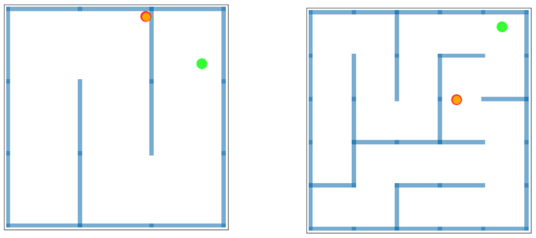
Hard Exploration for Maze Solving

Achraf AZIZE
Ecole Polytechnique
Palaiseau 91128
achraf.azize@polytechnique.edu

Abstract

In this work, we will try to solve the hard exploration problem in a maze environment using Curriculum Learning. We will be using Soft Actor Critic with Hindsight Experience Replay (SAC-HER) as a baseline. The experiments show that with a well-designed curriculum, the learning could be accelerated, even in very sparse hard task environments, and the final performance improved.



(a) Maze-Simple-v0 (b) Maze-Medium-v0

Figure 1: The basic Maze Environments

1. Introduction

Exploration in Reinforcement Learning is a big topic, very well studied in the literature. If in simple tabular RL, classical exploration techniques like epsilon-greedy, Upper Confidence Bounds or Thompson Sampling may be good, exploration in deep RL scenarios is still a difficult challenge. In this case, we can identify two classes of exploration approaches: Intrinsic Rewards as Exploration Bonuses and Memory-based Exploration.

In the Intrinsic Rewards as Exploration Bonuses, the idea is to augment the environment reward with an additional bonus reward to encourage extra exploration. This intrinsic reward is somewhat inspired by curiosity and surprise in psychology [4].

Memory-based Exploration tries to overcome the limitations of Reward-based exploration by relying on external memory. Many new approaches was proposed lately in this domain from episodic memory [2] or direct exporation [5].

In this work, we will try to implement another solution to hard exploration problems: Curriculum Learning [3]. An agent could learn much better when the examples are not randomly presented, but organized in a meaningful order which illustrates gradually more concepts, and gradually more complex ones. Here, we will present how to build such a curriculum in the context of maze solving. The experiments show that curriculum learning has both an effect on the speed of convergence of the training process, but also on the quality of the local minima obtained (curriculum learning can be seen as a particular form of continuation method).

2. The Basic Maze Environments

2.1. Overview

In this work, we will focus on two Maze environments: Maze-Simple-v0 (3×3 grid) and Maze-Medium-v0 (5×5 grid) from Figure 1. For both environments, the objective is to find the optimal movement policy which takes an agent from a starting point to the destination in a collision free path. It is a hard exploration task since the agent is only getting a reward when reaching the goal. The goal of this project is to speed up learning in those two environments by using curriculum learning.

Both maze environment are created using the depth-first algorithm [1] , with starting position and goal position generated randomly in the grid. Consequently, we can infer two interesting properties about this environments, that we will use after to build our curriculum.

A $n \times n$ grid maze environment has:

- $(n-1)^2$ internal obstacles
- The mean distance between the starting and goal positions is 0.527 (approximately)

[1]Christian Hill https://scipython.com/blog/making-a-maze/

2.2. MDP Formulation

In this section, we will use the same formalism introduced by Strudel *et al.* [7]. The state space $\mathcal{S}$ is the configuration space of the robot $\mathcal{C}$, $\mathcal{A}$ is the space of valid velocities at the current configuration and $\mathcal{O}$ is a representation of the workspace along with the goal configuration g. Given a robot configuration $q \in \mathcal{C}_{free}$ and v an admissible velocity vector, we denote $q(v)$ as the configuration reached by applying the velocity vector v to the robot for a fixed time. As the robot can hit into obstacles, we consider $q_{free}(v)$ which returns the last collision free configuration on the path connecting q to $q(v)$. Then the dynamics p is defined as $p(q, v) = q_{free}(v)$.

We define a reward function as follows. Given a goal configuration $g \in \mathcal{C}$, we define $r_{velocity}(q, v) = -\|v\|_2$ and

$$r_{task}(q,v,g) = \begin{cases} r_{goal} & \text{if } \|q_{free}(v) - g\| \leq \varepsilon, \\ r_{free} & \text{if } [q, q(v)] \subset \mathcal{C}_{free}, \\ r_{collision} & \text{else.} \end{cases} \quad (1)$$

with $r_{goal} > 0$, $r_{free} < 0$ and $r_{collision} < 0$. The reward function is then defined as $r(q,v,g) = r_{velocity}(q,v) + r_{task}(q,v,g)$. r_{task} rewards actions reaching g and penalizes actions which lead to a collision. Given two collision-free paths leading to the goal, the total reward $r(q, v, g)$ is highest for the shortest path. Maximizing the reward enables the policy path to be collision free and as short as possible.

The reward r_{task} defined above is sparse with respect to the goal: it is only positive if the agent reaches the goal during one episode, which may have a low probability in challenging environments.

2.3. Training

Throughout this work (and inside every training loop of the curriculum defined in Section 3.1), we will use the exact SAC+HER training implementation used by Strudel *et al.* [7].

The policy π and the Q-function are implemented as separate networks and trained jointly with alternate optimization following the actor-critic [6] scheme. We use a PointNet based policy with 3 hidden layers of size 256. The Q-function has a similar structure and the action is concatenated to each point of the point cloud.

Hindsight Experience Replay (HER) [1] is a technique to improve the sample efficiency of off-policy RL algorithms in the sparse and goal-conditioned setting. After collecting one rollout $s_0, ..., s_T$ which may or may not have reached the goal g, it consists in sampling one of the states as the new goal g' for this rollout. The rollout may not have reached g but in hindsight, it can be considered as a successful rollout to reach g'. HER can be seen as a form of implicit curriculum learning which accelerates the learning of goal-conditioned policies.

3. Curriculum Learning for maze solving

3.1. Curriculum Design

In this section, we will present a possible implementation of Curriculum Learning in the context of maze solving. The main intuition is that a small grid maze, with few obstacles and a small distance between the starting position and the goal should be an easier task to learn than a task with more obstacles or bigger distance. We will validate experimentally this hypothesis in Section 4.2 . The complexity of a task in a fixed maze grid is hence determined by the number of obstacles and the distance between start and finish. Since in Curriculum Learning, the idea is to present the tasks from easy to hard, we will start our curriculum from a very small distance (0.2 in the example from fig2) and no obstacles, then gradually add obstacles or/and distance. The way we chose to do it in a $n \times n$ gird is as follow:

- We start from the easiest task (smallest distance, with no obstacles), and we train our SAC+HER algorithm as explained in Section 2.3 for N epochs
- We evaluate the success rate by taking the average of 100 realisations. If the success rate is bigger than a success_to_next threshold, we pass the stage and move on to the next one, otherwise, we loop in the same stage until the success rate exceeds the threshold.
- The next stage is defined from the previous stage by adding more obstacles first (+ n_obstacles_step).
- If the number of obstacles reaches its max value ($(n-1)^2$ obstacles in a $n \times n$ grid maze built using the depth first algorithm), the next stage would be to add more distance (distance_step) but with 0 obstacles.

A detailed example in the case of a 3×3 maze curriculum is shown in the figure 2.

One simpler variation of this curriculum is to only train for N epochs in each stage, without having to exceed the success_to_next threshold to pass (which is equivalent to taking success_to_next = 0).

3.2. Curriculum Hyperparameters

The curriculum defined in Section 3.1 has 4 hyperparameters that need to be fixed:

- success_to_next: which is the minimum success rate to achieve to pass a stage
- N_epochs: the number of epochs to train in every stage before evaluating the success rate

- n_obstacles_step: the number of obstacles we add every stage
- distance_step: the distance step we add when the n_obstacles is maximum

Based on the remarks of Section 2.1, we can take for a $n \times n$ grid:

- $d \in [0.2, 0.8]$
- n_obstacles $\in \{0, n-1, 2(n-1), \ldots, (n-1)^2\}$

which means we start from $d = 0.2$ and n_obstacles $= 0$, with a n_obstacles_step $= n - 1$ and a distance_step $= 0.2$

3.3. Implementation Details

The code used in all experiments is based on the code from Strudel *et al.* [7], with some small modifications.

First, to make it easier to change the distance and the number of obstacles in a maze, we crated a new environment that takes this arguments directly (figure 3). Another modification is the ability to transfer the models weight directly between the different stages. To do so, we just add a –resume-from argument that loads the networks paramaters from the '.pkl' file. We finally use both modifications in the file curriculum.py, where the curriculum is defined. The complete modified code could be found here.

4. Experiments and results

4.1. Baseline

In this experiment, we only run the baseline algorithm SAC+HER from Strudel *et al.* [7] in Maze-Simple-v0 and Maze-Medium-v0. The results are shown in Figure 4. We could say that the agent is learning in Maze-Simple-v0, however it is hardly learning anything in the Maze-Medium-v0. We even run for longer epochs (1500) for Maze-Medium-v0, and the results are still looking bad (figure 5). That's what motivates using other techniques like Curriculum Learning, because classic RL algorithms could not achieve learning in hard exploration tasks.

4.2. Validation of the curriculum

In this experiment, we take the agent that learned for 400 epochs in the Maze-Simple-v0 from experiment 1, and we evaluate its performance in every stage of the curriculum defined in Sec3.1. The table 1 resumes the results. As expected, the agent performs better in the stages with less obstacles and less distance between start and finish.

4.3. Curriculum for Maze Simple

Since we already could achieve learning in Maze-Simple-v0, we only use the easier variation of the curriculum, where we only learn for 10 epochs in every stage (equivalent to taking success_to_next = 0). After going through our curriculum, we relaunch training in the Maze-Simple-v0 to compare. The results in Figure 6 show that the learning was indeed accelerated, and the final success rate achieved better.

4.4. Curriculum for Maze Medium, with transfer learning

In this first attempt to apply Curriculum Learning for Maze-Medium-v0, we first tried to directly transfer the weights from the agent that learned with curriculum learning in Maze-Simple-v0 from the previous experiment. Unlike the curriculum used for Maze Simple, this time we use the version of curriculum with success_to_next = 0.3, because otherwise (success_to_next = 0) the agent still doesn't achieve learning. However, after 1000 epochs, the agent was still stuck in the first stage (n_obstacles = 0 and distance = 0.2). This could be explained by the fact that observations from both environments are very different or that the knowledge from Maze-Simple-v0 was not transferable, because it biased the exploration. Even after changing some hyperparameters (smaller learning rate, bigger horizon), the agent was still stuck in the first stage.

4.5. Curriculum for Maze Medium, without transfer learning

Just like the previous experiment, we will use the version of curriculum with success_to_next = 0.3, but this time starting the learning from scratch (no transfer from Maze Simple). This time, after 1000 epochs, the agent successfully passed 7 stages of the curriculum. After passing the curriculum, we launch the learning in Maze-Medium-v0 to compare, and the results are shown in Figure 7. We can see that another time, the curriculum helped not only achieving a better performance, but enabling the learning in first place.

5. Conclusion

Exploration in very sparse reward environments is still a big challenge for Deep Reinforcement Learning algorithms. Designing a good curriculum could help overcome this difficulty, by both accelerating the learning and achieving better overall final performance. However, it is not always easy to design such curriculum. In the context of maze solving, the curriculum was straight forward, since the number of obstacles and distance to goal directly control how difficult a task could be. Generally, such an information is not given , which is the biggest limitation of this kind of methods. Solutions to automate the generation of such curriculumas are present in the literature, but still depend on human hand crafting at some level.

	n_obs = 0	n_obs = 2	n_obs = 4
d = 0.2	0.73	0.67	0.6
d = 0.4	0.7	0.65	0.45
d = 0.6	0.4	0.27	0.29
d = 0.8	0.2	0.16	0.12

Table 1: Validating the curriculum design for Maze-Simple-v0 by evaluation the success_rate of the trained agent from Maze-Simple-v0

References

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay, 2018. 2

[2] Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, Andew Bolt, and Charles Blundell. Never give up: Learning directed exploration strategies, 2020. 1

[3] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning, 2009. 1

[4] Cédric Colas, Tristan Karch, Olivier Sigaud, and Pierre-Yves Oudeyer. Intrinsically motivated goal-conditioned reinforcement learning: a short survey, 2020. 1

[5] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Go-explore: a new approach for hard-exploration problems, 2020. 1

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

[7] Robin Strudel, Ricardo Garcia, Justin Carpentier, Jean-Paul Laumond, Ivan Laptev, and Cordelia Schmid. Learning obstacle representations for neural motion planning, 2020. 2, 3

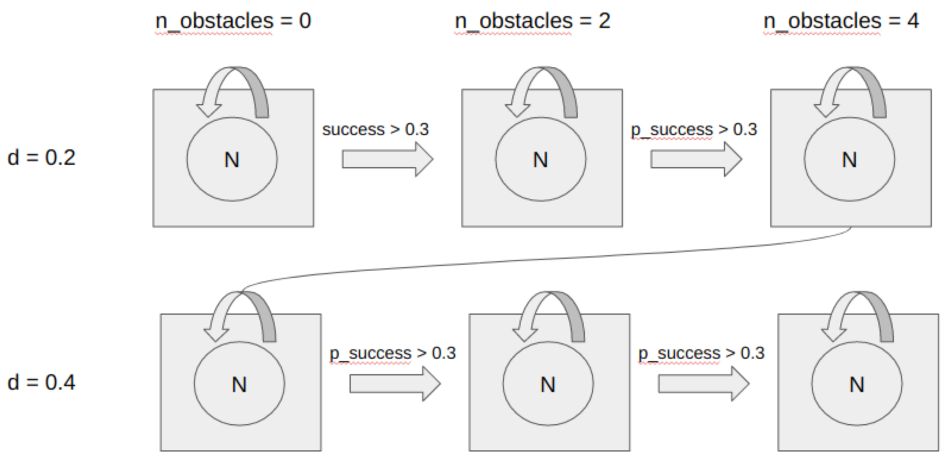
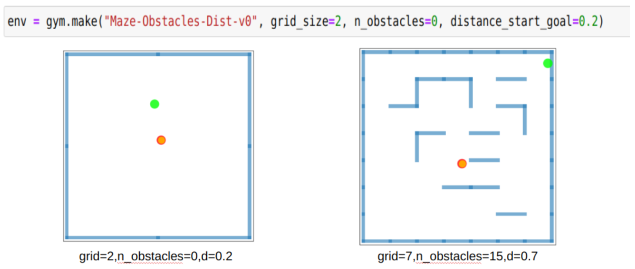
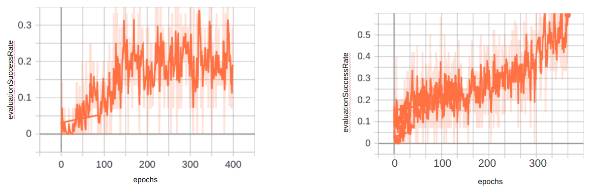


Figure 2: Example of a Maze Curriculum for Maze-Simple-v0



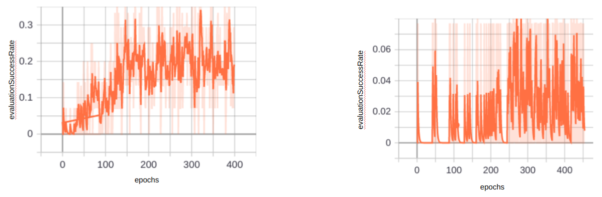
```
env = gym.make("Maze-Obstacles-Dist-v0", grid_size=2, n_obstacles=0, distance_start_goal=0.2)
```

Figure 3: The new Maze-Obstacles-Dist-v0 environment



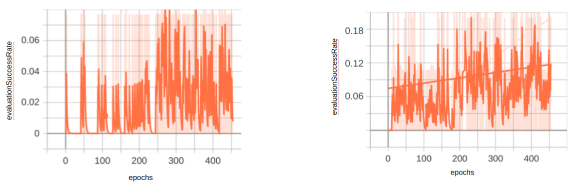
(a) Without Curriculum (b) With Curriculum

Figure 6: Results of using Curriculum Learning for Maze-Simple-v0 (evaluation_success_rate per epoch)



(a) Maze-Simple-v0 (b) Maze-Medium-v0

Figure 4: Results of Training the baseline (SAC+HER) for Maze-Simple-v0 and Maze-Medium-v0 (evaluation_success_rate per epoch)



(a) Without Curriculum (b) With Curriculum

Figure 7: Results of using Curriculum Learning for Maze-Medium-v0 (evaluation_success_rate per epoch)

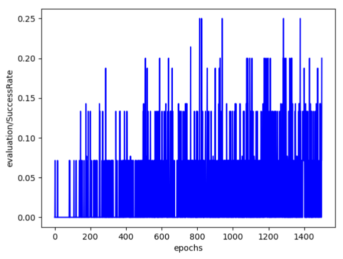


Figure 5: Results of training the baseline (SAC+HER) in Maze-Medium-v0 for 1500 epochs (evaluation_success_rate per epoch)