Implicitly Modeling Opponents *

WashU CSE 527A Project

Nicholas Sullivan
Washington University in St. Louis
d.sullivan@wustl.edu

Ravi Dantuluri
Washington University in St. Louis
d.ravi@wustl.edu

Abstract

Deep reinforcement learning (RL) techniques have been applied to solve classical games from game theory. Deterministic non-zero-sum games of complete information such as infinitely repeated Prisoners' Dilemma (Wang et al., 2018), Cournot Duopoly (Shi and Zhang, 2020), etc, have been used to benchmark algorithms. There also has been significant research on stochastic imperfect information games. It has been focused it has been focused particularly on zero-sum games. For example, techniques like ReBeL (Silver et al., 2016) have been shown to converge to a Nash Equilibrium (NE) in poker. Other works, namely, (Foerster et al., 2017) find solutions to non-zero-sum games. Both employ strategies to explicitly model other agents' behavior. We focus on two games from classic game theory, Iterative Prisoner's Dilemma (IPD) and Iterative Signaling Games (ISG). We show that dueling double DQNs with recurrent neural networks (RNNs), can implicitly model opponents. In IPD, agents employed tit-for-tat strategies to enforce cooperation. In ISG, agents maintained some semblance of cooperation but some deviation still occurred.

Introduction Multi-agent RL methods have flourished in the past few years. Most of the work is either focused on fully cooperative games or completely competitive games (zero-sum). In the real world, there are few fully cooperative situations due to the principle-agent problem, and very few zero-sum situations (outside of games). Prisoners' Dilemma is a commonly studied non-zero sum game. Two prisoners simultaneously choose to either cooperate or betray each other. The Nash Equilibrium of this game is for each prisoner to "Defect". If both players choose "Cooperate" it is profitable for each player to deviate to "Defect" preventing it from being a Nash Equilibrium. Cooperation

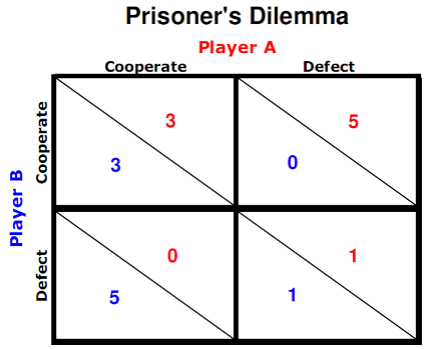

Figure 1: Prisoner's Dilemma example

occurs when the game repeats infinitely. In the infinitely repeated version of this game, agents can force their opponents to cooperate via punishment. Consider a scenario where both agents are cooperating. If one prisoner were to deviate from this cooperation, then the other prisoner would punish them by playing "Defect" for a set number of turns. Any agent thinking about defecting has to consider how many turns their opponent will play "Defect" for. If an agent values the future rewards for cooperating over the immediate reward for defecting plus the smaller future rewards due to the punishment then that punishment

*The LaTeX template is adopted from ACL style and formatting.

is effective. Effective punishments are different for different agents. The most important factor to consider is the other agents' discount factor. Impatient agents (meaning those with smaller discount factors) value their future less and thus require a harsher punishment to prevent them from defecting.

Signaling games are a type of dynamic Bayesian game. The game starts with *nature* choosing a type for the player(s) which are called *sender(s)*. This type is private information for the sender(s). The players who don't know the result of the stochastic element, are called the *receiver(s)*. They must infer the senders' type via the actions of the sender(s). Three distinct Nash equilibria arise from this seemingly simple game.

1. *Polling equilibrium*: occurs where the sender(s) plays the same action regardless of the result of the stochastic element. This means after the sender(s) actions the receiver(s) gain no new information.

2. *Separating equilibria*: occur when the sender(s) always chooses different signals based on the result of the stochastic element. This reveals the result of the stochastic element to the receiver(s) leading them to know the result of their actions.

3. *Semi-separating equilibrium*. This is essentially a mixed NE for the game. Senders will pick their actions with some probability. This can be a bluff! While bluffs have been extensively studied in zero-sum games there is a lack of results in non-zero-sum games where agents have the choice to co-operate.

Because of the lack of information, the sender can bluff the receiver. In the repeated version of the game, this problem is amplified. Cooperation in the repeated game is more complicated than in the stage game and in IPD. (Sugaya and Yamamoto, 2020) showed that both players can enforce cooperation for most signaling games. We only have used ones that fall under those conditions for obvious reasons.

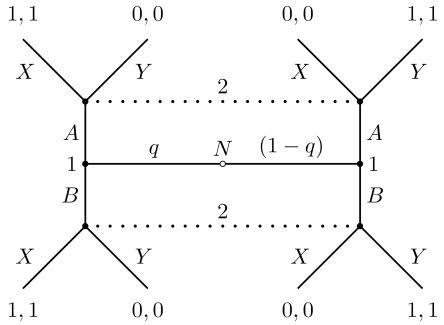

Figure 2: Signaling Game example

1 Lola

Lola agents(Foerster et al., 2017) explicitly model other agents. They do this by estimating their opponents' change in parameters. Instead of treating their opponents statically, updating optimizing

$$V(\theta_t^1, \theta_t^2)$$

they optimize

$$V(\theta_t^1, \theta_t^2 + \Delta\theta_t^2)$$

Each agent anticipates the opponent's updates by modeling opponents' parameters using maximum log-likelihood. The authors noted that their Lola agents successfully found and implemented the tit-for-tat strategy in IPD. This is when you punish your opponent for a single round.

2 Approach

When playing a repeated game against opponents we remember past turns and their temporal relationship. For example

in poker, someone might always bluff two hands after they win big. RNNs are very effective at learning data with temporal relationships. More importantly, punishments are inherently temporal. Deciding when to punish opponents and how long to punish opponents relies on understanding that opponent. Agents can also exploit weak other agents whose punishments aren't harsh enough. This can only be done if you know the temporal relationships of the current game meaning you know how and why your opponent punished you. We hypothesize that using RNNs in DQN, agents will allow the agents to implicitly model opponents and come up with effective strategies. We use dueling DQN with an RNN architecture and show that they can successfully learn to cooperate, through punishments on both repeated prisoner's dilemma (IPD) and repeated signaling games (ISG). We will refer to our architecture as RNN DQN and the baseline as linear DQN.

3 Methods

To implement, DQN we forked (XinJingHao, 2024)'s implementation of DQN. We modified this and added 2 RNNs for IPD and 3 RNNs for ISG. The RNNs' lengths were determined by the maximum length of the games (in our experiments they were mostly 100). The output of the RNNs would is concatenated and used as the input to a fully connected linear layer. Each linear layer had a hidden size of 200. We experimented with this and found 200 to find the solution while also maintaining performance. Here are the full results of our hidden state experimenting fig:3. The RNNs' inputs for IPD were all of the moves the opponent had made throughout the current game and the agents' moves made throughout the current game.

We algorithmically generated the IPD payoffs. We did this by first randomly generating two integers for each cooperation

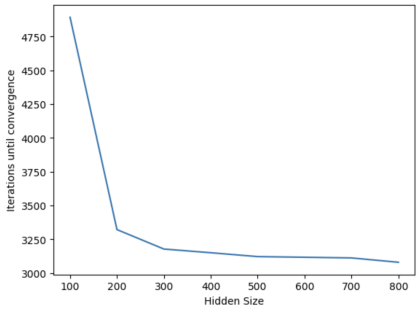

Figure 3: Percent of cooperative rounds during evaluation in Prisoner's dilemma

payoff. Then generating smaller integers at least 2 away from the original integer. This was the payoff if you cooperated while your opponent defected. Next, we generated the two largest integers for the payoff if you defected and your opponent cooperated. Lastly, we generated a middle integer for the double defect square. We found this made no difference in performance and thus we chose the example PD payoffs 1.

Because of our findings for IPD we used the payoff in the 10. We found the stage game Nash Equilibrium by hand. This game also satisfies the conditions in (Sugaya and Yamamoto, 2020) meaning both players can effectively punish one another.

We created these games by creating custom Gymnasium environments. We found that this would be the easiest course of action. Our state was different for each game. In the IPD, the state was all of the moves each player had played in the current game (padded with -1s). For ISG the state was all of the moves each player had played in the current game and all of the senders types. However, the receiver would only get the past sender's type but would get the current sender's move. The sender would get the past receiver moves but would get the current sender's type. This is due to the games' sequential structure.

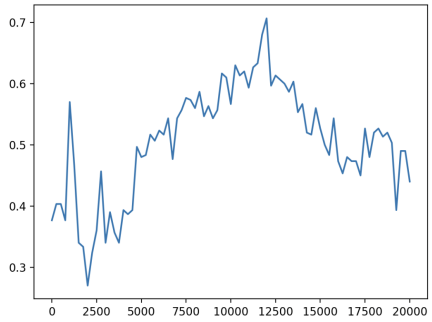

Figure 4: Percent of cooperative rounds during evaluation in Prisoner's dilemma

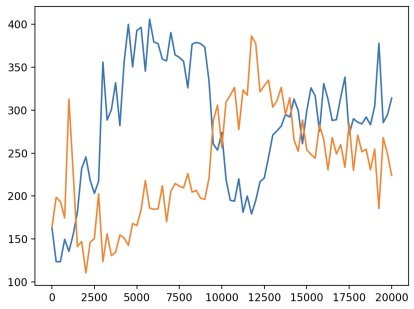

Figure 5: Payoff per round for Linear Agents in IPD

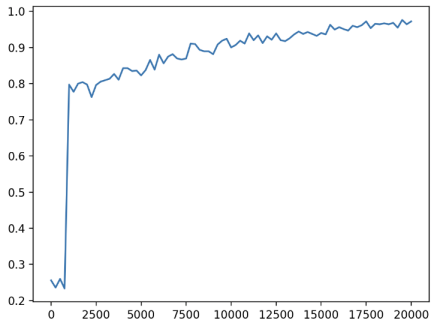

Figure 6: Cooperation per round for RNN agents in IPD

4 Experimental Setup

We played a maximum of 100 rounds of each game. For every training game, we randomize the number of rounds being played. For the evaluation games, we play the full 100 games. The input for agents in IPD is all moves made so far. The input for agents playing ISG is the moves played for the previous round (and the senders' move for the receiver) and the senders' type for every round (The receiver does **not** know the current senders' type). We used dueling-double DQN. The first 1,000 games were randomly played. We used a discount factor of .99, a learning rate of 1e-4, and exploratory noise of .1 with a decay rate of .99. We updated every 50 intervals and trained for a total of 20,000 steps. Additionally, we used a batch size of 256 and a memory size of 1e6. As a baseline, we used a dueling-double DQN with only linear layers. We evaluate the agents every 100 steps with a full-length game. We generated the games as described in the methods section.

5 Results

5.1 Prisoner's Dilemma

Linear agents completely fail to learn cooperation in both IPDs and ISGs. We can see here that the linear DQN fails to learn any cooperation. At around 12k iteration around 70% of the moves are cooperative. However, one agent takes advantage of the opponents' willingness to cooperate and starts to deviate.

The RNN DQN learns to cooperate quickly. After the 1,000 randomized steps, the ratio of cooperative to selfish moves is .8. It steadily increases throughout the training steps. Payoffs also steadily increase. They eventually converge to the maximum value value possible using cooperation.

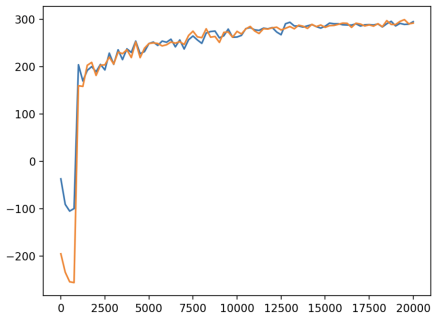

Figure 7: Payoff per round for RNN Agents in IPD

Figure 8: Payoff per round for Linear Agents in ISG

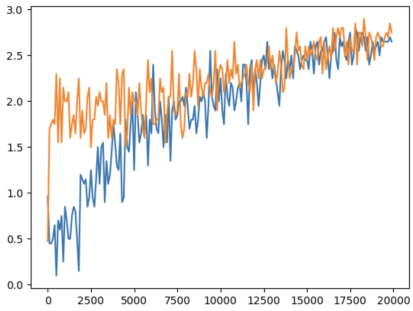

Figure 9: RNN agents payoffs over time in ISG

5.2 Signaling Games

Cooperation is harder in Signaling games. The linear DQN agent completely fails at any sort of cooperation. Through testing we found it did not react to a defection in the previous round. While the RNN agents seem to cooperate they still struggle.

5.3 Discussion

In IPD, RNN DQN agents quickly converged to cooperation. We saved the model after 20k steps and manually tested it. We found that it employed a tit-for-tat strategy consistently. On the other hand, linear DQN agents failed to converge to cooperation. Each agent would not react much to past moves. Instead, they would almost randomly choose when to cooperate and when to defect. When their opponent learned to cooperate more they would learn to defect. This is why the graphs seem cyclical.

In ISG, linear DQN agents failed to find any sort of cooperation. The sender agent learned to react to its type but played greedily. The receiver also learned to react to the current sender's move but failed to cooperate. For RNNs DQN, agents cooperated. However, looking at the evaluation games we found that agents played a kind of grim strategy. Each agent cooperated for roughly 70 periods then one agent would deviate. After this, both agents would greedily deviate much more. Through some testing, we found this was a form of punishment. We played deviate much sooner than round 7. At this point, the agent would defect for around 10% of rounds.

In IPD we also tested decreasing the discount factor of one agent to .7. This meant that the agent is not affected by a single round punishment and values the immediate deviation reward. In this case, the other agent learned to punish for 2 turns instead of just one. Linear agents did not seem to be affected by this and did not cooperate at all. In ISG it was a different story. Cooperation was unable to occur at all for linear and for RNN agents.

5.4 Horizon

We also tested out different lengths of history. Longer lengths tended to increase the state space so much that it took longer to converge. Agents would play more randomly in these scenarios. However, very small lengths (10) failed to produce any cooperation at all. They in fact defected each time. We hypothesize that this was because the games were so small that agents learned them as finitely repeated games. In finitely repeated games, you should always deviate on the last round. Because you know both agents are defecting on the last round both agents defect on the 2nd to last round. This makes

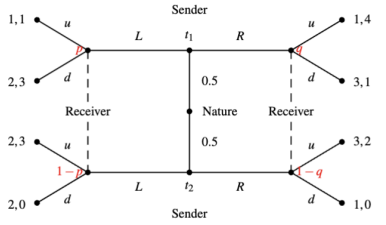

Figure 10: Example Signaling Game

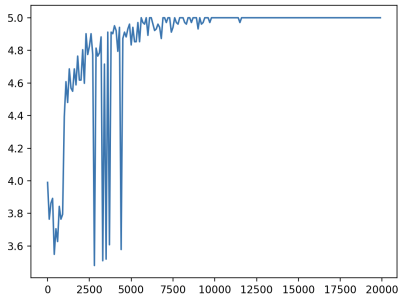

Figure 11: Linear Agent payoffs for a stage Signaling Game (payoffs different than in the repeated Signaling Game)

defecting each round a Nash Equilibrium.

5.5 Stage Game

Additionally, we found that both linear DQN and RNN DQN can find NE in both stage games (non-repeated). In ISG linear and RNN DQN could find-pooling and separating equilibrium. We found that both agents converged to the pooling NE when there was both a pooling equilibrium and a separating equilibrium. Here is an example of a signaling game. Its' separating equilibrium is at LR, DU, and pooling NE at LL, UU. The performance of which is seen in fig:11.

Our last test was to have the sender able to see all of the history while the receiver could only see the current state. In this scenario, the sender found a partial pooling equilibrium. This suggests that without taking the argmax agents would learn this mixed NE. Future work is needed.

For all of the SG stage games we tried the models converged to some *Perfect Bayesian Equilibrium* quickly.

5.6 Conclusion

We have found that RNNs can be used in DQN and perform extremely well. They utilize punishments to enforce cooperation in both a simple deterministic repeated game like Prisoner's Dilemma as well as in a more complicated noisy scenario, namely in Signaling Games. We believe that implicitly modeling opponents is viable in complex multi-agent settings.

References

Jakob N Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. 2017. Learning with opponent-learning awareness. *arXiv preprint arXiv:1709.04326* .

Yuanyuan Shi and Baosen Zhang. 2020. Multi-agent reinforcement learning in cournot games. In *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, pages 3561–3566.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of go with deep neural networks and tree search. *nature* 529(7587):484–489.

Takuo Sugaya and Yuichi Yamamoto. 2020. Common learning and cooperation in repeated games. *Theoretical Economics* 15(3):1175–1219.

Weixun Wang, Jianye Hao, Yixi Wang, and Matthew Taylor. 2018. Towards cooperation in sequential prisoner's dilemmas: a deep multiagent reinforcement learning approach. *arXiv preprint arXiv:1803.00162* .

XinJingHao. 2024. *XinJingHao/Duel-Double-DQN-Pytorch*. https://github.com/XinJingHao/Duel-Double-DQN-Pytorch.