Neurocomputing 449 (2021) 207–213

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Self-play reinforcement learning with comprehensive critic in computer games

Shanqi Liu, Junjie Cao, Yujie Wang, Wenzhou Chen, Yong Liu *

Institute of Cyber Systems and Control, ZheJiang University, China

ARTICLE INFO

Article history:
Received 4 July 2020
Revised 3 December 2020
Accepted 2 April 2021
Available online 06 April 2021
Communicated by Zidong Wang

Keywords:
Reinforcement learning
Self-play
Computer game

ABSTRACT

Self-play reinforcement learning, where agents learn by playing with themselves, has been successfully applied in many game scenarios. However, the training procedure for self-play reinforcement learning is unstable and more sample-inefficient than (general) reinforcement learning, especially in imperfect information games. To improve the self-play training process, we incorporate a comprehensive critic into the policy gradient method to form a self-play actor-critic (SPAC) method for training agents to play computer games. We evaluate our method in four different environments in both competitive and cooperative tasks. The results show that the agent trained with our SPAC method outperforms those trained with deep deterministic policy gradient (DDPG) and proximal policy optimization (PPO) algorithms in many different evaluation approaches, which vindicate the effect of our comprehensive critic in the self-play training procedure.

1. Introduction

Reinforcement learning has achieved excellent performance in a lot of board games and poker games such as Chess [1], Go [2], Gomoku [3], Kuhn poker [4] and Texas holdem poker [5]. Building computer programs with high performance in a non-trivial game can be a stepping stone toward solving more challenging real-world problems [6]. Recently, modern interactive computer games such as first-person shooters (e.g. ViZDoom [7]) and real-time strategy games (e.g. StarCraft [8]) have been the focus of AI research progressively.

Self-play reinforcement learning, i.e. agents learn by playing against the copy of themselves, replaces the loser with a copy of the winner in its training paradigm which nicely provides a perfect curriculum and offers rival opponents to agents. For example, game agents such as checker engines [9], AlphaGo [2], AlphaGo Zero [10] and AlphaZero [11] all benefit from the self-play reinforcement learning, which demonstrates that agents trained with self-play reinforcement learning methods are capable of outstripping human players in perfect-information games.

As for imperfect-information games with hidden information and stochasticity, they have been regarded as a beneficial domain for current AI research due to their promising application prospects. In many computer games of this kind, agents are usually trained with reinforcement learning by playing with a predetermined computer player (game bot) [7,8,12,13], and tend to overfit to the game bot. Meanwhile, Heinrich et al. [4] apply self-play Monte-Carlo Tree Search in Kuhn poker. A safe and nested subgame-solving technique for imperfect-information games have been proposed in the research of Brown et al. [5], which helps machines to defeat top humans players in no-limit Texas holdem poker. [14] indicates that Fictitious Self-Play (FSP) can converge to approximate Nash equilibria in playing River poker. In Limit Texas Holdem poker, based on significant domain expertise, Neural Fictitious Self-Play (NFSP) can achieve the performance of state-of-the-art according to [15]. Though those methods achieved good performance in training agents to play games with imperfect information, they usually rely on the extensive exploration and the memory of long-past information, instead of taking full advantage of the training paradigm of self-play in which the observation of the other player can facilitate the self-play training procedure.

In this paper, we propose a new self-play reinforcement learning method: self-play actor-critic (SPAC), in which a comprehensive critic speeds up the training procedure of self-play by combing the imperfect information observed of both players.

2. Related work

Self-play reinforcement learning has been widely studied and implemented on board games and pokers. In early time, Gelly and Silver studied 9x9 Go and proposed Heuristic UCT-RAVE algo-

* Corresponding author.
E-mail address: 11932025@zju.edu.cn (S. Liu).

https://doi.org/10.1016/j.neucom.2021.04.006
0925-2312/

rithm which incorporates prior knowledge to the Monte-Carlo tree search and provides a simple way to share experience between classes of related positions [16]. In 2010, Wiering studied the problem of learning to play Backgammon through a combination of self-play and expert knowledge methods [17], and pointed out that the best strategy to train agents to play Backgammon is playing against an expert which is difficult to access. [18] combines ADP with MCTS to train a neural network to play Gomoku with self-play, and competed for the candidate level of 5-star Gomoku. In computer Go, Silver et al. have trained two novel neural network agents, called AlphaGo and AlphaGo Zero, to achieve a state of the art results in the game of Go [2,10]. AlphaGo, which uses supervised learning, deep reinforcement learning, and MCTS, defeated top human Go players, through extensive use of domain knowledge and games played by human champions. In AlphaGo Zero, self-play games were generated by the best player which win by a margin of 55% when competing with the precedent best players. Furthermore, Silver et al. extended AlphaGo Zero to a general game-playing strategy, called AlphaZero, achieving state of the art in the games of Chess and Shogi [11]. In contrast to AlphaGo Zero, AlphaZero maintains a single policy that generates self-play games and is updated continually, without the evaluation step for the selection of the best player. Another works like [19] presents the MuZero algorithm which, by combining a tree-based search with a learned model, achieves superhuman performance in a range of challenging and visually complex domains. Although those works are successful in training agents to play board games, they are highly relying on the perfect information observed by the players, the rule of board games and domain knowledge of experts.

In playing Kuhn poker which is imperfect-information, [4] combines self-play MCTS with fictitious play, where the convergence to Nash equilibria is demonstrated with empirical results. Fictitious Self-Play (FSP) combines reinforcement learning and supervised learning in self-play training of agents to play poker games [14]. Based on FSP, Neural Fictitious Self-Play [15] is the first end-to-end self-play deep reinforcement learning method that can approach a Nash equilibrium of imperfect-information games. [20] trains a population of blueprint policies to solve poker games. Those reinforcement learning methods with self-play achieve good performance in poker games, where the explicit game rule can provide some information about the system and facilitate the self-play training procedure. During their self-play training procedure, both players are updated independently without exploiting the information from their opponent. In modern interactive computer games, from which the system can not be modeled, reinforcement learning can also achieve excellent performance. In ViZDoom, a first-person shooter computer game, [7] trains an agent with augmented Deep Recurrent Q-Networks by playing with the game bot. [21] modified PPO-RND method to solve sparse reward problem in ViZDoom. Though [8,22,23,12] have good performance in playing the real time strategy game StarCraft, their training procedures rely on playing with fixed computer opponents. [24] proposes a MOBA AI learning paradigm that methodologically enables playing full MOBA games with deep reinforcement learning.

Contrary to previous work in computer games, especially with imperfect information, we aim to train agents with self-play and facilitate the training procedure with a comprehensive critic to offset the impact of imperfect information, by exploiting the observation and action of the opponent.

3. Background

In standard Markov Decision Process (MDP), one agent sequentially chooses an action a_t according to a policy $\pi(a|s)$ based on the state s_t at time t. After taking the action a_t, state s_t transforms to the next state s_{t+1} according to the transition probability which satisfies Markov property and is entirely determined by the state-action pair one-time step before, i.e. $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$. Then the agent receives a scalar reward signal $r(s_t, a_t)$ from the environment. Deep Reinforcement Learning is one kind of deep learning algorithms that finds a policy π which can extract features with deep neural network and maximizes the expected discounted cumulative reward in one episode, i.e. $J(\theta) = E\left[\sum_t \gamma^t r(s_t, a_t)\right]$, where γ is a discount factor.

A multi-agent extension of MDP is called partially observable Markov games [25]. A Markov game, is defined by a set of states S describing the possible configurations of all agents, a collection of action sets $\{A_1, \ldots, A_k\}$ and a collection of observation sets $\{O_1, \ldots, O_K\}$ for *k*agents in the environment. State transitions depend on the current state and one action from each agent: $T : S \times A_1 \times \cdots \times A_k \to \mathbf{PD}(S)$, with $\mathbf{PD}(\mathrm{S})$ represents the set of probability distributions over the state space. Similar to standard MDP, each agent *i*will have an associated reward function, $R_i(S, A_i)$.

Policy gradient methods maximize the expected cumulative reward by estimating the performance gradient concerning the policy parameter vector $\theta : \nabla_\theta J(\pi_\theta)$, and updating the policy parameter vector with gradient ascent. In stochastic policy gradient methods [26], stochastic policy samples from a Gaussian distribution $\pi_\theta \sim N(\mu(s), \sigma(s)^2 I)$ with $\mu(s)$ and $\sigma(s)$ parameterized by θ. In deterministic policy gradient methods such as deterministic policy gradient (DPG) [27] and deep deterministic policy gradient (DDPG) [28]. DDPG is an actor-critic, model-free algorithm based on the deterministic policy gradient that can operate over continuous action spaces. The actor is the policy model $\pi_\theta(a|s)$ that selects the action a based on the observation s at each time step. The critic estimates the action-value function $Q(s, a)$ using off-policy data which is sampled by a noisy policy. The noisy policy improves the exploration by adding additive action noise to deterministic policy: $\hat{\pi}_\theta(s) = \pi_\theta(s) + w$, where $w \sim N(0, \sigma^2 I)$ represents uncorrelated Gaussian noise or $w \sim OU(0, \sigma^2 I)$ represents Ornstein–Uhlenbeck process noise. With the noise variance σ annealing during the training process, deterministic policy gradient methods make a trade-off between exploration and exploitation.

As the original version of the policy gradient algorithm, REINFORCE [29] tends to be of high variance due to the gradient estimation with the Monte Carlo method:

$$\nabla_\theta J(\theta) \approx \sum_{t=0}^{N-1} \nabla_\theta \log \pi_\theta(a_t|s_t) R_t, \tag{1}$$

where R_t represents the cumulative reward from time t to the end of one episode. Actor-critic methods use the value function $V(s_t)$, action-value function $Q(s_t, a_t)$ or advantage function $A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$ to substitute for the cumulative reward R_t, to reduce the variance of gradient estimation and improve the performance of policy gradient methods. For deterministic policy, according to the deterministic policy gradient theorem [27], the gradient of the objective $J(\theta)$ can be estimated as:

$$\nabla_\theta J(\theta) \approx \sum_{t=0}^{N-1} \nabla_\theta \mu_\theta(a_t|s_t) \nabla_a Q^\mu(s_t, a)|_{a=\mu_\theta(s_t)}. \tag{2}$$

Proximal policy optimization algorithms (PPO) [30] can be used to optimize the policy π with the reward R_t. The gradient of the policy π calculated with PPO can be derived as:

$$\nabla_\theta J(\theta) \approx E_{\tau_i}\left[A^{\pi_{\theta_{old}}}(o, a) \frac{\nabla_\theta \pi_\theta(a|o)}{\pi_{\theta_{old}}(a|o)} - \beta \nabla_\theta KL[\pi_{\theta_{old}}(\cdot|o), \pi_\theta(\cdot|o)]\right], \tag{3}$$

where $A^{\pi_{\theta_{old}}}$ represents the advantage function of the policy before updating, highly dependent on the reward R_t and *KL* stands for Kullback–Leibler divergence.

4. Method

In this work, we consider a special case of Markov game for self-play reinforcement learning, in which all agents interact with each other in a common environment. The definition of self-play MDP is the same as that of partially observable Markov games.

Actor-Critic methods for standard MDP can be directly applied to self-play settings by having the agent and the copy of itself learn two independent value functions based on their observations, actions and rewards. However, because both agents have their partial observation and are independently updating their policies according to two independent value functions, the environment appears non-stationary from the view of anyone agent, violating Markov assumptions of standard MDP. Inspired by the idea that if the actions taken by other agents are known, the environment is stationary [31], we propose a self-play reinforcement learning method with comprehensive critic taking into account the opponent's observations or observations and actions as well.

4.1. Comprehensive critic with opponent's observations

In competitive games which involve only physical interactions between agents, the learned policies can only use local information (i.e. competitors' observations) at execution time. To facilitate the training process and improve the performance of self-play, we modified original Actor-Critic methods to fit the frameworks of self-play, called Self-Play Actor-Critic (SPAC). In SPAC, there is one comprehensive critic for each agent, which is related to the opponent's observations. Although we exploit the comprehensive observations to train the comprehensive critic for each agent, the behavior of the agent only depends on its observation.

More concretely, we consider a game with two players with policies $\pi = \{\pi_1, \pi_2\}$ parameterized by $\theta = \{\theta_1, \theta_2\}$. The critic function can be represented as: $Q_i(S, a)$ or $A_i(S, a)$, where $i = \{1, 2\}$ represents for each player and a is its action. The comprehensive critic functions the comprehensive state S and its action as input, and outputs the Q-value or advantage for each agent i. The comprehensive state Scould consist of the observations of both agents, i.e. $S = \{O_1, O_2\}$, and additional state information if available. It is reasonable, in the training procedure of one agent with self-play, that we can obtain a comprehensive state which includes both agents' observations and additional information as well as the action of the other agent. The reward functions of both agents can be conflicting in the competitive setting.

For deterministic policy μ^i of agent $i$in self-play, the policy gradient for our SPAC can be derived according to the deterministic policy gradient theorem [27], similar to Eq. (2):

$$\nabla_\theta J(\theta) \approx \frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{N-1}\nabla_\theta \mu_\theta^i(a_t^i|o_t^i)\nabla_{a_i} Q_i(S_t, a_t^i)|_{a_i=\mu_i(o_i)}. \tag{4}$$

The comprehensive action-value function Q_i can be updated according to the minimization of the mean square TD error. The replay experience of self-play $(S_t, o_t^1, o_t^2, a_t^i, r_t^i, S_{t+1})$ can be stored in the replay buffer for off-policy updating of Q_i. With random sampling from the replay buffer and accessing the policies of both agents, the comprehensive action-value Q_i can be updated by minimizing:

$$\frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{N-1}[r_t^i + \gamma Q_i'(S_{t+1}, \mu_i'(o_t^i))) - Q_i(S_t, a_t^i)]^2, \tag{5}$$

where Q_i' is the delayed copy of Q_i and μ_i' is the set of target policies which are delayed copies of μ_θ^i and are easy to get in self-play. Using deterministic policy, our SPAC can take advantage of off-policy updating of DDPG [28], and can be seen as an extension of MADDPG [31] which optimizes separate policies in a multi-agent system. However, different from that of MADDPG, our SPAC with the deterministic policy is actually optimizing one single policy in a self-play setting.

4.2. Comprehensive critic with opponent's observations and actions

The main difference between these two ways of using global information is that if agents can access the opponents actions, the agent may learn a policy that can predict its opponents action, which will be useful in a competition setting.

The other influence of combining global information without action is that the environment is non-stationary now, the agent has to train in a non-stationary environment. This can cause unstable and agents will likely struggle to learn the above ground level policies because RL methods need the distribution of states to which those actions lead are stable to effectively value actions.

Similarly, we consider a game with two players with policies $\pi = \{\pi_1, \pi_2\}$ parameterized by $\theta = \{\theta_1, \theta_2\}$. The critic function can be represented as: $Q_i(S, a_1, a_2)$ or $A_i(S, a_1, a_2)$, where $i = \{1, 2\}$ represents for each player. The comprehensive critic function takes the actions of both agents $\{a_1, a_2\}$ and the comprehensive state Sas input, and outputs the Q-value or advantage for each agent i. The comprehensive state Scould consist of the observations of both agents, i.e. $S = \{O_1, O_2\}$.

For deterministic policy μ^i of agent $i$in self-play, the policy gradient similar to Eq. (4) but with opponent's actions:

$$\nabla_\theta J(\theta) \approx \frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{N-1}\nabla_\theta \mu_\theta^i(a_t^i|o_t^i)\nabla_{a_i} Q_i(S_t, a_t^1, a_t^2)|_{a_i=\mu_i(o_i)}. \tag{6}$$

The experience of self-play $(S_t, o_t^1, o_t^2, a_t^1, a_t^2, r_t^1, r_t^2, S_{t+1})$ can be stored in the replay buffer for off-policy updating of Q_i. The comprehensive action-value Q_i can be updated by minimizing:

$$\frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{N-1}[r_t^i + \gamma Q_i'(S_{t+1}, \mu_1'(o_t^1), \mu_2'(o_t^2)) - Q_i(S_t, a_t^1, a_t^2)]^2, \tag{7}$$

As for stochastic policy μ^i of agent $i$in self-play, the policy gradient similar to Eq. (3) but with opponent's actions:

$$\nabla_\theta J(\theta) \approx \frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{N-1}\nabla_\theta\left[A^{\pi_{\theta_{old}}^i}(S_t, a_t^1, a_t^2)\frac{\pi_\theta^i(a_t^i|o_t^i)}{\pi_{\theta_{old}}^i(a_t^i|o_t^i)} - \beta KL[\pi_{\theta_{old}}^i(\cdot|o_t^i), \pi_\theta^i(\cdot|o_t^i)]\right], \tag{8}$$

where $\pi_{\theta_{old}}^i$ represents the policy before optimization and $KL[\pi_{\theta_{old}}^i(\cdot|o_t^i), \pi_\theta^i(\cdot|o_t^i)]$ represents the KL divergence between stochastic policies before and after updating.

The overall framework of our SPAC is illustrated in Fig. 1.

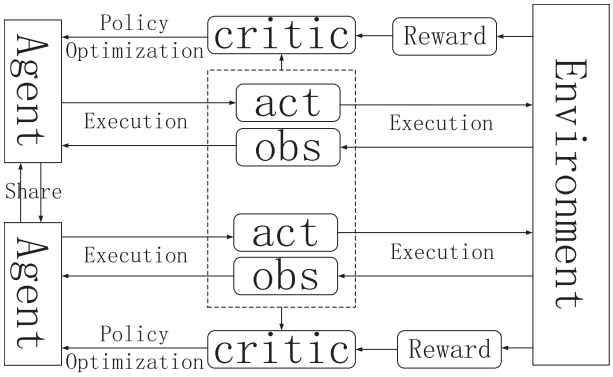


Fig. 1. Framework of SPAC.

5. Experiments

5.1. Simulated tasks

To evaluate the effect of the comprehensive critic in our SPAC, we select four suitable self-play reinforcement learning environments.

First of all, we build a simple competitive environment "Adversarial-Push" modified from the environments proposed in [32]. In "Adversarial-Push", there are two adversarial agents and one landmark inhabiting a two-dimensional world with continuous space and discrete-time interval. Both agents are punished for their distance to the landmark, and rewarded for preventing their opponents from approaching the landmark. So the reward functions of both agents can be formulated as: $R_1 = d_2 - 2 * d_1, R_2 = d_1 - 2 * d_2$, with d_i representing the distance of agent $i$to the landmark. Each agent should accomplish this by "physically pushing" another agent away from the landmark, temporarily occupying it. Here "physically pushing" means a collision with force. The force will motivate the agent with an acceleration $a = \frac{F}{m}$, with F, m represent the force executed on the agent and the mass of the agent. The observation of each agent comprises its velocity and the relative coordinate of the landmark, and the action is the force acting on their own. The environment "Adversarial-Push" is illustrated in Fig. 2, where blue and red circles represent the adversarial agents and the small green circle point represents the landmark (the goal of both agents).

Then, we choose a more complex game "Pong" in roboschool [33], which take into account the dynamics of the balls and is much more complex than the original pong game in Atari [34]. The observation of each agent comprises the positions and velocities of the paddles and balls, and the agents can control the velocity of the paddles as their actions.

We also use "Tennis" and "Soccer" in Unity mlagents [35], "Tennis" is two-player game where agents control rackets to hit a ball over the net. The agents must hit the ball so that the opponent cannot hit a valid return. And "Soccer" is an environment where four agents compete in a 2 vs 2 toy soccer game, which means agents must get the ball into the opponent's goal while preventing the ball from entering their own goal. All environments are illustrated in Fig. 2.

5.2. Effect of the comprehensive critic for self-play reinforcement learning

In this part, we evaluate the effect our SPAC in four self-play reinforcement learning environments. To prove the effectiveness of our SPAC, we firstly test training with the opponents observations and actions, we will discuss training without the opponents actions in Section 5.4.

Firstly, we implement DDPG [28] in the framework of self-play without a comprehensive critic, where both agents are trained separately without accessing the opponents' information and the loser will be replaced with a copy of the winner as that in our SPAC. In the implementation of DDPG and our SPAC with deterministic policy, the policies are all parameterized by a two-layer full connected network with 64 units per layer and "relu" active function, and are initialized randomly. In the experiments of game "Pong" we implement SPAC with stochastic policy and compare it with PPO as the baseline. Both policies are all parameterized by a two-layer full connected network with 64 units and 32 units, which are all initialized randomly. We use the γ equals to 0.99, the batch size is 32, the β is 0.95 and the learning rate is 10^{-4}. The choice between stochastic policy and deterministic policy can be decided by validation or estimation of the environments' randomness. According to our experience, in environments with deterministic state transition, the deterministic policy may be better. And stochastic policy may do better in environments with more randomness. In our experiments, we implement SPAC with the deterministic policy in "Adversarial-Push", "Tennis" and "Soccer" and stochastic policy in "Pong" as both games have different randomness.

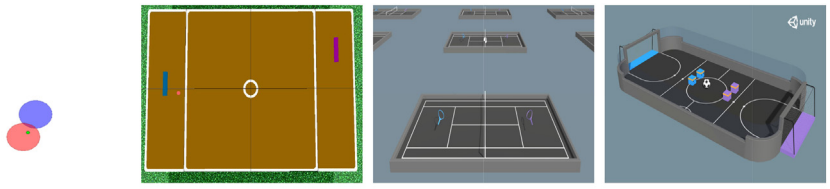
During the self-play training process, if one agent beats another by a certain score, its opponent will be replaced with the winner. Otherwise, both agents will be updated separately according to Eq. (3) or Eq. (4), provided with the observations or observations and actions of their opponents.

For fairly comparison, we evaluate the result policies trained with DDPG or PPO and our SPAC by competing in the environments presented above. This PK results will show exactly which player is doing better, for each algorithm maybe plays well during the self-play training process, but when it plays with another agent, it may not be so. The evaluation of both algorithms is based on the average episode reward in 50 competitions received by both agents. And for further comparison, we evaluate agents compete against the random and well-trained policy in "Tennis" and "Soccer" environments, as complementary experiments. These will mainly show that the baselines have learned decent policies rather than remain as ground level. A random policy is a policy which chooses a random action at each time step. And the well-trained policy is the policy using the final policy model, e.g. the final policy model trained by DDPG.

The "PK" evaluation results of all four environments are shown in Fig. 3. Because our experiments are based on a play against other agents, each image will contain only one set of experiments results.

Fig. 3 depicts the performance of the agents trained with DDPG or PPO and SPAC, plotted over the episodes required for self-play training. The "PK" result shows that our SPAC has a better performance comparing with DDPG or PPO that has no comprehensive critic, though with accidents when both training processes are not converged.

Fig. 4 depicts the results of complementary experiments, plotted over the episodes required for self-play training. The result shows that both the comparing algorithm such as DDPG and our SPAC have accepted a decent policy rather than remain as ground



(a) Adversarial-Push (b) Pong (c) Tennis (d) Soccer

Fig. 2. Environments.

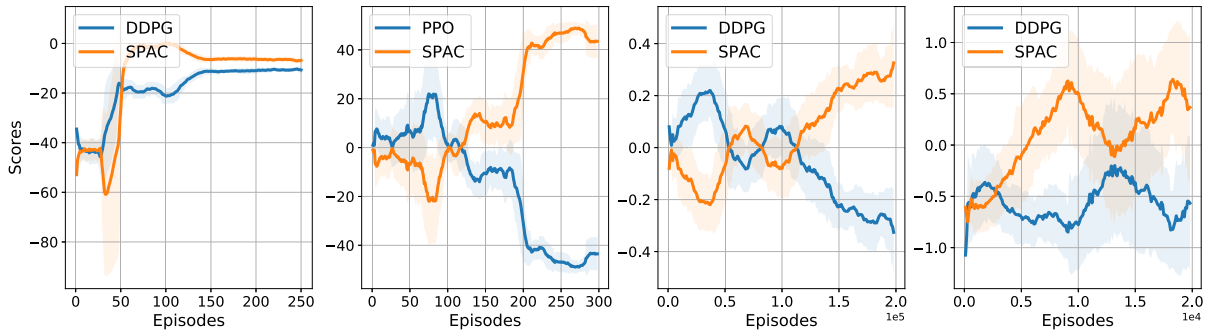


Fig. 3. Competitions between Trained Agents in four environments. From left to right are "PK" experiments tested in "push", "pong", "tennis" and "soccer". The "PK" results in the adversarial situation are measured by the return of the environments.

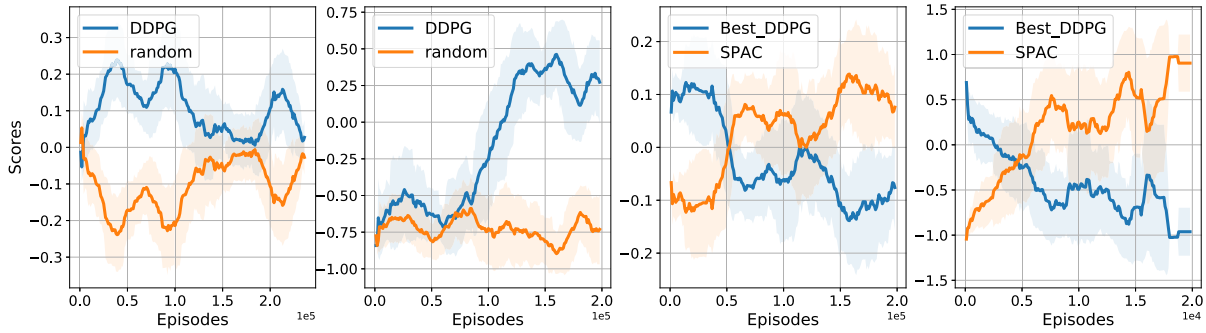


Fig. 4. Complementary experiments in tennis and soccer environments. From left to right are DDPG compete against the random policy in tennis and soccer, SPAC against the well-trained DDPG policy in tennis and soccer. The random policy is acting with total random actions and the best DDPG is a well-trained policy using DDPG.

level, this is essential to prove that all policy is well-trained, and our SPAC can actually learn a better policy rather than just beat the ground level agents. And from these results, we can find SPAC learn faster than the comparing algorithm, as SPAC can beat the best DDPG in an early stage.

These results show that our SPAC has a better performance compared with DDPG or PPO that has no comprehensive critic. This can be shown easily in the previous four sets of experiments, whether in a simple game like "Adversarial-Push" or a more complex game like "Pong", "Tennis" and "Soccer" which using stochastic policy or using deterministic policy separately. All experiment results show that SPAC performs better than their counterparts without comprehensive critic. In our SPAC, the policy is updated with Eq. (4) or Eq. (8) which is similar to that of DDPG Eq. (2) or PPO Eq. (3) with only the critic Qbe different. So it is the comprehensive critic that improves the performance of self-play reinforcement learning.

Table 1
Rewarding scheme for competitive situation.

	Left player misses the ball	Right player misses the ball
Left player rewards	−1	+1
Right player rewards	+1	−1

Table 2
Rewarding scheme for a collaborative situation

	Left player misses the ball	Right player misses the ball
Left player rewards	−1	−1
Right player rewards	−1	−1

5.3. Adversarial and cooperation settings

In the game "Pong" of roboschool, there are two agents play pong game with continuous observation space and continuous action space. We test our SPAC in the adversarial and cooperative situation of the pong game. In the adversarial game, each agent obtains an immediate reward when the ball gets past the other agent and an immediate punishment when it misses the ball. In the cooperative game, to encourage both players to keep the ball bouncing between each other, each agent obtains an immediate punishment whenever either agent misses the ball. The rewards in competitive and cooperative games are set as Tables 1 and 2. To accelerate the training process, we add rewards as well. Both agents get rewards when they catch the ball, which helps both agents learn the basic skill at the beginning of training.

We evaluate both algorithms in the paddle-bounces, the times of the ball bounces against the paddles in self-play, which can represent how well the agent plays pong game in self-play training directly. In the training session, the episode length is set as 200 frames, and the paddle-bounces are evaluated in 5 independent experiments with different random seeds, with the mean and confidence interval depicted in Fig. 5.

These results show that our SPAC can over-perform PPO in both adversarial and cooperative situations. We believe that SPAC perform better in cooperative situations because cooperative situations rely on using the opponent's observation more than adversarial situations.

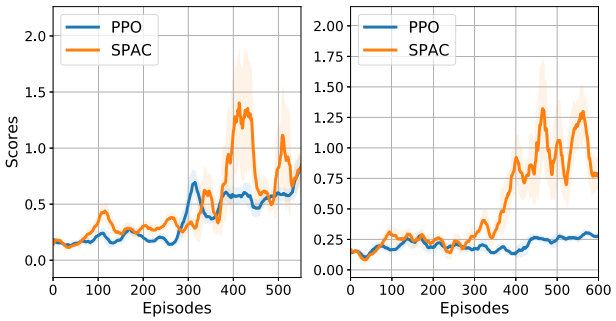


Fig. 5. The performance of the agents trained with PPO and SPAC, measured with paddle-bounces in self-play training. From left to right are tested in adversarial games and cooperative games.

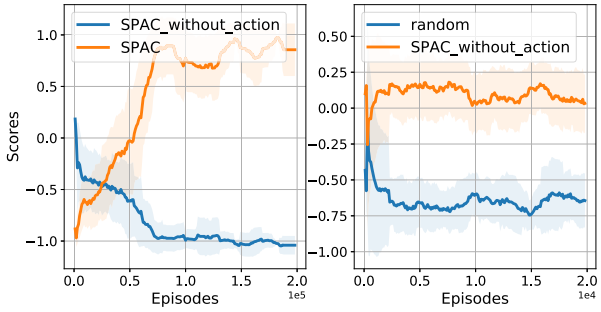


Fig. 6. The performance of the agents trained without the opponent's actions. Measured in the same ways as beyond. From left to right are SPAC trained without the opponent's actions competed with full SPAC and competed with random policy.

5.4. Training without the opponent's actions

In this setting, we use "soccer" to evaluate the policy training without the opponent's actions, for it is the most complex environment and that makes the results more convinced. We test our SPAC in both pieces of training with the opponents actions and without the opponents actions. Then we evaluate it by "pk" against SPAC trained with actions and random policy. The results are illustrated in Fig. 6.

These results show that our SPAC has a better performance compared with SPAC without the opponent's actions. This is clearly shown in "pk" result which means training with more global information can benefit training progress. This can also prove that it is a comprehensive critic that improves the performance of self-play reinforcement learning. In a word, using more global information, a better policy we can have.

6. Conclusion

In this paper, we introduce a new self-play reinforcement learning method, self-play actor-critic (SPAC), incorporating a comprehensive critic into the policy gradient method for self-play agents in playing computer games with imperfect information. The comprehensive critic evaluates the value function of one agent based on the comprehensive information of both agents in self-play, which offset the impact of imperfect information, thus can facilitate the training procedure of self-play. We evaluate the algorithm with extensive experiments based on four different environments, including both competitive and cooperative tasks. The result shows that the SPAC improves the training procedure of self-play and it outperforms DDPG and PPO in both adversarial and cooperative games. Furthermore, we test our methods in settings with and without the opponents actions. As a result, we figure out that more comprehensive information about both agents is useful and beneficial to the training process.

For further research, the application prospect of SPAC is promising, and it is easy to extend our SPAC to other self-play mechanisms such as fictitious self-play or games with more than two players.

CRediT authorship contribution statement

Shanqi Liu: Validation, Writing - original draft. **Junjie Cao:** Validation, Writing - original draft. **Yujie Wang:** Validation,Writing - review & editing. **Wenzhou Chen:** Validation, Writing - review & editing. **Yong Liu:** Supervision, Validation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

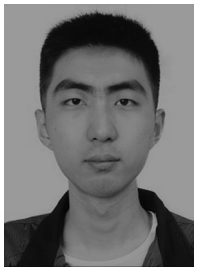
Acknowledgment

We would like to thank all people who offer help during this work.

References

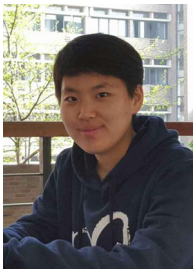
[1] E.A. Heinz, New self-play results in computer chess, in: International Conference on Computers and Games, Springer, 2000, pp. 262–276..
[2] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al., Mastering the game of go with deep neural networks and tree search, nature 529 (7587) (2016) 484..
[3] D. Zhao, Z. Zhang, Y. Dai, Self-teaching adaptive dynamic programming for gomoku, Neurocomputing 78 (1) (2012) 23–29.
[4] J. Heinrich, D. Silver, Self-play monte-carlo tree search in computer poker, in: Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014, pp. 19–25..
[5] N. Brown, T. Sandholm, Safe and nested subgame solving for imperfect-information games, in: Advances in Neural Information Processing Systems, 2017, pp. 689–699..
[6] J. Schaeffer, A gamut of games, AI Magazine 22 (3) (2001) 29.
[7] G. Lample, D.S. Chaplot, Playing fps games with deep reinforcement learning, AAAI (2017) 2140–2146.
[8] K. Shao, Y. Zhu, D. Zhao, Cooperative reinforcement learning for multiple units combat in starcraft, in: Computational Intelligence (SSCI), 2017 IEEE Symposium Series on, IEEE, 2017, pp. 1–6..
[9] A.L. Samuel, Some studies in machine learning using the game of checkers, IBM Journal of Research and Development 44 (1.2) (2000) 206–226.
[10] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al., Mastering the game of go without human knowledge, Nature 550 (7676) (2017) 354.
[11] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al., Mastering chess and shogi by self-play with a general reinforcement learning algorithm, arXiv preprint arXiv:1712.01815..
[12] Z. Tang, D. Zhao, Y. Zhu, P. Guo, Reinforcement learning for build-order production in starcraft ii, in: 2018 Eighth International Conference on Information Science and Technology (ICIST), IEEE, 2018, pp. 153–158.
[13] T. Rashid, G. Farquhar, B. Peng, S. Whiteson, Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning, Advances in Neural Information Processing Systems 33..
[14] J. Heinrich, M. Lanctot, D. Silver, Fictitious self-play in extensive-form games, in: International Conference on Machine Learning, 2015, pp. 805–813..
[15] J. Heinrich, D. Silver, Deep reinforcement learning from self-play in imperfect-information games, arXiv preprint arXiv:1603.01121..
[16] S. Gelly, D. Silver, Achieving master level play in 9 x 9 computer go., in: AAAI, vol. 8, 2008, pp. 1537–1540..
[17] M.A. Wiering, Self-play and using an expert to learn to play backgammon with temporal difference learning, JILSA 2 (2) (2010) 57–68.
[18] Z. Tang, D. Zhao, K. Shao, L. Lv, Adp with mcts algorithm for gomoku, in: Computational Intelligence (SSCI), 2016 IEEE Symposium Series on, IEEE, 2016, pp. 1–7..
[19] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al., Mastering atari, go, chess and shogi by planning with a learned model, arXiv preprint arXiv:1911.08265..

[20] N. Brown, T. Sandholm, Superhuman ai for multiplayer poker, Science 365 (6456) (2019) 885–890.
[21] J.-C. Chen, T.-H. Chang, Modified ppo-rnd method for solving sparse reward problem in vizdoom, in: 2019 IEEE Conference on Games (CoG), IEEE, 2019, pp. 1–4..
[22] S. Xu, H. Kuang, Z. Zhi, R. Hu, Y. Liu, H. Sun, Macro action selection with deep reinforcement learning in starcraft, in: Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, vol. 15, 2019, pp. 94–99..
[23] X. Xu, T. Huang, P. Wei, A. Narayan, T.-Y. Leong, Hierarchial reinforcement learning in starcraft ii with human expertise in subgoals selection, arXiv preprint arXiv:2008.03444..
[24] D. Ye, G. Chen, W. Zhang, B. Yuan, B. Liu, J. Chen, Z. Liu, F. Qiu, H. Yu, Y. Yin, et al., Towards playing full moba games with deep reinforcement learning, Advances in Neural Information Processing Systems 33..
[25] M. L. Littman, Markov games as a framework for multi-agent reinforcement learning, in: Machine Learning Proceedings 1994, Elsevier, 1994, pp. 157–163..
[26] R.S. Sutton, D.A. McAllester, S.P. Singh, Y. Mansour, Policy gradient methods for reinforcement learning with function approximation, in: Advances in Neural Information Processing Systems, Advances in Neural Information Processing Systems, 2000, pp. 1057–1063..
[27] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic policy gradient algorithms, in: ICML, ICML, 2014..
[28] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, arXiv preprint arXiv:1509.02971..
[29] R.J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Machine Learning 8 (3–4) (1992) 229–256.
[30] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347..
[31] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, I. Mordatch, Multi-agent actor-critic for mixed cooperative-competitive environments, in: Advances in Neural Information Processing Systems, 2017, pp. 6379–6390..
[32] I. Mordatch, P. Abbeel, Emergence of grounded compositional language in multi-agent populations, arXiv preprint arXiv:1703.04908..
[33] O. Klimov, J. Schulman, Roboschool (2017)..
[34] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, Openai gym, arXiv preprint arXiv:1606.01540..
[35] A. Juliani, V.-P. Berges, E. Vckay, Y. Gao, H. Henry, M. Mattar, D. Lange, Unity: A general platform for intelligent agents, arXiv preprint arXiv:1809.02627..



Shanqi Liu received his B.S. degree in control science and engineering from Zhejiang University in 2019. He is currently a Ph.D. Candidate of the institute of Cyber Systems and Control, Department of Control Science and Engineering, Zhejiang University. His research area is reinforcement learning and robotics.

Junjie Cao received the B.S. degree in Mechanical Engineering and Automation from Nanjing Tech University, Nanjing, China, in 2014 and the M.S. degree in Mechanical Engineering (Mechatronics) from Zhejiang University, Zhejiang, China, in 2017. He is currently working toward the Ph.D. degree at the College of Control Science and Engineering, Zhejiang University. His current research interests include machine learning, sequential decision making and robotics.



Yujie Wang, Ph.D. candidate of the Interdisciplinary Center of the Academy of Humanities and Social Sciences, Zhejiang University. Her main research areas are MIDAS data mining, thick-tail heterogeneous economic research. She also committed to applying advanced deep learning techniques in financial prediction, reinforcement learning methods in Forex scenario, and the expansion of the above theories in the fields of dynamic trade games and industrial organization.

Wenzhou Chen received the B.S. degree in automation from Zhejiang University of Technology in 2017. He is currently a Ph.D. Candidate of the institute of Cyber Systems and Control, Department of Control Science and Engineering, Zhejiang Uni- versity. His latest research interests include robot navigation and deep reinforcement learning.

Yong Liu received his B.S. degree in computer science and engineering from Zhejiang University in 2001, and the Ph.D. degree in computer science from Zhejiang University in 2007. He is currently a professor in the institute of Cyber Systems and Control, Department of Control Science and En- gineering, Zhejiang University. He has published more than 30 research papers in machine learning, computer vision, information fusion, robotics. His latest research interests include machine learning, robotics vision, information processing and granular computing. He is the corresponding author of this paper.