Nao QLearning Experiment

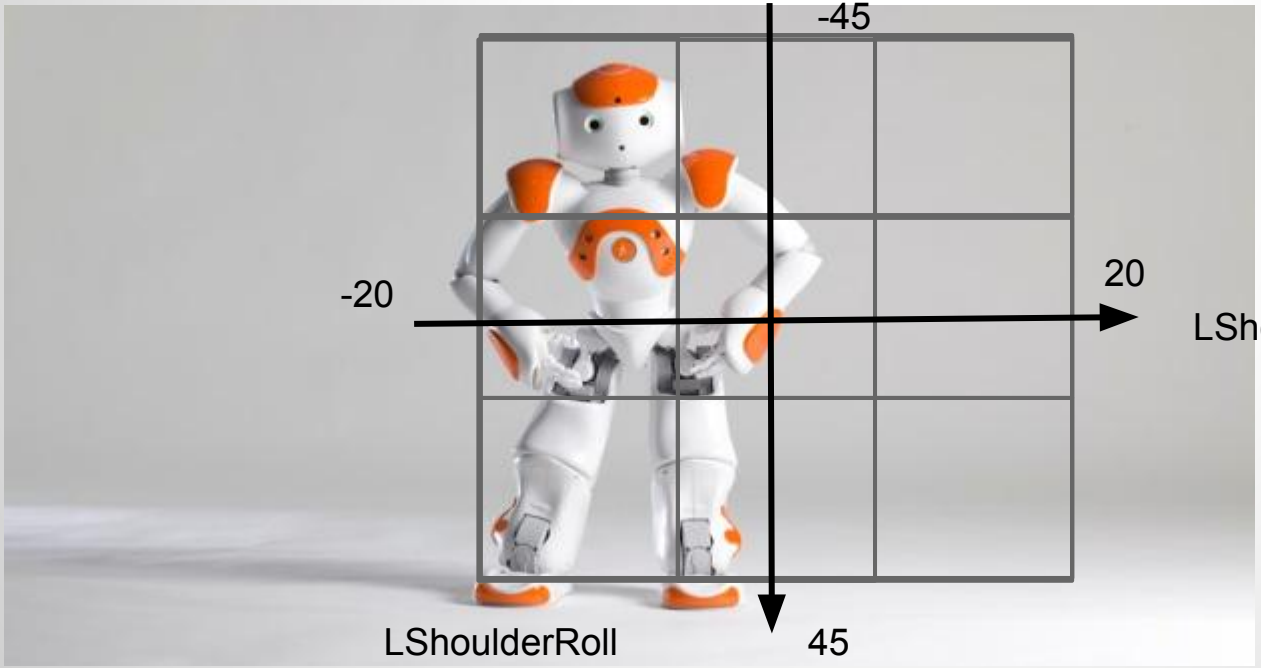
Guillaume Diallo-Mulliez, Etienne Masson Bachasson de Montalivet, Woody Rousseau

Goal

Using QLearning to teach Nao how to point at a given direction in space with its arm

States

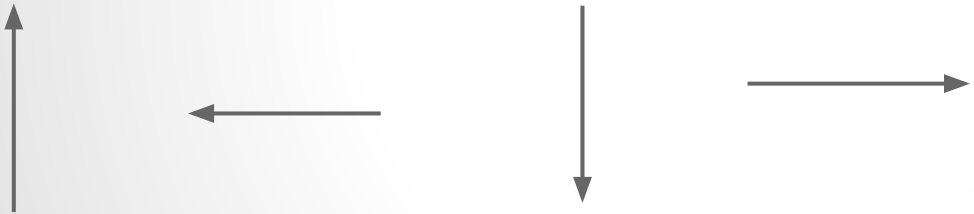
Space is discretized in 9 areas reached by NAO’s left arm



Actions

4 Actions are possible and are unidirectional

Checks are made to stay in bounds



Policies

3 Policies for picking the actions are implemented

- Random picks a random (feasible action)
- Optimal picks the best action based on the Q row
- Epsilon Greedy picks the optimal action with a ε probability, a random one otherwise

Rewards

A reward is expected at each performed action

- +1 if the head front sensor is touched (the arm got closer to the goal state)
- -1 if the head back sensor is touched (the arm got further from the goal state)
- +10 when the robot hears “Bravo” with speech recognition(the robot reached the goal state)

Q Learning

Q Matrix : Dimensions 9x4 (States x Actions)

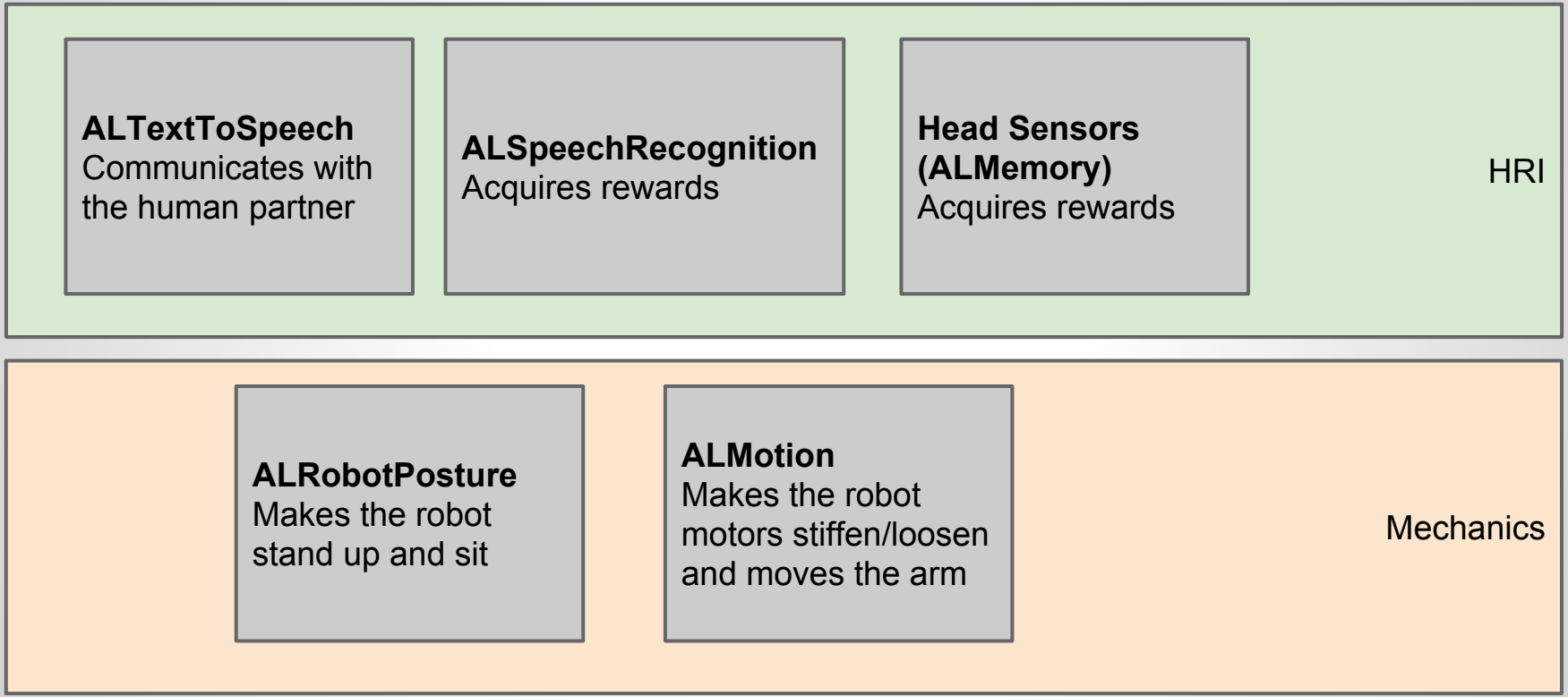
$$Q_{t+1}(s_t, a_t) = \underbrace{Q_t(s_t, a_t)}_{\text{old value}} + \underbrace{\alpha_t(s_t, a_t)}_{\text{learning rate}} \times \left[\overbrace{\underbrace{R_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \underbrace{\max_a Q_t(s_{t+1}, a_t)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q_t(s_t, a_t)}_{\text{old value}} \right]$$

The discount factor γ determines the importance of future rewards. A factor of 0 will make the agent "myopic" (or short-sighted) by only considering current rewards, while a factor approaching 1 will make it strive for a long-term high reward

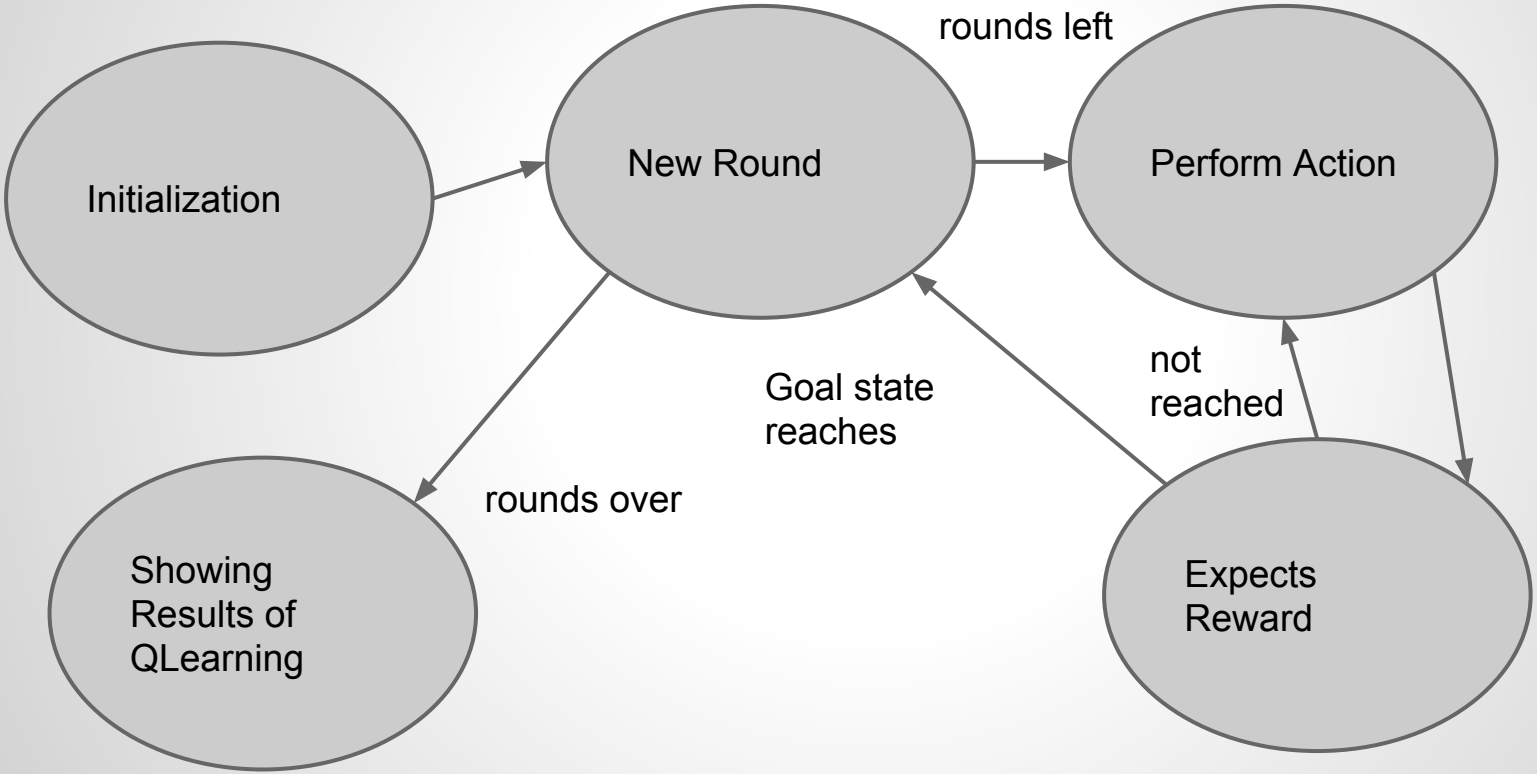
The learning rate determines to what extent the newly acquired information will override the old information. A factor of 0 will make the agent not learn anything, while a factor of 1 would make the agent consider only the most recent information.

All parameters (and picked policy) can be passed as CLI arguments to the main Python script

Nao Modules


ALTextToSpeech
Communicates with the human partner
ALSpeechRecognition
Acquires rewards
Head Sensors (ALMemory)
Acquires rewards
HRI
ALRobotPosture
Makes the robot stand up and sit
ALMotion
Makes the robot motors stiffen/loosen and moves the arm
Mechanics

State Machine



Results

Policy epsilon_greedy, Alpha=0.200000, Gamma=0.900000 , Epsilon=0.100000, 5 Rounds

Goal State : Down Right

```
Top Left =>    [[ 0.        0.236     0.        0.      ] => Going Down
Top =>          [ 0.        0.36      0.        0.      ] => Going Down
Top Right =>    [ 0.        0.       -0.2       0.      ] => Not Going Left
Left =>         [ 0.        0.814592  0.        0.      ] => Going Down
Center =>      [-0.164      0.       -0.123536  0.      ] => Not going up
Right =>[ 0.        0.        0.        0.      ] => Random
Bottom Left => [ 0.        0.        0.        2.48384 ] => Going Right
Bottom =>       [ 0.        0.       -0.164     6.7232  ] => Going Right
Bottom Right => [ 0.        0.        0.        0.      ]] => Goal State
```

Coding

All code available on Github

https://github.com/Guitoof/qnao

First tests in Matlab

- Basic QLearning script with keyboard input rewards (qlearning.m)
- States and Actions classes

Coding

Main coding in Python (naoqi)

- qnao.py : Main script, includes arguments parsing, state machine and QLearning algorithm
- action.py, state.py : Classes representing the states and actions
- policies.py : Class implementing the different policies
- reward.py : Recognizes sensor events and voices to acquire rewards
- config.py : Includes the configuration (nao IP and port)

> pip install -r requirements.txt

> python qnao.py --epsilon 0.7 --alpha 0.1

Questions

Feedback, Remarks ?