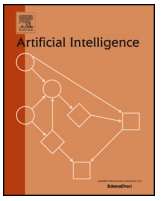
Artificial Intelligence 347 (2025) 104382

Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint



Reinforcement learning in convergently non-stationary environments: Feudal hierarchies and learned representations ☆

Diogo S. Carvalho *, Pedro A. Santos, Francisco S. Melo

Instituto Superior Técnico, University of Lisbon, and INESC-ID, Portugal

ARTICLE INFO

Keywords:
Reinforcement learning
Stochastic approximation

ABSTRACT

We study the convergence of Q-learning-based methods in convergently non-stationary environments, particularly in the context of hierarchical reinforcement learning and of dynamic features encountered in deep reinforcement learning. We demonstrate that Q-learning achieves convergence in tabular representations when applied to convergently non-stationary dynamics, such as the ones arising in a feudal hierarchical setting. Additionally, we establish convergence for Q-learning-based deep reinforcement learning methods with convergently non-stationary features, such as the ones arising in representation-based settings. Our findings offer theoretical support for the application of Q-learning in these complex scenarios and present methodologies for extending established theoretical results from standard cases to their convergently non-stationary counterparts.

1. Introduction

In reinforcement learning (RL), an agent interacts with its environment to maximize the accumulation of time-discounted rewards [32]. At each time step, the environment exists in a particular state; the agent selects and executes an action, which causes the environment to transition to a new state and provides the agent with a reward. Through a sequence of these interactions, the agent learns the optimal value of performing each action in each state, enabling it to achieve its objective by consistently selecting the action with the highest value [29]. The most prevalent algorithm for learning optimal action values is Q-learning, which guarantees convergence to the correct solution under certain conditions [35].

However, Q-learning is often applied in scenarios that violate a crucial assumption: that the environment is stationary. This assumption implies that the consequences of performing a given action in a specific state remain constant over time. For example, in hierarchical reinforcement learning, this assumption is compromised [11]. Focusing on feudal hierarchies [5], the agent operates at multiple levels of abstraction. At a high level, the agent selects a goal, while at a lower level, it strives to achieve that goal. Hierarchical reinforcement learning offers numerous benefits, including efficient learning, effective exploration, transferability, and interpretability. However, because learning occurs at both levels, the environment appears non-stationary from the high-level perspective. We are able to show that under certain conditions, despite non-stationary, the environment is convergent. Thus, we start by focusing on the convergence of Q-learning in convergently non-stationary environments (CNEs), particularly within feudal hierarchical reinforcement learning.

☆ This paper is an invited revision of a paper which first appeared at the 2023 European Conference on Artificial Intelligence (ECAI-23).

* Corresponding author.
E-mail address: diogo.s.carvalho@tecnico.ulisboa.pt (D.S. Carvalho).

https://doi.org/10.1016/j.artint.2025.104382
Received 16 December 2024; Received in revised form 7 May 2025; Accepted 7 June 2025

Available online 13 June 2025

But the reinforcement learning problem can be non-stationary, from the agent's point of view, even if the environment itself is stationary. For example, when there are too many states, it is common to use Q-learning with function approximation, where instead of directly observing the state of the environment, the agent observes a fixed set of features of the states and actions. If these features are not tabular representations of the states, the setting is troublesome, but there are a few results that establish convergence of regularized variations of Q-learning. However, normally, these features are not fixed and are updated throughout learning, making the environment non-stationary. We also investigate the convergence of regularized Q-learning with these CNEs with respect to the features.

In this work, we show that (i) tabular Q-learning converges with non-stationary convergent dynamics, consequently proving convergence of Q-learning in feudal hierarchies, and that (ii) regularized Q-learning with function approximation converges with non-stationary convergent features, consequently proving convergence of regularized Q-learning with deep neural networks whose hidden representations are updated through stochastic gradient descent over well-defined loss functions.

This work is significant to the field of Artificial Intelligence (AI) as it addresses key challenges in reinforcement learning related to dynamic and complex environments. Many AI applications, such as autonomous systems, robotics, and large-scale decision-making, operate in settings where assumptions of stationarity are violated, making theoretical guarantees of convergence critical. By demonstrating the convergence of Q-learning-based methods in CNEs, this work strengthens the foundational understanding of reinforcement learning. Furthermore, these results support the integration of deep neural networks into reinforcement learning frameworks, paving the way for more reliable, adaptive, and scalable AI systems in real-world scenarios.

2. Background

In this section, we formalize the process of interaction of an agent with its environment as a Markov decision process [29] and the reinforcement learning methods [32]. Beforehand, we provide notation to be used throughout the document.

Notation We denote random variables using upright letters, as in x or a. We use lowercase letters to denote functions, as in v or q, and calligraphic letters to denote sets, as in $\mathcal{X}$ or $\mathcal{A}$. Vectors are represented as bold lowercase letters. For example, $\mathbf{z}$ denotes a random vector and $\boldsymbol{z}$ an instance thereof. Operators are represented as upright uppercase letters such as H. Matrices are represented using bold uppercase letters, as in $\boldsymbol{M}$. If $\mathbf{z}$ is a random vector taking values in $\mathcal{Z}$ with distribution $\mu \in \Delta(\mathcal{Z})$, we use $\boldsymbol{z} \sim \mu$ or simply $\boldsymbol{z} \sim \mathbf{z}$ to denote that $\boldsymbol{z}$ is sampled according to the distribution μ and that we use $\mathbf{y} \sim \mu$ or just $\mathbf{y} \sim \mathbf{z}$ if $\mathbf{y}$ is a r.v. with the same distribution as $\mathbf{z}$. We write $\mathbb{E}_{\mathrm{x},\mathrm{a}\sim\mu}[f(\mathrm{x},\mathrm{a})]$ or simply $\mathbb{E}_{\mu}[f(\mathrm{x},\mathrm{a})]$ to denote the expectation of f when the random variables x and a follow distribution μ.

2.1. Markov decision processes

A (finite) Markov decision process (MDP) is a tuple $(\mathcal{X}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{X}$ is a finite set of n states, called the state space; $\mathcal{A}$ is a finite set of m actions, called the action space and P defines, for each state and action, a probability distribution over next states[1]; $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is the expected reward function and $\gamma \in [0, 1)$ is the discount factor. At time step t, the random variable x_t models the state of the process and the random variable a_t models the action taken by the agent. The random variable r_t models the reward received by the agent for taking action $a_t \sim \mathrm{a}_t$ in state $x_t \sim \mathrm{x}_t$ in a way that $\mathbb{E}\left[\mathrm{r}_t \mid \mathrm{x}_t = x, \mathrm{a}_t = a\right] = r(x, a)$. In our work we assume the expected reward in a bounded interval $[-r_{\max}, r_{\max}]$. After taking action a_t in state x_t, the environment transitions to a new state x_{t+1} according to the transition probabilities $P\left(x_t, a_t\right)$.

A policy is an agent's decision rule. The policy defines, for each state, a probability distribution over actions. We can, therefore, define that a policy is $\pi : \mathcal{X} \to \Delta(\mathcal{A})$.[2] A single interaction of an agent with its Markovian environment thus generates a tuple of state, action, reward and next state (x_t, a_t, r_t, x'_t). The Markov decision problem is the one of finding the policy that maximizes the expected sum of discounted rewards accumulated throughout the interaction of the agent with the environment, starting from a certain state. Formally, given a policy, we define the state value function $v^{\pi} : \mathcal{X} \to \mathbb{R}$ as

$$v^{\pi}(x) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \mathrm{r}_t \mid \mathrm{x}_0 = x\right], \tag{1}$$

where the expectation is taken with respect to the sequence of actions chosen, states visited and rewards received, and $\gamma \in [0, 1[$ is a discount factor.[3] The value function verifies the fixed-point equation:

$$v^{\pi}(x) = \mathbb{E}_{\pi}\left[\mathrm{r}_t + \gamma v^{\pi}(\mathrm{x}_{t+1}) \mid \mathrm{x}_t = x\right], \tag{2}$$

taking the expectation with respect to the action (implicit), reward and next-state. There exists a policy that maximizes the value of all states [29, Section 6.2]. We refer to it as optimal policy and denote it as π^* and its state value function as v^*. We also define the state-action value function, or just value function, as the expected sum of discounted rewards accumulated throughout the interaction of an agent with its environment, starting from a certain state and choosing a certain action, as $q^{\pi} : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ such that

[1] We can also represent P_a as an $n \times n$ matrix with component xx' given by $\mathbf{P}(\mathrm{x}_{t+1} = x' \mid \mathrm{x}_t = x, \mathrm{a}_t = a)$.

[2] We can also represent π as an $n \times m$ matrix, with component xa given by $\mathbf{P}(\mathrm{a}_t = a \mid \mathrm{x}_t = x)$ if $a_t \sim \pi(x_t)$.

[3] The discount factor guarantees that the sum converges and is, intuitively, an inverse of inflation rate.

$$q^{\pi}(x,a) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \mathrm{r}_t \mid \mathrm{x}_0 = x, \mathrm{a}_0 = a\right],$$

where the expectation is taken with respect to the same quantities as before. We have that

$$v^{\pi}(x) = \mathbb{E}_{\pi}\left[q^{\pi}(x,\mathrm{a})\right],$$

taking the expectation with respect to the actions $a \sim \pi(\cdot \mid x)$, and that the state-action value function verifies a fixed-point equation similar to (2). Specifically,

$$q^{\pi}(x,a) = \mathbb{E}_{\pi}\left[\mathrm{r}_t + \gamma q^{\pi}(\mathrm{x}_{t+1}, \mathrm{a}_{t+1}) \mid \mathrm{x}_t = x, \mathrm{a}_t = a\right], \tag{3}$$

where $\mathrm{a}_{t+1} \sim \pi(\cdot \mid x_{t+1})$ and the expectation is with respect to state, action, reward, next state and next action. We refer to the problem of evaluating a policy π, i.e., computing v^{π} or q^{π}, as the prediction problem. Then, we refer to the state-action value function of an optimal policy π^* as q^* and we know it verifies the Bellman fixed-point equation

$$q^*(x,a) = \mathbb{E}\left[\mathrm{r}_t + \gamma \max_{a' \in \mathcal{A}} q^*(\mathrm{x}_{t+1}, a') \mid \mathrm{x}_t = x, \mathrm{a}_t = a\right], \tag{4}$$

taking the expectation with respect to the state, action, next state and reward. The operator on the right-hand side of Equation (4) is called the Bellman operator, and can be defined from the space of value functions to itself such that $(\mathrm{H}q)(x,a) = \mathbb{E}\left[\mathrm{r}_t + \gamma \max_{a' \in \mathcal{A}} q^*(\mathrm{x}_{t+1}, a') \mid \mathrm{x}_t = x, \mathrm{a}_t = a\right]$. Knowing the optimal value function implies knowing an optimal policy.

Remark 1. Given the optimal value function q^*, an optimal policy π^* can be obtained directly from q^*, having $\pi^*(a \mid x) = 0$ if $a \notin \arg\max_{a' \in \mathcal{A}} q^*(x,a')$.

A policy defined from a value function as in the remark above is called a greedy policy. The implication of Remark 1 is that, if we want to compute an optimal policy, i.e., solve a Markov decision problem,[4] we can alternatively compute the optimal value function, from which an optimal policy is readily available. In our work, we focus on computing the optimal value function q^*.

When a model of the environment is available to the agent, including the transition probability matrices and the expected reward function, planning methods such as policy iteration and value iteration can be used to solve the Markov decision problem. When such model is not available, reinforcement learning methods, based on trial-and-error, can be used instead.

2.2. Reinforcement learning

Q-learning Q-learning is a reinforcement learning algorithm that learns the optimal value function, regardless of the policy the agent uses to interact with the environment. For that reason, it is called an off-policy algorithm. We know that the optimal value function verifies, at all state-action pairs, the Bellman fixed-point equation (4), which we write:

$$q^*(x,a) = \left(\mathrm{H}q^*\right)(x,a).$$

We can rearrange the equation to obtain

$$\mathbb{E}\left[\mathrm{r}_t + \gamma \max_{a' \in \mathcal{A}} q^*(\mathrm{x}_{t+1}, a') - q^*(\mathrm{x}_t, \mathrm{a}_t) \mid \mathrm{x}_t = x, \mathrm{a}_t = a\right] = 0 \tag{5}$$

for all state-action pairs. We define Q-learning with the stochastic update

$$q^*_{t+1}(x_t,a_t) = q^*_t(x_t,a_t) + \alpha_t\left(r_t + \gamma \max_{a' \in \mathcal{A}} q^*_t(x_{t+1},a') - q^*_t(x_t,a_t)\right).$$

Q-learning converges to q^* as long as state-action pairs are visited infinitely often [35]. We provide the background on stochastic approximation, which can be used to derive and analyze the convergence of the stochastic approximation algorithms presented in Appendix A.

Q-learning with function approximation Suppose now that there are too many states or actions, and we wish to approximate q^* verifying the fixed point equation (4) in a set of parameterized functions $\mathcal{Q} = \{q_{\boldsymbol{w}} : \mathcal{X} \times \mathcal{A} \to \mathbb{R}, \boldsymbol{w} \in \mathbb{R}^K\}$. Given a distribution μ over state-action pairs, reflecting the relevance or frequency of each pair, we can define an optimization problem such as

$$\arg\min_{\boldsymbol{w} \in \mathbb{R}^K} \frac{1}{2}\mathbb{E}_{\mu}\left[\left(\left(\mathrm{H}q_{\boldsymbol{w}}\right)(\mathrm{x},\mathrm{a}) - q_{\boldsymbol{w}}(\mathrm{x},\mathrm{a})\right)^2\right].$$

Now, we should differentiate the objective with respect to $\boldsymbol{w}$ and equaling 0. However, being a semi-gradient method, for Q-learning with function approximation we do not differentiate the target $\mathrm{H}q_{\boldsymbol{w}}$, only $q_{\boldsymbol{w}}$. Full-gradient or residual algorithms are generally slower and may converge to incorrect solutions [32]. In the semi-gradient case we are interested in, we obtain

[4] We can also refer to the Markov decision problem as the control problem.

$$\mathbb{E}_{\mu}\left[\left(\left(\mathrm{H}q_{\boldsymbol{w}}\right)(\mathrm{x},\mathrm{a})-q_{\boldsymbol{w}}(\mathrm{x},\mathrm{a})\right)\nabla q_{\boldsymbol{w}}(\mathrm{x},\mathrm{a})\right]=0.$$

We then have an equation that is of the form over which we can build a stochastic approximation algorithm:

$$\mathbb{E}_{\mu}\left[\left(\mathrm{r}+\gamma\max_{a'\in\mathcal{A}}q_{\boldsymbol{w}}(\mathrm{x}',a')-q_{\boldsymbol{w}}(\mathrm{x},\mathrm{a})\right)\nabla q_{\boldsymbol{w}}(\mathrm{x},\mathrm{a})\right]=0.$$

The Q-learning algorithm with function approximation becomes

$$\boldsymbol{w}_{t+1}=\boldsymbol{w}_t+\alpha_t\left(r_t+\gamma\max_{a'\in\mathcal{A}}q_{\boldsymbol{w}_t}(x_t',a')-q_{\boldsymbol{w}_t}(x_t,a_t)\right)\nabla q_{\boldsymbol{w}_t}(x_t,a_t).$$

Even though the algorithm can, in general, diverge [33,2], the introduction of regularizers guarantees the convergence of a variation of Q-learning [23,38], for instance through the update

$$\boldsymbol{w}_{t+1}=\boldsymbol{w}_t+\alpha_t\left(r_t+\gamma\max_{a'\in\mathcal{A}}q_{\boldsymbol{w}_t}(x_t',a')-\xi q_{\boldsymbol{w}_t}(x_t,a_t)\right)\nabla q_{\boldsymbol{w}_t}(x_t,a_t)-\alpha_t\eta\boldsymbol{w}_t,$$

where $\xi,\eta>0$ regularize the values and the parameters, respectively. $\xi=1$ and $\eta=0$ result in the original algorithm, and increasing any of these parameters increases regularization. We call the resulting algorithm regularized Q-learning. In the case of linear function approximation, each value function $q_{\boldsymbol{w}}$ is given by a linear combination of given features $\phi:\mathcal{X}\times\mathcal{A}\to\mathbb{R}^K$, i.e., $q_{\boldsymbol{w}}(x,a)=\phi^T(x,a)\boldsymbol{w}$. In this case, $\nabla q_{\boldsymbol{w}_t}(x,a)=\phi(x,a)$.

While the standard setting of Q-learning is well investigated, it is not common to take into account non-stationarities in the analysis. Non-stationarities are a reality in many predicted applications of reinforcement learning, be it because the environment changes as a response to changes in the transition dynamics, reward functions etc., or because the features change. In this work, we address this gap, by framing hierarchical reinforcement learning, where the environment is non-stationary, and deep reinforcement learning with supervised representation learning signals.

2.3. Convergently non-stationary environments

In this work we introduce and analyze a setting where the environment, from the point of view of the agent, is non-stationary but convergent.

Definition 1 *(Convergently non-stationary environments (CNEs))*. We define a convergently non-stationary environment (CNE) as a reinforcement learning setting in which the agent interacts with a process whose effective structure evolves over time but converges in the limit.

This includes two cases:

1. **Converging dynamics:** The environment is defined by a sequence of Markov Decision Processes $(\mathcal{X},\mathcal{A},P_t,r_t,\gamma)$, where the transition probabilities and reward function converge pointwise:

$$P_t(x'\mid x,a)\to P^*(x'\mid x,a),\quad r_t(x,a)\to r^*(x,a),\quad \text{for all } (x,a).$$

2. **Converging representations:** The agent uses time-dependent features $\phi_{\boldsymbol{u}_t}:\mathcal{X}\times\mathcal{A}\to\mathbb{R}^K$ to evaluate state-action pairs, where the parameter $\boldsymbol{u}_t\in\mathbb{R}^D$ evolves over time and converges to a limit u^*, and ϕ is Lipschitz-continuous in $\boldsymbol{u}_t$. Consequently, the feature map converges pointwise:

$$\phi_{\boldsymbol{u}_t}(x,a)\to\phi_{\boldsymbol{u}^*}(x,a),\quad \text{for all } (x,a).$$

In both cases, the non-stationarity vanishes asymptotically.

In our work, we establish conditions under which Q-learning converges in CNEs of both types. In the case of converging representations, we assume an offline reinforcement learning setting in which the data distribution is fixed and independent of the current value estimates, ruling out cyclic dependencies between feature learning and value learning.

3. Related work

While the convergence of Q-learning with linear function approximation, including tabular representations, has been the focus of substantial research, the inclusion of non-stationarities has not been considered in many analyses.

There is work on the asymptotic behavior of reinforcement learning in structured non-Markovian or abstracted environments. Li et al. [21] offer a unified framework for state abstraction in MDPs and analyze whether various forms of abstraction preserve optimality and learnability. Hutter [12] analyzes state aggregation schemes that go beyond MDPs, showing that optimal value functions can still be recovered from aggregated representations under certain structural assumptions. Most relevant to our work, Majeed and Hutter [26] prove that Q-learning can converge in a class of non-MDPs, provided that the optimal Q-values are history-independent

throughout the interaction. Hutter et al. [13] characterize convergence of TD-like algorithms under function approximation, identifying conditions on the shape of the feature space necessary for correct learning. While these contributions extend reinforcement learning theory beyond the classical Markov decision process setting, they typically assume that the value function is stationary or that the underlying environment does not change during learning. In contrast, we consider environments that are explicitly non-stationary—either due to changing dynamics or evolving representations—but which converge in the limit. Our results address a gap left open by these earlier analyses, as we do not assume history independence or ergodicity, and instead directly analyze convergence under convergent non-stationarity.

The case of deep reinforcement learning, where the features change throughout learning, has also been considered in some recent works. One approach proposes a loss function that decreases over time under the assumption that a target network converges at each step [34]. However, having a monotonically decreasing loss does not guarantee that the learned parameters or Q-values themselves converge. Cai et al. [4] and Xu and Gu [36] provide finite-time convergence guarantees for Q-learning with over-parameterized neural networks. These results hold in the infinite-width limit, where the architecture approximates a tabular representation, but do not imply convergence in practical finite settings.

In this work, we aim to shed light on the asymptotic behavior of Q-learning in the presence of structured non-stationarity. Specifically, we analyze convergence in convergently non-stationary environments, including both evolving dynamics, such as those induced by hierarchical learning, and evolving feature representations, as occur in deep reinforcement learning. We provide theoretical guarantees for convergence in both cases, under assumptions that reflect the practical structure of these settings.

4. Non-stationary dynamics

We start by focusing on CNEs where the transition or reward functions change over time, resulting in a sequence of Markov decision processes $\mathcal{M}_t = (\mathcal{X}, \mathcal{A}, P_t, r_t)$. Even though we provide a convergence result that is general for CNEs with respect to the dynamics, we focus the exposition on the particular case of feudal hierarchies.

4.1. Feudal hierarchies

Feudal hierarchies decompose a complex decision-making problem into multiple layers of abstraction, where each layer is controlled by a different agent. As introduced by Dayan and Hinton [5], a higher-level agent sets abstract goals for a lower-level agent, which in turn performs primitive actions to achieve these goals during a low-level episode. At the end of this episode, the high-level agent issues a new goal and the process repeats. This approach is particularly useful in hierarchical reinforcement learning, where it enables temporal abstraction and better exploration.

However, when learning policies at both levels simultaneously, a source of non-stationarity emerges: the environment faced by the high-level agent depends on the evolving behavior of the low-level agent. For example, if the high-level agent selects a meaningful goal but the low-level agent fails to achieve it, due to being early in training, then the high-level decision may be incorrectly penalized. This feedback loop makes it difficult to assess the long-term impact of high-level decisions and can hinder convergence [14,20,28].

Our goal in this section is to show that under mild assumptions, this source of non-stationarity can be handled: we prove that Q-learning at the high level converges, provided that the low-level policy converges. To do so, we first formalize the feudal structure as a pair of interconnected Markov decision processes, analyze the stability of their interaction, and then establish convergence of the high-level learning process.

Let $\mathcal{M} = (\mathcal{X}, \mathcal{A}, P, r, \gamma)$ be a standard MDP. We use superscripts h and l to denote the high-level and low-level components of a two-level hierarchy, respectively. We define the hierarchical MDP as a pair $(\mathcal{M}^h, \mathcal{M}^l)$:

$$\mathcal{M}^h(\pi^l) := \left(\mathcal{X}, \Omega, P^h(\pi^l), r^h(\pi^l), \gamma^h\right),$$
$$\mathcal{M}^l := \left(\mathcal{X} \times \Omega, \mathcal{A}, P^l, r^l, 1\right).$$

Here, Ω is the high-level action space (also called the *goal space*), and simultaneously part of the state space for the low-level agent. The transition function P^h and reward function r^h in the high-level MDP depend on the policy π^l of the low-level agent. The low-level MDP takes as input the current state and goal, and outputs primitive actions.

This formalism captures the non-stationarity faced by the high-level agent: as π^l changes during training, so do P^h and r^h.

4.2. Example

Suppose our shallow reinforcement learning agent, corresponding to the original Markov decision process $\mathcal{M}$, is an ant set on a 7×7 maze grid world depicted in Fig. 1. The example is inspired by the Ant Maze environment from [6], which was also adapted by [28], and by the Windy Maze environment by [22]. Even though cells are numbered, we omit the numbers from the figure to avoid visual overhead. The goal state of the environment is the southwestern-most cell on the grid. The ant receives a positive unit reward at the goal state and null otherwise, describing the reward structure r. Adding some difficulty to the ant's navigation task, the grid world is set on a windy place. At each time step, the wind blows *north*, *south*, *east* or *west* with uniform equal probability. The pairs composed of the ant's position and the wind direction form the state space $\mathcal{X}$. The ant is not strong enough to move in a contrary direction to the wind and, therefore, the ant can either stand at its current position or let itself move in the direction of the wind.

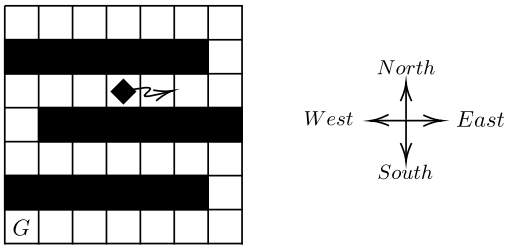


Fig. 1. Depiction of the windy ant maze grid world. The goal state is marked as G. The position of the ant is marked with the diamond. The wind direction is indicated as a bent arrow. At the current time step, wind is blowing *east*. The optimal action in the shallow environment is therefore to *stand*. In the hierarchical environment, the optimal goal to be set by the agent at the high level is *west*. With the current wind direction, for such goal, the optimal action at the low level is to *stand*, which gives a null reward.

Both actions are successful unless the wind blows the ant toward a wall cell, in which case the ant stays in the same cell, forming the transition probabilities P. The action space $\mathcal{A}$ is then composed of the actions *stand* and *move*.

Now we move on to describing a feudal hierarchical structure over the Markov decision process presented. We start with the high level $\mathcal{M}^h$. At the high level, the agent has the same state space $\mathcal{X}$ as in the shallow environment, observing both the position of the ant and wind direction. It can then set a goal for the low level to achieve. Such goals are to move *north*, *south*, *east* or *west*, forming the goal action space Ω. The high level observes a new state and outputs a new goal after the low-level episode terminates. The number of low-level time steps between high-level observations is random with a geometric distribution of $\gamma \in [0, 1)$. The transition probabilities depend on the policy operating at the low level as well as the termination of the low-level episode. The reward received by the high level is the sum, over the length of the low-level episode, of rewards it would have received in the shallow environment, r. If the high-level discount factor is $\gamma^{\frac{1}{\gamma}}$, on average, the rewards of the high level are discounted by the same factor as in the shallow Markov decision process, γ.

Finally, we focus on the low-level process $\mathcal{M}^l$. The state of the agent at the low level is the triple of ant's position, wind direction and goal. Its action space is equal to $\mathcal{A}$. When the low level takes an action, the transitions happen as in the shallow environment, modeled by P^l, regardless of the goal. The low-level agent receives a positive unit reward if it moves in the direction of the goal, a negative unit reward if it moves in a direction away from the goal, and a null reward if its position remains unchanged, forming r^l.

We establish our results under the following assumption, which can be described as having a uniform stopping probability at each step of the low level.

Assumption 1. The stopping time of a low-level episode, τ, is a random variable supported in $\mathbb{N}_0$ with probability mass $\xi(t) := P(\tau = t) = (1-\gamma)\gamma^t$.

Under the assumption, we establish that the high-level Markov decision process, despite non-stationary, is stable, in that it converges if the low-level policy converges.

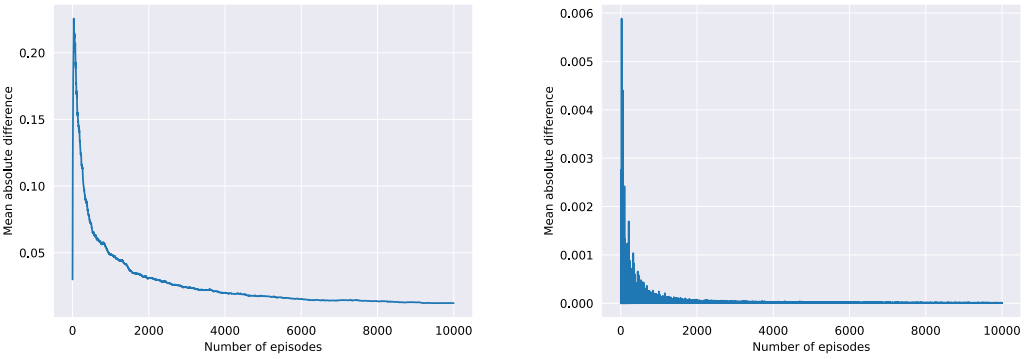
Proposition 1. *Consider the Markov decision processes $\mathcal{M}$, $\mathcal{M}^h$ and $\mathcal{M}^l$ and a converging sequence of low-level policies $\pi_t^l \to \pi^l$, $t \in \mathbb{N}$. For each high-level policy π^h, it holds that:*

1. $P^h(\pi_t^l) \to P^h(\pi^l)$;
2. $r^h(\pi_t^l) \to r^h(\pi^l)$;
3. $v_{\pi^h}^h(\pi_t^l) \to v_{\pi^h}^h(\pi^l)$;
4. $q_{\pi^h}^h(\pi_t^l) \to q_{\pi^h}^h(\pi^l)$.

Proof Sketch. We give the idea for the proof, referring to Appendix B.1.1 for the details. We start by noting that, in metric spaces, if an element $f_t \to f^*$, we have that $G(f_t) \to G(f^*)$ if and only if G is continuous. Therefore, the proposition equivalently states that, for every high-level policy π^h, P^h, r^h, $V_{\pi^h}^h$ and $Q_{\pi^h}^h$ are continuous in the low-level policies π^l. We prove the continuity of each component. □

As we know that the low-level policy converges, from the previous result we know that the high-level Markov decision process converges. Now, we establish convergence of Q-learning on converging Markov decision processes, and thus complete our convergence result.

We prove that Q-learning converges in non-stationary but convergent Markov decision processes by showing that it converges on the high level of a feudal hierarchy. We consider the low level learns using any learning algorithm at any learning rate, as long as it converges, and the high level uses Q-learning. We show that, even though the algorithms operate over a non-stationary Markov decision process, convergence still holds. We introduce some notation. $r_t(x, \omega)$ is a reward sample distributed according to $r^h(\pi_t^l)(x, \omega)$; $y_t(x, \omega)$ a next state sample distributed according to $P^h(\pi_t^l)(x, \omega)$; $q_t^{*,h}$ is the optimal value function of the high-level Markov decision



(a) Distance between low-level value functions with different high-level policies.

(b) Distance between consecutive updates of high-level value functions.

Fig. 2. Experimental validation of the convergence of feudal hierarchies.

process $\mathcal{M}^h(\pi_t^l)$, where π_t^l is the low-level policy at time step t. Consider a sequence of low-level policies π_t^l converging to the optimal low-level policy $\pi^{*,l}$. Q-learning on the high level performs the updates

$$q_{t+1}^h(x,\omega) = q_t^h(x,\omega) + \alpha_t(x,\omega)\Big(r_t(x,\omega) + \gamma^{\frac{1}{\gamma}} \max_{\omega'} q_t^h\big(y_t(x,\omega),\omega'\big) - q_t^h(x,\omega)\Big).$$

Theorem 1. *Suppose that Q-learning with tabular features operates in a CNE with respect to the dynamics, for example, the high-level of a feudal hierarchical setting. Let the high-level learning rates $\alpha_t : \mathcal{X} \times \Omega \to [0,1]$ be such that, for all (x,ω), $\sum_t \alpha_t(x,\omega) = \infty$ and $\sum_t \alpha_t^2(x,\omega) < \infty$. Then, Q-learning converges to $q^{*,h}$.*

Proof Sketch. We give the idea for the proofs, referring to Appendix B.1.2 for the details. The proof follows from verifying the conditions on an established convergence result on stochastic processes [31], which we reproduce. Consider the random iteration

$$\Delta_{t+1}(z) = \big(1 - \alpha_t(z)\big)\Delta_t(z) + \alpha_t(z)F_t(z),$$

where $z \in \mathcal{Z}$ and $\mathcal{Z}$ is finite. Consider also the sequence of sigma-fields $\mathcal{F}_t$ such that $\big(\alpha_t(z), \Delta_t(z), F_t(z)\big) \in \mathcal{F}_t$. The sequence $\{\Delta_t\}_{t\in\mathbb{N}_0}$ converges to 0 under the following conditions, for each z in $\mathcal{Z}$:

1. $\|\mathbb{E}[F_t(z) \mid \mathcal{F}_t]\| \leq \gamma\|\Delta_t\| + c_t$, $c_t \to 0$;
2. $var(F_t(z) \mid \mathcal{F}_t) \leq K(1 + \|\Delta_t\|)^2$.

To apply the convergence result, we consider the stochastic processes

$$\Delta_t(x,\omega) = q_t^h(x,\omega) - q^{*,h}(x,\omega)$$

$$F_t(x,\omega) = r_t(x,\omega) + \gamma^h \max_{\omega'} q_t^h\left(y_t(x,\omega),\omega'\right) - q^{*,h}(x,\omega)$$

and show that they satisfy the conditions referred. □

4.3. Experiments

We showcase that our theoretical findings are empirically verified in the Ant Maze example described before. We start by showing that the low-level policy converges to the same limit solution, regardless of the high-level policy it is trained with. For that, we learn the low-level policy using two different high-level ϵ-greedy policies, one where $\epsilon = 1$ (a random policy) and another where $\epsilon = 0.5$. We use a low-level learning rate $\alpha_t = 0.01$ and a low-level exploration rate $\epsilon = 0.1$. In both, the high-level policy visits all state-action pairs infinitely often. The discount factor is $\gamma = 0.9$. As we can see in Fig. 2a, as training progresses, the distance between the two low-level value functions converges to zero, implying that their respective greedy policies converge to the same limiting policy, which we know is the optimal at the low level.

Then, we also show that the high-level policy converges. We use a high-level learning rate $\beta_t = 0.01$. As we can see in Fig. 2b, even though the low-level policy is changing, the high-level policy is converging, as the distance between consecutive updates of the Q-function is converging to zero.

4.4. Discussion

Our results contribute to a growing line of work on the asymptotic behavior of reinforcement learning in structured non-stationary or non-Markovian environments [12,13,26,21]. As discussed in Section 3, these works typically assume that the environment is either stationary or that the optimal Q-values are independent of the full interaction history.

In contrast, we establish a convergence result for Q-learning at the high level of a feudal hierarchy under the assumption that the low-level process converges. This setting corresponds to a convergently non-stationary environment with respect to the dynamics, where the effective transition and reward functions at the high level evolve over time and converge. Unlike the framework of Majeed and Hutter [26], we do not assume that the optimal Q-values are history-independent. In our setting, the optimal high-level Q-values are generally time-dependent throughout training, but we show that they converge to a time-independent limit $q^{*,h}$ when the lower-level behavior stabilizes.

Additionally, our result does not require ergodicity of the sequence of MDPs, a common assumption in classical convergence analyses. This generality allows us to cover realistic hierarchical reinforcement learning scenarios where exploration may be constrained or structured.

Overall, our contribution extends convergence guarantees to a class of non-stationary problems that are not captured by existing theory, while leveraging structural properties that arise naturally in hierarchical learning.

5. Non-stationary features

Let us consider neural network architectures as parameterized non-linear functions. A state-action pair (x, a) is the input, and is processed by non-linear features parameterized by $\boldsymbol{u}$, $\phi_{\boldsymbol{u}}$. Then, $\phi_{\boldsymbol{u}}(x, a)$ is passed on to a linear layer parameterized by $\boldsymbol{v}$. The output is $q_{\boldsymbol{v},\boldsymbol{u}}(x, a)$. We consider $q_{\boldsymbol{v},\boldsymbol{u}} : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ such that $q_{\boldsymbol{v},\boldsymbol{u}}(x, a) = \phi_{\boldsymbol{u}}^T(x, a)\overline{\boldsymbol{v}}$, where $\bar{\cdot} : \mathbb{R}^K \to \mathcal{B}_\rho$ projects $\boldsymbol{v}$ to a ball of radius ρ in $\mathbb{R}^K$. We refer to the parameters of the final linear layer, $\boldsymbol{v} \in \mathbb{R}^K$, as the final parameters, to the parameters of the non-linear hidden layers, $\boldsymbol{u} \in \mathbb{R}^D$, as the hidden parameters and to $\phi_{\boldsymbol{u}} : \mathcal{X} \times \mathcal{A} \to \mathbb{R}^K$ as the features. We argue that learning the final parameters $\boldsymbol{v}$ at a faster time-scale than the hidden parameters $\boldsymbol{u}$ allows us to decouple the convergence analysis of the two and consequently establish that a regularized version of Q-learning with convergence guarantees for the stationary features case, will also converge if the features are non-stationary but convergent. We define the Q-learning scheme of the final linear layer merging ideas from [23] and [38], as

$$\boldsymbol{v}_{t+1} = \boldsymbol{v}_t + \alpha_t\big(r_t + \gamma \max_{a' \in \mathcal{A}} q_{\boldsymbol{v}_t,\boldsymbol{u}_t}(x_t', a') - \xi q_{\boldsymbol{v}_t,\boldsymbol{u}_t}(x_t, a_t)\big)\phi_{\boldsymbol{u}_t}(x_t, a_t) - \alpha_t \eta \boldsymbol{v}_t,$$

where $\xi \geq 1$, $\eta \geq 0$ are regularization hyper-parameters and the positive learning rates in $\{\alpha_t\}_{t\geq 0}$ are such that $\sum_{t=0}^{\infty} \alpha_t = \infty$ and $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$. We assume the features are updated much slower than the final layer through a stochastic approximation scheme with well-behaved noise and that they converge.

Assumption 2. We assume the following holds:

1. The hidden parameters $\boldsymbol{u}$ are updated according to
$$\boldsymbol{u}_{t+1} = \boldsymbol{u}_t + \beta_t\big(g(\boldsymbol{v}_t, \boldsymbol{u}_t) + \boldsymbol{n}_{t+1}\big);$$
2. The vector field $g : \mathbb{R}^{K+D} \to \mathbb{R}^D$ is Lipschitz-continuous;
3. The sequence of random vectors $\{\mathbf{n}_t\}_{t\in\mathbb{N}}$ has zero mean and finite variance;
4. the learning rates in $\{\beta_t\}_{t\in\mathbb{N}_0}$ are such that $\sum_{t=0}^{\infty} \beta_t = \infty$, $\sum_{t=0}^{\infty} \beta_t^2 < \infty$ and $\frac{\beta_t}{\alpha_t} \to 0$;

We start by establishing our main result, guaranteeing that regularized Q-learning with linear function approximation will converge with non-stationary features that converge. Later on we will discuss that the convergence of the features can be established through various supervised representation signals.

Theorem 2. *Suppose that Q-learning with linear function approximation operates in a CNE with respect to the representations. Further suppose that the evolution of the features satisfies Assumption 2 and, moreover,*

1. *For all t, (x_t, a_t, x_t', r_t) is sampled from $\mathcal{D} = \{(x_i, a_i, x_i', r_i), i \in \mathbb{N}_0\}$ with distribution μ.*
2. *For all $\boldsymbol{u} \in \mathbb{R}^D$, the features $\phi_{\boldsymbol{u}}$ are such that, for all $(x, a) \in \mathcal{X} \times \mathcal{A}$, $\|\phi_{\boldsymbol{u}}(x, a)\|_2 \leq 1$;*
3. *For all $\boldsymbol{u} \in \mathbb{R}^D$, the features $\phi_{\boldsymbol{u}}$ and the distribution μ are such that the $K \times K$ matrix $\boldsymbol{\Sigma}_{\boldsymbol{u}} := \mathbb{E}_\mu\big[\phi_{\boldsymbol{u}}(\mathrm{x}, \mathrm{a})\phi_{\boldsymbol{u}}^T(\mathrm{x}, \mathrm{a})\big]$ is positive-definite and its minimum eigenvalue is σ.*

Finally, suppose that the regularization parameter $\xi > 1$ is large enough, specifically that $\xi > \frac{\gamma}{\sigma}$; that $\eta > 0$ is small enough, specifically that $\eta < \xi\sigma - \gamma$; and that the radius of the ball $\mathcal{B}_\rho$, $\rho > 0$, is also large enough, specifically that $\rho > \frac{r_{\max}}{\xi\sigma - \gamma - \eta}$.

Then, it holds that $\boldsymbol{v}_t \to \boldsymbol{v}^*(\boldsymbol{u}^*)$, with $\boldsymbol{v}^* : \mathbb{R}^D \to \mathbb{R}^K$ such that

$$\boldsymbol{v}^*(\boldsymbol{u}) = \big(\xi\boldsymbol{\Sigma}_{\boldsymbol{u}} + \eta\mathbf{I}\big)^{-1}\,\mathbb{E}\Big[\big(r(\mathrm{x}, \mathrm{a}) + \gamma \max_{a' \in \mathcal{A}} q_{\boldsymbol{v}^*(\boldsymbol{u}),\boldsymbol{u}}(\mathrm{x}', a')\big)\phi_{\boldsymbol{u}}(\mathrm{x}, \mathrm{a})\Big].$$

Fig. 3. Transition diagram of the process in the $v \to 2v$ counter example.

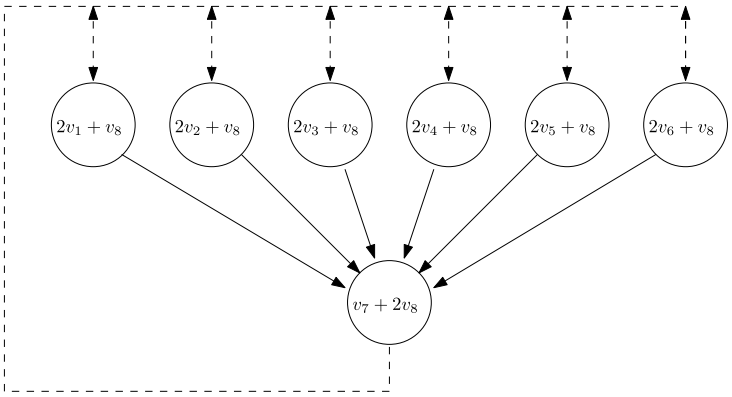


Fig. 4. Transition diagram of the process in the Star example.

Proof Sketch. We give here a roadmap for the proof, which follows from verifying the conditions of Theorem 3 in Appendix A. Throughout, we refer to auxiliary lemmas that we prove in Appendix B.2.

The Q-learning algorithm presented is a two-time-scale stochastic approximation algorithm where the fast component takes the form

$$\boldsymbol{v}_{t+1} = \boldsymbol{v}_t + \alpha_t \left(f(\boldsymbol{v}_t, \boldsymbol{u}_t) + \mathbf{m}_{t+1} \right),$$

with $f : \mathbb{R}^{K+D} \to \mathbb{R}^K$ the expected update

$$f(\boldsymbol{v}, \boldsymbol{u}) = \mathbb{E}\left[\left(r(\mathrm{x}, \mathrm{a}) + \gamma \max_{a' \in \mathcal{A}} q_{\boldsymbol{v},\boldsymbol{u}}(\mathrm{x}', a') - \xi q_{\boldsymbol{v},\boldsymbol{u}}(\mathrm{x}, \mathrm{a}) \right) \phi_u(\mathrm{x}, \mathrm{a}) \right] - \epsilon v$$

and $M_t \in \mathbb{R}^K$ its noise. Borkar [3, Chapter 6; Theorem 2] provides conditions under which the stochastic process above converges. The conditions include that f is Lipschitz and $\mathbf{m}_{t+1}$ is a martingale-difference sequence, which we show through Lemmas 4 and 5, respectively. In addition, Lemma 6 establishes that, for each $u \in \mathbb{R}^L$, the ordinary differential equation (o.d.e.)

$$\dot{\boldsymbol{v}}_t = f(\boldsymbol{v}_t, \boldsymbol{u})$$

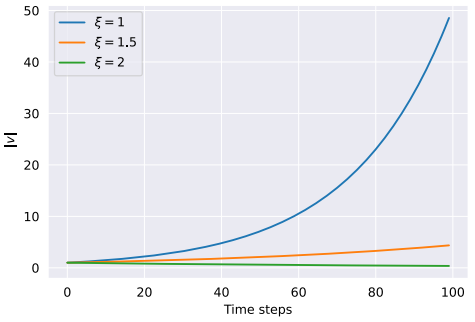
has a unique globally asymptotically stable equilibrium $v^*(u)$, using a Lyapunov argument. Since we show using Lemma 7 that, additionally, the iterates remain bounded, we conclude through Lemma 8 that they converge to the equilibrium w.p.1. □

5.1. Experiments

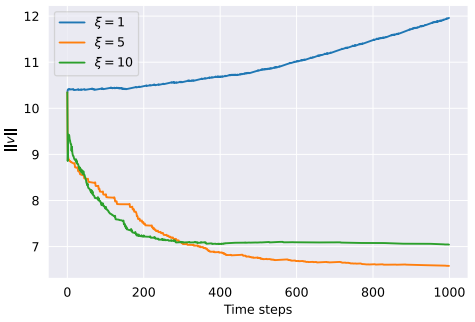
We showcase the convergence of the proposed method on two examples.

Linear $v \to 2v$ example In the $v \to 2v$ example of [33] there are two states and a single action. The first state always transitions to the second; the second state always transitions to itself. All rewards are 0 and, consequently, so are the values. We consider the features $\phi_u(x) = \psi(x) + u$, where $\psi(x)$ is 1 for the first state and 2 for the second state and $u \in \mathbb{R}$. Fig. 3 depicts the process. We consider $u_t = \frac{1}{t}(-1)^t \to 0$. The desirable behavior of Q-learning would be $v \to 0$. We use a learning rate $\alpha_t = 0.01$ and a discount factor $\gamma = 0.9$. Fig. 5a shows the results. We can see that, when $\xi = 1$, v diverges. As ξ increases, learning is more stable. When $\xi = 2$, v converges to the desired solution $v = 0$.

Star example The star example of [2], slightly modified by [32], has seven states and two actions. One of the actions transitions to any of the first six states uniformly, the other action transitions to the seventh state. All rewards are 0 and so are the values. The behavioral policy chooses the first action with probability $\frac{6}{7}$ and the second action with probability $\frac{1}{7}$. Therefore, the next-state distribution is uniform. The state features are $\phi_{\boldsymbol{u}}(x) = \psi(x) + \boldsymbol{u} \in \mathbb{R}^8$. For the first six states, for the first six states, $\psi(x)$ is 2 in the x-th component, 1 in the eighth component and 0 otherwise. For the seventh state, $\psi(x)$ is 1 in the seventh component, 2 in the eighth component and 0 otherwise. Fig. 4 depicts the process. We consider again converging hidden parameters $\boldsymbol{u}_t = (\frac{1}{t}, \frac{1}{t}, \frac{1}{t}, \frac{1}{t}, \frac{1}{t}, \frac{1}{t}, \frac{1}{t}, \frac{1}{t})(-1)^t \to 0$. We



(a) Norm of the approximation parameter v.



(b) Norm of the approximation parameters $\boldsymbol{v}$.

Fig. 5. Behavior of Q-learning with non-stationary features.

use a learning rate $\alpha_t = 0.01$ and a discount factor $\gamma = 0.9$. Fig. 5b shows the results obtained. When $\xi = 1$, the parameters $\boldsymbol{v}$ grow. However, we see that as ξ increases, $\boldsymbol{v}$ converges to the solution, as desired.

5.2. Discussion

In this section, we showed how to extend convergence properties of Q-learning from the linear function approximation case to the case when the features are non-stationary but convergent. For our purposes, we considered a regularized Q-learning method. In particular, we showed that if the features are the inner layers of a non-linear neural network, as long as those features converge, the final layer also converges.

Convergence of the features, i.e., satisfying Assumption 2, can be obtained through performing stochastic gradient descent on proper loss functions. Specifically, the features can be learned in an unsupervised way, for instance by learning an autoencoder, or in a semi-supervised way, for instance by learning to predict the features of the next state, which give rise to powerful representations [9, 17], or through reinforcement, in various forms.

We provide concrete examples of such updates in Appendix C, where, after recalling a general convergence result for stochastic gradient descent, we describe three common training strategies that give rise to convergently non-stationary features: unsupervised learning (e.g., autoencoding) in Appendix C.1; supervised auxiliary tasks (e.g., feature prediction) in Appendix C.2; and reinforcement-driven updates (e.g., representation learning driven by value gradients) in Appendix C.3. For each, we discuss how their induced feature dynamics can fit within our theoretical framework. These examples illustrate how convergently non-stationary features arise in practice, and help clarify how our convergence results apply to realistic deep reinforcement learning pipelines.

6. Conclusion

In this work, we investigated the convergence of Q-learning-based methods in convergently non-stationary environments (CNEs), addressing a significant gap in the existing literature. Our analysis revealed that Q-learning based methods can successfully converge with convergently non-stationary dynamics, particularly within the framework of feudal hierarchies. This finding underscores the robustness of Q-learning when applied to hierarchical reinforcement learning scenarios, where the interplay between different levels of abstraction leads to a dynamic environment. Furthermore, we demonstrated that regularized Q-learning with linear function approximation can achieve convergence in the presence of convergently non-stationary features, as seen in deep reinforcement learning models utilizing supervised representation signals. This result highlights the potential for leveraging Q-learning in deep learning contexts, where the features learned may vary over time but still converge.

Our work contributes to the theoretical foundation of reinforcement learning by providing insights into how Q-learning can be adapted to convergently non-stationary environments, thus broadening its applicability in real-world scenarios. Future research could build on these findings by exploring additional types of non-stationarities, such as those arising from dynamic state or action spaces, and investigating the practical implications of these theoretical results in various application domains.

In summary, this study not only confirms the convergence of Q-learning-based methods under specific non-stationary conditions but also opens new avenues for further exploration and development of reinforcement learning algorithms capable of navigating complex, changing environments.

CRediT authorship contribution statement

Diogo S. Carvalho: Writing – review & editing, Writing – original draft, Methodology, Investigation, Formal analysis, Conceptualization. **Pedro A. Santos:** Supervision. **Francisco S. Melo:** Supervision.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Diogo S. Carvalho reports financial support was provided by Fundação para a Ciência e Tecnologia. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work was partially supported by the Portuguese Fundação para a Ciência e a Tecnologia (FCT) under INESC-ID multi-annual funding (UIDB/50021/2020), and the projects RELEvaNT (PTDC/CCI-COM/5060/2021) and WSMART ROUTEs+ (2022.04180.PTDC). Diogo S. Carvalho acknowledges his FCT PhD grant (2020.05360.BD).

Appendix A. Stochastic approximation

In this appendix we go through stochastic approximation algorithms and tools for their convergence analysis. Robbins and Monro [30] introduced stochastic approximation to solve the equation

$$h(\boldsymbol{w}) = \mathbf{0}, \tag{A.1}$$

where $h : \mathbb{R}^K \to \mathbb{R}^K$. In stochastic approximation, we assume we do not know the function h and, consequently, cannot evaluate it exactly, regardless of the input. We assume we can, however, obtain on input $\boldsymbol{w} \in \mathbb{R}^K$ a noisy sample of $h(\boldsymbol{w})$. We write the noisy sample as $h(\boldsymbol{w}) + \boldsymbol{z}$, where $\boldsymbol{z}$ is a (sampled) noise random vector taking values in $\mathbb{R}^K$. To arrive at a solution $\boldsymbol{w}^*$ for (A.1), starting from an initial guess $\boldsymbol{w}_0$, the Robbins-Monro method iteratively updates its solution according to the rule

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha_t \left(h(\boldsymbol{w}_t) + \boldsymbol{z}_{t+1} \right), \tag{A.2}$$

where α_t is a positive, real-valued, and decreasing step-size that we call the learning rate. The theoretical analysis of the stochastic approximation algorithm (A.2) relies on observing that the update in (A.2) resembles the Euler method to solve the ordinary differential equation

$$\dot{\boldsymbol{w}}_t = h(\boldsymbol{w}_t), \tag{A.3}$$

which consists of the update rule

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \alpha h(\boldsymbol{w}_t). \tag{A.4}$$

More specifically, updates (A.2) and (A.4) differ only in the varying learning rates α_t and noise samples $\boldsymbol{z}_{t+1}$ that appear in the former. Following this remark, Borkar [3, Chapter 2] shows that if h is Lipschitz-continuous and the random variables $\{\boldsymbol{z}_{t+1}\}_{t \in \mathbb{N}_0}$ form a martingale-difference sequence, we can analyze the behavior of the stochastic approximation update by analyzing the ordinary differential equation (A.3) instead. In particular, if (A.3) has a unique globally asymptotically stable equilibrium $\boldsymbol{w}^*$ and $\boldsymbol{w}_t$ is updated according to (A.2), $\boldsymbol{w}_t$ converges to $\boldsymbol{w}^*$ with probability 1 (w.p.1) [3, Chapter 2, Theorem 2].[5]

Alternatively to the single stochastic approximation scheme presented, we can also conceive the case where different components of an iteration are updated according to different learning rate schedules. Particularly, we are mostly interested in the case where, in addition to the main component being updated according to a sequence of learning rates $\{\alpha_t\}_{t \in \mathbb{N}_0}$, a second component is updated according to a sequence of learning rates $\{\beta_t\}_{t \in \mathbb{N}_0}$ such that $\beta_t = o(\alpha_t)$.[6] Intuitively, the second component is updated much slower. Considering this possibility gives rise to two-time-scale stochastic approximation algorithms.

A.1. Two-time-scale stochastic approximation

Consider an iterative algorithm that calls for another iterative sub-routine on each iteration. Traditionally, we would run the sub-routine until near-convergence and then use its output on the first routine. In such case, an outer and an inner loop would be in place. However, we can sometimes achieve the same effect if the two components are updated concurrently but on different time-scales. The sub-routine calls for a larger learning rate and iterates faster. Two-time-scale stochastic approximation algorithms are composed of the iterations

$$\boldsymbol{v}_{t+1} = \boldsymbol{v}_t + \alpha_t (f(\boldsymbol{v}_t, \boldsymbol{u}_t) + \boldsymbol{m}_{t+1})$$
$$\boldsymbol{u}_{t+1} = \boldsymbol{u}_t + \beta_t (g(\boldsymbol{v}_t, \boldsymbol{u}_t) + \boldsymbol{n}_{t+1}),$$

[5] We say that a random sequence of $\boldsymbol{w}_t \in \mathbb{R}^K$ converges to $\boldsymbol{w}^*$ w.p.1 if $\mathbb{P}\left[\lim_{t \to \infty} \boldsymbol{w}_t = \boldsymbol{w}^*\right] = 1$. Convergence w.p.1 is the notion of convergence used throughout the document, unless stated otherwise.

[6] We say that $\beta_t = o(\alpha_t)$ if, for all $\epsilon > 0$, there exists t_0 such that, for all $t \geq t_0$, we have that $\beta_t < \epsilon \alpha_t$.

where $\beta_t = o(\alpha_t)$, $f : \mathbb{R}^{K_1+K_2} \to \mathbb{R}^{K_1}$, $g : \mathbb{R}^{K_1+K_2} \to \mathbb{R}^{K_2}$ and $\boldsymbol{m}_t$ and $\boldsymbol{n}_t$ are sampled according to $\mathbf{m}_t \in \mathbb{R}^{K_1}$ and $\mathbf{n}_t \in \mathbb{R}^{K_2}$ that are noise random variables. The difference in the learning rates between the fast and slow iterations allows us to analyze two-time-scale stochastic approximation algorithms as if the slow component is fixed and the fast component is in equilibrium throughout the execution of the algorithm. We can then use, as in the single time-scale stochastic approximation case, the stability of the corresponding ordinary differential equations to establish convergence of each update. We reproduce a convergence result for two-time-scale stochastic approximation algorithms, which we use to establish most of our convergence results throughout this research. It is due to Borkar [3].

Theorem 3. *Consider the following assumptions:*

1. *f and g are both Lipschitz-continuous.*
2. *$\{\mathbf{m}_t\}_{t\in\mathbb{N}}$ and $\{\mathbf{n}_t\}_{t\in\mathbb{N}}$ are martingale-difference sequences with respect to the sigma-fields $\mathcal{F}_t = \sigma(\boldsymbol{v}_\tau, \boldsymbol{u}_\tau, \mathbf{m}_\tau, \mathbf{n}_\tau, \tau \le t)$. Additionally, there exist $c_{\boldsymbol{m}}$ and $c_{\boldsymbol{n}}$ such that*

$$E[\|\mathbf{m}_t\|^2 \mid \mathcal{F}_t] \le c_{\boldsymbol{m}}(1 + \|\boldsymbol{v}_t\|^2 + \|\boldsymbol{u}_t\|^2),$$
$$E[\|\mathbf{n}_t\|^2 \mid \mathcal{F}_t] \le c_{\mathbf{n}}(1 + \|\boldsymbol{v}_t\|^2 + \|\boldsymbol{u}_t\|^2).$$

3. *The learning rates satisfy $\beta_t = o(\alpha_t)$, $\sum_t \beta_t = \infty$ and $\sum_t \alpha_t^2 < \infty$.*
4. *For each $\boldsymbol{u} \in \mathbb{R}^{K_2}$, o.d.e. $\dot{\boldsymbol{v}}_t = f(\boldsymbol{v}_t, \boldsymbol{u})$ has a unique globally asymptotically stable equilibrium $\boldsymbol{v}^*(\boldsymbol{u})$, where $\boldsymbol{v}^* : \mathbb{R}^{K_2} \to \mathbb{R}^{K_1}$ is a Lipschitz-continuous function.*
5. *O.d.e. $\dot{\boldsymbol{u}}_t = g\left(\boldsymbol{v}^*(\boldsymbol{u}_t), \boldsymbol{u}_t\right)$ has a unique globally asymptotically stable equilibrium $\boldsymbol{u}^*$.*
6. *It holds that $\sup_t \left(\|\boldsymbol{v}_t\| + \|\boldsymbol{u}_t\|\right) < \infty$ w.p.1.*

Then, $(\boldsymbol{v}_t, \boldsymbol{u}_t) \to (\boldsymbol{v}^*(\boldsymbol{u}^*), \boldsymbol{u}^*)$.

The assumptions that, in general, are harder to verify are Assumptions 4 and 5. These assumptions require existence and globally asymptotic stability of equilibrium points of o.d.e.s. To verify these assumptions on applications, we can use Lyapunov's second method [25] for o.d.e. stability, also known as the direct method, to establish the results. Finally, Assumption 6 demands that the iterates remain bounded. In most applications, the property is also not trivially established. However, a result from [16] guarantees boundedness of two-time-scale stochastic approximation iterates under conditions that are easier to establish, also resorting to global asymptotic stability of appropriate o.d.e.s.

We finish by noting that stochastic approximation is a method fit for various applications, due to its general formulation. In the specific case of reinforcement learning, there are sources of noise in the interaction of the agent with its environment. Those sources include the unavailability of the model, the stochasticity of the transitions, rewards and even of the action selection. At the same time, stochastic approximation allows us to analyze, theoretically, properties of the reinforcement learning algorithms such as asymptotic convergence. We formalize now the most common framework for reinforcement learning purposes.

Appendix B. Convergently non-stationary environments: detailed proofs

B.1. Non-stationary dynamics

B.1.1. Proof of Proposition 1

Before proving assertions 1 through 4, we recall that, in metric spaces, if an element $f_t \to f^*$, $G(f_t) \to G(f^*)$ if and only if G is a continuous mapping. Therefore, the theorem equivalently states that, for every high-level policy π^h, the functions P^h, r^h, $V^h_{\pi^h}$ and $Q^h_{\pi^h}$ are continuous on the low-level policies π^l. With the previous remark in mind, we start proving each assertion of the theorem.

1. We show that the transition probabilities $P^h(\pi^l)$ vary continuously with π^l for each ω. First note that the transition dynamics Markov reward process induced by the low-level policy on $\mathcal{M}^l$, $P^\omega_{\pi^l}$, is a continuous function of π^l. Now recall that

$$\left[P^h(\pi^l)\right]_{\omega,\cdot,\cdot} = \mathbb{E}[P^{\omega\,\tau}_{\pi^l}].$$

Finally note that

$$\begin{aligned}\mathbb{E}[P^{\omega\,\tau}_{\pi^l}] &= \sum_{\tau=0}^{\infty} \xi(\tau) P^{\omega\,\tau}_{\pi^l} \\ &= \sum_{\tau=0}^{\infty} (1-\gamma)\gamma^\tau P^{\omega\,\tau}_{\pi^l} \\ &= (1-\gamma)(I - \gamma P^\omega_{\pi^l})^{-1}.\end{aligned}$$

The inverse is well defined and continuous.

2. Recall that, for all $\omega \in \Omega$,

$$r^h(\pi^l)(x,\omega) = \mathbb{E}_{\pi^l|\omega}\Big[\sum_{t=0}^{\tau} r(x_t,a_t) \mid x_0 = x\Big].$$

After defining the real vector

$$[r^{\omega}_{\pi^l}]_x := \sum_a \pi^l\big((x,\omega),a\big) r(x,a),$$

we write the expected sum of discounted rewards

$$\begin{aligned}
&\mathbb{E}_{\pi^l|\omega}\Big[\sum_{t=0}^{\tau} r(x_t,a_t) \mid x_0 = x\Big] \\
&\quad = \Big[\sum_{\tau=0}^{\infty} \xi(\tau) \sum_{t=0}^{\tau} \gamma^t P^{\omega\, t}_{\pi^l} r^{\omega}_{\pi^l}\Big]_x \\
&\quad = \Big[\sum_{\tau=0}^{\infty} \xi(\tau)(I-\gamma P^{\omega}_{\pi^l})^{-1} \cdot \\
&\qquad \cdot \big(I-(\gamma P^{\omega}_{\pi^l})^{\tau+1}\big) r^{\omega}_{\pi^l}\Big]_x \\
&\quad = \Big[(I-\gamma P^{\omega}_{\pi^l})^{-1} \cdot \\
&\qquad \cdot \big(I-(1-\gamma)(I-\gamma^2 P^{\omega}_{\pi^l})^{-1} \cdot \\
&\qquad\quad \cdot \gamma P^{\omega}_{\pi^l}\big) r^{\omega}_{\pi^l}\Big]_x .
\end{aligned}$$

The conclusion follows again from continuity.

3. The value function $V^h_{\pi^h}(\pi^l) : \mathcal{X} \to \mathbb{R}$ is defined as

$$V^h_{\pi^h}(\pi^l)(x) = \mathbb{E}_{\pi^h}\Big[\sum_{t=0}^{\infty} (\gamma^h)^t r^h(\pi^l)(x_t,\omega_t) \mid x_0 = x\Big],$$

where $\omega_t \sim \pi^h(x_t,\cdot)$ and $x_{t+1} \sim P^h(\pi^l)(\cdot \mid x_t,\omega_t)$.
We observe that

$$V^h_{\pi^h}(\pi^l) = \big(I-\gamma^h P^h(\pi^l)\big)^{-1} r^h(\pi^l)$$

and that the expression varies continuously with π^l, by 1 and 2.

4. Building on assertion 3, we observe that

$$Q^h_{\pi^h}(\pi^l)(x,\omega) = \mathbb{E}\big[r^h(\pi^l)(x,\omega) + \gamma V^h_{\pi^h}(\pi^l)(x')\big]$$

continuously maps π^l at every state-action pair.

B.1.2. Proof of Theorem 1

We start with recalling a convergence result for stochastic processes [31]. Consider the random iteration

$$\Delta_t(z) = \big(1-\alpha_t(z)\big)\Delta_t(z) + \alpha_t(z) F_t(z),$$

where $z \in \mathcal{Z}$ and $\mathcal{Z}$ is finite. Consider also the sequence of σ-fields $\mathcal{F}_t$ such that $\big(\alpha_t(z), \Delta_t(z), F_t(z)\big) \in \mathcal{F}_t$. The sequence $\{\Delta_t\}$ converges to 0 w.p. 1 under the following conditions, for each z in $\mathcal{Z}$:

1. $\|\mathbb{E}[F_t(z) \mid \mathcal{F}_t]\| \le \gamma \|\Delta_t\| + c_t$, $c_t \to 0$ w.p. 1;
2. $var(F_t(z) \mid \mathcal{F}_t) \le K(1+\|\Delta_t\|)^2$.

Note that the assumptions on the learning rates require, in particular, infinitely often state action visitation. We refer to the original manuscript [31] for clarifying details over the learning rates.

Consider the stochastic processes

$$\begin{cases}
\Delta_t(x,\omega) = Q^h_t(x,\omega) - Q^{*,h}(x,\omega) \\
F_t(x,\omega) = r_{t+1}(x,\omega) + \\
\qquad +\gamma^h \max_{a'} Q^h_t\big(y_t(x,\omega),\omega'\big) - Q^{*,h}(x,\omega).
\end{cases}$$

1. After focusing on the definition of $F_t(x,\omega)$, we observe that

$$\begin{aligned}\mathbb{E}[F_t(x,\omega) \mid \mathcal{F}_t] &= \mathbb{E}\Big[r_{t+1}(x,\omega) + \gamma^h \max_{\omega'} Q_t^h\left(y_t(x,\omega),\omega'\right) - \\ &\quad - {Q_t}^{*,h}(x,\omega) \mid \mathcal{F}_t\Big] + \\ &\quad + \mathbb{E}\big[Q_t^{*,h}(x,\omega) - Q^{*,h}(x,\omega) \mid \mathcal{F}_t\big].\end{aligned}$$

From Theorem 1 we know the second term vanishes. For the first term we use the theorem again and also the contractiveness of the Bellman operator to conclude the assertion:

$$\begin{aligned}\left\|\mathbb{E}\Big[r_{t+1}(x,\omega)+\gamma^h \max_{\omega'} Q_t^h\left(y_t(x,\omega),\omega'\right) - Q_t^{*,h}(x,\omega) \mid \mathcal{F}_t\Big]\right\| &\leq \gamma^h \left\|Q_t^h - Q_t^{*,h}\right\| \\ &\leq \gamma^h \left\|\Delta_t\right\| + \gamma^h \left\|Q^{*,h} - Q_t^{*,h}\right\|.\end{aligned}$$

2. We must show that, for each t, the variance of $F_t(x,\omega)$ is bounded by a linear combination of 1, $\|Q_t^h(x,\omega)\|$ and $\|{Q_t^h}^2(x,\omega)\|$. We call for attention on the definition of $F_t(x,\omega)$ once more. Now we recall that the reward function has bounded variance (see Section 2). Additionally, the contribution of the term $Q^{*,h}$ to the variance is additive, since it is independent from Q_t^h given the history $\mathcal{F}_t$. Finally, the dependence of $F_t(x,\omega)$ on $Q_t^h(x,\omega)$ is, at most, linear. We conclude the result. The present proof, on a bound of the variance of the updates, closely resembles one from [31].

B.2. Non-stationary features: proof of Theorem 2

We present an outline of the proof, referring to auxiliary lemmas that we prove in the sequel. The Q-learning algorithm presented is a two-time-scale stochastic approximation algorithm where the fast component takes the form

$$\boldsymbol{v}_{t+1} = \boldsymbol{v}_t + \alpha_t\left(f(\boldsymbol{v}_t,\boldsymbol{u}_t) + M_{t+1}\right),$$

with $f : \mathbb{R}^{K+D} \to \mathbb{R}^K$ the expected update

$$f(\boldsymbol{v},\boldsymbol{u}) = \mathbb{E}\left[\left(r(x,a) + \gamma \max_{a'\in\mathcal{A}} q_{\boldsymbol{v},\boldsymbol{u}}(x',a') - \xi q_{\boldsymbol{v},\boldsymbol{u}}(x,a)\right)\phi_{\boldsymbol{u}}(x,a)\right] - \epsilon\boldsymbol{v}$$

and $M_t \in \mathbb{R}^K$ its noise. Borkar [3, Chapter 6; Theorem 2] provides conditions under which the stochastic process above converges. The conditions include that f is Lipschitz and M_{t+1} is a martingale-difference sequence, which we show through Lemmas 4 and 5, respectively. In addition, Lemma 6 establishes that, for each $\boldsymbol{u} \in \mathbb{R}^D$, the ordinary differential equation (o.d.e.)

$$\dot{\boldsymbol{v}}_t = f(\boldsymbol{v}_t,\boldsymbol{u})$$

has a unique globally asymptotically stable equilibrium $v^*(\boldsymbol{u})$, using a Lyapunov argument. Since we show using Lemma 7 that, additionally, the iterates remain bounded, we conclude through Lemma 8 that they converge to the equilibrium w.p.1.

Lemma 4. *The vector field $f : \mathbb{R}^{K+L} \to \mathbb{R}^K$ such that*

$$f(\boldsymbol{v},\boldsymbol{u}) = \mathbb{E}\left[\left(r(x,a) + \gamma \max_{a'\in\mathcal{A}} q_{\boldsymbol{v},\boldsymbol{u}}(x',a') - \xi q_{\boldsymbol{v},\boldsymbol{u}}(x,a)\right)\phi_{\boldsymbol{u}}(x,a)\right] - \epsilon\boldsymbol{v}$$

is Lipschitz-continuous.

Proof. We must show that

$$\|f(\boldsymbol{v},\boldsymbol{u}) - f(\boldsymbol{w},\boldsymbol{z})\| \leq L_f\|\boldsymbol{v}-\boldsymbol{w}\| + L_f\|\boldsymbol{u}-\boldsymbol{z}\|.$$

For a fixed transition tuple $(x,a,r,x') \in \mathcal{X}\times\mathcal{A}\times I_R\times\mathcal{X}$, we define $f_{x,a,r,x'} : \mathbb{R}^{K+D} \to \mathbb{R}^K$ such that

$$f_{x,a,r,x'}(\boldsymbol{v},\boldsymbol{u}) = \left(r + \gamma \max_{a'\in\mathcal{A}} q_{\boldsymbol{v},\boldsymbol{u}}(x',a') - \xi q_{\boldsymbol{v},\boldsymbol{u}}(x,a)\right)\phi_{\boldsymbol{u}}(x,a) - \epsilon\boldsymbol{v}.$$

We show that $f_{x,a,r,x'}$ is Lipschitz-continuous with a constant L_f that does not depend on (x,a,r,x'). Consequently, $f = \mathbb{E}[f_{x,a,r,x'}]$ is also Lipschitz-continuous with the same constant.

We assume (x,a,r,x') is fixed and we use the notation:

- $A(\boldsymbol{v},\boldsymbol{u}) = r + \gamma \max_{a'\in\mathcal{A}} q_{\boldsymbol{v},\boldsymbol{u}}(x',a')$;

- $B(\boldsymbol{v},\boldsymbol{u}) = \xi q_{\boldsymbol{v},\boldsymbol{u}}(x,a)$;
- $C(\boldsymbol{u}) = \phi_{\boldsymbol{u}}(x,a)$;
- $D(\boldsymbol{v}) = \epsilon \boldsymbol{v}$.

Then we have that $f_{x,a,r,x'}(\boldsymbol{v},\boldsymbol{u}) = A(\boldsymbol{v},\boldsymbol{u})C(\boldsymbol{u}) - B(\boldsymbol{v},\boldsymbol{u})C(\boldsymbol{u}) + D(\boldsymbol{v})$. We move on using triangle inequalities.

$$\begin{aligned}
\|f_{x,a,r,x'}(\boldsymbol{v},\boldsymbol{u}) - f_{x,a,r,x'}(\boldsymbol{w},\boldsymbol{z})\| &= \\
&= \|A(\boldsymbol{v},\boldsymbol{u})C(\boldsymbol{u}) - B(\boldsymbol{v},\boldsymbol{u})C(\boldsymbol{u}) - A(\boldsymbol{w},\boldsymbol{z})C(\boldsymbol{z}) + B(\boldsymbol{w},\boldsymbol{z})C(\boldsymbol{z}) + \\
&\quad D(\boldsymbol{v}) - D(\boldsymbol{w})\| \\
&\leq \|A(\boldsymbol{v},\boldsymbol{u})C(\boldsymbol{u}) - A(\boldsymbol{w},\boldsymbol{z})C(\boldsymbol{z})\| + \\
&\quad + \|B(\boldsymbol{v},\boldsymbol{u})C(\boldsymbol{u}) - B(\boldsymbol{w},\boldsymbol{z})C(\boldsymbol{z})\| + \|D(\boldsymbol{v}) - D(\boldsymbol{w})\| \\
&\leq \|A(\boldsymbol{v},\boldsymbol{u})\,(C(\boldsymbol{u}) - C(\boldsymbol{z}))\| + \|(A(\boldsymbol{v},\boldsymbol{u}) - A(\boldsymbol{w},\boldsymbol{z}))\,C(\boldsymbol{z})\| + \\
&\quad + \|B(\boldsymbol{v},\boldsymbol{u})\,(C(\boldsymbol{u}) - C(\boldsymbol{z}))\| + \|(B(\boldsymbol{v},\boldsymbol{u}) - B(\boldsymbol{w},\boldsymbol{z}))\,C(\boldsymbol{z})\| \\
&\quad + \|D(\boldsymbol{v}) - D(\boldsymbol{w})\|.
\end{aligned}$$

We observe that

$$\|A(\boldsymbol{v},\boldsymbol{u})\| \leq r_{max} + \gamma \max_{a' \in \mathcal{A}} \left\|\phi_{\boldsymbol{u}}(x',a')\right\| \left\|\overline{\boldsymbol{v}}\right\| \leq r_{max} + \gamma\rho$$

using Cauchy-Schwartz inequality, Assumption 2 and the projection of $\boldsymbol{v}$. Also,

$$\begin{aligned}
\|C(\boldsymbol{u}) - C(\boldsymbol{z})\| &\leq \left\|\phi_{\boldsymbol{u}}(x,a) - \phi_{\boldsymbol{z}}(x,a)\right\| \\
&\leq L_\phi \|\boldsymbol{u} - \boldsymbol{z}\|
\end{aligned}$$

from Assumption 2. We also have that

$$\begin{aligned}
\|A(\boldsymbol{v},\boldsymbol{u}) - A(\boldsymbol{w},\boldsymbol{z})\| &\leq \gamma \left\|\overline{\boldsymbol{v}} - \overline{\boldsymbol{w}}\right\| + \left\|\overline{\boldsymbol{w}}\right\| \|\boldsymbol{u} - \boldsymbol{z}\| \\
&\leq \gamma \|\boldsymbol{v} - \boldsymbol{w}\| + \rho \|\boldsymbol{u} - \boldsymbol{z}\|
\end{aligned}$$

for the same reasons. From Assumption 2,

$$\|C(\boldsymbol{u})\| = 1.$$

Now,

$$\|B(\boldsymbol{v},\boldsymbol{u})\| \leq \xi \left\|\phi_{\boldsymbol{u}}(x,a)\right\| \left\|\overline{\boldsymbol{v}}\right\| \leq \xi\rho$$

and

$$\begin{aligned}
\|B(\boldsymbol{v},\boldsymbol{u}) - B(\boldsymbol{w},\boldsymbol{z})\| &\leq \left\|\xi \left(\phi_{\boldsymbol{u}}(x,a) - \phi_{\boldsymbol{z}}(x,a)\right) \overline{\boldsymbol{u}}\right\| + \left\|\xi\phi_{\boldsymbol{z}}(x,a)(\overline{\boldsymbol{v}} - \overline{\boldsymbol{z}})\right\| \\
&\leq \xi\rho L_\phi \|\boldsymbol{u} - \boldsymbol{z}\| + \xi \|\boldsymbol{v} - \boldsymbol{w}\|.
\end{aligned}$$

Finally, we can see that

$$\|D(\boldsymbol{v}) - D(\boldsymbol{w})\| = \epsilon \|\boldsymbol{v} - \boldsymbol{w}\|.$$

Putting everything together with the help of Cauchy-Schwartz inequalities, the conclusion follows. □

Lemma 5. *The sequence of random vectors* $\{M_t\}_{t\geq 0}$ *such that*

$$\begin{aligned}
M_{t+1} = &\left(r_t + \gamma \max_{a' \in \mathcal{A}} q_{\boldsymbol{v}_t,\boldsymbol{u}_t}(x_t', a_t') - \xi q_{\boldsymbol{v}_t,\boldsymbol{u}_t}(x_t,a_t)\right) \phi_{\boldsymbol{u}_t}(x_t,a_t) - \\
&- \mathbb{E}\left[\left(r(x,a) + \gamma \max_{a' \in \mathcal{A}} q_{\boldsymbol{v},\boldsymbol{u}}(x',a') - \xi q_{\boldsymbol{v},\boldsymbol{u}}(x,a)\right) \phi_{\boldsymbol{u}}(x,a)\right]
\end{aligned}$$

is a martingale difference sequence verifying

$$\mathbb{E}\left[\left\|M_{t+1}\right\|^2 \mid \mathcal{F}_t\right] \leq c_M \left(1 + \left\|\boldsymbol{v}_t\right\|^2 + \left\|\boldsymbol{u}_t\right\|^2\right).$$

Proof. For $\{M_t\}_{t\geq 0}$ to be a martingale difference sequence, we must have that

1. $\mathbb{E}[M_{t+1} \mid \mathcal{F}_t] = 0, \quad \forall t > 0;$
2. $\mathbb{E}[\|M_t\|] < \infty, \quad \forall t > 0.$

From Assumption 1, we directly conclude property 1:

$$\mathbb{E}[M_{t+1} \mid \mathcal{F}_t] = \mathbb{E}[M_{t+1}] = 0.$$

To establish property 2, we observe that every term appearing on the definition of M_t is bounded by some constant. The same observation is sufficient to conclude the second moment is also bounded by some constant. □

Lemma 6. *For each $u \in \mathbb{R}^D$, the o.d.e.*

$$\dot{\boldsymbol{v}}_t = f(\boldsymbol{v}_t, \boldsymbol{u})$$

has a unique and globally asymptotically stable equilibrium $\boldsymbol{v}^*(u)$, where $\boldsymbol{v}^* : \mathbb{R}^D \to \mathbb{R}^K$ is Lipschitz-continuous.

Proof. We start by establishing the existence and uniqueness of an equilibrium, $\boldsymbol{v}^*(\boldsymbol{u})$, for each $\boldsymbol{u} \in \mathbb{R}^D$, by making use of Banach's fixed point theorem. We then show that $\boldsymbol{v}^*$ is Lipschitz-continuous. Finally, we show that $\boldsymbol{v}^*(\boldsymbol{u})$ is globally asymptotically stable using a Lyapunov argument.

Any solution to the o.d.e. must verify

$$\dot{\boldsymbol{v}}_t = f(\boldsymbol{v}_t, \boldsymbol{u}) = 0.$$

Equivalently, and ignoring for a moment the projection $\overline{\boldsymbol{v}}$ of $\boldsymbol{v}$ in q, any solution is time-invariant and we can, therefore, drop the dependency on t and, writing in the form of a fixed-point equation, we must have that

$$\boldsymbol{v} = \frac{1}{\xi}\Sigma_{\boldsymbol{u}}^{-1}\mathbb{E}\left[\left(r(x,a) + \gamma \max_{a' \in \mathcal{A}} q_{\boldsymbol{v},\boldsymbol{u}}(x',a')\right)\boldsymbol{\phi}_{\boldsymbol{u}}(x,a)\right] - \frac{\epsilon}{\xi}\Sigma_{\boldsymbol{u}}^{-1}\boldsymbol{v}.$$

We used Assumption 3 to invert the matrix $\Sigma_{\boldsymbol{u}}$. Let us refer to the right-hand side as $T : \mathbb{R}^K \to \mathbb{R}^K$. As R^K is a Banach space, contractivity of T will allow us to conclude the existence and uniqueness of solution to the fixed point equation above. We observe that

$$\begin{aligned}
&\|T(\boldsymbol{v}) - T(\boldsymbol{w})\| \leq \\
&\leq \left\|\frac{1}{\xi}\Sigma_{\boldsymbol{u}}^{-1}\gamma\mathbb{E}\left[\left(\max_{a' \in \mathcal{A}} q_{\boldsymbol{v},\boldsymbol{u}}(x',a') - \max_{a' \in \mathcal{A}} q_{\boldsymbol{w},\boldsymbol{u}}(x',a')\right)\boldsymbol{\phi}_{\boldsymbol{u}}(x,a)\right]\right\| + \\
&\quad + \left\|\frac{\epsilon}{\xi}\Sigma_{\boldsymbol{u}}^{-1}(\boldsymbol{v} - \boldsymbol{w})\right\| \\
&\leq \frac{\gamma}{\xi\sigma}\mathbb{E}\left[\max_{a' \in \mathcal{A}} |\boldsymbol{\phi}_{\boldsymbol{u}}(x',a') \cdot \boldsymbol{v} - \boldsymbol{\phi}_{\boldsymbol{u}}(x',a') \cdot \boldsymbol{w}| \left\|\boldsymbol{\phi}_{\boldsymbol{u}}(x,a)\right\|\right] + \\
&\quad + \frac{\epsilon}{\xi\sigma}\|\boldsymbol{v} - \boldsymbol{w}\| \\
&\leq \frac{\gamma + \epsilon}{\xi\sigma}\|\boldsymbol{v} - \boldsymbol{w}\|.
\end{aligned}$$

From Assumption 3, we have $\frac{\gamma+\epsilon}{\xi\sigma} < 1$, and contractivity follows. Consequently, there exists a unique solution $\boldsymbol{v}^* \in \mathbb{R}^K$ for each $\boldsymbol{u} \in \mathbb{R}^D$. Moreover, we can bound its norm as $\|\boldsymbol{v}^*(\boldsymbol{u})\| \leq \frac{1}{\xi\sigma - \epsilon}(r_{max} + \gamma\rho) < \rho$, implying that the solution lies inside the ball B_ρ.

We next show that the mapping $\boldsymbol{v}^* : \mathbb{R}^D \to \mathbb{R}^K$ is Lipschitz-continuous with respect to $\boldsymbol{u} \in \mathbb{R}^K$. The proof follows similar lines as the argument for the contractivity of T. To establish this, we write:

$$\begin{aligned}
&\left\|\boldsymbol{v}^*(\boldsymbol{u}) - \boldsymbol{v}^*(\boldsymbol{w})\right\| \leq \\
&\quad \leq \frac{\gamma}{\xi\sigma}\mathbb{E}\left[\max_{a' \in \mathcal{A}} |\boldsymbol{\phi}_{\boldsymbol{u}}(x',a') \cdot \boldsymbol{v}^*(\boldsymbol{u}) - \boldsymbol{\phi}_{\boldsymbol{w}}(x',a') \cdot \boldsymbol{v}^*(\boldsymbol{w})| \left\|\boldsymbol{\phi}_{\boldsymbol{u}}(x,a)\right\|\right] \\
&\quad \leq \frac{\gamma\rho}{\xi\sigma}L_\phi\|\boldsymbol{u} - \boldsymbol{w}\|.
\end{aligned}$$

Finally, having established uniqueness and existence of a Lipschitz-solution of the o.d.e., $\boldsymbol{v}^*(\boldsymbol{u})$, we prove it is globally asymptotically stable. We consider the Lyapunov function $l_{\boldsymbol{u}} : \mathbb{R}^K \to \mathbb{R}$ such that $l_{\boldsymbol{u}}(\boldsymbol{v}) = \frac{1}{2}\|\boldsymbol{v} - \boldsymbol{v}^*(\boldsymbol{u})\|^2$. We have that $l_{\boldsymbol{u}}(\boldsymbol{v}) = 0$ if and only if $\boldsymbol{v} = \boldsymbol{v}^*(\boldsymbol{u})$. We also have that $l_{\boldsymbol{u}}(\boldsymbol{v}) > 0$ if and only if $\boldsymbol{v} \neq \boldsymbol{v}^*(\boldsymbol{u})$. To establish globally asymptotic stability, it remains only to show that $\dot{l}_{\boldsymbol{u}}(\boldsymbol{v}) < 0$ whenever $\boldsymbol{v} \neq \boldsymbol{v}^*(\boldsymbol{u})$ and $\dot{l}_{\boldsymbol{u}}(\boldsymbol{v}) = 0$ otherwise. We start by writing

$$\dot{l}_{\boldsymbol{u}}(\boldsymbol{v}) = \nabla_{\boldsymbol{v}} l_{\boldsymbol{u}}(\boldsymbol{v}) \cdot \dot{\boldsymbol{v}}$$

$$
\begin{aligned}
&= \left(\boldsymbol{v} - \boldsymbol{v}^*(\boldsymbol{u})\right) \cdot f(\boldsymbol{v}, \boldsymbol{u}) \\
&= \left(\boldsymbol{v} - \boldsymbol{v}^*(\boldsymbol{u})\right) \cdot \\
&\quad \cdot \mathbb{E}\left[\left(r(x,a) + \gamma \max_{a' \in \mathcal{A}} q_{\boldsymbol{v},\boldsymbol{u}}(x',a') - \xi q_{\boldsymbol{v},\boldsymbol{u}}(x,a)\right)\phi_{\boldsymbol{u}}(x,a) - \epsilon \boldsymbol{v}\right] \\
&= \left(\boldsymbol{v} - \boldsymbol{v}^*(\boldsymbol{u})\right) \cdot \mathbb{E}\left[\left(r(x,a) + \gamma \max_{a' \in \mathcal{A}} q_{\boldsymbol{v},\boldsymbol{u}}(x',a')\right)\phi_{\boldsymbol{u}}(x,a) - \epsilon \boldsymbol{v}\right] - \\
&\quad \xi\left(\boldsymbol{v} - \boldsymbol{v}^*(\boldsymbol{u})\right)\Sigma_{\boldsymbol{u}}\boldsymbol{v}.
\end{aligned}
$$

Now, we subtract the quantity

$$
\mathbb{E}\left[\left(r(x, Aa) + \gamma \max_{a' \in \mathcal{A}} q_{\boldsymbol{v}^*(\boldsymbol{u}),\boldsymbol{u}}(x',a')\right)\phi_{\boldsymbol{u}}(x,a) - \epsilon \boldsymbol{v}^*(\boldsymbol{u})\right] - \xi \Sigma_{\boldsymbol{u}} \boldsymbol{v}^*(\boldsymbol{u}),
$$

which we know equals 0, multiplied by

$$
(\boldsymbol{v} - \boldsymbol{v}^*(\boldsymbol{u})).
$$

We can rearrange the resulting expression and obtain

$$
\begin{aligned}
\left(\boldsymbol{v} - \boldsymbol{v}^*(\boldsymbol{u})\right) \cdot \mathbb{E}\Bigg[&\left(r(x,a) + \gamma \max_{a' \in \mathcal{A}} q_{\boldsymbol{v},\boldsymbol{u}}(x',a') - \gamma \max_{a' \in \mathcal{A}} q_{\boldsymbol{v}^*(\boldsymbol{u}),\boldsymbol{u}}(x',a')\right) \cdot \\
&\cdot \phi_{\boldsymbol{u}}(x,a) - \epsilon\left(\boldsymbol{v} - \boldsymbol{v}^*(\boldsymbol{u})\right)\Bigg] - \\
&- \xi\left(\boldsymbol{v} - \boldsymbol{v}^*(u)\right)\Sigma_{\boldsymbol{u}}\left(\boldsymbol{v} - \boldsymbol{v}^*(\boldsymbol{u})\right).
\end{aligned}
$$

Since $\xi > 0$ is sufficiently large, we can conclude that $\dot{l}_{\boldsymbol{u}}(\boldsymbol{v}) < 0$ if $(\boldsymbol{v} - \boldsymbol{v}^*(\boldsymbol{u}))\Sigma_{\boldsymbol{u}}(\boldsymbol{v} - \boldsymbol{v}^*(\boldsymbol{u})) > 0$. Such is the case since we have positive-definiteness of $\Sigma_{\boldsymbol{u}}$ from Assumption 3. This concludes the result. □

Lemma 7. *For every $\boldsymbol{u} \in \mathbb{R}^D$, the sequence of vector fields $\{h_{c,\boldsymbol{u}}\}_{c \geq 1}$ such that $h_{c,\boldsymbol{u}} : \mathbb{R}^K \to \mathbb{R}^K$ and $h_{c,\boldsymbol{u}}(\boldsymbol{v}) = \frac{f(c\boldsymbol{v},\boldsymbol{u})}{c}$, for some continuous $h_{\infty,\boldsymbol{u}}$, verifies*

$$
h_{c,\boldsymbol{u}} \to h_{\infty,\boldsymbol{u}}
$$

uniformly on compacts. Additionally, the o.d.e.

$$
\dot{\boldsymbol{v}}_t = h_{\infty,\boldsymbol{u}}(\boldsymbol{v}_t)
$$

has the origin as its unique and globally asymptotically stable equilibrium.

Proof. We can expand the definition and observe

$$
h_{c,\boldsymbol{u}}(\boldsymbol{v}) = \frac{\mathbb{E}\left[\left(r(x,a) + \gamma \max_{a' \in \mathcal{A}} q_{c\boldsymbol{v},\boldsymbol{u}}(x',a') - \xi q_{c\boldsymbol{v},\boldsymbol{u}}(x,a)\right)\phi_{\boldsymbol{u}}(x,a)\right] - \epsilon c \boldsymbol{v}}{c}.
$$

We recall that q projects $\boldsymbol{v}$ back into B_ρ once $\|\boldsymbol{v}\| > \rho$. Therefore, as $c \to \infty$,

$$
h_{c,\boldsymbol{u}}(\boldsymbol{v}) \to -\epsilon \boldsymbol{v}
$$

uniformly on compacts. With $h_{\infty,\boldsymbol{u}}(\boldsymbol{v}) = -\epsilon \boldsymbol{v}$, we have that $\dot{\boldsymbol{v}}_t = h_{\infty,\boldsymbol{u}}(\boldsymbol{v}_t)$ has the origin as unique and globally asymptotically stable equilibrium. □

Lemma 8. *Let $V^* = \{((\boldsymbol{v}^*(\boldsymbol{u}), \boldsymbol{u}), \boldsymbol{u} \in \mathbb{R}^D\}$. If $\sup_{t \geq 0}\|\boldsymbol{u}_t\| < \infty$, then $(\boldsymbol{v}_t, \boldsymbol{u}_t) \to V^* \quad w.p.1$.*

Proof. Having established Lemmas 1 to 5, we can use Theorem 2 from Borkar [3, Chapter 6]. □

Lemma 9. *The finite composition of Lipschitz-continuous and Lipschitz-smooth function is Lipschitz-continuous and Lipschitz-smooth.*

Proof. Let $f, g : X \to Y$ such that f and g are Lipschitz-continuous with constants L_f and L_g and Lipschitz-smooth with constants $L_{\dot{f}}$ and $L_{\dot{g}}$ respectively. Notice that the derivatives of f and g are therefore bounded by C_f and C_g.

We show that $f \circ g$ is Lipschitz-continuous.

$$
\|(f \circ g)(x) - (f \circ g)(z)\| = \|f(g(x)) - f(g(y))\|
$$

$$\leq L_f \|g(x) - g(y)\|$$
$$\leq L_f L_g \|x - y\|.$$

We can also show $f \circ g$ is Lipschitz-smooth. We present the proof for the one-dimensional case.

$$\begin{aligned}
\left\|(f \circ g)'(x) - (f \circ g)'(z)\right\| &= \left\|f'(g(x))\, g'(x) - f'(g(y))\, g'(y)\right\| \\
&= \|f'(g(x))\, g'(x) - f'(g(y))\, g'(y) + \\
&\quad f'(g(x))\, g'(y) - f'(g(x))\, g'(y)\| \\
&\leq \left\|f'(g(x))\, g'(x) - f'(g(x))\, g'(y)\right\| + \\
&\quad \left\|f'(g(x))\, g'(y) - f'(g(y))\, g'(y)\right\| \\
&\leq \left\|f'(g(x))\right\| \left\|g'(x) - g'(y)\right\| + \\
&\quad \left\|g'(y)\right\| \left\|f'(g(x)) - f'(g(y))\right\| \\
&\leq C_f \dot{L}_g \|x - y\| + C_g L_{\dot{f}} \|g(x) - g(y)\| \\
&\leq C_f \dot{L}_g \|x - y\| + C_g L_{\dot{f}} L_g \|x - y\| \quad \square
\end{aligned}$$

Appendix C. Feature updates for convergent deep reinforcement learning

Through Theorem 2, we established that the regularized Q-learning method is convergent with features that are changing over time, throughout the learning process, as long as those features converge. Now, we present three learning settings for the hidden layers that we can show to satisfy the assumption, i.e., three learning settings under which convergence of the features is guaranteed.

We consider an L-times differentiable function $h : \mathbb{R}^D \to \mathbb{R}$. We want to find z^* such that

$$z^* = \min_{z \in \mathbb{R}^D} h(z).$$

The stochastic full-gradient-descent method for solving the equation above takes the form

$$z_{t+1} = z_t + \beta_t \left(\nabla_z h(z_t) + Y_{t+1}\right),$$

where the random variables Y_t have zero-mean and bounded variance. We have the following result from [27].

Theorem 10. *Suppose that the function h is Lipschitz-continuous, Lipschitz-smooth, coercive and not asymptotically flat. Then, we have that the set of critical points $Z^* = \{z : \nabla_z h(z) = 0\}$ is not empty. Further suppose that the random variables Y_t have zero-mean and finite variance. Then,*

$$z_t \to Z^*_\infty \quad w.p.1,$$

where $Z^*_\infty \subseteq Z^*$ is a bounded connected component over which h is constant.

Theorem 10 provides general conditions under which the parameters of a stochastic approximation method that is, particularly, stochastic full-gradient-descent of a loss function h, converge to a bounded region with constant value. While Q-learning performs only semi-gradient descent and divergence of the parameters is known to happen, it is possible to learn the parameters of the features through stochastic full-gradient descent and obtain convergence guarantees, possibly at the expense of optimality [32, Chapter 11]. In the sequel, we propose three such feature learning methods. One of the proposed methods is based on an unsupervised learning update; another is based on a semi-supervised learning update; the final is based on a reinforcement learning update. In light of Theorem 10, all the proposed feature learning methods are in accordance with Assumption 2 and are thus guaranteed to converge. In order to guarantee boundedness of the features, we should post-process them with a final sigmoid layer, $\sigma : \mathbb{R} \to (0, 1)$, such that $\sigma(x) = \frac{1}{1+e^{-x}}$.

We consider the Huber loss $H : \mathbb{R}^D \to \mathbb{R}$ such that

$$H(z) = \min_{p \in \{1,2\}} \frac{1}{p} \|\text{loss}(z)\|_p^p,$$

in the conditions demanded by the theorem. The Huber loss is the 2-norm if z is close to the origin and the 1-norm otherwise. Consequently, we have that $\nabla_{\text{loss}(z)} H(z)$ equals $\text{loss}(z)$ around the origin and $\text{sign}(\text{loss}(z))$ otherwise. The finite linear combination and the finite composition of Lipschitz-continuous and Lipschitz-smooth functions is also Lipschitz-continuous and Lipschitz-smooth.

C.1. Unsupervised learning

We can learn a linear map that reduces the input space, linearly, through principal component analysis. In the non-linear case, we can instantiate such learning using auto-encoder [24] or variational auto-encoder [15] architectures. Contrastive learning has also been used in feature extraction [19]. All these methods have no task information but can still be powerful if dimensionality is an issue or we want to transfer learning across tasks or even domains [10]. We focus here on the definition of a loss function over which the stochastic gradient descent method is guaranteed to converge: the auto-encoder. Formally, the auto-encoder performs stochastic gradient descent over the loss function

$$h(\boldsymbol{u}, \boldsymbol{s}) = \mathbb{E}\left[H\left(\kappa_{\boldsymbol{s}}\left(\phi_{\boldsymbol{u}}\left(\psi(x,a)\right)\right) - \psi(x,a)\right)\right],$$

where $\psi(x,a)$ is a Euclidean representation of (x,a) in $\mathbb{R}^P$. In the auto-encoder, an encoder $\phi_{\boldsymbol{u}} : \mathbb{R}^P \to \mathbb{R}^K$, $\boldsymbol{u} \in \mathbb{R}^D$ learns to map the features into a latent space and a decoder $\kappa_{\boldsymbol{s}} : \mathbb{R}^K \to \mathbb{R}^P$ learns to reconstruct the original input. The features $\phi_{\boldsymbol{u}}$ can then be normalized and inputted to the final layer of our regularized Q-learning method. The auto-encoder has been applied successfully in reinforcement learning tasks [18]. In practice, the stochastic gradient updates are:

$$\begin{aligned}
\boldsymbol{u}_{t+1} &= \boldsymbol{u}_t - \beta_t \nabla_{\boldsymbol{u}} \phi_{\boldsymbol{u}_t}(\psi(x,a)) \cdot \\
&\quad \cdot \nabla \kappa_{\boldsymbol{s}_t}\left(\phi_{\boldsymbol{u}_t}(\psi(x,a))\right) \nabla H\left(\kappa_{\boldsymbol{s}}\left(\phi_{\boldsymbol{u}}(\psi(x,a))\right) - \psi(x,a)\right), \\
\boldsymbol{s}_{t+1} &= \boldsymbol{s}_t - \beta_t \nabla_{\boldsymbol{s}} \kappa_{\boldsymbol{s}_t}\left(\phi_{\boldsymbol{u}}(\psi(x,a))\right) \cdot \\
&\quad \cdot \nabla \phi_{\boldsymbol{u}_t}(\psi(x,a)) \nabla H\left(\kappa_{\boldsymbol{s}}\left(\phi_{\boldsymbol{u}}(\psi(x,a))\right) - \psi(x,a)\right).
\end{aligned}$$

The process of learning the final linear layer parameterized by $\boldsymbol{v}$ evolves concurrently as described in Section 5. If β_t is $o(\alpha_t)$, convergence of both sets of parameters happens with probability 1.

C.2. Semi-supervised learning

We can also approach the feature learning problem in a semi-supervised way grounded in the theory of Markov decision processes [7]. Specifically, instead of only learning to approximate inputs that are close in the original space, we can learn to approximate them if they are close to each other in the Markov decision process. Bisimulation metrics [8] give us a way to perform such learning, by considering that state-action pairs are similar if they produce similar rewards and lead to similar states. We can thus define the loss function

$$\begin{aligned}
h(\boldsymbol{u}) = \mathbb{E}\Bigg[& H\Big(H\left(\phi_{\boldsymbol{u}}(x,a) - \phi_{\boldsymbol{u}}(\tilde{x},\tilde{A})\right) - H(r-\tilde{r}) - \\
& - \gamma H\left(\phi_{\boldsymbol{u}}(x',\cdot) - \phi_{\boldsymbol{u}}(\tilde{x}',\cdot)\right)\Big)\Bigg]
\end{aligned}$$

and perform again stochastic gradient descent. The technique has also provided positive results in practice, specifically when compared to unsupervised learning [37]. In this semi-supervised setting, the stochastic gradient updates take the form

$$\begin{aligned}
\boldsymbol{u}_{t+1} = \boldsymbol{u}_t - \beta_t \big(& \nabla_{\boldsymbol{u}}\left(\phi_{\boldsymbol{u}_t}(x_t,a_t) - \phi_{\boldsymbol{u}_t}(\tilde{x}_t,\tilde{a}_t)\right) \nabla H\left(\phi_{\boldsymbol{u}_t}(x_t,a_t) - \phi_{\boldsymbol{u}_t}(\tilde{x}_t,\tilde{a}_t)\right) - \\
& - \gamma \nabla_{\boldsymbol{u}}\left(\phi_{\boldsymbol{u}_t}(x_t,a_t) - \phi_{\boldsymbol{u}_t}(\tilde{x}_t,\tilde{a}_t)\right) \nabla H\left(\phi_{\boldsymbol{u}_t}(x_t,\cdot) - \phi_{\boldsymbol{u}_t}(\tilde{x}_t,\cdot)\right)\big) \cdot \\
& \cdot \nabla H\big(H\left(\phi_{\boldsymbol{u}_t}(x_t,a_t) - \phi_{\boldsymbol{u}_t}(\tilde{x}_t,\tilde{a}_t)\right) - H(r-\tilde{r}_t) - \\
& - H\left(\phi_{\boldsymbol{u}_t}(x_t,\cdot) - \phi_{\boldsymbol{u}_t}(\tilde{x}_t,\cdot)\right)\big),
\end{aligned}$$

where $(\tilde{x}_t, \tilde{a}_t, \tilde{x}'_t, \tilde{r}_t)$ are sampled independent and identically distributed. Again, the learning method used to update the hidden parameters u can happen alongside learning the final linear layer.

C.3. Reinforcement learning

Q-learning assumes bootstrapped targets and thus makes only stochastic semi-gradient descent over the loss function. In practice, that choice may produce good results but is often unstable. We propose that we could learn the features through full stochastic gradient descent and learn the final layer through the usual regularized stochastic semi-gradient descent method. The loss function is

$$h(\boldsymbol{u}, \boldsymbol{v}') = \mathbb{E}\left[H\left(r(x,a) + \gamma \max_{a' \in \mathcal{A}} \phi_{\boldsymbol{u}}(x',a') \cdot \boldsymbol{v}' - \phi_{\boldsymbol{u}}(x,a) \cdot \boldsymbol{v}'\right)\right].$$

Recent work from [1] proves that, often, the stochastic full-gradient descent is able to perform competitively with the stochastic semi-gradient method. The updates are

$$\boldsymbol{u}_{t+1} = \boldsymbol{u}_t + \beta_t \nabla H \left(r_t + \gamma \max_{a' \in \mathcal{A}} \phi_{\boldsymbol{u}_t}(x'_t, a') \cdot \boldsymbol{v}'_t - \phi_{\boldsymbol{u}_t}(x_t, a_t) \cdot \boldsymbol{v}'_t \right) \cdot \\ \cdot \left(\nabla_{\boldsymbol{u}} \phi_{\boldsymbol{u}_t}(x_t, a_t) \boldsymbol{v}'_t - \gamma \nabla_{\boldsymbol{u}} \max_{a' \in \mathcal{A}} \phi_{\boldsymbol{u}_t}(x'_t, a') \boldsymbol{v}'_t \right),$$

$$\boldsymbol{v}'_{t+1} = \boldsymbol{v}'_t + \beta_t \nabla H \left(r_t + \gamma \max_{a' \in \mathcal{A}} \phi_{\boldsymbol{u}_t}(x'_t, a') \cdot \boldsymbol{v}'_t - \phi_{\boldsymbol{u}_t}(x_t, a_t) \cdot \boldsymbol{v}'_t \right) \cdot \\ \cdot \left(\phi_{\boldsymbol{u}_t}(x_t, a_t) - \gamma \nabla_{\boldsymbol{v}'} \max_{a' \in \mathcal{A}} \phi_{\boldsymbol{u}_t}(x'_t, a') \boldsymbol{v}'_t \right).$$

For the gradient of the max operator, we may consider a smooth approximation parameterized by α, $\max_\alpha$, such that $\max_\alpha(\boldsymbol{v}_1, \dots, \boldsymbol{v}_n) = \frac{\sum_{i=1}^n \boldsymbol{v}_i e^{\alpha \boldsymbol{v}_i}}{\sum_{i=1}^n e^{\alpha \boldsymbol{v}_i}}$. As $\alpha \to \infty$, $\max_\alpha \to \max$. The hidden parameters $\boldsymbol{u}$ that are being updated can be used to learn the final parameters $\boldsymbol{v}$ using the regular semi-gradient Q-learning update. In practice, our proposed approach is halfway between the full-gradient and semi-gradient methods and is able to capture the stability of full-gradient and performance of semi-gradient. The additional computational and training costs are minimal.

Data availability

No data was used for the research described in the article.

References

[1] K.E. Avrachenkov, V.S. Borkar, H.P. Dolhare, K. Patil, Full gradient DQN reinforcement learning: a provably convergent scheme, in: Modern Trends in Controlled Stochastic Processes, Springer, 2021, pp. 192–220.
[2] L. Baird, Residual algorithms: reinforcement learning with function approximation, in: Proceedings of the International Conference on Machine Learning, 1995, pp. 30–37.
[3] V. Borkar, Stochastic Approximation: A Dynamical Systems Viewpoint, Cambridge University Press, 2008.
[4] Q. Cai, Z. Yang, J.D. Lee, Z. Wang, Neural temporal-difference learning converges to global optima, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc., 2019.
[5] P. Dayan, G.E. Hinton, Feudal reinforcement learning, in: S. Hanson, J. Cowan, C. Giles (Eds.), Advances in Neural Information Processing Systems, Morgan-Kaufmann, 1993, pp. 271–278.
[6] Y. Duan, X. Chen, R. Houthooft, J. Schulman, P. Abbeel, Benchmarking deep reinforcement learning for continuous control, in: International Conference on Machine Learning, PMLR, 2016, pp. 1329–1338.
[7] N. Ferns, P. Panangaden, D. Precup, Metrics for finite Markov decision processes, in: Uncertainty in Artificial Intelligence, 2004, pp. 162–169.
[8] N. Ferns, P. Panangaden, D. Precup, Bisimulation metrics for continuous Markov decision processes, SIAM J. Comput. 40 (2011) 1662–1714.
[9] D. Ghosh, M.G. Bellemare, Representations for stable off-policy reinforcement learning, in: International Conference on Machine Learning, PMLR, 2020, pp. 3556–3565.
[10] I. Higgins, A. Pal, A. Rusu, L. Matthey, C. Burgess, A. Pritzel, M. Botvinick, C. Blundell, A. Lerchner, DARLA: improving zero-shot transfer in reinforcement learning, in: International Conference on Machine Learning, 2017, pp. 1480–1490, JMLR.org.
[11] M. Hutsebaut-Buysse, K. Mets, S. Latré, Hierarchical reinforcement learning: a survey and open research challenges, Mach. Learn. Knowl. Extr. 4 (2022) 172–221.
[12] M. Hutter, Extreme state aggregation beyond Markov decision processes, Theor. Comput. Sci. 650 (2016) 73–91.
[13] M. Hutter, S. Yang-Zhao, S.J. Majeed, Conditions on features for temporal difference-like methods to converge, arXiv preprint, arXiv:1905.11702, 2019.
[14] Y. Jiang, S.S. Gu, K.P. Murphy, C. Finn, Language as an abstraction for hierarchical deep reinforcement learning, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc., 2019.
[15] D.P. Kingma, M. Welling, Stochastic gradient VB and the variational auto-encoder, in: Second International Conference on Learning Representations, ICLR, 2014, p. 121.
[16] C. Lakshminarayanan, S. Bhatnagar, A stability criterion for two timescale stochastic approximation schemes, Automatica 79 (2017) 108–114.
[17] C.L. Lan, S. Tu, A. Oberman, R. Agarwal, M.G. Bellemare, On the generalization of representations in reinforcement learning, arXiv preprint, arXiv:2203.00543, 2022.
[18] S. Lange, M. Riedmiller, A. Voigtländer, Autonomous reinforcement learning on raw visual input data in a real world application, in: The 2012 International Joint Conference on Neural Networks (IJCNN), IEEE, 2012, pp. 1–8.
[19] M. Laskin, A. Srinivas, P. Abbeel, CURL: contrastive unsupervised representations for reinforcement learning, in: International Conference on Machine Learning, PMLR, 2020, pp. 5639–5650.
[20] A. Levy, G. Konidaris, R. Platt, K. Saenko, Learning multi-level hierarchies with hindsight, in: Proceedings of the 7th International Conference on Learning Representations, 2017.
[21] L. Li, T.J. Walsh, M.L. Littman, Towards a unified theory of state abstraction for MDPs, ISAIM 4 (2006) 5.
[22] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J.E. Gonzalez, M.I. Jordan, I. Stoica, RLlib: abstractions for distributed reinforcement learning, in: International Conference on Machine Learning (ICML), 2018.
[23] H.D. Lim, D.W. Kim, D. Lee, Regularized Q-learning, arXiv preprint, arXiv:2202.05404, 2022.
[24] C.Y. Liou, W.C. Cheng, J.W. Liou, D.R. Liou, Autoencoder for words, Neurocomputing 139 (2014) 84–96.
[25] A.M. Lyapunov, The general problem of the stability of motion, Int. J. Control 55 (1992) 531–534.
[26] S.J. Majeed, M. Hutter, On Q-learning convergence for non-Markov decision processes, in: International Joint Conference on Artificial Intelligence (IJCAI), 2018, pp. 2546–2552.
[27] P. Mertikopoulos, N. Hallak, A. Kavis, V. Cevher, On the almost sure convergence of stochastic gradient descent in non-convex problems, in: H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc., 2020, pp. 1117–1128.
[28] O. Nachum, S.S. Gu, H. Lee, S. Levine, Data-efficient hierarchical reinforcement learning, in: S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc., 2018.
[29] M.L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, John Wiley & Sons, 2005.
[30] H. Robbins, S. Monro, A stochastic approximation method, Ann. Math. Stat. 22 (1951) 400–407.

[31] S. Singh, T. Jaakkola, M.L. Littman, C. Szepesvári, Convergence results for single-step on-policy reinforcement-learning algorithms, Mach. Learn. 38 (2000) 287–308.
[32] R. Sutton, A. Barto, Reinforcement Learning: An Introduction, 2nd ed., MIT Press, 2018.
[33] J.N. Tsitsiklis, B. Van Roy, Feature-based methods for large scale dynamic programming, Mach. Learn. 22 (1996) 59–94.
[34] Z.T. Wang, M. Ueda, A convergent and efficient deep Q network algorithm, arXiv preprint, arXiv:2106.15419, 2021.
[35] C.J. Watkins, P. Dayan, Q-learning, Mach. Learn. 8 (1992) 279–292.
[36] P. Xu, Q. Gu, A finite-time analysis of Q-learning with neural network function approximation, in: International Conference on Machine Learning, PMLR, 2020, pp. 10555–10565.
[37] A. Zhang, R. McAllister, R. Calandra, Y. Gal, S. Levine, Learning invariant representations for reinforcement learning without reconstruction, arXiv preprint, arXiv:2006.10742, 2020.
[38] S. Zhang, H. Yao, S. Whiteson, Breaking the deadly triad with a target network, in: International Conference on Machine Learning, PMLR, 2021, pp. 12621–12631.