An Inverse Reinforcement Learning Algorithm for Partially Observable Domains with Application on Healthcare Dialogue Management

Hamid R. Chinaei and Brahim Chaib-draa
Computer Science Department, Laval University
Québec, Québec, Canada
hrchinaei@damas.ift.ulaval.ca chaib@ift.ulaval.ca

***Abstract*—In this paper, we propose an algorithm for learning a reward model from an expert policy in partially observable Markov decision processes (POMDPs). The problem is formulated as inverse reinforcement learning (IRL) in the POMDP framework. The proposed algorithm then uses the expert trajectories to find an *unknown* reward model-based on the *known* POMDP model components. Similar to previous IRL work in Markov Decision Processes (MDPs), our algorithm maximizes the sum of the margin between the expert policy and the intermediate candidate policies. However, in contrast to previous work, the expert and intermediate candidate policy values are approximated using the beliefs recovered from the expert trajectories, specifically by approximating expert belief transitions.**

We apply our IRL algorithm to a healthcare dialogue POMDP where the POMDP model components are estimated from real dialogues. Our experimental results show that the proposed algorithm is able to learn a reward model that accounts for the the expert policy.

I. Introduction

Reinforcement learning (RL) is a formal framework for optimizing a *defined* reward model. In this context, RL in the Markov decision process (MDP) framework has been investigated in several applications such as dialogue management [1], autonomous navigation [2], Smartphone interruptibility [3], etc. However, defining the reward model is not straightforward in practical domains. Thus, inverse reinforcement learning (IRL) is used to approximate the reward function that some expert agent appears to be optimizing. To this end, [4] proposed multiple IRL algorithms in the MDP framework. These algorithms account for the case in which the expert policy is represented explicitly as well as the case where the expert policy is known only through observed *expert trajectories*.

Moreover, there is a basic assumption in MDPs: the true state of the environment is fully observable. In practical problems, however this assumption is generally too strong. For instance, in spoken dialogue management, the assumption of complete observability can fail due to automatic speech recognition (ASR) errors. In this context, partially observable Markov decision processes (POMDPs) have shown robust performance to ASR errors as they maintain multiple hypotheses of the user intention [5].

IRL in POMDPs, in short POMDP-IRL, is particularly challenging due to the difficulty in solving POMDPs. Recently, [6] proposed POMDP-IRL algorithms by extending MDP-IRL algorithms of [4] to POMDPs. In particular, [6] provided a general framework for POMDP-IRL by modelling the expert policy as a finite state controller (FSC) and used point-based policy iteration [7] as POMDP solver. The trajectory-based algorithms in [6] also required the FSC-based POMDP solver [7]. Since such algorithms spent most of the time solving the intermediate policies, they suggested modifying them to be able to use other solvers such as point-based value iteration (PBVI) ones.

In this paper, we extend the trajectory-based IRL algorithm of [4] to POMDPs. We assume that the model components are known similar to the methods of [4] as well as [6]. Our algorithm also maximizes the sum of the margin between the expert policy and the intermediate candidate policies. In contrast to previous work that relies on states, we use a finite number of expert beliefs that occurred in expert trajectories. For this, we approximate a belief transition model similar to the state transition model in MDPs. The approximated belief transition model is used to approximate the value of the expert policy and the value of intermediate candidate policies.

Note that trajectory-based POMDP-IRL addresses a practical problem in spoken dialogue management. In particular, in dialogue management, the transition and observation models can be calculated from data, such as dialogues collected by a Wizard-of-Oz setting [6]. Then, the dialogue trajectories can be used to learn a reward model for expert policy. To this end, we apply our IRL algorithm on a healthcare dialogue POMDP for which we learned the POMDP model components from real dialogues collected by the SmartWheeler [8] platform. SmartWheeler is a robotic wheelchair for persons with disabilities, which is described in Section IV-B.

In the rest of this paper, first we describe the necessary background for MDPs and POMDPs in Section II. We propose an IRL algorithm for POMDPs using expert trajectories in Section III. Then, in Section IV we evaluate our proposed algorithm on two well-known POMDP benchmarks and the learned dialogue POMDP from the SmartWheeler dialogues. Finally, we conclude this work in Section V.

II. Background

A MDP is a machine learning framework for decision making under uncertainty in Markovian domains. An MDP can be defined as $(S, A, T, \gamma, R, s_0)$ where, S is the set of discrete states, and A is the set of discrete actions. Moro ever, the transition model T consists of the probabilities of state transitions (dynamics of the system): $T(s, a, s') = Pr(s_{t+1} = s' | a_t = a, s_t = s)$ where s is the current state and s' is the next state. Furthermore, the reward model is $R(s, a)$ that specifies the reward of taking action a in the state s. Finally, the initial state is denoted by s_0.

In the MDP framework the objective is to find an optimal policy π^*, where for any state s, π^* specifies an action $a = \pi^*(s)$ that maximizes the expected value of future rewards starting from state s: $V^\pi(s) = E_{s_t \sim T}[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) | \pi, s_0 = s]$. The expected value of the reward can be computed recursively by Bellman equation defined as: $V^\pi(s) = [R(s, \pi(s)) + \gamma \sum_{s' \in S} T(s, \pi(s), s') V^\pi(s')]$.

A POMDP is a more generalized AI model for planning under uncertainty. The POMDP framework considers that the states are only partially observable through some *observations*. Thus, a POMDP is represented as a tuple $(S, A, T, \gamma, R, O, \Omega, b_0\}$. That is, a POMDP model includes an MDP model and adds O, the set of observations, Ω is the observation model defined as $\Omega(a, s', o') = Pr(o' | a, s')$ for the probability of observing o' after taking the action a which resulted in the state s'. Finally, the initial belief over all states is denoted by b_0.

The primary difference between POMDPs and MDPs is that, where a MDP has a particular current state, a POMDP (theoretically) maintains a probability distribution over all possible states. This distribution is known as a *belief state*. Given a belief state b, $b(s)$ is the probability assigned to a particular state s. After an action a is taken and observation o' is received, a new belief state, b', can be computed. This is done by using a *state estimator function* ($SE(b, a, o')$) to calculate the probability of some new state s', $b'(s')$

$$\begin{aligned} b'(s') &= SE(b, a, o') = Pr(s' | b, a, o') \\ &= \eta \Omega(a, s', o') \sum_{s \in S} b(s) T(s, a, s') \end{aligned} \quad (1)$$

where η is the normalization factor.

Notice that the important property of the belief state is that it is a sufficient statistic. In other words, the belief state at time t, b_t, summarizes the initial belief b_0, as well as all the actions taken and all the observations received. Formally, we have: $b_t(s) = Pr(s | b_0, a_0, o_0, \ldots, a_{t-1}, o_{t-1})$. In the POMDP framework the objective is to find an optimal Markov policy π^*, where for any belief state b, π^* specifies an action $a = \pi^*(b)$ that maximizes the expected discount of future rewards starting from belief b_0: $V^\pi(b) = E_{b_t \sim SE}[\sum_{t=0}^{\infty} \gamma^t R(b_t, \pi(b_t)) | \pi, b_0 = b]$, where

$$R(b, a) = \sum_{s \in S} b(s) R(s, a) \quad (2)$$

Then, we have the optimal policy π^* as:$\pi^*(b) = \arg\max_\pi V^\pi(b)$. The expected value of the reward can be computed recursively using Bellman equation for V^π.

$$V^\pi(b) = R(b, \pi(b)) + \gamma \sum_{o' \in O} Pr(o' | b, \pi(b)) V^\pi(b') \quad (3)$$

Note that, if the POMDP has a deterministic observation model and a deterministic initial belief, we can approach a POMDP as a MDP. This can be seen in Equation (1). By starting with a deterministic initial belief, the next belief will be deterministic if the observation model is deterministic.

III. An IRL algorithm for POMDPs

The IRL task is to find a reward model in which the expert's policy is both optimal and maximally separated from other policies. To do this, some candidate reward models and their subsequent policies are generated from the expert's behaviour. The candidate reward model is approximated by maximizing the value of the expert policy with respect to all previous candidate policies. The new candidate reward model and policy are then used to approximate another new set of models. This process iterates until the difference in values of successive candidate policies is less than some threshold. The final candidate reward model is the solution to the IRL task [4].

We propose an IRL algorithm in the POMDP framework, called as POMDP-IRL-BT (BT for belief transitions). In fact, POMDP-IRL-BT is similar to the basic trajectory-based MDP-IRL algorithm proposed in [4]. However, instead of states we use the finite number of expert beliefs that occurred in expert trajectories. Moreover, instead of transition model in MDPs we approximate a belief transition for expert beliefs.

Formally, we formulate the IRL problem as a POMDP without a reward model, noted as POMDP$\backslash R = \{S, A, T_a, O, \Omega, b_0, \gamma\}$, so that we can calculate the optimal policy of the POMDP given any choice of candidate reward model. Having t candidate policies $\pi_1, \ldots, \pi_t$, the next candidate reward is estimated by maximizing $\boldsymbol{d}^t$, the sum of the margins between value of expert policy and each learned candidate policy. Then, the objective function is $\boldsymbol{d}^t = (\boldsymbol{v_b}^{\pi_E} - \boldsymbol{v_b}^{\pi_1}) + \ldots + (\boldsymbol{v_b}^{\pi_E} - \boldsymbol{v_b}^{\pi_t})$, where $\boldsymbol{v_b}^{\pi_E}$ is an approximation of the value of the expert policy that occurred in expert trajectories. Moreover, each $v_{\boldsymbol{b}}^{\pi_t}$ is an approximation of value of the candidate policy π_t which is calculated by approximating expert belief transitions.

To illustrate these approximations, consider the value function for POMDPs shown in Equation (3). Using vector representation, we can rewrite Equation (3) as:

$$\boldsymbol{v}_b^\pi = \boldsymbol{r}_b^\pi + \gamma \boldsymbol{P}^\pi \boldsymbol{v}_b^\pi \quad (4)$$

where

- $\boldsymbol{v}_b^\pi$ is a vector of size N: the number of expert beliefs in which $\boldsymbol{v}_b^\pi(b) = V^\pi(b)$ (from Equation (3)).
- $\boldsymbol{r}_b^\pi$ is a vector of size N in which $\boldsymbol{r}_b^\pi(b) = R(b, \pi(b))$, where $R(b, a)$ comes from Equation (2).
- $\boldsymbol{P}^\pi$ is a matrix of size $N \times N$ that is the *belief transition* matrix for policy π, where:

$$\boldsymbol{P}^\pi(b, b') = \sum_{o' \in O} [Pr(o'|b, \pi(b)) \; \textit{ifClosest}((\textit{SE}(b, \pi(b), o'), b')]$$

where *SE* is the state estimator function in Equation (1) and *ifClosest*(b'', b') determines if b' is the closest expert belief to b'', the belief created as result of state estimator function. Formally, we define *ifClosest*(b'', b') as:

$$\textit{ifClosest}(b'', b') = \begin{cases} 1, & \text{if } b' = \arg\min_{b_n} |b'' - b_n| \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

where b_n is one of the N expert beliefs that appeared within the expert trajectories.

$P^\pi(b, b')$ is an approximate belief state transition model. It is approximated in three steps. First, the next belief b'' is estimated using the *SE* function. Second, the *ifClosest* function is used to find, b', the nearest belief that occurred within the expert trajectories. Finally, the transition probability between b and b' is updated using Equation (5). This avoids handling the excessive number of new beliefs created by the *SE* function. More importantly, this procedure supports the use of IRL on a fixed number of beliefs, such as expert beliefs from a fixed number of trajectories.

We construct the rest of formulations similar to MDPs. The reward model, R, is represented using the vector of features $\boldsymbol{\phi}$ so that each $\phi_i(s, a)$ is a basis function for the reward model. However, in POMDPs, we need to extend state features to beliefs. To do so, we define vector $\phi(b, a)$ as: $\boldsymbol{\phi}(b, a) = \sum_{s \in S} b(s) \boldsymbol{\phi}(s, a)$. Then, matrix $\boldsymbol{\Phi}_b^\pi$ is an $N \times K$ matrix of belief action features for policy π, defined as:

$$\boldsymbol{\Phi}_b^\pi = \begin{pmatrix} \boldsymbol{\phi}(b_0, \pi(b_0))^{\mathrm{T}} \\ \dots \\ \boldsymbol{\phi}(b_{N-1}, \pi(b_{N-1}))^{\mathrm{T}} \end{pmatrix}$$

For the expert policy π_E, we define $\boldsymbol{\Phi}_b^{\pi_E}$ as:

$$\boldsymbol{\Phi}_b^{\pi_E} = \begin{pmatrix} \boldsymbol{\phi}(b_0, \pi_E(b_0))^{\mathrm{T}} \\ \dots \\ \boldsymbol{\phi}(b_{N-1}, \pi_E(b_{N-1}))^{\mathrm{T}} \end{pmatrix}$$

By manipulation of Equation (4), we have $\boldsymbol{v}_{\boldsymbol{b}}^\pi = (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)^{-1} \boldsymbol{r}_{\boldsymbol{b}}^\pi$. The reward model can be represented as linear combination of features: $\boldsymbol{r}_b^\pi = \boldsymbol{\Phi}_b^\pi \boldsymbol{\alpha}$. As such, the vector of values $\boldsymbol{v}_{\boldsymbol{b}}^\pi$ can be represented as the multiplication of vector of feature weights $\boldsymbol{\alpha}$ and vector $\boldsymbol{x}_{\boldsymbol{b}}^\pi$. So, we have $\boldsymbol{v}_b^\pi = \boldsymbol{x}_b^\pi \boldsymbol{\alpha}$, where $\boldsymbol{x}_b^\pi$ is a matrix of size $N \times K$ defined as:

$$\boldsymbol{x}_b^\pi = (\boldsymbol{I} - \gamma \boldsymbol{P}^\pi)^{-1} \boldsymbol{\Phi}_b^\pi \quad (6)$$

We have a similar equation for the expert policy: $\boldsymbol{v}_b^{\pi_E} = \boldsymbol{x}_b^{\pi_E} \boldsymbol{\alpha}$, where $\boldsymbol{x}_b^{\pi_E}$ is a matrix of size $N \times K$ defined as:

$$\boldsymbol{x}_b^{\pi_E} = (\boldsymbol{I} - \gamma \boldsymbol{P}^{\pi_E})^{-1} \boldsymbol{\Phi}_b^{\pi_E} \quad (7)$$

where $\boldsymbol{P}^{\pi_E}$ is an $N \times N$ matrix where each element $\boldsymbol{P}^{\pi_E}(b_i, b_j)$ is the probability of transiting from b_i to b_j with expert action $\pi_E(b_i)$.

Algorithm 1 shows POMDP-IRL-BT. Similar to MDP-IRL, this algorithm maximizes the sum of the margins between the expert policy π_E and the candidate policies π_t (line 7 of Algorithm 1). The POMDP-IRL-BT algorithm is based on the belief transition model, as opposed to MDP-IRL which is based on transition of completely observable states.

Let's go through Algorithm 1 in detail. The algorithm starts by randomly initiating values for $\boldsymbol{\alpha}$ to generate the initial candidate reward model R^1 in line 1. Then, the algorithm finds the policy of R^1, denoted by π_1 in line 2, using a model-based POMDP algorithm such as Perseus [9]. In line 3, $\boldsymbol{P}^{\pi_1}$ is constructed, which is used to calculate $\boldsymbol{x}_b^{\pi_1}$ from Equation (6). Then, in line 4, the expert policy π_E is used to construct $\boldsymbol{P}^{\pi_E}$ which is used to calculate $\boldsymbol{x}_b^{\pi_E}$ from Equation (7). From line 5 to line 16, the algorithm iterates to learn the expert reward model by solving the linear program in line 7 with the constraints shown in line 8. In this optimization, we also constrain the value of the expert's policy to be greater than that of other policies in order to ensure that the expert's policy is optimal.

IV. Experiments

This section presents the POMDP-IRL-BT experiments on two sets of problems. First in Section IV-A, we apply the algorithm on two POMDP benchmarks. Then, in Section IV-B, we apply the algorithm on the learned dialogue POMDP from real dialogues collected by SmartWheeler. The problems and their results are summarized in Table I.

For each problem, we assumed an expert reward model R^{π_E} and used the POMDP model to find the expert policy π_E. The learned expert policy is used to sample N expert trajectories to be used in the algorithm. For our experiments, we used Perseus which is a PBVI POMDP solver [9]. PBVI solvers use a finite number of beliefs for solving a POMDP model approximately. The number of beliefs is a parameter that is generally hand-tuned. For our experiments, the solver used 10000 random samples to find the optimal policy of each candidate reward. The other parameter is max-time for execution of the algorithm, which is set to 1000.

We evaluate the learned reward model R using its optimal policy π known as the learned policy. For each problem, we generate two expert trajectories of the same size, one for IRL training and the other for IRL testing. To evaluate the learned policy, we find the number of *matched* actions between the learned policy and the expert policy on the testing trajectory (see Table I).

Input: $POMDP \backslash R = \{S, A, T, \gamma, O, \Omega, b_0\}$, expert trajectories and a vector of features $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_K)$, convergence rate ϵ, and maximum Iteration $maxT$
Output: Finds the reward model R where $R = \sum_i \alpha_i \phi_i(s, a)$, by approximating $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K)$

Choose the initial reward R^1 by randomly initializing feature weights $\boldsymbol{\alpha}$;
Set $\Pi = \{\pi_1\}$ by finding π_1 using POMDP with candidate reward model R^1 and a PBVI variant POMDP solver;
Set $X = \{\boldsymbol{x}_b^{\pi_1}\}$ by calculating $\boldsymbol{x}_b^{\pi_1}$ using $\boldsymbol{P}^{\pi_1}$ and Equation (6);
Calculate $\boldsymbol{x}_b^{\pi_E}$ from Equation (7);
for $t = 1$ *to maxT* **do**
    Find values for $\boldsymbol{\alpha}$ by solving the linear program:
    maximize$_\alpha$ $[((\boldsymbol{x}_b^{\pi_E} - \boldsymbol{x}_b^{\pi_1}) + \ldots + (\boldsymbol{x}_b^{\pi_E} - \boldsymbol{x}_b^{\pi_t}))\, \boldsymbol{\alpha}]$;
    subject to $-1 \leq \alpha_i \leq +1 \; \forall i \; 1 \leq i \leq K$;
    $R^{t+1} = \sum_i \alpha_i^t \phi_i(s, a)$;
    if $max_i |\boldsymbol{\alpha}_i^t - \boldsymbol{\alpha}_i^{t-1}| \leq \epsilon$ **then**
        return R^{t+1};
    end
    else
        $\Pi = \Pi \cup \{\pi_{t+1}\}$ by finding π_{t+1} using POMDP with candidate reward model R^{t+1} and a PBVI variant POMDP solver;
        Set $X = X \cup \{\boldsymbol{x}_b^{\pi_{t+1}}\}$ by calculating $\boldsymbol{x}_b^{\pi_{t+1}}$ using $\boldsymbol{P}^{\pi_{t+1}}$ and Equation (6);
    end
end

Algorithm 1: POMDP-IRL-BT: Inverse Reinforcement Learning for POMDPs using Belief Transitions.

A. Experiments on POMDP Benchmarks

We applied our proposed algorithm on the *Tiger* and *Rock Sample* problems, that are respectively the smallest and largest benchmarks for the experiment in [6]. For the explanation of the two problems the reader is refereed to [6]. We used the two benchmarks with the same specification used in [6] that is summarized in Table I.

In the tiger problem, there are 2 states, 2 observations, 3 actions, and the discount factor is set to 0.75. We generated two trajectories of size 20 to be used for training and testing. In the rock sample problem, there are 129 states, 2 observations, 8 actions, and the discount factor is 0.95. The size of training and testing trajectories is 200, with the maximum step size of 20.

Recall that IRL needs features for learning the reward model. For both benchmarks, we used *state-action-wise* features in the same way they are defined in [6]. That is, the features include the indicator function for each state action pair. Thus, the feature size for tiger problem equals $6 = 2 \times 3$, and for rock sample equals $1032 = 129 \times 8$.

We performed the algorithm on the training expert trajectory using state-action-wise features. Table I shows that the learned policies for the tiger and rock sample problems matched with the expert policy respectively for the 100% and 99.5% of the beliefs in their testing trajectories. In words, the proposed algorithm is able to learn a reward model for the expert policy fairly well.

B. Experiments on the SmartWheeler Domain

The SmartWheeler project aims at developing an intelligent wheelchair that minimizes the physical and cognitive load required in steering it [8]. Table II shows a sample of SmartWheeler dialogues captured for training dialogue models. The first line, noted as u_1, shows a transcription of the user utterance. The following line, noted as $\tilde{u}_1$, is the 1-Best ASR result of the utterance. Finally, the last line, noted as a_1, shows the action executed by the SmartWheeler. For instance, in the second turn, the user has uttered *turn right a little*, however, it has been mis-recognized as *10 writer little*. The action performed by the SmartWheeler for this user utterance is PLEASE REPEAT YOUR COMMAND, called *query action*, for gathering more certainty about the user intention. More details about SmartWheeler and its collected dialogues can be found in [8], and [10].

We used the dialogues collected by SmartWheeler to learn a dialogue POMDP [10]. The learned dialogue POMDP model for SmartWheeler consists of 11 states that are the learned user intentions from SmartWheeler dialogues. The states include: s_1: *move-forward-little*, s_2: *move-backward-little*, s_3: *turn-right-little*, s_4: *turn-left-little*, s_5: *follow-left-wall*, s_6: *follow-right-wall*, s_7: *turn-degree-right*, s_8: *go-door*, s_9: *set-speed-med*, s_{10}: *follow-person*, and s_{11} *stop*. There are 24 actions among which only 12 actions appeared in the expert policy. One is the query action, described above, and there are 11 actions called as command actions, each of which is the right action for its subsequent state. For instance, TURN RIGHT LITTLE is the right action for s_3: *turn-right-little*. The expert policy is the optimal policy of the assumed expert reward model, defined below.

There are 11 observations for the dialogue POMDP, called intention observation in [10], each of which shows its subsequent state with high probability and other states with low probability. The observation model, as well as the transition model, have been learned from dialogues using maximum likelihood methods.

For the reward model of the expert R^{π_E}, similar to previous works, we considered +1 reward for the SmartWheeler performing the right action at each state, and 0 otherwise. Moreover, for the general query action the reward is considered as +0.4 in every state.

For the choice of features, we automatically learned *Keyword* features. That is, a vector consists of SmartWheeler 1-top words for each user intention. The 1-top words are keywords and have been automatically learned during learning the user intentions [10]. The keyword features

Table I
THE PROBLEM SPECIFICATIONS AND THE RESULTS FOR POMDP-IRL-BT EXPERIMENTS USING STATE-ACTION-WISE FEATURES.

Problem	$\|S\|$	$\|A\|$	$\|O\|$	γ	feature size	trajectory size	# of matched actions	matched percentage
Tiger	2	3	2	0.75	6	20	20	100%
Rock Sample[4,3]	129	8	2	0.95	1032	200	199	99.5%
SmartWheeler	11	24	11	0.90	264	200	200	100%

Table II
SMARTWHEELER SAMPLE DIALOGUES.

u_1 :	turn right
$\tilde{u}_1$:	[turn right]
a_1 :	TURN RIGHT
u_2 :	turn right a little
$\tilde{u}_2$:	[10 writer little]
a_2 :	PLEASE REPEAT YOUR COMMAND
u_3 :	turn right a little
$\tilde{u}_3$:	[turn right to lead a]
a_3 :	TURN RIGHT A LITTLE
u_4 :	stop
$\tilde{u}_4$:	[stop]
a_4 :	STOP

Table III
KEYWORD FEATURES FOR SMARTWHEELER.

	forward	*backward*	***right***	***left***	*turn*	*go*	*for*	*top*	*stop*
s_1	1	0	0	0	0	0	0	0	0
s_2	0	1	0	0	0	0	0	0	0
s_3	0	0	**1**	0	0	0	0	0	0
s_4	0	0	0	**1**	0	0	0	0	0
s_5	0	0	0	**1**	0	0	0	0	0
s_6	0	0	**1**	0	0	0	0	0	0
s_7	0	0	0	0	1	0	0	0	0
s_8	0	0	0	0	0	1	0	0	0
s_9	0	0	0	0	0	0	1	0	0
s_{10}	0	0	0	0	0	0	0	1	0
s_{11}	0	0	0	0	0	0	0	0	1

for each state of SmartWheeler dialogue POMDP is represented in Table III. Notice that states 3 (s_3: *turn-right-little*) and 6 (s_6: *follow-right-wall*) share the same features, i.e., *right*. Moreover, states 4 (s_4: *turn-left-little*) and 5 (s_5: *follow-left-wall*) share the same feature, i.e., *left*. We used keyword-action-wise features. That is, the features include the indicator functions for each pair of state-keyword and action. Thus, the feature size for SmartWheeler problem equals $216 = 9 \times 24$.

We performed POMDP-IRL-BT on the SmartWheeler training expert trajectory. We observed that the policy of the learned reward is the same as the expert policy for 194 beliefs out of the 200 beliefs inside the testing trajectory. For all 6 errors, the expert action is TURN RIGHT LITTLE, i.e., the right action for state *turn-right-little*, while the action of the learned reward suggests FOLLOW RIGHT WALL. However, this error does not occur in all the cases which the expert action is TURN RIGHT LITTLE in testing trajectory.

Afterwards, we used state-action-wise features as described in Section IV-A. The size of state-action-wise features for the SmartWheeler POMDP equals $264 = 11 \times 24$, which is a slight increase compared to the size of keyword features. The result of the experiment on the SmartWheeler POMDP using state-action-wise features is shown in Table I. The table shows that the learned policy is exactly the same as the expert policy for the 200 beliefs inside the testing trajectory using state-action-wise features, i.e., 100% matched with the expert policy. In words, POMDP-IRL-BT is able to learn a reward model for the expert policy based on the learned dialogue POMDP from SmartWheeler dialogues.

Comparison to the Monte Carlo estimator: As stated in the introduction, [6] provided a general framework for IRL in POMDPs by assuming that expert policy is represented in the form FSC and thus using PBPI [7] as POMDP solver. In their trajectory-based algorithms, the authors used the Monte Carlo estimator for approximating the policy values. Specifically, using an expert trajectory of size N, the value of expert policy is calculated by Monte Carlo estimator as: $V^{\pi_E}(b_0) = \boldsymbol{\alpha}^T \sum_{t=0}^{N-1} \gamma^t \phi(b_t, a_t)$.

We implemented the Monte Carlo estimator for the value of policies in Algorithm 1 and used the Perseus software [9] as the POMDP solver. In the experiments below, our implementation of Monte-Carlo in POMDP-IRL-BT algorithm is labelled POMDP-IRL-MC (MC for Monte Carlo estimator). The two algorithms are then compared based on the following two criteria: 1) percentage of the learned actions that matches to the expert actions, and 2) CPU time spent by the algorithm as the number of expert trajectories increases.

Figure 1 shows percentage of the matched actions between the learned and expert policies as the number of iterations increases, using state-action features. The figure shows that this percentage reaches to 100% in POMDP-IRL-BT, while it reaches to 97% in POMDP-IRL-MC. Furthermore, the figure shows that the best performance of POMDP-IRL-BT occurs slightly earlier than that of POMDP-IRL-MC (iteration 13 vs. iteration 17).

Finally, Figure 2 demonstrates the spent time by POMDP-IRL-BT and POMDP-IRL-MC as the number of expert trajectories increases (in the log base). The results show that by increasing the number of expert trajectories POMDP-IRL-BT demands considerably more time than POMDP-IRL-MC. This increase is due to the size of belief transition matrix increasing, Equation (5), as the number of expert trajectories increases. In other words, the belief transition matrix demands much more time to be constructed as the

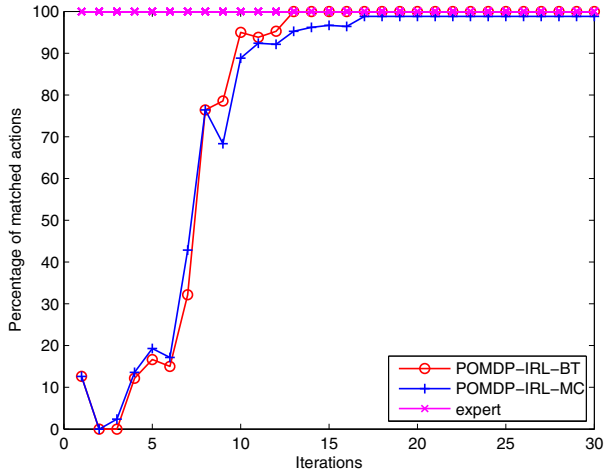

Figure 1. Comparison of the POMDP-IRL algorithms using state-action-wise features on the dialogue POMDP learned from SmartWheeler.

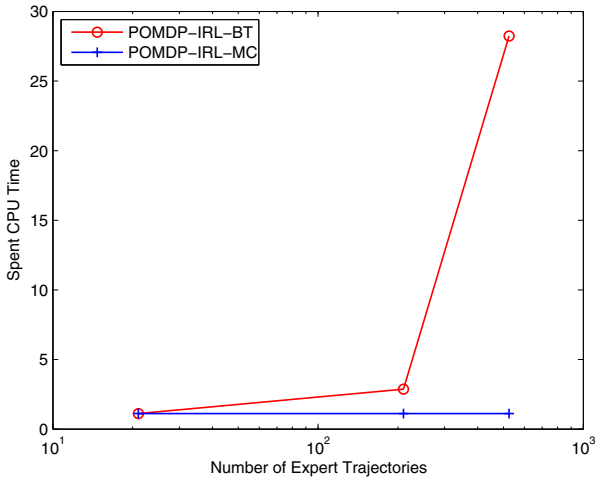

Figure 2. Spent CPU time by POMDP-IRL algorithms on SmartWheeler, as the number of expert trajectories (training data) increases.

number of beliefs in expert trajectories increases. Also, note that this matrix is constructed for each candidate policy, which in turn increases the CPU time. In the case of large number of expert trajectories, POMDP-IRL-BT can still be useful. For instance, we can use all expert trajectories to estimate the transition and observation models, but, select part of the expert trajectories to learn the reward model.

V. Conclusion

We have introduced a new IRL algorithm, POMDP-IRL-BT, for learning the reward model of expert policy in partially observable environments. This method requires the model components of the underlying POMDP to be known, while the expert policy can be represented implicitly using expert trajectories. We evaluated this algorithm on two well-known POMDP benchmarks. Moreover, we evaluated our algorithm on a dialogue POMDP for which we had learned the model components from a healthcare robot. Our experimental results on all the domains showed that the proposed algorithm is able to learn a reward model that accounts for the expert policy. In the future, we are going to further research on POMDP-IRL for larger practical domains with automatically learned features.

Acknowledgement

The authors thank Jason D. Williams and Suhrid Balakrishnan for helpful discussions in the early development of this work. The authors also thank Joelle Pineau for providing them with the Smartwheeler data. The dataset has been collected with contributions from researchers at McGill University, École Polytechnique de Montréal, Université de Montréal, and the Centres de réadaptation Lucie-Bruneau and Constance-Lethbridge. Last but not least, the authors thank Ethan Selfridge for his help in proofreading the paper.

References

[1] M. Frampton and O. Lemon, "Recent research advances in reinforcement learning in spoken dialogue systems," *Knowledge Engineering Review*, vol. 24, no. 4, pp. 375–408, 2009.

[2] E. D. S. Costa and M. M. G. Jr., "Autonomous Navigation in Dynamic Environments with Reinforcement Learning and Heuristic," in *2010 9th International Conference on Machine Learning and Applications*. IEEE, 2010, pp. 37–42.

[3] R. Fisher and R. Simmons, "Smartphone Interruptibility using Density-weighted Uncertainty Sampling with Reinforcement Learning," in *2011 10th International Conference on Machine Learning and Applications Workshops*. IEEE, 2011, pp. 436–441.

[4] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *Proceedings of the 17th International Conference on Machine Learning (ICML'00), Stanford, CA, USA*, 2000.

[5] J. D. Williams and S. Young, "Partially observable Markov decision processes for spoken dialog systems," *Computer Speech and Language*, vol. 21, pp. 393–422, 2007.

[6] J. Choi and K.-E. Kim, "Inverse reinforcement learning in partially observable environments," *Journal of Machine Learning Research*, vol. 12, pp. 691–730, 2011.

[7] S. Ji, R. Parr, H. Li, X. Liao, and L. Carin, "Point-based policy iteration," in *Proceedings of the 22nd national conference on Artificial Intelligence - Volume 2 (AAAI'07), Vancouver, British Columbia, Canada*, 2007.

[8] J. Pineau, R. West, A. Atrash, J. Villemure, and F. Routhier, "On the feasibility of using a standardized test for evaluating a speech-controlled smart wheelchair," *International Journal of Intelligent Control and Systems*, vol. 16, no. 2, pp. 124–131, 2011.

[9] M. Spaan and N. Vlassis, "Perseus: Randomized point-based value iteration for POMDPs," *Journal of Artificial Intelligence Research*, vol. 24, no. 1, pp. 195–220, 2005.

[10] H. R. Chinaei, B. Chaib-draa, and L. Lamontagne, "Learning observation models for dialogue POMDPs," in *Proceedings of the 24th Canadian conference on advances in Artificial Intelligence (Canadian AI'12), Toronto, Ontario, Canada*, 2012.