Machine Learning Project 2018-2019: Report 1

Andreas Stieglitz, Hendrik Serruys, Emile Breyne

March 2019

1 Literature

[1]: Chapter 3 introduces non-cooperative game theory. More specifically, the normal-form game is defined, as well as game analysis with respect to Pareto Optimality and Nash equilibrium. Chapter 7 deals with reinforcement learning as well as evolutionary learning (replicator dynamics for symmetric games). Both techniques are essential for the first part of this project. However, what is missing here is an extension of the single-population replicator equations to multi-populations (non-symmetric games). Chapter 7 also discusses fictitious play, which is important for opponent modelling (Task 2).

[2]: Section 3.2.2 discusses Replicator Dynamics. Both single-population and multi-population dynamics are discussed separately. The implementation of the latter in this project was based on this thesis.

[3]: Section 5 and 6 discuss the implementation of Probabilistic Learning Automata (PLA) for simple two player games (like the Prisoner's Dilemma and the Matching Pennies game). We have used this source to implement PLA as a basic RL technique.

[4]: Chapter 14 of this course book provides a first discussion on RL. It provides an overview of algorithms such as Value Iteration, Q-learning, SARSA and TD(λ). Furthermore, a suggestion is made of how to model the switch from exploration to exploitation, which allows the agent to gradually move from randomly selecting actions towards selecting actions with a high Q-value. We have implemented this idea in our Q-learning script.

2 Task 1: warming up 'Learning & Dynamics'

2.1 Independent learning in benchmark matrix games

Two basic reinforcement learning algorithms are implemented: Q-learning and Probabilistic Learning Automata (PLA). For **Q-learning**, the initial policy is chosen uniformly at random and all data structures are reset at the beginning of every new game. In **PLA**, every agent is an automaton and we calculate per player the probability of choosing a certain action. At the beginning of a new game, the probabilities are reset uniformly at random, summing to one per player. Every game consists of 1000 episodes. After each episode, the agents may adapt their strategy. In total, 10000 games are played and all results are averaged over these games.

2.1.1 The Prisoner's Dilemma (PD) game

In this social dilemma game, two players are given the following actions: (C)ollaborate or (D)efect. The following strategy profiles are **Pareto optimal**: (C, C), (C, D) and (D, C). The strategy profile (D, D) is not Pareto optimal as it is **Pareto dominated** by (C, C). However, strategy profile (D, D) is the only (strict) Nash equilibrium. With Q-learning, close to 100% of the games played finished with the strict Nash equilibrium strategy profile (D, D). With PLA, the average probability found per player to defect is near 100%. It is clear that the strict Nash equilibrium prevails once again. When combining the 2 learning algorithms (one player using Q-learning and the other one using PLA at the same time), the same result was achieved.

2.1.2 The Matching Pennies game

In this zero-sum game, two players (Even and Odd) pick either Heads (H) or Tails (T). All strategy profiles are Pareto optimal (characteristic to a zero-sum game). The unique Nash equilibrium of this game is a mixed strategy where both players pick H or T with equal probability. With Q-learning, all four strategy profiles are on average used with equal probability. For PLA, the average probability per player to pick Heads is about 50%. This means that for both basic reinforcement learning techniques, the mixed Nash equilibrium strategy is reached. When combining the 2 learning algorithms (one player using Q-learning and the other one using PLA at the same time), the same result was achieved.

2.2 Dynamics of learning in benchmark matrix games

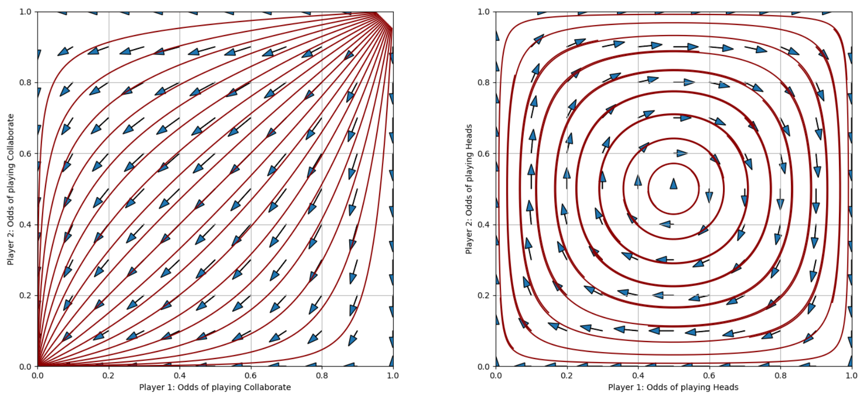
The replicator dynamics are shown in figure 1. These were generated based on handpicked initial estimates to obtain equidistant dynamics.

2.2.1 Prisoner's Dilemma

As expected, the replicator dynamics converge to the strict Nash equilibrium (D, D). The bending of the trajectories can be understood as follows. When both players are initially collaborating, one player will start to defect to increase his reward. Soon, the other will learn that it can only gain a reward if he too starts defecting.

2.2.2 Matching Pennies

The mixed strategy 50% heads/50% tails is the Nash equilibrium, but the replicator dynamics do not converge to it. Instead, they follow circles around the center. When the step size is too large, the result will spiral outward, since no point on the tangent of a circle lies within the circle. For an infinitesimally small step, the resulting strategy stays a certain distance from the mixed Nash Equilibrium strategy, but does not converge towards it. The Nash Equilibrium is therefore not an evolutionary stable strategy.



(a) Prisoner's Dilemma (b) Matching Pennies

Figure 1: Trajectory and directional field plots

2.3 Conclusion

For the social Prisoner's Dilemma game, neither basic single-agent RL algorithms nor replicator dynamics are able to produce the Pareto optimal strategy. In the zero-sum Matching Pennies game, the Nash Equilibrium is in fact a stable strategy profile for RL algorithms. However, it is not an evolutionary stable strategy. Both these experiments suggest that there is room for cooperating multi-agent algorithms which would provide an agent with more information about his environment and other opponent agents who affect him and the environment. This could ultimately increase all agents' rewards in games like PD. A further experiment might be to combine RL algorithms and replicator dynamics such that different agent populations compete against each other. This could potentially be used to measure the performance of different algorithms in a multi-agent setting.

3 Tasks 2 & 3: a brief plan of action

We have chosen to use Python for the first task as it has many libraries available for mathematics, machine learning, visualization... and is strongly supported by an active community. We have agreed to use Python for the second and third task as well.

3.1 Task 2

The Q-learning method should be the same as implemented in task 1. Although more research will need to be done, we believe it should be possible to build an implementation for part two, the fictitious play method, based on e.g. [1] or [5]. We still have to figure out how to combine the 2 algorithms for the 3rd assignment.

3.2 Task 3

For now, it's too early to provide a concrete plan to tackle part three. Once part two is finished, we will have another team meeting to explore the Harvest Game and its code, to expand our knowledge on Deep Learning and to brainstorm on how to accurately model the game state with it. We expect that one of the most difficult steps of task 3 is to find an efficient way of generalizing the different state-action pairs, as there will be way more possible states than in the other games (one). We believe we should use an algorithm based on opponent modelling here. The challenge will be predicting the actions of other agents with enough accuracy. Adding the ability to punish devious agents sounds promising, although it will complicate the entire system.

References

[1] Yoav Shoham & Kevin Leyton-Brown (2009) *Multiagent Systems. Algorithmic, Game-Theoretic, and Logical Foundations*, Cambridge University Press

[2] Daan Bloembergen (2010) *Analyzing Reinforcement Learning Algorithms Using Evolutionary Game Theory*

[3] Ath. Kehagias (1994) *Probabilistic Learning Automata and the Prisoner's Dilemma*

[4] Hendrik Blockeel. Machine Learning and Inductive Inference (course notes). KU Leuven, 2017.

[5] 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.