A Deep Reinforcement Learning Approach for Finding Non-Exploitable Strategies in Two-Player Atari Games

Zihan Ding

Princeton University

September 16, 2022

A Deep Reinforcement Learning Approach for Finding Non-Exploitable Strategies in Two-Player Atari Games

Zihan Ding*
Princeton University
zihand@princeton.edu

Dijia Su*
Princeton University
dsu@princeton.edu

Qinghua Liu
Princeton University
qinghual@princeton.edu

Chi Jin
Princeton University
chij@princeton.edu

Motivations

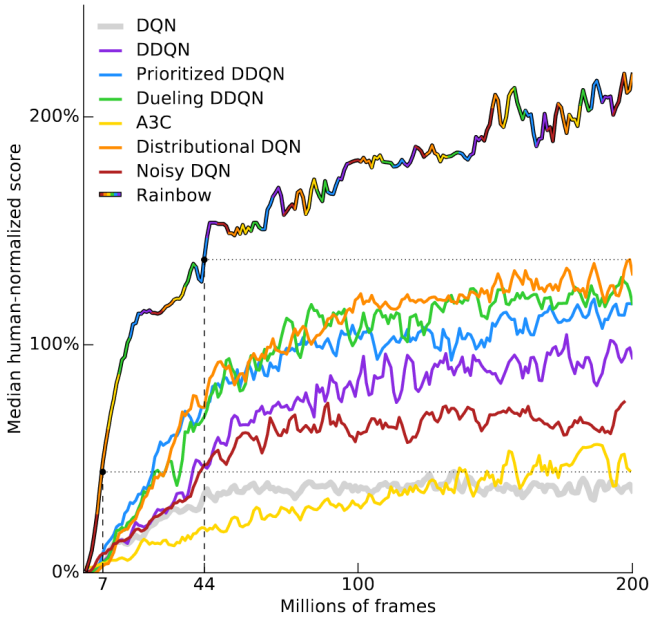
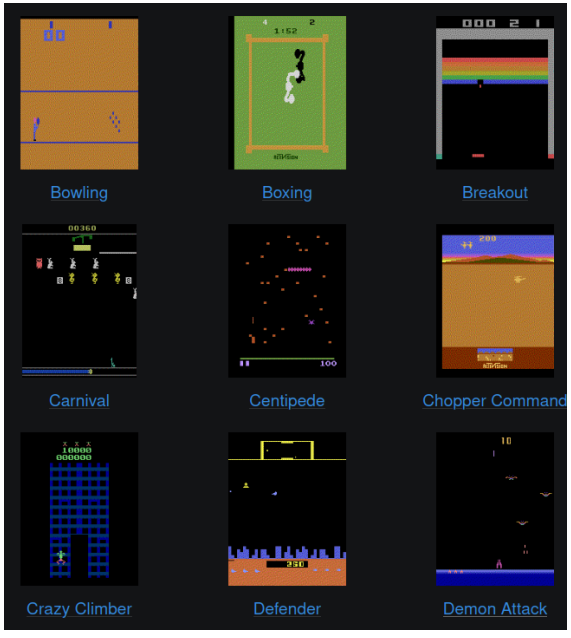


Figure from Rainbow paper [Hessel et al., 2018]



Atari games (from OpenAI Gym website)

- Single-agent Deep Reinforcement Learning (DRL) has achieved great progress, e.g. Atari games, although the theoretical optimum is unknown...
- How about Multi-agent RL setting?

Motivations

- Large-scale MOBA or RTS game, like DeepMind AlphaStar: fictitious self-play, population-based training, tons of engineering efforts...

Figure 1: AlphaStar in game

Motivations

- Extensive-form game, like Texas hold'em: state/action abstraction, Monte Carlo counterfactual regret minimization [Brown and Sandholm, 2018].

  It has incomplete information, tree structure.

Motivations

- How about multi-agent Atari games?

  Thanks to PettingZoo [Terry et al., 2021], multi-agent Atari games with 2 to 4 players.

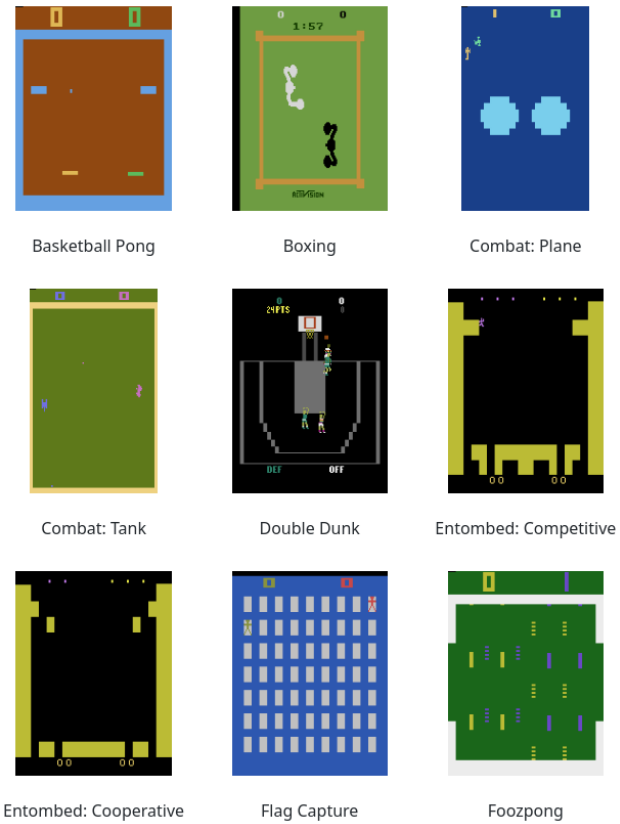


Figure 2: Multi-agent Atari games in PettingZoo

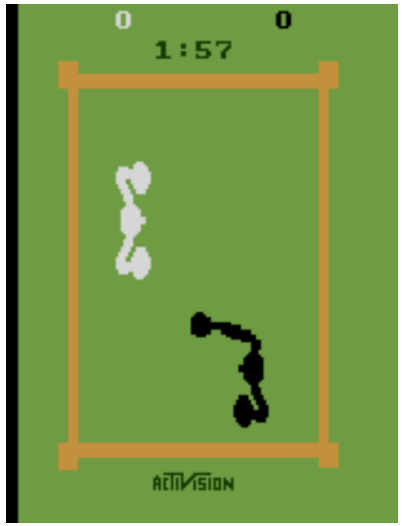
We focus on two-player zero-sum setting, assuming complete information (stacked frames if necessary).

Settings

Markov Game

Markov game (MG) is defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{B}, \mathbb{P}, r, \gamma)$.

- $\mathcal{S}$: state space
- $\mathcal{A}, \mathcal{B}$: action spaces for max- and min-player respectively
- $\mathbb{P}(\cdot|s, a, b)$: state transition distribution
- r: $\mathcal{S} \times \mathcal{A} \times \mathcal{B} \to \mathbb{R}$ reward function
- $\gamma \in [0, 1]$ discount factor



Two-player Boxing game in PettingZoo

Methods

Previous approaches

1. *Self-Play* (SP) [Fudenberg et al., 1998], iterative best response:

$$x_i^{t+1} = \mathrm{Br}_i(x_{-i}^t), \forall i \in [n] \tag{1}$$

Not converge for zero-sum games.

Methods

Previous approaches

2. *Fictitious Play* (FP) [Brown, 1951], iterative best response to opponent's historical average strategy:

$$\widehat{x_i^t} = \mathrm{Br}_i(x_{-i}^t = \frac{1}{t}\sum_{\tau=0}^{\tau=t-1} \Pr(a_{-i}^\tau = a, a \in \mathcal{A})) \tag{2}$$

$$x_i^{t+1} = (1 - \frac{1}{t})x_i^t + \frac{1}{t}\widehat{x_i^t}, \forall i \in [n] \tag{3}$$

FP achieves ϵ-NE with a convergence rate of $(1/\epsilon)^{\Omega(\mathcal{A})}$ for a normal-form game with action space $\mathcal{A}$ [Daskalakis and Pan, 2014].

Neural fictitious self-play (NFSP) [Heinrich and Silver, 2016] is a function approximation version of FP, using DRL.

Methods

Previous approaches

3. *Double Oracle* (DO) [McMahan et al., 2003], iterative best response to the opponent's Nash equilibrium strategy on a restricted action set:

$$\widehat{x_i^t} = \text{Br}_i(\text{Nash}_{-i}(\{\widehat{x_{-i}^\tau} | \tau = 0, 1, \ldots, t-1\})) \tag{4}$$

$$x_i^{t+1} = \text{Nash}(\{\widehat{x_i^\tau} | \tau = 0, 1, \ldots, t\}), \forall i \in [n] \tag{5}$$

The DO has convergence rate of $\mathcal{O}(|\mathcal{A}|)$ for a normal-form game with action space $\mathcal{A}$.

However, direct converting of Markov game to normal-form game will generate a new action space **exponential** in horizon H, number of states $|\mathcal{S}|$ and number of actions $|\mathcal{A}|$ in original Markov game.

Policy space response oracle (PSRO) [Lanctot et al., 2017] is a function approximation version of DO, using DRL.

Methods

Our method for MG: close to *Nash Q-learning* [Hu and Wellman, 2003] and *Nash Value Iteration (Nash VI)* [Liu et al., 2021], but with function approximation, named *Nash Deep Q-Network (Nash-DQN)*.

Given a payoff matrix $\mathrm{A} \in \mathbb{R}^{A \times B}$, the Nash equilibrium (NE) is,

$$(\mu^\star, \nu^\star) = \mathrm{Nash}(\mathrm{A}) = \arg\max_{\mu} \arg\min_{\nu} \mu^\intercal \mathrm{A} \nu \tag{6}$$

A is estimated with $Q(s, \cdot, \cdot)$ at each step of MG.
The Nash Bellman Equation,

$$Q^{target}(s, a, b) = r + \gamma \mathbb{E}_{s' \sim \mathbb{P}}[\max_{\mu} \min_{\nu} \mu^\intercal Q(s', \cdot, \cdot) \nu] \tag{7}$$

Nash Q-learning uses the soft update in tabular case,

$$Q(s, a, b) = \alpha \cdot Q^{target}(s, a, b) + (1 - \alpha) \cdot Q(s, a, b) \tag{8}$$

and ϵ-greedy for exploration.

Methods

In *Nash-DQN*,

- Q is parameterized with Q_ϕ
- Update with mean squared error between Q_ϕ and Q^{target}
- Maintain a sample replay buffer $\mathcal{D}$, take mini-batch SGD with a certain batch size
- Use a target network $Q_{\phi^{target}}$ for stabilize training, with delayed update
- ϵ-greedy for exploration

Methods

Algorithm 1 Nash Deep Q-Network (NASH_DQN)

Initialize replay buffer $\mathcal{D} = \emptyset$, counter $i = 0$, Q-network Q_ϕ
Initialize target network parameters: $\phi^{\text{target}} \leftarrow \phi$.
for episode $k = 1, \ldots, K$ **do**
  reset the environment and observe s_1.
  for $t = 1, \ldots, H$ **do**
    % collect data
    sample actions (a_t, b_t) from $\begin{cases} \text{Uniform}(\mathcal{A} \times \mathcal{B}) & \text{with probability } \epsilon \\ (\mu_t, \nu_t) = \text{NASH}(Q_\phi(s_t, \cdot, \cdot)) & \text{otherwise.} \end{cases}$
    execute actions (a_t, b_t), observe reward r_t, next state s_{t+1}.
    store data sample $(s_t, a_t, b_t, r_t, s_{t+1})$ into $\mathcal{D}$
    % update Q-network
    randomly sample minibatch $\mathcal{M} \subset \{1, \ldots, |\mathcal{D}|\}$.
    for all $j \in \mathcal{M}$ **do**
      compute $(\hat{\mu}, \hat{\nu}) = \text{NASH}(Q_{\phi^{\text{target}}}(s_{j+1}, \cdot, \cdot))$
      set $y_j = r_j + \gamma \hat{\mu}^\top Q_{\phi^{\text{target}}}(s_{j+1}, \cdot, \cdot)\hat{\nu}$.
    Perform m steps of GD on loss $\sum_{j \in \mathcal{M}} (y_j - Q_\phi(s_j, a_j, b_j))^2$ to update ϕ.
    % update target network
    $i = i + 1$; if $i \% N = 0$: $\phi^{\text{target}} \leftarrow \phi$.

Methods

We also proposes an exploiter version called *Nash-DQN with Exploiter*,

$$(\mu, \cdot) = \text{Nash}(Q_\phi(s, \cdot, \cdot)) \tag{9}$$
$$\nu = \arg\min_{\nu} \mu^\intercal \tilde{Q}_\psi(s, \cdot, \cdot)\nu.$$

where $\tilde{Q}_\psi$ is the parameterized best-response value function, or the Q-network for the exploiter (min-player),

$$\tilde{Q}^{target} = r + \gamma \mathbb{E}_{s' \sim \mathbb{P}}[\min_b \hat{\mu}^\intercal \tilde{Q}_{\psi^{target}}(s', \cdot, b)] \tag{10}$$

$$(\hat{\mu}, \cdot) = \text{Nash}(Q_{\phi^{target}}(s, \cdot, \cdot)) \tag{11}$$

Update also with mean squared loss between $\tilde{Q}_\psi$ and $\tilde{Q}^{target}$.
We hope the exploiter helps with the exploration, since it always tries to exploit the first player.

Methods

In implementation, a critical step is the subroutine of solving NE for each $Q(s, \cdot, \cdot)$, this happens for each step in both exploration and update.

Solving NE in matrix is Linear Programming (LP) problem, using existing solvers:

Solver	Time per Sample (s)	Solvability
Nashpy (equilibria)	0.751	all
Nashpy (equilibrium)	0.0016	not for some degenerate matrices
ECOS	0.0015	all
MWU (single)	0.008	all (but less accurate)
MWU (parallel)	fast, depends on batch size	all (but less accurate)
CVXPY	0.009	all
PuLP	0.020	all
Scipy	–	not for some
Gurobipy	0.01	not for some

Solvability is important because Q estimation can be random during the learning process.

Theoretical Analysis

Theorem

For tabular Markov games, the optimistic Nash VI and Nash VI Exploiter can find ϵ-approximate Nash equilibria in poly$(S, A, B, (1-\gamma)^{-1}, \epsilon^{-1}, \log(1/\delta))$ steps with probability at least $1-\delta$. Here S is the size of states, A, B are the size of two players' actions respectively, and γ is the discount factor.

Proofs in [Jin et al., 2021, Liu et al., 2021].

The shows clear contrast with previous approaches FP, DO with exponential dependency.

Experiments

I. Tabular Markov Game

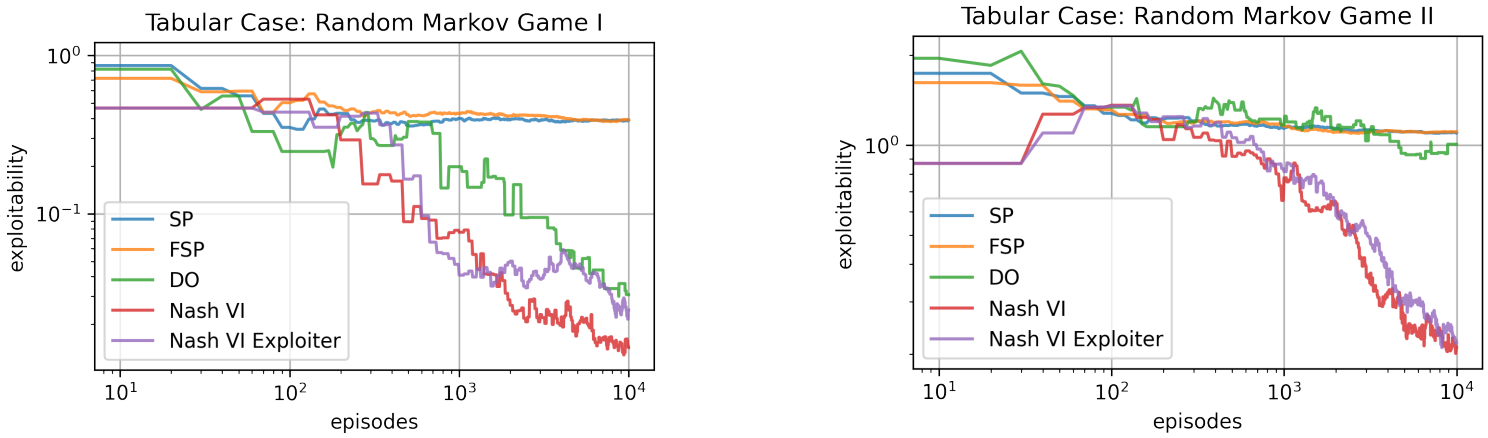


Figure 3: Tabular case experiments on two randomly generated Markov games.

- Randomly generated transition probabilities $\in [0, 1]$ and rewards $\in [-1, 1]$
- Left: $|\mathcal{S}| = |\mathcal{A}| = |\mathcal{B}| = H = 3$; Right: $|\mathcal{S}| = |\mathcal{A}| = |\mathcal{B}| = H = 6$
- All methods **without** function approximation

Experiments

I. Tabular Markov Game

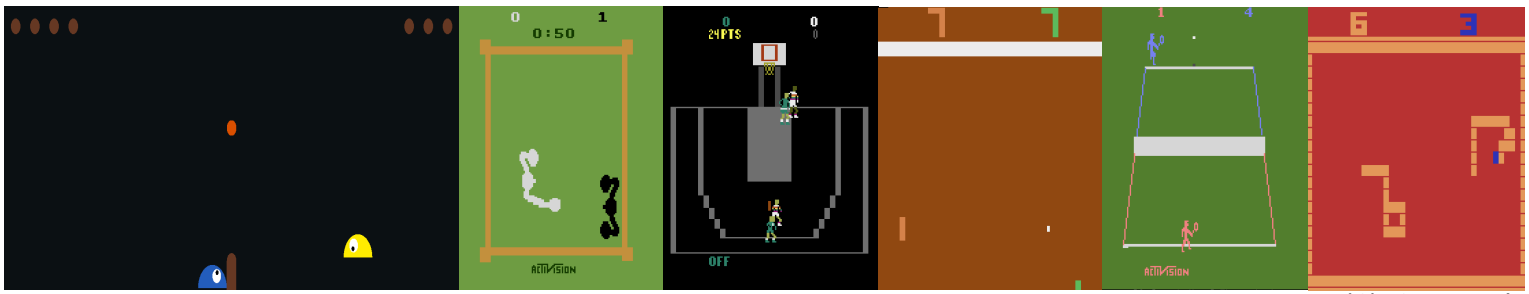
Table 1: Approximate exploitability (lower is better) in two tabular Markov games

EnvMethod	SP	FSP	NFSP	PSRO	**Nash DQN**	**Nash DQN Exploiter**	Oracle Nash
Tabular Env I	0.448	0.379	0.317	0.134	0.096	0.020	−0.027
Tabular Env II	1.239	0.694	0.379	0.569	**0.017**	0.071	−0.082

- All methods **with** function approximation, DQN as base RL agent
- Oracle Nash is the ground truth NE strategy (0 exploitability if fully exploited)
- Exploitation (best response) is achieved with DQN (under-exploiting happens)

Experiments

II. Two-Player Video Game



(1). SlimeVolley (2). Boxing (3). Double Dunk (4). Pong (5). Tennis (6). Surround

Figure 4: Screen shots of the six two-player video games.

Table 2: Approximate exploitability (lower is better) for six two-player video games.

EnvMethod	SP	FSP	NFSP	PSRO	**Nash DQN**	**Nash DQN Exploiter**
SlimeVolley	−0.049	0.514	0.069	0.000	0.000	−0.099
Boxing	24.907	93.683	24.544	66.891	−55.471	22.490
Double Dunk	7.039	6.067	4.564	7.256	−0.539	1.702
Pong	4.207	5.196	4.396	5.217	−3.336	−2.920
Tennis	2.970	2.355	3.207	2.465	−0.425	0.069
Surround	1.782	1.574	1.594	1.603	0.904	1.462

- All methods **with** function approximation, DQN as base RL agent
- Exploitation (best response) is achieved with DQN
- Use RAM observation, episode length truncated to 300

Experiments

II. Two-Player Video Game

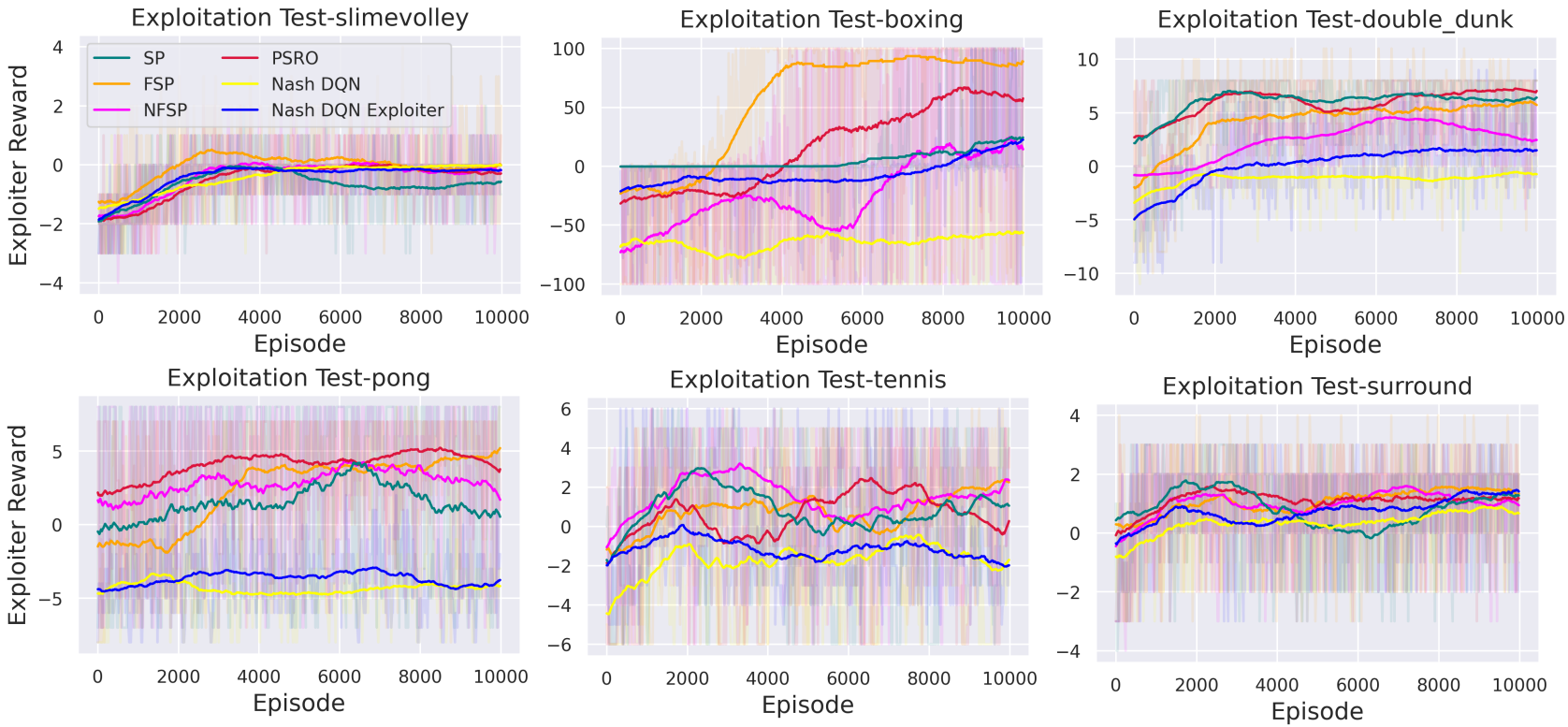
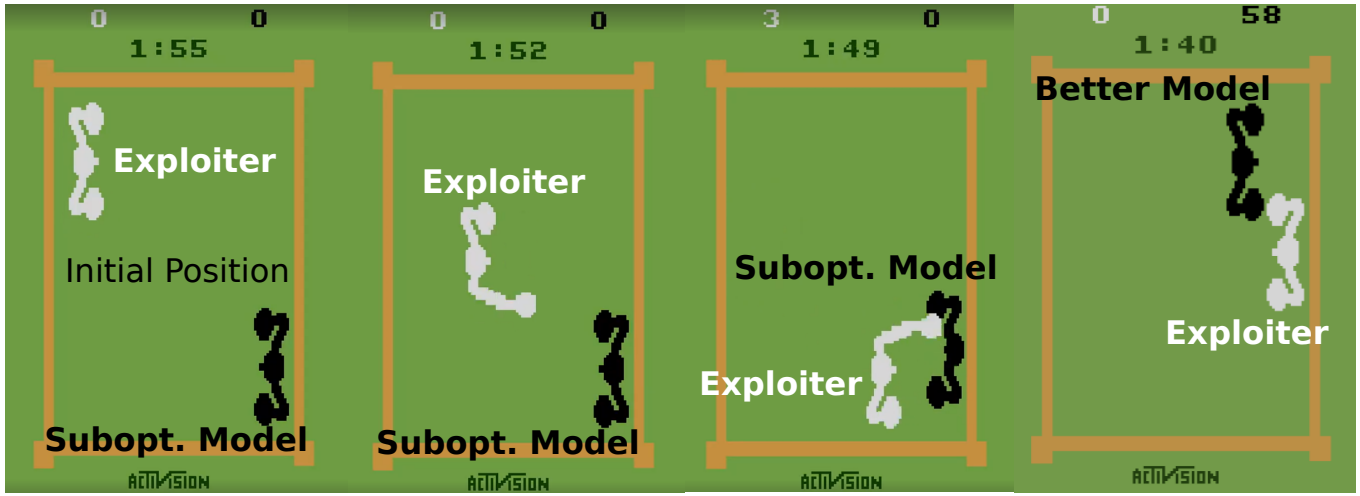


Figure 5: The exploiter learning curves for exploitation tests on six two-player zero-sum video games.

Experiments

II. Two-Player Video Game

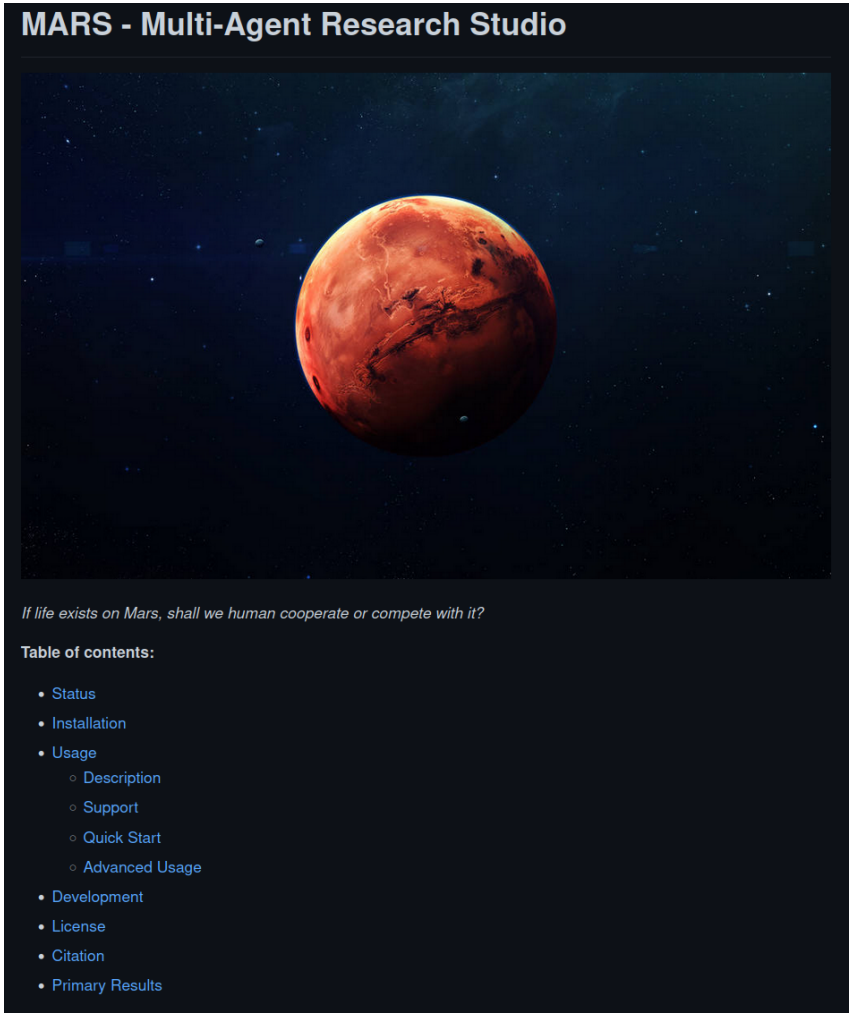


(a) (b) (c) (d)

Figure 6: The key frames in *Boxing* exploitation test: (a-c) shows a sub-optimal model exploited by the exploiter. (d) shows our proposed algorithms learn hard-to-exploit policy robust against the turning-around strategy of the exploiter.

Code Library

https://github.com/quantumiracle/MARS



MARS - Multi-Agent Research Studio

If life exists on Mars, shall we human cooperate or compete with it?

Table of contents:

Next Steps

Promising results are achieved with current methods, but there are limitations as well, in next steps:

- Direct train on image frames
- Thorough experiments on full episode length settings (takes at least 10x longer time than present)
- Optimize Nash solving subroutine
- Policy gradient style algorithm
- Independent learning process for each player

Q&A

References

George W Brown. Iterative solution of games by fictitious play. *Activity analysis of production and allocation*, 13(1):374–376, 1951.

Noam Brown and Tuomas Sandholm. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374): 418–424, 2018.

Constantinos Daskalakis and Qinxuan Pan. A counter-example to karlin's strong conjecture for fictitious play. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 11–20. IEEE, 2014.

Drew Fudenberg, Fudenberg Drew, David K Levine, and David K Levine. *The theory of learning in games*, volume 2. MIT press, 1998.

Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games, 2016.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

Junling Hu and Michael P Wellman. Nash q-learning for general-sum stochastic games. *Journal of machine learning research*, 4(Nov): 1039–1069, 2003.

Chi Jin, Qinghua Liu, and Tiancheng Yu. The power of exploiter: Provable multi-agent rl in large state spaces. *arXiv preprint arXiv:2106.03352*, 2021.

Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Perolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning, 2017.

Qinghua Liu, Tiancheng Yu, Yu Bai, and Chi Jin. A sharp analysis of model-based reinforcement learning with self-play. In *International Conference on Machine Learning*, pages 7001–7010. PMLR, 2021.

H Brendan McMahan, Geoffrey J Gordon, and Avrim Blum. Planning in the presence of cost functions controlled by an adversary. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 536–543, 2003.

J Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis S Santos, Clemens Dieffendahl, Caroline Horsch, Rodrigo Perez-Vicente, et al. Pettingzoo: Gym for multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34:15032–15043, 2021.