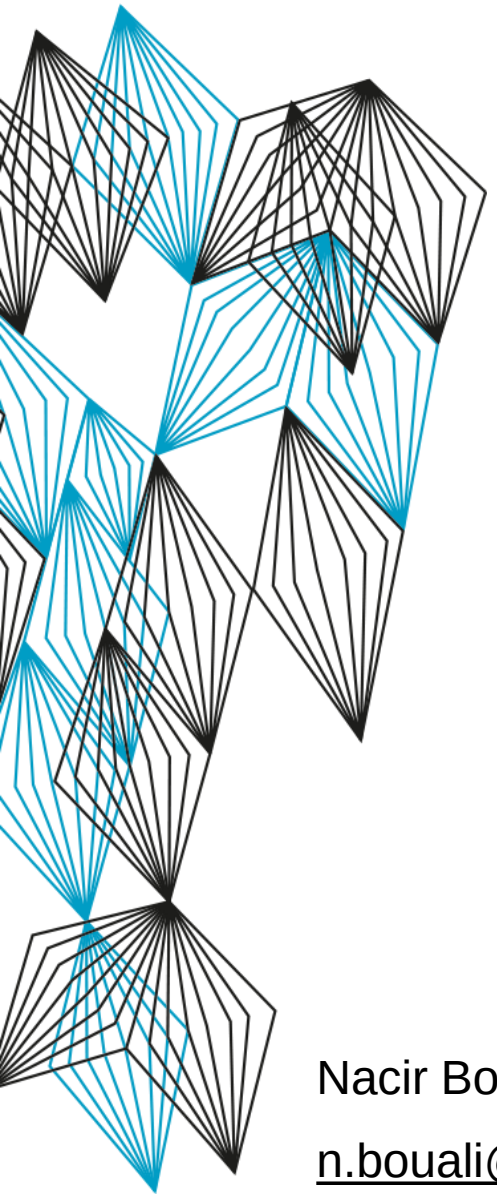


ARTIFICIAL INTELLIGENCE & CYBER SECURITY

REINFORCEMENT LEARNING

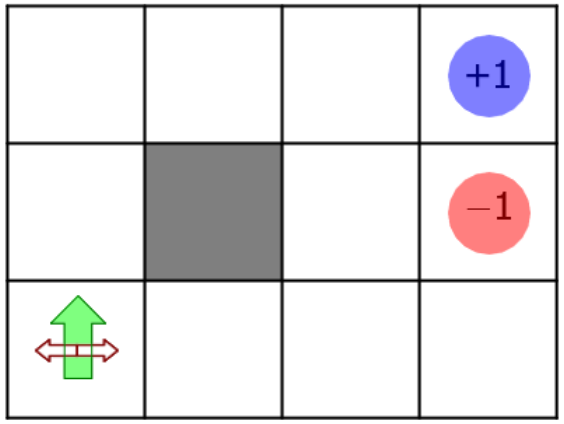
POLICY ITERATION

Nacir Bouali

n.bouali@utwente.nl

UNIVERSITY OF TWENTE.

POLICY ITERATION



- Start with random initial policy π’
- π =π’
- Compute the utilities of the states under π’

Advantages;

- Simplified Bellman equations:

$$U_i(s) = R(s) + \gamma \sum_{s'} p(s'|s, \pi_i(s))\, U_i(s')$$

(Which are linear equations and can be solved exactly for small state spaces)

- for each s in S, is there an action such that
  $[R(s) + \gamma \sum_{s' \epsilon S} P(s'|s, a) U(\pi, s')] > U(\pi, s)$
  - Then π’(s)=a otherwise no change.

POLICY ITERATION EXAMPLE

- Grid world with 4 states as in the previous example.
- Stochastic actions:
  - With probability 0.8, the intended action is executed
  - With probability 0.1, the agent moves perpendicular to the intended direction (on either side)
  - Collision with a wall: no movement
- Terminal state has a reward of +1.
- Non-terminal states have a reward of 0.
- Can we find the optimal policy?

POLICY ITERATION EXAMPLE

- Start by a random policy;
  - $\Pi(s_0)$=Left, $\Pi(s_1)$=Right, $\Pi(s_2)$=UP
- 1. $\Pi = \Pi'$
- 2. $U_i(s) = R(s) + \gamma \sum_{s'} p(s'|s, \pi_i(s))\, U_i(s')$
  - Equations
    - U_0=0+0.5(1*U_0)
    - U_1=0+0.5(0.8*U_2+0.2*U_1)
    - U_2=0+0.5(0.8*U_2+0.1*U_1+0.1*1)
    - Solutions; U_0=0; U_1=0.38; U_2=0.086

Use numpy to solve the system of equations

POLICY ITERATION EXAMPLE

- 3. for each s in S, is there an action such that $[R(s) + \gamma \sum_{s' \epsilon S} P(s'|s, a) U(\pi, s')] > U(\pi, s)$
  - S_0 ⟵right: $0+0.5(0.8U_1+0.2U_0) =0.047>0$;
  - S_0 ⟵up: $0+0.5(0.1U_1+0.9U_0) =0.0038>0$;
  - S_0 ⟵down: $0+0.5(0.1U_1+0.9U_0) =0.0038>0$;

Right returns the highest.

- Repeat the process until Π= Π’

0	0	0	+1

LINEAR ALGEBRA IN PYTHON

- $U_0=0+0.5(1*U_0)$ $0=-0.5*U_0$
- $U_1=0+0.5(0.8*U_2+0.2*U_1)$ $0=0.4*U_2-0.9*U_1$
- $U_2=0+0.5(0.8*U_2+0.1*U_1+0.1*1)$ $-0.05=0.05U_1-0.6U_2$

```
import numpy as np
V=np.array([[-0.5,0,0],[0,-0.9,0.4],[0,0.05,-0.6]])
C=np.array([0,0,-0.05])
inverseV=np.linalg.inv(V)
solution=np.dot(inverseV,C)
print(solution)
```

```
[0.         0.03846154 0.08653846]
```

$$C=\begin{pmatrix}0\\0\\-0.05\end{pmatrix}$$

$$V=\begin{pmatrix}-0.5 & 0 & 0\\0 & -0.9 & 0.4\\0 & 0.05 & -60.\end{pmatrix}$$

$$S=\begin{pmatrix}S1\\S2\\S3\end{pmatrix}=inv_v.C$$