Reinforcement Learning in ARGoS

– Intelligent Robotic Systems –

Andrea Roli
andrea.roli@unibo.it
Dept. of Computer Science and Engineering (DISI)
Alma Mater Studiorum Università di Bologna

Path Following

One of the most common tasks for robots working in real-world contexts is to follow a path. For instance, we may want robots able to move objects to target areas while constraining their paths by following a trace painted on the ground (we suppose the path is always free, so no collision issues).

The robot is equipped with ground sensors and uses these values to define its trajectory. We could write a control program such that the robot is able to attain this task, but this is also an interesting context to try RL.

Learning to follow a path

You may find an implementation of Q-learning for Path Following in `Qlearning-argos.zip`.[1]

The code is composed of the following files:

- `Qlearning.lua`: main module containing the implementation of Q-learning.
- `circuit-learning.lua`,`circuit-learning.argos`: implementation of Q-learning for path following.
- `train-script.sh`: bash script to run the learning process. You can of course adapt it to your needs: you just want to run the training for a given number of epochs.
- `circuit-testing.lua`,`circuit-testing.argos`: to test the robot after training. To test it just execute: `argos3 -c circuit-testing.argos`

[1]The code is by Matteo Magnini, with minor changes.

Note that the Q-table is stored as a `csv` file and it is updated at the end of each learning epoch. You have to create the first table before running the training phase by executing `create_Q-table.lua` with parameters: `Qtable-circuit.csv 256 5`

Images for the floor to be used in training and testing are in folder `img`.

Suggestions for some experiments with the code:

1. Before exploring the code, think of a possible implementation you would make. For example, how would you define states and actions? And the reward function?
2. Go back to step 1 until you are sure you exactly know what to do if you had to program RL yourself!
3. Now you can explore the code, starting from `Qlearning.lua`
4. Assign values to the parameters of the algorithm (in `circuit-learning.lua`).
5. Run the learning code with batches of 10 or 20 epochs up to 50 (at least); at the end of each epoch test the behaviour of the robot. You may want to apply a sound experimental evaluation by running the robot from different initial conditions and collect statistics by using the script `test-script.sh`.
6. Repeat the overall learning phase by trying with different parameter values.
7. Try also different combinations of train/test images.

Food for thought

- Are there alternative ways for defining the reward function?
- What is the impact of parameter values?
- Is there a parameter that is more critical than others?
- How could you assess the overall performance of the technique?
- Is there a way to estimate the convergence of the algorithm?