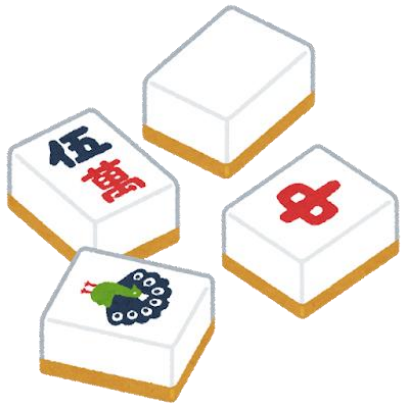
Sample-Efficient Reinforcement Learning of Partially Observable Markov Games

Qinghua Liu
Csaba Szepesvári
Chi Jin

Abstract

- This paper discusses reinforcement learning for player AI in multiplayer general **Partially Observable Markov Games (POMGs)**—competitive games in which players are only given partial information—as well as **Multi-Agent Reinforcement Learning (MARL).**



Mahjong

Real-time strategy game

Multi-agent robot system

Abstract

- In one of the MARL settings—**self-play**—a simple algorithm combining **optimism** and **maximum likelihood estimation (MLE)** proves useful.
- In settings involving **adversarial opponents**, variants of this algorithm yield better results.

Introduction

- Multi-Agent Reinforcement Learning under Partial Observability (MARL)

For example, in poker, a player's hand is hidden from other players.

When information is hidden, it is **difficult** for agents to learn and plan. This is because the current state alone provides **insufficient information**, requiring agents to keep track of past states, among other reasons. (Non-Markovian nature)

Introduction

- This paper examines a general mathematical model known as **Partially Observable Markov Games (POMGs).**

POMGs... involve multiple players, each of whom possesses only their own information.
Examples include games like UNO.
Agents must infer what is happening based solely on the information they can observe.

Introduction

- In this paper, we develop a user-friendly algorithm called **“weakly revealing POMGs.”**
  This approach **eliminates the worst-case scenario** in POMGs, where no information is obtained and reaching the correct solution requires an enormous number of trials.
  ・We do not consider the case where all players have absolutely no information.
  ・We guarantee that taking an action will yield some kind of hint (information that distinguishes between states).

Preliminary

- **Novelty** of This Paper
  - **Formulation** of multi-player POMGs

$$(H, S, \{\mathcal{A}_i\}_{i=1}^n, \{\mathcal{O}_i\}_{i=1}^n; \mathbb{T}, \mathbb{O}, \mu_1; \{r_i\}_{i=1}^n)$$

H: Length of a single episode
S: State space (set of all possible states)
Ai: Action space of the i-th player
Oi: Observation space of the i-th player
T: Set of transition matrices. The probabilities of the next state occurring.
O: Set of emission matrices. The probability distribution of what the player observes in the current state
ri: Reward function of the i-th player

3 WeaklyRevealing Partially Observable Markov Games

- This chapter defines the prerequisites for efficient learning even in partially observable Markov generalizations (POMGs).

  1、Undercomplete POMGs
  2、Overcomplete POMGs

3 WeaklyRevealing Partially Observable Markov Games

- 1、Undercomplete POMGs

A scenario in which the number of **observations (O)** is greater than or equal to the number of possible **states (S).**($S < O$)

Example: National Flag Quiz

3 WeaklyRevealing Partially Observable Markov Games

- 1、Undercomplete POMGs

Number of states (S)……Number of countries (196)

Number of observations (O)……Colors and shapes * 196

Jamaica

A situation where there are more clues than answers

3 WeaklyRevealing Partially Observable Markov Games

- 1、Undercomplete POMGs
  This proves that no matter how similar two different states may be, when the observations of all agents are combined, the agents **can recognize them as “different states.”**

Algorithm 1 OMLE-Equilibrium

1: **Initialize:** $\mathcal{B}^1 = \{\hat{\theta} \in \Theta : \ \min_h \sigma_S(\hat{\mathbb{O}}_h) \geq \alpha\}$, $\mathcal{D} = \{\}$
2: **for** $k = 1, \ldots, K$ **do**
3: compute π^k =Optimistic_Equilibrium$(\mathcal{B}^k)$
4: follow π^k to collect a trajectory $\tau^k = (\mathbf{o}_1^k, \mathbf{a}_1^k, \ldots, \mathbf{o}_H^k, \mathbf{a}_H^k)$
5: add (π^k, τ^k) into $\mathcal{D}$ and update

$$\mathcal{B}^{k+1} = \left\{\hat{\theta} \in \Theta : \sum_{(\pi,\tau)\in\mathcal{D}} \log \mathbb{P}_{\hat{\theta}}^{\pi}(\tau) \geq \max_{\theta'\in\Theta} \sum_{(\pi,\tau)\in\mathcal{D}} \log \mathbb{P}_{\theta'}^{\pi}(\tau) - \beta\right\} \bigcap \mathcal{B}^1$$

6: output π^{out} that is sampled uniformly at random from $\{\pi^k\}_{k\in[K]}$

3 WeaklyRevealing Partially Observable Markov Games

- 2、 Overcomplete POMGs
  There are far more hidden states than observations (S > O), making this a more complex and realistic scenario.
  Because there is **too little information** from the observations, a single observation is **insufficient to distinguish** between states.

  ―> Introduce Assumption 2 (multistep α -weakly revealing condition)

3 WeaklyRevealing Partially Observable Markov Games

- 2、 Overcomplete POMGs
  Assumption 2 (multistep α-weakly revealing condition)
  By observing a sequence of actions over **multiple steps**, it is **possible to distinguish** between hidden states.
  Even if a single step is insufficient for discrimination, combining multiple steps can yield weakly revealing information.

4 Learning Equilibria with Self-play

- 4.1 Undercomplete partially observable Markov games
  When the number of observations exceeds the number of states. (Case 3.1)

  OMLE-Equilibrium
  An algorithm that combines **optimism** and **maximum likelihood estimation (MLE)**

Algorithm 1 OMLE-Equilibrium

Initialize: $\mathcal{B}^1 = \{\hat{\theta} \in \Theta : \ \min_h \sigma_S(\hat{\mathbb{O}}_h) \geq \alpha\}, \mathcal{D} = \{\}$
for $k = 1, \ldots, K$ **do**
  compute π^k =Optimistic_Equilibrium$(\mathcal{B}^k)$
  follow π^k to collect a trajectory $\tau^k = (\mathbf{o}_1^k, \mathbf{a}_1^k, \ldots, \mathbf{o}_H^k, \mathbf{a}_H^k)$
  add (π^k, τ^k) into $\mathcal{D}$ and update

$$\mathcal{B}^{k+1} = \left\{\hat{\theta} \in \Theta : \sum_{(\pi,\tau)\in\mathcal{D}} \log \mathbb{P}_{\hat{\theta}}^{\pi}(\tau) \geq \max_{\theta'\in\Theta} \sum_{(\pi,\tau)\in\mathcal{D}} \log \mathbb{P}_{\theta'}^{\pi}(\tau) - \beta\right\} \bigcap \mathcal{B}^1$$

output π^{out} that is sampled uniformly at random from $\{\pi^k\}_{k\in[K]}$

4 Learning Equilibria with Self-play

- 4.1 Undercomplete partially observable Markov games
  ・Optimistic equilibrium computation
  The agent selects the model that is **most advantageous to itself** given the current situation.
  Without considering the other players, it first makes the choice that is most beneficial to itself.

  ・Confidence set update
  Continue running the **selected model** and use the resulting data to narrow down the models based on the principle of “maximum likelihood estimation (MLE).”
  From the actual data and the set of models, eliminate those with the lowest consistency.

4 Learning Equilibria with Self-play

- 4.1 Undercomplete partially observable Markov games

Subroutine 1 Optimistic_Equilibrium($\mathcal{B}$)

1: **for** $i \in [n]$ **do**
2: let $\overline{V}_i \in \mathbb{R}^{|\Pi_1^{\text{det}}| \times \cdots \times |\Pi_n^{\text{det}}|}$ with its π^{th} entry equal to $\sup_{\hat{\theta} \in \mathcal{B}} V_i^{\pi}(\hat{\theta})$ for $\pi \in \Pi_1 \times \cdots \times \Pi_n$
3: return EQUILIBRIUM$(\overline{V}_1, \ldots, \overline{V}_n)$

Calculation of the specific **optimistic equilibrium**:
For every player i and every combination π of fixed strategies for all players, the **upper bound on value, V^{π}_i**, is calculated.

4 Learning Equilibria with Self-play

- 4.2 Overcomplete partially observable Markov games

“Learning Methods in a More Challenging and Realistic Setting Where the Number of Hidden States (S) Exceeds the Number of Observations (O)”(3.2)

Challenges:

- In a single step, there is **insufficient information** to identify the current latent state.
- In a single step, it is **impossible to distinguish** between different groups of states.

4 Learning Equilibria with Self-play

- 4.2 Overcomplete partially observable Markov games
  Algorithm 2…An improved version of Algorithm 1

Algorithm 2 multi-step OMLE-Equilibrium

Initialize: $\mathcal{B}^1 = \{\hat{\theta} \in \Theta : \ \min_h \sigma_S(\hat{\mathbb{M}}_h) \geq \alpha\}$, $\mathcal{D} = \{\}$
for $k = 1, \ldots, K$ **do**
  compute π^k =Optimistic_Equilibrium$(\mathcal{B}^k)$
  for $h = 0, \ldots, H - m$ **do**
    execute policy $\pi^k_{1:h} \circ \text{uniform}(\mathcal{A})$ to collect a trajectory $\tau^{k,h}$
    then add $(\pi^k_{1:h} \circ \text{uniform}(\mathcal{A}), \tau^{k,h})$ into $\mathcal{D}$
  update $\mathcal{B}^{k+1} = \left\{\hat{\theta} \in \Theta : \sum_{(\pi,\tau)\in\mathcal{D}} \log \mathbb{P}^{\pi}_{\hat{\theta}}(\tau) \geq \max_{\theta'\in\Theta} \sum_{(\pi,\tau)\in\mathcal{D}} \log \mathbb{P}^{\pi}_{\theta'}(\tau) - \beta\right\} \bigcap \mathcal{B}^1$
output π^{out} that is sampled uniformly at random from $\{\pi^k\}_{k\in[K]}$

Algorithm 1 OMLE-Equilibrium

Initialize: $\mathcal{B}^1 = \{\hat{\theta} \in \Theta : \ \min_h \sigma_S(\hat{\mathbb{O}}_h) \geq \alpha\}$, $\mathcal{D} = \{\}$
for $k = 1, \ldots, K$ **do**
  compute π^k =Optimistic_Equilibrium$(\mathcal{B}^k)$
  follow π^k to collect a trajectory $\tau^k = (\mathbf{o}^k_1, \mathbf{a}^k_1, \ldots, \mathbf{o}^k_H, \mathbf{a}^k_H)$
  add (π^k, τ^k) into $\mathcal{D}$ and update
    $\mathcal{B}^{k+1} = \left\{\hat{\theta} \in \Theta : \sum_{(\pi,\tau)\in\mathcal{D}} \log \mathbb{P}^{\pi}_{\hat{\theta}}(\tau) \geq \max_{\theta'\in\Theta} \sum_{(\pi,\tau)\in\mathcal{D}} \log \mathbb{P}^{\pi}_{\theta'}(\tau) - \beta\right\} \bigcap \mathcal{B}^1$
output π^{out} that is sampled uniformly at random from $\{\pi^k\}_{k\in[K]}$

4 Learning Equilibria with Self-play

- 4.2 Overcomplete partially observable Markov games
- Active Sampling
  Algorithm 2

In addition to the **optimistic model** π k, we collect data by **randomly selecting actions** from intermediate step h

5: execute policy $\pi^k_{1:h} \circ \text{uniform}(\mathcal{A})$ to collect a trajectory $\tau^{k,h}$
then add $(\pi^k_{1:h} \circ \text{uniform}(\mathcal{A}), \tau^{k,h})$ into $\mathcal{D}$

Algorithm 1
We are using **only the optimistic model** π k

4: follow π^k to collect a trajectory $\tau^k = (\mathbf{o}^k_1, \mathbf{a}^k_1, \ldots, \mathbf{o}^k_H, \mathbf{a}^k_H)$

4 Learning Equilibria with Self-play

- 4.2 Overcomplete partially observable Markov games
- Active Sampling
  Algorithm 2

Perform a series of observations across **multiple steps** and narrow down the models to those that meet the threshold α.

Algorithm 2 multi-step OMLE-Equilibrium

1: **Initialize:** $\mathcal{B}^1 = \{\hat{\theta} \in \Theta : \min_h \sigma_S(\hat{\mathbb{M}}_h) \geq \alpha\}, \mathcal{D} = \{\}$

Algorithm 1
Narrow down the models based on whether they meet the α criterion starting from the **one step**

Algorithm 1 OMLE-Equilibrium

1: **Initialize:** $\mathcal{B}^1 = \{\hat{\theta} \in \Theta : \min_h \sigma_S(\hat{\mathbb{O}}_h) \geq \alpha\}, \mathcal{D} = \{\}$

4 Learning Equilibria with Self-play

- 4.2 Overcomplete partially observable Markov games
- Active Sampling
  Algorithm 2
  Even if a single step isn’t enough to determine the outcome, it can distinguish based on the **history of multiple steps**.
  It is actively working to extract information (the randomness mentioned earlier).

5 Playing against Adversarial Opponents

The **previous chapters** covered the **self-play setup**, where you learn by playing against yourself.

In this chapter, we'll move on to a more practical and challenging scenario where **you can only control your own player** and have **no idea what hostile strategies the other players** will employ.

5 Playing against Adversarial Opponents

- 5.1 Statistical hardness

When fighting hostile opponents in a **POMG** (a setting where each player can only see their own observations and actions).

This section demonstrates that, no matter how favorable the conditions for the learner may be, it is **impossible to minimize the damage.**

・Even when the enemy's status is known after a single observation and they are using only fixed strategies that are completely predictable in advance, it is **said that damage cannot be minimized.**

5 Playing against Adversarial Opponents

- 5.2 Positive results

To address the issue of statistical hardness(5.1), we are introducing a game replay feature. After a match ends, players will be able to view all observations and action logs from every other player.
例）雀魂

5 Playing against Adversarial Opponents

- 5.2 Positive results

Algorithm 3 OMLE-Adversary

1: **Initialize:** $\mathcal{B}^1 = \{\hat{\theta} \in \Theta : \sigma_S(\hat{\mathbb{O}}) \geq \alpha\}, \mathcal{D} = \{\}$
2: **for** $k = 1, \ldots, K$ **do**
3: learner computes $(\cdot, \pi_1^k) = \text{argmax}_{\hat{\theta} \in \mathcal{B}^k, \hat{\pi}_1 \in \Pi_1} \min_{\hat{\pi}_{-1} \in \Pi_{-1}} V_1^{\hat{\pi}_1 \times \hat{\pi}_{-1}}(\hat{\theta})$
4: opponents pick policies π_{-1}^k
5: execute policy $\pi^k = \pi_1^k \times \pi_{-1}^k$ to collect $\tau^k = (\mathbf{o}_1^k, \mathbf{a}_1^k, \ldots, \mathbf{o}_H^k, \mathbf{a}_H^k)$
6: add (π^k, τ^k) into $\mathcal{D}$ and update

$$\mathcal{B}^{k+1} = \left\{ \hat{\theta} \in \Theta : \sum_{(\pi,\tau) \in \mathcal{D}} \log \mathbb{P}_{\hat{\theta}}^{\pi}(\tau) \geq \max_{\theta' \in \Theta} \sum_{(\pi,\tau) \in \mathcal{D}} \log \mathbb{P}_{\theta'}^{\pi}(\tau) - \beta \right\} \bigcap \mathcal{B}^1 \qquad (3)$$

- Features
  - Identifies a **model that is favorable to the agent**, even in the **worst-case scenario**, based on the current belief set B^k.
  - After each play, collects the complete trajectories (τ ^k) of all agents' actions and observations.
  - Eliminates suboptimal model candidates based on the obtained data.

5 Playing against Adversarial Opponents

- 5.2 Positive results

Algorithm 3 minimizes the difference between the maximum score achievable in the worst-case scenario and the score actually obtained.

However, a drawback is that it can only be used when states can be distinguished from a single observation (3.1).
It cannot be used when the environment becomes more complex and distinguishing states requires multiple steps (3.2).