Learning Monitoring Strategies to Compensate for Model Uncertainty

Eric A. Hansen, Paul R. Cohen
Experimental Knowledge Systems Laboratory
Department of Computer Science
University of Massachusetss, Amherst, MA 01003
hansen@cs.umass.edu

Abstract

This paper addresses the need for monitoring the environment given an action model that is uncertain or stochastic. Its contribution is to describe how monitoring costs can be included in the framework of Markov decision problems, making it possible to acquire cost-effective monitoring strategies using dynamic programming or related reinforcement learning algorithms.

Statement of the Problem

In realistic domains, an agent must generate plans from an action model that is imperfect and uncertain. Even if the model can be improved by learning, there is usually a limit to how accurate it can be made; to some degree, the effects of actions are stochastic. When an agent cannot predict the effects of its actions with certainty, it must monitor the state of its environment. (In the rest of this paper, the terms "monitoring" and "sensing" are used interchangeably.) However if a cost is incurred for monitoring, it may be prohibitively expensive for an agent to monitor every feature of its environment continuously. Hence there is a need for cost-effective monitoring strategies.

Recent work that considers sensing costs in learning stategies for robotic sensing attempts to make sensing more efficient by selecting a subset of the available features of the environment to sense (Chrisman & Simmons, 1991; Tan, 1991). An interesting aspect of this work is that it deals with the issue of incomplete state descriptions (Whitehead & Ballard, 1991). An assumption it makes, however, is that the agent senses its environment (or selected features of it) at a fixed periodic interval, typically at the beginning of each time step or decision cycle. In this paper, we start by questioning this assumption. We consider an agent that can decide for itself when to sense the world, and how long to wait before sensing again. So our focus is less on the problem of what features of the environment to sense than when and how often to sense them.

For both generality and rigor, we have attempted to treat this problem as a Markov decision problem using methods based on dynamic programming. This framework has been used in recent work to make useful connections between planning and learning (Sutton, 1990). In this framework, an action model takes the form of a state transition function, $P_{xy}(a)$, that gives the conditional probability that action a taken in state x produces state y. In addition, a payoff function, $R(x,a)$, gives the expected single-step payoff for taking action a in state x. The problem is to find a policy that optimizes payoff in the long term. This policy can be found with dynamic programming, or various reinforcement learning algorithms that have been shown to have a theoretical basis in dynamic programming (Barto, Sutton, & Watkins, 1990).

In a conventional Markov decision problem, the state of the environment is automatically monitored at each time step without considering the costs this might incur. Because this is exactly the assumption we wish to question, the key step in our work is to show how to express monitoring costs and formulate monitoring strategies in the framework of a Markov decision problem. Given a conventional Markov decision problem with single-step state-transition probabilities and a single-step reward function, we show how to transform it into another Markov decision problem in which the agent does not automatically monitor each time step, but considers sensing costs in deciding when and how often to monitor. It is not possible to describe all the details of this approach in this short paper, but a simple example is followed by a brief commentary.

A simple example

Consider a Markov chain with five states labelled from 0 to 4, in which the highest numbered state, 4, is an *absorbing state*. The action set for the controller is $A = \{null, restart\}$. If the controller does nothing (i.e., performs the null action) at a given time step, the Markov chain has the state transition probabilities shown in figure 1.

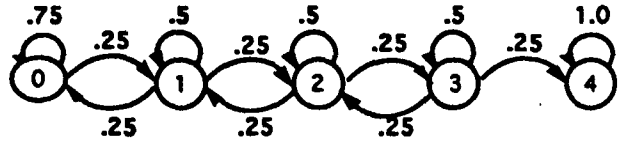

Figure 1. State transition probabilites for the null action

The controller can restart the Markov chain once it enters the absorbing state, 4. Restarting it restores it to state 0 and has the state transition probabilites shown in figure 2.

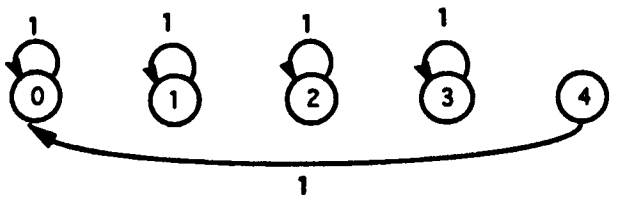

Figure 2. State transition probabilites for restart.

For example, a restart in state 4 effects a transition to state 0 but a restart in state 2 does not change the state. (We assume the effect of a restart is instantaneous, while the the effect of the null action takes one time step.)

There is a cost, -2, assesed for each time step the process is in the absorbing state, and no cost assessed when it is in the other states. A cost, -3, is incurred for restarting the process, and there is no cost for the null action. So the payoff function for this problem is:

$$R(x,null) = \begin{cases} -2 & \text{if } x = 4 \\ 0 & \text{otherwise} \end{cases}$$
$$R(x,restart) = -3$$

The optimal control policy is self-evident; the process should be restarted whenever it enters its absorbing state. This policy can also be found by computing the optimal value function, below, by using some method of infinite-horizon dynamic programming such as policy iteration.

$$V(x) = \max_{a \in A} \left\{ R(x,a) + \lambda \sum_{y \in S} P_{xy}(a) V(y) \right\}$$

Using a discount factor of $\lambda = 0.95$, the optimal policy for this problem and its expected cost are displayed in table 1.

state	0	1	2	3	4
control action	null	null	null	null	restart
monitoring interval	1	1	1	1	1
expected cost (not including monitoring cost)	-1.06	-1.28	-1.77	-2.64	-4.06
expected cost (including monitoring cost)	-21.06	-21.28	-21.77	-22.64	-24.06

Table 1. Optimal policy for original Markov decision problem

The last row of the table shows how the expected cost would increase if a monitoring cost, $M = -1$, is assessed each time step. In a conventional Markov decision problem, the controller has no control over costs incurred by monitoring because the state of the process is monitored automatically at each time step. We show that a more cost-effective policy can be found by defining a new Markov decision problem in which monitoring costs are considered and the controller does not automatically monitor each time step but decides for itself how many steps to wait before monitoring again. This new Markov decision problem is constructed from the original one by defining a new action set:

$$A^* = \{null, restart\} \times \{1,2,3,\ldots\}$$

where a tuple, $\langle a,m \rangle$, represents a decision to take action, a, and monitor again m time steps later.

A multi-step state transition function is defined by matrix multiplication,

$$P^*_{xy}(\langle a,m \rangle) = \left(P(a) P^{m-1}(null) \right)_{xy}$$

where a is either *null* or *restart*, and $P(null)$ and $P(restart)$ are the single-step state transition probability matrices displayed graphically in figures 1 and 2.

A multi-step expected payoff function is defined in terms of the original payoff function as follows:

$$R^*(x,\langle a,m \rangle) = M + R(x,a) + \sum_{j=1}^{m} \left(\lambda^{j-1} P^*_{xy}(\langle a,j \rangle) R(y,null) \right)$$

where $M = -1$, again, is the cost incurred for monitoring.

The optimal value function for this new decision problem is

$$V(x) = \max_{\langle a,m \rangle \in A^*} \left\{ R^*(x,\langle a,m \rangle) + \lambda^m \sum_{y \in S} P^*_{xy}(\langle a,m \rangle) V(y) \right\}$$

It can be solved only if the number of action and monitoring interval pairs, $\langle a,m \rangle$, that must be evaluated in each state, is finite. One way to ensure this is to put a bound on the maximum monitoring interval m. A less arbitrary solution is to reason that the expected value in state x for action a is a unimodal function of the monitoring interval, m. This is a perfectly reasonable assumption since it amounts to saying that the closer the monitoring interval is to the optimum, the better it is. This allows an optimal monitoring interval for each state-action pair to be found by bounded search or simple gradient ascent.

The optimal policy and its expected cost have been computed by dynamic programming and are displayed in table 2.

state	0	1	2	3	4
control action	null	null	null	null	restart
monitoring interval	11	8	5	2	11
expected cost (including monitoring costs)	-4.98	-5.70	-6.90	-8.44	-7.98

Table 2. Optimal policy for the redefined Markov decision problem.

Two observations can be made about this policy. First, taking monitoring costs into consideration makes it possible to compute a more efficient policy in which the state of the Markov process does not have to be monitored

at each time step. Second, the controller no longer monitors at a fixed periodic rate. In this example, its rate of monitoring increases as it gets closer to the costly absorbing state. This reflects the general idea that there are regions of a problem space or environment in which a controller needs to "be more careful", so to speak, by monitoring more frequently. We have found that a pattern much like this emerges from a number of different problem-solving situations. Atkin and Cohen (1993) describe an example in which it is worthwhile for an agent to monitor more frequently as it approaches a goal.

Dynamic programming is not the only way to determine a policy once monitoring costs and strategies have been expressed in a Markov decision problem. Various reinforcement learning algorithms have been developed that have a theoretical basis in dynamic programming and can be used either for direct learning, when an explicit cost and probability model are not available, or for real-time planning and learning when a model is available but the problem space is too large to perform full passes of dynamic programming in real-time (Barto, Bradtke, & Singh, 1993). Although not described here, we have designed a simple extension of the Q-learning algorithm that learns to monitor as part of its control policy. It works by using a separate stochastic Gaussian unit for each state-action pair to find the optimal monitoring interval by gradient ascent.

Work in Progress

Two interesting classes of policies can be found using our approach. The first is illustrated by the previous example: an agent observes the current state, performs a single action (which could be the null action), and then waits some number of steps before monitoring again. This is typical of problems in which, for example, a process is monitored over time to detect when a corrective action should be taken. It also applies to problems in which a single action is continued for a period of time and periodically monitored to decide whether to continue it or not.

However this class of policies is only a special case of a more general class in which, in each state, a policy specifies an open-loop sequence of actions to perform before monitoring again some number of time steps later. The difference between the first class and the more general class is that, in the latter, a sequence of different actions is taken before monitoring again, rather than a single action. This broader class of policies is more closely related to planning problems in which a sequence of actions is taken to get from a start state to a goal state. However the first class of policies, described here, corresponds to an important set of practical monitoring problems; moreover, finding an optimal policy in the first case is computationally simpler. While the number of possible action and monitoring interval pairs, $\langle a,m \rangle$, in the first class is a linear function of the monitoring interval, the number of possible open-loop sequences of actions in the general class grows exponentially as a function of the monitoring interval.

In both these classes of problems, the purpose of monitoring is simply to determine the current state of the environment. Another purpose of monitoring, one we have not yet considered, might be to check whether the model used by the agent to generate its behavior is correct or needs to be revised. A very simply example of this is described in (Grefenstette & Ramsay, 1992), where a monitor detects discrepancies between the parameters of an agent's model and the environment to decide whether to update the model parameters and trigger a learning algorithm to revise the agent's behavior. It might be interesting to explore, within a framework like the one described here, how uncertainty about whether an action model is correct might influence the rate of monitoring.

Acknowledgments

Thanks to Andrew Barto and Satinder Singh for helping to clarify some of these ideas in discussion, and to Scott Anderson for comments on a draft.

This research is supported by the Defense Advanced Research Projects Agency under contract #F49620-89-C-00113; by the Air Force Office of Scientific Research under the Intelligent Real-time Problem Solving Initiative, contract #AFOSR-91-0067; and by Augmentation Award for Science and Engineering Research Training, PR No. C-2-2675. The US Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon.

References

Atkin, M. and Cohen, P. (1993). Genetic programming to learn an agent's monitoring strategy. In this proceedings.

Barto, A.G., Sutton, R.S., and Watkins, C.J.C.H. (1990). Learning and sequential decision making. In *Learning and Computational Neuroscience: Foundations of Adaptive Networks*. M. Gabriel and J. W. Moore (Eds.), MIT Press, pp. 539-602.

Barto, A., Bradtke, A., and Singh, S. (1993). Learning to act using real-time dynamic programming. Computer Science Technical Report 93-02, University of Massachusetts, Amherst.

Chrisman, L. and Simmons, R. (1991). Sensible planning: Focusing perceptual attention. *Proceedings AAAI-91*, pp. 756-761.

Grefenstette, J.J., & Ramsey, C.L. (1992). An approach to anytime learning. *Proceedings of the Ninth International Conference on Machine Learning*, pp. 189-195.

Sutton, R.S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. *Proceedings of the Seventh International Conference on Machine Learning*, pp. 216-224,.

Tan, M. (1991). Learning a cost-sensitive internal representation for reinforcement learning. In *Proceedings of the Eighth International Workshop on Machine Learning*, pp. 358-362.

Whitehead, S.D., & Ballard, D.H. (1991). Learning to perceive and act by trial and error. *Machine Learning* 7:45-83.