- Mdnotes File Name: undefined

# Extracted Annotations (2021-12-25)

> "Self-play, where the agents compete with themselves, is often used to generate training data for iterative policy improvement." (Zhong et al 2020:1)

*Self-play definition(note on p.1)*

> "Typical rules include choosing the latest agent, the best agent, or a random historical agent. However, these rules may be inefficient in practice and sometimes do not guarantee convergence even in the simplest matrix games" (Zhong et al 2020:1)

> "This paper proposes a new algorithmic framework for competitive self-play reinforcement learning in two-player zero-sum games." (Zhong et al 2020:1)

*Contribution(note on p.1)*

> "Our method simultaneously trains several agents and intelligently takes each other as opponents based on a simple adversarial rule derived from a principled perturbation-based saddle optimization method." (Zhong et al 2020:1)

> "competing against a randomly chosen historical agent leads to the emergence of more diverse behaviors (Bansal et al., 2017) and more stable training than against the latest agent (Al-Shedivat et al., 2018)." (Zhong et al 2020:1)

> "In population-based training (Jaderberg et al., 2019; Liu et al., 2019) and AlphaStar (Vinyals et al., 2019), an elite or random agent is picked from the agent population as the opponent" (Zhong et al 2020:1)

> "Unfortunately, these rules may be inefficient and sometimes ineffective in practice since they do not necessarily enjoy last-iterate convergence to the "average-case optimal" solution even in tabular matrix games." (Zhong et al 2020:1)

*Drawback (note on p.1)*

> "In fact, in the simple Matching Pennies game, self-play with the latest agent fails to converge and falls into an oscillating behavior, as shown in Sec. 5 ." (Zhong et al 2020:1)

> "Reinforcement learning (RL) from self-play has drawn tremendous attention over the past few years. Empirical successes have been observed in several challenging tasks, including Go (Silver et al., 2016; 2017; 2018), simulated hide-and-seek (Baker et al., 2020), simulated sumo wrestling (Bansal et al., 2017), Capture the Flag (Jaderberg et al., 2019), Dota 2 (Berner et al., 2019), StarCraft II (Vinyals et al., 2019), and poker (Brown & Sandholm, 2019), to name a few." (Zhong et al 2020:1)

*Previous work (note on p.1)*

> "During RL from selfplay, the learner collects training data by competing with an opponent selected from its past self or an agent population." (Zhong et al 2020:1)

*Methodology of self-pay (note on p.1)*

> "Self-play presumably creates an auto-curriculum for the agents to learn at their own pace. At each iteration, the learner always faces an opponent that is comparably in strength to itself, allowing continuous improvement." (Zhong et al 2020:1)

*Self-play auto-curriculum(note on p.1)*

> "y the opponents are selected often follows human-designed heuristic rules in prior work." (Zhong et al 2020:1)

> "AlphaGo (Silver et al., 2016) always competes with the latest agent, while the later generation AlphaGo Zero (Silver et al., 2017) and AlphaZero (Silver et al., 2018) generate self-play data with the maintained best historical agent." (Zhong et al 2020:1)

> "Nash equilibrium is a fundamental solution concept that characterizes the desired "average-case optimal" strategies (policies). When each player assumes other players also play their equilibrium strategies, no one in the game can gain more by unilaterally deviating to another strategy. Nash, in his seminal work (Nash, 1951 ), has" (Zhong et al 2020:2)

*Nash equilibrium (note on p.2)*

> "To summarize, we apply ideas from the perturbation-based methods of classical saddle point optimization to the model-free self-play RL regime. This results in a novel population-based policy gradient method with a principled adversarial opponent-selection rule. Analogous to the standard model-free RL setting, we assume only "naive" players (Jafari et al., 2001) where the game dynamic is hidden and only rewards for their own actions are revealed." (Zhong et al 2020:2)

*Methodology (note on p.2)*

> "Reinforcement learning trains a single agent to maximize the expected return in an environment (Sutton & Barto, 2018)." (Zhong et al 2020:2)

> "Multiagent reinforcement learning (MARL), of which two-agent is a special case, concerns multiple agents taking actions in the same environment (Littman, 1994)." (Zhong et al 2020:2)

> "Self-play is a training paradigm to generate data for MARL and has led to great successes, achieving superhuman performance in several

> domains (Tesauro, 1995; Silver et al., 2016; Brown & Sandholm, 2019)." (Zhong et al 2020:2)

> "Applying RL algorithms naively as independent learners in MARL sometimes produces strong agents (Tesauro, 1995) but not always." (Zhong et al 2020:2)

> "People have studied ways to extend RL algorithms specifically to MARL, e.g., minimax-Q (Littman, 1994), Nash-Q (Hu & Wellman, 2003), WoLF-PG (Bowling & Veloso, 2002), etc. However, most of these methods are designed for tabular RL only, therefore not readily applicable to continuous state action spaces or complex policy functions where gradient-based policy optimization methods are preferred." (Zhong et al 2020:2)

> ". Monte Carlo Tree Search (MCTS) is also effective in Go (Silver et al., 2016)." (Zhong et al 2020:3)

> "However, Tree search requires learners to know (or at least learn) the game dynamics." (Zhong et al 2020:3)

> "The other ones typically require maintaining some historical quantities." (Zhong et al 2020:3)

> "In Fictitious play, the learner best-responds to a historical average opponent, and the average strategy converges." (Zhong et al 2020:3)

> "Furthermore, most of those algorithms are designed only for discrete state action games." (Zhong et al 2020:3)

> "Special care has to be taken with neural net function approximators (Heinrich & Silver, 2016)." (Zhong et al 2020:3)

*Formulation for the reward in such games(note on p.3)*

> "adversarial rule (Eq. 3) is adopted in Alg. 1 L6 to choose the opponents adaptively. The intuition is that v i and u i are the most challenging

> opponents in the population for the current x i and y i" (Zhong et al 2020:4)

> "This setting is particularly relevant to practical robotics research, as we believe success in this simulation could be transferred into the realworld." (Zhong et al 2020:8)

> "When training x i , our method finds the strongest opponent (that incurs the largest loss on x i ) from the population, whereas the baselines always choose (possibly past versions of) y i . Since the candidate set contains y i , the "fall-back" case is to use y i as opponent in our method. We report the frequency that y i is chosen as opponent for x i (and x i for y i likewise)." (Zhong et al 2020:14)

> "This gives a sense of how often our method falls back to the baseline method." (Zhong et al 2020:14)