Abstract—Predictive State Representations (PSRs) are dynamical systems models that keep track of the system’s state using predictions of future observations. In contrast to other models of dynamical systems, such as partially observable Markov decision processes, PSRs produce more compact models and can be consistently learned using statistics of the execution trace and spectral decomposition. In this paper we make a connection between rank minimization problems and learning PSRs. This allows us to derive a new algorithm based on nuclear norm minimization. In addition to estimate automatically the dimension of the system, our algorithm compares favorably with the state of the art on randomly generated realistic problems of different sizes.

I. INTRODUCTION

In machine learning and artificial intelligence, planning a sequence of actions to maximize future rewards in a discrete-time partially observable nonlinear dynamical system is a fundamental problem. Such systems are often represented by latent variable models called Hidden Markov Models (HMMs) or, in the controlled setting, Partially Observable Markov Decision Process (POMDP). In these models, the discrete hidden state of the system, that evolves over time in a Markovian way, is sensed through noisy incomplete observations. Inference and learning in POMDPs is a difficult task and planning is possible only for very small models due to the large number of parameters and latent variables. Existing methods, like Expectation Maximization [1] or Gibbs sampling, failed to learn an accurate model, being prone to get stuck in local minima and do not scale to large state spaces. Indeed, latent variable models keep track of a distribution over hidden states through what is called the belief and stands for an observed variable models keep track of a distribution over hidden states, PSRs use a compact representation [3] of the state. Predictive State Representations (PSRs) [2] are compact representations that generalize POMDPs. Unlike POMDPs for which the belief scales exponentially with the number of hidden states, PSRs use a compact representation [3] of the current state by maintaining occurrence probabilities of a set of future events, called tests, conditioned on past events, called histories. In addition, as they deal with observable quantities, learning and inference is easier than for POMDPs. Planning in PSRs can be done using algorithms from POMDPs with very small changes [4], [5]. In this paper we focus on linear PSRs. Linear PSRs of dimension $d$ are less expressive than the nonlinear ones but can still model any POMDPs with at most $d$ hidden states.

Recently, a new algorithm [4] has been proposed to learn PSRs taking advantage of their spectral properties to identify a low dimensional subspace in which the system dynamics lie. Basically, the low dimensional subspace is identified using the thin singular value decomposition (SVD) of an estimated correlation matrix between past and future events. Then, as the PSR is linear, its parameters can be learned by linear regression between estimated correlation matrices projected on this subspace. This spectral learning has been shown to be consistent. In addition, [6]–[8] provided finite sample bounds on the learned joint and conditional probabilities when the observations are generated by an HMM. However, this algorithm has some drawbacks. First, in practice, one has to rely on regularization during the regression step to reduce the effect of imperfect estimation of correlation matrices. Secondly, one needs to provide the algorithm with the true dimension of the system. When the true dimension of the system is unknown, consistency and bounds do not hold. Underestimation of the dimension can map a subspace too small to explain the dynamics [9]. On the other hand, overestimation of the dimension leads to a less compact representation that is more sensitive to estimation noise. Estimating correctly the dimension can be formulated as a rank minimization problem.

The matrix rank minimization problem has recently attracted much renewed interest as this problem arises in many engineering and statistical modeling applications [10], [11]. For example, rank minimization appears in minimum order linear system identification [12]–[15], low-rank matrix completion, low-dimensional euclidean embedding problems and image compression. In all these problems, the complexity of a model can be expressed by the rank of an appropriate matrix. Thus, resolution relies on finding the lowest rank matrix that is consistent with observations, where the consistency is formulated as convex constraints. In general, rank minimization is NP-hard, and one has to rely on heuristics using convex relaxations [15]. A popular one minimizes the nuclear norm of the matrix, which is the sum of the singular values, instead of its rank. Indeed, since the singular values of a matrix are all positive, the nuclear norm is equal to the $l_1$ norm of the vector formed from the singular values. As it has been shown for compressed sampling [16] and sensing [17], the $l_1$ heuristic provably recovers the sparsest solution under soft assumptions. As a result, minimizing the nuclear norm will lead to sparsity...
in the vector of singular values, or equivalently to a low-rank
matrix.

In this paper, we propose a new algorithm 1) able to estimate
the rank of the system depending on noise in estimators, 2)
less sensitive to the regularization parameter 3) producing
better performance. Our algorithm is formulated as a rank
minimization problem which is approximately solved by nu-
clear norm minimization (NNM). In addition to propose a new
algorithm, our contribution is twofold. First we provide, for
the first time, an experimental analysis on how regularization in
the regression and the size of the low dimensional subspace
impacts performances. Second, we cast the learning PSRs
problem to a rank minimization one that is extensively studied
in the literature. First, some related work on PSR learning and
NNM for system identification are reviewed. Next, we intro-
duce Transformed PSRs and the spectral learning algorithm
it comes with. The NNM problem is also defined. Then, in
section III, we show how our algorithm makes use of NNM to
learn Transformed PSRs. Section IV presents the experimental
design. The results are shown in section V. Finally, sections VI
and VIII conclude and propose directions for future researches.

II. BACKGROUND

A. Linear Predictive State Representation

A predictive state is a sufficient statistic of past events
to predict the future. Let \( \mathcal{A} \) be the discrete set of actions
and \( \mathcal{O} \) the discrete set of observations, an history \( h_t \) :=
\( (a_0, a_1, o_0, a_2, o_2, ..., a_t, o_t) \) is a succession of taken actions and
observations received since system started up to time \( t \). We
denote by \( \mathbb{P}(o_{t+1}|a_{t+1}, h_t) \) the probability of observing \( o_{t+1} \)
after taken action \( a_{t+1} \) and observing history \( h_t \). We call a
sequence \( h_{\tau} := (a_0, o_0, a_1, o_1, a_2, o_2, ..., a_{\tau}, o_{\tau}) \) of size \( p \)
a succession of \( p \) action-observation pairs in the future. We
denote by \( \tau^A \) the sequence of actions of \( \tau \) and \( \tau^O \) the sequence
of observations.

PSRs have the property that any future can be predicted
from the occurrence probability of a small set of tests, called
the core tests. In the following we denote by \( \mathcal{Q} \) a set of
core tests. The current predictive state, denoted by \( \mathbf{m}_t \), can be
written as a vector of conditional probabilities of the tests in
\( \mathcal{Q} := \{ \tau_1, ..., \tau_{|Q|} \} \) given the current history \( h_t \) :

\[
\mathbf{m}_t := [\mathbb{P}(\tau^O_i | \tau^A_j, h_t)]_{\tau_i \in \mathcal{Q}}.
\]

More generally, a set of tests \( \mathcal{T} \) is core if and only if we can
define a predictive state,

\[
\mathbf{m}_t := [\mathbb{P}(\tau^O_i | \tau^A_j, h_t)]_{\tau_i \in \mathcal{T}},
\]

such that for any test \( \tau \), there exists \( f_{\tau} \) such that
\( \mathbb{P}(\tau^O | \tau^A, h_t) = f_{\tau}(\mathbf{m}_t^\top) \). In this paper, we restrict ourselves
to linear PSRs. Thus, we assume that for any test \( \tau \) there exists
a vector \( \mathbf{r}^\tau_i \in \mathbb{R}(|\mathcal{Q}|) \) such that,

\[
\mathbb{P}(\tau^O_i | \tau^A_j, h_t) = f_{\tau}^i(\mathbf{m}_t^\top).
\]

Let \( \mathcal{M}_\mathcal{O} := [\mathbf{r}^\tau_{a_{\tau}, o_{\tau}}] \) be the matrix built by stacking the row
vectors \( \mathbf{r}^\tau_{a_{\tau}, o_{\tau}} \), for each test in the core set, then we have

\[
[\mathbb{P}(\tau^O_i | \tau^A_j, h_t)]_{\tau_i \in \mathcal{Q}} = \mathcal{M}_\mathcal{O}\mathbf{m}_t.
\]

Let \( \mathbf{m}_t^\top \) a normalization vector such that \( \forall i \mathbf{m}_t^\top \mathbf{m}_t = 1 \), then
prediction of the next observation given the next action can be
done by,

\[
\mathbb{P}(o|a, h_t) = \mathbf{m}_t^\top \mathbb{P}(\tau^O_i | \tau^A_j, h_t)_{\tau_i \in \mathcal{Q}} = \mathbf{m}_t^\top \mathcal{M}_\mathcal{O}\mathbf{m}_t,
\]

where the normalization vector is marginalizing out the prob-
ability of the state from the joint probability \( \mathbb{P}(\tau^O_i | \tau^A_j, h_t) \).

We can now perform a state update through Bayesian filtering.
After executing action \( a \) and observing \( o \), we have,

\[
\mathbf{m}_{t+1} = \left[ \mathbb{P}(\tau^O_i | \tau^A_j, h_{t+1}) \right]_{\tau_i \in \mathcal{Q}} = \left[ \mathbb{P}(\tau^O_i | \tau^A_j, (h_t, a, o)) \right]_{\tau_i \in \mathcal{Q}} = \left[ \mathbb{P}((a|\tau^O_i)(\tau^A_j), h_t) \right]_{\tau_i \in \mathcal{Q}} = \left[ \mathbb{P}((a|\tau^O_i)(\tau^A_j), h_t) \right]_{\tau_i \in \mathcal{Q}} = \left[ \mathbb{P}(o|a, h_t) \right]_{\tau_i \in \mathcal{Q}} = \mathbf{m}_t^\top \mathbb{P}(o|a, h_t),
\]

\[
\mathbf{m}_{t+1} = \mathcal{M}_\mathcal{O}\mathbf{m}_t.
\]

We denote by \( \mathbf{m}_1 \) the initial predictive state. The vectors
\( \mathbf{m}_t \), \( \mathbf{m}_\infty \) and the matrices \( \mathcal{M}_\mathcal{O} \) define the PSR parameters.
Finally, we say that a core set is minimal if all the tests
it contains are linearly independent. This means that the
occurrence probability of each test cannot be written as a linear
combination of probabilities of other tests in the set. In the
sequent, \( \mathcal{Q} \) stands for a minimal core set.

B. Learning Transformed PSR

The learning problem can be decomposed into two parts.
The discovery problem aims at finding a minimal core set of
tests, whereas the estimation problem is concerned by
estimating the PSR parameters. Note that predicting (1) and
filtering (2) equations are invariant to any linear invertible
transformation \( J \) of the parameters. Let,

\[
b_\infty := J\mathbf{m}_\infty, B_{ao} := JM_{ao}J^{-1}, b_1 := J\mathbf{m}_1,
\]

we still have that,

\[
\mathbb{P}(o|a, h_t) = \mathbf{b}_\infty^\top B_{ao}\mathbf{b}_t,
\]

\[
b_{t+1} = B_{ao}b_t.
\]

These new parameters defined a transformed version of the
original PSR, we refer to as the Transformed PSR (TPSR [18]).
To solve the discovery problem, the idea behind spectral
learning is to start with a huge set of tests \( \mathcal{T} \) such that it
is core; then by spectral decomposition to recover the PSR
parameters up to a linear transformation without building
explicitly the minimal core set of tests. In practice, the set \( \mathcal{T} \)
is constructed from all fixed size sequence of action-observation
pairs appearing in the learning trajectory. We define also the set
of all histories \( \mathcal{H} \), constructed in the same way. In [4], TPSR
parameters can be recovered from observable statistics. Indeed,
let \( \mathcal{P}_{\mathcal{T}, \mathcal{H}} \in \mathbb{R}(|\mathcal{T}| \times |\mathcal{H}|) \) be the vector of occurrence probabilities of
histories \( \mathcal{H} \). Similarly, we define \( \mathcal{P}_{\mathcal{T}, \mathcal{H}} \in \mathbb{R}(|\mathcal{T}| \times |\mathcal{H}|) \) to be
the matrix containing the joint occurrence probabilities of an history followed by a test for all tests and histories in $T \times H$. Finally, for all actions and observations $(a, o) \in A \times O$, let $P_{T,ao,\mathcal{H}} \in \mathbb{R}^{T \times |H|}$ be the joint probabilities’ matrices of a test occurring after an history just followed by the action-observation pair $ao$. To summarize, we have,

$$
P_{\mathcal{H}} = [P(h_i)]_i, \quad P_{\mathcal{T,\mathcal{H}}_i} = [P(\tau, h_i)]_{i,j}, \quad \forall a, o \quad P_{T,ao,\mathcal{H}} = [P((aoT), h_j)]_{i,j}.
$$

In [4], authors proved that $P_{T,ao,\mathcal{H}}$ and $P_{T,ao,\mathcal{H}}$ have rank at most $d := |\Omega|$. In addition, they showed that if,

$$
U, S, V = SD_k(\hat{P}_{T,\mathcal{H}}),
$$

where $SVD_k$ is the thin SVD, in a way that $\hat{U}$ contains only the associated columns to the $k$ largest singular values, then, $J = U^T R$ with $R = [r_{i,j}^T]$ is an invertible linear transformation. In addition, noting by $X^T$ the Moore pseudo-inverse of a matrix $X$, we have,

$$
b_1 = U^T P_{T,\mathcal{H}} e, \quad b_2^i = \hat{P}_{H}(U^T P_{T,\mathcal{H}})^i, \quad \forall ao \quad B_{ao} = U^T P_{T,ao,\mathcal{H}}(U^T P_{T,\mathcal{H}})^i,
$$

where $e$ is a vector with all coordinates equals to 0 but the $i$-th one equals to 1 where $i$ stands for the place of the empty history in $\mathcal{H}$. If we want to take for the initial belief, the stationary distribution of the system, we can change $e$ to be $1$. This means that the initial state would correspond to the averaged history encountered in the training set.

The learning algorithm works by building empirical estimates $\hat{P}_{\mathcal{H}}, \hat{P}_{T,\mathcal{H}}$ and $\hat{P}_{T,ao,\mathcal{H}}$ of the matrices $P_{\mathcal{H}}, P_{T,\mathcal{H}}$ and $P_{T,ao,\mathcal{H}}$. The best way to build these estimates is to repeatedly sample an history $h$ according to the same distribution $\omega$ and then execute a sequence of actions and record the resulting observations. For instance, $\omega$ can be the distribution of histories resulting from the execution of a fixed exploration policy after a reset. So, to gather such data, the system must allow to be reset in a state distributed as $\omega$. When reset is not available, estimates can be approximately built from a single long trajectory by dividing it into subsequences. This approach called the suffix-algorithm [20], produces still good estimates in practice. Once the estimates $\hat{P}_{\mathcal{H}}, \hat{P}_{T,\mathcal{H}}, \hat{P}_{T,ao,\mathcal{H}}$ are computed, provided with an estimation $k$ of the true rank $d$, we can find the low dimension subspace by plugging the estimate $\hat{P}_{T,\mathcal{H}}$ into equation (3),

$$
\hat{U}, \hat{S}, \hat{V} = SVD_k(\hat{P}_{T,\mathcal{H}}).
$$

Finally, by left multiplying the estimates by $\hat{U}^T$, we project them on the identified subspace and the TPSR parameters can be recovered by linear regression. During this last step, Tikhonov regularization with parameter $\lambda$ improves the robustness to estimation errors involved in the first step. This final step is sum up through the following computations:

$$
b_1 = \hat{U}^T \hat{P}_{T,\mathcal{H}} e, \quad b_2^\infty = \hat{P}_{H}(\hat{U}^T \hat{P}_{T,\mathcal{H}})^{\lambda}, \quad \forall ao \quad \hat{B}_{ao} = \hat{U}^T \hat{P}_{T,ao,\mathcal{H}}(\hat{U}^T \hat{P}_{T,\mathcal{H}})^{\lambda},
$$

where $X^{\lambda} = (X^TX + \lambda I)^{-1}X^T$ is the regularized Moore pseudo-inverse.

C. Low Rank Matrix Denoising by Nuclear Norm Minimization

Let $\mathbb{R}^{n \times m}$ be the space of $n \times m$ matrices endowed with the standard trace inner product $(X, Y) = \text{tr}(X^TY)$. We denote by $\|X\|_F$ and $\|X\|_2$, the Frobenius norm and the spectral norm respectively of $X$. Let $\sigma_i(X)$ be the $i$-th largest singular value of $X$, the nuclear norm of $X$ is defined to be $\|X\|_\nu = \sum_{i=1}^{\min\{m,n\}} \sigma_i(X)$. Assume that $\hat{X} \in \mathbb{R}^{n \times m}$ is a low rank matrix perturbed by a small noise, i.e. $\hat{X} = X + E$ with $\|E\|_F \leq \delta$. We formulate the low rank matrix denoising problem as,

$$
\min_{X \in \mathbb{R}^{n \times m}} \|X\|_\nu
$$

subject to $\|X - \hat{X}\|_F \leq \delta$. \hspace{1cm} (P_1)

As mentioned in the introduction solving directly this problem is NP-hard. Minimizing the nuclear norm provides an interesting alternative as an heuristic for low-rank approximation problems that cannot be handled via SVD. In particular, NNM aimed approximation problems with structured low-rank matrices and problems that include additional constraints or objectives. This surrogate problem can be written as

$$
\min_{X \in \mathbb{R}^{n \times m}} \|X\|_\nu
$$

subject to $\|X - \hat{X}\|_F \leq \delta$. \hspace{1cm} (P_2)

In some sense, (P_2) is the tightest convex relaxation [21] of the NP-hard rank minimization problem. Here we focus on the closely related problem, which highlights better the trade off between ensuring that $X$ is low rank and sticking to observations $\hat{X}$,

$$
\min_{X \in \mathbb{R}^{n \times m}} \mu\|X\|_\nu + \frac{1}{2}\|X - \hat{X}\|_F^2. \hspace{1cm} (P_3)
$$

for a given $\mu$. In fact, (P_3) is equivalent to (P_2) for some $\delta$ unknown a priori. Now, we present a closed form solution to (P_3). Consider the SVD of the matrix $X$ of rank $r$,

$$
X = USV^T, \quad \Sigma = \text{diag}(\{\sigma_i\}_{1 \leq i \leq r}).
$$

For each $\tau \geq 0$, we introduce the singular value thresholding (SVT) operator $D_\tau$ defined as follows,

$$
D_\tau(X) = UD_\tau(\Sigma)V^T, \quad D_\tau(\Sigma) = \text{diag}(\{\sigma_i - \tau\}_+),
$$

where $x_+ = \max(0, x)$. It has been shown [21] that $D_\mu(X)$ achieves the minimum in ($P_3$). So,

$$
D_\mu(X) = \min_{X \in \mathbb{R}^{n \times m}} \mu\|X\|_\nu + \frac{1}{2}\|X - \hat{X}\|_F^2. \hspace{1cm} (P_4)
$$

In contrast, the truncation of the singular values in the SVD, as it’s done in the algorithm presented in section II, achieves the following minimum,

$$
US_kV^T = \min_{X \in \mathbb{R}^{n \times k}} \|X - \hat{X}\|_F
$$

subject to rank$(X) = k$, where $S_k \in \mathbb{R}^{n \times r}$ is a diagonal matrix containing the $k$ largest singular values of $X$. 


III. LEARNING TRANSFORMED PSR BY THRESHOLDING SINGULAR VALUES

Identifying a low dimensional subspace from the noisy estimates of the correlation matrices is a critical step in learning TPSR. Lower is the dimension, more compact the model will be but one has to account for the quantity of the noise in order to select an appropriate size for the subspace. If errors in the estimations are large, searching for a small subspace can lead to a big approximation error, defined as the distance between the true subspace and the approximate one. Indeed, some of the dimensions can be used to model the noise instead of the system dynamics. In this case, it is better to look for a bigger subspace and map both the system dynamics and a part of the noise, because this noise can be next reduced during regression by means of regularization. When errors in estimates are small, one can directly try to find the lowest rank estimates of the correlation matrices by stacking all the correlation matrices as follow,

\[
\begin{pmatrix}
\hat{P}^T_H \\
\hat{P}_{T,H} \\
\hat{P}_{T,a_0i,1,H} \\
\vdots \\
\hat{P}_{T,a_0i,A\vert O\vert,H}
\end{pmatrix}
\]

Notice that \( \text{rank}(\hat{F}) = \text{rank}(\hat{P}_{T,H}) = d \) as the PSR is linear (see equations (4), (5), (6)). So, \( \hat{F} \) is supposed to be low rank. Next, by solving problem \((P_3)\) for \( \hat{F} \), we obtain low rank estimates \( \hat{P}^T_H, \hat{P}_{T,H} \) and for all actions and observations \( \hat{P}_{T,a_0i,H} \) of the correlation matrices \( \hat{P}^T_H, \hat{P}_{T,H} \) and \( \hat{P}_{T,a_0i,H} \) that lie into a low dimensional space and are close to the empirical estimates \((\hat{P}^T_H, \hat{P}_{T,H}, \hat{P}_{T,a_0i,H})\).

\[
\begin{pmatrix}
\hat{P}^T_H \\
\hat{P}_{T,H} \\
\hat{P}_{T,a_0i,1,H} \\
\vdots \\
\hat{P}_{T,a_0i,A\vert O\vert,H}
\end{pmatrix} = D\mu(\hat{F}),
\]

Finally, these low rank estimates are plugged into the equations (3), (4), (5) and (6), to obtain the TPSR parameters,

\[
\begin{align*}
\hat{U}, \hat{S}, \hat{V} &= \text{SVD}_1(\hat{P}_{T,H}), \\
\hat{b}_1 &= \hat{P}_{T,H}e, \\
\hat{b}_1^\infty &= \hat{P}_{T,H}^U(\hat{U}\hat{P}_{T,H})^{\lambda_t}, \\
\forall a_o \quad \hat{B}_{ao} &= \hat{U}\hat{P}_{T,a_0o,H}(\hat{U}\hat{P}_{T,H})^{\lambda_t},
\end{align*}
\]

where \( t \) is the index of the smallest positive singular value of \( \hat{P}_{T,H} \). Note that our algorithm insert himself into the one presented in section II. By denoising the estimates using their low rank property, it changes the identified subspace but then the main steps remain unchanged. Thus, our algorithm, by improving the subspace identification, leads to better performances while maintaining all the nice properties of the algorithm in section II. Thus, it offers a consistent learning of a model capable of dealing with the curse of dimensionality, by compactly modeling states, and the curse of history, by tracking states that can keep a trace of old past events.

IV. EXPERIMENTAL DESIGN

We choose to show the benefits of our approach on the restricted setting of POMDPs, as they are easier to generate than PSRs and simplifies the discussion. The same conclusions could be drawn for PSRs. In our experiments, as we have more observations than states and as the observation matrix is full rank, tests and histories consist of single observation. However, the presented algorithms could also handle more difficult cases simply by using longer tests and histories, as PSRs do. As a POMDP with a given policy reduce to an HMM, algorithms are directly evaluated on HMMs. Recall [6] that the rank of \( \hat{P}_{T,H} \) equals the one of the transition matrix for HMMs. This property allows us to compare with the algorithm that uses a truncation of the SVD with the true rank.

In our experiments we refer to the algorithm described in section II where the (usually unknown) rank of \( T \) is used to find the low dimensional subspace as SVD(d). We refer to our algorithm as SVT. Finally, we evaluated the algorithm of section II but using the rank estimated from SVT instead of the true rank. This algorithm is mentioned as SVD(l). These three algorithms are evaluated on randomly generated HMMs. To approach realistic settings, we generate transition and observation matrices with a low branching factor, i.e. each state can transit to (resp. generate) only a small subset of the states (resp. observations). The probabilities are then drawn uniformly. To draw a transition matrix with rank \( r \), we only generate \( r \) of its columns and then randomly duplicate them to get a square matrix. We choose to evaluate the three algorithms on two kind of HMMs qualified in the next by small and big. The small kind has 5 states, 10 observations and the transition matrix is full rank. Branching factor for small HMMs is 3 for transitions and observations. Big HMMs have 20 states, 25 observations and the rank of the transition matrix is 10. The branching factor is 5 for transitions and for observations. We evaluate the algorithms for different size of the training set. We set \( \mu = \frac{s_{max}(\hat{F})}{100} \) in simulations. For each size, we draw 10 HMMs and for each HMMs, we draw 20 fixed sized sets of trajectories. Each trajectory is made of 20 steps. Then, for each simulation we try different values of the regularization parameter \( \lambda \) used in the regression.

V. RESULTS

A. Effect of the singular values thresholding

In this section, we would like to point out the difference between SVD(l) and SVT by taking an example. In figure 1, we draw the singular values of a set of matrices. SVD(l) and SVT differs in two points. First, in SVT, the singular values are soft-thresholded, whereas, in SVD(l), they are truncated. Second, in SVT, we work on the stacked matrix \( \hat{F} \) whereas,
in SVD(l), only \( \hat{P}_{T,H} \) is used. In figure 1, we show on an example that soft-thresholding the singular values of \( \hat{F} \) has a different effect than soft-thresholding the singular values of \( \hat{P}_{T,H} \) because the differences between singular values of \( \hat{P}_{T,H} \) and \( \hat{P}_{T,H} \) are not constant. Similar analysis can be conducted for \( \hat{P}_{H} \) and \( \hat{P}_{T,so,H} \). These subtle differences between SVD(l) and SVT lead to better performances for SVT.

**B. Rank estimation**

Figure 2 shows how the estimated rank decreases with the size of the training set for both kind of HMMs. For small training sets, the rank is overestimated, about twice its final value. Because the high noise is distributed in all dimensions, our algorithm has difficulties to find a correct approximation lying in a small dimensional subspace of the observed covariance matrices. However, the estimated rank converges quickly to his final value after a thousand trajectories. Recall that for small HMMs the true rank is 5. For the big one, we set the true rank to 10. Thus, our algorithm tends to slightly underestimate the true rank. In the next section, we show that it does not decrease the performance, even it improves it. As we kept the same value of \( \mu \) in both settings (small and big HMMs), we show our algorithm can correctly estimate the rank without the need to be tweaked for different problems.

**C. One step ahead prediction**

The quality of the learned TPSR is assessed by the mean square error of the predicted distribution over the next observation computed by the learned model relatively to the one of the HMM. The score is then averaged over each step and each trajectory. The test set consists of 200 trajectories, drawn at each simulation, of 20 steps. For each trajectory, the predictive state is tracked and the next observation is predicted. This is the predicted distribution that we use to compute the score. As all the algorithms produced parameters that are approximately the ones of a PSR, computed probabilities can sometimes be larger than 1 or negative. To deal with that issue we threshold the probability vector by zero and one and then normalize it, in order to get a distribution. Figures 3 and 4 show how the mean square error decreases with the training set size. We plot curves for different values of \( \lambda \) and another curve where \( \lambda \) is set to its best value for each training set size and for each

![Figure 1. Effect of the singular values thresholding](image_url)

![Figure 2. Change in estimated rank depending on the training set size.](image_url)
algorithm. The y-axis is log scaled for all curves but the one showing the mean square error for the best values of λ. In spite of we kept the same value of μ, for both sizes of HMMs, the results and the following analysis are similar.

First, we can analyze how the regularization in regression impacts performances. Globally, simulations show a bias-variance trade-off. When the training set is small, all algorithms have to rely on high values of λ to best remove the estimation noise. But, when the training set grows, regularization introduces a bias that decreases the performance. In the limit, removing the regularization is the best option.

Secondly, by comparing SVD(d) and SVD(l), we can see that the estimated rank, produces much better results for low values of λ when the training set is big and for high values of λ when the training set is small. When the value of λ is fitted to the training set size, the two algorithms produce similar results for medium-sized training set; whereas, for huge training set, using the true rank seems slightly better. So, with the estimated rank, one can use a lower value of λ to achieve the same performance. As regularization in regression introduces a bias, our analysis highlights the importance of adapting the rank to errors in estimates.

At last, we compare SVT to SVD(l) to understand differences in subspace identification between soft-thresholding the singular values of the stack matrix \( \hat{F} \) and truncating the singular values of \( \mathcal{P}_{\mathcal{T}, \mathcal{H}} \). Overall, SVT does a better job than SVD(l), excepting for the combination of a big regularization and big training set, which is by definition and experimentally a bad setting. In other words, increasing regularization works better for SVT when the training set is small. For big training set, increasing regularization tends to fade and even invert the superiority of SVT over SVD(l).

VI. DISCUSSION

In this paper, we reformulated the PSRs learning problem as rank minimization. The low rank and linear properties allow our algorithm to fade out the noise due to sampling in the estimates \( \mathcal{P}_{\mathcal{T}, \mathcal{H}} \), \( \mathcal{P}_{\mathcal{T}, \mathcal{H}} \), and \( \mathcal{P}_{\mathcal{T}, \mathcal{H}} \). Indeed, as the noise is distributed in all dimensions, correctly identifying a low dimensional subspace helps to remove the noise orthogonal to the system dynamics. The estimation noise that lies along the system dynamics in the low dimensional subspace still must be latter remove using regularized regression. If the low dimensional subspace is not correctly identified, then system dynamics can lie outside of it, making the TPSR parameters hard to recover. In the experiments, we pointed out two advantages of SVT over the previous algorithm to learn TPSR.

First, it adapts the dimension of the model to the noise in estimates. Thus, when the training set is small, the noise is big and the estimated rank is high. This avoids the bad scenario where the system dynamics lie outside the identified subspace. Of course, this comes with the drawback of having a less compact model and keeping more noise in the projected estimates. However, this noise can be efficiently removed by regularization during regression. This explains why on figure 4 and 3, in the small training set case, SVT can do better than SVD(d) if the regularization parameter is big enough (\( \lambda \in \{10^{-4}, 10^{-5}\} \)), as SVT keeps in its low dimensional subspace the system dynamics.

Secondly, using all estimates stacked in the matrix \( \hat{F} \), helps to find a closer approximated low dimensional subspace to the true one. Using NNM can take advantage of the linear property that holds between the estimates. As a result, the set of admissible subspaces is more constrained by the linearity and by the fact that all estimates, and not only \( \mathcal{P}_{\mathcal{T}, \mathcal{H}} \), have to be close to their values once projected.

Finally, the experiments show that the first advantage we just pointed out is mitigated because, even if regularization in SVT performs well to remove noise when the rank is overestimated, it introduce a bias. This is why, when \( \lambda \) is adjusted to training set size, differences between curves fade out. However, SVT still comes with a small improvement due to the second advantage pointed out. In addition, when training sets are large, SVT does not have to rely on regularization because it finds the smallest dimensional subspace as shown on the curves with \( \lambda = 0 \).

VII. RELATED WORK

The first work [18] that applies spectral decomposition of statistics on execution trace built estimates from conditional probabilities instead on the joint ones. This resulted in very noisy estimates that lead to poor practical performances. Then, in [6], authors proposed, for the first time, the algorithm presented in the next section to learn HMMs. They also gave a consistency proof, based on finite sample bounds, for estimation of the joint and conditional probability of observations sequence. Their algorithm and proof have then been modified in [7] to handle reduced rank HMMs. Reduced rank HMMs are HMMs with a low rank transition matrix. The same authors also extended this algorithm to learn TPSR [4]. However, they did not generalize the finite sample bounds and only proved the consistency. An online version of this algorithm is proposed in [19]. In [8], authors focused on improving the finite sample bounds in the case of HMMs. To achieve their goal, they proposed slight modifications to the original algorithm. Finally, in [5], ideas from compressed sensing have been used to learn compressed PSRs, which are TPSRs where the low rank assumption is replaced with a sparsity assumption. Using random projections, they proposed a more efficient algorithm in term of execution speed. However, random projections introduce a bias which results in poorer performances but good enough to still handle very large partially observable planning problems. They also made a finite sample analysis of their algorithm. Lastly, in [9] some researchers have studied effects of learning PSRs with a dimension less than the true one. However, their analysis holds only in absence of noise in estimates. Rank minimization is an extensively studied NP-hard problem for which numerous convex relaxations have been proposed. Some of them are well detailed in [15] and have been applied with success to subspace identification of linear systems [12]—[14]. These studies take advantage of the NNM formulation to add linear constraints and outperform methods using only the SVD.

VIII. CONCLUSION AND PERSPECTIVE

In this paper, we showed how learning PSRs can be reformulate as a rank minimization problem. Using a convex relaxation, we demonstrated this problem can be solved by
nuclear norm minimization and derived a new algorithm. Experiments demonstrated that it better approximates the system dynamics subspace for two reasons. First, it adapts the dimension of the approximate subspace to the noise in estimates. Secondly, it uses all estimates and takes advantage of the linearity between them to identify the subspace. Accordingly, our algorithm relies less on regularization to remove the noise due to sampling and pushes back the bias-variance trade-off.
The connections made between learning PSR and solving a rank minimization problem could be exploited further. First, as NNM can also handle convex constraints, we could add constraints specific to PSR. For example, we could ensure as NNM can also handle convex constraints, we could add admissible subspaces for the rank minimization problem, we can reduce the set of dimensional subspace. This approach has been very successful for linear systems [12]–[15]. Ideally, we could constrain the set of admissible subspaces enough to make it equal to the space spanned by correlation matrices of PSRs. Thus, any solutions to the NNM would correspond to a valid PSR. Then, we could look for a PSR of minimal dimension for which the correlation matrices are not far from the estimated ones. In addition, it could be interesting to adapt our algorithm to solve $P_2$, as $\delta$ can be estimated in probability using the concentration of measure phenomenon. Finally, in future research we would like to investigate theoretical properties of our algorithm as the ones of NNM have been and are still extensively studied [10], [11].

REFERENCES