Learning option MDPs from small data

Ashkan Zehfroosh,[1] Herbert G. Tanner,[1] and Jeffrey Heinz[2]

***Abstract*— Learning from small data is a challenge that presents itself in applications of human-robot interaction (HRI) in the context of pediatric rehabilitation. Discrete models of computation such as an Markov decision process (MDP) can be used to capture the dynamics of HRI, but the parameters of those models are usually unknown and (human) subject dependent. This paper combines an abstraction method for MDPs, with a parameter estimation method originally developed for natural language processing, designed specifically to operate on small data. The combination expedites learning from small data and offers more accurate models that lend themselves to more effective decision-making. Numerical evidence in support of the approach is offered in a comparative study on a small grid-world example.**

I. Introduction

Autonomous agents are often called to make decisions in partially known environments, based on incomplete information. A standard Artificial Intelligence (AI) approach to this problem is the application of reinforcement learning [1]. Learning algorithms in this framework involve a *learning stage*; during that phase, there is typically not enough information to theoretically guarantee optimal decision-making. While at this stage, the emphasis is on fast completion rather than best performance. Some HRI applications, however, particularly in the context of early pediatric rehabilitation, present instances of the learning problem where (i) on one hand performance is paramount because interaction time is brief and precious, and on the other, (ii) precisely because of limited rehabilitation dosage and small data sets, the learner may not converge and computed policies may be far from optimal [2]. The focus of this paper therefore is to accelerate the *learning stage* when learning from *small data*.

Markovian, discrete mathematical models have been used in many applications to describe the dynamics of interaction between robots and humans. The most common Markovian model in human intent prediction is a partially observable Markov decision process (POMDP) which treats human *action* as an observable variable, and encodes human *intent* as a hidden state [3]. The POMDP model is computationally taxing (NEXP-complete), and even infinite horizon versions of such problems are known to be undecidable under different optimality criteria [4]. A less computationally demanding model is a mixed observability Markov decision process (MOMDP), which treats components of the state, as opposed to the whole state vector, as being unobservable. Such a model has been used for intention-aware robot motion planning [5], where states of the world are considered observable, and only a finite number of human intentions are deemed unobservable. For each one of the human intentions, the model is essentially fully known. Optimal policies for robots are then computed by forming a *belief* over human intention. Crucial underlying assumptions in MOMDP approaches is that (i) human behavior is always rational, which supports the construction of a meaningful prior model, and (ii) a large amount of observation data are available for learning.

In contrast to POMDP and MOMDP models, an MDP is relatively simple, and has also been utilized in HRI applications, although not to the extend of the aforementioned models. In one example of use of an MDP in an aerospace HRI application, [6], the system is initiated with some inaccurate prior, and attempts to refine and update its MDP model through observations. Because of the size of the model, however, the update process is restricted to select out of a finite set of parameters. Still, given that an MDP has a relatively smaller number of tunable parameters compared to more sophisticated POMDP and MOMDP models, it presents itself as an attractive (from a computational expediency standpoint) choice when small data is available.

The application that motivates this work is in the space of early pediatric motor rehabilitation. In this context, young children (infants) with motor disabilities are socially interacting with robots in play activities. The goal of this play-based interaction is to encourage physical activity on behalf of the special needs children. This is because early development is tied to learning, and learning depends on exploration [7]–[9]. The more the infants explore their environment, as well as test and hone their own locomotion skills, the faster they can develop, not only motor-wise, but also cognitively and socially. To apply HRI in this paradigm, some scenarios are designed by considering infants' abilities and interests based on their age and level of impairment [10]–[12]. The objective of the described HRI problem is to design an automated decision-algorithm for the robot to keep the infant engaged in physical activity. This problem is arguably more challenging than other HRI applications, since the automation system is called to effectively work in perhaps more complex and dynamic environments compared to earlier HRI pediatric studies [13]. In addition, there is little prior information about the infant's preferences, the interaction time in clinical studies for infants hardly provides sufficient data for machine learning algorithms. Models and methods that can handle sparsity in training data are therefore expected to perform better than alternatives, and this is why

[1]Ashkan Zehfroosh, and Bert Tanner are with the Department of Mechanical Engineering, University of Delaware. {ashkanz, btanner}@udel.edu

[2]Jeff Heinz is with the Department of Linguistics and institute for advanced computational science, Stony Brook University. jeffrey.heinz@stonybrook.edu

This work was supported by NIH under grant # R01HD87133.

mathematical abstractions of this type of HRI in the form of some MDP appear to be appropriate here [2].

When the dynamics of HRI are to be described by an MDP, the designer is faced with the problem of populating the model with the appropriate values of its parameters. For an MDP, arguably the most critical set of parameters is the collection of transition probabilities. Often, those transition probabilities are unknown —they have to be learned. Then a parameter estimation process is initiated, and a typical method for doing that can be maximum likelihood (ML), which can come with formal guarantees of convergence [14]. Depending on the application, however, the training data size required to obtain convergence in those terms may be unreasonably big. When ML is used with data obtained from a small amount of observations, the approximation of the probabilities obtained is usually crude and unacceptably inaccurate [2].

The problem of learning the parameters of an MDP model based on sparse data can been approached from four main directions. One is closely tied to probably asymptotically correct (PAC) analysis; questions considered important here are how to place a bound on the minimal amount of data required for different learning algorithms in order to guarantee some metric of compatibility with true observations at a given level of confidence [15]. Another approach is focused on convergence *speed*: what modifications can be introduced to a learning algorithm to make it learn faster from small data sets [16], [17]. A third viewpoint is linked to abstraction, namely finding a model representation that is just coarse enough to sufficiently reduce the size of the search space for the parameters that need to be learned [18]. Finally, there are efforts that essentially try some type of "interpolation" between the sparse data to improve learning objectives when the training set is small [2]. If one is willing to forego the certainty of provable theoretical convergence and trade it off for increased learning performance on small data sets, they can replace ML with *smoothing*. Smoothing [19] is a technique that has traditionally been applied in the application space of natural language processing (NLP), and is designed to interpolate over sparse data sets. Motivated by its success in NLP, smoothing was proposed and formulated to learn the unknown transition probabilities in a conventional MDP, from small sets of observations [2]. When applied to learning MDP transition probabilities, it has demonstrated increased performance compared to ML [2]. The present paper combines elements from the last two types of the approaches, namely using abstract model representations that facilitate learning by reducing the size of the search space, together with learning techniques specifically developed for operation on small data sets.

The computationally expedient abstract representations considered here involve *options* [18]. Options essentially express a way discretizing and making finite the search space of policies. And while all finite-horizon policies policies based on a finite set of primitive actions will theoretically form a finite set, options are judiciously chosen (sub-)policies that are likely to be effective over a relatively wide range of states. It turns out that the resulting abstraction has the potential to reduce drastically the amount of data required for learning [18].

The main innovation in this paper, therefore, is the combination of smoothing with a partially known Semi-Markov decision process (SMDP), and the use of this new construction for the development of a learning framework that is adapted to learning from small data. Once the small body of data is processed, and the SMDP parameters are estimated, a standard decision-making algorithm is executed. The policies resulting from this process are numerically tested and assessed on a small grid-world example.

The rest of the paper is organized in the following sequence. Section II provides some technical background. Section III describes the problem that this paper addresses in technical terms, and then Section IV presents the solution developed in the context of options and smoothing. Section V contains the numerical results on the grid-world example, and Section VI finally closes the paper with some conclusions.

II. Technical Preliminaries

A finite MDP M is a tuple $\{S, A_s, R_s, P^a_{ss'}, \gamma\}$ where S is finite set of states; A_s is finite set of available actions in each state s; R_s is the *reward* assigned to each state s; $P^a_{ss'}$ is the *probability of transition* from state s to state s' by performing action a; and γ is a *discount factor*. A *policy* π is a mapping from states to probabilities of performing some action in A_s, i.e. $\pi : S \times A_s \rightarrow [0, 1]$. A policy is said to be *optimal* if it maximizes the expected sum of discounted reward, $\mathbb{E}\{\sum_{t=0}^{\infty} \gamma^t R_{s_t}\}$, in which the discount factor γ reflects the preference of immediate rewards over future ones, and the variable t is used to denote the current discrete time step, while s_t and a_t are the state, and action, taken at time t, respectively. The *value* of state s under policy π is defined as

$$V^\pi(s) = \sum_{a \in A_s} \pi(s, a) \left[R_s + \gamma \sum_{s'} P^a_{ss'} V^\pi(s') \right]$$

An optimal policy maximizes the value on all states, and the latter is then denoted V^*_s.

Options [18] are subsets of policies (sub-policies) that are only applicable to certain groups of states. In other words, a decision-making algorithm can only choose among the particular set of sub-policies when actions need to be executed within this group of states. Formally, an option o is a tuple $\{I, \pi, \beta\}$, where $I \subseteq S$ is group of states where the particular option can be initiated, π is the policy that the option executes, and $\beta : S \rightarrow [0, 1]$ is a function that for a given state gives the probability that the option *terminates* at that state. For example, if an option labeled turn-on-light is available, the policy π that is executed for that option is likely a sequence of actions that include reaching, touching, and flipping some light switch; in this particular case, the termination probability β is 1 in all states where the light is on, and I contains all states where the light is off. The *option reward function* R^o_s is the expected value of the accumulated rewards by executing option o in state s until termination,

and with n representing the random number of steps after which option o terminates, it is expressed as

$$R_s^o = \mathbb{E}\{r_{t+1} + \gamma\, r_{t+2} + \ldots + \gamma^{n-1}\, r_{t+n}\}$$

The *option transition probabilities* are denoted $P_{ss'}^o$. They express the probability that the system terminates option o at state s' after initiating it at state s. If $P^o(s', n)$ denotes the probability that option o terminates at s' after n steps, and m is the maximum number of steps allowed, the transition probabilities can be evaluated as

$$P_{ss'}^o = \sum_{n=1}^{m} P^o(s', n)\, \gamma^n$$

The set of all options available at state s is denoted O_s, and the collection of all options is $O = \bigcup_{s \in S} O_s$. Then a *policy over options* can be defined as a map $\mu : O \times S \to [0, 1]$. With these definitions at hand, an MDP with set of options O becomes a SMDP, which is a direct extension of the classical MDP [18]. In this extension, o plays the role of a, μ that of π, O_s replaces A_s, and R_s^o takes the place of R_s.

The value iteration update rule for an SMDP is given as

$$V_s \leftarrow \max_{o \in O_s} R_s^o + \sum_{s' \in S} P_{ss'}^o V_{s'}$$

and converges to the optimal option value for state s. It is known that if the optimal policy can actually be constructed by the set of available options, then the optimal option value in the SMDP is the same as the optimal value V_s^* in the associated MDP [18].

Quite common in early applications of NLP, *smoothing* was shown to interpolate much more effectively when presented with sparse data compared to other contemporary methods [19]. Within the context of NLP, and in its Kneser-Ney variant [19], smoothing would operate on finite contiguous sequences of letters, or symbols. The learning algorithm keeps a record of the frequency of appearance of those sequences, called *factors*, in the data. The main difference compared to a typical ML estimation algorithm, is that smoothing —acknowledging the fact that the training sample may be too small to explicitly express all possibilities— assigns nonzero probabilities even to factors that have not been encountered in the data.

The goal of this paper is to formulate a learning framework that is better suited for learning the parameters (transition probabilities) of MDP models from small data, so as to boost the performance of subsequent decision making. *Small data*, in this context, is formally understood in terms of the VC dimension. In learning theory [20], a sample size for learning is considered *small* when the ratio of the cardinality of the set of training examples n, to the VC dimension of functions of the learning machine d, is smaller than some (arbitrary) constant, say $n/d < 20$.

III. Problem Statement

Consider an instance of HRI, where the dynamics of interaction between human and machine is modeled by an MDP M. Transitions in M express the human's reaction to the robot's actions. It is assumed that the responses of the human to the robot actions are observable, and the objective is to design a policy to dictate the robot's actions. However, both the transition probabilities in M are not known confidently a priori.

In HRI applications where automated decision-making is informed by partially known Markovian models, learning modules can be incorporated to complete the missing information. During that initial learning phase, the learner observes instances of HRI and based on these observations, estimates the unknown parameters of the model. In many instances, however, the available observations are scarce. The central question for the paper, therefore, is how to construct that learning algorithm so that it is capable to provide better approximations of the model parameters compared to a typical parameter estimation method, so as to increase the performance the decision-making algorithm that subsequently uses the model.

IV. Technical approach

The hypothesis in this paper is that if the states in M can be grouped in such a way so that only those particular type of policies (the options) would make sense in each group, the resulting abstraction makes the learning problem for the transition probabilities significantly easier. Due to the sparsity of the training data available, the strategy therefore is to combine a computationally expedient Markovian model (an SMDP) together with a learner designed to cope with small data sets (smoothing). Once M is abstracted into an SMDP $\mathcal{M}$, the unknown parameters take the form of option transition probabilities, and option reward functions, $P_{ss'}^o$ and R_s^o, respectively. The discount factor is taken as $\gamma = 1$.

The abstraction of M into $\mathcal{M}$ is performed in a straightforward way as follows. Transitions (by options) in $\mathcal{M}$ are considered as pairs of states, or bi-gram elements (sequences of symbols of length two) consisted of the the state at which an option was initiated followed by the state at which it was terminated. For example, when an option takes M from s_{i-1} to s_i, for example, with $s_{i-1}, s_i \in S$, the subsequence will be of the form $s_{i-1}s_i$. The frequency of occurrence in the data of a transition from s' to s'' by option o is denoted $c_o(s's'')$. The frequency of occurrence of a transition from s' to s'' by option o *terminated after exactly* n steps is denoted $c_o^{(n)}(s's'')$.

A maximum number m of steps taken in an option is set, to prevent the system from being trapped executing an ineffective option. As a result, the model is semi-Markovian: the termination of an option depends not only on the previous state in M that was visited, but also on m. In the same spirit, updates on the model are made only after termination of options; not while an option is being executed. Consequently, the intra-option model learning method [18] cannot be used here.

The learning algorithm now keeps a record of the frequency of appearance in the data of all different $s_{i-1}s_i$ sequences, corresponding to each option being executed. Let the cumulative reward of option o, when initiated at state s

and terminated after n steps be denoted $R_s^o(n)$; this belongs in the observables provided to the learning module. The average cumulative reward obtained by executing the option N times from state s, is updated through an incremental learning rule as

$$R_s^o = \frac{N-1}{N} R_s^o + \frac{1}{N} R_s^o(n) \tag{1}$$

Now take a discount constant parameter $\zeta \in (0,1)$, denote $|\cdot|$ the cardinality of a set, and define a normalizing constant

$$\lambda_{s_{i-1}s_i}(o) \triangleq \frac{\zeta \, |\{s' : 0 < c_o(s_{i-1}s')\}|}{\sum_{s'} c_o(s_{i-1}s')} \times \frac{\sum_{n=1}^{m} \gamma^n c_o^{(n)}(s_{i-1}s_i)}{c_o(s_{i-1}s_i)}$$

The Kneser-Ney smoothing process approximates the probability of reaching state s_i by executing option o as

$$P^o_{s_i s_{i-1}} = \frac{\lambda_{s_{i-1}s_i}(o) \, \big|\{s' : 0 < c_o(s's_i)\}\big|}{\big|\{s's'' : 0 < c_o(s's'')\}\big|} + \frac{\max\{c_o(s_{i-1}s_i) - \zeta, 0\} \, \sum_{n=1}^{m} \gamma^n c_o^{(n)}(s_{i-1}s_i)}{c_o(s_{i-1}s_i) \sum_{s'} c_o(s_{i-1}\, s')} \tag{2}$$

It should be noted that (2) assigns nonzero probabilities to *all* possible pairs $s_{i-1}s_i$ associated with option o —-that is, even those that have not been observed. The magnitude of those probabilities is determined in part by the discount parameter. Combining the described updating process with ε-greedy exploration approach [21], the whole learning and decision-making process can be described in Algorithm 1.

Input: set of states S, set of options O, prior option transition probabilities $P^o_{s_i s_{i-1}}$, reward function R_s, prior option reward function R_s^o, coefficient c, convergence threshold ϵ.
Set: $N(s) = 0$, $c_o(ss') = 0$, $c_o^{(n)}(ss') = 0$ $\varepsilon_t(s) = 0$, $\forall s, s' \in S$, $\forall o \in O$; current state s_t.
While $\max_{s' \in S} |P^o_{ss'} - p^o_{ss'}| < \epsilon$
Do
1) $N(s_t) \coloneqq N(s_t) + 1$
2) $\varepsilon_t(s_t) \coloneqq c/N(s_t)$
3) Value-iteration (current MDP) $\rightarrow V^*$:
   - with probability $1 - \varepsilon_t(s_t)$: $o_t \coloneqq \arg\max_{o \in O_{s_t}} [R^o_{s_t} + \gamma \sum_{s'} P^o_{s_t s'} V^*_{s'}]$
   - with probability $\varepsilon_t(s_t)$: $o_t \coloneqq \mathsf{Random}(O_{s_t})$
4) Execute the option till termination after n steps and observe new state s_n
5) $c_{o_t}(s_n, s_t) \coloneqq c_{o_t}(s_n, s_t) + 1$
6) $c^{(n)}_{o_t}(s_n, s_t) \coloneqq c^{(n)}_{o_t}(s_n, s_t) + 1$
7) $p^o_{s_t s_n} := P^o_{s_t s_n}$
8) Update system parameters:
   - $P^o_{s_t s_n} \leftarrow$ (1)
   - $R^o_{s_t} \leftarrow$ (2)
9) $s_t \leftarrow s_n$
End

Algorithm 1: Learning and decision-making loop.

V. Validation

In this section, Algorithm 1 is tested on a small grid-world (see Fig. 1). The grid-world is designed in a way that the states can be divided into two groups or partitions. The first (left) partition includes states $\{1, 2, 7, 8, 11, 12, 13\}$, while the second (right) partition extends through states $\{3, 4, 5, 9, 10, 14, 15\}$. To all partition states, a reward of -1 is assigned. In contrast, states $\{6, 16\}$ are goal states and are assigned rewards 1.5 and 1 respectively.

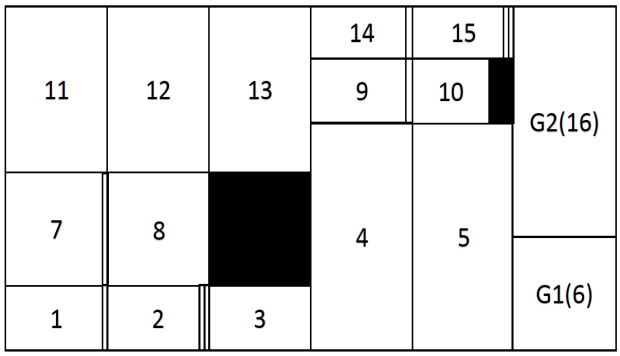

Fig. 1. The grid-world with two partitions.

With the exception of the goal states, the system always has four primitive actions available: down (d), left (l), up (u), and right (r). Nominally, the execution of each one of these primitive actions will successfully transition the system along the intended direction with probability 0.7; the system will stay in the same cell with probability 0.1, and may make a random move with probability 0.2. In cells with right boundary marked with double line, the probabilities for those events are 0.4, 0.4, and 0.2, respectively. For cells with triple lines on their right boundary, the probabilities are 0.1, 0.7, and 0.2, respectively. The prior for the probability of moving in the intended direction is 1.

Only two *options* are available in each partition to escape: either through cell 13 or 2 in the left partition and 5 or 15 for the right one. The maximum number of steps (read: execution of primitive actions) allowed for each option is set at $m = 10$. Fig. 2 illustrates one of these options in the left partition.

The option set is chosen here in a way that the true optimal policy can indeed be synthesized by the available options. As a consequence, the optimal option value should the same as the conventional optimal value for all states. Let V^* be the vector of optimal values in all states, and V^l be the optimal value resulting from the model produced by the learning algorithm. Those values are computed using value iteration [22]. The difference between these two vectors serves as a metric of performance, quantifying in some sense the decision-making error as a result of a coarse model. Figure 3 depicts the Euclidean norm of the difference between the value vectors, $\|V^* - V^l\|_2$ as a function of observations, for different combinations of models and learning methods: (a) utilization of an MDP as the underlying

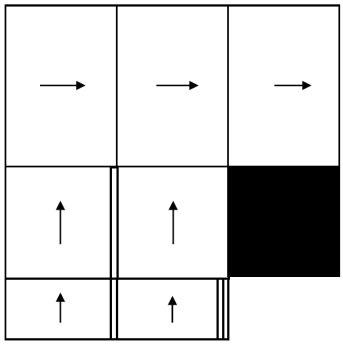

Fig. 2. The policy for the option that takes the system out of the left partition and into the right through cell 13.

model and estimation of the transition probabilities using ML; (b) utilization of the same MDP as the underlying model, and estimation of the transition probabilities using smoothing; (c) use of an SMDP as the underlying model, and estimation of its transition probabilities and reward functions through ML; and finally (d) use of the SMDP and estimation of the transition probabilities and reward functions via smoothing. When an ML estimation method approximates the option reward function, in the form of the average cumulative reward obtained by executing the option N times from state s, through the same incremental learning rule (1). However, the option transition probabilities are updated, using the indicator function $\delta_{ss'}$, which is equal to 1 if $s = s'$ and is 0 otherwise, in the following way:

$$P^o_{ss''} = \frac{N-1}{N} P^o_{ss''} + \frac{1}{N} \gamma^n \delta_{s's''} \tag{3}$$

The update step 8 in Algorithm 1 is performed using (1) and (3), instead. The magnitude of the norm depicted in the figure is the average over twenty different instantiations of the particular learning problem having been presented with the same number of observations.

After around 4000 steps, all methods perform comparably, and the discrepancy decreases even more with more observations, as expected. The utilization of smoothing MDP results in noticeably better performance compared to ML with small data. Interestingly, however, the difference in performance is not that stark when smoothing is applied on the SMDP. Smoothing does appear to perform better, but the difference diminishes quickly with the number of observations. The difference is more significant at very small number of observations. One of the reasons is that that the SMDP model itself is capable of absorbing some of the uncertainty related to small data sets, and another is that in both cases, the reward functions are still estimated identically through (1).

While an MDP model for the grid-world of Fig. 1 involves 228 transitions, the associated SMDP model includes 308 transitions, and requires the estimation of an additional 16 option reward functions. Thus, at first sight, the computational benefits are not obvious; the abstraction appears to be "finer" than the concrete model. The secret lies in the data: because of the grouping and in conjunction with the small data set, most of the transitions in SMDP are never encountered in the training set, in contrast to the case of the MDP. As a result, for the same number of observations, Algorithm 1 focuses more on a subset of the transitions in SMDP while it "spreads" itself over a much larger set of transitions when executed on an MDP. In fact out of the 308 SMDP transition probabilities, only about 77 were updated by Algorithm 1.

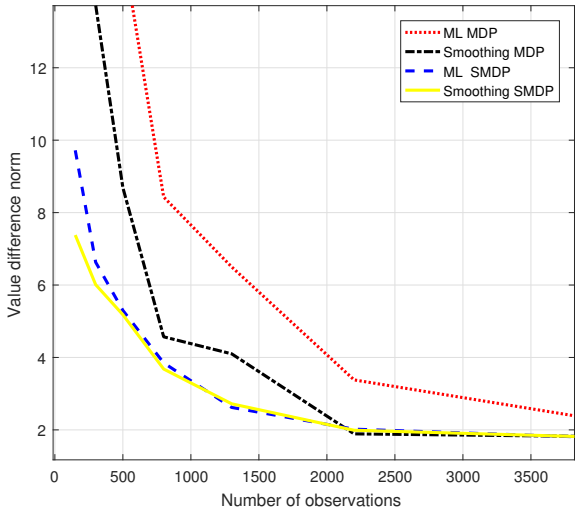

Fig. 3. The evolution of the difference between value vectors as a function of observations.

Due to the inherent variability introduced by small training data sets, it is informative to look at the variance of the metric depicted in Fig. 3. Table I is revealing: using options with SMDP models appears to significantly reduce the *variability in the performance* of the decision-making algorithm, especially on small data sets.

TABLE I
VARIANCE OF VALUE DIFFERENCE NORM

# of observations	ML	Smoothing	Option ML	Option smoothing
150	54.41	45.69	2.41	1.74
300	16.24	14.52	3.05	1.84
500	23.56	10.37	3.27	1.92
800	14.23	7.34	0.77	0.83
1300	11.90	4.37	0.78	0.60
2200	3.82	2.09	0.22	0.27
3900	1.57	1.12	0.31	0.39

Another interesting test relates to the speed by which the algorithm converges to the truly optimal policy (recall that the latter can be known). Table II lists the average number of states where the learned policy is *different* from optimal one, for increasing number of observations. Again, the use

of an SMDP appears advantageous, and smoothing slightly outperforms ML at data sets of very small size.

TABLE II
AVERAGE NUMBER OF STATES WITH A LEARNED POLICY DIFFERENT FROM THE OPTIMAL POLICY

# of observations	ML	Smoothing	Option ML	Option smoothing
150	6	4.5	4.1	3.6
300	4.3	4.1	2.6	2.4
500	3.1	2.7	2.4	1.5
800	2.5	2.2	1.4	1.3
1300	1.9	1.5	1.2	0.9
2200	0.9	0.9	0.2	0.3
3900	1.0	0.9	0.1	0.1

Along a similar line, Table III lists the average number of *successfully completed* episodes (i.e. reaching one of the goal states), given different number of observations. During the initial learning stages, when the algorithm is presented with very few data, ML and smoothing perform equally well on the SMDP, and significantly better than when applied on the MDP.

TABLE III
AVERAGE NUMBER OF COMPLETED EPISODES

# of observations	ML	Smoothing	Option ML	Option smoothing
150	0.9	1.5	13.7	13.9
300	5.1	5.4	24.9	25.4
500	15.7	17.1	43.0	45.0
800	45.8	41.5	65.8	65.7
1300	110.5	112.0	117.5	114.7
2200	194.7	187.1	171.3	173.3
3900	329.0	320.2	301.4	309.2

VI. CONCLUSION

In instances of HRI with dynamics described by imperfect discrete Markovian models which have to be refined using a small body of observations, the combination of an SMDP with options as the underlying model, and smoothing as the unknown parameter estimation method, appears to be particularly powerful. Because both aspects of this approach are well suited to function on small data sets, they combine well to yield improved performance in the initial phases of the learning process. The benefits seem to diminish when the training set grows significantly, but when obtaining data is particularly expensive or impossible, the application of this combination may as well make the difference.

REFERENCES

[1] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[2] A. Zehfroosh, E. Kokkoni, H. G. Tanner, and J. Heinz. Learning models of human-robot interaction from small data. In *2017 25th Mediterranean Conference on Control and Automation*, pages 223–228, July 2017.

[3] Nikolaos Mavridis. A review of verbal and non-verbal human-robot interactive communication. *Robotics and Autonomous Systems*, 63(P1):22–35, 2015.

[4] D Bernstein, R Givan, N Immerman, and S Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

[5] Tirthankar Bandyopadhyay, Kok Sung Won, Emilio Frazzoli, David Hsu, Wee Sun Lee, and Daniela Rus. Intention-aware motion planning. *Algorithmic Foundations of Robotics X: Proceedings of the Tenth Workshop on the Algorithmic Foundations of Robotics*, 86:475–491, 2013.

[6] Catharine LR McGhan, Ali Nasir, and Ella M Atkins. Human intent prediction using markov decision processes. *Journal of Aerospace Information Systems*, 5(12):393–397, 2015.

[7] J. J. Campos, D. I. Anderson, M. A. Barbu-Roth, E. M. Hubbard, M. J. Hertenstein, and D. Witherington. Travel broadens the mind. *Infancy*, 1(2):149–219, 2000.

[8] M. W. Clearfield. The role of crawling and walking experience in infant spatial memory. *Journal of Experimental Child Psychology*, 89:214–241, 2004.

[9] Eric A Walle and Joseph J Campos. Infant language development is related to the acquisition of walking. *Developmental Psychology*, 50(2):336–348, 2014.

[10] Karen Adolph. Motor development. *Handbook of child psychology and developmental science*, 2:114–157, 2015.

[11] Laura A Prosser, Laurie B Ohlrich, Lindsey A Curatalo, Katharine E Alter, and Diane L Damiano. Feasibility and preliminary effectiveness of a novel mobility training intervention in infants and toddlers with cerebral palsy. *Developmental Neurorehabilitation*, 15(4):259–66, 2012.

[12] Karina Pereira, Renata Pedrolongo Basso, Ana Raquel Rodrigues Lindquist, Louise Gracelli Pereira da Silva, and Eloisa Tudella. Infants with Down syndrome: percentage and age for acquisition of gross motor skills. *Research in Developmental Disabilities*, 34(3):894–901, 3 2013.

[13] Elizabeth S. Kim, Lauren D. Berkovits, Emily P. Bernier, Dan Leyzberg, Frederick Shic, Rhea Paul, and Brian Scassellati. Social robots as embedded reinforcers of social behavior in children with autism. *Journal of Autism and Developmental Disorders*, 43(5):1038–1049, 2013.

[14] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

[15] Alexander L Strehl, Lihong Li, and Michael L Littman. Reinforcement learning in finite MDPs: PAC analysis. *Journal of Machine Learning Research*, 10(Nov):2413–2444, 2009.

[16] David Chapman and Leslie Pack Kaelbling. Input Generalization in Delayed Reinforcement Learning: An Algorithm and Performance Comparisons. *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, 2:726–731, 1991.

[17] Michael Kearns and Daphne Koller. Efficient reinforcement learning in factored MDPs. *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, 16:740–747, 1999.

[18] Richard S. Sutton, Doina Precup, and Satinder Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181 – 211, 1999.

[19] Stanley F. Chen and Joshua Goodman. An Empirical Study of Smoothing Techniques for Language Modeling. *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, 13:310–318, 1996.

[20] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

[21] Junling Hu and Michael P Wellman. Nash Q-Learning for General-Sum Stochastic Games. *Journal of Machine Learning Research*, 4(6):1039–1069, 2003.

[22] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 2010.