FACULTY OF ENGINEERING SCIENCE

KU LEUVEN

Machine Learning Project
Exploring Multi-Agent Learning

Report 2
Opponent modelling & Deep reinforcement learning

Johan Edstedt (r0734133)
Ondřej Kobza (r0734132)
Jens-Andreas Rensa (r0733324)

July 2019

Abstract

In this paper, we experiment with opponent modelling and deep reinforcement learning in a multi-agent setting. In the first part of our paper, we experiment with different opponent modelling techniques in the classic zero-sum game; rock-paper-scissors. We implement Q-learning, fictitious play and a combined method, and find that the combined method yields best results. In the second part, we create agents which can play a complex game with many states, a partially observable temporally extended Markov game, "The Harvest Game". This is characterized by a large state-space and game theoretical problems such as the tragedy of the commons. In such a setting, we need more sophisticated algorithms. We therefore experiment with the deep-learning algorithm A3C. We are presenting five architectures based on A3C, which differ in complexity and the types of applied neural networks. We see than even the simple agents yield functional results in short training time.

Contents

1 Introduction

The real world is multi agent. The outcomes of the actions each of us takes is not only determined by the action itself, but also the actions of our peers. Training autonomous agents in such an environment poses a set of challenges. They not only have to take their own actions into account, but also the actions of other agents operating withing the same environment. Furthermore, this dynamic may lead to social dilemmas, where a local strategy for a agent may lead to an overall bad outcome for all agents as a whole, thus reducing long-term prospects. One example of this is the tragedies of the commons.[2]

On the first part of the report we explore how other actors in the environment may be modeled. The overall goal of is to get a better understanding how opponent modelling might improve learning in multi-agent environments. We implement three different methods for this. First, a Q-learning approach, where agents only are aware of their own actions and observed rewards. Second, a fictitious play [1] approach where agents only take opponents behaviour into account. Finally, combine the two into an agent who uses both the observed rewards from their actions, and the actions of the opponent into account. We do this within the context of a Rock-Paper-Scissors game.

In the second part of the report, we try to extend agents to a partially observable temporally extended Markov game, "The Harvest Game". The purpose of the Harvest game is for agents to collect apples which gives the agent reward. In this multi-agent game, each agent must take into account not only its own score (or rewards) but also which apples to eat (if all apples in a region are eaten, no more apples can grow there). If the long term cumulative reward is to be maximized it is clear that the agents must somehow cooperate to prevent overgrazing.

This idea is inspired by real life, e.g., human society is built upon cooperation. It has been observed that many humans exhibit inequity averse behaviour in social dilemmas, which might be a key piece to ensure cooperation. To try to capture this behaviour within the context of reinforcement learning models, we construct a utility function which takes inequity between agents into account. This function penalizes agents with a relatively large score difference compared to its peers, constructing a inequity averse model.[3] With that in mind we investigate and compare the learning of agents with different reward functions. We also investigate to which extent different agent architectures plays a roll in total payoff.

2 Opponent modelling

To observe how different methods of opponent modeling performs, we first a need a context to test these models in. We utilize the classic Rock-Paper-Scissors game. The game is a classic zero-sum game where the reward given to an agent fully depends on the action of the other agent. This game can be characterized by the pay-off matrix presented in table 1. In this game, it can be observed that all states are Pareto optimal, due to the nature of zero sum games. Furthermore, only the point where rock, paper and scissors are played with equal probability is a nash-equilibrium.

	Rock:	Paper	Scissors:
Rock:	0	-1	1
Paper:	1	0	-1
Scissors:	-1	1	0

Table 1: Pay-off matrix for the Rock-Paper-Scissors game

2.1 Q-learning agents

In our first approach, Q-learning, the agent is not explicitly modelled. Instead, the agent tries to directly learn the expected reward of the actions which it can perform, by observing the received reward. The agent will pick the action with the highest perceived rewards, and will break ties randomly. Although the opponent is not explicitly modelled, the opponents strategy will propagate to the agent through the rewards. The main problem with this strategy is inability to directly observe changes in the opponents strategy if one simply selects the highest Q-value. To combat this problem, we instead choose a move with the probability given by the boltzmann distribution over the Q-values.

In this setting, the agents will adapt to each other, and reach a strategy similar to a nash equilibrium. However, each time the agents learn form the opponent, a small deviation will be made from this strategy. This is shown in the learning dynamics of figure 1, where the frequency of moves shows the long term strategy of the agent, and the learning evolutionary dynamics shows how the short-term strategy changes.

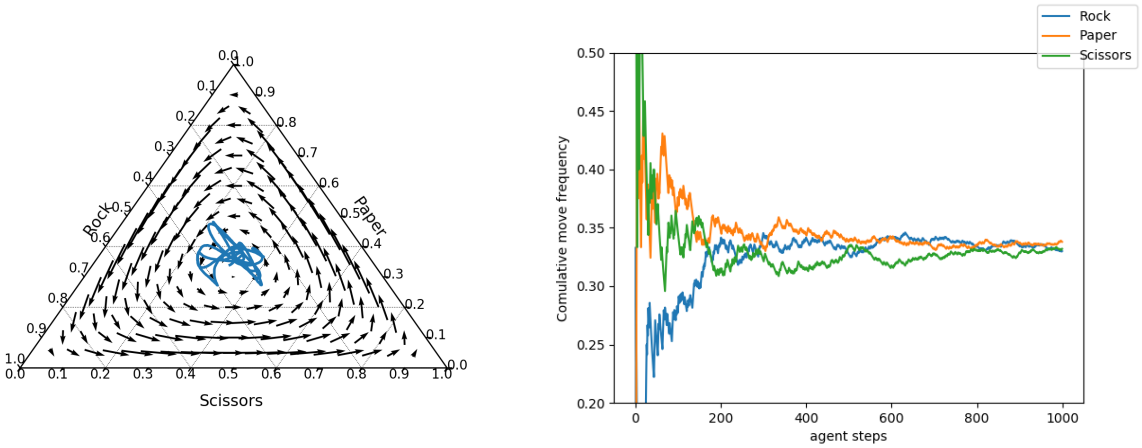

Fig. 1. Q-Learning: [A] Frequency of played moves over time, [B] learning rate of 0.01, [C] learning rate of 0.1. We observe that the agents long term strategy converge to the nash equilibrium, and small deviations from this equilibrium is found on the short term

2.2 Fictitious play

Second, we implemented a fictitious play algorithm. In this approach the agent will model the opponent directly though it's behaviour. It will observe the actions made by each opponent, and store the counts of each move. Based on this, it will calculate probability of an opponent making a move, purely based on the previously observed moves. With this information, as well as the given reward function, the agent will calculate the potential reward from each possible action. It will then choose an action based on the following formula:

$$\arg\max_{action}(OpponentActions \cdot Payoff Matrix)$$

Where the actions are integer values 0,1,2 corresponding to Rock, Paper, Scissors. OpponentActions is a list of length 3, each field in the list corresponds to how many times a particular action was played by the opponent. The learning dynamics of such an agent can be again visualized by learning traces and dynamics trajectories as can be seen in figure 2-A. We note that the short term strategy of the agent is diverging. As the move-counts which the agent stores increases, so will the number of moves

required to change the agents strategy. This results in an increasingly longer period before strategy changes. This increase can be observed in figure 2-B.

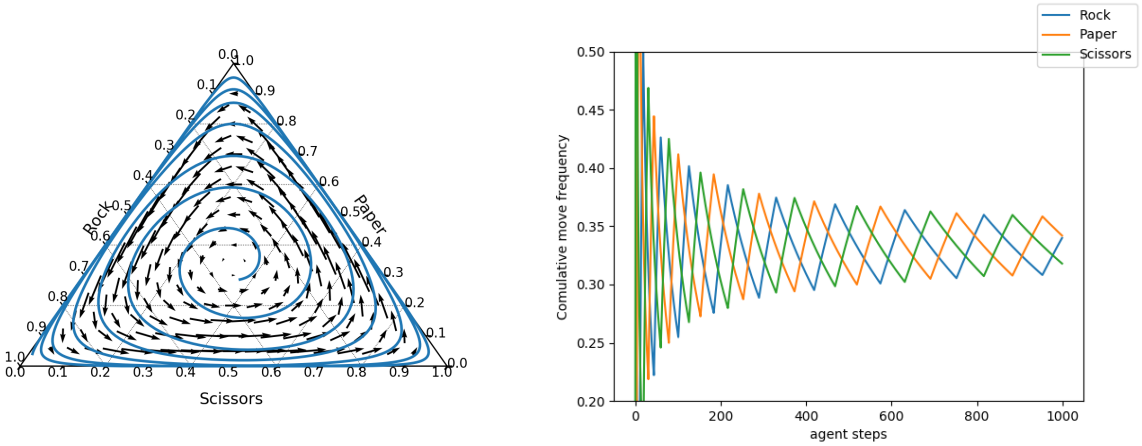

Fig. 2. Fictitious play: [A]Replicator and evolutionary dynamics for two agents in Rock-Paper-Scissors game. [B] The cumulative move frequency over time

2.3 Combined agent

Finally, we combined the two learning approaches. In this setting, the agent will utilize both methods from Q-learning and Ficticious play. It will first of all store all the moves made by the opponent, and use this to calculate the probability of the opponenent playing a given move. Furthermore, it will use the observed reward to maintain an approximation of the rewards from each move. In contract to normal Q-learning, the agent will try to learn a mapping from both its own moves and the opponents move to the reward. The agent will then combine this information by the use of the following formula:

$$probability\ of\ move = softmax(OpponentActionsFrequencies \cdot Qvalues)$$

We have observed, that after a few thousands of iterations, the Q-values converge to values in the pay-off matrix converges to the reward function. Thus, the behaviour should be similar to the standard fictitious play agent. However, we observe from figure3 that the agent converges to the nash equilibrium in both the short and long term, and is more stable than the Q-learning approach. This change can be attributed to the main change between this and the fictitious play model: Choosing the moves based on the soft-max probabilities.

2.4 Conclusions

We have shown that basic tabular Q-learning without opponent modelling can converge to the Nash-equilibrium, however the convergence is unstable and slow. While the opponent modelling in fictitious play is powerful it produces very exploitable agents, because of the assumption of the opponent using a stationary strategy mix. We have also shown that combining Q-learning with opponent modelling, using Boltzmann probabilities to choose actions improves convergence and causes the agents to converge quickly to a Nash equilibrium, and remain fairly stable in this.

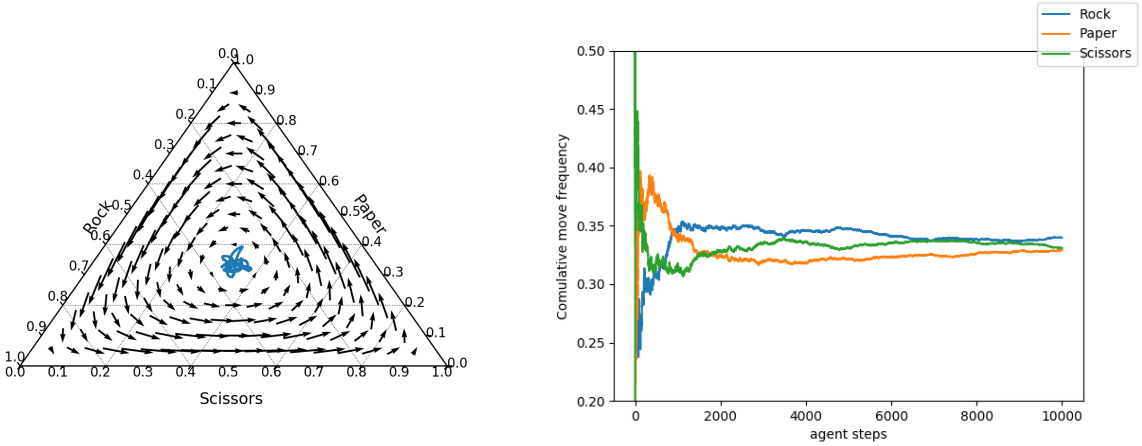

Fig. 3. Combined agents: [A] Replicator and evolutionary dynamics for two agents in Rock-Paper-Scissors game. [B] The cumulative move frequency over time

3 The Apples game

3.1 Problem statement

The game presented several challenges. Since the game presents both incomplete information as well as temporal dependencies as well as a much larger state space, agents needed to be more sophisticated. Our goal was to investigate how agents can learn to cooperate and to perform desired behaviour in this kind of environment.

3.2 Encoding the local state-space

The Apples game is an incomplete information game where agents can only see a 15x15 block of information around them. We made use of this fact by encoding the positions of nearby apples to a 15x15 matrix with the agent position implicitly centered. Nearby snakes were in the same way encoded into another separate matrix. Since different actions should be performed depending on the orientation of the agent, the matrix was rotated to the agents subjective point of view. With $2^{15+15+2} = 2^{32}$ different states a tabular method seemed infeasible. Our approach to solve this issue will be discussed in the following section.

3.3 Deep Reinforcement Learning

3.3.1 Models

For environments with very large state-spaces, tabular methods are usually difficult to get to work, and requires clever tricks to reduce state-space dimensionality. We decided to instead use a deep learning approach with neural networks using the A3C algorithm [4]. A3C agents utilize two networks, the policy network and the critic network. One actor(policy) network that for a given state gives a probability distribution over actions, while the value(critic) network approximates the expected value of being in a given state. We investigated five network architectures (each, actor and critic have the same architecture, [note that this does not necessarily mean, that the networks share weights in all of the models], except the last layer: the number of neurons and activation function is different - for an actor the last layer is a fully-connected softmax layer encoding a probability distribution over the

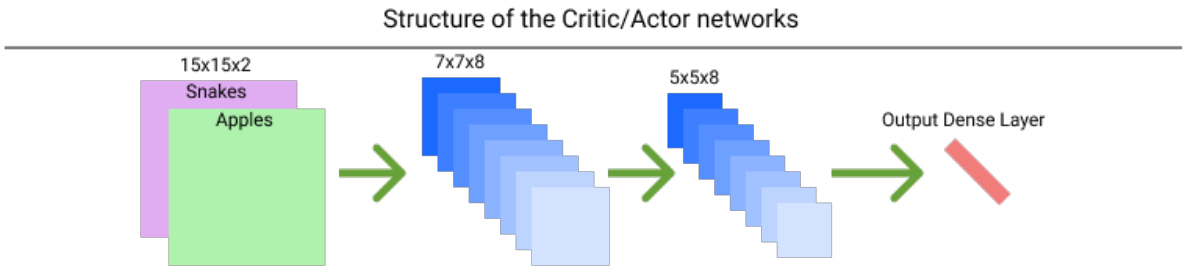

Fig. 4. Structure of the actor and critic networks in the final model

action space i.e. the policy, whereas for the critic, there is only one neuron with linear activation function):

1. A CNN architecture,
2. A frame-stacked CNN,
3. A single output layer,
4. An LSTM,
5. Combined CNN & LSTM.

From empirical testing we found that, while the LSTM provides a theoretical advantage in being able to capture temporal information, the increased complexity of the model slowed down training by about 100x. This made training the model incompatible with our timeconstraints and hence an less computationally heavy model was preferred. It was found empirically that the CNN model performed the best, the structure of the model can be seen in figure 4. The convolutional blocks consisted of a 3x3 kernel convolutional layer + relu activation layer + batch normalization layer.

3.3.2 Exploration/Exploitation

The agents used the A3C algorithm, which is an on-policy algorithm. However, the actions were randomly selected from the action distribution encoded by the network, hence the network could explore actions which it did not think were optimal in a given state. On top of that a type of regularization was used where high entropy policies were rewarded, i.e. policies with low certainty. A problem with this regularization is that it introduces a new hyperparameter, the "magnitude" of the entropy in the loss function, which needs to be tuned. We found empirically that 0.01 was a reasonable value, by observing when the agents would collapse into random behaviour.

3.4 Utility functions

We investigated three different reward functions for the agents. They are listed below.

3.4.1 Raw individual reward

The raw individual reward is simply $r_i^t = score_i^t - score_i^{t-1}$ where $i = 1...N$ represents an agent. This reward function incentivizes greedy behaviour of the agents, since no regard is taken to the other players of the game.

3.4.2 Group reward

The group reward is defined by $r_i^t = \frac{1}{N}\sum_j score_j^t - score_j^{t-1}$, here the joint reward is given to all agents. One problem with this reward function is the "spurious" reward problem, where agents get rewarded for actions which may have no connection to the reward. Another issue is the "lazy agent" problem. However, since we employed self-play to train the models this is less relevant.

3.4.3 Inequity averse utility function

The inequity averse utility function is an agent's subjective utility function, which is determined by the following formulas (according to [3] utility function for inequity aversion in sequential dilemmas):

$$u_i(s_i^t, a_i^t) = r_i(s_i^t, a_i^t) - \frac{\alpha_i}{N-1}\sum_{j \neq i} \max(e_j^t(s_j^t, a_j^t) - e_i^t(s_i^t, a_i^t), 0) - \frac{\beta_i}{N-1}\sum_{j \neq i} \max(e_i^t(s_i^t, a_i^t) - e_j^t(s_j^t, a_j^t), 0)$$

Where $j = 1...N, \quad i = 1...N$ are particular agents, N..number of agents, α is a weight determining disadvantage aversion, β is a weight determining advantage aversion, a is an action, s is a state and e is a temporal smoothed reward defined as follows:

$$e_j^t(s_j^t, a_j^t) = \gamma\lambda e_j^{t-1}(s_j^{t-1}, a_j^{t-1}) + r_j^t(s_j^t, a_j^t)$$

Where γ is discount factor and λ is a hyperparameter.

3.4.4 Joint reward utility function

We have observed, that the hyperparameters α and β should be set to similar value, because if the α is much higher than β, the agents tend to play only "fire" action - they are avoiding disadvantages too much, whereas if α is much lower than β, agents are generally ending up with lower scores than when the two parameters have similar values. Also, we have observed, that these values should be rather small than big.

3.4.5 Game-theoretic perspective

It is difficult to say which states are pareto optimal, but states where snakes avoid apples as much as possible are probably optimal, since taking more apples would reduce rewards for other agents.

We can also find nash-equilibria, e.g. a situation where every agent overgrazes maximally is such a state. However, since the agents are equipped with tools for "punishing" greedy agents, the hope is that these kinds of equilibria are unstable and the desired behaviour will emerge through proper design of the reward function.

In theory it is possible for agents using any of these utility functions to exhibit desired behaviour. However recent research (Hughes et al. 2018)[3] suggests that inequity aversion can "steer" agents away from selfish behaviour.

3.4.6 Predicted behaviour of the agents

It has been observed (Perolat et al. 2017)[5], that different phases of learning can be seen in social dilemmas. Roughly speaking a naive phase where agents explore their environment should first be seen. This phase should give high collective reward since the agents are too "dumb" to exploit their environment to their advantage, but smart enough not to shoot since it loses reward. This is followed by an exploitation phase, the tragedy, where agents deplete the resources and get lower rewards. At some point agents start punishing other agents who exploit the resources which turns into a third phase, "maturity", where agents harvest responsibly and converge to some sort of sustainable behaviour.

3.5 Experiments

For each utility function we trained our models running 4 games of 8 agents simultaneously, i.e. 32 agents in parallel on a hyperthreaded CPU. The agents were trained using self play, i.e. all of the agents shared the same networks and updated the weights asynchronously, see [4] for details. The game was run with standard rates of apple spawning, and with 5 starting apples. For the inequity averse model we set $\alpha, \beta = 0.02, 0.001$, with a time-smoothing of $\gamma = 0.9$ and $\lambda = 1$. The episodes for training was usually set to between 50-250 timesteps. An agent specific optimizer was used, i.e. no shared update statistics was used during training of the agents, which should in theory perform slightly worse than an asynchronus version.

3.6 Results

3.6.1 Observed Behaviours

Maybe not surprisingly the difficulty of training went up the more agents were on the board simultaneously. For 1-3 agents, we observed that all reward functions worked well, with agents converging relatively quickly to desired behaviour[1]. In our extended experiments we used 8 agents per game, which proved a much more difficult environment as might be expected, for example indivdual reward which worked well for fewer agents performed poorly[2] and exhibited unwanted behavior. We could not find learning phases that were as clearly defined as in [5]. In some cases agents seemed to cycle through certain kinds of strategies while never converging, it is however possible that we did not train the agents for a sufficient amount of to exhibit the typical learning phases.

Agent behaviour would sometimes collapse into a single strategy. Some examples are

- Only shooting for first part of game (i.e. when there were few apples)
- Going in circles
- Only going straight
- Going only left/right
- Completely random behaviour, i.e. no convergence

Only shooting was observed for agents which were strongly disadvantage averse. Another reason may be that agents with 0 score could not get a lower score by shooting or getting shot, which meant that there was no punishment for shooting early on. Going in circles was usually observed when a lot of

[1] https://www.youtube.com/watch?v=ADToViwPcuc
[2] https://www.youtube.com/watch?v=pjX7agW2IOA

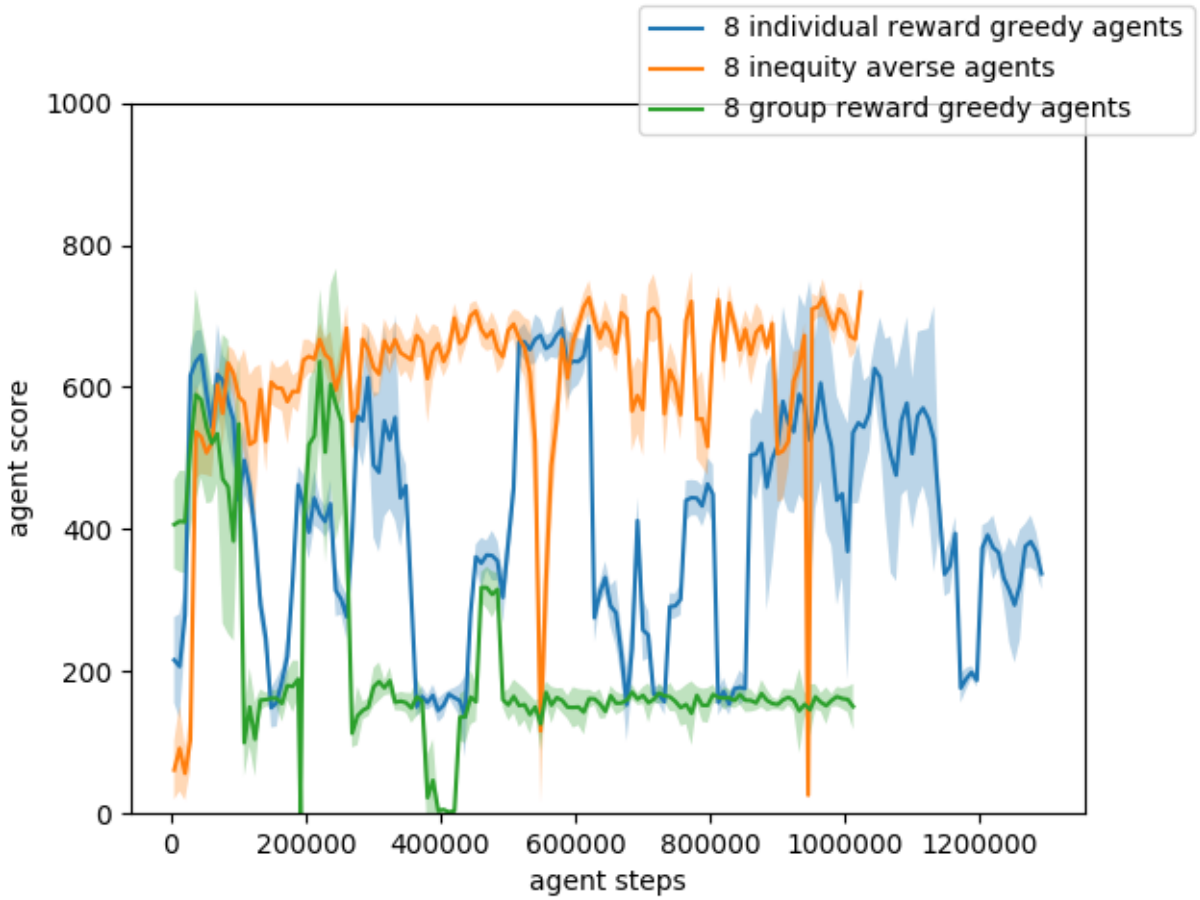

Fig. 5. Learning curves for different utility functions, observe that these plots are based on previous versions of the agent architecture

agents were shooting, likely to avoid getting shot. Only going straight was also observed, or going only left/right. Completely random behaviour was also observed in cases where the regularization was too harsh.

These kind of behaviours were very common for the naive reward functions, and is reflected in figure 5. Here it can be seen that the total reward goes through cycles, especially for the individual reward. When using a "gentle" inequity aversion, with the parameters described above, we found that this behaviour was lessened and reward was generally more stable and higher. However, when α was set too high we observed that behaviour would often collapse into only shooting. The group reward agents fared the worse, and learning was generally not possible. This is probably caused by the extremely noisy reward, where all behaviours get equally rewarded.

3.6.2 Best Results

Our best results was had with the inequity averse utility function with a low value of $\alpha, \beta = 0.02, 0.001$. Typical results gave a mean score of 650 in rounds with standard apple spawn rate and 5 starting apples with 8 agents. In this youtube clip, typical behavior of the agents can be observed. Agents start out by avoiding apples and sometimes shooting eachother. Once enough apples are around the agents start consuming.

3.6.3 Inequity Averse Model Parameters

We did not find good results with previously reported[3] values of α, β. Setting $\alpha = 5$ caused our agents to collapse into only firing. This behaviour may have been caused by the score being capped at 0, since repeatedly firing at 0 score gave no punishment to the agents. We instead found that setting low values of $\alpha\beta$ gave the benefits of individual rewards, i.e. less punishment, while retaining training stability which the selfish reward did not exhibit.

One encouraging result is that, as can be seen in figure 5, the inequity averse model does not fall into the trap of "tragedy". Instead we see a relatively smooth monotone improvement during training, which suggests that the agents never devolve into exploitative behaviours.

3.7 Future work

Agents using a long term memory would be an interesting extension since this would enable agents to remember actions of other players which could lead to more complex behaviour. Further it would be interesting to see the effect of using other players received rewards as input to the model, e.g. a model might learn inequity aversion by itself. Another aspect would be mixing different kinds of agents, which has in fact already been done see e.g. [3], to investigate further how interactions between these kinds of agents unfold.

3.8 Conclusions

We have shown that inequity aversion seems to improve convergence to desired emergent behaviour of agents in the Apples game by incentivizing cooperation. We have also shown that some other naive reward functions, while they work well for fewer agents, do not converge well when many agents are present, and tends to collapse into unwanted strategies.

References

[1] George W. Brown. Iterative solution of games by fictitious play. In T. C. Koopmans, editor, *Activity Analysis of Production and Allocation*. Wiley, New York, 1951.

[2] Garrett Hardin. The tragedy of the commons, 1999.

[3] Edward Hughes, Joel Z. Leibo, Matthew G. Philips, Karl Tuyls, Edgar A. Duéñez-Guzmán, Antonio García Castañeda, Iain Dunning, Tina Zhu, Kevin R. McKee, Raphael Koster, Heather Roff, and Thore Graepel. Inequity aversion resolves intertemporal social dilemmas. *CoRR*, abs/1803.08884, 2018.

[4] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783, 2016.

[5] Julien Pérolat, Joel Z. Leibo, Vinícius Flores Zambaldi, Charles Beattie, Karl Tuyls, and Thore Graepel. A multi-agent reinforcement learning model of common-pool resource appropriation. *CoRR*, abs/1707.06600, 2017.