Lab 7
Time to Explore: DQN with RND and ICM

April, 2025

1 Goal of the Lab

In this lab you will implement two *state of the ar*t methods of exploration for sparse reinforcement learning tasks. **Notice that this lab has to be delivered through the *racó* for evaluation.**

2 Introduction

As you have learned from the lecture, for sparse reward problems where the agent has to thoroughly explore the state space to get a glimpse of what his task is (sparse reward setting), exploration techniques like ϵ-greedy are often not enough to "stumble over" the desired goal conditions.

This is already the case for relatively simple problems like the `MountainCar`[1] environment. In this environment, the agent has to strategically accelerate a car back and forth to escape from a mountain valley and arrive at the top of the mountain at the right (depicted in Figure 1). In the default implementation, the reward function is $R_a(s, s') = -1$ for every step until the episode is finished, i.e. after a maximum of 200 time steps or the agent arrived at the goal. As it turns out, using value-function approximators like DQN, this reward definition is not as sparse as it seems on first impressions (try to explain why this might be the reason. Think about how the value estimates of visited vs. not-visited states change over the course of training.). To convert the `MountainCar` environment into a genuinely challenging sparse-reward problem, we wrote an environment wrapper inside env.py and changed the reward function to:

no informativa i esparsa

$$R_a(s, s') = \begin{cases} 200.0, & \text{if } s' \text{ is a goal-state} \\ 0.0, & \text{otherwise} \end{cases} \tag{1}$$

[1] `https://gymnasium.farama.org/environments/classic_control/mountain_car/`

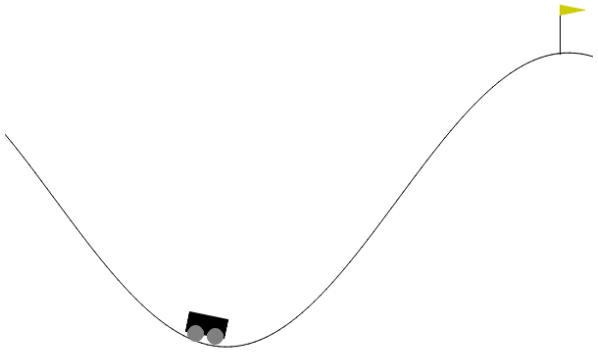
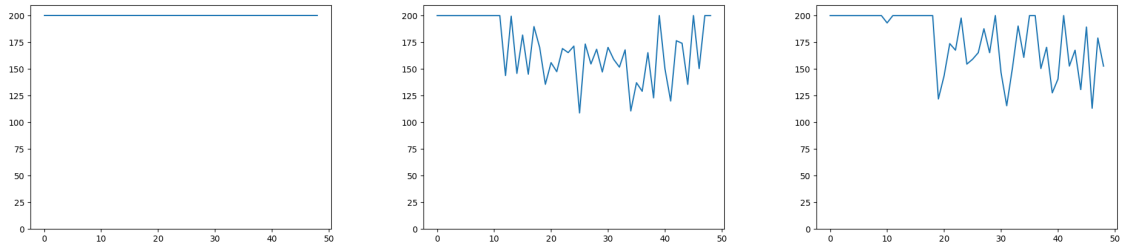


Figure 1: The `MountainCar-v0` OpenAI gym environment.



(a) Epsilon greedy (b) ICM (c) RND

Figure 2: Length of episodes for each of the three explosion methods implemented with learning experiences.

no necessitam paper, però se pot mirar per curiositat

Your task is to implement *Exploration by random network distillation* (RND) [1] and *Curiosity-driven Exploration by Self-supervised Prediction* (ICM) [2], two state of the art RL exploration methods. In the following, we give a rough sketch of the algorithms and provide a more detailed, textual pseudo-code inside the code skeletons. We also advise you to quickly look at the respective papers for more information.

In short, for both methods, the goal is to have some metric that encapsulates how often the agent has already visited a certain state. Whereas in discrete state spaces, we could just keep count of state visitations inside a table, for continuous state spaces this is harder to integrate.

RND: The idea of RND is to use a randomly initialized, fixed target network and a second predictor network, where the predictor learns to predict the outputs of the target network based on states as input. Even though the target network outputs are random, they are deterministic (stay the same) for every state input. This way, the more often the agent has already visited a certain state, the better the predictor will be able to approximate the target network outputs. In RND, the intrinsic reward is then defined as the error between the prediction and the target outputs.

ICM: The goal of ICM is to quantify the novelty of states through a forward model. Based on the current observation and the performed action, ICM predicts the next observation and uses the error between the actual next observation, and the prediction as an intrinsic exploration reward. Because learning a forward model inside the observation space has many drawbacks (think about visual observations), ICM performs the forward prediction inside a learned feature space. To do this, a feature network encodes the observations into a low-dimensional feature representation. A second network, the so-called inverse-dynamics network, is trained to predict the feature encoding of the next observation based on the feature encoding of the current observation and the performed action. This way, only action-relevant features are learned and stochastic effects outside the agent's control (think about the noisy TV problem) are filtered out. Finally, the forward-dynamics network predicts the feature vector of the next observation based on the encoding of the current observation and the performed action. Similar to RND, we use the error as an intrinsic reward.

3 Programming Tasks

1. Implement RND and ICM inside `exploration.py`. You will have to implement `calculate_loss` and `calculate_reward` functions of both exploration modules.

2. Run your implementation and verify that they are working. For reference, Figure 2 displays result plots of our solution.

4 Final work

With previous work, you can obtain up to an 8 in the grade for the lab. If you want to obtain the full mark on this lab, you should implement the following:

3. In order to compare the success of the exploring mechanism, adapt your HER implementation from the previous lab to the sparse `MountainCar` environment.

To make the comparison fair, consider 0 reward when no solution found in HER and 1 (or 200) when the agent reaches the goal state.

Also, notice that the `MountainCar` environment define states with continuous values (differently than the *bits* problem that was discrete). To deal with continuous states, we will determine that we have achieved a goal state when we are close enough to it. Close enough means that the distance (euclidean, for instance) of the current state to the goal state is smaller than a given threshold ϵ you should define.

Some Notes

- It can happen that sometimes the methods do not converge. This is due to the fact that these methods are highly dependent on good hyper-parameters. For example, if the RND predictor is too powerful and learns to fast, the intrinsic reward becomes non-informative very quickly. This is equally true for ICM. We tried our best to find reliable hyperparameters, but RL can be tricky sometimes.
- If your implementation arrives at the goal sometimes, it is probably correct.
- To highlight the exploration capabilities of RND and ICM, we only use ϵ-greedy for vanilla DQN.

References

[1] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=H1lJJnR5Ym

[2] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML'17. JMLR.org, 2017, p. 2778–2787. [Online]. Available: https://pathak22.github.io/noreward-rl/resources/icml17.pdf