A Learning Rate Analysis of Reinforcement Learning Algorithms in Finite-Horizon

Frédérick Garcia
INRA/BIA, Auzeville BP 27
31326 Castanet Tolosan cedex
France
fgarcia@toulouse.inra.fr

Seydina M. Ndiaye
INRA/BIA, Auzeville BP 27
31326 Castanet Tolosan cedex
France
ndiaye@toulouse.inra.fr

Abstract

Many reinforcement learning algorithms, like Q-Learning or R-Learning, correspond to adaptative methods for solving Markovian decision problems in infinite-horizon when no model is available. In this article we consider the particular framework of non-stationary finite-horizon Markov Decision Processes. After establishing a relationship between the finite-horizon total reward criterion and the average-reward criterion in finite-horizon, we define $Q_{\mathcal{H}}$-Learning and $R_{\mathcal{H}}$-Learning for finite-horizon MDPs. Then we introduce the Ordinary Differential Equation (ODE) method to conduct a learning rate analysis of $Q_{\mathcal{H}}$-Learning and $R_{\mathcal{H}}$-Learning. $R_{\mathcal{H}}$-Learning appears to be a version of $Q_{\mathcal{H}}$-Learning with matrix-valued stepsizes, the corresponding gain matrix being very close to the optimal matrix which results from the ODE analysis. Experimental results confirm that performance hierarchy.

1 Introduction

The search for optimal policies in Markov Decision Processes has been deeply studied according to different optimality criteria and has led to the definition of the well known Bellman optimality equations, and dynamic programming algorithms [Puterman, 1994]. Most Reinforcement Learning (RL) algorithms that have been recently developed [Kaelbling et al., 1996, Bertsekas and Tsitsiklis, 1996] take a stochastic optimization approach to solve these optimality equations, by directly learning the optimal policies from iterated observations of rewards and state transitions, without a priori knowledge about the system.

In this paper we consider the case of non stationary Markov decision problems in a finite horizon. Despite being an accurate modelling of many applications concerning the management of industrial production systems, finite-horizon MDPs have not been yet specifically considered in reinforcement learning. This article is a first attempt to fill this gap.

Our work relies on two parts. First we propose a reformulation of the two main classical optimality criteria, expected total reward criterion and average expected reward criterion, given the finite-horizon assumption. After establishing an equivalence between them, we conclude that it is possible to use the two adapted reinforcement learning algorithms, $Q_{\mathcal{H}}$-Learning and $R_{\mathcal{H}}$-Learning, to learn optimal policies for non stationary finite-horizon MDPs.

Secondly, we conduct an analysis of the respective rates of convergence of $Q_{\mathcal{H}}$-Learning and $R_{\mathcal{H}}$-Learning. Surprisingly, $R_{\mathcal{H}}$-Learning appears to be a version of $Q_{\mathcal{H}}$-Learning with matrix-valued stepsizes. Furthermore, the ordinary differential equation (ODE) method enables to determine a theoretical optimal matrix-valued gain, and it appears that the gain corresponding to $R_{\mathcal{H}}$-Learning is numerically and structurally very close to that optimal gain. The experimental study we conducted confirms these results: in most situations we tested, $R_{\mathcal{H}}$-Learning performs better than $Q_{\mathcal{H}}$-Learning, and the implementation of the optimal matrix-valued gain defines a reinforcement algorithm that surpasses $R_{\mathcal{H}}$-Learning.

2 Reinforcement Learning in Finite-Horizon

2.1 Non-Stationary MDP in Finite-Horizon

The majority of reinforcement learning algorithms solve stationary infinite-horizon Markov Decision Problems. Given a state-space S and an action-space A, the dynamic of a Markov decision process is characterized as follows: at each time step $t \in T$, the execution of action $a_t \in A$ in state $x_t \in S$ leads to the new state $x_{t+1} \in S$ with a probability $p(x_{t+1} \mid x_t, a_t)$, and to the instantaneous reward $r(x_t, a_t)$.

A Markov Decision Problem is defined by adding to that process a performance criterion to maximize over a set of decisional policies. This criterion is a measure of the expected sum of the rewards along a trajectory, and policies are functions that indicate the action a_t to execute given informations about the past trajectory at time t. For stationary infinite-horizon Markov decision problems, most of the perfomance criteria lead to the existence of stationary optimal policies, i.e. functions π that map states in S to actions in A.

In finite-horizon problems, trajectories are sequences of exactly N transitions, with $T = \{1, \ldots, N\}$. The performance criterion considered in that case is the *finite total expected reward* criterion $V^{\pi}(x) = E_{\pi}[r(x_1, a_1) + r(x_2, a_2) + ... + r(x_N, a_N) \mid x_1 = x]$ where $x \in S$ and E_{π} is the expected value given the policy π.

When dealing with finite-horizon MDPs, the stationary assumption cannot be considered anymore. A first reason is that even for stationary finite-horizon MDP models (time-independent spaces S and A, transition probabilities $p()$ and rewards $r()$), optimal policies are no longer stationary, and are functions of $T \times S$ into A: $t, x \overset{\pi}{\rightarrow} \pi(t, x)$ [Puterman, 1994].

More practically, in most of the problems of industrial production process control that lead to finite-horizon MDPs, the main cause of non-stationarity of optimal policies is the non-stationarity of the MDP model itself: is is very common to have different state-spaces, decision-spaces, transition probabilities and reward values at each decision step. In order to take into account this characteristic, we consider the following formal model of finite-horizon MDP: 1) to each time step $i \in T = \{1, 2, \ldots, N\}$ is associated a finite state space S_i and a finite decision space A_i; 2) for each step $i \in \{1, 2, \ldots, N-1\}$, the execution of action $a_i \in A_i$ from the state $x_i \in S_i$ leads to the new state $x_{i+1} \in S_{i+1}$ with a probability $p_i(x_{i+1} \mid x_i, a_i)$, and with the instantaneous reward $r_i(x_i, a_i)$; 3) at the last decision step, the system receives a reward $r_N(x_N, a_N)$ after the execution of a_N in x_N, and stops.

For this particular kind of MDP a policy π can be decomposed into a set $\{\pi_1, \pi_2, ..., \pi_N\}$ of policies $\pi_i : S_i \rightarrow A_i$. For each decision step, a value function associated to π is defined as $V_i^{\pi}(x) = E_{\pi}[\sum_{t=i}^{t=N} r_t(x_t, \pi_t(x_t)) \mid x_i = x]$.

We say a policy π is optimal if it maximizes the value function V_1^{π} on S_1. For this criterion, the classical Bellman optimality equations that characterize optimal policies are

$$V_i^*(x) = \max_a \left\{ r_i(x, a) + \sum_{y \in S_{i+1}} p_i(y \mid x, a) V_{i+1}^*(y) \right\} \tag{1}$$

for all $x \in S_i$, $i \in \{1, .., N\}$ and $V_{N+1}^* = 0$ [Puterman, 1994]. Then $\pi_i^*(x) = \text{argmax}_a \{r_i(x, a) + \sum_{y \in S_{i+1}} p_i(y \mid x, a) V_{i+1}^*(y)\}$. This optimality equation has a single solution $V^* = \{V_1^*, ..., V_N^*\}$, that can be easily obtained by a dynamic programming algorithm in $O(N.n_A.n_S^2)$ complexity (for state spaces S_i and decision spaces A_i of constant size n_S and n_A) when transition probabilities and reward function are known [Puterman, 1994]. The associated learning problem is to adaptatively estimate the optimal value functions V_i^* and the corresponding policies π_i^*, from observed transitions and rewards when the Markov decision process is not known.

2.2 Q$_{\mathcal{H}}$-Learning in Finite-Horizon

Q-Learning [Watkins, 1989] is based on the rewriting of the Bellman optimality equation, replacing the $V^{\pi}(x)$ value function of a policy by a new function $Q^{\pi}(x, a)$: for all $x \in S_i$, $a \in A_i$ $Q_i^{\pi}(x, a) = r_i(x, a) + \sum_{y \in S_{i+1}} p_i(y \mid x, a) V_{i+1}^{\pi}(y)$.

We have $V^{\pi}(x) = Q^{\pi}(x, \pi(x))$, and the optimality equation becomes

$$Q_i^*(x, a) = r_i(x, a) + \sum_{y \in S_{i+1}} p_i(y \mid x, a) \max_b Q_{i+1}^*(y, b),$$

for all $x \in S_i$ and $a \in A_i$. Then $V^*(x) = \max_a Q^*(x, a)$ and $\pi^*(x) = \text{argmax}_a Q^*(x, a)$.

Q-Learning is a reinforcement learning algorithm allowing the iterative generation of the solution Q^* and the optimal policies π^*. The algorithm consists in updating at each iteration n the estimation Q_n of the value function Q^*, from the current observed transition and reward $< x_n, a_n, y_n, r_n >$. Q-Learning is a natural candidate for solving finite-horizon MDPs. It is indeed easy to transform an N-step non-stationary MDP into an infinite-horizon process, by adding an artificial final absorbing state x_{abs}, that is reward-free and such that all actions a_N in A_N lead with probability 1 to x_{abs} (figure 1).

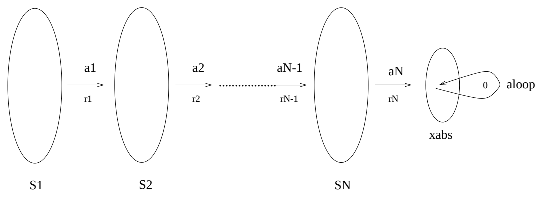

Figure 1: infinite process with absorbing state

Hence the first reinforcement learning algorithm we propose for finite horizon MDPs is:

Finite-Horizon Q$_{\mathcal{H}}$-Learning

Observe $< x_n, a_n, y_n, r_n >$
Update

$$Q_{n+1}(x,a) = Q_n(x,a) + \alpha_n(x,a).e_n \tag{2}$$

with $e_n =$

$$\begin{cases} r_n + \max_b Q_n(y_n,b) - Q_n(x,a) \text{ if } (x,a)=(x_n,a_n), x_n \in S_i, i<N \\ r_n - Q_n(x,a) \text{ if } (x,a)=(x_n,a_n), x_n \in S_N \\ 0 \text{ otherwise} \end{cases}$$

If $x_n \notin S_N$ set $x_{n+1} = y_n$;
otherwise choose randomly x_{n+1} in S_1.
If $x_{n+1} \in S_j$ select a_{n+1} in A_j

In this algorithm $Q_0(x,a) = 0$ and $\alpha_n(x,a)$ are small learning rates decaying over time. The state exploration is classically determined by the dynamic of the process (that is, $x_{n+1} = y_n$), until the last decision step is reached and we restart a new trajectory by choosing randomly a new initial state in S_1. The specific learning rule for S_N is equivalent to directly setting $V^*_{N+1}(x_{abs}) = Q^*_{N+1}(x_{abs}, a_{loop}) = 0$. The action selection is as usual based on an exploration function.

Let us assume that each pair (x,a) in $S_i \times A_i$ is visited an infinite number of times, and that $\sum_n \alpha_n(x,a) = \infty$ and $\sum_n \alpha_n^2(x,a) < \infty$. The convergence of Q-Learning [Watkins and Dayan, 1992, Jaakkola et al., 1994, Tsitsiklis, 1994] in case of no-discounting ($\gamma = 1$) and with the presence of reward-free absorbing states proves that this finite-horizon Q$_{\mathcal{H}}$-Learning algorithm will converge in probability 1 towards the optimal value function: $\forall x \in S_i$, $a \in A_i$, $\lim_{n\to\infty} Q_n(x,a) = Q^*_i(x,a)$ a.s. with $V^*_i(x) = \max_{a \in A_i} Q^*_i(x,a)$.

2.3 R$_{\mathcal{H}}$-Learning and the average-reward criterion

The average reward criterion was introduced in Reinforcement Learning by Schwartz through the R-Learning algorithm [Schwartz, 1993]. It has been studied since then by many researchers [Singh, 1994, Ok and Tadepalli, 1996, Mahadevan, 1996b]. The goal is to search for *gain-optimal* policies that maximize the expected payoff per step, which is a very natural measure of optimal acting:

$$\rho^\pi(x) = \lim_{n\to\infty} E_\pi[\frac{1}{n}\sum_{t=1}^{n} r_t \mid x_1 = x].$$

For the particular case of *unichain* MDPs (that is, for all policy π, the Markov chain $\{x_n\}_n$ contains a single recurrent class of states, and a possibly empty set of transient states), the average reward associated to each policy is independent of the state : $\rho^\pi(x) = \rho^\pi(y) = \rho^\pi$. For simplicity reasons, most of the results concerning average reward criterion in Reinforcement Learning have been established with this unichain assumption [Mahadevan, 1996b].

A more selective optimality criterion can be defined. It is based on a new value function U^π of a policy π, called *bias value* [Puterman, 1994]. For all state $x \in S$ we have

$$U^\pi(x) = \lim_{n\to\infty} E_\pi[\sum_{t=1}^{n}(r_t - \rho^\pi) \mid x_1 = x].$$

A policy π^* is said to be *bias-optimal* (or *T-optimal* in [Schwartz, 1993]) if it is gain-optimal, and if $U^{\pi^*}(x) \geq U^\pi(x)$ for all x and all policy π.

The existence of optimal stationary policies for gain and bias optimality has been shown [Puterman, 1994]. For all unichain MDPs, there exists a pair (U^*, ρ^*) solution of the Bellman equation for the average criterion:

$$U^*(x) + \rho^* = \max_a \left(r(x,a) + \sum_y p(y \mid x,a) U^*(y) \right), \tag{3}$$

for all $x \in S$, such that the average reward of the policy π^* that maximizes the right-hand side of (3) is the optimal average reward ρ^*. Furthermore, if $(U^{*\prime}, \rho^{*\prime})$ is also a solution of (3), then $\rho^* = \rho^{*\prime}$. The solutions U^* of (3) are not unique, since for each solution (U^*, ρ^*), the pair $(U^* + k, \rho^*)$ is also a solution (one can show that this is a complete characterization of the set of solutions for unichain MDP models [Puterman, 1994]).

That last remark shows that (3) is not sufficient to produce bias optimal policies. Another optimality equation, based on a third notion of value function called *bias offset*, is generally required [Puterman, 1994, Mahadevan, 1996a].

In order to adapt the average-reward criterion to finite-horizon MDPs, we first transform the initial process $S_1 \to S_N$ into a new infinite process $S_1 \to S_N \circlearrowright S_1$. The natural solution we propose is to close artificially the loop between S_N and S_1 by adding a uniform transition: $\forall x \in S_N, \forall a \in A_N, \forall y \in S_1, p_N(y \mid x, a) = \frac{1}{n_{S_1}}$ (figure 2).

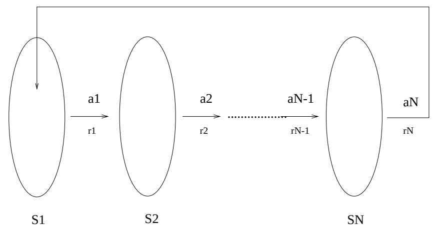

Figure 2: infinite process with looping on S_1

For the new MDP $S_1 \to S_N \circlearrowright S_1$ we proved the following proposition [Garcia and Ndiaye, 1998]:

Proposition 1 *For the cycling process $S_1 \to S_N \circlearrowright S_1$, for all policy π,*

$$\forall x \in S_1 \quad \rho^\pi(x) \quad = \quad \frac{1}{N}\frac{1}{n_{S_1}}\sum_{x_1 \in S_1} V_1^\pi(x_1) = \rho^\pi$$

$$\begin{aligned}\forall x \in S_{i,\, i=1,\ldots,N} \quad U^\pi(x) \quad &= \quad V_i^\pi(x) - (N-i+1)\rho^\pi, \\ \sum_{x_1 \in S_1} U^\pi(x_1) \quad &= \quad 0.\end{aligned}$$

The first aspect of this result, the state independence of ρ^π, is not surprising since the looping $S_N \circlearrowright S_1$ transforms the original MDP into a *unichain* process. More interesting are the next equalities; the second one establishes an equivalence between the bias-value function, the average reward and the value function in finite-horizon, and the last one completely determines this bias-value function. From that properties we proved the following theorem [Garcia and Ndiaye, 1998]:

Theorem 1 *If (U^*, ρ^*) is a solution of average-reward Bellman equation (3) for $S_1 \to S_N \circlearrowright S_1$ with the constraint $\sum_{x_1 \in S_1} U^*(x_1) = 0$, and if π^* is an associated gain-optimal policy, then the value functions $V_i^*(x) = U_i^*(x) + (N-i+1)\rho^*$ are solutions of finite-horizon Bellman equation (1), and π^* is a policy that maximizes $V_i^\pi(x)$ for $x \in S_i$, $i = 1, \ldots, N$.*

That result shows that there is an equivalence between the finite-horizon and average-reward criteria, and a solution of (3) necessary leads to a solution of (1). The following corollary characterizes more deeply this equivalence [Garcia and Ndiaye, 1998].

Corollary 1 *If (U^*, ρ^*) is solution of (3) for $S_1 \to S_N \circlearrowright S_1$ with $\sum_{x_1 \in S_1} U^*(x_1) = 0$, then (U^*, ρ^*) also defines a bias-optimal solution.*

From these results, it appears natural to use R-Learning for solving finite-horizon MDPs. The second reinforcement learning algorithm we propose, called R$_\mathcal{H}$-Learning, is an adaptation of R-Learning with an update rule for states in S_N that directly integrates the final condition $R_N^*(x,a) = r_N(x,a) - \rho^\pi$ for $x \in S_N$, $a \in A_N$:

Finite-Horizon R$_\mathcal{H}$-Learning

Observe $< x_n, a_n, y_n, r_n >$
Update

$$\begin{aligned}R_{n+1}(x,a) &= R_n(x,a) + \alpha_n(x,a).e_n \\ \rho_{n+1} &= \rho_n + \beta_n.e'_n \\ e_n &= \begin{cases} r_n - \rho_n + \max_b R_n(y_n,b) - R_n(x,a) \\ \text{ if } (x,a)=(x_n,a_n)\ x_n \in S_i,\ i<N \\ r_n - \rho_n - R_n(x,a) \\ \text{ if } (x,a)=(x_n,a_n)\ x_n \in S_N \\ 0 \text{ otherwise} \end{cases} \\ e'_n &= \begin{cases} r_n - \rho_n + \max_b R_n(y_n,b) - R_n(x_n,a_n) \\ \text{ if } x_n \in S_i,\ i<N\ a_n = \pi_n(x_n) \\ r_n - \rho_n - R_n(x_n,a_n) \\ \text{ if } x_n \in S_N\ a_n = \pi_n(x_n) \\ 0 \text{ otherwise} \end{cases}\end{aligned}$$

If $x_{n+1} = y_n \in S_j$ select a_{n+1} in A_j

3 A learning rate analysis of $\mathrm{Q}_{\mathcal{H}}$-Learning and $\mathrm{R}_{\mathcal{H}}$-Learning

The simulations we conducted from a random finite-MDP generator (see [Garcia and Ndiaye, 1998] and section 4) have shown experimentally that $\mathrm{Q}_{\mathcal{H}}$-Learning and $\mathrm{R}_{\mathcal{H}}$-Learning always converge to an optimal policy, and that $\mathrm{R}_{\mathcal{H}}$-Learning is most of the time faster than $\mathrm{Q}_{\mathcal{H}}$-Learning. However, it still does not exist any definitive theoretical results about the convergence of R-Learning-like algorithms, with a mixed iteration on R_n and ρ_n.

The aim of this section is to introduce a comparison of the respective learning rates of convergence of $\mathrm{Q}_{\mathcal{H}}$-Learning and $\mathrm{R}_{\mathcal{H}}$-Learning. The analysis we propose below is made possible by an original equivalent transformation of $\mathrm{R}_{\mathcal{H}}$-Learning into a new reinforcement algorithm, the form of which is closer to $\mathrm{Q}_{\mathcal{H}}$-Learning. We first present that transformation.

3.1 An equivalent formulation of $\mathrm{R}_{\mathcal{H}}$-Learning

Just consider the second equation of Proposition 1. It sets a direct relation between the value functions U^π and V^π of a policy π, that can be directly translated in terms of functions R^π and Q^π : $\forall x \in S_i$, $\forall a \in A_i$ $\;R_i^\pi(x,a) = Q_i^\pi(x,a) - (N-i+1)\rho^\pi$. From that observation we propose to transform the iteration on R_n in the $\mathrm{R}_{\mathcal{H}}$-Learning algorithm by an iteration on Q_n. With this aim, we define the new series $\{Q_n\}_n$:

$$\forall x \in S_i,\ \forall a \in A_i \quad Q_n(x,a) = R_n(x,a) + (N-i+1)\rho_n \tag{4}$$

with $\{R_n\}_n$ and $\{\rho_n\}_n$ the two series of the $\mathrm{R}_{\mathcal{H}}$-Learning algorithm. That transformation leads to the following equivalent reinforcement algorithm:

Finite-Horizon $\mathrm{R}_{\mathcal{H}}$-Learning - Q formulation

$$\begin{array}{rcl} Q_{n+1}(x,a) & = & Q_n(x,a) + \gamma_n(x,a).e_n \qquad (5) \\ \rho_{n+1} & = & \rho_n + \beta_n.e_n \text{ if } a_n = \pi_n(x_n) \\ e_n & = & \begin{cases} r_n + \max_b Q_n(y_n,b) - Q_n(x_n,a_n) \\ \quad \text{if } x_n \in S_i,\, i<N \\ r_n - Q_n(x_n,a_n) \text{ if } x_n \in S_N \end{cases} \\ \gamma_n(x,a) & = & \begin{cases} \alpha_n(x,a) + (N-i+1)\beta_n \\ \quad \text{if } (x,a)=(x_n,a_n),\, x_n \in S_i,\, a_n = \pi_n(x_n) \\ \alpha_n(x,a) \\ \quad \text{if } (x,a)=(x_n,a_n),\, a_n \neq \pi_n(x_n) \\ (N-i+1)\beta_n \\ \quad \text{if } (x,a)\neq(x_n,a_n),\, x \in S_i,\, a_n = \pi_n(x_n) \\ 0 \text{ otherwise} \end{cases} \end{array}$$

As we can see, the two series $\{Q_n\}_n$ and $\{\rho_n\}_n$ are now decoupled. Furthermore, since $\pi_n(x) = \mathrm{argmax}_a R_n(x,a) = \mathrm{argmax}_a (Q_n(x,a) - (N-i+1)\rho_n) = \mathrm{argmax}_a Q_n(x,a)$, the $\{\rho_n\}_n$ iteration is even not necessary to determine the current policy π_n.

Hence the two algorithms $\mathrm{Q}_{\mathcal{H}}$-Learning and $\mathrm{R}_{\mathcal{H}}$-Learning in finite-horizon can be considered as two different updating rules of the same value function Q. More precisely, the main difference between $\mathrm{Q}_{\mathcal{H}}$-Learning and $\mathrm{R}_{\mathcal{H}}$-Learning can now be clearly associated to the number of components $Q_n(x,a)$ which are modified at each iteration of the algorithm. In $\mathrm{Q}_{\mathcal{H}}$-Learning, we only update the component $Q_n(x_n,a_n)$. In $\mathrm{R}_{\mathcal{H}}$-Learning, if $(x,a) \neq (x_n,a_n)$, $Q(x,a)$ can still be updated if the action a_n corresponds to a greedy action for the state x_n in the policy π_n.

3.2 Reinforcement Learning and the ODE method

Now that we have seen that $\mathrm{R}_{\mathcal{H}}$-Learning is a parallel version of $\mathrm{Q}_{\mathcal{H}}$-Learning, we intend to compare their respective rates of convergence. The theoretical tool we have chosen is the Ordinary Differential Equation (ODE) method recently introduced in reinforcement learning [Bertsekas and Tsitsiklis, 1996, Kushner and Yin, 1997]. The ODE method results from the combination of dynamical systems and stochastic approximation techniques. The classical theory of stochastic approximation introduced by Robbins and Monro [Robbins and Monro, 1951] concerns the analysis of adaptive stochastic algorithms

$$\theta_{n+1} = \theta_n + \gamma_n H(\theta_n, X_{n+1}) \tag{6}$$

where θ_n is the parameter vector, and X_n the input random vector bringing some information on θ_n at time n. The application of this theory to the domain of Reinforcement Learning has led to general proofs of convergence for Q-Learning or $TD(\lambda)$ [Jaakkola et al., 1994, Tsitsiklis, 1994]. The ODE method was initially proposed by Ljung [Ljung, 1977], and then has been the source of many works, as in [Kushner and Clark, 1978, Benaïm, 1996]. It consists in the introduction of the averaged differential equation $\dfrac{d\theta}{dt} = \overline{H}(\theta)$ where $\overline{H}(\theta) = \lim\limits_{n\to\infty} E[H(\theta, X_n)]$, the behaviour of which can be compared to the asymptotic behaviour of (6).

The use of the ODE method for analysing learning algorithms like neural nets has originally been introduced by Benaïm [Benaïm, 1995]. An application to

the analysis of reinforcement learning algorithms has already
been considered in [Bertsekas and Tsitsiklis, 1996, Kushner and Yin, 1997], where convergence analysis of Q-Learning are presented. The point we want to emphasize in this article is that the ODE method can also be applied to study the learning rates of reinforcement learning algorithms like $\mathrm{Q}_{\mathcal{H}}$-Learning and $\mathrm{R}_{\mathcal{H}}$-Learning.

The representation we adopt for this study is the following: the parameter vector θ to estimate is the optimal value function Q^*, X_n represents the observation at time n, and is defined as $X_n = (x_{n-1}, a_{n-1}, x_n)$, $H()$ is the update rule of $\mathrm{Q}_{\mathcal{H}}$-Learning in finite-horizon. Here $H(Q,(x,a,y))$ is set to the vector:

$$\begin{array}{c} \\ \\ (x,a) \\ \\ \\ \end{array} \left(\begin{array}{c} 0 \\ \vdots \\ \begin{cases} r(x,a) + \max_b Q(y,b) - Q(x,a) & \text{if } x \in S_i, i < N \\ r(x,a) - Q(x,a) & \text{if } x \in S_N \end{cases} \\ \vdots \\ 0 \end{array} \right)$$

Thus the two algorithms $\mathrm{Q}_{\mathcal{H}}$-Learning and $\mathrm{R}_{\mathcal{H}}$-Learning can be described as

$$Q_{n+1} = Q_n + \frac{1}{n}\Gamma H(Q_n, X_{n+1}) \qquad (7)$$

where $\Gamma = \Gamma^{Q_{\mathcal{H}}}$ or $\Gamma^{R_{\mathcal{H}}}$ is an adaptive gain matrix. For $\beta_n = 0$ and $\alpha_n(x,a) = \frac{1}{n}$, $\Gamma^{Q_{\mathcal{H}}} = \Gamma^{R_{\mathcal{H}}} = I$ which corresponds to the simplest version of $\mathrm{Q}_{\mathcal{H}}$-Learning. Therefore, within the ODE method, $\mathrm{Q}_{\mathcal{H}}$-Learning and $\mathrm{R}_{\mathcal{H}}$-Learning can be considered as two discrete approximations with adaptive matrix-valued gains $\Gamma^{Q_{\mathcal{H}}}$ and $\Gamma^{R_{\mathcal{H}}}$ of the same differential equation:

$$\frac{dQ}{dt} = h(Q) \qquad (8)$$

with $h(Q) = \lim_{n \to \infty} E_Q[H(Q, X_n)]$. $h(Q)$ can be calculated from the stationary distribution μ^Q of the Markov chain $\{X_n\}_n$ given a constant parameter Q: $h(Q) = \sum_X H(Q,X)\mu^Q(X)$.

To calculate the stationary distribution μ^Q we have to take into account the fact that for reinforcement learning algorithms, the input sequence $\{X_n\}_n$ is a Markov process controlled by the parameter vector Q_n itself [Benveniste et al., 1990]. For Markovian exploration functions, where a_n depends probabilistically of x_n and Q_n, μ^Q is given by:

$$\forall X = (x,a,x'), \quad \mu^Q(X) = \mu_E^Q(x) P_{exp}^Q(a \mid x) P(x' \mid x,a)$$

where μ_E^Q is the stationary distribution of the Markov chain $\{x_n\}_n$ defined by $P(x' \mid x) = \sum_{a \in A_i} p_i(x' \mid x,a) P_{exp}^Q(a \mid x)$ for $x \in S_i$, and $P_{exp}^{Q_n}(a_n \mid x_n)$ is the selection probability of the exploration function. Since we added a uniform return form S_N to S_1, we have $\forall x \in S_1$, $\mu_E^Q(x) = \frac{1}{N.n_{S_1}}$. Iteratively, μ_E^Q can be computed on each state-space S_i as: $\forall x \in S_{i+1}$, $\mu_E^Q(x) = \sum_{z \in S_i} P(x \mid z) \mu_E^Q(z)$.

We easily check that Q^* is a stable attractive point of (8). First we can see that $h(Q^*) = 0$. Moreover, we can calculate the jacobian matrix of h on that point:

$$h_Q(Q^*) = (\frac{d}{dQ}h)(Q^*) = \qquad (9)$$

$$\begin{array}{cc} & \quad\quad\quad\quad (x',a') \\ (x,a) & \left(\begin{array}{cc} & \vdots \\ \cdots & \begin{cases} -\mu^*(x,a) & \text{if } (x,a)=(x',a') \\ p_i(x' \mid x,a)\mu^*(x,a) & \text{if } x \in S_i, i<N, \\ & \quad x' \in S_{i+1}, a' = \pi^*(x') \\ 0 & \text{otherwise} \end{cases} \end{array} \right) \end{array}$$

with $\pi^*(x) = \mathrm{argmax}_a\, Q^*(x,a)$ and $\mu^*(x,a) = \mu_E^{Q^*}(x) P_{exp}^{Q^*}(a \mid x)$. The eigenvalues of $h_Q(Q^*)$ are equal to $-\mu^*(x,a)$. They are strictly negative with the simple assumptions that the Markov chain $\{x_n\}_n$ is recurrent at $Q = Q^*$, and that $\forall x, a\ P_{exp}^{Q^*}(a \mid x) > 0$.

Based on that material, we can now focus on the problem of the learning rate analysis, and its application to the comparison between $\mathrm{Q}_{\mathcal{H}}$-Learning and $\mathrm{R}_{\mathcal{H}}$-Learning in finite-horizon MDPs.

3.3 Optimal matrix-valued learning rates

The use of a matrix-valued gain to guide and accelerate the convergence of a stochastic adaptive algorithm is a classic result of stochastic approximation theory [Benveniste et al., 1990, Kushner and Yin, 1997].

For the algorithm (7) the gain matrices Γ that maintain Q^* as a stable equilibrium of the new ODE

$$\frac{dQ}{dt} = \Gamma h(Q),$$

are characterized by $\forall \lambda$ eigenvalue of $\frac{1}{2}I + \Gamma . h_Q(Q^*)$, $\mathcal{R}e(\lambda) < 0$. Among all these matrices, it is possible to prove [Benveniste et al., 1990, Kushner and Yin, 1997] that the one that minimizes the asymptotic variance $\lim_{n \to \infty} ||Q_n - Q^*||^2$ is defined by

$$\Gamma^* = -{h_Q}^{-1}(Q^*) \tag{10}$$

As we can see the knowledge of the target parameter Q^* is generally required and an adaptive matrix-valued gain Γ_n that converges toward Γ^* is often used. In our case, Γ^* can be calculated by inverting (9). We obtain the following upper triangular optimal matrix [Garcia and Ndiaye, 1998]:

$$\Gamma^* = \begin{array}{cc} & (x',a') \\ (x,a) & \left(\begin{array}{cc} & \vdots \\ \cdots & \begin{cases} \frac{1}{\mu^*(x,a)} & \text{if } (x,a)=(x',a') \\ \frac{P^{Q^*}(x'|x,a)}{\mu^*(x',a')} & \text{if } x \in S_i, x' \in S_j, \\ & \quad i<j, a'=\pi^*(x') \\ 0 & \text{otherwise} \end{cases} \end{array} \right) \end{array}$$

where $P^Q(x' \mid x, a)$ is the probability of going from $x \in S_i$ to $x' \in S_j$, $1 \leq i < j \leq N$, in $j - i$ steps, by first executing the action a, and then by following the current policy $\pi^Q(x) = \text{argmax}_a Q(x, a)$ for the last $j - i - 1$ steps.

3.4 Comparison between $\Gamma^{Q_\mathcal{H}}$, $\Gamma^{R_\mathcal{H}}$ and Γ^*

For the two algorithms $\text{Q}_\mathcal{H}$-Learning (2) and $\text{R}_\mathcal{H}$-Learning (5) that we consider in this paper, the gains $\Gamma^{Q_\mathcal{H}}$ and $\Gamma^{R_\mathcal{H}}$ are adaptive gains that depend on n, but also on X_n and Q_n.

In order to be able to compare these matrix-valued gains with the optimal gain Γ^*, it is necessary to consider their asymptotic behaviour. If we assume classically that $\alpha_n(x, a) = \frac{\alpha_0}{N(x,a)}$ where $N(x, a)$ is the total number of times the pair (x, a) was visited at time n, and $\beta_n = \frac{\beta_0}{n}$, we show in [Garcia and Ndiaye, 1998] that $\Gamma^{Q_\mathcal{H}}$ and $\Gamma^{R_\mathcal{H}}$ converge respectively toward:

$$\Gamma_\infty^{Q_\mathcal{H}} = \begin{pmatrix} \ddots & & 0 \\ & \frac{\alpha_0}{\mu^*(x,a)} & \\ 0 & & \ddots \end{pmatrix}$$

$$\Gamma_\infty^{R_\mathcal{H}} = \begin{array}{cc} & \vdots (x',a') \\ \underset{\cdots}{(x,a)} & \left(\begin{cases} \frac{\alpha_0}{\mu^*(x,a)} + & (N-i+1)\beta_0 \\ & \text{if } (x,a)=(x',a'), x \in S_i , a=\pi^*(x) \\ \frac{\alpha_0}{\mu^*(x,a)} & \text{if } (x,a)=(x',a'), a \neq \pi^*(x) \\ (N-i+1)\beta_0 & \text{if } (x,a) \neq (x',a'), x \in S_i, a'=\pi^*(x') \\ 0 & \text{if } (x,a) \neq (x',a'), a' \neq \pi^*(x') \end{cases} \right) \end{array}$$

A first remark about these matrix-valued gains is that the stability condition on Q^* implies that $\alpha_0 > \frac{1}{2}$. This explains some empirical results concerning R-Learning which reveal that higher initial values of α_0 are to be preferred to lower values [Mahadevan, 1996b].

It is now interesting to compare $\Gamma_\infty^{Q_\mathcal{H}}$, $\Gamma_\infty^{R_\mathcal{H}}$ and Γ^*. For $\alpha_0 = 1$ and a small β_0, the three matrices have more or less the same diagonal values, which is a confirmation of the good choice $\alpha_n(x, a) = \frac{1}{N(x,a)}$, asymptotically equivalent to $\frac{1}{n\mu^*(x,a)}$.

Another important similarity between $\Gamma_\infty^{R_\mathcal{H}}$ and Γ^* is about the structure of the matrices : both of them have exactly the same null columns.

4 Simulations

In order to experimentally compare $\text{Q}_\mathcal{H}$-Learning, $\text{R}_\mathcal{H}$-Learning and the Γ^*Q-Learning corresponding to (7) with $\Gamma = \Gamma^*$, we have developed a random finite-MDP generator. At each step i, a set S_i of n_S states and a set A_i of n_A actions are defined. Each transition from S_i to S_{i+1} is characterized by a set of n_A transition matrices $p_i(. \mid ., a)$ and n_A reward vectors $r_i(., a)$. The problem parameters are N, n_S and n_A. The reward values $r_i(s, a)$ and the probabilities $p_i(s' \mid s, a)$ are drawn in $[0, 1]$ from a random number generator, with the constraints $\sum_{s'} p_i(s' \mid s, a) = 1$.

For a given random MDP, we first calculate the exact finite-horizon optimal policy π^* with the classical N-step backward dynamic programming algorithm, using the p_i and r_i values. Then we calculate μ^*, P^{Q^*}, and finally the Γ^* optimal gain.

We evaluated the performance of the 3 algorithms $\text{Q}_\mathcal{H}$-Learning, $\text{R}_\mathcal{H}$-Learning and Γ^*Q-Learning on different random MDPs [Garcia and Ndiaye, 1998].

The learning parameters α_n et β_n were defined by

$$\alpha_n(x, a) = \frac{\alpha_0}{N(x, a)} \text{ and } \beta_n = \frac{\beta_0}{n/N},$$

where $N(x, a)$ is the number of times the action a has been chosen in the state x. We used $\alpha_0 = 1$ and $\beta_0 = 0.4$ for all simulations, with a semi-uniform exploration function ($\tau = 90\%$). These choices of learning rates were made to optimize the behaviour of $\text{Q}_\mathcal{H}$-Learning and $\text{R}_\mathcal{H}$-Learning on the set of problems we considered.

We chose $\rho_{\pi_n} = \frac{1}{N.n_S} \sum_{x \in S_1} V_1^{\pi_n}(x)$ as a performance measure of the current policy π_n at iteration n, and the variability of this policy π_n for different learning trajectories was taken into account by calculating the mean of the value ρ_{π_n} on M different runs (we took $M = 5$). More precisely we considered for each algorithm the two criteria $C_1 : \rho_{\pi_n}/\rho^*$ for $n = 500N$, and $C_2 : \rho_{\pi_n}/\rho^*$ for $n = 5000N$, where $\rho^* = \frac{1}{N.n_S} \sum_{x \in S_1} V_1^*(x)$ is the optimal average gain of π^*.

The first surprising fact we noticed was that most of the time the Γ^*Q-Learning did not converge. An explanation we found is that for large sized problems, the initial gains $\frac{1}{\mu^*(x,a).n}$ of Γ^*Q-Learning are too large, and make the series $\{Q_n\}_n$ leaves its convergence domain. To fix that problem we decided to replace Γ^* by an adapative matrix Γ_n^* asymptotically equivalent to Γ^*, where $\frac{n}{N(x,a)}$ is used instead of $\frac{1}{\mu^*(x,a)}$. The results we finally obtained showed that Γ_n^*Q-Learning is a bit better than $R_{\mathcal{H}}$-Learning, and that both of them are always faster than $Q_{\mathcal{H}}$-Learning, as illustrated in table 3 and figure 4.

ρ_{π_n}/ρ^* %	C_1 $(n = 500N)$			C_2 $(n = 5000N)$		
N, n_S, n_A	$Q_{\mathcal{H}}$	$R_{\mathcal{H}}$	Γ_n^*Q	$Q_{\mathcal{H}}$	$R_{\mathcal{H}}$	Γ_n^*Q
5,25,10	79.54	89.71	92.55	97.90	99.39	99.42
5,50,10	71.79	84.57	89.93	96.50	98.15	99.12
5,50,50	65.26	78.56	80.32	94.53	95.87	95.95
5,100,10	62.87	78.82	87.19	91.79	95.85	97.91
5,300,10	56.42	64.32	82.24	77.88	89.24	93.27
10,25,10	74.35	90.57	95.90	94.62	99.28	99.83
10,50,50	61.72	77.72	88.06	92.72	95.81	97.61
10,100,10	61.21	79.01	91.41	87.21	95.80	97.94
50,50,10	63.43	85.24	86.94	72.38	98.32	97.39
50,50,50	58.53	79.65	84.52	74.54	95.67	96.87

Figure 3: Relative evaluation of $Q_{\mathcal{H}}$-Learning, $R_{\mathcal{H}}$-Learning and Γ_n^*Q-Learning.

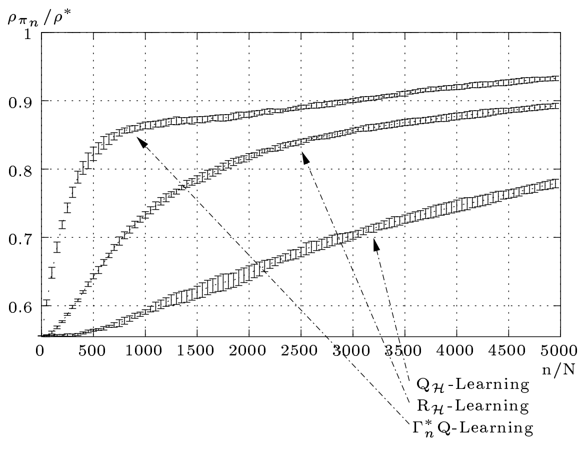

Figure 4: n_A=10, n_S=300, N=5 (5 runs).

5 Conclusion

The underlying goal of this article was to tackle the problem of using reinforcement learning algorithms in the framework of finite-horizon Markov Decision Processes. Two main results have been obtained.

First we proved the equivalence between the total reward criterion and the average-reward criterion in finite horizon. An interesting conclusion is that classical Q-Learning and R-Learning algorithms can be adapted to define $Q_{\mathcal{H}}$-Learning and $R_{\mathcal{H}}$-Learning algorithms in finite horizon. Both of these algorithms converge experimentally toward the optimal V_i^* value functions, with a convergence proof for $Q_{\mathcal{H}}$-Learning.

The other important result is about the comparison between the learning rates of $Q_{\mathcal{H}}$-Learning and $R_{\mathcal{H}}$-Learning. It appears that $R_{\mathcal{H}}$-Learning can be seen as a version of $Q_{\mathcal{H}}$-Learning using matrix-valued stepsizes, where several components of the Q function are updated simultaneously. Furthermore, we showed that this stepsize matrix is structurally and numerically very close to the optimal gain matrix proposed by the ODE method, and that $R_{\mathcal{H}}$-Learning performs very similarly to the learning algorithm corresponding to this optimal gain matrix. Consequently we argue in favor of using $R_{\mathcal{H}}$-Learning when solving finite-horizon MDPs.

Different open questions still deserve to be considered. First we would like to know whether it is possible to derive from Γ_n^*Q-Learning an equivalent reinforcement learning algorithm where only one component is updated at each state transition, like it is the case for $R_{\mathcal{H}}$-Learning in its initial formulation. For the moment, independently of the fact that it requires to know P^{Q^*} and μ^*, Γ_n^*Q-Learning cannot be used in practice since it is too much slow.

Another question we are currently considering is to exploit the equivalent Q formulation of $R_{\mathcal{H}}$-Learning for proving its convergence. Some recent theoretical results concerning the ODE method could be sufficient, like in [Benaïm et al., 1998].

Finally, we are trying to generalize our results concerning the convergence of $Q_{\mathcal{H}}$-Learning and $R_{\mathcal{H}}$-Learning to Q-Learning and R-Learning within the classical framework of stationary infinite MDPs.

Acknowledgements

We thank Michel Benaïm for fruitful discussions about the ODE method.

References

[Benaïm, 1995] Benaïm, M. (1995). Convergence Theorem for Hybrid Learning Rules. *Neural Computation*, 7(1):19–25.

[Benaïm, 1996] Benaïm, M. (1996). A Dynamical System Approach to Stochastic Approximations. *SIAM Control and Optimisation*, 34(2):437–472.

[Benaïm et al., 1998] Benaïm, M., Fort, J. C., and Pages, G. (1998). Almost sure convergence of the one-dimensional kohonen algorithm. *Advances in Applied Probability.* To appear.

[Benveniste et al., 1990] Benveniste, A., Metivier, M., and Priouret, P. (1990). *Adaptive Algorithms and Stochastic Approximation.* Springer-Verlag, Berlin,New York.

[Bertsekas and Tsitsiklis, 1996] Bertsekas, D. P. and Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming.* Athena Scientific, Belmont (MA).

[Garcia and Ndiaye, 1998] Garcia, F. and Ndiaye, S. (1998). Reinforcement Learning in finite-horizon: Optimality criteria and algorithms. Technical report, Department of Biometry and Artificial Intelligence, INRA, Toulouse, France.

[Jaakkola et al., 1994] Jaakkola, T., Jordan, M., and Singh, S. (1994). On the Convergence of Stochastic Iterative Dynamic Programming Algorithms. *Neural Computation*, 6:1185–1201.

[Kaelbling et al., 1996] Kaelbling, L., Littman, M., and Moore, A. W. (1996). Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research (JAIR)*, 4:237–285.

[Kushner and Clark, 1978] Kushner, H. and Clark, D. (1978). *Stochastic Approximation Methods for Constrained and Unconstrained Systems.* Springer-Verlag, Berlin,Heildelberg,New York.

[Kushner and Yin, 1997] Kushner, H. and Yin, G. (1997). *Stochastic Approximation Algorithms and Applications.* Springer.

[Ljung, 1977] Ljung, L. (1977). Analysis of recursive stochastic algorithms. *IEEE Transactions on Automatic Control*, 22:551–575.

[Mahadevan, 1996a] Mahadevan, S. (1996a). An Average-Reward Reinforcement Learning Algorithm for Computing Bias-Optimal Policies. In *AAAI National Conference*, volume 13.

[Mahadevan, 1996b] Mahadevan, S. (1996b). Average Reward Reinforcement Learning: Foundations, Algorithms and Empirical Results. *Machine Learning*, 22:159–196.

[Ok and Tadepalli, 1996] Ok, D. and Tadepalli, P. (1996). Auto-exploratory average reward reinforcement learning. In *AAAI National Conference*, volume 13.

[Puterman, 1994] Puterman, M. (1994). *Markov Decision Processes.* John Wiley and Sons, New York.

[Robbins and Monro, 1951] Robbins, H. and Monro, S. (1951). A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407.

[Schwartz, 1993] Schwartz, A. (1993). A Reinforcement Learning Method for Maximizing Undiscounted Rewards. In *International Conference on Machine Learning*, volume 10.

[Singh, 1994] Singh, S. (1994). Reinforcement Learning Algorithms for Average-Payoff Markovian Decision Processes. In *AAAI International Conference*, volume 12.

[Tsitsiklis, 1994]
Tsitsiklis, J. N. (1994). Asynchronous Stochastic Approximation and Q-Learning. *Machine Learning*, 16:185–202.

[Watkins, 1989] Watkins, C. (1989). *Learning from Delayed Rewards.* PhD thesis, Cambridge University, Cambridge, England.

[Watkins and Dayan, 1992] Watkins, C. and Dayan, P. (1992). Q-Learning. Technical Note. *Machine Learning*, 8(3):279–292.