Markov Decision Processes
Review and Outline

- Adversarial Games
  - Minimax search
  - $\alpha-\beta$ search
  - Evaluation functions
  - Multi-player, non-0-sum

- Stochastic Games
  - Expectimax

- Markov Decision Processes
- Reinforcement Learning
Agents vs. Environment

- An **agent** is an entity that *perceives* and *acts*.

- A **rational agent** selects actions that *maximize its utility function*.

Deterministic vs. **stochastic**

*Fully observable* vs. partially observable
Non-Deterministic Search
Example: Grid World

- A maze-like problem
  - The agent lives in a grid
  - Walls block the agent’s path

- Noisy movement: actions do not always go as planned
  - 80% of the time, the action North takes the agent North (if there is no wall there)
  - 10% of the time, North takes the agent West; 10% East
  - If there is a wall in the direction the agent would have been taken, the agent stays put

- The agent receives rewards each time step
  - Small “living” reward each step (can be negative)
  - Big rewards come at the end (good or bad)

- Goal: maximize sum of rewards
Grid World Actions

Deterministic Grid World

Stochastic Grid World
An MDP is defined by:
- A set of states $s \in S$
- A set of actions $a \in A$
- A transition function $T(s, a, s')$
  - Probability that $a$ from $s$ leads to $s'$, i.e., $P(s' \mid s, a)$
  - Also called the model or the dynamics

For now, we give this as input to the agent
Markov Decision Processes

- An MDP is defined by:
  - A set of states \( s \in S \)
  - A set of actions \( a \in A \)
  - A transition function \( T(s, a, s') \)
    - Probability that \( a \) from \( s \) leads to \( s' \), i.e., \( P(s' \mid s, a) \)
    - Also called the model or the dynamics
  - A reward function \( R(s, a, s') \)
    - Sometimes just \( R(s) \) or \( R(s') \)

\[
\begin{align*}
  R(s_{32}, N, s_{33}) &= -0.01 \\
  R(s_{32}, N, s_{42}) &= -1.01 \\
  R(s_{33}, E, s_{43}) &= 0.99
\end{align*}
\]

Cost of breathing

R is also a Big Table!

For now, we also give this to the agent
Markov Decision Processes

- An MDP is defined by:
  - A set of states $s \in S$
  - A set of actions $a \in A$
  - A transition function $T(s, a, s')$
    - Probability that $a$ from $s$ leads to $s'$, i.e., $P(s' | s, a)$
    - Also called the model or the dynamics
  - A reward function $R(s, a, s')$
    - Sometimes just $R(s)$ or $R(s')$
  - A start state
  - Maybe a terminal state

- MDPs are non-deterministic search problems
  - One way to solve them is with expectimax search
  - We’ll have a new tool soon
What is Markov about MDPs?

- “Markov” generally means that given the present state, the future and the past are independent.

- For Markov decision processes, “Markov” means action outcomes depend only on the current state.

\[
P(S_{t+1} = s' | S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, A_{t-1}, \ldots, S_0 = s_0)
\]

\[
= P(S_{t+1} = s' | S_t = s_t, A_t = a_t)
\]

- This is just like search, where the successor function could only depend on the current state (not the history).

Andrey Markov (1856-1922)
In deterministic single-agent search problems, we wanted an optimal plan, or sequence of actions, from start to a goal.

For MDPs, we want an optimal policy $\pi^*: S \rightarrow A$

- A policy $\pi$ gives an action for each state
- An optimal policy is one that maximizes expected utility if followed
- An explicit policy defines a reflex agent

Optimal policy when $R(s, a, s') = -0.4$ for all non-terminals $s$
Optimal Policies

\[ R(s) = -0.01 \]

\[ R(s) = -0.03 \]

\[ R(s) = -0.4 \]

\[ R(s) = -2.0 \]
Example: Racing
Example: Racing

- A robot car wants to travel far, quickly
- Three states: Cool, Warm, Overheated
- Two actions: Slow, Fast
- Going faster gets double reward

<table>
<thead>
<tr>
<th>State</th>
<th>Slow Action</th>
<th>Fast Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cool</td>
<td>0.5</td>
<td>+1</td>
</tr>
<tr>
<td>Warm</td>
<td>0.5</td>
<td>+1</td>
</tr>
<tr>
<td>Overheated</td>
<td>0.5</td>
<td>+2</td>
</tr>
</tbody>
</table>

Rewards:
- Slow: +1
- Fast: +1, +2, -10

Transition Probabilities:
- Slow to Slow: 0.5
- Slow to Fast: 0.5
- Fast to Slow: 0.5
- Fast to Fast: 0.5
- Fast to Overheated: 1.0
Racing Search Tree
Each MDP state projects an expectimax-like search tree.

- A state is a tuple $(s, a)$.
- A transition is a tuple $(s, a, s')$.
- The transition function $T(s, a, s') = P(s' | s, a)$.
- The reward function $R(s, a, s')$. 

Diagram:

- A node labeled $s$ is a state.
- Edges from $s$ to $s'$ and $a$.
- Transition $(s, a, s')$.
- Probability distribution $P(s' | s, a)$.
Utilities of Sequences
Utilities of Sequences

- What preferences should an agent have over reward sequences?

- More or less? \([1, 2, 2]\) or \([2, 3, 4]\)

- Now or later? \([0, 0, 1]\) or \([1, 0, 0]\)
Discounting

- It’s reasonable to maximize the sum of rewards
- It’s also reasonable to prefer rewards now to rewards later
- One solution: values of rewards decay exponentially
Discounting

- **How to discount?**
  - Each time we descend a level, we multiply in the discount once

- **Why discount?**
  - Think of it as a gamma chance of ending the process at every step
  - Also helps our algorithms converge

- **Example: discount of 0.5**
  - $U([1,2,3]) = 1 \times 1 + 0.5 \times 2 + 0.25 \times 3$
  - $U([1,2,3]) < U([3,2,1])$
Quiz: Discounting

- Given:
  - Actions: East, West, and Exit (only available in exit states a, e)
  - Transitions: deterministic

- Quiz 1: For $\gamma = 1$, what is the optimal policy?
  
- Quiz 2: For $\gamma = 0.1$, what is the optimal policy?

- Quiz 3: For which $\gamma$ are West and East equally good when in state d?

  $1_{\gamma} = 10 \cdot \gamma^3$
Infinite Utilities?!  

- Problem: What if the game lasts forever? Do we get infinite rewards?

- Solutions:
  - Finite horizon: (similar to depth-limited search)
    - Terminate episodes after a fixed $T$ steps (e.g. life)
    - Policy $\pi$ depends on time left
  - Discounting: use $0 < \gamma < 1$
    \[
    U([r_0, \ldots r_\infty]) = \sum_{t=0}^{\infty} \gamma^t r_t \leq R_{\text{max}}/(1 - \gamma)
    \]
    - Smaller $\gamma$ means smaller “horizon” – shorter term focus
  - Absorbing state: guarantee that for every policy, a terminal state will eventually be reached (like “overheated” for racing)
Recap: Defining MDPs

- Markov decision processes:
  - Set of states $S$
  - Start state $s_0$
  - Set of actions $A$
  - Transitions $P(s' \mid s,a)$ (or $T(s,a,s')$)
  - Rewards $R(s,a,s')$ (and discount $\gamma$)

- MDP quantities so far:
  - Policy = Choice of action for each state
  - Utility = sum of (discounted) rewards
Solving MDPs