

Lecture 20: Markov Decision Processes: Control

10:30 AM, Apr 9, 2009

Contents

1 Overview	1
2 Definitions and An Example	1
2.1 Markov Decision Processes	1
2.2 Example	2
3 State Values	2
4 Action Values	4
5 Value Iteration	4
5.1 Example (cont'd)	5
6 Policy Iteration	6
6.1 Policy Evaluation	6
6.2 Policy Improvement	6
6.3 Example (cont'd)	7

1 Overview

In this lecture, we extend our discussion of Markov reward processes to Markov decision processes (MDP). For MDPs, we pose and solve the control problem—the search for an optimal policy. Specifically, we describe value iteration and policy iteration, two dynamic programming algorithms that are used to compute an optimal policy in an MDP.

2 Definitions and An Example

What follows generalizes the definition of Markov reward processes presented in Lecture 19.

2.1 Markov Decision Processes

An agent operating in a Markovian environment transitions from state to state, in general making decisions and obtaining rewards along the way, as follows: at time t ,

1. state is s_t
2. choose action a_t
3. receive reward r_t
4. transition to state s_{t+1} with probability $P[s_{t+1} | s_t, a_t]$

Markov decision processes (MDPs) model such agent-environment interactions. A (discrete-time) **Markov decision process** is a tuple $\langle S, A, R, P \rangle$, where time is discrete: *i.e.*, $t \in T = \{0, 1, \dots\}$, and

- S is a finite set of states ($s \in S$)
- A is a finite set of actions ($a \in A$)
- $R : S \times A \rightarrow \mathbb{R}$ is a reward function
- $P : S \times A \rightarrow \Delta(S)$ is a probability transition function (or matrix)
 $\Delta(S)$ is the set of probability distributions over S

2.2 Example

Each TAC Travel flight “auction” (viewed in isolation) is an example of an MDP. Let us simplify one such auction and model it as an MDP.

The state is defined in terms of the price of the flight and the time remaining until the end of the auction. Specifically, the state space is the cross product of the set of possible prices, say $\mathcal{P} = \{150, 160, \dots, 590, 600\}$ and the time, which we assume varies discretely from $t = 0$ through time $T = 30$, unioned with a designated state END. Let p_t denote the price at time t .

The set of possible actions A includes *buy now* (B) and *(re)consider later* (C). Rewards depend on the flight’s valuation. Assuming v represents this valuation, $R(p_t, B) = v - p_t$ and $R(p_t, C) = 0$, for all p_t ; in addition, $R(\text{END}, a) = 0$, for all actions $a \in A$. Finally, transition probabilities depend on states and actions: for all prices $p \in \mathcal{P}$, actions $a \in A$, and times $t \in \{0, \dots, T\}$,

$$\begin{aligned}
 P[\text{END} | p_t = p, a_t = B] &= 1.0 \\
 P[p_{t+1} = p + 10 | p_t = p, a_t = C] &= 0.5 \\
 P[p_{t+1} = p - 10 | p_t = p, a_t = C] &= 0.5 \\
 P[\text{END} | \text{END}, a_t = a] &= 1.0 \\
 P[\text{END} | p_T = p, a_T = a] &= 1.0
 \end{aligned}$$

At state p_t , what is the optimal action?

3 State Values

A **policy** is a map from states to actions: *i.e.*, $\pi : S \rightarrow A$. The state value $V^\pi(s)$ associated with state s under policy π is defined as the expected reward that is accrued from state s on by following

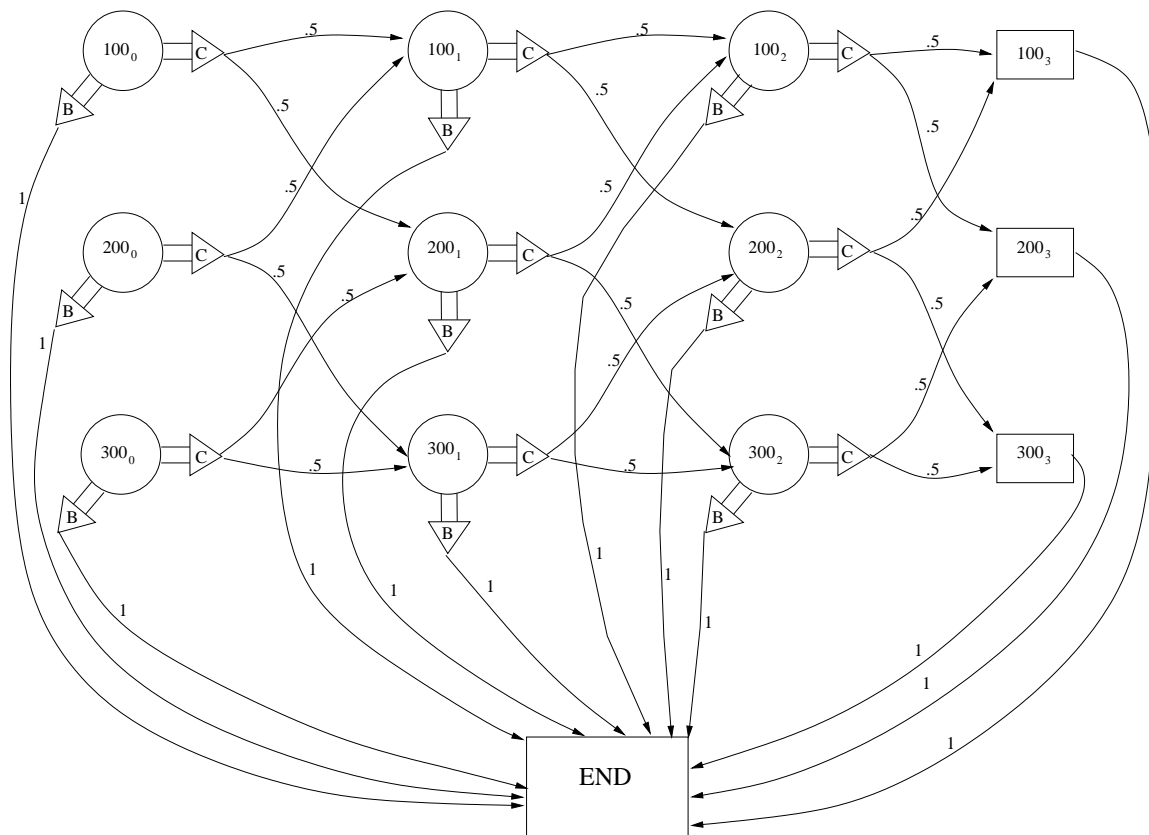


Figure 1: TAC Travel Flight Auctions as an MDP: States are indicated by circles. Fat arrows indicate actions; they are labeled with rewards. Skinny arrows indicate transitions; they are labeled with probabilities.

policy π . By Bellman’s theorem (for state values), this state value can be equivalently expressed as the sum of the immediate reward obtained by taking action $\pi(s)$ in state s and the discounted expected value of the next state s' , assuming the policy π is followed from state s' on:

$$V^\pi(s) = R(s, \pi(s)) + \gamma \mathbb{E}[V^\pi(s')] \tag{1}$$

Policy π dominates policy $\hat{\pi}$ (notation $\pi \gg \hat{\pi}$) iff $V^\pi(s) \geq V^{\hat{\pi}}(s)$ for all states $s \in S$. We seek an optimal policy: *i.e.*, π^* *s.t.* $\pi^* \gg \pi$, for all policies π . It suffices to restrict our attention to *deterministic, stationary* policies π , in which the same **pure** (*i.e.*, non-randomized) action is taken every time state s is visited. (Why?)

An optimal policy can be computed by solving Bellman’s **optimality** equations:

$$V(s) = \max_a \{R(s, a) + \gamma \mathbb{E}[V(s')]\} \tag{2}$$

These equations state that a state’s value is that which can be obtained by choosing the action that maximizes the sum of the immediate reward at the current state and the discounted expected value of the next state. As in the case of Markov reward processes, to find a solution to this system of ($|S|$) equations (with $|S|$ unknowns), we rely on Banach’s fixed point theorem. The optimal value function V^* is the unique solution to this system of equations.

Exercise: Show that the mapping implicit in Equation 2 is a contraction on (\mathbb{R}^S, L_∞) .

Given V^* , the optimal policy π^* maps state s into an optimal action, as follows:

$$\pi^*(s) \in \arg \max_a \{R(s, a) + \gamma \mathbb{E}[V^*(s')]\} \quad (3)$$

While the optimal value function V^* is unique, the optimal policy π^* need not be unique.

4 Action Values

The action value $Q^\pi(s, a)$ associated with state s and action a under policy π is defined as the expected reward that is accrued from state s on by following policy π , except at state s , where it is assumed that action a is taken instead of action $\pi(s)$. By Bellman's theorem (for action values), this value can be equivalently expressed as the sum of the immediate reward obtained by taking action a in state s and the discounted expected value of the next state s' , assuming the policy π is followed from state s' on:

$$Q^\pi(s, a) = R(s, a) + \gamma \mathbb{E}[V^\pi(s')] \quad (4)$$

$$= R(s, a) + \gamma \mathbb{E}[Q^\pi(s', \pi(s'))] \quad (5)$$

Restating Bellman's optimality equations in terms of action values yields:

$$Q(s, a) = R(s, a) + \gamma \mathbb{E}[V(s')] \quad (6)$$

$$= R(s, a) + \gamma \mathbb{E}[\max_a Q(s', a)] \quad (7)$$

As in the case of Markov reward processes, to find a solution to this system of $(|S \times A|)$ equations (with $|S \times A|$ unknowns), we rely on Banach's fixed point theorem. The optimal action-value function Q^* is the unique solution to this system of equations.

Exercise: Show that the mapping implicit in Equation 7 is a contraction on (\mathbb{R}^S, L_∞) .

Given Q^* , the optimal policy π^* maps state s into an optimal action, as follows:

$$\pi^*(s) \in \arg \max_a Q^*(s, a) \quad (8)$$

While the optimal action-value function Q^* is unique, the optimal policy π^* need not be unique.

5 Value Iteration

The value iteration algorithm, which is based on Equation 2, updates as follows:

$$V(s) \leftarrow \max_a \{R(s, a) + \gamma \sum_{s'} P[s' | s, a] V(s')\} \quad (9)$$

Equivalently,

$$Q(s, a) \leftarrow R(s, a) + \gamma \sum_{s'} P[s' | s, a] V(s') \quad (10)$$

$$V(s) \leftarrow \max_a Q(s, a) \quad (11)$$

The algorithm, which is depicted in Table 1, first computes the value of each state for all actions, and then sets each state's value to be the greatest value achieved among all courses of action. The actions that yield the optimal state values can be extracted as the optimal policy.

VALUE_ITERATION(MDP, γ, ϵ)	
Inputs	discount factor γ convergence test ϵ
Output	optimal state-value function V^*
Initialize	$V = 0$ and $V' = \infty$
<pre> while max_s V(s) - V'(s) > ϵ do 1. V' = V 2. for all s ∈ S (a) for all a ∈ A i. Q(s, a) = R(s, a) + $\gamma \sum_{s'} P[s' s, a]V(s')$ (b) V(s) = max_a Q(s, a) return V </pre>	

Table 1: Value Iteration *à la* Gauss-Seidel.

5.1 Example (cont'd)

The following tables depict the computation of state and action values and the optimal policy in a TAC flight auction, assuming 3 prices, namely \$100, \$200, and \$300, and 4 time steps, with $V = 500$ and $\gamma = 1$.

$Q(s, a)$	$t = 0$		$t = 1$		$t = 2$		$t = 3$	
	B	C	B	C	B	C	B	C
300	200	300	200	275	200	250	200	0
200	300	337.5	300	325	300	300	300	0
100	400	362.5	400	350	400	350	400	0

$V(s)$	$t = 0$	$t = 1$	$t = 2$	$t = 3$
300	300	275	250	200
200	337.5	325	300	300
100	400	400	400	400

$\pi(s)$	$t = 0$	$t = 1$	$t = 2$	$t = 3$
300	C	C	C	B
200	C	C	C/B	B
100	B	B	B	B

The optimal policy is fairly intuitive: It prescribes that an agent should buy if all time has elapsed, regardless of price. Similarly, an agent should be if ever the price hits the lower bound. Otherwise, if time remains and the price is not rockbottom, it is optimal to consider buying later, since there is some chance of seeing the price drop.

6 Policy Iteration

Policy iteration is a two-phase dynamic programming method for computing optimal policies in an MDP. The first phase, **policy evaluation**, computes the state values for the current (fixed) policy via Equation 1. The second phase, **policy improvement**, improves upon the current policy (whenever possible) in a greedy fashion. Policy improvement updates based on Equations 4 and 8.

In practice, value iteration is faster than policy iteration *per iteration*; however, policy iteration takes far fewer iterations to converge. One modified version of policy iteration does not wait for the policy evaluation phase of policy iteration to converge, and instead produces approximations of V^π . This modification leads to substantial speedups in the runtime of policy iteration.

6.1 Policy Evaluation

Policy evaluation in Markov decision processes computes state values given some policy exactly as state values are evaluated in Markov reward processes:

$$V^\pi(s) \leftarrow R(s, \pi(s)) + \gamma \sum_{s'} P[s' | s, \pi(s)] V^\pi(s') \quad (12)$$

6.2 Policy Improvement

The convergence of policy iteration follows from the **policy improvement theorem** and the **one-shot deviation principle**. The former states that greedy improvements (i.e., improvements in immediate rewards) lead to improved policies. Conversely, the latter states: if there are no greedy improvements to a policy to be had, then the policy is optimal. Hence, by repeatedly improving a policy in a greedy fashion until no further improvements can be made, one arrives at the optimal policy. A proof of the one-shot deviation principle for Markov Decision Processes appears in Blackwell's paper entitled Discounted Dynamic Programming (*Annals of Mathematical Statistics*, 1965). Here is the formal statement of the policy improvement theorem.

Theorem: Given policies π_1 and π_2 , if $Q^{\pi_1}(s, \pi_2(s)) \geq Q^{\pi_1}(s, \pi_1(s))$ for all states $s \in S$, then $V^{\pi_2}(s) = Q^{\pi_2}(s, \pi_2(s)) \geq Q^{\pi_1}(s, \pi_1(s)) = V^{\pi_1}(s)$ for all $s \in S$.

Proof: (Sketch)

$$\begin{aligned} V^{\pi_1}(s) &= Q^{\pi_1}(s, \pi_1(s)) \\ &\leq Q^{\pi_1}(s, \pi_2(s)) \\ &= R(s, \pi_2(s)) + \gamma \mathbb{E}[V^{\pi_1}(s')] \\ &= R(s, \pi_2(s)) + \gamma \mathbb{E}[Q^{\pi_1}(s', \pi_1(s'))] \\ &\leq R(s, \pi_2(s)) + \gamma \mathbb{E}[Q^{\pi_1}(s', \pi_2(s'))] \\ &= R(s, \pi_2(s)) + \gamma \mathbb{E}[R(s', \pi_2(s')) + \gamma \mathbb{E}[V^{\pi_1}(s'')]] \\ &= R(s, \pi_2(s)) + \gamma \mathbb{E}[R(s', \pi_2(s'))] + \gamma^2 \mathbb{E}[V^{\pi_1}(s'')] = \dots = V^{\pi_2}(s) \end{aligned}$$

The policy improvement steps in the policy iteration algorithm are as follows:

$$Q^\pi(s, a) \leftarrow R(s, a) + \gamma \sum_{s'} P[s' | s, a] V^\pi(s') \quad (13)$$

$$\pi(s) \in \arg \max_a Q(s, a) \tag{14}$$

6.3 Example (cont'd)

The following tables depict the iterative computation of policies, state, and action values in one TAC flight auction. The flight's valuation is 500.

Initialization

π	$t = 0$	$t = 1$	$t = 2$	$t = 3$
300	<i>B</i>	<i>B</i>	<i>B</i>	<i>B</i>
200	<i>B</i>	<i>B</i>	<i>B</i>	<i>B</i>
100	<i>B</i>	<i>B</i>	<i>B</i>	<i>B</i>

Iteration 0

V^π	$t = 0$	$t = 1$	$t = 2$	$t = 3$
300	200	200	200	200
200	300	300	300	300
100	400	400	400	400

$Q^\pi(s, a)$	$t = 0$		$t = 1$		$t = 2$		$t = 3$	
	<i>B</i>	<i>C</i>	<i>B</i>	<i>C</i>	<i>B</i>	<i>C</i>	<i>B</i>	<i>C</i>
300	200	250	200	250	200	250	200	0
200	300	300	300	300	300	300	300	0
100	400	350	400	350	400	350	400	0

π	$t = 0$	$t = 1$	$t = 2$	$t = 3$
300	<i>C</i>	<i>C</i>	<i>C</i>	<i>B</i>
200	<i>C/B</i>	<i>C/B</i>	<i>C/B</i>	<i>B</i>
100	<i>B</i>	<i>B</i>	<i>B</i>	<i>B</i>

Iteration 1

V^π	$t = 0$	$t = 1$	$t = 2$	$t = 3$
300	287.5	275	250	200
200	300	300	300	300
100	400	400	400	400

<p>POLICY_ITERATION(MDP, γ, ϵ)</p> <p>Inputs discount factor γ convergence test ϵ</p> <p>Output optimal policy π^*</p> <p>Initialize $\pi \neq \pi'$</p>
<p>while $\pi \neq \pi'$ do</p> <ol style="list-style-type: none"> 1. $\pi' = \pi$ 2. $V^\pi = \text{POLICY_EVALUATION}(\text{MDP}, \pi, \gamma, \epsilon)$ 3. $\pi = \text{POLICY_IMPROVEMENT}(\text{MDP}, V^\pi, \gamma)$ <p>return π</p>
<p>POLICY_EVALUATION(MDP, π, γ, ϵ)</p> <p>Inputs policy π discount factor γ convergence test ϵ</p> <p>Output state-value function V^π</p> <p>Initialize $V = 0$ and $V' = \infty$</p>
<p>while $\max_s V(s) - V'(s) > \epsilon$ do</p> <ol style="list-style-type: none"> 1. $V' = V$ 2. for all $s \in S$ <ol style="list-style-type: none"> (a) $V(s) = R(s, \pi(s)) + \gamma \sum_{s'} P[s' s, \pi(s)] V(s')$ <p>return V</p>
<p>POLICY_IMPROVEMENT(MDP, V, γ)</p> <p>Inputs value function V discount factor γ</p> <p>Output improved policy π</p>
<p>for all $s \in S$</p> <ol style="list-style-type: none"> 1. for all $a \in A$ <ol style="list-style-type: none"> (a) $Q(s, a) = R(s, a) + \gamma \sum_{s'} P[s' s, a] V(s')$ 2. $\pi(s) \in \arg \max_a Q(s, a)$ <p>return π</p>

Table 2: Policy Iteration.

```

MODIFIED_POLICY_ITERATION(MDP,  $\gamma$ )
  Inputs   discount factor  $\gamma$ 
  Output   optimal policy  $\pi^*$ 
  Initialize  $\pi \neq \pi'$ 

while  $\pi \neq \pi'$  do
  1.  $\pi' = \pi$ 
  2. for all  $s \in S$  /* approximate policy evaluation */
      (a)  $V(s) = R(s, \pi(s)) + \gamma \sum_{s'} P[s' | s, \pi(s)]V(s')$ 
  3. for all  $s \in S$  /* policy improvement */
      (a) for all  $a \in A$ 
          i.  $Q(s, a) = R(s, a) + \gamma \sum_{s'} P[s' | s, a]V(s')$ 
          (b)  $\pi(s) \in \arg \max_a Q(s, a)$ 
return  $\pi$ 
    
```

Table 3: Modified Policy Iteration

$Q^\pi(s, a)$	$t = 0$		$t = 1$		$t = 2$		$t = 3$	
	<i>B</i>	<i>C</i>	<i>B</i>	<i>C</i>	<i>B</i>	<i>C</i>	<i>B</i>	<i>C</i>
300	200	287.5	200	275	200	250	200	0
200	300	337.5	300	325	300	300	300	0
100	400	350	400	350	400	350	400	0

π	$t = 0$	$t = 1$	$t = 2$	$t = 3$
300	<i>C</i>	<i>C</i>	<i>C</i>	<i>B</i>
200	<i>C</i>	<i>C</i>	<i>C/B</i>	<i>B</i>
100	<i>B</i>	<i>B</i>	<i>B</i>	<i>B</i>

Iteration 2

V^π	$t = 0$	$t = 1$	$t = 2$	$t = 3$
300	300	275	250	200
200	337.5	325	300	300
100	400	400	400	400

$Q^\pi(s, a)$	$t = 0$		$t = 1$		$t = 2$		$t = 3$	
	<i>B</i>	<i>C</i>	<i>B</i>	<i>C</i>	<i>B</i>	<i>C</i>	<i>B</i>	<i>C</i>
300	200	300	200	275	200	250	200	0
200	300	337.5	300	325	300	300	300	0
100	400	362.5	400	350	400	350	400	0

π	$t=0$	$t=1$	$t=2$	$t=3$
300	<i>C</i>	<i>C</i>	<i>C</i>	<i>B</i>
200	<i>C</i>	<i>C</i>	<i>C/B</i>	<i>B</i>
100	<i>B</i>	<i>B</i>	<i>B</i>	<i>B</i>

As the new policy does not differ from the old, policy iteration has converged. Moreover, the current values of V^π represent the values of the optimal policy.