Gradient Ascent Activity-based Credit Assignment with History-dependent Reward

Oussama Sabri, Luc Lehéricy, Alexandre Muzy

Gradient Ascent Activity-based Credit Assignment with History-dependent Reward

Oussama Sabri[1], Luc Lehéricy[2], and Alexandre Muzy[3]

[1] University Medical Center Hamburg-Eppendorf (UKE), Hamburg, Germany
o.sabri@uke.de
[2] Laboratoire JAD, CNRS, Université Côte d'Azur, Nice, France
luc.lehericy@univ-cotedazur.fr
[3] Laboratoire I3S, CNRS, Université Côte d'Azur, Nice, France
alexandre.muzy@univ-cotedazur.fr

Abstract. In reinforcement learning, credit assignment with history-dependent reward is a key problem to solve for being able to model agents: (i) associating the returns from their environment with their past (series of) actions, and (ii) figuring out which past decisions are responsible for the current achievement of their goal. Usual approaches simplify this problem by assuming an immediate reward for each action. Our first result is to propose a general and formal framework in which the credits assigned to actions are updated based on a gradient of expected rewards from past actions. This framework is able to model complex tasks that require fulfilling sub-tasks in order, each sub-task consisting of a specific sequence of actions. Our second result is to propose an algorithm using the activity of actions to increase (resp. decrease) the credits of necessary (resp. unnecessary) past actions. We illustrate our algorithm on a task inspired by a behavioral learning task of rodents in a maze.

Keywords: Credit assignment · Policy gradient algorithm · Non-Markovian rewards.

1 Introduction

In usual Reinforcement Learning (RL), the environment delivers a reward based on the current state of the system or action of the agent. This is called a Markovian reward case [4]. However, in general, the rewards delivered by the environment might also depend on the past states/actions of the agent: this is called the non-Markovian, or history-dependent, reward case.

For complex and non-Markovian rewards, a natural question is whether some specific actions are essential to obtain larger rewards. The credit assignment problem [6], which consists in assigning credit or blame to a series of decisions achieved by the agent, aims to answer this question. Identifying key actions allows adjustments to be made in order to improve the decision making performance.

The credit assignment problem is closely related to the concept of "switch state" (or action) [2]. In a series of actions, a switch action is an action that

needs to be chosen to obtain a positive reward from the execution of a series of actions. If a switch action is not chosen, the reward from the series is null. Switch actions are difficult to evaluate using temporal credit assignment [16] [14] since the reward function sums all contributions without informing the agent of which actions contributed the most.

Activity-based Credit Assignment [8] [9] has been developed to better assign credit to individual actions in tasks where discounting future rewards is inadequate. The return from the environment is then computed as a relationship between a measure of the activity of actions and the global reward obtained at the end of an episode (a series of actions). Here, we consider the activity as a measure of the profitability of the actions, in the sense that the agent wants to optimize the reward using high profitability actions. This corresponds to realistic situations where an agent achieves a complex task through a series of actions in which some actions can be discarded. The activity offers a way to figure out which actions should be taken.

As an application, we consider the problem of learning paths in a T-maze [7], as shown in Fig. 2a. An action consists in choosing and crossing a corridor between two crossroads or bends of the maze, and the activity of the action is the length of the corridor. We show that when taking activity into account, our algorithm identifies the shortest correct paths, while without the activity, the agent tends to make superfluous detours.

The policy underlying our algorithms are based on giving each possible action a credit and selecting an action with a probability given by the softmax of its credit. It is not necessary to store the whole series of actions as in usual approaches dealing with history-dependent rewards [4]: the credits are enough. The convergence of the algorithm is ensured by a gradient-based computation of the credit of each action. This is an advantage over other approaches that are not guaranteed to converge [2] [3] [12] or that require converting delayed rewards into immediate rewards [4].

2 Related work

In RL, two main approaches allow accounting for history-dependent rewards: (i) Deterministic finite automata [4] [12] used to learn series of actions, with no credit assignment and reinforcing the full series [4], and (ii) Stochastic gradient methods in which policies are approximated by a function and updated according to the gradient of the expected discounted reward [15] [18]. Following this approach, better estimations can be obtained by recomputing all action probabilities at each state / step [11]. More references can be found in [4] [12] [11] [2] [3] [12].

The convergence of deterministic automata depends on the convergence of the underlying RL algorithm used. In these approaches, non-Markovian situations are typically assumed to be Markovian to be able to use usual RL algorithms [4] [12]. Stochastic gradient approaches, such as the one we introduce, are more computationally intensive but provide a formal way to study and ensure the convergence of the resulting algorithms [15].

The general gradient ascent approach presented here allows for any return from the environment: not only the reward obtained at the end of an episode but possibly other metrics on (series of) actions, such as their activity. We will see that including some activity metrics over the series of actions in the learning policy can lead to improved results.

Regarding possible applications, the method obtained here allows to consider more complex experiments than the human decision making ones presented in [2]. Indeed, these experiments consider the case where a series of actions is correct if and only if it goes through a given "switch state" and reaches a specific end point, which is a particular case of our setting. Our application is an example of free learning of rodents in a maze where multiple correct series of actions are possible. By free learning we mean that the rodents are not taught the correct series of actions and need to find them by themselves. Note that although the behavior of real rodents in a maze has been studied in [7], our focus is on the models and algorithms rather than on realistically imitating rat brain activity and behavior. This generalizes the setting presented in [4], which considers only one correct series of actions. In particular, we allow the insertion of detours in the middle of correct series of actions: a correct series of actions is a sequence of groups of consecutive actions that need to be executed in order, with optional loops inserted between the groups. The reward of the series is zero when one group is missed.

In cognitive neuroscience, this kind of experiment [7] is more challenging to study. This is due to the fact that animals have to figure out and discover the series of actions leading to the rewards while so many useless intermediate actions can be inserted in the series without affecting the rewards. This is a challenging example of credit assignment problem.

Credit assignment can be summed up as follow: each time the agent gets a reward, it assigns credit (or blame) to past actions leading to this reward. Its objective is to identify the most rewarding sequence of actions. Discounted reward approaches, that estimate the worth of an action by the expected future reward without accounting for past actions, do not work in this setting, where obtaining a reward requires picking specific sequences of actions.

3 Problem formulation

3.1 Notations and definitions

Let $\mathbb{N} = \{1, 2, \dots, \}$ be the set of positive integers. For any $n \in \mathbb{N}$, write $[n]$ the set $\{1, 2, \dots, n\}$. Let $n \in \mathbb{N}$ be a number of states and $m \in \mathbb{N}$ be a number of actions. Let $\mathcal{S} = [n]$ be the set of all states the agent can be in. Let $\mathcal{F} = \{\mathcal{A}(s)\ ,\ s \in \mathcal{S}\}$ be the partition of the set of all actions $\mathcal{A} = [m]$, where $\mathcal{A}(s)$ is the set of all actions available to the agent when it is in state s.

For all $s \in \mathcal{S}$, let $\psi(s, \cdot)$ be a function from $\mathcal{A}(s)$ to $\mathcal{S}$ defined by

$$\begin{aligned} \psi(s,\cdot)\ :\ \mathcal{A}(s)\ &\longrightarrow \mathcal{S} \\ a\ &\longmapsto \psi(s,a) = s'. \end{aligned}$$

The function ψ describes the (deterministic) state transitions, that is, $\psi(s,a)$ is the state the agent finds itself in after taking action a in state s. Define the activity function μ as

$$\begin{aligned}\mu \,:\, \mathcal{S}\times\mathcal{A} &\longrightarrow \mathbb{R}_+^* \\ (s,a) &\longmapsto \mu(s,a).\end{aligned}$$

This function returns the cost of taking action a in state s.

A *trajectory* is a vector $(T,S,A) \coloneqq (T,(S_1,\dots,S_T),(A_1,\dots,A_{T-1}))$, where $T \in \mathbb{N}$ is the *length* of the trajectory, $S = (S_t)_{t\in[T]} \in \mathcal{S}^T$ is a vector of states, and $A = (A_t)_{t\in[T-1]} \in \mathcal{A}^{T-1}$ is a vector of actions. By convention, a trajectory of length one is of the form $(1,S_1,\varnothing)$. In what follow, the index $t \geqslant 1$ refers to a *time step*. A trajectory (T,S,A) is called *valid* if and only if

$$\forall t \in [T-1], \quad S_{t+1} = \psi(S_t, A_t).$$

Note that if the trajectory (T,S,A) is valid, then $A_t \in \mathcal{A}(S_t)$ for each $t \in [T-1]$. The total activity of a trajectory (T,S,A) is

$$M(T,S,A) = \sum_{t\in[T-1]} \mu(S_t, A_t).$$

The experiment proceeds as follows: at the first time step $t = 1$, the agent, starting in s_1, picks an action a_1 in $\mathcal{A}(s_1)$ according to a policy π_θ, for some policy parameter θ, observes the activity $\mu(s_1,a_1)$, then moves to the next state $s_2 = \psi(s_1,a_1)$. The agent continues this procedure until the environment ends the experiment and returns a reward to the agent, who can then update the policy parameter θ. We call one step of this procedure an *episode*, that is, sampling a trajectory and collecting the reward. We write $T^{(e)}$ the stopping time step which corresponds to the length of the trajectory picked during the e^{th} episode.

For each $e \geqslant 1$, let $(T^{(e)}, S^{(e)}, A^{(e)}) = (T^{(e)}, (S_t^{(e)})_{t\in[T^{(e)}]}, (A_t^{(e)})_{t\in[T^{(e)}-1]})$ be the random vector that is the valid trajectory chosen by the agent during the e^{th} episode, and let $M^{(e)} = M(T^{(e)}, S^{(e)}, A^{(e)})$ be the total activity of that trajectory. We assume that $T^{(e)}$ is a stopping time for the filtration induced by the sequence $((S_t^{(e)}, A_t^{(e)}))_{t\geqslant 1}$, that is, the event $\{T^{(e)} = T\}$ depends only on $(S_t^{(e)})_{t\in[T]}$ and $(A_t^{(e)})_{t\in[T]}$ (or simply $(A_t^{(e)})_{t\in[T]}$, since $S^{(e)}$ is determined by $A^{(e)}$ and ψ).

Fix $s^\star \in \mathcal{S}$. For any family $\theta = (\theta_{t,s,a})_{t\in\mathbb{N},s\in\mathcal{S},a\in\mathcal{A}(s)}$ of real numbers, let π_θ be the distribution such that if $(T,S,A) = (T,(S_t)_{t\in[T]},(A_t)_{t\in[T-1]}) \sim \pi_\theta$, then $S_1 = s^\star$ and the random variables $(S_t)_{2\leqslant t\leqslant T}$ and $(A_t)_{t\in[T-1]}$ are such that:

$$\begin{aligned}&\forall t \geqslant 2,\ \forall s \in \mathcal{S}, \quad \pi_\theta(S_t = s \,|\, S_1, A_1, \dots, S_{t-1}, A_{t-1}, t \leqslant T) \\ &\qquad = \mathbb{1}_{s=\psi(S_{t-1},A_{t-1})}\ , \\ &\forall t \geqslant 1,\ \forall a \in \mathcal{A}, \quad \pi_\theta(A_t = a \,|\, S_1, A_1, \dots, S_{t-1}, A_{t-1}, \\ &\qquad S_t, t \leqslant T-1) = \mathbb{1}_{a\in\mathcal{A}(S_t)} \frac{e^{\theta_{t,S_t,a}}}{\sum_{a'\in\mathcal{A}(S_t)} e^{\theta_{t,S_t,a'}}}\ .\end{aligned}$$

Hence,

$$\pi_\theta(T,S,A) = \mathbb{1}_{\substack{(T,S,A)\\ \text{is valid}}} \mathbb{1}_{S_1=s^\star} \mathbb{P}(T|S,A) \prod_{t=1}^{T-1} \frac{e^{\theta_{t,S_t,A_t}}}{\sum_{a'\in\mathcal{A}(S_t)} e^{\theta_{t,S_t,a'}}},$$

$$= \mathbb{1}_{\substack{(T,S,A)\\ \text{is valid}}} \mathbb{1}_{S_1=s^\star} \mathbb{P}(T|S,A) \prod_{t=1}^{T-1} \mathrm{softmax}_{\mathcal{A}(S_t)}(\theta_{t,S_t,\cdot}, A_t),$$

where for all $a \in \mathcal{A}$, $\mathrm{softmax}_{\mathcal{A}(S_t)}(\theta_{t,S_t,\cdot}, a)$ is the a-th component of the softmax function evaluated on $\theta_{t,S_t,\cdot}$, restricted on the subset $\mathcal{A}(S_t) \subset \mathcal{A}$. It defines a probability vector on $\mathcal{A}$ which gives mass 1 to $\mathcal{A}(S_t)$. Therefore, conditionally to $t \leqslant T-1$, the probability to choose an action $a \in \mathcal{A}$ at time step t depends only on S_t. For all $t \in \mathbb{N}$ and $s \in \mathcal{S}$, we call $(\theta_{t,s,a})_{a\in\mathcal{A}(s)}$ the *credit* vector over the set of actions available in state s at time step t.

The distribution π_θ is how the agent picks a trajectory starting from $s^\star$. We assume that the distribution of $T^{(e)}$ conditionally to $(A^{(e)}, S^{(e)})$, which is a function of the environment, does not depend on $s^\star$ or θ.

3.2 Environmental feedback

Assume that a trajectory can have a length at most $T_{\max} \in \mathbb{N}^*$ before the environment stops it.

Let $r \in \mathbb{N}$ and $\mathscr{T} = (\mathcal{T}_1^*, \dots, \mathcal{T}_r^*)$ be a vector of valid trajectories, where $\mathcal{T}_i^* = (T_i^*, S_i^*, A_i^*)$. We call $\mathscr{T}$ the family of target sub-trajectories. Note that these trajectory might not start at $s^\star$.

Given a valid trajectory (T,S,A), consider the function

$$\phi_{(T,S,A)} : [T] \times [r] \longrightarrow \{0,1\}$$
$$(t,i) \longmapsto \mathbb{1}_{t\geqslant T_i^*} \mathbb{1}_{(S_{t-T_i^*+1},\dots,S_t)=S_i^*} \mathbb{1}_{(A_{t-T_i^*+1},\dots,A_{t-1})=A_i^*}.$$

$\phi_{(T,S,A)}(t,i)$ returns whether or not an instance of the target sub-trajectory $\mathcal{T}_i^*$ is present in the trajectory (T,S,A) and ends at time t.

Let $\eta(t,(T,S,A))$ be the greatest integer $k \in [r]$ such that there exists integers $1 \leqslant t_1 < \cdots < t_k \leqslant T$ with $t_{i+1} - t_i \geqslant T_i^*$ for each $i \in [k-1]$ such that $\phi_{(T,S,A)}(t_i,i) = 1$ for each $i \in [k]$. If there exists no such k, let $\eta(t,(T,S,A)) = 0$ instead.

In other words, the function $(t,(T,S,A)) \longmapsto \eta(t,(T,S,A))$ counts the number of sub-trajectories from the family $\mathscr{T}$ that have been encountered in order up to time t in the trajectory (T,S,A).

Fig.1(a) shows an example of a trajectory of length $T = 12$. In this case, only two target sub-trajectories have been visited among the three targets illustrated in Fig.1(b). Examples of the value of η would be: $\eta(7,(T,S,A)) = 1$ and $\eta(11,(T,S,A)) = 2$. The *reward* of a trajectory (T,S,A) is the sum of the lengths of the target sub-trajectories encountered in order:

$$R(T,S,A) = \sum_{i=1}^{\eta(T,(T,S,A))} T_i^*.$$

For instance, the reward in the example of Fig.1(a) is $R(12, S, A) = 8$. In what follows, we write $R^{(e)} = R(T^{(e)}, S^{(e)}, A^{(e)})$ the reward of the trajectory from the e^{th} episode.

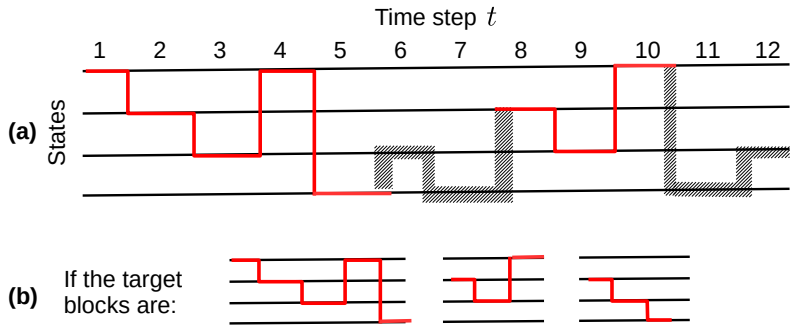


Fig. 1: Illustration of a trajectory of length 12. (a): the horizontal lines represent states (4 states in this case). Red sections correspond to the target sub-trajectories, and the grey sections are intermediate trajectories between the target sub-trajectories. These intermediate trajectories do not affect the reward. (b): The target sub-trajectories.

Finally, assume that if a trajectory $(T, S, A) \sim \pi_\theta$ for some θ, then its length T satisfies almost surely

$$T = \min\Big\{\{t \in [T] : \eta(t, (T, S, A)) = r\} \cup \{T_{\max}\}\Big\}, \tag{1}$$

that is, the environment stops the episode as soon as all target sub-trajectories have been encountered in the right order, or if the maximal trajectory length $T_{\max}$ is reached without encountering all the target sub-trajectories.

3.3 optimization problem

Write $\mathbb{E}_\theta$ the expectation under the policy π_θ. The goal of the agent is to maximise the expected value of an objective X:

$$\begin{aligned}\theta \longmapsto & \mathbb{E}_\theta[X^{(1)}] \\ = & \sum_{(T,S,A) \text{ trajectory}} \pi_\theta(T^{(1)} = T, S^{(1)} = S, A^{(1)} = A) \\ & \times \mathbb{E}[X^{(1)} \,|\, T^{(1)} = T, S^{(1)} = S, A^{(1)} = A].\end{aligned}$$

In the simulations, we consider two choices of X: when it is the reward $X = R$, and when it is the ratio between reward and total activity $X = R/M$. Until then, we not make any assumption on X, other than that the conditional expectation $\mathbb{E}_\theta[X^{(1)} \,|\, T^{(1)} = T, S^{(1)} = S, A^{(1)} = A]$ does not depend on the credit vector θ: it is a function of the environment and the trajectory, not of the agent.

The gradient of the expected value of the objective can be expressed as follows.

Theorem 1. *Let $B^{(1)}$ be a random variable that is independent of $(S_t^{(1)}, A_t^{(1)})$ conditionally to $\{T^{(1)} \geqslant t\}$, and $(\tilde{T}^{(1)}, \tilde{S}^{(1)}, \tilde{A}^{(1)})$ be a random trajectory with the same distribution as $(T^{(1)}, S^{(1)}, A^{(1)})$, then for all $t \in \mathbb{N}$, $s \in \mathcal{S}$ and $a \in \mathcal{A}$,*

$$\frac{\partial \mathbb{E}_\theta[X^{(1)}]}{\partial \theta_{t,s,a}} = \mathbb{E}_\theta\Big[\mathbb{1}_{t \leqslant T^{(1)}-1}\mathbb{1}_{S_t^{(1)}=s}\big(\mathbb{1}_{A_t^{(1)}=a} - \pi_\theta(\tilde{A}_t^{(1)} = a \,|\, \tilde{S}_t^{(1)} = s, t \leqslant \tilde{T}^{(1)} - 1)\big) \times (X^{(1)} - B^{(1)})\Big].$$

This results in a gradient ascent algorithm whose convergence is ensured by the following result.

Theorem 2. *Assume that there exists $r > 0$ such that $|X^{(e)}| \leqslant r$ and $|B^{(e)}| \leqslant r$ for all $e \geqslant 1$. Let $(\alpha_e)_{e \geqslant 1}$ be a sequence of nonnegative real numbers such that $\sum_{e \geqslant 1} \alpha_e = +\infty$ and $\sum_{e \geqslant 1} \alpha_e^2 < +\infty$. Update $(\theta^{(e)})_{e \geqslant 1}$ according to the rule: for each $t \in [T_{\max}]$, $a \in \mathcal{S}$ and $a \in \mathcal{A}(s)$,*

$$\theta_{t,s,a}^{(e+1)} = \theta_{t,s,a}^{(e)} + \alpha_e \mathbb{1}_{t \leqslant T^{(e)}-1}\mathbb{1}_{S_t^{(e)}=s}\Big(\mathbb{1}_{A_t^{(e)}=a} - \pi_{\theta^{(e)}}(A_t^{(e)} = a | S_t^{(e)} = s, t \leqslant T^{(e)} - 1)\Big) \times \Big(X^{(e)} - B^{(e)}\Big), \tag{2}$$

For each e, t, s and a, let $\widetilde{\theta}_{t,s,a}^{(e)} = \theta_{t,s,a}^{(e)} - \max_{a' \in \mathcal{A}(s)} \theta_{t,s,a'}^{(e)}$. Then $(\mathbb{E}_{\theta^{(e)}}[X^{(e)}])_{e \geqslant 1}$ converges and the limit points of $(\widetilde{\theta}^{(e)})_{e \geqslant 1}$ are all in the same connex component of the set of zeroes of the gradient of $\widetilde{\theta} \in \Theta \longmapsto \mathbb{E}_{\widetilde{\theta}}[X^{(1)}]$, where Θ is the (compact) set of credit vectors $\widetilde{\theta}$ such that $\widetilde{\theta}_{t,s,a} \in [-\infty, 0]$ and $\max_{a' \in \mathcal{A}(s)} \widetilde{\theta}_{t,s,a'} = 0$ for all t, s.

These theorems are proved in Appendix B and C respectively.

3.4 GAtACA Algorithm

Take the update rule of equation (2). We choose the baseline as a discounted mean: $B^{(e)} = \frac{1-\gamma}{1-\gamma^{e-1}} \sum_{k=1}^{e-1} \gamma^{e-k-1} X^{(k)}$, with $B^{(1)} = 0$ and $\gamma = 0.99$. This algorithm and choice of baseline are similar to the REINFORCE algorithms introduced in [18], except that it accounts for history-dependent rewards and explicitly allows activity measures on state-action pairs.

Algorithm 1 describes the general algorithm Gradient Ascent Activity-based Credit Assignment (GAtACA), which assigns credits to actions in a series of decisions based on a reward $R^{(e)}$ collected at the end of each episode.

When $X^{(e)} = R^{(e)}$, the GAtACA algorithm can be seen as an extension of the usual REINFORCE algorithm to non-Markovian rewards. This baseline algorithm is coined REINFORCER, where R stands for the reward from the environment. Note that the usual REINFORCE algorithm assumes Markovian rewards. Reformulating our setting into a Markovian one, by aggregating states and actions to obtain one reward per (state, action) pair, for instance, would exponentially increase the complexity of the model, with one choice per possible trajectory, making it intractable.

When $X^{(e)} = R^{(e)}/M^{(e)}$, we will see that using the activity allows to obtain high reward trajectories with short length. In the next section, we compare GAtACA and our baseline, REINFORCER.

Algorithm 1 GAtACA(Environment $(\mathcal{S}, \mathcal{A}, \psi)$, $s^\star \in \mathcal{S}$, $\mathscr{T}$ vector of r trajectories, $\alpha > 0$, $\varepsilon > 0$)

Initialize: $e = 1$, $B^{(1)} = 0$, $\theta^{(1)}_{t,s,a} = 0$ for all t, s, a
repeat
  $t \leftarrow 1$ and $S_1^{(e)} \leftarrow s^\star$
  while $t \leqslant T^{(e)}$ as defined by Equation (1) **do**
    $A_t^{(e)} \sim$ categorical distribution with parameter $\mathrm{softmax}_{\mathcal{A}(S_t^{(e)})}(\theta_{t,S_t^{(e)},\cdot})$
    $S_{t+1}^{(e)} \leftarrow \psi(S_t^{(e)}, A_t^{(e)})$
    $t \leftarrow t + 1$
  end while
  $M^{(e)} \leftarrow \sum_{u=1}^{T^{(e)}-1} \mu(S_u^{(e)}, A_u^{(e)})$
  $R^{(e)} \leftarrow R(T^{(e)}, S^{(e)}, A^{(e)})$
  Compute $X^{(e)}$ from $R^{(e)}$ and $M^{(e)}$
  for all $t = 1, 2, \ldots, T^{(e)} - 1$ **do**
    for all $a \in \mathcal{A}(S_t^{(e)})$ **do**
      $\theta^{(e+1)}_{t,S_t^{(e)},a} \leftarrow \theta^{(e)}_{t,S_t^{(e)},a} + \alpha(\mathbb{1}_{A_t^{(e)}=a} - \mathrm{softmax}(\theta_{t,S_t^{(e)},\cdot})(a))(X^{(e)} - B^{(e)})$
    end for
  end for
  $B^{(e+1)} \leftarrow \frac{1-\gamma}{1-\gamma^e} \sum_{k=1}^{e} \gamma^{e-k} X^{(k)} = \frac{1}{1-\gamma^e}(X^{(e)} + \gamma(1-\gamma^{e-1})B^{(e)})$
  $e \leftarrow e + 1$
until $|X^{(e)} - B^{(e)}| < \varepsilon$
return $(\theta^{(e)}_{t,s,a})_{t,s,a}$.

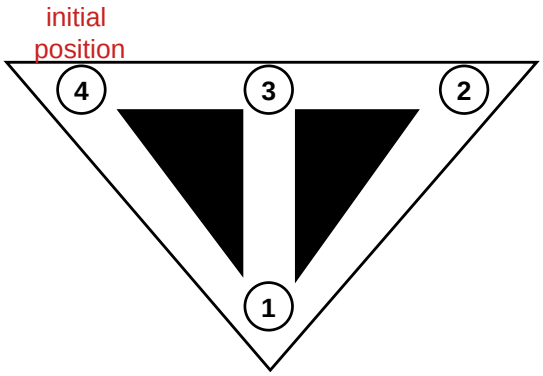
4 Application and results

4.1 Online learning of a behavioral task

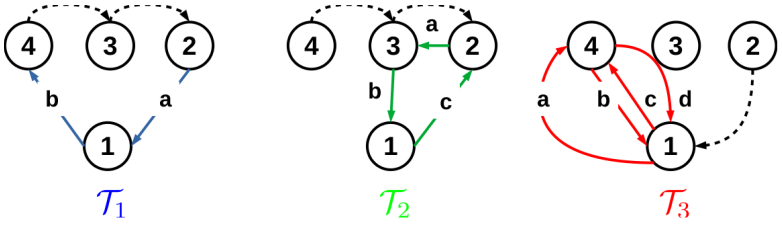
Consider the T-maze shown in Fig. 2a. There are $n = 4$ possible states, $\mathcal{S} = [n]$. Bottom and top intersections offer three possible actions (that is, walking along the corridor until the next intersection), while corners offer two. The activity $\mu(s, a)$ is the length of the corridor taken by choosing action a from crossroad s. The oblique corridors of the maze have length 5, the vertical corridor has length 4 and the two horizontal corridors each have length 3. This activity can be easily extended to more complex metrics, such as the cognitive activity defined as the number of cognitive processes involved, or the neuronal activity defined as the mean firing rate of neuronal networks, and the maze setting generalized to more complex environments, for instance with randomized transitions between states.

The goal of the agent (a rodent in [7]) is to learn the correct series of actions in the maze as shown in Fig. 2b. However, the agent does not know precisely which actions are correct or incorrect with respect to the target. It only receives a dose of sugar when finishing the trajectory[4]. Note that the last state of a target sub-trajectory may not be the first state of the next target sub-trajectory. Several

[4] The trajectories learned in [7] by real rodents were much simpler, to give them a chance of finding them.



(a) Structure of the T-maze: at each intersection and corner, the rodent can pick a corridor. The initial position is fixed in position 4.



(b) Indices of the 3 correct (target) series of actions : $\mathcal{T}_1^\star = (2, 1, 4)$, $\mathcal{T}_2^\star = (2, 3, 1, 2)$ and $\mathcal{T}_3^\star = (1, 4, 1, 4, 1)$. The order of the choices is indicated by characters: a, b, c and d. The shortest path to initiate the correct series of actions is indicated with dashed arrows.

Fig. 2: Maze choices and correct series of actions.

series of actions, including unnecessary loops, can be inserted between target sub-trajectories.

This application is a spatio-temporal credit assignment problem [3]. The spatial problem arises from the fact that sometimes, only a small subset of all possible states and actions are relevant or contribute to a correct path. For example, for the target sub-trajectory $\mathcal{T}_3^\star$ in Fig.2b, only the left and bottom corners are relevant. The temporal problem arises from the fact that a state can be encountered more than once during an episode, and the best action is not necessarily the same at different time steps. Thus, the problem is to identify the set of actions and their order to maximize the reward.

At each episode $e \geqslant 1$, the agent starts in a fixed state, the upper left corner, as shown in Fig. 2a. The learning phase starts with a uniform policy, and the credit vector is updated according to either the REINFORCER or GAtACA algorithms.

4.2 Results

In this section, we compare the REINFORCER algorithm, based only on the reward (objective $X = R$), to GAtACA, based on both reward and activity (objective $X = R/M$), for learning series of actions in the T-maze problem.

Fig. 3 presents the rewards and trajectory lengths obtained by both algorithms. Both are able to find trajectories containing the three correct target sub-trajectories in the right order that are shorter than the maximum trajectory length, $T_{\max} = 30$ in this example.

The trajectories generated by REINFORCER are longer than the ones generated by GAtACA, as shown in Fig. 3: the length of the trajectories sampled by GAtACA converges to 15 time steps, the minimum of steps to get the highest possible reward, while REINFORCER's average trajectory length remain larger than 17. Fig. 4, described below, illustrates the end policy of GAtACA and REINFORCER. REINFORCER's most likely trajectory contains an unnecessary loop: the actions between time steps 9 and 13 could be condensed into GAtACA's actions between time steps 9 and 11.

This is due to the fact that GAtACA accounts for the total distance travelled along the trajectory, in addition to the reward, while REINFORCER does not. For instance, at the beginning of each episode, the initial position of the agent is 4. The shortest path to initiate the first target sub-trajectory $\mathcal{T}_1^*$ is to go through state 3. REINFORCER does not differentiate between passing by 1 or 3, because it does not account for activities (corridor lengths in this case), but GAtACA does. Note that the objective that is optimized by GAtACA is the ratio of reward over activity: while in this situation it produces trajectories that are optimal for both reward and length, it is not always the case.

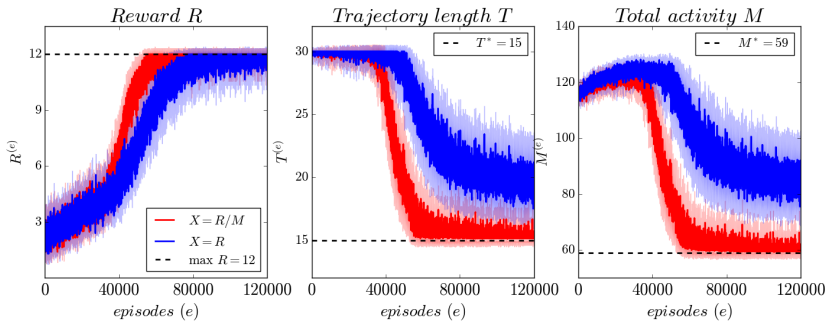


Fig. 3: Reward (left), trajectory length (middle) and activity (right) of the trajectories generated by REINFORCER (blue) and GAtACA (red) during the learning phase. The dashed lines show the maximum reward and minimum length of a trajectory with the maximum reward respectively. The curves are averaged over 30 realizations, the 95% confidence intervals (see Appendix A) are shown in a lighter color.

To simplify the visual representation of the policies in Figures 4 and 5 (in Appendix D), we pick one trajectory and, for each state along this trajectory, represent the policy vector at this time step in this state. This trajectory is taken as a greedy approximation $(T_{\max}, \hat{S}, \hat{A})$ of the most likely trajectory, defined as follows: $\hat{S}_1 = s^\star$ and for all $t \in [T_{\max} - 1]$

$$\begin{cases} \hat{A}_t = \min(\operatorname{argmax}_{a \in \mathcal{A}} \theta_{t,\hat{S}_t,a}), \\ \hat{S}_{t+1} = \psi(\hat{S}_t, \hat{A}_t). \end{cases} \tag{3}$$

Recall that for any t, s, a, the probability $\pi_\theta(A_t = a \,|\, S_t = s, t \leqslant T-1)$ that the agent chooses action a in state s at time t is given by $e^{\theta_{t,s,a}} / \sum_{b \in \mathcal{A}(s)} e^{\theta_{t,s,b}}$. Thus, maximizing $\pi_\theta(A_t = a \,|\, S_t = s, t \leqslant T-1)$ in a is equivalent to maximizing $\theta_{t,s,a}$ in a. Thus, this trajectory is built by picking the most probable action at each time step. When the policy converges to a Dirac distribution, this greedy approximation is eventually the most probable trajectory and converges the limit trajectory.

The temporal evolution of the policies at each time step, averaged over 30 realizations, for the REINFORCER and GAtACA algorithms, is shown in Fig. 5 of Appendix D. Fig. 4 shows the end policy of a single realization of REINFORCER and GAtACA, and shows that the REINFORCER policy does not converge to a Dirac distribution, while the GAtACA policy does. Note that in both cases, even if the policy does not converge to a Dirac, the expected reward under the policy does converge to the maximum possible reward, see Fig. 3.

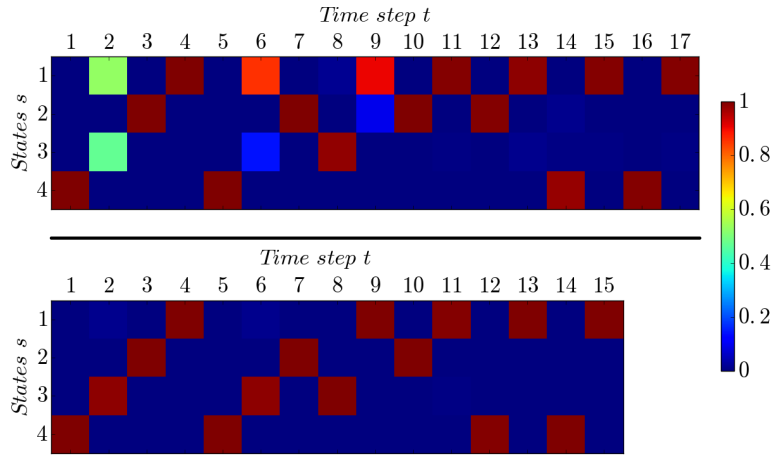


Fig. 4: Probability, for each time step t, to reach state s at time step $t+1$ when in the state $\hat{S}_t$ at time step t, according to the REINFORCER (top) and GAtACA (bottom) policies. $\hat{S}_t$ is the greedy approximation of the most likely trajectory, defined in Equation (3).

4.3 Link between GAtACA and REINFORCER

REINFORCER can be seen as an instance of GAtACA, with its objective function being chosen as $X = R$ instead of $X = R/M$. Beyond this connection, the two algorithms can be seen as the same gradient ascent algorithm as REINFORCER, with the difference that the learning rate α in REINFORCER is constant and the learning rate in GAtACA depends on the environment and trajectory: writing GAtACA as an instance of the REINFORCER algorithm is as simple as taking $\alpha_{\text{REINFORCER}} \leftarrow \frac{\alpha_{\text{GAtACA}}}{M^{(e)}}$ during episode e.

We take advantage of this similarity to choose α in order to have comparable rates of convergence. As shown in Fig. 3, the average activity of the trajectories at the start of the learning phase is around 120. We thus took $\alpha_{\text{GAtACA}} = 120\alpha_{\text{REINFORCER}}$, and indeed the evolution of the reward under the two policies is comparable in the early stages of Fig. 3. The rate of convergence under the two policies only start to diverge when the activity under GAtACA drops, which is indeed the kind of behavior expected when changing the learning rate of a gradient ascent algorithm.

Note that this does not solve the issue of choosing the best possible α, which is itself a known and challenging issue with gradient ascent/descent algorithms, see [10] and references therein. GAtACA's tendency to have larger learning rate when its activity is smaller could be combined with data-driven methods that select an intentionally underestimated learning rate α based on the first few episodes. This would hasten the convergence of the policy when the activity drops, which coincides with the drop of trajectory length.

5 Conclusion

We propose a general gradient ascent approach for solving the credit assignment assignment problem in case of history-dependent reward. This leads to the general form of the GAtACA algorithm, which optimizes an expected objective function X of the environment. This objective may involve a reward collected at the end of an episode as well as any other metrics, called activities, on the actions themselves. We prove that this algorithm converges to a zero of the gradient of the objective function.

In our example, we took the objective function as being the ratio of a benefit (reward) over a cost (sum of the activities, that is the total distance travelled) to show the interest of the activity-based credit assignment with respect to a more usual reward-based credit assignment. Activity-based credit assignment is shown to properly converge in the T-maze example where multiple optimal solutions are possible. This trade-off between reward and distance travelled proved to be a better way to solve the credit assignment problem than merely optimizing the expected reward, as it produces the shortest trajectories that still obtain the highest possible reward. Other activity metrics than corridor length, such as neuronal activity of the agent, as well as randomizing the state transition probabilities of the environment, could be worth exploring for further applications.

Our approach allows to account for non-Markovian rewards, where the best choice of action depends on past actions. We do so by allowing the credit of each action, and through them the probability of choosing these actions, to depend on the current time step, while keeping the choice independent on the past. In contrast, a simpler (possibly Partially Observed) Markovian Decision Process (MDP [5] and POMDP [17]) would have an exponentially larger complexity: instead of basing its decision on the current state, the agent would need to base it on the current state and a certain number of previous states and actions. When accounting for T past actions and n possible states, this results in n^T possible states, before even taking actions into account, compared to n with our approach, which quickly becomes intractable. Moreover, these models assume that a reward is made available at each time step, and that it depends only on the current state and action chosen, which is not the case here.

Acknowledgements This work has been supported by the French government, through the UCA^{JEDI} Investments in the Future project managed by the National Research Agency (ANR) with the reference number ANR-15-IDEX-01. It is part of the Computabrain project.

References

1. Bertsekas, D.P., Tsitsiklis, J.N.: Gradient convergence in gradient methods with errors. SIAM Journal on Optimization **10**(3), 627–642 (2000). https://doi.org/10.1137/S1052623497331063, https://doi.org/10.1137/S1052623497331063
2. Clarke, A.M., Friedrich, J., Tartaglia, E.M., Marchesotti, S., Senn, W., Herzog, M.H.: Human and machine learning in non-markovian decision making. PLOS ONE **10**(4), 1–15 (04 2015). https://doi.org/10.1371/journal.pone.0123105, https://doi.org/10.1371/journal.pone.0123105
3. Friedrich, J., Urbanczik, R., Senn, W.: Spatio-temporal credit assignment in neuronal population learning. PLoS computational biology **7**(6), e1002092 (2011)
4. Gaon, M., Brafman, R.: Reinforcement learning with non-markovian rewards. Proceedings of the AAAI Conference on Artificial Intelligence **34**(04), 3980–3987 (Apr 2020). https://doi.org/10.1609/aaai.v34i04.5814, https://ojs.aaai.org/index.php/AAAI/article/view/5814
5. Gosavi, A.: Simulation-based optimization: Parametric Optimization Techniques and Reinforcement Learning. Springer (2015)
6. Minsky, M.: Steps toward artificial intelligence. Proceedings of the IRE **49**(1), 8–30 (1961). https://doi.org/10.1109/JRPROC.1961.287775
7. Moussa, R., Poucet, B., Amalric, M., Sargolini, F.: Contributions of dorsal striatal subregions to spatial alternation behavior. Learning & memory **18**(7), 444–451 (2011)
8. Muzy, A.: Exploiting activity for the modeling and simulation of dynamics and learning processes in hierarchical (neurocognitive) systems. Computing in Science & Engineering **21**(1), 84–93 (2019)
9. Muzy, A., Zeigler, B.P.: Activity-based credit assignment heuristic for simulation-based stochastic search in a hierarchical model base of systems. IEEE Systems Journal **11**(4), 1916–1927 (2017)

10. Nemirovski, A., Juditsky, A., Lan, G., Shapiro, A.: Robust stochastic approximation approach to stochastic programming. SIAM Journal on optimization **19**(4), 1574–1609 (2009)
11. Petit, B., Amdahl-Culleton, L., Liu, Y., Smith, J., Bacon, P.L.: All-action policy gradient methods: A numerical integration approach. arXiv preprint arXiv:1910.09093 (2019)
12. Rens, G., Raskin, J.F., Reynoaud, R., Marra, G.: Online learning of non-markovian reward models. In: Proceedings of the Thirteenth International Conference on Agents and Artificial Intelligence. Scitepress (2020)
13. Sabri, O., Lehéricy, L., Muzy, A.: Multi-agent learning via gradient ascent activity-based credit assignment. Scientific Reports **13**(1), 15256 (Sep 2023). https://doi.org/10.1038/s41598-023-42448-9, https://doi.org/10.1038/s41598-023-42448-9
14. Sutton, R.S.: Learning to predict by the methods of temporal differences. Machine learning **3**(1), 9–44 (1988)
15. Sutton, R.S., McAllester, D., Singh, S., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. Advances in neural information processing systems **12** (1999)
16. Sutton, R.S.: Temporal credit assignment in reinforcement learning. Ph.D. thesis, University of Massachusetts Amherst (1984)
17. Toro Icarte, R., Waldie, E., Klassen, T., Valenzano, R., Castro, M., McIlraith, S.: Learning reward machines for partially observable reinforcement learning. Advances in neural information processing systems **32** (2019)
18. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning **8**(3), 229–256 (1992)

A Simulation environment and parameters

The simulation results in this paper are Monte Carlo simulations performed over $N = 30$ realizations, using the pseudo-random number generator Mersenne Twister from of library numpy 1.11.1, with *seed* = 7052984. The same seed is used for REINFORCER and GAtACA to guaranty the same environment for different learning objective.

Each realization, indexed by $r \in [N]$, at episode $e \geqslant 1$ produces $Y^{(e),[r]}$ for any variable Y ($R^{(e)}$ and $T^{(e)}$ in Figure 3). Therefore, the mean and the standard deviation of Y over the realizations is given as

$$\forall e \geqslant 1 \longmapsto \quad \overline{Y}^{(e)} = \frac{1}{N}\sum_{r=1}^{N} Y^{(e),[r]} = \text{mean}\Big(\big(Y^{(e),[r]}\big)_{r\in[N]}\Big),$$
$$\forall e \geqslant 1 \longmapsto \quad \sigma_Y^{(e)} = \text{SD}\Big(\big(Y^{(e),[r]}\big)_{r\in[N]}\Big).$$

The confidence interval (CI) at episode e is given as follow

$$\left[\overline{Y}^{(e)} - c\frac{\sigma_Y^{(e)}}{\sqrt{N}}, \overline{Y}^{(e)} + c\frac{\sigma_Y^{(e)}}{\sqrt{N}}\right],$$

where $c \approx 2.048$ is the 97.5^{th} percentile of a Student distribution with $N - 1 = 29$ degrees of freedom.

The total number of episodes is $1.5 \cdot 10^5$. In all simulations, we consider a constant learning rate $\alpha = \frac{0.07}{120}$ in REINFORCER (which optimizes $X = R$), and $\alpha = 0.07$ in GAtACA (which optimizes $X = R/M$), so that αX is of the same order in the two algorithms. The reasoning behind this choice of α is explained in Section 4.3.

B Proof of Theorem 1

Consider the objective function: $\theta \longmapsto \mathbb{E}_\theta[X^{(1)}]$. By definition,

$$\begin{aligned}\mathbb{E}_\theta[X^{(1)}] &= \sum_{(T,S,A)\ \text{trajectory}} \pi_\theta(T^{(1)} = T, S^{(1)} = S, A^{(1)} = A)\mathbb{E}[X^{(1)}|T^{(1)} = T, S^{(1)} = S, A^{(1)} = A] \\ &= \sum_{(T,S,A)\ \text{trajectory}} \mathbb{1}_{(T,S,A)\text{ is valid}}\mathbb{1}_{S_1=s^\star}\mathbb{P}(T|S,A)\left(\prod_{t=1}^{T-1}\frac{e^{\theta_{t,S_t,A_t}}}{\sum_{b\in\mathcal{A}(S_t)} e^{\theta_{t,S_t,b}}}\right)\nu_{T,S,A}\end{aligned}$$

Recall that the conditional expectation $\nu_{T,S,A} := \mathbb{E}[X^{(1)}|T^{(1)} = T, S^{(1)} = S, A^{(1)}AC]$ does not depend on θ.

Let us compute its gradient. Let $t \in [T_{\max}]$, $s \in \mathcal{S}$ and $a \in \mathcal{A}$.

$$\frac{\partial \mathbb{E}_\theta[X^{(1)}]}{\partial \theta_{t,s,a}} = \sum_{\substack{(T,S,A) \text{ trajectory}}} \mathbb{1}_{(T,S,A) \text{ is valid}} \mathbb{1}_{S_1=s^\star} \mathbb{P}(T^{(1)} = T \,|\, S^{(1)} = S, A^{(1)} = A) \nu_{T,S,A} \times \frac{\partial}{\partial \theta_{t,s,a}} \left(\prod_{u=1}^{T-1} \frac{e^{\theta_{u,S_u,A_u}}}{\sum_{b \in \mathcal{A}(S_u)} e^{\theta_{u,S_u,b}}} \right). \tag{4}$$

Then,

$$\frac{\partial}{\partial \theta_{t,s,a}} \left(\prod_{u=1}^{T-1} \frac{e^{\theta_{u,S_u,A_u}}}{\sum_{b \in \mathcal{A}(S_u)} e^{\theta_{u,S_u,b}}} \right) = \mathbb{1}_{t \leqslant T-1} \left(\prod_{\substack{u \in [T-1] \\ u \neq t}} \frac{e^{\theta_{u,S_u,A_u}}}{\sum_{b \in \mathcal{A}(S_u)} e^{\theta_{u,S_u,b}}} \right) \underbrace{\frac{\partial}{\partial \theta_{t,s,a}} \frac{e^{\theta_{t,S_t,A_t}}}{\sum_{b \in \mathcal{A}(S_t)} e^{\theta_{t,S_t,b}}}}_{(*)}, \tag{5}$$

and

$$\begin{aligned}
(*) &= \mathbb{1}_{S_t=s} \frac{\mathbb{1}_{A_t=a} e^{\theta_{t,s,A_t}} \sum_{b \in \mathcal{A}(s)} e^{\theta_{t,s,b}} - e^{\theta_{t,s,A_t}} e^{\theta_{t,s,a}}}{\left(\sum_{b \in \mathcal{A}(j)} e^{\theta_{t,s,b}} \right)^2} \\
&= \mathbb{1}_{S_t=s} \frac{e^{\theta_{t,s,A_t}}}{\sum_{b \in \mathcal{A}(j)} e^{\theta_{t,s,b}}} \left(\mathbb{1}_{A_t=a} - \frac{e^{\theta_{t,s,a}}}{\sum_{b \in \mathcal{A}(j)} e^{K_{t,s,b}}} \right) \\
&= \mathbb{1}_{S_t=s} \pi_\theta(A_t^{(1)} = A_t \,|\, S_t^{(1)} = s, t \leqslant T^{(1)} - 1)(\mathbb{1}_{A_t=a} - \pi_\theta(A_t^{(1)} = a \,|\, S_t^{(1)} = s, t \leqslant T^{(1)} - 1)).
\end{aligned}$$

Substituting $(*)$ in Eq. (5),

$$\begin{aligned}
\frac{\partial}{\partial \theta_{t,s,a}} \left(\prod_{u=1}^{T-1} \frac{e^{\theta_{u,S_u,A_u}}}{\sum_{b \in \mathcal{A}(S_u)} e^{\theta_{u,S_u,b}}} \right) &= \mathbb{1}_{t \leqslant T-1} \prod_{u \in [T-1]} \pi_\theta(A_u^{(1)} = A_u \,|\, S_u^{(1)} = S_u, T^{(1)} = T) \\
&\quad \times \mathbb{1}_{S_t=s} (\mathbb{1}_{A_t=a} - \pi_\theta(A_t^{(1)} = a \,|\, S_t^{(1)} = s, t \leqslant T^{(1)} - 1)),
\end{aligned}$$

so that the gradient of the objective function (Eq.4) becomes

$$\begin{aligned}
\frac{\partial \mathbb{E}_\theta[X^{(1)}]}{\partial \theta_{t,s,a}} &= \sum_{\substack{(T,S,A) \text{ trajectory}}} \pi_\theta(T^{(1)} = T, S^{(1)} = S, A^{(1)} = A) \\
&\quad \times \mathbb{1}_{t \leqslant T-1} \mathbb{1}_{S_t=s} (\mathbb{1}_{A_t=a} - \pi_\theta(A_t^{(1)} = a \,|\, S_t^{(1)} = s, t \leqslant T^{(1)} - 1)) \nu_{T,S,A} \\
&= \mathbb{E}_\theta[\mathbb{1}_{t \leqslant T^{(1)}-1} \mathbb{1}_{S_t^{(1)}=s} (\mathbb{1}_{A_t^{(1)}=a} - \pi_\theta(\tilde{A}_t^{(1)} = a \,|\, \tilde{S}_t^{(1)} = s, t \leqslant \tilde{T}^{(1)} - 1)) X^{(1)}],
\end{aligned}$$

where $(\tilde{T}^{(1)}, \tilde{S}^{(1)}, \tilde{A}^{(1)})$ is a copy of $(T^{(1)}, S^{(1)}, A^{(1)})$ that follows the same distribution. Finally, let $B^{(1)}$ be a random variable that is independent of $(S_t^{(1)}, A_t^{(1)})$

conditionally to $\{t \leqslant T^{(1)} - 1\}$, then

$$\begin{aligned}
&\mathbb{E}_\theta[\mathbb{1}_{t\leqslant T^{(1)}-1}\mathbb{1}_{S_t^{(1)}=s}(\mathbb{1}_{A_t^{(1)}=a} - \pi_\theta(\tilde{A}_t^{(1)} = a \,|\, \tilde{S}_t^{(1)} = s, t \leqslant \tilde{T}^{(1)} - 1))B^{(1)}] \\
&\quad = \mathbb{E}_\theta[\mathbb{1}_{t\leqslant T^{(1)}-1}\mathbb{E}_\theta[\mathbb{1}_{S_t^{(1)}=s}(\mathbb{1}_{A_t^{(1)}=a} - \pi_\theta(\tilde{A}_t^{(1)} = a \,|\, \tilde{S}_t^{(1)} = s, t \leqslant \tilde{T}^{(1)} - 1))B^{(1)} \,|\, t \leqslant T^{(1)} - 1]] \\
&\quad = \mathbb{E}_\theta[\mathbb{1}_{t\leqslant T^{(1)}-1}\mathbb{E}_\theta[\mathbb{1}_{S_t^{(1)}=s}(\mathbb{1}_{A_t^{(1)}=a} - \pi_\theta(\tilde{A}_t^{(1)} = a \,|\, \tilde{S}_t^{(1)} = s, t \leqslant \tilde{T}^{(1)} - 1)) \,|\, t \leqslant T^{(1)} - 1]\mathbb{E}_\theta[B^{(1)} \,|\, t \leqslant T^{(1)} - 1]] \\
&\quad = 0,
\end{aligned}$$

since

$$\begin{aligned}
&\mathbb{E}_\theta[\mathbb{1}_{S_t^{(1)}=s}(\mathbb{1}_{A_t^{(1)}=a} - \pi_\theta(\tilde{A}_t^{(1)} = a \,|\, \tilde{S}_t^{(1)} = s, t \leqslant \tilde{T}^{(1)} - 1)) \,|\, t \leqslant T^{(1)} - 1] \\
&\quad = \mathbb{E}_\theta[\mathbb{1}_{S_t^{(1)}=s} \underbrace{\mathbb{E}_\theta[\mathbb{1}_{A_t^{(1)}=a} - \pi_\theta(\tilde{A}_t^{(1)} = a \,|\, \tilde{S}_t^{(1)} = s, t \leqslant \tilde{T}^{(1)} - 1) \,|\, S_t^{(1)} = s, t \leqslant T^{(1)} - 1]}_{=0} \,|\, t \leqslant T^{(1)} - 1].
\end{aligned}$$

C Convergence of the algorithm

The proof is similar to the proof of Theorem 2 of [13].

We replace θ by $\widetilde{\theta}$ to solve convergence issues: the distributions $\pi_{\theta^{(e)}}$ and $\pi_{\widetilde{\theta}^{(e)}}$ are the same when the elements of $\theta^{(e)}$ are finite, but the limit of $(\pi_{\theta^{(e)}})_{e\geqslant 1}$ may not be well-defined when there exists t, s, a and $a' \neq a$ such that $\theta^{(e)}_{t,s,a}$ and $\theta^{(e)}_{t,s,a'}$ tend to $+\infty$ at the same time. In contrast, for any sequence $\widetilde{\theta}^{(e)} \longrightarrow \widetilde{\theta}^{\infty} \in \Theta$, $\pi_{\widetilde{\theta}^{(e)}} \longrightarrow \pi_{\widetilde{\theta}^{\infty}}$.

An example where the gradient is zero is when $\#\{a : \widetilde{\theta}_{t,s,a} = -\infty\} = |\mathcal{A}(s)| - 1$ for all t, s: in this case, the agent always makes the same choice when in state s at time t. Any finite change of θ does not change the distribution of the trajectory chosen, and hence does not change $\mathbb{E}_\theta[X^{(1)}]$.

Note that this theorem requires the step size to tend to zero, but not too fast. This is a usual assumption in gradient descent, though how to choose α_e in practice is a delicate issue with no universal answer. In our algorithm and Equation (2), we chose to take α_e constant. Despite not being covered by the above theorem, the simulations show that the algorithm does converge toward the optimal solution.

Proof. Proposition 3 of [1] allows to prove the convergence of algorithms of the form

$$\theta^{(e+1)} = \theta^{(e)} + \alpha_e(s^{(e)} + w^{(e)}),$$

where $s^{(e)}$ is an approximation of the gradient of a target function $\theta \mapsto f(\theta)$ in $\theta^{(e)}$ and $w^{(e)}$ is a perturbation of this gradient. In our case, $f(\theta) = \mathbb{E}_\theta[X^{(1)}]$, $s^{(e)} = \nabla f(\theta^{(e)})$ and

$$w^{(e)} = (\mathbb{1}_{A_t^{(e)}=a} - \pi_{\theta^{(e)}}(A_t^{(e)} = a | S_t^{(e)} = s, t \leqslant T^{(e)} - 1))(X^{(e)} - B^{(e)}) - s^{(e)}.$$

The assumptions of Proposition 3 of [1] on $s^{(e)}$ are clearly satisfied, and the assumptions on $w^{(e)}$ are checked by Theorem 1 and by the fact that $w^{(e)}$ is bounded by $4r$.

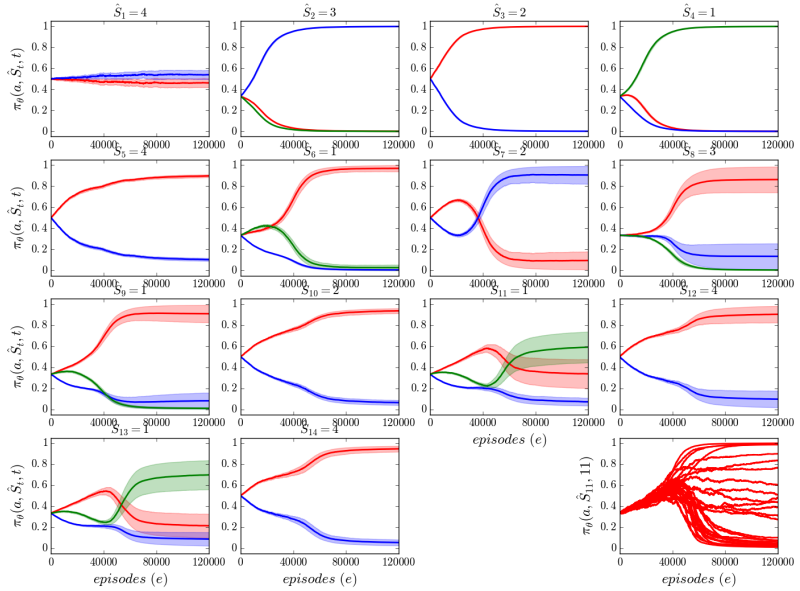
Our addition to their result, that the limit points belong to the same convex component, follows from the fact that $\|\widetilde{\theta}^{(e+1)} - \widetilde{\theta}^{(e)}\|_\infty \leqslant 2r\alpha_e \longrightarrow 0$. Thus, by compactness of Θ, any two limit points of $(\widetilde{\theta}^{(e)})_{e\geqslant 1}$ are connected by a continuous path of limit points of $(\widetilde{\theta}^{(e)})_{e\geqslant 1}$, and thus are in the same connex component.

D Additional figures

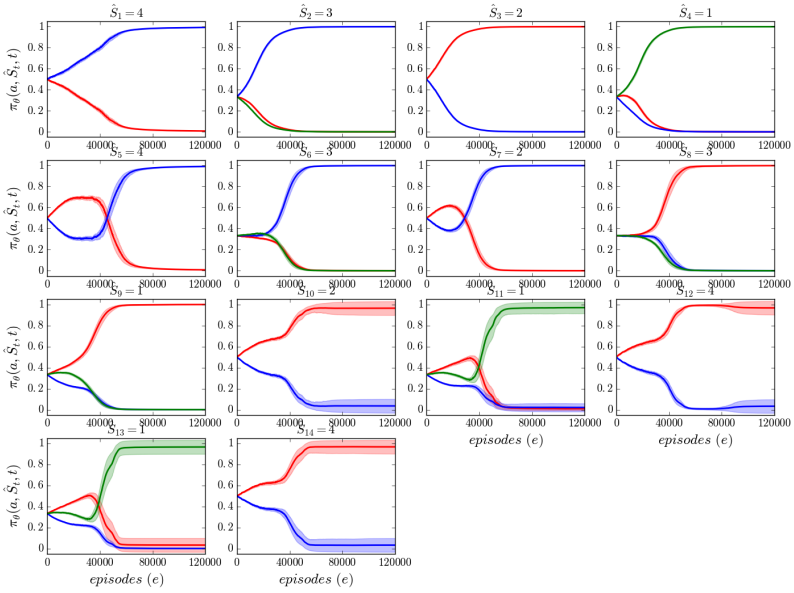
In this Section, we show the evolution of the policies REINFORCER and GAtACA during the learning phase.

Fig. 5 shows the evolution of the policies over time along the greedy approximation $(T_{\max}, \hat{S}, \hat{A})$ of the most likely trajectory, based on the policy whose credit is the average of the credits returned at the end of the $N = 30$ realizations of the algorithms. Note that there is no reason for the greedy approximation chosen for REINFORCER and GAtACA to be the same, and indeed $\hat{S}_2$ is different for REINFORCER and GAtACA in Fig. 5.

The GAtACA policies converge toward a Dirac distribution (since their component probabilities tend to either 0 or 1), while the REINFORCER policies remain hesitant at several time steps.



(a) REINFORCER policy evolution. The bottom right graph shows all $N = 30$ realizations contributing to the red mean curve for time step $t = 11$, showing that the policy did not converge.



(b) GAtACA policy evolution.

Fig. 5: Policy components $\pi_\theta(a, \hat{S}_t, t) := \pi_\theta(A_t^{(1)} = a \,|\, S_t^{(1)} = \hat{S}_t, t \leqslant T^{(1)} - 1)$ when in state $\hat{S}_t$ at time step t (only the first 14 time steps are shown) for REINFORCER (a) and GAtACA (b). $\hat{S}_t$ is the greedy approximation of the most likely trajectory, defined in Equation (3), following the policy based on the final credit averaged over the $N = 30$ realizations. The full line is the mean policy value, the diffuse region is the 95% confidence interval, defined in Section A. The state of the trajectory at each time step is written at the top of each graph.