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

Lecture 12: Markov Decision Processes and Value Iteration

Instructor: Frederic Koehler

Announcements

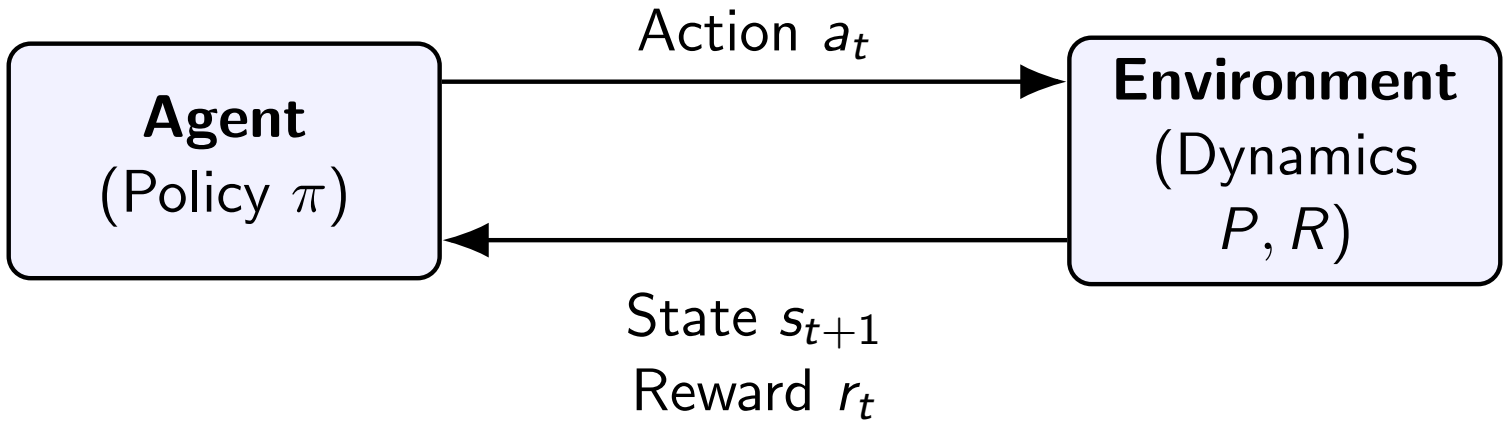
HW2 has been uploaded, you should have received an email. See course website.

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 $\gamma \approx 1$.

Visualizing MDP Dynamics



The Interaction Loop:

1. Agent observes state s_t.
2. Agent executes action $a_t \sim \pi(\cdot|s_t)$.
3. Environment returns reward $r_t \sim R(s_t, a_t)$ and transitions to new state $s_{t+1} \sim P(\cdot|s_t, a_t)$.

Policies and Value Functions

Policy π: A mapping from states to distributions over actions, $\pi : \mathcal{S} \to \Delta(\mathcal{A})$.

Value Function $V^\pi(s)$: Expected discounted return from state s.

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \,\middle|\, s_t = s \right]$$

Q-Function $Q^\pi(s, a)$: Expected discounted return taking action a in state s, then following π.

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[r_t + \gamma \sum_{k=1}^{\infty} \gamma^k r_{t+k} \,\middle|\, s_t = s, a_t = a \right]$$

The Bellman Equations

For a fixed policy π, the values satisfy recursive relationships:

Bellman Expectation Equation for Q^π

$$Q^\pi(s,a) = \mathbb{E}[R(s,a)] + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s,a) V^\pi(s')$$

Connection between V and Q

$$V^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) Q^\pi(s,a) = \mathbb{E}_{a \sim \pi(\cdot|s)}[Q^\pi(s,a)]$$

Bellman Optimality Equations

We seek the optimal policy π^* such that $V^{\pi^*}(s) \geq V^{\pi}(s)$ for all s, π.

The optimal value functions V^* and Q^* satisfy:

$$V^*(s) = \max_{a \in \mathcal{A}} Q^*(s, a)$$

$$Q^*(s, a) = \mathbb{E}[R(s, a)] + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s, a) V^*(s')$$

Optimal Policy:

$$\pi^*(s) = \operatorname*{argmax}_{a \in \mathcal{A}} Q^*(s, a)$$

Solving for π^*: Value Iteration

Algorithm:

1. Initialize $V_0(s) = 0$ for all $s \in \mathcal{S}$.
2. Repeat until convergence ($||V_{k+1} - V_k||_\infty < \epsilon$):

$$V_{k+1}(s) \leftarrow \max_{a \in \mathcal{A}} \left(\mathbb{E}[R(s,a)] + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s, a) V_k(s') \right)$$

3. Output greedy policy w.r.t. final V.

Note: Convergence is guaranteed because the Bellman operator is a γ-contraction (since $\gamma < 1$, by Banach fixed point theorem).

An illustrative example

MDPs are closely related to shortest path/flow problems.

- Shortest path MDP: we receive a reward of 1 dollar once we reach target t, with some decay factor $\gamma < 1$.
- States are vertices of a graph, action at each step is to choose a neighbor of current vertex to move to.
- Optimal policy gets to t as fast as possible.
- At each vertex, it moves to a neighbor following a shortest path to the root.

In this context, value iteration is essentially Dijkstra's algorithm.

Connection to Contextual Bandits ($\gamma = 0$)

If the discount factor $\gamma = 0$, the problem simplifies significantly.

- Future states do not contribute to the value ($0 \times V(s') = 0$).
- The objective is to maximize immediate reward given current state s.

Reduction

$$Q^*(s, a) = \mathbb{E}[R(s, a)]$$

$$\pi^*(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, \mathbb{E}[R(s, a)]$$

Here, the state s is equivalent to the **Context** in a Contextual Bandit problem. The transition dynamics P become irrelevant for optimization.

Learning and Regret

Consider an agent interacting with the MDP over T episodes (who typically does not know the full dynamics of the MDP).

- In episode t, the agent starts at state $s_{t,0}$ and follows policy π_t.
- We compare the agent's performance against the optimal policy π^*.

Cumulative Regret Definition

$$\text{Regret}(T) = \sum_{t=1}^{T} (V^*(s_{t,0}) - V^{\pi_t}(s_{t,0}))$$

- **Sub-linear Regret** ($\text{Regret}(T) = o(T)$): Implies the agent is learning and converging to optimal behavior.
- **Linear Regret** ($\text{Regret}(T) \propto T$): The agent is failing to learn (e.g., constant random actions).

Finite Horizon MDP

Sometimes (e.g. in the learning setting), it is nice to setup the MDP so that each episode only has a finite number of steps to play.

- If the MDP has a terminal state T and is guaranteed to transition to T within $H \geq 0$ steps, we call H the horizon.
- In the finite horizon setting, we are allowed to set $\gamma = 1$ (no time discounting).
- This is how MDPs are formulated in the Foster-Rakhlin notes.

Understanding some difficult aspects of RL

We will go through some simple and well-known examples illustrating important challenges for reinforcement learning algorithms.

- First example: the state space is often huge (exponentially large), and some problems with exponentially large state space are very (information-theoretically) difficult.
- Second example: unlike bandits, pure random exploration is not a good idea in some MDPs. Simple examples are solvable by smarter strategies.

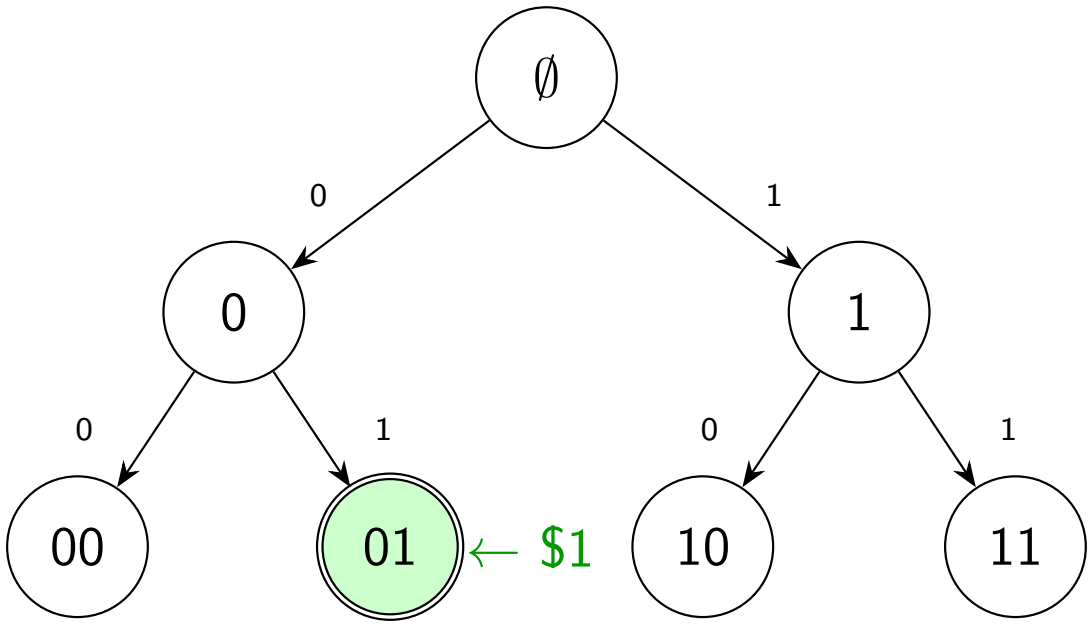
Example: The Binary Combination Lock

Consider an MDP representing a password input system.

- **Goal**: Enter a secret binary password of length H.
- **State Space** $\mathcal{S}$: The set of all binary strings of length $< H$.
  - Total states $\approx 2^H$ (a binary tree).
- **Action Space** $\mathcal{A}$: $\{0, 1\}$.
- **Transitions**: Deterministic. Action a appends bit a to current string.
- **Rewards**:
  - $r(s, a) = 1$ if the resulting string is the correct password w^*.
  - $r(s, a) = 0$ otherwise.

This is a **sparse reward** problem. The agent receives no feedback until it hits the specific target state.

Visualizing the Hardness of Exploration



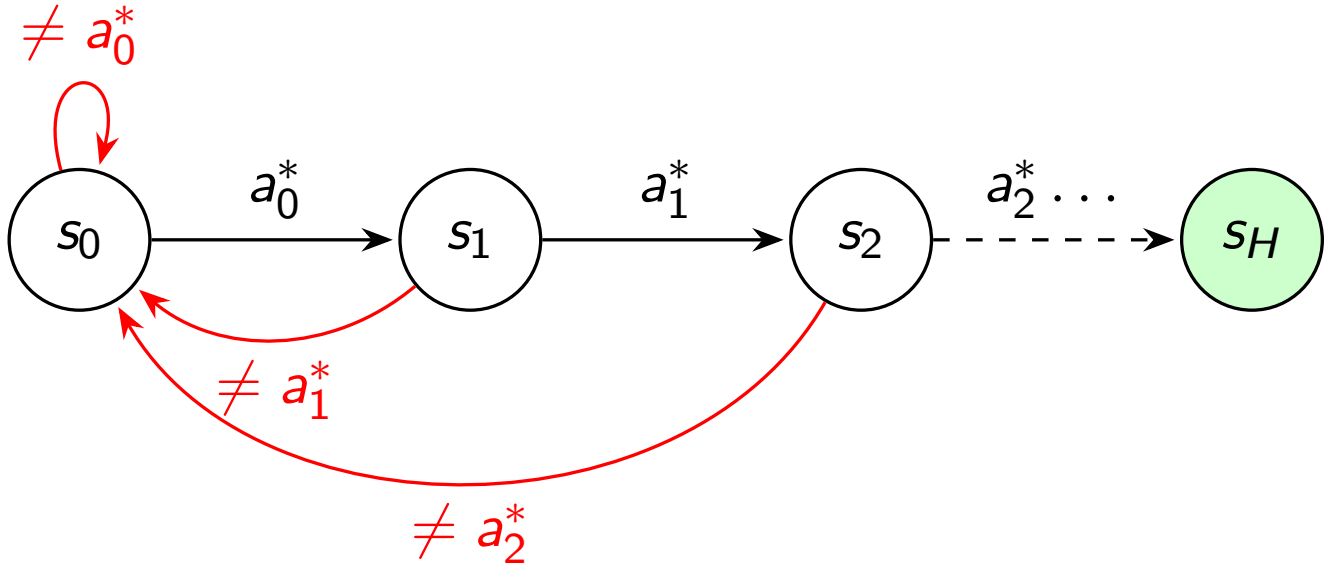
- There are 2^H leaf nodes. Only **one** has a reward.
- Without intermediate rewards ("shaping"), the agent learns nothing from failures.
- **Worst Case:** The agent must try every path.
- **Sample Complexity:** $\Omega(|\mathcal{S}|) = \Omega(2^H)$. This shows that RL can be exponentially hard in the horizon H without further assumptions.

Example: The Combination Lock with Resets

Consider a variant where errors are costly.

- **Goal**: Enter a secret binary password of length H.
- **State Space** $\mathcal{S}$: $\{0, 1, \ldots, H\}$.
  - State k represents "correctly entered first k digits".
  - **Size is Small**: $|\mathcal{S}| = H + 1$ (Linear, not Exponential!).
- **Transitions**:
  - **Correct Action**: $s_k \xrightarrow{a=w_k^*} s_{k+1}$ (Progress).
  - **Wrong Action**: $s_k \xrightarrow{a \neq w_k^*} s_0$ (**Reset to Start**).
- **Reward**: 1 only at state H.

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.