Decision Theory: Markov Decision Processes

CPSC 322 Lecture 33

March 31, 2006
Textbook §12.5

Lecture Overview

Recap

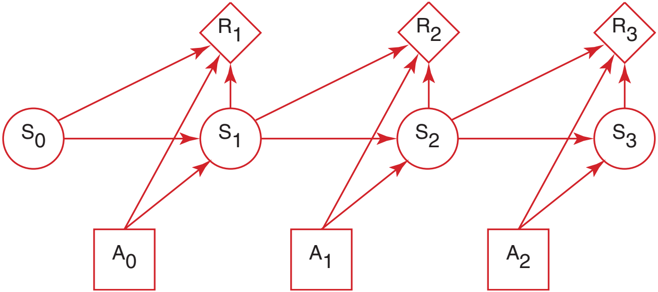
Rewards and Policies

Value Iteration

Asynchronous Value Iteration

Markov Decision Processes

- A Markov decision process augments a stationary Markov chain with actions and values:

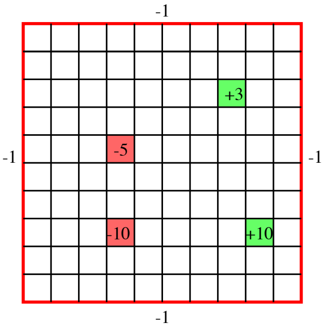


Markov Decision Processes

An MDP is defined by:

- set S of states.
- set A of actions.
- $P(S_{t+1}|S_t, A_t)$ specifies the dynamics.
- $R(S_t, A_t, S_{t+1})$ specifies the reward. The agent gets a reward at each time step (rather than just a final reward).
    - $R(s, a, s')$ is the reward received when the agent is in state s, does action a and ends up in state s'.

Example: Simple Grid World



- Actions: up, down, left, right.
- 100 states corresponding to the positions of the robot.
- Robot goes in the commanded direction with probability 0.7, and one of the other directions with probability 0.1.
- If it crashes into an outside wall, it remains in its current position and has a reward of -1.
- Four special rewarding states; the agent gets the reward when leaving.

Planning Horizons

The planning horizon is how far ahead the planner looks to make a decision.

- The robot gets flung to one of the corners at random after leaving a positive ($+10$ or $+3$) reward state.
  - the process never halts
  - infinite horizon
- The robot gets $+10$ or $+3$ entering the state, then it stays there getting no reward. These are absorbing states.
  - The robot will eventually reach the absorbing state.
  - indefinite horizon

Lecture Overview

Recap

Rewards and Policies

Value Iteration

Asynchronous Value Iteration

Rewards and Values

Suppose the agent receives the sequence of rewards $r_1, r_2, r_3, r_4, \ldots$. What value should be assigned?

- total reward $V = \sum_{i=1}^{\infty} r_i$
- average reward $V = \lim_{n \to \infty} \frac{r_1 + \cdots + r_n}{n}$
- discounted reward $V = \sum_{i=1}^{\infty} \gamma^{i-1} r_i$
  - γ is the discount factor
  - $0 \leq \gamma \leq 1$

Policies

- A stationary policy is a function:

$$\pi : S \rightarrow A$$

  Given a state s, $\pi(s)$ specifies what action the agent who is following π will do.
- An optimal policy is one with maximum expected value
  - we'll focus on the case where value is defined as discounted reward.
- For an MDP with stationary dynamics and rewards with infinite or indefinite horizon, there is always an optimal stationary policy in this case.

Value of a Policy

- $Q^\pi(s,a)$, where a is an action and s is a state, is the expected value of doing a in state s, then following policy π.
- $V^\pi(s)$, where s is a state, is the expected value of following policy π in state s.
- Q^π and V^π can be defined mutually recursively:

$$\begin{aligned} V^\pi(s) &= Q^\pi(s,\pi(s)) \\ Q^\pi(s,a) &= \sum_{s'} P(s'|a,s)\left(r(s,a,s') + \gamma V^\pi(s')\right) \end{aligned}$$

Value of the Optimal Policy

- $Q^*(s, a)$, where a is an action and s is a state, is the expected value of doing a in state s, then following the optimal policy.
- $V^*(s)$, where s is a state, is the expected value of following the optimal policy in state s.
- Q^* and V^* can be defined mutually recursively:

$$
\begin{aligned}
Q^*(s,a) &= \sum_{s'} P(s'|a,s)\left(r(s,a,s') + \gamma V^*(s')\right) \\
V^*(s) &= \max_a Q^*(s,a) \\
\pi^*(s) &= \arg\max_a Q^*(s,a)
\end{aligned}
$$

Lecture Overview

Recap

Rewards and Policies

Value Iteration

Asynchronous Value Iteration

Value Iteration

- Idea: Given an estimate of the k-step lookahead value function, determine the $k+1$ step lookahead value function.
- Set V_0 arbitrarily.
  - e.g., zeros
- Compute Q_{i+1} and V_{i+1} from V_i:

$$\begin{aligned} Q_{i+1}(s,a) &= \sum_{s'} P(s'|a,s)\left(r(s,a,s') + \gamma V_i(s')\right) \\ V_{i+1}(s) &= \max_a Q_{i+1}(s,a) \end{aligned}$$

- If we intersect these equations at Q_{i+1}, we get an update equation for V:

$$V_{i+1}(s) = \max_a \sum_{s'} P(s'|a,s)\left(r(s,a,s') + \gamma V_i(s')\right)$$

Pseudocode for Value Iteration

procedure value_iteration(P, r, θ)
inputs:
  P is state transition function specifying $P(s'|a, s)$
  r is a reward function $R(s, a, s')$
  θ a threshold $\theta > 0$
returns:
  $\pi[s]$ approximately optimal policy
  $V[s]$ value function
data structures:
  $V_k[s]$ a sequence of value functions
begin
  for $k = 1 : \infty$
    for each state s
      $V_k[s] = \max_a \sum_{s'} P(s'|a, s)(R(s, a, s') + \gamma V_{k-1}[s'])$
    if $\forall s \ |V_k(s) - V_{k-1}(s)| < \theta$
      for each state s
        $\pi(s) = \arg\max_a \sum_{s'} P(s'|a, s)(R(s, a, s') + \gamma V_{k-1}[s'])$
      return π, V_k
end

Lecture Overview

Recap

Rewards and Policies

Value Iteration

Asynchronous Value Iteration

Asynchronous Value Iteration

- You don't need to sweep through all the states, but can update the value functions for each state individually.
  - This converges to the optimal value functions, if each state and action is visited infinitely often in the limit.
- You can either store $V[s]$ or $Q[s,a]$.
- This algorithm forms the basis of several reinforcement learning algorithms
  - how should an agent behave in an MDP if it doesn't know the transition probabilities and the reward function?

Asynchronous VI: storing $Q[s,a]$

- Repeat forever:
  - Select state s, action a;
  - $Q[s,a] \leftarrow \sum_{s'} P(s'|s,a) \left(R(s,a,s') + \gamma \max_{a'} Q[s',a'] \right);$

Pseudocode for Asynchronous Value Iteration

procedure asynchronous_value_iteration(P, r)
inputs:
    P is state transition function specifying $P(s'|a, s)$
    r is a reward function $R(s, a, s')$
returns:
    π approximately optimal policy
    Q value function
data structures:
    real array $Q[s, a]$
    action array $\pi[s]$
begin
    repeat
        select a state s
            select an action a
                $Q[s, a] = \sum_{s'} P(s'|a, s)(R(s, a, s') + \gamma \max_{a'} Q[s', a'])$
    until some stopping criteria is true
    for each state s
        $\pi[s] = \arg\max_a Q[s, a]$
    return π, Q
end