Machine Learning: Project (2024-2025)

Multi-Agent Learning in Canonical Games and Knights Archers Zombies

Giuseppe Marra, Wannes Meert

February 2025

1 Introduction and Related Literature

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, and robotic teams.

This assignment covers topics in multi-agent reinforcement learning (MARL). We assume elementary knowledge of single-agent reinforcement learning (a good reference when less familiar with RL is Sutton and Barto [8]). 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 in combination with multi-agent learning, which is non-stationary and reflects a moving target problem (see [7] or [6] for basic concepts about game theory when less familiar).

In this assignment we first tackle some canonical games from the 'pre-Deep Learning' period. To learn how (multi-agent) reinforcement learning and game theory relate to each other, you will work with tabular RL methods using ε-greedy and Boltzmann exploration [10, 3] and interpret the evolution of the learned policy. Next, we move to the Knights Archers and Zombies game using RL and ML [5, 4].

We will be working in the PettingZoo environment[1] and recommend the RLlib[2] framework for the RL algorithms. Learning how to use advanced, state-of-the-art software toolboxes for AI is part of the project and we expect you to explore the documentation (including manuals, docstrings, code examples, etc.).

2 Approach

You work on this project in a team of ≤2 students. Note that since a lot of code is available within the available frameworks we do expect students to show a good understanding of the techniques deployed, and be able to conduct a knowledgeable conversation about the techniques used during the defense of the project. We use the Dept CS Assignment Commons: ES-GW-TP-TS-NPP-NVP.[3] In summary: All resources are allowed but code or text that is claimed to be authored by the team and cannot be explained or reappears in other submissions/sources is assumed to be copied and may be cause for sanctions. We expect that your code runs on the computers at the Department of Computer Science, and that you participate in the tournament.

Please direct questions that you have about the project to the Toledo forum or the classroom discussion moments such that all students can benefit from the discussion.

[1] https://pettingzoo.farama.org/environments/butterfly/knights_archers_zombies/

[2] https://docs.ray.io/en/latest/rllib/index.html

[3] https://wms.cs.kuleuven.be/cs/english/study/assignment-commons

3 Deadlines

3.1 Form Groups — Before February 28th, 23:59

Mail the team member names to wannes.meert@kuleuven.be and giuseppe.marra@kuleuven.be.

3.2 Submit Draft of Report (optional, not graded) — Before March 23nd, 23:59

If you submit a draft report of tasks 1 and 2 via Toledo, feedback will be provided individually.

3.3 Submit agent to tournament — Before May 14th, 23:59

For the final evaluation of your agents for Tasks 3 and 4 we will play a tournament, in which each agent will play many games. The tournament is played with all submitted agents and a range of simple baseline agents. This will be used to assess whether your agent learned how to play the game.

To participate in the tournament, follow the predetermined template and upload your agent to the departmental computers. See `https://github.com/ML-KULeuven/ml-project-2024-2025` for technical instructions. If you work in a team, choose the directory of one member. Test your (preliminary) code as early as possible on the departmental computers. An implementation that does not run reduces your score.

3.4 Submit Report — Before May 14th, 23:59

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

- Mention the directory on the dept. computers where your **code** for all tasks and agents are stored.
- Formulate your **design choices** as research questions and answer them.
- Write out the (scientifically supported) **conclusions** you draw from your experiments.
- Be concrete and precise about methods, formulas and numbers. A scientific text is **reproducible**.
- Clearly **cite** sources.
- Report, per person, the **time each of you spent** on the project, and how it was divided over the tasks.
- An appendix is allowed for additional results or figures you want to refer to during the discussion (pages >10). There is no guarantee the appendix is read and the first 10 pages need to be self-contained.

3.5 Peer assessment — Before May 14th, 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. This information is used only by the professor and assistants and is not communicated further.

3.6 Oral discussion — Week of May 19th

Discussion about your report and code. Slots will be available on Toledo.

nash - stag, stag and rabbit, rabbit
pareto - stag stag (otherwise both are worse)

		Player 2	
		S	H
Player 1	S	$1, 1$	$0, 2/3$
	H	$2/3, 0$	$2/3, 2/3$

(a) Stag hunt

		Player 2	
		S_1	S_2
Player 1	S_1	$12, 12$	$0, 11$
	S_2	$11, 0$	$10, 10$

(b) Subsidy game

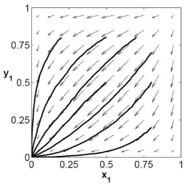
		Player 2	
		H	T
Player 1	H	$0, 1$	$1, 0$
	T	$1, 0$	$0, 1$

(c) Matching Pennis

nash - mixed strategy nash, randomly choose h or t
pareto - none

		Player 2	
		C	D
Player 1	C	$3, 3$	$0, 5$
	D	$5, 0$	$1, 1$

(d) Prisoner's Dilemma



(e) Example of empirical policy traces of the learning behavior overlaid on the vector field of the corresponding replicator dynamics.

Figure 1: Matrix games

nash - defect defect
pareto - silent silent

4 Tasks

Your report should discuss the following tasks (mention the task numbers).

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

Task 1: Literature Study

You are expected to read the relevant sections in the provided references to understand the terms in this assignment. Describe the 3 papers that influenced your approach the most, and explain why.

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

Task 2: Learning & Dynamics: Matrix Games

Here we learn how to play four benchmark matrix games: Stag Hunt, Subsidy Game, Matching Pennies and Prisoner's Dilemma. Use the payoff tables in Figure 1. These games belongs to different categories of games, i.e. social dilemma, zero-sum or coordination game.

Goal

You train a policy with basic RL algorithms for both players per benchmark matrix game using independent learning. Both players use the same RL algorithm. You can use self-play (agents use the same model).

1. List for each game the Nash equilibria and Pareto Optimal states. [$1/8$ points]
2. Implement yourself (a) ε-greedy Q-learning, (b) Boltzmann Q-learning, and (c) Lenient Boltzmann Q-learning [2]. Plot multiple empirical (time-averaged) learning trajectories. Thus, show how the policy changes over multiple iterations of the learning step (see figure 1e for an example). Explain the behavior and the differences between algorithms. Investigate and report on whether the learning algorithms converge to a Nash equilibrium and/or a Pareto optimal state (or why not). [$5/8$ points]
3. For matrix games, we can analytically verify whether your learning trajectories are behaving as expected by computing the replicator equations and plotting the directional (vector) field plots [2]. Do this for Boltzmann Q-learning and Lenient Boltzmann Q-learning. You can compute the equations yourself and make quiver plot or (for Boltzmann Q-learning) use a library like OpenSpiel.[4] [$2/8$ points]

[*With this task you can earn* $8/20$ *points of your overall mark.*]

[4] https://openspiel.readthedocs.io/en/latest/

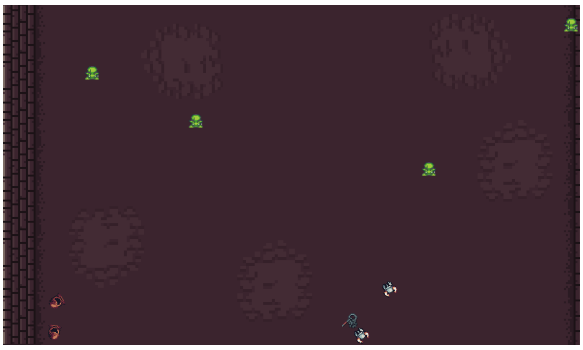


Figure 2: Knights-Archers-Zombies (KAZ) environment.

Task 3: Playing the Knights-Archers-Zombies game

In this task you will train an agent to control a single archer in the Knights-Archers-Zombies game (KAZ, Figure 2) in the PettingZoo environment[5]. In this game, an archer needs to hit as many zombies as possible before either it gets hit or a zombie reaches the bottom line.

You will employ Reinforcement Learning (RL) techniques [1, 8, 9] to develop your solution. In simple matrix games, learning action probabilities (i.e. policies or strategies) directly is feasible because matrix games are stateless, synchronous, single-step interaction games. In fact, in a matrix game, players choose their actions simultaneously, get the corresponding reward and the game resolves in a single step. This is not the case anymore in KAZ, where current actions influence not only immediate rewards but also future states (i.e., a Markov Decision Process). For instance, if only one zombie is present, the best move might be to shoot immediately. Conversely, facing multiple zombies may require repositioning before attacking.

Due to the vast number of state-action combinations in KAZ, standard model-free approaches using tabular representations are impractical. You cannot just store the probabilities (or values) of all the actions for all the possible states. Instead, generalization techniques are necessary, where the information learned in one state can be transferred and re-used in other states. One common approach is leveraging deep neural networks to predict the value or action distribution of states. This requires designing and/or learning features that effectively describe states and actions, allowing the model to generalize well.

Feature representation strategies include:

- Manual feature engineering: Preprocessing states to extract or compute features that simplify learning.
- Automated feature learning: Using raw data as input to a deep learning model. Since states are vectorized, they often contain varying numbers of objects and require padding to a fixed size.[6] Careful adaptation of the model to handle state and state-action pair representations is crucial.
- Hybrid approach: Combining manual and automated feature extraction.

You may leverage implementations from PettingZoo[7] and Ray RLlib[8]. Machine learning models, such as deep neural networks, can represent state-action value functions (Q-values), state value functions (V-values), or directly learn policies. You are free to choose any RL technique, such as deep Q-learning, policy gradient methods, or Proximal Policy Optimization (PPO)—many of which are available in RLlib.

[5] https://pettingzoo.farama.org/environments/butterfly/knights_archers_zombies/

[6] https://pettingzoo.farama.org/environments/butterfly/knights_archers_zombies/#vectorized-default

[7] https://pettingzoo.farama.org/index.html

[8] https://docs.ray.io/en/latest/rllib/index.html

Goal

Your objectives for this task are (both are required to earn points):

1. **Implementation & Evaluation** : You will develop and train an agent for a single-archer KAZ environment using the provided template. This implementation will use PettingZoo for environment interaction and state processing. You may choose a machine learning library to develop you agents; we provide code examples and recommend the RLlib library. You are expected to evaluate your agent on your machine and include such evaluations in your report. You compare your agent against simple baselines you come up with (e.g., random play, always shooting diagonally, etc.).

2. **Central Evaluation (Tournament)**: You upload the agent you trained, and the training code, to the departmental computers. Your agent will play a number of randomly initialized games. The random seeds used for the evaluation are not disclosed in advance, so your agent must handle any possible game (i.e. any possible zombie appearance and configuration). The average reward is computed and used to rank your agent with respect to baseline agents and agents implemented by other students.

Use the results of your evaluation, along with relevant literature, to justify your design choices. Explicitly describe your model architecture (e.g., network structure, input tensor format), the observed gameplay behavior (e.g., what strategies does your model learn?) and the learning statistics you used to analyze performance.

Important: In submitting code for evaluation, you must not alter the original reward scheme provided by PettingZoo for the KAZ environment. However, you are allowed to modify the reward structure of the environment when training on your machine.

[*With this task you can earn* $^{7}/_{20}$ *points of your overall mark.*]

Task 4: Playing Multi-Agent Knights-Archers-Zombies

In this final task, you will explore and potentially implement a multi-agent version of the KAZ environment, where two archers cooperate to eliminate zombies. *This is a more advanced task so approach this only when you have completed the previous tasks.*

Goal

Your objectives for this task are:

1. **Discussion** ($^{2}/_{20}$ points) Analyze how your approach from Task 3 needs to adapt for a multi-agent setting. Use relevant literature and RLlib documentation to explore communication mechanisms and learning strategies and algorithms. Discuss the expected benefits of these strategies.

2. **Implementation** & Evaluation ($^{2}/_{20}$ points): Implement and evaluate a multi-agent version of the KAZ environment featuring two cooperating archers.

[*With this task you can earn* $^{4}/_{20}$ *points of your overall mark.*]

References

[1] Hendrik Blockeel. *Machine Learning and Inductive Inference (course notes)*. KU Leuven, 2024.

[2] Daan Bloembergen et al. "Evolutionary Dynamics of Multi-Agent Learning: A Survey". In: *J. Artif. Intell. Res. (JAIR)* 53 (2015), pp. 659–697.

[3] Lucian Busoniu, Robert Babuska, and Bart De Schutter. "A Comprehensive Survey of Multiagent Reinforcement Learning". In: *IEEE Trans. Systems, Man, and Cybernetics, Part C* 38.2 (2008).

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

[5] Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444.

[6] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009. URL: http://www.masfoundations.org/mas.pdf.

[7] Lukas Schäfer Stefano V. Albrecht Filippos Christianos. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press, 2024. URL: https://www.marl-book.com.

[8] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. 2nd. Cambridge, MA: MIT Press, 2017. URL: http://incompleteideas.net/book/the-book-2nd.html.

[9] Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Morgan & Claypool, 2010. URL: https://sites.ualberta.ca/~szepesva/RLBook.html.

[10] Karl Tuyls and Gerhard Weiss. "Multiagent Learning: Basics, Challenges, and Prospects". In: *AI Magazine* 33.3 (2012), pp. 41–52.