A Predictive Model for Imitation Learning in Partially Observable Environments

Abdeslam Boularias
Department of Computer Science, Laval University, Canada, G1K 7P4
boularias@damas.ift.ulaval.ca

Abstract

Learning by imitation has shown to be a powerful paradigm for automated learning in autonomous robots. This paper presents a general framework of learning by imitation for stochastic and partially observable systems. The model is a Predictive Policy Representation (PPR) whose goal is to represent the teacher's policies without any reference to states. The model is fully described in terms of actions and observations only. We show how this model can efficiently learn the personal behavior and preferences of an assistive robot user.

1 Introduction

Domestic and assistive robots are becoming increasingly prevalent in the research community, and the hope is that in the next 5-10 years, they will also permeate our living and working environments. The environments of such systems are typically complex, stochastic and partially observable, and the planning task should take into account these problems in order to get robust performance. There are two main models used to represent these types of environments: Partially Observable Markov Decision Processes (POMDPs) [14], and more recently, Predictive State Representations (PSRs) [10]. However, learning a model of the environment often requires a large amount of sampled data before getting accurate parameters. Imitative learning [1, 2] is a well-known technique that provides a fast and efficient way of acquiring new skills without the need for extensive experimentation. It is also justified somewhat by the fact that much of biological learning is done by imitation. In assistive robotics, where the human-machine interaction is omnipresent, imitative learning can be a powerful learning paradigm. In the imitation process, a *learner* agent observes the actions of a *teacher* agent, and tries to find the relation between the actions and the different encountered situations. A growing body of work has been done over the past few years on the imitative learning framework for fully observable systems, but there has been less focus on applying these ideas to partially observable environments. Some of the proposed solutions [6, 15] use a POMDP model to represent the policy of the teacher, and then learn the parameters of this POMDP by using the Expectation-Maximization method [9], which needs considerable data for learning, and is subject to local minima unless a good initialization of the parameters is provided. Initializing the parameters of a policy is a difficult problem because the states of a given policy correspond to mental states rather than to physical situations. In this paper, we propose a new method to learn the teacher's policy by using predictive representations [10], which are generally easier to learn than state-based representations [7]. We start by giving an overview of the two problems that will be used as a benchmark for our approach, then we present a background on POMDP and PSR frameworks. We describe the new imitative learning framework, and conclude by presenting an empirical analysis of our algorithm.

2 Motivating problems

Intelligent wheelchairs are one instance of assistive robots where the human user is constantly interacting with the machine. They are generally designed to assist individuals with mobility impairments who have difficulty with fine motor skills, and therefore cannot easily or safely operate a standard motorized wheelchair. The intelligent controller is designed to reduce their physical (and sometimes cognitive) load. Many of these individuals however are not willing to relinquish full control to an intelligent controller, thus a shared control solution is preferred.

As part of this shared control, the intelligent wheelchair should be able to detect the anomalous behavior of the user and to switch into automatic control mode (or alert the user). This problem can be cast as a problem of learning a policy for the human controller, and thus is a suitable candidate for the methods we present in this paper.

The second problem we are interested in addressing is the classical pathfinding problem. In general, this problem can be solved efficiently if the environment dynamics are known in advance (e.g. using A*-type methods, or the more

complex POMDP framework). But these approaches work best when the optimization criterion is simple (e.g. minimum distance). Note however that the shortest path is not always the user's preferred path, and in some cases the user may wish to choose between several short paths. This problem can also be cast in the imitation learning framework, and solved with the methods outlined below. We will return to these problems in the latter part of this paper.

3 Background

3.1 POMDPs

Formally, a POMDP [14] is defined by the following parameters: the system states set $\mathcal{S}$; the agent actions set $\mathcal{A}$; the observations set O; the transition and observation function[1] $T(s_i, a_i, o_i, s_{i+1})$ which returns the probability of transiting from the state s_i to the state s_{i+1} and observing o_i after having executed a_i; and the reward function $R(s_i, a_i)$.

In POMDPs, the hidden state is represented by a probability distribution over the states, called the belief state: $b_i = [Pr(s_i = s^0), Pr(s_i = s^1), \ldots, Pr(s_i = s^{|\mathcal{S}|-1})]^T$. The Markov property guarantees that the belief state and the full history of the system contain exactly the same information about the actual state. In fact, we start with an initial belief state b_0, and every time we execute an action a_i and receive an observation o_i, we update the belief state b_i by:

$$b_{i+1}(s^j) = \frac{\sum_{s^k \in \mathcal{S}} b_i(s^k) T(s^k, a_i, o_i, s^j)}{\sum_{s^k \in \mathcal{S}} \sum_{s^l \in \mathcal{S}} b_i(s^k) T(s^k, a_i, o_i, s^l)} \quad (1)$$

Baum-Welch algorithm (an adaptation of the Expectation-Maximization method) is considered as the standard algorithm for learning the parameters of a POMDP [9]. It finds the parameters that locally maximize the likelihood of a given sequence of actions and observations.

3.2 PSRs

PSRs [12] are an alternative model for representing partially observable environments without reference to hidden variables. The fundamental idea of PSRs is to replace the probabilities on states by probabilities on future possible trajectories, called *tests*. A test $t = a^1 o^1 \ldots a^k o^k$ is an ordered sequence of actions and observations. The probability of t starting at time step i is defined by:

$$Pr(t|h_i) = Pr(o_{i+1} = o^1, \ldots o_{i+k} = o^k | h_i, a_{i+1} = a^1, \ldots a_{i+k} = a^k)$$

where $h_i = a_0 o_0 \ldots a_i o_i$ is the history until the step i.

The *system-dynamics matrix* D is an infinite matrix where $\forall t, h : D(t,h) = Pr(t|h)$. This matrix can be seen as a general model, that can describe any discrete dynamical system, since it returns the probability of any event after any history. In practice, we can only generate a finite submatrix of D that should contain the same information as the full matrix. For a large category of dynamical systems, we can prove that the probability of any test t depends only on the probabilities of a few tests, called *core tests*, which constitute a sufficient statistic for the system.

We indicate the core tests by $q^1, q^2, \ldots q^N$. Q indicates the set of these tests, and $Pr(Q|h_j) = (Pr(q^1|h_j), Pr(q^2|h_j), \ldots, Pr(q^N|h_j))^T$ is the probability vector for the core tests at time step j, which is equivalent to the belief state in POMDPs. We have then:

$$Pr(t^i|h_j) = f_{t^i}(Pr(Q|h_j)) \quad (2)$$

where f_{t^i} is a function associated to the test t^i, this function is independent of the history, and allows us to calculate the probability of t^i by using only the probabilities of Q. The Bayes update function for PSRs is given by:

$$Pr(q^i|h_j ao) = \frac{Pr(aoq^i|h_j)}{Pr(ao|h_j)} = \frac{f_{aoq^i}(Q|h_j)}{f_{ao}(Q|h_j)} \quad (3)$$

4 Predictive Policy Representations (PPRs)

In PPRs [3, 16], the roles of the actions and observations are switched. The probability that a test $t = o^0, a^1, \ldots o^{k-1}, a^k$ succeeds is given by:

$$Pr(t|h_i) = Pr(a_{i+1} = a^1, \ldots, a_{i+k} = a^k | h_i, o_i = o^0, \ldots, o_{i+k-1} = o^{k-1})$$

The history h_i ends with an action and not an observation as in PSRs, because tests in PPRs start with an observation. In fact, step i is the moment after executing a_i and before perceiving o_i. We also consider that all the histories start with a fictive observation o^* which has probability 1 (the default observation). A test in PPRs can be seen as a question regarding what the agent will do when it perceives a specified sequence of observations.

After executing an action a and perceiving an observation o, the following Bayes function is used to update the probabilities of the core tests:

$$Pr(q^i|h_j oa) = \frac{Pr(oaq^i|h_j)}{Pr(oa|h_j)} = \frac{m^T_{oaq^i} Pr(Q|h_j)}{m^T_{oa} Pr(Q|h_j)} \quad (4)$$

We can see from this equation that to calculate the new probability of core test q after executing a and perceiving o, we need only to know for the probabilities $Pr(oaq^i|h_j)$ and $Pr(oa|h_j)$ of the tests oa and oaq.

A Predictive Policy Representation is defined by the following parameters: Q, the core tests list; $Pr(Q|\emptyset)$, the initial probabilities of the core tests; $\forall a \in A, \forall o \in O : m_{oa}$, the vector associated to test oa; $\forall a \in A, \forall o \in O, \forall q^i \in Q : m_{oaq^i}$, the

[1] We denote by a_i (*resp.* o_i, s_i) the current action (*resp.* observation, state) at time step i, and by a^i (*resp.* o^i, s^i) a given action (*resp.* observation, state) in the set $\mathcal{A}$ (*resp.* O, $\mathcal{S}$).

vector associated to test oaq^i, composed by test oa followed by test q^i.

The *policy matrix* P is defined by the infinite set of all possible tests and histories. An entry $P(h_j,t^i)$ is given by $Pr(t^i|h_j)$, the probability that the actions indicated in the test t^i will be executed by the agent, such that the current history of the system is h_j and the future observations will be the observations indicated in t^i. The core tests of a linear PPR corresponds to a basis of the policy matrix P. Interestingly, we can prove [3] that the rank of the policy matrix corresponding to a given Finite-State Controller (FSC [5]) is upper-bounded by the number of states of this controller.

The PPR model functions as follows: we start by initializing the probabilities $Pr(Q|\emptyset)$ of the core policy tests. After observing o_{j+1} (or the default observation o^* if we did not yet execute any action, i.e. $h_j = \emptyset$), the probabilities of the core policy tests are used to calculate a distribution of probabilities over the next actions by:

$$Pr(a_{j+1} = a^i|h_j o_{j+1}) = m^T_{o_{j+1}a^i} Pr(Q|h_j) \qquad (5)$$

We select an action a_{j+1} according to this distribution, we execute this action, and then we use this new information (o_{j+1},a_{j+1}) to update our core test probabilities by using Equation 4. We add the event (o_{j+1},a_{j+1}) at the end of the history h_j, which becomes h_{j+1}, and we repeat the same process for the next steps.

5 Imitation Learning with PPRs

In partially observable environments, the learner agent can perceive only the actions and the observations of the teacher, and it should be able to extract the teacher's policy that is generating this control. This task can be formulated easily as a problem of learning the parameters of a PPR model, which is equivalent to the problem of learning the parameters of PSRs. Many algorithms for learning PSRs have recently been proposed, some of them [8, 16] assume that the system can be restarted to an initial state, which is the case of systems where the objective is to attend some final states, and others [11, 13, 17] are more related to the systems running infinitely, without reset. The latter algorithms can be classified in two categories: *online* algorithms [11, 13], and *off-line* algorithms [17], depending on the fact that the learning is performed after every action and observation, or after a complete stream.

5.1 Learning PPRs Vs learning FSCs

Contrary to FSCs, PPRs can be learned by using Monte Carlo estimation methods, which are guaranteed to converge to the accurate parameters. This is due to the fact that the PPR core tests can be observed and counted during the learning process, whereas the latent states in FSCs cannot be observed. In fact, to calculate the probability of a core test q at history h, we should only repeat h several times and count the number of times q occurs after h. But to calculate the probability of a state s at history h, even if we repeat h several times, we should use the parameters of the FSC to estimate $Pr(s|h)$ since we cannot observe s but only its *effects*. The FSC parameters cannot be known during the process of learning them. Hence, FSCs learning algorithms like Baum-Welch algorithm [9] start with random parameters, and then alternate between the phase of estimating the belief states b_i at each step i of the training sequence and the phase of optimizing the FSC parameters, until no more optimization can be achieved. The result depends heavily on the initial values, and is subject to local optima.

Input: Train sequences $T_1 = (o^* a_{1,T_1} o_{2,T_1} a_{2,T_1} \dots)$, $\dots T_n = (o^* a_{1,T_n} o_{2,T_n} a_{2,T_n} \dots)$;
Output: A PPR model $\langle Q, Pr(Q|\emptyset), \forall a \in \mathcal{A}, \forall o \in O : m_{oa}, \forall a \in \mathcal{A}, \forall o \in O, \forall q^i \in Q : m_{oaq^i} \rangle$;

```
Construct the matrix P̂ by calculating the estimated probabilities P̂(h,t) = P̂r(t|h) for all the sequences t and h occurring in the training sequences T_i;
Initialize the core tests list Q with the empty test ∅, Pr(∅|∅) = 1;
foreach q ∈ Q, a ∈ A, o ∈ O do
    if the column of oaq in P̂ is linearly dependent on the columns of the core tests q' ∈ Q;
    then
        Find the vector m_oaq by solving the linear system P̂(h,oaq) = P̂(h,Q) m_oaq, ∀h;
        if q = ∅ then m_oa = m_oaq
    else
        Add the test oaq to the list of core tests Q;
        Pr(oaq|∅) = P̂(∅,oaq);
    end
end
```

Algorithm 1: Analytical Discovery and Learning.

5.2 Learning with biased samples

We are interested here in controlled systems where the goal can be specified as a final state, so we assume that the system can be restarted. If the final state is not defined, we can cut the stream at different points, and consider that the system has been restarted at these points. We propose an algorithm (Algorithm 1) close to the ADL (Analytical Discovery and Learning), described in [8]. First, the learner agent collects several samples $T_1 = (o^* a_{1,T_1} o_{2,T_1} a_{2,T_1} \dots)$, $T_2 = (o^* a_{1,T_2} o_{2,T_2} a_{2,T_2} \dots) \dots T_n = (o^* a_{1,T_n} o_{2,T_n} a_{2,T_n} \dots)$ by observing the teacher's behavior. The next step consists in constructing partially the policy matrix $\hat{P}$, with the available

tests and histories. We use two tables *Trial* and *Success*, where $Trial[t^i = o^*a_1o_2a_2\ldots o_ma_m] = \#o^*xo_2x\ldots o_mx$, the number of times where the observations of the test t^i have been perceived in the samples, regardless to the actions of t^i, and $Success[t^i = o^*a_1o_2a_2\ldots o_ma_m] = \#o^*a_1o_2a_2\ldots o_ma_m$, i.e. $Success[t^i]$ counts the times where the actions and the observations of t^i matched with a prefix of a sampled sequence. We can now estimate the probabilities $\hat{P}(t^i|\emptyset)$ with the following estimator ($\emptyset$ is the empty history):

$$\hat{P}(t^i = o^*a_1o_2a_2\ldots o_ma_m|\emptyset) = \prod_{i=1}^{m} \frac{Success[o^*a_1o_2a_2\ldots o_ia_i]}{\sum_{j=0}^{|\mathcal{A}|} Trial[o^*a_1o_2a_2\ldots o_ia^j]}$$

This estimator was proposed in [4] to deal with the problem of bias induced by the system dynamics. It's originally used to learn the parameters of a dynamical system when the exploration policy is not *blind*, i.e. the executed actions depend on the observations and are not totally random. This problem is inherent to PPR parameters learning, because the observations returned by the system depend on the executed actions, unless the system dynamics is totally random.

Based on the first row of the matrix $\hat{P}$ (corresponding to the empty history), we use the Bayes update equation to calculate iteratively the probabilities of the other rows (histories): $\hat{P}(t^i|h_j) = \frac{\hat{P}(t^i|\emptyset)}{\hat{P}(h_j|\emptyset)}$. Given the matrix $\hat{P}$, the next step is to find the core policy tests. The first independent test is always the null test. If q is a core test, then for every action a and every observation o, we check if the column of the extended test oaq depends linearly on the columns of the already discovered core tests, and calculate the weight vector in that case. To do so, we use a Gauss-Jordan elimination, the eigenvalues will correspond then to diagonal elements, and the number of eigenvalues λ_i s.t $\lambda_i \geq \varepsilon$ corresponds approximately to the rank of the matrix, the cutoff ε is used to reduce the number of discovered core tests when we have a small amount of training data. The algorithm stops when all the extensions of all the core tests in Q are linearly dependent on Q. In fact, we can prove that if a test is linearly dependent on other tests, then all its extensions will be also dependent [10]. However, the main drawback of this algorithm is in estimating the matrix $\hat{P}$, since we have to calculate the probability of each possible test, and we can have until $(|O||A|)^t$ different tests for t steps. But in practice, most of these tests are not experienced given that they are generated according to the specific environment dynamics and the teacher's policy rather than uniformly.

5.3 Learning with heterogenous observations

In general, the observations can be different from the teacher to the learner (the actions also can be different, but we do not consider this case in the present paper). We use O^T to indicate the teacher's observations set, and O^L to indicate the learner's observations set. The other important issue is that the observations of the teacher are unknown to the learner, assuming that the learning is completely passive, the teacher and the learner are not sharing their information, and the learner can observe only the teacher's actions and his own observations. However, both of O^T and O^L observations depend on the same parameter, which is the common system state. For simplicity, we will adopt in this section FSC notations instead of PPR, but the result can be generalized to this latter model.

Let $\mathcal{P} = \langle S, A, O^T, O^L, T\rangle$ be the POMDP model of the system (our learning framework does not require specification of a reward function), where $T(s_i, a, \langle o^T, o^L\rangle, s_{i+1})$ is the probability of transiting from the state s_i to the s_{i+1}, after executing the action a, while the teacher perceives o^T and the learner perceives o^L. We consider that the teacher is following a stationary policy, described by an internal finite-state controller $\mathcal{C}^T = \langle S^T, \psi^T, \eta^T\rangle$, $\psi(s^T, a)$ is the probability of choosing action a when the controller is in state s^T, and $\eta^T(s_i^T, a, o^T, s_{i+1}^T)$ is the probability of transiting to state s_{i+1}^T when the controller is the state s_i^T, executes the action a and receives the observation o^T. We will show that even for the learner agent, the teacher is following a stationary policy described by another finite state controller $\mathcal{C}^L$. From the learner's point of view, the teacher is part of the system (under the stationarity assumption), so we can regroup the system dynamics model $\mathcal{P}$ and the teacher's policy $\mathcal{C}^T$ in one uncontrolled POMDP model $\mathcal{P} \times \mathcal{C}^T = \langle S \times S^T, A, O^L, T'\rangle$, where

$$T'(\langle s_i, s_i^T\rangle, a, o^L, \langle s_{i+1}, s_{i+1}^T\rangle) = Pr(\langle s_{i+1}, s_{i+1}^T\rangle, a, o^L | \langle s_i, s_i^T\rangle)$$
$$= \sum_{o^T \in O^T} \psi^T(s_i^T, a) T(s_i, a, \langle o^T, o^L\rangle, s_{i+1}) \eta^T(s_i^T, a, o^T, s_{i+1}^T)$$

From the joint model $\mathcal{P} \times \mathcal{C}^T$, we can extract a model $\mathcal{C}^L = \langle S \times S^T, \psi^L, \eta^L\rangle$, which describes only the policy of the teacher as seen by the learner, and makes abstraction of the environment dynamics.

$$\psi^L(\langle s_i, s_i^T\rangle, a) = \psi^T(s_i^T, a)$$

and

$$\eta^L(\langle s_i, s_i^T\rangle, a, o^L, \langle s_{i+1}, s_{i+1}^T\rangle) = \frac{T'(\langle s_i, s_i^T\rangle, a, o^L, \langle s_{i+1}, s_{i+1}^T\rangle)}{\sum_{(s_{i+1}, s_{i+1}^T)} T'(\langle s_i, s_i^T\rangle, a, o^L, \langle s_{i+1}, s_{i+1}^T\rangle)}$$

The controller $\mathcal{C}^L$ contains $|S| \times |S^T|$ states, we can use any learning algorithm to find the parameters of the policy described by C^L, which represents the teacher's policy from the learner's point of view. The learned policy contains at most $|S| \times |S^T|$ states (or core policy tests), the divergence between the learned policy and the actual policy depends on the divergence between the two observation models.

6 Empirical Analysis

6.1 The simulated environment

We used a simulated environment to test our learning framework on the two problems defined in Section 2. The

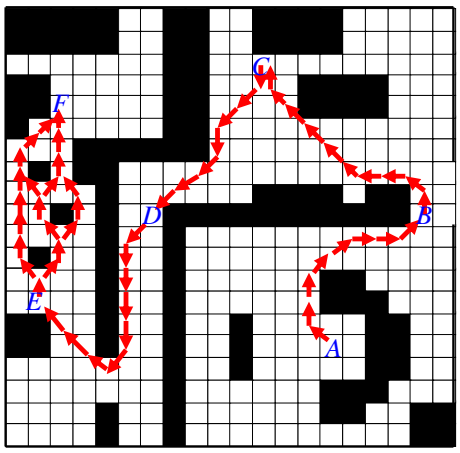

Figure 1. Simulated indoor environment of the robotic wheelchair.

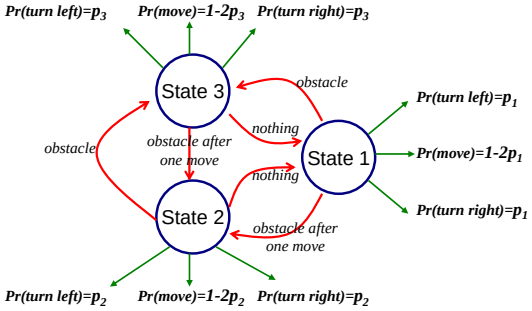

Figure 2. The finite state controller used to simulate the wheelchair user's behavior. In a normal behavior, we have $p_1 = 1/10, p_2 = 9/20, p_3 = 1/2$. In an abnormal behavior, we have $p_1 = 1/3, p_2 = 1/3, p_3 = 1/2$, the user tends to move closer to obstacles.

environment is represented by a 20×20 grid (Figure 1), the system state is the position of the wheelchair on the grid (275 free positions) and its direction (8 orientations), the system contains 2200 different states. We consider three actions: move forward, turn left, and turn right. The turn actions result on a rotation of $\pi/4$. For the obstacle avoidance problem, we consider only three observations: no obstacle, obstacle after one move, and obstacle ahead. These observations are the same for both the teacher and the learner. For the pathfinding problem, we consider that the wheelchair observes presence/absence of obstacles in the 8 adjacent positions, we have then 256 observations, whereas the teacher observes the exact state. The observations are aliased, but they are deterministic, and the actions have stochastic effects: they succeed with probability 0.9, and have no effect

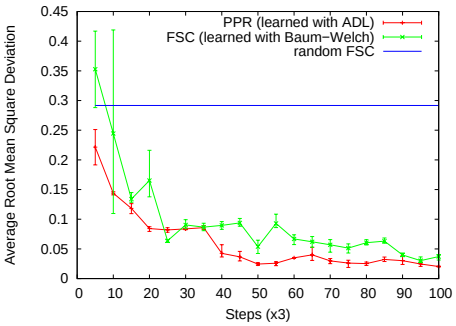

Figure 3. The average RMSD on predicting the user's normal actions in the obstacle avoidance problem, as a function of the training steps.

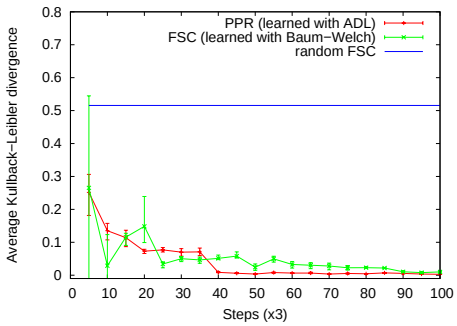

Figure 4. The average KL-divergence between the predictions of the learned model and those of the accurate model.

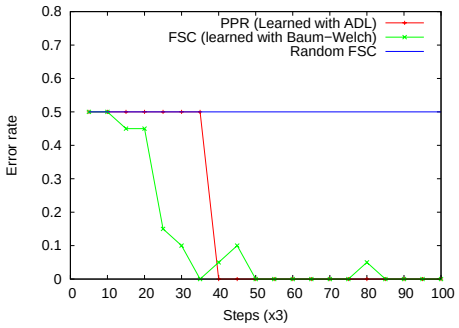

Figure 5. The average error rate in recognizing the user's anomalous behavior.

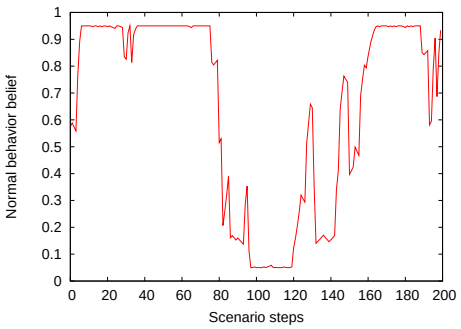

Figure 6. Detecting an abnormal behavior sequence within a normal behavior sequence, by using the learned PPR model. The anomalous behavior starts at step 75 and ends at step 125.

with probability 0.1.

6.2 The obstacle avoidance problem

We have simulated the user's behavior by means of the Finite-State Controller described in Figure 2. In normal situation (*state 1*), the user moves forward with probability 0.8, and turn left or right, with probability 0.2. When an obstacle one step away is observed (*state 2*), the user tries to avoid it by turning left or right with probability 0.9, but still there is a small probability, 0.1, that the user moves forward, because he could be looking for something on the obstacle (the obstacle is his goal). Last, if the obstacle is ahead (*state 3*), the user would turn left or right with probability 1, and would never move in the direction of the obstacle. In abnormal behavior, we suppose that the user chooses uniformly her/his actions in states 1 and 2, but she/he still avoids obstacles in state 3 by turning left or right with probability 1. If we consider that the user can hit obstacles, the problem becomes no longer interesting since abnormal behavior could be detected each time the user moves towards an obstacle.

To learn the user's personal normal behavior, we randomly generate the initial position of the wheelchair on the grid, then we use the parameters of the environment (defined as a POMDP), and the parameters of the policy (defined as an FSC), to generate one action and one observation at each time step, we repeat this process L times. We also consider a reset point after every three actions.

Given that the rank of the estimated matrix $\hat{P}$ is very sensitive to the smallest variations in the probabilities estimation, the number of discovered core tests can be very high for the smallest values of L. So, to force the PPR learning algorithm to keep a few core tests, the cutoff threshold ε was set to 0.9. PPR learning algorithm does not require any prior parameter besides ε.

We used the same training data to learn the parameters

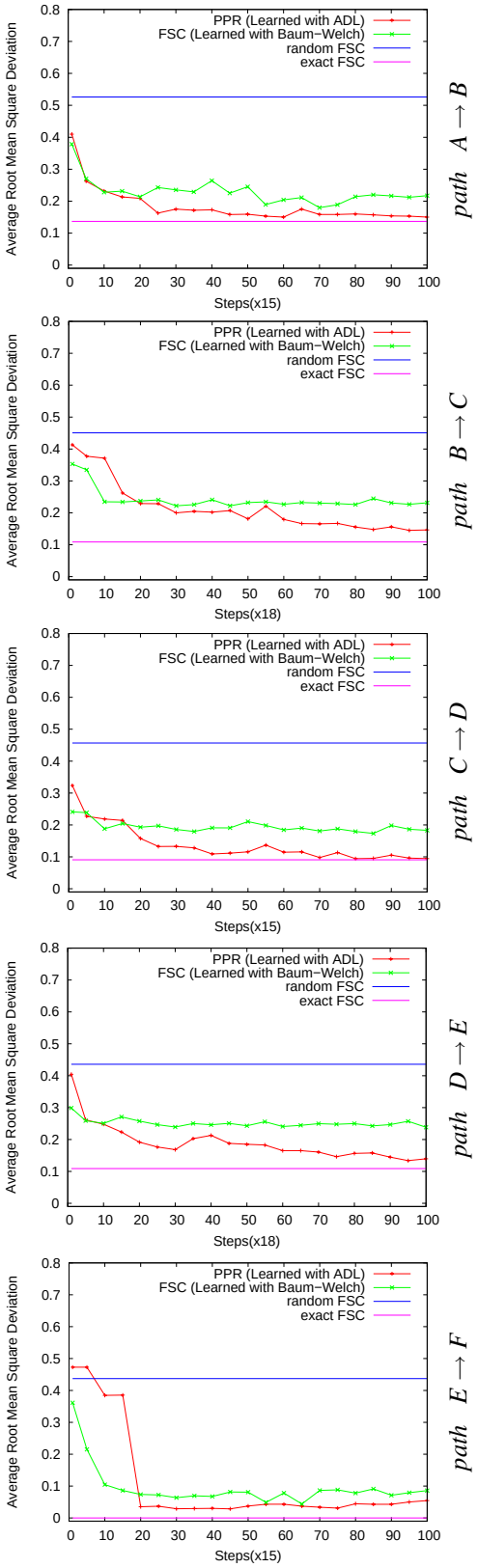

Figure 7. The RMSD error on predicting the user's actions in the pathfinding problem.

of the corresponding finite controller with the Baum-Welch algorithm. Given a prior model, the Baum-Welch algorithm calculates beliefs on the hidden states after each step, and uses this expectation as a basis to find new parameters which better explain the observed actions. This operation is repeated until a fixed point is reached. In general, the fixed point corresponds to a local maximum. As a prior model for the Baum-Welch algorithm, we considered a 3-state controller with random transition and emission probabilities.

To test the accuracy of our learned policies, we simulate a trajectory with length $L = 100$ steps, and at each step i, we calculate for each next action a the probability $\hat{Pr}(a|h_i)$ that the user will execute the action a according to the learned policy model, and compare it to $Pr(a|h_i)$, the true probability given by the finite-state controller of Figure 2. We calculate then the *Root Mean Square Deviation (RMSD)*:

$$RMSD = \sqrt{\frac{1}{L \times |\mathcal{A}|} \sum_{i=1}^{L} \sum_{a \in \mathcal{A}} (\hat{Pr}(a|h_i) - Pr(a|h_i))^2}$$

Figure 3 shows the different values of RMSD, averaged over 20 trials, as function of the sampled data size. As expected, an accurate PPR model can be learned after only 40 episodes of 3 steps, with an average error of 0.04. The error continues to decrease slowly as we use more training steps. The FSC parameters are also learned quickly, but we must wait until step 100 to get an error of 0.04. Figure 4 shows the Kullback-Leibler (KL) divergence between the learned policies and the true policy, we notice that PPR distributions are slightly more accurate than FSC distributions.

The learned policies are used to distinguish the normal behavior state from the anomalous behavior state. We define a small POMDP containing these two states, the actions probabilities in every state are defined by the corresponding policy, and the belief state is updated with Bayes function (Equation 1). The initial belief state is set to $(1/2, 1/2)$, and the belief state at a given step indicates the most likely type of behavior that has been followed during the last a few steps. We sampled 10 normal scenarios and 10 abnormal scenarios, and reported in Figure 5 the error rate in classifying scenarios. Both PPR and FSC were able to properly classify the behavior. We observed that for the smallest training samples, PPR misclassifies the anomalous behaviors because of a missing core policy test that does not occur in the smallest samples and its probability was automatically set to 0. Finally, Figure 6 shows how the deviations from a normal behavior to an abnormal behavior could be quickly detected by using the learned PPR policy. The user starts an anomalous behavior after step 75, and we can detect it just a few steps later. After step 125, the user comebacks to her/his normal motion, and the belief on the normal state of the user increases gradually and becomes almost equal to 1 after step 160. Notice that this deviation is not reflected in hitting obstacles, but just in the probabilities of choosing the same actions (Figure 2).

6.3 The pathfinding problem

This problem is much more complex than the first one, because we have now a larger set of observations, the trajectory is longer, and more importantly, the user's observations do not match the wheelchair's observations. The user has a perfect observation of the state, whereas the wheelchair can observe only the obstacles surrounding it. To make the problem more challenging, we consider that in a given position, the wheelchair will receive the same observation whatever its direction, so it can never realize if a turn action has succeeded or not because the observation will be the same. The user's policy is described in Figure 2 in terms of (state, action), the controller corresponding to this policy in terms of learner's local observations for each part of the itinerary contains 20 states at least, most of them are aliased because the direction was not included in the observations. There are 5 reset points in the global path, each one corresponds to a subgoal (usual destinations of the user like doors, tables ...). Recall that the goal is not to find the shortest paths, but to learn the user's personal preferences. We repeatedly generated the trajectory from the start to the arrival positions (where the first actions are often turn actions), and used the stream of (local observations, actions) to learn approximately the underlying user's policy, even though the user's behavior can never be reproduced exactly in this case. We used a random model to initialize Baum-Welch algorithm, but the number of states in the accurate model was given to the algorithm. Figure 7 indicates the results on the RMSD, averaged over 20 trajectories. We notice that in this problem too, PPR is learned with fewer training data compared to FSC, and it converges quickly to a minimal error around 0.10. This error of 0.10, indicated in Figure 7 by the error of the exact FSC, corresponds to the divergence between the true policy of the teacher, and this policy as seen by the learner, because of the difference between the two observation models (see Section 5.4).

For the last path $(E \rightarrow F)$, we considered that both of the teacher and the learner use the same observation model. In this case, the observation corresponds to the presence or absence of an obstacle ahead the wheelchair, and its direction, we have then 2×8 observations. Based on these partial observations, the teacher can follow different paths to move from E to F. Here again, we notice that the divergence of PPR learning algorithm decreases faster than FSC learning algorithm. Contrary to the previous paths, the final error (0.05) in this path is closer to 0 because both of the teacher and the user have the same model of observation. The error of PPR falls down quickly after 15 repetitions because of a second core test $(\langle \text{obstacle ahead, direction north} \rangle, \langle \text{turn left} \rangle)$ that is recog-

nized as independent, with $\varepsilon = 0.1$, only after the 15^{th} repetition.

7 Conclusion

Learning by imitation is very useful in situations where the task of the robot involves interaction with humans. This method can significantly accelerate the learning process, compared to fully-autonomous methods. Moreover, the imitative learning can personalize the robot policies for every new user and environment. Most of the previous work on in this topic is concerned with deterministic, and/or fully observable environments. In this paper, we presented a new imitation learning method which is based on Predictive Policy Representations (PPRs). The advantage of PPRs, compared to Finite-State Controllers, is that they are state-free, and based completely on observable quantities (actions and observations). We tested this model on two problems related to the assistive robotics. The first one is to learn a user's behavior regarding obstacles avoidance, and to recognize any deviation from this behavior in order to alert the user, or to switch to an automated control. The second problem is to imitate the user's motion policy to reach a given goal in an indoor environment. In both problems, PPR outperform the Baum-Welch algorithm for FSC, and the corresponding policies could be learned after a few repetitions, without any prior knowledge on the environment, the target task, or the user's preferences. Future extension of this work will go in two directions: The first one is to bound the divergence between the teacher's policy and the learner's policy when they use different models of transition and observation. The other extension of this work is to consider using Predictive Policy Representations in the problem of reinforcement learning in partially observable domains.

8 Acknowledgment

The author would like to thank Joelle Pineau and Brahim Chaib-draa for their helpful discussions and suggestions concerning the topic of this paper.

References

[1] C. G. Atkeson and S. Schaal. Robot learning from demonstration. In *Proceedings of the Fourteenth International Conference on Machine Learning*, 1997.

[2] A. Billard. Imitation: A review. In *The Handbook of Brain Theory and Neural Network. 2nd Edition. Michael A. Arbib (editor), pages 566-569. MIT Press*, 2002.

[3] A. Boularias and B. Chaib-draa. Planning in Decentralized POMDPs with Predictive Policy Representations. In *Proceedings of ICAPS Multiagent Planning Workshop*, 2008.

[4] M. Bowling, P. McCracken, M. James, J. Neufeld, and D. Wilkinson. Learning predictive state representations using non-blind policies. In *Proceedings of the Twenty-Third International Conference on Machine Learning (ICML)*, 2006.

[5] E. Hansen. Solving pomdps by searching in policy space. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, 1998.

[6] J. Hoey and J. Little. Value-directed human behavior analysis from video using partially observable markov decision processes. *IEEE transactions on pattern analysis and machine intelligence, vol*, 29, no. 7, 2007.

[7] M. James. *Using Predictions for Planning and Modeling in Stochastic Environments*. PhD thesis, The University of Michigan, 2005.

[8] M. R. James and S. Singh. Learning and discovery of predictive state representations in dynamical systems with reset. In *Proceedings of the twenty-first international conference on Machine learning*, 2004.

[9] S. Koenig and R. Simmons. Unsupervised learning of probabilistic models for robot navigation. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 1996.

[10] M. Littman, R. S. Sutton, and S. Singh. Predictive representations of state. In *Advances in Neural Information Processing Systems 14 (NIPS)*, 2002.

[11] P. McCracken and M. Bowling. Online learning of predictive state representations. In *Advances in Neural Information Processing Systems 18 (NIPS)*, 2006.

[12] S. Singh, M. James, and M. Rudary. Predictive state representations: A new theory for modeling dynamical systems. In *Uncertainty in Artificial Intelligence: Proceedings of the Twentieth Conference (UAI)*, 2004.

[13] S. Singh, M. L. Littman, N. K. Jong, D. Pardoe, and P. Stone. Learning predictive state representations. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

[14] R. D. Smallwood and E. J. Sondik. The optimal control of partially observable markov decision processes over a finite horizon. *Operations Research*, 21(5):1557–1566, 1971.

[15] D. Verma and R. P. N. Rao. Goal-based imitation as probabilistic inference over graphical models. In *Advances in Neural Information Processing Systems 18 (NIPS)*, 2006.

[16] E. Wiewiora. Learning predictive representations from a history. In *Proceedings of the 22nd international conference on Machine learning*, 2005.

[17] B. Wolfe, M. R. James, and S. Singh. Learning predictive state representations in dynamical systems without reset. In *Proceedings of the 22nd international conference on Machine learning*, 2005.