In *The IJCAI-2005 Workshop on Planning and Learning in A Priori Unknown or Dynamic Domains* (IJCAI 2005), Edinburgh, Scotland, UK, August 2005.

Bayesian Models of Nonstationary Markov Decision Processes

Nicholas K. Jong and **Peter Stone**
Department of Computer Sciences
University of Texas at Austin
Austin, Texas 78712
{nkj,pstone}@cs.utexas.edu

Abstract

Standard reinforcement learning algorithms generate polices that optimize expected future rewards in a priori unknown domains, but they assume that the domain does not change over time. Prior work cast the reinforcement learning problem as a Bayesian estimation problem, using experience data to condition a probability distribution over domains. In this paper we propose an elaboration of the typical Bayesian model that accounts for the possibility that some aspect of the domain changes spontaneously during learning. We develop a reinforcement learning algorithm based on this model that we expect to react more intelligently to sudden changes in the behavior of the environment.

1 Introduction

Reinforcement learning (RL) research provides algorithms for generating universal plans from experience, given minimal prior knowledge about the domain [Sutton and Barto, 1998]. Classical RL algorithms assume only that the domain obeys the Markov property: the effects of each action depend only on the currently observed state. However, the behavior of many interesting domains depends on factors that are difficult or impossible to represent in the state space. A robot's effectors may change unexpectedly due to damage. An overturned truck may render a highway suddenly impassable. An opened door in a previously explored area may grant access to new opportunities. Standard RL algorithms adapt only gradually to such drastic changes to the overall system. Enough experience after the change must accumulate to outweigh the outdated knowledge. The agent may *never* notice a change that occurs in a region of the state space that the learned behavior doesn't visit.

In this paper, we consider statistical methods for detecting changes in the domain in a more timely manner. Intuitively, an intelligent agent should notice when the environment ceases to behave as expected. Such an agent should consider throwing out or discounting its old model of the relevant aspects of the environment. We adopt a Bayesian framework that allows us to reason explicitly about uncertainty over the domain [Strens, 2000]. We elaborate the standard probabilistic model to represent the possibility of domain change. We then propose an algorithm that employs statistical inference techniques to behave more robustly in the presence of domain change.

2 Background

The standard domain formalism in RL research is the Markov decision problem (MDP). An MDP $\langle S, A, P, R \rangle$ comprises a finite set of states S, a finite set of actions A, a transition function $P : S \times A \times S \to [0, 1]$, and a reward function $R : S \times A \to \mathbb{R}$. Executing an action a in a state s yields an expected immediate reward of $R(s, a)$ and causes a transition to state s' with probability $P(s, a, s')$. A policy $\pi : S \to A$ specifies an action $\pi(s)$ for every state s and induces a value function $V^\pi : S \to \mathbb{R}$ that satisfies the Bellman equations $V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} P(s, \pi(s), s') V^\pi(s')$, where $\gamma \in [0, 1]$ is a discount factor for future reward that may be necessary to make the equations satisfiable. For every MDP at least one optimal policy π^* exists that maximizes the value function at every state simultaneously. To compute an optimal policy from a fully specified MDP, a number of algorithms are available, including dynamic programming, policy iteration, and linear programming [Littman *et al.*, 1995].

In the RL problem, only the state space S and the action space A are known a priori, but standard approaches assume that the transition function P and reward function R are fixed. An important class of RL algorithms are model-based: they compute policies by first estimating P and R and then solving the estimated MDP. Although solving an MDP is too computationally expensive to perform after every time step, algorithms such as Prioritized Sweeping [Moore and Atkeson, 1993] describe how to propagate incremental updates to a model through a policy learned through dynamic programming. However, model-based algorithms are particularly vulnerable to nonstationary domains, since they typically employ maximum likelihood estimates of parameters of the model given all the available experience data. Hence, changes in the domain will tend to averaged into a large body of outdated prior experience.

In this paper, we elaborate a model-based algorithm called Bayesian dynamic programming [Strens, 2000]. For each state-action pair (s, a), this approach interprets $P(s, a, \cdot)$ as the parameters of an a priori unknown multinomial distribution and $R(s, a)$ as the mean of an unknown normal dis-

tribution.[1] We represent our initial uncertainty about these unknown distributions as prior distributions over their parameters. The joint distribution over all the presumed-independent state-action pairs yields a probability distribution over MDPs. Since conjugate families of prior distributions exist for both the multinomial and normal distributions, we can compactly represent and efficiently update these distributions over MDPs. At the beginning of each training episode, Bayesian dynamic programming samples a hypothetical model of the domain from this distribution and then behaves according to the policy obtained from solving the model.

3 A Probabilistic Model of Change

We propose a simple model of environmental change: after every episode and for each state-action pair, the associated multinomial successor-state distribution and normal reward distribution reset with some small probability to distributions drawn from the respective original priors. This model caters to the fact that only small parts of a domain may change at a time. A more sophisticated model might also capture the fact that a change in one state-action pair makes a change in another pair more likely, but such models may be quite complex.

Our Bayesian model of the domain must change to accomodate this probability of reset. Suppose that we have k complete training episodes of data. Consider a particular state-action pair (s, a). Let T denote the episode number when the state-action pair (s, a) last reset, so $T = 0$ is the hypothesis that the behavior of a at state s has never changed and $T = k$ is the hypothesis that the behavior reset at the beginning of the current episode. Let P denote the successor state distribution given (s, a). Then we have a hierarchical distribution over P, given by $\Pr(P = \vec{p}) = \sum_t \Pr(P = \vec{p}|T = t) \cdot \Pr(T = t)$, where $\Pr(P = \vec{p}|T = t)$ is the result of the standard Bayesian conditioning process, but using only a suffix of all the data. Similar reasoning applies to the distribution over R, the reward function evaluated at (s, a). When we sample a hypothesis MDP from our Bayesian model, we must now first draw a sample hypothesizing the last time each state-action pair reset (according to the distribution $\Pr(T)$, before sampling the pieces of the transition and reward functions from posterior distributions conditioned on the corresponding suffixes of the data.

No conjugate family of prior distributions exists for $\Pr(T)$, so in practice we approximate this distribution by maintaining the relative probabilities for a small subset of episode numbers. The computation of these relative probabilities poses another obstacle. Suppose that we have a prior distribution $\Pr(T)$ that we want to update given experience data D that we assume all come from the same distribution. Then from Bayes' Theorem we have $\Pr(T = t|D) \propto \Pr(D|T = t) \cdot \Pr(T = t)$. We can rewrite the model likelihood as an integral over Bayesian models conditioned on a suffix of the data: $\Pr(D|T = t) = \int\int \Pr(D|P = \vec{p}, R = r, T = t) \cdot \Pr(P = \vec{p}|T = t) \cdot \Pr(R = r|T = t)\ d\vec{p}\,dr$. Computing the model likelihood exactly is infeasible, but we can approximate it with Monte Carlo integration by sampling some number of values for P and R, again conditioned on the appropriate suffix of the data.

Our Bayesian model of the state-action pair (s, a) is therefore approximate in two ways. First, we approximate $\Pr(T)$ with a bounded-size sample of the most probable values of T. Each point in this sample has a scalar weight and a Bayesian model of P and R, represented exactly as the appropriate parameters to the conjugate priors for these distributions. The second approximation is in our Bayesian update of our model given new data. We reweight the sample by multiplying each weight by the model likelihood, estimated using Monte Carlo integration.

4 Application of the Bayesian Model

We can use the Bayesian model elaborated above directly with Strens' Bayesian dynamic programming algorithm [Strens, 2000]. After each episode, we add to our sample of $\Pr(T)$ the hypothesis for each state-action pair that it reset before that episode. We give this hypothesis some portion of the weight equal to our prior probability of domain change at each episode. Then we condition the weights and each model of P and R on the data from that episode. To keep the sample size reasonable, we select a value of T to discard. Finally, for the next episode, we sample an MDP from the hierarchical model.

This Bayesian approach to recognizing domain change allows us to avoid unilateral commitments to either keeping or discarding old data. Additionally, in the absence of evidence to the contrary, it gradually increases the belief that neglected state-action pairs have reset. If the prior distribution over the reward function is optimistic, then the agent will eventually choose to explore the action again.

Unfortunately, Bayesian dynamic programming does not always work so well when the state-action pair that changed is part of the learned policy. If a previously reliable action suddenly fails entirely, the solution to the sampled MDP may cause the agent stubbornly to retry expensive actions until timing out. The same phenomenon can occur in model-free methods such as Q-learning: depending on the learning rate, the agent may spend quite some time in a negative-reward loop.

We propose a small modification of Bayesian dynamic programming. If the number of visits to a state in a single episode exceeds a certain threshold, we begin to conduct statistical goodness-of-fit tests to evaluate our hypotheses for P and R at the appropriate state-action. If the data collected during that episode cause us to reject the sample of P or R, we immediately resample them from a Bayesian model conditioned on only that data. We then update the policy as necessary to solve the updated MDP. Note that even though we resample from a distribution assuming that a reset occurred, we still update the Bayesian model as usual at the end of the episode.

[1] The MDP formalism does not require the rewards to be normally-distributed, but this assumption seems fairly innocuous given that we care only about the mean of the reward distribution.

5 Future Work and Discussion

The implementation and evaluation of the algorithm described above remains to be done. However, we believe that this approach of building a Bayesian model of domain uncertainty is very promising. One concern is the computational cost of Bayesian inference, but the proposed modifications to Bayesian dynamic programming should not worsen the runtime much. The Monte Carlo integration only occurs for state-action pairs executed during an episode, and the primary cost of this procedure is the sampling of P and R that the algorithm already performs once for every state-action pair. The goodness-of-fit tests only occur upon revisiting a state several times in the same episode, and some form of exponential backoff can help prevent spurious testing. (Testing again after one additional data point is unlikely to produce a different result.)

The ideas described in this paper are reminiscent of our previous usage of a Bayesian model of MDPs to infer state abstractions [Jong and Stone, 2005] from the solution of one MDP for use in similar MDPs. A particularly promising avenue of future research is the usage of a single Bayesian model to reason about both dynamic domains and state abstraction simultaneously. In this framework we can imagine inducing structure such as state abstraction that continues to aid learning despite continual changes in the reward and transition functions.

Acknowledgments

This research was supported in part by NSF CAREER award IIS-0237699 and DARPA grant HR0011-04-1-0035.

References

[Jong and Stone, 2005] Nicholas K. Jong and Peter Stone. State abstraction discovery from irrelevant state variables. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, 2005.

[Littman *et al.*, 1995] Michael L. Littman, Thomas L. Dean, and Leslie Pack Kaelbling. On the complexity of solving Markov decision problems. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, 1995.

[Moore and Atkeson, 1993] Andrew W. Moore and Christopher G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, 13:103–130, 1993.

[Strens, 2000] Malcolm Strens. A Bayesian framework for reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 943–950, 2000.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.