Agents as Processes

Agents carry out actions:

- forever **infinite horizon**
- until some stopping criteria is met **indefinite horizon**
- finite and fixed number of steps **finite horizon**

Decision-theoretic Planning

What should an agent do when

- it gets rewards (and punishments) and tries to maximize its rewards received
- actions can be stochastic; the outcome of an action can't be fully predicted
- there is a model that specifies the (probabilistic) outcome of actions and the rewards
- the world is fully observable

World State

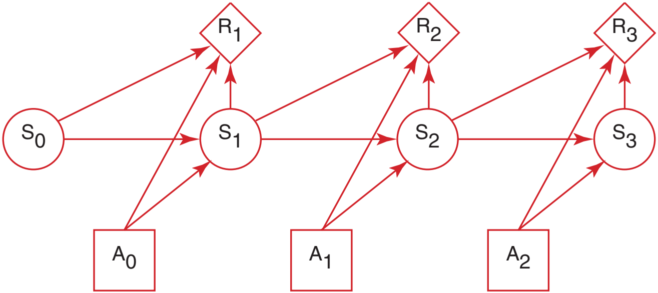
- The world state is the information such that if you knew the world state, no information about the past is relevant to the future. Markovian assumption.
- Let S_i be the state at time i

$$P(S_{t+1}|S_0, A_0, \ldots, S_t, A_t) = P(S_{t+1}|S_t, A_t)$$

  $P(s'|s, a)$ is the probability that the agent will be in state s' immediately after doing action a in state s.
- The dynamics is stationary if the distribution is the same for each time point.

Decision Processes

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

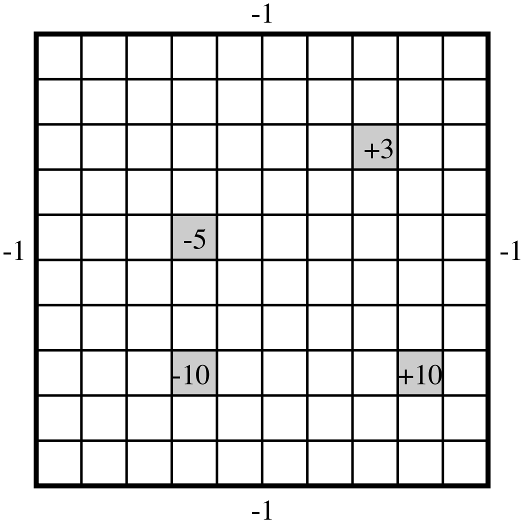


Markov Decision Processes

For an MDP you specify:

- 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 expected reward received when the agent is in state s, does action a and ends up in state s'.

Example: Simple Grid World



Grid World Model

- 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

Information Availability

What information is available when the agent decides what to do?

- fully-observable MDP the agent gets to observe S_t when deciding on action A_t.
- partially-observable MDP (POMDP) the agent has some noisy sensor of the state. It needs to remember its sensing and acting history.

[This lecture only considers FOMDPs]

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 $V = \lim_{n \to \infty} (r_1 + \cdots + r_n)/n$
- discounted reward $V = r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \cdots$
  γ is the discount factor $0 \leq \gamma \leq 1$.

Properties of the Discounted Reward

- The discounted value of rewards $r_1, r_2, r_3, r_4, \ldots$ is

$$\begin{aligned} V &= r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \cdots \\ &= r_1 + \gamma(r_2 + \gamma(r_3 + \gamma(r_4 + \ldots))) \end{aligned}$$

- If $V(t)$ is the value obtained from time step t

$$V(t) = r_t + \gamma V(t+1)$$

- $1 + \gamma + \gamma^2 + \gamma^3 + \cdots = 1/(1-\gamma)$
  Therefore $\dfrac{\text{minimum reward}}{1-\gamma} \le V(t) \le \dfrac{\text{maximum reward}}{1-\gamma}$
- We can approximate V with the first k terms, with error:

$$V - (r_1 + \gamma r_2 + \cdots + \gamma^{k-1} r_k) = \gamma^k V(k+1)$$

Policies

- A stationary policy is a function:

$$\pi : S \to A$$

  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 discounted reward.
- For a fully-observable MDP with stationary dynamics and rewards with infinite or indefinite horizon, there is always an optimal stationary policy.

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:

$$\begin{aligned} Q^\pi(s, a) &= \\ V^\pi(s) &= \end{aligned}$$

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:

$$
\begin{aligned}
Q^*(s, a) &= \\
V^*(s) &= \\
\pi^*(s) &=
\end{aligned}
$$

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.
- Compute Q_{i+1}, V_{i+1} from V_i.
- This converges exponentially fast (in k) to the optimal value function.

  The error reduces proportionally to $\dfrac{\gamma^k}{1-\gamma}$

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 $Q[s, a]$.

Asynchronous VI: storing $V[s]$

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

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

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

Policy Iteration

- Set π_0 arbitrarily, let $i = 0$
- Repeat:
  - evaluate $Q^{\pi_i}(s, a)$
  - let $\pi_{i+1}(s) = argmax_a Q^{\pi_i}(s, a)$
  - set $i = i + 1$
- until $\pi_i(s) = \pi_{i-1}(s)$

Policy Iteration

- Set π_0 arbitrarily, let $i = 0$
- Repeat:
  - evaluate $Q^{\pi_i}(s, a)$
  - let $\pi_{i+1}(s) = argmax_a Q^{\pi_i}(s, a)$
  - set $i = i + 1$
- until $\pi_i(s) = \pi_{i-1}(s)$

Evaluating $Q^{\pi_i}(s, a)$ means finding a solution to a set of $|S| \times |A|$ linear equations with $|S| \times |A|$ unknowns.

It can also be approximated iteratively.

Modified Policy Iteration

Set $\pi[s]$ arbitrarily;
Set $Q[s, a]$ arbitrarily;
Repeat forever:

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