TITLE GOES HERE

Anonymous Submission

Abstract

Fancy Abstract...

Introduction

Learning models of partially observable dynamical systems is very important in practice — several alternatives...

General algorithms are not designed to exploit frequen patterns/structure in sequences of observations to speed-up learning — but in practice with large observations and highly structured environments this might be necessary to achieve decent results

We propose a new model of predictive state representation for environments with discrete observations: the multi-step PSR (M-PSR)

We show how the standard spectral learning for PSR extends to M-PSR

Then we present a data-driven algorithm for selecting a particular M-PSR from data sampled from a structured partially observable environment

We evaluate the performance of our algorithms in an extensive collection of synthetic environments and conclude that...

M-PSR: Definition and Learning

Multi-Step PSR

A linear *predictive state representation* for an autonomous dynamical system with discrete observations in a set Σ is a tuple $\mathcal{A} = \langle \Sigma, \boldsymbol{\alpha}_\lambda, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}_{\sigma \in \Sigma} \rangle$ where: BLA BLA.

To define our model for multi-step PSR we basically augment a PSR with two extra objects: a set of *multi-step observations* $\Sigma' \subset \Sigma^+$ containing non-empty strings formed by basic observations, and a *coding function* $\kappa : \Sigma^\star \to \Sigma'^\star$ that given a string of basic observations produces an equivalent string composed using multi-step observations. The choice of Σ' and κ can be quite application-dependent, in order to reflect the particular patterns arising from different environments. However, we assume this objects satisfy a basic set of requirements for the sake of simplicity and to avoid degenerate situations:

1. The set Σ' must contain all symbols in Σ; i.e. $\Sigma \subseteq \Sigma'$
2. The function κ satisfies $\partial(\kappa(x)) = x$ for all $x \in \Sigma^\star$, where $\partial : \Sigma'^\star \to \Sigma^\star$ is the *decoding morphism* between free monoids given by $\partial(z) = z \in \Sigma^\star$ for all $z \in \Sigma'$. Note this implies that $\kappa(\epsilon) = \epsilon$, $\kappa(\sigma) = \sigma$ for all $\sigma \in \Sigma$, and κ is injective.

Using these definitions, a *multi-step PSR* (M-PSR) is a tuple $\mathcal{A}' = \langle \Sigma, \Sigma', \kappa, \boldsymbol{\alpha}_\lambda, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}_{\sigma \in \Sigma'} \rangle$.

Examples

Base PSR for Duration Models BLA BLA...

Spectral Learning Algorithm

We extend (Boots, Siddiqi, and Gordon 2011)...

Fully Data-Driven Learning

Experiments

We assess the performance of the Base System on labyrinth environments. The robot is positioned in a starting location and it stochastically navigates the environment. We split the experiments into two cases. The first is the case of timing, where $\sum = \{\sigma\}$ and the second is with multiple observations. For timing, the goal is to make predictions about how long the agent will survive the environment. One can also ask conditional queries such as how long the agent should expect to survive given that x seconds have elapsed. For multiple observations we place the agent in the environment, let it roam for a fixed time length and then remove the agent. The goal for the multiple observation labyrinth will be to make predictions about seeing observation sequences. In both cases, we analyse the performance for PSRs of different model sizes with a fixed observation set.

Double Loop Timing

For timing, we start by considering a double loop environment. The agent starts at the intersection of the two loops. At the intersection, the agent has a 50 percent chance of entering either loop. Exit states are located halfway between each loop. At an exit state, the agent has a 50 percent probability of exiting the environment. Observations are generated by placing the agent in the environment: $\sigma^1 00$ corresponds to the agent surviving 100 time units. In figure 1, we learn a PSRs with 10000 observation sequences. We generate 10 PSRs and average their performance. The error is computed with the following norm:

$||f - \hat{f}|| = \sqrt{\sum_{x \in observations}(f(x) - \hat{f(x)})^2}$.

We use this norm because of a bound presented by [AUTHORS], which states that ().

Pacman Timing

We proceed to work with timing in a Pacman environment. The transition structure of this environment is shown in Figure 2. Again we use 10000 observation sequences per PSR and compute the average of 10 PSRs.

Multiple Observations

To test whether the Base System would translate to multiple observations we construct a double loop environment where one loop is green and the other is blue. The lengths of each loop are also varied. We fix the length of observations to be $loop1 + loop2 * 3$. To build empirical estimates of probabilities we set f(x)=prefix-occc(x)/num-strings-length¿=x.

Conclusion

Acknowledgments

Funding and friends...

References

Boots, B.; Siddiqi, S.; and Gordon, G. 2011. Closing the learning planning loop with predictive state representations. *International Journal of Robotic Research*.