Today

- More MDPs: policy iteration
- Reinforcement learning
  - Passive learning
  - Active learning

Recap: MDPs

- Markov decision processes (MDPs)
  - A set of states \( s \in S \)
  - A model \( T(s,a,s') \)
    - Probability that the outcome of action \( a \) in state \( s \) is \( s' \)
  - A reward function \( R(s) \)

- Solutions to an MDP
  - A policy \( \pi(s) \)
  - Specifies an action for each state
  - We want to find a policy which maximizes total expected utility = expected (discounted) rewards

Bellman Equations

- The value of a state according to \( \pi \)
  \[ U^\pi(s) = R(s) + \gamma \sum_{s'} T(s',a,s') U^\pi(s') \]
- The policy according to a value \( U \)
  \[ \pi^U(s) = \arg \max_a \sum_{s'} T(s',a,s') U(s') \]
- The optimal value of a state
  \[ U^*(s) = R(s) + \gamma \max_a \sum_{s'} T(s',a,s') U^*(s') \]

Recap: Value Iteration

- Idea:
  - Start with (bad) value estimates (e.g. \( U_0(s) = 0 \))
  - Start with corresponding (bad) policy \( \pi_0(s) \)
  - Update values using the Bellman relations (once)
    \[ U_{i+1}(s) = R(s) + \gamma \sum_{s'} T(s',a,s') U_i(s') \]
  - Update policy based on new values
    \[ \pi_{i+1}(s) = \arg \max_a \sum_{s'} T(s',a,s') U_{i+1}(s') \]
  - Repeat until convergence

Policy Iteration

- Alternate approach:
  - Policy evaluation: calculate exact utility values for a fixed policy
  - Policy improvement: update policy based on values
  - Repeat until convergence
- This is policy iteration
  - Can converge faster under some conditions
### Policy Evaluation

- If we have a fixed policy $\pi$, use a simplified Bellman update to calculate utilities:

$$U^{\pi}(s) = R(s) + \gamma \sum_{s'} T(s,a,s') U^{\pi}(s')$$

- Unlike in value iteration, policy does not change during update process.
- Converges to the expected utility values for this $\pi$.
- Can also solve for $U$ with linear algebra methods instead of iteration.

### Policy Improvement

- Once values are correct for current policy, update the policy:

$$\pi_{t+1}(s) = \arg \max_a \sum_{s'} T(s,a,s') U^{\pi}(s')$$

- Note:
  - Value iteration: update $U$, $\pi$, $U$, $\pi$, $U$...
  - Policy iteration: $U$, $U$, $U$... $\pi$, $U$, $U$, $U$... $\pi$
  - Otherwise, basically the same!

### Reinforcement Learning

- Reinforcement learning:
  - Still have an MDP:
    - A set of states $s \in S$
    - A model $T(s,a,s')$
    - A reward function $R(s)$
  - Still looking for a policy $\pi(s)$
  - New twist: don't know $T$ or $R$
    - I.e. don't know which states are good or what the actions do
    - Must actually try actions and states out to learn

### Example: Animal Learning

- RL studied experimentally for more than 60 years in psychology
  - Rewards: food, pain, hunger, drugs, etc.
  - Mechanisms and sophistication debated
  - Example: foraging
    - Bees learn near-optimal foraging plan in field of artificial flowers with controlled nectar supplies
    - Bees have a direct neural connection from nectar intake measurement to motor planning area

### Example: Autonomous Helicopter

- Example: Autonomous Helicopter
Example: Backgammon

- Reward only for win / loss in terminal states, zero otherwise
- TD-Gammon learns a function approximation to U(s) using a neural network
- Combined with depth 3 search, one of the top 3 players in the world
- (We’ll cover game playing in a few weeks)

Example: Direct Estimation

- **Episodes:**
  - (1,1) -1 up
  - (1,2) -1 up
  - (1,3) -1 right
  - (2,3) -1 right
  - (3,3) -1 right
  - (3,2) -1 up
  - (4,3) +100

  
  - U(1,1) = (92 + -106) / 2 = -7
  - U(3,3) = (99 + 97 + -102) / 3 = -31.3

Example: Model-Based Learning

- **Idea:**
  - Learn the model empirically (rather than values)
  - Solve the MDP as if the learned model were correct

- **Empirical model learning**
  - Simplest case:
    - Count outcomes for each s,a
    - Normalize to give estimate of T(s,a,s')
  - Discover R(s) the first time we enter s
  - More complex learners are possible (e.g. if we know that all squares have related action outcomes "stationary noise")

Example: Model-Free Learning

- **Big idea:** why bother learning T?
  - Update each time we experience a transition
  - Frequent outcomes will contribute more updates (over time)

- **Temporal difference learning (TD)**
  - Policy still fixed!
  - Move values toward value of whatever successor occurs

  $U^T(s) = R(s) + \gamma \sum_{s'} T(s'|s)U^T(s', \pi(s'))$

  $U^T(s) \leftarrow U^T(s) + \alpha \left( R(s) + \gamma U^T(s') - U^T(s) \right)$

- [DEMO]

Passive Learning

- **Simplified task**
  - You don’t know the transitions T(s,a,s')
  - You don’t know the rewards R(s)
  - You DO know the policy \( \pi(s) \)
  - Goal: learn the state values (and maybe the model)

- In this case:
  - No choice about what actions to take
  - Just execute the policy and learn from experience
  - We’ll get to the general case soon
**Example: Passive TD**

\[ U^\pi(s) \leftarrow U^\pi(s) + \alpha \left[ r(s) + \gamma U^\pi(s') - U^\pi(s) \right] \]

(1,1) -1 up
(1,2) -1 up
(1,3) -1 right
(2,3) -1 right
(3,3) -1 right
(3,2) -1 left
(4,3) +100

Take \( \gamma = 1 \), \( \alpha = 0.1 \)

---

**Example: Greedy Active Learning**

- Imagine we find the lower path to the good exit first
- Some states will never be visited following this policy from (1,1)
- We’ll keep re-using this policy because following it never collects the regions of the model we need to learn the optimal policy

---

**(Greedy) Active Learning**

- In general, want to learn the optimal policy
- Idea:
  - Learn an initial model of the environment:
    - Solve for the optimal policy for this model (value or policy iteration)
    - Refine model through experience and repeat

---

**What Went Wrong?**

- Problem with following optimal policy for current model:
  - Never learn about better regions of the space
- Fundamental tradeoff: exploration vs. exploitation
  - Exploration: must take actions with suboptimal estimates to discover new rewards and increase eventual utility
  - Exploitation: once the true optimal policy is learned, exploration reduces utility
  - Systems must explore in the beginning and exploit in the limit

---

**Next Time**

- Active reinforcement learning
  - Q learning
  - Balancing exploration / exploitation

- Function approximation
  - Generalization for reinforcement learning
  - Modeling utilities for complex spaces