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. □

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. □

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: $\mathcal{O} = \begin{bmatrix} C & CA & CA^2 & ... & CA^{k-1}\end{bmatrix}^\top$ 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 $A \in \mathbb{R}^{3\times 3}$, $C \in \mathbb{R}^{2\times 3}$, $Q \in \mathbb{R}^{3\times 3}$, $R \in \mathbb{R}^{2\times 2}$. 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 $\mathcal{N}(\mathtt{1}, \Sigma_s)$. 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.