Markov Decision Process

• Components:
  – **States** $s$, beginning with initial state $s_0$
  – **Actions** $a$
    • Each state $s$ has actions $A(s)$ available from it
  – **Transition model** $P(s' | s, a)$
    • Markov assumption: the probability of going to $s'$ from $s$ depends only on $s$ and $a$ and not on any other past actions or states
  – **Reward function** $R(s)$
• **Policy** $\pi(s)$: the action that an agent takes in any given state
  – The “solution” to an MDP
Overview

• First, we will look at how to “solve” MDPs, or find the optimal policy when the transition model and the reward function are known.

• Next time, we will consider reinforcement learning, where we don’t know the rules of the environment or the consequences of our actions.
Example 1: Game show

- A series of questions with increasing level of difficulty and increasing payoff
- Decision: at each step, take your earnings and quit, or go for the next question
  - If you answer wrong, you lose everything

1. Q1: Correct: $61,100
   - Incorrect: $0
   - Quit: $100
2. Q2: Correct: $100,000
   - Incorrect: $0
   - Quit: $1,100
3. Q3: Correct: $50,000
   - Incorrect: $0
   - Quit: $11,100
4. Q4: Correct: $61,100
   - Incorrect: $0
   - Quit: $11,100
Example 1: Game show

- Consider $50,000 question
  - Probability of guessing correctly: 1/10
  - Quit or go for the question?
- What is the expected payoff for continuing?
  \[ 0.1 \times 61,100 + 0.9 \times 0 = 6,110 \]
- What is the optimal decision?

<table>
<thead>
<tr>
<th>Question</th>
<th>Correct</th>
<th>Incorrect</th>
<th>Quit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td></td>
<td>$0</td>
<td>$100</td>
</tr>
<tr>
<td>Q2</td>
<td></td>
<td>$0</td>
<td>$1,100</td>
</tr>
<tr>
<td>Q3</td>
<td></td>
<td>$0</td>
<td>$11,100</td>
</tr>
<tr>
<td>Q4</td>
<td></td>
<td>$0</td>
<td>$61,100</td>
</tr>
</tbody>
</table>

Correct: $61,100
Incorrect: $0
Quit: $100
Quit: $1,100
Quit: $11,100
Quit: $61,100
Example 1: Game show

- What should we do in Q3?
  - Payoff for quitting: $1,100
  - Payoff for continuing: $0.5 \times 11,100 = $5,550

- What about Q2?
  - $100 for quitting vs. $4,162 for continuing

- What about Q1?
  - $100 for quitting vs. $3,746 for continuing

U = $11,100

9/10
Correct

Incorrect: $0
Quit: $100

U = $4,162

9/10
Correct

Incorrect: $0
Quit: $1,100

U = $5,550

3/4
Correct

Incorrect: $0
Quit: $11,100

U = $11,100

1/10
Correct: $61,100

Incorrect: $0
Quit: $11,100
Example 2: Grid world

Transition model:

$R(s) = -0.04$ for every non-terminal state
Example 2: Grid world

Optimal policy when \( R(s) = -0.04 \) for every non-terminal state
Example 2: Grid world

- Optimal policies for other values of $R(s)$:

\[
\begin{align*}
R(s) &< -1.6284 \\
-0.4278 &< R(s) < -0.0850 \\
-0.0221 &< R(s) < 0 \\
R(s) &> 0
\end{align*}
\]
Markov Decision Process

- Components:
  - States $s$
  - Actions $a$
    - Each state $s$ has actions $A(s)$ available from it
  - Transition model $P(s' \mid s, a)$
    - Markov assumption: the probability of going to $s'$ from $s$ depends only on $s$ and $a$, and not on any other past actions and states
  - Reward function $R(s)$

- The solution:
  - Policy $\pi(s)$: mapping from states to actions
Partially observable Markov decision processes (POMDPs)

- Like MDPs, only state is not directly observable
  - States $s$
  - Actions $a$
  - Transition model $P(s' | s, a)$
  - Reward function $R(s)$
  - Observation model $P(e | s)$

- We will only deal with fully observable MDPs
  - Key question: given the definition of an MDP, how to compute the optimal policy?
Maximizing expected utility

• The optimal policy should maximize the expected utility over all possible state sequences produced by following that policy:

\[ \sum_{\text{state sequences starting from } s_0} P(\text{sequence}) U(\text{sequence}) \]

• How to define the utility of a state sequence?
  – Sum of rewards of individual states
  – Problem: infinite state sequences
Utilities of state sequences

• Normally, we would define the utility of a state sequence as the sum of the rewards of the individual states

• **Problem:** infinite state sequences

• **Solution:** *discount* the individual state rewards by a factor $\gamma$ between 0 and 1:

$$U([s_0, s_1, s_2, \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots$$

$$= \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \frac{R_{\text{max}}}{1 - \gamma} \quad (0 < \gamma < 1)$$

  – Sooner rewards count more than later rewards
  – Makes sure the total utility stays bounded
  – Helps algorithms converge
Utilities of states

• Expected utility obtained by policy $\pi$ starting in state $s$:

$$U^\pi(s) = \sum_{\text{state sequences starting from } s} P(\text{sequence}) U(\text{sequence})$$

• The “true” utility of a state, denoted $U(s)$, is the expected sum of discounted rewards if the agent executes an optimal policy starting in state $s$.

• Reminiscent of minimax values of states…
Finding the utilities of states

- What is the expected utility of taking action $a$ in state $s$?
  \[ \sum_{s'} P(s'| s, a)U(s') \]

- How do we choose the optimal action?
  \[ \pi^*(s) = \arg \max_{a \in A(s)} \sum_{s'} P(s'| s, a)U(s') \]

- What is the recursive expression for $U(s)$ in terms of the utilities of its successor states?
  \[ U(s) = R(s) + \gamma \max_a \sum_{s'} P(s'| s, a)U(s') \]
The Bellman equation

- Recursive relationship between the utilities of successive states:

\[ U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' | s, a) U(s') \]
The Bellman equation

• Recursive relationship between the utilities of successive states:

\[ U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'| s, a) U(s') \]

• For N states, we get N equations in N unknowns
  – Solving them solves the MDP
  – We could try to solve them through expectimax search, but that would run into trouble with infinite sequences
  – Instead, we solve them algebraically
  – Two methods: value iteration and policy iteration
Method 1: Value iteration

- Start out with every $U(s) = 0$
- Iterate until convergence
  - During the $i$th iteration, update the utility of each state according to this rule:

  $$U_{i+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a)U_i(s')$$

- In the limit of infinitely many iterations, guaranteed to find the correct utility values
  - In practice, don’t need an infinite number of iterations…
Value iteration

- What effect does the update have?

\[
U_{i+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) U_i(s')
\]
Method 2: Policy iteration

• Start with some initial policy $\pi_0$ and alternate between the following steps:
  
  – **Policy evaluation:** calculate $U^{\pi_i}(s)$ for every state $s$
  
  – **Policy improvement:** calculate a new policy $\pi_{i+1}$ based on the updated utilities

  $$\pi^{i+1}(s) = \arg \max_{a \in A(s)} \sum_{s'} P(s'|s,a) U^{\pi_i}(s')$$
Policy evaluation

• Given a fixed policy $\pi$, calculate $U^\pi(s)$ for every state $s$
• The Bellman equation for the optimal policy:

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'| s, a) U(s')$$

  – How does it need to change if our policy is fixed?

$$U^\pi(s) = R(s) + \gamma \sum_{s'} P(s'| s, \pi(s)) U^\pi(s')$$

  – Can solve a linear system to get all the utilities!
  – Alternatively, can apply the following update:

$$U_{i+1}(s) \leftarrow R(s) + \gamma \sum_{s'} P(s'| s, \pi_i(s)) U_i(s')$$