Online and Reinforcement Learning

2024-2025

Home Assignment 4

Yevgeny Seldin and Christian Igel
Department of Computer Science
University of Copenhagen

The deadline for this assignment is **5 March 2025, 20:59**. You must submit your *individual* solution electronically via the Absalon home page.

A solution consists of:

- A PDF file with detailed answers to the questions, which may include graphs and tables if needed. Do *not* include your full source code in the PDF file, only selected lines if you are asked to do so.
- A .zip file with all your solution source code with comments about the major steps involved in each question (see below). Source code must be submitted in the original file format, not as PDF. The programming language of the course is Python.

Important Remarks:

- IMPORTANT: Do NOT zip the PDF file, since zipped files cannot be opened in *SpeedGrader*. Zipped PDF submissions will not be graded.
- Your PDF report should be self-sufficient. I.e., it should be possible to grade it without opening the .zip file. We do not guarantee opening the .zip file when grading.
- Your code should be structured such that there is one main file (or one main file per question) that we can run to reproduce all the results presented in your report. This main file can, if you like, call other files with functions, classes, etc.
- Handwritten solutions will not be accepted.

1 Policy Gradient Methods `(50 points) [Christian]`

1.1 Baseline

The policy gradient theorem can be generalized to include a baseline, e.g.,

$$\nabla_{\boldsymbol{\theta}} J(\pi) = \sum_s \mu^\pi(s) \sum_a \nabla_{\boldsymbol{\theta}} \pi(s,a) \left(Q^\pi(s,a) - b(s)\right) \quad ,$$

where $b(s) : S \to \mathbb{R}$ is an arbitrary baseline function. Here S denotes the state space, J the performance measure, Q^π and μ^π are the state-value function and the stationary state distribution when following policy π, respectively, and $\boldsymbol{\theta}$ are the parameters of the policy π.

$\sum_a \nabla_{\boldsymbol{\theta}} \pi(s,a) b(s)$ acts as a control variate. Prove that $\mathbb{E}\left[\sum_a \nabla_{\boldsymbol{\theta}} \pi(s,a) b(s)\right] = 0$.

1.2 Lunar

In this assignment, we will use the basic policy gradient algorithm REINFORCE to safely land a lunar landing module on the moon.

Please have a look at the notebook `LunarLanderAssignment.ipynb`, which implements the basic REINFORCE algorithm Williams (1992). Do not start too late with this part of the assignment, because it may not be straight-forward to get the notebook running on your system. The *Lunar Lander* RL environment is part of the Gymnasium framework, see `https://gymnasium.farama.org/environments/box2d/lunar_lander/` for details of the RL task. You may need to install Gymnasium via `pip install gymnasium[box2d]`. You may also need to install some other packages (e.g., SWIG, `https://www.swig.org`).

The notebook uses a softmax policy (see lecture slide "Example: Softmax policy"). The preferences of the softmax policy are just linear functions of the features. It is not trivially clear that this policy class is powerful enough to get some reasonable results, but it gives you clearly better than random policies (we will see advanced algorithms that learn better solutions later). However, running the RL in the notebook will not always give you a nice solution with the given hyperparameters, you may need to rerun it to get a good policy.

The algorithm requires the score function, that is the gradient $\nabla \ln \pi(s, a)$ of the policy w.r.t. the policy parameters.

Your task is to compute the score function and to implement a membership function

```
def gradient_log_pi(self, s, a):
    """
        Computes the gradient of the logarithm of the policy
        :param s: state feature vector
        :param a: action
        :return: gradient of the logarithm of the policy
    """
```

in the class `Softmax_policy` that computes the score function analytically. This can be done in less than additional 10 lines.

You can check if your are on the right track using the membership function `gradient_log_pi_test`, which approximates the gradient numerically.

Deliverables:

1. Derive the analytical expression for the score function of the softmax policy in the report.
2. Show the code for the implementation of the analytical score function `gradient_log_pi(self, s, a)` in the report.
3. Briefly describe how you verified that your implementation is correct.

2 Improved Parametrization of UCB1 (0 points) [Yevgeny] (Optional, but highly recommended)

Solve Exercise 5.5 Part 1 in Yevgeny's lecture notes.

3 Introduction of New Products (25 points) [Yevgeny]

Solve Exercise 5.6 in Yevgeny's lecture notes.

4 Empirical comparison of FTL and Hedge (25 points) [Yevgeny]

Solve Exercise 5.8 in Yevgeny's lecture notes.

References

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.