Error Reducing Sampling in Reinforcement Learning

Bruno Scherrer, Shie Mannor

Error reducing sampling in reinforcement learning

Bruno Scherrer
CSAIL, MIT
NE43-783
200 Technology square
Cambridge MA, 02139-4307

Shie Mannor
LIDS, MIT
Room 35-406
77 Massachussetts Avenue
Cambridge MA, 02139-4307

Abstract

In reinforcement learning, an agent collects information interacting with an environment and uses it to derive a behavior. This paper focuses on efficient sampling; that is, the problem of choosing the interaction samples so that the corresponding behavior tends quickly to the optimal behavior. Our main result is a sensitivity analysis relating the choice of sampling any state-action pair to the decrease of an error bound on the optimal solution. We derive two new model-based algorithms. Simulations demonstrate a quicker convergence (in the sense of the number of samples) of the value function to the real optimal value function.

Introduction

In reinforcement learning, an agent collects information interacting with an environment and uses it to derive a behavior. This paper focuses on efficient sampling; that is, the problem of choosing the interaction samples so that the corresponding behavior tends quickly to the optimal behavior. The problem we consider here is different from the well-known exploration-exploitation dilemma (Kumar, 1985), in which an agent wants to collect information *while* optimizing its interaction. In this paper we consider the case where the agent wants to find the samples that will allow it to tend to the optimal behavior with fewer samples, *while* not caring about its exploration performance.

A typical setting where the present work might be useful is when the agent has a *practice* epoch at its disposal when its performance does not matter. For instance, it might be a computer game player which is practicing before a competition like the famous Back-Gammon TD-player (Tesauro, 1995), or a robot which learns in a non-harmful environment (e.g. on Earth) before actually going to a similar risky environment (e.g. on Mars) (Bernstein *et al.*, 2001). Another case where performance during training is irrelevant is *neurodynamic programming* (Bertsekas and Tsitsiklis, 1996), where reinforcement learning methods are used to solve very large MDPs in simulation. Tackling this sampling issue is all the more relevant when sampling has a high cost (in the robot example, interacting with the world costs a lot of time and energy). In all these problems, we want the computed behavior to tend to the optimal behavior quickly *with the number of samples*.

Our approach is the following: we first derive a confidence bound on the optimal value function, then we make a sensitivity analysis relating the choice of sampling any state-action pair to the tightening of this confidence bound. Our main result is Theorem 3, where we actually predict how sampling in a given state-action pair will tighten the confidence bound. With such an analysis, an agent can, step after step, choose to sample the state-action pair that will tighten its confidence on its behavior quality the most. Going even further, section 5 introduces an Error MDP, whose optimal policy corresponds to the best sampling strategy for tightening the confidence bound *in the long-term*.

Most work in reinforcement learning sampling analysis (Bertsekas and Tsitsiklis, 1996; Even-Dar *et al.*, 2003; Kearns and Koller, 1999; Kearns and Singh, 1998) rely on the maximum L_∞ norm. Though sufficient for many convergence results, L_∞ bounds are often disappointing as they don't give a precise picture of where and why the approximation is bad. In this paper, we provide a confidence bound with respect to the L_1 norm. Such a bound allows us to have a more precise insight of *where* and *how much* in the state-action space, sampling error on the parameters R and T incur a global cost on the value function.

The paper is organized as follows. Section 1 presents the core of reinforcement learning: we briefly present the theory of optimal control with Markov decision processes (MDPs) and the certainty equivalence method for reinforcement learning. Section 2 reviews recent results for analyzing approximations in the MDP framework. In section 3, we apply this analysis to the reinforcement learning problem and prove the key theorem of this paper: Theorem 3 shows how to estimate the effect of sampling a particular state-action pair on the approximation error. Section 4 then describes two new algorithms that are based on this key theorem. Section 5 illustrates experimentally and discusses the results of these algorithms. Finally, section 6 provides a discussion of the related literature.

1 The model

Markov Decision processes (MDP) (Puterman, 1994) provide the theoretical foundations of challenging problems to researchers in artificial intelligence and operation research. These problems include optimal control and reinforcement learning (Sutton and Barto, 1998).

A Markov Decision Process is a controlled stochastic process satisfying the Markov property with rewards (numerical values) assigned to state-action pairs. Formally, an MDP $\mathcal{M}$ is a four-tuple $\langle S, A, T, R\rangle$ where S is the state space, A is the action space, T is the transition function and R is the reward function. T is the state-transition probability distribution conditioned by the action; for all state-action pairs (s,a) and possible subsequent states s': $T(s,a,s') \overset{def}{=} \mathbb{P}(s_{t+1}=s'|s_t=s,\ a_t=a)$. $R(s,a)\in\mathbb{R}$ is the random variable which corresponds to the instantaneous reward for taking action $a\in A$ in state S. We assume throughout this paper that R is bounded. Then, without loss of generality, we also assume that it is contained in the interval $(0, R_{max})$.

Given an MDP $\langle S, A, T, R\rangle$, the optimal control problem consists in finding a sequence of actions $(a_0, a_1, a_2, ...)$ that maximises the expected long-term discounted sum of rewards: $\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)|\, s_0=s,\ a_t\right]$ where the expectation is over the runs of the Markov chain induced by $(a_0, a_1, a_2, ...)$, and $\gamma\in(0,1)$ is a discount factor. A well-known fundamental result is that an optimal sequence of actions can be derived from a deterministic function $\pi : S \to A$, called policy, which prescribes which action to take in every state. The value function of a policy π at state s is the expected long-term discounted amount of rewards if one follows policy π from state s: $V^\pi(s) \overset{def}{=} \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)|\, s_0=s,\ a_t=\pi(s_t)\right]$ where the expectation is over the runs of the Markov chain induced by π, and satisfies for all states s: $V^\pi(s) \overset{def}{=} \mathbb{E}[R(s,\pi(s))] + \gamma\sum_{s'}T(s,\pi(s),s')V^\pi(s')$. The Q-function of a policy π for state-action pair (s,a) is the expected long-term discounted amount of rewards if one does action a from state s and then follows the policy π: $Q^\pi(s,a) \overset{def}{=} \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)|\, s_0=s,\ a_t=\left\{\begin{array}{l} a \text{ if } t=0\\ \pi(s_t) \text{ else}\end{array}\right.\right]$ and satisfies: $Q^\pi(s,a) \overset{def}{=} \mathbb{E}[R(s,a)] + \gamma\sum_{s'}T(s,a,s')V^\pi(s')$.

Given these notations, the optimal control problem amounts to finding an optimal policy π^* whose value V^*, called the optimal value function, is the greatest for all states: $\forall s\in S,\ V^*(s) = \max_\pi V^\pi(s)$. Such an optimal policy exists and its value function V^*, is the unique fixed point a contraction mapping, so that for all states s: $V^*(s) = \max_a\left(\mathbb{E}[R(s,a)] + \gamma\sum_{s'}T(s,a,s')V^*(s')\right)$. The corresponding optimal Q-function $Q^*(s,a) \overset{def}{=} \mathbb{E}[R(s,a)] + \gamma\sum_{s'}T(s,a,s')V^*(s')$ is particularly interesting as it enables us to derive a deterministic optimal policy $\pi^*(s)$ as follows: $\pi^*(s) = \arg\max_a Q^*(s,a)$. A standard algorithm for solving optimal control is Policy Iteration (Puterman, 1994) which converges to the optimal solution in a finite number of iterations.

The reinforcement learning problem is a variant of optimal control where the MDP parameters (R and T) are initially unknown, and therefore must be estimated through sample experiments (Sutton and Barto, 1998). While optimal control only involves planning, reinforcement learning involves both learning (estimation of the parameters) and planning and is therefore a slightly more difficult problem. A standard and natural solution to this problem, known as the certainty equivalence method (Kumar and Varaiya, 1986), consists in estimating the unknown parameters R and T, and then deriving a policy from these estimates. Let $\#(s,a)$ be the number of times one has taken action a in state s. Let $\#(s,a,s')$ be the number of times one arrived in state s' after having taken action a in state s. Let $\Sigma R(s,a)$ be the cumulative amount of rewards received when taking action a in state s. The idea of the certainty equivalence method is to solve the MDP $\hat{\mathcal{M}} = \langle S, A, \hat{T}, \hat{R}\rangle$ where

$$\hat{R}(s,a) \overset{def}{=} \frac{\Sigma R(s,a)}{\#(s,a)} \text{ and } \hat{T}(s,a,s') \overset{def}{=} \frac{\#(s,a,s')}{\#(s,a)}. \qquad (1)$$

are the maximum-likelihood estimates of R and T. After a finite number of samples, choosing the optimal policy given this empirical model is clearly an approximation. The next sections provide an explicit analysis of this approximation.

2 The approximation error

In this section, we review some recent general results about approximation in MDPs. We will apply them to the reinforcement learning case in the next section.

Recall that, in the discounted optimal control problem, we want to find the optimal value function V^* which satisfies for all states s: $V^*(s) = \max_a [B_a V^*](s)$ where B_a, often referred to as the Bellman operator, returns for any real-valued function W on S, and any action a, a new real-valued function of s: $[B_a W](s) \overset{def}{=} \mathbb{E}[R(s,a)] + \gamma \sum_{s'} T(s,a,s')W(s')$. Consider that, instead of using this Bellman operator B_a, we use a slightly different Bellman operator $\hat{B}_a$, which for any real-valued function W on S, and any action a, returns the following function of s : $[\hat{B}_a W](s) \overset{def}{=} \hat{R}(s,a) + \gamma \sum_{s'} \hat{T}(s,a,s')W(s')$. We shall call $\hat{B}_a$ the approximate Bellman operator as it is based on some approximate parameters $\hat{R}$ and $\hat{T}$. For any policy π, let $\hat{V}^\pi$ be the value of the policy based on these approximate parameters. Similarly, let $\hat{V}^*$ be the corresponding optimal value function and $\hat{\pi}^*$ the corresponding optimal policy. In the remaining of this section, we show how to analyze the error due to using $\hat{B}_a$ instead of B_a.

Suppose $e(s,a)$ is an upper bound of the error if using the approximate parameters to operate on the real optimal value function V^*: $\left|[B_a V^*](s) - [\hat{B}_a V^*](s)\right| \leq e(s,a)$. As e measures how much error applying once the approximate Bellman operator will incur, we call it the 1-step error. Though in practice, the 1-step error might be difficult to estimate as it depends on unknown quantities (B_a and V^*), next section will show how to estimate it in the reinforcement learning case. Let $\mathbb{1}$ be the indicatrice function. For any transition function $\tilde{T}$ and any policy π, let $I_{\tilde{T},\pi}$ be the discounted sum of occupations of the dynamical system whose dynamics is characterized by $\tilde{T}$ and π; $I_{\tilde{T},\pi}$ is the solution of a linear system of size $|S|$: $\forall s \in S$, $I_{\tilde{T},\pi}(s) = 1 + \gamma \sum_{s'} \tilde{T}(s', \pi(s'), s) I_{\tilde{T},\pi}(s')$. The following theorem (Munos and Moore, 2000) allows to compute the approximation error:

Theorem 1
1) Given a policy π, if $E^\pi(s)$ satisfies for all states s, $E^\pi(s) = e(s,\pi(s)) + \gamma \sum_{s'} \hat{T}(s,\pi(s),s')E^\pi(s')$ then for all states s, $|V^\pi(s) - \hat{V}^\pi(s)| \leq E^\pi(s)$.
2) If $E^*(s)$ satisfies for all states s, $E^*(s) = \max_a e(s,a) + \gamma \max_a \left(\sum_{s'} \hat{T}(s,a,s')E^*(s')\right)$ then for all states s, $|V^*(s) - \hat{V}^*(s)| \leq E^*(s)$.
3) We can quantify the relation between the 1-step error e and these approximation error bounds: $\frac{\partial \|E^\pi\|_1}{\partial e(s,a)} = \mathbb{1}_{\{a=\pi(s)\}} I_{\hat{T},\pi}(s)$ and $\frac{\partial \|E^*\|_1}{\partial e(s,a)} = \mathbb{1}_{\{a=\pi_e(s)\}} I_{\hat{T},\pi_{E^*}}(s)$ where $\pi_e(s) \overset{def}{=} \arg\max_a e(s,a)$ and $\pi_{E^*}(s) \overset{def}{=} \arg\max_a \left(\sum_{s'} \hat{T}(s,a,s').E^*(s')\right)$ are the policies that incur the worst errors in the equation characterizing E^*.

Note that, in contrast to the optimal value function V^* which depends upon the real transition function T, the approximation errors E^π and E^* depend upon the estimate transition function $\hat{T}$. These approximation errors can thus be easily computed. Indeed, the theorem we have just described can be practically exploited in the following manner. Given $e(s,a)$ for all state-action pairs (s,a) (we show in the next section how to derive such a quantity), the error policy $\pi_e = \arg\max_a e(s,a)$ can be derived in a straightforward way. Then the approximation error E^*, and the policy π_E can be computed with an algorithm similar to Policy Iteration (see algorithm 1). By using the triangle inequality, it is also

Algorithm 1 `Error`
```
Input: a state space $S$, an action space $A$, an approximate $\hat{T}$, an upper bound on the one-step error $e$, the policy $\pi_e$, and a discount factor $\gamma$
Output: the approximation error E and the policy $\pi_E$
  Initialize $\pi_E$ arbitrarily
  repeat
    1. Find the solution $(E(s_1), E(s_2), ...)$ of the linear system which satisfies for all states $s$:
       $E(s) = e(s, \pi_e(s)) + \gamma \sum_{s' \in S} \hat{T}(s, \pi_E(s), s')E(s')$
    2. Update the policy $\pi_E$ for all states s:
       $\pi_E(s) \leftarrow \arg\max_{a \in A} \left( \sum_{s' \in S} \hat{T}(s,a,s')E(s') \right)$
  until convergence
```

easy to derive a bound on the distance between the *real* value of the optimal policy π^* and the *real* value of the optimal policy $\hat{\pi}^*$ derived from the approximate model:

$$\|V^* - V^{\hat{\pi}^*}\|_1 \leq \|V^* - \hat{V}^*\|_1 + \|\hat{V}^* - V^{\hat{\pi}^*}\|_1 \leq \|E\|_1 \tag{2}$$

with $E \overset{def}{=} E^* + E^{\pi^*}$, and to relate it to the 1-step error:

$$\frac{\partial \|E\|_1}{\partial e(s,a)} = \mathbb{1}_{\{a=\pi_{E^*}(s)\}} I_{\hat{T},\pi_{E^*}}(s) + \mathbb{1}_{\{a=\hat{\pi}^*(s)\}} I_{\hat{T},\hat{\pi}^*}(s). \tag{3}$$

3 The sampling error

The analysis of the previous section shows how to compute an error bound while using the optimal policy of the approximate model given the 1-step error e. In this section, we provide two key theorems that link the approximation analysis and reinforcement learning. The proofs are deferred to the appendix.

Our first important result shows how to relate the 1-step error e to the amount of sampling $\#(s,a)$ in each state-action (s,a):

Theorem 2
Fix $\delta > 0$. Then with probability at least $1-\delta$, for all

state-action pairs (s,a):

$$\left|[B_a V^*](s) - [\hat{B}_a V^*](s)\right| \leq \frac{\lambda\mu}{\sqrt{\#(s,a)}}$$

where $\lambda \overset{def}{=} R_{max}\left(1+\frac{\gamma|S|}{1-\gamma}\right)$ *and* $\mu \overset{def}{=} \sqrt{\frac{1}{2}\log\frac{2|S||A|(|S|+1)}{\delta}}$ *are two constant numbers.*

As a corollary, if we set $e(s,a) \overset{def}{=} \frac{\lambda\mu}{\sqrt{\#(s,a)}}$, and if we derive E as described in previous section (see Equation 2), we obtain that $\|V^* - V^{\hat{\pi}^*}\|_1 \leq \|E\|_1$ with probability at least $1-\delta$. In other words, this analysis provides a confidence bound on the quality of the policy given the number of samples $\#(s,a)$ in every state-action pair.

Because in the analysis so far, we have considered the L_1 norm instead of the usual L_∞ norm, we can predict the effect of sampling on the error bounds $E^* \geq |V^* - \hat{V}^*|$ and $E \geq |V^* - V^{\hat{\pi}^*}|$ we introduced in the previous section (Theorem 1 and Equation 2). Suppose the agent is about to sample some state-action pair (s,a). Before it does so, the agent has upper bounds of the error $\|E^*\|_1$ and $\|E\|_1$ which hold with high probability. After the agent has actually sampled, it might compute new error bounds $\|E'^*\|_1$ and $\|E'\|_1$. Let $\Delta\|E^*\|_1 \overset{def}{=} \|E'^*\|_1 - \|E^*\|_1$ and $\Delta\|E\|_1 \overset{def}{=} \|E'\|_1 - \|E\|_1$ be the variations of these error bounds when sampling some state-action pair. Let us also define the function $f(k) \overset{def}{=} \frac{1}{\sqrt{k+1}} - \frac{1}{\sqrt{k}}$. We can predict the evolution of the error bounds without actually having to compute them for all state-action pairs:

Theorem 3
Fix $\delta > 0$. Then with probability at least $1-\delta$, sampling action a in state s will affect the error bounds as follows:

$$\Delta\|E^*\|_1 = \mathcal{S}^*(s,a) + o\left(f(\#(s,a))\right)$$
$$\Delta\|E\|_1 = \mathcal{S}(s,a) + o\left(f(\#(s,a))\right)$$

with

$$\mathcal{S}^*(s,a) \overset{def}{=} \lambda\mu 1\!\!1_{\{a=\pi_e(s)\}} I_{\hat{T},\pi_E^*}(s)|f(\#(s,a))| \quad (4)$$

$$\mathcal{S}(s,a) \overset{def}{=} \lambda\mu\left(1\!\!1_{\{a=\pi_e(s)\}} I_{\hat{T},\pi_E^*}(s) + 1\!\!1_{\{a=\hat{\pi}^*(s)\}}.I_{\hat{T},\hat{\pi}^*}(s)\right)|f(\#(s,a))| \quad (5)$$

As f is quickly decreasing to 0 ($f(k) \sim k^{-\frac{3}{2}}$ when $k \to \infty$), the $o(.)$ term is more and more negligible as the number of samples grows. This fundamental result prescribes to sample the state-action pair (s,a) for which the scores $\mathcal{S}(s,a)$ or $\mathcal{S}^*(s,a)$ are the biggest. We show how to practically exploit this information through two algorithms in the next section.

4 Two sampling algorithms

This section provides two new algorithms for efficient sampling in reinforcement learning that are based on the analysis of the previous sections. We consider two cases, the off-line case and the online case. In the off-line case, the agent can sample any action from any state whenever it wants to. The on-line case is a bit more tricky: at any time, the agent is in one state, it chooses an action and then gets to a new state and can only sample from this new state; the on-line case is in other words constrained by the real interaction dynamics.

The off-line case algorithm (see algorithm 2) is rather straight-forward. It can be used with any of the two score function $\mathcal{S}^*$ and $\mathcal{S}$ defined just after Theorem 3. At each iteration, the agent computes the approximate value function, derives the corresponding error bound, estimates the effect of sampling (this involves 2 linear systems inversions for $\mathcal{S}^*$ and 4 for $\mathcal{S}$), and samples the state-action pair that will decrease the approximation error bound the most.

Algorithm 2 `Off-line sampling`

```
Input: a state space S, an action space A and a discount factor $\gamma$
Output: an approximate value function $V$, a policy $\pi$, and a confidence bound $E$
  Initializations: $\hat{R}(s,a) \leftarrow 0$, $\hat{T}(s,a,s') = 0$, $e(s,a) \leftarrow +\infty$, $\pi_e$ arbitrary.
  repeat
    $(\hat{V},\hat{\pi}) \leftarrow$ SolveMDP($\langle S,A,\hat{T},\hat{R}\rangle,\gamma$).
    $(E,\pi_E) \leftarrow$ Error($S,A,\hat{T},e,\pi_e,\gamma$).
    Compute one of the score functions $\mathcal{S}(s,a)$ for all $(s,a)$ (Equation 4 or 5).
    Sample the state-action pair $(s,a)$ that maximizes $\mathcal{S}(s,a)$.
    Update the parameters $\hat{R}(s,a)$ and $\hat{T}(s,a,.)$ given the observed result of sampling (Equation 1).
    Update the error parameters $e(s,a)$ and $\pi_e(s,a)$.
  until stopped
```

In the on-line case, one wants the agent to take a sequence of actions that minimizes the approximation error in the long-term. Indeed, the agent needs not only choose the best current sample, it must also plan to go to regions of the state space where sampling is useful. To do so, we introduce an Error MDP $\mathcal{M}^{\mathcal{S}} = \langle S,A,\hat{T},\mathcal{S}\rangle$, whose optimal policy maximizes the long-term decrease of error (see algorithm 3). Here we consider the score $\mathcal{S}$ but our arguments are similar for S^*. If the agent follows the optimal policy of this Error MDP, it should expect to maximize the discounted sum of error decreases $\sum_{t=0}^{\infty}\gamma^t\mathcal{S}(s_t,a_t)$. As every new sample might change the score $\mathcal{S}$, the Error MDP must

Algorithm 3 `On-line sampling`

```
Input: a state space S, an action space A, a starting state $s$, and a discount factor $\gamma$
Output: an approximate value function $V$, a policy $\pi$, and a confidence bound $E$
  Initializations: $\hat{R}(s,a) \leftarrow 0$, $\hat{T}(s,a,s') = 0$, $e(s,a) \leftarrow +\infty$, $\pi_e$ arbitrary.
  repeat
    $(\hat{V},\hat{\pi}) \leftarrow$ SolveMDP($\langle S,A,\hat{T},\hat{R}\rangle,\gamma$).
    $(E,\pi_E) \leftarrow$ Error($S,A,\hat{T},e,\pi_e,\gamma$).
    Compute one of the score functions $\mathcal{S}(s,a)$ for all $(s,a)$ (Equation 4 and 5).
    $(.,\pi_{explore}) \leftarrow$ SolveMDP($\langle S,A,\hat{T},\mathcal{S}\rangle,\gamma$).
    Execute the action: $a \leftarrow \pi_{explore}(s)$.
    Update the current state $s$, and the parameters $\hat{R}(s,a)$ and $\hat{T}(s,a,.)$ (Equation 1).
    Update the error parameters $e(s,a)$ and $\pi_e(s,a)$.
  until stopped
```

in theory be solved at each time step. The choice of one action (which thus involves solving an MDP) might look like a heavy computation. Nevertheless, it is easy to see that after sampling action a in state s, any score $\mathcal{S}$ will at most vary by $\frac{2\lambda\mu}{1-\gamma}f(\#(s,a))$. Let $V_t^{\mathcal{S}}$ denote the optimal value function of the Error MDP $\mathcal{M}^{\mathcal{S}}$ at time t, then $\|V_{t+1}^{\mathcal{S}} - V_t^{\mathcal{S}}\|_\infty \leq \frac{2\lambda\mu}{(1-\gamma)^2}f(\#(s,a))$. In other words, while time goes, the variation of $V_t^{\mathcal{S}}$ gets smaller and smaller. This suggests that starting the resolution of $\mathcal{M}^{\mathcal{S}}$ at time $t+1$ with its solution at time t will speed up the process. We study the practical efficiency of these algorithms in the next section.

5 Experiments

We have experimented our two algorithms, each with the two possible score criteria $\mathcal{S}^*$ and $\mathcal{S}$, and compared them to two standard sampling approaches: 1) Random Sampling: At any iteration, one chooses an action a uniformly at random in A (in the offline case, one also chooses s uniformly at random). 2) Exhaustive Sampling: At any iteration, one chooses the action a (in the offline case, the state-action pair (s,a)) that has been experienced the less.

We considered two classes of problems: 1) Random Grid MDPs: We created a set of random 5×5 grid MDPs, with 4 actions, where transitions are local: the next-state distribution from the state of coordinates (x,y) on the grid only includes the states $\{(x \pm 1, y \pm 1)\}$. 2) Howard's Automobile Replacement problem: We consider this 40-state 41-action MDP as it is defined in (Howard, 1960), because it often stands for a test-bed in the optimization literature.

In all experiments, we set the discount factor γ to 0.99 and we randomly chose a starting state. We measured, sample after sample, the efficiency of the 4 different exploration strategies by computing 1) the *real* relative distance between the real optimal value function and the approximate optimal function $C^* \overset{def}{=} \frac{\|V^*-\hat{V}\|_1}{\|V^*\|_1}$ and 2) the *real* relative distance between the real optimal value function and the real value of the approximate optimal policy $C \overset{def}{=} \frac{\|V^*-V^{\hat{\pi}^*}\|_1}{\|V^*\|_1}$. Recall that $\|E^*\|_1$ (resp. $\|E\|_1$) constitutes an upper bound of $\|V^*\|_1 C^*$ (resp. $\|V^*\|_1 C$), and that using score $\mathcal{S}^*$ (resp. $\mathcal{S}$) is aimed at reducing $\|E^*\|_1$ (resp. $\|E\|_1$). In figure 1 and 2, we display a typical performance evolution of C^* and C for the offline and the online cases; this gives 4 sub-figures by problem. We show the performance evolution for the 4 different exploration strategies: using score $\mathcal{S}^*$, using score $\mathcal{S}$, exhaustive sampling, random sampling; this thus gives 4 curves. For each curve, we ran the simulations 20 times and display the median and the ranges (after having removed the 4 worst and best values). For all these curves, the quicker they go to 0 the better.

A first glance at figures 1 and 2 leads to the following general observation: the score $\mathcal{S}$ is better than $\mathcal{S}^*$ for decreasing C *and* C^*. This is somehow surprising, as $\mathcal{S}$ was designed to minimize C while $\mathcal{S}^*$ was designed to minimize C^*; indeed, one would have expected that using $\mathcal{S}^*$ would be more efficient for decreasing C^*. We think that the reason why this effect stands is related to the fact that the score $\mathcal{S}$ depends upon the current approximate optimal policy $\hat{\pi}^*$ whereas $\mathcal{S}^*$ does not (compare definitions of $\mathcal{S}$ and $\mathcal{S}^*$ in Theorem 3). Thus, the current knowledge of the approximate optimal policy $\hat{\pi}^*$ tends to favour sampling along the state-action pairs which belong to $\hat{\pi}^*$, and therefore can be seen as a heuristic that accelerates the convergence.

Then, our main experimental result is the following: our algorithms with score $\mathcal{S}$ lead to a clear quicker decrease of C^* than both standard approaches for all problems (see figures 1-a, 1-b, 2-a, 2-b). If we consider the error measure C, our algorithms lead to a clear improvement in all cases except figure 2-b' where there does not seem to be a significant improvement, although there is actually one (figures 2-b and 2-b' correspond to two measures of the same experiments and the improvement is clear for C^* in figure 2-b.).

More general observations can be derived from all the experiments we have run and whose results are not shown in this paper. The convergence acceleration is always smoother and easier to notice for the relative distance between the approximate value function and the real optimal value function (C^*) than for the relative distance between the real value of the approximate optimal policy $\hat{\pi}^*$ and the optimal value (C). The reason for C to show less clear improvement is probably related

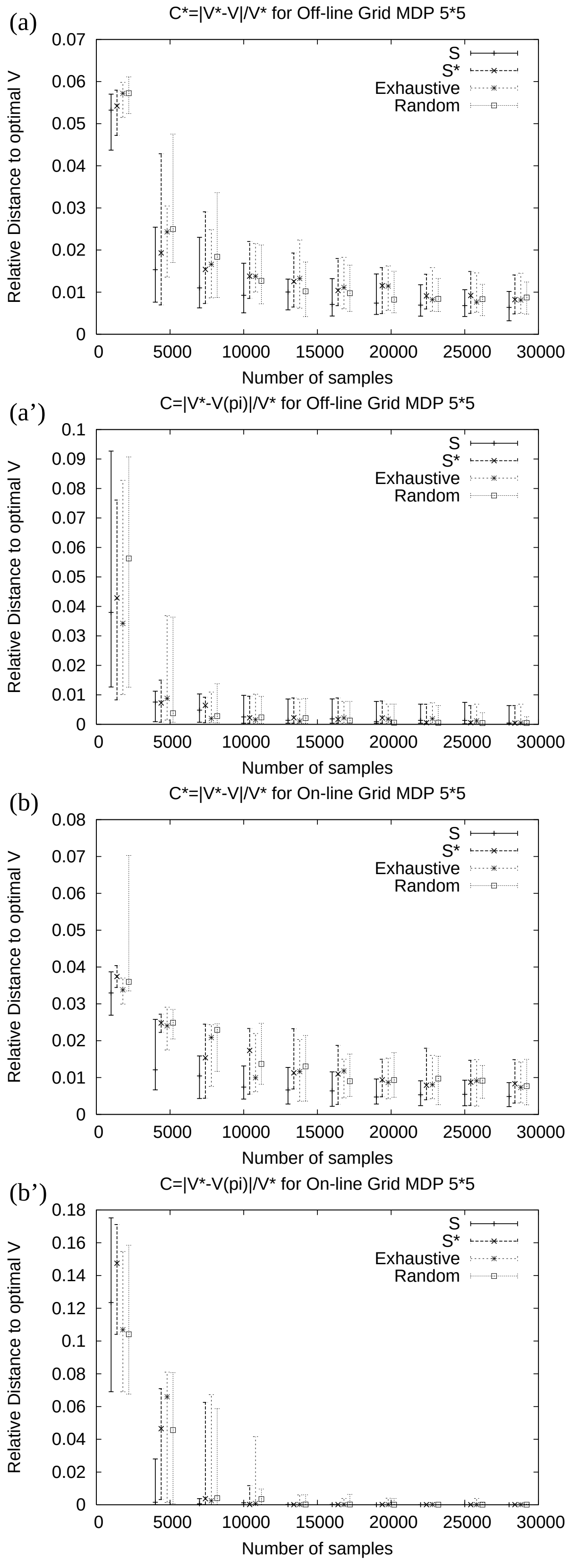

Figure 1: Error measure evolution for Random Grid MDP simulations: a (resp. a') shows C^* (resp. C) for the off-line algorithm; b (resp. b') shows C^* (resp. C) for the on-line algorithm.

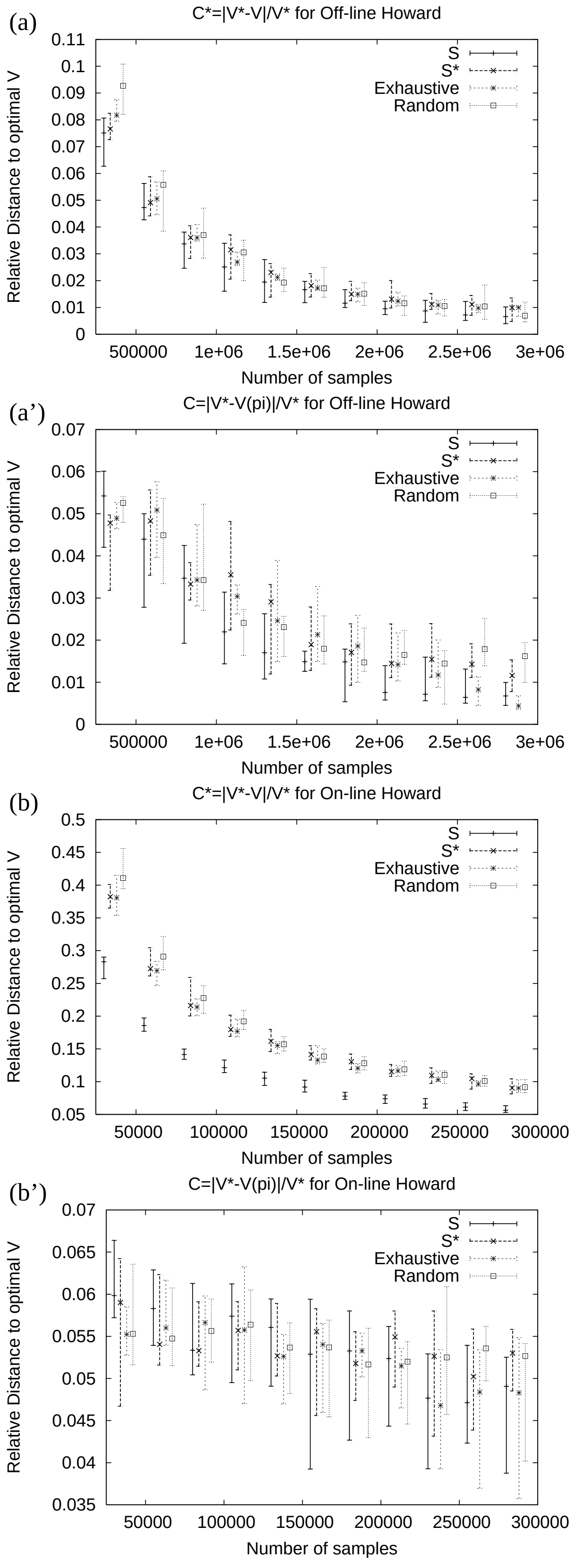

Figure 2: Error measure evolution for Howard's automobile replacement problem simulations: a (resp. a') shows C^* (resp. C) for the off-line algorithm; b (resp. b') shows C^* (resp. C) for the on-line algorithm.

to its more general complex shape: C only changes by discontinuous jumps each time a new sample leads to a change of the approximate policy $\hat{\pi}^*$. Another general observation is that the efficiency for decreasing C^* by our algorithms with score $\mathcal{S}$ is usually more striking for the *on-line* reinforcement learning problem than for the *off-line* case (compare for instance 2-a and 2-b). This is particularly interesting, as it is likely that problems for which samples have a high cost are also *on-line* problems.

6 Discussion

We showed through simulations in the previous section that our algorithms can in practice speed-up the convergemce towards the optimal value function and the optimal policy. We here discuss our contributions to the literature.

This work can be seen as an extension of the L_1 norm error analysis of (Munos and Moore, 2000) referred to in this paper as Theorem 1, to the reinforcement learning problem. A key issue in such an approach is to estimate in a sound way the 1-step error which depends upon unknown parameters (B_a and V^*). The work we have presented in this paper shows that, for the reinforcement learning problem, standard tools of the statistical theory allow us to derive an upper bound of this 1-step error which is true with high probability, and we can consequently exploit the strength of the L_1 norm error analysis as explained in (Munos and Moore, 2000).

The idea of optimizing the choices of samples based on a gradient descent of the error on the value function was also suggested in (Munos, 2001). However the approach of the author for analyzing the error is quite different; from our viewpoint, the most important difference is that the author does not provide any confidence bounds on the approximate policy value, or even on the approximate value, which might be crucial in practice for deciding when it is reasonable to stop sampling. Furthermore, the author does not provide any specific algorithm for the reinforcement learning problem nor does show any empirical evaluation. If given a gradient analysis of the error given the sampling, the algorithm for sampling in the off-line case can be derived in a rather straight-forward way, the on-line case algorithm, which shows the most interesting improvements in our simulations, and the very idea of introducing an Error MDP, are completely new contributions.

A closely related work is (Even-Dar *et al.*, 2003), where the exploration-exploitation dilemma is tackled with confidence bounds on the Q-values and an action elimination procedure which progressively restricts the set of possible actions to sample in each state. Using such an action elimination procedure (and even though their algorithm samples uniformly in the actions that have not been eliminated yet), the authors show that an eventual non uniform sampling strategy is better than the random sampling strategy. As we also derive confidence bounds on the policy, we could have incorporated an action elimination procedure in our algorithm. We decided not to do so. Indeed, our simulations show that just our gradient-based approach can be efficient for reducing the number of samples. A natural subsequent work will evaluate the combination of action elimination with our gradient approach in the minimization of sampling we considered in our paper, and in the exploration-exploitation dilemma.

From a computational viewpoint, our algorithms (especially the on-line algorithm) require, for choosing every next sample, to solve a couple of MDPs and to invert a couple of linear systems. We already argued that it would be, in practice, quite efficient to use the solutions of the Error MDP at time t, as a starting point for finding the solutions at time $t+1$. Considering this complexity, we would like to stress here the fact that there is a potential leverage on what we might call a complexity-quality trade-off. Because all the computations we consider (for the MDPs as for the linear system inversions) are contraction mapping fixed points, it is straightforward to imagine variations of these computations, where one just applies a small number of (possibly asynchronous) back-ups of the contraction operators and use the corresponding approximate solutions for choosing the samples. This leverage principle for contraction mapping was used many times in the reinforcement learning literature (for instance, one can think of the DYNA architecture (Sutton, 1991) as an asynchronous version of the natural certainty equivalence method introduced in section 2). Here again for the current paper, we decided not to over-complexify the analysis with such details so that the theoretical justifications of our algorithms remain clear, and also so that we could analyze the very process of doing sampling based on a sensitivity analysis without perturbing it with some other approximation issues.

Conclusion

In this paper, we provided a sensitivity analysis and two new algorithms which enable us to reduce the amount of sampling in reinforcement learning. Simulations show that our algorithms provide a quicker convergence (in the sense of the number of samples) of the value function to the real optimal value function than standard approaches. In the near future, we will investigate incorporating action elimination and using lower complexity variations of the algorithms presented in this paper. We will also relate our work to the well-known exploration-exploitation dilemma. Further future directions include deriving model-free versions of our algorithms and combining them with function approximation in a *neurodynamic way* (Bertsekas and Tsitsiklis, 1996) for tackling large state space problems.

References

Bernstein, D., Zilberstein, S., Washington, R., and Bresina, J. (2001). Planetary rover control as a markov decision process. In *AAAI Spring Symposium: Game Theoretic and Decision Theoretic Agents*.

Bertsekas, D. and Tsitsiklis, J. (1996). *Neurodynamic Programming*. Athena Scientific.

Even-Dar, E., Mannor, S., and Mansour, Y. (2003). Action elimination and stopping conditions for reinforcement learning. In *ICML*, pages 162–169.

Hoeffding, W. (1963). Probability inequalities for sums of bounded random variables. *American Statistical Association Journal*, **58**, 13–30.

Howard, R. (1960). *Dynamic programming and Markov processes*. MIT press.

Kearns, M. and Koller, D. (1999). Efficient reinforcement learning in factored MDPs. In *Seventeenth International Joint Conference on Artificial Intelligence*.

Kearns, M. and Singh, S. (1998). Near-optimal reinforcement learning in polynomial time. In *Proc. 15th International Conf. on Machine Learning*, pages 260–268. Morgan Kaufmann, San Francisco, CA.

Kumar, P. (1985). A survey of some results in stochastic adaptive control. *SIAM Journal of Control and Optimization*, **23**, 329–38.

Kumar, P. and Varaiya, P. (1986). *Stochastic Systems: Estimation, Identification, and Adaptive Control*. Prentice Hall, Englewood Cliffs, New Jersey.

Munos, R. (2001). Efficient resources allocation for markov decision processes. In *NIPS*, pages 1571–1578.

Munos, R. and Moore, A. (2000). Rates of convergence for variable resolution schemes in optimal control. In *International Conference on Machine Learning*.

Puterman, M. (1994). *Markov Decision Processes*. Wiley, New York.

Sutton, R. (1991). DYNA, an Integrated Architecture for Learning, Planning and Reacting. In *Working Notes of the AAAI Spring Symposium on Integrated Intelligent Architectures*.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning, An introduction*. BradFord Book. The MIT Press.

Tesauro, G. (1995). Temporal difference learning and TD-Gammon. *Communications of the ACM*, **38**(3), 58–68.

Appendix

A. Proof of Theorem 2

Let us define $\Delta R(s,a) \overset{def}{=} |R(s,a) - \hat{R}(s,a)|$ and $\Delta T(s,a,s') \overset{def}{=} |T(s,a,s') - \hat{T}(s,a,s')|$. Hoeffding's inequality (Hoeffding, 1963) gives:

$$\mathbb{P}\left(\left\{\frac{\Delta R(s,a)}{R_{max}} \geq \frac{\mu}{\sqrt{\#(s,a)}}\right\}\right) \leq 2e^{-2\left(\frac{\mu}{\sqrt{\#(s,a)}}\right)^2 \#(s,a)}.$$

Given the definition of μ, we can infer:

$$\mathbb{P}\left(\left\{\frac{\Delta R(s,a)}{R_{max}} \geq \frac{\mu}{\sqrt{\#(s,a)}}\right\}\right) \leq \frac{\delta}{|S||A|(|S|+1)}.$$

Similarly for T, we can write:

$$\mathbb{P}\left(\left\{\Delta T(s,a,s') \geq \frac{\mu}{\sqrt{\#(s,a)}}\right\}\right) = \frac{\delta}{|S||A|(|S|+1)}.$$

Using the union bound we can deduce:

$$\mathbb{P}\left(\left\{\exists (s,a), \frac{\Delta R(s,a)}{R_{max}} \geq \frac{\mu}{\sqrt{\#(s,a)}}\right\} \cup \left\{\exists (s,a,s'), \Delta T(s,a,s') \geq \frac{\mu}{\sqrt{\#(s,a)}}\right\}\right) \leq \delta.$$

Then, using the triangle inequality, we have with probability at least $1-\delta$, for all (s,a):

$$\Delta R(s,a) + \frac{\gamma R_{max}}{1-\gamma}\sum_{s'} \Delta T(s,a,s') \leq \frac{\mu R_{max}}{\sqrt{\#(s,a)}} + \frac{\gamma R_{max}}{1-\gamma}|S|\frac{\mu}{\sqrt{\#(s,a)}} = \frac{\lambda\mu}{\sqrt{\#(s,a)}}.$$

Eventually, the theorem follows from the triangle inequality which allows to say that for all (s,a):

$$\left|[B_a V^*](s) - [\hat{B}_a V^*](s)\right| \leq \Delta R(s,a) + \frac{\gamma R_{max}}{1-\gamma}\sum_{s'} \Delta T(s,a,s'). \qquad \square$$

B. Proof of Theorem 3

Let us first concentrate on E^*. We use a Taylor development to prove Theorem 3:

$$\Delta\|E^*\|_1 = \frac{\partial\|E^*\|_1}{\partial e(s,a)}\Delta e(s,a) + o(\Delta e(s,a)).$$

where $\Delta e(s,a)$ is the variation of $e(s,a)$ if the agent samples (s,a). We have:

$$\Delta e(s,a) = \frac{\lambda\mu}{\sqrt{\#(s,a)+1}} - \frac{\lambda\mu}{\sqrt{\#(s,a)}} = \lambda\mu f(\#(s,a)).$$

Theorem 1 gives the value of $\frac{\partial\|E^*\|_1}{\partial e(s,a)}$.

The proof is similar for E. $\square$