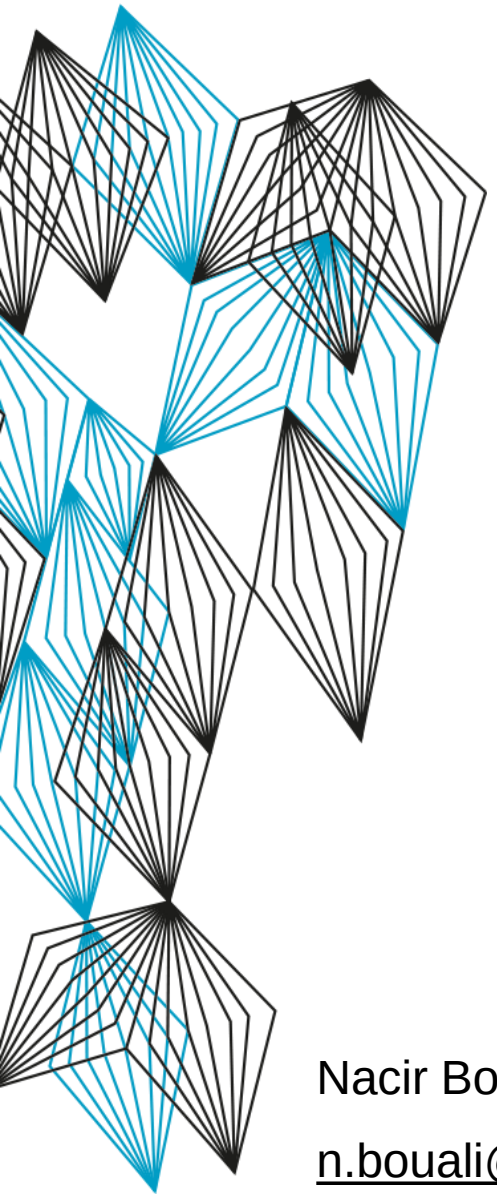


ARTIFICIAL INTELLIGENCE & CYBER SECURITY

REINFORCEMENT LEARNING
VALUE ITERATION

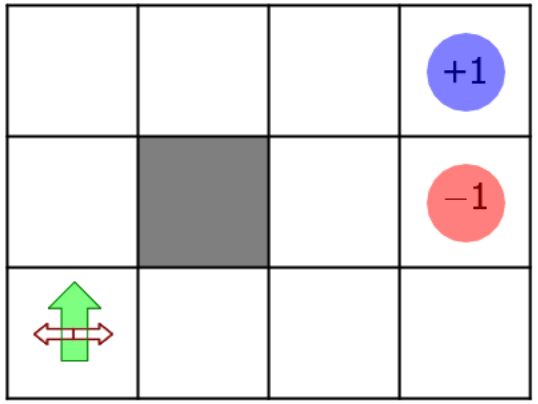
Nacir Bouali

n.bouali@utwente.nl

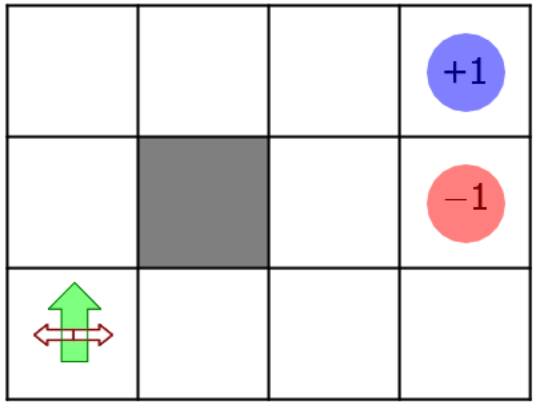
UNIVERSITY OF TWENTE.

MARKOV DECISION PROCESS

- Defined by;
  - Set of states, *S*
  - Set of actions A, and for each state s ∈ S a set of possible actions a ∈ A(s)
  - Probabilistic transition function, p(s′ | s, a)
  - Reward function R(s) (can be positive or negative)
- Sequential Decision Problem
- Fully Observable stochastic environment
- Markovian transition model
- Additive reward
- In a given MDP, what is the best course of action?
- In other words: what is the optimal policy?



MARKOV DECISION PROCESS



- Example (Grid World)
- Sequential Decision Process
- 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 states have rewards +1 and −1
- Non-terminal states have a reward of -0.04

VALUE ITERATION

- Value-iteration algorithm
  - Initialize utilities to zero, except for terminal states (reward)
  - Iteratively update (Bellman update):

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} p(s'|s,a)\, U_i(s')$$

  - Until largest utility change is smaller than a given threshold ϵ
  - Convergence:
    - Converges to a unique solution
    - Speed of convergence depends heavily on γ

0	0	0	1
0		0	−1
0	0	0	0

Iteration 0

−0.04	−0.04	0.76	1
−0.04		−0.04	−1
−0.04	−0.04	−0.04	−0.04

Iteration 1

VALUE ITERATION

- Value-iteration algorithm
  - Initialize utilities to zero, except for terminal states (reward)
  - Iteratively update (Bellman update):

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} p(s'|s,a)\, U_i(s')$$

  - Until largest utility change is smaller than a given threshold ϵ
  - Convergence:
    - Converges to a unique solution
    - Speed of convergence depends heavily on γ

−0.08	0.56	0.832	1
−0.08		0.464	−1
−0.08	−0.08	−0.08	−0.08

Iteration 2

0.392	0.738	0.89	1
−0.12		0.572	−1
−0.12	−0.12	0.315	−0.12

Iteration 3

VALUE ITERATION

- Value-iteration algorithm
  - Initialize utilities to zero, except for terminal states (reward)
  - Iteratively update (Bellman update):

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} p(s'|s,a)\, U_i(s')$$

  - Until largest utility change is smaller than a given threshold ϵ
  - Convergence:
    - Converges to a unique solution
    - Speed of convergence depends heavily on γ

0.577	0.819	0.906	1
0.25		0.629	−1
−0.16	0.188	0.394	0.1

Iteration 4

0.698	0.849	0.914	1
0.472		0.648	−1
0.162	0.313	0.492	0.185

Iteration 5

VALUE ITERATION

- Value-iteration algorithm
  - Initialize utilities to zero, except for terminal states (reward)
  - Iteratively update (Bellman update):

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} p(s'|s,a)\, U_i(s')$$

  - Until largest utility change is smaller than a given threshold ϵ
  - Convergence:
    - Converges to a unique solution
    - Speed of convergence depends heavily on γ

0.812	0.868	0.918	1
0.762		0.66	−1
0.705	0.655	0.611	0.388

After convergence

VALUE ITERATION

- Assume that we have the grid world (below), with a reward in each state of 0, and for the final state +1. We also have a discount factor $\gamma=0.5$
- Assume that the actions allowed are U,D,L,R. Succeeding 80%, and 20% are perpendicular to the direction of the action.

- What is the optimal policy?

- Value-iteration algorithm
  - Initialize utilities to zero, except for terminal states (reward)

0	0	0	+1

VALUE ITERATION

- Iteratively update (Bellman update):

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} p(s'|s,a)\, U_i(s')$$

- $U_0=R(s_0)+0.5 \max [0.8U_1+0.2U_0, 1U_0, 0.9U_0+0.1U_1, 0.9U_0+0.1U_1]=$

  $0+0.5\max[0,0,0,0]=0$

- $U1=R(S1)+0.5\max[0.8U2+0.2U1, 0.8U0+0.2U1, 0.8U1+0.1U0+0.1U2, 0.8U1+0.1U0+0.1U2]=$

  $0+0.5\max[0,0,0,0]=0$

- $U_2=R(s_2)+0.5 \max [0.8U_3+0.2U_2, 0.8U_1+0.2U_2, 0.8U_2+0.1U_3+0.1U_1, 0.8U_2+0.1U_3+0.1U_3]=$

  $0+0.5 \max[0.8+0,0,0.1,0.1]=0.4$

0	0	0	+1

Iteration 0

0	0	0.4	+1

Iteration 1

VALUE ITERATION

- Iteratively update (Bellman update):

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} p(s'|s,a)\, U_i(s')$$

- $U_0=R(s_0)+0.5 \max [0.8U_1+0.2U_0, 1U_0, 0.9U_0+0.1U_1, 0.9U_0+0.1U_1]=$
  $0+0.5\max[0,0,0,0]=0$
- $U_1=R(S_1)+0.5\max[0.8U_2+0.2U_1, 0.8U_0+0.2U_1, 0.8U_1+0.1U_0+0.1U_2,\ 0.8U_1+0.1U_0+0.1U_2]=$
  $0+0.5\max[0.32,0,0.04,0.04]=0.16$
- $U_2=R(s_2)+0.5 \max [0.8U_3+0.2U_2,\ 0.8U_1+0.2U_2, 0.8U_2+0.1U_3+0.1U_1,\ 0.8U_2+0.1U_3+0.1U_3]=$
  $0+0.5 \max[0.88,0.08,0.42,0.42]=0.44$

0	0	0.4	+1

Iteration 1

0	0.16	0.44	+1

Iteration 2

VALUE ITERATION

0	0	0.4	+1

Iteration 1

0	0.16	0.44	+1

Iteration 2

0.064	0.192	0.444	+1

Iteration 3

0.083	0.196	0.444	+1

Iteration 4

0.087	0.197	0.444	+1

Iteration 5

0.087	0.197	0.444	+1

Iteration 6

VALUE ITERATION

0.087	0.197	0.444	+1

- We can now derive our policy from the utility values we converged to;
  - $л(s_0)$=argmax{$0.8U_1+0.2U_0$, U_0, $0.9U_0+0.1U_1$, $0.9U_0+0.1U_1$}
    =argmax{0.175, 0.087, 0.098, 0.098}
  - Repeat for S_1 and S_2
  - The optimal policy will then be:
    - $л(s_0)$= Right
    - $л(s_1)$= Right
    - $л(s_2)$= Right

VALUE ITERATION

- We can now derive our policy from the utility values we converged to;
  - $\pi(s_0)=\text{argmax}\{0.8U_1+0.1U_0+0.1U_5,\ 0.9U_0+0.1U_5,\ \mathbf{0.8U_5+0.1U_1+0.1U_0},\ 0.9U_0+0.1U_1\}$

    $=\text{argmax}\{0.67,0.71,\mathbf{0.74},0.7\}$
  - Repeat for S_1 and S_2 …

0.812	0.868	0.918	1
0.762		0.66	−1
0.705	0.655	0.611	0.388

After convergence

VALUE ITERATION - EXERCISE

- Assume that we have the grid world, with a reward in each state of +2, and for the final state +1. We also have a discount factor γ=0.1
- Assume that the actions allowed are U,D,L,R. Succeeding 80%, and 20% are perpendicular to the direction of the action.

- Verify that you get the same utilities at iteration 4?
- Check that the policy should be: Right, Down, Left for S_0,S_1 and S_2 respectively.

0	0	0	+1

Iteration 0

2.22	2.22	2.22	+1

Iteration 4