Learning to Filter with Predictive State Inference Machines

Wen Sun[†] WENSUN@CS.CMU.EDU
Arun Venkatraman[†] ARUNVENK@CS.CMU.EDU
Byron Boots[*] BBOOTS@CC.GATECH.EDU
J. Andrew Bagnell[†] DBAGNELL@RI.CMU.EDU
[†]Robotics Institute, Carnegie Mellon University, USA
[*]College of Computing, Georgia Institute of Technology, USA

Abstract

Latent state space models are a fundamental and widely used tool for modeling dynamical systems. However, they are difficult to learn from data and learned models often lack performance guarantees on inference tasks such as filtering and prediction. In this work, we present the PREDICTIVE STATE INFERENCE MACHINE (PSIM), a data-driven method that considers the inference procedure on a dynamical system as a composition of predictors. The key idea is that rather than first learning a latent state space model, and then using the learned model for inference, PSIM directly learns predictors for inference in predictive state space. We provide theoretical guarantees for inference, in both realizable and agnostic settings, and showcase practical performance on a variety of simulated and real world robotics benchmarks.

1. Introduction

Data driven approaches to modeling dynamical systems is important in applications ranging from time series forecasting for market predictions to filtering in robotic systems. The classic generative approach is to assume that each observation is correlated to the value of a latent state and then model the dynamical system as a graphical model, or latent state space model, such as a *Hidden Markov Model* (HMM). To learn the parameters of the model from observed data, Maximum Likelihood Estimation (MLE) based methods attempt to maximize the likelihood of the observations with respect to the parameters. This approach has proven to be highly successful in some applications (Coates et al., 2008; Roweis & Ghahramani, 1999), but has at least two shortcomings. First, it may be difficult to find an appropriate parametrization for the latent states. If the model is parametrized incorrectly, the learned model may exhibit poor performance on inference tasks such as Bayesian filtering or predicting multiple time steps into the future. Second, learning a latent state space model is difficult. The MLE objective is non-convex and finding the globally optimal solution is often computationally infeasible. Instead, algorithms such as Expectation-Maximization (EM) are used to compute locally optimal solutions. Although the maximizer of the likelihood objective can promise good performance guarantees when it is used for inference, the locally optimal solutions returned by EM typically do not have any performance guarantees.

Spectral Learning methods are a popular alternative to MLE for learning models of dynamical systems (Boots, 2012; Boots et al., 2011; Hsu et al., 2009; Hefny et al., 2015). This family of algorithms provides theoretical guarantees on discovering the global optimum for the model parameters under the assumptions of infinite training data and realizability. However, in the non-realizable setting — i.e. model mismatch (e.g., using learned parameters of a Linear Dynamical System (LDS) model for a non-linear dynamical system) — these algorithms lose any performance guarantees on using the learned model for filtering or other inference tasks. For example, Kulesza et al. (2014) shows when the model rank is lower than the rank of the underlying dynamical system, the inference performance of the learned model may be arbitrarily bad.

Both EM and spectral learning suffer from limited theoretical guarantees: from model mismatch for spectral methods, and from computational hardness for finding the global optimality of non-convex objectives for MLE-based methods. In scenarios where our ultimate goal is to infer some quantity from observed data, a natural solution is to skip the step of learning a model, and instead directly optimize the inference procedure. Toward this end, we generalize the *supervised message-passing Inference Machine* approach

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48. Copyright 2016 by the author(s).

of Ross et al. (2011b); Ramakrishna et al. (2014); Lin et al. (2015). Inference machines do not parametrize the graphical model (e.g., design of potential functions) and instead directly train predictors that use incoming messages and local features to predict outgoing messages via black-box supervised learning algorithms. By combining the model and inference procedure into a single object — an *Inference Machine* — we directly optimize the end-to-end quality of inference. This unified perspective of learning and inference enables stronger theoretical guarantees on the inference procedure: the ultimate task that we care about.

One of the principal limitations of inference machines is that they require supervision. If we only have access to observations during training, then there is no obvious way to apply the inference machine framework to graphical models with latent states. To generalize Inference Machines to dynamical systems with latent states, we leverage ideas from *Predictive State Representations* (PSRs) (Littman et al., 2001; Singh et al., 2004; Boots et al., 2011; Hefny et al., 2015). In contrast to latent variable representations of dynamical systems, which represent the belief state as a probability distribution over the unobserved state space of the model, PSRs instead maintain an *equivalent* belief over sufficient features of future observations.

We propose PREDICTIVE STATE INFERENCE MACHINES (PSIMs), an algorithm that treats the inference procedure (filtering) on a dynamical system as a composition of predictors. Our procedure takes the current predictive state and the latest observation from the dynamical system as inputs and outputs the next predictive state (Fig. 1). Since we have access to the observations at training, this immediately brings the supervision back to our learning problem — we quantify the loss of the predictor by measuring the likelihood that the actual future observations are generated from the predictive state computed by the learner. PSIM allows us to treat filtering as a general supervised learning problem handed-off to a black box learner of our choosing. The complexity of the learner naturally controls the trade-off between computational complexity and prediction accuracy. We provide two algorithms to train a PSIM. The first algorithm learns a sequence of non-stationary filters which are provably consistent in the realizable case. The second algorithm is more data efficient and learns a stationary filter which has reduction-style performance guarantees.

The three main contributions of our work are: (1) we provide a reduction of unsupervised learning of latent state space models to the supervised learning setting by leveraging PSRs; (2) our algorithm, PSIM, directly minimizes error on the inference task—closed loop filtering; (3) PSIM works for general *non-linear* latent state space models and guarantees filtering performance even in agnostic setting.

2. Related Work

In addition to the MLE-based approaches and the spectral learning approaches mentioned in Sec. 1, there are several supervised learning approaches related to our work. Data as Demonstrator (DaD) (Venkatraman et al., 2015) applies the Inference Machine idea to fully observable Markov chains, and directly optimizes the *open-loop* forward prediction accuracy. In contrast, we aim to design an *unsupervised* learning algorithm for *latent* state space models (e.g., HMMs and LDSs) to improve the accuracy of *closed loop* prediction–Bayesian filtering. It is unclear how to apply DaD to learning a Bayesian filter. Autoregressive models (Wei, 1994) on k-th order fully observable Markov chains (AR-k) use the most recent k observations to predict the next observation. The AR model is not suitable for latent state space models since the beliefs of latent states are conditioned on the *entire* history. Learning mappings from entire history to next observations is unreasonable and one may need to use a large k in practice. A large k, however, increases the difficulty of the learning problem (i.e., requires large computational and samples complexity).

In summary, our work is conceptually different from DaD and AR models in that we focus on unsupervised learning of latent state space models. Instead of simply predicting next observation, we focus on predictive state—a distribution of future observations, as an alternative representation of the beliefs of latent states.

3. Preliminaries

We consider uncontrolled discrete-time time-invariant dynamical systems. At every time step t, the latent state of the dynamical system, $s_t \in \mathbb{R}^m$, stochastically generates an observation, $x_t \in \mathbb{R}^n$, from an observation model $P(x_t|s_t)$. The stochastic transition model $P(s_{t+1}|s_t)$ computes the predictive distribution of states at $t+1$ given the state at time t. We define the belief of a latent state s_{t+1} as the distribution of s_{t+1} given all the past observations up to time step t: $\{x_1, ..., x_t\}$, which we denote as h_t.

3.1. Belief Propagation in Latent State Space Models

Let us define b_t as the belief $P(s_t|h_{t-1})$. When the transition model $P(s_{t+1}|s_t)$ and observation model $P(x_t|s_t)$ are known, the belief b_t can be computed by a special-case of message passing called forward belief propagation:

$$b_{t+1} = \frac{1}{P(x_t|h_{t-1})} \int_{s_t} b_t P(s_{t+1}|s_t) P(x_t|s_t) ds_t. \quad (1)$$

The above equation essentially maps the belief b_t and the current observation x_t to the next belief b_{t+1}.

Consider the following linear dynamical system:

$$\begin{aligned} s_{t+1} &= As_t + \epsilon_s, \quad \epsilon_s \sim \mathcal{N}(0, Q), \\ x_t &= Cs_t + \epsilon_x, \quad \epsilon_x \sim \mathcal{N}(0, R), \end{aligned} \tag{2}$$

where $A \in \mathbb{R}^{m\times m}$ is the transition matrix, $C \in \mathbb{R}^{n\times m}$ is the observation matrix, and $Q \in \mathbb{R}^{m\times m}$ and $R \in \mathbb{R}^{n\times n}$ are noise covariances. The Kalman Filter (Van Overschee & De Moor, 2012) update implements the belief update in Eq. 1. Since $P(s_t|h_{t-1})$ is a Gaussian distribution, we simply use the mean $\hat{s}_t$ and the covariance Σ_t to represent $P(s_t|h_{t-1})$. The Kalman Filter update step can then be viewed as a function that maps $(\hat{s}_t, \Sigma_t)$ and the observation x_t to $(\hat{s}_{t+1}, \Sigma_{t+1})$, which is a nonlinear map.

Given the sequences of observations $\{x_t\}_t$ generated from the linear dynamical system in Eq. 2, there are two common approaches to recover the parameters A, C, Q, R. Expectation-Maximization (EM) attempts to maximize the likelihood of the observations with respect to parameters (Roweis & Ghahramani, 1999), but suffers from locally optimal solutions. The second approach relies on Spectral Learning algorithms to recover A, C, Q, R up to a linear transformation (Van Overschee & De Moor, 2012).[1] Spectral algorithms have two key characteristics: 1) they use an observable state representation; and 2) they rely on method-of-moments for parameter identification instead of likelihood. Though spectral algorithms can promise global optimality in certain cases, this desirable property does not hold under model mismatch (Kulesza et al., 2014). In this case, using the learned parameters for filtering may result in poor filtering performance.

3.2. Predictive State Representations

Recently, predictive state representations and observable operator models have been used to learn from, filter on, predict, and simulate time series data (Jaeger, 2000; Littman et al., 2001; Singh et al., 2004; Boots et al., 2011; Boots & Gordon, 2011; Hefny et al., 2015). These models provide a compact and complete description of a dynamical system that is easier to learn than latent variable models, by representing state as a set of predictions of observable quantities such as future observations.

In this work, we follow a predictive state representation (PSR) framework and define state as the distribution of $f_t = [x_t^T, ..., x_{t+k-1}^T]^T \in \mathbb{R}^{kn}$, a k-step fixed-sized time window of *future* observations $\{x_t, ..., x_{t+k-1}\}$ (Hefny et al., 2015). PSRs assume that if we can predict everything about f_t at time-step t (e.g., the distribution of f_t), then we also know everything there is to know about the state of a dynamical system at time step t (Singh et al., 2004). We

[1]Sometimes called *subspace identification* (Van Overschee & De Moor, 2012) in the linear time-invariant system context.

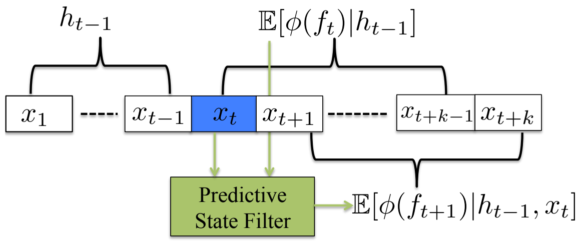


Figure 1. Filtering with predictive states for a k-observable system. At time step t, the filter uses the belief $\mathbb{E}[\phi(f_t)|h_{t-1}]$ and the latest observation x_t as feedback, outputs the next belief $\mathbb{E}[\phi(f_{t+1})|h_{t-1}, x_t]$.

assume that systems we consider are k-observable[2] for $k \in \mathbb{N}^+$: there is a bijective function that maps $P(s_t|h_{t-1})$ to $P(f_t|h_{t-1})$. For convenience of notation, we will present our results in terms of k-observable systems, where it suffices to select features from the next k observations.

Following Hefny et al. (2015), we define the predictive state at time step t as $\mathbb{E}[\phi(f_t)|h_{t-1}]$ where ϕ is some feature function that is sufficient for the distribution $P(f_t|h_{t-1})$. The expectation is taken with respect to the distribution $P(f_t|h_{t-1})$: $\mathbb{E}[\phi(f_t)|h_{t-1}] = \int_{f_t} \phi(f_t)P(f_t|h_{t-1})df_t$. The conditional expectation can be understood as a function of which the input is the random variable h_{t-1}. For example, we can set $\mathbb{E}[\phi(f)|h_{t-1}] = \mathbb{E}[f, ff^T|h_{t-1}]$ if $P(f_t|h_{t-1})$ is a Gaussian distribution (e.g., linear dynamical system in Eq. 2); or we can set $\phi(f) = [x_t \otimes ... \otimes x_{t+k-1}]$ if we are working on a discrete models (discrete latent states and discrete observations), where x_t is an indicator vector representation of the observation and $\otimes$ is the tensor product. Therefore, we assume that there exists a bijective function mapping $P(f|h_{t-1})$ to $\mathbb{E}[\phi(f_t)|h_{t-1}]$. For any test f_t', we can compute the probability of $P(f_t'|h_{t-1})$ by simply using the predictive state $\mathbb{E}[\phi(f_t)|h_{t-1}]$. Note that the mapping from $\mathbb{E}[\phi(f_t)|h_{t-1}]$ to $P(f_t'|h_{t-1})$ is not necessarily linear.

To filter from the current predictive state $\mathbb{E}[\phi(f_t)|h_{t-1}]$ to the next state $\mathbb{E}[\phi(f_{t+1})|h_t]$ conditioned on the most recent observation x_t (see Fig. 1 for an illustration), PSRs additionally define an extended state $\mathbb{E}[\zeta(f_t, x_{t+k})|h_{t-1}] = \int_{(f_t, x_{t+k})} \zeta(f_t, x_{t+k})P(f_t, x_{t+k}|h_{t-1})d(f_t, x_{t+k})$, where ζ is another feature function for the future observations f_t and one more observation x_{t+k}. PSRs explicitly assume there exists a linear relationship between $\mathbb{E}[\phi(f_t)|h_{t-1}]$ and $\mathbb{E}[\zeta(f_t, x_{t+k})|h_{t-1}]$, which can be learned by *Instrumental Variable Regression* (IVR) (Hefny et al., 2015). PSRs then additionally assume a nonlinear conditioning operator that can compute the next predictive state with the extended state and the latest observation as inputs.

[2]This assumption allows us to avoid the cryptographic hardness of the general problem (Hsu et al., 2009).

4. Predictive State Inference Machines

The original Inference Machine framework reduces the problem of learning graphical models to solving a set of classification or regression problems, where the learned classifiers mimic message passing procedures that output marginal distributions for the nodes in the model (Langford et al., 2009; Ross et al., 2011b; Bagnell et al., 2010). However, Inference Machines cannot be applied to learning latent state space models (unsupervised learning) since we do not have access to hidden states' information.

We tackle this problem with predictive states. By using an observable representation for state, observations in the training data can be used for supervision in the inference machine. More formally, instead of tracking the hidden state s_t, we focus on the corresponding predictive state $\mathbb{E}[\phi(f_t)|h_{t-1}]$. Assuming that the given predictive state $\mathbb{E}[\phi(f_t)|h_{t-1}]$ can reveal the probability $P(f_t|h_{t-1})$, we use the training data f_t to quantify how good the predictive state is by computing the likelihood of f_t. The goal is to learn an operator F (the green box in Fig. 1) which *deterministically* passes the predictive states forward in time conditioned on the latest observation:

$$\mathbb{E}[\phi(f_{t+1})|h_t] = F\Big(\mathbb{E}[\phi(f_t)|h_{t-1}], x_t\Big), \tag{3}$$

such that the likelihood of the observations $\{f_t\}_t$ being generated from the sequence of predictive states $\{\mathbb{E}[\phi(f_t)|h_{t-1}]\}_t$ is maximized. In the standard PSR framework, the predictor F can be regarded as the composition of the linear mapping (from predictive state to extended state) and the conditioning operator. Below we show if we can correctly filter with predictive states, then this is equivalent to filtering with latent states as in Eq. 1.

4.1. Predictive State Propagation

The belief propagation in Eq. 1 is for latent states s_t. We now describe the corresponding belief propagation for updating the predictive state from $\mathbb{E}[\phi(f_t)|h_{t-1}]$ to $\mathbb{E}[\phi(f_{t+1})|h_t]$ conditioned on the new observation x_t. Since we assume that the mapping from $P(s_t|h_{t-1})$ to $P(f_t|h_{t-1})$ and the mapping from $P(f_t|h_{t-1})$ to $\mathbb{E}[\phi(f_t)|h_{t-1}]$ are both bijective, there must exist a bijective map q and its inverse q^{-1} such that $\mathrm{q}(P(s_t|h_{t-1})) = \mathbb{E}[\phi(f_t)|h_{t-1}]$ and $\mathrm{q}^{-1}(\mathbb{E}[\phi(f_t)|h_{t-1}]) = P(s_t|h_{t-1})$,[3] then the message passing in Eq. 1 is also equivalent to:

$$\begin{aligned}&\mathbb{E}[\phi(f_{t+1})|h_t] = \mathrm{q}(P(s_{t+1}|h_t)) \qquad (4)\\&= \mathrm{q}\Big(\int_{s_t} \frac{P(s_t|h_{t-1})P(s_{t+1}|s_t)P(x_t|s_t)}{P(x_t|h_{t-1})} ds_t\Big)\\&= \mathrm{q}\Big(\int_{s_t} \frac{\mathrm{q}^{-1}(\mathbb{E}[\phi(f_t)|h_{t-1}])P(s_{t+1}|s_t)P(x_t|s_t)}{P(x_t|h_{t-1})} ds_t\Big)\end{aligned}$$

[3]The composition of two bijective functions is bijective.

Eq. 4 explicitly defines the map F that takes the inputs of $\mathbb{E}[\phi(f_t)|h_{t-1}]$ and x_t and outputs $\mathbb{E}[\phi(f_{t+1})|h_t]$. This map F could be non-linear since it depends the transition model $P(s_{t+1}|s_t)$, observation model $P(x_t|s_t)$ and function q, which are all often complicated, non-linear functions in real dynamical systems. We do not place any parametrization assumptions on the transition and observation models. Instead, we parametrize and restrict the class of predictors to encode the underlying dynamical system and aim to find a predictor F from the restricted class. We call this framework for inference the PREDICTIVE STATE INFERENCE MACHINE (PSIM).

PSIM is different from PSRs in the following respects: (1) PSIM collapses the two steps of PSRs (predict the extended state and then condition on the latest observation) into one step—as an Inference Machine—for closed-loop update of predictive states; (2) PSIM directly targets the filtering task and has theoretical guarantees on the filtering performance; (3) unlike PSRs where one usually needs to utilize linear PSRs for learning purposes (Boots et al., 2011), PSIM can generalize to non-linear dynamics by leveraging non-linear regression or classification models.

Imagine that we can perform belief propagation with PSIM in predictive state space as shown in Eq. 4, then this is *equivalent* to classic filter with latent states as shown in Eq. 1. To see this, we can simply apply q^{-1} on both sides of the above equation Eq. 4, which exactly reveals Eq. 1. We refer readers to the Appendix for a detailed case study of the stationary Kalman Filter, where we explicitly show this equivalence. Thanks to this equivalence, we can learn accurate inference machines, even for partially observable systems. We now turn our focus on learning the map F in the predictive state space.

4.2. Learning Non-stationary Filters with Predictive States

For notational simplicity, let us define trajectory as τ, which is sampled from a unknown distribution $\mathcal{D}_\tau$. We denote the predictive state as $m_t = \mathbb{E}[\phi(f_t)|h_{t-1}]$. We use $\hat{m}_t$ to denote an approximation of m_t. Given a predictive state m_t and a noisy observation f_t conditioned on the history h_{t-1}, we let the loss function[4] $d(m_t, f_t) = \|m_t - \phi(f_t)\|_2^2$. This squares loss function can be regarded as matching moments. For instance, in the stationary Kalman filter setting, we could set $m_t = \mathbb{E}[f_t|h_{t-1}]$ and $d(m_t, f_t) = \|m_t - f_t\|_2^2$ (matching the first moment).

[4]Squared loss in an example Bregman divergence of which there are others that are optimized by the conditional expectation (Banerjee et al., 2005). We can design $d(m_t, f_t)$ as negative log-likelihood, as long as it can be represented as a Bregman divergence (e.g., negative log-likelihood of distributions in exponential family).

Algorithm 1 PREDICTIVE STATE INFERENCE MACHINE (PSIM) with Forward Training

```
1: Input: $M$ independent trajectories $\tau_i, 1 \leq i \leq M$;
2: Set $\hat{m}_1 = \frac{1}{M}\sum_{i=1}^{M}\phi(f_1^i)$;
3: Set $\hat{m}_1^i = \hat{m}_1$ for trajectory $\tau_i, 1 \leq i \leq M$;
4: for t = 1 to T do
5:     For each trajectory $\tau_i$, add the input $z_t^i = (\hat{m}_t^i, x_t^i)$ to $D_t$ as feature variables and the corresponding $f_{t+1}^i$ to $D_t$ as the targets;
6:     Train a hypothesis $F_t$ on $D_t$ to minimize the loss $d(F(z), f)$ over $D_t$;
7:     For each trajectory $\tau_i$, roll out $F_1, ..., F_t$ along the trajectory (Eq. 6) to compute $\hat{m}_{t+1}^i$;
8: end for
9: Return: the sequence of hypothesis $\{F_t\}_{t=1}^N$.
```

We first present a algorithm for learning non-stationary filters using *Forward Training* (Ross & Bagnell, 2010) in Alg. 1. Forward Training learns a non-stationary filter for each time step. Namely, at time step t, forward training learns a hypothesis F_t that approximates the filtering procedure at time step t: $\hat{m}_{t+1} = F_t(\hat{m}_t, x_t)$, where $\hat{m}_t$ is computed by $F_{t-1}(\hat{m}_{t-1}, x_{t-1})$ and so on. Let us define $\hat{m}_t^i$ as the predictive state computed by rolling out $F_1, .., F_{t-1}$ on trajectory τ_i to time step $t-1$. We define f_t^i as the next k observations starting at time step t on trajectory τ_i. At each time step t, the algorithm collects a set of training data D_t, where the feature variables z_t consist of the predictive states $\hat{m}_t^i$ from the previous hypothesis F_{t-1} and the local observations x_t^i, and the targets consist of the corresponding future observations f_{t+1}^i across all trajectories τ_i. It then trains a new hypothesis F_t over the hypothesis class $\mathcal{F}$ to minimize the loss over dataset D_t.

PSIM with Forward Training aims to find a good sequence of hypotheses $\{F_t\}$ such that:

$$\min_{F_1\in\mathcal{F},...F_T\in\mathcal{F}} \mathbb{E}_{\tau\sim\mathcal{D}_\tau}\Big[\frac{1}{T}\sum_{t=1}^{T} d(F_t(\hat{m}_t^\tau, x_t^\tau), f_{t+1}^\tau)\Big], \quad (5)$$

$$s.t.\;\; \hat{m}_{t+1}^\tau = F_t(\hat{m}_t^\tau, x_t^\tau), \forall t \in [1, T-1], \quad (6)$$

where $\hat{m}_1 = \arg\min_m \sum_{t=1}^{M} d(m, f_1^i)$, which is equal to $\frac{1}{T}\sum_{i=1}^{T}\phi(f_i^i)$. Let us define ω_t as the joint distribution of feature variables z_t and targets f_{t+1} after rolling out $F_1, ..., F_{t-1}$ on the trajectories sampled from $\mathcal{D}_\tau$. Under this definition, the filter error defined above is equivalent to $\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{(z,f)\sim\omega_t}\Big[d(F_t(z), f)\Big]$. Note essentially the dataset D_t collected by Alg. 1 at time step t forms a finite sample estimation of ω_t.

To analyze the consistency of our algorithm, we assume every learning problem can be solved perfectly (risk minimizer finds the Bayes optimal) (Langford et al., 2009). We first show that under infinite many training trajectories, and in realizable case — the underlying true filters $F_1^*, ..., F_T^*$ are in the hypothesis class $\mathcal{F}$, Alg. 1 is consistent:

Theorem 4.1. *With infinite many training trajectories and in the realizable case, if all learning problems are solved perfectly, the sequence of predictors $F_1, F_2, ..., F_T$ from Alg. 1 can generate exact predictive states $\mathbb{E}[\phi(f_t^\tau)|h_{t-1}^\tau]$ for any trajectory $\tau \sim \mathcal{D}_\tau$ and $1 \leq t \leq T$.*

We include all proofs in the appendix. Next for the agnostic case, we show that Alg. 1 can still achieve a reasonable upper bound. Let us define $\epsilon_t = \min_{F\sim\mathcal{F}}\mathbb{E}_{(z,f)\sim\omega_t}[d(F(z), f)]$, which is the minimum batch training error under the distribution of inputs resulting from hypothesis class $\mathcal{F}$. Let us define $\epsilon_{\max} = \max_t\{\epsilon_t\}$. Under infinite many training trajectories, even in the model agnostic case, we have the following guarantees for filtering error for Alg. 1:

Theorem 4.2. *With infinite many training trajectories, for the sequence $\{F_t\}_t$ generated by Alg. 1, we have:*

$$\mathbb{E}_{\tau\sim\mathcal{D}_\tau}\Big[\frac{1}{T}\sum_{t=1}^{T} d(F_t(\hat{m}_t^\tau, x_t^\tau), f_{t+1}^\tau)\Big] = \frac{1}{T}\sum_t \epsilon_t \leq \epsilon_{max}.$$

Theorem. 4.2 shows that the filtering error is upper-bounded by the average of the minimum batch training errors from each step. If we have a rich class of hypotheses and small noise (e.g., small Bayes error), ϵ_t could be small.

To analyze finite sample complexity, we need to split the dataset into T disjoint sets to make sure that the samples in the dataset D_t are i.i.d (see details in Appendix). Hence we reduce forward training to T independent passive supervised learning problems. We have the following agnostic theoretical bound:

Theorem 4.3. *With M training trajectories, for any $F_t^* \in \mathcal{F}, \forall t$, we have with probability at least $1-\delta$:*

$$\mathbb{E}_{\tau\sim\mathcal{D}_\tau}\Big[\frac{1}{T}\sum_{t=1}^{T} d(F_t(\hat{m}_t^\tau, x_t^\tau), f_{t+1}^\tau)\Big]$$
$$\leq \mathbb{E}_{\tau\sim\mathcal{D}_\tau}\Big[\frac{1}{T}\sum_{t=1}^{T} d(F_t^*(\hat{m}_t^\tau, x_t^\tau), f_{t+1}^\tau)\Big]$$
$$+ 4\nu\bar{\mathcal{R}}(\mathcal{F}) + 2\sqrt{\frac{T\ln(T/\delta)}{2M}}, \quad (7)$$

where $v = \sup_{F,z,f} 2\|F(z) - f\|_2$, $\bar{\mathcal{R}}(\mathcal{F}) = \frac{1}{T}\sum_{t=1}^{T}\mathcal{R}_t(\mathcal{F}))$ and $\mathcal{R}_t(\mathcal{F})$ is the Rademacher number of $\mathcal{F}$ under ω_t.

As one might expect, the learning problem becomes harder as T increases, however our finite sample analysis shows the average filtering error grows sublinearly as $\tilde{O}(\sqrt{T})$.

Although Alg. 1 has nice theoretical properties, one shortcoming is that it is not very data efficient. In practice, it is possible that we only have small number of training trajectories but each trajectory is long (T is big). This means that we may have few training data samples (equal to the number of trajectories) for learning hypothesis F_t. Also, instead of learning non-stationary filters, we often prefer to learn a stationary filter such that we can filter indefinitely. In the next section, we present a different algorithm that utilizes *all* of the training data to learn a stationary filter.

4.3. Learning Stationary Filters with Predictive States

The optimization framework for finding a good stationary filter F is defined as:

$$\min_{F\in\mathcal{F}} \mathbb{E}_{\tau\sim\mathcal{D}_\tau}\frac{1}{T}\sum_{t=1}^{T} d(F(\hat{m}_t, x_t), f_{t+1}), \tag{8}$$

$$s.t \quad \hat{m}_{t+1} = F(\hat{m}_t, x_t), \forall t\in[1, T-1], \tag{9}$$

where $\hat{m}_1 = \arg\min_m \sum_{t=1}^{M} d(m, f_1^i) = \frac{1}{T}\sum_{i=1}^{T}\phi(f_i^i)$. Note that the above objective function is non-convex, since $\hat{m}_t$ is computed recursively and in fact is equal to $F(...F(F(\hat{m}_1, x_1), x_2)...)$, where we have $t-1$ nested F. As we show experimentally, optimizing this objective function via Back-Propagation likely leads to local optima. Instead, we optimize the above objective function using an iterative approach called Dataset Aggregation (DAgger) (Ross et al., 2011a) (Alg. 2). Due to the non-convexity of the objective, DAgger also will not promise global optimality. But as we will show, PSIM with DAgger gives us a sound theoretical bound for filtering error.

Given a trajectory τ and hypothesis F, we define $\hat{m}_t^{\tau,F}$ as the predictive belief generated by F on τ at time step t. We also define $z_t^{\tau,F}$ to represent the feature variables $(\hat{m}_t^{\tau,F}, x_t^\tau)$. At iteration n, Alg. 2 rolls out the predictive states using its current hypothesis F_n (Eq. 9) on all the given training trajectories (Line. 2). Then it collects all the feature variables $\{(\hat{m}_t^{i,F_n}, x_t^i)\}_{t,i}$ and the corresponding target variables $\{f_{t+1}^i\}_{t,i}$ to form a new dataset D_n' and aggregates it to the original dataset D_{n-1}. Then a new hypothesis F_n is learned from the aggregated dataset D_n by minimizing the loss $d(F(z), f)$ over D_n.

Alg. 2 essentially utilizes DAgger to optimize the non-convex objective in Eq. 8. By using DAgger, we can guarantee a hypothesis that, when used during filtering, performs nearly as well as when performing regression on the aggregate dataset D_N. In practice, with a rich hypothesis class $\mathcal{F}$ and small noise (e.g., small Bayes error), small regression error is possible. We now analyze the filtering performance of PSIM with DAgger below.

Let us fix a hypothesis F and a trajectory τ, we define $\omega_{F,\tau}$ as the uniform distribution of (z, f): $\omega_{F,\tau} = \mathcal{U}\Big[(z_1^{\tau,F}, f_2^\tau), ..., (z_T^{\tau,F}, f_{T+1}^\tau)\Big]$. Now we can rewrite the filtering error in Eq. 8 as $L(F) = \mathbb{E}_\tau[\mathbb{E}_{z,f\sim\omega_{F,\tau}}[d(F(z), f)]|\tau]$. Let us define the loss function for any predictor F at iteration n of Alg. 2 as:

Algorithm 2 PREDICTIVE STATE INFERENCE MACHINE (PSIM) with DAgger

```
1: Input: $M$ independent trajectories $\tau_i$, $1\le i\le M$;
2: Initialize $D_0 \leftarrow \emptyset$ and initalize $F_0$ to be any hypothesis in $\mathcal{F}$;
3: Initialize $\hat{m}_1 = \frac{1}{M}\sum_{i=1}^{M}\phi(f_1^i)$
4: for n = 0 to N do
5:     Use $F_n$ to perform belief propagation (Eq. 9) on trajectory $\tau_i$, $1\le i\le M$
6:     For each trajectory $\tau_i$ and each time step $t$, add the input $z_t^i = (m_t^{i,F_n}, x_t^i)$ encountered by $F_n$ to $D'_{n+1}$ as feature variables and the corresponding $f_{t+1}^i$ to $D'_{n+1}$ as the targets ;
7:     Aggregate dataset $D_{n+1} = D_n \cup D'_{n+1}$;
8:     Train a new hypothesis $F_{n+1}$ on $D_{n+1}$ to minimize the loss $d(F(m,x), f)$;
9: end for
10: Return: the best hypothesis $\hat{F}\in\{F_n\}_n$ on validation trajectories.
```

$$L_n(F) = \mathbb{E}_\tau[\mathbb{E}_{z,f\sim\omega_{F_n,\tau}}[d(F(z), f)]|\tau]. \tag{10}$$

As we can see, at iteration n, the dataset D_n' that we collect forms an empirical estimate of the loss L_n:

$$\hat{L}_n(F) = \frac{1}{M}\sum_{\tau=1}^{M}\mathbb{E}_{z,f\sim\omega_{F_n,\tau}}\big(d(F(z), f)\big). \tag{11}$$

We first analyze the algorithm under the assumption that $M=\infty$, $\hat{L}_n(F) = L_n(F)$. Let us define *Regret* γ_N as: $\frac{1}{N}\sum_{n=1}^{N} L_n(F_n) - \min_{F\in\mathcal{F}}\frac{1}{N}\sum_{n=1}^{N} L_n(F) \le \gamma_N$. We also define the minimum average training error $\epsilon_N = \min_{F\in\mathcal{F}}\frac{1}{N}\sum_{n=1}^{N} L_n(F)$. Alg. 2 can be regarded as running the Follow the Leader (FTL) (Cesa-Bianchi et al., 2004; Shalev-Shwartz & Kakade, 2009; Hazan et al., 2007) on the sequence of loss functions $\{L_n(F)\}_{n=1}^{N}$. When the loss function $L_n(F)$ is strongly convex with respect to F, FTL is no-regret in a sense that $\lim_{N\to\infty}\gamma_N = 0$. Applying Theorem 4.1 and its reduction to no-regret learning analysis from (Ross et al., 2011a) to our setting, we have the following guarantee for filtering error:

Corollary 4.4. *(Ross et al., 2011a) For Alg. 2, there exists a predictor $\hat{F}\in\{F_n\}_{n=1}^{N}$ such that:*

$$L(\hat{F}) = \mathbb{E}_\tau\big[\mathbb{E}_{z,f\sim\omega_{\hat{F},\tau}}(d(\hat{F}(z), f))|\tau\big] \le \gamma_N + \epsilon_N.$$

As we can see, under the assumption that L_n is strongly convex, as $N\to\infty$, γ_N goes to zero. Hence the filtering error of $\hat{F}$ is upper bounded by the minimum batch training

	N4SID	IVR	PSIM-Linear$_d$	PSIM-Linear$_b$	PSIM-RFF$_d$	Traj. Pwr
Robot Drill Assembly	2.87±0.2	2.39 ±0.1	2.15±0.1	2.54±0.1	**1.80±0.1**	27.90
Motion Capture	7.86± 0.8	6.88± 0.7	5.75±0.5	9.94±2.9	**5.41± 0.5**	107.92
Beach Video Texture	231.33±10.5	213.27±11.5	164.23±8.7	268.73±9.5	**130.53±9.1**	873.77
Flag Video Texture	3.38e3±1.2e2	3.38e3±1.3e2	1.28e3±7.1e1	1.31e3±6.4e1	**1.24e3±9.6e1**	3.73e3

Table 1. Filter error (1-step ahead) and standard deviation on different datasets. We see that using PSIM with DAgger with both RFF and Linear outperforms the spectral methods N4SID and IVR, with the RFF performing better on almost all the datasets. DAgger (20 iterations) trains a better linear regression for PSIM than back-propagation with random initialization (400 epochs). We also give the average trajectory power for the true observations from each dataset.

error that could be achieved by doing regression on D_N within class $\mathcal{F}$. In general the term ϵ_N depends on the noise of the data and the expressiveness of the hypothesis class $\mathcal{F}$. Corollary. 4.4 also shows for fully realizable and noise-free case, PSIM with DAgger finds the optimal filter that drives the filtering error to zero when $N \to \infty$.

The finite sample analysis from (Ross et al., 2011a) can also be applied to PSIM. Let us define $\hat{\epsilon}_N = \min_{F\in\mathcal{F}} \frac{1}{N}\hat{L}_n(F)$, $\hat{\gamma}_N \geq \frac{1}{N}\sum_{n=1}^{N}\hat{L}_n(F_n) - \min_{F\in\mathcal{F}}\frac{1}{N}\sum_{n=1}^{N}\hat{L}_n(F)$, we have:

Corollary 4.5. *(Ross et al., 2011a) For Alg. 2, there exists a predictor $\hat{F} \in \{F_n\}_{n=1}^N$ such that with probability at least $1-\delta$:*

$$L(\hat{F}) = \mathbb{E}_\tau\left[\mathbb{E}_{z,f\sim\omega_{\hat{F},\tau}}(d(\hat{F}(z),f))|\tau\right] \leq \hat{\gamma}_N + \hat{\epsilon}_N + L_{\max}\left(\sqrt{\frac{2\ln(1/\delta)}{MN}}\right). \quad (12)$$

5. Experiments

We evaluate the PSIM on a variety of dynamical system benchmarks. We use two feature functions: $\phi_1(f_t) = [x_t, ..., x_{t+k-1}]$, which stack the k future observations together (hence the message m can be regarded as a prediction of future k observations $(\hat{x}_t, .., \hat{x}_{t+k-1})$), and $\phi_2(f_t) = [x_t, ..., x_{t+k-1}, {x_t}^2, ..., x_{t+k-1}^2]$, which includes second moments (hence m represents a Gaussian distribution approximating the true distribution of future observations). To measure how good the computed predictive states are, we extract $\hat{x}_i$ from $\hat{m}_t$, and evaluate $\|\hat{x}_i - x_i\|_2^2$, the squared distance between the predicted observation $\hat{x}_i$ and the corresponding true observation x_i. We implement PSIM with DAgger using two underlying regression methods: ridge linear regression (**PSIM-Linear**$_d$) and linear ridge regression with Random Fourier Features (**PSIM-RFF**$_d$) (Rahimi & Recht, 2007)[5]. We also test PSIM with back-propagation for linear regression (**PSIM-Linear**$_b$). We compare our approaches to several baselines: Autoregressive models (**AR**), Subspace State Space System Identification (**N4SID**) (Van Overschee & De Moor, 2012), and PSRs implemented with **IVR** (Hefny et al., 2015).

[5]With RFF, PSIM approximately embeds the distribution of f_t into a Reproducing Kernel Hilbert Space.

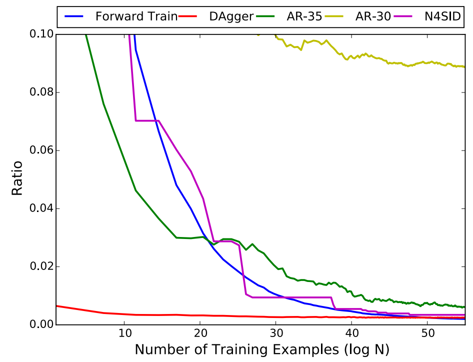


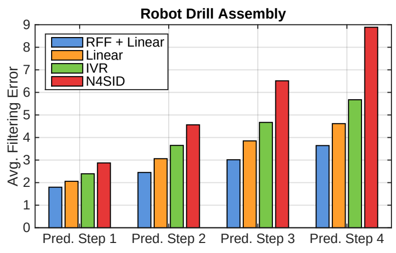
Figure 2. The convergence rate of different algorithms. The ratios (y-axis) are computed as $\log(\frac{e}{e_F})$ for error e from corresponding algorithms. The x-axis is computed as $\log(N)$, where N is the number of trajectories used for training.

5.1. Synthetic Linear Dynamical System

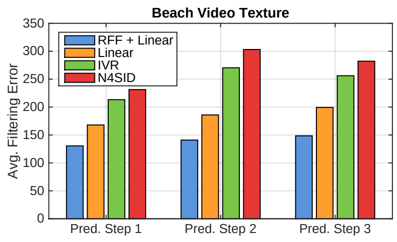
First we tested our algorithms on a synthetic linear dynamical system (Eq. 2) with a 2-dimensional observation x. We designed the system such that it is exactly 2-observable. The sequences of observations are collected from the linear stationary Kalman filter of the LDS (Boots, 2012; Hefny et al., 2015). The details of the LDS are in Appendix.

Since the data is collected from the stationary Kalman filter of the 2-observable LDS, we set $k = 2$ and use $\phi_1(f_t) = [x_t, x_{t+1}]$. Note that the 4-dimensional predictive state $\mathbb{E}[\phi_1(f_t)|h_t]$ will represent the exact conditional distribution of observations (x_t, x_{t+1}) and therefore is equivalent to $P(s_t|h_{t-1})$ (see the detailed case study for LDS in Appendix). With linear ridge regression, we test PSIM with forward training, PSIM with DAgger, and AR models (AR-k) with different lengths (k steps of past observations) of history on this dataset. For each method, we compare the average filtering error e to e_F which is computed by using the underlying linear filter F of the LDS.

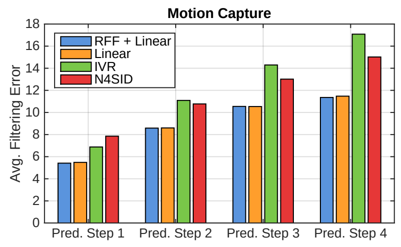
Fig. 2 shows the convergence trends of PSIM with DAgger, PSIM with Forward Training, and AR as the number of training trajectories N increases. The prediction error for AR with $k = 5, 10, 20$ is too big to fit into the plot. PSIM with DAgger performs much better with few training data while Forward Training eventually slightly surpasses DAgger with sufficient data. The AR-k models need long histories to perform well given data gnereated by latent state space models, even for this 2-observable LDS. Note AR-



(a) Robot Drill Assembly



(b) Beach Video Texture



(c) Motion Capture

Figure 3. Filter error for multiple look ahead steps for the future predictions shown for a few of the datasets. We see across datasets that the performance of both IVR and N4SID are significantly worse than using PSIM with either linear or random Fourier feature + linear learner. For some datasets, the nonlinearity of the random Fourier features helps to improve the performance.

35 performs regression in a 70-dimensional feature space (35 past observations), while PSIM only uses 6-d features (4-d predictive state + 2-d current observation). This shows that *predictive state* is a more compact representation of the history and can reduce the complexity of learning problem.

5.2. Real Dynamical Systems

We consider the following three real dynamical systems: (1) *Robot Drill Assembly*: the dataset consists of 96 sensor telemetry traces, each of length 350, from a robotic manipulator assembling the battery pack on a power drill. The 13 dimensional noisy observations consist of the robot arm's 7 joint torques as well as the the 3D force and torque vectors. Note the fixed higher level control policy for the drill assembly task is not given in the observations and must be learned as part of the dynamics; (2) *Human Motion Capture*: the dataset consists of 48 skeletal tracks of 300 timesteps each from a Vicon motion capture system from three human subjects performing walking actions. The observations consist of the 3D positions of the various skeletal parts (e.g. upperback, thorax, clavicle, etc.); (3) *Video Textures*: the datasets consists of one video of *flag* waving and the other one of waves on a *beach*.

For these dynamical systems, we do not test PSIM with Forward Training since our benchmarks have a large number of time steps per trajectory. Throughout the experiments, we set $k = 5$ for all datasets except for video textures, where we set $k = 3$. For each dataset, we randomly pick a small number of trajectories as a validation set for parameter tuning (e.g., ridge, rank for N4SID and IVR, band width for RFF). We partition the whole dataset into ten folds, train all algorithms on 9 folds and test on 1 fold. For the feature function ϕ_1, the average one-step filtering errors and its standard deviations across ten folds are shown in Tab. 1. Our approaches outperforms the two baselines across all datasets. Since the datasets are generated from complex dynamics, PSIM with RFF exhibits better performance than PSIM with Linear. This experimentally supports our theorems suggesting that with powerful regressors, PSIM could perform better. We implement PSIM with back-propagation using Theano with several training approaches: gradient descent with step decay, RMSProp (Tieleman & Hinton, 2012) and AdaDelta (Zeiler, 2012) (see Appendix. E). With random initialization, back-propagation does not achieve comparable performance, except on the flag video, due to local optimality.We observe marginal improvement by using back-propogation to refine the solution from DAgger. This shows PSIM with DAgger finds good models by itself (details in Appendix. E). We also compare these approaches for multi-step look ahead (Fig. 3). PSIM consistently outperforms the two baselines.

To show predictive states with larger k encode more information about latent states, we additionally run PSIM with $k = 1$ using ϕ_1 . PSIM (DAgger) with $k = 5$ outperforms $k = 1$ by 5% for robot assembly dataset, 6% for motion capture, 8% for flag and 32% for beach video. Including belief over longer futures into predictive states can thus capture more information and increase the performance.

For feature function ϕ_2 and $k = 5$, with linear ridge regression, the 1-step filter error achieved by PSIM with DAgger across all datasets are: 2.05 ± 0.08 on Robot Drill Assembly, 5.47 ± 0.42 on motion capture, 154.02 ± 9.9 on beach video, and $1.27e3 \pm 13e1$ on flag video. Comparing to the results shown in the **PSIM-Linear**$_d$ in column of Table. 1, we achieve slightly better performance on all datasets, and noticeably better performance on the beach video texture.

6. Conclusion

We introduced PREDICTIVE STATE INFERENCE MACHINES, a novel approach to directly learn to filter with latent state space models. Leveraging ideas from PSRs, PSIM reduces the unsupervised learning of latent state space models to a supervised learning setting and guarantees filtering performance for general non-linear models in both the realizable and agnostic settings.

Acknowledgements

This material is based upon work supported in part by: DARPA ALIAS contract number HR0011-15-C-0027 and National Science Foundation Graduate Research Fellowship Grant No. DGE-1252522. The authors also thank Geoff Gordon for valuable discussions.

References

Bagnell, J Andrew, Grubb, Alex, Munoz, Daniel, and Ross, Stephane. Learning deep inference machines. *The Learning Workshop*, 2010.

Banerjee, Arindam, Guo, Xin, and Wang, Hui. On the optimality of conditional expectation as a bregman predictor. *Information Theory, IEEE Transactions on*, 51(7):2664–2669, 2005.

Bastien, Frédéric, Lamblin, Pascal, Pascanu, Razvan, Bergstra, James, Goodfellow, Ian J., Bergeron, Arnaud, Bouchard, Nicolas, and Bengio, Yoshua. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

Boots, Byron. *Spectral Approaches to Learning Predictive Representations*. PhD thesis, Carnegie Mellon University, 2012.

Boots, Byron and Gordon, Geoffrey J. Predictive state temporal difference learning. In *NIPS*, 2011.

Boots, Byron, Siddiqi, Sajid M, and Gordon, Geoffrey J. Closing the learning-planning loop with predictive state representations. *The International Journal of Robotics Research*, 30(7): 954–966, 2011.

Cesa-Bianchi, Nicolo, Conconi, Alex, and Gentile, Claudio. On the generalization ability of on-line learning algorithms. *Information Theory, IEEE Transactions on*, 50(9):2050–2057, 2004.

Coates, Adam, Abbeel, Pieter, and Ng, Andrew Y. Learning for control from multiple demonstrations. In *ICML*, pp. 144–151, New York, NY, USA, 2008.

Duchi, John, Hazan, Elad, and Singer, Yoram. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

Hazan, Elad, Agarwal, Amit, and Kale, Satyen. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, 2007.

Hefny, Ahmed, Downey, Carlton, and Gordon, Geoffrey J. Supervised learning for dynamical system learning. In *Advances in Neural Information Processing Systems 28*, 2015.

Hsu, Daniel, M. Kakade, Sham, and Zhang, Tong. A spectral algorithm for learning hidden markov models. In *COLT*, 2009.

Jaeger, Herbert. Observable operator models for discrete stochastic time series. *Neural Computation*, 12(6):1371–1398, 2000.

Kulesza, Alex, Rao, N Raj, and Singh, Satinder. Low-rank spectral learning. In *Proceedings of the 17th Conference on Artificial Intelligence and Statistics*, 2014.

Langford, John, Salakhutdinov, Ruslan, and Zhang, Tong. Learning nonlinear dynamic models. In *Proceedings of the 26th International Conference on Machine Learning (ICML-09)*, pp. 75, 2009.

Lin, Guosheng, Shen, Chunhua, Reid, Ian, and van den Hengel, Anton. Deeply learning the messages in message passing inference. In *Advances in Neural Information Processing Systems*, pp. 361–369, 2015.

Littman, Michael L., Sutton, Richard S., and Singh, Satinder. Predictive representations of state. In *NIPS*, pp. 1555–1561. MIT Press, 2001.

Mohri, Mehryar, Rostamizadeh, Afshin, and Talwalkar, Ameet. *Foundations of machine learning*. MIT press, 2012.

Rahimi, Ali and Recht, Benjamin. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pp. 1177–1184, 2007.

Ramakrishna, Varun, Munoz, Daniel, Hebert, Martial, Bagnell, James Andrew, and Sheikh, Yaser. Pose machines: Articulated pose estimation via inference machines. In *Computer Vision–ECCV 2014*, pp. 33–47. Springer, 2014.

Ross, Stéphane and Bagnell, J. Andrew. Efficient reductions for imitation learning. In *AISTATS*, pp. 661–668, 2010.

Ross, Stéphane, Gordon, Geoffrey J, and Bagnell, J.Andrew. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, 2011a.

Ross, Stephane, Munoz, Daniel, Hebert, Martial, and Bagnell, J Andrew. Learning message-passing inference machines for structured prediction. In *CVPR*, pp. 2737–2744, 2011b.

Roweis, Sam and Ghahramani, Zoubin. A unifying review of linear gaussian models. *Neural computation*, 11(2):305–345, 1999.

Shalev-Shwartz, Shai and Kakade, Sham M. Mind the duality gap: Logarithmic regret algorithms for online optimization. In *NIPS*, pp. 1457–1464, 2009.

Singh, Satinder, James, Michael R., and Rudary, Matthew R. Predictive state representations: A new theory for modeling dynamical systems. In *UAI*, 2004.

Srebro, Nathan, Sridharan, Karthik, and Tewari, Ambuj. Optimistic rates for learning with a smooth loss. *arXiv preprint arXiv:1009.3896*, 2010.

Tieleman, Tijmen and Hinton, Geoffrey. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4: 2, 2012.

Van Overschee, Peter and De Moor, BL. *Subspace identification for linear systems: TheoryImplementationApplications*. Springer Science & Business Media, 2012.

Venkatraman, Arun, Hebert, Martial, and Bagnell, J Andrew. Improving multi-step prediction of learned time series models. *AAAI*, 2015.

Wei, William Wu-Shyong. *Time series analysis*. Addison-Wesley publication, 1994.

Zeiler, Matthew D. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

A. Proof of Theorem. 4.1

Proof. We prove the theorem by induction. We start from $t = 1$. Under the assumption of infinite many training trajectories, $\hat{m}_1$ is exactly equal to m_1, which is $\mathbb{E}_\tau(\phi(f_1))$ (no observations yet, conditioning on nothing).

Now let us assume at time step t, we have all computed $\hat{m}_j^\tau$ equals to m_j^τ for $1 \leq j \leq t$ on any trajectory τ. Under the assumption of infinite training trajectories, minimizing the empirical risk over D_t is equivalent to minimizing the true risk $\mathbb{E}_\tau[d(F(m_t^\tau, x_t^\tau), f_{t+1}^\tau)]$. Since we use sufficient features for distribution $P(f_t|h_{t-1})$ and we assume the system is k-observable, there exists a underlying deterministic map, which we denote as F_t^* here, that maps m_t^τ and x_t^τ to m_{t+1}^τ (Eq. 4 represents F_t^*). Without loss of generality, for any τ, conditioned on the history h_t^τ, we have that for a noisy observation f_t^τ:

$$\begin{aligned}\phi(f_{t+1}^\tau)|h_t^\tau &= \mathbb{E}[\phi(f_{t+1}^\tau)|h_t^\tau] + \epsilon && (13)\\ &= m_{t+1}^\tau + \epsilon && (14)\\ &= F_t^*(m_t^\tau, x_t^\tau) + \epsilon, && (15)\end{aligned}$$

where $\mathbb{E}[\epsilon] = 0$. Hence we have that F_t^* is the operator of conditional expectation $\mathbb{E}[\big(\phi(f_{t+1})|h_t\big)|m_t, x_t]$, which exactly computes the predictive state $m_{t+1} = \mathbb{E}[\phi(f_{t+1}^\tau)|h_t^\tau]$, given m_t^τ and x_t^τ on any trajectory τ.

Since the loss d is a squared loss (or any other loss that can be represented by Bregman divergence), the minimizer of the true risk will be the operator of conditional expectation $\mathbb{E}[\big(\phi(f_{t+1})|h_t\big)|m_t, x_t]$. Since it is equal to F^* and we have $F^* \in \mathcal{F}$ due to the realizable assumption, the risk minimization at step t exactly finds F_t^*. Using $\hat{m}_t^\tau$ (equals to m_t^τ based on the induction assumption for step t), and x_t^τ, the risk minimizer F^* then computes the exact m_{t+1}^τ for time step $t+1$. Hence by the induction hypothesis, we prove the theorem. $\square$

B. Proof of Theorem. 4.2

Under the assumption of infinitely many training trajectories, we can represent the objective as follows:

$$\mathbb{E}_{\tau\sim\mathcal{D}}\frac{1}{T}\sum_{t=1}^{T} d(F_t(\hat{m}_t^\tau, x_t^\tau), f_{t+1}^\tau) = \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{(z,f)\sim\omega_t}\big[d(F_t(z), f)\big] \tag{16}$$

Note that each F_t is trained by minimizing the risk:

$$F_t = \arg\min_{F\sim\mathcal{F}} \mathbb{E}_{(z,f)\sim\omega_t}\big[d(F(z), f)\big]. \tag{17}$$

Since we define $\epsilon_t = \min_{F\sim\mathcal{F}}\mathbb{E}_{(z,f)\sim\omega_t}\big[d(F(z), f)\big]$, we have:

$$\mathbb{E}_{\tau\sim\mathcal{D}}\frac{1}{T}\sum_{t=1}^{T} d(F_t(\hat{m}_t^\tau, x_t^\tau), f_{t+1}^\tau) = \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{(z,f)\sim\omega_t}\big[d(F_t(z), f)\big] \leq \frac{1}{T}\sum_{t}\epsilon_t. \tag{18}$$

Defining $\epsilon_{max} = \max_t\{\epsilon_t\}$, we prove the theorem.

C. Proof of Theorem. 4.3

Proof. Without loss of generality, let us assume the loss $d(F(z), f) \in [0, 1]$. To derive generalization bound using Rademacher complexity, we assume that $\|F(z)\|_2$ and $\|f\|_2$ are bounded for any $z, f, F \in \mathcal{F}$, which makes sure that $d(F(z), f)$ will be Lipschitz continuous with respect to the first term $F(z)$[6].

Given M samples, we further assume that we split M samples into T disjoint sets $S_1, ..., S_T$, one for each training process of F_i, for $1 \leq i \leq T$. The above assumption promises that the data S_t for training each filter F_t is i.i.d. Note that each S_i now contains M/T i.i.d trajectories.

Since we assume that at time step t, we use S_t (rolling out $F_1, ..., F_{t-1}$ on trajectories in S_t) for training F_t, we can essentially treat each training step independently: when learning F_t, the training data z, f are sampled from ω_t and are i.i.d.

Now let us consider time step t. With the learned $F_1, ..., F_{t-1}$, we roll out them on the trajectories in S_t to get $\frac{M}{T}$ i.i.d samples of $(z, f) \sim \omega_t$. Hence, training F_t on these $\frac{M}{T}$ i.i.d samples becomes classic empirical risk minimization problem. Let us define loss class as $\mathcal{L} = \{l_F : (z, f) \to d(F(z), f) : F \in \mathcal{F}\}$, which is determined by $\mathcal{F}$ and d. Without loss of generality, we assume $l(z, f) \in [0, 1], \forall l \in \mathcal{L}$. Using the uniform bound from Rademacher theorem (Mohri et al., 2012), we have for any $F \in \mathcal{F}$, with

[6]Note that in fact for the squared loss, d is 1-smooth with respect to its first item. In fact we can remove the boundness assumption here by utilizing the existing Rademacher complexity analysis for smooth loss functions (Srebro et al., 2010).

probability at least $1-\delta'$:

$$\mathbb{E}_{z,f\sim\omega_t}[d(F(z),f)] - \frac{T}{M}\sum_i d(F(z^i),f^i)| \tag{19}$$

$$\leq 2\mathcal{R}_t(\mathcal{L}) + \sqrt{\frac{T\ln(1/\delta')}{2M}}, \tag{20}$$

where $\mathcal{R}_t(\mathcal{L})$ is Rademacher complexity of the loss class $\mathcal{L}$ with respect to distribution ω_t. Since we have F_t is the empirical risk minimizer, for any $F_t^*\in\mathcal{F}$, we have with probability at least $1-\delta'$:

$$\mathbb{E}_{z,f\sim\omega_t}[d(F_t(z),f)] \leq \mathbb{E}_{z,f\sim\omega_t}[d(F_t^*(z^i),f^i)] + 4\mathcal{R}_t(\mathcal{L}) + 2\sqrt{\frac{T\ln(1/\delta')}{2M}}. \tag{21}$$

Now let us combine all time steps together. For any $F_t^*\in\mathcal{F}$, $\forall t$, with probability at least $(1-\delta')^T$, we have:

$$\begin{aligned}
&\mathbb{E}_{\tau\sim\mathcal{D}_\tau}\Big[\frac{1}{T}\sum_{t=1}^{T} d(F_t(\hat{m}_t^\tau,x_t^\tau),f_{t+1}^\tau)\Big] = \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{z,f\sim d_t}\big[d(F_t(z),f)\big]\\
&\leq \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{z,f\sim\omega_t}[d(F_t^*(z),f)] + 4\bar{\mathcal{R}}(\mathcal{L}) + 2\sqrt{\frac{T\ln(1/\delta')}{2M}}\\
&= \mathbb{E}_{\tau\sim\mathcal{D}_\tau}\big[\frac{1}{T}\sum_{t=1}^{T} d(F_t^*(\hat{m}_t^\tau,x_t^\tau),f_{t+1}^\tau)\big] + 4\bar{\mathcal{R}}(\mathcal{L}) + 2\sqrt{\frac{T\ln(1/\delta')}{2M}},
\end{aligned} \tag{22}$$

where $\bar{\mathcal{R}}(\mathcal{L}) = (1/T)\sum_{t=1}^{T}\mathcal{R}_t(\mathcal{L})$ is the average Rademacher complexity. Inequality. 22 is derived from the fact the event that the above inequality holds can be implied by the event that Inequality. 21 holds for every time step t ($1\leq t\leq T$) independently. The probability of Inequality. 21 holds for all t is at least $(1-\delta')^T$.

Note that in our setting $d(F(z),f) = \|F(z)-f\|_2^2$, and under our assumptions that $\|F(z)\|_2$ and $\|f\|_2$ are bounded for any $z,f,F\in\mathcal{F}$, $d(F(z),f)$ is Lipschitz continuous with respect to its first item with Lipschitz constant equal to ν, which is $\sup_{F,z,f} 2\|F(z)-f\|_2$. Hence, from the composition property of Rademacher number (Mohri et al., 2012), we have:

$$\mathcal{R}_t(\mathcal{L}) \leq \nu\mathcal{R}_t(\mathcal{F}), \quad \forall t. \tag{23}$$

It is easy to verify that for $T\geq 1$, $\delta'\in(0,1)$, we have $(1-\delta')^T \geq 1-T\delta'$. Let $1-T\delta' = 1-\delta$, and solve for δ', we get $\delta' = \delta/T$. Substitute Eq. 23 and $\delta'=\delta/T$ into Eq. 22, we prove the theorem. $\square$

Note that the above theorem shows that for fixed number training examples, the generalization error increase as $\tilde{O}(\sqrt{T})$ (sublinear with respect to T).

D. Case Study: Stationary Kalman Filter

To better illustrate PSIM, we consider a special dynamical system in this section. More specifically, we focus on the stationary Kalman filter (Boots, 2012; Hefny et al., 2015) [7]:

$$\begin{aligned}
s_{t+1} &= As_t + \epsilon_s, \;\; \epsilon_s\sim\mathcal{N}(0,Q),\\
x_t &= Cs_t + \epsilon_x, \;\; \epsilon_x\sim\mathcal{N}(0,R).
\end{aligned} \tag{24}$$

As we will show, the Stationary Kalman Filter allows us to explicitly represent the predictive states (sufficient statistics of the distributions of future observations are simple). We will also show that we can explicitly construct a bijective map between the predictive state space and the latent state space, which further enables us to explicitly construct the predictive state filter. We will show that the predictive state filter is closely related to the original filter in the latent state space.

The k-observable assumption here essentially means that the observability matrix: $\mathcal{O} = \begin{bmatrix} C & CA & CA^2 & \ldots & CA^{k-1}\end{bmatrix}^\top$ is full (column) rank. Now let us define $P(s_t|h_{t-1}) = \mathcal{N}(\hat{s}_t,\Sigma_s)$, and $P(f_t|h_{t-1}) = \mathcal{N}(\hat{f}_t,\Sigma_f)$. Note that Σ_s is a constant for a stationary Kalman filter (the Kalman gain is converged). Since Σ_f is purely determined by Σ_s, A, C, R, Q, it is also a constant. It is clear now

[7]For a well behaved system, the filter will become stationary (Kalman gain converges) after running for some period of time. Our definition here is slightly different from the classic Kalman filter: we focus on filtering from $P(s_t|h_{t-1})$ (without conditioning on the observation x_t generated from s_t) to $P(s_{t+1}|h_t)$, while traditional Kalman filter usually filters from $P(s_t|h_t)$ to $P(s_{t+1}|h_{t+1})$.

that $\hat{f}_t = \mathcal{O}\hat{s}_t$. When the Kalman filter becomes stationary, it is enough to keep tracking $\hat{s}_t$. Note that here, given $\hat{s}_t$, we can compute $\hat{f}_t$; and given $\hat{f}_t$, we can reveal $\hat{s}_t$ as $\mathcal{O}^{\dagger}\hat{f}_t$, where $\mathcal{O}^{\dagger}$ is the pseudo-inverse of $\mathcal{O}$. This map is bijective since $\mathcal{O}$ is full column rank due to the k-observability.

Now let us take a look at the update of the stationary Kalman filter:

$$\hat{s}_{t+1} = A\hat{s}_t - A\Sigma_s C^T(C\Sigma_s C^T + R)^{-1}(C\hat{s}_t - x_t) = A\hat{s}_t - L(C\hat{s}_t - x_t), \tag{25}$$

where we define $L = A\Sigma_s C^T(C\Sigma_s C^T + R)^{-1}$. Here due to the stationary assumption, Σ_s keeps constant across time steps. Multiple $\mathcal{O}$ on both sides and plug in $\mathcal{O}^{\dagger}\mathcal{O}$, which is an identity, at proper positions, we have:

$$\begin{aligned}\hat{f}_{t+1} &= \mathcal{O}\hat{s}_{t+1} = \mathcal{O}A(\mathcal{O}^{\dagger}\mathcal{O})\hat{s}_t - \mathcal{O}L(C\mathcal{O}^{\dagger}\mathcal{O}\hat{s}_t - x_t)\\ &= \mathcal{O}A\mathcal{O}^{\dagger}\hat{f}_t - \mathcal{O}L(C\mathcal{O}^{\dagger}\hat{f}_t - x_t) = \tilde{A}\hat{f}_t - \tilde{L}(\tilde{C}\hat{f}_t - x_t) \qquad (26)\\ &= \begin{bmatrix}\tilde{A} - \tilde{L}\tilde{C} & \tilde{L}\end{bmatrix}\begin{bmatrix}\hat{f}_t\\ x_t\end{bmatrix}, \qquad (27)\end{aligned}$$

where we define $\tilde{A} = \mathcal{O}A\mathcal{O}^{\dagger}$, $\tilde{C} = C\mathcal{O}^{\dagger}$ and $\tilde{L} = \mathcal{O}L$. The above equation represents the *stationary* filter update step in predictive state space. Note that the *deterministic* map from $(\hat{f}_t, \Sigma_f)$ and x_t to $(\hat{f}_{t+1}, \Sigma_f)$ is a linear map (F defined in Sec. 4 is a linear function with respect to $\hat{f}_t$ and x_t). The filter update in predictive state space is very similar to the filter update in the original latent state space except that predictive state filter uses operators $(\tilde{A}, \tilde{C}, \tilde{Q})$ that are linear transformations of the original operators (A, C, Q).

We can do similar linear algebra operations (e.g., multiply $\mathcal{O}$ and plug in $\mathcal{O}^{\dagger}\mathcal{O}$ in proper positions) to recover the stationary filter in the original latent state space from the stationary predictive state filter. The above analysis leads to the following proposition:

Proposition D.1. *For a linear dynamical system with k-observability, there exists a filter in predictive state space (Eq. 27) that is equivalent to the stationary Kalman filter in the original latent state space (Eq. 25).*

We just showed a concrete bijective map between the filter with predictive states and the filter with the original latent states by utilizing the observability matrix $\mathcal{O}$. Though we cannot explicitly construct the bijective map unless we know the parameters of the LDS (A,B,C,Q,R), we can see that learning the linear filter shown in Eq. 27 is equivalent to learning the original linear filter in Eq. 25 in a sense that the predictive beliefs filtered from Eq. 27 encodes as much information as the beliefs filtered from Eq. 25 due to the existence of a bijective map between predictive states and the beliefs for latent states.

D.1. Collection of Synthetic Data

We created a linear dynamical system with $A \in \mathbb{R}^{3\times3}$, $C \in \mathbb{R}^{2\times3}$, $Q \in \mathbb{R}^{3\times3}$, $R \in \mathbb{R}^{2\times2}$. The matrix A is full rank and its largest eigenvalue is less than 1. The LDS is 2-observable. We computed the constance covariance matrix Σ_s, which is a fixed point of the covariance update step in the Kalman filter. The initial distribution of s_0 is set to $\mathcal{N}(1, \Sigma_s)$. We then randomly sampled 50000 observation trajectories from the LDS. We use half of the trajectories for training and the left half for testing.

E. Additional Experiments

With linear regression as the underlying filter model: $\hat{m}_{t+1} = W[\hat{m}_t^T, x_t^T]^T$, where W is a 2-d matrix, we compare PSIM with back-propagation using the solutions from DAgger as initialization to PSIM with DAgger, and PSIM with back-propagation with random initialization. We implemented PSIM with Back-propagation in Theano (Bastien et al., 2012). For random initialization, we uniformly sample non-zero small matrices to avoid gradient blowing up. For training, we use mini-batch gradient descent where each trajectory is treated as a batch. We tested several different gradient descent approaches: regular gradient descent with step decay, AdaGrad (Duchi et al., 2011), AdaDelta (Zeiler, 2012), RMSProp (Tieleman & Hinton, 2012). We report the best performance from the above approaches. When using the solutions from PSIM with DAgger as an initialization for back-propagation, we use the same setup. We empirically find that RMSProp works best across all our datasets for the inference machine framework, while regular gradient descent generally performs the worst.

	PSIM-Linear (DAgger)	**PSIM-Linear** (Bp)	**PSIM-Linear** (DAgger + Bp)
Robot Drill Assembly	2.15	2.54	**2.09**
Motion Capture	5.75	9.94	**5.66**
Beach Video Texture	164.23	268.73	**164.08**

Table 2. Comparison between PSIM with DAgger, PSIM with back-propagation using random initialization, and PSIM with back-propagation using DAgger as initialization with ridge linear regression.

Tab. 2 shows the results of using different training methods with ridge linear regression as the underlying model.

Additionally, we test back-propagation for PSIM with Kernel Ridge regression as the underlying model: $\hat{m}_{t+1} = W\eta(\hat{m}_t, x_t)$, where η is a pre-defined, deterministic feature function that maps $(\hat{m}_t, x_t)$ to a reproducing kernel Hilbert space approximated with Random

Fourier Features (RFF). Essentially, we lift the inputs $(\hat{m}_t, x_t)$ into a much richer feature space (a scaled, and transition invariant feature space) before feeding it to the next module. The results are shown in Table. 3. As we can see, with RFF, back-propagation achieves better performance than back-propagation with simple linear regression (PSIM-Linear (Bp)). This is expected since using RFF potentially captures the non-linearity in the underlying dynamical systems. On the other hand, PSIM with DAgger achieves better results than back-propagation across all the datasets. This result is consistent with the one from PSIM with ridge linear regression.

	PSIM-RFF (Bp)	**PSIM-RFF** (DAgger)	**RNN**
Robot Drill Assembly	2.54	**1.80**	1.99
Motion Capture	9.26	**5.41**	9.6
Beach Video Texture	202.10	**130.53**	346.0

Table 3. Comparison between PSIM with DAgger, PSIM with back-propagation using random initialization with kernel ridge linear regression, and Recurrent Neural Network. For RNN, we use 100 hidden states for Robot Drill Assembly, 200 hidden states for motion capture, and 2500 hidden states for Beach Video Texture.

Overall, several interesting observations are: (1) back-propagation with random initialization achieves reasonable performance (e.g., good performance on flag video compared to baselines), but worse than the performance of PSIM with DAgger. PSIM back-propagation is likely stuck at locally optimal solutions in some of our datasets; (2) PSIM with DAgger and Back-propagation can be symbiotically beneficial: using back-propagation to refine the solutions from PSIM with DAgger improves the performance. Though the improvement seems not significant over the 400 epochs we ran, we do observe that running more epochs continues to improve the results; (3) this actually shows that PSIM with DAgger itself finds good filters already, which is not surprising because of the strong theoretical guarantees that it has.