Machine Learning Project 2021-2022

Multi-Agent Learning in Canonical Games, Kuhn and betting-abstracted no-limit Texas Hold'em poker (OpenSpiel)

Karl Tuyls, Wannes Meert

February 2022

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 marketplaces, 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 topics in multi-agent reinforcement learning (RL). We will assume elementary knowledge of single-agent reinforcement learning (a good introduction when less familiar with RL is available at `http://incompleteideas.net/book/the-book-2nd.html` [20]).
When moving from single-agent RL to multi-agent RL (MAL), 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 (see `http://www.masfoundations.org/mas.pdf` [16] for some basic concepts when less familiar) in combination with multi-agent learning, which is non-stationary and reflects a moving target problem (for some references see [5, 22, 2]). In [22] a good intuitive introduction to MAL is provided, while [5] and [2] provide more technical descriptions of algorithms and replicator dynamics. 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 [9].

In this assignment we start by tackling some canonical games from the 'pre-Deep Learning' period (for some Deep RL references see [10, 6]), after which we transition to more complex Poker games and delve into deep multi-agent reinforcement learning.

Note that we will be organizing Q&A sessions that will cover some of these game theoretic and multi-agent learning concepts. It is recommended to attend these.

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 is 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**: Biased Rock-Paper-Scissors Game, Dispersion Game, Battle of the Sexes and the Subsidy Game.
- **Kuhn Poker** is a simple K-player card poker game by Harold E. Kuhn [8]. 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. You will play a 2-player game with 3 cards.
- **FCPA Poker** is a two-player (smaller) variant of Heads-up No-limit Texas hold'em poker, the most popular card game played worldwide and a standard domain for research on imperfect information games. As the size of the latter game can be intractably large [7], researchers have defined smaller games to drive the research forward [18]. One way to make a smaller version is using abstractions, and the abstraction we will use is the action abstraction, where the abstract game uses only a subset of the actions of the original game. Action abstractions are particularly interesting as it has been shown that one can achieve super-human performance in the full game even if the agent's actions are restricted to a small abstracted subset.[1] Indeed, all the state-of-the-art agents currently use very aggressive action abstraction and use only a few actions (e.g., fold, call, potbet) [4, 14, 23, 3].

The following approaches become very relevant for agents to deploy in order to deal with such games:

1. **Reinforcement Learning**: Reinforcement Learning (RL) [1, 20, 21] 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 or make implementations of your own, i.e. in RL, you will need to represent states and state-action pairs. Simple games will be stateless models, but for more complex settings you will need to use models that can generalize states. 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.

[1] The agent still has to be able to respond to any action of the opponent.

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 neighbours, 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.) [13, 12, 17]. 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 FCPA poker.

3 Approach

Your task is to investigate and implement multi-agent learning approaches in canonical games and subsequently in the provided Kuhn and FCPA 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 FCPA poker. All software included in OpenSpiel can be used.

Generic improvements to the 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 Deadlines

4.1 Form Groups — **Before March 1st, 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 Submit Draft of Report (optional) — **Before March 18th, 23:59**

If you submit a decent draft of tasks 1 and 2, (high-level) feedback will be provided individually.

4.3 Submit Report and Prototype — **Before May 19th, 23:59**

Submit your report (PDF, ≤10 pages) to Toledo. Your report should fulfill the following criteria:

- Confirm the directory on the departmental computers where your code and agent are stored (see Task 4).
- 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**.

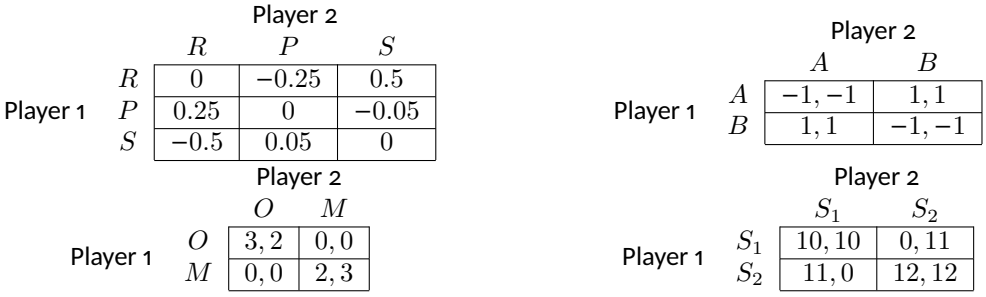


Figure 1: Biased Rock-Paper-Scissors Game, Dispersion Game, Battle of the Sexes and Subsidy Game

- 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.

4.4 Peer assessment

Before May 19th, 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.

4.5 Oral discussion

Week of May 23rd

Pick a slot on Toledo to discuss your report and answer questions.

5 Tasks

Your report should discuss the following tasks.

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

Task 1: Literature

Describe briefly 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.[2]

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

Task 2: 'Learning & Dynamics' in OpenSpiel

Use the payoff tables of Figure 1 for your work.

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

- **[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 the four benchmark matrix games in Figure 1. Each of these games represents a category of games, 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).
- **[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/phase plots to illustrate your work. Figure 3a illustrates an example phase plot of replicator dynamics in the matching pennies game, with the Nash equilibrium at $(0.5, 0.5)$ (not shown here but expected is the trajectory of your agent while training). 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.

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

Task 3: Kuhn poker

Kuhn Game Rules

You are now 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 player takes their first action. At this point there are three information sets or states (indicated as dashed ovals in Figure 4). 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 4). 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.

Goals

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 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 [15].

In a two-player, zero-sum game, NashConv reduces to the sum of exploitabilities [11]. Formal definitions explaining how to compute both metrics are available in the literature, see e.g. [11, 19, 15].

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

Task 4: FCPA poker

FCPA Game Rules

One widely-used abstraction in Poker is called Fold, Call, Pot, All-In (FCPA) hold 'em, or simply just **FCPA no-limit poker**, consisting of just these four actions. FCPA Poker is played with a standard 52-card deck. The goal of the game is to win money via betting on the strength of a player's set of five cards (poker hand) determined by a fixed and commonly-known ranking order.

Each player starts with a stack size, S, of *chips*. Each game starts with the first player betting b chips (the *small blind*) and the second player betting B chips (*big blind*), which are placed in the centre pool called the *pot*. Each player is then dealt 2 randomly-drawn private cards from the deck and the game starts.

Each turn, players can choose one of four actions: fold (f), check/call (c), pot-sized bet / raise (p), and all-in (a):

- A fold action is similar to resign: it ends the game immediately, and all the money in the pot is given to the other player.
- When a player calls they bet an amount of money such that they match the other player's contribution to the pot. Playing a call action when both players have contributed the same amount (bet of 0 chips) is called a *check*.
- When a player players a pot-size bet / raise, they first bet enough chips to match the other player's contribution (if necessary), and then they bet (raise) an number of chips equal to the pot size at that point. This action is only legal when a player has at least a number of chips required by this bet size.
- When a player "goes all-in", they contribute the rest of their chips to the pot. The first all-in bet forces the other player to either fold or stay in by responding with an all-in bet.

The game is played over four rounds in order: pre-flop, flop, turn, and river. The small blind starts in the first round and the big blind starts every other round. The pre-flop is the first round where players have only received their private cards. At the start of the flop, three public community cards are randomly dealt face-up on the table. Community cards are cards that both players can use to make their best five-card poker hand, in addition to their private cards. At the start of the turn and river, another single community card is dealt face-up drawn randomly from the deck. A round ends when a player calls or both players have checked.

As values you should use $(S, b, B) = (20000, 100, 150)$.[3] So, for example, on the very first turn the first player has 19950 chips, and the second player has 19900 chips, and there are 150 chips in the pot. The initial legal moves for the first player are $\{f, c, p, a\}$. If the first player folds, the game ends and they lose 50 chips. If the first player calls, they would add 50 chips to the pot and the flop round would begin. If, instead, they played a pot-sized bet, they would first add 50 and then another 200; at that stage, the next player could then respond with a fold, or call (adding another 200 to the pot making it 600 total), or raise again via another pot-sized bet (adding 200 to bring the pot to 600 and then betting another 600). At any time any player can decide to go all-in, forcing the other player to either fold or also go all-in.

[3] Pay attention that for this assignment we use custom values, not the default values for OpenSpiel. See the Github repository.

Other than a player's stack, there is no limit to how many bets occur in each round or how large the pot gets (this is why Doyle Brunson considers no-limit Poker "the only pure game left"[4]!) If both players go all in, then the game proceeds as if both players check from that point on.

If both players are still in at the end of the river round, a *showdown* occurs. Both players then reveal their private cards, and whichever player can make the best five-card poker hand[5] (using their private cards and five community cards) wins the pot. In all cases, the utility is the amount of chips a player after the game ended minus their starting amount (S).

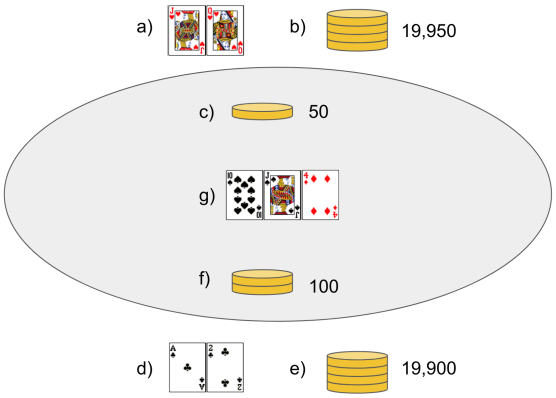


Figure 2: a) Player 1 private cards b) Player 1 stack c) Player 1 commitment d) Player 0 private cards e) Player 0 stack f) Player 0 commitment g) public cards

Goals

You will need to develop an agent that has learned to play FCPA poker, making use of the code infrastructure provided in OpenSpiel (again the environment and all algorithms can be used for this purpose). In the case of FCPA it won't be possible anymore to evaluate your agent using the exploitability and/or NashConv metrics due to the size of the game. Therefore, we expect you to evaluate your agent head-2-head against standard baselines such as random play, always fold, always call, $50/50$ call-fold etc. Use the results of your evaluation and/or literature to motivate your design choices. Being able to run and experiment with your agent (and for us to be able to reproduce this) is part of the evaluation. Thus explicitly describe your model (e.g. network structure) and discuss the learning statistics you used.

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. The tournament is played with all submitted agents and a range of standard baseline agents. This will be used to assess that your agent knows how to play the game. On these outcomes we will also do a game theoretic analysis to determine the winners to make it more fun.

To participate in the tournament, a copy of your code and agent needs to be available in a directory on the departmental computers that will be provided and your agent needs to follow a predetermined template. See `https://github.com/ML-KULeuven/ml-project-2021-2022` for technical instructions. Include all the code you used and have written to perform the experiments and explain how to use it in a readme file. If you work in a team, choose the directory of one team member to copy the code to. Test your (preliminary) code as early as possible. An implementation that does not run on the server reduces your score.

[4] `https://www.youtube.com/watch?v=QAJkvEJ2QjE`
[5] `https://en.wikipedia.org/wiki/List_of_poker_hands`

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

6 FCPA Poker Example Gameplay

Due to poker conventions: big blind is labelled Player 0, small blind is Player 1.

Preflop

- Player 0 is dealt $2\spadesuit, 3\heartsuit$. Player 1 is dealt $A\clubsuit, 2\diamondsuit$.
- Player 1 bets small blind 50
- Player 0 bets big blind 100
- Player 1 plays a pot bet (raising their commitment to 300 ($= 50 + 50 + 200$))
- Player 0 calls

Flop

- Current pot size: 600.
- Flop is dealt: $A\heartsuit, 4\spadesuit, Q\spadesuit$
- Player 0 plays a pot bet (raising their commitment to 900 ($= 300 + 600$))
- Player 1 calls

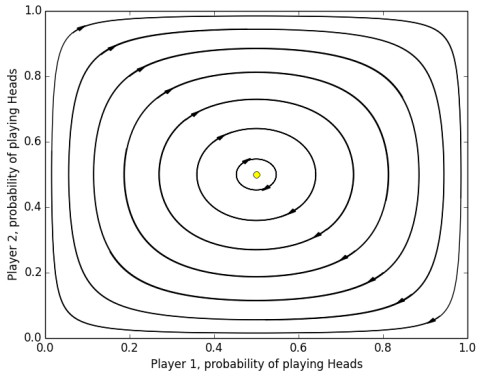
Turn

- Current pot size: 1800.
- One more card is dealt: $A\heartsuit, 4\spadesuit, Q\spadesuit, A\spadesuit$
- Player 0 checks
- Player 1 checks

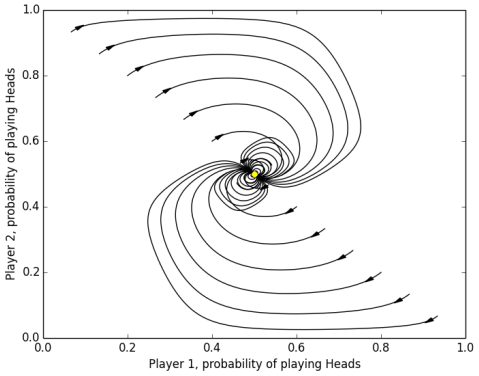
River

- Current pot size: 1800.
- One more card is dealt: $A\heartsuit, 4\spadesuit, Q\spadesuit, A\spadesuit 5\heartsuit$
- Player 0 checks
- Player 1 plays a pot bet (raising their commitment to 2700 ($= 900 + 1800$)
- Player 0 calls.

Reveal As no player folded, the players reveal their cards and the player with the strongest combination wins. The best 5-card combination player 1 can get is $A\clubsuit, A\heartsuit, Q\spadesuit, 4\spadesuit, 5\heartsuit$ (pair of aces). The best 5-card combination player 0 can get is $A\heartsuit, 2\spadesuit, 3\heartsuit, 4\spadesuit, 5\heartsuit$ (straight). As straight is stronger than pair, player 0 wins 2700 chips.



(a) Evolutionary dynamics of the Matching Pennies game.



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

Figure 3: Example of dynamics

References

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

[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] Noam Brown, Anton Bakhtin, Adam Lerer, and Qucheng Gong. Combining deep reinforcement learning and search for imperfect-information games. *Advances in Neural Information Processing Systems*, 33, 2020.

[4] Noam Brown and Tuomas Sandholm. Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018.

[5] 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.

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

[7] Michael Johanson. Measuring the size of large no-limit poker games. *CoRR*, abs/1302.7008, 2013.

[8] 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.

[9] 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.

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

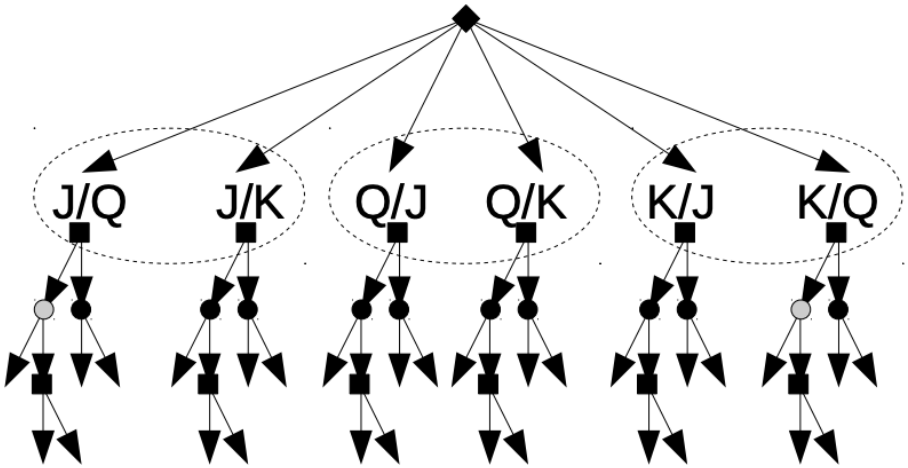


Figure 4: Information sets.

[11] 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.

[12] 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.

[13] 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.

[14] Matej Moravčík and Martin Schmid et al. DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 358(6362), 2017.

[15] 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.

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

[17] 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.

[18] Finnegan Southey, Michael P Bowling, Bryce Larson, Carmelo Piccione, Neil Burch, Darse Billings, and Chris Rayner. Bayes' bluff: Opponent modelling in poker. *arXiv preprint arXiv:1207.1411*, 2012.

[19] 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.

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

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

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

[23] Ryan Zarick, Bryan Pellegrino, Noam Brown, and Caleb Banister. Unlocking the potential of deep counterfactual value networks. *arXiv preprint arXiv:2007.10442*, 2020.