Efficient PAC Reinforcement Learning in Regular Decision Processes

Alessandro Ronca Giuseppe De Giacomo

DIAG – Sapienza Università di Roma

Setting: Agent and Environment

An agent interacts with its environment, performing *actions*, and receiving *observations* and *rewards*. To learn, the agent has the opportunity to *stop* and possibly start over. This process generates strings of the form

$$a_1, o_1, r_1, \ldots, a_n, o_n, r_n$$

which we call *episodes*. They form the experience of the agent, which is the basis to improve its behaviour.

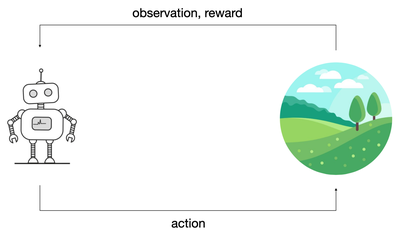


Figure 1. An agent interacting with its environment.

Non-Markov Decision Processes (NMDPs)

NMDPs are decision processes where the Markov property does not necessarily hold. Thus, their dynamics can depend on the past observations.

$$\mathcal{P} = \langle A, O, R, \mathbf{T}, \mathbf{R}, \gamma \rangle$$

The components are as follows:

- actions A, observations O, and rewards R;
- transition function $\mathbf{T} : O^* \times A \to \mathbb{P}(O)$;
- reward function $\mathbf{R} : O^* \times A \times O \to R$;
- discount factor $\gamma \in (0, 1)$.

Also policies are history-dependent. A *policy* is a function $\pi : O^* \to \mathbb{P}(A)$ that maps histories of observations to probability distributions over actions. The *value* $\mathbf{v}_\pi$ of a policy π is also a function of the history, $\mathbf{v}_\pi : O^* \to \mathbb{R}$. A policy is (near-)optimal if its value is (near-)maximum on every history.

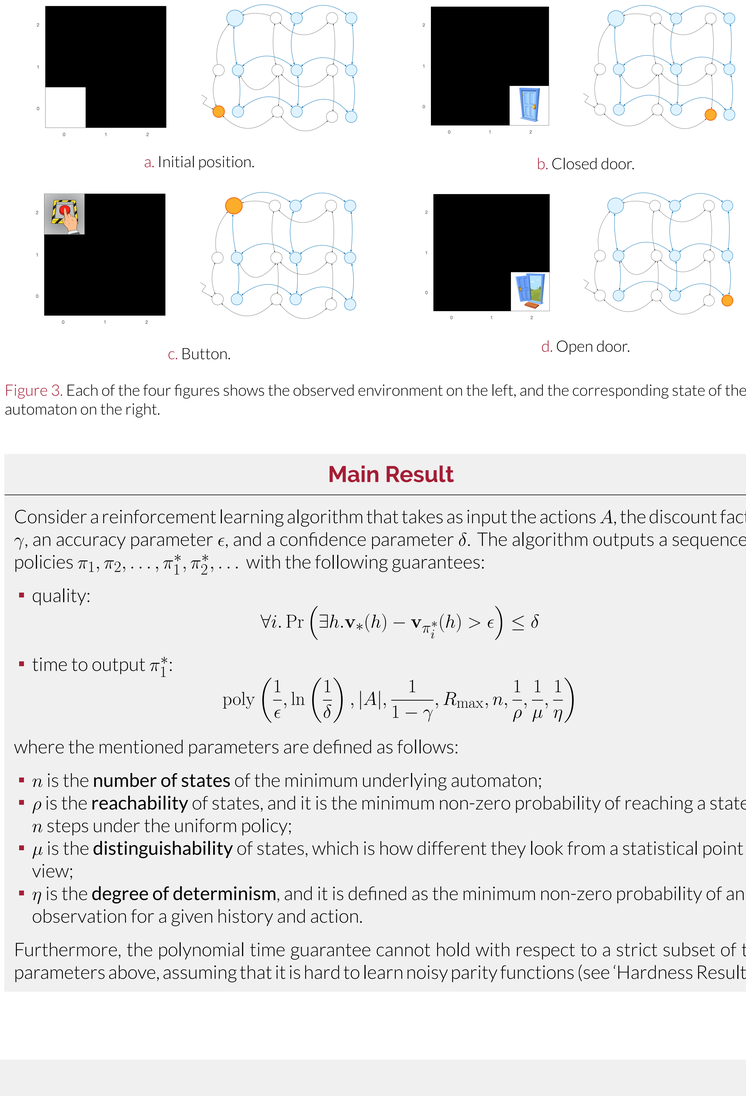
Regular Decision Processes (RDPs)

RDPs have been first introduced in [2] as the class of NMDPs where the functions $\mathbf{T}$ and $\mathbf{R}$ can be represented using the temporal logic on finite traces LDL_f. It has the same expressive power as (i) Monadic Second-Order Logic on finite ordered traces, and (ii) Regular Expressions. Equivalently, here we use finite-state transducers:

- a finite-state transducer that, on every history $h \in O^*$, outputs the function $\mathbf{T}_h : A \to \mathbb{P}(O)$ induced by $\mathbf{T}$ when its first argument is h;
- a finite-state transducer that, on every history $h \in O^*$, outputs the function $\mathbf{R}_h : A \times O \to R$ induced by $\mathbf{R}$ when its first argument is h.

Example of an RDP

As an example of RDP, consider a grid. At each moment, the agent observes the coordinates of its current cell, and the content of the cell. The cell at the right-bottom corner contains a door, which is initially closed, and it opens when the agent pushes a button located in the cell at the left-top corner (by simply visiting the cell). Such dynamics are captured by the automaton shown below. There are two states for each cell, except for the button cell which has only one. Each state records the current position and whether the button cell has been visited.



a. Initial position. b. Closed door. c. Button. d. Open door.

Figure 3. Each of the four figures shows the observed environment on the left, and the corresponding state of the automaton on the right.

Main Result

Consider a reinforcement learning algorithm that takes as input the actions A, the discount factor γ, an accuracy parameter ϵ, and a confidence parameter δ. The algorithm outputs a sequence of policies $\pi_1, \pi_2, \ldots, \pi_1^*, \pi_2^*, \ldots$ with the following guarantees:

- quality:
$$\forall i.\ \Pr\left(\exists h. \mathbf{v}_*(h) - \mathbf{v}_{\pi_i^*}(h) > \epsilon\right) \leq \delta \tag{1}$$
- time to output π_1^*:
$$\mathrm{poly}\left(\frac{1}{\epsilon}, \ln\left(\frac{1}{\delta}\right), |A|, \frac{1}{1-\gamma}, R_{\max}, n, \frac{1}{\rho}, \frac{1}{\mu}, \frac{1}{\eta}\right) \tag{2}$$

where the mentioned parameters are defined as follows:

- n is the **number of states** of the minimum underlying automaton;
- ρ is the **reachability** of states, and it is the minimum non-zero probability of reaching a state in n steps under the uniform policy;
- μ is the **distinguishability** of states, which is how different they look from a statistical point of view;
- η is the **degree of determinism**, and it is defined as the minimum non-zero probability of an observation for a given history and action.

Furthermore, the polynomial time guarantee cannot hold with respect to a strict subset of the parameters above, assuming that it is hard to learn noisy parity functions (see 'Hardness Results').

Technique

Probabilistic-Deterministic Finite Automata (PDFA)

Our technique is based on PDFA. A PDFA is a tuple $\langle Q, \Sigma, \zeta, \tau, \lambda, q_0 \rangle$ where:

- Q is a finite set of states, Σ is a finite alphabet, ζ is the end-of-string symbol, q_0 is the inital state;
- $\tau : Q \times \Sigma \to Q$ is a deterministic transition function;
- $\lambda : Q \to \mathbb{P}(\Sigma \cup \{\zeta\})$ specifies, for every state, the probability of generating a certain symbol, or terminating.

PDFA are PAC-learnable in time polynomial in the number of states, the alphabet size, and state distinguishability [1].

Capturing RDPs via PDFA

Consider action-observation-reward traces generated under a fixed policy, that chooses to stop eventually with probability one. The probability distribution over such traces is captured by a PDFA.

A simple RL algorithm for RDPs

1. Learn a PDFA from traces generated under the uniform policy (with constant stop probability);
2. Map the PDFA to an MDP;
3. Solve the MDP (e.g., value iteration) to obtain a Markov policy;
4. Return the Markov policy composed with the transition function of the PDFA.

Guarantees

The algorithm can be shown to have PAC-style guarantees:

- From the classic RL literature we know that a sufficiently accurate approximation of an MDP allows one to compute a near-optimal policy, cf. [4].
- We can show that every near-optimal policy for the MDP built from the PDFA yields a near-optimal policy for the original MDP when composed with the transition function;
- The accuracy of the MDP is bound to the accuracy of the PDFA;
- An accurate PDFA can be learned in polynomial time [1].

Hardness Results

None of the parameters in Equation (2) is redundant. It can be shown as follows:

- The parameters $\frac{1}{\epsilon}, \ln\left(\frac{1}{\delta}\right), |A|, \frac{1}{1-\gamma}, R_{\max}$ are already necessary in MDPs;
- Without a number of steps that grows with $n, \frac{1}{\rho}, \frac{1}{\eta}$, the agent may not observe events that are necessary to choose a good policy;
- For $\frac{1}{\mu}$, there is a reduction from learning noisy parity functions, already used for PDFA [3].

References

[1] Borja Balle, Jorge Castro, and Ricard Gavaldà. Learning probabilistic automata. *Theor. Comp. Sci.*, 2013.

[2] Ronen I. Brafman and Giuseppe De Giacomo. Regular decision processes: A model for non-Markovian domains. In *IJCAI*, 2019.

[3] Michael J. Kearns, Yishay Mansour, Dana Ron, Ronitt Rubinfeld, Robert E. Schapire, and Linda Sellie. On the learnability of discrete distributions. In *STOC*, 1994.

[4] Michael J. Kearns and Satinder P. Singh. Near-optimal reinforcement learning in polynomial time. *Mach. Learn.*, 2002.

ronca@diag.uniroma1.it