CSE 4/546: Reinforcement Learning
Fall 2024

Instructor: Alina Vereshchaka

Assignment 3 - Actor-Critic

Due Date: November 21, Thu, 11:59pm

1 Assignment Overview

The goal of the assignment is to help you understand the concept of policy gradient algorithms, to implement the actor-critic algorithm and apply it to solve Gymnasium environments. We will train our networks on multiple RL environments.

Part I [60 points] - Implementing Advantage Actor Critic (A2C/A3C)

In this part we will implement an Advantage Actor Critic (A2C/A3C) algorithm and test it on any simple environment. A2C is a synchronous version of the A3C method.

1. Implement the A2C algorithm. You are welcome to implement A2C or A3C version of the algorithm. Any other variations will not be evaluated. You may use any framework (Tensorflow/PyTorch). **Implement at least 2 actor-learner threads.**

Algorithm S3 Asynchronous advantage actor-critic - pseudocode for each actor-learner thread.

// Assume global shared parameter vectors θ *and* θ_v *and global shared counter* $T = 0$
// Assume thread-specific parameter vectors θ' *and* θ'_v
Initialize thread step counter $t \leftarrow 1$
repeat
  Reset gradients: $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$.
  Synchronize thread-specific parameters $\theta' = \theta$ and $\theta'_v = \theta_v$
  $t_{start} = t$
  Get state s_t
  repeat
    Perform a_t according to policy $\pi(a_t|s_t;\theta')$
    Receive reward r_t and new state s_{t+1}
    $t \leftarrow t + 1$
    $T \leftarrow T + 1$
  until terminal s_t **or** $t - t_{start} == t_{max}$
  $R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t, \theta'_v) & \text{for non-terminal } s_t \text{// Bootstrap from last state} \end{cases}$
  for $i \in \{t-1, \ldots, t_{start}\}$ **do**
    $R \leftarrow r_i + \gamma R$
    Accumulate gradients wrt θ': $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i;\theta')(R - V(s_i;\theta'_v))$
    Accumulate gradients wrt θ'_v: $d\theta_v \leftarrow d\theta_v + \partial \left(R - V(s_i;\theta'_v)\right)^2 / \partial \theta'_v$
  end for
  Perform asynchronous update of θ using $d\theta$ and of θ_v using $d\theta_v$.
until $T > T_{max}$

2. Train your implemented algorithm on any environment. Check "Suggested environments" section.

3. Show and discuss your results after applying the A2C/A3C implementation on the environment. Plots should include the total reward per episode.
4. Provide the evaluation results. Run your environment for at least 10 episodes, where the agent chooses only greedy actions from the learnt policy. Plot should include the total reward per episode.
5. If you use your Grid World environment, render one evaluation episode to visualize how the agent behaves.

Part II [Total: 40 points] - Solving Complex Environments

In this part, test the A2C/A3C algorithm implemented in Part I on any other two complex environments.

1. Choose an environment. At least one of the environments has to be among "Suggested environments - Part II". The environment with multiple versions is considered as one environment.
2. Apply the A2C/A3C algorithm to solve it. You can adjust the neural network structure or hyperparameters from your base implementation.
3. Show and discuss your results after applying the A2C/A3C implementation on the environment. Plots should include the total reward per episode.
4. Provide the evaluation results. Run your environment for at least 10 episodes, where the agent chooses only greedy actions from the learnt policy. Plot should include the total reward per episode.
5. If you use your Grid World environment, render one evaluation episode to visualize how the agent behaves.
6. Go to Step 1. Provide the results for TWO environments.

Suggested environments

Part I

- Your grid world defined in A1 or A2
- CartPole
- Acrobot
- Mountain Car
- Pendulum
- Lunar Lander

Part II

- Any multi-agent environment
- Car Racing
- Bipedal Walker
- MuJoCo Ant
- Any Atari env
- Any other complex environment that you will use for your Final Project

In your report:

1. Discuss the A2C/A3C algorithm you implemented.
2. What is the main difference between the actor-critic and value based approximation algorithms?
3. Briefly describe THREE environments that you used (e.g. possible actions, states, agent, goal, rewards, etc). You can reuse related parts from your Assignment 2 report.
4. Show and discuss your results after training your Actor-Critic agent on each environment. Plots should include the reward per episode for THREE environments. Compare how the same algorithm behaves on different environments while training.
5. Provide the evaluation results for each environments that you used. Run your environments for at least 10 episodes, where the agent chooses only greedy actions from the learnt policy. Plot should include the total reward per episode.
6. If you are working in a team of two people, we expect equal contribution for the assignment. Provide contribution summary by each team member.

Extra Points [max +12 points]

- **Implement a different version of actor-critic [7 points]**
  Extend your A2C/A3C algorithm to a more complex version, e.g. PPO/TRPO/DDPG/TD3/SAC. Compare the results after applying it to the same THREE environments used in the assignment. Provide three rewards dynamic plots for each environment with the results of two algorithms: A2C/A3C and improved version. Discuss the results.
- **Solve Mujoco Environment [5 points]**
  Solve one of the MuJoCo environments with any Actor Critic algorithm. You can use your A2C/A3C implementation or implement an advanced version.

2 References

- NeurIPS Styles (docx, tex)
- Overleaf (LaTex based online document generator) - a free tool for creating professional reports
- Gymnasium environments
- Atari Environments
- Lecture slides
- Asynchronous Methods for Deep Reinforcement Learning

3 Assignment Steps

1(a). Register your team (Due date: Nov 7)

- You may work individually or in a team of up to 2 people. The evaluation will be the same for a team of any size.
- Register your team at UBlearns > Groups. You have to enroll in a team on UBLearns even if you are working individually. Your teammates for A2 and A3 should be different.

1(b). For a team of 2 students (Due date: Nov 10)

- Create a private GitHub repository for the project and add our course GitHub account as a collaborator: @ub-rl
- Each team member should regularly push their progress to the repository. For example, you can sync the repository daily to reflect any updates or improvements made
- In your report include a link to the repository along with the contribution table. Additionally, add screenshot(s) of your commit history.

1(c). Submission Format

- **Report:** The report should be delivered as a separate pdf file, and it is recommended for you to use the NIPS template to structure your report. You may include comments in the Jupyter Notebook, however you will need to duplicate the results in the separate pdf file.

  Report should follow the following naming convention:
  UBIT $TEAMMATE1\_TEAMMATE2\_assignment3\_report.pdf$
  (e.g. $avereshc\_nitinvis\_assignment3\_report.pdf$)

- **Code:** Python is the only code accepted for this project. You can submit the code in Jupyter Notebook with the saved results. You can submit multiple files, but they all need to have a clear name. After executing command Jupyter Notebook, it should generate all the results and plots you used in your report and should be able to be printed out in a clear manner.

  Name your Jupyter Notebooks following the pattern:
  $TEAMMATE1\_TEAMMATE2\_assignment3\_part1.ipynb$
  and $TEAMMATE1\_TEAMMATE2\_assignment3\_part2.ipynb$
  (e.g. $avereshc\_nitinvis\_assignment3\_part1.ipynb$)

- **Model Parameters:** Saved weights of the model(s) as a pickle file or .h5, so that the grader can fully replicate your results. Name your .pickle, .h5 or .pth files using the following pattern:
  $TEAMMATE1\_TEAMMATE2\_assignment3\_part1\_a2c\_cartpole.pickle$
  $TEAMMATE1\_TEAMMATE2\_assignment2\_part1\_a2c\_lunar_{l}ander.pickle$
  (e.g. $avereshc\_nitinvis\_assignment3\_part1\_a2c\_cartpole.pickle$)

2. Submit final results (Due date: November 21, Thu)

- Fully complete all parts of the assignment
- Submit at UBLearns > Assignments
- You can submit multiple files, but they all need to be labeled clearly.
- Your Jupyter notebook should be saved with the results
- Include all the references at the end of the report. Metion all the materials that have been used to complete the assignment
- If you are working in a team of two, we expect equal contribution for the assignment. Each team member is expected to make a code-related contribution. Provide a contribution summary by each team member in a form of a table below. If the contribution is highly skewed, then the scores of the team members may be scaled w.r.t the contribution.

Team Member	Assignment Part	Contribution (%)

Important Notes

- Only files submitted on UBlearns are considered for evaluation.
- Files from local devices, GitHub, Google Colab, Google Docs, or other locations are not accepted.
- Regularly submit your work-in-progress on UBlearns and always verify submitted files by downloading and opening them.

4 Academic Integrity

This assignment can be completed individually or in a team of two students. Teams can not be the same for A2 & A3 assignments. The standing policy of the Department is that all students involved in any academic integrity violation (e.g. plagiarism in any way, shape, or form) will receive an F grade for the course. The catalog describes plagiarism as "Copying or receiving material from any source and submitting that material as one's own, without acknowledging and citing the particular debts to the source, or in any other manner representing the work of another as one's own.". Updating the hyperparameters or modifying the existing code is not part of the assignment's requirements and will result in a zero. Please refer to the UB Academic Integrity Policy.

5 Important Information

This assignment can be completed in groups of two or individually. Teams can not be the same for A2 & A3 assignments.

6 Late Days Policy

You can use up to 5 late days throughout the course toward any assignments' checkpoint or final submission. You don't have to inform the instructor, as the late submission will be tracked in UBlearns. If you work in teams the late days used will be subtracted from both partners. E.g. you have 4 late days and your partner has 3 days left. If you submit one day after the due date, you will have 3 days and your partner will have 2 days left.

7 Important Dates

November 7, Thu, 11:59pm - Register your team (UBlearns > Groups)

November 21, Thu, 11:59pm - Assignment 3 is Due