Machine Learning Project 2019-2020

Multi-Agent Learning in Canonical Games, Kuhn and Leduc Poker (OpenSpiel)

Karl Tuyls, Wannes Meert

February 2020

Read this text completely. Deviations from the instructions may result in lower scores.

1 Introduction

We live in a multi-agent world and to be successful in that world, agents, and in particular, artificially intelligent agents, need to learn to take into account the agency of others. They will need to compete in market places, cooperate in teams, communicate with others, coordinate their plans, and negotiate outcomes. Examples include self-driving cars interacting in traffic, personal assistants acting on behalf of humans and negotiating with other agents, swarms of unmanned aerial vehicles, financial trading systems, and robotic teams.

This assignment covers some topics in multi-agent reinforcement learning (RL). We will assume elementary knowledge of single-agent reinforcement learning (a good introduction is available at `http://incompleteideas.net/book/the-book-2nd.html`).
When moving from single-agent RL to multi-agent RL, Game Theory plays an important role as it is a theory of interactive decision making. Throughout the assignment you will use some elementary game theoretic concepts [12] (see `http://www.masfoundations.org/mas.pdf`) in combination with multi-agent learning, which is non-stationary and reflects a moving target problem (for some references see [3, 18, 2]). We will be working in the OpenSpiel open-source software framework, see `https://github.com/deepmind/open_spiel`. OpenSpiel is a collection of environments, algorithms and game-theoretic analysis tools for research in general reinforcement learning and search/planning in games. OpenSpiel supports n-player zero-sum, cooperative and general-sum games [6].

In this assignment we start by tackling some canonical games from the 'pre-Deep Learning' period (for some Deep RL references see [7, 4]), after which we transition to more complex Poker games and delve into deep multi-agent reinforcement learning.

Note that we will be organizing an introductory session on 3 March (including Q&A) that will cover some of these game theoretic and multi-agent learning concepts. It is highly recommended to attend this.

2 Problem

In this assignment you will be using machine learning (reinforcement learning) and game theory techniques in the context of multi-agent settings. We will focus on zero-sum games and social dilemmas: zero-sum games correspond to situations in which the total sum of gains and losses of all players involved equal to zero; social dilemmas are known as games in which several agents participate and there is a tension between what is optimal to do at the individual level (in the

short term) and what is beneficial to do at the group-level (in the long term). Famous examples of the former include the *Rock-Paper-Scissors* game and *Poker*, and of the latter include the *Prisoner's Dilemma* game and the *Tragedy of the Commons*. In this assignment we will be particularly interested in:

- **Benchmark matrix games**: prisoner's dilemma, matching pennies game, battle of the sexes, and rock-paper-scissors.
- **Kuhn Poker** is a simple K-player card poker game by Harold E. Kuhn [5]. Each player starts with 2 chips, antes 1 chip to play, receives a face-down card from a deck of $K+1$ such that one card remains hidden, and either bets (raise or call) or folds until all players are in (contributed equally to the pot) or out (folded). Amongst those that are in, the player with the highest-ranked card wins the pot. In a 2-player game there are thus 3 cards.
- In **Leduc Poker** [14], players instead have limitless chips, one initial private card, and ante 1 chip to play. Bets are limited to 2 and 4 chips, respectively, in the first and second round, with two raises maximum in each round. A public card is revealed before the second round so at a showdown, the player whose card pairs the public card or has the highest-ranked card wins the pot.

Therefore the following approaches become very relevant for agents to deploy in order to deal with such situations:

1. **Reinforcement Learning**: Reinforcement Learning (RL) [1, 16, 17] can be used to learn a policy for playing games. Often, given the very large number of state-action combinations, standard model-free techniques such as Q-learning, with a table of state-action pairs will not work; instead some kind of generalization is needed (predicting the Q-value of unseen state-action pairs), often done using neural networks. There will be a need to design or learn features describing states and actions that correlate with the quality of state-action pairs.
2. **Game Theory**: Game Theory is a theory of interactive decision making, in which the players or agents involved take actions of which the success depends on the actions and preferences of the other players involved. The most elementary concepts include normal form games (an abstract model of a game) and Nash equilibrium. Intuitively, a Nash equilibrium is a strategy profile (a tuple in which each player plays one strategy) in which none of the players have an incentive to unilaterally deviate from their strategy given that the other players keep their strategy fixed. This means they cannot improve their payoff or reward by changing their strategy when the other players keep their strategy fixed.

For both approaches you can use implementations from OpenSpiel and make additional implementations of your own, i.e. in RL, you will need to represent states and state-action pairs. In simple games this will be stateless, for more complex settings you will need to use models that can generalize. To this end, you will have to think about features that can accurately represent the game state and that allow for learning models that generalize well. The features can be designed by an expert, or learned automatically. The Q or V function will be expressed in terms of these features. Learning this function can be done using almost any of the machine learning techniques you have seen in the machine learning course: k-nearest neighbors, random forests, neural networks, inductive logic programming, etc. In this assignment we invite you to study Deep Learning for this purpose and use e.g. techniques from the OpenSpiel framework (DQN, policy gradient etc) [10, 9, 13]. In Game Theory you will need to use/implement simple games and strategic tools such as Nash and replicator equations. For the final part of the assignment, you will investigate two more complex games, i.e. Kuhn and Leduc poker.

3 Approach

Your task is to investigate and implement multi-agent learning approaches in canonical games and subsequently in the provided Kuhn and Leduc poker games. Additionally, the goal is to study the behaviour of the learning algorithms in game theoretic terms in the canonical games: do the outcomes form a Nash Equilibrium? Is the equilibrium Pareto optimal? How does the behaviour evolve over time and are agents behaving in a fair manner?
The final goal is to implement first an agent that learns to play Kuhn poker, and subsequently an agent that learns to play Leduc poker. All software included in OpenSpiel can be used.

Extensions to Openspiel software are welcome and contribute to a positive evaluation (it is not allowed to share your machine learning approach or algorithm with others).
You will work on this project in a team of two or three students. Note that since a lot of code is available within the OpenSpiel framework we do expect students to show a good understanding of the techniques deployed, and we expect them to be able to conduct a knowledgeable conversation about the techniques deployed during the defence of the project.

4 Tasks

The assigment is divided in three tasks, which are described below.

4.1 Form Groups

Before Feb 21st, 23:59

Mail to **both** wannes.meert@cs.kuleuven.be and karl.tuyls@kuleuven.be whether you work on this project in a team of two or three and include the names of all team members (one email per team suffices).

4.2 Report 1

Before March 20th, 23:59

In a first phase of the project you are expected to concisely report about the first steps of the project that are outlined below. Mail a report (PDF, $\leq$4 pages) to the aforementioned email addresses.

Literature:

- Describe what **literature** you have read and what you have learned from it. This is not just an annotated bibliography, but a critical analysis of the existing work and how your project relates to it.[1]

Task 1: warming up 'Learning & Dynamics' in OpenSpiel

- **[Independent learning in benchmark matrix games]** In a first step the purpose is to implement/use one or more basic reinforcement learning algorithms (e.g. Q-learning, Learning Automata) and experiment with these in four benchmark matrix games that are multi-agent: (1) the 2-player prisoner's dilemma, (2) matching pennies games, (3) the battle of the sexes game, and (4) biased Rock-Paper-Scissors. Each of these games represent a category of game, i.e. social dilemma, zero-sum or coordination game. Here you need to investigate and report on whether the learning algorithms converge to Nash equilibria, and whether these are (Pareto) optimal? (note that both agents should be using an RL algorithm; if they use the same RL algorithm this is called self-play)

[1] http://www.writing.utoronto.ca/advice/specific-types-of-writing/literature-review

- **[Dynamics of learning in benchmark matrix games]** Start by implementing/using the basic form of replicator equations (selection mechanism only) and examine their directional field plots for the same games and compare these to the learning trajectories. Provide trajectory and directional field or phase plots to illustrate your work. Figure 1a illustrates an example phase plot of replicator dynamics in the matching pennies game, with the Nash equilibrium at $(0.5, 0.5)$. Now make your own implementation of 2-player lenient boltzmann Q-learning dynamics in OpenSpiel, and do the same analysis (tune the involved parameters and report about their values). You can find the dynamics of lenient boltzmann in [2].
- Draw some overall conclusions about both steps relating the observed learning and dynamics behaviours in the four benchmark games.

Tasks 2 & 3. Additionally, we also expect you to describe briefly how you plan to address tasks 2 and 3.

[*With task 1 you can earn 6 out of 20 points of your overall mark.*]

4.3 Report 2

Before May 15th, 23:59

Submit your report (PDF, ≤10 pages) and prototype as a tabular policy (both as 1st and 2nd player; you can use class PolicyFromCallable for this) to wannes.meert@cs.kuleuven.be using the Belnet Filesender.[2] You will get an automated notification from the Filesender service when we receive your submission. Your report should fulfill the following criteria:

- Clearly state the **problem statement**.
- Formulate your design choices as **research questions** and answer them.
- Write out the **conclusions** you draw from your experiments together with a scientifically supported motivation for these conclusions.
- Be concrete and precise about methods, formulas and numbers throughout the text. A scientific text should allow for **reproducibility**.
- Clearly **cite** all your sources.
- Report the total **time each of you spent** on the project, and how it was divided over the different tasks mentioned.

Task 2: Kuhn poker In a second phase we are going to explore the simple (turn-taking) 2-player Kuhn poker game. In 2-player Kuhn Poker each player is dealt a single private card from a deck of three cards (e.g. labelled J,Q,K). There are six possible ways the cards can be dealt to the two players, followed by some player betting actions (see the earlier description). This implies there are six possible initial states, where the first players takes their first action. At this point there are three information sets or states (indicated as dashed ovals in Figure 2). An information state is a set of histories where player i is to act, that cannot be distinguished by player i due to information not known by i (see Figure 2). In 2-player Kuhn at the start the first player cannot distinguish between J/Q and J/K (or Q/J and Q/K, or K/J and K/Q), because they only differ in the second player's card, which the first player does not see. In game theoretic terms, a player makes decisions at these information states. In poker, information states consist of the actions taken by all players so far, the public cards revealed so far, and the player's own private cards.

The goal is now for you to develop an agent learning to play Kuhn poker (this can both be tabular and using a Neural Net). You can do this by e.g. using a single policy network for both

[2] https://filesender.belnet.be

players, which takes as input the current information state of the game and whose output is a softmax distribution over the actions of the game. The state of the game is represented as a fixed-size vector encoding public information, private information, and the game history.

Evaluate your agent during learning using the **exploitability** and/or **NashConv** metrics; visualise and discuss your findings.

- Exploitability of a strategy of the other player(s) is expressed as how much a player could gain if he switched to a best response against the other player(s). A strategy profile has 0 exploitability if and only if it is an exact Nash equilibrium.
- NashConv measures the 'distance' of a joint policy to a Nash equilibrium (i.e., lower NashConv is better), see [11].

In a two-player, zero-sum game, NashConv reduces to the sum of exploitabilities [8]. Formal definitions explaining how to compute both metrics are available in the literature, see e.g. [8, 15, 11].

[*With task 2 you can earn 7 out of 20 points of your overall mark.*]

Task 3: Leduc poker As a final part of this project we would like you to tackle 2-player Leduc Poker. Leduc poker is a significantly larger game than Kuhn poker with two rounds and a 6-card deck in two suits, e.g. {JS,QS,KS,JH,QH,KH}. In the first round a single private card is dealt to each player. In the second round a single board card is revealed. There is a two chip bet maximum, with raise amounts of 2 and 4 in the first and second round, respectively.

You will need to develop an agent learning to play Leduc poker, making use of the code infrastructure provided in OpenSpiel (again the environment and all algorithms can be used for this purpose). As before, evaluate your agent using the exploitability and/or NashConv metrics.

Include all the code you used and have written to perform the experiments with the final report (explain how to use it). Running and experimenting your application is part of the evaluation. Make sure that you (only) use the information states of the game as input to your approach (NN, or tabular). Submit the mapping between the information states as strings and distributions over actions as a csv file, both for the agent as 1st and 2nd player, i.e. a tabular representation of your policies as in Kuhn poker.

For the final evaluation of your agent we will play a round robin (pairwise) tournament, in which each agent will play many games (pairwise) against all other agents. On these outcomes we will do a game-theoretic analysis to determine the winners.
[*With task 3 you can earn 7 out of 20 points of your overall mark.*]

4.4 Peer assessment

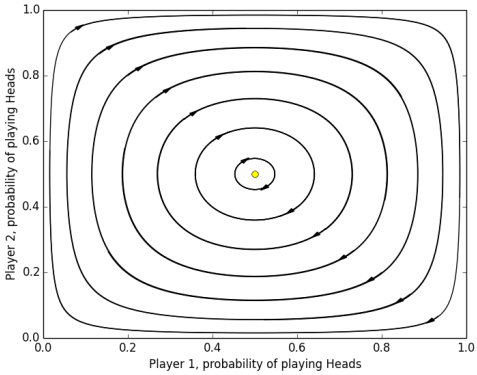
Before May 15th, 23:59, individually

Send by email a peer-assessment of your partner's efforts. This should be done on a scale from 0-4 where 0 means "My partner did not contribute", 2 means "I and my partner did about the same effort", and 4 means "My partner did all the work". Add a short motivation to clarify your score. If you are in a team of more than two people, send a score for each partner. This information is used only by the professor and his assistants and is not communicated further.

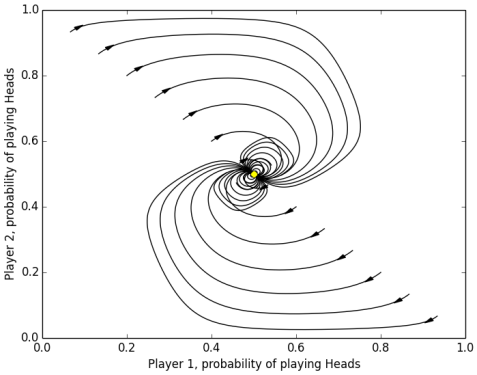
[*The final mark per task is determined by the combination of the reports and the oral discussion*]

References

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



(a) Evolutionary dynamics of the Matching Pennies game.



(b) Average Time Evolutionary dynamics of the Matching Pennies game.

[2] Daan Bloembergen, Karl Tuyls, Daniel Hennes, and Michael Kaisers. Evolutionary dynamics of multi-agent learning: A survey. *J. Artif. Intell. Res. (JAIR)*, 53:659–697, 2015.

[3] Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Trans. Systems, Man, and Cybernetics, Part C*, 38(2):156–172, 2008.

[4] Ian J. Goodfellow, Yoshua Bengio, and Aaron C. Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016.

[5] Harold W. Kuhn. Simplified two-person poker. In Harold W. Kuhn and Albert W. Tucker, editors, *Contributions to the Theory of Games*, pages 97–103. Princeton University Press, 1950.

[6] Marc Lanctot, Edward Lockhart, Jean-Baptiste Lespiau, Vinícius Flores Zambaldi, Satyaki Upadhyay, Julien Pérolat, Sriram Srinivasan, Finbarr Timbers, Karl Tuyls, Shayegan Omidshafiei, Daniel Hennes, Dustin Morrill, Paul Muller, Timo Ewalds, Ryan Faulkner, János Kramár, Bart De Vylder, Brennan Saeta, James Bradbury, David Ding, Sebastian Borgeaud, Matthew Lai, Julian Schrittwieser, Thomas Anthony, Edward Hughes, Ivo Danihelka, and Jonah Ryan-Davis. Openspiel: A framework for reinforcement learning in games. *ArXiv*, abs/1908.09453, 2019.

[7] Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[8] Edward Lockhart, Marc Lanctot, Julien Pérolat, Jean-Baptiste Lespiau, Dustin Morrill, Finbarr Timbers, and Karl Tuyls. Computing approximate equilibria in sequential adversarial games by exploitability descent. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 464–470, 2019.

[9] 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.

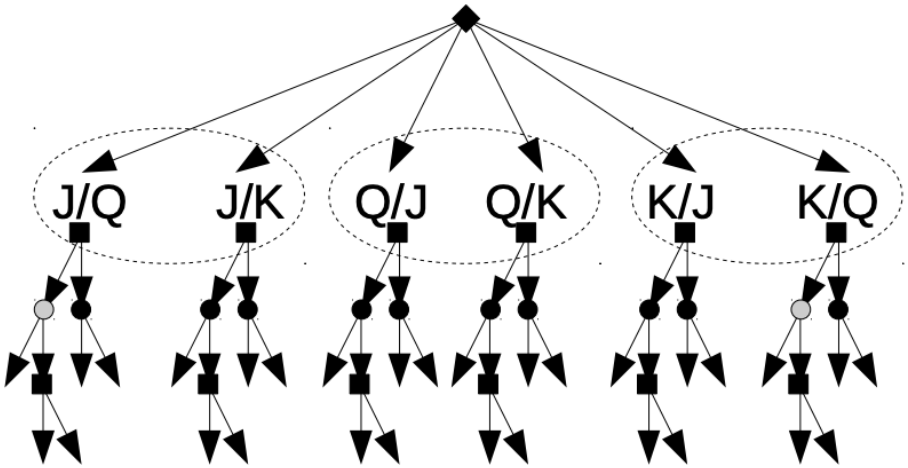


Figure 2: Information sets.

[10] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[11] Shayegan Omidshafiei, Daniel Hennes, Dustin Morrill, Rémi Munos, Julien Pérolat, Marc Lanctot, Audrunas Gruslys, Jean-Baptiste Lespiau, and Karl Tuyls. Neural replicator dynamics. *CoRR*, abs/1906.00190, 2019.

[12] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009.

[13] 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. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[14] Finnegan Southey, Michael Bowling, Bryce Larson, Carmelo Piccione, Neil Burch, Darse Billings, and D. Chris Rayner. Bayes? bluff: Opponent modelling in poker. In *UAI*, 2005.

[15] Sriram Srinivasan, Marc Lanctot, Vinícius Flores Zambaldi, Julien Pérolat, Karl Tuyls, Rémi Munos, and Michael Bowling. Actor-critic policy optimization in partially observable multiagent environments. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 3426–3439, 2018.

[16] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press, 2nd edition, 2017.

[17] Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Morgan & Claypool, 2010.

[18] Karl Tuyls and Gerhard Weiss. Multiagent learning: Basics, challenges, and prospects. *AI Magazine*, 33(3):41–52, 2012.