ULiège
Prof. Damien Ernst

INFO8003-1
Optimal decision making
for complex problems

Assignment 2

Reinforcement Learning in a Continuous Domain

1 DOMAIN

We describe the domain below:

- State space: $X = \{(p, s) \in \mathbb{R}^2 | |p| \leq 1, |s| \leq 3\}$ and a *terminal state*[1].
  - A terminal state is reached if $|p_{t+1}| > 1$ or $|s_{t+1}| > 3$.
- Action space: $U = \{4, -4\}$.
- Dynamics: $\dot{p} = s$, $\dot{s} = \frac{u}{m(1+Hill'(p)^2)} - \frac{gHill'(p)}{1+Hill'(p)^2} - \frac{s^2 Hill'(p)Hill''(p)}{1+Hill'(p)^2}$, where $m = 1$, $g = 9.81$ and

$$Hill(p) = \begin{cases} p^2 + p & \text{if} \quad p < 0 \\ \frac{p}{\sqrt{1+5p^2}} & \text{otherwise.} \end{cases}$$

  - The discrete-time dynamics is obtained by discretizing the time with the time between t and $t + 1$ chosen equal to $0.100s$.
- Integration time step: 0.001.
- Reward signal:

$$r(p_t, s_t, u_t) = \begin{cases} -1 & \text{if} \quad p_{t+1} < -1 \quad \text{or} \quad |s_{t+1}| > 3 \\ 1 & \text{if} \quad p_{t+1} > 1 \quad \text{and} \quad |s_{t+1}| \leq 3 \\ 0 & \text{otherwise.} \end{cases}$$

- Discount factor: $\gamma = 0.95$.

[1] A terminal state can be seen as a regular state in which the system is stuck and for which all the future rewards obtained in the aftermath are zero.

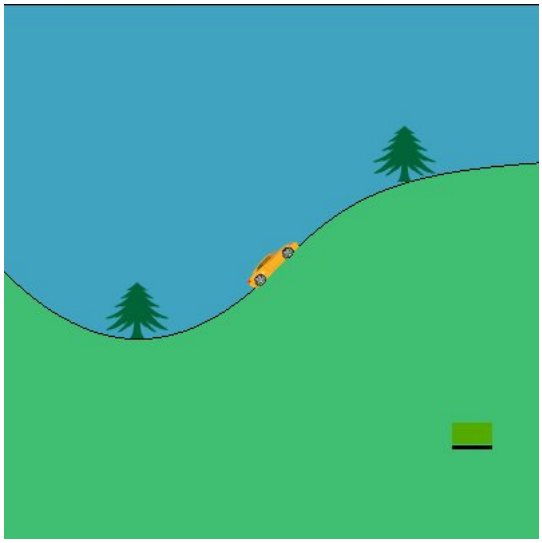

Figure 1: Display of the position $p = 0$ and the speed $s = 1$ of the car.

- Time horizon: $T \to +\infty$.

This domain is a *car on the hill* problem, and will be referred by this name from now on.

Make sure you (i) rigorously test your implementation at each stage of the assignment and (ii) store any intermediate result to avoid redundant calculations.

2 Implementation of the Domain

Implement the different components of the *car on the hill* problem. Your implementation of the dynamics should exploit the Euler integration method. Make sure your implementation handles the terminal state case. Test your results by simulating a simple policy.

3 Expected Return of a Policy in Continuous Domain

Implement a routine which estimates the expected return of a policy for the car on the hill problem. Your routine should exploit the Monte Carlo principle. Test your routine with a simple policy (e.g., always applies action $u = 4$).

4 Visualization

Implement a routine which produces a video from any *car on the hill* trajectory. Your routine needs to use the function from this *Python script* (or an equivalent implementation) which produces an image of a given state from the *car on the hill* problem.

5 FITTED-Q-ITERATION

Implement a routine which computes $\widehat{Q}_N$ for $N = 1, 2, 3 ...$ using *Fitted-Q-Iteration*. Use the following supervised learning techniques:

- Linear/logistic regression,
- Extremely Randomized Trees,
- Neural networks.
  - You need to build yourself and motivate your neural network structure.

These techniques are implemented in the *scikit-learn* and *Keras* programming libraries. Propose several strategies for generating sets of one-step system transitions that will be used in your experiments. Display $\widehat{Q}_N$ for each supervised learning algorithm. Derive the policy $\widehat{\mu}^*_N$ from $\widehat{Q}_N$. Estimate and display the expected return of $\widehat{\mu}^*_N$.

6 PARAMETRIC Q-LEARNING

Implement a routine which estimates the Q-function with *Q-learning* when a parametric approximation architecture of $Q(x, u, a^*)$ is used. Use neural networks and radial basis functions as approximation architectures. Derive the policy $\widehat{\mu}_*$ from $\widehat{Q}(x, u, a^*)$. Estimate and display the expected return of $\widehat{\mu}^*$. Design an experiment protocol to compare *FQI* and *parametric Q-learning*.