A Heuristic for Encouraging Cooperation in Multi-Agent Reinforcement Learning

Martin Allen    Douglas MacFarland

Abstract
When multiple individuals work together as a team with a common goal, they can achieve much that is out of reach to a single agent. Unfortunately, team-work involves a lot of complexity. In particular, when agents work in decentralized environments, where each individual has its own private information, not always shared with all its teammates, optimal planning and decision making quickly grow infeasible. Multi-agent reinforcement learning (RL) is an approach that approximates solutions while avoiding the problems inherent in full planning. This work extends single-agent Q-learning, a popular RL technique, to the multiagent case. The main drawback of Q-learning is that, like most RL techniques, Q-learning involves random exploration of the state and action space of the problem. In the cooperative multiagent case, chance occurrences of coordination can become very unlikely. This work examines ways in which the technique can be improved by the introduction of heuristics that guide the learning process in non-random ways.

Reinforcement Learning

Agents observe, act, and receive reward in an uncertain world.

- A shared reward function leads to cooperation
- Locally, partial information leads to decentralization
- Actions affect a shared environment, leading to interaction
- All these factors combine to make coordination very difficult.

The Q-Learning Algorithm

\[Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]\]

One step Q-Learning

- Initialize \(Q(s, a)\) arbitrarily.
- Initialize \(\epsilon\).
- Repeat (for each step of episode):
  - Choose \(a\) for \(s\) using policy derived from \(Q\) (e.g., \(\epsilon\)-greedy).
  - Take action \(a\), observe \(r, s'\).
  - \(Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_a Q(s', a') - Q(s, a)]\)
- until \(s\) is terminal.

Q-Learning pseudo-code

Box Pushing Variant

Parameters

- \(\alpha\) (5) - The learning rate.
- \(\epsilon\) (5) - The probability of taking an off-policy exploratory action.
- \(\gamma\) (5) - The discount factor.
- -reduce (500,000) - The number of iterations before the value of \(\epsilon\) is reduced.
- move penalty (-1) - The reward penalty for moving.
- wait penalty (1.2) - The reward penalty for waiting.
- failure penalty (-1,-5) - The reward penalty for taking an action that failed.
- small reward (50) - The reward for getting a small box into the goal zone.
- big reward (1,000) - The reward for getting the large box into the goal zone.
- pushing probability (1.5) - The probability of succeeding to push a box when an attempt is made.

Challenges

- Complexity - The use of a heuristic does not remove the inherent complexity of the problem.
- Applicability - It is not trivial to find a heuristic that is both useful and that preserves the decentralized nature of the problem.
- Sensitivity - As shown on the right, the heuristic may be sensitive to some of the parameters.
- Resources - The current heuristic does require a one-time generation of the full state-transition probability table. For more complex problems, generating that table could be incredibly resource intensive.

Empirical Data

Empirical Data

Contact Information

Martin Allen   Douglas MacFarland
(mallen@cs.uwlax.edu)   (macfarla.doug@uwlax.edu)
Department of Computer Science
University of Wisconsin - La Crosse
La Crosse, WI 54601