DATA 37200: Learning, Decisions, and Limits
(Winter 2026)

Lecture 14: Learning Tabular MDPs (+intro to multiplayer)

Instructor: Frederic Koehler

Context

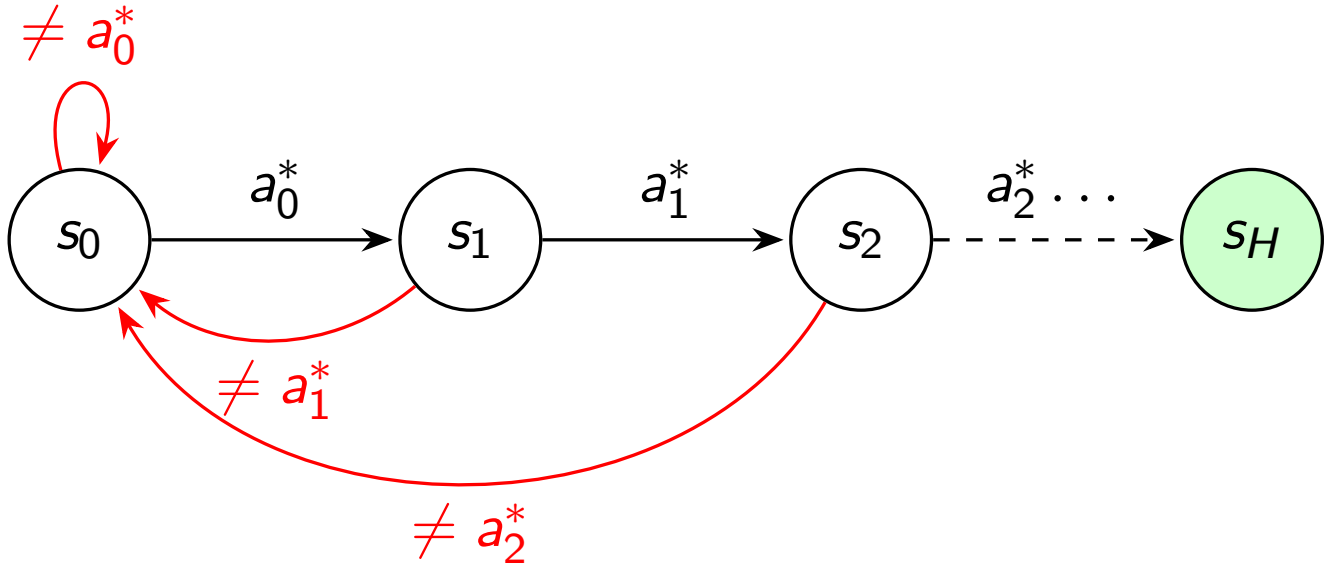
- Two important families of optimally solvable MDPs: tabular MDP (not-too-large state and action space) and LQR (control theory, last class).
- What if we want to solve an MDP, but, e.g. transitions are unknown?
  - We can learn them from data.
  - LQR: system identification via random control + linear regression. (Last time.)
  - Tabular MDP: can learn using policies based on the principle of "optimism under uncertainty".
- Reminder: naïve exploration fails (next two slides).

The Markov Decision Process (MDP) Formalism

An MDP is defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$.

- **State Space** $\mathcal{S}$: The set of all possible states.
- **Action Space** $\mathcal{A}$: The set of available actions.
- **Transition Dynamics** $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$:
  - $P(s' \mid s, a)$ is the probability of transitioning to s' given state s and action a.
- **Reward Distribution** R:
  - Upon taking action a in state s, receive random reward $r \sim R(s, a)$.
  - Expected immediate reward: $r(s, a) = \mathbb{E}[r \mid s, a]$.
- **Discount Factor** $\gamma \in [0, 1)$:
  - Determines the present value of future rewards. Can approximate no discount by setting $\gamma \approx 1$.

Failure of Naive Exploration



Any mistake sends the agent back to the start s_0. (Or: terminates the episode.)

- A random policy (picking actions uniformly) reaches s_H with probability $(1/2)^H$.
- Even though the state space is tiny ($O(H)$), random exploration takes exponential time.
- **Lesson:** To solve this efficiently (polynomial in H), an agent must remember which actions caused resets.

Optimism under uncertainty

- Optimism under uncertainty: a design strategy for bandit/RL algorithms. Assume that every option is "as good as possible" given the data we have seen.
- Morally, Bayesian with a prior weighted towards high rewards.
- In the previous example: if we have previously tried inputting 0 and it keeps ending our game without reward, we will try inputting 1 because we are optimistic this will go better.

Optimism vs pessimism

In an optimistic strategy, we keep guessing upper bounds on the reward of the optimal policy. Then we test if the upper bound is actually achievable: if yes, we are done. If no, we learn something. PRO: Good for making sure we don't miss anything.

CON: when state and action space are large, optimism might waste too much time exploring. One reason why other approaches (e.g. policy gradient) are preferable in some applications.

- Pessimism may have a benefit.
- Optimism in practice: when playing Go against a fixed opponent, we can always try playing weird strategies and hope they respond poorly to them. This has clear upsides and downsides...

Today: two approaches to learning in unknown MDPs

- UCB Value Iteration: combine value iteration with a UCB optimism bonus.
- R_{max}: Greedy, but pretend that any option we haven't explored well has very high reward.
- For either method, I will focus on high-level details and not go through a detailed proof that the method works. You can see the course webpage for links to the details (which requires a lot of notation).

Comment: conceptual proof difficulty

- We would like to say: we explore enough to learn the entire MDP, and then we can for sure find the optimal policy.
- However: some states might very hard to reach. (E.g. situations which arise after crashing a video game.)
- This is true even in tabular case: transition probabilities can be very small but nonzero.
- Because there is always something potentially unknown, optimism is more crucial than in bandit case.

R_{max}: Model-Based Optimism

- **Key Idea:** "Innocent until proven guilty." If we haven't visited a state-action pair (s, a) enough times, assume it's amazing.
- We classify each (s, a) pair based on visit counts $N(s, a)$:
  - **Known:** $N(s, a) \geq m$ (for some threshold m).
  - **Unknown:** $N(s, a) < m$.
- Build an optimistic, fictitious MDP $\hat{\mathcal{M}}$:
  - For **known** (s, a), use the empirical transition probabilities $\hat{P}$ and rewards $\hat{r}$ calculated from our data.
  - For **unknown** (s, a), assume it transitions with probability 1 to a fictitious absorbing state s_{heaven} that yields the maximum possible reward indefinitely.

The R_{max} Algorithm Loop

1. Initialize all (s, a) counts to zero (everything is unknown).
2. Construct the optimistic MDP $\hat{\mathcal{M}}$ using current data.
3. Compute the optimal policy $\hat{\pi}^*$ for $\hat{\mathcal{M}}$ (e.g., via Value Iteration).
4. Execute $\hat{\pi}^*$ in the *real* environment for one episode.
5. Update visit counts $N(s, a)$ and empirical estimates $\hat{P}$, $\hat{r}$.
6. Repeat from Step 2.

The "Explore or Exploit" Lemma:
By following $\hat{\pi}^*$, the agent will either gather high rewards in the known parts of the MDP (Exploit) or quickly reach unknown states and learn about them (Explore).

There are only $|\mathcal{S}| \times |\mathcal{A}|$ state-action pairs to explore, so eventually the policy will be rarely reaching new states. Then by optimism, we are done.

From Hard Thresholds to Soft Bonuses

- R_{max} uses a **hard threshold**: a state is entirely unknown until visited m times, then suddenly becomes completely known.
- **UCB-VI (Upper Confidence Bound Value Iteration)** uses a **soft bonus** approach instead.
- Instead of modifying the MDP transitions directly to point to s_{heaven}, we add an explicit exploration bonus $b(s, a)$ directly to our reward/value estimates.
- The bonus scales with our statistical uncertainty: it is large when $N(s, a)$ is small, and decays as $N(s, a) \to \infty$.
- **For simplicity:** we consider UCB-VI in the case of a finite horizon H with $\gamma = 1$ (no discount factor). This matches the [FR] lecture notes.

UCB Value Iteration (UCB-VI)

- Maintain empirical estimates $\hat{P}$ and $\hat{r}$ just like R_{max}.
- Perform Value Iteration (e.g., for finite horizon H), adding the bonus $b(s, a)$:

$$Q_h(s, a) = \min\left(H, \hat{r}(s, a) + b(s, a) + \sum_{s'} \hat{P}(s' \mid s, a) V_{h+1}(s')\right)$$

$$V_h(s) = \max_a Q_h(s, a)$$

- Bonus can be derived from Hoeffding's inequality:

$$b(s, a) = 2H\sqrt{\frac{\log(2SAHT/\delta)}{N(s, a)}}$$

- **Note:** The $\min(H, \dots)$ ensures our optimistic Q-values don't exceed the maximum possible return H.

UCB-VI: Theoretical guarantee

UCB-VI as described above achieves regret

$$R_T \lesssim H|S|\sqrt{|A|T}\sqrt{\log(|S||A|HT/\delta)}.$$

(Reminder: regret is over T episodes and measures gap between our total reward and expected reward under optimal policy.)

With more work (tighter bonus terms etc), a variant can achieve around $\tilde{O}(\sqrt{H|S||A|T})$ which is close to tight. See Section 5.6 of [FR] notes for references.

Concluding this section

- We have now covered the basics of single-player RL.
- We will return to some additional topics (policy gradient, RLHF) later.
- Single player RL assumed a fixed environment (MDP). But we are also interested in applications like playing Go, card games, economic transactions, ... which may involve other "strategic agents".
- To discuss these kinds of situations, we need to introduce some game theory.

Two-player zero-sum games

- The simplest setting with multiple agents: two players with diametrically opposed interests.
- **Zero-sum property:** Whatever Player 1 wins, Player 2 loses. The rewards satisfy $R_1 + R_2 = 0$.
- Often represented as a matrix game (Normal Form).
  - Player 1 chooses a row (action a_1).
  - Player 2 chooses a column (action a_2).
  - The matrix entry gives the payoff to Player 1 (which is the loss for Player 2).
- **Examples:** Rock-Paper-Scissors, Matching Pennies.
- **Key concept:** To avoid being exploited by the opponent, players usually need to randomize their actions (*mixed strategies*).

The Minimax Theorem and Nash Equilibrium

- How should we play a zero-sum game? Assume the opponent is perfectly rational and wants to minimize our reward.
- **Minimax value:** Player 1 wants to choose a policy $\pi_1 \in \Delta(\mathcal{A}_1)$ that maximizes their worst-case payoff against any opponent policy $\pi_2 \in \Delta(\mathcal{A}_2)$:

$$V^* = \max_{\pi_1} \min_{\pi_2} \mathbb{E}[R(\pi_1, \pi_2)]$$

- **von Neumann's Minimax Theorem (1928):** For finite two-player zero-sum games, the order of max and min doesn't matter!

$$\max_{\pi_1} \min_{\pi_2} \mathbb{E}[R] = \min_{\pi_2} \max_{\pi_1} \mathbb{E}[R] = V^*$$

- A pair of strategies achieving this is a **Nash Equilibrium**. Neither player has an incentive to unilaterally deviate.

Example: Rock-Paper-Scissors

- We represent the game using a **payoff matrix** for Player 1 (the "Row Player"). Player 2 (the "Column Player") wants to minimize these numbers.
- Payoffs: $+1$ (win), -1 (loss), 0 (tie).

	Rock	**Paper**	**Scissors**
Rock	0	-1	$+1$
Paper	$+1$	0	-1
Scissors	-1	$+1$	0

- **How to read:** If Player 1 plays Paper and Player 2 plays Rock, the entry is $+1$ (Player 1 wins).
- **Nash Equilibrium:** To be unexploitable, both players must randomize, playing each action with exactly probability $1/3$. The *value* of the game is 0.

Adding State: Two-Player Zero-Sum Markov Games

- Matrix games are single-step. What if the game evolves over time with states?
- **Markov Games (Stochastic Games):** A generalization of MDPs to multiple players.
- A 2-player zero-sum Markov Game is defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}_1, \mathcal{A}_2, P, R, \gamma)$:
  - **State Space** $\mathcal{S}$
  - **Action Spaces** $\mathcal{A}_1$ for Player 1, $\mathcal{A}_2$ for Player 2.
  - **Transitions** $P : \mathcal{S} \times \mathcal{A}_1 \times \mathcal{A}_2 \to \Delta(\mathcal{S})$. The next state depends on *both* actions!
  - **Reward Distribution** R: Player 1 wants to maximize the expected reward $r(s, a_1, a_2)$, and Player 2 wants to minimize it.

Solving Markov Games: Minimax Value Iteration

- Can we compute the optimal value function $V^*(s)$ for a Markov Game?
- Yes, by combining Value Iteration with the Minimax theorem.
- **Minimax Bellman Equation:**

$$V^*(s) = \max_{\pi_1 \in \Delta(\mathcal{A}_1)} \min_{\pi_2 \in \Delta(\mathcal{A}_2)} \mathbb{E}_{a_1 \sim \pi_1, a_2 \sim \pi_2} Q(s, a_1, a_2)$$

where

$$Q(s, a_1, a_2) = r(s, a_1, a_2) + \gamma \sum_{s'} P(s' \mid s, a_1, a_2) V^*(s')$$

- **Intuition:** At each state s, the players are effectively playing a local matrix game where the payoffs are the immediate rewards plus the discounted future values.

Beyond Zero-Sum: Prisoner's Dilemma

- Not all multi-agent environments are strictly competitive. In **general-sum games**, the sum of the rewards is not necessarily constant.
- Two suspects are arrested. They can either **Cooperate** (with each other, by remaining silent) or **Defect** (betray the other).
- Because it's not zero-sum, the matrix must show the payoff for *both* players as a tuple: (R_1, R_2).

	P2: Cooperate	**P2: Defect**
P1: Cooperate	$(-1, -1)$	$(-3, 0)$
P1: Defect	$(0, -3)$	$(-2, -2)$

- **The Dilemma:** No matter what Player 2 does, Player 1 is always better off Defecting ($0 > -1$ and $-2 > -3$).
- **Nash Equilibrium:** Both players choose **Defect** yielding $(-2, -2)$. But if they could somehow trust each other, they could both get a better outcome at $(-1, -1)$.