ULiège
Prof. Damien Ernst

INFO8003-1
Optimal decision making
for complex problems

Project

Inverted Double Pendulum : Searching High-Quality Policies to Control an Unstable Physical System.

1 Implementation and Deliverables

You need to deliver (i) your cleaned and well documented source code used for this project and (ii) a report which is outlined accordingly of the project description. Your report also need to explain possible improvements of your approach. Please note that both the source code and the report are mandatory, and that you need to implement the relevant reinforcement learning algorithms *yourself*.

2 Domain (6 points)

The Double Inverted Pendulum is illustrated by Figure 1. The agent interacts with the environment by applying an horizontal force on a cart in which a two-link pendulum is mounted in its center and are initially set to (nearly) upright position above the cart. The goal of the agent is to steadily keep the two-link pendulum in this position. The agent continuously receives a feedback which corresponds to (i) a penalty proportional to the distance between the upright position and the current state, (ii) a positive bonus as an incentive to keep interacting with the environment and (iii) a penalty proportional to the velocity of the cart. A terminal state of the environment is reached when the distance between the upright and the current state is above a given thresold.

As first task, provide a formalization of the Double Inverted Pendulum environment (excepted the dynamics) based on this *source code*. Provide also characterizations of this environment (e.g., deterministic or stochastic...). The format of your formalization should be inspired from the paper [1]. You should also install all needed dependencies to be able to exploit the environment source code.

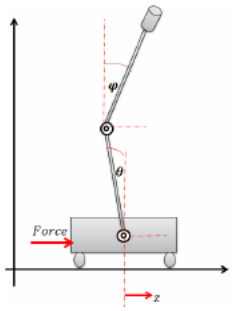

Figure 1: Double Inverted Pendulum environment.

3 Policy Search Techniques (14 points)

Design your own policy search technique based on the reinforcement learning literature. Provide references upon which your approach is based. Your algorithm has to be able to learn both discrete and continuous policies. Design an experimental protocol which assesses the performance of your algorithm with both policies, compared to a classical algorithm seen during the lectures (e.g., FQI with ensemble of trees). Display the performance of your policies at the end of each episode in terms of expected discounted cumulative reward. You may propose any other metrics you want that are relevant with your approach.

See below a few references to direct policy search techniques.

References

[1] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(Apr):503–556, 2005.

[2] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018.

[3] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[4] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015.

[5] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[6] Yuhuai Wu, Elman Mansimov, Shun Liao, Roger B. Grosse, and Jimmy Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. *CoRR*, abs/1708.05144, 2017.