Reinforcement Learning WS 2024/25
Instructor: Georg Martius <georg.martius@uni-tuebingen.de>
TAs: René Geist, Pavel Kolev, Mikel Zhobro, Pierre Schumacher
Final Project
Competition: 24.02.2025, In-person get-together 25.02.2025 14:15-16:00 in N10 (Morgenstelle). Report, Video and Code is due: 26.02.2025 23:55
Last Q&A session: Jan. 28th+30th
December 6, 2024, Version 1.0

1 Reinforcement Learning Project

The final project of the lecture will be on developing a Reinforcement learning agent for a small game. You can form teams of 2 or 3 people. You will implement different reinforcement learning algorithms and evaluate their performance on simpler tasks first and then apply them to a simulated `hockey` game. We are using the gymnasium API (former Open AI gym), see `https://gymnasium.farama.org/` with a custom environment. It uses the same packages as we had for the exercises. The code for the hockey game is at the git repository `https://github.com/martius-lab/hockey-env` (*HockeyEnv* / `Hockey-v0` / `Hockey-One-v0`). Try out the notebook `Hockeyenv.ipynb`.
All teams will compete against each other in the game at the end. Details on the tournament mode will follow. There are intermediate checkpoints, see below, that we will discuss in the remaining tutorial sessions.
Please register your team as soon as possible, but not later than 20th of January at this google form[1]. You will then get account information for the teaching compute cluster.
For the final evaluation, you have to prepare:

(a) A report using the latex template uploaded to ILIAS next to the project description, with:

  (a) an introduction, including a description of the game/problem **(1/2 page)**
  (b) a methods section, including relevant implementation details, modifications and math, e.g. the objective functions of the implemented algorithms **(min 1 page per algorithm/person)**
  (c) an experimental evaluation for all the methods and environments (it needs to including the performance against the basic opponent (weak)). **(min 1 page per algorithm/person)**
  (d) a final discussion, including a comparison of the different algorithms **($\approx$ 1 page)**

[1] `https://forms.gle/tRzxWyCZF9GzQRkT7`

For a single person, the report should not be longer than **5** pages excluding references.

For a team of two people, the report should not be longer than **8** pages excluding references.

For a team of three people, the report should not be longer than **11** pages excluding references.

Each **team member** should have **one algorithm** implemented and his/her independent contribution should be clearly marked in the **report and the source code**. **Significant modifications** of the same base algorithm are possible, for instance, based on DQN, two modifications mentioned in the *Rainbow* paper [8] would be significant per person. For instance, just adding some Gaussian noise to the actions would not be significant. For a top grade self-play or other techniques to account the "game" aspect are expected.

Regarding the usage of existing code you have the following options:

- you implement an algorithm yourself (except the ones we had in the exercises), or
- you base your implementation on some existing code (needs to be specified where it comes from) and you make a non-trivial modification. A non-trivial modification is something that is or might be a main contribution in a paper.

If in doubt, contact the tutors during the exercise sessions in advance.

(b) A video-presentation:

- 3 min for a single person
- 4 min for two-person team
- 6 min for three-person team

(c) The source code

The submission deadline for the report, video, and code is on **26.02.2025 23:55** via ILIAS.
The code needs to be running in the tournament starting on the 24.08. from 10am onward for 12h. Test runs will happen earlier and will be announced.

Requirements:

(a) Teams of two should implement 2 algorithms

(b) Teams of three should implement 3 algorithms

(c) In order to pass the exam report, presentation and code have to be handed in on time.

(d) The code has to run and if the hockey does not work at least a simple environment must be solved.

Usage of GPT Large language models are great tools for various tasks and can help you be more efficient, however, you need to know how to do things yourself properly first, before relying on the LLMs. Since this is part of YOUR education, the usage of chatGPT, or similar, for writing the report is not allowed.

Grading

(a) The mark will be determined based on all parts. You are expected to deliver a nicely written report, a clear presentation, and a good performance.

(b) The final mark will be computed from the individual scores with the following weighting: 60% report, 20% presentation, 20% performance.

(c) The minimum performance is measures against the basic opponent (in weak mode). If an agent cannot win against it consistently (more than 55% of times) performance grade is 5.0. If you are not taking part in the tournament, your performance score can only be 3.0. If in the tournament, you consistently win against the strong basic opponent, you get a 2.0. Better marks are given according to the rank in the tournament. The top 3 teams in the competition get an overall bonus of 0.35 marks. For teams until rank 9 the bonus is 0.2 marks.

1.1 Checkpoint 1: Get your algorithms up and running

Start with the Pendulum-v0 from exercise 8 and 9 or with other simple environments. Be aware that some environments (such as the pendulum) contains an exploration problem that is not necessarily quickly solved by all algorithms. You can also try LunarLander-v2 (exercise 9) or HalfCheetah. Important is to see that the reward is optimized and the behavior is reasonable. This should allow you to debug your code.

Implement your algorithms of choice. I recommend to consider off-policy algorithms: Dueling Deep Q-learning (DDQN) [10], Deep deterministic policy gradient (DDPG) [9] or Twin Delayed DDPG (TD3) [4], Soft/Natural Actor Critic (SAC [5]) or model-based algorithms such as Dreamer [6] or TD-MPC [7]. The versions from our exercises can be a good starting point, but need to be modified (as we provided a solution). You can also use existing implementations. Inspiration for modifications are:

- *Rainbow* paper [8] (for DQN)
- *Pink-noise* paper for exploration [3]
- *CrossQ*: batch normalization for DeepRL [1]
- *RND*: enhance exploration by using adding an exploration reward [2]
- *Two-hot Q*: Using the categorical Q-value representation using symexp twohot loss, see page 7 in [6].

Make appropriate analysis and track your performance etc. Don't forget this procedure during the rest of the project. Remember that you want to create a report with plots giving detail about the training, comparisons etc.

1.2 Checkpoint 2: Hockey – learning to handle the puck

Start working on the Hockey game[2]. The repository provides the environment and a little notebook to see how the environment works. It is actually installable via pip, see the `README.md` file.
The game has some training camp modes that you can test. These are:

TRAIN SHOOTING hitting a static ball into the goal (other agent is static)

TRAIN DEFENSE defending your goal against incoming shots (other agent is static)

However, it usually does not help much to train the agent in stages, so you finally need to train using

NORMAL normal gameplay against another agent.

You can enable the game-modes with
`HockerEnv(mode=NORMAL|TRAIN_SHOOTING|TRAIN_DEFENSE)`.

1.3 Checkpoint 3: Play in normal mode against the basic opponent

The basic opponent has a weak and a strong mode. Train your agent against it in normal game mode. Think how to exploit the fact that you are using an off-policy algorithm.

1.4 Checkpoint 4: Self-play

Let your agents play against each other in normal game mode. Make appropriate analysis and track your performance etc. Experiment with different tournament modes.

1.5 Final

Updates on the tournament server etc will come.

References

[1] Aditya Bhatt, Daniel Palenicek, Boris Belousov, Max Argus, Artemij Amiranashvili, Thomas Brox, and Jan Peters. Crossq: Batch normalization in deep reinforcement learning for greater sample efficiency and simplicity. In *The Twelfth International Conference on Learning Representations*, 2024.

[2] `https://github.com/martius-lab/laser-hockey-env`

[2] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019.

[3] Onno Eberhard, Jakob Hollenstein, Cristina Pinneri, and Georg Martius. Pink noise is all you need: Colored noise exploration in deep reinforcement learning. In *Proceedings of the Eleventh International Conference on Learning Representations (ICLR)*, May 2023.

[4] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1587–1596, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[5] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[6] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models, 2024.

[7] Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. In *The Twelfth International Conference on Learning Representations*, 2024.

[8] Matteo Hessel, Joseph Modayil, H. V. Hasselt, T. Schaul, Georg Ostrovski, W. Dabney, Dan Horgan, B. Piot, Mohammad Gheshlaghi Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, 2018.

[9] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[10] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1995–2003, New York, New York, USA, 20–22 Jun 2016. PMLR.