Predictive State Representations with State Space Partitioning

Yunlong Liu[1] Yun Tang[1] Yifeng Zeng[1,2]
[1]Department of Automation, Xiamen University, Xiamen, China
[2]School of Computing, Teesside University, UK
ylliu@xmu.edu.cn tydqhqy@qq.com yifeng.zeng.dk@gmail.com

ABSTRACT

Predictive state representations (PSRs) are powerful methods of modeling dynamical systems by representing state through observational data. Most of the current PSR techniques focus on learning a complete PSR model from the entire state space. Consequently, the techniques are often not scalable due to the dimensional curse, which limits applications of PSR. In this paper, we propose a new PSR learning technique. Instead of directly learning a complete PSR at one time, we learn a set of local models each of which is constructed on a sub-state space and then combine the learnt models. We employ the *landmark* technique to partition the entire state space. We further show the theoretical guarantees on the learning performance of the proposed technique and present empirical results on multiple domains.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Theory

Keywords

Predictive State Representations; state space partitioning; landmark

1. INTRODUCTION

One of the most challenging problems in artificial intelligence is concerned with agents operating in stochastic and partially observable environments, i.e., how an agent can plan and act optimally under uncertainty. One commonly used technique is to model the system first, and then the problem can be solved using the developed model.

Thus far, the most general framework to model such controlled dynamical systems is *partially observable Markov decision processes* (POMDPs) [11]. However, it is well known that the POMDP model is based on unobserved or hidden states and requires significant amount of prior knowledge when learning such a model in practice. As an alternative, Littman *et al.* [12] proposed a new framework called *predictive state representations* (PSRs) to model such systems by defining and operating on the PSR state. PSRs define state using a prediction vector, which specifies probabilities over a number of tests. A *test* is a sequence of action-observation pairs to occur in the future and can be done on the system. The PSR state summarizes all of the information about the past. Unlike the POMDP model which is based on unobserved or hidden states, the PSR model is expressed entirely in terms of observable quantities. Compared to learning POMDP, learning PSR should be easier and less prone to local minimum problems. Additionally, PSRs are more compact than POMDPs [1].

Much progress has been made since PSRs were first described by Littman *et al.* [12]. However, the current PSR techniques mainly focus on learning the model from the entire state space. For linear PSRs, the vast majority of the literature on PSRs, one can directly derive the PSR model from a system dynamics matrix (SDM) that contains conditional probabilities of all possible tests given the past sequences and can be used to describe the underlying dynamical system [17]. The rank of the SDM is the linear dimension (denoted as n) of the dynamical system. To learn a PSR model on the entire state space, on one hand, we need to determine the n linearly independent columns of the SDM for state representation; on the other hand, we need to learn the model parameters and use these parameters for the prediction computation and state update, which involves at least $|A||O|$ full rank square matrices of rank n, where A is the set of actions that can be executed at each time step and O the set of possible observations of the underlying dynamical system. Given a complex setting of state space, learning PSR becomes rather expensive and time-consuming.

In this paper, we present a decomposition technique to learn the PSR models. Observing that the learning complexity is mainly due to the high dimension of state space, we proceed to learn local PSR models on a set of sub-state spaces and then complete the PSR model by combining the learnt models. To partition the entire space, we resort to the landmark technique that provides sufficient statistics of states in the learning. More importantly, we show that the learning performance of the proposed technique can be guaranteed theoretically. We demonstrate its performance in multiple domains.

We organize the remainder of this paper as follows. We briefly review the PSR model in Section 2. Subsequently, we propose the new PSR learning algorithm in Section 3. The technique first partitions the entire state space and

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

landmarks. Some practical and popular application, like MEDUSA [10], retrieves states at the middle point of the entire process and the states can also serve as the landmarks. Alternatives can seek for the advice from domain experts or exploit high-quality sensors to label some states, which are well accepted in many applications.

Algorithm 1 describes the partitioning of the state space through the landmark techniques. We partition the whole training data into multiple sets of training sequences that correspond to different sub-state spaces. Each set is marked by a landmark and contains the sequences of alternating actions and observations between its landmark and other landmarks, i.e., *seq* is the sequence of actions and observations between $currLandmark$ and $prevLandmark$, which includes $currLandmark$, but does not include $preLandmark$.

As landmark can serve as state and be used as the initial state of the underlying system [9, 13], for each sub-state space, the landmark of it is used as its initial state. We then proceed to learn local PSR models from the subsets of training data.

3.2 Learning Local Models

Current methods for learning a complete model of a system generally assume that any entry of the SDM can be estimated using the training data. However, for PSRs with state space partitioning, the training data is decomposed into multiple sets, and each set of data is used for learning a local model, i.e., the model of a sub-state space. In such cases, some entries of the SDM required for obtaining a local model cannot be estimated using only one set of data. For example, entry $p(t|h)$ cannot be estimated using only one set of data if h exists in one sub-state space and t exists in both this sub-state space and other sub-state spaces.

Obtaining Missing Values. According to the state space partitioning mechanism, for the prediction $p(t|h)$, if sequence h exists in one sub-state space and ht exists in more than one sub-state spaces, there exists landmark(s) in ht, and ht can be denoted $ht = ht_1t_2\cdots t_n$. Here ht_1 is the sequence in the first sub-state space, t_2 is the sequence in the second sub-state space, etc.. ht_1, t_2, ..., and t_{n-1} all end with a landmark. In such cases, $p(t|h)$ cannot be estimated using only one set of the partitioned data. Note that t_1 can be null.

Taking the example of $ht = ht_1t_2$ in two sub-state spaces, the following equation holds:

$$p(t|h) = p(t_1|h)p(t_2|ht_1) = p_1(t_1|h)p_2(t_2|\phi) \qquad (2)$$

where $p_j(t|h)$ is the prediction for test t at history h in the j^{th} sub-state space and ϕ the null history in the corresponding sub-state space. We define $p(t_1|h) = 1$ when t_1 is null.

Since ht_1 ends with a landmark, the landmark can serve as state and be used as the initial state of the next sub-state space. Hence the equation, $p(t_2|ht_1) = p_2(t_2|\phi)$, holds. $p_1(t_1|h)$ can be estimated using the set of data corresponding to the first sub-state space and $p_2(t_2|\phi)$ can also be estimated using the set of data corresponding to the second sub-state space. The missing value $p(t|h)$ can be obtained.

For test that exists in more than two sub-state spaces, the same mechanism can be applied to achieve the prediction.

Learning Local Models. Existing PSR learning algorithms can be used to obtain each local model from the corresponding partitioned training data.

3.3 Combining Local Models

We formally define the PSR model with state space partitioning as one tuple below.

DEFINITION 3. *A PSR model with state space partitioning is a tuple:* $\langle A, O, l_1\cdots l_N, Q^1\cdots Q^N, M, p(Q^1|\phi)\cdots p(Q^N|\phi)\rangle$.

where A is the set of actions, O the set of observations,N the number of the sub-state spaces, l_i the landmark corresponding to sub-state space i, Q^i the set of core tests for local model l_i, M the set of update parameters M^i_{ao} and m^i_{ao} for all a,o and all local models l_i, $p(Q^i|\phi)$ the initial prediction vector for local model l_i which can be estimated using the corresponding set of training data. Each local model l_i is specified by the tuple $\langle A, O, Q^i, M^i_{ao}, m^i_{ao}, p(Q^i|\phi)\rangle$, and the PSR state in sub-state space i is denoted by the concatenation of the local model and the prediction vector for that local model, i.e., as $[l_i, p(Q^i)]$.

Then, the prediction for any test t given any history h, $p(t|h)$, can be calculated as follows.

First, we determine the sub-state space to which the suffix of history h belongs. Let c be the sub-state space and h^c the suffix of history h that exists in c. Then, the equation $p(t|h) = p(t|h^c)$ holds as h^c starts with a landmark.

Second, we compute $p(t|h)$ in different cases: 1)if the test t only exists in sub-state space c, we compute $p(t|h)$ directly using the local model of c; 2)if the test t exists in multiple sub-state spaces, we compute $p(t|h)$ by linking landmarks in different sub-state spaces. Considering the test t in two sub-state spaces, c and $c+1$, we write t as $t = t^ct^{c+1}$ where $t^c = a_1o_1\cdots a_ko_k$, $t^{c+1} = a_{k+1}o_{k+1}\cdots a_no_n$, and t^c ends with the landmark of sub-state space c+1, then

$$\begin{aligned} p(t|h) =& p(t|h^c) = p(t^ct^{c+1}|h^c) = p(t^c|h^c)p(t^{c+1}|h^ct^c) \\ =& p_c(t^c|h^c)p_{c+1}(t^{c+1}|\phi) = p(Q^c|h^c)^T M^c_{a_1o_1}\cdots \\ & M^c_{a_{k-1}o_{k-1}} m^c_{a_ko_k} p(Q^{c+1}|\phi)^T M^{c+1}_{a_{k+1}o_{k+1}}\cdots \\ & M^{c+1}_{a_{n-1}o_{n-1}} m^{c+1}_{a_no_n} \end{aligned} \qquad (3)$$

Similarly, $p(t|h)$ can be calculated if t exists in more than two sub-state spaces.

Note that unlike the local PSR models proposed in [5, 18, 19, 20], which can make only certain predictions in some specific situations, our PSR model can make any conditional prediction about the system.

4. ALGORITHM CORRECTNESS AND COMPLEXITY ANALYSIS

In the following, we show that our method can be guaranteed to find the full core tests in each sub-state space. In addition, we analyze the data efficiency and time complexity of learning a PSR model with state space partitioning.

4.1 Algorithm Correctness

A sub-state space resulting from partitioning the full state space is a region in which every state is reachable from an initial state. We can consider the sub-state space as a separate dynamical system that is usually represented by a finite POMDP model. To learn a PSR model, a common approach for determining core tests uses an iterative procedure [7], i.e., finding the set of test Q where the rank of matrix of $p(Q|H')$ equals to the rank of matrix $p(aoQ|H')$

for all $a \in A$, $o \in O$, and H' is the set of histories. In Theorem 1 and Lemma 1, we show the correctness of learning PSR models on a sub-state space.

THEOREM 1. *A full set of core tests can be correctly found on a sub-state space by the iterative method.*

PROOF. Let a sub-state space be a tuple $< S, \mathcal{T}, \mathcal{O}, b_o, A, O >$, where S is a finite set of states of the world, $\mathcal{T}$ the *state-transition function*, $\mathcal{O}$ the *observation function*, and b_o the *initial belief state*.

For any test t at any history h, if t can be produced by a linear combination of Q found through the iterative procedure, aot can also be produced by a linear combination of Q for any $a \in A$ and $o \in O$:

$$p(aot|h) = b_h^T U(aot) \quad (4)$$
$$= b_h^T T^a O^{ao} U(t) \quad (5)$$
$$= b_h^T T^a O^{ao} U(Q) w_t \quad (6)$$
$$= b_h^T U(aoQ) w_t \quad (7)$$
$$= b_h^T U(Q) W w_t \quad (8)$$
$$= b_h^T U(Q) w_{t'} \quad (9)$$
$$= p(Q|h) w_{t'} \quad (10)$$

The components of $U(t)$ are the probabilities of the test t when applied from each underlying state of the POMDP. Q is the set of core tests found by the iterative method. $.^T$ is the transpose operator. w_t is the $|Q| \times 1$ weight vector for test t and W is a $|Q| \times |Q|$ matrix with its i^{th} column be the weight vector for the test aoq_i.

Simultaneously, at the end of the iterative procedure, ao for any $a \in A$, $o \in O$ can be produced by a linear combination of Q, then any test t can be produced by a linear combination of Q. Hence a full set of core tests Q is correctly found. □

LEMMA 1. *The learnt model parameters converge to the true model parameters as more training data are added.*

PROOF. The model parameters m_{ao}, M_{ao} are learnt by the following equations.

$$\hat{m}_{ao} = \hat{p}(Q|H)^{-1}\hat{p}(ao|H), \quad \hat{m}_{aoq_i} = \hat{p}(Q|H)^{-1}\hat{p}(aoq_i|H) \quad (11)$$

where m_{aoq_i} is the i^{th} column of M_{ao}.

As shown in Theorem 1, the set of core tests Q and the set of core histories H can be correctly found. Then, as more training data were included, the law of large numbers guarantees that the empirical estimation of $\hat{p}(t|h)$ converges to the true $p(t|h)$, i.e., $\hat{p}(T|H)$, $\hat{p}(ao|H)$ and $\hat{p}(aoq_i|H)$ converge to the true $p(T|H)$, $p(ao|H)$ and $p(aoq_i|H)$ respectively. Thus, the estimation of $\hat{m}_{ao}$ (and $\hat{M}_{ao}$) will converge to the true model parameters m_{ao} (and M_{ao}). □

4.2 Data Efficient Learning

Compared to other PSR learning algorithms, learning PSR by partitioning the state space requires less training data to learn a comparable model. For our approach, a single action-observation sequence can usually be partitioned into multiple training sequences using the landmarks. For example, given the landmark a^2o^2, we aim to estimate $p(a^1o^1|ha^2o^2)$ for any history h using three training sequences that start from the initial states: $s_1 = a^1o^1a^2o^2a^1o^3a^2o^3$,$s_2 = a^2o^2a^1o^2a^3o^3$,$s_3 = a^2o^1a^2o^2a^1o^1a^1o^3$. We first convert the training sequences into five training sequences: $s_1 = a^1o^1$, $s_2 = a^2o^2a^1o^3a^2o^3$, $s_3 = a^2o^2a^1o^2a^3o^3$, $s_4 = a^2o^1$, $s_5 = a^2o^2a^1o^1a^1o^3$. As stated above, $p(t|ha^2o^2) = p(t|a^2o^2)$ because a^2o^2 is a landmark. Then, $p(a^1o^1|ha^2o^2) = p(a^1o^1|a^2o^2) = \frac{1}{3}$. However, for other PSR learning approaches, $p(a^1o^1|ha^2o^2)$ is directly estimated using the original training sequences, and for any history h, $p(a^1o^1|ha^2o^2) = 1$ or 0. Obviously, the entry estimated by our method is more accurate since our algorithm exploits the training data more efficiently.

4.3 Complexity Analysis

Instead of directly learning a complete PSR model, we learn local models on sub-state spaces and then combine the learnt models to compose the complete model. Intuitively, a local model may be far simpler than the complete model of the original system, and can be no more complex. We formally state it below.

PROPOSITION 1. *The linear dimension of any local model is not larger than the linear dimension of the dynamical system.*

PROOF. The rows and columns of the SDM corresponding to a local model are part of rows and columns of the SDM of the original system. Thus, the rank of the SDM corresponding to a local model can be no more than the rank of the SDM of the original system, i.e., if the linear dimension of a dynamical system is n the linear dimension of any local model does not exceed n. □

In practice, while the SDM of the sub-state space is estimated using part of the whole training data, the SDM of the dynamical system is estimated using the whole training data that contains more rows and columns. The linear dimension of any local model is usually less than the linear dimension of the dynamical system.

Another advantage of our method is on the reduction of computational complexity. The state-of-the-art PSR learning algorithms are the spectral approaches that perform the singular value decomposition (SVD) operation on the matrix $p(T, H)$ [15], where $p(T, H)$ is a $T \times H$ matrix containing the joint probabilities of all tests $t \in T$ and all histories $h \in H$. With the setting of $|H| = |T| = n$, the number of the SVD operation is $O(n^3)$ in the spectral approach [15]. On the other hand, our approach conducts $|LM|$ of SVD operations on the matrix $p(T, H_i)$ where $i = 1, \cdots, LM$ and $|LM|$ is the number of the landmarks used to partition the data. Assuming that on the average, the size of each set of history H_i is $\frac{n}{|LM|}$. Thus, in our approach, SVD on these matrices requires $O(|LM|n((\frac{n}{|LM|}))^2) = O(\frac{n^3}{|LM|})$ operations.

5. EXPERIMENTAL RESULTS

We implemented the algorithm on learning PSR models with state space partitioning (PSRs with SSP). To learn local models on sub-state spaces, we employ the traditional iterative algorithm [7] in the implementation. Note that other existing PSR learning techniques can also be used to learn local PSR models. A landmark can be determined by checking whether the rank of the memory's corresponding sub-SDM is equal to 1[9, 13]. Many techniques for identifying/verifying landmarks have been well designed and implemented. For example, the techniques for identifying states can also be used to identify landmarks [21]. However, it is

not our interests to automatically find the landmarks in this paper. Our experiments assume that a set of landmarks are known as a prior in the learning.

5.1 Problem Domains

We tested the algorithm on three benchmarks [4], namely *Cheese Maze*, *Hallway* and *Hallway2*, in Figs. 2 - 4. *Cheese Maze* has 11 states and 7 observations. *Hallway* and *Hallway2* are relatively large domains by PSR standards, where previous iterative algorithms that learn complete models had difficulties. Differ from the environments used as the test bed for the iterative algorithms of learning a complete model of a system [7, 9, 14, 22], which are usually with several observations and no more than 20 states. In *Hallway*, the number of states is 48 (11 rooms with 4 orientations, plus 4 landmarks) and the number of observations is 20 (each possible combination of the presence of a wall in each of the 4 relative directions, plus the 4 landmarks that are visible when the agent is in the four particular locations). In *Hallway2*, the number of states is 71 (4 orientations in 16 rooms, plus 7 landmarks) and the number of observations is 23 (all combinations of walls, plus 7 landmarks).

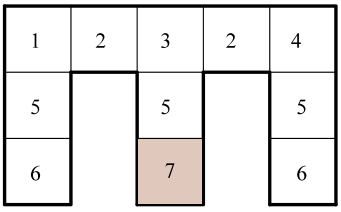


Figure 2: *Cheese Maze* with 11 states and 7 observations. Landmarks are observations 1, 3 and 4.

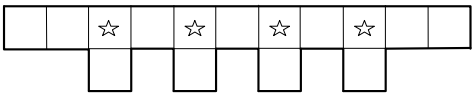


Figure 3: *Hallway* with 48 states and 20 observations. The stars are set as landmarks.

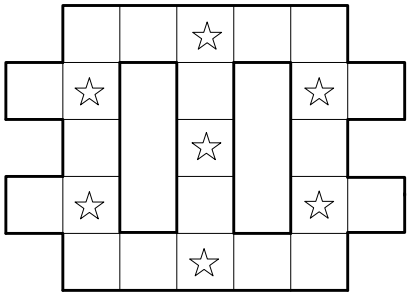


Figure 4: *Hallway2* with 71 states and 23 observations. The stars are seven landmarks.

5.2 Comparative Techniques

Many algorithms have been proposed for learning PSR models on some large scale systems. However, the algorithms focus on learning approximate predictive representations, i.e., learning a local model of the underlying system with the goal of making only certain predictions in some certain situations. We cannot compare them with our approach directly since we aim to learn a complete model. On the other hand, we observe that the transformed PSR (TPSR) [3, 16] model learning technique has shown good model learning performance and potential applications in realistic domain, we compared our new PSR learning technique with TPSR.

Additionally we implemented the expectation maximization (EM) algorithm for learning the POMDP models of the underlying system and compared it to our approach on learning PSR models.

5.3 Measurements

For each trial in each environment, a training data sequence using a uniformly-random action at each time step was generated to obtain the complete model of the underlying system. We computed the difference between the true predictions and the predictions given by the learned model over a test data sequence. The difference measures the accuracy of the learnt model, which is a normalized version of the measurement used in the previous work [5].

We used two error functions in the measurement. One error function is the average one-step prediction error per time step. It is computed in Eq. 12.

$$\frac{1}{L}\sum_{t=1}^{L}\frac{1}{|O|}\sum_{o\in O}(p(o|h_t,a_{t+1})-\hat{p}(o|h_t,a_{t+1}))^2 \tag{12}$$

where $p(o|h_t,a_{t+1})$ is the probability calculated from the true POMDP model of the problem domain and $\hat{p}(o|h_t,a_{t+1})$ the probability obtained from the learnt model. $L(=10,000)$ is the length of the test sequence in our experiments.

The other is the average four-step prediction error per time step on the test sequence. To reduce the computation complexity, we compute it in Eq. 13.

$$\begin{aligned}&\frac{1}{L}\sum_{t=1}^{L}(p(o_{t+1}o_{t+2}o_{t+3}o_{t+4}|h_ta_{t+1}a_{t+2}a_{t+3}a_{t+4})\\&-\hat{p}(o_{t+1}o_{t+2}o_{t+3}o_{t+4}|h_ta_{t+1}a_{t+2}a_{t+3}a_{t+4}))^2\end{aligned} \tag{13}$$

where $\hat{p}(\cdot|\cdot)$ is the estimation computed by the comparative methods. As the number of joint observations over four steps tends to be extremely large in the domains, it is intractable to compute $p(o_{t+1}o_{t+2}o_{t+3}o_{t+4}|h_ta_{t+1}a_{t+2}a_{t+3}a_{t+4})$ for all possible observations. We compute the $p(\cdot|\cdot)$ in the test sequences where a_t happens to be the action chosen at time t.

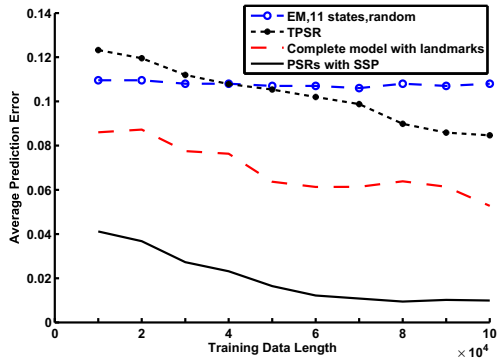
5.4 Results

We compare the performance of three methods (PSRs with SSP, TPSR and EM algorithms) in the aforementioned problem domains. We implemented two versions of the EM algorithm. One is with random initial states and the other has been supplied with almost correct initial states. We ran each algorithm for ten trials and plot the prediction error functions (Y-axis) over the training sequence lengths (X-axis) in Figs. 5- 7.

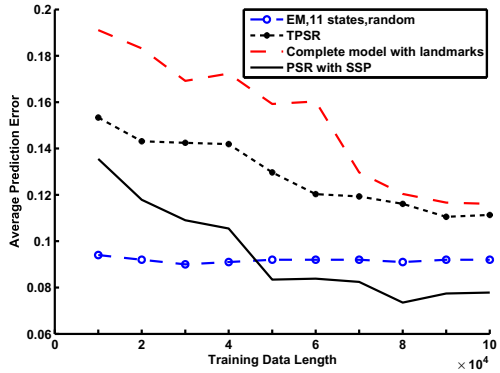
5.4.1 Cheese Maze

In *Cheese Maze*, with 80% of the time the agent moves in its intended direction and with 10% of the time it moves in

one of the directions perpendicular to its intended direction. In Fig. 2, observations 1, 3 and 4 are used as the landmarks for the state space partitioning. Obviously, other memories that uniquely determine the underlying state, such as $1S5$, $1E2$, $3S5$, etc., can also be used as landmarks for this environment, where S and E are the *GoSouth* and *GoEast* actions respectively.



(a)



(b)

Figure 5: (a)One-step; (b)four-step prediction error in *Cheese Maze*.

Fig. 5 shows the average predication errors of three methods in *Cheese Maze*. The performance of the EM algorithm with close to correct initial states is not shown in the figure since the algorithm achieves very low prediction error (around 0.007 in the one-step and four-step predictions). Given the almost correct input, the EM algorithm approaches the true POMDP model of the *Cheese Maze* domain with a reasonable size of 11 states.

For the one-step and four-step predictions (respectively in Fig. 5(a) and Fig. 5(b)), our algorithm outperforms both the EM with random initial states and TPSR learning techniques. Given a small amount of training data, our algorithm results in a large prediction error particularly in the four-step prediction. However, as the training-sequence length increases, our algorithm reduces its prediction error while the EM algorithm with random initial values does not improve its performance. Notice that the TPSR learning technique does not perform well and is even worse than the EM algorithm with random initial values in the four-step prediction.

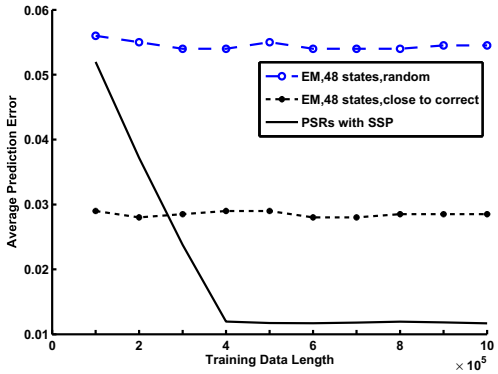
To further investigate the performance of state space partitioning approach, we estimate the sub-SDMs for the corresponding data partitioned by the same landmarks and combine them into one entire SDM. The SDM is used directly to learn a complete model (Complete model with landmarks). As can be seen in Fig. 5(a) and Fig. 5(b), both for the one-step and four-step predictions, models learned using state space partitioning approach perform significantly better than models on the entire state space. The explanation is that the more complex a SDM is, the more difficult to find the correct core tests, which results in less accurate models.

At the same time, as can be seen from the domain in Fig. 2. Rather than learning a complete model on the environment with 11 states, we learn 3 local models with 5, 7, and 5 states respectively. Hence the learning is more effective and efficient.

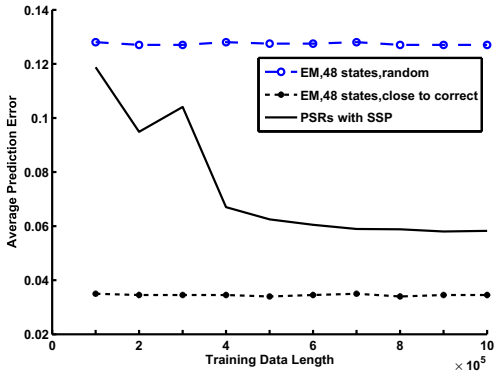
5.4.2 *Hallway and Hallway2*

The domains of *Hallway* and *Hallway2* are rather large and were seldom used to test the PSR learning techniques. To the best of our knowledge, neither of the existing algorithms has learnt the complete PSR models in the two domains.

In our experiments, we set 4 and 7 landmarks respectively in *Hallway* and *Hallway2*, which are marked as the stars in Figs. 3 and 4. In these squares, the agent will fully disambiguate its location.



(a)



(b)

Figure 6: (a)One-step; (b)four-step prediction error in *Hallway*.

Figs. 6 and 7 show the average one-step and four-step prediction errors of the comparative methods in both domains. As the TPSR approach needs to estimate the entries in the matrix $p(T, H)$, the calculation becomes extremely inaccurate due to a large number of states in the domains. The prediction conducted by TPSR is much worse than that of

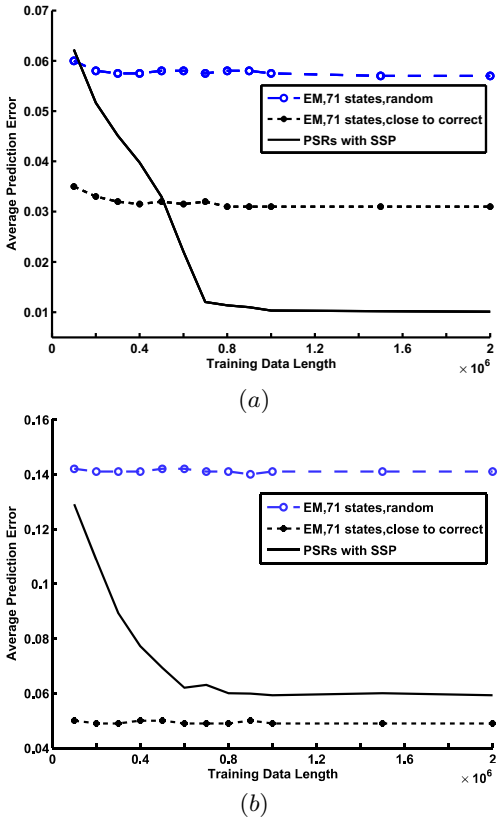


Figure 7: (*a*)One-step; (*b*)four-step prediction error in *Hallway2*.

the EM algorithm with random initial states, which is eliminated from the plots in Figs. 6 and 7.

As expected, our algorithm continues the good performance in two large domains and exhibits significantly lower prediction errors than the EM algorithm with random initial states. Meanwhile, we observe that our algorithm does not outperform the EM algorithm with close to correct initial states on the four-step prediction in both domains. This is mainly due to the incomplete calculation in Eq. 13 where only a relatively small set of observations (compared to more than 20^4 possible joint observations over four time steps in the two domains) are counted in the test sequences.

Overall, our algorithm achieves the acceptable prediction in the two large domains. This is benefited from the decomposition strategy in the new learning technique. For example, in *Hallway*, our algorithm learns 4 local models with 18, 15, 15, and 18 states respectively, which is much more efficient than learning a complete model with 48 states in total. In *Hallway2*, we avoid to learn a complete model on an environment with 71 states. Instead, the algorithm learns 7 local models with 24, 19, 19, 11, 19, 19 and 24 states respectively.

6. RELATED WORK

Much effort has been devoted to learning PSR models. In the work [7, 9, 13, 22], Monte-Carlo-style estimation was used to construct elements of SDM and then PSRs including the tests whose predictions constitute state and the model parameters can be derived from the SDM. McCracken and Bowling [14] learnt PSR models through a gradient descent approach, which provides online PSR algorithms. Liu *et al.* [13] presented an approach to learning the completed model by using landmarks to obtain the entries of the S-DM. However, unlike our approach, these methods directly learn the PSR model on the entire state space thereby being subject to the dimensional curse.

Rosencrantz *et al.* [16] developed the TPSR model that uses the principle component analysis for learning the model parameters in uncontrolled dynamical systems. Instead of maintaining probability distributions over the outcomes of a set of tests, they keep linear combinations of the probabilities to alleviate the discovery problem. Recently, some spectral algorithms for learning TPSR parameters were developed and they are computationally efficient and statistically consistent [2, 3]. Hamilton *et al.* [6] presented the compressed PSR models. The technique learns approximate PSR models of uncontrolled dynamical systems by exploiting a particularly sparse structure presented in some domains, which allows for an increase in both the efficiency and predictive power. These algorithms have been applied to some large domains. With landmarks, our partitioning approach is a general learning strategy and can be integrated into such algorithms. By adopting state space partitioning approach, as discussed earlier and shown in the experimental results, beside the reduction of computational complexity, more accurate learned model can be found.

To avoid the problem of the entire state space based methods, some work provides local approaches to modeling the observational data [5, 18, 19, 20]. Although in the work of [19], several local models can be combined to compute the predictions of intersections/unions of test of interest by making a strong assumption that the local models are mutually conditionally independent, most of these works learn only the local PSR model of the underlying system. In contrast, our approach still constructs the complete PSR model of the system using the observed data and can generate any conditional prediction about the system.

James *et. al.* [9] proposed the memory-based PSR (mPSR) that exploits the memory of the past to facilitate the model learning. More recently, Ong *et.al.* [15] presented the mixed observability PSR (MO-PSR) model by partitioning the SDM based on the fully observable variable of the last observation. However, some important differences exist between their approaches and ours: (i) Our approach first partitions the training data, then uses these partitioned data to estimate the corresponding sub-SDM and learn the local model. However, their methods first estimate the SDM using the training data, and rather than partitioning the training data, they partition the SDM, then each sub-SDM is used to learn the corresponding local model. As a result, our method is more efficient and required less training data to learn a model of better accuracy; (ii) our method can ensure the correctness of core tests discovery and model parameters learning. In the mPSR and MO-PSR models, each local model is learnt using a submatrix of the SDM of the original system in which the histories (rows) end with the same memory/full observable variable of the last observation. However, the states at the histories ending with the same memory/full observable variable of the last observation are usually not adjacent. For example, in the *Cheese Maze* environment (in Fig. 2), the states at the histories ending with memory "observation 5" are not adjacent. In such cas-

es, the sub-state space cannot be represented by a POMDP model. Consequently, equations 5 and 6 do not hold, and the correctness of learning core tests is not guaranteed.

7. CONCLUSION AND FUTURE WORKS

To reduce the complexity of learning a complete PSR model, we resort to learning local models on sub-state spaces and then combine the learnt models. By learning local models, the new algorithm achieves data efficiency resulting in a good scalability. The learning performance is also guaranteed in a theoretical way. Particularly we test the new algorithm in *Hallway* and *Hallway2*, where previous iterative algorithms that learn the complete model had difficulties.

As mentioned in the experiments, the traditional iterative algorithm was used to learn local models in this research. Obviously, other more modern learning algorithms, such as local approaches to modeling the observed data or the spectral methods proposed recently, can also be extended directly to learn local models. We will investigate the improved performance in more complex domains.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. 61375077 and No. 61375070) and Specialized Research Fund for the Doctoral Program of Higher Education of China (No. 20100121120022).

REFERENCES

[1] B. Balle, W. Hamilton, and J. Pineau. Methods of moments for learning stochastic languages: Unified presentation and empirical comparison. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 1386–1394, 2014.

[2] B. Boots and G. Gordon. An online spectral learning algorithm for partially observable nonlinear dynamical systems. In *Proceedings of the 25th National Conference on Artificial Intelligence (AAAI)*, 2011.

[3] B. Boots, S. Siddiqi, and G. Gordon. Closing the learning planning loop with predictive state representations. *International Journal of Robotic Research*, 30:954–956, 2011.

[4] A. R. Cassandra. Tony pomdp file repository page, 1999.

[5] M. Dinculescu and D. Precup. Approximate predictive representations of partially observable systems. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 895–902, 2010.

[6] W. L. Hamilton, M. M. Fard, and J. Pineau. Modelling sparse dynamical systems with compressed predictive state representations. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 178–186, 2013.

[7] 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 (ICML)*, pages 417–424, 2004.

[8] M. R. James and S. Singh. Sarsalandmark: An algorithm for learning in pomdps with landmarks. In *8th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 585–591, 2009.

[9] M. R. James, B. Wolfe, and S. Singh. Combining memory and landmarks with predictive state representations. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2005.

[10] R. Jaulmes, J. Pineau, and D. Precup. A formal framework for robot learning and control under model uncertainty. In *IEEE International Conference on Robotics and Automation*, ICRA 2007, pages 2104–2110, 2007.

[11] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.

[12] M. L. Littman, R. S. Sutton, and S. Singh. Predictive representations of state. In *Advances In Neural Information Processing Systems (NIPs)*, pages 1555–1561, 2001.

[13] Y. Liu and R. Li. Discovery and learning of models with predictive state representations for dynamical systems without reset. *Knowledge-Based Systems*, 22:557–561, 2009.

[14] P. Mccracken and M. Bowling. Online discovery and learning of predictive state representations. In *Advances in Neural Information Processing Systems (NIPs)*, pages 875–882, 2006.

[15] S. C. W. Ong, Y. Grinberg, and J. Pineau. Mixed observability predictive state representations. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence (AAAI)*, pages 746–752, 2013.

[16] M. Rosencrantz, G. Gordon, and S. Thrun. Learning low dimensional predictive representations. In *Proceedings of the Twenty-first International Conference on Machine Learning (ICML)*, pages 695–702, 2004.

[17] S. Singh and M. R. James. Predictive state representations: A new theory for modeling dynamical systems. In *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 512–519, 2004.

[18] E. Talvitie. Learning partially observable models using temporally abstract decision trees. In *Advances in Neural Information Processing Systems (NIPs)*, pages 827–835, 2012.

[19] E. Talvitie and S. P. Singh. Simple local models for complex dynamical systems. In *Proceedings of th Twenty-Second Annual Conference on Neural Information Processing Systems*, pages 827–835, 2008.

[20] E. Talvitie and S. P. Singh. Learning to make predictions in partially observable environments without a generative model. *J. Artif. Intell. Res. (JAIR)*, 42:353–392, 2011.

[21] D. Wierstra and M. Wiering. Utile distinction hidden markov models. In *Proceedings of the Twenty-first International Conference on Machine Learning (ICML)*, pages 108–115, 2004.

[22] B. Wolfe, M. R. James, and S. Singh. Learning predictive state representations in dynamical systems without reset. In *Proceedings of the Twenty-Second International Conference on Machine Learning (ICML)*, pages 980–987, 2005.