Lecture 25: Markov Decision Processes

Scribe: *David Millard*

There is a recitation on smoothing on Tuesday 2018-09-04.

1 Review of MDPs

Recall that a Markov decision process (MDP) is defined as (S, A, T, R) where

- S is a discrete state space.
- A is a discrete action space.
- $T : S \times A \rightarrow \Pi(S)$ is a transition function giving $p(s'|a, s)$, the probability of transition to state s' given action a and previous state s.
- $R : S \rightarrow \mathbb{R}$ is function mapping states to numerical rewards.

Given a policy π, we can compute the value of being in a given state s for a finite time horizon

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{T} R(s_t) \middle| \pi_t\right] \tag{1}$$

or for an infinite time horizon

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) \middle| \pi_t\right] \tag{2}$$

2 Example: Grid world

We consider the simple grid world given in Figure 1.

-0.04	-0.04	-0.04	$+1$
-0.04		-0.04	-1
-0.04	-0.04	-0.04	-0.04

Figure 1: Grid world

Our goal is to produce an optimal policy

$$\pi^*(s) = \underset{\pi}{\operatorname{argmax}}\, V^{\pi}(s) \tag{3}$$

Note

Why have a policy at all? Why not generate a plan and then execute it?

A policy gives you the option for closed loop control. If we end up in a state that deviates from our plan's expectations, we can still recover and act optimally.

We describe the value function for an optimal policy as follows

$$V^*(s_t) = \max_a \left[R(s_t) + \gamma \sum_{s_{t+1}} P(S_{t+1}|s_t, a_t) \underbrace{V^*(s_{t+1})}_{\substack{\text{expected} \\ \text{future} \\ \text{rewards}}} \right] \tag{4}$$

Intuitively, an optimal policy should optimize the immediate reward and the discounted expected future rewards.

Note

We have n equations and n unknowns, so can't we solve the system of equations?

No, because $\max_{a_t}$ is nonlinear, we must solve some other way.

3 Value iteration algorithm

Algorithm 1: Value iteration algorithm

Result: $\pi*$ an optimal policy (within ε)
$V_0(s) \leftarrow 0$;
repeat
  for $s \in S$ **do**
    $V_{t+1}(s) \leftarrow \max_a \left[R(s) + \sum_{s'} P(s'|s,a) V_t(s')\right]$;
  end for
until $\max_s |V_{t+1}(s) - V_t(s)| < \varepsilon$;

Applying the value iteration algorithm to our grid world example, we get

−0.04	−0.04	−0.04	+1
−0.04		−0.04	−1
−0.04	−0.04	−0.04	−0.04

(a) $t = 1$

−0.08	−0.08	−0.68	+1
−0.08		−0.08	−1
−0.08	−0.08	−0.08	−0.08

(b) $t = 2$

0.81	0.89	0.92	+1
0.76		0.66	−1
0.71	0.66	0.61	0.39

(c) $t = T$

Figure 2: Value iteration on grid world

Note

What if -0.04 approaches 0? The robot will be extremely conservative, and run into the wall instead of attempting to move near -1.

What if -0.04 approaches -5? The robot will go straight towards either -1 or $+1$, since either are better than staying still.