Lego© Mindstorms NXT and Q-Learning: a Teaching Approach for Robotics in Engineering

Ángel Martínez-Tenor, Juan-Antonio Fernández-Madrigal and Ana Cruz-Martín

Systems Engineering and Automation, University of Málaga (Spain), Andalucía Tech

ICERI 2014

PROBLEM STATEMENT

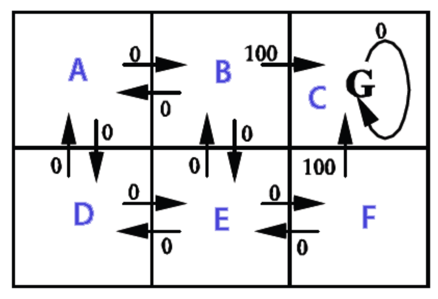
Robotics: Common subject in Engineering degrees

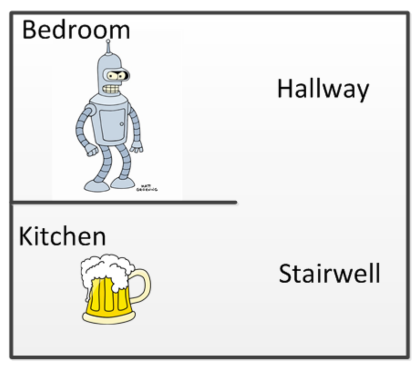
↳ **Cognitive Robotics:** Now covered in Postgraduate programs

↳ **Reinforcement Learning (RL):** Decision-Making Machine learning

↳ **Q-learning algorithm:** Simple, effective and well-known RL algorithm

Simulated robots





Embodied agent

- Obstacle avoidance
- Line follower
- Walking
- Phototactic behaviour

Agent — **Environment**

action →

← reward

← new state

RL does not require a model of the environment, overcoming this limitation by making observations

GOAL: Develop a teaching framework integrating:

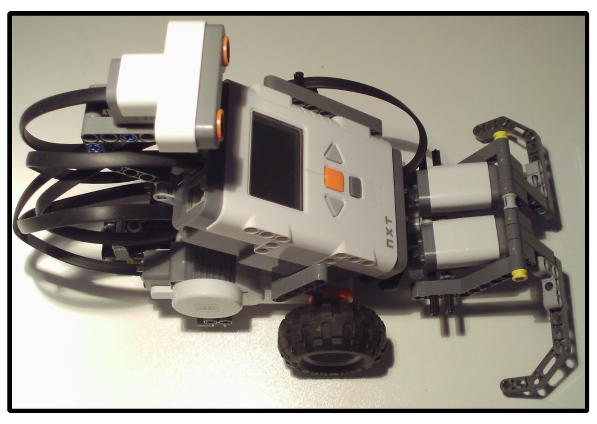
- Q-learning
- Practical activities
- Real Mobile Robots

Allowing the students to get a better understanding of the robotic learning problem

APPROACH

Materials

LEGO MINDSTORMS NXT Education Base Set



Developed work

Offline Design & Simulation

- Assembly
- Modeling
- Simulation
- Parameters Tuning

↓ Offline Q-learning Template Octave / Matlab

→

Robot implementation

- Q-learning on robot
- CPU limitations studies: Overflow & Precision losses
- Robot vs Simulation analysis
- Parameters Optimization

↓ NXT Robot Q-learning NXC (main & NXT_io library)

Q-learning pseudocode

```
% Q-learning algorithm parameters
    N_STATES, N_ACTIONS,INITIAL_POLICY
% Experiment parameters
    N_STEPS, STEP_TIME

% Variables
    s,a,sp  % (s,a,s')
    R       % Obtained Reward
    alpha   % Learning rate parameter
    GAMMA   % Discount rate parameter
    Policy  % Current Policy
    V       % Value function
    Q       % Q-matrix
    step    % Current step

% Initial state
    V = 0, Q = 0, Policy = INITIAL_POLICY
    s = Observe_state() % sensors-based observation

    for step = 1:N_STEPS  %---- Main loop ---------------
        a = Exploitation_exploration_strategy()

        robot_execute_action(a), wait(STEP_TIME)

        sp = observe_state(), R = obtain_Reward()

        Q(s,a) = (1-alpha)*Q(s,a) + alpha*(R+GAMMA*V(sp))

        V(s)_update()
        Policy(s)_update()
        alpha_update()
        s = sp  % update current state
    end  %----------------Main loop end------------------
```

Obstacle-avoidance wandering tasks

Wander -1:

States	
s1	no contact
s2	left bumper contact
s3	right bumper contact
s4	both contacts

Actions	
a1	stop
a2	left-turn
a3	right-turn
a4	move forward

Wander -2:

Evolved from Wander-1	
s1	no contact & obstacle near
s5	no contact & obstacle far

Wander -3:

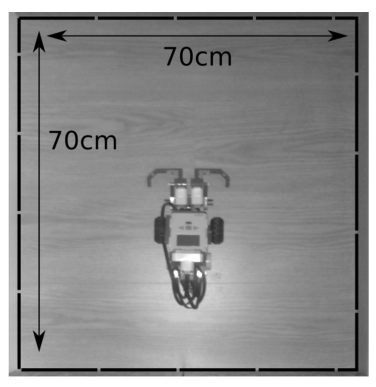
States	
s1	no contact & obstacle range 0
s2	left contact & obstacle range 0
s3	right contact & obstacle range 0
s4	both contacts & obstacle range 0
s5	no contact & obstacle range 1
s6	left contact & obstacle range 1
s7	right contact & obstacle range 1
s8	both contacts & obstacle range 1
s9	no contact & obstacle range 2
s10	left contact & obstacle range 2
s11	right contact & obstacle range 2
s12	both contacts & obstacle range 2
s13	no contact & obstacle range 3
s14	left contact & obstacle range 3
s15	right contact & obstacle range 3
s16	both contacts & obstacle range 3

Actions	
a1	stop
a2	left-turn (both wheels)
a3	right-turn (both wheels)
a4	move forward
a5	left wheel forward
a6	right wheel forward
a7	left wheel backward
a8	right wheel backward
a9	move backward

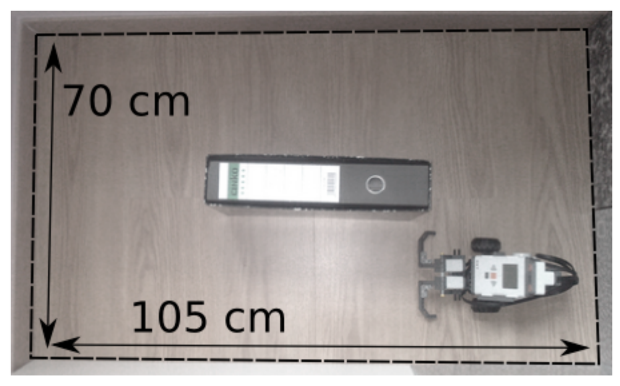
Range	Distance (cm)
0	< 25
1	25-50
2	50-75
3	> 75

Scenarios

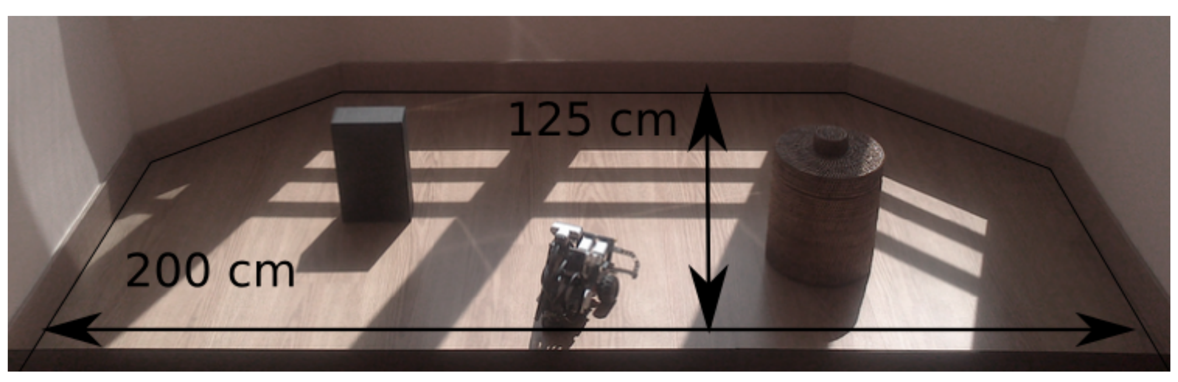
A



B



C



Optimal parameters

Rewards	
1	moved forward (no collide)
0.1	turned (no collide)
-0.2	one bumper collides
-0.7	both bumpers collide

Parameter	Value
Robot Speed	50 (of [0,100])
Step Time	250 ms
Number of Steps	2000
Exploration	e-greedy 30%
Discount rate γ	0.9
Learning rate α	0.02
FP	10000 (4 decimals)
Q-matrix cell size	4 bytes (long)

RESULTS

Q-learning method for a NXT mobile robot

Simple — Applicable — Stable

All learning tasks were performed successfully leading to an optimal or pseudo-optimal policy

Benefits for students

Simulation templates
- Allow a thorough analysis of the Q-learning parameters

Real Robot templates
- Easily adapted to different tasks with minimal changes
- Get a complete vision of the learning problem filling the gap between theory and practice