Tragedy-of-the-Commons RL Environment

December 4, 2025

Background and Motivation

Commons and resource-management problems are common in ecology, economics, and social systems.

In these settings, a shared resource is harvested by multiple agents. If all agents harvest aggressively, they may collapse the resource; if agents harvest too little, they may starve individually.

Reinforcement Learning (RL) provides a way to model and study how an agent might behave in a population of non-learning (or randomly acting) agents, and whether the agent can learn sustainable harvesting strategies.

This assignment represents the first step in my larger project which is to experiment with whether multiple reinforcement-learning agents can coexist sustainably in a commons-type environment without explicit coordination or predefined collaboration. In particular, I'm trying to see if a population of N agents can learn to converge on an equilibrium harvesting strategy, one where each agent extracts enough resources to survive over time, yet where the group collectively avoids over-exploitation and prevents collapse of the shared resource.

In this assignment, you will work with a simplified simulation environment inspired by the "tragedy of the commons." The environment is described below.

Environment Description

- There is a single renewable resource with maximum capacity $R_{\max}$.
- At the start of each episode, the resource is initialized at $R_{\max}$.
- In each "round" (time step):
    1. The order of all agents (the learning agent plus K non-learning "non-agents") is shuffled uniformly at random.
    2. Each non-agent before the learning agent extracts a random amount uniformly in $[0, \text{max\_nonagent\_take}]$.
    3. The learning agent then chooses how much to extract (via a discrete action). The agent *observes only*:

- Current resource level R, and
- Number of non-agents who acted before it in this round.

4. After the agent, the remaining non-agents extract resource (random as above).
5. After all extractions, the resource regenerates according to a logistic growth rule:

$$\Delta R = \text{regen\_rate} \times R \times \left(1 - \frac{R}{R_{\max}}\right).$$

6. If resource drops to zero (or below), the environment collapses; episode terminates with negative reward; the agent dies.
7. If the agent extracts below a minimal survival threshold, the agent starves; episode terminates with negative reward.

- Otherwise, the episode continues.

This environment captures key aspects of commons dilemmas:

- renewable but limited resource (via logistic growth),
- competition (multiple extractors),
- partial observability (agent does not know future extraction),
- decision tradeoff between short-term gain and long-term sustainability.

Tasks

attempt the following tasks.

1. **Implement the reward function.** In the `TocEnv` class, complete the method `_reward(self, agent_action)`. Here is my idea for the reward function:
   - If the agent's extraction is below `survival_min_take`, the episode ends with a negative reward (e.g. -1).
   - If resource is ≤ 0 (after any extraction), the episode ends with a negative reward (e.g. -1).
   - Otherwise, the agent receives a positive reward for surviving (e.g. +1), and the episode continues.
2. **Choose number of buckets for the discretization of the observation space.** Because the first observation in the observation vector is continuous this discretization method will allow for the use of Q-learning and SARSA as they require discrete state spaces. e.g.:

$$\text{state} = \left(\lfloor R/(R_{\max}/N) \rfloor,\ n_{\text{before}}\right)$$

   where n_{before} is number of non-agents before the agent in the current round. The more buckets N the larger the state space becomes. But it also allows for the observation to be more accurate. So there is a trade-off.

3. **Implement a tabular Q-learning algorithm.** Write a function or class that uses the environment + discretization + Q-learning to learn a policy. Use ε-greedy exploration, learning rate α, discount γ, etc. Track total reward per episode over training.

4. **Implement a tabular SARSA algorithm.** Similarly, implement SARSA (on-policy TD), using the same discretization. Track reward per episode over training.

5. **Compare performance via plots.** After training both algorithms for a sufficient number of episodes (e.g. 5,000–20,000), produce:

   - A plot of average (or smoothed) reward per episode vs episodes, comparing Q-learning and SARSA.
   - A plot of ending resource level per episode (to evaluate ecological collapse or stability).

6. **Write a short reflection.** Reflect on:

   - Differences between Q-learning and SARSA behavior (e.g. risk-taking vs conservative harvesting).
   - Whether the learned policies tend to sustain the resource or collapse it.
   - How partial observability (unknown future extractions) affects strategy.
   - Why might the resource graphs show a lot of variance.
   - What modifications (e.g. reward shaping, environment changes) might encourage better and/or different outcomes.