University of Liège

INFO8003-1 - Optimal decision making for complex problems Searching High-Quality Policies to Control an Unstable Physical System

2022 - 2023

Aurélien NOIRAULT s228267

Benoît LU, s141188

Mai 2023

1 Introduction

2 Domain

In this project, we consider different environments with continuous action space to evaluate our implementations' performances. The two considered environments are Inverted Pendulum and Inverted Double Pendulum. We will consider the implementation provided by the gymnasium Python library.

2.1 Inverted Pendulum

In this environment, we aim to balance a pole placed on the top of a cart. To do so, we must applythe right amount of force to the cart.

- State Space : $S = (pos, ang, linVel, angVel)$ with
  - pos : the position of the cart along the axe
  - ang : the angle of the pole on the cart
  - $linVel$: the linear velocity of the cart
  - $angVel$: the angular velocity of the pole
- Action Space : $A = -3.0, 3.0$
- Reward : 1 while the episode did not stop
- End of an Episode : the episode ends if $|ang| > 0.2rad$
- Starting state : (0.0, 0.0, 0.0, 0.0, with a uniform random noise of range $[-0.1, 0.1]$

2.2 Inverted Double Pendulum

This environment is a variant of the previous one. In this one, a second pole is added on top of the first one. Similarly, the goal is to balance the pole of the top (the second pole).

- State Space : $A' = (pos, sin_1, sin_2, cos_1, cos_2, linVel, angVel, force_1, force_2, force_3)$ with :
  - pos : the position of the cart along the axe
  - sin_1 : sine of the angle of the first pole
  - sin_2 : sine of the angle of the second pole
  - cos_1 : cos of the angle of the first pole
  - cos_2 : cos of the angle of the second pole
  - $linVel$: the linear velocity of the cart
  - $angVel$: the angular velocity of the angle between the poles

  - $force_1$: constraint force -1
  - $force_2$: constraint force -2
  - $force_3$: constraint force -3
- Action space : $A = [-1.0, 1.0]$
- Reward : $alive_bonus - distance_penalty - velocity_penalty$ with
  - $alive\_bonus$: 10 while the second pole is upward
  - $distance\_penalty$: by how much the tip of the second pendulum move
  - penalty for moving too fast
- End of an episode : if the second pole is no longer upward
- Starting state : (0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0) with a uniform random noise of range $[-0.1, 0.1]$

3 Comparison of the models

We have developed three models : FittedQ, REINFORCE and PPO Clip. Each of them are details like so:

- FittedQ is implemented using Extremely Random Trees with 10 estimators as it has been used in the previous project and provided good result in comparison with the Multilayer Perceptron (MLP) and the Linear Regression. While MLP provides better fitting, its training time was too time-consuming and therefore was left aside after several attempts and debugging phases.
- REINFORCE was developped as a Policy Gradient Algorithm because it was directly mentioned in the project statement. We have used a MLP of 3 dense layers with 2 separate output layers. It is defined as the class `NormalDistribParam` in the file `utils.py`.
- PPO Clip was picked as an actor critic and it was also implemented using the same neural network as REINFORCE, however it is not fully implemented. To be explicit, it has run several times but its metrics (number of steps and rewards) are collapsing after a few iterations.

Because of organization issues and constraints of times we cannot provide you with a plot for FittedQ, however we can mention that using a dataset of 20.000 one-step transition and 20 iterations led to episodes of 30 to 160 steps in the inverted pendulum case and of 10-30 steps in the double pendulum case.
REINFORCE gave very good results in the inverted pendulum case closing up to 1000 average step.

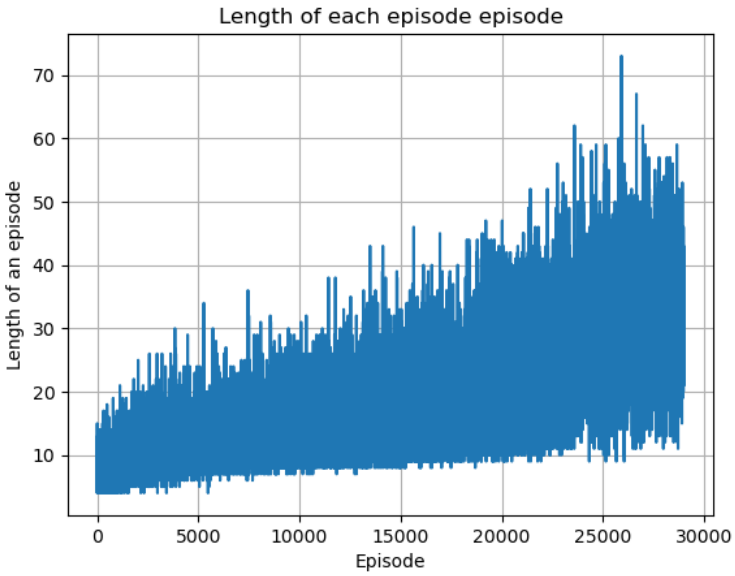

Figure 1: Double Pendulum length of episode using REINFORCE

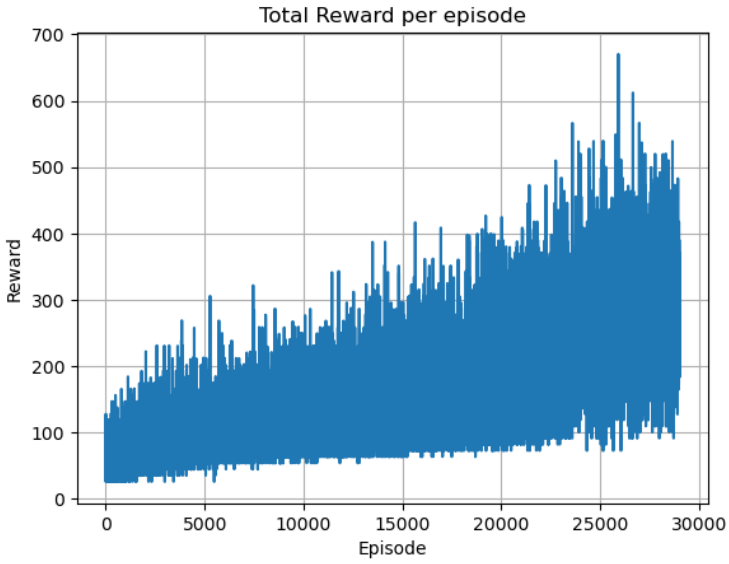

Figure 2: Double Pendulum average rewards using REINFORCE

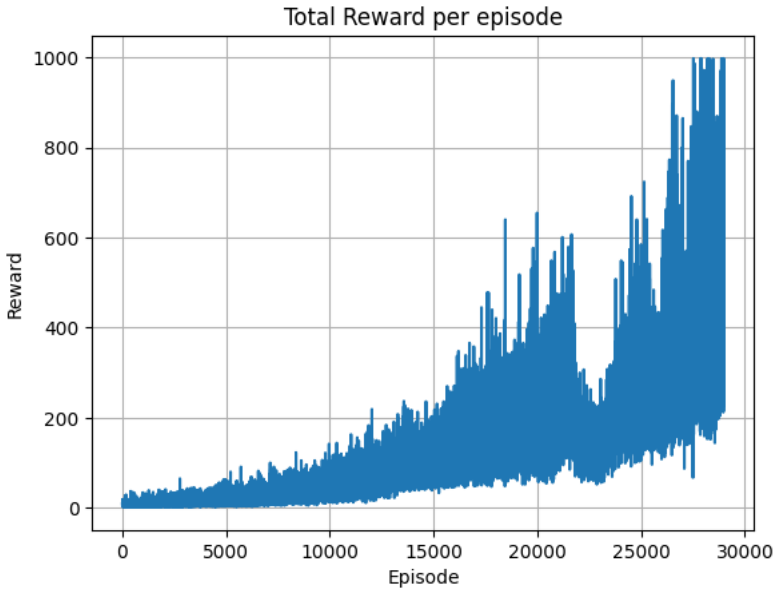

Figure 3: Inverted Pendulum average rewards using REINFORCE

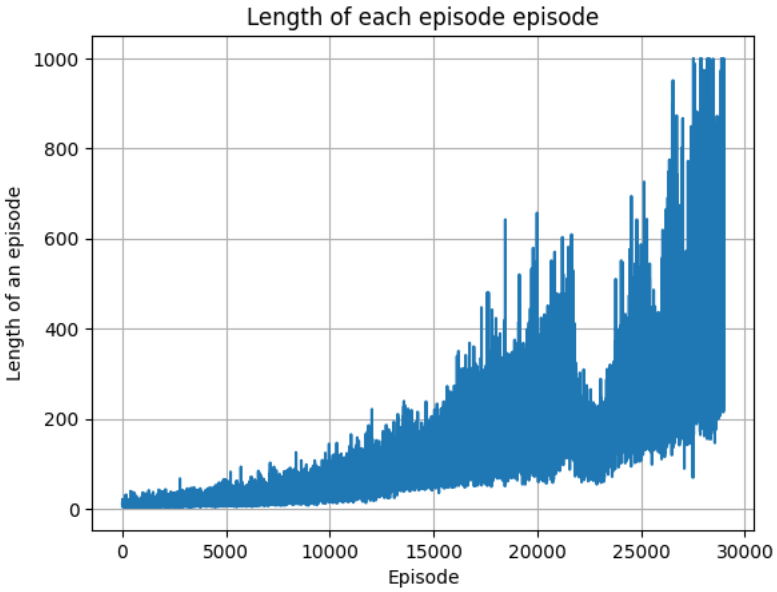

Figure 4: Inverted Pendulum length of episode using REINFORCE