ITCS 6150
Intelligent Systems

Lecture 19
Making Complex Decisions
Chapter 17
Imagine a robot with only local sensing

- Traveling from A to B
- Actions have uncertain results – might move at right angle to desired
- We want robot to “learn” how to navigate in this room

Sequential Decision Problem

Figure 17.1 (a) A simple $4 \times 3$ environment that presents the agent with a sequential decision problem. (b) Illustration of the transition model of the environment: the “intended” outcome occurs with probability 0.8, but with probability 0.2 the agent moves at right angles to the intended direction. A collision with a wall results in no movement. The two terminal states have reward +1 and −1, respectively, and all other states have a reward of −0.04.
Similar to 15-puzzle problem

**How is this similar and different from 15-puzzle?**

- Let robot position be the blank tile
- Keep issuing movement commands
- Eventually a sequence of commands will cause robot to reach goal

Our model of the world is incomplete
How about other search techniques

**Genetic Algorithms**

- Let each “gene” be a sequence of L, R, U, D
  - Length unknown
  - Poor feedback

---

**Figure 17.1** (a) A simple $4 \times 3$ environment that presents the agent with a sequential decision problem. (b) Illustration of the transition model of the environment: the “intended” outcome occurs with probability 0.8, but with probability 0.2 the agent moves at right angles to the intended direction. A collision with a wall results in no movement. The two terminal states have reward $+1$ and $-1$, respectively, and all other states have a reward of $-0.04$. 
Building a policy

How might we acquire and store a solution?

• Is this a search problem?
  – Isn’t everything?
• Avoid local mins
• Avoid dead ends
• Avoid needless repetition

**Key observation:** if the number of states is small, consider evaluating states rather than evaluating action sequences
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
Example solutions

Typos in book!

Figure 17.2  (a) An optimal policy for the stochastic environment with $R(s) = -0.04$ in the nonterminal states. (b) Optimal policies for four different ranges of $R(s)$. 
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
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

Assumption

• If I say I will prefer state a to state b tomorrow, I must also say I prefer state a to state b today
• State preferences are stationary

Additive Rewards

• \( U[(a, b, c, ...)] = R(a) + R(b) + R(c) + ... \)

Discounted Rewards

• \( U[(a, b, c, ...)] = R(a) + \gamma R(b) + \gamma^2 R(c) + ... \)
  - \( \gamma \) is the discount factor between 0 and 1
  - What does this mean?
Evaluating infinite horizons

How can we compute the sum of infinite horizon?

- $U((a, b, c, \ldots)) = R(a) + R(b) + R(c) + \ldots$
- If discount factor, $\gamma$, is less than 1

$$U_h([s_0, s_1, s_2, \ldots]) = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \sum_{t=0}^{\infty} \gamma^t R_{\text{max}} = \frac{R_{\text{max}}}{1 - \gamma}$$

- note $R_{\text{max}}$ is finite by definition of MDP
Evaluating infinite horizons

How can we compute the sum of infinite horizon?

- If the agent is guaranteed to end up in a terminal state eventually
  - We’ll never actually have to compare infinite strings of states
  - We can allow \( \gamma \) to be 1
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

$$\pi^* = \arg\max_{\pi} E \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi \right]$$
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...

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

Let $\gamma = 1$ and $R(s) = -0.04$

Notice:

- Utilities higher near goal reflecting fewer $-0.04$ steps in sum
Restating the policy

I had 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

\[ \pi^*(s) = \text{argmax}_a \sum_{s'} T(s, a, s') U(s') \]
Putting pieces together

We said the utility of a state was:

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

The policy is maximum expected utility

$$\pi^*(s) = \underset{\pi}{\text{argmax}} \sum_{s'} T(s, a, s') U(s')$$

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

$$U(s) = R(s) + \gamma \max_a \sum_{s'} T(s, a, s') U(s')$$
What a deal

**Much cheaper to evaluate:**

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

**Instead of:**

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

**Richard Bellman invented the top equation**

- Bellman equation (1957)
Example of Bellman Equation

Revisit 4x3 example

Utility at cell (1, 1)

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

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

  - $n$ equations and $n$ unknowns… I can solve this, right?
  - No, because of nonlinearity caused by $\text{argmax}(\ )$
  - We’ll use an iterative technique
Iterative solution of Bellman equations

- Start with arbitrary initial values for state utilities
- Update the utility of each state as a function of its neighbors

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

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

- Repeat this process until an equilibrium is reached
Bellman Update

- Iterative updates look like this
  \[ U_{i+1}(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} T(s, a, s') U_i(s') \]

- After infinite Bellman updates, we are guaranteed to reach an equilibrium that solves Bellman equations
- The solutions are unique
- The corresponding policy is optimal
  - Sanity check... utilities for states near goal will settle quickly and their neighbors in turn will settle
  - Information is propagated through state space via local updates
Policy Iteration

*Imagine someone gave you a policy*

- How good is it?
  - Assume we know $\gamma$ and $R$
  - Eyeball it?
  - Try a few paths and see how it works?
  - Let’s be more precise…
Policy iteration

**Checking a policy**

- Just for kicks, let’s compute a utility (at this particular iteration of the policy, \( i \)) for each state according to Bellman’s equation:

\[
U_i(s) = R(s) + \gamma \sum_{s'} T(s, \pi_i(s), s') U_i(s')
\]
Policy iteration

Checking a policy

- But we don’t know $U_i(s')$
- No problem
  - $n$ Bellman equations
  - $n$ unknowns
  - equations are linear
- We can solve for the $n$ unknowns in $O(n^3)$ time using standard linear algebra methods
Policy iteration

**Checking a policy**

- Now we know $U(s)$ for all $s$
- For each $s$, compute
  \[
  \max_a \sum_{s'} T(s, a, s') U[s']
  \]
  - This is the best action
  - If this action is different from policy, update the policy

\[
U_i(s) = R(s) + \gamma \sum_{s'} T(s, \pi_i(s), s') U_i(s')
\]