Approximate Predictive Representations of Partially Observable Systems

Monica Dinculescu MDINCULESCU@GMAIL.COM
McGill University, School of Computer Science, 3480 University st. Montreal, QC, H3A2A7, CANADA

Doina Precup DPRECUP@CS.MCGILL.CA
McGill University, School of Computer Science, 3480 University st. Montreal, QC, H3A2A7, CANADA

Abstract

We provide a novel view of learning an approximate model of a partially observable environment from data and present a simple implementation of the idea. The learned model abstracts away unnecessary details of the agent's experience and focuses only on making certain predictions of interest. We illustrate our approach in small computational examples, demonstrating the data efficiency of the algorithm.

1. Introduction

Reasoning about the consequences of actions in a stochastic, partially observable domain is a key desirable feature of intelligent agents. Traditionally, the framework of choice for modeling this process has been provided by Partially Observable Markov Decision Processes (POMDPs) (see e.g., Kaelbling et al., 1998). It is well understood how to plan good courses of action in a POMDP, if the model of the environment is given. If the model is not known, the traditional solution is to use Expectation Maximization (EM) to acquire it from data. This approach has been demonstrated in several practical applications (e.g. Shatkay & Kaelbling, 1997). However, the empirical evidence to date suggests that this approach works well mainly if one starts with a good initial model. If the initial model is imprecise, EM typically ends up in a bad (but locally optimal) solution.

Recent work on predictive state representations (PSRs) (Littman et al., 2002) is aimed at addressing this problem. Their proposed representation is based on predicting the conditional probability of future observation sequences, conditioned on future sequences of actions and on the past history. Because there is no hidden (or latent) state in the model, in principle, such a representation should be easier to learn from data. Linear PSRs, which have been explored most, are an exact model of the system in the sense that they can predict the probability of any future, given any history. Similar exact models have been provided in other work (e.g., Rivest & Schapire, 1995; Rosencrantz et al., 2004; Hundt et al. (2006)). At the heart of the linear PSR construction lies the computation of a "system dynamics" matrix (Singh et al., 2004), which contains conditional probabilities of future action-observation sequences (called *tests*) given past sequences (i.e. *histories*). Then, one needs to determine the linearly independent columns of this matrix. However, this operation is not numerically stable, so it only works well if a lot of data has been accumulated, and the estimates of the probabilities are very precise. In environments with many observations and actions, computing all these probabilities is too expensive. This has lead to a wave of recent work on learning approximate predictive representations (which we review in detail in Sec. 8). The common thread in all this work is the idea that one has to give up building an exact model, which is able to predict all possible future behaviors, and replace it with a more modest goal of predicting only specific quantities of interest. The approaches differ greatly in terms of the computational mechanisms involved.

In this paper, we attempt to provide a unified way of thinking about this problem. There are two key ideas. First, the only goal of the model is to maintain only particular types of predictions (e.g., predictions about specific observations, about the total rewards that the agent might obtain etc.). One might think of this model as *incomplete*, in the sense that it can only make certain predictions, not all predictions. Second, the history of the agent has to be summarized in such a way as to make these (restricted) predictions possible. We take the view that the predictions of the model will be maintained via a *regression* process, which maps a history representation to a prediction value for the quantity of interest. Our goal is to find automatically a good representation of the history. We propose a very simple approach, inspired by the mechanism of eligibility traces routinely used in reinforcement learning (Sutton & Barto, 1998). We show through small computational experiments that this approach is capable of building good predictive models from small amounts of experience.

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010.

The paper is structured as follows. Sec. 2 presents background and notation. Sec. 3 describes our formalization for building an incomplete predictive model. Sec. 4 presents an approximate representation with guaranteed prediction accuracy. The down side of this representation is that it may be as expensive to compute as a linear PSR. To alleviate this problem, Sec. 5 presents an approach for modeling the history information. Sec. 6 presents a learning algorithm which constructs an approximate state representation from data. Sec. 7 provides experimental results. In Sec. 8 we discuss the relationship of our approach to related work, and in Sec. 9 we conclude and discuss future work.

2. Background

We consider the case of an agent interacting with a dynamical system at discrete time steps, by performing actions from a discrete set $\mathcal{A}$ and receiving observations from a set O. For simplicity, we assume that O is discrete, though our approach can be generalized to continuous observations in a straightforward way. At each time step τ, the agent takes an action $a_\tau \in \mathcal{A}$ and receives an observation $o_\tau \in O$. A *history* is a sequence of past actions and observations; we denote by h_τ the action-observation sequence received up to τ: $h_\tau = a_0 o_0 a_1 o_1 \dots a_\tau o_\tau$.

An action-observation sequence starting at time $\tau+1$ is called a *test*, t_τ (cf. Littman et al., 2002). Given a test $t_\tau = (a_{\tau+1} o_{\tau+1} \dots a_{\tau+k} o_{\tau+k})$, we denote by $\omega(t)$ the sequence of observations of the test, $(o_{\tau+1}, \dots o_{\tau+k})$ and by $\sigma(t)$ the sequence of actions of the test, $(a_{\tau+1}, \dots a_{\tau+k})$. The *prediction* for test t given history h, $p(t|h)$, is defined as the conditional probability that $\omega(t)$ occurs, if the sequence of actions $\sigma(t)$ is executed (Singh et al., 2004):

$$p(t|h) = Pr(\omega(t)|h, \sigma(t)).$$

Let T be the set of all tests and H the set of all histories. Given an ordering (e.g. lexicographic) over H and T, one can define a matrix $\mathcal{D}$, called the *system-dynamics matrix*, in which rows correspond to histories and columns correspond to tests. The entries in the matrix are the predictions of tests, i.e., $\mathcal{D}_{i,j} = p(t_j|h_i)$. Singh et al. (2004) showed that, even though this matrix is infinite, if histories and tests are generated from a POMDP model, the matrix has finite rank. Linear PSR models are based on finding the set of linearly independent columns of this matrix, called core tests, and maintaining their predictions. While this approach is theoretically elegant, it is problematic in practice. If the data is not coming from a Markovian underlying system, for example, or if the underlying system has continuous states, the rank of the matrix may be infinite. Even if the environment is Markovian, but has many observations and actions, building a sufficient portion of the system-dynamics matrix, to a sufficient degree of accuracy, may not be feasible. Instead, to simplify the task, the agent can choose to only answer a restricted set of questions, i.e., make a limited number of predictions about the future. Much of the recent approximate PSR work has been driven by this idea, with the agent's interest being specified in different ways, for example, as subsets of the tests of interest, or linear combinations of tests (e.g. Talvitie & Singh, 2008). We now propose a way to formalize it.

3. Specifying interest

Definition 1. *A* **test probe**, *f, is a mapping, $f : O^* \to \mathbf{R}$. The* **prediction of a test** *t given a history h and a probe f is defined as the expected value of $f(t)$ given h:*

$$p_f(t|h) = p(t|h) f(\omega(t)),$$

Finally, the **prediction for a sequence of actions** *$a_{\tau+1} \dots a_{\tau+k}$ given a history h and a probe f is defined as the expected value of f over all tests t having the action sequence as a skeleton:*

$$p_f(a_{\tau+1} \dots a_{\tau+k}|h) = \sum_{t:\sigma(t)=a_{\tau+1}\dots a_{\tau+k}} p_f(t|h)$$

Note that the probe is defined only on the observation sequence, in order to ensure that the prediction can still be properly conditioned on the sequence of actions for the test, $\sigma(t)$ (and thus independent of the agent's policy).

This is a very general formulation and previous work can be derived as special cases. A full linear PSR will have f equal to 1 for all tests. If the agent is only interested in predicting a linear combination of the form $\sum_i w_i p(t_i)$, we can define $f(t_i) = w_i, \forall i$ and $f(t) = 0$ for all other tests; this corresponds to transformed PSRs (TPSRs), defined by Rosencrantz et al. (2004). If the agent wants to ignore all observations except the last one in a test, f can be defined as 1 for all tests ending in the desired observation and 0 for all tests with the same skeleton but not ending in the desired observation (this approach generalizes in a straightforward way to union tests, cf. Talvitie & Singh, 2008). If the observations consist of a vector and certain components are conditionally independent (cf. Wolfe et al., 2008), then different fs can be used for the parts of the observation vector that can be modeled independently. An important special case is the one in which the observations contain rewards; in this case, f could be defined as the future sum of discounted rewards and the predictions associated with action sequences would be akin to value functions for sequences of actions.

4. Learning predictions of interest

Suppose that instead of containing values $p(t|h)$, the system dynamics matrix would contain values $p_f(t|h)$, where

f is a given probe. If two rows of this matrix, corresponding to histories h_1 and h_2, would be equal, then the corresponding histories yield the same predictions for all tests, and they can be collapsed into one, aggregate class. Similarly, if two tests t_1 and t_2 had the same predictions for all histories, they could be collapsed into an equivalence class. This process gives rise to a smaller matrix.

However, if the matrix is learned from data, exact equality in the expectation estimates is unlikely to happen. Instead, we will allow for small errors in these predictions. We can partition the set of all possible tests T into a set of clusters T', such that for each cluster $T_i \in T'$,

$$\forall t_1, t_2 \in T_i, \forall h \in H, |p_f(t_1|h) - p_f(t_2|h)| \leq \varepsilon_f, \quad (1)$$

where ε_f is a small real value (chosen by the experimenter). Note that the mapping of tests to such clusters is not unique. Since every pair of tests in a cluster is similar, any single test within the cluster can be used to uniquely identify a specific cluster. We associate with each cluster T'_i a *representative test* $t'_i \in T'_i$, which can be chosen arbitrarily (e.g., the lexicographically shortest member). The set T' can then be considered to contain only the representative tests from each cluster. The prediction for a cluster T_i and history h is:

$$p_f(T_i|h) = \frac{1}{|T_i|} \sum_{t \in T_i} p_f(t|h)$$

Note that because of the way in which the clusters are constructed, all elements of a cluster will be within ε_f of this prediction. We define the *probed prediction vector of a history* to be:

$$p_f(T'|h) = [p_f(t'_1|h), p_f(t_2|h), \ldots, p_f(t'_k|h)],$$

where k is the number of clusters in T'.

Similarly, we can partition the set of all possible histories H into a set of clusters H', such that for each cluster $H_i \in H'$

$$\forall h_1, h_2 \in H_i, \forall t \in T, |p_f(t|h_1) - p_f(t|h_2)| \leq \varepsilon_g, \quad (2)$$

where ε_g is a parameter. As before, a *representative history* $h'_i \in H'_i$ can be chosen, with H' being the set of all clusters; but by definition, all histories in a cluster will be within ε_g of this value.

The *probed prediction for a history cluster* H'_i is the average value of the probed prediction vectors of the histories in the class:

$$p_f(T'|H'_i) = \frac{1}{|H'_i|} \sum_{h_i \in H'_i} p_f(T'|h_i).$$

The vector defined above gives the probed predictions for the representative tests in T'. Given a new history that can be mapped to a cluster H'_j, we can get the probed prediction for a test of interest t'_i by looking at the i^{th} column of the vector $p_f(T'|H'_j)$. We use simple averaging but, if the agent's behavior were biased by a particular policy, the averaging could be done based on the data received (so more likely histories would naturally be given more weight). We can collect all the probed predictions for H' in a set denoted $p_f(T'|H')$.

One possibility for obtaining T' and H' is to construct a fragment of the system dynamics matrix, then use Eq. (1) and (2) to collapse its rows and columns. The following theorem gives an error guarantee for this approach.

Theorem 1. *Suppose T' respects Eq. 1 and H' respects Eq. 2. The the maximum prediction error will be at most $2(\varepsilon_f + \varepsilon_g)$.*

Proof: Fig. 1 depicts the main idea of the proof. The large dots represent the histories, and the small dots are the predictions for specific tests. Within each circle, i.e. each cluster of tests, all predictions are ε_f away from the center, and thus the difference in prediction is at most the diameter of the circle, which is $2\varepsilon_f$. Similarly, histories are only assigned to the same cluster H_i if their predictions of the tests of interest are ε_g away from the center. So within each cluster of histories, any two predictions will have a total error of at most $2(\varepsilon_f + \varepsilon_g)\diamond$.

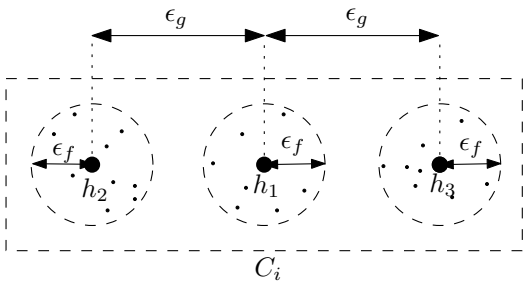

Figure 1. Maximum prediction error

Unfortunately, this approach does not scale well with the number of action-observations pairs; depending on ε_f and ε_g, one may need to build a very large fragment of the system dynamics matrix to get the representation. We now turn our attention to methods which are faster, but may provide rougher approximations to the predictions of interest.

5. History representations

In the rest of the paper, we will assume that f is given and the goal is to learn predictions p_f from data. Given the definition, the simplest approach is to use *discriminative learning*, where the history (or the history-action sequence) is treated as an input and the output to be predicted is p_f. Existing results (Ng & Jordan, 2002) suggest that discriminative learning may have advantages compared to generative

models (like those that might be learned by EM) in terms of the quality of the solution obtained. Intuitively, this should be especially true for temporal predictions, in which small errors in the model estimates may cause predictions to drift considerably for long sequences of observations. With this view, the problem of learning a predictive representation becomes a traditional supervised learning problem that can be solved in a straightforward way *if a mapping of histories H into a finite set of features Φ is given*. However, finding such a good encoding automatically may be very hard (this is akin to the feature construction problem). In the context of learning predictive models, an early attempt at this task is the work on utile distinction memory (McCallum, 1995), which learns an action-value function based on a variable-length history representation. More recently, Wolfe & Barto (2006) used decision trees to learn similar predictions, but in a fully observable environment.

To make this problem computationally tractable, we restrict our attention to mappings in which $\Phi = \mathbb{R}$ (i.e., each history is mapped to a real number). Furthermore, we would like the value of this function to be updated incrementally as new action-observation pairs are received.

Definition 2. *A* **history probe** *is a function $g : H \rightarrow \mathbb{R}$ defined recursively as:*

$$\begin{aligned} g(ao) &= \theta_{ao} \\ g(hao) &= \varphi g(h) + \gamma\theta_{ao}, \end{aligned}$$

where γ, φ and $\theta_{ao}, \forall a \in \mathcal{A}, o \in O$ are parameters.

This particular form of history representation is inspired by eligibility traces; previous work (Loch & Singh, 1998; Bellemare & Precup, 2007) observed that these functions produce good representations for partially observable systems, because they provide implicitly a form of *memory* to remember past events. Note that in an MDP, g would be defined for states and the usual eligibility traces can be represented by setting $\theta = 1$, φ to the product of the discount factor and the eligibility parameter (usually denoted λ) and $\gamma = 1$. However, for a general partially observable environment, we would like all these parameters to be learned from data. We now turn our attention to learning algorithms for such representations.

6. Learning approximate state representations

Given a history probe g, we could create clusters of histories, where histories h_1 and h_2 are mapped together if their value under g is close:

$$|g(h_1) - g(h_2)| \leq \varepsilon_g,$$

where ε_g is a small real value. The history probe value of a cluster is:

$$g(H'_i) = \frac{1}{|H'_i|} \sum_{h \in H'_i} g(h).$$

The tuple $\langle T', H', f, g, p_f(T'|H') \rangle$ forms an approximate system dynamics matrix, which we will call the *approximate agent state representation* (AASR). This representation can be easily updated as the agent collects experience: whenever a history h is observed, the agent computes the associated value $g(h)$ and uses it to attribute the history to an appropriate cluster. Any history h is uniquely mapped to the cluster $i^* = \arg\max_i |g(H'_i) - g(h)|$ that has the closest history probe value. The probed predictions for the tests of interest can be obtained from $p_f(T'|H'_{i^*})$ (defined as before, but for the new history clusters). Furthermore, as the agent observes more data, the clusters (and thus the predictions for the tests of interest) can be updated.

However, if histories are collapsed based solely on the g values, there is no guarantee that histories assigned to the same cluster actually have similar predictions. We would like to *construct a history probe* such that g values are close for histories with similar predictions and very different for histories with different predictions. Thus, we want to learn a parameter configuration $\{\varphi, \gamma, \theta\}$ that minimizes the distance (in terms of the value of the history probe) between histories in the same cluster, while maximizing the distance among all clusters. Formally, we pick θ with

$$\max_{\{\varphi,\gamma,\theta\}} \sum_{i,j} [g(H'_i) - g(H'_j)]^2, \forall H'_i, H'_j \text{ clusters, such that}$$

$$\forall h_i, h_j \in H'_i, |g(h_i) - g(h_j)| \leq \max_{T_i \in T'} |p_f(T_i|h_i) - p_f(T_i|h_j)|$$

Because the distance between histories in the same cluster is bounded above by a prediction error, which is a probability value, we must normalize the values of the history probes by dividing each $g(h)$ by the theoretical maximal value, computed as follows:

$$\begin{aligned} g(a_0 o_0 \ldots a_n o_n) &= \varphi g(a_0 o_0 \ldots a_{n-1} o_{n-1}) + \gamma\theta_n \\ &= \varphi[\varphi g(a_0 o_0 \ldots a_{n-2} o_{n-2}) + \gamma\theta_{n-1}] + \gamma\theta_n \\ &\ldots \\ &= \varphi^n \theta_0 + \gamma \sum_{i=0}^{n-1} \varphi^i \theta_{n-i} \end{aligned}$$

where $\theta_i = \theta_{a_i o_i}$. In the limit, as the length of the history goes to infinity, the first term goes to zero, and using the geometric series, the second term converges to $\gamma \frac{R}{1-\varphi}$, where $R = \max_i \theta_i$ is the highest weight over all observations.

To solve this optimization problem, we begin by constructing a small system-dynamics matrix from the short term

experience (e.g., a couple of steps in the past). Short sequences occur frequently, so the matrix can be computed easily and accurately. Then we determine the exact clusters of histories and tests of interests. This is similar to the clustering procedure described before, with the exception that histories are now collapsed according to their predictions, not according to the history probe. First, the set T' of tests of interest are formed, such that:

$$\forall h |p_f(t_1|h) - p_f(t_2|h)| \leq \varepsilon_f.$$

Then, clusters of histories, H', are formed, such that:

$$\forall T_i, |p_f(T_i|h_1) - p_f(T_i|h_2)| \leq \varepsilon_g$$

where ε_g , ε_f are small real values. Finally, we perform a search in parameter space to solve the optimization problem above. Note that more efficient solution methods can be used here; we use line search just for simplicity. More details on the algorithm (including pseudocode) are available in (Dinculescu, 2010).

The short-term system dynamics matrix, along with the parametrized g function learned from it are the new predictive model, which we call the *local agent state representation* (LASR). Predictions of tests already in the system-dynamics matrix are made as explained above: the history h is mapped to the cluster H_i closest to $g(h)$ and $p_f(t|H_i)$ is read. To make long term predictions, we assume that predictions are *compositional*, i.e.:

$$p_f(t_1 t_2|h) = p_f(t_2|ht_1)$$

The agent is now guaranteed to have errors in prediction - for example, whether or not ht_1 can occur, it will be mapped to an actual cluster and a valid, possibly non-zero prediction will be returned. Note that compositionality can be checked from data and a new representation could be learned if compositionality is violated; however, this goes beyond the scope of this paper and we leave it for future work.

7. Experimental Results

We illustrate our learning algorithm on three small environments.

7.1. Tunnel World

The first environment is a small probabilistic domain, shown in Fig. 2. The agent transitions from state i to state $i-1$ with probability p and to state $i+1$ with probability $1-p$. In our experiments, $p = 0.7$. There are two deterministic observations: dark (D) and light (L). The starting state is always the rightmost state (s_4 in the figure).

The tests of interest are given by a binary test probe: for any test t whose observation sequence contains light, $f(\omega(t)) = 1$; otherwise, $f(\omega(t)) = 0$. The history probe has the same form and it only takes into consideration the observation sequence starting at the last time light was seen.

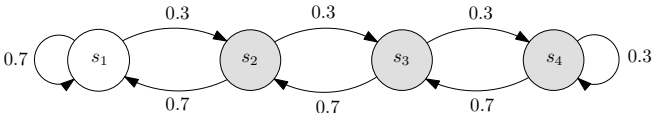

Figure 2. Tunnel World

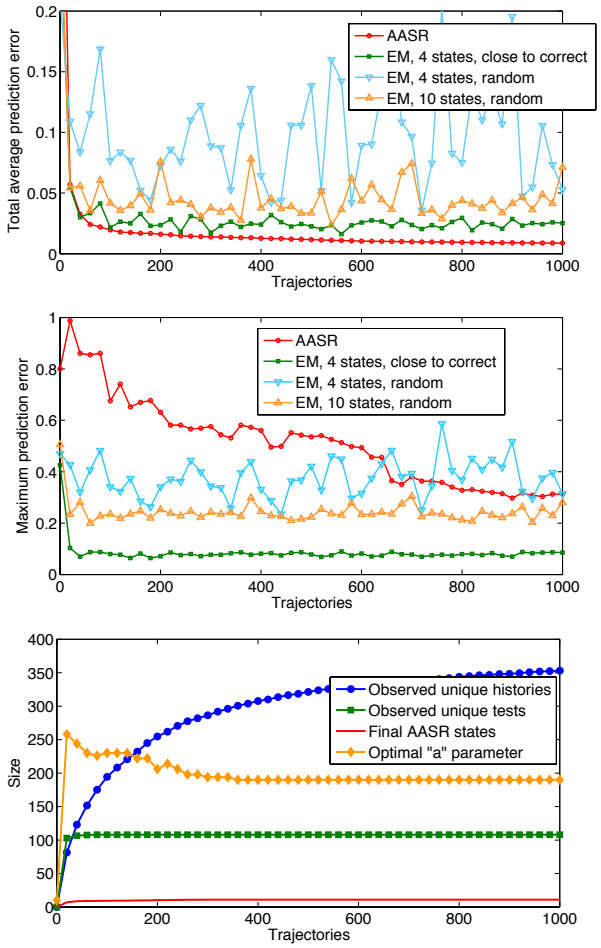

Figure 3. Tunnel World: Average prediction error (top), maximum prediction error (middle) and total number of states (bottom)

The data consists of action-observation trajectories of length 16. From these, we extract the initial $p(t|h)$ predictions through counting, while requiring the histories in the table to be no longer than 10 time steps, and the tests no longer than 6. All results are averaged over 10 independent runs. We used $\gamma = 1$, $\varphi = 0.8$ and $\varepsilon_f = 0.04$. The parameter for the light observation, θ_L, is learned.

The top panel of Fig. 3 presents the total average predic-

tion error (i.e. the difference, over all history and test pairs, between the true predictions and the predictions given by the learned model). The middle panel shows the maximum prediction error over all history-test pairs. We compare these errors with those obtained with four models learned using Expectation Maximization (EM): two models that fit the data to 10 states (the size of the learned model), and two that fit the data to 4 states, the actual number of states in the underlying environment. The initial values of these models is either uniform, random, or in the case of the 4-state model, initialized very closely to the correct values.

We find that the initial values of the EM algorithm greatly affect the accuracy of the model, with all models ending up in local optima. The proposed algorithm performs better overall, even though its maximum prediction is initially higher; this is because with the amount of data seen, the system dynamics matrix entries for rare history-test pairs are inaccurate and they improve with more data. Once a good representation is found, the number of states in the approximate model does not increase as more data is seen, although its accuracy increases. This is because the error comes from the inaccurate $p(t|h)$ estimations and not from incorrect clustering. This result can be seen in the bottom panel of Fig. 3. The states of the final model correspond to histories that determine the position in the world.

7.2. Gridworld

We now consider a larger domain, similar to the one used in Sutton et al. (2006), pictured in the left panel of Fig. 4. Each grid cell has 4 different orientations, so there are 6x6x4 states. There are 2 actions, forward (F) and turn left (L), which changes the orientation without changing the cell. Each action has a 5% probability of failing (in which case the agent remains in the same state). The domain is also partially observable: if the agent is next to a wall and facing it, it will observe the wall's colour, otherwise it will observe white.

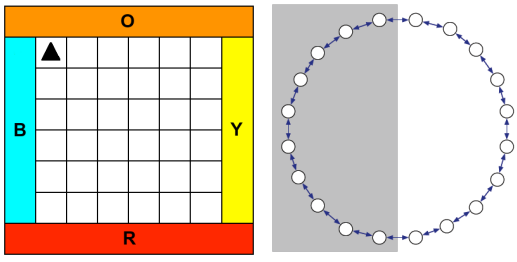

Figure 4. Gridworld environment (left) and half-moon environment (right)

We sample trajectories consisting of 40 transitions in which the action is chosen uniformly randomly (histories of length 30, tests of length 10). The start state of each trajectory is the top left corner, oriented towards the orange wall. The goal of the task is to make predictions about whether the agent will see the orange wall again, given the history so far and a sequence of actions to be taken. $\theta_O = 1$ and $\varepsilon_f = 0.04$. The final model has no more than 50 states, regardless of the complexity and size of the trajectories, as can be seen in the top panel of Fig. 5. The bottom panel contains a plot of the average prediction error for selected history-test pairs that vary in both the length of the predicted tests, as well as the observed history. The results are averaged over 10 runs and contain very tight 95% confidence intervals. As expected, shorter tests have a smaller prediction error, which converges very quickly, implying that a model that predicts immediate futures can be learned from a very small number of samples. Longer tests are seen with much lower frequency; as a result, their initial $p(t|h)$ predictions are inaccurate and they take longer to learn.

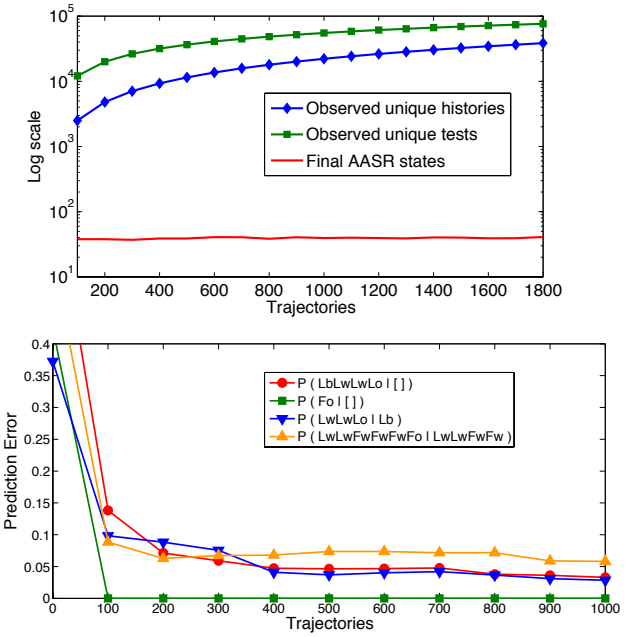

Figure 5. Grid world. Top: number of states in the learned model as the amount of data increases; Bottom; prediction errors for several selected tests

7.3. Half-moon world

We illustrate the benefit of using local models on the Half-Moon world used in (Koop, 2007), depicted in the right panel of Fig. 4. This domain exhibits temporal coherence, i.e. observations are consistent over a short period of time. There are twenty states; in half of them the agent observes black, in the others, white. The agent floats with equal probability between adjacent states. The agent has a very short term memory available: trajectories of length 5, which are split in histories up to length 3 and tests up to

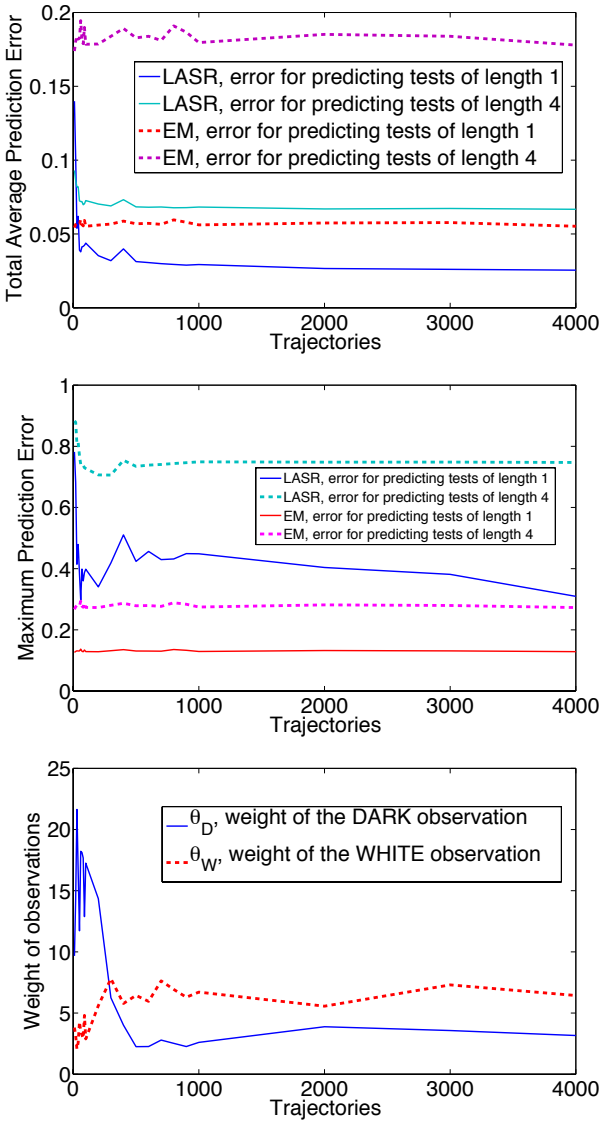

Figure 6. Half Moon world: Maximum error (top panel), average error (middle panel) and parameter values (bottom panel) as a function of amount of data

length 2. The start state is the first white state at the top, and the f probe assigns a value of 1 to any test containing Dark and 0 otherwise, thus answering the question "Will I see Dark in the near future".

Results (averaged over 10 independent runs) are presented in Fig. 6. The top panel compares the error in prediction of the learned LASR with a 20-state model learned with Expectation Maximization (EM), where the initial parameters are initialized very close to the correct ones. We use two different scenarios, one in which the testing data is composed of histories of length 10 and tests of length 1 (thus effectively answering the question "Will I see Dark"

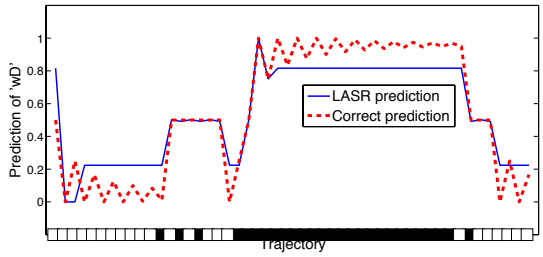

Figure 7. Half Moon world - Trajectory starting from 12:00

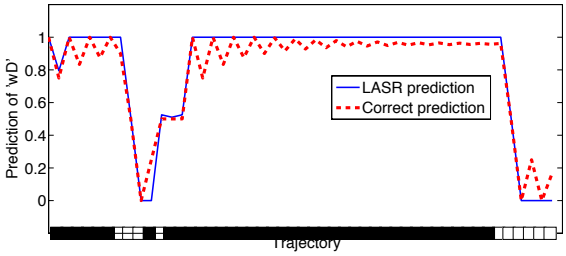

Figure 8. Half Moon world - Trajectory starting from 9:00

in the next step), and a second in which the testing data has tests of length 4. In both scenarios the LASR model has a smaller total average prediction error (over all history-test pairs in the training set) when compared to EM. In the middle panel we present the maximum prediction error of the models. In both scenarios the learned model has a higher maximum prediction error than EM, due to the compositionality assumption. Because the algorithm does not try to learn the dynamics of the system, it will sometimes return non-zero predictions for history-test pairs that do not actually occur in the environment. In the bottom panel we present the change in the two θ parameters over time. It is interesting to note that the local model prefers assigning a higher weight to the White observation (in the history probe), even though it is not the observation for the tests.

In Figs. 7 and 8 we present two sample trajectories, one starting from the top white state (12 o'clock), the other starting from the 15th state (in the dark zone). The horizontal axis shows the observation at each time step, and the vertical axis shows the prediction of seeing dark in the next time step. In both cases, predictions are high quality, albeit with small oscillations.

8. Related work

Some learning algorithms have been proposed for PSRs. McCracken & Bowling (2006) propose an on-line learning method for PSRs. We cannot compare results directly, since their goal is to learn a complete model. Our algorithm has a less ambitious goal (maintaining only predictions of

interest) but as a result can use less data, as shown by comparing the reported amounts of time steps in their paper. James et al. (2005) gradually construct a PSR by adding memory. At any intermediate stage of learning, their representation is approximate, but in the end their goal is to make complete predictions.

TD networks (Sutton & Tanner, 2005) specify a predictive model structure that can include approximations and learn the parameters necessary to make the predictions of interest (as specified by the network). They construct the predictions using bootstrapping from other predictions, while we use Monte-Carlo-style estimation.

Our approach is most similar to the work of Talvitie & Singh (2008), who provide a local, non-parametric approach to modeling sequences of observations. Our work has two differences. First, we use action-observations sequences, rather than just observations. Second, they assume that the form of the models is given, whereas we learn it from data. Wolfe et al. (2008) provide approximate PSR models under the assumption that the observations are multivariate and different parts of the observation vector can be assumed to be conditionally independent. Still (2009) proposes the use of information-theoretic criteria to summarize the system's history in a finite number of states; our optimization problem is different, as we do not require information to be maximized. Our intuition is that the problem we define is easier to solve, because it does not require simulated annealing, but an empirical comparison would be valuable. Talvitie & Singh (2009) learn approximate representations whose goal is to update accurately the values of certain features even in the absence of a PSR model; these features are used for control. In our work, once the tests of interest are specified, all their predictions are maintained continuously.

9. Conclusions and future work

We presented a blueprint for creating approximate representations of partial environments from data and a simple algorithm which makes these ideas concrete. The results are very promising, but more experimentation, in larger domains, is necessary. Better methods could be used to solve the optimization problem that we defined (e.g., the problem could be formulated using L_1 regularization). In more complex tasks, a mapping of histories into reals may not be sufficient, and instead one may need to use a mapping into a feature vector. The methods that we proposed extend in a straightforward way to this case, but the data efficiency remains to be established through experimentation. Finally, we are currently exploring the idea of checking if the compositionality property holds and changing the representation accordingly.

Acknowledgements

This work was supported in part by and NSERC fellowship to Monica Dinculescu and by ONR award no. N000140910462. We thank Jordan Frank, Erik Talvitie and Rich Sutton for very helpful discussions and the anonymous reviewers for their suggestions.

References

Bellemare, M.G. and Precup, D. Context-driven predictions. In *IJCAI*, 2007.

Dinculescu, M. Learning approximate representations of partially observable systems. MSc thesis, McGill University, 2010.

Hundt, C., Panangaden, P., Pineau, J., and Precup, D. Representing systems with hidden state. In *AAAI*, 2006.

James, M. R. and Singh, S. Learning predictive state representations in dynamical systems without reset. In *ICML*, 2005.

Kaelbling, L. P., Littman, M. L., and Cassandra, A. R. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998.

Koop, A. Investigating experience: Temporal coherence and empirical knowledge representation. MSc thesis, University of Alberta, 2007.

Littman, M., Sutton, R.S., and Singh, S. Predictive representations of state. In *NIPS*, 2002.

Loch, J. and Singh, S. Using eligibility traces to find the best memoryless policy in partially observable markov decision processes. In *ICML*, pp. 323–331. Morgan Kaufmann, 1998.

McCallum, A.K. *Reinforcement learning with selective perception and hidden state*. PhD thesis, Rochester University, 19955.

McCracken, P. and Bowling, M. Online discovery and learning of predictive state representations. In *NIPS*, 2006.

Ng, A. Y. and Jordan, M. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *NIPS*, 2002.

Rivest, R. L. and Schapire, R. E. Diversity-based inference of finite automata. *Journal of the ACM*, 41(3):555–589, 1994.

Rosencrantz, M., Gordon, G., and Thrun, S. Learning low dimensional predictive representations. In *ICML*, 2004.

Shatkay, H. and Kaelbling, L.P. Learning topological maps with weak local odometric information. In *IJCAI*, 1997.

Singh, S., James, M., and Rudary, M. R. Predictive state representations: A new theory for modeling dynamical systems. In *UAI*, 2004.

Still, S. Information theoretic approach to interactive learning. *European Physics Letters*, 85, 2009.

Sutton, R. S. and Tanner, B. Temporal-difference networks. In *NIPS*, 2005.

Sutton, R. S., Rafols, E. J., and Koop, A. Temporal abstraction in temporal-difference networks. In *NIPS*, 2006.

Sutton, R.S. and Barto, A.G. *Reinforcement learning: An introduction*. MIT Press, 1998.

Talvitie, E. and Singh, S. Simple local models for complex dynamical systems. In *NIPS*, 2008.

Talvitie, E. and Singh, S. Maintaining predictions over time without a model. In *IJCAI*, 2009.

Wolfe, A. P. and Barto, A. G. Decision tree methods for finding reusable MDP homomorphisms. In *AAAI*, 2006.

Wolfe, B., James, M. R., and Singh, S. Approximate predictive state representations. In *AAMAS*, 2008.