Population-based Evaluation in Repeated Rock-Paper-Scissors as a Benchmark for Multiagent Reinforcement Learning

Marc Lanctot
Google DeepMind

John Schultz
Google DeepMind

Neil Burch
Sony AI

Max Olan Smith
University of Michigan

Daniel Hennes
Google DeepMind

Thomas Anthony
Google DeepMind

Julien Pérolat
Google DeepMind

ABSTRACT

Progress in fields of machine learning and adversarial planning has benefited significantly from benchmark domains, from checkers and the classic UCI data sets to Go and Diplomacy. In sequential decision-making, agent evaluation has largely been restricted to few interactions against experts, with the aim to reach some desired level of performance (e.g. beating a human professional player). We propose a benchmark for multiagent learning based on repeated play of the simple game Rock, Paper, Scissors along with a population of forty-three tournament entries, some of which are intentionally sub-optimal. We describe metrics to measure the quality of agents based both on average returns and exploitability. We then show that several RL, online learning, and language model approaches can learn good counter-strategies and generalize well, but ultimately lose to the top-performing bots, creating an opportunity for research in multiagent learning.

KEYWORDS

reinforcement learning, population, benchmark, roshambo

1 INTRODUCTION

How should agents be evaluated when learning with other learning agents? One metric is simply the average return over an agent's lifetime. Another is the agent's robustness against a potential nemesis whose goals are only to minimize the agent's return. The first is the conventional metric used in the evaluation of reinforcement learning (RL) agents, while the second is quite common among game-theoretic AI techniques for competitive imperfect information games. In our paper [1], we argue our position that neither of these is generally sufficient in isolation: good agents should both maximize return *and* be robust to adversarial attacks.

The classical method to demonstrate superior AI performance is head-to-head matches or direct comparisons of average return, against the strongest known agents. This method has driven progress since the beginning: from Samuel's checkers program, to chess, Go, poker, modern real-time games, and so on. On the other hand, game-theoretic approaches to learning result in agents that play approximate Nash equilibrium strategies. The extent to which current AI systems are robust to adversarial attacks is unclear. There is evidence that even expert level AI agents can be demonstrably susceptible to adversarial behavior. While current evaluation methodologies over-emphasize the single metric of cumulative reward or performance against experts, we argue that the more important problem is the lack of benchmarks that prioritize the evaluation of agents in a more general way, where multiple metrics could lead to a better understanding of an agent's capabilities.

In this two-page abstract, we summarize a recently proposed benchmark based on the Repeated Rock, Paper, Scissors (RRPS) [1]. The benchmark contains a suite of bots of varying skill levels used in previous competitions along with a new metric for ranking RRPS agents that takes into account both cumulative reward and exploitability. One-shot RPS is a well-understood two-player zero-sum game whose game-theoretic optimal strategy is well-known, and by construction maximizing rewards against fallible opponents naturally leads to behavior that is potentially exploitable. For learning agents to find exploits in the opponents, they must correctly deduce their strategies from observations. We train agents using several modern approaches against the population and independently trained against copies of themselves. These approaches show promise: out-of-distribution generalization of exploitative behavior, a clear lack of exploitable behavior, and a good balance between these two metrics. Ultimately, none of the agents are able to outperform the top two participants in head-to-head matches while being more robust to exploits, leading to a challenge and opportunity for novel multiagent reinforcement learning research.

2 REPEATED ROCK, PAPER, SCISSORS

A normal-form game has a set of players $\mathcal{N} = \{1, 2, \cdots, n\}$. A matrix game is a two-player game with a set of actions per player $\mathcal{A}_1$ and $\mathcal{A}_2$, a joint action set $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2$, and utility functions for each player $i \in \mathcal{N}$, $u_i : \mathcal{A} \rightarrow \Re$. A zero-sum game is one where $\forall a \in \mathcal{A}, \sum_{i=1}^{n} u_i(a)$. Rock, Paper, Scissors (RPS), also called RoShamBo, is a two-player zero-sum matrix game where Rock beats Scissors, Paper beats Rock, and Scissors beats Paper. RPS is a commonly-used first example in the study of game theory because there is a unique fully-mixed Nash equilibrium where players choose each action uniformly and every pure strategy is fully exploitable. The sequential version is repeated: there are $K = 1000$ identical plays of RPS. At state s_0, agents simultaneously decide their actions and agent i receives intermediate reward $r_{t,i}$ by joint action a_t composed of all agents' actions combined. Every episode has length K and the full (undiscounted) return is defined as $G_{0,i} = \sum_{t=0}^{K-1} r_{t,i}$. The

Proc. of the Adaptive and Learning Agents Workshop (ALA 2024), Avalos, Müller, Wang, Yates (eds.), May 6-7, 2024, Online, https://ala2024.github.io/. 2024.

agent's observation is the most recent R actions taken by both players where "full recall" refers to $R = K$.

2.1 Population-Based Evaluation

In early 2000s, Darse Billings ran two Repeated Rock, Paper, Scissors (RRPS) competitions. In each competition, participants were asked to submit a bot to play RRPS, all played within a one-second time limit. Each program had full recall, the entire action sequence in each episode, but nothing more that would identify the other bots. The majority of the entries in the competition were hand-crafted heuristic bots that were developed independently by different programmers. The resulting population consists of 43 bots: 25 entrant bots and 18 seed bots from the first competition, whose approaches and strategies vary significantly and strength range from simple (and exploitable) to quite competitive (and difficult to exploit).

In RRPS, an agent i plays with policy π_i. It is important to both maximize expected return ($\textsc{PopulationReturn}(\pi_i)$) against the population but also to minimize exploitability. The exploitability of an agent's policy is the expected return that a best response opponent would achieve against it. Computing exact exploitability quickly becomes infeasible as R grows, but can be approximated by RL; in the bot population it was found to range from 4.8 to 1000, with an average of 420.3. We introduce a more practical way to approximate exploitability by simply enumerating all opponents P:

$$\textsc{WithinPopExpl}(\pi_i) = \max_{\pi_{-i} \in P} \mathbb{E}_{a \sim (\pi_i, \pi_{-i})}[G_{0,-i}].$$

This metric across bots ranged from 1.2 to 1000, with several reaching this upper-bound, 316.1 on average. While this within-population exploitability generally under-estimates the full exploitability, it was found to recover 50-100% of the RL-learned exploitability consistently across all 43 agents (75.2% on average). One simple way to rank agents under both metrics is to assume they both matter equally: $\textsc{AggregateScore}(\pi_i) = \textsc{PopulationReturn}(\pi_i) - \textsc{WithinPopExpl}(\pi_i)$. These two metrics (population return and within-population exploitability) capture the essence of the RPS game in the repeated setting: ultimately the goal is to maximize return (by predicting opponents' choices), but agents cannot be too exploitable in doing so because that risks giving up reward to adaptive opponents. Hence, aggregate score acts as a summary of how well an agent is performing on both fronts within is population, allowing agent designers to compare a single number. Table 1 shows results for all agents (both bots and learning agents).

Bot/Agent Names	Pop. Return	W.P. Expl.	Agg. Score
Greenberg (bot)	288.15	3.65	284.50
iocainebot (bot)	255.00	5.00	250.00
PopRL	258.00	10.98	247.02
biopic (bot)	196.36	36.66	159.70
LLM (Chinchilla 70B)	201.00	45.80	155.20
ContRM	164.77	16.27	148.51
boom (bot)	169.11	27.93	141.19
shofar (bot)	152.01	16.87	135.14
Q-learning ($R = 10$)	-0.52	8.62	8.10
R-NaD	$[-10, 5]$	$[20, 40]$	$[-50, -25]$

Table 1: RRPS agents and bots ranked by AggregateScore.

3 LEARNING TO PLAY REPEATED RPS

We tried several baseline agents trained in self-play: tabular Q-learning with varying $R \in \{1, 3, 5, 10\}$, DQN, A2C, and Boltzmann DQN. Most baseline agents achieved negative population return or low positive return and were quite exploitable. The best baseline was Q-learning with a recall of 10 with an aggregate score of 8.2.

Contextual regret minimization (ContRM) is another tabular method. One natural choice for making decisions in RRPS is using an adversarial bandit algorithm. One way to frame RRPS as a contextual regret minimization problem is to completely separate each possible recalled history for $R > 0$ into separate contexts, with independent regret minimizing algorithms running in each context. We tried four different regret minimization algorithms with contexts determined by recall lengths $R \in \{1, 2, 3\}$. The best combination achieved an aggregate score of 148.51.

The large language model (LLM) agent queries a 70-billion Chinchilla base model (without any additional fine-tuning) using a simple prompt. The query is to predict the next opponent action and the agent plays the best response to the prediction. This agent performed surprisingly well, performing better than 40/43 bots, achieving an aggregate score of 155.20.

Regularized Nash Dynamics (R-NaD), the algorithm behind the human-level Stratego agent DeepNash, achieved a slightly negative aggregate score, which is not far away from what the uniform random policy achieves. This algorithm achieves a strategy that is hard to exploit but it will not exploit the other players.

Finally, we try a new population-based RL algorithm: PopRL. PopRL is based on the IMPALA RL algorithm and uses a recurrent neural network. PopRL adds opponent identification as an auxiliary task. During training: at the start of each episode, an opponent is sampled which is either one of the 43 bots, or a copy of itself. PopRL then learns to predict which agent it is playing against in addition to maximizing return. During evaluation, the supervised opponent label is not present. PopRL was the best performing learned agent, achieving an aggregate score of 247.02. However, it was unable to beat the top two hand-crafted bots Greenberg and IocianeBot.

4 AVAILABILITY, USE, AND CHALLENGE

The environment, bot population, examples, and some learning agents are freely available in OpenSpiel (https://github.com/google-deepmind/open_spiel). This benchmark was used as the basis of an assignment for a university course in artificial intelligence, with over 170 participants. We hope to encourage studying agent capabilities beyond return maximization. Due to the simplicity of RPS, it has a low barrier to entry. On the other hand, due to the sequential nature of the repeated game and population-based evaluation, adaptation is a key challenge. We hope it will inspire new learning approaches that can outperform the top two bots.

REFERENCES

[1] Neil Burch Max Olan Smith Daniel Hennes Thomas Anthony Julien Perolat Marc Lanctot, John Schultz. 2023. Population-based Evaluation in Repeated Rock-Paper-Scissors as a Benchmark for Multiagent Reinforcement Learning. *Transactions on Machine Learning Research* (2023).