FTML practical session 11: 2023/06/03

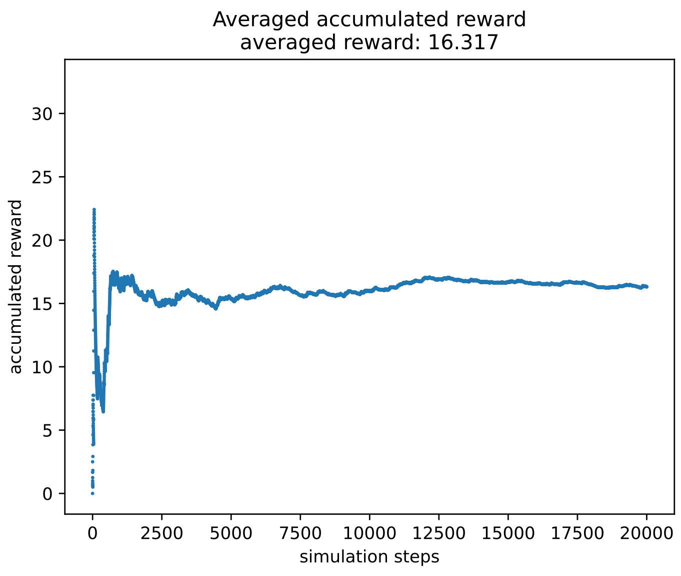

TABLE DES MATIÈRES

INTRODUCTION

1 EXPLOITATION/EXPLORATION COMPROMISE

Reinforcement learning consists in learning a **policy** : a function that computes the best action that an agent shoule take in a given situation, in order to maximize an aggregation of rewards.
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 **exploration_exploitation**. 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 :

- **main.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 **main.py**.

main.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 that consists in always going left is around 16, with a little bit of variance.

1.2 Objective

However, it is possible to do way better (> 45 !) by using a smarter policy.

Write different, **stochastic** policies that achieve better performances than the default policy.

You will need to

- import you policy in **main.py**
- replace line 51 by a line that calls your policy instead of the default policy.

1.3 Ressources

Many approaches are valid in order to write a good policy for the agent. Here are some suggestions :

- try the classical ϵ-greedy policies. Explore with probability ϵ, exploit with probablity $1 - \epsilon$.

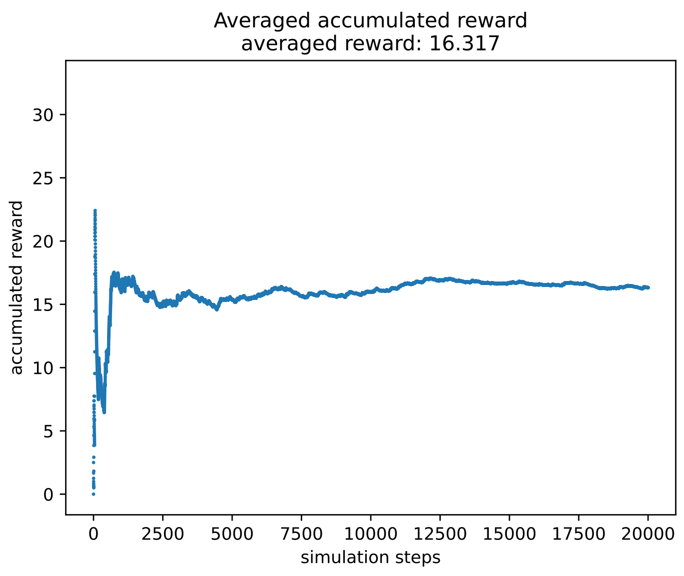

Figure 1 – Convergence of the average reward obtained by the agent with the default policy.

— propose your own, intuitive heuristics.

— use methods inspired by reinforcement learning approaches, like value iteration. See sections I.3 and I.4 of the classical book "Reinforcement Learning : an introduction".
`http://incompleteideas.net/book/RLbook2020.pdf`

2 OVERPARAMETRIZED AND UNDERPARAMETRIZED REGIMES

During the lectures, we have introduced the notions of **overparametrized** and **underparametrized** regimes. When doing empirical risk minimization, we say that we are overparametrized if there exists an estimator in the considered space of estimators, with an empirical risk equal to 0. We say that this estimator **interpolates** the dataset. Otherwise, we are underparametrized. It is experimentally observed that some overparametrized neural networks have a tendency to generalize well and do little overfitting.

It seems that the distinction between the two regimes has important consequences on the behavior of SGD algorithms applied to the optmization of neural networks. Specifically, the oscillations of SGD are different between the two regimes, and this depends on the magnitude of the learning rate. Namely, the oscillations can be way smaller in the case of the overparametrized regime.

Generate two datasets : an underparametrized one and an overparametrized one for the class of one hidden layer neural networks. Technically this means that you can generate the data with neural networks. Then, optimize another neural network, with SGD, for different learning rates, in order to observe something like the following figures 3 and 2.

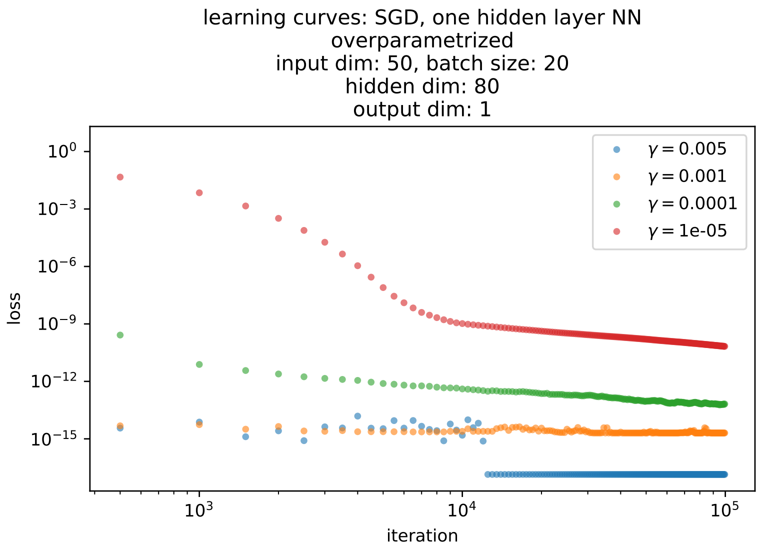

FIGURE 2 – Learning in the overparametrized regime.

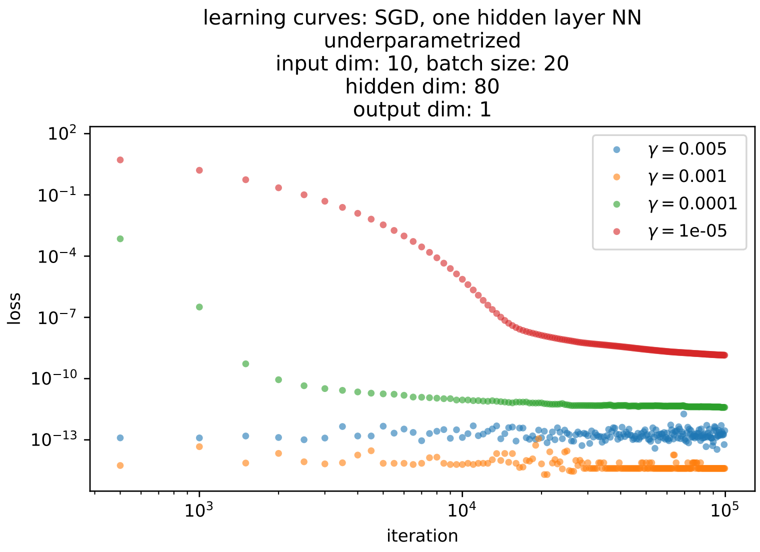

FIGURE 3 – Learning in the underparametrized regime.