Reinforcement Learning
Project work

November 7, 2021

1 Introduction

In the project work, we will implement and apply some more advanced RL algorithms in continuous control tasks. The project work includes two parts. First, two widely used reinforcement learning algorithms, **TD3** [1] and **PPO** [2], will be implemented. For this part, we will offer the base code so you can start easily. After finishing this part, you can train a policy to balance a inverted pendulum and to control a halfcheetah running forward. In the second part, you need to read some research papers and implement their proposed algorithms based on the code finished in Part I. The candidate algorithms in Part II include

- MBPO [3]]: How to use model-based approach to improve sample efficiency?
- REDQ [4]: How to significantly improve sample efficiency of model-free RL with ensembles?
- SAC [5]: SAC introduces the maximal entropy RL, and it offers you a view of treating RL as probabilistic inference.
- TD3_BC [6]: An offline RL method that learns policy without interacting with the environment.

According to your preference, you can choose one of them to understand the paper and to implement the algorithm. Also, in the second part, you are free to choose other interesting papers to implement, but some papers are hard to reproduce, so be careful to choose them. This project work is supposed to be done in groups of 2 students. If you need to find a partner for the project, please join the **project** channel on Slack and advertise yourself.

Furthermore, it's possible to choose an alternative project by proposing their own topic to combine RL with their research. The proposal of the alternative project should be submitted before 29th Oct (but please contact TAs asap to discuss your project) and **the deadline for both standard project and the alternative project is 05.12.2021, 23:55.** We mainly recommend this for doctoral students who want to explore possible ways of integrating reinforcement learning in their research.

2 Part I

In the first part, you will implement two commonly used methods in RL, especially when action space is continuous – TD3 an PPO. This part constitutes 50 % of the total project points. Also, we recommend to work on the task in the order they are presented in this document (since Part II largely depends on Part I).

2.1 Algorithm 1: Twin Delayed Deep Deterministic Policy Gradient (TD3)

Read the Twin Delayed Deep Deterministic Policy Gradient (TD3) paper.

2.1.1 Question 1

What are the problems with the deep deterministic policy gradient (DDPG) algorithm? How does TD3 solve these problems? (5')

2.1.2 Question 2

For policy gradient methods seen in Exercise 5, we update the agent using only on-policy data, while in TD3 we can use off-policy data. Why is this the case? (5')

2.1.3 Question 3

Finish the implementation of the TD3 algorithm and train the agent with both InvertedPendulumBulletEnv-v0 and HalfCheetahBulletEnv-v0 environments. The places where you need to make modifications to the code are marked as *TODOs* in the `td3.py` file. Train your agents with three random seeds for both environments. Include the training plots in the project report and attach the agents' weights to your submission, with filenames ending with `td3.pth` containing the run ID (1, 2, and 3, each with a different random seed) and the environment name.

We suggest to develop and test your implementation based on InvertedPendulumBulletEnv-v0 environment , and then train it on HalfCheetahBulletEnv. For the InvertedPendulum environment, you can set the "–max_timesteps=200,000" and for the HalfCheetah, it's "–max_timesteps=1,000,000". (5')

2.1.4 Question 4

Now let's analyze the sensitivity of TD3 to hyperparameters. Choose one hyperparameter, e.g., target action noise, exploration noise, or policy update frequency, that you think heavily influence the training and explain why. Then, train your agent with the modified hyperparameter on both InvertedPendulumBulletEnv-v0 and HalfCheetahBulletEnv-v0 environments (3 random seeds). Show the training plots and submit the trained model of HalfCheetah with name "td3_q4.pth". (5')

2.1.5 Question 5

After playing with the TD3 algorithm, could you find any aspect that could be improved? Please list three of them. Also, please propose a potential solution to one of the problems you listed. You can answer this question by providing a paper link and explaining in your own words how the proposed approach solves/mitigates the problem. (5')

2.2 Algorithm 2: Proximal Policy Optimization Algorithms (PPO)

The proximal policy optimization (PPO) was introduced in [2] and presented in Lecture 6.

2.2.1 Question 1

Why does clipping the $\frac{\pi_\theta(a|s)}{\pi_{old}(a|s)}$ ratio stabilize the training? What is the relationship between TRPO [7] and PPO? (5')

2.2.2 Question 2

Please finish the implementation of the PPO algorithm. The places where you need to make modifications to the code are marked as *TODOs* in the `ppo.py` file. Similar to Question 3, train the agent on both the InvertedPendulumBulletEnv-v0 and the HalfCheetahBulletEnv-v0 environments.Train your agents with three random seeds for both environments. Include the training plots in the project report and attach the agents' weights to your submission, with filenames ending with `ppo.pth` containing the run ID (1, 2, and 3, each with a different random seed) and the environment name. For the training curve, you can reference 2. (10')

2.3 Question 3

In PPO, the target value is calculated by generalized advantage estimation (GAE) [8], as shown in the second equation. Explain the relationship between n-step advantage and GAE. Why is GAE better than n-step advantage? (10') [Hint: you can reference this paper: https://arxiv.org/abs/1506.02438]

- n-step advantage: $A^t = -V(s_t) + r_t + \gamma r_{t+1} + \cdot + \gamma^{T-t+1} r^{T-1} + \gamma^{T-t} - V(s_T)$

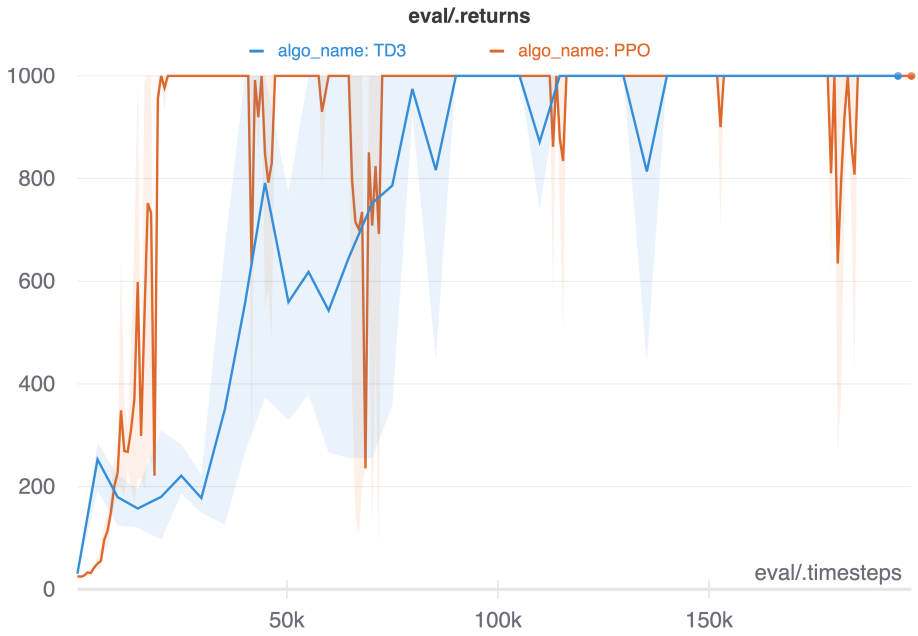

Figure 1: Results of PPO and TD3 on the InvertedPendulum environment.

- GAE: $A^t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdot + (\gamma\lambda)^{T-t+1}\delta_{T-1}$, where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$

3 Part II

In this part, you will read one selected research paper and implement the corresponding algorithm. We select four candidates for you, but you are free to choose other papers interest you. For the papers given by us, we will offer the reference training curve. The score will be given based on whether the algorithm is correctly implemented and **your report**. Please note that we will give more score on your report, since it reflects your understanding of the methods.

3.1 The selected algorithms

Model-based Policy Optimization (MBPO) [3] Sample efficiency is one bottleneck to apply RL in real world especially in the model free methods we have been using so far in the exercises. Model-based RL is one approach to mitigate this problem. In MBPO sample efficiency is improved by learning a dynamics model of the environment and using it to generate synthetic data. It should be noticed that, we tested this algorithm on Openai Gym environment but not make it work properly on PyBullet environment yet. So if you want to implement this algorithm, **please contact us**. We can help you to setup the Openai Gym environment and offer base codes.

Randomized Ensembled Double Q-learning (REDQ) While Model-based RL methods are known to improve sample efficiency they come with their own downsides. For example

one needs to learn good dynamics models which is challenging and often requires a lot of computational resources. Therefore, in this paper, you will investigate a model-free approach to improving sample efficiency.

Soft Actor Critic (SAC) SAC is another commonly used RL method. This paper introduces the maximal entropy RL framework. If you want have a deeper understanding, this tutorial (https://arxiv.org/abs/1805.00909) is a good option. This paper will show you how to treat RL as probabilistic inference. For the training curve of SAC and REDQ, you can reference

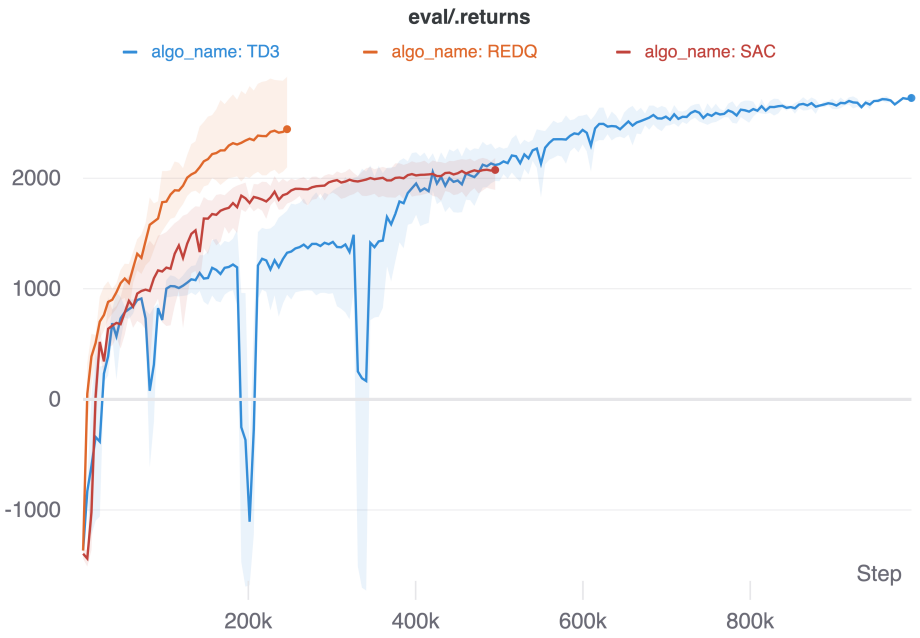

Figure 2: Results of TD3, SAC and REDQ on the HalfCheetah environment.

A Minimalist Approach to Offline Reinforcement Learning (TD3_BC) RL needs to keep interacting with the environment to learn an agent. This might be problematic in real world because some samples are expensive to obtain (e.g., in robotics, self-driving car, etc). This method introduces you a way to learn an agent from pre-collected dataset without interacting with environment. This tutorial (https://arxiv.org/abs/2005.01643) gives you a good overview of offline RL.

We offer one dataset under the "part2" folder named "halfcheetah_mixed.pickle". Please test your algorithm with different data size: 100%, first 50%, first 25% and first 10% (for each data size, run your algorithm with three random seeds).

You can use pickle.load to load them. These datasets are stored in a dictionary with keys: "states", "actions", "next_states", "rewards" and "not_dones".

The reference training curve is Fig 3

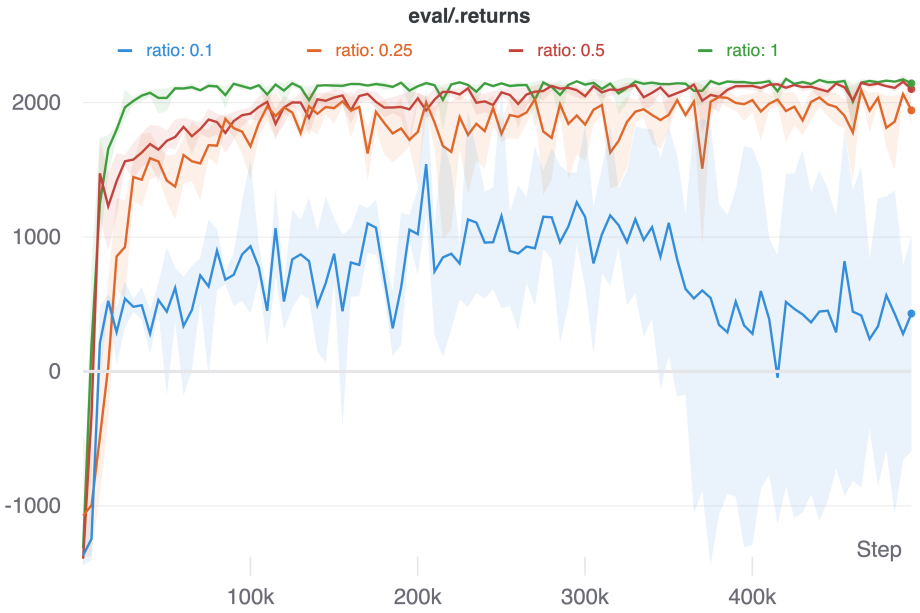

Figure 3: Results of TD3_BC on 100%, 50%, 25% and 10% data.

3.2 Questions

3.2.1 Question 1

Please correctly implement your algorithm and show the training plots against the TD3/PPO with three random seeds. The plots should include both InvertedPendulum and HalfCheetah environments. Also, you need to describe the network structure and training procedure as well as hyperparemeters. A clear way to show the hyperparameters is using a table.

If your algorithm failed to learn (especially when the algorithm is selected by yourself), include the training plots as well and try to investigate why it didn't perform as expected.

To save the training time, you can use less training timesteps by changing the "–max_timesteps", e.g., SAC: 500,000, REDQ: 250,000, TD3_BC: 500,000.

You should submit the code and the trained model on HalfCheetah named "<algo_name>.pth" (20')

3.2.2 Question 2

Let's analysis your algorithm by performing an ablation study. You could modify one design option that you expect to influence the training performance. Then train the agent using the modified code and compare the results to the original algorithm. Training your agent on HalfCheetah environment is enough but with three random seeds.

For example, in case of the MBPO algorithm we can investigate the role the ensemble plays in modeling dynamics by replacing it with a feed forward neural network. Retraining the policy

and comparing it to baseline MBPO results will allow us to evaluate the performance differences cause by this change. Notice that, hyperparameters are not considered as "design options".

You can set a flag in parser to decide whether to run normal training or ablation study. Please submit the agent's weight with name "<algo_name>.pth" (10')

4 Report & Grading

The report should follow the following structure,:

- Introduction
- Describe the chosen method. (10')
- Answers to the questions in part 1. (50')
- Answers to the questions part 2. (30')
- Conclusion. (2')

We will give **8 points** for the report quality:

- Structure (2')
- Formatting (2')
- Plots included and labeled (2')
- Language (2')

5 Hints

5.1 Computational power for training — Maari

You can train your models on the computers in the Maari building. In order to connect remotely, first use SSH to log in to `kosh.aalto.fi` or `lyta.aalto.fi`, and from there use the `ssh` command to connect to one of the CS computers.

The machine available for students can be found here.

Remember that there might be other people working on these machines - please check the usage with `htop` and `nvidia-smi` before running heavy training.

Also, don't run your programs directly on Kosh or Lyta - those servers were not designed to handle heavy workloads, and are meant to be used as gateways to access other Aalto machines.

5.2 Training over multiple days

In order to leave your processes running in the background after you've closed your SSH connection or left the machine, you can use the `screen` command. When you start `screen`, it will open a new terminal, in which you can start your lengthy process. When you want to log out of the machine (or close the SSH session), you can detach your `screen` terminal by pressing `Ctrl+a` and `d` (one after another). Doing so will close the terminal and leave your processes running in the background.

To reconnect to an existing `screen` session, use `screen -Rd`.

5.3 Remote development

We also offer a simple tutorial on how to develop remotely with VSCode in the project folder named "remote_tutorial.md", you can reference it to setup your work flow.

For more info and examples refer to the setup instructions on mycourses.

6 Submission

Your submission should include:

```
first name_last name_student no.
    |- Report_first name_last name_student no.pdf
    |- part1
        |- codes, named ppo.py, td3.py, <others>.py
        |- trained agents, named ppo/<env_name>/<seed>/ppo_<env>_<seed>.pth,
                                 td3/<env_name>/<seed>/td3_<env>_<seed>.pth,
                                 td3_q4/<env_name>/<seed>/td3_q4_<env>_<seed>.pth
    |- part2
        |- code, named <algo_name>.py
        |- trained agents. The folder names are following part1.
```

References

[1] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*. PMLR, 2018, pp. 1587–1596.

[2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017. [Online]. Available: http://arxiv.org/abs/1707.06347

[3] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," *arXiv preprint arXiv:1906.08253*, 2019.

[4] X. Chen, C. Wang, Z. Zhou, and K. Ross, "Randomized ensembled double q-learning: Learning fast without a model," *arXiv preprint arXiv:2101.05982*, 2021.

[5] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[6] S. Fujimoto and S. S. Gu, "A minimalist approach to offline reinforcement learning," *arXiv preprint arXiv:2106.06860*, 2021.

[7] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," *CoRR*, vol. abs/1502.05477, 2015. [Online]. Available: http://arxiv.org/abs/1502.05477

[8] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.