# RL mini-project : Mountain Car environment

Authors : Harfouch Rayan & Tara[1] Fjellman

Spring 2024

## 1 Introduction

To be written after the rest of the report is done.

## 2 Environment

The Mountain Car environment is a classic reinforcement learning problem. The agent is placed in a valley between two hills and has to learn to reach the top of the right hill. The agent can only apply a force of -1, 0 or 1 to the car. The car has a maximum speed of 0.07 and a maximum position of 0.6. The agent receives a reward of -1 at each time step, and a reward of 0 when it reaches the top of the hill. The episode ends when the agent reaches the top of the hill or after 200 steps. The state space is continuous and has two dimensions, the position and the velocity of the car. The action space is discrete and has three dimensions, the force applied to the car. The environment is implemented in the OpenAI gym library.

## 3 Random agent

To analyse the performance of the agent, one can analyse the duration of the episodes as a function of time. Indeed, if the agent learns the task, the duration of the episodes should have a decreasing trend in time. This because at the start the agent doesn't know the task, and so the episode is truncated at 'full_ep_len=200', while as it learns it, it should manage to finish the task before reaching the step limit.

Running the the agent for 100 episodes, we obtain fig ... . Looking at it, it is clear that the agent does not lean the task, as all the durations coincide with the truncation time.

## 4 DQN

### 4.1 Vanilla version

Running 100 episodes just as for the random agent and computing their duration along with the cumulated reward per episode, we get fig ... . Looking at it we see that the behaviour is the same as for the random agent, i.e. the agent does not learn the task.

This is due to the sparsity of the reward.

### 4.2 Evolution metrics

loss per episode over the samples ? would expect reduction actually the task is getting harder while we explore more the loss trend is not expected to be decreasing before the steady state ? what is this steady state ? average cumulative reward per episode -¿ more like duration-normalised episode-cumulative reward

[1]My administrative name is Tobia, so you will find me under that name in EPFL related databases.

### 4.3 Heuristic reward

#### 4.3.1 Heuristic reward function

To alleviate the sparsity of the reward, we introduce a heuristic reward function which can be used as auxiliary reward for the DQN agent. Intuitively, we want this function to make the agent learn it should reach the top of the hill, which is placed on the right of the environnement. We therefore decide to take a heuristic function that assigns a reward if the agent is on the right of its (average) starting point. To make sure the agent tries and reach higher and higher on the hill, we decide to make this function be a growing one with the position. A simple and versatile proposal is to take

$$f(x) = A\mathbb{I}[x > \bar{x}\_0]x^n$$

for $A > 0$ some constant and a given $n \in \mathbb{N}^\star$. In practice we most often take $n = 3$ and $A = 0.1$.

#### 4.3.2 Auxiliary reward scaling

For the auxiliary reward to make sense it should be sufficiently large for it to reward, with relative scale * to be used as baseline with no sparse reward, right ?? * if too small, does not speed up training as problem is just as hard * if too large, we dominate the environnement reward * so to solve the two need to specify the same task * eg. if take degree=1 not trivial that we actually learn ?? * could even have instabilities ?? * 1.e-1 seems to work well * test 1.e0 and 2.e-2 ?? (sort of dicotomy + prior knowledge search) * test the higher reward with low degree func ? * be on the lookout for instabilities

## 5 RND reward

### 5.1 Normalisation

* of the input * we subtract the mean to get some deviance from what we know, which is what we are trying to capture in the RND method (the mean state will be mapped to 0 ??, while the deviant states will be mapped to something else) * the division by the std makes sure that this quantity is of right scale for NN input (in general would want to have the right order) * but then why not the in Q net ? can't scale the environment to gain stability ? * of the output * by default could be centred at whatever value * we center it because we want to avoid getting free reward (we want to stop giving rewards to states we have visited often, i.e. we have had time to fit) * to make sure we have control of the scale through our reward-factor parameter we make sure we are of order one before scaling * clamping is there to avoid problems in heavy tail distributions ? * as NNs are not linear * over what should we normalise ? * can use whole buffer or batch * can do it for no extra expense by storing buffer mean and std and using update rule for mean quantities $f\_n + 1(\mathbf{x}) = f\_n(\mathbf{x}\_n) + \frac{1}{n}(x\_n + 1 - x\_0)$ * they say not to compute at "few initial steps" but can wait until buffer is full * we wait for the buffer to be full to start training as we want representative input to be given (and buffer length is not huge compared to training time) * we could try and a posteriori compute the RND reward for the initial buffer but it turns out to mess up the gradients as it requires modifying variables that are used for the training of the RND net

### 5.2 Reward-factor scale

* if too big * we just keep trying new stuff and ignore the environement reward ?? * if too small * nothing happens, we continue being slow at learning the task * right size ? * crude scale : over an episode the rewards should be comparable ?? * * handling gradients * they are set to zero after each update * we use target network to only include part of the loss in the differenciation * dataloaders are usually used for training (to shuffle data and create batches) * can't add samples to it, and should not re-define it from scratch * we just keep a list instead, sampling with random indices * we only start training once the replay buffer is full * implementation of target net was done with help of chat gpt (command to import the weights from another net)