4. Assignment 3: Advanced Actor Critic - PPO and/or SAC

4.1. Motivation

In the assignments so far, we have covered some of the big milestones in the advent of Deep Reinforcement Learning. The methods introduced are very illustrative, as we have moved from a method with only a value network (DQN), to one employing only a policy network (REINFORCE), ending up with a method that uses both (Actor-Critic). However, these methods are hardly state-of-the-art today.

In this assignment, you will have the opportunity to study and implement a state-of-the-art Algorithm. More specifically, you may choose between Proximal Policy Optimisation (PPO-clipped) (cf. (Schulman et al., 2017) the original paper, or spinning up for a conceptual explanation); and Soft-Actor-Critic (SAC) (cf. (Haarnoja et al., 2018; 2019) the original papers, or spinning up for a conceptual explanation). Both algorithms still follow the principal idea of an actor-critic method, but they make modifications to the value and/or policy losses.

4.2. Tasks

The assignment will resolve around two major parts: A more theoretical component and some experiments.

Theory

Explain the algorithm, that you have chosen. More specifically, we want you to explain the employed policy (actor) loss and the employed (critic) loss. Moreover, you should provide some motivation about the origin of that loss, and explain how this relates to the ”basic” Policy losses in Eq. (2) and (3) respectively.

Moreover, you should provide an explanation of how this algorithm works (if you use pseudo-code at most 8 lines).

Experiments

We again stay in the Cartpole environment. Implement the algorithm that you chose. We expect an implementation based on the pseudo-code you provided and an explanation of all engineering tricks you employ, like mini-batching, general advantage estimation and alike. If you implement it from scratch, none of these tricks have to be employed, but if you scrape the code from somewhere, we expect you to at least understand all the tricks that are implemented.

Note please, that if you use engineering tricks, that are not mentioned sufficiently, we consider this as a case of fraud, as you might have simply copy-pasted code, which will result in a fail on this assignment.

The results of this implementation should be compared to your results from Assignment 1 & 2, you may also reuse the baseline provided there as a point of reference. Discuss your results.

4.3. Task-Checklist

1.1 Explain the Algorithm of your choice: Motivation (high-level), loss function and explain the steps in the algorithm

1.2 Relate your algorithm back to AC/A2C, i.e. what are the differences?

2.1 Implement the Algorithm

2.2 Explain ALL engineering tricks you use, that are outside of the basic method (minibatching, etc.) and provide a reference if needed

2.3 Plot your results, reference back to the results of Assignment 1 & 2.

4.4. Submission

Make sure to nicely document everything that you do. Your final submission consists of:

(i) Your source code, together with a requirements file and a README with instructions that allows us to easily (single command per experiment / sub task) rerun your experiments on a university machine booted into Linux (DSLab or computer lab). To generate a requirements file boot up a terminal, activate your virtual/conda environment (if you used one), navigate to the folder in which you want to generate the file and run the following command: `python -m pip freeze > requirements.txt`.

(ii) You have to use the ICML Latex template for your Report. (Obtainable here)

(iii) Chapter Structure: Abstract (at most quarter of a page), Introduction, Theory (Theory only, around one page), Experiments (setup & results; explaining environment should take at most quarter of a page), Discussion (discussion of results & weaknesses), Conclusion (reflect own work & outline future work).

(iv) A self-contained scientific pdf report of 4-6 pages with figures etc. This report contains an explanation of the techniques, your experimental design, results (performance statistics, other measurements,...), and overall conclusions, in which you briefly summarize the goal of your experiments, what you have done, and what you have observed/learned.

If you have any questions about this assignment, please visit our lab sessions on Friday where we can help you out. In case you cannot make it, you can post questions about the contents of the course on the Brightspace discussion forums, where other students can also read and reply to your questions.

4.5. Deadline

May 09, 23:59:59

References

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 1861–1870. PMLR, July 2018.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., and Levine, S. Soft Actor-Critic Algorithms and Applications, January 2019.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Mnih, V. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal Policy Optimization Algorithms, August 2017.

Sutton, R. S. Reinforcement learning: An introduction. *A Bradford Book*, 2018.

Towers, M., Kwiatkowski, A., Terry, J., Balis, J. U., De Cola, G., Deleu, T., Goulao, M., Kallinteris, A., Krimmel, M., KG, A., et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.