Oases of Cooperation: An Empirical Evaluation of Reinforcement Learning in the Iterated Prisoner's Dilemma

Peter Barnett, John Burden [1]

[1] Centre for the Study of Existential Risk, University of Cambridge
peterbarnettnz@gmail.com

Abstract

In the creation of safe AI systems it is extremely important to ensure cooperative behaviour of these systems, even when there are incentives to act selfishly. In many cases, even when game-theoretic solutions allow for cooperation, actually getting the AI systems to converge on these solutions through training is difficult. In this paper we empirically evaluate how reinforcement learning agents can be encouraged to cooperate (without opening themselves up to exploitation) by selecting appropriate hyperparameters and environmental perceptions for the agent. Our results in the multi-agent scenario indicate that in hyperparameter-space there are isolated "oases" of mutual cooperation, and small changes in these hyperparameters can lead to sharp drops into non-cooperative behaviour.

Introduction

In a world where AI systems are becoming ever more ubiquitous, it is increasingly important that these systems cooperate effectively. The notion of cooperation is certainly not unique to AI, it is a key facet of human society as well as certain animal populations such as ant colonies. Large parts of AI research, however, have focused on either single systems interacting alone in a domain or multiple agents directly competing against each other, where cooperation cannot occur. Non-cooperative behaviour can easily arise when systems have conflicting goals. Even when goals of systems are aligned, non-cooperation can occur if the agent is valuing its own contribution more than "global" outcome. For example, consider an vacuum cleaner that is rewarded for the amount of dirt it cleans from the floor. In a scenario with multiple vacuum cleaners, even though they have the exact same goal, they each have an incentive to incapacitate other cleaners other than itself in order to clean more of the dirt themselves. This can be mitigated (as with humans) by getting the system to instead value the global outcome. However, this is often easier said than done, as has been shown in Game Theoretic analyses, notably the Prisoner's Dilemma, where the Nash Equilibrium (NE) occurs at non-cooperation (Rapoport, Chammah, and Orwant 1965). Even when Game Theory can encourage cooperation, with many AI systems there is the added difficulty of training them to reach these points of cooperation. Within deep reinforcement learning (RL) there are often no guarantees about the type of policy that will ultimately be reached.

In this paper we present an empirical exploration of encouraging cooperative and robust strategies for the Iterated Prisoner's Dilemma. We aim to identify the properties of training and of the environment representation that enable more cooperative policies to be learnt.

		Player 2	
		Cooperate	Defect
Player 1	Cooperate	(3, 3)	(0, 5)
	Defect	(5, 0)	(1, 1)

Table 1: Payoff matrix for the Prisoner's Dilemma. The tuples are the rewards for each player depending on the actions taken, for example if Player 1 plays cooperate and Player 2 plays defect, then Player 1 receives a reward of 0 and Player 2 receives a reward of 5.

Iterated Prisoner's Dilemma

In the Iterated Prisoner's Dilemma (IPD) two players play a series of Prisoner's Dilemma games, each turn the players receive reward according to the payoff matrix in Table 1 depending on the actions of both players. This is a very simple game, where each player locally has an incentive to defect, but mutual cooperation is globally preferable. In the case of a single PD game, the only Nash Equilibrium (NE) is mutual defection (Rapoport, Chammah, and Orwant 1965). In the IPD with a finite (and known) number of rounds, the only NE is still mutual defection each turn which follows from a simple backwards-induction argument. However, if the number of rounds is not known (or infinite) then it has been shown that mutual defection is no longer the *only* NE (Aumann 1959), opening opportunities for cooperation.

A well known strategy for the IPD is tit-for-tat, arising in (and winning) Axelrod's seminal IPD tournament (Axelrod 1980), tit-for-tat cooperates on the first turn, and after that always plays the opponents previous move. This strategy rewards the other player for continuing to cooperate and punishes defections, but also allows for "forgiveness".

Related Work

Previous empirical evaluations of reinforcement learning agents' performance on the IPD have been limited to tabular Q-learning and (very) small recurrent neural networks (Sandholm and Crites 1996), occurring before the rise of deep reinforcement learning of the last decade. These studies showed the difficulty of attaining reliable cooperation within the the IPD domain, particularly with the neural network-based approaches. More recent work (Harper et al. 2017) has shown that RL techniques can learn very strong policies for the IPD, though the focus here was on training a single agent to perform well in a tournament setting rather than encouraging cooperative behaviour between agents.

In (Vassiliades and Christodoulou 2010), the authors demonstrate that evolutionary algorithms can be used to improve the rate of cooperation by altering the payoff values within the IPD game matrix (while retaining requisite conditions). (Vassiliades, Cleanthous, and Christodoulou 2009) demonstrates that so-called spiking neural networks can be made to perform more cooperatively in the IPD by giving the networks "stronger memory" in the form of longer eligibility traces.

In (Wang et al. 2019) the authors extend the concept of the Iterated Prisoner's Dilemma into so-called Sequential Prisoner's Dilemma, where the focus is on creating a class of environments that require temporally-extended cooperation in the form of 2D gridworld environments that mimic the dynamics of the prisoner's dilemma.

Distinguishing our work in this paper is an empirical analysis of training more robust agents within the single-agent setting, where we widen the range of hyperparameters where the tit-for-tat strategy is learnt, as well as identifying which properties contribute to encouraging cooperation within the multi-agent scenario. Our focus is not on training agents that achieve high scores against other agents, but rather on trying to ensure and encourage mutual cooperation.

Environment

Reinforcement Learning (RL) (Sutton and Barto 2018) is a machine learning paradigm in which an agent interacts with an environment. The agent learns to take actions based on observations of the environment and learns to update its policy based on received reward. The agent's goal is to find a policy π^* that maximises the expected cumulative reward received over a single episode.

Within RL the environment is often assumed to be a Markov Decision Process (MDP), where the next state depends only on the current state and the action taken, and not explicitly on any previous state. Intuitively, this can be thought of as the system being "memoryless". However, for agents to implement strategies in iterated games they must have a memory of previous moves. We can therefore store an agent's "memory" in the environment; the environment is used as a ledger for moves that each player has made so far. An alternative to this would be for observations to simply be the turns played for the previous game, and use agents with an internal memory such as an LSTM network (Hochreiter and Schmidhuber 1997).

To implement the IPD as an MDP, we must define a suitable state-space, action-space, reward function and transition function. The action-space is simply $\{cooperate, defect\}$. For each game there is a vector of length four representing whether each agent cooperated or defected. These vectors are then essentially concatenated and flattened. The resulting possible vectors are the set of possible states subject to the constraints that an agent cannot both cooperate and defect in a single game and must select at least one. The reward function simply rewards each agent appropriately according to the payoff matrix in Table 1. Finally, the transition function "shifts down" each component s_i of s four indices, and setting s_0 through s_3 appropriately to match the agent's choices in the game that has just occurred.

The initial state has every entry as 0, and the environment "fills up" as games are played. For ease of learning good strategies, the most recent turn is always at the start, the second most recent is second, and so on. A strategy such as tit-for-tat only requires knowledge of the last turn, and so it should be easier to learn this kind of strategy if the last turn is always in the same place. The environment could be designed such that the first game is first and so on, but this would mean that the agent would have to learn to focus on a different part of the observation each turn. Having the most recent move in the same place needn't stop the agent from learning policies which require a longer memory, as all the game history is still stored in the environment.

The observation an agent receives can be any length 'window' of previous games, ranging from length 1 (where there agent only receives knowledge of the last game), to the length of the entire history.

Evaluation Metrics

The RL agents are trained using the rewards received from playing multiple episodes of the IPD. However, the total rewards for each IPD episode isn't a perfect evaluation metric. For example, when playing against a fixed opponent employing tit-for-tat, a strong (but not quite optimal) strategy is to always cooperate. But if an agent learns to simply always cooperate, then it is vulnerable to opponents playing other strategies (for example, always-defect).

For this reason, it seems desirable for agents to learn a strategy similar to tit-for-tat. This means than an agent will cooperate with cooperative opponents, but not be as vulnerable to opponents defecting. We can evaluate how similar a learned strategy is to tit-for-tat by deducing the fraction of moves in which the agent would choose the same action as tit-for-tat. We refer to this measure as "tit-for-tat similarity".

This is calculated by sampling valid observations from the state-space, and for each observation checking the action of the agent against tit-for-tat. A similarity of 1 means the agent is playing exactly tit-for-tat, a fraction of 0.5 means the agent is playing a strategy uncorrelated with tit-for-tat (for example, playing randomly, always-cooperate, or always-defect), and a fraction of 0 means that the agent is always playing the action tit-for-tat doesn't play, "tat-for-tit". In this sampling approach we ensure that observations are generated so that each stage of the game is equally likely.

This measure can be expressed succinctly as

$$\text{tit-for-tat similarity} = \frac{1}{n}\sum_{i}^{n} \mathbb{1}\left(\pi_{RL}(s_i) = \pi_{tft}(s_i)\right)$$

where π_{RL} and π_{tft} are the policies of the RL agent and tit-for-tat respectively and are applied to the randomly generated state s_i, $\mathbb{1}$ is the indicator function, and n is the number of sampled states.

Additionally, we can calculate the rate of agent cooperation on sampled states, further allowing us to determine the type of policy learnt. This can be similarly expressed as

$$\text{cooperation similarity} = \frac{1}{n}\sum_{i}^{n} \mathbb{1}\left(a_{RL}(s_i) = \text{cooperate}\right)$$

Experimental details

Within our experiments we will make use of two common RL algorithms: Deep Q-Networks (DQN) (Mnih et al. 2013) and Proximal Policy Optimization (PPO) (Schulman et al. 2017). These two algorithms were selected to help make the empirical evaluation more general; DQN uses value-iteration and PPO uses policy-iteration. Further, DQN is off-policy while PPO is on-policy.

In this work we have used the DQN and PPO implementations from the Ray RLlib library (Liang et al. 2018). The default configurations were used, with the following modifications: DQN used a noisy network, PPO used a minibatch size of 32 and trained on 20 epochs for each batch. Unless otherwise specified, an episode consisted of 100 PD games. As per the default RLlib configurations, one training iteration is 1000 timesteps for DQN, and 4000 timesteps for PPO.

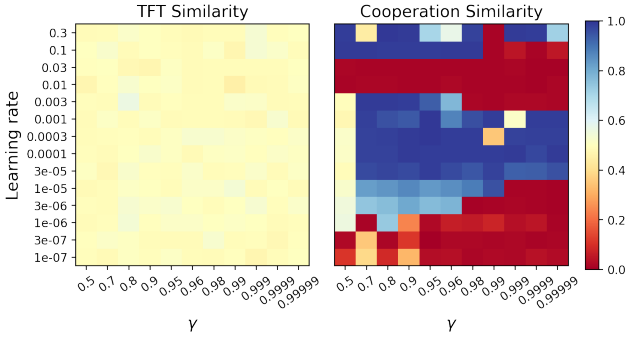
Single Agent Training

We begin by training single agents fixed policies to investigate what strategies developed. Hyperparameter sweeps were performed over the learning rate and the discount rate γ. For each hyperparameter configuration the agents were trained for 200 iterations, and then evaluated to see what type of strategies had developed. The IPDs were 100 games in length, and the agents received the full history as observations. In these experiments DQN has a a hidden layer network of [1024, 512, 256, 32, 8] and PPO has [124,16].

Learning with tit-for-tat

The similarity measures for tit-for-tat and cooperation for agents trained against a tit-for-tat policy are shown in Figure 1. Agents trained against a tit-for-tat policy learned simple strategies, often always cooperating or always defecting. The learned strategies were not at all correlated with tit-for-tat. There were regions of the hyperparameter space where agents learned always-cooperate or always-defect, this was especially prominent for the DQN agents. This is generally what is expected; the policy yielding the largest reward against a fixed tit-for-tat agent is to cooperate in every game except the last. However, the choice of hyperparameters can prevent this policy from being learnt. An agent that is too slow to learn how to respond to tit-for-tat will likely receive more defections from the fixed policy and instead learn to also defect in order to minimise losses. We can see that in all cases, the policy learnt is essentially uncorrelated with tit-for-tat. It's worth highlighting that while we want the agents to be cooperative in application, due to the fact that the evaluation is done using the agent's response to sampled valid states it isn't necessarily better for agents to have a very large cooperation similarity, in this case the agent is likely opening itself up to exploitation from possible opponents.

(a) DQN trained against tit-for-tat



(b) PPO trained against tit-for-tat

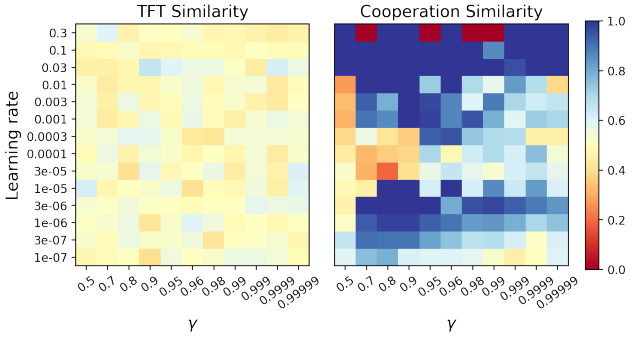


Figure 1: The tit-for-tat similarity and cooperativity for RL agents trained against tit-for-tat, across a range of values for the learning rate and discount rate γ. Different algorithms (DQN and PPO), and different numbers of hidden layers were used. These were evaluated after 200 training iterations. These agents do not learn strategies close to tit-for-tat. There are hyperparameter regions where agents are more likely to learn always-cooperate or always-defect.

Learning with tit-for-tat-then-defect

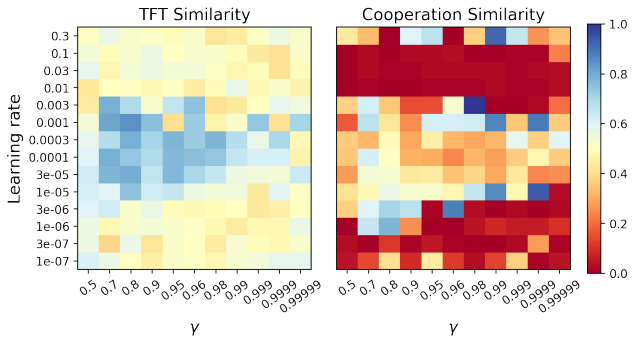
In order to address the agent's vulnerability we train our agent against a different strategy, this new strategy begins each IPD by following a tit-for-tat policy, but then on a random turn begins defecting forever. The intention behind this strategy is that the RL agent will learn to cooperate (as it did against pure tit-for-tat) until the fixed opponent begins defecting, when the agent then needs to switch to defection in order to "defend" itself. Against this "tit-for-tat-then-defect"

opponent the agent needs a policy much closer to tit-for-tat in order to perform well.

Learning a strategy similar to tit-for-tat may be helpful for multi-agent training, if the two agents are both playing a strategy close to tit-for-tat then cooperating and continuing to play tit-for-tat will be rewarded, and defecting punished. This would mean both players playing tit-for-tat could be stable and self-reinforcing.

The results for RL agents learning against this tit-for-tat-then-defect policy are plotted in Figure 2. Similarly to when playing against tit-for-tat there are regions of the hyperparameter-space which lead to different strategies being learned. There are regions of the space which robustly lead to strategies very similar to tit-for-tat (as high as a 0.85 similarity).

(a) DQN trained against tit-for-tat-then-defect



(b) PPO trained against tit-for-tat-then-defect

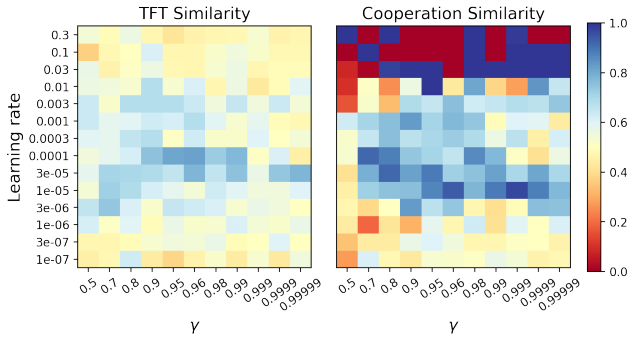


Figure 2: Similarity scores to tit-for-tat and cooperation for RL agents trained against the tit-for-tat-then-defect strategy. Otherwise the same setup as Figure 1. Learning against this strategy lead the agents to learn strategies similar to tit-for-tat. The highest tit-for-tat similarity found for the different agents represented in separate rows was 0.85 (top row), 0.80 (middle row), 0.82 (bottom row). There were regions of this hyperparameter space which lead to agents learning strategies more similar to tit-for-tat. For DQN, particularly high and low learning rates seemed to lead to the agent learning always-defect.

For the DQN agents, the regions where they learn strategies similar to tit-for-tat are quite sharply defined, and outside these regions the agents generally learn to almost always defect. The regions where the DQN agents learn strategies correlated with tit-for-tat seems defined primarily by the choice of learning rate; if the learning rate is either too large or too small the agents will learn to defect. For the PPO agents, the regions where they learn strategies correlating to tit-for-tat is not as well defined. There again seems to be a band of learning rates where the PPO agents can learn strategies which are both generally cooperative and similar to tit-for-tat. Outside of this band, if the learning rate is too large then the agents tend to learn simple always-cooperate or always-defect strategies, and if the learning rate is too small, it appears that no learning has occurred as behaviour appears to be mostly random — there is no correlation with tit-for-tat or cooperation.

Figure 2 highlights concerns for AI systems in the real world; there are sudden jumps in the hyperparameter-space where agents suddenly transition from playing a strategy close to tit-for-tat to playing always-defect. This behaviour is especially clear with the DQN agents. Although there may be some range of hyperparameters which is "safe" in terms of agents trying to cooperate, straying outside of this range at all may be "unsafe" Additionally, for PPO, the region of the hyperparameter-space which generally leads to policies being more similar to tit-for-tat still contains learnt policies which are uncorrelated with tit-for-tat. Here the choice of hyperparameters which leads to the most similar policy (learning rate of 0.0001, discount rate of 0.96) to tit-for-tat is directly adjacent to a policy which is completely uncorrelated with tit-for-tat (learning rate of 0.0003, discount rate of 0.95). Similar behaviour is seen in Figure 1 where regions of always-cooperate are right next to regions of always-defect. For PPO, if the learning rate is too high then potentially desirable (if vulnerable) always-cooperate strategies are found right next to potentially dangerous always-defect strategies.

Multi-Agent training

So far, we have only considered training an agent against a fixed opponent. In more realistic scenarios the both agents can adapt their own policies in response to the other's. We carried out the same experiment as in the previous section but with both agents learning in order to see how RL agents would behave when they can both learn. In this scenario, each player in the IPD is an RL agent, and the training of both agents happens simultaneously. Similar hyperparameter sweeps to the single agent training were performed. In general the multi-agent training was run for 100 training iterations, where a single iteration is still 1000 timesteps for DQN, and 4000 timesteps for PPO. All agents in this section have network sizes of [1024, 512, 256, 32, 8].

Naive Training

We begin by comparing agents with the same algorithm and architecture (but no weight sharing) for a fixed number of games. Both PPO and DQN agents were evaluated for tit-for-tat and cooperation similarities as shown in Figure 3. This time, as well as using the sampling approach, we also show the "observed" results for the tit-for-tat and cooperation similarities which are found by seeing what actions

the two agents take when in the IPD with each other. These observed results are found by having two agents play 20 episodes (2000 games) IPDs together. These sampled similarity and observed similarity can be very different: The sampled similarity is based on a uniform distribution over all valid states, but the observed similarity is based on a distribution of states dependent on the opponent's policy. Within the evaluation of multi-agent games both of these measures are important.

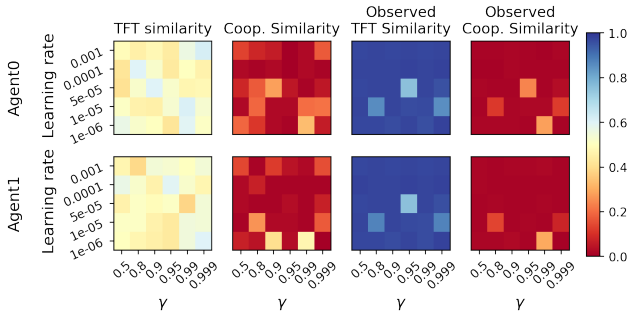
For DQN in Figure 3a, the agents quite robustly learn to always defect, and do not learn strategies close to tit-for-tat. This is likely due to the agents being unable to overcome the local "optimum" of mutual defection. These agents appear to play "tit-for-tat" based on their opponent's move, but this is a trivial, illusory case due to mutual defection. We can see from their behaviour given random states that they are actually playing always-defect rather than tit-for-tat. For PPO in Figure 3b, the agents appeared similarly incapable of learning tit-for-tat. These agents still defected the majority of the time, but not nearly as much as DQN. The results here are expected; the number of games is finite, fixed, and in a sense "known" to the agent because the agent can observe the vector representing the state-space "fill up". The Nash Folk Theorem for finitely repeated games (Benoit and Krishna 1985) shows that in this case (and by backward induction) that we expect this mutual-defection.

Random Game Length

The always-defect behaviour in the naive training for the DQN agents is not unexpected because the IPDs are of fixed length. It is known that the only Nash Equilibrium here is always-defect (Rapoport, Chammah, and Orwant 1965). The agents both have incentives to defect on the last round, because the opponent cannot retaliate to punish this defection. If they both defect on the last round, then they also have incentives to defect on the second last round, and so on. But if games do not have a fixed length, then it is possible for cooperation to be sustained indefinitely (Aumann 1959). Although this is possible according to game theory, it is a separate question whether RL algorithms are capable of reaching and sustaining this cooperation equilibrium. This is because multi-agent training is difficult for RL agents: for each agent the environment is non-stationary. The agents don't explicitly model their opponent, and so the opponent actions are treated as part of the environment dynamics. Because both agents are learning, this means that the effective environments are changing during training. This can cause an agent to start learning a good strategy which becomes obsolete as the other other agent changes in response.

To see if we can achieve stable levels of cooperation, experiments identical to those demonstrated in Figure 3 were run, but instead of a fixed game length the games end on a random turn (so the IPDs can have anywhere from 1 to 100 games). Having a random game length did not help increase the cooperativity or correlation with tit-for-tat of the learned policies (Figure 4). Additionally, having both a random length and no "done" signal at the end of each episode was investigated and did not improve these metrics. Even though cooperation is one of the possible Nash Equilibria

(a) DQN



(b) PPO

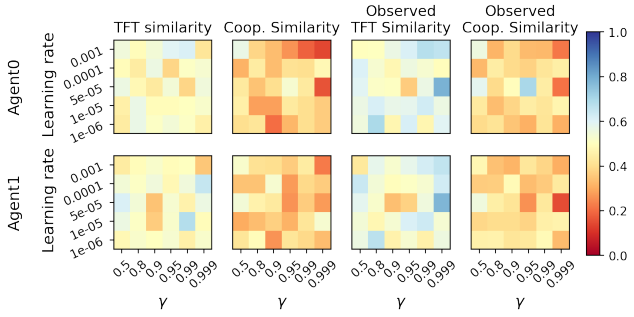
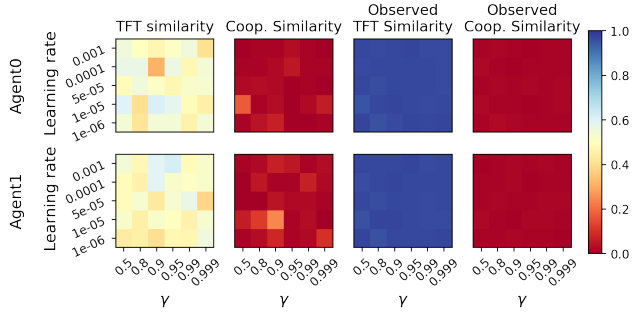


Figure 3: The tit-for-tat similarity and the cooperation similarity for two DQN agents (3a) and two PPO agents (3b) trained with each other in an IPD, for a range of learning rates and discount rates γ. The first two columns are evaluated by seeing what the agents would play given random observations of turns played so far, the second two columns show how the agents act when playing with each other. The two rows are for the two agents.

in this scenario, it is not easily found by RL agents starting from random initialisation.

For AI systems interacting with each other in the real world, there are likely to be scenarios where one system may defect for individual gain. These results from simple IPDs show that just because cooperation is a strong strategy according to game-theory it can still be difficult to achieve convergence with typical RL algorithms. This is likely due to the fact that strategies such as tit-for-tat are not very stable, slight deviations from tit-for-tat, caused by agent exploration or slightly imperfect agent, can prevent cooperation from occurring and being learnt.

(a) DQN



(b) PPO

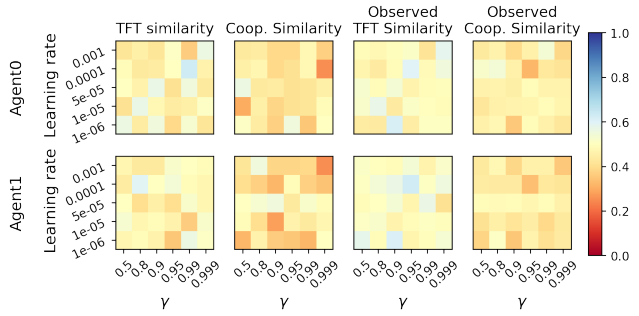


Figure 4: The tit-for-tat similarity and the cooperation similarity for the DQN (4a) and PPO (4b) agents, as in Figure 3 but here the IPDs are all of random length. Training with random lengths does not seem to effect the learned strategies.

Observation Window Size

So far we been using a window-length of 100, the observation the agent receives contains the entire history of both players' interaction with the game. We also experiments evaluating the effect of having shorter window lengths. There are two primary reasons that this may be beneficial. First, training may be faster simply because there are fewer features for the agent to use and therefore more episodes and games can be played in the same amount of time. Second,for short windows there are fewer possible strategies that can be developed. By essentially shortening the agents' memory, we effectively reduce their ability to hold onto a "grudge".

Agents were trained in the same setup as in Figure 4, playing against another agent of the same type in IPDs of random length. But rather than the observation being the entire history of moves that had been played, the observations were only of the last 1 move or last 5 moves, this is referred to as the window-length. The results are plotted in Figure 5. DQN agents with a window-length of 1 and PPO agents with a window-length of 5 (Figures 5a and 5d) performed very similarly to the multi-agent training seen so far; the DQN agents here learned to always defect, and the PPO agents did not learn strategies which were cooperative or similar to tit-for-tat.

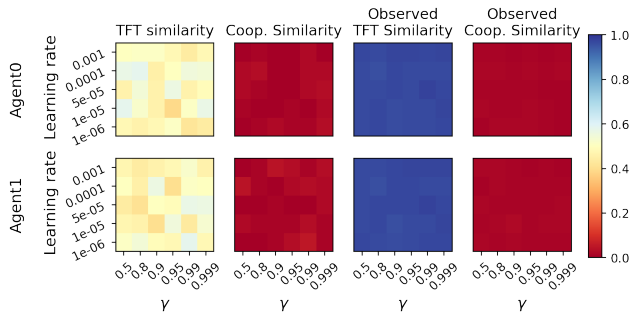
There were more interesting results for the PPO agents with a window-length of 1 and the DQN agents with a window-length of 5 (Figures 5b and 5c) . These agents often learned to always defect, but within these hyperparameter sweeps there are a few small "oases" of cooeprative behaviour, but these are few and far between. Here the PPO agents with a learning rate of 1×10^{-6} and a discount rate of 0.9 were able to learn to cooperate when playing with each other. However, right next to this in the hyperparameter space (learning rate of 1×10^{-5} and a discount rate of 0.9), one agent has learned tit-for-tat while the other has learned to defect; this results in the agents both generally defecting when playing with each other. The DQN agents with a state length of 5 here have not learned to cooperate quite as robustly, but there is a small region in the hyperparameter space around learning rate of 5×10^{-5} and discount rate of 0.8 where the agents seems generally less likely to always defect and even develop moderately cooperative strategies.

Pretraining

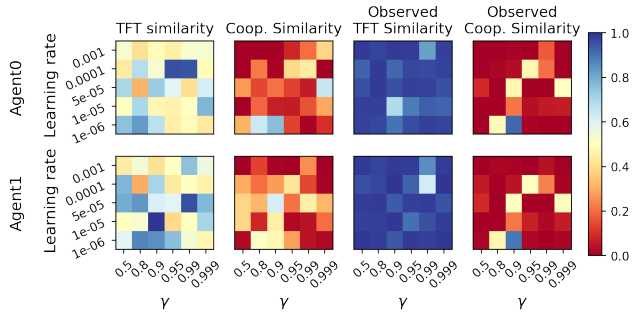
When naively training RL agents against each other in IPDs the default option seems to be for the agents to fall into defect-defect equilibria (especially for DQN agents), even if there are certain small oases in the hyperparameter-space where they do learn to cooperate. A potential strategy to avoid this defect-defect behaviour would be to pretrain agents against fixed policies before training them with each other. If the agents have learned a strategy similar to tit-for-tat from the pretraining, then this learned behaviour may be good for playing against another learning RL agent. If the agent receives a high reward for continuing to play its learned strategy (which is close to tit-for-tat), then this behaviour will be reinforced and hopefully stable. We can then also see if the behaviour of generally cooperative agents will be retained if they start playing against another cooperative agent.

Pairs of RL agents were initially trained with a fixed policy in order to instil a certain behaviour; training with tit-for-tat-then-defect to make them learn a policy correlated with tit-for-tat, and training with standard tit-for-tat to make them learn a cooperative policy. Then these agents were trained with each other to see the effect that this initial training had on the policies they developed. The similarity to tit-for-tat and cooperativity (evaluated with the previously used sampling approach) were calculated throughout training with the

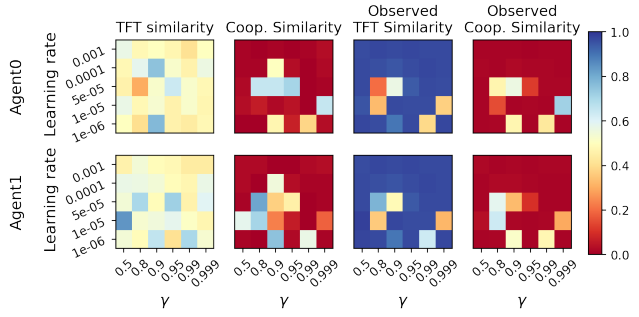
(a) DQN, state length: 1



(b) PPO, state length: 1



(c) DQN, state length: 5



(d) PPO, state length: 5

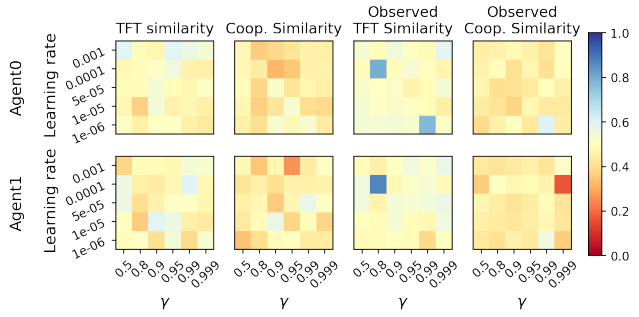


Figure 5: The tit-for-tat similarity and the cooperation similarity of DQN and PPO agents with state lengths of 1 and 5. The IPDs here are all of random length. DQN with a state length of 1, and PPO with a state length of 5 lead to similar results as previously seen. PPO with a state length of 1, and DQN with a state length of 5 are able to learn strategies which are correlated with tit-for-tat and cooperative.

fixed policies and throughout training with each other (Figure 6).

For the DQN agents (Figure 6a), the training to be cooperative is extremely non-robust, both agents learn to cooperate during the pretraining with tit-for-tat, and then extremely rapidly both learn to always defect. Pretraining with tit-for-tat-then-defect appears to somewhat help with sustaining cooperation; both agents develop policies which reach around 0.8 tit-for-tat similarity during pretraining, and although this decreases when they train with each other they do not immediately learn to always defect. However, these agents still defect more often than cooperate and the training with each other is quite unstable.

The PPO agents pretrained with tit-for-tat do not learn to always cooperate and also do not rapidly collapse into always-defect, as is consistent with Figures 3b and 4b. Throughout training these agents remain marginally more likely to cooperate than defect. When training PPO agents with the tit-for-tat-then-defect policy, these agents learn strategies slightly more similar to tit-for-tat than the DQN agents learned (Figure 6a). The tit-for-tat similarity rapidly decays as the agents begin to play against each other, but rather than falling into defecting these PPO agents develop cooperative polices. The PPO agents here learn strategies which cooperate at 0.9 cooperation similarity, although these learned strategies are not similar to tit-for-tat.

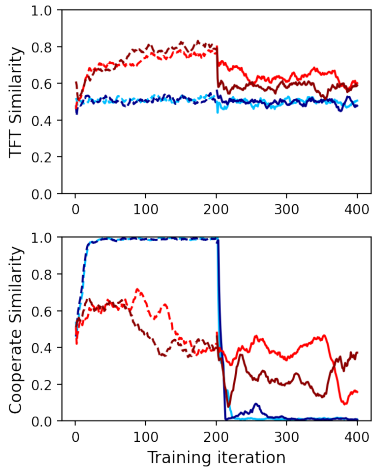
Conclusion and Future Work

We have presented an empirical analysis emphasising the difficulty in achieving cooperation with reinforcement learning agents in the Iterated Prisoner's Dilemma. We have highlighted an approach to reduce agent vulnerability when learning against a fixed policy in the single-agent setting by using more adversarial opponents. We have further tried to identify which properties can help improve agents' behaviour to be more cooperative and employ strategies like tit-for-tat. By far the most effective approach has been to reduce the history of interaction available to the agent to the previous game, however hyperparameters still play a vital role in achieving this behaviour. In general we have found very sharp changes in behaviour arising from small alterations to hyperparameters, finding only a few oases of cooperation in an otherwise vast desert of defection.

There are enumerable ways to continue this work, we have simply shown that RL agents can struggle to achieve the types of strategies advocated by game-theory. Solutions to this could take the form of more robust training or better incentives to prevent uncooperative behaviour. These would be of large benefit to the AI and RL safety communities, and would be tentative first steps towards the design and implementation of AI systems that can be safely deployed in the real world where similar (though often far more complex) problems arise.

The code to reproduce the research in this paper can be found here: https://github.com/peterbarnettnz/rl-ipd

(a) DQN



(b) PPO

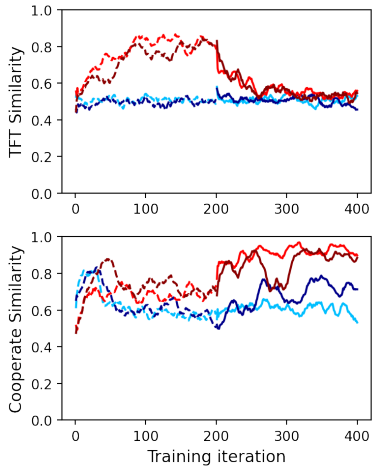


Figure 6: The tit-for-tat similarity and the cooperation similarity for DQN and PPO RL agents. Dashed lines indicate the agents are pretraining against a fixed policy, solid lines indicate the same agents are now training against an agent trained with the same fixed policy. The red lines are for the agents pretrained with tit-for-tat-then-defect, the blue lines are for agents trained with tit-for-tat (each shade of a colour represents a different agent). The DQN agents have a learning rate of 1×10^{-3} and a discount rate of 0.8, the PPO agents have a learning rate of 1×10^{-4} and a discount rate of 0.96. The agents were trained with the fixed policies for 200 training iterations, and a further 200 training iterations with the other RL agent.

References

Aumann, R. J. 1959. Acceptable Points in General Cooperative n-Person Games. In *Contributions to the Theory of Games (AM-40), Volume IV*, 287–324. Princeton University Press.

Axelrod, R. 1980. Effective Choice in the Prisoner's Dilemma. *Journal of Conflict Resolution*, 24(1): 3–25.

Benoit, J.-P.; and Krishna, V. 1985. Finitely Repeated Games. *Econometrica*, 53(4): 905–922.

Harper, M.; Knight, V.; Jones, M.; Koutsovoulos, G.; Glynatsi, N. E.; and Campbell, O. 2017. Reinforcement learning produces dominant strategies for the Iterated Prisoner's Dilemma. *PLOS ONE*, 12(12): 1–33.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-term Memory. *Neural computation*, 9: 1735–80.

Liang, E.; Liaw, R.; Nishihara, R.; Moritz, P.; Fox, R.; Goldberg, K.; Gonzalez, J. E.; Jordan, M. I.; and Stoica, I. 2018. RLlib: Abstractions for Distributed Reinforcement Learning. In *International Conference on Machine Learning (ICML)*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Rapoport, A.; Chammah, A. M.; and Orwant, C. J. 1965. *Prisoner's dilemma: A study in conflict and cooperation*, volume 165. University of Michigan press.

Sandholm, T. W.; and Crites, R. H. 1996. Multiagent reinforcement learning in the Iterated Prisoner's Dilemma. *Biosystems*, 37(1): 147–166.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book. ISBN 0262039249.

Vassiliades, V.; and Christodoulou, C. 2010. Multiagent Reinforcement Learning in the Iterated Prisoner's Dilemma: Fast cooperation through evolved payoffs. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, 1–8.

Vassiliades, V.; Cleanthous, A.; and Christodoulou, C. 2009. Multiagent Reinforcement Learning with Spiking and Non-Spiking Agents in the Iterated Prisoner's Dilemma. In Alippi, C.; Polycarpou, M.; Panayiotou, C.; and Ellinas, G., eds., *Artificial Neural Networks – ICANN 2009*, 737–746. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-04274-4.

Wang, W.; Hao, J.; Wang, Y.; and Taylor, M. 2019. Achieving Cooperation through Deep Multiagent Reinforcement Learning in Sequential Prisoner's Dilemmas. In *Proceedings of the First International Conference on Distributed Artificial Intelligence*, DAI '19. New York, NY, USA: Association for Computing Machinery. ISBN 9781450376563.

Acknowledgements

This work was done as part of the Cambridge Existential Risks Initiative (CERI) Summer Research Fellowship 2021, we would like to thank CERI for their organisation and support. We would also like to thank the Stanford Existential Risks Initiative for additional funding and support.