CS 594 Modern Reinforcement Learning

Lecture 5: MARL and MCTS

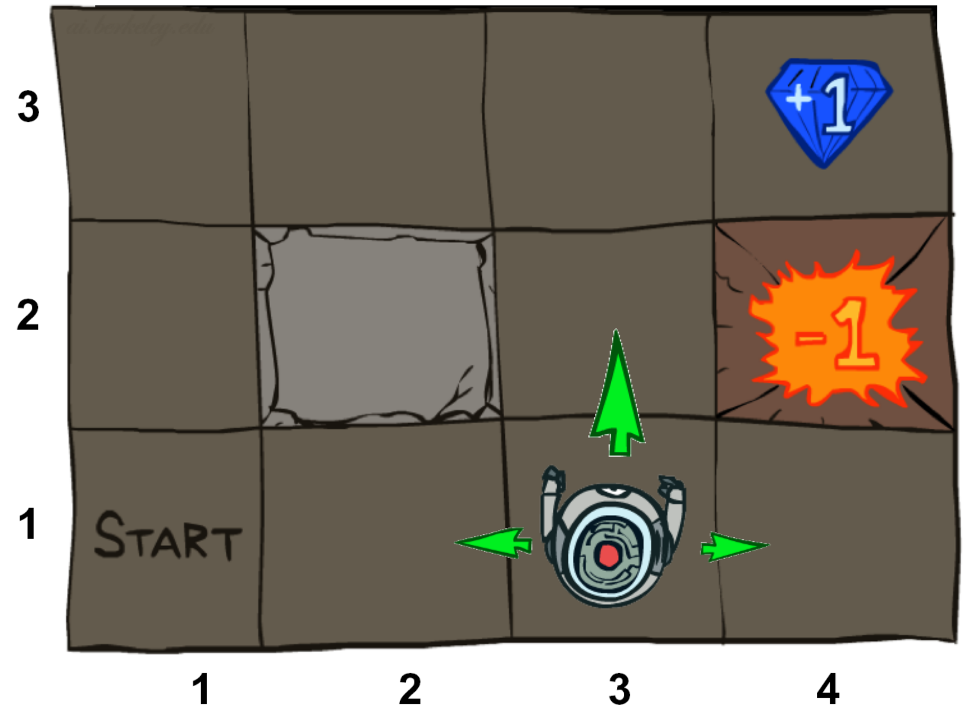
Announcements

- HW2 Released
- Resources being added to course website

Markov Decision Processes

- Trajectory $S_0, A_0, R_0, S_1, A_1, R_1, \ldots$

- A (finite) MDP is defined by:
  - A finite set of states $s \in S$
  - A finite set of actions $a \in A$
  - A finite set of rewards $r \in R$
  - Dynamics $p(s',r|s,a) = \Pr(S_t=s',R_t=r|S_{t-1}=s,A_{t-1}=a)$
  - Discount Rate γ
  - Possibly start state (distribution), terminal state



- Derived Quantities:
  - State transition probabilities $p(s'|s,a) = \Pr(S_t=s'|S_{t-1}=s,A_{t-1}=a) = \Sigma_{r\in R}\, p(s',r|s,a)$
  - Expected rewards $r(s,a) = E[R_t|S_{t-1}=s,A_{t-1}=a] = \Sigma_{r\in R} r\, \Sigma_{s'\in S}\, p(s',r|s,a)$
  - Or $r(s,a,s') = E[R_t|S_{t-1}=s,A_{t-1}=a, S_t=s'] = \Sigma_{r\in R} r\, p(s',r|s,a) / p(s'|s,a)$

Markov Games (a.k.a. Stochastic Games)

- A finite set of players $i \in N$
- A finite set of states $s \in S$
- A finite set of actions for each player $a_i \in A_i$
- A finite set of rewards $r \in R$
- Dynamics $p(s', r_1, \ldots, r_n \mid s, a_1, \ldots, a_n)$

Independent Q-learning

- $q(s,a) = q(s,a) + \alpha\left(r + \gamma \max_{a''}[q(s',a'')] - q(s,a)\right)$

- What might go wrong?

- Environment is non-stationary

Joint Q-learning

- $q_i(s, a_1, a_2) = q_i(s, a_1, a_2) + \alpha\big(r_i + v_i(s') - q_i(s, a_1 a_2)\big)$

- How should we compute $v_i(s')$?

- Special case: $r_2 = -r_1$
  - “Zero Sum”

- Implies $v_2 = -v_1$ and $q_2 = -q_1$

Min-Max-Q

- Player 1 wants to maximize q_1, player 2 wants to minimize it

- von Neumann Minimax Theorem:
  - $v_*(s) = \max_{\pi_1(s)} \min_{\pi_2(s)} q_1(s, \pi_1(s), \pi_2(s)) = \min_{\pi_2(s)} \max_{\pi_1(s)} q_1(s, \pi_1(s), \pi_2(s))$

- $q_1(s, a_1, a_2) = q_1(s, a_1, a_2)$
  $+\alpha \left(r_i + \max_{\pi_1} \min_{a_2} q_1(s, a_1, a_2) - q_1(s, a_1 a_2) \right)$

Converges under Robbins Munro conditions!

Nash-Q

- $q_i(s, a_1, a_2) = q_i(s, a_1, a_2) + \alpha\big(r_i + v_i(s') - q_i(s, a_1 a_2)\big)$
- $v_i(s') = E_{\pi_1, \pi_2}[q_i(s', a_1{}', a_2{}')]$
- Need a way of predicting what the other player will do
- Compute an equilibrium
- Converges under more complex conditions

Three Learning Settings

- Planning with known dynamics
- Reinforcement Learning
- Simulation
- With simulator, can reset and "roll out" repeatedly from a state

Rollout Algorithms

- I am in state s and have a policy π. What should I do?
- Want to learn something smarter than just $\pi(s)$
- Idea: estimate $q_\pi(s, a)$ via Monte Carlo
- Intuitively converges via Policy Improvement Theorem!

Rollout Algorithms

Rollout Algorithms

Monte Carlo Tree Search (MCTS)

Repeat until bored:

1) Selection: Use current estimates to choose a leaf of tree policy
2) Expansion: (Optional) add children of leaf to the tree policy
3) Simulation: Roll out from (new) leaf
4) Backup: Update all relevant MC estimates based on return

Monte Carlo Tree Search (MCTS)

Multi-agent Monte Carlo Tree Search (MCTS)

Summary

- MARL – Need to predict actions of others
  - Not too hard in two-player, zero-sum settings
  - Tricky in general

- Rollouts
  - Use Monte Carlo estimates for policy improvement
  - Can focus search (MCTS)
  - Works even with multiple agents