2025

Mountain Car

PROJECT CARD

SARA SOBSTYL I ALICJA BANASZEWSKA

Spis treści

Description 2

Algorithm 3

Mountain Car

Description

This environment is part of the **Classic Control** suite, which offers benchmark tasks for studying control algorithms and reinforcement learning techniques.

Goal

The goal of the MDP is to strategically accelerate the car to reach the goal state on top of the right hill.

Observation Space

Each observation is an array of two values:

Num	Observation	Min	Max	Unit
0	Car position (x-axis)	-1.2	0.6	Meters (m)
1	Car velocity	-0.07	0.07	Velocity (v)

Action Space

The agent can take one of three discrete actions:

- **0:** Accelerate left
- **1:** Coast (no acceleration)
- **2:** Accelerate right

Transition Dynamics

Given an action, the environment updates according to:

$velocity_{t+1} = velocity_t + (action - 1) * force - cos(3 * position_t) * gravity$

$position_{t+1} = position_t + velocity_{t+1}$

Where:

- **force = 0.001**
- **gravity = 0.0025**

Velocity and position are clipped to their respective ranges. Collisions with boundaries reset velocity to 0.

Reward

Each timestep incurs a reward of **-1** until the car reaches the goal. This structure encourages solving the task in as few steps as possible.

Starting State

At the beginning of an episode:

- **Position** is randomly initialized between **-0.6** and **-0.4**.
- **Velocity** starts at **0**.

Episode End

An episode terminates when:

- The car's **position ≥ 0.5** (goal achieved), or
- **200** steps have elapsed (timeout).

Algorithm

To solve this problem, we plan to implement a **Q-learning** algorithm. Q-learning is a model-free reinforcement learning technique that learns an optimal action-selection policy for any Markov Decision Process (MDP).

The agent will maintain a **Q-table** where each entry represents the expected future rewards for a given state-action pair. At each timestep:

- The agent selects an action using an exploration-exploitation strategy.
- It observes the reward and the next state.
- It updates the Q-value based on the **Bellman equation**:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max_{a'} Q(s', a') - Q(s,a)]$$

where:

- **α** is the learning rate,
- **γ** is the discount factor,
- **r** is the immediate reward,
- **s'** is the next state.

Through repeated interaction and updates, the agent learns to maximize its cumulative reward by reaching the goal efficiently.