Learning Approximate Predictive Representations

Monica Dinculescu MDINCU@CS.MCGILL.CA
Reasoning and Learning Lab, McGill University, Montréal, Québec, Canada

Reasoning about the consequences of actions in a stochastic, partially observable domain is a key desirable feature of intelligent agents. In environments with many observations and actions building a fully detailed model of the world may not be possible and may require too much data. Instead, an agent may want to only answer a restricted set of questions, i.e. make a limited number of predictions about the future. In this abstract we propose an approach for automatically learning such an approximate model of a partially observable environment from data, with respect to a set of predictions of interest.

Let $\mathcal{A}$ be a finite set of actions, and $\mathcal{O}$ a finite set of observations, such that at each time step i, the agent takes an action $a_i \in \mathcal{A}$ and the environment produces an observation $o_i \in \mathcal{O}$. We define a *history* at time t to be a sequence of actions and observations, $h = a_0 o_0 a_1 o_1 \dots a_t o_t$. We call a future sequence starting at time $t+1$ a test. A *prediction* is the probability that, the set of observations of the test occurs, if the set of actions is executed. Predictions of tests can be thought of as questions the agent wants to answer about the world.

1. Summarizing the data

We now consider a specific way to restrict the set of tests of interest by defining a mechanism called a *probe* to filter the data available to the agent. Intuitively, filtering the space of future tests collapses the space of predictions the agent is interested in. Further, filtering histories extracts predictive information from the time series. We define a probe f on tests as a mapping on observation sequences, $f : \mathcal{O}^* \to \mathbb{R}$, with the following restriction: given observation sequences ω_1, ω_2, and an observation $o \in \mathcal{O}$, then $f(\omega_1) = f(\omega_2) \Rightarrow f(\omega_1 o) = f(\omega_2 o)$. The intuition is that if an agent cannot distinguish two tests at time step t, any future observations will not change this. Similarly, we can define a probe on histories as a mapping on action-observation sequences, $g : (\mathcal{A} \times \mathcal{O})^* \to \mathbb{R}$. Although many functions could be considered as potential history probes, only some would be useful. In particular, we would like to find probes that maintain enough information to predict well the outcomes of the test probes (i.e., the class of tests in which we are interested). A good heuristic for aggregating histories that hold the same amount of predictive information is based on the eligibility trace notion found in reinforcement learning. Eligibility traces are a means of temporarily assigning credit for the occurrence of an event (in our case, the occurrence of observations in our set of interest). For example, given a history $h = a_0 o_0 \dots a_\tau o_\tau \dots a_{\tau+k} o_{\tau+k}$, let g be a probe that assigns r_1 for the occurence of an observation o, and r_2 otherwise. Assume τ to be the last time of occurence of o. Then, using a decay factor γ, $g(h) = \gamma^k r_1 + \gamma^{k-1} r_2 + \dots \gamma^0 r_2$.

2. Learning a predictive history model

We can now build a predictive machine, whose states are summaries of observed histories, and observations are predictions of tests of interest. The approach, presented in Algorithm 1, is based on three successive clustering processes. First, the set of observed tests is filtered using the test probe, forming the class of tests of interest. Secondly, the f probe can be used to filter the set of observed histories, forming classes of histories that are indistinguishable to the agent, i.e. they predict all tests with the same accuracy. The granularity of the clustering is variable, and allows for approximate constructions. This clustering gives the states (summaries of histories) and observations (predictions of tests) of the machine. Once these are formed, a third layer of clustering is performed, this time using the g probe on histories. The idea here is to further minimize the size of the final model, while maintaining its predictive power. Finally, deterministic transitions are added in a natural way: given a history h we can obtain its successor states by concatenating all possible action-observation pairs. It is also interesting to note that f and g can be applied to the space of histories in any order, with considerably different results. The probe on tests is very specific and easy to construct, if the set of observations of interest is known. In most AI environments, this could be the observation denoting a goal state, for example. However, the

Algorithm 1 Learning the history model

Input: history test pairs: (h, t); probes: f, g; error threshold: ϵ_f, ϵ_g
`Compute probed predictions` $\langle h|t\rangle_f$
$T' =$ `cluster tests using` f:
(i.e. $t_1 \sim_f t_2$ if $\forall h$ histories, $|\langle h|t_1\rangle_f - \langle h|t_2\rangle_f| \leq \epsilon_f$)
$H' =$ `cluster histories using` f.
(i.e. $h_1 \sim_f h_2$ if $\forall t$ tests, $|\langle h_1|t\rangle_f - \langle h_2|t\rangle_f| \leq \epsilon_f$)
$H' =$ `cluster histories using` g.
(i.e. $h_1 \sim_g h_2$ if $|g(h_1) - g(h_2)| \leq \epsilon_f$)
`Create model`

probe on histories is entirely given by heuristics and an insight in the domain. Thus, applying it before the probe on tests can significantly affect the correctness of the machine.

3. Experiment

We illustrate our results on a small probabilistic domain, shown in Figure 1. The agent transitions from state i to state $i+1$ with probability p, and to state $i+1$ with probability $1-p$, where $p = 0.7$. There are two deterministic observations, dark(D), and light(L). We imagine that the agent is interested in tests distinguishing the light observation, and thus both the f and g probes collapse the space of tests and histories with respect to this observation.

Figure 1. Tunnel Example

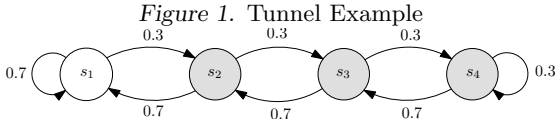
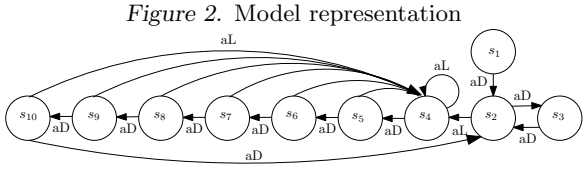


Figure 2 contains the final model representation learned by the algorithm on a particular run, where $s1 = aD$, $s2 = aDaD$, $s3 = aDaDaD$, $s4 = aDaDaL$, $s6 = aDaDaLaD$, ..., $s10 = aDaDaLaDaDaDaDaD$. The size of the machine greatly depends on the choice of ϵ_f and ϵ_g parameters. As these values increase, the number of states in the machine decreases to a minimum of 4, corresponding exactly to the 4 states in the underlying model, which is already minimal (i.e. for each of the 4 states there exist a test whose prediction distinguishes it from the other 3).

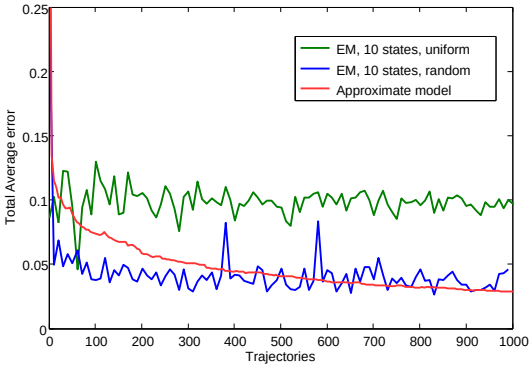
Finally, in Figure 3 we present the total average prediction error over all pairs of histories and tests, averaged over 10 trials. We compare the error obtained from our algorithm with that obtained by two models created using Expectation Maximization(EM) on a model with 10 states: one in which the parameters are

Figure 2. Model representation



initialized to uniform values, and the second in which they are initialized to random values. Note that the initial values of the algorithm greatly affect its accuracy, the former initialization performing significantly worse. However, it can be seen than both EM models reach local optima, and do not improve as more data is seen.

Figure 3. Error comparison



References

Hundt, C., Panangaden, P., Pineau, J., & Precup, D. (2006). Representing systems with hidden state. *AAAI*.

Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1995). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, *101*, 99–134.

Littman, M. L., Sutton, R. S., & Singh, S. (2001). Predictive representations of state. *In Advances In Neural Information Processing Systems 14* (pp. 1555–1561). MIT Press.

Singh, S., Littman, M. L., Jong, N. K., Pardoe, D., & Stone, P. (2003). Learning predictive state representations. *In The Twentieth International Conference on Machine Learning (ICML-2003* (pp. 712–719). AAAI Press.