Making Complex Decisions
Chapter 17(1-2)

Building a policy

• Specify a solution for any initial state
  – Construct a policy that outputs the best action for any state
    • policy = \( \pi \)
    • policy in state \( s = \pi(s) \)
  – Complete policy covers all potential input states
  – Optimal policy, \( \pi^* \), yields the highest expected utility
    • Why expected?
      – Transitions are stochastic
Using a policy

- An agent in state $s$
  - $s$ is the percept available to agent
    $\pi^*(s)$ outputs an action that maximizes expected utility

- The policy is a description of a simple reflex

Striking a balance

- Different policies demonstrate balance between risk and reward
  - Only interesting in stochastic environments (not deterministic)
  - Characteristic of many real-world problems

- Building the optimal policy is the hard part!
Attributes of optimality

- We wish to find policy that maximizes the utility of agent during lifetime
  - Maximize $U([s_0, s_1, s_2, \ldots, s_n])$

- But is length of lifetime known?
  - Finite horizon – number of state transitions is known
    - After timestep $N$, nothing matters
  - $U([s_0, s_1, s_2, \ldots, s_n]) = U([s_0, s_1, s_2, \ldots, s_n, s_{n+1}, s_{n+k}])$ for all $k>0$
  - Infinite horizon – always opportunity for more state transitions

Time horizon

- Consider spot (3, 1)
  - Let horizon = 3
  - Let horizon = 8
  - Let horizon = 20
  - Let horizon = inf
  - Does $\pi^*$ change?

- Nonstationary optimal policy
Evaluating state sequences

• Additive Rewards
  \[ U[(a, b, c, \ldots)] = R(a) + R(b) + R(c) + \ldots \]

• Discounted Rewards
  \[ U[(a, b, c, \ldots)] = R(a) + \gamma R(b) + \gamma^2 R(c) + \ldots \]
  \( \gamma \) is the discount factor between 0 and 1
  • What does this mean?

Evaluating a policy

• Each policy, \( \pi \), generates multiple state sequences
  – Uncertainty in transitions according to \( T(s, a, s') \)

• Policy value is an expected sum of discounted rewards observed over all possible state sequences
  \[ \pi^* = \arg\max_{\pi} \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi \right] \]
Building an optimal policy

• Value Iteration
  – Calculate the utility of each state
  – Use the state utilities to select an optimal action in each state
  – Your policy is simple – go to the state with the best utility
  – Your state utilities must be accurate
    • Through an iterative process you assign correct values to the state utility values

Utility of states

• The utility of a state $s$ is…
  – the expected utility of the state sequences that might follow it
    • The subsequent state sequence is a function of $\pi(s)$
• The utility of a state given policy $\pi$ is…

$$U^\pi(s) = E \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi, s_0 = s \right]$$
Restating the policy

- Previous slide said you go to state with highest utility
- Actually…
  - Go to state with maximum expected utility
    - Reachable state with highest utility may have low probability of being obtained
    - Function of available actions, transition function, resulting states

Putting pieces together

- We said the utility of a state was:

\[ U^*(s) = E \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi, s_0 = s \right] \]

- The policy is maximum expected utility

\[ \pi^*(s) = \arg\max_a \sum_{s'} T(s, a, s')U(s') \]

- Therefore, utility of a state is the immediate reward for that state and expected utility of next state

\[ U(s) = R(s) + \gamma \max_a \sum_{s'} T(s, a, s')U(s') \]
Example

• Revisit 4x3 example
• Utility at cell (1, 1)

\[
U(1, 1) = -0.04 + \gamma \max \{ 0.8U(1, 2) + 0.1U(2, 1) + 0.1U(1, 1), \quad \text{(Up)} \\
0.9U(1, 1) + 0.1U(1, 2), \quad \text{(Left)} \\
0.9U(1, 1) + 0.1U(2, 1), \quad \text{(Down)} \\
0.8U(2, 1) + 0.1U(1; 2) + 0.1U(1, 1) \} \quad \text{(Right)}
\]

• Consider all outcomes of all possible actions to select best action and assign its expected utility to value of next-state in equation

Using Bellman Equations to solve MDPs

• Consider a particular MDP
  – \( n \) possible states
  – \( n \) Bellman equations (one for each state)
  – \( n \) equations have \( n \) unknowns (\( U(s) \) for each state)
    • Iterative technique to solve