Group Sparse Optimization for Learning Predictive State Representations

Yifeng Zeng[a,*], Biyang Ma[b], Bilian Chen[b,**], Jing Tang[a], Mengda He[a]

[a]*School of Computing, Teesside University, UK.*
[b]*Department of Automation, Xiamen University, Xiamen 361005, China.*

Abstract

Predictive state representations (PSRs) are a commonly used approach for agents to summarize the information from history generated during their interaction with a dynamical environment and the agents may use PSRs to predict the future observation. Existing works have shown the benefits of PSRs for modelling partially observable dynamical systems. One of the key issues in PSRs is to discover a set of tests for representing states, which is called core tests. However, there is no very efficient technique to find the core tests for a large and complex problem in practice. In this paper, we formulate the discovering of the set of core tests as an optimization problem and exploit a group sparsity of the decision-making matrix to solve the problem. Then the PSR parameters can be obtained simultaneously. Hence, the model of the underlying system can be built immediately. The new learning approach doesn't require the specification of the number of core tests. Furthermore, the embedded optimization method for solving the considered group Lasso problem, called alternating direction method of multipliers (ADMM), can achieve a global convergence. We conduct experiments on three problem domains including one extremely large problem domain and show promising performances of the new approach.

Keywords:
Predictive state representations, group sparse, alternating direction method of multipliers

1. Introduction

In the past decades, a number of representations for modeling dynamical systems under uncertainty have been proposed for designing a rational, autonomous agent. However, challenge still exists for learning dynamical systems particularly for a partially observable domain with a large observation space. Currently, one of the most popular modeling frameworks, namely predictive state representations (PSRs), has been investigated as a general framework for offering an effective approach to model partially observable systems [11]. Unlike the latent state approach of partially observable Markov decision processes (POMDPs) [15],

*Corresponding author
**Principal corresponding author
Email addresses: y.zeng@tees.ac.uk (Yifeng Zeng), biyangma@stu.xmu.edu.cn (Biyang Ma), blchen@xmu.edu.cn (Bilian Chen), J.Tang@tees.ac.uk (Jing Tang), M.He@tees.ac.uk (Mengda He)

PSRs represent state of dynamical systems as a vector of predictions about observable events in future. Thus PSRs model the systems in a more concise way.

Since PSR models use observable quantities they can overcome the shortcomings of the traditional models and effectively solve a single agent forecasting and decision-making problem. In the issue of learning PSR models, we need to address two main research problems: one is to find a small sufficient set of tests to represent states in a dynamical system while the other is to learn the model parameters that maintain a probability distribution over the success of these tests as the dynamical system progresses. Much efforts has focused on the former problem. Many researchers have applied numerous algorithms in a small problem domain that can search the state space, e.g., [13, 21, 29]. For a larger domain in practice, since the state space of dynamical systems is often very large, the searching technique may not be an efficient approach to find the sufficient set of tests. Consequently, the line of research on learning PSR still attracts much attention in the intelligent agent and its relevant community.

With the benefit of using the system dynamics matrix to describe the dynamical system proposed by Singh et al. [27], researchers have made great progress in learning PSR models, and a matrix-based modeling method has became a popular tool. Indeed, much research work has demonstrated several ways of finding core tests and learning model parameters. The traditional iterative method [14, 29] can only be used in a toy problem. The spectral learning [2, 17, 18] and compressed sensing [10, 11] approaches avoid the complex discovery problem by maintaining a large enough set of tests that almost contain a sufficient set of tests representing the states, which may be prohibitively expensive; then by means of matrix dimension reduction method, the PSR models can be learnt. These two methods are considered to be the most successful for learning PSRs. The sub-state space method [21] first partitions the state space by using the landmark information and then applies traditional iterative methods to find a sufficient set of tests in each sub-state space. Most of the existing methods have shown expected performance on learning PSRs, while their capability is still limited by a large observation space since the methods typically consider a combinatorial number of observation sequences. In addition, when available training data is insufficient and the system dynamics matrix is filled with noise, it would be too inaccurate to obtain a satisfying PSR model particularly in a large problem domain. Meanwhile, too many tests will cause the manipulation of many high dimensional matrices, and the computational cost may be too high to afford. In general, we need to further explore new techniques for learning PSRs in a large and complex problem domain.

In this paper, we investigate the system dynamics matrix and learn the PSR models through optimization. The key underlying idea is to take advantage of group sparsity structure of the optimization variables. The discovery and learning problems are optimized and solved simultaneously. Here, we use alternating direction method of multipliers (ADMM) to solve the relevant optimization problem efficiently. Thus, the set of tests for representing state can be obtained and the model of the underlying system can be built straightforwardly and so does the model parameters. We conduct experiments on three problem domains including one extremely large domain. The experimental results demonstrate the effectiveness of our approach.

The rest of this paper is organized as follows. In Section 2, we introduce notations, fundamental theory and technical background of PSRs. Section 3 is devoted to the theoretical

analysis of learning the PSR models of dynamical systems. We conduct experimental study on several domains in Section 4. In Section 5, we discuss relevant works on learning PSRs. Finally we conclude our work and give some suggestions on the future work in Section 6.

2. Preparations

2.1. Basic matrix algebra

Throughout this paper, the usual lower case letters (e.g., x), the boldface lower case letters (e.g., $\boldsymbol{x} = (x_i)$), and the capital letters (e.g., $X = (x_{ij})$) denote scalars, vectors, and matrices, respectively. Denote $\mathbb{R}$ to be the space of real number. The space of n-dimensional vectors is denoted by $\mathbb{R}^n$ and the space of $m \times n$ matrices is denoted by $\mathbb{R}^{m\times n}$.

We denote by $\|\cdot\|_p$ where $1 \leq p \leq \infty$ to be the L_p norm of a vector, i.e., if $\boldsymbol{x} \in \mathbb{R}^n$,

$$\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p}.$$

The Donoho's L_0 norm of a vector is denote by $\|\cdot\|_0$, which is defined to be the number of nonzero entries of the vector. The most commonly used norm for a vector is the L_2 norm, or the Euclidean norm, given by $\|\boldsymbol{x}\|_2 = (\sum_{i=1}^{n} {x_i}^2)^{1/2}$.

For the norm of a matrix, similar to the vector case, the most common one is the Frobenius norm. For example, the Frobenius norm of a matrix $X \in \mathbb{R}^{m\times n}$ is defined as

$$\|X\|_F = \left(\sum_{i=1}^{m}\sum_{j=1}^{n} {x_{ij}}^2\right)^{1/2}.$$

A commonly used matrix norm in studying *group sparsity* is so-called the $L_{2,1}$ norm, which is the sum of the L_2 norms of the row vectors of a matrix. Specifically, if $X \in \mathbb{R}^{m\times n}$ and $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m \in \mathbb{R}^n$ are the row vectors of X, i.e., $X^{\mathrm{T}} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m)$, then the $L_{2,1}$ norm of X is

$$\|X\|_{2,1} = \|(\|\boldsymbol{x}_1\|_2, \|\boldsymbol{x}_2\|_2, \ldots, \|\boldsymbol{x}_m\|_2)\|_1 = \sum_{i=1}^{m} \|\boldsymbol{x}_i\|_2 = \sum_{i=1}^{m}\left(\sum_{j=1}^{n} {x_{ij}}^2\right)^{1/2}.$$

In the above definition, each row is considered as a group. Since $\|\boldsymbol{x}_i\|_2 = 0$ is equivalent to the vector $\boldsymbol{x}_i = 0$, the term group sparsity is used to describe this phenomenon: the matrix has a natural grouping of its components and the components within a group are likely to be either all zeros or all nonzeros. The $L_{2,1}$ norm is used in some convex optimization models to facilitate group sparsity replacing the $L_{2,0}$ norm of a matrix, which is defined to be the number of nonzero row vectors, i.e.,

$$\|X\|_{2,0} = \|(\|\boldsymbol{x}_1\|_2, \|\boldsymbol{x}_2\|_2, \ldots, \|\boldsymbol{x}_m\|_2)\|_0.$$

In fact, one can extend the $L_{2,0}$ and $L_{2,1}$ norms of a matrix to the $L_{p,0}$ and $L_{p,1}$ norms for any $1 \leq p \leq \infty$ as follows:

$$\|X\|_{p,0} = \|(\|\boldsymbol{x}_1\|_p, \|\boldsymbol{x}_2\|_p, \ldots, \|\boldsymbol{x}_m\|_p)\|_0,$$
$$\|X\|_{p,1} = \|(\|\boldsymbol{x}_1\|_p, \|\boldsymbol{x}_2\|_p, \ldots, \|\boldsymbol{x}_m\|_p)\|_1.$$

2.2. Second-order cone program

One of the tools to be used in this paper is second-order cone programs (SOCP). An SOCP is an optimization model by minimizing a linear function over the intersection of an affine space and some second-order cones, which can be written as

$$\begin{array}{ll} \min\limits_{\boldsymbol{x}} & \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \\ \text{s.t.} & \boldsymbol{a}_i{}^{\mathrm{T}}\boldsymbol{x} + b_i \leq \|C_i\boldsymbol{x} + \boldsymbol{d}_i\|_2,\ i = 1, 2, \ldots, m \\ & F\boldsymbol{x} = \boldsymbol{g}, \end{array}$$

where the problem parameters are $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{a}_i \in \mathbb{R}^n$, $b_i \in \mathbb{R}$, $C_i \in \mathbb{R}^{m_i \times n}$, $\boldsymbol{d}_i \in \mathbb{R}^{m_i}$, $F \in \mathbb{R}^{m \times n}$ and $\boldsymbol{g} \in \mathbb{R}^m$, and the decision variable is $\boldsymbol{x} \in \mathbb{R}^n$. It is a convex optimization problem, generalizing the linear program when all the matrices C_i's are zeros. An SOCP can be solved with great efficiency by interior point methods. For details, one is referred to Boyd and Vandenberghe [5].

2.3. Technical Background

Linear PSRs are a systematically-studied type of PSRs [27] and form the basis of our work. An agent interacts with dynamical environments by executing actions and receiving observations. The dynamical environment considered here is a discrete-time, controlled dynamical system, which produces a sequence of actions and observations with one action and one observation per time step. With the benefits of the linear PSR models, the agent can capture system's dynamics through probability distributions over tests conditioned on histories, make prediction on occurrence of observation information after performing a series of actions, and further use this information to predict the optimal sequential actions for a specific task.

For a dynamical system, suppose that $\mathcal{A} = \{a^1, a^2, \ldots, a^{|A|-1}, a^{|A|}\}$ is a set of all executable actions that an agent can operate and $\mathcal{O} = \{o^1, o^2, \ldots, o^{|O|-1}, o^{|O|}\}$ is a set of all observations that the agent may observe. A *test* is a sequence of action-observation pairs, e.g., test $t = a_1o_1a_2o_2 \ldots a_ko_k$; correspondingly, we have the action sequence $t^a = a_1a_2 \ldots a_k$ and the observation sequence $t^o = o_1o_2 \ldots o_k$. A *history* has the same structure as a test, which is used to describe the whole sequence of past action-observation pairs, e.g., history $h = a_1^h o_1^h a_2^h o_2^h \ldots a_l^h o_l^h$. The prediction of a test t given prior history h, denoted by $p(t|h)$, is defined as

$$p(t|h) = prob(o^{l+1} = o_1, \ldots, o^{l+k} = o_k | h, a^{l+1} = a_1, \ldots, a^{l+k} = a_k).$$

For any set of tests $Q = \{q_1, q_2, \ldots, q_n\}$, its prediction vector (or state vector) is $p(Q|h) = [p(q_1|h), \ldots, p(q_n|h)]^{\mathrm{T}}$. If $p(Q|h)$ forms a *sufficient statistic* at any history in the dynamical system, i.e., all tests can be predicted based on $p(Q|h)$ (in other words, there exists a function f_t such that $p(t|h) = f_t(p(Q|h))$ for any test t), then the set Q is called *core tests*. The process of finding Q is called *discovery problem*, while the computation of the projection vectors by using Q to represent all other tests is called *learning problem*.

Particularly, for a linear system, a significantly different approach for studying PSRs was proposed by Littman et al. [19] who introduced the concept of system dynamics matrix D. The matrix D has an infinite number of rows and columns, where the rows correspond to

all possible histories and the columns correspond to all possible tests. Each element is the prediction of a test at a given history, i.e., $D_{ij} = p(t_j|h_i)$. Then, our goal is to find a maximal set Q of linearly independent columns of matrix D, so that all columns are a weighted sum of those columns. Hence, at any history h, for any test t, $p(t|h)$ can be calculated by

$$p(t|h) = p(Q|h)^{\mathrm{T}}\, \boldsymbol{m}_t, \tag{1}$$

where $\boldsymbol{m}_t \in \mathbb{R}^{n\times 1}$ is called the *projection vector*. As time passes, the linear PSRs should update its state (i.e., prediction vector). The update procedure calculates the new state $p(Q|hao)$ from the old state $p(Q|h)$ after agent taking the action a and seeing the observation o from history h. For any core test $q_i \in Q$, we can calculate as follows:

$$p(q_i|hao) = \frac{p(aoq_i|h)}{p(ao|h)} = \frac{p(Q|h)^{\mathrm{T}}\boldsymbol{m}_{aoq_i}}{p(Q|h)^{\mathrm{T}}\boldsymbol{m}_{ao}}, \tag{2}$$

where $\boldsymbol{m}_{aoq_i}$ and $\boldsymbol{m}_{ao}$ are the $\boldsymbol{m}_t$ for each one-step test (ao) and each one-step extension (aoq_i), $\forall a \in \mathcal{A}, o \in \mathcal{O}, q_i \in Q$, respectively. Eq. (2) is obtained first by Bayes formula and followed by Eq. (1). Moreover, Eq. (2) can be written in a more compact form by defining matrices $M_{ao} \in \mathbb{R}^{n\times n}$, where the i-th column is $\boldsymbol{m}_{aoq_i}, \forall a \in \mathcal{A}, o \in \mathcal{O}$, thus we have

$$p(Q|hao) = \left(\frac{p(Q|h)^{\mathrm{T}} M_{ao}}{p(Q|h)^{\mathrm{T}}\boldsymbol{m}_{ao}}\right)^{\mathrm{T}}. \tag{3}$$

The vectors $\boldsymbol{m}_{ao}$ and matrices $\{M_{ao}\}$ $(\forall a \in \mathcal{A}, o \in \mathcal{O})$ are called the model parameters of the linear PSRs. If the core tests Q are found, the parameters can be computed easily as follows:

$$\begin{aligned} \boldsymbol{m}_{ao} &= p(Q|H)^{-1}p(ao|H) \\ \boldsymbol{m}_{aoq_i} &= p(Q|H)^{-1}p(aoq_i|H) \end{aligned} \tag{4}$$

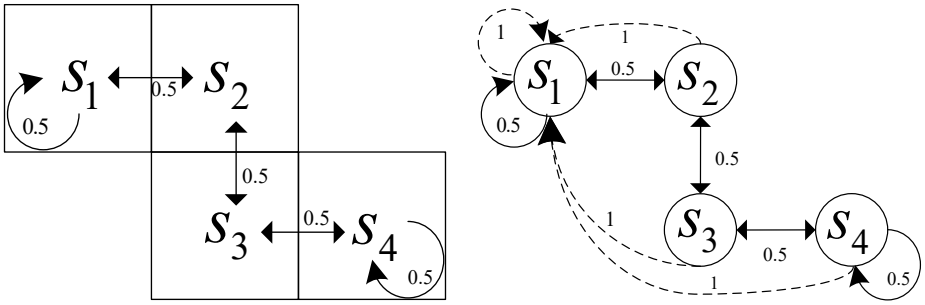
where H is the set of possible histories and $^{-1}$ denotes (pseudo)inverse of a matrix. The probabilities e.g., $p(Q|H)$, $p(ao|H)$ and $p(aoq_i|H)$ can be estimated using the training data.

In summary, a linear PSR model in a partially observable system is formulated with the parameters $< \mathcal{A}, \mathcal{O}, Q, \{\boldsymbol{m}_{ao}\}, \{M_{ao}\}, p(Q|\phi) >$: the set of actions $\mathcal{A}$, the set of observations $\mathcal{O}$, the set of core tests Q, the model parameters $\boldsymbol{m}_{ao}$ and $\{M_{ao}\}$ for all $a \in \mathcal{A}, o \in \mathcal{O}$, and an initial prediction vector $p(Q|\phi)$, where ϕ is the null history. Unless stated otherwise, the term PSR refers to a linear PSR in the rest of this paper.

2.4. Robot Navigation Example as a PSR

In this part, we propose a robot navigation example to elaborate how to use PSRs to model a dynamical system. Fig. 1 shows a robot navigation problem. This simulated dynamical system is modeled using discrete time and consists of 4 states (denoted by S_1, S_2, S_3 and S_4), 2 actions (move and reset, denoted by f and r, respectively) and 2 observations (denoted by 0 and 1). Initially at ϕ, the robot is in state S_1. Its target is to reach state S_4.

In the navigation, the robot executes action f and it moves to its adjacent grid or stay still with equal probability, e.g., the robot is in state S_1 at some time, then the next time step it will in S_1 or S_2 with the probability 0.5, see Fig. 1(a). Also the robot can execute



(a) Robot navigation in a grid world (b) Action execution in the robot navigation

Fig. 1. A graphical illustration of robot navigation problem

action r and return to initial state S_1 with the probability 1, and the robot will always receive observation 0; while, in target state S_4, it will get observation 1, see Fig. 1(b).

We proceed to show how to use a PSR to model this dynamical system. After a large number of samples for modeling the dynamical system are extracted from the robot's experience, we could use these insufficient samples to build the system dynamics matrix D, where each row is corresponding to each possible history and each column is corresponding to each possible test, see Fig. 2. Without loss of generality, the histories and tests are arranged in length-lexicographical ordering. Then we remove all the columns and rows with zeros since the entries never occur and hence do not play a role in the system behavior. For every entry in matrix D, there are many methods to estimate these prediction values in an unbiased way and one simple way is to use multiple trajectories of experience to estimate the prediction $p(t|h)$.

D	r0	f0	f1	r0r0	r0f0	f0r0	f0f0	f0f1	r0r0r0	⋯
∅	0.51	0.49	0	0.25	0.25	0.24	0.25	0	0.1281	⋯
r0	0.5	0.5	0	0.25	0.25	0.25	0.25	0	0.1209	⋯
f0	0.5	0.5	0	0.25	0.25	0.25	0.25	0	0.1209	⋯
r0r0	0.5	0.5	0	0.24	0.26	0.26	0.24	0	0.1255	⋯
r0f0	0.5	0.5	0	0.25	0.25	0.24	0.22	0.03	0.1278	⋯
f0r0	0.49	0.51	0	0.23	0.26	0.27	0.24	0	0.1071	⋯
f0f0	0.5	0.5	0	0.25	0.25	0.25	0.25	0	0.1209	⋯
r0r0r0	0.48	0.52	0	0.25	0.23	0.27	0.25	0	0.1413	⋯
r0r0f0	0.52	0.48	0	0.27	0.25	0.23	0.22	0.03	0.136	⋯
r0f0r0	0.5	0.5	0	0.25	0.26	0.24	0.26	0	0.1261	⋯
r0f0f0	0.48	0.45	0.06	0.25	0.27	0.25	0.22	0.01	0.1071	⋯
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

Fig. 2. The system dynamics matrix D

Then we model the robot navigation example system by the following PSR. One possible set of core tests is $Q = \{r0, f0, r0r0, r0f0\}$. The initial prediction vector is $p(Q|\phi) = [0.51, 0.49, 0.25, 0.25]^{\mathrm{T}}$. After that, we calculate all the model parameters m_{ao} and M_{ao} to make predictions. For example, the probability of a future test $t = f0f0f0f0$ on condition

of the history $h = r0$ can be calculated by

$$p(t|h) = 0.0284\,p(r0|h) + 0.0312\,p(f0|h) + 0.0059\,p(r0r0|h) + 0.0027\,p(r0f0|h).$$

3. Analysis of Discovering and Learning Techniques in PSRs

This section is devoted to studying the optimization models for PSR problems.

3.1. Optimization models

As discussed in Section 2.3, one key issue in PSR problems is to find the core tests Q to represent the state. Once it is settled, the parameters of the model can be easily updated and the prediction can be computed. Similar to the technique introduced in [19], our first main task is to construct the system dynamics matrix D. If we have the core tests Q selected from D, then all the columns of the matrix D can be linearly represented by a coefficient matrix, say X, such that

$$D = DX. \tag{5}$$

In general, the number of solutions for a system such as (5) will be infinite. We are interested to find a solution as simple as possible, i.e., the matrix X is group sparsity where the groups used to represent D are the row vectors of X. Therefore, it is naturally to search a solution X that has the minimum number of nonzero rows. This can be formulated as an optimization model as below:

$$\begin{array}{lll} (P) & \min\limits_{X} & \|X\|_{p,0} \\ & \text{s.t.} & D = DX. \end{array}$$

Unlike other learning PSR approaches, we do not need to specify the number of core tests in advance as it is automatically determined by the optimal solution of (P). Remark that for any p satisfying $1 \le p \le \infty$ in the model (P), its solutions will be the same. We do not specify p at the moment as this would give us the flexibility in the solution methods of (P). One important property of this model is the following.

Theorem 3.1. *If X^* is an optimal solution of (P), then* $\operatorname{rank}(D) = \|X^*\|_{p,0}$.

Proof. First, as the matrix X^* only has $\|X^*\|_{p,0}$ number of nonzero rows, the rank of X^* must be no more than $\|X^*\|_{p,0}$, i.e.,

$$\operatorname{rank}(X^*) \le \|X^*\|_{p,0}.$$

Besides, as $D = DX^*$, we have $\operatorname{rank}(D) \le \operatorname{rank}(X^*)$. Therefore we arrive at

$$\operatorname{rank}(D) \le \|X^*\|_{p,0}. \tag{6}$$

On the other hand, let $r = \operatorname{rank}(D)$ and without loss of generality we suppose the first r columns of D are linear independent. Let $D = (\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_n)$ where $\boldsymbol{d}_i$'s are column vectors of D. Each column of D can be written as a linear combination of $\{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_r\}$, in particular, let

$$\boldsymbol{d}_k = x_{1k}\boldsymbol{d}_1 + x_{2k}\boldsymbol{d}_2 + \cdots + x_{rk}\boldsymbol{d}_r, \quad k = 1, 2, \ldots, n.$$

Define

$$X_0 = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{r1} & x_{r2} & \cdots & x_{rn} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix} \in \mathbb{R}^{n\times n}.$$

We have $DX_0 = D$ and X_0 is a feasible solution of (P). By checking the number of obvious zero vectors in X_0, it follows that

$$\|X^*\|_{p,0} \leq \|X_0\|_{p,0} \leq r = \text{rank}\,(D).$$

The claimed result is proved by combing the above inequality with (6). □

According to the formulation (P), the core tests Q can be obtained from D and the model parameters can be obtained from the nonzero rows of the solution X to the model (P). Therefore, the discovering problem and learning problem of PSR models are solved at the same time. As mentioned before, the parameter p can be chosen any value in the interval $[1, \infty]$. In the following discussions, we let $p = 2$ as a natural choice. Besides, $p = \infty$ is a good alternative. Well, the difficulty of the model (P) lies in the L_0 norm in the objective function, which is NP-hard to compute in general. Due to the efficiency of receiving sparse solutions via L_1 norm relaxation problems [1, 6, 7, 12, 16], we now consider the L_1 norm relaxation of (P), i.e.,

$$(R) \quad \begin{array}{ll} \min\limits_{X} & \|X\|_{2,1} \\ \text{s.t.} & D = DX. \end{array}$$

This relaxation is actually a convex optimization model, specifically, a second-order cone program discussed in Section 2.2. To see why, let $\boldsymbol{x}_i$'s be the row vectors of X, i.e., $X^{\mathrm{T}} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m)$. We have

$$\|X\|_{2,1} = \sum_{i=1}^{m} \|\boldsymbol{x}_i\|_2.$$

Let $t_i = \|\boldsymbol{x}_i\|_2$ for $i = 1, 2, \ldots, m$ in order to replace the objective function of (R) being a linear function. We get an equivalent formulation of (R).

$$\begin{array}{ll} \min\limits_{X,\boldsymbol{t}} & \sum_{i=1}^{m} t_i \\ \text{s.t.} & \|\boldsymbol{x}_i\|_2 = t_i,\ i = 1, 2, \ldots, m \\ & D = DX. \end{array}$$

The constraint $\|\boldsymbol{x}_i\|_2 = t_i$ can be replaced by $\|\boldsymbol{x}_i\|_2 \leq t_i$. This is because that the objective is to minimize the sum of all t_i's and an optimal solution makes the inequality being tight for all i. Therefore we arrive the following.

Proposition 3.2. *Problem (R) is equivalent to the following second-order cone problem*

$$(S) \quad \begin{array}{ll} \min\limits_{X,\boldsymbol{t}} & \sum_{i=1}^{m} t_i \\ \text{s.t.} & \|\boldsymbol{x}_i\|_2 \leq t_i,\ i = 1, 2, \ldots, m \\ & D = DX. \end{array}$$

where $X^{\mathrm{T}} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m)$ *and* $\boldsymbol{t} = (t_1, t_2, \ldots, t_m)^{\mathrm{T}}$.

An SOCP can be solved by various methods. If the size of (S) is not large, it can be solved by the state-of-the-art solver called CVX [9] to obtain an optimal solution. For large size problems, standard algorithms are usually computationally expensive. Several efficient first-order algorithms have been proposed, e.g., a spectral projected gradient method [30], an accelerated gradient method [20] and block-coordinate descent algorithms [24].

3.2. Group LASSO and the ADMM method

Ideally, for a given matrix D, one is able to find sparse matrix X such that $D = DX$. However, in practice, the data matrix D could be noisy. Restriction on $D = DX$ shall be relaxed such that they are as close as possible, i.e., $\|D - DX\|_F$ is to be minimized as well. This actually follows from the well known least absolute shrinkage and selection operator (LASSO) problem. Inspired by [31], we know that the group LASSO could impose sparsity on groups of variables (features) via the Ł$_{2,1}$-regularization, which is a popular extension of the L_1-regularization (LASSO). Therefore, the model (R) could be reformulated as a group LASSO problem as follows:

$$(P_1) \quad \min_{X} \quad \tfrac{1}{2}\|DX - D\|_F^2 + \lambda \|X\|_{2,1}.$$

(P_1) is an unconstrained optimization problem. A common approach to efficiently solve this problem is the alternating direction method of multipliers (ADMM). Details on ADMM is referred to [4]. The ADMM is actually based on a variable-splitting technique and an augmented Lagrangian framework. Specifically, by introducing a new variable E for the model (P_1), we have

$$\begin{aligned} (P_2) \quad \min_{X,E} \quad & \tfrac{1}{2}\|DX - D\|_F^2 + \lambda \|E\|_{2,1} \\ \text{s.t.} \quad & X = E. \end{aligned}$$

The augmented Lagrange function is then given by

$$L_\mu(X, E, Z) = \frac{1}{2}\|DX - D\|_F^2 + \lambda \|E\|_{2,1} + \operatorname{tr}\left(Z^{\mathrm{T}}(X - E)\right) + \frac{\mu}{2}\|X - E\|_F^2,$$

where $\mu > 0$ is a penalty parameter, Z is the Lagrange multiplier, and $\operatorname{tr}(\cdot)$ is the trace operator.

The ADMM generally consists of the following iterations:

$$X^{k+1} = \arg\min_{X} L_\mu(X), \tag{7}$$

$$E^{k+1} = \arg\min_{E} L_\mu(E), \tag{8}$$

$$Z^{k+1} = Z^k + \mu(X^{k+1} - E^{k+1}). \tag{9}$$

First we fix E, Z and update X. The objective function $L_\mu(X)$ is equivalent to

$$L_\mu(X) = \frac{1}{2}\|DX - D\|_F^2 + \operatorname{tr}\left(Z^{\mathrm{T}}X\right) + \frac{\mu}{2}\|X - E\|_F^2 + c_0,$$

where c_0 is some constant. The optimal solution can be obtained analytically by

$$X = \left(D^{\mathrm{T}}D + \mu I\right)^{-1}(D^{\mathrm{T}}D - Z - \mu E).$$

Next we fixed X, Z and update E. The objective function $L_\mu(E)$ is now rewritten as

$$L_\mu(E) = \lambda \|E\|_{2,1} - \mathrm{tr}\left(Z^{\mathrm{T}}E\right) + \frac{\mu}{2}\|E - X\|_F^2 = \lambda \|E\|_{2,1} + \frac{\mu}{2}\left\|E - \left(X + \frac{1}{\mu}Z\right)\right\|_F^2.$$

This problem can also be solved analytically via m subproblems. Let $G = X + \frac{1}{\mu}Z$. The row vectors of the optimal E can be obtained by the row vectors of G, i.e.,

$$\boldsymbol{e}_i = \max\left\{\|\boldsymbol{g}_i\|_2 - \frac{\lambda}{\mu}, 0\right\}\frac{\boldsymbol{g}_i}{\|\boldsymbol{g}_i\|},$$

where $\boldsymbol{e}_i$ and $\boldsymbol{g}_i$ is the i-th row vector of the matrix E and G, respectively.

The ADMM procedure to solve the model (P_2) is summarized as below.

Algorithm 3.3. *ADMM for the model* (P_2).

- *Input: the matrix* D*, a parameter* $\lambda > 0$ *and a termination tolerance* ϵ.

0 Initialize E^0*,* Z^0 *and* μ*;*

1 Update X^{k+1} *by* (7)*,* E^{k+1} *by* (8) *and* Z^{k+1} *by* (9)*, sequentially;*

2 If the stopping criterion is satisfied, stop; Otherwise, go to Step 1.

- *Output: an optimal* X.

The stopping criterion of the above algorithm can be set as

$$\|X^{k+1} - E^{k+1}\|_F \le \epsilon \text{ or } \max\left\{\|X^{k+1} - X^k\|_F, \|E^{k+1} - E^k\|_F\right\} \le \epsilon.$$

The former one is called the primal residual and the latter one is the difference between successive iterations. The global convergence of Algorithm 3.3 can be easily established, similar to the global convergence of [4, Section 3].

Theorem 3.4. *The sequence* $\{(X^k, E^k, Z^k)\}$ *produced by Algorithm 3.3 converges to an optimal solution of* (P_2) *from any starting point.*

It is worth mentioning that the above convergent property satisfies for any parameter $\mu > 0$.

3.3. Discovering and Learning Problems

We summarize our approach in Fig. 3, which includes three main steps:

① Construct the system dynamics matrix D and formulate the PSR problem as an optimization problem;

② Solve the optimization problem to get core tests Q and coefficient matrix X;

③ Extract information from matrix X or compute it by Eq. (4) to get the model parameters $\{\boldsymbol{m}_{ao}\}$ and $\{M_{ao}\}$.

Remark that when matrix X is obtained, we store its nonzero row indices, say, $I = \{r_1, r_2, \ldots, r_q\}$, and denote the corresponding matrix by $\bar{X}$. Then we could easily find a set of the linearly independent columns from D, whose column indices coincide with set I, that is, the core tests $Q = \{D_{\cdot,r_1}, D_{\cdot,r_2}, \ldots, D_{\cdot,r_q}\}$ are found, which solves the discovery problem. Meanwhile, each column of matrix $\bar{X}$ is a weighted vector, which corresponds to the linear projection vector $\{\boldsymbol{m}_{ao}\}$ or $\{M_{ao}\}$. Hence, the learning problem is settled. To conclude this section, we can simultaneously solve the discovering problem and learning problem of PSR models via one optimization.

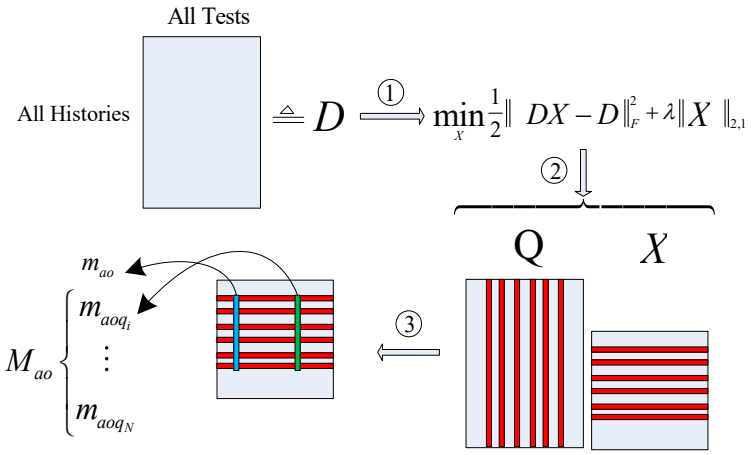


Fig. 3. The framework of our approach

4. Experimental Results

We test our approach on three different standard benchmarks that are commonly used in the literatures, i.e., Hallway and Hallway2 [8], and Poc-Man [26]. All of them are large domains and were originally defined as *POMDPs*. Throughout our experiments, the agents were given random exploration policies, and both the sequences of actions and the sequences of observations were recorded as training datasets. From those, we propose an optimization problem based on ADMM for discovering the core tests to represent states, and then the PSR model of the underlying system can be learnt directly, which produces probability distributions over action-observation sequences. To measure the learnt PSR model of a given domain, we test the empirical performance of the model by simulating a sequence of action-observations (whose length starts from 1 to 10) and evaluate the predictions of the model.

4.1. Experimental Setting

4.1.1. Problem Domains

Hallway and Hallway 2. In the Hallway domain, an agent interacts with its environment by taking actions and receiving observations from the environment in order to reach

the target place. In the interaction with the environment, the agent executes actions with disturb and receives observations with noise, which causes the domain partially observable and thus increases the challenge of modeling the dynamical system. To the best of our knowledge, all the existing algorithms have difficulty to learn the complete PSR models in the two domains.

In these two domains, as shown in Figs. 4 and 5, the agent in each grid has four states considering the agent's current orientation and increases the indices in a clockwise direction. The agent can only conduct actions like no operation, move forward, move backward, turn left and turn right, and receive observations about whether the four directions around have walls, reach the goal or stay in the (cross) grid or not.

In Hallway, the number of states is 61 (15 rooms with 4 orientations, plus the goal state) and the number of observations is 21 (each possible combination of the presence of a wall in each of the 4 relative directions, which results in 16, plus the 4 observations when the agent is in one of the cross grids and faces to the South, and then add the agent in the goal grid), see Fig. 4. In Hallway2, the number of states is 92 (4 orientations in 23 rooms) and the number of observations is 17 (all possible combinations of walls, plus the agent in the goal grid), see Fig. 5.

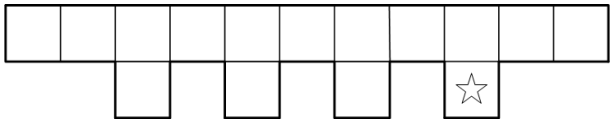


Fig. 4. Hallway with 61 states and 21 observations. The star is the goal grid.

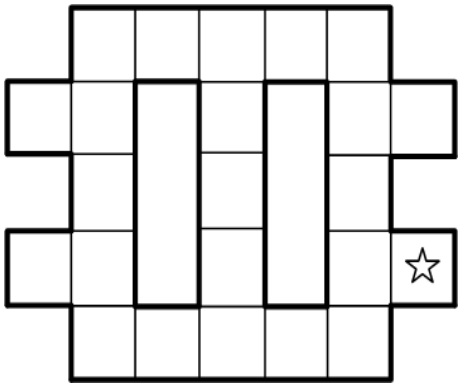


Fig. 5. Hallway2 with 92 states and 17 observations. The star is the goal grid.

Poc-Man. The PocMan domain is a partially observable version of the popular video game Pac-Man [26] for testing the performance of PSR models when confronts a large dynamical system. In PocMan domain, the goal of the agent is to search around for randomly placed food pellets while navigating in the maze and avoiding be catched by any of four ghosts just like in the video game. However, only local environment states and partial observations are accessible for the agent to accomplish its mission, which is not the same as the video game version. To learn a perfect predictive representation of this domain is a challenge, because it has a large number of state space($|S| \approx 10^{56}$) and observation space($|O| \approx 2^9$), see Fig. 6.

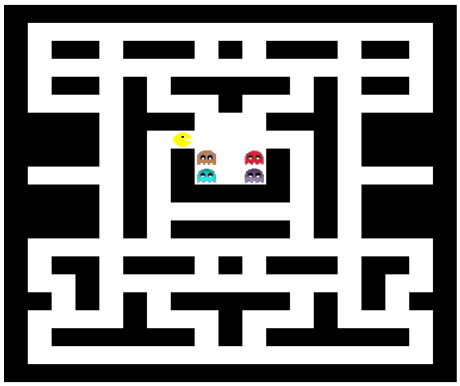


Fig. 6. Pocman with a large number of states and observations.

4.1.2. Comparative Methods

We aim to learn a complete model of PSR in the large scale systems as elaborated above. Note that many algorithms focus on learning a local model of the underlying system with the aim of making only predictions in specific situations. Thus, these algorithms are not included in the comparison. For the Hallway and Hallway2 problem, we compare our new learning PSR technique with a spectral based method, i.e., the transformed PSR (TPSR) [2, 3]) model since it has shown good model learning performance. For the Poc-Man problem, the compressed PSR (CPSR) approach [10, 11] is also used for comparison, since it is suitable for a domain with a particular sparse structure and is able to produce a good quality model.

4.1.3. Performance Measurements

The main purpose of a dynamical system is to predict the probabilities of different observations when an action is executed given an arbitrary history h_t. For each trial in the environment, a training data sequence using a random action at each time step was generated to obtain the complete model of the underlying system. We evaluate the learnt models in terms of prediction accuracy, which computes the gap between the true predictions and the predictions given by the learnt model over all test data sequences. For Hallway and Hallway2, we can obtain the true value of each step prediction from related POMDP files. However, in Poc-Man, we cannot obtain the true predictions and use Monte-Carlo rollout predictions [10] instead.

Two error functions were used in our numerical experiments. One is called root mean squared error (*RMSE*) that measures the divergence of one-step prediction error per time step on the test sequences, which is given by Eq. (10):

$$RMSE = \sqrt{\frac{1}{H \times L} \sum_{t=1}^{H} \sum_{i=0}^{L-1} (\hat{p}(o_{t+i+1}|h_{t+i}a_{t+i+1}) - p(o_{t+i+1}|h_{t+i}a_{t+i+1}))^2}. \tag{10}$$

The other is called absolute error (*AE*) that computes the average of absolute errors of one-step prediction per time step, as shown in Eq. (11).

$$AE = \frac{1}{H} \sum_{t=1}^{H} |\hat{p}(o_{t+1}|h_t a_{t+1}) - p(o_{t+1}|h_t a_{t+1})|. \tag{11}$$

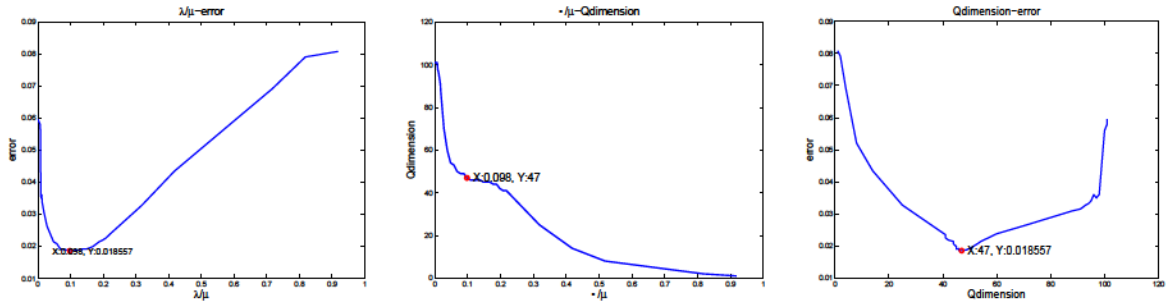
In Eqs. (10) and (11), $p(\cdot)$ is the probability obtained from the true POMDP model or the Monte-Carlo rollout prediction, and $\hat{p}(\cdot)$ is the estimated probability computed by the the learnt model. H is the total number of test sequences, and tests of length L starting from 1 to 10 were used, respectively. Eq. (10) will be used in Section 4.2 and Eq. (11) will be used in Section 4.3.

For each trial in the environment, a training data sequence using a random action at each time step was generated to learn the PSR model of the underlying system. We use 10,000 training sequences to learn a model representation for Hallway and Hallway2 domains, and 1000 for Poc-Man domain. In the sampling process, the step-length of sequences for Hallway and Hallway2 is 10 while it is 5 for Poc-Man.

4.2. Parameter Sensitivity Analysis

As indicated in Theorem 3.4, the value μ is irrelevant to the convergence result. Here, we use $\mu = 0.619$ in the analysis. In this part, we investigate the performance of various values of the penalty parameter λ of optimization problem (P_1) on all the three different domains, i.e., Hallway, Hallway2 and Poc-Man.

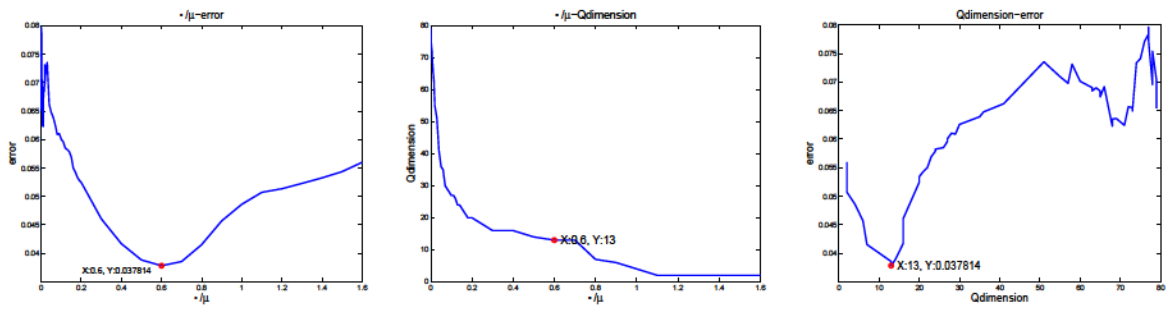
The results are summarized in Figs. 7, 8, and 9. In each figure for each domain, we show the effect of λ on the average prediction error, that of on the number of core tests that discovered, and the relationship between the number of core tests and the average prediction error, which corresponds to the (a), (b) and (c) part of the figure, respectively. The average prediction error (y-axis) in Figs. 7, 8, and 9 is computed by Eq. (10). The results are all calculated on 1000 test trials.



(a) error curve (b) dimension curve (c) dimension-error curve

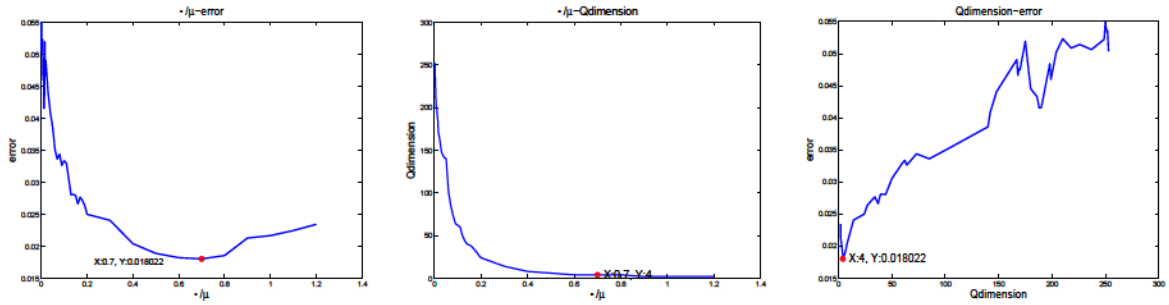
Fig. 7. Parameter sensitivity analysis of Hallway domain

Figs. 7, 8, and 9 show that the best performance of our learnt PSR model on those three domains can be obtained by adjusting the parameter λ, which are marked as red points. From that, we can get the minimum prediction error and its corresponding value of λ and the number of core tests. Figs. 7(a),8(a) and 9(a) show that the value of parameter λ does not affect the prediction error significantly, and its value in certain range could be acceptable for Hallway, Hallway2 and Poc-Man domains. Moreover, Figs. 7(c), 8(c) and 9(c) illustrate that a suitable number of core tests are needed for representing each domain to keep the prediction error low. If it is too small, then the information for state representation may not be enough; otherwise, the noise and irrelevant information may be included. For both cases,



(a) error curve (b) dimension curve (c) dimension-error curve

Fig. 8. Parameter sensitivity analysis of Hallway2 domain



(a) error curve (b) dimension curve (c) dimension-error curve

Fig. 9. Parameter sensitivity analysis of Poc-Man domain

the prediction errors will be increased. Notice that the fluctuations in Fig. 9 may be caused by the complexity of Poc-Man domain and the noise in the sampling progress.

4.3. Results

In this part, we present the numerical results in terms of the model accuracy and the algorithm runtime for each domain. Each result is obtained by running 1,000 test trials and computing the average performance. For each domain, we use the value of parameter λ that achieve the lowest average prediction error presented in Section 4.2, i.e., the red points showed in Figs. 7(a), 8(a) and 9(a). Accordingly, the number of core tests used to represent the states can be obtained, i.e., the y-axis of red points in Figs. 7(b), 8(b) and 9(b), which is 47, 13, and 4 for Hallway, Hallway2 and Poc-Man, respectively. For a fair comparison, we set the number of compressed dimension used by the TPSR algorithm to be the model dimension that learnt by our algorithm.

Fig. 10 compares the one-step prediction accuracy of the evaluated methods in the three different domains. In these figures, the x-axis is the step length of action-observations, the y-axis is the mean prediction error of 1,000 trials calculated by Eq. (11). As can be seen from Fig. 10, for almost all cases, our PSR-ADMM algorithm performs very well and produce more competitive predictions than other algorithms in all horizons of Hallway, Hallway2 and Poc-Man domains. In the Poc-Man domain, CPSR outperforms TPSR, and although it is a domain suitable for the CPSR approach, our approach is able to learn a more accurate

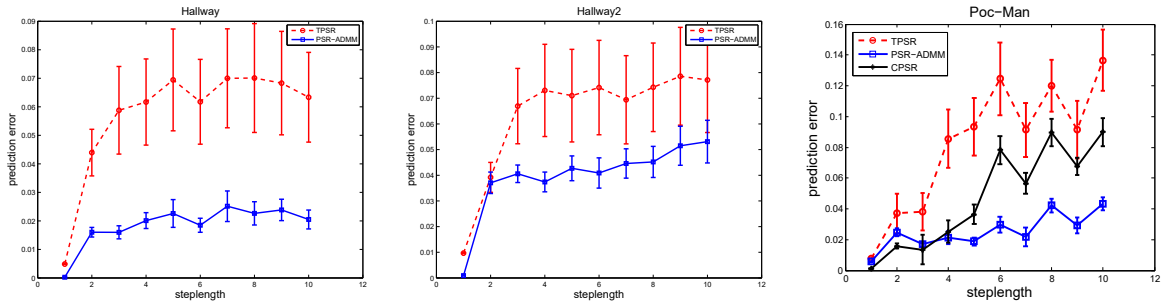
model.

Fig. 11 shows the runtime of each algorithm, which includes the time cost in estimating and building the model of dynamical system. As showed in Fig. 11, our approach significantly reduces the runtime on the three different domains, as compared to TPSR algorithm and CPSR algorithm. The runtime of the TPSR approach is prohibitively expensive because it requires a singular value decomposition (SVD) on a large-sized system dynamics matrix.

In summary, the good performance of our PSR-ADMM approach is partially due to the fact that

- The PSR model parameters can be obtained directly from the matrix X without extra computations.
- We do not need the specification of the size of core tests.
- The subproblems in alternating direction method of multipliers (ADMM) have closed-form solutions, and the method achieves a global convergence and converges fast as shown in numerical examples.
- We use the original information from system dynamics matrix as core tests.

However, the TPSR and CPSR methods require an input of the number for the size of core tests, which directly impacts the solution quality. Moreover, the core tests obtained by the TPSR and the CPSR methods do not use the original information of D (the information is compressed or extracted from SVD), which may also compromise the solution quality.

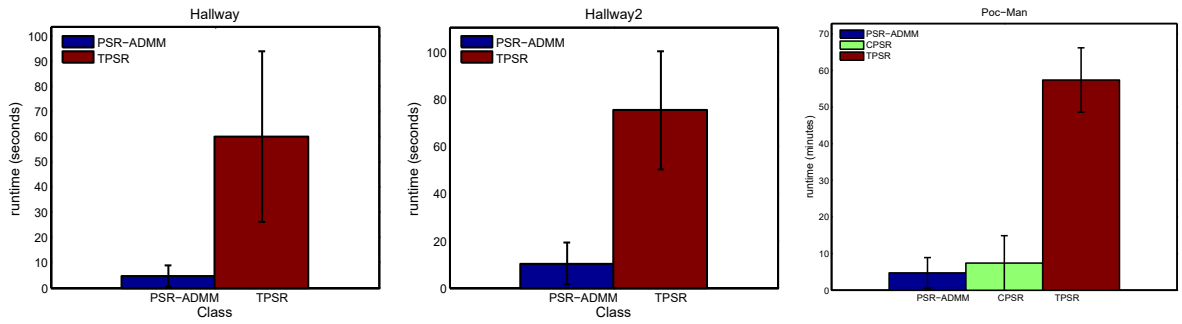


(a) Hallway (b) Hallway2 (c) Poc-Man

Fig. 10. Prediction errors in three different problem domains.

5. Related work

Littman et al. [19] proposed an algorithm based on sufficient training data for modeling the PSR model of the dynamical system. Specifically, this work is based on the assumption that the core tests are known, and then uses gradient descent method to learn from the training data for getting the PSR model. Later on, McCracken et al. [23] put forward constrained gradient descent method, thus the efficiency of the PSR model has been improved greatly, where core tests can be quickly discovered using very small amounts of data, and accuracy of predictions of the dynamical system can be improved as well when more data is



(a) Hallway (b) Hallway2 (c) Poc-Man

Fig. 11. Runtime in building the models for three different domains.

available. James et al. [13] studied a special class of controlled dynamical systems that have a reset operation and provided the first discovery algorithm and a new learning algorithm for PSRs. Moreover, James et al. [14] proposed a model called memory-PSRs (that uses both memories of the past, and predictions of the future) and also detected landmarks. The detection and recognition of landmarks is advantageous because they can serve to reset a model that has gotten off-track, which happens usually when the model is learned from samples. However, the dynamical systems cannot be reset in practice. For this reason, some research efforts were dedicated to non-resettable dynamical systems, such as suffix-history algorithm [29] and learning PSRs from a single sequence (i.e., history) [28].

Inspired by the basic theory of PSR models, researchers have developed a number of derivative models, such as spectral learning approach [2, 3, 17, 18, 25], compressed sensing approach [10, 11], etc. Among these models, researchers often addressed the discovery problem by specifying a large enough set of tests that contains a sufficient subset for state representation. Rosencrantz et al. [25] proposed an algorithm called transformed PSR (TPSR), which helps to alleviate the discovery problem and learns the parameters of TPSR efficiently based on principal components analysis (PCA). In a recent few years, some variants of TPSR were proposed. William et al. [10, 11] presented compressed transformed PSRs algorithm for a relatively large domain with a particularly sparse structure. Compared to TPSR, CPSR allows for an increase in the efficiency and predictive power. Furthermore, Kulesza et al. [17] introduced a TPSR-based model with a weighted loss function to overcome the consequence of discarding arbitrarily small singular values of the system dynamics matrix. They showed that the algorithm can effectively reduce the prediction error within the error bounds; however, it required the training data to be sufficiently sampled. Kulesza et al. [18] also did a research on data inadequate sampling situation and the corresponding algorithm ensures the accuracy of the PSR models in both theory and practice, which significantly reduces prediction errors compared to standard spectral learning approaches. On the other side, there exist other kind of approaches for learning PSR models. Liu et al. [21] employed the landmark technique to partition the entire state space, and learnt each separate sub-state space. The learnt models are combined finally to represent the entire space. More recently, Liu et al. [22] formulated the discovery problem as a sequential decision making problem, which can be solved using Monte-carlo tree search (MCTS). Thus the two main problems in learning PSR models can be solved thoroughly.

6. Conclusion and Future Work

In this paper, by utilizing the tool of linear algebra, we formulate an optimization problem for discovering the set of core test. With the benefit of group sparsity structure of coefficient matrix X, we update the parameters of PSR models as well. Hence, the discovery problem and learning problems are solved simultaneously and the model of the underlying system can be built straightforwardly. Due to the difficulties in solving the $L_{2,0}$-norm optimization problem, we consider its relaxation form and propose a group Lasso problem, and then employ the ADMM method, where the global convergence is guaranteed. Experimental results show that our algorithm is capable of learning PSR models in a large and complex domain. As shown in the experiments, our method is significantly improved as compared to two other popular methods, i.e., TPSR and CPSR. Moreover, it is worth mentioning that we do not need an input to specify the number of core tests, as it is automatically decided by the optimization process. In the future, we may develop more effective techniques to solve this optimization problem, and explore other ways for learning the PSR models, e.g., incremental learning by partitioning the system dynamics matrix D, taking advantage of the sparsity of matrix D or adding the sparsity constraint on X accordingly.

Acknowledgments

The authors would like to thank Dr. Zhening Li for useful discussions. This work was supported in part by the National Natural Science Foundation of China (Grants 11301436, 11671335, 61375077 and 61375070).

References

[1] A. Bhattacharyya, V. Nair, An explicit sparse recovery scheme in the L1-norm. arXiv preprint arXiv:1411.2344, 2014.

[2] B. Boots, G. J. Gordon, An online spectral learning algorithm for partially observable dynamical systems, AAAI'11 Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, 2011, pp.293–300.

[3] B. Boots, S. M. Siddiqi, G. J. Gordon, Closing the learning-planning loop with predictive state representations, The International Journal of Robotics Research, 30(7) (2011) 954–966.

[4] S. Boyd, N. Parikh, E. Chu, B. Peleato: and J. Eckstein, Distributed optimization and statistical learning via the alternating direction method of multipliers, Foundations and Trends in Machine Learning, 3(1) (2011) 1–122.

[5] S. Boyd, L. Vandenberghe, Convex Optimization, Cambridge University Press, 2004.

[6] E.J. Candès, Y. Plan, Near-ideal model selection by ℓ_1 minimization, The Annals of Statistics, 37 (2009) 2145–2177.

[7] E.J. Candès, M. Wakin, S. Boyd, Enhancing sparsity by reweighted ℓ_1 minimization, Journal of Fourier analysis and applications, 14 (2008) 877–905.

[8] A. R. Cassandra, Exact and Approxiamte Algorithm for Partially Observable Markov Decision Processess, Ph.D.Thesis, Brown University, 1998.

[9] M. Grant , S. Boyd,Y. Ye, CVX: Matlab software for disciplined convex programming, version 2.0 beta. http://cvxr.com/cvx, September 2013.

[10] W.L. Hamilton, M.M. Fard, J. Pineau, Modelling sparse dynamical system with compressed predictive state representations, In Proceedings of the 30th International Conference on Machine learning, 28 (2013) 178–186.

[11] W.L. Hamilton, M.M. Fard, J. Pineau, Effcient learning and planning with compressed predictive states, Journal of Machine learning Research, 15 (2014) 3395–3439.

[12] P. Indyk, M. Ruzic, Near-optimal sparse recovery in the ℓ_1 norm, Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on. IEEE, 2008, pp.199–207.

[13] M. R. James, S. Singh, Learning and discovery of predictive state representations in dynamical systems with reset, In Proceedings of the 21st International Conference on Machine Learning, Banff, Alberta, Canada, 2004, pp.695-702.

[14] M. R. James, B. Wolfe, S. Singh, Combining memory and landmarks with predictive state representations, In Proceedings of the International Joint Conference on Artificial Intelligence(IJCAI), 2005, pp.734-739.

[15] L.P. Kaelbling, M.L. Littman, A.R. Cassandra, Planning and acting in partially observable stochastic domains, Artificial intelligence, 101(1) (1998) 99-134.

[16] M.A. Khajehnejad, W.Xu, A.S.Avestimehr, B. Hassibi, Analyzing weighted ℓ_1 minimization for sparse recovery with nonuniform sparse models, IEEE Transactions on Signal Processing, 59(5) (2011) 1985-2001.

[17] A. Kulesza, N. Jiang, S. Singh, Low-Rank spectral learning with weighted loss functions, In Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS), San Diego, CA, USA, JMLR:W&CP, 38 (2015) 517–525.

[18] A. Kulesza, N. Jiang, S. Singh, Spectral learning of predictive state representations with insufficient statistics, AAAI'15 Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI Press, 2015, pp.2715–2721.

[19] M. L. Littman, R.S. Sutton,S. P. Singh, Predictive representations of state. In T. G. Dietterich, S. Becker, & Z. Ghahramani (Eds.), Advances in Neural Information Processing Systems (NIPS), 14(2001) 1555–1561.

[20] J. Liu, S. Ji,J. Ye, SLEP: Sparse Learning with Efficient Projections, Arizona State University, 2009. http://www.public.asu.edu/jye02/Software/SLEP.

[21] Y. Liu, Y. Tang, Y. Zeng, Predictive state representations with state space partitioning, In Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2015, pp.1259–1266.

[22] Y. Liu, H. Zhu, Y. Zeng, Z. Dai, Learning predictive state representations via monte-carlo tree search, Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI), 2016, pp.3192-3198.

[23] P. McCracken, M. Bowling,Online discovery and learning of predictive state representations, Advances in Neural Information Processing Systems(NIPs), 18 (2005) 875–882.

[24] Z. Qin, K. Scheinberg, D. Goldfarb, Efficient block-coordinate descent algorithms for the group lasso, Mathematical Programming Computation, 5(2) (2013) 143-169.

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

[26] D. Silver, J. Veness, Monte-carlo planning in large pomdps, In Advances in Neural Information Processing Systems, 23 (2010) 2164-2172.

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

[28] E. Wiewiora, Learning predictive representations from a history, In Proceedings of the Twenty-Second International Conference on Machine Learning, Bonn, Germany, 2005, pp.964-971.

[29] B. Wolfe, M. 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), 2005, pp.980–987.

[30] E. Van den Berg, M. Schmidt, M. P. Friedlander, and K. Murphy, Group sparsity via linear-time projection, Dept. Comput. Sci., Univ. British Columbia, Vancouver, BC, Canada, 2008.

[31] M. Yuan,Y. Lin, Model selection and estimation in regression with grouped variables, Journal of the Royal Statistical Society, Series B., 68(1) (2006) 49–67.