Maintaining Predictions Over Time Without a Model

Erik Talvitie
Computer Science and Engineering
University of Michigan
etalviti@umich.edu

Satinder Singh
Computer Science and Engineering
University of Michigan
baveja@umich.edu

Abstract

A common approach to the control problem in partially observable environments is to perform a direct search in policy space, as defined over some set of features of history. In this paper we consider *predictive* features, whose values are conditional probabilities of future events, given history. Since predictive features provide direct information about the agent's future, they have a number of advantages for control. However, unlike more typical features defined directly over past observations, it is not clear how to maintain the values of predictive features over time. A model could be used, since a model can make any prediction about the future, but in many cases learning a model is infeasible. In this paper we demonstrate that in some cases it is possible to learn to maintain the values of a set of predictive features even when a learning a model is infeasible, and that natural predictive features can be useful for policy-search methods.

1 Introduction

A common approach to the control problem in partially observable environments is a direct search in policy space with respect to some set of features of history. In practice, features are typically directly computed from the sequence of past observations. They may be some features of the last k observations for some finite k or the presence/absence of some particular subsequence of observations. These *historical features* have the advantage of being easy to maintain over time, as new observations are added to history. Their disadvantage, however, is that it can be difficult to know which historical features are important to create a good policy without a good deal of *a priori* knowledge about the environment. In this paper we will focus on *predictive features*. These features take the form of a conditional probability of some future event, given history and a set of future actions the agent might take. Predictive features, in contrast with historical features, have direct consequences for control, as they provide information about the effects of possible behaviors the agent might engage in. As such, it may be easier to select a set of predictive features that are likely to be informative about the optimal action to take (e.g. "Will the agent reach the goal state when it takes this action?" or "Will taking this action damage the agent?"). Furthermore, some important knowledge may be expressed fairly compactly in terms of a prediction that would be complex to specify purely in terms of past observations. Literature on PSRs [Littman *et al.*, 2002] shows that an arbitrary-length history can be fully captured by a finite set of short-term predictions. On the other hand, unlike historical features, it is not immediately clear how to compute the value of a predictive feature given a particular history.

As an example, consider the game of Three Card Monte. The agent is shown three cards, one of which is the "special card." The dealer then flips over the cards to hide their faces and proceeds to mix them up by swapping the positions of two cards at every time step. The agent can observe which cards the dealer swaps. At the end of the game, the agent must identify which card is the special card. To perform well in this game, the agent need only answer one question: "Where is the special card?" The location of the special card can be expressed very easily in terms of predictive features: we could have one feature for each card whose value is the probability that if that card were flipped over, it would be the special card. However, it is not obvious, without extensive knowledge about the dynamics of the system, how to compute the values of these features for a given history.

One way to maintain predictive features is using a model. If the agent has a model of the environment it can make *any* prediction about the future, at any history. However, in complex environments, a complete model may be unavailable, and may furthermore be infeasible to learn from experience.

A model has the ability to simulate the system. As such, it has not only the ability to predict the current location of the special card, but also its location in the future. In Three Card Monte, in order to predict the location of the special card in the future, it is necessary to predict how the dealer will behave. If the decision-making process the dealer uses to choose swaps is very complex, learning a model of the system may be infeasible. Note, however, that a model's abilities to simulate future events is unnecessary in Three Card Monte. In order to make decisions, the agent needs only predict the *current* location of the special card at any given moment, with no need to predict what the dealer will do in the future. As such, one would hope that the complexity of maintaining the values of the predictive features representing the correct position of the special card would be independent of

the complexity of predicting what cards the dealer will swap, and therefore simpler than that of learning a model.

An alternate approach that would avoid the use of a model would be to treat a predictive feature as simply a function from histories to probabilities, and perform some kind of regression. Of course, this leads directly back to the original feature selection problem: which features of history are important? Furthermore, the predictions of interest may have complex and long-range dependence on history that could be difficult to capture with a reasonable set of features. In Three Card Monte, the location of the card depends on the entire history since the beginning of the game. It is precisely these long-range dependencies that model-learning methods are designed to discover. So, in our approach, we will leverage model-learning techniques in order to learn to maintain the values of predictive features over time, but we will not learn a complete model of the system. We accomplish this by modeling a transformation of the system called the *prediction profile system*. A model of the prediction profile system can be used to provide the values of the predictions of interest in the original system at any given time step, but will make no other predictions. Essentially we give up a model's ability to project what its predictions will be in the future. We will demonstrate our technique on three example domains, including Three Card Monte, which prove too complex for standard POMDP model-learning methods, but for which it is possible to maintain the values of predictive features important for finding a good policy.

2 Background

In this work we focus our attention on discrete dynamical systems. The agent has a finite set A of actions that it can take and the environment has a finite set O of observations that it can produce. At every time step τ, the agent chooses an action $a_\tau \in A$ and the environment emits an observation $o_\tau \in O$. The sequence $h_\tau = a_1 o_1 a_2 o_2 ... a_\tau o_\tau$ is the *history* at time τ. The history at time zero, $h_0 = \phi$, is the *null history*. We call a possible future sequence of actions and observations $t = a_{\tau+1} o_{\tau+1} ... a_{\tau+k} o_{\tau+k}$ a *test*. If the agent takes the action sequence in t and observes the observation sequence in t, we say that test t *succeeded*. A *prediction* $p(t|h)$ is the probability that test t succeeds after history h. Formally,

$$p(t|h) \stackrel{\text{def}}{=} \Pr(o_{\tau+1} ... o_{\tau+k} | h, a_{\tau+1} ... a_{\tau+k}). \quad (1)$$

We let T be the set of all tests and $H \stackrel{\text{def}}{=} \{t \in T | p(t|\phi) > 0\}$ be the set of all histories. We assume that the agent has a finite set $T^I \subset T$ of *tests of interest* that it wishes to accurately predict at any history. The predictions for the tests of interest are the agent's predictive features. The tests of interest could be elements of T, though in general they could also be abstract tests that capture more sophisticated predictions.

2.1 Policy Gradient Methods

Policy gradient methods have been very successful in providing a viable option for model-free control in partially observable domains. Though there are differences between various algorithms, the common thread is that they assume a parametric form for either the agent's policy, or for the value function. The goal, then, is to alter those parameters in the direction of the gradient with respect to expected reward. In this paper we will use Online GPOMDP with Average Reward Baseline [Weaver and Tao, 2001], or OLGARB (we refer readers to the original paper for details). We will assume we have a set of features f_i of history, and that the agent's policy takes the form:

$$\Pr(a|h;\vec{w}) = \frac{e^{\sum_i w_{i,a} f_i(h)}}{\sum_{a'} e^{\sum_i w_{i,a'} f_i(h)}} \quad (2)$$

where the parameters $\vec{w}$ are real-numbered weights specific to the feature and the action being considered.

2.2 Models

In this paper we will make use of two modeling representations. Partially observable Markov decision processes (POMDP) [Monahan, 1982] are a standard modeling method and we will learn POMDP models as a baseline of comparison to our method. POMDPs are typically trained using Expectation-Maximization (EM), which performs gradient ascent to find a local maximum of likelihood. As a part of our method, we will make use of looping predictive suffix trees (LPST) [Holmes and Isbell, 2006], which are specialized to deterministic systems. Though we consider stochastic systems, one part of our method involves learning a model of a deterministic system (as described in the next section). Under some conditions on the training data, LPST training converges to the optimal model.

3 Prediction Profile System

Let T^I be a set of m tests of interest whose predictions are expected to be useful features for control. We would like a function $g : H \to \mathbb{R}^m$ that maps histories to predictions for the tests of interest. A *prediction profile* is a vector of predictions for the tests of interest. In this paper, we will focus on the case where there are finitely many distinct prediction profiles. This is a restrictive assumption, though we believe there are many systems and sets of tests that have this property. For instance, in Three Card Monte, there are three possible prediction profiles, one for each position of the special card.

Say there are n prediction profiles. Then let $L = \{1...n\}$ be a finite set of labels for the prediction profiles. Then we can define g', a redefinition of g, such that $g' : H \to L$ is a mapping from histories to prediction profile *labels*. The goal of this paper is to find a compact representation of g'. We will accomplish this by defining and then modeling a new dynamical system called the *prediction profile system* (or PP).

Definition. The observations of PP, $O_{PP} \stackrel{\text{def}}{=} L$, are prediction profile labels. The actions, $A_{PP} \stackrel{\text{def}}{=} A \times O$ are action/observation pairs from the original system.

Because the observations from the original system are folded into the actions of PP, a model of the prediction profile system *conditions* on the dynamics of the original system, but need not make *predictions* about them. The dynamics of the prediction profile system are governed by

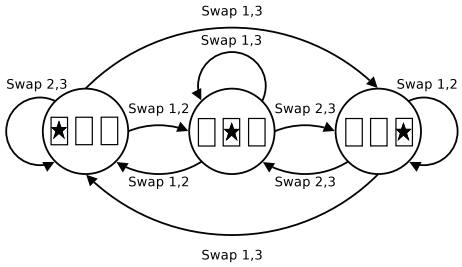

Figure 1: The prediction profile system for Three Card Monte. Transitions are labeled with the dealer's swaps. States are labeled with the predicted position of the special card.

g'. Specifically, for any action sequence $a_1 a_2 ... a_k$ with $a_1, ..., a_k \in A_{PP}$, PP emits the observation sequence $g'(a_1)g'(a_1 a_2)...g'(a_1 ... a_k)$. As an example, we show the prediction profile system for Three Card Monte in Figure 1. This shows how the prediction about the current location of the card can be maintained as the dealer makes swaps, without predicting which swaps the dealer will make.

In general, the prediction profile system is non-Markovian. As a result, the prediction profile is not a form of predictive state [Littman *et al.*, 2002]. The next prediction profile depends in general upon the entire history of the prediction profile system, rather than just the current prediction profile. Note, however, that even if the original system is stochastic the prediction profile system is always deterministic, because every history corresponds to exactly one prediction profile. The stochastic observations of the original system have been folded into the actions of the prediction profile system.

If we have a model of the prediction profile system, we can recover g' in order to obtain the prediction profile for any history. A given history is a sequence of action/observation pairs. This is converted to a sequence of prediction profile actions in A_{PP}. This sequence of actions results deterministically in a unique sequence of prediction profile observations, the last of which is the prediction profile that provides the predictions of interest given the current history.

So, a model of the prediction profile system provides the values for a set of predictive features at any history and *no other predictions*. It therefore lacks the ability to project what its predictions will be in the future, and therefore any ability to simulate the system. As a result, a prediction profile model cannot be used for model-based control techniques. For these reasons, we say that a prediction profile model is not a model of the system. However, as we will see, a prediction profile model's ability to provide accurate values for a set of predictive features at any given time step can be valuable for model-free control.

3.1 Related Work

The idea of model only some aspects of the observations of a dynamical system is not new. In some recent examples, Wolfe [2006] and Rudary [2008] both learn models that split the observation into two pieces, one of which is modeled while the other is treated as an action, or an "exogenous input." A prediction profile model, instead of predicting some piece of the next observation, predicts the *values of some predictions* at the next time step, which we believe to be a novel, and in some ways more general idea.

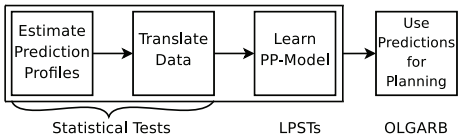

Figure 2: Flow of our algorithm.

Temporal Difference Networks (TD-nets) [Tanner and Sutton, 2005] can be used to approximately maintain a set of predictive features without a model. However, there is as yet no method for constructing a TD-net that makes accurate (or near accurate) predictions for a given set of tests of interest.

The prediction profile system is also similar in spirit to finite state controllers for POMDPs. In some cases, it is possible to represent the optimal policy for a POMDP as a finite state machine that takes action/observation pairs as inputs and that outputs actions. Multiple authors (e.g. [Hansen, 1998]) provide techniques for learning finite state controllers. However, these algorithms require access to a complete POMDP model of the world to begin with which, in our setting, is assumed to be impractical at best.

4 Learning a Prediction Profile Model

We now turn our attention to the main problem of the paper: learning a prediction profile model from experience. We assume we are given data in the form of a set S of trajectories of interaction with the original system. The learning procedure we will present has three main steps (pictured in Figure 2). First, we estimate both the number of distinct prediction profiles and their values. Second, we translate the trajectories in S to a set S' of trajectories of interaction with the *prediction profile system*. Finally, we use the translated trajectories to train a model of PP. We then use the obtained prediction profile model to maintain predictive features for use in a policy gradient algorithm.

4.1 Estimating the Prediction Profiles

Given S, we would like to determine how many distinct prediction profiles there are, as well as their values. Note that we can very simply obtain an estimated prediction profile for any history in the agent's experience. The estimated prediction for a test of interest t at a history h is[1]:

$$\hat{p}(t|h) = \frac{\text{\# times } t \text{ succeeds from } h}{\text{\# times } acts(t) \text{ taken from } h}. \quad (3)$$

Due to sampling error, it is unlikely that any of these estimated profiles will be exactly the same. We can use statistical tests (chi-square test under the appropriate approximation

[1]Bowling et al. [2006] note that the estimator in Equation 3 is biased when the exploration policy depends upon past observations and provide an unbiased estimator for general exploration policies. Our experiments use a uniform random policy. Adapting our algorithm to general exploration policies is interesting future work.

conditions and Fisher's exact test otherwise) to find histories that have *significantly* different prediction profiles.

To compare the profiles of two histories, we perform a test of homogeneity for each test of interest. If the test associated with any test of interest rejects the null hypothesis that the test has the same probability of succeeding in both histories, then the two histories have different prediction profiles.

In order to find the set of distinct prediction profiles, we maintain an initially empty set of exemplar histories. We search over all histories in the agent's experience, comparing each history's profile to the exemplar histories' profiles. If the candidate history's profile is significantly different from the profiles of all exemplar histories, we add the candidate as a new exemplar. In the end, we use the estimated profiles of the exemplar histories as the set of prediction profiles. In order to obtain the best estimates possible, we order our search to prioritize histories with lots of data.

The procedure we have outlined has two main sources of complexity. The first is the sample complexity of estimating the prediction profiles. It can take a great deal of exploration to see each history enough times to obtain good statistics, especially as the number of actions and observations increases. This issue could be addressed by adding generalization to our estimation procedure, so that data from one sample trajectory could improve the estimates of many similar histories. In our experiments we will use observation aggregation as a simple form of generalization. The second is the complexity of searching for prediction profiles, as this involves exhaustively enumerating the histories in the agent's experience. It would be valuable to develop heuristics to identify histories likely to provide new profiles. In our experiments we limit the search to short histories, as long histories will tend to have less data, and will therefore be less likely to provide new profiles.

4.2 Generating Prediction Profile Trajectories

Having generated a finite set of distinct prediction profiles, we now turn to translating the agent's experience into sequences of action/observation pairs and those prediction profiles. These trajectories will be used to train a model of the prediction profile system.

The process of translating a raw action/observation sequence s into a prediction profile trajectory s' is straightforward and, apart from a few practical concerns, follows directly from the definition of the prediction profile system given in Section 3. Recall that, for an action/observation sequence $s = a_1o_1...a_ko_k$, the corresponding action sequence for the prediction profile system is simply $acts(s') = \{a_1, o_1\}...\{a_k, o_k\}$. The corresponding observation sequence is $obs(s') = g'(a_1o_1)g'(a_1o_1a_2o_2)...g'(a_1o_1...a_ko_k)$.

Of course we do not have access to g' to generate $obs(s')$. We will approximate it by using statistical tests to compare the profile of each prefix of s to the exemplar histories' profiles. Since we considered every history in the agent's experience in the construction of the set of exemplar histories we are guaranteed that the profile every prefix of s matches the profile of at least one exemplar history. We take the profile of that matching exemplar to be the result of g'.

Unfortunately, there are some circumstances in which we cannot determine a unique matching exemplar. It is possible for the test of homogeneity to fail to reject the null hypothesis for two or more exemplar histories. This indicates that there is not enough data at the prefix to distinguish between multiple possible matching profiles. This is especially common when two or more profiles have similar values. In this case we cannot know which profile belongs at that point in the sequence. Rather than choose an arbitrary (and possibly incorrect) profile, we simply cut the trajectory short, ignoring all subsequent action/observation pairs. This is somewhat data inefficient, though we found it resulted in significantly better performance than the alternative. The development of a principled method for choosing a profile in these situations would likely improve the data efficiency of our algorithm.

4.3 Learning a Prediction Profile Model

We now have a set S' of trajectories of interaction with PP. We can use these trajectories to train a model of the prediction profile system using any model-learning technique.

We will use LPSTs to represent our model of PP. Note that, because we consider stochastic systems, LPSTs could not be used to learn a model of the original system. However, because PP is deterministic, they are well suited to our purpose. That said, prediction profile LPSTs (PP-LPSTs) can fail when some history suffixes do not occur in the training data. This can cause the PP-LPST not to make any prediction about the next observation. To address this eventuality, in our implementation we keep track of the empirical stationary distribution of prediction profiles in the training sequences. To generate a prediction for a test of interest in the absence of a prediction from the PP-LPST, we provide the expected prediction with respect to the empirical stationary distribution over profiles. We then randomly draw a prediction profile from the empirical stationary distribution and use that profile as the observation to update the PP-LPST.

5 Experiments

In this section we will apply our methods to three example problems. For each problem we provide an initial training set of experience used to learn a prediction profile model. In this "training phase" the reward the agent obtains is unimportant. So, in the training trajectories, the agent chooses amongst its actions with equal probability.

The free parameter of our algorithm is the significance value of the statistical tests, α. Given the large number of contingency tests that will be performed on the same dataset, which can compound the probability of a false negative, α should be set fairly low. In all our experiments we set $\alpha = 0.00001$, though we tried several reasonable values and achieved similar results. As discussed above, we will also set the maximum length history we will consider when we search for prediction profiles. This cutoff allows us to avoid considering long histories, as there are many long histories to search over and they are unlikely to provide new prediction profiles.

After we have learned a model of PP, we use its output as features for the policy gradient algorithm OLGARB during the "control phase." Specifically, for each test of interest t we split the unit interval up into 10 equally-sized bins b and provide a binary feature $f_{t,b}$ that is 1 if the prediction of t lies

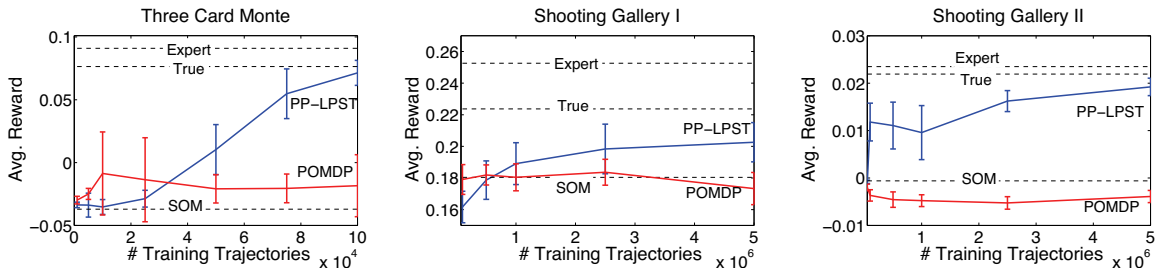

Figure 3: Control performance using features maintained by a PP-LPST with different amounts of training data compared to a POMDP model (POMDP), the true predictions (True), second-order features (SOM), and an expert policy (Expert).

in bin b, and 0 otherwise. We also provide binary features f_o, for each possible observation o. The feature $f_o = 1$ if o is the most recent observation and 0, otherwise. We set the free parameters of OLGARB, the learning rate and discount factor, to 0.01 and 0.95, respectively, run it for 1,000,000 steps and report the average reward obtained.

We compare the performance of OLGARB using predictive features maintained by a PP-LPST trained using our method to its performance using the predictions of a POMDP model trained on the raw data. Because these problems are too complex to feasibly train a POMDP with the correct number of underlying states, we trained a 30-state POMDP (stopping EM after a maximum of 50 iterations)[2]. As baselines, we also compare to OLGARB using the true predictions as features (the best we could hope to do), OLGARB using second-order Markov features (the two most recent observations, as well as the action between them) but no predictive features, and an expert-coded policy.

5.1 Three Card Monte

The first domain is Three Card Monte. The agent is presented with three cards. Initially, the card in the middle (card 2) is the "special card." The agent has four actions available to it: $watch$, $flip1$, $flip2$, and $flip3$. If the agent chooses a flip action, it observes whether the card it flipped over is the special card. If the agent chooses the $watch$ action, the dealer can swap the positions of two cards, in which case the agent observes which two cards were swapped, or the dealer can ask for a guess. The agent receives no reward for choosing $watch$. If the dealer asks for a guess and the agent flips over the special card, the agent gets reward of 1. If the agent flips over one of the other two cards, or doesn't flip a card, it gets reward of -1. If the dealer has not asked for a guess, then $watch$ results in 0 reward and any flip action results in -1 reward. The agent has three tests of interest, and they take the form $flipX\ special$, for each card X.

Note that we have a representational choice. We could consider all three tests of interest at once, or we could treat them separately and learn a PP-LPST for each one. The advantage to splitting them up is that there are fewer prediction profiles for each individual PP-LPST (there are 3 prediction profiles when they are grouped, but only 2 profiles each if they are separate). On the other hand, if they are grouped together, the prediction profile system is Markov (that is, the next prediction profile depends only on the current prediction profile and the next action/observation pair). If we were to split them up, each individual prediction profile system would be non-Markov, making learning more difficult. In this case, we group them together and learn one PP-LPST for all three tests. We will see an example where learning separate PP-LPSTs is beneficial in the next experiment.

As discussed previously, the complexity of this system is directly related to the complexity of the dealer's decision-making process. In our experiments, when the agent chooses "watch" the dealer swaps the pair of cards it has swapped the least so far with probability 0.5; with probability 0.4 it chooses uniformly amongst the other pairs of cards; otherwise it asks for a guess. Since the dealer is keeping a count of how many times each swap was made, the process governing its dynamics effectively has infinitely many latent states. The prediction profile system, on the other hand, has only 3 states, regardless of the dealer's complexity (see Figure 1).

For the training phase we used length 10 trajectories of experience with the Three Card Monte problem. Figure 3a shows the results of the control phase for various amounts of training data, averaged over 20 trials. The predictive features are clearly useful for performance, and the PP-LPST learns to maintain them correctly. As we expected, the POMDP model was unable to accurately predict the tests of interest. Second-order features were also essentially uninformative.

5.2 Shooting Gallery I

In our second experiment, the agent is at a shooting gallery (see Figure 4a). The agent has 11 actions: $lookX$, for each of 10 positions X, and $shoot$. When the agent looks at a position (marked by the 'X' in the figure), it observes whether there is a target in that position and in the two adjacent positions (unshaded region). When the agent shoots, the bullet hits the current position with probability 0.8 and to either side each with probability 0.1. Whenever the agent hits a target, it gets reward of 1. At every time step the gallery resets with probability 0.2. When the gallery resets, the agent receives a special observation and each target turns on with probability 0.3. In order to do well, the agent must remember targets it has seen so it can go back to shoot them, and so it can prioritize targets (clusters of targets are best due to the inaccuracy of the gun). So, the agent has 10 tests of interest: $lookX\ target$, for each position X. In this case, learning one PP-LPST for all 10 tests

[2]We obtained similar results with 5, 10, 15, 20, and 25 states.

of interest would result in over 50,000 prediction profiles. We will learn a separate PP-LPST for each test of interest, each having only 3 profiles. Note that this system has 2^{10} latent states, which is too many to feasibly learn a POMDP. Also, unlike Three Card Monte, the prediction profiles are non-deterministic, so more data is required to distinguish profiles. In the training phase we used length 4 trajectories, restricting our search to length 2 histories. Figure 3b shows the results of the control phase. In this problem, OLGARB with the true predictions does not perform as well as the hand-coded policy. This is likely due to the algorithm converging to a local maximum. The PP-LPST does not learn to perfectly maintain the predictive features, though the values it provides are more useful for control than those provided by the POMDP model.

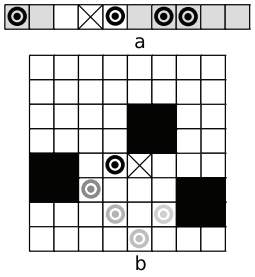

Figure 4:
(a) Shooting Gallery I
(b) Shooting Gallery II

5.3 Shooting Gallery II

For our final experiment, we have another type of shooting gallery, pictured in Figure 4b. The agent has a gun aimed at a fixed position on an 8×8 grid (marked by the 'X') . A target moves diagonally, bouncing off of the edges and 2×2 obstacles (an example trajectory is pictured). The agent's task is to shoot the target. The agent has two actions: $watch$ and $shoot$. When the agent chooses $watch$, it gets 0 reward. If the agent chooses $shoot$ and the target is in the crosshairs in the step after the agent shoots, the agent gets reward of 10, otherwise it gets a reward of -5. Whenever the agent hits the target, the shooting range resets: the agent receives a special observation, each 2×2 square on the range is made an obstacle with probability 0.1, and the target is placed in a random position. There is also a 0.01 probability that the range will reset at every time step. The difficulty is that the target is "sticky." Every time step with probability 0.7 it moves in its current direction, but with probability 0.3 it sticks in place. Thus, looking only at recent history, the agent may not be able to determine the target's current direction. The agent needs to know the probability that the target will be in its sights in the next step, so clearly the test of interest is: $watch\ target$.

Due to the number of possible configurations of obstacles and positions of the target, this system has roughly 4,000,000 observations and even more latent states. This results in a large number of possible histories, each with only a small probability of occurring. As discussed, this can lead to a large sample complexity for obtaining good estimates of prediction profiles. We address this with a simple form of generalization: observation aggregation. We will treat two observations as the same if the target is in the same position and if the configuration of obstacles in the immediate vicinity of the target is the same. Even with this aggregation, there are over 6000 action/observation pairs. We assume two histories have the same prediction profile if they have the same aggregate observations. This allows us to use one sample trajectory to improve the estimates for several histories. We performed the same observation aggregation when training the POMDP model. We trained our method using length 4 trajectories, restricting our search for prediction profiles to length 3 histories. Results are shown in Figure 3c. Again, the PP-LPST learns to maintain the predictive features well, resulting in good control performance while the POMDP model gives less useful information than the second order Markov features.

6 Conclusions

We have demonstrated that in some cases, it is possible to learn to maintain a small set of predictions of interest by learning a model of how the *values* of those predictions change over time. Empirically we were able to learn to maintain predictive features useful for policy-gradient techniques, even when learning a model was infeasible.

Acknowledgements

Erik Talvitie was supported under the NSF GRFP. Satinder Singh was supported by NSF grant IIS-0413004. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

[Bowling *et al.*, 2006] Michael Bowling, Peter McCracken, Michael James, James Neufeld, and Dana Wilkinson. Learning predictive state representations using non-blind policies. In *International Conference on Machine Learning 23 (ICML)*, pages 129–136, 2006.

[Hansen, 1998] Eric Hansen. *Finite-Memory Control of Partially Observable Systems*. PhD thesis, University of Massachussetts, Amherst, MA, 1998.

[Holmes and Isbell, 2006] Michael Holmes and Charles Isbell. Looping suffix tree-based inference of partially observable hidden state. In *International Conference on Machine Learning 23 (ICML)*, pages 409–416, 2006.

[Littman *et al.*, 2002] Michael Littman, Richard Sutton, and Satinder Singh. Predictive representations of state. In *Advances in Neural Information Processing Systems 14 (NIPS)*, pages 1555–1561, 2002.

[Monahan, 1982] George E. Monahan. A survey of partially observable markov decisions processes: Theory, models, and algorithms. *Management Science*, 28(1):1–16, 1982.

[Rudary, 2008] Matthew Rudary. *On Predictive Linear Gaussian Models*. PhD thesis, University of Michigan, 2008.

[Tanner and Sutton, 2005] Brian Tanner and Richard Sutton. Temporal difference networks. In *Neural Information Processing Systems 17 (NIPS)*, pages 1377–1384, 2005.

[Weaver and Tao, 2001] Lex Weaver and Nigel Tao. The optimal reward baseline for gradient-based reinforcement learning. In *Uncertainty in Artificial Intelligence 17 (UAI)*, pages 538–545, 2001.

[Wolfe, 2006] Alicia Peregrin Wolfe. Pomdp homomorphisms. Presented at The NIPS-2006 Workshop on Grounding Perception, Knowledge and Cognition in Sensory-Motor Experience. Available at `http://www-all.cs.umass.edu/~pippin/publications/NIPSRLWorkshop2006Pape%rPOMDPHMs.pdf`, 2006.