FTML practical session 11: 2023/06/03

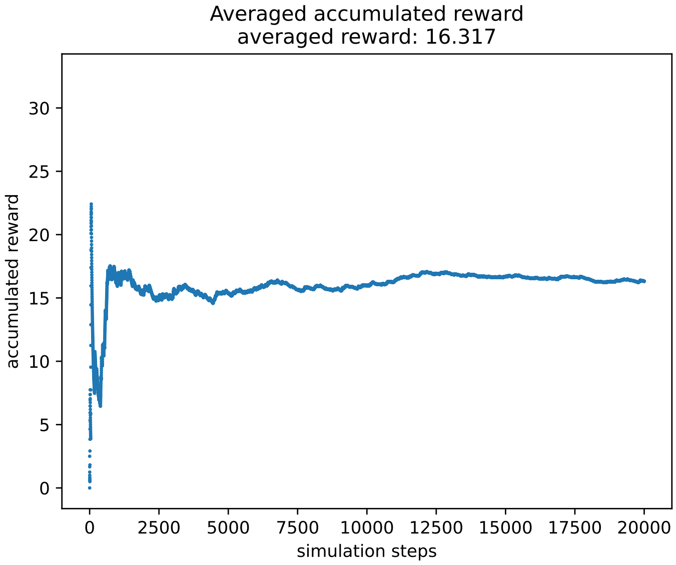


TABLE DES MATIÈRES

INTRODUCTION

1 EXPLOITATION/EXPLORATION COMPROMISE

In this exercise we work with the notion of stochastic policy, applied to a simple agent in a 1-dimensional world.

1.1 Setting

We consider a one dimensional world, with 8 possible positions, as defined in the folder **project/exercise_4**. An agent lives in this world, and can perform one of 3 actions at each time step : stay at its position, move right or move left.

In this folder, you can find 3 files :

- **simulation.py** is the main file that you can run to evaluate a policy.
- **agent.py** defines the Agent class. This simple agent only has two attributes.
  - **position** : its position
  - **known_rewards** : represents the knowledge of the agent about the rewards in the worlds (see below)
- **default_policy.py** implements a default policy that consists in always going left.

Some rewards are placed in this world randomly, and are randomly updated perdiodically, at a fixed frequency. This means that a good agent should update its policy periodically as well and adapt to the new rewards. The agent knows about a reward in the world if its position has been on the same position as the reward, but each time the rewards are updated, the agents forgets all this knowledge, as implemented line 46 in **simulation.py**.

simulation.py computes the statistical amount of reward obtained by the agent and plots the evolution of this quantity in **images/**. As you can see in the **images/** folder, the average accumulated reward with the default policy is around 16, with a little bit of variance.

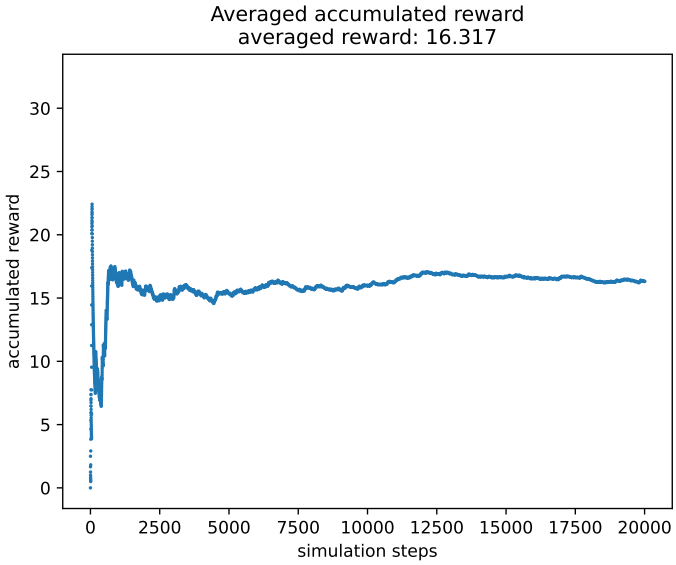


FIGURE 1 – Convergence of the average reward obtained by the agent with the default policy.

1.2 Objective

Write a different, **stochastic** policy in a separate file named **<group_name>_policy.py** that achieves a better performance than the default policy. **<group_name>_** should be the name of one of the students of your group, or any name that identifies your group.

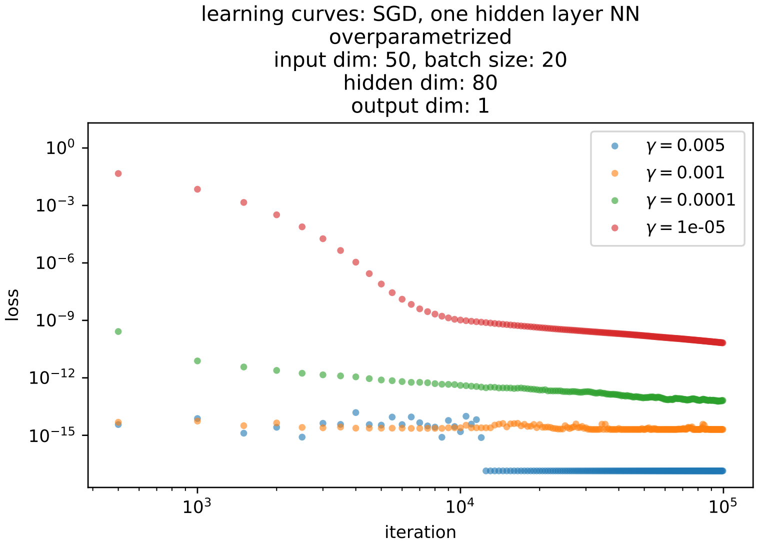
You will need to

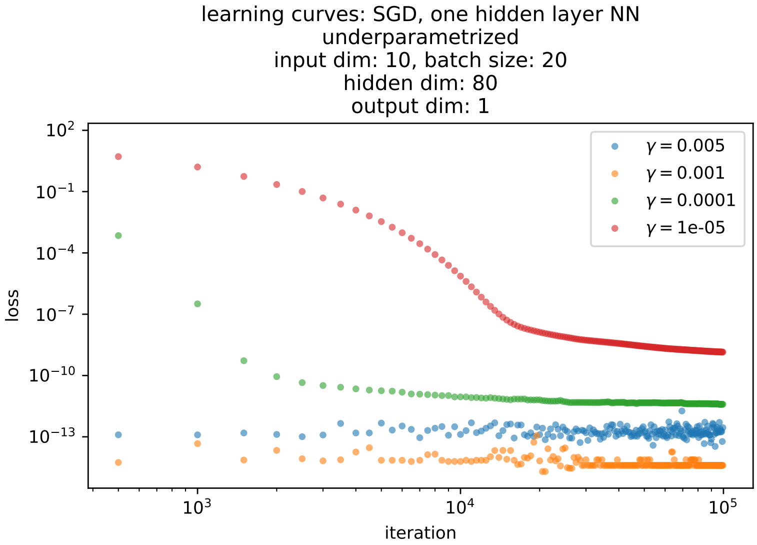
— import you policy in **simulation.py**

— replace line 51 by a line that calls your policy instead of the default policy.

Your objective is to obtain a final average reward of at least 20.

2 OVERPARAMETRIZED AND UNDERPARAMETRIZED REGIMES


learning curves: SGD, one hidden layer NN
overparametrized
input dim: 50, batch size: 20
hidden dim: 80
output dim: 1
10^{0}
10^{-3}
10^{-6}
10^{-9}
10^{-12}
10^{-15}
loss
10^{3}
10^{4}
10^{5}
iteration
$\gamma = 0.005$
$\gamma = 0.001$
$\gamma = 0.0001$
$\gamma =$ 1e-05


learning curves: SGD, one hidden layer NN
underparametrized
input dim: 10, batch size: 20
hidden dim: 80
output dim: 1
10^{2}
10^{-1}
10^{-4}
10^{-7}
10^{-10}
10^{-13}
loss
10^{3}
10^{4}
10^{5}
iteration
$\gamma = 0.005$
$\gamma = 0.001$
$\gamma = 0.0001$
$\gamma =$ 1e-05