

---

# Practical Learning of Predictive State Representations

---

Carlton Downey<sup>1</sup> Ahmed Hefny<sup>1</sup> Geoffrey Gordon<sup>1</sup>

## Abstract

Over the past decade there has been considerable interest in spectral algorithms for learning Predictive State Representations (PSRs). Spectral algorithms have appealing theoretical guarantees; however, the resulting models do not always perform well on inference tasks in practice. One reason for this behavior is the mismatch between the intended task (accurate filtering or prediction) and the loss function being optimized by the algorithm (estimation error in model parameters).

A natural idea is to improve performance by refining PSRs using an algorithm such as EM. Unfortunately it is not obvious how to apply apply an EM style algorithm in the context of PSRs as the Log Likelihood is not well defined for all PSRs. We show that it is possible to overcome this problem using ideas from Predictive State Inference Machines (Sun et al., 2016).

We combine spectral algorithms for PSRs as a consistent and efficient initialization with PSIM-style updates to refine the resulting model parameters. By combining these two ideas we develop Inference Gradients, a simple, fast, and robust method for practical learning of PSRs. Inference Gradients performs gradient descent in the PSR parameter space to optimize an inference-based loss function like PSIM. Because Inference Gradients uses a spectral initialization we get the same consistency benefits as PSRs. We show that Inference Gradients outperforms both PSRs and PSIMs on real and synthetic data sets.

## 1. Introduction

Predictive state representations (PSRs) (Littman et al., 2001) are a class of models for filtering, prediction, and simulation

<sup>1</sup>Carnegie Mellon University, Pittsburgh, PA, USA. Correspondence to: Carlton Downey <cmdowney@cs.cmu.edu>, Ahmed Hefny <ahefny@cs.cmu.edu>, Geoffrey Gordon <ggordon@cs.cmu.edu>.

of discrete time dynamical systems. PSRs provide a compact representation of a dynamical system by representing state as a set of predictions of features of future observations. This representation is known as a *predictive state*, in contrast to the *latent state* present in models such as Hidden Markov Models (HMMs).

PSRs are an important class of models because, due to the (noisy but direct) observability of their parameters, there exist spectral algorithms for learning PSR model parameters which are statistically consistent, computationally efficient, and globally optimal. In contrast, competing techniques based on the likelihood function (such as Expectation Maximization (EM)) are often slow to converge (Wu, 1983) and are highly susceptible to local optima.

Despite the appealing theoretical properties of spectral algorithms for PSRs these models have seen limited use in practice due to underwhelming experimental performance on many problems. In an effort to overcome this issue several authors have proposed using an EM style algorithm to post process/fine tune the model parameters of a PSR learned using a spectral algorithm (Jiang et al., 2016; Shaban et al., 2015). Unfortunately EM algorithms are based on the log likelihood function, which is not well defined for all PSRs. Specifically, filtering and prediction in PSRs learned using a spectral algorithm can produce states which are unbounded in size, or which correspond to negative probabilities.

We show that this problem can be solved by considering an alternative objective function which is not based on the log likelihood. We present a simple, computationally efficient algorithm which allows us to fine tune the model parameters of an arbitrary PSR and results in large improvements in experimental performance.

Sun et al. (Sun et al., 2016) introduce the idea of Predictive State Inference Machines (PSIMs) which directly train models to perform inference. They show that models trained in this way can outperform both latent state models and models trained via spectral techniques. Furthermore they establish asymptotic and finite sample bounds on the filtering performance of the resulting models. However the form of these results implies that the finite sample performance of their algorithm may depend heavily on how it is initialized. Furthermore the results assume that the training of the in-

ference model can be performed perfectly, which may not hold for the non-linear filtering models that can emerge in discrete systems. In this setting, having a good initialization becomes even more important.

We combine spectral algorithms for PSRs as a consistent and efficient initialization with ideas from PSIMs to refine the model parameters in order to achieve good predictive performance. By combining these two ideas we get Inference Gradients, a simple, fast, and robust method for practical learning of PSRs. Inference Gradients allows us to perform gradient descent in the PSR parameter space to optimize an inference based loss function without being forced to define a proxy to the log likelihood. We show that Inference Gradients outperforms both PSRs and PSIMs on two real and synthetic data sets.

## 2. Background

### 2.1. Dynamical Systems

A dynamical system is a stochastic process (i.e., a distribution over sequences of observations) such that, at any time, the distribution of future observations is fully determined by a vector  $s_t$  called the latent state. Note that the distribution of  $s_t$  depends only on history. The process is specified by three distributions: the initial state distribution  $P(s_1)$ , the state transition distribution  $P(s_{t+1} | s_t)$ , and the observation distribution  $P(o_t | s_t)$ . Given a dynamical system, one of the fundamental tasks is to perform inference, where we predict future observations given a history of observations. Typically this is accomplished by maintaining a distribution or belief over latent states  $b_{t|t-1} = P(s_t | o_{1:t-1})$ , where  $o_{1:t-1}$  denotes the first  $t-1$  observations.  $b_{t|t-1}$  represents both our knowledge and our uncertainty about the true state of the system.

Two core inference tasks are filtering and prediction. In filtering, given the current belief  $b_t = b_{t|t-1}$  and a new observation  $o_t$ , we calculate an updated belief  $b_{t+1} = b_{t+1|t}$  that incorporates  $o_t$ . In prediction, we project our belief into the future: given a belief  $b_{t|t-1}$  we estimate  $b_{t+k|t-1} = P(s_{t+k} | o_{1:t-1})$  for some  $k > 0$  (without incorporating any intervening observations).

### 2.2. Predictive State Representations (PSRs)

The classical approach for modelling a dynamical system is to explicitly estimate the initial, transition, and observation distributions. Estimates of these distributions can be used compute and update a belief over latent states, which in turn allows us to make predictions of future observations. PSRs take an alternative approach: instead of maintaining a belief  $b_t$  over latent states  $s_t$ , they maintain a *predictive state* represented by the expected value of a sufficient statistic of future observations (Jaeger, 1999; Littman et al., 2001;

Singh et al., 2004; Boots & Gordon, 2010; Boots et al., 2009; Hefny et al., 2015). In this work we will use the PSR formulation of (Hefny et al., 2015; Venkatraman et al., 2016).

In more detail, we define a predictive state  $q_t = q_{t|t-1} = E[\psi_t | o_{1:t-1}]$ , where  $\psi_t = \psi(o_{t:t+k-1})$  is a vector of features of future observations. The features are chosen such that  $q_t$  determines the distribution of future observations  $P(o_{t:t+k-1} | o_{1:t-1})$ .<sup>1</sup> Filtering then becomes the process of mapping a predictive state  $q_t$  to  $q_{t+1}$  conditioned on  $o_t$ , while prediction maps a predictive state  $q_t = q_{t|t-1}$  to  $q_{t+k|t-1} = E[\psi_{t+k} | o_{1:t-1}]$  without intervening observations.

PSRs are based on the idea that it is sufficient to maintain  $q_t$  in order to make predictions about future observations conditioned on the history. The key benefit offered by PSRs is that by using an observable representation of state we can develop efficient globally convergent algorithms (Singh et al., 2004).

### 2.3. Spectral Learning of PSRs

The classical approach to learning a dynamical system is to optimize the model parameters by maximizing the likelihood of the observed data. Unfortunately the likelihood function is often highly non-convex, leading to local optima and sub-optimal model parameters. Spectral algorithms offer an alternate approach to learning: they use the method of moments to set up a system of equations that can be solved in closed form to recover estimates of the desired parameters. In this process, they typically factorize a matrix or tensor of observed moments—hence the name “spectral”.

Hefny et al. (Hefny et al., 2015) show that spectral learning of dynamical systems can be formulated as solving a sequence of regression problems. We follow this approach, referred to as *two-stage regression* or **2SR**, in our work. Under this framework, spectral learning of PSRs corresponds to learning a set of regression models. In the case of a discrete observation space<sup>2</sup>, it can be shown that two stage regression with linear models is equivalent to learning an ordinary PSR, that is, an initial state  $q_1$ , a normalizer  $b_\infty$ , and set of linear operators  $B_i$  for  $i \in \{1, \dots, k\}$  such that:

$$q_{t+1} = \frac{B_{o_t} q_t}{b_\infty^T B_{o_t} q_t}$$

<sup>1</sup>For convenience we assume that the system is  $k$ -observable: that is, the distribution of all future observations is determined by the distribution of the next  $k$  observations. (Note: not by the next  $k$  observations themselves.) At the cost of additional notation, this restriction could easily be lifted.

<sup>2</sup>While we focus on the discrete setting for ease of exposition, the main idea of this work can be extended to continuous settings.

Given a set of training examples  $(h_t, o_t, \psi_t, \psi_{t+1})$  for  $1 \leq t \leq T$ , we can estimate these parameters as follows:

$$q_1 = \frac{1}{T} \sum_{t=1}^T \psi_t \quad (1)$$

$$B_i = \left( \sum_{t=1}^T \psi_t h_t^\top \right)^+ \left( \sum_{t=1}^T 1(o_t = i) \psi_{t+1} h_t^\top \right) \quad (2)$$

$$b_\infty^T = \left( \sum_{t=1}^T h_t^\top \right) \left( \sum_{t=1}^T \psi_t h_t^\top \right)^+ \quad (3)$$

Note that while each  $B_i$  is linear, the state update is *non-linear* due to the normalization term.

#### 2.4. Predictive State Inference Machines (PSIMs)

PSIMs (Sun et al., 2016) constitute an inference method for dynamical systems that combines (1) the idea of a predictive state from PSRs (where the state is a prediction of future statistics) and (2) inference machines (Langford et al., 2009; Ross et al., 2011) where, instead of learning generative model parameters that are then fed into a fixed inference function, we directly learn the inference function.

PSIMs directly learn an inference function by minimizing predictive loss on the training set. In contrast, a PSR specifies a data *generation* model, and the goal of the spectral algorithm is to identify the parameters of this model by matching training statistics. These parameters indirectly determine the inference function.

More formally, the goal is to learn a function  $f$  that can deterministically pass the predictive states forward in time conditioned on the latest observation – that is,  $\hat{q}_{t+1} = f(\hat{q}_t, o_t)$  such that the likelihood of the observations  $o_t$  generated from the sequence of predictive states is maximized.

As this is a sequential prediction problem over the predictive states, we use DAgger (Ross et al., 2011) to optimize the inference machine. The choice of learner for  $f$  can be any no-regret regression or classification algorithm.

We note that PSIM updates can be implemented as gradient descent on the loss function  $L(f) = \frac{1}{T} \sum_{t=1}^T \|\hat{q}_t - q_t\|_2^2$  where  $\hat{q}_t$  is the state estimate at time  $t$  produced by filtering, while  $q_t$  is the true state at time  $t$ . This procedure is outlined in Algorithm 1:

### 3. Related Work

The classic approach for refining a dynamical system model is the EM algorithm. EM iteratively adjusts the model parameters to locally optimize the log likelihood. Unfortunately inference in PSRs can produce states which are

---

#### Algorithm 1 PSIM

---

**Input:** sequence of observations  $o_{1:T}$ , learning rate  $\alpha \in \mathbb{R}^+$ , number of iterations  $n \in \mathbb{N}^+$ .

```

 $f \leftarrow$  An arbitrary hypothesis.
for  $i = 1$  to  $n$  do
     $\hat{q}_1 \leftarrow q_1$ 
    for  $t = 1$  to  $T$  do
         $\hat{q}_{t+1} \leftarrow f(\hat{q}_t, o_t)$ 
         $f \leftarrow f - \alpha \Delta L(f)$ 
    end for
end for

```

---

unbounded in size, or which correspond to negative probabilities. This means that the log likelihood is not well defined for arbitrary PSR parameters. Several authors attempt to solve this problem in a variety of ways.

Jiang et al. (Jiang et al., 2016) propose a gradient descent algorithm for improving the performance of PSRs where they optimize a proxy to the log likelihood. Given a PSR  $\mathcal{B} = (b_1, b_\infty, \{B_i\}_{i=1}^\infty)$  and a sequence of observations  $o = o_1, \dots, o_n$ , the negative log likelihood is:

$$\ell(\mathcal{B}; o) = -\log(P_{\mathcal{B}}(o)) = -\log(b_\infty B_n \dots B_1 b_1) \quad (4)$$

Because the log likelihood is not well defined for an arbitrary PSR, Jiang et al. choose to optimize a related loss function which rectifies and re-normalizes each predicted observation probability distribution:

$$\ell(\mathcal{B}; o) = -\log \left( \frac{|P_{\mathcal{B}}(o)|}{\sum_{y \in O^{\parallel o \parallel}} |P_{\mathcal{B}}(y)|} \right)$$

where  $O^{\parallel o \parallel}$  is the space of all observation sequences with the same length as  $o$ . This yields the gradient

$$\begin{aligned} \frac{d}{dB_i} \ell(\mathcal{B}; o) &= \frac{d}{dB_i} \log \left( \sum_{y \in O^{\parallel o \parallel}} \|P_{\mathcal{B}}(y)\| \right) \\ &\quad - \frac{d}{dB_i} \log \|P_{\mathcal{B}}(o)\| \end{aligned}$$

This expression is analytically intractable, so they propose a stochastic gradient descent procedure where they approximate this expression using contrastive divergence. They show that this approach can be used to significantly improve the performance of a PSR initialized via spectral techniques. This approach allows for gradient descent on a surrogate to the log loss, however the resulting algorithm has a complex update rule, and can be slow in practice.

Shaban et al. (Shaban et al., 2015) propose a two pass algorithm for learning a dynamical system model, where they first learn a PSR using a spectral algorithm, then subsequently convert the PSR into a valid HMM using an exterior

point method. Their approach is different from ours as they produce an HMM as the final model rather than a PSR. Additionally, their algorithm focuses on model parameter optimization rather than optimal performance on inference tasks.

Sun et al. (Sun et al., 2016) (see background) attempt to learn an inference model directly using the observable features of PSRs, but without the PSR model. This approach means that they are no longer learning a model class which includes HMMs. Furthermore their algorithm requires a good initialization in order to perform well.

These methods offer significant improvements over PSRs trained using two-stage regression on many problems, however they also come with disadvantages. We would like an algorithm with PSIM-like behavior, but (1) on a model class such as PSRs and (2) which includes HMMs and a good initialization.

## 4. Inference Gradients

One reason that two-stage regression for PSRs can perform poorly on finite data sets is the mismatch between the intended task (accurate filtering or prediction) and the loss function being optimized by the algorithm (estimation error in model parameters). In the realizable setting, under appropriate regularity conditions, spectral algorithms asymptotically converge to the true model parameters. Unfortunately these results tell us little about how models learned from finite data will perform on filtering or prediction tasks in practice. Indeed, PSRs learned using a spectral algorithm can produce poor predictions when applied recursively to the original training data, even if they are close to the true model parameters (Sun et al., 2016)! Rather than a guarantee that the learned model parameters will be close to the true model parameters, we would prefer a guarantee that the learned model will perform well on inference tasks.

Additional insight into this problem can be gained by examining how we learn a model using two stage regression, and what happens when that model is used to perform filtering. Let  $(q_1, \{B_i\}_{i=1}^k, b_\infty)$  be a discrete PSR learned via two stage regression as described in Section 2.3. Let  $o_1, \dots, o_T$  be a sequence of observations and  $q_1, \dots, q_T$  the corresponding sequence of true states. Let  $\hat{q}_t$  be the estimated belief over states at time  $t$  produced by filtering. Given state  $q_t$  we apply the PSR update  $\hat{q}_{t+1} = \frac{B_{o_t} \hat{q}_t}{b_\infty^T B_{o_t} \hat{q}_t}$  to produce  $\hat{q}_{t+1}$ .

It can be seen from Equation (2) that each  $B_i$  is the solution to a regression problem from expected features of the future at time  $t$  to expected features of the future at time  $t + 1$ . In other words we learn a model which makes good one step inference predictions for all states  $q_t$  encountered in the training set. Unfortunately, using (2) to perform inference results in states not present in the training set due to model

estimation error. This causes poor subsequent predictions, as our model was not optimized to make good predictions on states outside of the training set.

We would like to modify the learned PSR so that it makes good predictions on states encountered during inference, not just the states encountered when solving the two-stage regression.

To solve this problem, we turn to PSIM. The PSIM framework specifically trains a model to make good predictions on states encountered during inference. Essentially, PSIM uses an initial model to perform filtering on a sequence of observations to produce a sequence of states, then updates the model so that the states produced during inference are closer (in expectation) to the true states.

One issue with PSIM is that the training data for PSIM is generated from the initial model. If the initial model is poor, the resulting training data will also be poor, and updating the model based on this data is likely to be of little benefit. Adapting PSIM to PSRs allows us to generate a good PSIM initialization using two-stage regression. This offers an alternative viewpoint: that two-stage regression applied to PSR allows us to fully unlock the potential of PSIMs by providing a consistent way to initialize them.

A second issue is that it is not obvious how to apply PSIM to PSRs. In PSRs we use a weird functional form consisting of a linear update followed by a normalization step. This functional form does not fit nicely within the PSIM framework. We show that by using the gradient descent formulation of PSIM it is possible to perform PSIM-style updates on an arbitrary PSR.

We apply PSIM-style updates to a PSR initialized using 2-stage regression. Our proposed updates correspond to gradient descent on the loss function  $L(\{B_i\}_{i=1}^k) = \sum_t \frac{1}{2} \|q_{t+1} - \hat{q}_{t+1}\|_2^2$ . Since  $q_t$  is not observed we replace it with the observed features of future observations  $\psi_t$  as an unbiased estimate of  $q_t$  giving us the loss function.

$$\hat{L}(\{B_i\}_{i=1}^k) = \frac{1}{T} \sum_{t=1}^T \frac{1}{2} \|\psi_{t+1} - \hat{q}_{t+1}\|_2^2 \quad (5)$$

The gradient of this function can be found in Lemma 1. The proof of Lemma 1 can be found in Appendix A.

### Lemma 1.

$$\begin{aligned} \frac{\partial}{\partial B_{o_t}} \|\psi_{t+1} - \hat{q}_{t+1}\|_2^2 &= \frac{\partial}{\partial B_{o_t}} \left\| \psi_{t+1} - \frac{B_{o_t} \hat{q}_t}{b_\infty^T B_{o_t} \hat{q}_t} \right\|_2^2 \\ &= \left( \frac{b_\infty \hat{q}_{t+1}^T - I}{b_\infty^T B_{o_t} \hat{q}_t} \right) (\psi_{t+1} - \hat{q}_{t+1}) \hat{q}_t^T \end{aligned}$$

This gradient combined with 2-stage regression and PSIM style updates gives us the following simple algorithm for learning PSRs:

**Algorithm 2** Learning a PSR with One-Step Inference Gradients

---

**Input:** sequence of observations  $o_{1:T}$ , corresponding set of features of future observations  $\psi_{1:T}$ , learning rate  $\alpha \in \mathbb{R}^+$ , number of iterations  $n \in \mathbb{N}^+$ .

```

 $(q_1, \{B_i\}_{i=1}^k, b_\infty) \leftarrow \text{2S-Regression}(o_{1:T}, \psi_{1:T})$ 
for  $i = 1$  to  $n$  do
     $\hat{q}_1 \leftarrow q_1$ 
    for  $t = 1$  to  $T$  do
         $\hat{q}_{t+1} \leftarrow \frac{B_{o_t} \hat{q}_t}{b_\infty^T B_{o_t} \hat{q}_t}$ 
         $B_{o_t} \leftarrow B_{o_t} - \alpha \left( \frac{b_\infty \hat{q}_{t+1}^T - I}{b_\infty^T B_{o_t} \hat{q}_t} \right) (\psi_{t+1} - \hat{q}_{t+1}) \hat{q}_t^T$ 
    end for
end for

```

---

## 5. Multi-step Inference Gradients

In step  $t$  of Algorithm 2 we take a gradient step such that  $\hat{q}_{t+1}$  is closer to the true state  $\psi_{t+1}$ . However it is important to note that changing  $B_{o_t}$  will also change future predicted states  $\hat{q}_{t+2}, \hat{q}_{t+3}$ , etc. In fact making  $\hat{q}_{t+1}$  closer to  $\psi_{t+1}$  may actually cause future predicted states to be worse. This effect is not taken into account in PSIM, where the objective function being optimized is the mean squared loss of all one-step updates.

This discussion suggests that we should be optimizing each  $B_i$  with respect to its long term effect on inference predictions. With this in mind we extend the results from the previous section from one-step Inference Gradients to  $h$ -step Inference Gradients. In an  $h$ -step inference gradient we take into account the effect of changing the observable operator at the current observation on the error in the  $h$ th future observation.

**Lemma 2.**

$$\begin{aligned} & \frac{\partial}{\partial B_{o_t}} \frac{1}{2} \|\psi_{t+h} - \hat{q}_{t+h}\|_2^2 \\ &= B_{o_{t+1:t+h}}^T \left( \frac{b_\infty \hat{q}_{t+h}^T - I}{b_\infty^T B_{o_{t+1:t+h}} \hat{q}_t} \right) (\psi_{t+h} - \hat{q}_{t+h}) \hat{q}_t^T \end{aligned}$$

where  $B_{o_{t:t+h}}$  is defined as  $B_{o_{t:t+h}} = B_{o_t} \dots B_{o_{t+h}}$ .

The proof of Lemma 2 is a simple application of the chain rule and can be found in appendix B.

We fix a horizon  $H$  and average the  $h$  step gradients for  $h \in \{1, \dots, H\}$  to get the final gradient update, resulting in Algorithm 3.

Note that Multi-step inference gradients are equivalent to Backpropagation Through Time (BPTT) for recurrent neural networks. Indeed, the PSR update Equation 1 defines a special form of a recurrent structure.

**Algorithm 3** Learning a PSR with Multi-Step Inference Gradients

---

**Input:** sequence of observations  $o_{1:T}$ , corresponding set of features of future observations  $\psi_{1:T}$ , learning rate  $\alpha \in \mathbb{R}^+$ , horizon  $H \in \mathbb{N}^+$ , number of iterations  $n \in \mathbb{N}^+$ .

```

 $(q_1, \{B_i\}_{i=1}^k, b_\infty) \leftarrow \text{2S-Regression}(o_{1:T}, \psi_{1:T})$ 
for  $i = 1$  to  $n$  do
     $\hat{q}_1 \leftarrow q_1$ 
    for  $t = 1$  to  $T$  do
         $\Delta \leftarrow 0$ 
        for  $h = 1$  to  $H$  do
             $\hat{q}_{t+h} \leftarrow \frac{B_{o_{t:t+h-1}} \hat{q}_t}{b_\infty^T B_{o_{t:t+h-1}} \hat{q}_t}$ 
             $\Delta \leftarrow \Delta + \frac{\partial}{\partial A} \frac{1}{2} \|\psi_{t+h} - \hat{q}_{t+h}\|_2^2$ 
        end for
         $\hat{q}_{t+1} \leftarrow \frac{B_{o_t} \hat{q}_t}{b_\infty^T B_{o_t} \hat{q}_t}$ 
         $B_{o_t} \leftarrow B_{o_t} - \alpha \Delta$ 
    end for
end for

```

---

## 6. Evaluation Metrics

We evaluate each model using a variety of metrics in order to examine how the model performs on various inference tasks.

As discussed earlier the log likelihood is not well defined for all PSR model parameters, hence we evaluate all models on the *Proxy Negative Log Likelihood (PNLL)* proposed by (Jiang et al., 2016) (see equation 4). A good PSR is one which assigns high PNLL to sequences of observations in the test set. We expect the model of (Jiang et al., 2016) to perform well on PNLL, as their model directly optimizes this function. We note that PNLL is a specific artificial metric defined by (Jiang et al., 2016) for their model, and that while suggestive, does not necessarily imply good predictive performance.

Dynamical systems models such as PSRs are used in practice to predict future observations. These predictions can be one-step predictions (the next observation) or multi-step predictions (an observation at some point in the future). An example of this would be predicting the next character/word in a string of text. An appropriate evaluation metric for this task is the *One-Step Prediction Accuracy (OSPA)*, i.e. the fraction of (1-step) predicted observations which match the true (1-step) observations. A larger OSPA is better.

Finally we are also interested in the quality of a PSR as a model. Our previous metrics have evaluated the quality of our PSR based on its ability to perform tasks for us. However at its core a PSR is model which allows us to transform states into new states, where each state corresponds to observable quantities. As mentioned in section 4 we would

like a model which makes good state updates on all states encountered during inference. Therefore our final evaluation metric will be the *L<sub>2</sub> state error (L<sub>2</sub>SE)* as defined in Equation 5. A smaller L<sub>2</sub>SE is better.

Note that inference can result in pathological states, which cause very large values of the squared loss. In fact it was this exact behavior which motivated this work. To allow us to analyze this behavior we examine both the mean and median of the L<sub>2</sub>SE. The median provides us with a robust estimate of the general trend, while the mean provides us with insight into the number and magnitude of these pathological states.

A good model will perform well on all of these metrics. It is particularly important for our trust in the algorithm that if we optimize our model with respect to one metric that it should perform well on other metrics. Note that Inference Gradients and PSIM optimize with respect to L<sub>2</sub>SE, while Jiang et al. optimize with respect to PNLL.

## 7. Experiment: Synthetic HMM

In our first experiment we compare the performance of Inference Gradients, Multi-step Inference Gradients, 2-stage regression, PSIMs, and the likelihood based gradient approach of (Jiang et al., 2016) on a synthetic HMM. This mirrors an experiment performed in (Jiang et al., 2016).

### 7.1. Dataset

We generate  $10^4$  length 10 observation sequences from a randomly generated ring-topology HMM with 20 states and 20 observations. Each state has at most 2 possible observations, chosen at random. The transition matrix follows a ring topology, where each state can only transition to its two neighbors or to itself. All non-zero entries of the transition matrix, the emission matrix, and the initial state distribution are picked uniformly randomly from  $[0, 1]$  and then normalized. We split the observation sequences equally into training and test data.

### 7.2. Parameters

We use all strings of length 1 and 2 as our features. We evaluate the performance of the learned PSRs on each of the metrics discussed in Section 6. We average results over 100 independent trials.

For all methods we normalize gradients to have an L<sub>1</sub> norm of one, and use a fixed learning rate of  $10^{-3}$  (selected via cross validation). We believe this is the fairest way of comparing the performance of these disparate approaches, as it means the only difference between gradient updates is the direction of the gradient step.

### 7.3. Discussion

Results are presented in Figure 1. We note that the random baseline and 2SR baseline remain constant across all iterations, and hence correspond to flat lines in all figures. Additionally all results are presented in terms of the number of iterations, however in practice Inference Gradients was also several orders of magnitude faster than the approach of Jiang et al. for the same number of iterations, and had comparable speed to PSIM.

- **Proxy Negative Log Likelihood:** Inference Gradients, Multi-step Inference Gradients, and Jiang et al. all decrease the PNLL, while all other approaches increase the PNLL. Jiang et al. decrease the PNLL the most, which is not surprising given that their approach performs gradient descent on a surrogate to the PNLL. It is important to note that Inference Gradients and Multi-step Inference Gradients both decrease the PNLL even though they are performing gradient descent on the L<sub>2</sub>SE. Finally it is clear that the 2SR initialization is extremely important, as both Inference Gradients and Multi-step Inference Gradients increase the negative log likelihood when initialized randomly.

- **One Step Prediction Accuracy:** All approaches improve the OSPA. Multi-step Inference Gradients results in the best OSPA, with Inference Gradients a close second. Note that Multi-step Inference Gradients is outperforming Inference Gradients on OSPA despite being optimized for Multi-step prediction. This suggests that optimizing model parameters based on long term effects is a good strategy. The first few iterations of Jiang et al. result in a small increase in OSPA, however OSPA quickly plateaus, with subsequent iterations failing to further improve OSPA, despite continuing decreases in the PNLL. This result highlights the importance of evaluating the learned PSR according to multiple metrics, and suggests that optimizing the PSR with respect to the negative log likelihood may not be the best approach for optimizing a PSR if the goal is to use that PSR for prediction tasks. Once again we note the importance of the 2SR initialization: the OSPA of randomly initialized Inference Gradients and randomly initialized Multi-step Inference Gradients is significantly lower. Furthermore it appears to plateau, suggesting the possibility of local optima. Even if this is not the case it will take far more iterations to achieve the same level of performance.

- **Median L<sub>2</sub> state error:** The median L<sub>2</sub>SE results mirror the OSPA results, with Multi-step Inference Gradients outperforming all other approaches. This is to be expected as Inference Gradients optimizes the model with respect to L<sub>2</sub>SE.

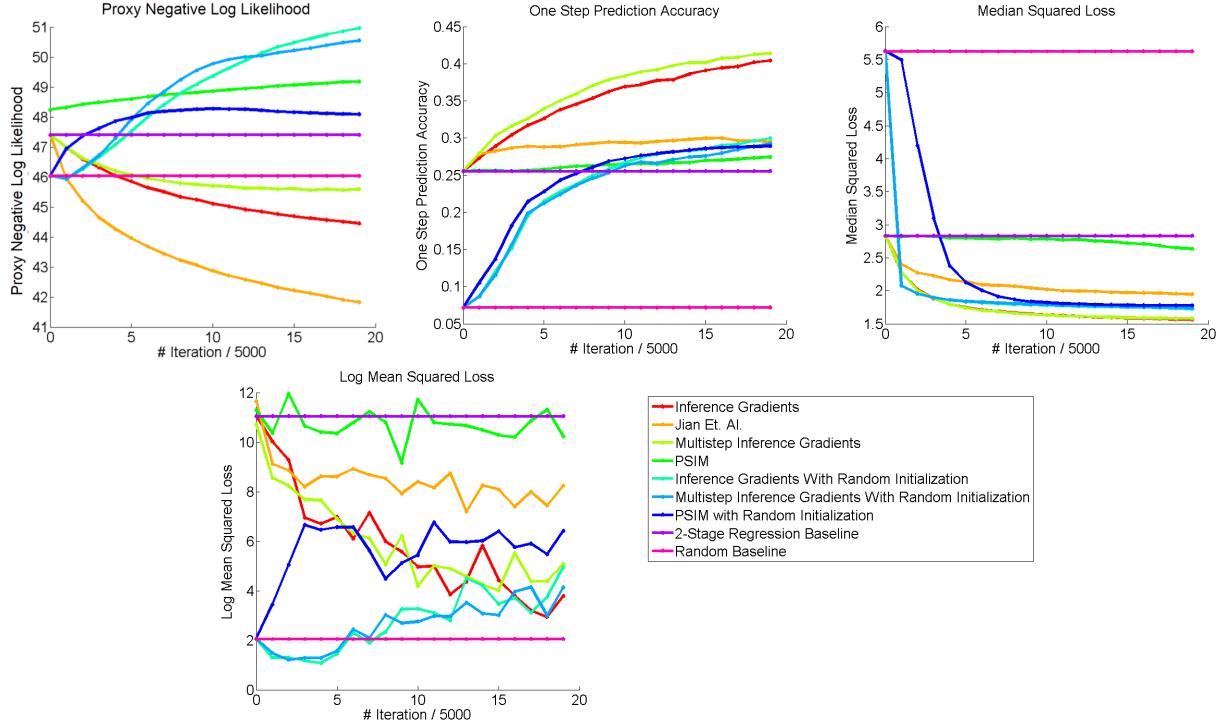


Figure 1. Comparison of various approaches on the ring topology synthetic HMM dataset with respect to the number of iterations.

- **Log Mean L2 state error:** All approaches which were not randomly initialized improve log mean L2SE, while all randomly initialized approaches increase it. One of the original inspiration behind Inference Gradients was to use techniques from imitation learning to improve model predictions on unseen states. We theorized that improving model predictions on unseen states would result in improved performance on other metrics. *This result shows that Inference Gradients is achieving exactly this behavior.* This decrease shows that Inference Gradients decreases the number and magnitude of pathological states encountered during inference. It is interesting to note that randomly initialized approaches increase log mean L2SE while decreasing median L2SE. In other words these methods improve most states, while producing more rogue states.

## 8. Experiment: English Text

In our second experiment we compare the performance of the same set of approaches on an English text dataset. We note that in this dataset OSPA corresponds to the accuracy when the model is used to predict the next character in a text string. This experiment also mirrors one performed by Jiang et al. (Jiang et al., 2016).

### 8.1. Dataset

We sample random length  $10^5$  excerpts from the Penn Tree Bank (PTB) split evenly into training and test data.

### 8.2. Parameters

We use all strings of length 1 and 2 as our features. We evaluate the performance of the learned PSRs on each of the metrics discussed in Section 6. We average results over 100 independent trials. For all methods we normalize gradients to have an L1 norm of one, and use a fixed learning rate of  $10^{-3}$  (selected via cross validation).

### 8.3. Discussion

Results are shown in Figure 2. These results match those from the previous experiment. The only (minor) difference is that Multi-step Inference Gradients gives a larger performance improvement over Inference Gradients on this, more complex, dataset.

<sup>2</sup>Cross validating each method independently resulted in similar optimal learning rates for all methods, hence for simplicity we fixed the learning rate to be constant for all methods. We note that this offers the additional benefit of allowing us to compare performance based only on the direction of the gradient.

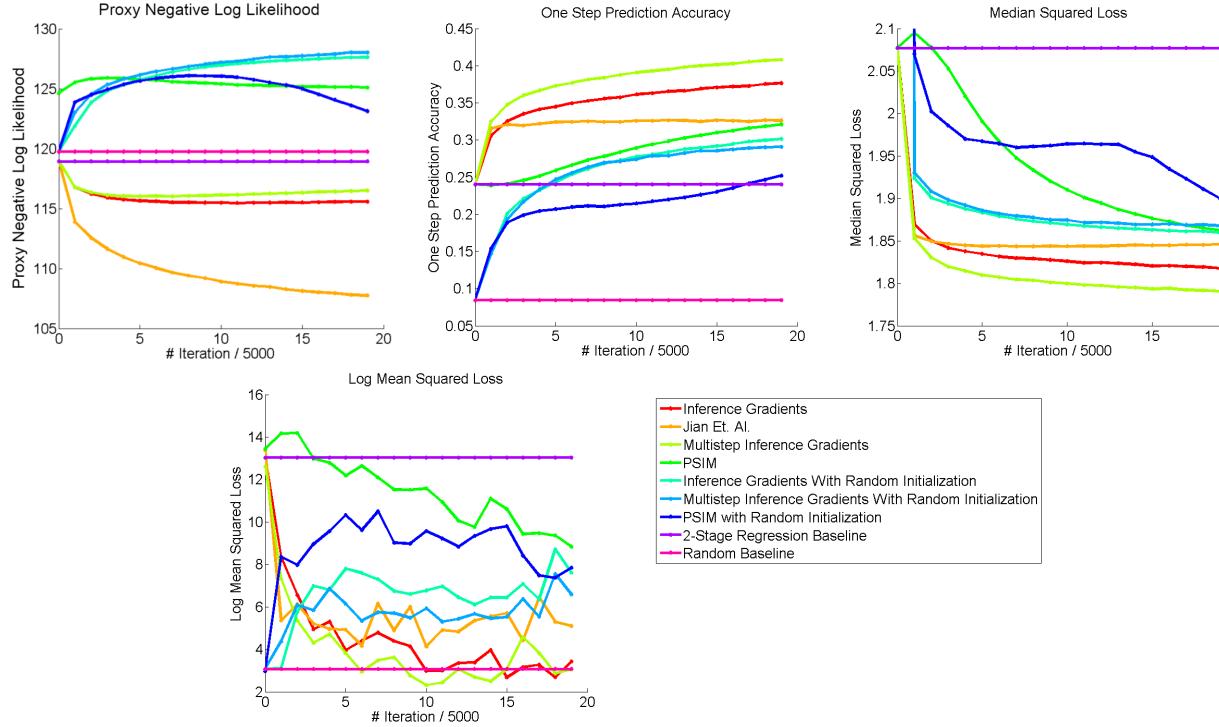


Figure 2. Comparison of various approaches on an english text dataset with respect to the number of iterations.

## 9. Conclusions

We presented Inference Gradients, a simple, fast, and robust method for practical learning of PSRs.

Spectral algorithms for learning PSRs are single pass algorithms. This means that they are not designed to minimize errors resulting from recursive operations such as the state updates in filtering and prediction. Therefore the PSR often makes poor predictions on the additional states encountered during inference. We present a simple algorithm which refines the PSR to make good state predictions from all states encountered during inference on test data. Our approach consists of simple gradient updates inspired by ideas from imitation learning.

This approach provides us with a method to refine the model parameters of a PSR which has been initialized using a spectral algorithm. This refinement optimizes predictive performance and minimizes the predictive error without resorting to approximations to the Likelihood. We show that our approach outperforms the current state-of-the-art gradient descent algorithm for PSRs on several metrics.

## References

Boots, Byron and Gordon, Geoffrey J. Predictive state temporal difference learning. *CoRR*, abs/1011.0041, 2010.

Boots, Byron, Siddiqi, Sajid M., and Gordon, Geoffrey J. Closing the learning-planning loop with predictive state representations. *CoRR*, abs/0912.2385, 2009.

Hefny, Ahmed, Downey, Carlton, and Gordon, Geoffrey J. Supervised learning for dynamical system learning. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 1963–1971, 2015.

Jaeger, Herbert. Observable operator models for discrete stochastic time series. *Neural Computation*, 12:1371–1398, 1999.

Jiang, Nan, Kulesza, Alex, and Singh, Satinder P. Improving predictive state representations via gradient descent. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pp. 1709–1715, 2016.

Langford, John, Salakhutdinov, Ruslan, and Zhang, Tong. Learning nonlinear dynamic models. *CoRR*, abs/0905.3369, 2009.

Littman, Michael L., Sutton, Richard S., and Singh, Satinder. Predictive representations of state. In *In Advances In Neural Information Processing Systems 14*, pp. 1555–1561. MIT Press, 2001.

Ross, Stéphane, Munoz, Daniel, Hebert, Martial, and Bagnell, J. Andrew. Learning message-passing inference machines for structured prediction. In *The 24th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, Colorado Springs, CO, USA, 20-25 June 2011*, pp. 2737–2744, 2011. doi: 10.1109/CVPR.2011.5995724.

Shaban, Amirreza, Farajtabar, Mehrdad, Xie, Bo, Song, Le, and Boots, Byron. Learning latent variable models by improving spectral solutions with exterior point methods. In *Proceedings of The International Conference on Uncertainty in Artificial Intelligence (UAI-2015)*, 2015.

Singh, Satinder, James, Michael R., and Rudary, Matthew R. Predictive state representations: A new theory for modeling dynamical systems. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence, UAI '04*, pp. 512–519, Arlington, Virginia, United States, 2004. AUAI Press. ISBN 0-9749039-0-6.

Sun, Wen, Venkatraman, Arun, Boots, Byron, and Bagnell, J. Andrew. Learning to filter with predictive state inference machines. In *Proceedings of the 2016 International Conference on Machine Learning (ICML-2016)*, 2016.

Venkatraman, Arun, Sun, Wen, Hebert , Martial, Bagnell, J. Andrew (Drew), and Boots, Byron. Online instrumental variable regression with applications to online linear system identification. In *Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, February 2016.

Wu, C. F. Jeff. On the convergence properties of the em algorithm. *Ann. Statist.*, 11(1):95–103, 03 1983. doi: 10.1214/aos/1176346060.

## A. Proof of Lemma 1

$$\frac{\partial}{\partial A} \frac{1}{2} \left\| y - \frac{Ax}{z^T Ax} \right\|_2^2 = \left( \frac{zx^T A^T - (z^T Ax)I}{(z^T Ax)^2} \right) (y - \hat{y})x^T$$

*Proof.*

$$\begin{aligned}
 & \frac{\partial}{\partial A_{ij}} \frac{1}{2} \left\| y - \frac{Ax}{z^T Ax} \right\|_2^2 \\
 &= \frac{\partial}{\partial A_{ij}} \frac{1}{2} \sum_k \left( y_k - \frac{\sum_b A_{kb}x_b}{\sum_{kj} z_k A_{kj}x_j} \right)^2 \\
 &= \sum_k \left( \frac{\partial}{\partial A_{ij}} \frac{-\sum_b A_{kb}x_b}{\sum_{ab} z_a A_{ab}x_b} \right) \left( y_k - \frac{\sum_b A_{kb}x_b}{\sum_{ab} z_a A_{ab}x_b} \right) \\
 &= \sum_{k \neq i} \left( \frac{\partial}{\partial A_{ij}} \frac{-\sum_b A_{kb}x_b}{\sum_{ab} z_a A_{ab}x_b} \right) \left( y_k - \frac{\sum_b A_{kb}x_b}{\sum_{ab} z_a A_{ab}x_b} \right) + \left( \frac{\partial}{\partial A_{ij}} \frac{-\sum_b A_{ib}x_b}{\sum_{ab} z_a A_{ab}x_b} \right) \left( y_i - \frac{\sum_b A_{ib}x_b}{\sum_{ab} z_a A_{ab}x_b} \right) \\
 &= \sum_{k \neq i} \left( \frac{z_i x_j \sum_b A_{kb}x_b}{(z^T Ax)^2} \right) \left( y_k - \frac{\sum_b A_{kb}x_b}{z^T Ax} \right) + \left( \frac{-x_j z^T Ax + z_i x_j \sum_b A_{ib}x_b}{(z^T Ax)^2} \right) \left( y_i - \frac{\sum_b A_{ib}x_b}{z^T Ax} \right) \\
 &= \sum_k \left( \frac{z_i x_j \sum_b A_{kb}x_b}{(z^T Ax)^2} \right) \left( y_k - \frac{\sum_b A_{kb}x_b}{z^T Ax} \right) + \left( \frac{-x_j z^T Ax}{(z^T Ax)^2} \right) \left( y_i - \frac{\sum_b A_{ib}x_b}{z^T Ax} \right) \\
 &= z_i \left( \frac{\sum_b x_b \sum_k A_{kb}(y_k - \hat{y}_k)}{(z^T Ax)^2} \right) x_j - \left( \frac{1}{z^T Ax} \right) (y_i - \hat{y}_i) x_j \\
 &= z_i \left( \frac{(y - \hat{y})^T Ax}{(z^T Ax)^2} \right) x_j - \left( \frac{1}{z^T Ax} \right) (y_i - \hat{y}_i) x_j \\
 &= \left( \left( \frac{zx^T A^T - (z^T Ax)I}{(z^T Ax)^2} \right) (y - \hat{y})x^T \right)_{ij}
 \end{aligned}$$

□

## B. Proof of Lemma 2

$$\frac{\partial}{\partial A} \frac{1}{2} \left\| y - \frac{A \frac{Bx}{z^T Bx}}{z^T A \frac{Bx}{z^T Bx}} \right\|_2^2 = \frac{\partial}{\partial A} \frac{1}{2} \left\| y - \frac{ABx}{z^T ABx} \right\|_2^2$$

*Proof.*

$$\begin{aligned}
 & \frac{\partial}{\partial A_{ij}} \frac{1}{2} \left\| y - \frac{ABx}{z^T ABx} \right\|_2^2 \\
 &= \frac{\partial}{\partial A_{ij}} \frac{1}{2} \sum_k \left( y_k - \frac{\sum_{ab} B_{ka} A_{ab} x_b}{\sum_{abc} z_a B_{ab} A_{bc} x_c} \right)^2 \\
 &= \sum_k \left( \frac{\partial}{\partial A_{ij}} \frac{-\sum_{ab} B_{ka} A_{ab} x_b}{\sum_{abc} z_a B_{ab} A_{bc} x_c} \right) \left( y_k - \frac{\sum_{ab} B_{ka} A_{ab} x_b}{\sum_{abc} z_a B_{ab} A_{bc} x_c} \right) \\
 &= \sum_k \left( \frac{-B_{ki} x_j (\sum_{abc} z_a B_{ab} A_{bc} x_c) + \sum_{ab} B_{ka} A_{ab} x_b (\sum_a z_a B_{ai} x_j)}{(\sum_{abc} z_a B_{ab} A_{bc} x_c)^2} \right) \left( y_k - \frac{\sum_{ab} B_{ka} A_{ab} x_b}{\sum_{abc} z_a B_{ab} A_{bc} x_c} \right) \\
 &= \sum_k \left( \frac{-B_{ki} x_j (z^T B Ax) + [BAx]_k ([zB]_i x_j)}{(z^T B Ax)^2} \right) \left( y_k - \left[ \frac{BAx}{(z^T B Ax)} \right]_k \right) \\
 &= \left( \frac{[(y - \hat{y})^T B]_i x_j (z^T B Ax) + ((y - \hat{y})^T B Ax) ([zB]_i x_j)}{(z^T B Ax)^2} \right) \\
 &= \left( \frac{B^T z x^T A^T B^T - B^T (z^T B Ax)}{(z^T B Ax)^2} \right) (y - \hat{y}) x^T
 \end{aligned}$$

□