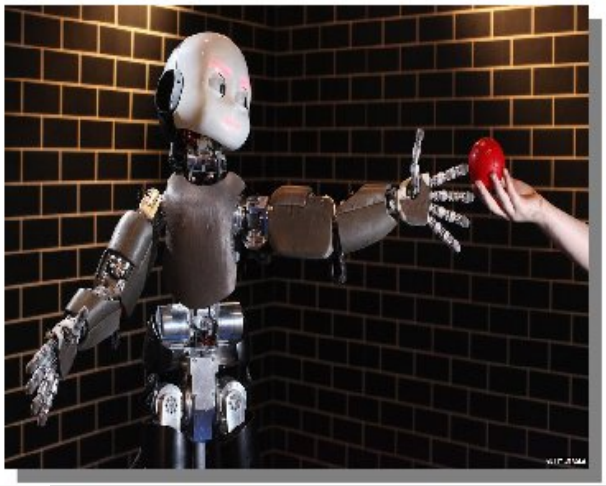
Verification of Agents learning through Reinforcement

Shashank Pathak[12] Giorgio Metta[12] Luca Pulina[3]
Armando Tacchella[2]

Robotics, Brain and Cognitive Sciences (RBCS)
Istituto Italiano di Tecnologia (IIT), Via Morego, 30 – 16163 Genova – Italy
Shashank.Pathak@iit.it - Giorgio.Metta@iit.it

Dipartimento. di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi (DIBRIS)
Università degli Studi di Genova, Via Opera Pia, 13 – 16145 Genova – Italy
Armando.Tacchella@unige.it

POLCOMING, Università degli Studi di Sassari



Some relevant features:

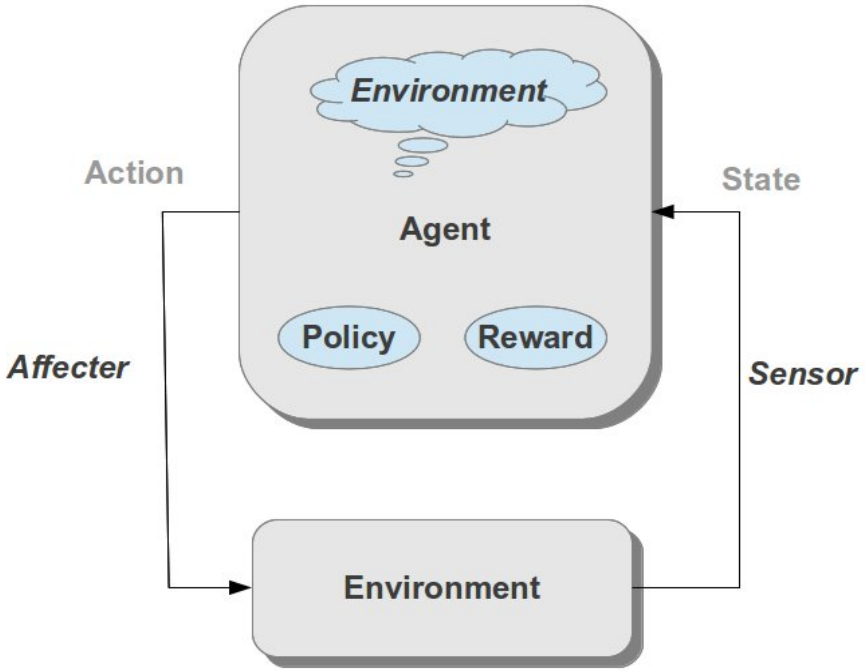


Figure : Reinforcement Learning

- Learning through experiences ie $(S_t, A_t, R_t, S_{next})$
- Objective is to attain a policy $\pi(s_i) \rightarrow A_i$
- Secondly, policy should be maximizing some measure of "rewards" R_i

Finite-window rewards

Assume:
finite time-horizon $t \in (t, T)$ and discounting γ with $\gamma \in [0, 1)$

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-t-1} r_T$$

and that we define, *Value* as expected-value of this averaged-reward $V^\pi(s) = E^\pi(R_t | s_t = s)$

$$V(s_t) \rightarrow V(s_t) + \alpha(R_t - V(s_t))$$

We would have *update*:

$$V(s_t) \rightarrow V(s_t) + \alpha\delta, \quad \delta = (r_{t+1} + \gamma V(s_{t+1}) - V(s_t)) \qquad (1)$$

ISTITUTO ITALIANO DI TECNOLOGIA

Air hockey

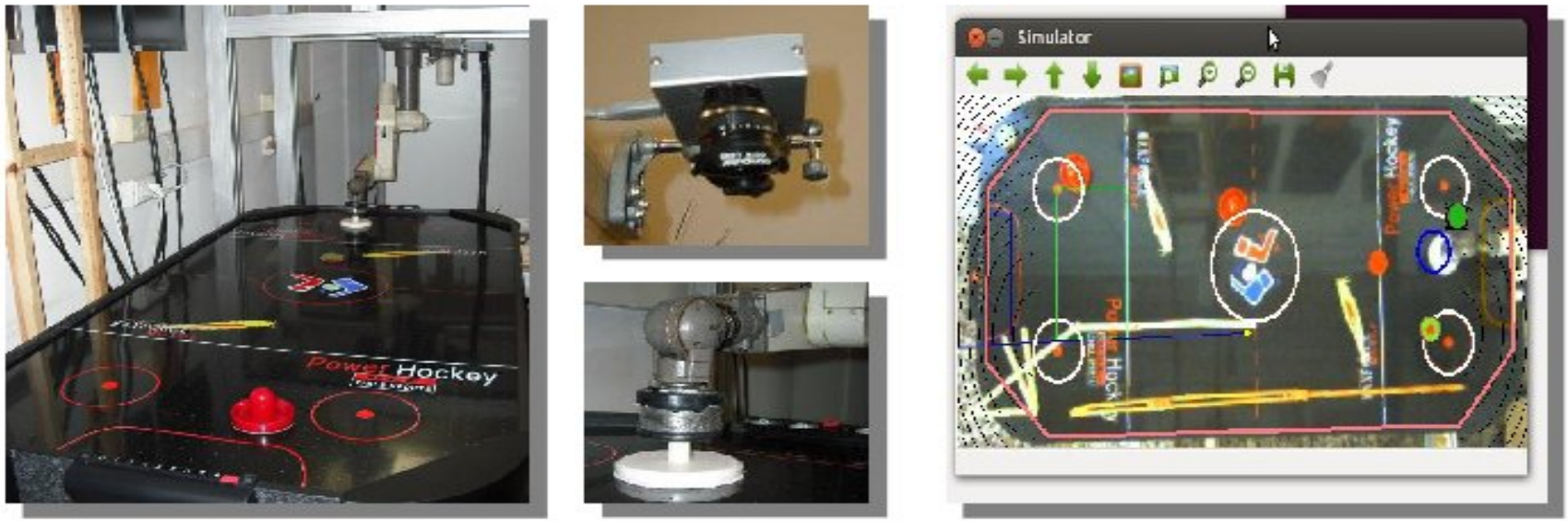


Figure : Platform and simulator

ISTITUTO ITALIANO DI TECNOLOGIA

Reasons for picking up air hockey

- Air hockey is a challenging platform and has been used in past to demonstrate learning
- As a robotic setup, it has been included as one of the benchmark for robotics & humanoids
- Our previous work has been performed on real air hockey and supervised learning

Simulator

For the current study, we chose simulator instead of real setup

- Our goal was to demonstrate safety in a model-free learning approach and ways to improve it
- Some sophiticated semi-supervised approaches are needed to apply RL on real setup
- Showing benefits of verification and repair was independent to these approaches
- Simulation or at least some logging would be required even if real setup were used

ISTITUTO ITALIANO DI TECNOLOGIA

Simulator ...

Simulator was implemented with C++ using some libraries like OpenCV, Boost and Pantheios

- For simplicity no game engine was used, rather 2D Physics was implemented
- Also physical and geometric considerations were made
- Extensive logging and a GUI based parameter search was done

Learning Problem

Given: an air hockey platform and a robotic arm.
Objective: to learn to defend the goal as good as possible

- Action of robotic arm was constrained to be minimum-jerk trajectory
- Joint-kinematics and safety
- State was defined in trajectory-space rather than cartesian coordinates
- Discrete state and discrete actions were considered

LEARN

Algorithm 1 Pseudo-code for learning to play Air hockey using Reinforcement Learning

Initialize $Q \leftarrow \mathbf{0}$; $\Delta t \leftarrow 20$ms
function LEARN(N_e, N_b, N_r)
  for all $i \in \{1, \ldots, N_e\}$ **do**
    Send *Start* signal to Simulator
    $j \leftarrow 1$
    repeat
      Receive $s_j \leftarrow (p_j, \alpha_j, \theta_j)$ from Simulator
      $\Delta\theta_j \leftarrow$ COMPUTEPOLICY(Q, s_j)
      Send $(\Delta\theta_j, \Delta t)$ to Simulator
      Receive $s_{j+1} \leftarrow (p_{j+1}, \alpha_{j+1}, \theta_{j+1})$ and
        $f_{j+1} \leftarrow (m, g, w, r)$
      $r_{j+1} \leftarrow$ COMPUTEREWARD((s_{j+1}, f_{j+1})
      $E_j \leftarrow (s_j, \Delta\theta_j, r_{j+1}, s_{j+1}, f_{j+1})$
      $Q \leftarrow$ UPDATE(Q, E_j)
      $j \leftarrow j + 1$
      if $(j = N_b)$ **then**
        for all $k \in \{1, \ldots, N_r\}$ **do**
          Choose random $m \in \{1, \ldots, N_b\}$
          $Q \leftarrow$ UPDATE(Q, E_j)
        end for
        $j \leftarrow 1$
      end if
    until $r =$ TRUE
  end for
  return Q

Verification of DTMC

- Discrete state-action space, allowed to model learned policy as a Discrete Time Markov Chain
- Learnt policy $\pi(s) \rightarrow a$ was Softmax distribution over Q-values,

$$\pi(s, a_i) = \frac{e^{\kappa Q(s,a_i)}}{\sum_{a \in A} e^{\kappa Q(s,a)}} \tag{2}$$

- Next states were observed via simulation and probabilities were adjusted imperically
- We considered 2 approaches: *unsafe* states as *failure* and as *fault*

Verification of DTMC: Unsafe states as failures

- unsafe flag $\Longrightarrow$ halt
- On practical setups, there are usually low-level control
- Some approaches to address this: Lyapunov candidates, safety conscious rewarding etc
- For sake of generality and yet effectiveness, we used safety conscious rewarding schema while avoided Lyapunov candidates
- In our case, safety of the agent is reachability probability on unsafe states
- Using safety property, we used both PRISM and MRMC, to get qualitative measure of safety

Repairing DTMC

- Intuition: badness of a state depends on forward proximity to a bad state.
- In general, changing Q-values in ways similar to *eligibility trace* would make policy safer
- While this is more effective than incorporating safety while learning, it could deteriorate learnt policy
- Our experiments show it need not be the case

Repairing DTMC: Using COMICS

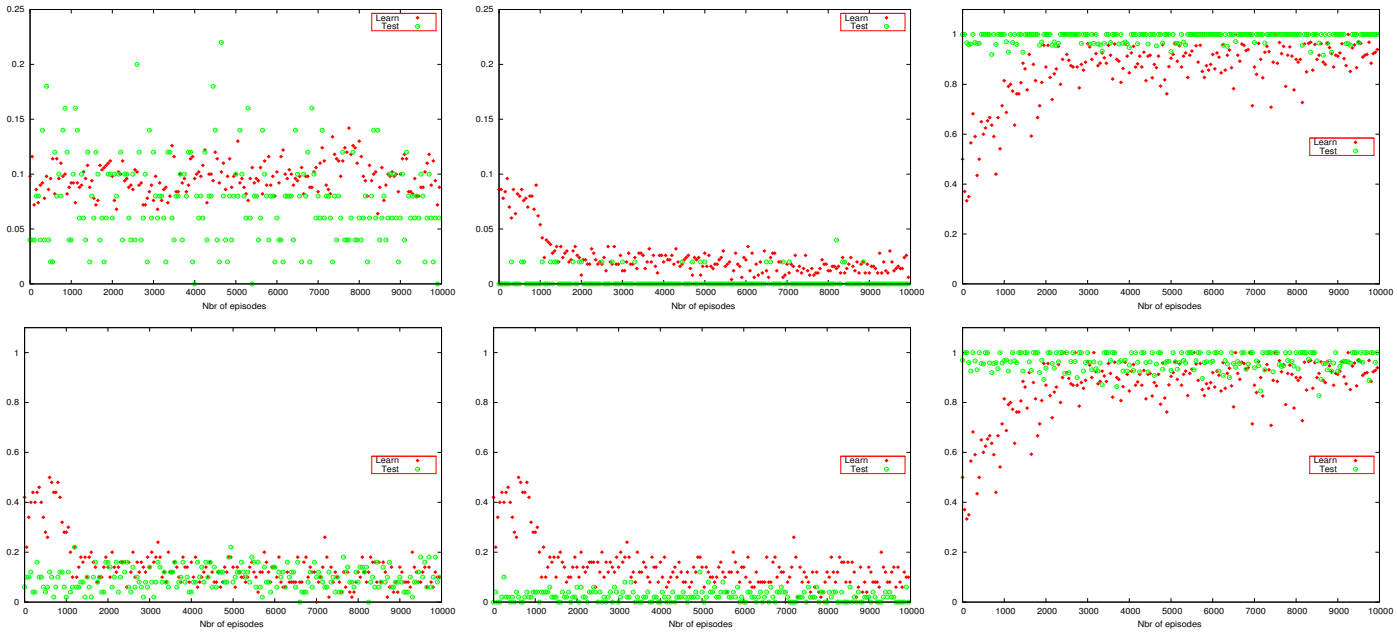
- We used tool COMICS to generate the counter-example
- We then proceeded with repairing the paths
- The overall algorithm was

Algorithm 2 Pseudo-code for Verification and Repair of Learn

Given agent $\mathcal{A}$, learning algorithm ▸ Learn , safety bound P_{bound}
Using $\mathcal{A}$ perform Learn
Obtain policy $\pi(s, a)$
Construct a DTMC $\mathcal{D}$ from policy $\pi(s, a)$
Use MRMC or PRISM on $\mathcal{D}$ to obtain P_{unsafe} of violating $\mathcal{P}$
repeat
    repeat
        Use COMICS to generate set $\mathbf{S}_{unsafe}$ negating $\mathcal{P}$ with bound P_{unsafe}
        Apply Repair on $\mathbf{S}_{unsafe}$
    until $\mathbf{S}_{unsafe} = \{\phi\}$
    $P_{unsafe} \leftarrow P_{unsafe} - \epsilon$, $\epsilon \in (0, P_{unsafe} - P_{bound}]$
until $P_{unsafe} < P_{bound}$

ISTITUTO ITALIANO DI TECNOLOGIA

Results



Thanks to audience and my colleagues [1]!
Questions or comments?

[1] *Armando Tacchella*, *Giorgio Metta*, & *Luca Pulina*