Learning to Smooth with Bidirectional Predictive State Inference Machines

Wen Sun[1], Roberto Capobianco[2], Geoffrey J. Gordon[1], J. Andrew Bagnell[1], and Byron Boots[3]
[1] School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213
[2] Department of Computer, Control and Management Engineering, Sapienza University of Rome, Italy
[3] School of Interactive Computing, Georgia Institute of Technology, Atlanta, GA 30332
[1]{wensun, ggordon, dbagnell}@cs.cmu.edu, [2]capobianco@dis.uniroma1.it, [3]bboots@cc.gatech.edu

Abstract

We present the Smoothing Machine (SMACH, pronounced "smash"), a dynamical system learning algorithm based on chain Conditional Random Fields (CRFs) with latent states. Unlike previous methods, SMACH is designed to optimize prediction performance when we have information from both past and future observations. By leveraging Predictive State Representations (PSRs), we model beliefs about latent states through *predictive states*—an alternative but equivalent representation that depends directly on observable quantities. Predictive states enable the use of well-developed supervised learning approaches in place of local-optimum-prone methods like EM: we learn regressors or classifiers that can approximate message passing and marginalization in the space of predictive states. We provide theoretical guarantees on smoothing performance and we empirically verify the efficacy of SMACH on several dynamical system benchmarks.

1 INTRODUCTION

In time series, smoothing is the process of inferring *a posteriori* latent state information given a model as well as past and future observations. Many applications leverage smoothing for inference, ranging from machine learning and robotics to biological science. For example, in a Linear Dynamical System (LDS), a Kalman smoother is frequently used to compute the posterior distribution of the system's states. Similarly, in Optical Character Recognition (OCR), smoothing is used to predict the word corresponding to a sequence of of handwritten characters.

Chain CRFs [Lafferty et al., 2001] are remarkably successful probabilistic graphical models for these applications. As a discriminative approach, CRFs can capture detailed structural properties of the observations and states with a reasonable number of parameters. (In this context, the states are often called *labels*.) Given a parametrization of the CRF, along with a training data set consisting of pairs of ground truth labels and observations, we can learn parameters for the CRF by maximizing the (log) conditional likelihood of the training labels given the training observations. For appropriate feature parametrizations, the log-likelihood objective is convex and one can achieve globally optimal solutions.

In this work, we are interested in *latent* chain CRFs, which generalize CRFs by adding a layer of latent variables between the labels and observations (Figure 1a). These latent variables increase the expressiveness of the underlying probabilistic model [Stratos et al., 2013], but also make the optimization problem more challenging: the log-likelihood objective becomes non-convex. Iterative approaches such as Expectation-Maximization (EM) can only compute locally optimal solutions, making the search for a globally optimal solution computationally infeasible. Though the global maximizer of the likelihood can promise good performance if we can find it, the locally optimal solutions that are found in practice typically do not have any performance guarantees.

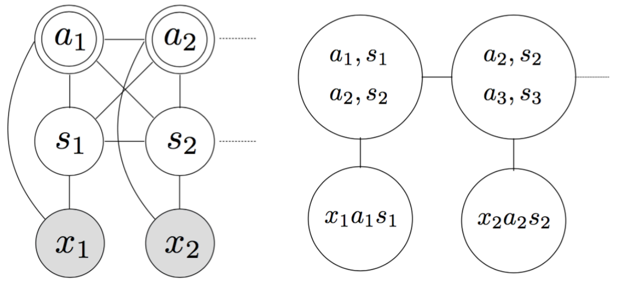
To tackle the problem of local optima, we borrow ideas from spectral learning methods and Predictive State Representations (PSRs). In the last decade, these methods have been successfully used for learning and inference in latent state space models such as linear dynamical systems and hidden Markov models [Jaeger, 2000, Hsu et al., 2009, Boots, 2012, Boots et al., 2011, Song et al., 2010, Boots et al., 2013, Hefny et al., 2015]. The main idea is that, instead of tracking latent states directly, we track observable quantities such as expectations of features of future observations. If the features are selected to be sufficient statistics of the latent state distribution, knowing the predictive state is equivalent to knowing the underlying state of the system [Jaeger, 2000, Hefny et al., 2015, Sun et al., 2016]. Spectral learning algorithms provide theoretical guarantees: their estimating equations typically have a unique global solution, which converges to the true model parameters under the assumption of realizability (no model mismatch). However, if there is model mismatch, no theoretical guarantees are available for the learned model or the associated inference tasks [Kulesza et al., 2014].

To gain the benefits of spectral learning methods even in the case of model mismatch, we propose a novel algorithm, the *Smoothing Machine* (SMACH). Our method builds on recent work on *Predictive State Inference Machines* (PSIMs) [Sun et al., 2016, Venkatraman et al., 2016]. Like a PSR, a PSIM uses predictive states to represent beliefs about latent states. But, unlike a PSR, a PSIM directly learns a closed-loop filter as an inference machine that propagates the predictive state forward in time. By focusing on inference in predictive state space rather than latent state space, a PSIM reduces the difficult problem of learning a latent state space model to supervised learning and achieves guaranteed performance for its inference task (filtering or forward belief propagation) even in the presence of model mismatch.

A PSIM learns a filter that uses past and current observations to predict current latent information. Recently, Venkatraman et al. [2016] applied PSIM to forward message passing in a graphical model with partially observable states. However, this approach is suboptimal for estimating latent states in an offline setting where future observations are also available. So, SMACH extends PSIMs by learning a smoother that takes account of both past *and* future observations. Similar to PSIMs and PSRs, SMACH replaces latent states by predictions of observable quantities. Different from classic messages in graphical models, predictive messages represent the distributions of the sufficient statistics of observable quantities (e.g., labels and observations).

SMACH treats message passing as a sequence of predictions. It directly learns three predictors for approximating message passing in the predictive state space: one for forward predictive state message passing, another for backward predictive state message passing, and a third for combining messages at a given time step. The first predictor learns to recursively compute forward messages that encode information about past events (all past observations up to now), while the second predictor learns to compute backward messages that encode information about future events (all future observations after now). The last predictor predicts the current label given the local observation and the corresponding forward and backward predictive states. At testing time, SMACH uses the first two predictors to compute the forward and backward predictive states along the chain, and then uses the last predictor to combine the forward and backward predictive states with the local observations to predict the labels.

The main advantages of SMACH are: (1) It leverages predictive states to reduce the problem of learning latent chain CRFs to a supervised setting. This reduction enables us



(a) Latent chain CRF (b) Junction Tree

Figure 1: A form of latent chain CRF and its junction tree. The labels a (double circled) are observable in training but are latent and need to be inferred during testing. Latent state s is never observable and x (gray) is always observable. During testing, given all observations $x_{1:T}$, the inference task is to compute the posterior distribution $P(a_t \mid x_{1:T})$ for all t.

to avoid local optima and use well developed theorems of supervised learning to quantify the smoothing performance of SMACH. (2) Similar to PSIM, SMACH combines the two phases of modeling and learning into a siingle task, to directly optimize the ultimate inference task of smoothing. Hence, no parametrization is needed for an explicit probabilistic graphical model (e.g., potential functions for cliques). (3) The learned predictors implicitly encode the information of the underlying probabilistic models. This enables the use of arbitrarily powerful regressors or classifiers as predictors for message passing and for the computation of marginal distributions and hence provides resistance to model mismatch. Finally, (4) as we will show, our formulation fully generalizes the special case of a *chain CRF* without latent states (i.e., removing s from Figure 1a).

2 PRELIMINARIES

In this work we consider a general latent chain CRF structure, as shown in Figure 1a. The model consists of three different types of variables: (1) Labels a_t that can be either continuous (e.g, positions of a mobile robot) or discrete (e.g., labels of a hand-written character). We assume that the ground truth labels are available in the training data, while the labels are latent at test time and need to be predicted. (2) Latent states s_t (either continuous or discrete) that are hidden both at training and test time. (3) Observations x_t (either continuous or discrete), available both at training and test time. A sequence of labels and observations $\{a_1, x_1, ..., a_T, x_T\}$ defines a trajectory τ. In order to perform the *smoothing* inference process, we are interested in computing the posterior distribution of the label a_t, conditioned on all observations $\{x_1, ..., x_T\}$: $P(a_t \mid x_{1:T}), \forall t$.

To discuss message passing in the latent chain CRF of Figure 1a, we first need to generate the corresponding junction tree representation, as shown in Figure 1b. The junction tree allows inference algorithms to avoid the inner loops in the original latent CRF model and perform message passing as follows [Koller and Friedman, 2009]. First, pick the node (a_1, s_1, a_2, s_2) as the root and perform backward message passing starting from each leaf (x_t, a_t, s_t). The message at the separation set (a_t, s_t) between the node $(a_t, s_t, a_{t+1}, s_{t+1})$ and the node (x_t, a_t, s_t) can be represented by $P(x_t \mid a_t, s_t)$, where x_t is the evidence. We define the backward message at the separation set (a_t, s_t), going from the node $(a_t, s_t, a_{t+1}, s_{t+1})$ to the node $(a_{t-1}, s_{t-1}, a_t, s_t)$, as $b_t \equiv P(a_t, s_t \mid x_{t:T})$, which we compute recursively as:

$$b_{t-1} \propto \sum_{a_t s_t} P(a_{t-1}, s_{t-1} \mid a_t, s_t) P(x_t \mid a_t, s_t) b_t. \quad (1)$$

We assume without loss of generality that the (forward) transition probability $P(a_{t+1}, s_{t+1} \mid a_t, s_t)$ is time-invariant, and we write $P(a_t, s_t \mid a_{t+1}, s_{t+1})$ for the corresponding backward transition probability. After all backward messages are computed, starting from the root, we pass messages forward. The forward message at the separation set (a_t, s_t) from the node $(a_{t-1}, s_{t-1}, a_t, s_t)$ to the node $(a_t, s_t, a_{t+1}, s_{t+1})$ is represented by $P(a_t, s_t \mid x_{1:t-1})$, which we denote $\tilde{b}_t$. The forward message is also computed recursively:

$$\tilde{b}_{t+1} \propto \sum_{s_t, a_t} P(a_{t+1}, s_{t+1} \mid a_t, s_t) P(x_t \mid a_t, s_t) \tilde{b}_t. \quad (2)$$

With forward and backward messages, at the node $(a_t, s_t, a_{t+1}, s_{t+1})$ the marginal message $P(a_t \mid x_{1:T})$, which we denote as $\hat{b}_t$, is computed as:

$$\hat{b}_t \propto \sum_{s_t, a_{t+1}, s_{t+1}} \frac{P(a_{t+1}, s_{t+1} \mid a_t, s_t) \tilde{b}_t P(x_t \mid a_t, s_t) b_{t+1}}{P(a_{t+1}, s_{t+1})}. \quad (3)$$

If we assume that $P(a, s)$ is time-invariant, the above equation can be regarded as a time-invariant, deterministic function that maps $\tilde{b}_t$, b_{t+1}, and the local observation x_t to $\hat{b}_t$.

3 OBSERVABILITY AND PREDICTIVE STATES

To perform message passing as described in Sec. 2, classic MLE-based approaches first parametrize latent CRFs and then learn the parametrization from training data. Learning the parameters of the transition models and the observations models is hard due to the latent states: the log likelihood of the observations and labels is non-convex. As a consequence, MLE-based approaches need to use iterative methods such EM, which can only promise local optimality and therefore lack performance guarantees.

Our approach leverages Predictive State Representations (PSRs) to overcome the difficulties of dealing with latent states. PSRs use *predictive states*, which consist of expectations of observable quantities, as an alternative, equivalent representation of latent belief states. Below, we first introduce the definition of *observability*, which extends the classic observability definition from latent state space models (e.g., LDS, HMM) to latent chain-CRFs.

3.1 OBSERVABILITY

Let us focus on a particular separation set (a_t, s_t) between the two nodes $(a_{t-1}, s_{t-1}, a_t, s_t)$ and $(a_t, s_t, a_{t+1}, s_{t+1})$ on the junction tree modelled in Figure 2. The forward belief in this separation set is represented as $P(a_t, s_t \mid x_{1:t-1})$. We define forward k-observability as follows:

Definition (Forward k-observability) There exists a constant $k \in \mathbb{N}^+$ such that the relationship between $P(a_t, s_t \mid x_{1:t-1})$ and $P(a_{t:t+k}, x_{t:t+k-1} \mid x_{1:t-1})$ is bijective.

Intuitively, forward k-observability means that the distribution of *future* observable quantities (labels a and observations x) uniquely determines the latent forward belief, conditioned on the previous observations. We define backward k-observability similarly:

Definition (Backward k-observability) There exists a constant $k \in \mathbb{N}^+$ such that the relationship between $P(a_t, s_t \mid x_{t:T})$ and $P(a_{t-k:t}, x_{t-k:t-1} \mid x_{t:T})$ is bijective.

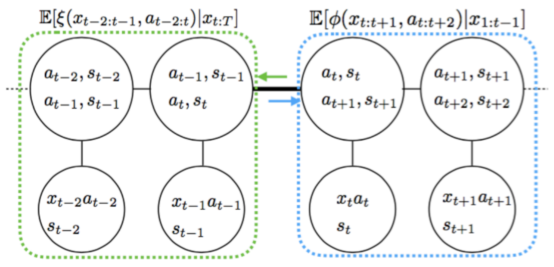
Intuitively, backward k-observability means that the distribution of *past* observable quantities (labels a and observations x) uniquely determines the latent backward belief state, conditioned on future observations.[1]

In the Appendix, we conduct a case study of observability on one special form of latent chain-CRF—the Refinement Hidden Markov Model (RHMM) [Stratos et al., 2013]. Specifically, first we reduce the RHMM to a regular HMM, and then we leverage the well-defined concept of observability of HMMs to define the forward and backward observability of the original RHMM. In fact, when we set k big enough to cover the entire time windows of past and future, our definition is actually agrees with the *past* and *future* random variables defined by Stratos et al. [2013].

3.2 PREDICTIVE STATES

The definition of observability provides us with a different way to represent the beliefs containing latent states: we can use the joint distributions of future (past) labels

[1] In principle, the statistics could depend on the entire sequence of observations $x_{t:T}$ (or $x_{1:t}$). The restriction to a k-step window of visible variables simplifies the notation and is commonly used in practice.



$\mathbb{E}[\xi(x_{t-2:t-1}, a_{t-2:t})|x_{t:T}]$
$\mathbb{E}[\phi(x_{t:t+1}, a_{t:t+2})|x_{1:t-1}]$
a_{t-2}, s_{t-2} a_{t-1}, s_{t-1}
a_{t-1}, s_{t-1} a_t, s_t
a_t, s_t a_{t+1}, s_{t+1}
a_{t+1}, s_{t+1} a_{t+2}, s_{t+2}
$x_{t-2} a_{t-2}$ s_{t-2}
$x_{t-1} a_{t-1}$ s_{t-1}
$x_t a_t$ s_t
$x_{t+1} a_{t+1}$ s_{t+1}

Figure 2: Illustration of the forward predictive state (blue) and backward predictive state (green) with $k = 2$.

and observations to represent the forward (backward) beliefs about latent states. Let us define a feature function ϕ that computes sufficient statistics of $a_{t:t+k}, x_{t:t+k-1}$, such that $P(a_{t:t+k}, x_{t:t+k-1} \mid x_{1:t-1})$ can be represented by $\mathbb{E}[\phi(a_{t:t+k}, x_{t:t+k-1}) \mid x_{1:t-1}]$. (In many cases, a good feature function is a kernel mean embedding, which is guaranteed to yield sufficient statistics for a wide range of distributions.) Similarly we define a feature function ξ that computes sufficient statistics of $a_{t-k:t}, x_{t-k:t-1}$, such that $P(a_{t-k:t}, x_{t-k:t-1} \mid x_{t:T})$ can be represented by $\mathbb{E}[\xi(a_{t-k:t}, x_{t-k:t-1}) \mid x_{t:T}]$. (In practice, ξ could be the same as ϕ.) Under the assumption that the system is forward k-observable and backward k-observable, we can now replace the beliefs about latent states $P(a_t, s_t \mid x_{1:t-1})$ and $P(a_t, s_t \mid x_{t:T})$ with the predictive messages $\mathbb{E}[\phi(a_{t:t+k}, x_{t:t+k-1}) \mid x_{1:t-1}]$ and $\mathbb{E}[\xi(a_{t-k:t}, x_{t-k:t-1}) \mid x_{t:T}]$ respectively. For notational simplicity, let us define $f_t = (a_{t:t+k}, x_{t:t+k-1})$ and $h_t = (a_{t-k:t}, x_{t-k:t-1})$. We define the *Forward Predictive State* (FPS) m_t at step t, and the *Backward Predictive State* (BPS) v_t at step t as:

$$m_t = \mathbb{E}[\phi(f_t) \mid x_{1:t-1}], \quad v_t = \mathbb{E}[\xi(h_t) \mid x_{t:T}]. \tag{4}$$

Note that our formulation fully generalizes the special case of a *chain CRF* without latent states (i.e., removing all s and edges connected to s from Figure 1a). In this case, by setting $k = 0$, the forward predictive state m_t becomes $\mathbb{E}[\phi(a_t) \mid x_{1:t-1}]$ and the backward predictive state v_t becomes $\mathbb{E}[\xi(a_t) \mid x_{t:T}]$. With sufficient feature functions ϕ and ξ, m_t and v_t are then equivalent to $P(a_t \mid x_{1:t-1})$ and $P(a_t \mid x_{t:T})$, which are the classic forward and backward beliefs on a chain CRF.

As we will show in the next section, we can use PSIM to compute forward and backward predictive states.

4 ALGORITHM

We now present the *Smoothing Machines* (SMACH) algorithm (Alg. 2). As introduced in the previous section, we first leverage PSIM to learn stationary filters for the generation of forward and backward predictive states. The final step of SMACH is to combine the forward and backward predictive states together with observations to learn a predictor for smoothing. Below we briefly introduce PSIM and how PSIM can be used for computing predictive states.

Algorithm 1 PSIM with DAgger (Forward Pass)

```
1: Input: $M$ independent trajectories $\tau_i$, $1 \leq i \leq M$;
2: Initialize $D_0 \leftarrow \emptyset$ and initialize $F_0 \in \mathcal{F}_1$;
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. 6) on trajectory $\tau_i$, $1 \leq i \leq M$
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 $F_{n+1} \in \mathcal{F}_1$ on $D_{n+1}$ to minimize the loss $d(F(m,x), f)$;
9: end for
10: Return: the best hypothesis $F_f \in \{F_n\}_n$ on validation trajectories.
```

4.1 PREDICTIVE STATE INFERENCE MACHINES

We utilize *Predictive State Inference Machines* (PSIM) [Sun et al., 2016] to directly compute the predictive states m_t and v_t. PSIM is a discriminative learning approach that learns a filter (black box represented by any regressors or classifiers) to mimic the predictive message passing process: by taking the incoming predictive message and the local observation as inputs, it outputs the next predictive message. Specifically, for forward predictive message passing, given a trajectory $\tau = \{a_1, x_1, ..., a_T, x_T\}$ sampled from the distribution $\mathcal{D}$, PSIM aims at finding a hypothesis $F_f \in \mathcal{F}_1$ to optimize the following objective:

$$\min_{F_f \in \mathcal{F}_1} \mathbb{E}_\tau \sum_{t=1}^{T} \|\hat{m}_t^\tau - \phi(f_t^\tau)\|^2; \tag{5}$$

$$\text{s.t. } \hat{m}_{t+1}^\tau = F_f(\hat{m}_t^\tau, x_t^\tau). \tag{6}$$

Namely, PSIM finds a hypothesis that can mimic forward predictive message passing, and the quality of the computed predictive messages are measured by the loss $\|\hat{m}_t - \phi(f)_t\|^2$ (i.e., moment matching).

Similar to forward predictive message passing, we can use PSIM for backward predictive message passing. In this case, PSIM aims at finding a hypothesis $F_b \in \mathcal{F}_2$ ($\mathcal{F}_2$ and $\mathcal{F}_1$ could either be the same or different hypothesis classes) to optimize the *backward* filtering performance as:

$$\min_{F_b \in \mathcal{F}_2} \mathbb{E}_\tau \sum_{t=1}^{T} \|\hat{v}_t^\tau - \xi(h_t^\tau)\|^2; \tag{7}$$

$$\text{s.t. } \hat{v}_t^\tau = F_b(\hat{v}_{t+1}^\tau, x_t^\tau). \tag{8}$$

In its original form [Sun et al., 2016], two optimization approaches are adopted for finding F_f or F_b: one uses Forward Training [Ross and Bagnell, 2010] to learn a non-stationary filter, while the other uses Data Aggregation [Ross et al., 2011b] to learn a stationary filter. However, as pointed out by the authors, even if the use of PSIM with Forward Training provably guarantees hypothesis consistency (i.e., the learned filters are equal to the true underlying filters), this solution is often impractical due to its data inefficiency. Conversely, PSIM with Data Aggregation is significantly more data efficient, both in the sample complexity analysis and in the empirical analysis [Sun et al., 2016]. Therefore, in this paper, we use PSIM with Data Aggregation to learn stationary filters for computing approximated predictive forward messages $\hat{m}$ and backward messages $\hat{v}$. The detailed description of the PSIM with DAgger for computing the hypothesis F_f for forward predictive message passing is provided in Alg. 1. The computation of backward predictive states will be similar, since we only need to reverse the belief propagation order (Line 5) using Eq. 7 and replace $f, \hat{m}, \mathcal{F}_1, F_f$ with $h, v, \mathcal{F}_2, F_b$ respectively.

Theoretically, PSIM ensures that the filtering errors resulting from the learned F_f and F_b are upper bounded as:[2]

$$\mathbb{E}_\tau \frac{1}{T} \sum_{t=1}^{T} [\|\hat{m}_t^\tau - \phi(f_t)\|^2] \le \epsilon_m, \tag{9}$$

$$\mathbb{E}_\tau \frac{1}{T} \sum_{t=1}^{T} [\|\hat{v}_t^\tau - \xi(h_t)\|^2] \le \epsilon_v, \tag{10}$$

where ϵ_m and ϵ_v are the regression or classification error on the aggregated dataset [Ross et al., 2011a, Sun et al., 2016]. In practice, both ϵ_m and ϵ_v can be small when we have an expressive hypothesis class and small noise (e.g., small Bayes error) [Ross et al., 2011c].

Therefore, by using PSIM, we can learn two operators F_f and F_b that mimic the forward and backward predictive state passing procedures. With these two operators, we can compute the approximated backward predictive state $\hat{v}_t$ and the forward predictive state $\hat{m}_t$ for the separation set (a_t, s_t) at any time step t (Figure 2).

[2]The original PSIM work by Sun et al. [2016] only provides theoretical bounds for forward message passing. However, for backward message passing, the same bounds directly apply if we reverse message passing direction.

Algorithm 2 Smoothing Machines (SMACH)

```
1: Input: $M$ training trajectories $\tau_i = \{x_t^i, a_t^i\}_{t=1}^{T_i}$, $1 \le i \le M$; Hypothesis class $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$.
2: Run PSIM on $\{\tau_i\}_{t=1}^{M}$ to learn the forward model $F_f \in \mathcal{F}_1$.
3: Run PSIM on $\{\tau_i\}_{t=1}^{M}$ to learn the backward model $F_b \in \mathcal{F}_2$ .
4: Initialize dataset $\mathcal{S} = \emptyset$.
5: for each $\tau_i$, $1 \le i \le M$ do
6:     Roll out $F_f$ forward to generate forward predictive states $\{\hat{m}_t^{\tau_i}\}_{t=1}^{T_i}$ .
7:     Roll out $F_b$ backward to generate backward predictive states $\{\hat{v}_t^{\tau_i}\}_{t=2}^{T_i+1}$ .
8:     Compose input feature $\hat{z}_t^{\tau_i} = (\hat{m}_t^{\tau_i}, \hat{v}_{t+1}^{\tau_i}, x_t^{\tau_i})$ for $1 \le t \le T_i$ .
9:     Add input and output pair $\{(\hat{z}_t^\tau, a_t^{\tau_i})\}_{t=1}^{T_i}$ into $\mathcal{S}$.
10: end for
11: Compute the marginal hypothesis:
        $$\hat{G} = \arg\max_{G \in \mathcal{F}_3} \mathbb{E}_{(z,a)\sim\mathcal{S}}\, \ell(G(z), a). \quad (11)$$
12: Return: $F_f, F_b, \hat{G}$.
```

4.2 SMOOTHING MACHINES

The SMACH algorithm is presented in Alg. 2. SMACH first learns a stationary forward filter $F_f \in \mathcal{F}_1$ and a backward filter $F_b \in \mathcal{F}_2$: given training data, the SMACH algorithm uses PSIM to learn a hypothesis F_f that can compute and pass the forward predictive states $\hat{m}_t$ (Line 2) and, independently, uses PSIM to learn a hypothesis F_b that can compute and pass the backward predictive states $\hat{v}_t$ (Line 3). Due to their independence, learning F_f and F_b can be executed in parallel.

To learn the final message product and marginalization step, we first generate the predictive states by rolling out F_f and F_b on the training trajectories (Line 6 and 7). Next, the algorithm collects the pairs of forward messages $\hat{m}_t$ and backward messages $\hat{v}_{t+1}$, together with the local observation x_t, as the input feature, with the corresponding label a_t as the output. Finally, SMACH learns a classifier or regressor $\hat{G}$ as shown in Eq. 11 (Line 11) by minimizing a loss function ℓ that measures the prediction error. When a_t is discrete, ℓ can be a common classification loss, such as hinge loss, softmax, or cross entropy.

At test time, given a sequence τ, we only have access to the observations $\{x_t^\tau\}$ and need to predict $\{a_t^\tau\}$. With the learned $F_f, F_b, \hat{G}$, we simply roll F_f forward to compute $\{\hat{m}_t^\tau\}$ and roll F_b backward to compute $\{\hat{v}_t\}$, in parallel. With all the messages available, we then use $\hat{G}(\hat{m}_t^\tau, \hat{v}_{t+1}^\tau, x_t^\tau)$ to predict the label. This whole process uses all observations $\{x_t^\tau\}_t$ to predict a_t a posteriori.

4.3 DISCUSSION

Our learning algorithm shares some similarities with the well-known Baum-Welch algorithm for HMMs. Baum-Welch iterates between estimating the posterior distributions of latent states (using forward and backward pass), and optimizing the parameters. SMACH also performs a forward and backward pass to compute the messages represented by predictive states. The key difference is that we leverage predictive state representations (PSRs) to reduce the problem of learning latent chain-CRFs back to the supervised setting, where we can avoid local optimality issues and give strong theoretical guarantees (Sec. 5).

5 THEORETICAL ANALYSIS

Let us assume that we use PSIM to learn F_f to pass predictive states forward on any given sequences of observations $\{x_1, ..., x_T\}$ from τ as $\hat{m}^\tau_{t+1} = F_f(\hat{m}^\tau_t, x^\tau_t)$, and learn F_b to pass predictive states backward as $\hat{v}^\tau_t = F_b(\hat{v}^\tau_{t+1}, x^\tau_t)$. Let us define $\Delta_{m^\tau_t} = m^\tau_t - \hat{m}^\tau_t$, and $\Delta_{v^\tau_t} = v^\tau_t - \hat{v}^\tau_t$, as the difference between the underlying true predictive states (equivalent to the original beliefs $b_t, \tilde{b}_t$ due to the existence of the bijective map) and the predictive states computed from the learned F_f and F_b, respectively.

Eq. 9 and Eq. 10 quantify the filtering error from the approximated predictive states computed from the learned hypothesis F_f and F_b respectively. The ultimate goal of smoothing is to use both past and future information to compute the posterior distribution of a label more accurately. As one might expect, if we can exactly compute the forward and backward messages, namely $\hat{m}_t = m_t$ and $\hat{v}_{t+1} = v_{t+1}$ for any t and τ, then, in a realizable case, we can exactly learn a hypothesis that takes $\hat{m}_t$, $\hat{v}_t$ and the local observations as inputs and outputs the posterior distributions of the labels. However, one may wonder if we can still achieve strong prediction guarantees on the learned model even if we can only approximately compute m_t and v_t. Moreover, we may be interested in quantifying the performance of the learned model based on the accuracy of the approximated predictive states (e.g., using Δ_{m_t} and Δ_{v_t}). In order to perform this type of analysis, we first introduce some lemmas and notation.

The following lemma extends the results for PSIM shown in Eq. 9 and 10, by explicitly quantifying $\Delta_{\hat{m}_t}$ and $\Delta_{\hat{v}_t}$:

Lemma 5.1. *Given Eq.* 9 *and* 10 *from PSIM, for* $\Delta_{\hat{m}_t}$ *and* $\Delta_{\hat{v}_t}$*, we have:*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_\tau[\|\Delta_{m^\tau_t}\|^2] \leq 2(\epsilon_m + \delta_m); \tag{12}$$

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_\tau[\|\Delta_{v^\tau_t}\|^2] \leq 2(\epsilon_v + \delta_v); \tag{13}$$

where $\delta_m = \frac{1}{T}\sum_{t=1}^T \mathbb{E}_\tau[\|m^\tau_t - \phi(f_t)\|^2]$ *and* $\delta_v = \frac{1}{T}\sum_{t=1}^T \mathbb{E}_\tau[\|v^\tau_t - \xi(h_t)\|^2]$.

Proof for this lemma and the lemmas/theorems in the remainder of this paper are all included in Appendix.

The above lemma states that the size of Δ_m (or Δ_v) is upper bounded by the sum of ϵ_m (or ϵ_v), which is the risk resulting from the predicted messages, and δ_m (or δ_v), which is the Bayes error resulting from the system itself. Note that the Bayes error is purely determined by the underlying systems and has nothing to do with the learning algorithms. In general, we cannot guarantee the elimination of the Bayes errors. This is because when aiming at learning a stationary F_f (or F_b) for forward (or backward) predictive message passing, the objective (Eq. 5) that PSIM tries to optimize is non-convex. Even ideally assuming that PSIM is *risk* consistent, i.e., it finds an F that has the same filtering error (risk) as the true underlying filter for predictive states, which is the best we can do for a general non-convex objective, we still cannot promise that the learned F is exactly the true underlying filter. Therefore, even if F is risk consistent with respect to the underlying true filter, we cannot promise that the approximated predictive state $\hat{m}_t$ generated from F would be exactly equal to the true underlying state m_t. More detailed explanation is included in Appendix. However, as shown in Lemma. 5.1, the smaller the filtering error from PSIM, the better approximation we get for the predictive states.

To analyze the performance of the learned marginalization hypothesis $\hat{G}$, we first assume that the loss function $\ell(\cdot,\cdot)$ that we are using for classification is Lipschitz continuous, with constant L_1 with respect to the first item ($G(z)$). Commonly used classification loss functions have this property (e.g., hinge loss, logistic loss). Additionally, we assume that $G(\cdot)$, for any $G \in \mathcal{F}_3$, is also Lipschitz continuous with constant L_2 with respect to z. This is also a reasonable assumption for common hypothesis classes such as linear, quadratic and, in fact, any differentiable hypotheses with bounded first derivatives.

Let us define G^* as the minimizer of the true risk measured under the true messages (m^τ_t, v^τ_t):

$$G^* = \arg\max_{G\in\mathcal{F}_3} \mathbb{E}_\tau[\ell(G(z^\tau_t), a^\tau_t)], \tag{14}$$

where z^τ_t is composed as $(m^\tau_t, v^\tau_{t+1}, x^\tau_t)$ with the true underlying predictive messages m^τ_t, v^τ_{t+1} on the sequence τ. With sufficient feature functions ϕ and ξ, m^τ_t and v^τ_t will be equivalent to b^τ_t and $\tilde{b}^\tau_t$ subject to a bijective map. Hence, in a realizable case, G^* encodes as much information as the true underlying marginalization step in the original latent chain-CRF, as shown in Eq. 3. Let us first analyze the learned model $\hat{G}$ under the assumption of infinite many training trajectories ($M \to \infty$):

Theorem 5.2. *Assuming* F_f *and* F_b *from PSIM generate the messages* $\hat{m}_t$ *and* $\hat{v}_t$ *satisfying Eq. 12 and 13, the* *marginalization hypothesis $\hat{G}$ obtained from Eq. 11 has the following property:*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{\tau\sim\mathcal{D}}[\ell(\hat{G}(\hat{z}_t^\tau), a_t^\tau)]$$
$$\leq \frac{1}{T}\mathbb{E}_{\tau\sim\mathcal{D}}[\ell(G^*(z_t^\tau), a_t^\tau)] + O(\epsilon_m + \epsilon_v + \delta_m + \delta_v).$$

The above theorem shows that if PSIM learns good hypothesis F_f and F_b for estimating messages (e.g., ϵ_m and ϵ_m are small) and the underlying noise of the dynamical systems is not big, the learned hypothesis $\hat{G}$ can achieve competitive smoothing performance with respect to G^*—the global minimizer of the smoothing risk with access to the true messages (e.g., m_t^τ, v_t^τ).

For Theorem 5.2 we assumed an infinite number of training sequences, although in practice, we only have a finite number of sequences. To show the finite sample complexity of our algorithm, we additionally assume that the training sequences $\tau_1, ..., \tau_M$ are separated in two halves. We use the first half to learn F_f and F_b with PSIM, and the second half to generate messages $\hat{m}$ and $\hat{v}$ with F_f and F_b, and then train $\hat{G}$ for the marginalization step. This assumption ensures that, at every time step t, the generated messages $\{\hat{m}_t^{\tau_{M/2}}, \hat{m}_t^{\tau_{M/2+1}}, ..., \hat{m}_t^{\tau_M}\}$ are i.i.d sampled (note that F_f is independent with respect to $\tau_{M/2}, ..., \tau_M$, and each τ is i.i.d sampled). Let us define the distribution $\hat{d}_t$ as the distribution of $\hat{z}_t^\tau = (\hat{m}_t^\tau, \hat{v}_{t+1}^\tau, x_t^\tau)$. Note that $\{(\hat{m}_t^{\tau_i}, \hat{v}_{t+1}^{\tau_i}, x_t^{\tau_i})\}_{i=M/2}^{M}$ will be i.i.d sampled from $\hat{d}_t$. For convenience, we assume that we have $2M$ training sequences. We use the first M sequences for PSIM to learn F_f and F_b and the remaining M for learning $\hat{G}$.

Using the finite sample analysis for PSIM with Data Aggregation as the optimization tool,we first extend Lemma. 5.1 to the corresponding high probability bounds:

Lemma 5.3. *Using PSIM with Data Aggregation, with probability $1-\delta$, we have:*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_\tau[(\Delta_{m_t^\tau})] \leq 2\hat{\gamma}_m + 2\hat{\epsilon}_m + 2\delta_m + O(\sqrt{\frac{\ln(1/\delta)}{MN}});$$
$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_\tau[(\Delta_{v_t^\tau})] \leq 2\hat{\gamma}_v + 2\hat{\epsilon}_v + 2\delta_v + O(\sqrt{\frac{\ln(1/\delta)}{MN}});$$

where N is number of iterations $PSIM$ used and $\hat{\gamma}_m$ and $\hat{\gamma}_v$ converge to zero as $N \to \infty$

We present the finite sample analysis using Rademacher complexity. We define $\mathcal{R}_t(\mathcal{F}_3)$ as the Rademacher number of the hypothesis class $\mathcal{F}_3$ under distribution $\hat{d}_t$, and $\bar{\mathcal{R}}(\mathcal{F}_3) = (\mathcal{R}_1(\mathcal{F}_3) + ... + \mathcal{R}_T(\mathcal{F}_3))/T$, as the average Rademacher number across T time steps. In our analysis, we assume we know T or the upper bound of T.

	KF	EKF	KS	iEKS	SMACH-0	Avg $\|a_t\|_2$
Cart-Pole	12.80	2.14	4.13	2.02	**0.29**	0.65
Bicycle	0.093	0.068	0.091	0.067	**0.065**	2.01
Helicopter	$\sim$	2.49	$\sim$	2.19	**2.17**	21.98
Swimmer	$\sim$	1.90	$\sim$	1.72	**0.69**	9.61

Table 1: Prediction error of different approaches without latent states (a_t encodes the exact state of the system).

Theorem 5.4. *Given $2M$ training sequences for Alg. 2, as $N \to \infty$, with probability $1-\delta$, for any $G^* \in \mathcal{F}_3$, we have:*

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{\tau\sim\mathcal{D}}[\ell(\hat{G}(\hat{z}_t^\tau), a_t^\tau)] \leq \frac{1}{T}\sum_{t=1}^{T}\mathbb{E}_{\tau\sim\mathcal{D}}[\ell(G^*(z_t^\tau), a_t^\tau)]$$
$$+ \tilde{O}(\sqrt{\frac{\ln(1/\delta)}{M}}) + O(\bar{\mathcal{R}}(\mathcal{F}_3) + \hat{\epsilon}_m + \hat{\epsilon}_v + \delta_v + \delta_m).$$

6 EXPERIMENTS

We test SMACH on two different types of datasets: (1) datasets with continuous labels a_t (SMACH needs to perform *regression*), which are collected from several simulated robotics dynamical systems, and (2) datasets whose labels a_t are discrete (SMACH needs to perform *classification*), which are from two domains: Optical Character Recognition, a sequential image recognition task, and questions and answers recognition [McCallum et al., 2000], a Natural Language Processing task.

6.1 REGRESSION TASKS

To show that our approach is able to deal with complicated, non-linear dynamical models of robotics systems, we test our approach on four classic simulated dynamics models: (1) Cart-Pole Balancing, (2) Bicycle Balancing [Ernst et al., 2005], (3) Helicopter Hover [Abbeel et al., 2005] and (4) Swimmer [Tassa et al., 2008]. The simulated models are available from RLPy [Geramifard et al., 2013].

We compare SMACH to classic physics-based algorithms: the Kalman Filter (KF), the Kalman Smoother (KS), the Extended Kalman Filter (EKF), and the iterated Extended Kalman Smoother (iEKS) [Bell, 1994]. We first consider the chain-CRF structure without latent states (no s_t in Fig. 1a). Hence, the label a_t represents the *full* state of the robot, and x_t is generated from a_t through a stochastic observation model. Since there are no latent states, we set $k = 0$. Here, we define the smoothing error as the average of the prediction errors $\|a - \hat{a}\|_2$.

In our setup, we allow KF, KS, EKF, iEKS to access the real underlying dynamical models and observation models. For KF and KS, we linearize the real dynamics and observation model around the balancing state. Note that directly comparing to KF, KS EKF, and iEKS is not fair since these approaches have access to the true stochastic dynamical models, while SMACH only has access to the data generated from the models. Nevertheless, as we can see from

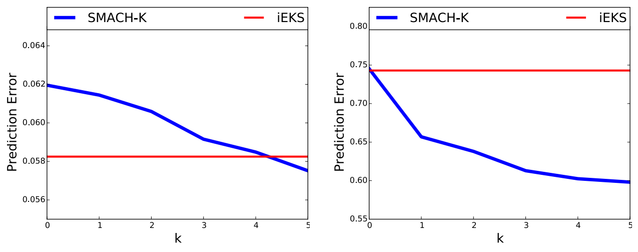
Tab. 1, by using Kernel Ridge regression (i.e., $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_t$ are Reproducing Kernel Hilbert Spaces), SMACH outperforms these classic physics-based approaches on all four testbeds.[3] We only tested SMACH-0 here as we do not have latent states s_t in the setup here. As we expected, increasing k (using predictive states) does not give noticeable improvement on the prediction error.

To test the performances of SMACH with latent states, we separate the full state into two parts: labels a_t and latent states s_t (see Appendix for the detailed components of s_t and a_t). Under this setup, SMACH is never allowed to access s_t. Due to the existence of latent states, we set $k \geq 1$ (i.e. we use predictive states). Figure 3a and 3b show the performance of SMACH as k increases on the bicycle balancing and swimmer datasets. A similar trend is observed on the other two datasets. Overall, the smoothing performance improves as k increases, which illustrates that longer predictive states can potentially capture more information about latent states. Interestingly, SMACH outperforms iEKS (note that we still give iEKS full access to the latent states s_t in order to perform Kalman smoothing, but we only report smoothing prediction error with respect to label a_t). Since iEKS can be understood as applying the Gauss-Newton algorithm on the log likelihood, which could be non-convex due to non-linear dynamics and observation models, it is likely that iEKS will be stuck at locally optimal solutions.

Since KF, KS, EKF, and iEKS have access to the perfect stochastic models, this set of experiments also supports one of our main claims: separately learning the models and then using the learned models to perform inference may result in decreased performance. This can happen even if we can leverage powerful learning algorithms to learn the perfect models (e.g., using Gaussian Process [Deisenroth et al., 2012] or Hilbert space embeddings [Nishiyama et al., 2016] to model dynamics) due to the unavoidable approximations that one usually needs in order to make inference computationally tractable (e.g., linearize the real/learned dynamics and observations models as KF, KS, GP-EKF and GP-EKS [Ko and Fox, 2009] do). These approximations on the learned/real models may cancel out the benefits derived from the availability of perfect models. SMACH, instead, directly learns powerful predictors to optimize the smoothing performance. Since the predictors operate as black boxes to directly perform the smoothing operation, no further approximation is needed for inference.

6.2 CLASSIFICATION TASKS

We evaluate SMACH on classification tasks belonging to two different domains: Optical Character Recognition and



(a) Bicycle (b) Swimmer

Figure 3: Performance of SMACH with respect to k (length of predictive states) for models with latent states.

questions and answers recognition (FAQ) [McCallum et al., 2000]. In this case, for fair comparison to previous approaches, we use standard feature design. We compare SMACH to two families of algorithms: (1) search-based prediction algorithms such as SEARN [Daumé III et al., 2009], DAgger [Ross et al., 2011b], and PSIM [Sun et al., 2016], (2) CRF [Lafferty et al., 2001] and its extensions, such as Hidden-unit CRF [van der Maaten et al., 2011] and NeuroCRFs [Do et al., 2010].

Optical Character Recognition The OCR dataset contains 6877 handwritten words. Each word is represented as a series of handwritten characters and there are 52152 total characters. Each character is a binary 16×8 image, leading to 128-dimensional binary feature vectors. Each character is one of the 26 letters in the English alphabet. We define the binary vector for character at slot t as c_t. The task is to predict the identity of each character, given a sentence. In our experiments, we use a 7-step time window: we represent x_t as $[c_{t-3}, ..., c_t, ..., c_{t+3}]$ and we test SMACH with different k (SMACH-k).

We first test SMACH on a small experimental setting, where we separate the dataset into 10 folds and perform training on one fold while testing on the remaining 9 folds. Table 2 shows the comparison between SMACH and other closely related approaches.[4] SMACH outperforms SEARN, DAgger and PSIM, which are the state-of-art search-based prediction algorithms. Compared to probabilistic graphical model approaches, SMACH also has better performance than CRFs. From Table 2, we see that from DAgger (equivalent to PSIM-0) to PSIM-1 and PSIM-2 the prediction error decreases when we use, in the predictive messages, more steps of future/past labels and observations. This is consistent with the claim from Sun et al. [2016] that a larger k can lead to more accurate predictive states. As a result, SMACH-2 surpasses SMACH-1, since SMACH-2 uses more accurate predictive messages computed from PSIM-2. This evidence agrees with Theo-

[3]Note that, for iEKS, we use the solutions from EKF as initialization. Simply using random initialization, or the average of the training trajectories as initialization, does not perform well.

[4]The results of SVM^{struct}, SVM^{struct}, M^{3}N, CRF are from [Nguyen and Guo, 2007]; SVM^{struct2} and CRF^2 are from [Keerthi and Sundararajan, 2007]

SVM^{struct}	SVM^{struct2}	M^{3}N	SEARN
21.16	19.24	25.08	27.02
CRF	**CRF^2**	**Hidden unit CRF**	**DAgger**
32.30	19.97	18.36	30.02
PSIM-1	**PSIM-2**	**SMACH-1**	**SMACH-2**
26.11	23.89	18.41	16.21

Table 2: Comparison of SMACH with previous related approaches on the small OCR experimental setting.

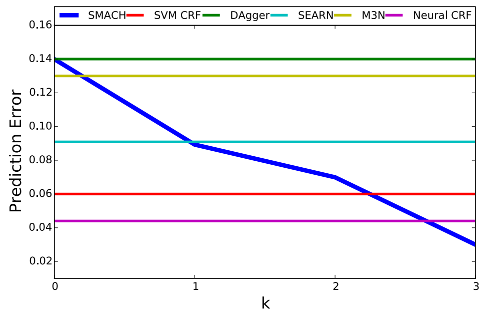


Figure 4: Performance of SMACH-k on large OCR task.

rem 5.4: when we have more accurate predictive messages for the marginal step (e.g., smaller ϵ), the marginal step has a smaller generalization error. We also ran SMACH on a larger setting (training on 9 folds and testing on the remaining one) with different k, reporting the results in Figure 4.[5] From the graph, we note that larger k can greatly improve smoothing performance. SMACH-3 achieves an average prediction error of 3.5%, while Hidden unit CRF achieves a 2.0% prediction error. Note, however, that SMACH-2 achieves a smaller error in the small experiment.

Recognizing questions and answers We also test SMACH on the FAQ dataset from McCallum et al. [2000]. We use the same feature representation from McCallum et al. [2000], where each sentence is described using a 24-dimensional binary vector. Each sentence in the FAQ data set is labeled by one of four labels: (1) question, (2) answer, (3) header, or (4) footer. We use linear SVM for PSIM for computing backward and forward predictive states. For the marginalization step, we use two classifiers: Linear SVM with Random Fourier feature (RFF-SVM) and Random Forest (RF) with 60 trees and maximum depth 30.

From Table 3, we see that both SMACH with RFF-SVM and SMACH with RF achieve performances which are comparable to Hidden-unit CRF [van der Maaten et al., 2011] with Stochastic Gradient Descent as the optimization (we note that SMACH performs slightly better than hidden-unit CRF with other optimizers like Perceptron and BFGS). SMACH with RF gives slightly better performance than SMACH with RFF-SVM. This indicates that using a powerful classifier for the marginal step can potentially enhance the performance. We also tested $k = 1$, leading to a slightly worse performance than $k = 0$. Since we are only allowed to use one file (one sequence) to train the model and test on all the remaining files in the same group [McCallum et al., 2000], when we increase k, we may not have enough training data, since the complexity of the hypothesis classes increases as k becomes larger.

[5]The result of SVM+CRF is from [Hoefel and Elkan, 2008].

Method	Error (%)
Linear SVM [Do et al., 2010]	9.87
Linear CRF [Maaten et al., 2011]	6.54
NeuroCRFs [Do et al., 2010]	6.05
Hidden-unit CRF [Maaten et al., 2011]	4.43
DAgger	7.47
SMACH-0 with RFF-SVM	5.10
SMACH-0 with RF	5.01

Table 3: Comparison of SMACH with related approaches on the FAQ dataset.

7 CONCLUSION

In this paper we present Smoothing Machines (SMACH), a data-driven approach that directly optimizes *smoothing* performance on time series with latent states. We use the concepts of predictive states and inference machines to directly learn functions that mimic forward and backward message passing in our system. Under this setting, we can achieve strong performance guarantees for the ultimate inference task (i.e., smoothing) in the agnostic setting. We show that SMACH outperforms classic physics-based smoothing algorithms on dynamical systems with complicated non-linear transition models. We also show that in the presence of latent states, using more features (i.e., a longer time window) can boost the algorithm's performance. Additionally, our experimental results on classification tasks are promising. We leave the application of SMACH on more complicated NLP tasks (e.g., Part of Speech Tagging) as future work.

ACKNOWLEDGEMENTS

This material is based upon work supported by DARPA ALIAS contract number HR0011-15-C-0027 and NSF CRII Award No. 1464219.

References

Pieter Abbeel, Varun Ganapathi, and Andrew Y Ng. Learning vehicular dynamics, with application to modeling helicopters. In *NIPS*, pages 1–8, 2005.

Bradley M Bell. The iterated Kalman smoother as a gauss-newton method. *SIAM Journal on Optimization*, 4(3): 626–636, 1994.

Byron Boots. *Spectral Approaches to Learning Predictive*

Representations. PhD thesis, Carnegie Mellon University, 2012.

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

Byron Boots, Arthur Gretton, and Geoffrey J. Gordon. Hilbert space embeddings of predictive state representations. In *UAI-2013*, 2013.

Hal Daumé III, John Langford, and Daniel Marcu. Search-based structured prediction. *Machine learning*, 75(3): 297–325, 2009.

Marc Peter Deisenroth, Ryan Darby Turner, Marco F Huber, Uwe D Hanebeck, and Carl Edward Rasmussen. Robust filtering and smoothing with Gaussian processes. *Automatic Control, IEEE Transactions on*, 57(7):1865–1871, 2012.

Trinh Do, Thierry Arti, et al. Neural conditional random fields. In *AISTATS*, pages 177–184, 2010.

Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6:503–556, 2005.

Alborz Geramifard, Robert H Klein, and JP How. Rlpy: The reinforcement learning library for education and research. *Machine Learning Open Source Software*, 2013.

Ahmed Hefny, Carlton Downey, and Geoffrey J Gordon. Supervised learning for dynamical system learning. In *NIPS*, pages 1954–1962, 2015.

Guilherme Hoefel and Charles Elkan. Learning a two-stage SVM/CRF sequence classifier. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 271–278. ACM, 2008.

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

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

S Sathiya Keerthi and Sellamanickam Sundararajan. CRF versus SVM-struct for sequence labeling. Technical report, Technical report, Yahoo Research, 2007.

Jonathan Ko and Dieter Fox. GP-Bayesfilters: Bayesian filtering using gaussian process prediction and observation models. *Autonomous Robots*, 27(1):75–90, 2009.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Alex Kulesza, N Raj Rao, and Satinder Singh. Low-rank spectral learning. In *AISTATS*, 2014.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *ICML*. Morgan Kaufmann, 2001.

Laurens Maaten, Max Welling, and Lawrence K Saul. Hidden-unit conditional random fields. In *AISTATS*, pages 479–488, 2011.

Andrew McCallum, Dayne Freitag, and Fernando CN Pereira. Maximum entropy markov models for information extraction and segmentation. In *ICML*, volume 17, pages 591–598, 2000.

Nam Nguyen and Yunsong Guo. Comparisons of sequence labeling algorithms and extensions. In *ICML*, 2007.

Yu Nishiyama, Amir Hossein Afsharinejad, Shunsuke Naruse, Byron Boots, and Le Song. The nonparametric kernel Bayes smoother. In *AISTATS*, 2016.

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

Stéphane Ross, Geoffrey J Gordon, and J. Andrew Bagnell. No-regret reductions for imitation learning and structured prediction. In *AISTATS*, 2011a.

Stéphane Ross, Geoffrey J Gordon, and J Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. *AISTATS*, 2011b.

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

Le Song, Byron Boots, Sajid M Siddiqi, Geoffrey J Gordon, and Alex J Smola. Hilbert space embeddings of hidden markov models. In *ICML*, pages 991–998, 2010.

Karl Stratos, Alexander M Rush, Shay B Cohen, and Michael Collins. Spectral learning of refinement hmms. In *CoNLL*, pages 56–64, 2013.

Wen Sun, Arun Venkatraman, Byron Boots, and J Andrew Bagnell. Learning to filter with predictive state inference machines. In *ICML*, 2016.

Yuval Tassa, Tom Erez, and William D Smart. Receding horizon differential dynamic programming. In *Advances in neural information processing systems*, pages 1465–1472, 2008.

Laurens van der Maaten, Max Welling, and Lawrence K Saul. Hidden-unit conditional random fields. In *AISTATS*, pages 479–488, 2011.

Arun Venkatraman, Wen Sun, Martial Hebert, Byron Boots, and J. Andrew Bagnell. Inference machines for nonparametric filter learning. In *IJCAI-2016*, 2016.