CS294-40 Learning for Robotics and Control Lecture 9 - 9/25/2008

Partially Observable Systems

Lecturer: Pieter Abbeel *Scribe: Martin M. Sørensen*

1 Lecture outline

- Partially observable systems
- Information states
- Separation principle and multi-armed bandits intro

2 Example: Machine repair problem

[This example is similar to an example in Bertsekas, Dynamic Programming and Optimal Control, 2nd Ed., Chapter 5.]

The machine in this example can be in one of two following states P and $\bar{P}$ which correspond to the machine functioning properly and not functioning properly respectively. Two different actions can be chosen which are C = *Continue* or R = *Repair.* Given the following rewards and transition possibilities,

$$\begin{array}{ll} R(P,C) = 2 & P(P|P,C) = \frac{2}{3} \\ R(\bar{P},C) = 0 & P(P|\bar{P},C) = 0 \\ R(x,R) = 1 & P(P|P,R) = \frac{2}{3} \\ & P(P|\bar{P},R) = \frac{1}{3} \end{array}$$

the transition model depicted in figure 1 can be made.

Figure 1: Transition model of machine repair problem including the reward of being in a certain state given an action, and the transition probabilities

We are interested in finding the optimal policy using the value function

$$\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{H} R(s,a)\right]$$

The problem, as currently phrased, is readily solved using value iteration. Concretely, initialize $V_{H+1} = 0$ and then run the Bellman backups:

$$V_k(s) \leftarrow \max_{a\in\{R,C\}} \left[R(s,a) + \sum_{s'} P(s'|s,a)V_{k+1}(s')\right]$$

$$\mu_k = \arg \max_{a\in\{R,C\}} \left[R(s,a) + \sum_{s'} P(s'|s,a)V_{k+1}(s')\right]$$

The running time scales as $O(H|S|^2|A|)$, which is *linear* in the horizon H.

3 Partially Observable Systems

If there is some noise on our observations, then the states become partially observable as illustrated in Figure 2, which illustrates the observation model. (Note, for the purposes of this example: we assume the control policy does not get to observe the rewards, the only avialable observations are illustrated in Figure **??**.)

Figure 2: Observation model of two states where there is some noise on the observations.

Initially $P(s_0 = \hat{P}) = P(s_0 = P) = \frac{1}{2}$

In case of uncertainty about the state of the system, we can treat the history as the state for control purposes. The problem can then be solved by using dynamic programming from back to start over these "history-states," often called "information states."

Unfortunately, by doing so the complexity blows up exponentially in the horizon H. Figure 3 illustrates this.

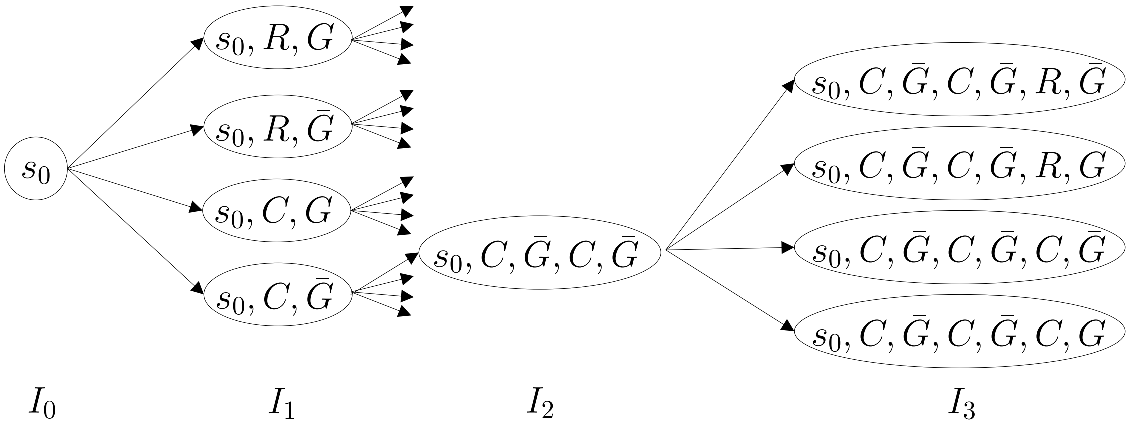
s_0
s_0, R, G
$s_0, R, \bar{G}$
s_0, C, G
$s_0, C, \bar{G}$
$s_0, C, \bar{G}, C, \bar{G}$
$s_0, C, \bar{G}, C, \bar{G}, R, \bar{G}$
$s_0, C, \bar{G}, C, \bar{G}, R, G$
$s_0, C, \bar{G}, C, \bar{G}, C, \bar{G}$
$s_0, C, \bar{G}, C, \bar{G}, C, G$
I_0
I_1
I_2
I_3

Figure 3: Illustrate how the problem quickly blows up due to the uncertainty in the observations

We now consider the dynamic programming expression to back-up from time 3 to time 2:

$$V_2(I_2) = \max_{a \in \{R,C\}} \{\mathbb{E}\left[R + V(I_3)|I_2\right]\}$$

Concretely, if we pick Repair as the action then we have:

$$\begin{aligned} Q_2(I_2, a = R) &= \mathbb{E}[R|I_2, a_2 = R] + \mathbb{E}[V(I_3)|I_2, a_2 = R] \\ &= P(P|I_2, a_2 = R)R(P, R) + P(\bar{P}|I_2, a_2 = R)R(\bar{P}, R) + \sum_{I_3} \underbrace{P(I_3|I_2, a_2 = R)} V(I_3) \end{aligned}$$

The transition probability highlighted by the under bracket can be found from the transition model and the observation model. In particular, let $I_3 = (I_2, a_2 = a, o_3 = o)$, namely the successor of the information state I_2, which is achieved by taking action a and ending up observing o. Then we have:

$$
\begin{aligned}
P(s_3 = s', I_3 = (I_2, a_2 = a, o_3 = o) | I_2, a_2 = a) =& P(o_3 = o | s_3 = s') P(s_3 = s' | s_2 = s, a_2 = a) P(s_2 = s | I_2) \\
P(I_3 = (I_2, a_2 = a, o_3 = o) | I_2, a_2 = a) =& \sum_{s'} P(s_3 = s', I_3 = (I_2, a_2 = a, o_3 = o) | I_2, a_2 = a) \\
P(s_3 = s' | I_3) =& P(s_3 = s', I_3) / \sum_{s} P(s_3 = s, I_3)
\end{aligned}
$$

We can similarly forward compute the information state transition probabilities for all times k, and initialize with $P(s_0 = s | I_0)$.

This completes the (fairly succinct) exposition of the computations required for value iteration over the information states—namely, computing the information state space transition probabilities, and the information state Bellman back-ups.

In practice these computations tend to quickly become infeasible, as the number of information states grows exponential with the horizon H.

4 Policy search

To avoid the problem blowing up, we could posit a parameterized policy and then numerically optimize the policy's parameters through simulation.

For example, for the repair problem, we could posit the following policy:

$$
\begin{aligned}
\text{if } P(s = P) < \beta \rightarrow & \text{ repair} \\
\text{otherwise} \rightarrow & \text{ continue}
\end{aligned}
$$

To evaluate the goodness of a policy, we can run several simulations (using various random seeds). Assuming boundedness of the reward function, a small number of simulations will be sufficient to evaluate the goodness of a specific parameter setting β. We could then use a gradient descent procedure to optimize β. When doing so, it is advisable to fix the random seeds, so runs for different values of β are "better comparable." (See Ng and Jordan's paper in UAI 2000 on "Pegasus.")

In general, it will be hard to find succinct policy parameterizations that include a near-optimal policy. Moreover, the optimization problem could have many local optima.

4.1 Hallway Example

Picture 4 illustrates a hallway with six areas each containing a hidden reward. It is clear that the goal here is to go to the area with the +1000 reward. However, since the hallway problem is symmetric there is no way of knowing how we are turning, and hereby we risk to get a -1000 reward. In this case the optimal policy would be to go to one of the corners and receive a low reward/penalty which will give the orientation, and then go to the area with the +1000 reward.

Figure 4: Example with hallway where the grey areas have hidden rewards

However, when we only have partial observability, and can't distinguish between top and bottom, the optimal policy becomes to first walk down the hallway, then verify what reward we get in one of the alcoves. This will result in a belief state that gives us certainty about our orientation relative to the hallway. Then we move to the +1000 reward state.

5 2 Settings

In next lectures, we will consider two special settings, in which, despite the partial observability, we can still find optimal policies (relatively) efficiently.

1. Linear Systems + Quadratic Cost → Separation Principle
2. Multi-armed Bandit

5.1 Separation Principle

When the optimal policy of a partially observable system can be decomposed into (i) an optimal state estimator, and (ii) an optimal controller for the same system when it is fully observable, then the "separation principle" applies to this system.

5.2 Multi-armed Bandit

Example: 2 Armed Slot Machine

$$\text{2 arms} \to \text{2 actions} \left\langle \begin{array}{l} \text{Pull 1} \to \text{Payoff} = \begin{cases} 1, P = \theta_1 \\ 0, P = 1-\theta_1 \end{cases} \\ \text{Pull 2} \to \text{Payoff} = \begin{cases} 1, P = \theta_2 \\ 0, P = 1-\theta_2 \end{cases} \end{array} \right.$$

Each bandit has pay probability of θ_1 and θ_2 respectively, which are both unknown. In this example the information state would be,

$$I = \left(\begin{array}{l} \text{successful pulls of bandit 1,} \\ \text{unsuccessful pulls of bandit 1,} \\ \text{successful pulls of bandit 2,} \\ \text{unsuccessful pulls of bandit 2} \end{array} \right)$$

Intuitively you would think that this problem would grow with the number of arms. But when you pull one arm you don't affect the state of other arms. Therefore there is only one state (assuming the problem is fully observable), and the problem can be solved by finding an optimal policy on one arm using dynamic programming.