Multi-Agent Reinforcement Learning

Yashashwi Singhania

IIT BHU

April 2024

Introduction

Multiagent system can be defined as a group of autonomous, interacting entities sharing a common environment, which they percieve with sensors and upon which act with actuators.

The reward feedback system is more in supervised learning than in RL due to agents always doing the correct action, but least in unsupervised because the agent has to discover correct actions on its own without any performance feedback.
Therefore we use RL because of its simplicity and generality.

Different Games

Stochastic Game:

A stochastic game (SG) is a tuple $\langle X, U_1, \cdots, U_n, \rho_1, \cdots, , \rho_n \rangle$, where n is the number of agents, X is the discrete set of environment states, U_i are the discrete sets of actions available to the agents, yielding the joint action set $\mathbf{U} = U_1 \times \cdots \times U_n$, $f : X \times \mathbf{U} \times X \to [0, 1]$ is the state transition probability function, and $\rho_i : X \times \mathbf{U} \times X \to \mathbb{R}$, are the reward functions of the agents.

Static Game:

A static game is a SG with $X = \emptyset$. When played repeatedly by the same agents, the static game is called a **repeated game.**

Stage Game:

A stage game is the static game that arises when the state of an SG is fixed to some value. Since in general the agents visit the same state of an SG multiple times, the stage game is a **repeated game.**

Benefits of MARL

A speedup of MARL can be realized thanks to parallel computation when the agents exploit decentralizes structured of task.

Sharing experience helps the agent learn faster and better about similar tasks. For instance-

- Agents can exchange information using communication.
- Skilled agents may serve as teachers for the learner.
- The learner may watch and imitate the skilled agents.

When one or more agents fail in a multiagent system, the remaining agents can take over some of their tasks. This implies that MARL is inherently robust. Furthermore, by design, most multiagent systems also allow the easy insertion of new agents into the system, leading to a high degree of scalability.

Challenges in MARL

1. **Curse of Dimensionality:** As the number of agents and the dimensionality of the state and action spaces increase, the computational complexity of MARL grows exponentially.
2. **Difficulty in Specifying Goals:** Defining a good goal in MARL is challenging because agents' returns are correlated, and maximizing one agent's return may affect others.
3. **Nonstationarity:** Multi-agent learning is inherently nonstationary because all agents are learning simultaneously, causing the environment to change as each agent updates its policy.
4. **Exploration-Exploitation Tradeoff:** MARL faces the exploration-exploitation tradeoff, where agents must balance between exploiting their current knowledge and exploring to gather more information.
5. **Need for Coordination:** The effects of an agent's actions depend on the actions of other agents, requiring coordination for effective behavior.

Goals and Strategies

The learning goals in MARL incorporate two main aspects: **stability** and **adaptation**. Stability refers to the convergence of the learning dynamics of an agent to a stationary policy. Adaptation refers to the ability of an agent to maintain or improve performance as other agents change their policies.

The goals are typically formulated for static games. Some of these goals can be extended to dynamic games by requiring that the conditions are satisfied stagewise for all states of the dynamic game.

Convergence to equilibria is a basic stability requirement. It means the agents strategies should eventually converge to a coordinated equilibrium. Nash equilibria are most frequently used. Simlarly rationality is added as an adaptation criterion. Rationality is defined as the requirement that the agent converges to a best response when the other agents remain stationary.

Goals and Strategies

An alternative to rationality is no-regret concept. It is defined as the requirement that the agent achieves a return that is at least as good as the return of any stationary strategy, regardless of the strategies of the other agents.

Targeted Optimality are adaptation requirements expressed in the form of average reward bounds. Targeted optimality demands an average reward, against a targeted set of algorithms, which is at least the average reward of a best response. Compatibility prescribes an average reward level in self-play, i.e., when the other agents use the learner's algorithm. Safety demands a safety-level average reward against all other algorithms.

An opponent-independent algorithm converges to a strategy that is part of an equilibrium solution regardless of what the other agents are doing. An opponent-aware algorithm learns models of the other agents and reacts to them using some form of best response.

Algorithms

A. Fully Cooperative Tasks

In a fully cooperative SG, the agents have the same reward function and the learning goal is to maximize the common discounted return. If a centralized controller were available, the task would reduce to a MDP. In this case, the goal could be achieved by learning the optimal joint-action values with Q-learning

$$Q_{k+1}(x_k, \mathbf{u}_k) = Q_k(x_k, \mathbf{u}_k) + \alpha[r_{k+1} + \gamma \max_{\mathbf{u'}} Q_k(x_{k+1}, \mathbf{u'}) - Q_k(x_k, \mathbf{u}_k)] \quad (1)$$

and using the greedy policy. However, the agents are independent decision makers, and a coordination problem arises even if all the agents learn in parallel the common optimal Q-function. It might seem that the agents could use greedy policies applied to Q to maximize the common return.

$$\overline{h^*}_i(x) = \arg\max_{u_i} \max_{u_1,\ldots,u_{i-1},u_{i+1},\ldots,u_n} Q^*(x, u) \quad (2)$$

Algorithms

A. Fully Cooperative Tasks - (1) *Coordination-Free Methods*

The Team Q-learning algorithm avoids the coordination problem by assuming that the optimal joint actions are unique. Then, if all the agents learn the common Q-function in parallel with (1), they can safely use (2) to select these optimal joint actions and maximize their return.

The Distributed Q-learning algorithm solves the cooperative task,only in a deterministic problems, without assuming coordination and with limited computation. Each agent maintains a local policy $\overline{h}_i(x)$, and a local Q-function $Q_i(x, u_i)$, depending only on its own action. The local Q-values are updated only when the update leads to an increase in the Q-value

$$Q_{i,k+1}(x_k, u_k) = \max_{u_i} \left(Q_{i,k}(x_k, u_k), r_{k+1} + \gamma \max_{u_i} Q_{i,k}(x_{k+1}, u_i) \right) \qquad (3)$$

Algorithms

A. Fully Cooperative Tasks - (1) *Coordination-Free Methods*

This ensures that the local Q-value always captures the maximum of the joint-action Q-values. The local policy is updated only if the update leads to an improvement in the Q-values:

$$\overline{h}_{i,k+1}(x_k) = \begin{cases} u_{i,k} & \text{if } \max_{u_i} Q_{i,k+1}(x_k, u_i) > \max_{u_i} Q_{i,k}(x_k, u_i), \\ h_{i,k}(x_k) & \text{otherwise.} \end{cases} \quad (4)$$

This ensures that the joint policy is always optimal with respect to the global Q_k. Under the conditions that the reward function is positive and $Q_{i,0} = 0 \ \forall i$, the local policies of the agents provably converge to an optimal joint policy.

Algorithms

A. Fully Cooperative Tasks - (2) *Coordination-Based Methods*

Coordination graphs simplify coordination when the global Q-function can be additively decomposed into local Q-functions that only depend on the actions of a subset of agents. For instance, in an SG with four agents, the decomposition might be:

$$Q(x, u) = Q_1(x, u_1, u_2) + Q_2(x, u_1, u_3) + Q_3(x, u_3, u_4)$$

The decomposition might be different for different states. Typically (like in this example), the local Q-functions have smaller dimensions than the global Q-function. Maximization of the joint Q-value is done by solving simpler, local maximizations in terms of the local value functions, and aggregating their solutions. Under certain conditions, coordinated selection of an optimal joint action is guaranteed.

Algorithms

A. Fully Cooperative Tasks - (3) *Indirect Coordination Methods*

Indirect coordination methods bias action selection toward actions that are likely to result in good rewards or returns.

a.) Static Tasks: Joint Action Learners learn joint- action values and employ empirical models of the other agents strategies. Agent i learns models for all the other agents $j \neq i$, using

$$\hat{\sigma}_j^i(u_j) = \frac{C_j^i(u_j)}{\sum_{\widetilde{u}_j \in U_j} C_j^i(\widetilde{u}_j)} \tag{5}$$

where $\hat{\sigma}_j^i$ is agent i 's model of agent j 's strategy and $C_j^i(u_j)$ counts the number of times agent i observed agent j taking action u_j.

Algorithms

A. Fully Cooperative Tasks - (3) *Indirect Coordination Methods*

The Frequency Maximum Q-value heuristic is based on the frequency with which actions yielded good rewards in the past. Agent i uses Boltzmann action selection, plugging in modified Q-values $\tilde{Q}_i$ computed with the formula

$$\tilde{Q}_i(u_i) = Q_i(u_i) + \nu \frac{C^i_{\max}(u_i)}{C^i(u_i)} r_{\max}(u_i)$$

where $r_{\max}(u_i)$ is the maximum reward observed after taking action u_i, $C^i_{\max}(u_i)$ counts how many times this reward has been observed, $C^i(u_i)$ counts how many times u_i has been taken, and ν is a weighting factor. The algorithm only works for deterministic tasks, where variance in the rewards resulting from the agent's actions can only be the result of the other agents' actions.

Algorithms

A. Fully Cooperative Tasks - (3) *Indirect Coordination Methods*

b.) Dynamic Tasks:

- In Optimal Adaptive Learning, virtual games are constructed on top of each stage game of the SG. Optimal joint actions are rewarded 1, rest are rewarded 0.
- An algorithm is introduced that, by biasing the agent toward recently selected optimal actions, guarantees convergence to a coordinated optimal joint action for the virtual game, and therefore, to a coordinated joint action for the original stage game.
- Thus, OAL provably converges to optimal joint policies in any fully cooperative SG. It is the only currently known algorithm capable of achieving this.
- This, however, comes at the cost of increased complexity: each agent estimates empirically a model of the SG, virtual games for each stage game, models of the other agents, and an optimal value function for the SG.

Algorithms

A. Fully Cooperative Tasks - (4) *Open Issues*

- All methods rely on exact state measurements and often need precise measurements of other agents' actions.
- Any perception difference among agents can disrupt the consistency of Q-functions and policies.
- Communication can help ensure consistency by allowing data exchange.
- Most algorithms face the curse of dimensionality, except Distributed Q-learning and FMQ, which work in restricted settings.

Algorithms

B. Explicit Coordination Mechanisms

- An approach to solve the coordination problem is to ensure that ties are broken by all agents in the same way. This requires coordination or negotiation of random action choices.
- Mechanisms are based on social conventions, roles, and communication and can be used for any type of task.
- Both social conventions and roles restrict the action choices of the agents. An agent role restricts the set of actions available to that agent prior to action selection.

Algorithms

B. Explicit Coordination Mechanisms

- Social conventions encode a priori preferences toward certain joint actions, and help break ties during action selection. If properly designed, roles or social conventions eliminate ties completely.
- A simple social convention relies on a unique ordering of agents and actions. These two orderings must be known to all agents. Combining them leads to a unique ordering of joint actions, and coordination is ensured if the first joint action in this ordering is selected by all the agents.
- Communication can be used to negotiate action choices, either alone or in combination with the aforementioned techniques. When combined with these techniques, communication can relax their assumptions and simplify their application.

Algorithms

C. Fully Competitive Tasks

In a fully competitve game ($\sum \rho_i = 0$) the minimax principle can be applied: maximize one's benefit under the worst-case assumption that the opponent will always endeavor to minimize it.
The minimax- Q algorithm employs the minimax principle to compute strategies and values for the stage games, and a temporal-difference rule to propagate the values across state-action pairs.

$$
\begin{aligned}
h_{1,k}\left(x_k, \cdot\right) &= \arg \mathbf{m}_1\left(Q_k, x_k\right) \\
Q_{k+1}\left(x_k, u_{1,k}, u_{2,k}\right) &= Q_k\left(x_k, u_{1,k}, u_{2,k}\right) \\
&\quad + \alpha\left[r_{k+1} + \gamma \mathbf{m}_1\left(Q_k, x_{k+1}\right)\right. \\
&\quad \left. - Q_k\left(x_k, u_{1,k}, u_{2,k}\right)\right]
\end{aligned}
$$

$\mathbf{m}_1$ is the minimax return of agent 1

$$
\mathbf{m}_1(Q, x) = \max_{h_1(x,\cdot)} \min_{u_2} \sum_{u_1} h_1\left(x, u_1\right) Q\left(x, u_1, u_2\right)
$$

Algorithms

D. Mixed Tasks

In mixed SGs, no constraints are imposed on the reward functions of the agents. Sometimes even cooperating agents may encounter situations where their immediate interests are in conflict.
A significant number of algorithms in this category are designed only for static tasks (i.e., repeated, general-sum games). In repeated games, one of the essential properties of RL, delayed reward, is lost. However, the learning problem is still nonstationary due to the dynamic behavior of the agents that play the repeated game. This is why most methods in this category focus on adaptation to other agents.

1. Single-Agent RL: Q-learning can be directly applied to the multiagent cases. However, the nonstationarity of the MARL problem invalidates most of the single-agent RL theoretical guarantees. Despite the limitations these are widely applied mainly because of its simplicity.

Algorithms

D. Mixed Tasks - (2) *Agent-Independent Methods*

Algorithms that are inde- pendent of the other agents share a common structure based on Q-learning, where policies and state values are computed with game-theoretic solvers for the stage games arising in the states of the SG. This is similar to minimax-Q algorithm; the only difference is that for mixed games, solvers can be different from minimax.

Denoting by $\{Q\cdot,_k(x,\cdot)\}$ the stage game arising in state x and given by all the agents' Q-functions at time k, learning takes place according to

$$
\begin{aligned}
&h_{i,k}(x,\cdot) = \text{solve}_i\left\{Q\cdot,_k(x_k,\cdot)\right\}\\
&Q_{i,k+1}(x_k,\boldsymbol{u}_k) = Q_{i,k}(x_k,\boldsymbol{u}_k)\\
&+\alpha\left[r_{i,k+1}+\gamma\cdot\text{eval}_i\left\{Q\cdot,_k(x_{k+1},\cdot)\right\}\right.\\
&\left.-Q_{i,k}(x_k,\boldsymbol{u}_k)\right]
\end{aligned}
$$

solve$_i$some type of equilibrium (a strategy), and **eval**$_i$ gives expected return given this equilibrium.

Algorithms

D. Mixed Tasks - (3) *Agent-Tracking Methods*

Agent-tracking algorithms estimate models of the other agents' strategies or policies and act using some form of best-response to these models. Convergence to stationary strategies is not a requirement. Each agent is assumed capable to observe the other agents' actions.

a.) Static Methods: In the fictitious play algorithm, agent i acts at each iteration according to a best response to the models $\hat{\sigma}^i_1, \ldots, \hat{\sigma}^i_{i-1}, \hat{\sigma}^i_{i+1}, \ldots, \hat{\sigma}^i_n$. The models are computed empirically using (5). Fictitious play converges to a Nash equilibrium in certain restricted classes of games, among which are fully cooperative, repeated games.

The MetaStrategy algorithm, combines modified versions of fictitious play, minimax, and a gametheoretic strategy to achieve the targeted optimality, compatibility, and safety goals.

To compute best responses, the fictitious play and MetaStrategy algorithms require a model of the static task, in the form of reward functions.

Algorithms

D. Mixed Tasks - (3) *Agent-Tracking Methods*

The Hyper- Q algorithm uses the other agents' models as a state vector and learns a Q-function $Q_i\left(\hat{\sigma}_1, \ldots, \hat{\sigma}_{i-1}, \hat{\sigma}_{i+1}, \ldots, \hat{\sigma}_n, u_i\right)$ with an update rule similar to Q-learning. By learning values of strategies instead of only actions, Hyper$Q$ should be able to adapt better to nonstationary agents. One inherent difficulty is that the action selection probabilities in the models are continuous variables. This means the classical, discrete-state Q-learning algorithm cannot be used. Less understood, approximate versions of it are required instead.

b.) Dynamic Tasks: The Nonstationary Converging Policies (NSCP) algorithm computes a best response to the models and uses it to estimate state values.

Algorithms

D. Mixed Tasks - (3) *Agent-Tracking Methods*

$$
\begin{aligned}
h_{i,k}\left(x_k, \cdot\right) &= \arg \mathbf{br}_i\left(Q_{i,k}, x_k\right) \\
Q_{i,k+1}\left(x_k, \boldsymbol{u}_k\right) &= Q_k\left(x_k, \boldsymbol{u}_k\right) + \alpha\left[r_{i,k+1} + \gamma \mathbf{br}_i\left(Q_{i,k}, x_{k+1}\right)\right. \\
&\quad \left. - Q_k\left(x_k, \boldsymbol{u}_k\right)\right]
\end{aligned}
$$

where the best-response value operator br is implemented as

$$
\mathbf{br}_i\left(Q_i, x\right) = \max_{h_i(x,\cdot)} \sum_{u_1,\ldots,u_n} h_i\left(x, u_i\right)
$$

$$
Q_i\left(x, u_1, \ldots, u_n\right) \prod_{j=1, j\neq i}^{n} \hat{h}_j^i\left(x, u_j\right).
$$

$$
\hat{h}_j^i\left(x, u_j\right) = \frac{C_j^i\left(x, u_j\right)}{\sum_{\tilde{u}_j \in U_j} C_j^i\left(x, \tilde{u}_j\right)}
$$

Algorithms

D. Mixed Tasks - (4) *Agent-Aware Methods*

Agent-aware algorithms target convergence, as well as adaptation to the other agents. Some algorithms provably converge for particular types of tasks (mostly static), others use heuristics for which convergence is not guaranteed.

a.) Static Methods: The AWESOME algorithm uses fictitious play, but monitors the other agents and, when it concludes that they are nonstationary, switches from the best response in fictitious play to a centrally precomputed Nash equilibrium.

IGA and WoLF-IGA work in two-agent, two-action games, and use similar gradient update rules

$$\begin{cases} \alpha_{k+1} = \alpha_k + \delta_{1,k} \dfrac{\partial E\left\{r_1 \mid \alpha, \beta\right\}}{\partial \alpha} \\ \beta_{k+1} = \beta_k + \delta_{2,k} \dfrac{\partial E\left\{r_2 \mid \alpha, \beta\right\}}{\partial \beta} \end{cases} \tag{6}$$

Algorithms

D. Mixed Tasks - (4) *Agent-Aware Methods*

b.) Dynamic Tasks: Win-or-Learn-Fast Policy Hill-Climbing (WoLF-PHC) is a heuristic algorithm that updates Q-functions with the Q-learning rule, and policies with a WoLF rule inspired from (6)

$$
\begin{aligned}
h_{i,k+1}(x_k, u_i) &= h_{i,k}(x_k, u_i) \\
&+ \begin{cases} \delta_{i,k} & \text{if } u_i = \arg\max_{\tilde{u}_i} Q_{i,k+1}(x_k, \tilde{u}_i) \\ -\frac{\delta_{i,k}}{|U_i|-1} & \text{otherwise} \end{cases} \qquad (7) \\
\delta_{i,k} &= \begin{cases} \delta_{\text{win}} & \text{if winning} \\ \delta_{\text{lose}} & \text{if losing.} \end{cases}
\end{aligned}
$$

The Extended Optimal Response (EXORL) heuristic applies a complementary idea in two-agent tasks: the policy update is biased in a way that minimizes the other agent's incentive to deviate from its current policy. Thus, convergence to a coordinated Nash equilibrium is expected.

Algorithms

D. Mixed Tasks - (5) *Open Issues*

- **Limited Applicability**: Static, repeated games have limited applicability due to their inability to address dynamic scenarios.
- **Extension to Dynamic Settings**: Algorithms must be extended to dynamic settings to broaden their applicability.
- **Challenges with Exact Task Models**: Most algorithms assume exact task models, which are rarely available.
- **Curse of Dimensionality and Sensitivity**: Many algorithms suffer from the curse of dimensionality and sensitivity to imperfect observations.
- **Bias towards Static Solutions**: Game theory biases towards static solutions in dynamic settings, potentially leading to ineffective strategies.
- **Understanding Single-Agent RL in Mixed SGs**: Research is needed to understand the conditions under which single-agent RL techniques work effectively in mixed stochastic games.