Spectral learning for structured partially observable environments

Lucas Langer
lucas.langer@mail.mcgill.ca

1. PROBLEM AND MOTIVATION

We consider the problem of learning models of time series data in partially observable environments. Typical applications arise in robotics and reinforcement learning with HMMs and POMDPs being the models of choice. We take interest in environments with structured observations. Standard learning algorithms are not designed to exploit patterns which arise in many practical applications. As a result, we focus on extending a current learning algorithm to exploit such structure. Our approach yields both better predictive accuracy and computational performance when learning smaller models as one does in practice.

2. BACKGROUND AND RELATED WORK

Predictive state representations (PSR) are used as a model for computing a probability distribution over observations in a dynamical system [**?**]. There exists a well known spectral algorithm which learns a PSR from empirical data [**?**]. The algorithm makes use of Hankel matrices and singular value decomposition. One can control the number of states in the PSR by only including states with high singular values. The reason for using fewer states is twofold. First, noise in empirical data artificially creates extra states with low singular values. Secondly, reducing the number of states is necessary in practice for computational performance.

Learning of PSRs began with work on non-spectral methods [**?**]. Spectral algorithms emerged later and became of particular interest because they delivered theoretical guarantees far better than other methods. [**?**]. On the applied side, spectral learning of PSRs has shown promise in planning with timing information [**?**] and in natural language processing for dependency parsing [**?**].

3. PRELIMINARIES AND NOTATION

3.1 Notation

We now introduce some notation. Let $\sum$ be the alphabet of observations symbols. Then we use $\sum *$ to denote the set of all finite strings which can be made from $\sum$.

A PSR is defined by three parameters: f = $\langle \alpha_0, \{A_S\}, \alpha_\infty \rangle$. Here α_0 is a row vector (1xn), α_∞ is a column vector (nx1), and $A_s, s \in S$ are transition matrices (nxn) between states associated with the observation s. The dimension of these parameters n corresponds to the number of states in the PSR.

3.2 Spectral Learning of PSRs

placeholder

3.3 Computations

Let x $\in \sum *$ be a query string with a decomposition $x = x_1x_2x_3...x_k$ with $x_i \in S$. Then $f(x) = \alpha_0 \cdot A_{x_1} \cdot A_{x_2}...A_{x_k} \cdot \alpha_{\text{inf}}$. Note that multiple decompositions of x may be possible.

The standard way of computing with PSRs has been to include $\{A_s\}$ s $\in \sum$ and decompose a string x character by character.

3.4 Selection of the Base

Given observation sequences one would like to automatically determine which operators to learn. From and intuitive standpoint, operators should cover all frequent and long substrings that occur in observations. Here, we present an iterative greedy algorithm for choosing such operators.

Algorithm 1 Base Selection Algorithm

```
1: procedure BASE SELECTION
2:     BaseSystem ← {s} ∈ ∑
3:     i ← 0
4:     while i < desiredBaseSize do
5:         for each substring s ∈ Observations  do
6:             for each observation in  do
7:             end for
8:
```

4. APPROACH AND UNIQUENESS

In our work, we extend the standard PSR learning algorithm by developing a new machinery for performing queries which we call the Base System. The main idea in the Base System is to include transition operators for sequences of observations in addition to those for single observations. We first apply the Base System to predict the time spent by an agent in a stochastic environment. We then progress to systems

with multiple observations. Finally, we develop two heuristics: one for choosing operators from data, and another for applying operators in an effective order. The former uses an iterative greedy algorithm, while the latter uses a dynamic programming algorithm.

5. RESULTS AND CONTRIBUTIONS

In the experiments that follow, we produce observations by simulating robot motion in stochastic labyrinth environments. The robot explores the labyrinths until it leaves through one of the doors. We compare PSRs learned with the generic algorithm to PSRs learned with different degrees of the Base System. To measure the performance of a PSR, we compare predictions to the actual probability distribution over observations.

5.1 Double Loops

In the first experiment we look at the time spent in double loop labyrinths.

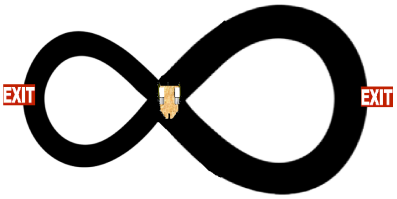

Figure 1: Double Loop Environment

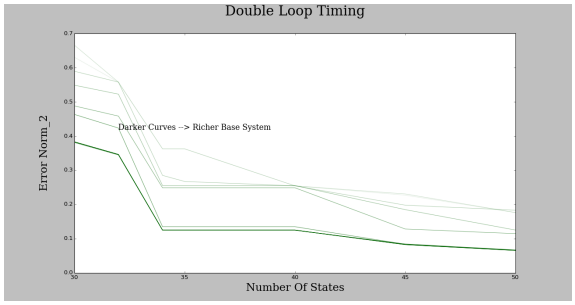

Figure 2: No Noise in Loops

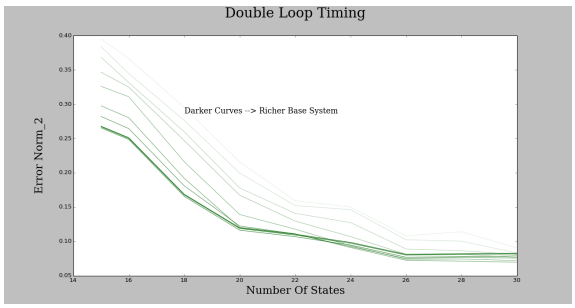

Figure 3: Noise in Loops

The PSR with the Base System significantly lower errors across all model sizes (Figures 2 and 3). In particular, we note that noise in the durations of loops doesn't harm the performance of the Base System.

5.2 PacMan Labyrinth

In the second experiment, we look at timing for a PacMan-Type labyrinth. In addition, we use state weights from the learned PSRs to predict distances between the robot and objects in the environment.

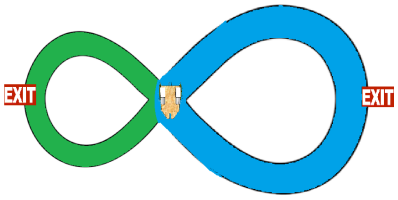

Figure 4: Multiple Observation Environment

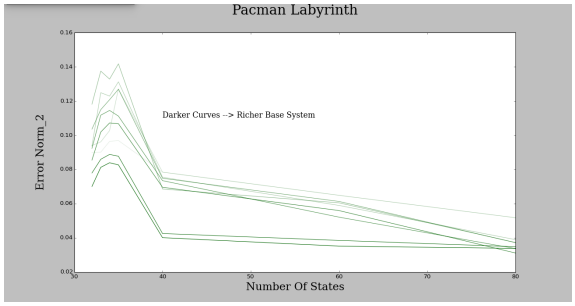

Figure 5: Timing Predictions in Pacman

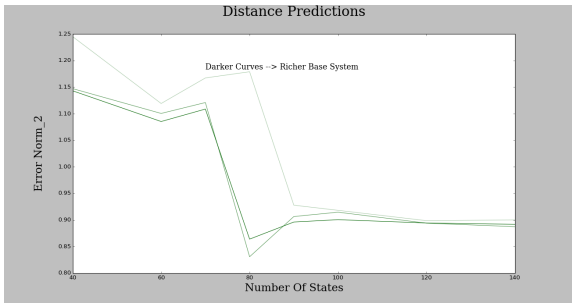

Figure 6: Distance Predictons

The Base System outperforms the naive PSR for the Pacman environment (Figure 5). It also does better for predicting distances (Figure 6).

5.3 Multiple Observations

Next, we change our set of observations to wall colors of the labyrinth.

The Base System also does better for loops with multiple observations (Figure 7).

6. CONCLUSION AND FUTURE WORK

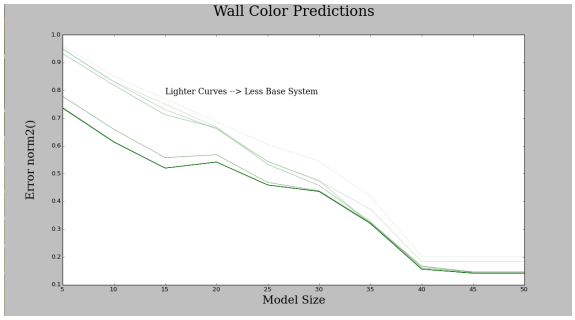

Figure 7: Predicting Wall Colors

In this work, we showed a way to significantly improve performance of truncated models in applied environments. For future work, we leave a theoretical analysis of the Base System and further optimization of its construction .