractable solution to the problem and an approximation algorithm for the solution. We then showed the efficacy of this model by experiments where we successfully identified user types and presented them with correct action set. The directions for future work are many. Indeed, this paper just introduces a novel framework for this very interesting problem with many practical applications [Rosenfeld *et al.*, 2012]. In the future we hope to expand it in different ways, including more effective algorithms for modelling problems with richer description of user behaviour, handling further issues arising in realistic physical systems (e.g., gesture based natural user interfaces) and validating our theory in systematic human behavioural experiments.

A Proof of Lemma 1

Proof. From [Weissman *et al.*, 2003] (via an application of Hoeffding bound), we know that a sample $\bar{s}_{1:k}$ generated by $T(\cdot|s, a; \tau)$ will satisfy the following:

$$Pr(\|Pr_{\bar{s}_{1:k}} - T(\cdot|s, a; \tau)\|_1 \geq \epsilon') \leq \delta \tag{10}$$

where $\delta = \exp[2K - k\epsilon'^2]$ with $K = \ln(2^{|\mathcal{A}'|} - 2)$. Hence, by setting $\|Pr_{s_{1:k}} - T(\cdot|s, a; \tau)\|_1 \triangleq \epsilon_1$, we get that

$$Pr(\|Pr_{\bar{s}_{1:k}} - T(\cdot|s, a; \tau)\|_1 \geq \epsilon_1) \leq \exp[2K - k\epsilon_1^2]$$

and so

$$Pr(\|Pr_{\bar{s}_{1:k}} - T(\cdot|s, a; \tau)\|_1 \leq \epsilon_1) > 1 - \exp[2K - k\epsilon_1^2] = f_H[s_{1:k}, T(\cdot|s, a; \tau)]$$

Of course, $Pr(\|Pr_{\bar{s}_{1:k}} - T(\cdot|s, a; \tau)\|_1 \leq \epsilon_1)$ is simply the probability in the statement of the lemma, and so this completes the proof. □

Acknowledgements

This work has taken place in the Robust Autonomy and Decisions group within the School of Informatics. Research of the RAD Group is supported by the UK Engineering and Physical Sciences Research Council (grant number EP/H012338/1) and the European Commission (TOMSY Grant Agreement 270436, under FP7-ICT-2009.2.1 Call 6).

References

[Auer *et al.*, 2002] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, 32:48–77, 2002.

[Bandyopadhyay *et al.*, 2012] T. Bandyopadhyay, K. S. Won, D. Hsu E. Frazzoli, W. S. Lee, and D. Rus. Intention-aware motion planning. In *Proceedings Workshop of Algorithmic Foundations, WAFR-2012*, 2012.

[Chickering and Paek, 2007] David Maxwell Chickering and Tim Paek. Personalizing influence diagrams: applying online learning strategies to dialogue management. *User Modeling and User-Adapted Interaction*, 17(1-2):71–91, 2007.

[Choi *et al.*, 1999] Samuel P. M. Choi, Dit=Yan Yeung, and Nevin Lianwen Zhang. Hidden-mode markov decision processes. In *Proceedings of the* 16^{th} *International Joint Conference on Artificial Intelligence*, 1999.

[Doshi-Velez, 2009] F. Doshi-Velez. The infinite partially observable markov decision process. In *Proc. of the* 20^{th} *Neural Information Processing Systems Conference*, 2009.

[Fern and Tadepalli, 2010] Alan Fern and Prasad Tadepalli. A computational decision theory for interactive assistants, advances in neural information processing systems. In *Proceedings of the* 23^{rd} *Conference on Neural Information Processing Systems*, 2010.

[Jannach *et al.*, 2011] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. *Recommender Systems An Introduction*. Cambridge University Press, 2011.

[Ong *et al.*, 2010] Sylvie C. W. Ong, Shao Wei Png, David Hsu, and Wee Sun Lee. Planning under uncertainty for robotic tasks with mixed observability. *I. J. Robotic Res.*, 29(8):1053–1068, 2010.

[Puterman, 1994] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, 1994.

[Rosenfeld *et al.*, 2012] A. Rosenfeld, Z. Bareket, C.V. Goldman, S. Kraus, D.J. LeBlanc, and O. Tsimhoni. Toward adapting cars to their drivers. *AI Magazine*, Winter:46 – 58, 2012.

[Sabbadin *et al.*, 2007] R. Sabbadin, J. Lang, and N. Ravaonjanahary. Purely epistemic markov decision process. In *Proceedings of the* 22^{nd} *National Conference on Artificial Intelligence*, 2007.

[Satinder Singh *et al.*, 2002] Diane Litman Satinder Singh, Michael Kearns, and Marilyn Walker. Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system. *Journal of Artificial Intelligence Research*, 16:105–133, 2002.

[Shani *et al.*, 2005] Guy Shani, David Heckerman, and Ronen I. Brafman. An mdp-based recommender system. *Jouornal of Machine Learning Research*, 6:1265–1295, 2005.

[Sisbot *et al.*, 2007] E. A. Sisbot, L. F. Marin-Urias, R. Alami, and T. Simeon. A human aware mobile robot motion planner. *IEEE Transactions on Robotics*, 23(5):874–883, 2007.

[Sutton and Barto, 1998] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[Weissman *et al.*, 2003] Tsachy Weissman, Erik Ordentlich, Gadiel Seroussi, Sergio Verdu, and Marcelo J. Weinberger. Inequalities for the L1 deviation of the empirical distribution. Technical Report HPL-2003-97R1, Hewlett-Packard Labs, 2003.