Today’s Outline

- MDPs
  - Finding the optimal policy
  - Policy iteration
  - Q-value iteration
- Reinforcement Learning
  - Introduction
Recall: MDPs

- An MDP is defined by:
  - States \( s \in S \)
  - Actions \( a \in A \)
  - Transition function \( T(s,a,s') = P(s' | s,a) \)
  - Reward function \( R(s,a,s') \)
  - Start state

Recall: Value Iteration

- How do we compute \( V^*(s) \) for all states \( s \)?
- Use iterative method called Value Iteration:
  - Start with \( V_0^*(s) = 0 \)
  - Given \( V_i^* \), calculate the values for all states for depth \( i+1 \):
    \[
    V_{i+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[ R(s,a,s') + \gamma V_i(s') \right]
    \]
  - Repeat until convergence
Example: Value Iteration (Movie)

Optimal Policy: Computing Actions

- Which action to chose in state $s$:
  - Given optimal $Q^*$?
    $$\text{Best action} = \arg\max_a Q^*(s, a)$$
  - Given optimal values $V^*$?
    $$\text{Best action} = \arg\max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$
Value Iteration Complexity

- **Problem size:**
  - $|A|$ actions and $|S|$ states

- **Each Iteration**
  - For all $s$:
    $$V_{i+1}(s) = \max_a \sum_{s'} T(s, a, s') \left[ R(s, a, s') + \gamma V_i(s') \right]$$
  - Time: $O(|A| \cdot |S|^2)$
  - Space: $O(|S|)$

- **Num of iterations**
  - Can prove that it can be exponential in the discount factor $\gamma$

---

Is there a faster alternative to value iteration?

Yeah, crazy little thing called policy iteration!
# Policy Iteration: Motivation

- Problem with value iteration:
  
  \[ V_{i+1}(s) = \max_a \sum_{s'} T(s, a, s') \left[ R(s, a, s') + \gamma V_i(s') \right] \]

  - Considering all actions makes each iteration slow

- What if we compute values for some fixed policy \( \pi(s) \)?

  \[ V^\pi(s) = \sum_{s'} T(s, \pi(s), s') \left[ R(s, \pi(s), s') + \gamma V^\pi(s') \right] \]

  Look, no max, so fast!

---

# Policy Iteration

- Start with an arbitrary policy \( \pi_0 \)
- Repeat until policy converges:
  1. **Policy evaluation (fast):** With fixed current policy \( \pi_k \), iterate values until convergence:

     \[ V^\pi_{i+1}(s) = \sum_{s'} T(s, \pi_k(s), s') \left[ R(s, \pi_k(s), s') + \gamma V^\pi_i(s') \right] \]

  2. **Policy improvement (slow but infrequent):** Based on converged values in (2), update policy by choosing best action using one-step look-ahead:

     \[ \pi_{k+1}(s) = \arg \max_a \sum_{s'} T(s, a, s') \left[ R(s, a, s') + \gamma V^\pi_k(s') \right] \]
Policy Iteration Complexity

- Problem size:
  - $|A|$ actions and $|S|$ states

- Each Iteration
  - Time: $O(|S|^3 + |A| \cdot |S|^2)$
  - Space: $O(|S|)$

- Num of iterations
  - Unknown, but can be fast in practice
  - Convergence is guaranteed

One last variation: Q-Value Iteration

- Value iteration updates values for states:
  $$V_{i+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') \left[ R(s, a, s') + \gamma V_i(s') \right]$$

  Equivalent to:
  $$V_{i+1}(s) \leftarrow \max_a Q_{i+1}(s, a)$$

  Why not update $Q$-values instead of $V$?!

  $$Q_{i+1}(s, a) \leftarrow \sum_{s'} T(s, a, s') \left[ R(s, a, s') + \gamma V_i(s') \right]$$
  i.e.,
  $$Q_{i+1}(s, a) \leftarrow \sum_{s'} T(s, a, s') \left[ R(s, a, s') + \gamma \max_{a'} Q_i(s', a') \right]$$
**Q-Value Iteration**

Initialize each Q-state: \( Q_0(s,a) = 0 \)

Repeat

For all Q-states s,a

Compute \( Q_{i+1}(s,a) \) from \( Q_i \) by Bellman update:

\[
Q_{i+1}(s, a) = \sum_{s'} T(s, a, s') \left[ R(s, a, s') + \gamma \max_{a'} Q_i(s', a') \right]
\]

Until \( \max_{s,a} |Q_{i+1}(s,a) - Q_i(s,a)| < \varepsilon \)
(i.e., until convergence of all Q values;
\( \varepsilon \) is a small positive value)

---

**Example: Q-Value Iteration**

![Value Iteration](image1)

![Q-Value Iteration](image2)

Numbers show \( V(s) \)

Numbers show \( Q(s,a) \)
What if we don’t know the transition model $T(s,a,s')$ and reward model $R(s,a)$?!

Enter…Reinforcement Learning (RL)

Agent doesn’t know what actions do
Agent doesn’t know which states are good
Try different actions and learn policy by trial-and-error!
Example: Robotic Learning

Crawler robot

(from http://sysplan.nams.kyushu-u.ac.jp/gen/papers/JavaDemoML97/robodemo.html)

Example: Animal Learning

- **RL studied experimentally for more than 80 years in psychology and brain science**
  - Rewards: food, pain, hunger, drugs, etc.
  - Evidence for RL in the brain via a chemical called dopamine
- **Example: foraging**
  - Bees can learn near-optimal foraging policy in field of artificial flowers with controlled nectar supplies

Yum!
RL solves the “Credit Assignment” Problem

I'm in state 43, reward = 0, action = 2
- “ “ “ 39, “ = 0, “ = 4
- “ “ “ 22, “ = 0, “ = 1
- “ “ “ 21, “ = 0, “ = 1
- “ “ “ 21, “ = 0, “ = 1
- “ “ “ 13, “ = 0, “ = 2
- “ “ “ 54, “ = 0, “ = 2
- “ “ “ 26, “ = 100,

But which of the actions along the way actually helped you get there??
RL solves this Credit Assignment problem

Yippee! I got to a state with a big reward!

The Reinforcement Learning (RL) Problem

- **Given:** Set of states \( S \) and actions \( A \)
  - Do not know transition probabilities \( T \)
  - Do not know reward function \( R \)

- **Interact with environment at each time step \( t \):**
  - Environment gives new state \( s_t \) and reward \( r_t \)
  - Choose next action \( a_t \)

- **Goal:** Learn policy \( \pi \) that maximizes expected discounted sum of rewards
Two main approaches to RL

- **Model-based approaches:**
  - Explore environment & learn model \( T=P(s'|s,a) \) and \( R(s,a,s') \)
  - Use model to compute policy MDP-style
  - Works well when state-space is small

- **Model-free approach:**
  - Don’t learn a model
  - Learn value function (Q value) or policy *directly*
  - Works better when state space is large

Comparison of approaches

- **Model-based approaches:**
  
  Learn \( T + R \)
  
  \(|S|^2|A| + |S||A|\) parameters \( (\text{E.g., } 200^2 \times 10 + 200 \times 10 = 402,000) \)

- **Model-free approach:**
  
  Learn \( Q \)
  
  \(|S||A|\) parameters \( (\text{E.g., } 200 \times 10 = 2,000) \)
Next Time

- Model-Free Reinforcement learning
  - Q-learning
- To Do
  - Finish Chapter 17
  - Read Chapter 21
  - Start Project #3