Planning Lab - Lesson 3

Markov Decision Process (MDP)

Luca Marzari and Alessandro Farinelli

University of Verona
Department of Computer Science

Contact: luca.marzari@univr.it

November 9, 2022

Start Your Working Environment

Start the previously installed (lesson 1) conda environment *planning-lab*

```
> cd Planning-Lab
> conda activate planning-lab
> jupyter notebook
```

To open the assignment navigate with your browser to: lesson_3/lesson_3_problem.ipynb

NumPy

What is it

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- *a powerful N-dimensional array object*
- *sophisticated (broadcasting) functions*
- *tools for integrating C/C++ and Fortran code*
- *useful linear algebra, Fourier transform, and random number capabilities*

What is it for

Fast array manipulation and mathematical operations. Think of it as a MATLAB like environment for Python: try to speed up the computations writing code in a vectorial fashion.

Where to find it

http://www.numpy.org

Assignments

- Your assignments for this lesson are at: *lesson_3/lesson_3_problem.ipynb*.
  You will be required to implement value iteration and policy iteration algorithms
- In the following you can find pseudocodes for such algorithms

Value Iteration

function VALUE-ITERATION(mdp, ϵ) **returns** a utility function
  inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$,
    rewards $R(s)$, discount γ
   ϵ, the maximum error allowed in the utility of any state
  local variables: U, U', vectors of utilities for states in S, initially zero
    δ, the maximum change in the utility of any state in an iteration

  repeat
   $U \leftarrow U'$; $\delta \leftarrow 0$
   for each state s **in** S **do**
    $U'[s] \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\, U[s']$
    if $|U'[s] - U[s]| > \delta$ **then** $\delta \leftarrow |U'[s] - U[s]|$
  until $\delta < \epsilon(1-\gamma)/\gamma$
  return U

Policy Iteration

```
function POLICY-ITERATION(mdp) returns a policy
  inputs: mdp, an MDP with states S, actions A(s), transition model P(s' | s, a)
  local variables: U, a vector of utilities for states in S, initially zero
                   π, a policy vector indexed by state, initially random

  repeat
     U ← POLICY-EVALUATION(π, U, mdp)
     unchanged? ← true
     for each state s in S do
        if max_{a ∈ A(s)} Σ_{s'} P(s' | s, a) U[s'] > Σ_{s'} P(s' | s, π[s]) U[s'] then do
           π[s] ← argmax_{a ∈ A(s)} Σ_{s'} P(s' | s, a) U[s']
           unchanged? ← false
  until unchanged?
  return π
```

To implement the *Policy-Evaluation* step, use the following formula:

$$U_i(s) = R(s) + \gamma \sum_{s'} P(s' \mid s, \pi_i(s)) U_i(s') .$$