Decision making in stochastic environments

CE417: Introduction to Artificial Intelligence
Sharif University of Technology
Spring 2015

Soleymani

Sequential decision problems

- $S$: Set of states
- $A$: Set of actions

- Goal: Finding a mapping from states to actions in order to maximize reward signal
  - Main difficulty: delayed reward
Environment properties

- Deterministic vs. stochastic
  - Stochastic: stochastic reward & transition

- Known vs. unknown
  - Unknown: Agent doesn't know the precise results of its actions before doing them
    - Reinforcement learning

- Fully observable vs. partially observable
  - Partially observable: Agent doesn't necessarily know all about the current state
    - POMDPs
Deterministic environment

- Deterministic
  - Transition and reward functions

- At time $t$:
  - Agent observes state $s_t \in S$
  - Then chooses action $a_t \in A$
  - Then receives reward $r_t$, and state changes to $s_{t+1}$

- How to choose action $a_t$ in state $s_t$ that maximizes the utility (return)?

\[ R_t = r_t + r_{t+1} + r_{t+2} + \cdots \]
Example: Robot grid world

- Deterministic reward and transition
- Optimal policy
Stochastic environment

- Stochastic environment
  - Stochastic transition and/or reward

- How to choose a policy that maximizes the expected return:

  \[ E[R_t] = E[r_t + r_{t+1} + r_{t+2} + \cdots] \]

  starting from state \( s_t \)?
Markov Decision Process (MDP)

- Markovian transition model

\[ P(s_{t+1}, r_t | s_t, a_t, r_{t-1}, s_{t-1}, a_{t-1}, r_{t-2}, ...) = P(s_{t+1}, r_t | s_t, a_t) \]

- Markov Decision Process: A sequential decision problem for a fully observable, stochastic environment with a Markovian transition model and additive rewards
MDP: Recycling Robot example

- \( S = \{\text{high, low}\} \)
- \( A = \{\text{search, wait, recharge}\} \)
  - \( A(\text{high}) = \{\text{search, wait}\} \)
  - \( A(\text{low}) = \{\text{search, wait, recharge}\} \)
- \( R_{\text{search}} > R_{\text{wait}} \)

\[
P(s_{t+1} = \text{high} | s_t = \text{high}, a_t = \text{search})
\]

Available actions in the ‘high’ state

[Sutton & Barto Book]
Example

- Intended outcome of actions occurs with probability 0.8 and with probability 0.2 the agent move at right $+90$ and $-90$ of the intended action.

- Two terminal states with reward $+1$ and $-1$

- Rewards of the other actions (except to those leading to $+1$ and $-1$ squares) are $-0.04$. 
Policy

- Policy shows what the agent should do for any state that it might reach.

- The quality of policy is measured by the expected utility of the possible environment histories generated by that policy:

\[ E[r_t + r_{t+1} + r_{t+2} + \cdots | s_t = s, \pi] \]

- Optimal policy: yields the highest expected utility.
  - For every possible state \( s \in S \) shows an action that maximizes the expected return.
Utility

- **Finite horizon vs. infinite horizon**
  - **Finite horizon**: after a fixed time $N$ nothing matters
    \[ U[s_0, ... s_N] = U[s_0, ... s_{N+k}] \forall k > 0 \]
  - **Infinite horizon**: no fixed time limit

- **Additive rewards**
  \[ U[s_0, ... s_N] = E[r_0 + r_1 + ... + r_N] \]

- **Discounted rewards**
  \[ U[s_0, ... s_N] = E[r_0 + \gamma r_1 + ... + \gamma^N r_N], \quad 0 < \gamma < 1 \]

We consider infinite horizon utility for its simplicity.
Discounted rewards

- The discount factor $\gamma$ describes the preference of an agent for current rewards over future rewards.
  - When $\gamma$ is close to 0, rewards in the distant future are insignificant.
  - When $\gamma$ is 1, discounted rewards are equivalent to additive rewards.

Why discounted rewards?

- if the environment does not contain a terminal state (or agent never reaches one) then undiscounted additive rewards will generally be infinite.
- Discounted rewards is a solution to this problem

$$U[s_0, s_1, \ldots] = \sum_{t=0}^{\infty} \gamma^t r_t \leq \sum_{t=0}^{\infty} \gamma^t r_{\text{max}} = \frac{r_{\text{max}}}{1 - \gamma}$$
Utility (or state-value) function for policy $\pi$

- Given a policy $\pi$, expected utility (called value function) is defined as:

$$V^\pi(s) = E\{\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s, \pi\}$$

- $V^\pi(s)$: How good for the agent to be in the state $s$ when its policy is $\pi$
Recursive definition for $U^\pi(S)$

\[
V^\pi(s) = E\{\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t = s, \pi\}
\]

\[
= E\{r_t + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, \pi\}
\]

\[
= \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} (R^a_{ss'} + \gamma E\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_{t+1} = s', \pi\})
\]

Bellman Equations

\[
V^\pi(s) = \sum_a \pi(s,a) \sum_{s'} P^a_{ss'} (R^a_{ss'} + \gamma V^\pi(s'))
\]
Example

- Deterministic example

\[ V^\pi(s) = \sum_{s'} p^{\pi(s)}_{ss'} \left( R^{\pi(s)}_{ss'} + \gamma V^\pi(s') \right) \]
Grid-world example

Actions that move the agent outside of the grid:
- location unchanged
- reward -1

In state $A$, All actions $\rightarrow A'$ and their rewards is $10$
In state $B$, All actions $\rightarrow B'$ and their rewards is $5$

Reward of all other actions is $0$

$V^\pi$ for random policy

$\gamma = 0.9$
Optimal policy

- Policy $\pi$ is better than (or equal to) $\pi'$ (i.e. $\pi \geq \pi'$) iff
  \[ V^\pi(s) \geq V^{\pi'}(s), \quad \forall s \in S \]

- **Optimal policy** $\pi^*$ is better than (or equal to) all other policies ($\forall \pi, \pi^* \geq \pi$)

- For any MDP, a deterministic optimal policy exists!
  - $V^{\pi^*}(s)$ will be abbreviated as $V^*(s)$

- **Optimal policy** $\pi^*$:
  \[ \pi^*(s) = \arg\max_\pi V^\pi(s), \quad \forall s \in S \]
Optimal policy

\[ \pi^*(s) = \arg\max_{\pi} V^\pi(s) \]

\[ \pi^*(s) \] shows optimal action in state \( s \)

- Optimal policy in a state is independent of the initial state (and other previous states):

\[ \pi^*(s) = \arg\max_{a \in A(s)} \sum_{s'} P_{ss'}^a \left( R_{ss'}^a + \gamma V^{\pi^*}(s') \right) \]
Optimal policies share the same optimal state-value function:

\[ V^*(s) = \max_{\pi} V^\pi(s), \quad \forall s \in S \]

\[ V^*(s) = \max_{a \in A(s)} \sum_{s'} P_{ss'}^a \left( R_{ss'}^a + \gamma V^*(s') \right) \]
Optimal policy: example 1 (deterministic env.)
Optimal policy: example 2

a) gridworld

b) $V^*$

c) $\pi^*$
Optimal policy: example 3

<table>
<thead>
<tr>
<th></th>
<th>0.812</th>
<th>0.868</th>
<th>0.918</th>
<th>+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.762</td>
<td>0.660</td>
<td>-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.705</td>
<td>0.655</td>
<td>0.611</td>
<td>0.388</td>
<td></td>
</tr>
</tbody>
</table>

Optimal policy
Main steps in solving Bellman optimality equations

- Two kinds of steps, which are repeated in some order for all the states until no further changes take place

\[
\pi(s) = \arg\max_{a \in \mathcal{A}(s)} \left\{ \sum_{s'} \mathcal{P}_{ss'}^a (R_{ss'}^a + \gamma V^\pi(s')) \right\}
\]

\[
V^\pi(s) = \sum_{s'} \mathcal{P}_{ss'}^{\pi(s)} (R_{ss'}^{\pi(s)} + \gamma V^\pi(s'))
\]

- There are variants of algorithms
  - Order of the above steps
  - One can also do them for all states at once, or state by state, or more often to some states than others.
Value Iteration algorithm

1) Initialize $V(s)$ arbitrarily
2) Repeat until convergence
   for $s \in S$
   
   
   $$V(s) \leftarrow \max_{a \in \mathcal{A}(s)} \sum_{s'} p_{ss'}^a (r_{ss'}^a + \gamma V(s'))$$

$V(s)$ converges to $V^*(s)$

Dynamic programming
If \( \max_{s \in S} |V^{old}(s) - V(s)| < \epsilon \), then the value of the greedy policy differs from the optimal policy by no more than \( \frac{2\epsilon\gamma}{1-\gamma} \).

Value Iteration

- Needs complete knowledge of the transitions and reward function
  - It is model-based
- It is time and memory expensive