Final Project Document for **OpenPrison**

Prepared for

Franz Kurfess

CSC580 Artificial Intelligence

Winter 2024

Computer Science Department

Cal Poly, College of Engineering

Prepared By

Nathan McCutchen, Nicholas Tan, Daniel Gonzalez, Christine Widden, Luke Rowe

Affiliation: Cal Poly San Luis Obispo, Computer Science and Software Engineering Department

Completion Date: 3/19/2024

Table of Contents

Abstract

The Prisoner's Dilemma is a classic game theory thought experiment from William Poundstone's 1993 book of the same name. In essence, two agents are given the choice to either collaborate with the other or defect. If both agents collaborate, they are both rewarded. If one agent collaborates and the other defects, the one that defects is rewarded greatly, and the one that collaborates gets nothing. If both defect, both receive very small rewards. We used deep reinforcement learning to train an AI agent against a pool of popular prisoner’s dilemma algorithms such as “Tit-for-Tat”. Ultimately, it was found that the learned behavior of our agent depended heavily on the distribution of the opponents it trained against. The highest-performing agent of those we trained learned to emulate “Tit-for-Tat” but did not outperform it.

Introduction, Background and Related Work

Introduction

The prisoner's dilemma is a well-known thought experiment in game theory that explores the tensions between self-interest and cooperation. In this classic scenario, two prisoners are separated and given the opportunity to either cooperate with their partner by staying silent, or to defect and testify against the other in an attempt to receive a lighter sentence. The dilemma arises because the Nash equilibrium results in both prisoners defecting, leading to a suboptimal outcome compared to if they had cooperated. This project explores how intelligent agents developed using deep learning techniques could approach and potentially "solve" the prisoner's dilemma in novel ways compared to traditional algorithms. Existing approaches to computationally modeling the prisoner's dilemma often rely on hardcoded strategies like "tit-for-tat" or evolutionary algorithms. In contrast, this work aims to train an agent using deep reinforcement learning to learn adaptive strategies in an environment simulating the prisoner's dilemma scenario. From a technical perspective, it represents an interesting challenge for deep reinforcement learning involving elements of game theory, multi-agent cooperation/competition, and emergent behavior. Can an agent using the flexible function approximation capabilities of deep learning discover more nuanced strategies than classical approaches?

The intended audience for this project encompasses those interested in intelligent agents and multi-agent systems, deep reinforcement learning, game theory, and the study of cooperation and social dilemmas. It may also appeal to readers intrigued by the philosophical implications of increasingly capable AI systems. The following sections will describe the technical approach, results, and analysis in more detail.

Background

In 1980, political scientist Robert Axelrod held a pioneering computer tournament to investigate successful strategies for the iterated prisoner's dilemma game. Contestants from around the world submitted game-playing programs that would play the prisoner's dilemma repeatedly against all other entries. Surprisingly, the winner was not a highly complex strategy, but an extremely simple one submitted by Anatol Rapoport called "tit-for-tat." [3] This strategy starts by cooperating on the first move, and then simply replicates whatever the opponent played on the previous move thereafter. So it cooperates until the opponent defects, and then it defects on the subsequent moves until the opponent cooperates again.

Axelrod held a second tournament allowing entrants to revise their strategies after analyzing the results from the first. The winner was again tit-for-tat, bolstering the conclusion that reciprocal cooperation represented a successful "meta-strategy" in this iterated game setting. While simple, tit-for-tat codifies insights about human cooperation that have parallels in evolutionary biology, personal relationships, and international relations. Axelrod's tournaments sparked immense interest in the prisoner's dilemma from perspectives spanning economics, psychology, and computer science. This project takes inspiration from Axelrod's pioneering work using computational means to model the prisoner's dilemma. However, rather than manually programming rules for different strategies, we investigate whether more general reinforcement learning techniques can discover effective policies for tackling this cooperation/defection game scenario.

Related Work

There have been various previous attempts to apply adaptive agents and machine learning techniques to the prisoner's dilemma scenario. Sandholm and Crites used a reinforcement learning approach called neuro-evolutionary programming to train an agent to play the iterated prisoner's dilemma against itself [7]. Their agent learned strategies incorporating elements of tit-for-tat through trial-and-error exploration. Ghang and Rendell developed a cooperative reinforcement learning framework where two agents could learn jointly and overcome the rational strategy of constant defection in

one-shot prisoner's dilemmas [8]. Their approach used a stochastic game formulation combined with artificial neural networks as policy approximators. More recently, Wang et al. trained agents using a multi-agent variant of the REINFORCE policy gradient algorithm to play repeated prisoner's dilemma games [9]. Their agents could learn sophisticated strategies outperforming tit-for-tat when facing specific memorized sequences of defections and cooperations from an opponent agent. Our work builds upon these prior machine learning approaches to the prisoner's dilemma, leveraging modern deep reinforcement learning methods.

Difficulty & Relevance

The process of setting up the actual prisoner's dilemma game environment itself was not particularly challenging. However, the most difficult and time-consuming aspect of this project was implementing the Proximal Policy Optimization (PPO) reinforcement learning algorithm to train the neural network agent. Designing a stable and effective PPO implementation required careful tuning of hyperparameters, reward shaping strategies, and exploration vs exploitation tradeoffs. Additionally, ensuring adequate training convergence often required lengthy training times across multiple random seeds. Thus, the difficulty for this project lies within the range of 6 to 7.

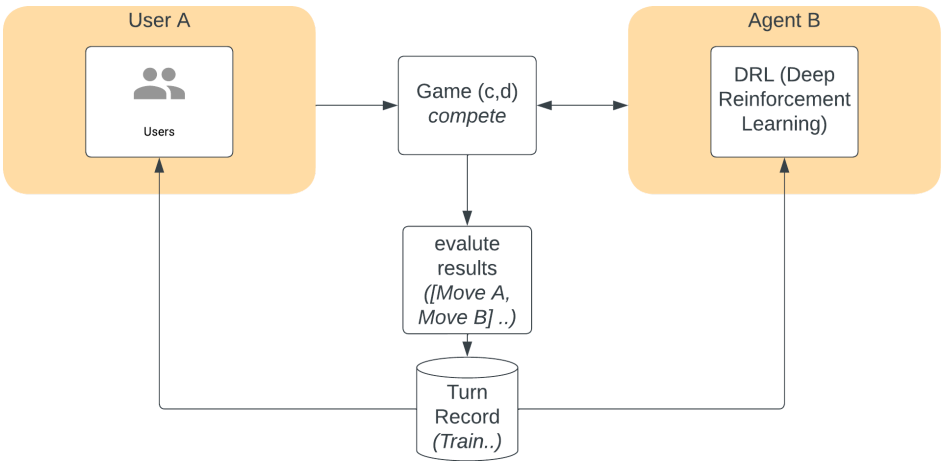
In terms of relevance to this class, the prisoner's dilemma is a canonical problem in game theory that directly models the interaction of self-interested agents or players. The strategies employed by the agents, whether hard-coded or learned, are therefore highly relevant. More broadly, this project explores how intelligent agents based on general machine learning models like neural networks could tackle situations involving cooperation, competition, and social dilemmas. Understanding these behaviors has important implications as AI systems become increasingly autonomous and deployed in multi-agent environments.

System Design, Implementation, and Validation

System Design

Our prototype features a functional command line interface, offering users the option to engage in the Prisoner's Dilemma game against either another user or a pre-existing agent/actor. Within this interface, users have the flexibility to select which agent they wish to compete against, with options including tit-for-tat or grim trigger, among others that we have implemented. Although the number of rounds is currently set arbitrarily, the game concludes by displaying the scores of the two prisoners at the end of the final round, with victory awarded to the actor with the highest score (closest to 0). Our program environment is designed to easily accommodate the integration of different actors/strategies, enabling users to engage with a diverse range of agents, including those developed by us following training.

We also implemented our own agent using a Recurrent PPO model. PPO is a deep reinforcement learning (DRL) algorithm that attempts to maximize reward by searching for optimal behavior, taking small steps at a time. We used the MLP LSTM Policy which uses multilayer perceptron (MLP) layers and Long Short-Term Memory (LSTM) layers. This is suitable when the game state representation requires sequential processing, which is appropriate for the Prisoner's Dilemma turn-based game.



Our final implementation utilizes a Recurrent Proximal Policy Optimization (PPO) model. Built on Stable Baselines 3 and OpenAI's Gymnasium, our setup enables dynamic competitions between diverse agents, showcasing the potential of AI to understand and engage in complex strategic interactions efficiently. The system supports user input which competes against a pre-existing agent, during these simulations inputs are recorded and can be reused for future agent models.

Implementation

Python is our language of choice due to its versatility in handling AI tasks. Leveraging Python, we developed a custom training environment/simulation designed for running iterated prisoner's dilemma sessions, both for training and testing purposes. To facilitate our deep reinforcement learning (DRL) model training, we rely on the StableBaselines3 framework as our primary tool which provides a set of reliable implementations of RL (Reinforcement Learning) algorithms in Pytorch. This library is the successor to the original Stable Baselines, which was built on TensorFlow.

Our features dynamically initialize to accommodate either single or multiple pre-defined opponents, along with customizable episode lengths and strategic lookahead capabilities for depth analysis. Its core functionalities include generating observations based on the current game state, executing game steps with reward calculation, and providing the option for complex lookahead strategies to anticipate the optimal achievable scores. The environment also supports resetting for new episodes, ensuring versatility in testing various strategies under different conditions.

Results

In our experiments, we found that the behavior of the deep reinforcement learning (DRL) agents varied depending on the distribution of opponent strategies present in the training pool. This demonstrated the adaptive nature of the DRL approach - the agents could learn diverse policies tailored to the specific multi-agent environment they were exposed to. The most common emergent behavior we observed was one similar to the hard-coded "Always Defect" logic agent. This uncooperative, exploit-oriented strategy tended to arise when the opponent pool contained a "Random" agent that followed no cohesive policy. In these conditions, the DRL agent learned that defecting against other opponents was the optimal way to accrue rewards over the training episodes.

The second most frequently observed behavior from the DRL agents aligned closer to the classic "Tit-for-Tat" strategy. The agents seemed to learn the benefits of initially cooperating, then reciprocating their opponent's previous moves to encourage cooperation when faced with a mix of gameplans. We noted this tit-for-tat-esque behavior most consistently when the opponent pool consisted of an ensemble including Tit-for-Tat, Generous Tit-for-Tat, Always Defect, Always Cooperate, and Grim Trigger agents. Against this diverse but cooperative-leaning set of opponents, the DRL agents converged to a policy of initial cooperation with retaliatory defections against exploitative behavior.

Validation

Due to time constraints, we were unable to rigorously test and validate our deep reinforcement learning agent's performance across the full suite of desired evaluation metrics for this project. A more comprehensive empirical analysis remains an important area for future work.

Ideally, we would benchmark the agent's effectiveness by measuring criteria such as:

- Win rate of a single iterated game per actor (against a specific opponent strategy)

- Win rate across multiple iterated games (against various opponent strategy combinations)
- Cumulative rounds won across a single iterated game (vs. a single opponent)
- Cumulative rounds won across multiple games (vs. various opponents)
- Qualitative assessment of whether the agent exhibits intelligent, adaptive gameplay
- Comparative analysis of how closely the agent's behaviors resemble or differ from other established strategies

Obtaining robust quantitative results across all of these metrics would allow us to better validate the agent's performance and directly compare it against theoretical optimal policies as well as other machine learning approaches. For example, it would be very insightful to analyze how the agent's win rate and cumulative scores vary when facing different regimes of opponents over many thousands of iterations. Does it learn to ruthlessly exploit aimless "Random" strategies? Can it sustain mutually cooperative relationships with other reciprocal agents? How does it fare against periodically exploitative game plans?

Such rigorous testing and validation is crucial for ensuring the agent generalizes properly and does not simply memorize idiosyncratic patterns from its original training experience. Continuing this evaluation across diverse environments is an exciting future direction to further understand the strengths and limitations of general reinforcement learning methods for multi-agent collaboration and competition scenarios like the iterated prisoner's dilemma.

Future Work and Conclusions

Future Work

To enhance the ethical and social impact of the Prisoner's Dilemma AI project, future work could focus on integrating a feedback mechanism that guides users on improving their project's alignment with ethical and social justice principles. This might include visual data analysis on ethical considerations and adapting the AI model based on user feedback and usability studies. Introducing A/B testing tools for empirical evaluation and expanding to speech interaction features are also promising avenues. These improvements aim to refine the AI's technical performance while ensuring its development and use are ethically responsible and socially inclusive.

Furthermore, implementing a version of our system that accounts for cross-episode performance may result in more interesting emergent behaviors, which is something that may be worth exploring. We also imagined an alternate version of IPD where agents can communicate intentions before each turn. However, we were unable to reach this stretch goal within the time frame of the quarter. Yet, emergent behavior of DRL models trained with pre-turn communication could be an interesting avenue for future research within our project.

Conclusions

The results from our deep reinforcement learning agent on the iterative prisoner's dilemma task revealed some interesting emergent behaviors and strategies. However, we believe the nature of these behaviors stemmed primarily from the narrow reward structure used during training. Specifically, our reward function only accounted for the payoff matrix results from a single isolated iteration of the game. While this incentivized the agent to learn to maximize its reward for any given individual match, it did not accurately represent the agent's overall performance across a sequence of games against dynamically changing opponents. In the canonical prisoner's dilemma, agents

are scored based on their cumulative rewards over many repeated game iterations. This allows for policies and strategies that potentially makeLocalWords at one game step to pay off in the long run through reciprocal cooperation or preventing exploitation. Our single-game episodic reward structure failed to fully capture these longer-term strategic elements.

To better align with the true spirit of the iterated prisoner's dilemma, it would be necessary to introduce a concept of cross-episode performance scoring into the reward function. This could involve having the agent's total reward accumulate over multiple successive game iterations, perhaps against an ensemble of held-out opponent strategies. However, implementing such a cross-episode reward setup seems quite difficult to achieve given the current state of our system's architecture and training pipeline. It would likely require an overhaul of how experiences are sampled across episodes and how long-term reward signals are propagated through the agent's neural network. This represents an important area for future research.

Despite the limitations discussed above, we believe this project demonstrated the potential power of deep reinforcement learning techniques to discover intelligent strategies in simple multi-agent dilemma scenarios. With key improvements to the reward scheme and training process, these kinds of general-purpose adaptive agent models could potentially make important contributions to our understanding of cooperation, social dynamics, and equilibria concepts from game theory. The prisoner's dilemma served as an illustrative starter problem, but many other rich opportunities await in the growing field of multi-agent reinforcement learning.

References

[1] SB3-contrib. “Proximal Policy Optimization (PPO) - Recurrent Policies.” Read the Docs, SB3-contrib, https://sb3-contrib.readthedocs.io/en/master/modules/ppo_recurrent.html.

[2] Stable Baselines Team. “Stable Baselines Getting Started.” Google Colab, Colab Research, https://colab.research.google.com/github/Stable-Baselines-Team/rl-colab-notebooks/blob/sb3/stable_baselines_getting_started.ipynb.

[3] Axelrod Development Team. “Overview of Strategies.” Axelrod Documentation, Axelrod Development Team, https://axelrod.readthedocs.io/en/fix-documentation/reference/overview_of_strategies.html.

[4] CleanRL. “PPO Implementation Details.” CleanRL Blog, CleanRL, https://ppo-details.cleanrl.dev//2021/11/05/ppo-implementation-details/.

[5] Farama. “Gymnasium.” Farama, https://gymnasium.farama.org/index.html.

[6] Brownlee, Jason. “When to Use MLP, CNN, and RNN Neural Networks.” Machine Learning Mastery, Machine Learning Mastery, 20 June 2019, https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/.

[7] Sandholm, T.W. and Crites, R.H., 1995. Multiagent reinforcement learning in the iterated prisoner's dilemma. Biosystems, 37(1-2), pp.147-166.

[8] Ghang, Y.H. and Rendell, L., 2004, September. Coevolutionary prisoner's dilemma game model for cooperative reinforcement learning algorithm. In International Conference on Parallel Problem Solving from Nature (pp. 341-350). Springer, Berlin, Heidelberg.

[9] Wang, R., Liang, Z., Wang, Y., Wang, X., Sun, J., Yuan, Y. and Xing, E.P., 2022. Learning Policies for the Infinitely Repeated Prisoner's Dilemma via Monte Carlo Tree Search. arXiv preprint arXiv:2204.04420.