An Online Spectral Learning Algorithm for Partially Observable Nonlinear Dynamical Systems

Byron Boots
Machine Learning Department
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Geoffrey J. Gordon
Machine Learning Department
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

Recently, a number of researchers have proposed *spectral* algorithms for learning models of dynamical systems—for example, Hidden Markov Models (HMMs), Partially Observable Markov Decision Processes (POMDPs), and Transformed Predictive State Representations (TPSRs). These algorithms are attractive since they are statistically consistent and not subject to local optima. However, they are *batch* methods: they need to store their entire training data set in memory at once and operate on it as a large matrix, and so they cannot scale to extremely large data sets (either many examples or many features per example). In turn, this restriction limits their ability to learn accurate models of complex systems. To overcome these limitations, we propose a new *online* spectral algorithm, which uses tricks such as incremental Singular Value Decomposition (SVD) and random projections to scale to much larger data sets and more complex systems than previous methods. We demonstrate the new method on an inertial measurement prediction task and a high-bandwidth video mapping task and we illustrate desirable behaviors such as "closing the loop," where the latent state representation changes suddenly as the learner recognizes that it has returned to a previously known place.

Introduction

Many problems in machine learning and artificial intelligence involve discrete-time partially observable nonlinear dynamical systems. If the observations are discrete, then *Hidden Markov Models* (HMMs) (Rabiner 1989) or, in the controlled setting, *Partially Observable Markov Decision Processes* (POMDPs) (Sondik 1971) can be used to represent belief as a discrete distribution over latent states. *Predictive State Representations (PSRs)* (Littman, Sutton, and Singh 2002), *Transformed Predictive State Representations (TPSRs)* (Rosencrantz, Gordon, and Thrun 2004; Boots, Siddiqi, and Gordon 2010), and the closely related *Observable Operator Models (OOMs)* (Jaeger 2000) are generalizations of POMDPs that have attracted interest because they both have greater representational capacity than POMDPs and yield representations that are *at least* as compact (Singh, James, and Rudary 2004). In contrast to the latent-variable representations of POMDPs, predictive representations like PSRs, TPSRs, and OOMs represent the state of a dynamical system by tracking occurrence probabilities of a set of future events (called *tests* or *characteristic events*) conditioned on past events (called *histories* or *indicative events*).

Recently, spectral algorithms have been increasingly used to learn models of partially observable nonlinear dynamical systems such as HMMs (Hsu, Kakade, and Zhang 2009; Siddiqi, Boots, and Gordon 2010) and TPSRs (Rosencrantz, Gordon, and Thrun 2004; Boots, Siddiqi, and Gordon 2010; Boots and Gordon 2010). Most of these algorithms are *statistically consistent*, unlike the popular expectation maximization (EM) algorithm, which is subject to local optima. Furthermore, spectral learning algorithms are easy to implement with a series of linear algebra operations. Despite these attractive features, spectral algorithms have so far had an important drawback: they are *batch* methods (needing to store their entire training data set in memory at once) instead of *online* ones (with space complexity independent of the number of training examples and time complexity linear in the number of training examples).

To remedy this drawback, we propose a fast, online spectral algorithm for TPSRs. TPSRs *subsume* HMMs, PSRs, and POMDPs (Singh, James, and Rudary 2004; Rosencrantz, Gordon, and Thrun 2004). In fact, previous spectral learning algorithms for several types of HMMs (Hsu, Kakade, and Zhang 2009; Siddiqi, Boots, and Gordon 2010; Song et al. 2010) are more accurately described as TPSR learning algorithms *applied to* HMMs. Therefore, our algorithm also improves on past algorithms for these other models. Our method leverages fast, low-rank modifications of the thin singular value decomposition (Brand 2006), and uses tricks such as random projections to scale to extremely large numbers of examples and features per example. Consequently, the new method can handle orders of magnitude larger data sets than previous methods, and can therefore scale to learn models of systems that are too complex for previous methods.

Experiments show that our online spectral learning algorithm does a good job recovering the parameters of a nonlinear dynamical system in two partially observable do-

mains. In our first experiment we empirically demonstrate that our online spectral learning algorithm is unbiased by recovering the parameters of a small but difficult synthetic Reduced-Rank HMM. In our second experiment we demonstrate the performance of the new method on a difficult, high-bandwidth video understanding task.

Predictive State Representations

We take a Predictive State Representation (PSR) (Littman, Sutton, and Singh 2002) view of non-linear dynamical systems. A PSR is a compact description of a dynamical system that represents state as a set of predictions of observable experiments or *tests*. Specifically, a test is an ordered sequence of action-observation pairs $q = [a_1^q, o_1^q, \ldots a_k^q, o_k^q]$ that can be executed and observed at a given time. If the observations produced by the dynamical system match those specified by the test, the test is said to have *succeeded*. The *prediction* for q, $\mathbb{P}[q^{\mathcal{O}} \mid \text{do}(q^{\mathcal{A}})]$, is the probability of seeing observations $q^O = [o_1^q, \ldots, o_k^q]$, given that we *intervene* (Pearl 2000) to take the actions $q^A = [a_1^q, \ldots, a_k^q]$. The key idea behind a PSR is that, if we know the expected outcomes of all possible tests, then we know everything there is to know about state.

A *history* is an ordered sequence of action-observation pairs $h = [a_1^h, o_1^h, \ldots, a_t^h, o_t^h]$ that has been executed and observed prior to a given time. We write $\mathcal{Q}(h)$ for the *prediction vector* of success probabilities for a set of tests $\mathcal{Q} = \{q_i\}$ given a history h.

Formally a PSR consists of the tuple $\langle \mathcal{A}, \mathcal{O}, \mathcal{Q}, \mathscr{F}, m_1 \rangle$. $\mathcal{A}$ is the set of possible actions, and $\mathcal{O}$ is the set of possible observations. $\mathcal{Q}$ is a *core* set of tests, i.e., a set such that $\mathcal{Q}(h)$ is a sufficient statistic for predicting *all* tests: the prediction for any test τ is a function of $\mathcal{Q}(h)$. $\mathscr{F}$ is the set of functions f_τ which embody these predictions: $\mathbb{P}[\tau^{\mathcal{O}} \mid h, \text{do}(\tau^{\mathcal{A}})] = f_\tau(\mathcal{Q}(h))$. Finally, $m_1 = \mathcal{Q}(\epsilon)$ is the initial prediction vector.

We restrict ourselves to *linear* PSRs, in which all prediction functions are linear: $f_\tau(\mathcal{Q}(h)) = r_\tau^{\mathsf{T}} \mathcal{Q}(h)$ for some vector $r_\tau \in \mathbb{R}^{|\mathcal{Q}|}$. A core set $\mathcal{Q}$ for a linear PSR is said to be *minimal* if the tests in $\mathcal{Q}$ are linearly independent (Jaeger 2000; Singh, James, and Rudary 2004), i.e., no one test's prediction is a linear function of the other tests' predictions. Note that the restriction to linear prediction functions is only a restriction to linear relationships between conditional probabilities of tests; linear PSRs can still represent systems with *nonlinear* dynamics.

Since $\mathcal{Q}(h)$ is a sufficient statistic for all test predictions, it is a *state* for our PSR: i.e., we can remember just $\mathcal{Q}(h)$ instead of h itself. After action a and observation o, we can update $\mathcal{Q}(h)$ recursively: if we write M_{ao} for the matrix with rows $r_{ao\tau}^{\mathsf{T}}$ for $\tau \in \mathcal{Q}$, then we can use Bayes' Rule to show:

$$\mathcal{Q}(hao) = \frac{M_{ao}\mathcal{Q}(h)}{\mathbb{P}[o \mid h, \text{do}(a)]} = \frac{M_{ao}\mathcal{Q}(h)}{m_\infty^{\mathsf{T}} M_{ao}\mathcal{Q}(h)} \tag{1}$$

where m_∞ is defined by $m_\infty^{\mathsf{T}}\mathcal{Q}(h) = 1\ (\forall h)$.

Transformed PSRs

Transformed PSRs (TPSRs) (Rosencrantz, Gordon, and Thrun 2004; Boots, Siddiqi, and Gordon 2010) are a generalization of PSRs: for any invertible matrix J, if the parameters m_1, M_{ao}, and m_∞ represent a PSR, then the transformed parameters $b_1 = Jm_1$, $B_{ao} = JM_{ao}J^{-1}$, and $b_\infty = J^{-\top} m_\infty$ represent an *equivalent* TPSR. In addition to the initial TPSR state b_1, we define normalized conditional *internal states* b_t, which we can update similarly to Eq. 1:

$$b_{t+1} \equiv \frac{B_{ao_{1:t}} b_1}{b_\infty^{\mathsf{T}} B_{ao_{1:t}} b_1} = \frac{B_{a_t o_t} b_t}{b_\infty^{\mathsf{T}} B_{a_t o_t} b_t} \tag{2}$$

Pairs $J^{-1}J$ cancel during the update, showing that predictions are equivalent as claimed:

$$\begin{aligned} &\Pr[o_{1:t} \mid \text{do}(a_{1:t})] = m_\infty^{\mathsf{T}} M_{ao_{1:t}} m_1 \\ &= m_\infty^{\mathsf{T}} J^{-1} J M_{ao_{1:t}} J^{-1} J m_1 \\ &= b_\infty^{\mathsf{T}} B_{ao_{1:t}} b_1 \end{aligned} \tag{3}$$

By choosing the invertible transform J appropriately (see the next subsection), we can think of TPSRs as maintaining a small number of sufficient statistics which are *linear combinations* of predictions for a (potentially very large) core set of tests. This view leads to the main benefit of TPSRs over regular PSRs: given a core set of tests, we can find low dimensional parameters using spectral methods and regression instead of combinatorial search. In this respect, TPSRs are closely related to the transformed representations of LDSs and HMMs found by *subspace identification* (Van Overschee and De Moor 1996; Katayama 2005; Soatto and Chiuso 2001; Hsu, Kakade, and Zhang 2009). Furthermore, to make it practical to work with data gathered from complex real-world systems, we can learn from finite-dimensional features of the past and future, rather than an extremely large or even infinite core set of tests (see Section "Batch Learning of TPSRs," below). Additional details regarding the relationship between TPSRs and PSRs can be found in (Boots, Siddiqi, and Gordon 2010).

Relating TPSRs and PSRs For some PSR, let $\mathcal{Q}$ be a minimal core set of tests. Then, let $\mathcal{T}$ be a (larger) core set of tests, and let $\mathcal{H}$ be a mutually exclusive and exhaustive partition of the set of all possible histories. (Elements of $\mathcal{H}$ are called *indicative events* (Jaeger 2000).) And, let $\mathcal{AO}$ be the set of all possible action-observation pairs. Define $\phi^{\mathcal{H}}(h)$ for $h \in \mathcal{H}$ to be a vector of *indicative* features, i.e., features of history, and define $\phi^{\mathcal{AO}}(a, o)$ to be a vector of features of a present action and observation. Finally, define $\phi^{\mathcal{T}}(h)$ to be a vector of *characteristic* features: that is, each entry of $\phi^{\mathcal{T}}(h)$ is a linear combination of some set of test predictions.

For purposes of gathering data, we assume that we can sample from a sufficiently diverse distribution ω over histories. Note that this assumption means that we cannot estimate m_1 (equivalently, b_1), since we typically don't have samples of trajectories starting from m_1. Instead, we will estimate m_*, an arbitary feasible state. If we only have m_*, initial probability estimates will be approximate, but the difference will disappear over time as our process mixes. We will also assume that we execute a known exploration policy from each sampled history; with this assumption, it is possible to construct unbiased samples of $\phi^{\mathcal{T}}(h)$ by importance weighting (Bowling et al. 2006; Boots and Gordon 2010).

When our algorithms below call for samples of $\phi^{\mathcal{T}}(h)$, we use this importance weighting trick to provide them.

We define $\Phi^{\mathcal{T}}$, $\Phi^{\mathcal{H}}$, and $\Phi^{\mathcal{AO}}$ as matrices of characteristic, indicative, and present features respectively, with first dimension equal to the number of features and second dimension equal to $|\mathcal{H}|$. An entry of $\Phi^{\mathcal{H}}$ is the expectation of one of the indicative features given the occurrence of one of the indicative events and the exploration policy; an entry of $\Phi^{\mathcal{T}}$ is the expectation of one of our characteristic features given one of the indicative events; and an entry of $\Phi^{\mathcal{AO}}$ is the expectation of one of the present features given one of the indicative events and the exploration policy. We also define $\psi = \mathbb{P}[\mathcal{H}]$, $D = \operatorname{diag}(\psi)$, $R \in \mathbb{R}^{|\mathcal{T}|\times|\mathcal{Q}|}$ as the matrix with rows $r_{\tau_i}^{\mathsf{T}}$, $S \in \mathbb{R}^{|\mathcal{Q}|\times|\mathcal{H}|}$ as the expected state $\mathbb{E}[\mathcal{Q} \mid \mathcal{H}]$, and M as a $|\mathcal{Q}| \times |AO| \times |\mathcal{Q}|$ third-order tensor (each $|\mathcal{Q}| \times |\mathcal{Q}|$ slice, M_{ao}, of M is the transition matrix for a particular action-observation pair).

Given the above notation, we define several covariance and "trivariance" matrices which are related to the parameters of the PSR. In several of the following equations we use *tensor-matrix multiplication* $\times_v$, also known as a *mode-v product*: $\times_v$ multiplies the second dimension of a matrix with the vth mode of a tensor.

$$\begin{aligned}
[\mu_{\mathcal{H}}]_i \equiv& \mathbb{E}[\phi_i^{\mathcal{H}}(h)] \\
\implies \mu_{\mathcal{H}} =& \Phi^{\mathcal{H}}\psi && \text{(4a)} \\
[\Sigma_{\mathcal{AO},\mathcal{AO}}]_{i,j} \equiv& \mathbb{E}[\phi_i^{\mathcal{AO}}(a,o) \cdot \phi_j^{\mathcal{AO}}(a,o)] \\
\implies \Sigma_{\mathcal{AO},\mathcal{AO}} =& \Phi^{\mathcal{AO}} D \Phi^{\mathcal{AO}\mathsf{T}} && \text{(4b)} \\
[\Sigma_{\mathcal{T},\mathcal{H}}]_{i,j} \equiv& \mathbb{E}[\phi_i^{\mathcal{T}}(\tau^O) \cdot \phi_j^{\mathcal{H}}(h) \mid \operatorname{do}(\tau^A)] \\
\implies \Sigma_{\mathcal{T},\mathcal{H}} =& \Phi^{\mathcal{T}} R S D \Phi^{\mathcal{H}\mathsf{T}} && \text{(4c)} \\
[\Sigma_{\mathcal{T},\mathcal{AO},\mathcal{H}}]_{i,j,k} \equiv& \mathbb{E}[\phi_i^{\mathcal{T}}(\tau^O) \cdot \phi_j^{\mathcal{H}}(h) \cdot \phi_k^{\mathcal{AO}}(ao) \mid \operatorname{do}(\tau^A, a)] \\
\implies \Sigma_{\mathcal{T},\mathcal{AO},\mathcal{H}} =& \mathsf{M} \times_1 (\Phi^{\mathcal{T}} R) \times_2 (\Phi^{\mathcal{AO}} D) \times_3 (\Phi^{\mathcal{H}} D S^{\mathsf{T}}) && \text{(4d)}
\end{aligned}$$

Now, if we are given an additional matrix U such that $U^{\mathsf{T}}\Phi^{\mathcal{T}}R$ is invertible, we can use Equations 4a–d to define a TPSR whose parameters are only a similarity transform away from the original PSR parameters.

$$\begin{aligned}
b_* \equiv& U^{\mathsf{T}}\Sigma_{\mathcal{T},\mathcal{H}} e = (U^{\mathsf{T}}\Phi^{\mathcal{T}}R) m_* && \text{(5a)} \\
b_\infty^{\mathsf{T}} \equiv& \mu_{\mathcal{H}}^{\mathsf{T}} (U^{\mathsf{T}}\Sigma_{\mathcal{T},\mathcal{H}})^{\dagger} = m_\infty^{\mathsf{T}} (U^{\mathsf{T}}\Phi^{\mathcal{T}}R)^{-1} && \text{(5b)} \\
B_{ao} \equiv& \Sigma_{\mathcal{T},\mathcal{AO},\mathcal{H}} \\
& \times_1 U^{\mathsf{T}} \times_2 \Phi^{\mathcal{AO}\mathsf{T}} (\Sigma_{\mathcal{AO},\mathcal{AO}})^{-1} \times_3 (\Sigma_{\mathcal{T},\mathcal{H}}^{\mathsf{T}} U)^{\dagger} \\
=& (U^{\mathsf{T}}\Phi^{\mathcal{T}}R) M_{ao} (U^{\mathsf{T}}\Phi^{\mathcal{T}}R)^{-1} && \text{(5c)}
\end{aligned}$$

Batch Learning of TPSRs

The identities in Equation 5a–c yield a straightforward learning algorithm (Boots, Siddiqi, and Gordon 2010): we build empirical estimates $\hat{\mu}_{\mathcal{H}}$, $\widehat{\Sigma}_{\mathcal{AO},\mathcal{AO}}$, $\widehat{\Sigma}_{\mathcal{T},\mathcal{H}}$, and $\widehat{\Sigma}_{\mathcal{T},\mathcal{AO},\mathcal{H}}$ of the matrices defined in Equation 4. Once we have constructed $\widehat{\Sigma}_{\mathcal{T},\mathcal{H}}$, we can compute $\widehat{U}$ as the matrix of n leading left singular vectors of $\widehat{\Sigma}_{\mathcal{T},\mathcal{H}}$. Finally, we plug the estimated covariances and $\widehat{U}$ into Equation 5 to compute estimated PSR parameters. One of the advantages of subspace identification is that the complexity of the model can be tuned by selecting the number of singular vectors in $\widehat{U}$, at the risk of losing prediction quality.

As we include more data in our averages, the law of large numbers guarantees that our estimates converge to the true matrices $\mu_{\mathcal{H}}$, $\Sigma_{\mathcal{AO},\mathcal{AO}}$, $\Sigma_{\mathcal{T},\mathcal{H}}$, and $\Sigma_{\mathcal{T},\mathcal{AO},\mathcal{H}}$. So by continuity of the formulas above, if our system is truly a PSR of finite rank, our estimates $\widehat{b}_*$, $\widehat{b}_\infty$, and $\widehat{B}_{ao}$ converge, with probability 1, to the true parameters up to a linear transform—that is, our learning algorithm is *consistent*.[1]

An Efficient Batch Learning Algorithm

Unfortunately, it is difficult to implement the naïve algorithm in practice. For example, if there are a large number of features of tests or features of histories, we may not be able even to store the the tensor in Equation 4d. Therefore, we need to use a more efficient algorithm, one that does not explicitly build the tensor $\Sigma_{\mathcal{T},\mathcal{AO},\mathcal{H}}$.

The key idea is to compute a set of smaller-sized intermediate quantities from realizations of characteristic features $\phi^{\mathcal{T}}$, indicative features $\phi^{\mathcal{H}}$, and present features $\phi^{\mathcal{AO}}$, and then compute TPSR parameters from these quantities. In particular, we compute a subset of the empirical estimates from above: $\hat{\mu}_{\mathcal{H}}$, $\widehat{\Sigma}_{\mathcal{AO},\mathcal{AO}}$ and $\widehat{\Sigma}_{\mathcal{T},\mathcal{H}}$. From $\widehat{\Sigma}_{\mathcal{T},\mathcal{H}}$ we compute the rank-n singular value decomposition $\widehat{U}$, $\widehat{S}$, $\widehat{V}^{\mathsf{T}}$. Then, instead of computing $\widehat{\Sigma}_{\mathcal{T},\mathcal{AO},\mathcal{H}}$, we use the above matrices to compute a smaller tensor $\widehat{\mathsf{B}}_{\mathcal{AO}} = \widehat{\Sigma}_{\mathcal{T},\mathcal{AO},\mathcal{H}} \times_1 \widehat{U}^{\mathsf{T}} \times_3 (\widehat{\Sigma}_{\mathcal{T},\mathcal{H}})^{\dagger}$ directly. To get the final desired parameter $\widehat{B}_{ao}$ (Equation 5), we simply compute $B_{ao} = \widehat{\mathsf{B}}_{\mathcal{AO}} \times_2 \phi^{\mathcal{AO}}(ao)^{\mathsf{T}} (\Sigma_{\mathcal{AO},\mathcal{AO}})^{-1}$. The tensor $\widehat{\mathsf{B}}_{\mathcal{AO}}$ has much lower dimensionality than $\Sigma_{\mathcal{T},\mathcal{AO},\mathcal{H}}$, with first and third modes being no greater than n dimensions (the length of the latent state vector b_t). Therefore, we save substantially on both memory and runtime by avoiding construction of the larger tensor.

In more detail, we begin by estimating $\mu_{\mathcal{H}}$, the unnormalized[2] empirical expectation (over histories sampled from ω) of the indicative feature vector: if $\phi_t^{\mathcal{H}}$ is the feature vector for the tth history,

$$\widehat{\mu}_{\mathcal{H}} = \sum_{t=1}^{w} \phi_t^{\mathcal{H}} \qquad \text{(6a)}$$

Next, we estimate $\Sigma_{\mathcal{AO},\mathcal{AO}}^{-1}$, the inverse of the unnormalized empirical expectation of the product of present features:

$$\widehat{\Sigma}_{\mathcal{AO},\mathcal{AO}}^{-1} = \left(\sum_{t=1}^{w} \phi_t^{\mathcal{AO}} \phi_t^{\mathcal{AO}} \right)^{-1} \qquad \text{(6b)}$$

Next, each element of our estimate $\widehat{\Sigma}_{\mathcal{T},\mathcal{H}}$ is an unnormalized empirical expectation of the *product* of one indicative

[1] Continuity holds if we fix $\widehat{U}$; a similar but more involved argument works if we estimate $\widehat{U}$ as well.

[2] We can get away with using *unnormalized* empirical expectations in Equation 6a–d since the normalizations would just cancel when we compute the TPSR parameters in Equation 7a–c.

feature and one characteristic feature, if we sample a history from ω and then follow an appropriate sequence of actions. We can compute all elements of $\widehat{\Sigma}_{\mathcal{T},\mathcal{H}}$ from a single sample of trajectories if we sample histories from ω, follow an appropriate exploratory policy, and then importance-weight each sample (Bowling et al. 2006): $[\widehat{\Sigma}_{\mathcal{T},\mathcal{H}}]_{ij}$ is $\sum_{t=1}^{w} \lambda_t \phi_{it}^{\mathcal{T}} \phi_{jt}^{\mathcal{H}}$, where λ_t is an importance weight.

Next we compute $\widehat{U}\widehat{S}\widehat{V}^{\mathsf{T}}$, the rank-$n$ thin singular value decomposition of $\widehat{\Sigma}_{\mathcal{T},\mathcal{H}}$:

$$\langle \widehat{U}, \widehat{S}, \widehat{V} \rangle = \text{SVD}\left(\sum_{t=1}^{w} \lambda_t \phi_t^{\mathcal{T}} \phi_t^{\mathcal{H}} \right) \tag{6c}$$

The left singular vectors $\widehat{U}$ are directly needed, e.g., in Eq. 5. The right singular vectors $\widehat{V}$ and singular values $\widehat{S}$ can also be used to make computation of the other TPSR parameters more efficient: recall from Equation 4d that each element of $\widehat{\Sigma}_{\mathcal{T},ao,\mathcal{H}}$ is a weighted unnormalized empirical expectation of the product of one indicative feature, one characteristic feature, and one present feature. To efficiently construct the tensor $\widehat{\mathsf{B}}_{\mathcal{AO}}$, we combine Equation 4d with Equation 5c while making use of the singular value decomposition of $\widehat{\Sigma}_{\mathcal{T},\mathcal{H}}$:

$$\widehat{\mathsf{B}}_{\mathcal{AO}} = \sum_{t=1}^{w} \lambda_t \left(\widehat{U}^{\mathsf{T}} \phi_t^{\mathcal{T}} \right) \otimes \left(\phi_t^{\mathcal{AO}} \right) \otimes \left(\widehat{S}^{-1} \widehat{V}^{\mathsf{T}} \phi_t^{\mathcal{H}} \right) \tag{6d}$$

where $\otimes$ indicates tensor (outer) product. The idea here is to build the lower-dimensional $\widehat{\mathsf{B}}_{\mathcal{AO}}$ directly, by making use of the tall thin matrices $\widehat{U}$ and $\widehat{V}$, without first building the high-dimensional $\widehat{\Sigma}_{\mathcal{T},\mathcal{AO},\mathcal{H}}$.

Using these matrices we can efficiently compute estimates of the TPSR parameters in Equation 5:

$$\hat{b}_* = \widehat{S}\widehat{V}^{\mathsf{T}} e \tag{7a}$$

$$\hat{b}_\infty^{\mathsf{T}} = \hat{\mu}_{\mathcal{H}}^{\mathsf{T}} \widehat{V} \widehat{S}^{-1} \tag{7b}$$

$$\widehat{B}_{ao} = \widehat{\mathsf{B}}_{\mathcal{AO}} \times_2 \left(\phi^{\mathcal{AO}}(ao)^{\mathsf{T}} \cdot \widehat{\Sigma}_{\mathcal{AO},\mathcal{AO}}^{-1} \right) \tag{7c}$$

This learning algorithm works well when the number of features of tests, histories, and action-observation pairs is relatively small, and in cases where data is collected in batch. These restrictions can be limiting for many real-world data sets. In practice, the number of features may need to be quite large in order to accurately estimate the parameters of the TPSR. Additionally, we are often interested in estimating TPSRs from massive datasets, updating TPSR parameters given a new batch of data, or learning TPSRs online from a data stream. Below we develop several computationally efficient extensions to overcome these practical obstacles to learning in real-world situations.

Iterative Updates to TPSR Parameters

We first attack the problem of updating existing TPSR parameters given a batch of new information. Next, we look at the special case of updating TPSR parameters in an online setting (batch size 1), and develop additional optimizations for this situation.

Batch Updates We first present an algorithm for updating existing TPSR parameters given a new batch of characteristic features $\phi_{\text{new}}^{\mathcal{T}}$, indicative features $\phi_{\text{new}}^{\mathcal{H}}$, and present features $\phi_{\text{new}}^{\mathcal{AO}}$. Naïvely, we can just store empirical estimates and update them from each new batch of data: $\hat{\mu}_{\mathcal{H}} + \sum_{t=1}^{w} \phi_{\text{new}_t}^{\mathcal{H}}$, $\widehat{\Sigma}_{\mathcal{AO},\mathcal{AO}} + \phi_{\text{new}}^{\mathcal{AO}} \phi_{\text{new}}^{\mathcal{AO}\,\mathsf{T}}$, $\widehat{\Sigma}_{\mathcal{T},\mathcal{H}} + \phi_{\text{new}}^{\mathcal{T}} \phi_{\text{new}}^{\mathcal{H}\,\mathsf{T}}$, and $\widehat{\Sigma}_{\mathcal{T},\mathcal{AO},\mathcal{H}} + \sum_{t=1}^{w} \phi_{\text{new}_t}^{\mathcal{T}} \otimes \phi_{\text{new}_t}^{\mathcal{AO}} \otimes \phi_{\text{new}_t}^{\mathcal{H}}$. Then, after each batch, we can apply Equation 5a–c to learn new TPSR parameters $\hat{b}_*$, $\hat{b}_\infty^{\mathsf{T}}$, and $\widehat{B}_{ao}$.

This naïve algorithm is very inefficient: it requires storing each of the matrices in Equation 4a–d, updating these matrices given new information, and recomputing the TPSR parameters. However, as we have seen, it is also possible to write the TPSR parameters (Equation 7a–c) in terms of a set of lower-dimensional memory-efficient matrices and tensors (Equation 6a–d), made possible by the singular value decomposition of $\widehat{\Sigma}_{\mathcal{T},\mathcal{H}}$. The key idea is to update these lower-dimensional matrices directly, instead of the naïve updates suggested above, by taking advantage of numerical algorithms for updating singular value decompositions efficiently (Brand 2006).

We begin by simply implementing the additive update to the vector $\widehat{\mu}_{\mathcal{H}}$.

$$\hat{\mu}_{\mathcal{H}_{\text{new}}} = \hat{\mu}_{\mathcal{H}} + \sum_{t=1}^{w} \phi_t^{\mathcal{H}} \tag{8a}$$

The inverse empirical covariance of present features remains computationally expensive to update. If the new batch of data is large, and updates infrequent, then we can maintain the empirical covariance $\widehat{\Sigma}_{\mathcal{AO},\mathcal{AO}}$ separately from the inverse. We update $\widehat{\Sigma}_{\mathcal{AO},\mathcal{AO}}$, and after each update, we invert the matrix.

$$\widehat{\Sigma}_{\mathcal{AO},\mathcal{AO}_{\text{new}}}^{-1} = \left(\widehat{\Sigma}_{\mathcal{AO},\mathcal{AO}} + \phi_{\text{new}}^{\mathcal{AO}} \phi_{\text{new}}^{\mathcal{AO}\,\mathsf{T}} \right)^{-1} \tag{8b}$$

If we are updating frequently with smaller batches of data, we can instead use a low-rank update via the Sherman-Morrison formula. See Equation 9a for details.

The main computational savings come from using incremental SVD to update $\widehat{U}, \widehat{S}, \widehat{V}$ and $\widehat{\mathsf{B}}_{\mathcal{AO}}$ directly. The incremental update for $\widehat{U}, \widehat{S}, \widehat{V}$ is much more efficient than the naïve additive update when the number of new data points is much smaller than the number of features in $\phi^{\mathcal{T}}$ and $\phi^{\mathcal{H}}$. The incremental update for $\widehat{\mathsf{B}}_{\mathcal{AO}}$ saves time and space when the number of features in $\phi^{\mathcal{T}}$ and $\phi^{\mathcal{H}}$ is much larger than the latent dimension n.

Our goal is therefore to compute the updated SVD,

$$\langle \widehat{U}_{\text{new}}, \widehat{S}_{\text{new}}, \widehat{V}_{\text{new}} \rangle = \text{SVD}\left(\widehat{\Sigma}_{\mathcal{T},\mathcal{H}} + \phi_{\text{new}}^{\mathcal{T}} \Lambda \phi_{\text{new}}^{\mathcal{H}\,\mathsf{T}} \right),$$

where Λ is a diagonal matrix of importance weights $\Lambda = \text{diag}(\lambda_{1:t})$. We will derive the incremental SVD update in two steps. First, if the new data $\phi_{\text{new}}^{\mathcal{T}}$ and $\phi_{\text{new}}^{\mathcal{H}}$ lies entirely within the column spaces of $\widehat{U}$ and $\widehat{V}$ respectively, then we can find $\widehat{S}_{\text{new}}$ by projecting both the new and old data onto the subspaces defined by $\widehat{U}$ and $\widehat{V}$, and diagonalizing the

resulting small $(n \times n)$ matrix:

$$\langle \widehat{\mathcal{U}}, \widehat{S}_{\text{new}}, \widehat{\mathcal{V}} \rangle = \text{SVD}\left(\widehat{U}^\mathsf{T}\left(\widehat{\Sigma}_{\mathcal{T},\mathcal{H}} + \phi^{\mathcal{T}}_{\text{new}} \Lambda \phi^{\mathcal{H}}_{\text{new}}{}^\mathsf{T}\right)\widehat{V}\right)$$
$$= \text{SVD}\left(\widehat{S} + (\widehat{U}^\mathsf{T}\phi^{\mathcal{T}}_{\text{new}})\Lambda(\widehat{V}^\mathsf{T}\phi^{\mathcal{H}}_{\text{new}})^\mathsf{T}\right)$$

We can then compute $\widehat{U}_{\text{new}} = \widehat{U}\widehat{\mathcal{U}}^\mathsf{T}$ and $\widehat{V}_{\text{new}} = \widehat{V}\widehat{\mathcal{V}}^\mathsf{T}$, the rotations of $\widehat{U}$ and $\widehat{V}$ induced by the new data.

If the new data does *not* lie entirely within the column space of $\widehat{U}$ and $\widehat{V}$, we can update the SVD efficiently (and optionally approximately) following Brand (2006). The idea is to split the new data into a part within the column span of $\widehat{U}$ and $\widehat{V}$ and a remainder, and use this decomposition to construct a small matrix to diagonalize as above.

Let C and D be orthonormal bases for the component of the column space of $\phi^{\mathcal{T}}_{\text{new}}$ orthogonal to $\widehat{U}$ and the component of the column space of $\phi^{\mathcal{H}}_{\text{new}}$ orthogonal to $\widehat{V}$:

$$C = \text{orth}\left((I - \widehat{U}\widehat{U}^\mathsf{T})\phi^{\mathcal{T}}_{\text{new}}\right)$$
$$D = \text{orth}\left((I - \widehat{V}\widehat{V}^\mathsf{T})\phi^{\mathcal{H}}_{\text{new}}\right)$$

The dimension of C and D is upper-bounded by the number of data points in our new batch, or the number of features of tests and histories, whichever is smaller. (If the dimension is large, the orthogonalization step above (as well as other steps below) may be too expensive; we can accommodate this case by splitting a large batch of examples into several smaller batches.) Let

$$K = \begin{bmatrix} \widehat{S} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} \widehat{U}^\mathsf{T} \\ C^\mathsf{T} \end{bmatrix} \phi^{\mathcal{T}}_{\text{new}} \Lambda \phi^{\mathcal{H}}_{\text{new}}{}^\mathsf{T} [\, \widehat{V}\ D\,],$$

and diagonalize K to get the update to $\widehat{S}$:

$$\langle \widehat{\mathcal{U}}, \widehat{S}_{\text{new}}, \widehat{\mathcal{V}} \rangle = \text{SVD}\,(K) \tag{8c}$$

Finally, as above, $\widehat{\mathcal{U}}$ and $\widehat{\mathcal{V}}$ rotate the extended subspaces $[\,\widehat{U}\ C\,]$ and $[\,\widehat{V}\ D\,]$:

$$\widehat{U}_{\text{new}} = [\,\widehat{U}\ C\,]\,\widehat{\mathcal{U}} \tag{8d}$$
$$\widehat{V}_{\text{new}} = [\,\widehat{V}\ D\,]\,\widehat{\mathcal{V}} \tag{8e}$$

Note that if there are components orthogonal to $\widehat{U}$ and $\widehat{V}$ in the new data (i.e., if C and D are nonempty), the size of the thin SVD will grow. So, during this step, we may choose to tune the complexity of our estimated model by restricting the dimensionality of the SVD. If we do so, we may *lose information* compared to a batch SVD: if future data causes our estimated leading singular vectors to change, the dropped singular vectors may become relevant again. However, empirically, this information loss can be minimal, especially if we keep extra "buffer" singular vectors beyond what we expect to need.

Also note that the above updates do not necessarily preserve orthonormality of $\widehat{U}_{\text{new}}$ and $\widehat{V}_{\text{new}}$, due to discarded nonzero singular values and the accumulation of numerical errors. To correct for this, every few hundred iterations, we re-orthonormalize using a QR decomposition and a SVD:

$$\langle U_Q, U_R \rangle = \text{QR}\left(\widehat{U}_{\text{new}}\right)$$
$$\langle V_Q, V_R \rangle = \text{QR}\left(\widehat{V}_{\text{new}}\right)$$
$$\langle U_{QR}, S_{QR}, V_{QR} \rangle = \text{SVD}\left(U_R \widehat{S}_{\text{new}} V_R^\mathsf{T}\right)$$
$$\widehat{U}_{\text{new}} = U_Q U_{QR}$$
$$\widehat{V}_{\text{new}} = V_Q V_{QR}$$
$$\widehat{S}_{\text{new}} = S_{QR}$$

The updated SVD now gives us enough information to compute the update to $\widehat{\mathsf{B}}_{\mathcal{AO}}$. Let Λ be the diagonal tensor of importance weights $\Lambda_{i,i,i} = \lambda_i (i \in 1, 2, \ldots, t)$. Using the newly computed subspaces $\widehat{U}_{\text{new}}, \widehat{S}_{\text{new}}$, and $\widehat{V}_{\text{new}}$, we can compute an additive update from the new data.

$$\widehat{\mathsf{B}}_{\mathcal{AO}_{\text{new}}} = \left(\widehat{\Sigma}_{\mathcal{T},\mathcal{AO},\mathcal{H}} + \Lambda \times_1 \phi^{\mathcal{T}}_{\text{new}} \times_2 \phi^{\mathcal{AO}}_{\text{new}} \times_3 \phi^{\mathcal{H}}_{\text{new}}\right)$$
$$\times_1 \widehat{U}^\mathsf{T}_{\text{new}} \times_3 \widehat{S}^{-\mathsf{T}}_{\text{new}} \widehat{V}^\mathsf{T}_{\text{new}}$$
$$= \widehat{\mathsf{B}}_{\text{update}} + \widehat{\mathsf{B}}_{\text{new}} \tag{8f}$$

where $\widehat{\mathsf{B}}_{\text{update}}$ can be viewed as the projection of $\widehat{\mathsf{B}}_{\mathcal{AO}}$ onto the new subspace:

$$\widehat{\mathsf{B}}_{\text{update}} = \widehat{\Sigma}_{\mathcal{T},\mathcal{AO},\mathcal{H}} \times_1 \widehat{U}^\mathsf{T}_{\text{new}} \times_3 \widehat{S}^{-\mathsf{T}}_{\text{new}} \widehat{V}^\mathsf{T}_{\text{new}}$$
$$= \begin{bmatrix} \widehat{\mathsf{B}}_{\mathcal{AO}} & 0 \\ 0 & 0 \end{bmatrix} \times_1 \left(\widehat{U}^\mathsf{T}_{\text{new}} \begin{bmatrix} \widehat{U} & 0 \end{bmatrix}\right)$$
$$\times_3 \left(\widehat{S}^{-1}_{\text{new}} \widehat{V}^\mathsf{T}_{\text{new}} \begin{bmatrix} \widehat{V}\widehat{S} & 0 \end{bmatrix}\right)$$

and $\widehat{\mathsf{B}}_{\text{new}}$ is the projection of the additive update onto the new subspace:

$$\widehat{\mathsf{B}}_{\text{new}} = \Lambda \times_1 (\widehat{U}^\mathsf{T}_{\text{new}} \phi^{\mathcal{T}}_{\text{new}}) \times_2 (\phi^{\mathcal{AO}}_{\text{new}}) \times_3 (\widehat{S}^{-1}_{\text{new}} \widehat{V}^\mathsf{T}_{\text{new}} \phi^{\mathcal{H}}_{\text{new}})$$

Once the updated estimates in Equation 8a–f have been calculated, we can compute the new TPSR parameters by Equation 7a–c.

Online updates In the online setting (with just one sample per batch), the updates to $\widehat{\Sigma}_{\mathcal{AO},\mathcal{AO}}$ and $\widehat{S}$ are rank-1, allowing some additional efficiencies. The inverse empirical covariance $\widehat{\Sigma}^{-1}_{\mathcal{AO},\mathcal{AO}}$ can be updated directly using the Sherman-Morrison formula:

$$\widehat{\Sigma}^{-1}_{\mathcal{AO},\mathcal{AO}_{\text{new}}} = \widehat{\Sigma}^{-1}_{\mathcal{AO},\mathcal{AO}} - \frac{\widehat{\Sigma}^{-1}_{\mathcal{AO},\mathcal{AO}} \phi_1^{\mathcal{AO}} \phi_1^{\mathcal{AO}\mathsf{T}} \widehat{\Sigma}^{-1}_{\mathcal{AO},\mathcal{AO}}}{1 + \phi_1^{\mathcal{AO}\mathsf{T}} \widehat{\Sigma}^{-1}_{\mathcal{AO},\mathcal{AO}} \phi_1^{\mathcal{AO}}} \tag{9a}$$

Next we can compute the rank-1 update to the matrices $\widehat{U}, \widehat{S}$, and $\widehat{V}^\mathsf{T}$. We can compute C and D efficiently via a simplified Gram-Schmidt step (Brand 2006):

$$\widetilde{C} = (I - \widehat{U}\widehat{U}^\mathsf{T})\phi_1^{\mathcal{T}}$$
$$C = \widetilde{C}/\|\widetilde{C}\|$$
$$\widetilde{D} = (I - \widehat{V}\widehat{V}^\mathsf{T})\phi_1^{\mathcal{H}}$$
$$D = \widetilde{D}/\|\widetilde{D}\|$$

Finally, we can compute K by adding a rank-1 matrix to $\widehat{S}$:

$$K = \begin{bmatrix} \widehat{S} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} \widehat{U}^\mathsf{T} \\ C^\mathsf{T} \end{bmatrix} \phi_1^\mathcal{T} \lambda_1 \phi_1^{\mathcal{H}\mathsf{T}} \begin{bmatrix} \widehat{V} & D \end{bmatrix},$$

We then compute the updated parameters using Eqs. 8c–e.

Random Projections for High Dimensional Feature Spaces

Despite their simplicity and wide applicability, HMMs, POMDPs, and PSRs are limited in that they are usually restricted to discrete observations, and the state is usually restricted to have only moderate cardinality. In Section "Transformed PSRs," above, we described a feature-based representation for TPSRs that relaxes this restriction. Recently, Song et al. went a step further, and proposed a spectral learning algorithm for HMMs with continuous observations by representing distributions over these observations and continuous latent states as embeddings in an infinite dimensional Hilbert space (Song et al. 2010). These Hilbert Space Embeddings of HMMs (HSE-HMMs) use essentially the same framework as other spectral learning algorithms for HMMs and PSRs, but avoid working in the infinite-dimensional Hilbert space by the well-known "kernel trick."

HSE-HMMs have been shown to perform well on several real-world datasets, often beating the next best method by a substantial margin. However, they scale poorly due to the need to work with the kernel matrix, whose size is quadratic in the number of training points.

We can overcome this scaling problem and learn TPSRs that approximate HSE-HMMs using *random features* for kernel machines (Rahimi and Recht 2007): we construct a large but finite set of random features which let us *approximate* a desired kernel using ordinary dot products. Rahimi and Recht show how to approximate several popular kernels, including radial basis function (RBF) kernels and Laplacian kernels.) The benefit of random features is that we can use fast linear methods that do not depend on the number of data points to approximate the original kernel machine.

HSE-HMMs are no exception: using random features of tests and histories, we can *approximate* a HSE-HMM with a TPSR. If we combine random features with the above online learning algorithm, we can approximate an HSE-HMM very closely by using an extremely large number of random features. Such a large set of features would overwhelm batch spectral learning algorithms, but our online method allows us to approximate an HSE-HMM very closely, and scale HSE-HMMs to orders of magnitude larger training sets or even to *streaming* datasets with an inexhaustible supply of training data.

Experimental Results

We designed 3 sets of experiments to evaluate the statistical properties and practical potential of our online spectral learning algorithm. In the first experiment we show the convergence behavior of the algorithm. In the second experiment we show how online spectral learning combined with random projections can be used to learn a TPSR that closely approximates the performance of a HSE-HMM. In the third experiment we demonstrate how this combination allows us to model a high-bandwidth, high-dimensional video, where the amount of training data would overwhelm a kernel-based method like HSE-HMMs and the number of features would overwhelm a TPSR batch learning algorithm.

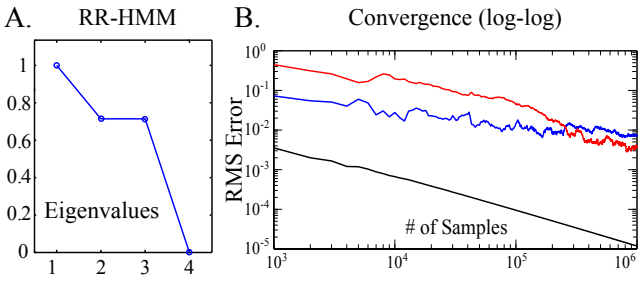

Figure 1: A synthetic RR-HMM. (A.) The eigenvalues of the true transition matrix. (B.) RMS error in the nonzero eigenvalues of the estimated transition matrix vs. number of training samples, averaged over 10 trials. The error steadily decreases, indicating that the TPSR model is becoming more accurate, as we incorporate more training data.

A Synthetic Example

First we demonstrate the convergence behavior of our algorithm on a difficult synthetic HMM from Siddiqi et al. (2010). This HMM is 2-step observable, with 4 states, 2 observations, and a rank-3 transition matrix. (So, the HMM is reduced rank (an "RR-HMM") and features of multiple observations are required to disambiguate state.) The transition matrix T and the observation matrix O are:

$$T = \begin{bmatrix} 0.7829 & 0.1036 & 0.0399 & 0.0736 \\ 0.1036 & 0.4237 & 0.4262 & 0.0465 \\ 0.0399 & 0.4262 & 0.4380 & 0.0959 \\ 0.0736 & 0.0465 & 0.0959 & 0.7840 \end{bmatrix}$$

$$O = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

We sample observations from the true model and then estimate the model using the algorithm of Section "Online updates." Since we only expect to recover the transition matrix up to a similarity transform, we compare the eigenvalues of $\widehat{B} = \sum_o \widehat{B}_o$ in the learned model to the eigenvalues of the transition matrix T of the true model. Fig. 1 shows that the learned eigenvalues converge to the true ones as the amount of data increases.

Slot Car Inertial Measurement

In a second experiment, we compare the online spectral algorithm with random features to HSE-HMMs with Gaussian RBF kernels. The setup consisted of a track and a miniature car (1:32 scale) guided by a slot cut into the track (Song et al. 2010). Figure 2(A) shows the car and the attached IMU (an Intel Inertiadot), as well as the 14m track, which contains elevation changes and banked curves. We collected the estimated 3D acceleration and velocity of the car at 10Hz. The data consisted of 3000 successive measurements while

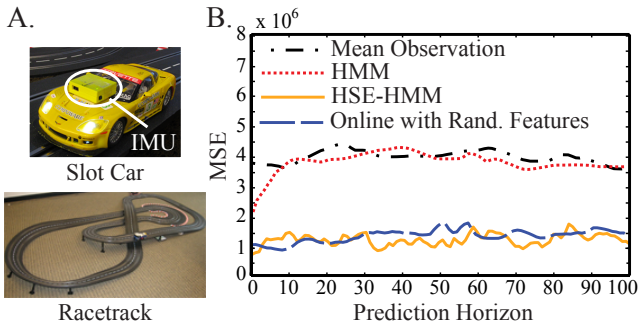

Figure 2: Slot car inertial measurement data. (A) The slot car platform: the car and IMU (top) and the racetrack (bottom). (B) Squared error for prediction with different estimated models. Dash-dot shows the baseline of simply predicting the mean measurement on all frames.

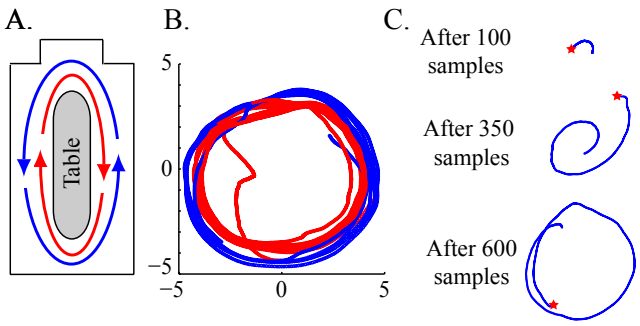

Figure 3: Modeling video. (A.) Schematic of the camera's environment. (B.) The second and third dimension of the learned belief space (the first dimension contains normalization information). Points are colored red when the camera is traveling clockwise and blue when traveling counterclockwise. The learned state space separates into two manifolds, one for each direction, connected at points where the camera changes direction. (The manifolds appear on top of one another, but are separated in the fourth latent dimension.) (C.) Loop closing: estimated historical camera positions after 100, 350, and 600 steps. Red star indicates current camera position. The camera loops around the table, and the learned map "snaps" to the correct topology when the camera passes its initial position.

the slot car circled the track controlled by a constant policy. The goal was to learn a model of the noisy IMU data, and, after filtering, to predict future readings.

We trained a 20-dimensional HSE-HMM using the algorithm of Song et al., with tests and histories consisting of 150 consecutive observations. We set the bandwidth parameter of the Gaussian RBF kernels with the "median trick," and the regularization (ridge) parameter was 10^{-4}. For details see Song et al. (2010).

Next we trained a 20-dimensional TPSR with random Fourier features to *approximate* the Gaussian RBF kernel. We generated 25000 features for the tests and histories and 400 features for current observations, and then used the online spectral algorithm to learn a model. Finally, to provide some context, we learned a 20-state discrete HMM (with 400 levels of discretization for observations) using the Baum-Welch EM algorithm run until convergence.

For each model, we performed filtering for different extents $t_1 = 100, 101, \ldots, 250$, then predicted an image which was a further $t_2 = 1, 2, \ldots, 100$ steps in the future. The squared error of this prediction in the IMU's measurement space was recorded, and averaged over all the different filtering extents t_1 to obtain means which are plotted in Figure 2(B).

The results demonstrate that the online spectral learning algorithm with a large number of random Fourier features does an excellent job matching the performance of the HSE-HMM, and suggest that the online spectral learning algorithm is a viable alternative to HSE-HMMs when the amount of training data grows large.

Modeling Video

Next we look at the problem of mapping from video: we collected a sequence of 11,000 160×120 grayscale frames at 24 fps in an indoor environment (a camera circling a conference room, occasionally switching directions; each full circuit took about 400 frames). This data was collected by hand, so the camera's trajectory is quite noisy. The high frame rate and complexity of the video mean that learning an accurate model requires a very large dataset. Unfortunately, a dataset of this magnitude makes learning an HSE-HMM difficult or impossible: e.g., the similar but less complex example of Song et al. used only 1500 frames.

Instead, we used random Fourier features and an online TPSR to approximate a HSE-HMM with Gaussian RBF kernels. We used tests and histories based on 400 sequential frames from the video, generated 100,000 random features, and learned a 50-dimensional TPSR. To duplicate this setup, the batch TPSR algorithm would have to find the SVD of a $100{,}000 \times 100{,}000$ matrix; by contrast, we can efficiently update our parameters by incorporating 100,000-element feature vectors one at a time and maintaining 50×50 and $50 \times 100{,}000$ matrices.

Figure 3 shows our results. The final learned model does a surprisingly good job at capturing the major features of this environment, including both the continuous location of the camera and the discrete direction of motion (either clockwise or counterclockwise). Furthermore, the fact that a general-purpose online algorithm learns these manifolds is a powerful result: we are essentially performing simultaneous localization and mapping in a difficult loop closing scenario, *without* any prior knowledge (even, say, that the environment is three-dimensional, or whether the sensor is a camera, a laser rangefinder, or something else).

Conclusion

We presented spectral learning algorithms for TPSR models of partially-observable nonlinear dynamical systems. In particular, we showed how to *update* the parameters of a TPSR

given new batches of data, and built on these updates to develop an efficient *online* spectral learning algorithm. We also showed how to use random projections in conjunction with TPSRs to efficiently approximate HSE-HMMs. The combination of random projections and online updates allows us to take advantage of powerful Hilbert space embeddings while handling training data sets that are orders of magnitude larger than previous methods, and therefore, to learn models that are too complex for previous methods.

Acknowledgements

Byron Boots and Geoffrey J. Gordon were supported by ONR MURI grant number N00014-09-1-1052. Byron Boots was supported by the NSF under grant number EEEC-0540865. We would additionally like to thank Sajid Siddiqi and David Wingate for helpful discussions.

References

Boots, B., and Gordon, G. 2010. Predictive state temporal difference learning. In Lafferty, J.; Williams, C. K. I.; Shawe-Taylor, J.; Zemel, R.; and Culotta, A., eds., *Advances in Neural Information Processing Systems 23*. 271–279.

Boots, B.; Siddiqi, S. M.; and Gordon, G. J. 2010. Closing the learning-planning loop with predictive state representations. In *Proceedings of Robotics: Science and Systems VI*.

Bowling, M.; McCracken, P.; James, M.; Neufeld, J.; and Wilkinson, D. 2006. Learning predictive state representations using non-blind policies. In *Proc. ICML*.

Brand, M. 2006. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and its Applications* 415(1):20–30.

Hsu, D.; Kakade, S.; and Zhang, T. 2009. A spectral algorithm for learning hidden Markov models. In *COLT*.

Jaeger, H. 2000. Observable operator models for discrete stochastic time series. *Neural Computation* 12:1371–1398.

Katayama, T. 2005. *Subspace Methods for System Identification*. Springer-Verlag.

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

Pearl, J. 2000. *Causality: models, reasoning, and inference*. Cambridge University Press.

Rabiner, L. R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* 77(2):257–285.

Rahimi, A., and Recht, B. 2007. Random features for large-scale kernel machines. In *Neural Infomration Processing Systems*.

Rosencrantz, M.; Gordon, G. J.; and Thrun, S. 2004. Learning low dimensional predictive representations. In *Proc. ICML*.

Siddiqi, S.; Boots, B.; and Gordon, G. J. 2010. Reduced-rank hidden Markov models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS-2010)*.

Singh, S.; James, M.; and Rudary, M. 2004. Predictive state representations: A new theory for modeling dynamical systems. In *Proc. UAI*.

Soatto, S., and Chiuso, A. 2001. Dynamic data factorization. Technical Report UCLA-CSD 010001, UCLA.

Sondik, E. J. 1971. The optimal control of partially observable Markov processes. PhD. Thesis, Stanford University.

Song, L.; Boots, B.; Siddiqi, S. M.; Gordon, G. J.; and Smola, A. J. 2010. Hilbert space embeddings of hidden Markov models. In *Proc. 27th Intl. Conf. on Machine Learning (ICML)*.

Van Overschee, P., and De Moor, B. 1996. *Subspace Identification for Linear Systems: Theory, Implementation, Applications*. Kluwer.