Efficiently Learning Linear-Linear Exponential Family Predictive Representations of State

David Wingate WINGATED@UMICH.EDU
Satinder Singh BAVEJA@UMICH.EDU
Computer Science and Engineering, University of Michigan, 2260 Hayward, Ann Arbor, MI 48109

Abstract

Exponential Family PSR (EFPSR) models capture stochastic dynamical systems by representing state as the parameters of an exponential family distribution over a short-term window of future observations. They are appealing from a learning perspective because they are fully observed (meaning expressions for maximum likelihood do not involve hidden quantities), but are still expressive enough to both capture existing models and predict new models. While maximum-likelihood learning algorithms for EFPSRs exist, they are not computationally feasible. We present a new, computationally efficient, learning algorithm based on an approximate likelihood function. The algorithm can be interpreted as attempting to induce stationary distributions of observations, features and states which match their empirically observed counterparts. The approximate likelihood, and the idea of matching stationary distributions, may apply to other models.

1. Introduction

One of the basic problems in modeling controlled, partially observable, stochastic dynamical systems is representing and tracking state. In a reinforcement learning context, the state of the system is important because it can be used to make predictions about the future, or to control the system optimally. Often, state is viewed as an unobservable, latent variable, but models with *predictive representations of state* (Littman et al., 2002) propose an alternative: PSRs represent state as *statistics about the future*.

Appearing in *Proceedings of the 25th International Conference on Machine Learning*, Helsinki, Finland, 2008.

The original PSR models used the probability of specific, detailed futures called *tests* as the statistics of interest. Recent work has introduced the more general notion of using parameters that model the distribution of length n futures as the statistics of interest (Rudary et al., 2005; Wingate, 2008). To clarify this, consider an agent interacting with the system. It observes a series of observations $o_1...o_t$, which we call a *history* h_t (where subscripts denote time). Given any history, there is some distribution over the next n observations: $p(O_{t+1}...O_{t+n}|h_t) \equiv p(F^n|h_t)$ (where O_{t+i} is the random variable representing an observation i steps in the future, and F^n is a mnemonic for *future*). We emphasize that this distribution directly models observable quantities in the system.

The *Exponential Family PSR* is a new family of models of partially observable, stochastic dynamical systems. EFPSR models assume that the distribution $p(F^n|h_t)$ has an exponential family form, and that the parameters of that distribution are the state of the system (Wingate, 2008). This idea has been shown to unify a number of existing models of dynamical systems: for example, if $p(F^n|h_t)$ is assumed to be Gaussian (and certain other choices are made), the model can capture any domain modeled by a Kalman filter.

Existing algorithms for learning EFPSR models from data are based on maximizing exact likelihood, but the algorithms are slow. This paper presents an efficient algorithm for one particular EFPSR, named the Linear-Linear EFPSR. We begin by presenting an approximate likelihood function, and then show that the terms needed to maximize it can be efficiently computed by virtue of the linearity of the Linear-Linear EFPSR's state update. The resulting algorithm is computationally efficient, and can be interpreted in terms of stationary distributions of observations, features and states. It allows us to begin to learn models of domains which are too large (in terms of the amount of data required, and in terms of the complexity of the observation space) to tackle with any other EFPSR.

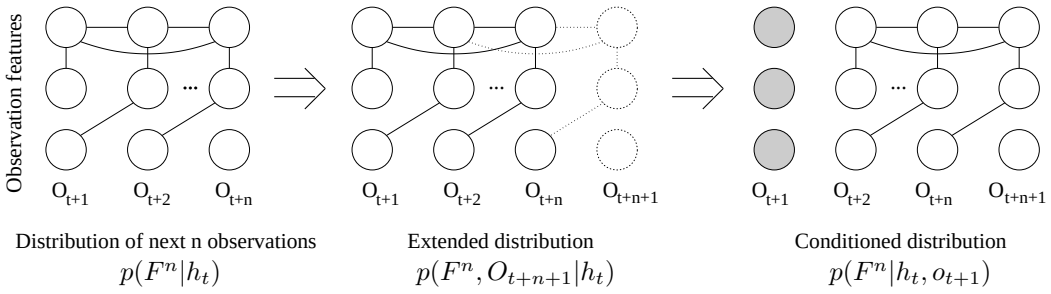

Figure 1. An illustration of extending and conditioning.

2. The Exponential Family PSR

We first review the EFPSR family of models, including how state is represented and how it is maintained.

State. The EFPSR defines state as the parameters of an exponential family distribution modeling $p(F^n|h_t)$, which is a window of n future observations. To emphasize that these parameters represent state, we will refer to them as s_t. The form of the distribution is:

$$p(F^n = f^n|h_t; s_t) = \exp\left\{s_t^\top \phi(f^n) - \log Z(s_t)\right\}, \quad (1)$$

with both $\{\ \phi(f^n),\ s_t\ \} \in \mathbb{R}^{l\times 1}$. The vector $\phi(f^n)$ is a feature vector which controls the particular form of the distribution. For example, $\phi(X) = [X, X^2]$, yields a Gaussian, but $\phi(X) = [X, \log(X)]$ yields a gamma. Since the distribution is over the future, ϕ can be thought of as *features of the future*.

As the agent interacts with the system, $p(F^n|h_t)$ changes because h_t changes; therefore the parameters s_t and hence state change. The feature vector $\phi(f^n)$ does not change over time.

Maintaining State. In addition to selecting the form of $p(F^n|h_t)$, there is a dynamical component: given the parameters of $p(F^n|h_t)$, how can we incorporate a new observation to find the parameters of $p(F^n|h_t, o_{t+1})$? That is, how can we update state? Our strategy is to *extend and condition*.

Extend. We assume that we have the parameters of $p(F^n|h_t)$, denoted s_t. We *extend* the distribution of $F^n|h_t$ to include O_{t+n+1}, which forms a new variable $F^{n+1}|h_t$, and we assume it has the distribution $p(F^n, O_{t+n+1}|h_t) = p(F^{n+1}|h_t)$. This is a temporary distribution over $(n+1)$ observations.

To perform the extension, we define an *extension function* which maps the current state vector to the parameters of the extended distribution:

$$s_t^+ = \text{extend}(s_t; \theta),$$

where θ is a vector of parameters controlling the extension function (and hence, the overall dynamics).

The extension function helps govern the kinds of dynamics that the model can capture. For example, in the PLG family of work, a linear extension allows the model to capture linear dynamics (Rudary et al., 2005), while a non-linear extension allows the model to capture non-linear dynamics (Wingate, 2008).

Condition. Once we have extended the distribution to model the $n+1$'st observation in the future, we then condition on the *actual* observation o_{t+1}, which results in the parameters of $p(F^n|h_{t+1})$:

$$s_{t+1} = \text{condition}(s_t^+, o_{t+1}),$$

which is our state at time $t+1$.

The entire process of extending and conditioning is illustrated in Fig. 1. We have drawn graphs to suggest that there can be structure in the distributions, and to informally hint at the fact that the form of the distribution does not change over time. This, and other constraints on the features and extension function, are discussed in detail elsewhere (Wingate, 2008).

2.1. The Linear-Linear EFPSR

The EFPSR is a family of models. Specific members of the family are chosen by selecting two things: the features ϕ, and an extension function. For example, if $p(F^n|h_t)$ is Gaussian, and a special extension function is chosen, the predictively defined version of a linear dynamical system (Kalman filter) is recovered.

The Linear-Linear EFPSR chooses features and an extension function designed to make it both analytically tractable and efficiently approximable. The extension function is linear, and features are chosen such that conditioning is always a linear operation (hence the name, "Linear-Linear"). In this paper, we also assume the base observations are vectors of binary random variables. If they are not, we assume that binary features are extracted from the observations, and discard the original observations (we call these *atomic* features, to distinguish them from the higher-order features defined by ϕ).

Features. Let each base observation O_t be a vector $\in \{0,1\}^d$; therefore, each $F^n|h_t \in \{0,1\}^{nd}$. We restrict all features comprising the feature vector ϕ to be conjunctions of the atomic binary variables in the base observations. For example, if each $O_t \in \{0,1\}^3$, there could be a feature $\phi(o_t)_k$ which is a conjunction of the second and third components of the observation: $\phi(o_t)_k = (o_t)_2(o_t)_3$. By selecting features this way, the resulting distribution can be conditioned with an operator that is nonlinear in the observation o_{t+1}, but linear in the state s_t. We therefore define the linear conditioning operator $G(o_{t+1})$ to be a matrix which transforms s_t^+ into s_{t+1}: $s_{t+1} = G(o_{t+1})s_t^+$. See (Wingate, 2008) for details.

Extension function. We choose a linear extension:

$$s_t^+ = As_t + B.$$

$A \in \mathbb{R}^{k\times l}$ and $B \in \mathbb{R}^{k\times 1}$ are our model parameters.

The combination of a linear extension and a linear conditioning operator means that the entire extend-and-condition operation (ie, state update) is a linear operation:

$$s_{t+1} = G(o_{t+1})\left(As_t + B\right).$$

This will be critical in the sequel.

3. Learning with Exact Likelihood

We now briefly sketch how to learn a Linear-Linear EFPSR model from data by maximizing exact likelihood. We do this to point out the two primary computational bottlenecks that motivate this paper.

We assume we are given a sequence of T observations, $[o_1 \cdots o_T]$, which we stack to create a sequence of samples from the $F^n|h_t$'s: $f_t|h_t = [o_{t+1} \cdots o_{t+n}|h_t]$. The likelihood of the training data is $p(o_1, o_2 ... o_T) = \prod_{t=1}^T p(o_t|h_t)$, but we will find it more convenient to measure the likelihood of the corresponding f_t's: $p(o_1, o_2 ... o_T) \approx n\prod_{t=1}^T p(f_t|h_t)$ (maximizing this is equivalent to maximizing the standard likelihood).

The expected log-likelihood of the training f_t's under the model defined in Eq. 1 is

$$\mathcal{LL} = \frac{1}{T}\left(\sum_{t=1}^{T} -s_t^\top \phi(f_t) - \log Z(s_t)\right) \qquad (2)$$

Our goal is to maximize this quantity. Any optimization method can be used to maximize the log-likelihood. Two popular choices are gradient ascent and quasi-Newton methods, such as (L-)BFGS, which require the gradient of the likelihood with respect to the parameters, which we will now compute.

We can differentiate with respect to our parameters:

$$\frac{\partial \mathcal{LL}}{\partial \{A, B\}} = \sum_{t=1}^{T} \frac{\partial \mathcal{LL}}{\partial s_t}^\top \frac{\partial s_t}{\partial \theta} \qquad (3)$$

and with respect to each state:

$$\begin{aligned}\frac{\partial \mathcal{LL}}{\partial s_t} &= \frac{\partial}{\partial s_t}\left[-s_t^\top \phi(f_t) - \log Z(s_t)\right] \\ &= \mathrm{E}_{s_t}[\phi(F^n|h_t)] - \phi(f_t) \qquad (4)\end{aligned}$$

where $\mathrm{E}_{s_t}[\phi(F^n|h_t)] \in \mathbb{R}^{l\times 1}$ is the vector of expected sufficient statistics at time t.

The gradient of s_t with respect to A is given by

$$\frac{\partial s_t}{\partial A} = G(o_{t+1})\left(A\frac{\partial s_{t-1}}{\partial A} + s_{t-1}^\top \otimes I\right),$$

where $\otimes$ is the Kronecker product, and I is an identity matrix the same size as A. The gradient of the state with respect to B is

$$\frac{\partial s_t}{\partial B} = G(o_{t+1})\left(A\frac{\partial s_{t-1}}{\partial B} + I\right).$$

Note that the gradients at time t are temporally recursive – they depend all previous gradients.

There are two bottlenecks which motivate this paper:

1. Computing $\mathrm{E}_{s_t}[\phi(F^n|h_t)]$ is a standard inference problem in exponential family models, and is computationally expensive because it scales exponentially with the number of atomic observation variables included in the domain of $p(F^n|h_t)$. Even approximate inference is NP-hard (Dagum & Luby, 1993), and it must be done T times.

2. The gradients are temporally recursive, but can be computed in a single pass through the data. However, the process is expensive. For the discussion, assume that we have l features in $\phi(f_t)$, and that we have k features in the extended distribution. This means that the matrix $A \in \mathbb{R}^{k\times l}$, that the vector $s_t \in \mathbb{R}^l$, and that there are kl total parameters describing A. The term $\partial s_t/\partial A$ is a matrix, with l rows and kl columns. Given $\partial s_{t-1}/\partial A$, part of computing $\partial s_t/\partial A$ involves multiplying $\partial s_t/\partial A$ by A. This is an expensive matrix-matrix multiplication, which scales poorly as the number of features in the model grows, and it must be performed T times to get the true gradient of the likelihood, which scales poorly as the size of the training set grows.

To summarize, the exact learning algorithm does not scale well with either the number of training samples T, the dimension of the observations, the window size n, or the number of features $|\phi|$.

4. Approximate Likelihood

We now turn to the main contribution of this paper. In order to achieve an efficient learning algorithm, we will present an approximate expression for likelihood, named $\widehat{\mathcal{LL}}$, and show that its gradient can be efficiently computed. We will also examine what happens in the limit as $T \to \infty$. The quantity $\widehat{\mathcal{LL}}$ could be used with any model, not just the Linear-Linear EFPSR, but we will show that the Linear-Linear EFPSR allows us to compute the needed terms easily.

We now present our approximate log-likelihood $\widehat{\mathcal{LL}}$, which is an approximate lower bound on the exact likelihood. To begin, we will make one central assumption:

Assumption 4.1. We assume that $\text{Cov}[s_t, \phi(f_t)] = 0$ and that $\text{Cov}[s_t, o_t] = 0$, $\forall t$.

This assumes that the state does not covary with observable quantities. It implies that $\text{E}[s_t^\top \phi(f_t)] = \text{E}[s_t]^\top \text{E}[\phi(f_t)]$, which will be repeatedly used in the following derivation. This is not as severe of an assumption as it may appear to be – in particular, that this does not imply that s_t and $\phi(f_t)$ are independent.

We derive $\widehat{\mathcal{LL}}$ using Assumption 4.1 and a lower bound based on Jensen's inequality:

$$\begin{aligned}
\mathcal{LL} &= \frac{1}{T}\left(\sum_{t=1}^{T} -s_t^\top \phi(f_t) - \log Z(s_t)\right) \\
&= \text{E}_T\left[-s_t^\top \phi(f_t) - \log Z(s_t)\right] \\
&= \text{E}_T\left[-s_t^\top \phi(f_t)\right] - \text{E}_T\left[\log Z(s_t)\right] \\
&\approx \text{E}_T\left[-s_t\right]^\top \text{E}_T\left[\phi(f_t)\right] - \text{E}_T\left[\log Z(s_t)\right] \\
&\geq \text{E}_T\left[-s_t\right]^\top \text{E}_T\left[\phi(f_t)\right] - \log Z(\text{E}_T\left[s_t\right]) \\
&\equiv \widehat{\mathcal{LL}}
\end{aligned}$$

where we have defined the operator

$$\text{E}_T[X] \equiv \frac{1}{T}\sum_{t=1}^{T} X.$$

The fourth line in the derivation follows because of Assumption 4.1. The fifth line is obtained by a double application of Jensen's inequality:

$$\begin{aligned}
\text{E}[-\log Z(s_t)] &= \text{E}\left[-\log(\int \exp(-s_t^\top \phi(F))dF)\right] \\
&\geq -\log(\text{E}\left[\int \exp(-s_t^\top \phi(F))dF\right]) \\
&\geq -\log(\int \exp(\text{E}\left[-s_t^\top \phi(F)\right])dF) \\
&\approx -\log(\int \exp(\text{E}\left[-s_t\right]^\top \text{E}\left[\phi(F)\right])dF) \\
&= -\log Z(\text{E}[s_t]).
\end{aligned}$$

Algorithm 1 LEARN-EFPSRS-W-APPROX-LL

Input: $\text{E}_T[o_t]$, $\text{E}_T[\phi(f_t)]$
Initialize $A = 0$, $B = 0$.
repeat
  $(\widehat{\mathcal{LL}}, \nabla_A \widehat{\mathcal{LL}}, \nabla_B \widehat{\mathcal{LL}})$=GRADS-OF-APPROX-LL($\text{E}_T[o_t], \text{E}_T[\phi(f_t)], A, B$)
  // Use the gradients in an optimizer. Steepest
  // descent would look like this:
  $A = A + \alpha \nabla_A \widehat{\mathcal{LL}}$
  $B = B + \alpha \nabla_B \widehat{\mathcal{LL}}$
until $\widehat{\mathcal{LL}}$ is maximized
Return A, B

The second and third lines follow because of the convexity of the functions $-\log$ and exp, and the fourth line follows by Assumption 4.1.

The approximate log-likelihood involves several new terms, which we now explain. Consider $\text{E}_T[s_t]$. Because this is an unconditional expectation, as $T \to \infty$, this can be interpreted as the stationary distribution of states induced by a particular setting of the parameters of the model.

At first glance, this term would appear to defeat the point of our approximations: it appears to depend on T and on the model parameters, which means that we would have to recompute it, at cost T, every time the parameters change (as they would inside any sort of optimization loop). Fortunately, because it is the stationary distribution of states, it can be efficiently computed in the case of the Linear-Linear EFPSR as the solution to a linear system of equations in a way that does not depend on T.

The other terms have similar interpretations. $\text{E}_T[\phi(f_t)]$ is empirically observed stationary distribution of features of n-step windows of observations. Since it does not depend on the model parameters, it can be computed once at the beginning of learning in a single pass through the data. The quantity $\log Z(\text{E}_T[s_t])$ is the log partition function Z computed using the vector $\text{E}_T[s_t]$, and can be computed in the same way as the partition function associated with any ordinary state s_t.

4.1. Computing the Approximate Likelihood

Can the approximate log-likelihood $\widehat{\mathcal{LL}}$ and its derivatives be computed efficiently? The answer is yes: Appendix A shows that in the case of the Linear-Linear EFPSR, both $\widehat{\mathcal{LL}}$ and the derivative of $\widehat{\mathcal{LL}}$ with respect to the model parameters can be computed efficiently. The computation does not depend on T (the amount

of training data), and only involves the solution to two sparse linear systems of equations. Inference must be performed on the graphical model only once. In addition, the expensive matrix-matrix multiplications are completely eliminated.

4.2. Algorithm Summary

Let us pause for a brief summary. The exact log-likelihood $\mathcal{LL}$ in Eq. 2 is intractable to maximize. However, we have introduced $\widehat{\mathcal{LL}}$, and shown that it and its derivatives can be computed efficiently. Putting everything together, we see that this learning algorithm is attempting:

- to find a setting of the parameters A and B
- which generate a stationary distribution of states $\mathrm{E}_T[s_t]$,
- based on a transition operator defined using the stationary distribution of observations $\mathrm{E}_T[o_t]$,
- which imply a stationary distribution of features of length n trajectories $\mathrm{E}_{\mathrm{E}_T[s_t]}[\phi(F^n|h_t)]$ as close as possible to the empirically observed stationary distribution of features of length n trajectories $\mathrm{E}_T[\phi(f_t)]$.

With gradients in hand, any optimization method may be used to find the optimal settings for A and B. The final gradient algorithm is shown in Algorithm 2 (in Appendix A), and a simple companion steepest descent optimizer is shown in Algorithm 1.

5. A Low-Rank Parameterization

We briefly turn our attention to the parameter matrices A and B. So far, we have implicitly assumed that the matrix A is reasonably sized, but this assumption is false in the case of a large number of features.

To clarify this, recall that our state s_t is a vector $\in \mathbb{R}^{l\times 1}$, where l is the number of features of the future. When we extend and condition, we implicitly compute s_t^+, which is a vector of parameters describing $n+1$ observations: $s_t^+ = As_t + B$. If we assume that there are k extended features, the A matrix is $\in \mathbb{R}^{k\times l}$.

One of the goals of EFPSRs is to be able to use many features in order to capture state. If the number of features l is very large (say, tens of thousands, or even millions), the number of extended features k will be even larger, and the matrix A will be too large to work with. For example, if there are 10,000 features, and if the extended distribution has 15,000 features, the matrix $A \in \mathbb{R}^{15{,}000\times 10{,}000}$, which is simply too large.

$\widehat{\mathcal{LL}}$ has another property which suggests a solution

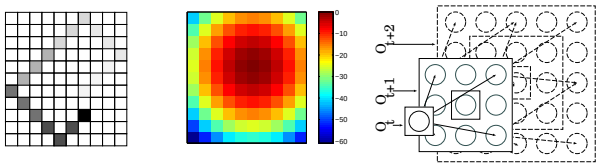

Figure 2. The setup of the bouncing ball problem.

to this problem: the gradients $\nabla_A \widehat{\mathcal{LL}}$ have a natural rank-one form, and therefore mesh well with singular value decomposition (SVD) update algorithms (Brand, 2006). Instead of maintaining the full matrix A, we can maintain a low-rank SVD of A. Given the SVD of A and a rank-one gradient update, the parameters of the updated SVD can be efficiently computed. The entire process can be meshed with a rank-aware line search. The advantage is that the full matrix A is never computed, but exact line searches can be conducted. See (Wingate, 2008) for more details.

6. Experiments and Results

We now evaluate the quality of our approximations. For large problems, we cannot compute the exact likelihoods to compare with, and since we are using approximate likelihoods, it is not clear what a comparison would mean. Instead, we use reinforcement learning to help measure the quality of the model: we use the states generated by the EFPSR as the input to an reinforcement learning planner. We conclude that our model is good if the RL agent is able to use it generate performance comparable to that of the true model. For comparison, we also tested RL using the raw observations as state (called the "reactive," or first-order Markov policy), and a random policy.

6.1. Planning in the EFPSR

We used the Natural Actor Critic (or NAC) algorithm (Peters et al., 2005) to test our model. NAC requires two things: a stochastic, parameterized policy and the gradients of the log probability of that policy. We used a softmax function of a linear projection of the state: the probability of taking action a_i from state s_t given the policy parameters θ is $p(a_i; s_t, \theta) = \exp\left\{s_t^\top \theta_i\right\} / \sum_{j=1}^{|\mathcal{A}|} \exp\left\{s_t^\top \theta_j\right\}$, where θ is to be learned. See (Wingate, 2008) for details.

6.2. Bouncing Ball

The first test domain is called the Bouncing Ball domain. In this domain, the observations are factored in a way that is closely related to the dynamics of the system. This domain was hand-crafted to be compatible with the EFPSR: the domain has significant structure

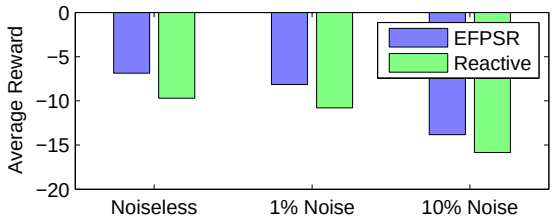

Figure 3. Results on the bouncing ball domain.

in the observations, and basically requires the use of a model which is able to capture that structure.

Figure 2 describes the domain pictorially. The left figure shows the the ball bouncing. At each timestep, the agent observes an 11x10 array of pixels which may be black or white. One of these pixels represents the "ball," which bounces diagonally around the box (shown as a gray trail in the figure). The agent has two actions: 0 means "do nothing," and 1 means "reverse the direction of the ball." The reward signal is shown in the middle. This domain is episodic: every 50 timesteps, the ball is reset to a random position. We define three different versions of the domain. In the noiseless version, the agent sees the exact position of the ball. This domain is second-order Markov with $11 \times 10 = 110$ observations. The second version adds a p=1% chance of flipping white pixels to black. This domain is no longer second-order Markov, and has 2^{110} possible observations. The third version uses p=10%.

Figure 2 shows the features we used, which are hand-coded to correspond with the known dynamics. We set $n = 2$ and added singleton features for each observation. Pairwise features were added for each variable to its diagonal neighbors in the next timestep (to capture the diagonal motion of the ball). The extended distribution $p(F^3|h_t)$ used quartets consisting of an action and observation at time t, and diagonal observations at time $t+1$ and $t+2$. There were 584 features describing $p(F^2|h_t)$ and 1,292 features describing $p(F^3|h_t)$.

We used the timeless gradients, the low rank approximation of A, and 100,000 training samples. Figure 3 collects the results. The EFPSR is able to consistently improve over the best reactive policy, generating a policy with 30% higher reward in the noiseless version, a policy with 25% higher reward when p=1%, and a policy with 13% higher reward when p=10%. It is an open question as to whether different feature sets would improve these results further.

6.3. Robot Vision Domain

Together, the combination of the Linear-Linear EFPSR, the approximate maximum likelihood objective function, and the low-rank decomposition of the parameter matrix allow experimentation on domains with hundreds of observation variables and tens of thousands of features, which is larger than any other model with a predictive representation of state. Here, we apply the entire suite of techniques to the task of visual navigation, where a robot must navigate a maze using only features of camera images as observations.

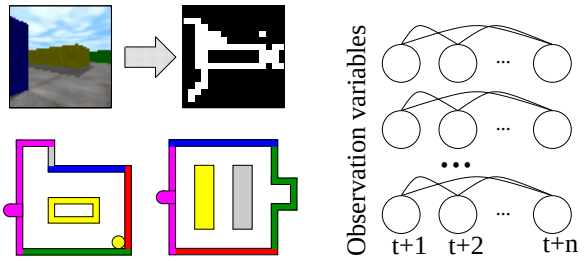

Figure 4. Setup of the vision domain.

Figure 4 explains the setup. The latent state space consists of a position x, y and orientation θ. The experiments used two different maps (bottom left). The agent has four actions: move forward, move backward, turn left and turn right. We tested two kinds of dynamics: in the "coarse" dynamics, the agent took large steps and turns, and in the "fine" dynamics, the agent took small steps and turns. The initial observations are 64x64 color images, from which binary features are extracted (upper left). We tried two different sets of binary features. The first set consisted of 884 features like edges, corners and colors, and the second feature set was a post-processed version of the first. The idea of the second set was to create higher-order features which represented things like walls and hallways. To do this, images from Maze #1 were clustered according to the latent states, and then the binary features were averaged together to create a sort of filter. New images were tested against each filter, triggering if the response exceeded a threshold. There were 373 of these features. Note that while the images were all taken from Maze #1, they were also used in Maze #2, where the colors, hall geometry, etc. were all different.

We set $n = 3$. For the feature vector $\phi()$, we used "streamer features." These connect each observation variable only to its temporal successors (Fig. 4, right). There were between 12,000 and 50,000 total features in the final feature set. We trained on 200,000 samples generated with a random policy. For the NAC parameters, we used a TD rate of $\lambda = 0.85$, a step-size $\alpha = 10.0$, gradient termination test $\epsilon = 0.001$ and remembering factor $\beta = 0.0$.

Figure 5 shows the results. The random policy performed the same in both domains, regardless of map or dynamics. Higher rewards were obtained in general

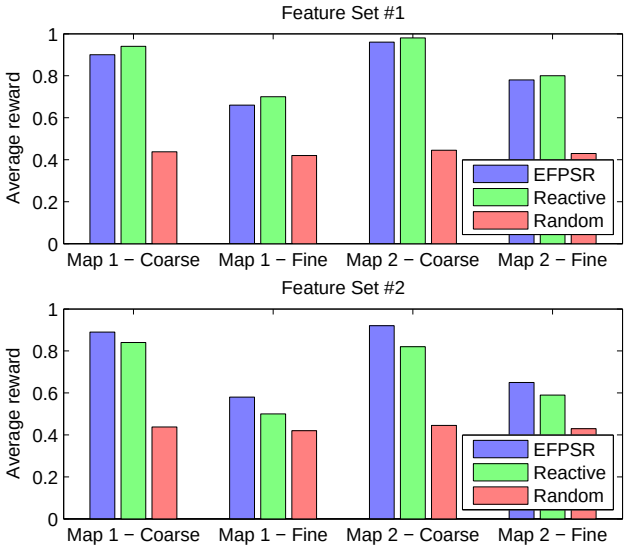

Figure 5. Results on the vision domain.

with coarse dynamics, regardless of map, feature set, or learning algorithm (presumably because the agent can reach high-reward regions more quickly).

The difference between the two feature sets that is most interesting. Using feature set #1, the EFPSR performs just under the performance of the reactive policy, regardless of map or dynamics. Perhaps this means that the EFPSR was unable to capture any meaningful dynamics, and instead learned to predict the identity function, with some noise. This would result in a policy equivalent to the reactive policy.

The results are reversed for feature set #2. Here, the EFPSR consistently outperforms the reactive policy. Together, these observations imply a coherent story. For both feature sets, we used the same set of streamer features. One plausible explanation for the results is that low-order conjunctions of more abstract features gives more modeling benefit than low-order conjunctions of granular, low-level features. It is easy to imagine that low-order conjunctions of granular features is insufficient to capture useful abstract structure in the domain. For example, to represent the corner of a wall, the agent might need a conjunction of 10 features, but we only had fourth order conjunctions. This was part of the motivation for feature set #2: because the camera images were clustered according to latent states, they were typically images of the same thing, from slightly different positions and angles. Using this feature set, the highest-order conjunction was still four or five, but these conjunctions may represent more abstract knowledge: if one feature represents "pink wall" and another represents "pink corner," perhaps a low-order conjunction could express "I'm looking at a pink wall, but if I turn left, I'll see a pink corner." The idea that low-order conjunctions of more abstract features gives more modeling benefit than low-order conjunctions of granular, low-level features suggests several directions for future improvement of these results.

Not reflected in the performance graphs is the computation required. Learning the model was relatively easy, taking only about 30 seconds. Because of the intensive rendering and relatively large size of the domains, the NAC algorithm required about a day to generate the policies to be reported. Informal calculations indicated that it would take about a week to get a single gradient with exact likelihood.

7. Conclusions and Future Work

We have presented a computationally efficient learning algorithm for the Linear-Linear EFPSR model and illustrated it on two domains. Our main contribution is an approximate likelihood, and the insight that maximizing it is equivalent to attempting to match stationary distributions. This idea may find traction in other learning problems. While evaluation of the model and learning algorithm is challenging, it is only by virtue of these approximations that we were able to attempt at all domains like the Bouncing Ball or the Robot Vision domain, which have continuous state spaces, rich observations, and tens of thousands of features. For both domains, we obtained better-than-reactive control policies, suggesting that information from history has successfully been incorporated into the state representation. This is a positive result considering the size of the data set and the number of features involved. Future work needs to address the problem of learning good atomic features and the graphical structure, since these appear to be key factors affecting performance.

A. Computing $\widehat{\mathcal{LL}}$ and Its Derivatives

To compute $\widehat{\mathcal{LL}}$ we must compute three terms: $\mathrm{E}_T[s_t]$ (the stationary distribution of states), $\mathrm{E}_T[\phi(f_t)]$ (which is computed once from data), and the log partition function $\log Z(\mathrm{E}_T[s_t])$. We begin with $\mathrm{E}_T[s_t]$. Recall that our goal is to compute this term in a way that is independent of T. This will be possible using Assumption 4.1, the linearity of the state update, and an insight related to stationary distributions:

$$\begin{aligned}
\mathrm{E}_T[s_t] &= \mathrm{E}_T\left[G(o_t)\left(As_{t-1}+B\right)\right] \\
&\approx \mathrm{E}_T\left[G(o_t)A\right]\mathrm{E}_T\left[s_{t-1}\right] + \mathrm{E}_T\left[G(o_t)\right]B \\
&= \mathrm{E}_T\left[G(o_t)A\right]\mathrm{E}_T\left[s_t\right] + B_G \\
&= G(\mathrm{E}_T[o_t])A\mathrm{E}_T\left[s_t\right] + B_G \\
&= \left(I - G(\mathrm{E}_T[o_t])A\right)^{-1} B_G \qquad (5)
\end{aligned}$$

where I is an appropriately sized identity matrix, and where $B_G = G(\mathrm{E}_T[o_t])B$. The second line follows by Assumption 4.1.

The third and fourth lines are both interesting for different reasons. The fourth line follows by the linearity of the operator $G(\cdot)$. The matrix $G(\mathrm{E}[o_t])$ can be interpreted as the expected transition operator, and is a simple function of the stationary distribution of observations $\mathrm{E}_T[o_t]$. The third line follows by the limiting properties of our expectations: we assume that $\mathrm{E}_T[s_t] = \mathrm{E}_T[s_{t-1}]$ because as $T \to \infty$, both represent the stationary distribution of states.

The result is that $\mathrm{E}_T[s_t]$ can be computed as the solution to a linear system of equations. Note that $G(E_T[o_t])$ will typically be very sparse, and a designer may force the A part to be sparse or low-rank. If so, a matrix-vector product can be computed efficiently, and an iterative solver should be used to solve Eq. 5.

Computing Derivatives. We now compute the derivatives of $\widehat{\mathcal{LL}}$ with respect to A and B:

$$\frac{\partial \widehat{\mathcal{LL}}}{\partial \{A, B\}} = \frac{\partial \widehat{\mathcal{LL}}}{\partial \mathrm{E}_T[s_t]}^\top \frac{\partial \mathrm{E}_T[s_t]}{\partial \{A, B\}}$$

We begin with the left-hand term:

$$\frac{\partial \widehat{\mathcal{LL}}}{\partial \mathrm{E}_T[s_t]} = \mathrm{E}_{\mathrm{E}_T[s_t]}[\phi(F)] - \mathrm{E}_T[\phi(f_t)] \equiv \Delta$$

This result has an appealing intuitive interpretation. $\mathrm{E}_{\mathrm{E}_T[s_t]}[\phi(F)]$ can be interpreted as the expected features that would be obtained if inference were performed using $\mathrm{E}_T[s_t]$ as the state – in other words, it represents the stationary distribution of features under the model. Since $\mathrm{E}_T[\phi(f_t)]$ represents the empirically observed stationary distribution, we see that the gradient wishes to match the two. If we use a variational method to compute the log partition function $\log Z(\mathrm{E}_T[s_t])$, which is needed to determine the value of the log-likelihood, then the expected features $\mathrm{E}_{\mathrm{E}_T[s_t]}[\phi(F)]$ are available as a byproduct of the optimization. This is a pleasing efficiency.

However, we are not done. We still must find the transition parameters which allow us to move the expected sufficient statistics closer:

$$\frac{\partial \mathrm{E}_T[s_t]}{\partial A} = (I - G(\mathrm{E}[o_t])A)^{-1} \left(\frac{\partial}{\partial A} G(\mathrm{E}[o_t])A\right) \mathrm{E}_T[s_t]$$

We now find it convenient to remember that the full derivative also includes the term $\partial \widehat{\mathcal{LL}} / \partial \mathrm{E}_T[s_t] \equiv \Delta$, which is a column vector. Let $\Gamma \equiv \Delta^\top (I - G(\mathrm{E}[o_t])A)^{\top -1} G(\mathrm{E}[o_t])$. Then:

$$\Delta^\top \frac{\partial \mathrm{E}_T[s_t]}{\partial A} = \frac{\partial}{\partial A} \Gamma A \mathrm{E}_T[s_t] = \Gamma^\top \mathrm{E}_T[s_t]^\top$$

Algorithm 2 GRADS-OF-APPROX-LL

Input: $\mathrm{E}_T[o_t]$, $\mathrm{E}_T[\phi(f_t)]$, A, B

// Compute stationary distribution of states
$\mathrm{E}_T[s_t] = (I - G(\mathrm{E}_T[o_t])A)^{-1} B$

// Use $\mathrm{E}_T[s_t]$ to perform inference
Compute $\mathrm{E}_{\mathrm{E}_T[s_t]}[\phi(F)]$ and $\log Z(\mathrm{E}_T[s_t])$

// Compute the approximate log-likelihood:
$\widehat{\mathcal{LL}} = -\mathrm{E}_T[s_t]^\top \mathrm{E}_{\mathrm{E}_T[s_t]}[\phi(F)] - \log Z(\mathrm{E}_T[s_t])$.

// Compute the gradient:
$\Delta = \mathrm{E}[\phi(f_t)] - \mathrm{E}_{\mathrm{E}_T[s_t]}[\phi(F)]$.
$\Gamma = \Delta^\top (I - G(\mathrm{E}[o_t])A)^{\top -1} G(\mathrm{E}[o_t])$
$\nabla_A \widehat{\mathcal{LL}} = \Gamma^\top \mathrm{E}_T[s_t]^\top$ ← note: a rank-one matrix
$\nabla_B \widehat{\mathcal{LL}} = G(\mathrm{E}_T[o_t])^\top \Delta$

Return $\widehat{\mathcal{LL}}, \nabla_A \widehat{\mathcal{LL}}, \nabla_B \widehat{\mathcal{LL}}$

The derivative with respect to B is similar:

$$\begin{aligned} \Delta^\top \frac{\partial \mathrm{E}_T[s_t]}{\partial B} &= \Delta^\top \frac{\partial}{\partial B} [G(\mathrm{E}[o_t])(A\mathrm{E}_T[s_t] + B)] \\ &= \frac{\partial}{\partial B} \Delta^\top G(\mathrm{E}[o_t])B = \Delta^\top G(\mathrm{E}[o_t]) \end{aligned}$$

The completed algorithm is shown in Algorithm 2.

Acknowledgments

Both authors were supported by NSF grant IIS-0413004. Any opinions, findings, and conclusions or recommendations expressed are those of the authors and do not necessarily reflect the views of the NSF.

References

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

Dagum, P., & Luby, M. (1993). Approximate probabilistic reasoning in bayesian belief networks is NP-hard. *Artificial Intelligence*, *60*, 141–153.

Littman, M. L., Sutton, R. S., & Singh, S. (2002). Predictive representations of state. *Neural Information Processing Systems (NIPS)* (pp. 1555–1561).

Peters, J., Vijayakumar, S., & Schaal, S. (2005). Natural actor-critic. *European Conference on Machine Learning (ECML)* (pp. 280–291).

Rudary, M. R., Singh, S., & Wingate, D. (2005). Predictive linear-Gaussian models of stochastic dynamical systems. *Uncertainty in Artificial Intelligence* (pp. 501–508).

Wingate, D. (2008). *Exponential family predictive representations of state*. PhD thesis, University of Michigan.