Reproducibility lab

In this lab, with a team you will pick a research questions and design an experiment to answer the question. You will then write a blog post describing your results. You are allowed to use existing code and existing environments as long as proper credit is given.

Teams

The lab is meant for teams of 3 or 4. If you want to work in a team of 5, talk to your TA during one of the classes. In general, we expect a clear additional effort from teams of 5.

Timeline

- Given the short time, it is important to start work in time. We suggest making groups and picking a topic on September 30th. **Sign up for a lab3 group on Canvas.** You can choose from the list of topic suggestions below, if you want to define your own topic, make sure the scope fits the timeline and discuss with a TA first.
- In all future tutorial sessions, some or all of the session can be used to work on the lab. The TA's will be available to answer questions and give you suggestions and feedback. The lab workload assumes you work on the lab outside of the sessions as well.
- We will do a peer feedback session in the tutorial sessions of october 14th. **Bring a draft of your blog post - as many copies as you have team members.** You should at least have initial answers up to and including step 4 (experimental design) in the overview below. If you do not have all the results yet, at least it is useful to get feedback on your writing and the design of the experiment. **Presence and participation at the session will contribute to your grade.**
- The feedback you receive will not influence your grade. However, you have the chance to improve your blog post based on the feedback you received, and almost an additional week to finalize experiments. You will also have the chance to ask the TAs any last minute questions during the tutorial sessions on October 16th and 17th.
- Final blog hand-in date: October 18th, 23:59.

The estimated workload for the project is 20 hours per person up to the feedback session, and time after it (up to 12 hours) to possibly finalize experiments and revise your blog post based on feedback. This means there is relatively little time. Thus, limit the scope of the projects. For example, getting big networks to run on difficult tasks often takes lot of tuning and computation time. Luckily, many questions can be answered by investigating simpler tasks and tabular or linear representations. *Choose simple enough tasks (e.g. tabular MDPs, mountain car) such that the project can be completed in the assigned time! Consider you will need to learn models several times to be able to judge the reliability of your results.*

Peer feedback

In order to get feedback on your experiment design, we will do a peer feedback session where you critically examine the project design of another student with an eye on the evaluation criteria below. Your feedback will not impact the grade of the other student. However, participation in the feedback session (in the tutorial session on October 14th) does count towards your own grade.

Steps

1. Choose one of the research questions below.
2. Describe why you think this research question is important for the understanding or application of reinforcement learning methods.
3. Briefly describe the main technique(s) you are investigating
4. What type of experiment(s) do you need to do to answer the research question, and what data do you need to collect? Describe (at least) the following points:
   (a) What environment(s) should you test your technique(s) on? How did you pick the number and type of environments?
   (b) What methods should you compare the chosen technique to? How did you pick these?
   (c) What hyperparameters should you set? How to ensure comparisons are fair with regard to the hyperparameters?
   (d) Which quantities do you need to measure? How did you pick these?
   (e) How many random runs do you need? How did you pick these?
5. Set-up and run the described experiments. You are free to use any code you find on-line, but be sure to sanity check the code and the results, and to give proper credit.
6. Report the results from your experiments. Make sure the presentation of the results is clear, and that we can also see what the level of confidence in the results is. If you use errorbars, state clearly what they represent.
7. Describe any conclusions you draw from your experiments.
8. Write a blog post including all answers given to the questions above. Make a story out of your answers. Submit your blogpost and code by October 18th, 23:59. You can submit a link to a live website, but also submit an offline version (e.g. html or pdf).

Evaluation criteria

We will evaluate the project based on the following criteria. Make sure that each of the aspects is explained in the blog post, otherwise, we can't award you points for it!

- **Presentation (20%).** Is the final blog-post clear and legible? Are figures or media and design elements (titles, captions) used effectively? Is the information easily accessible (avoid walls of text).
- **Motivation and research question (10%).** Does it become clear why you think that the central question in your project is important?
- **Explanation of the algorithms and techniques used (20%).** Do you clearly convey to your audience how the method does what it does, beyond equations or pseudocode?
- **Experimental design (20%).** Did you use appropriate techniques to set up your experiment? Definitely consider the suggestions in 'Deep Reinforcement Learning that Matters' [1], that will be discussed in class. How do you make sure comparisons are as fair as possible? How to prevent looking at noise rather than a real difference between methods?
- **Results and conclusions (20%).** Are the results clearly presented? Is the reliability of the results clear (e.g. is the spread of results clear, and is it clear how it was calculated?). Can the reader understand what graphs and tables mean? Do you draw conclusions from the results, and are the conclusions sufficiently supported by the experiments?
- **Feedback given in peer-review session (10%).**
- **Credit.** Clearly mention where you have used code (environments, algorithms) or other resources (e.g. figures) by other people in your blog post. **Presenting other people's work as yours constitutes plagiarism.**

Topics

Following is a list of suggested topics. It is ok to focus on a specific aspect of the topic. If you are unsure, talk to your TA. It is ok if multiple groups work *independently* on the same topic. Given the short term, start with a *simple* environment (you can always make it more complicated if you have time). You can consider grid-worlds, the cliff world from the lecture, the chain MDP in [2], the mountain car, pendulum balancing, etc.

1. Compare double Q-learning to Q-learning. In what environment is there a big difference? Are there environments where (single) Q-learning is better?
2. There are different types of experience replay, e.g. prioritised experience replay and hindsight experience replay. Compare two or more types of experience replay. Does the 'winner' depend on the type of environment?

3. In (close to) deterministic systems the performance of a greedy policy indicates the value of a state, and can thus be used as baseline [3, 4]. Compare the performance of the self-critic to a learned value function. How does the self-critic perform if the environment gets more stochastic?

4. If instead of taking the semi-gradient, we take the full gradient of the TD error, does this lead to problems in practice? Investigate this on environments with different properties.

5. Semi-gradient methods do not always coverge. DQN [5] uses a semi-gradient version of Q-learning. Can you find an environment where this method diverges? How much do the tricks in DQN (e.g. experience replay, target network) help to avoid divergence?

6. Compare natural gradient to 'vanilla' gradient for different types of policies. Can you find policies where there is a big difference between the two? Conversely, are there policies for which it does not matter a lot?

7. Study n-step bootstrapping in actor critic methods (e.g. generalized advantage estimation, [6]). What are the benefits and disadvantages compared to Monte-Carlo returns and 1-step methods?

References

[1] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *AAAI National Conference on Artificial Intelligence (AAAI)*, 2018.

[2] Nikos Vlassis and Marc Toussaint. Model-free reinforcement learning as mixture learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1081–1088. ACM, 2009.

[3] Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. Self-critical sequence training for image captioning. In *CVPR*, volume 1, page 3, 2017.

[4] WWM Kool and M Welling. Attention solves your TSP. *arXiv preprint arXiv:1803.08475*, 2018.

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[6] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *International Conference on Learning Representations (ICLR)*, 2015.