Exploiting Structure and Uncertainty of Bellman Updates in Markov Decision Processes

Davide Tateo, Carlo D'Eramo, Alessandro Nuara, Marcello Restelli, Andrea Bonarini
Department of Electronics, Information and Bioengineering
Politecnico Di Milano, Milano, Italy
Email: {davide.tateo, carlo.deramo, alessandro.nuara, marcello.restelli, andrea.bonarini}@polimi.it

***Abstract*—In many real-world problems stochasticity is a critical issue for the learning process. The sources of stochasticity come from the transition model, the explorative component of the policy or, even worse, from noisy observations of the reward function. For a finite number of samples, traditional Reinforcement Learning (RL) methods provide biased estimates of the action-value function leading to poor estimates and propagating them by the application of the Bellman operator. While some approaches rely on the fact that the estimation bias is the key problem in the learning process, we show that in some cases this assumption does not necessarily hold. We propose a method that exploits the structure of the Bellman update and the uncertainty of the estimation in order to better use the amount of information provided by the samples. We show theoretical considerations about this method and its relation w.r.t. Q-Learning. Moreover, we test it in environments available in literature in order to demonstrate its effectiveness against other algorithms that focus on bias and sample-efficiency.**

I. Introduction

It is well known that a key issue of Reinforcement Learning (RL) problems is the accuracy of the estimation of the action-value with a limited number of samples. While most algorithms guarantee the convergence of the estimates to the optimal action-value, in practice the presence of stochastic components lead to slow learning. In fact, the majority of real-world problems have significant sources of stochasticity: the environment could have stochastic transition and this complicates the estimation of the effectiveness of an action; most of the times it is necessary to use stochastic policies to guarantee that all states are visited infinitely many times, that is required to guarantee the convergence of the algorithm; the policy could change during the learning process resulting in very different behaviors; the reward function is often corrupted by noisy observations and, in other cases, the reward function is stochastic itself. Moreover, it usually happens that some deterministic environments are partially observable and, thus, are perceived by the agent as stochastic decision processes (e.g. Blackjack).

Since Monte-Carlo estimates of action-values are affected by high variance of the returns, the most successful RL algorithms are based on bootstrapping (e.g. Q-Learning [1]), that trades off the variance of the estimation with a consistent but biased estimator. However, with a finite number of samples, the bias of the estimation could be significantly relevant when propagating the action-values to the next state and, recursively, to all the other states. Recent works tried to deal with this issue, in particular focusing on the estimation of the maximum expected value. It is well known [2], [3] that the maximum operator (used in the Q-Learning update equation) is positively biased, thus it overestimates the optimal action-value. In highly stochastic environments, this overestimation leads to unstable learning and poor convergence rates. In order to avoid this issue, the Double Q-Learning algorithm [4] has been proposed. This algorithm uses the double estimator [5] to compute the maximum action-value to calculate the temporal-difference error: providing a negatively biased estimation (i.e. underestimating) of the maximum action-value, this approach can improve the performance especially in noisy environments. Another recently proposed approach is the Weighted Q-Learning [6] that computes a weighted average of the action-value functions estimates balancing between underestimation and overestimation.

However, an inaccurate estimation of the action value function does not always imply bad performance; indeed most of the policies are not dependent on the accuracy of the action-values, but instead they rely on their ordering. For instance, in the empirical section of this paper, we show how Speedy Q-Learning algorithm [7] reaches very good performance despite sometimes estimating the action-values very poorly. Starting from this considerations, we try to address this problem from another point of view. Indeed, we do not focus on the estimation of the maximum expected value, but we care about weighting the new samples according to the uncertainty of the current estimates. In fact, we propose this work starting from the consideration that it is not sufficient to accurately estimate the maximum expected value of the Q-function of the next state, but we also need to consider how these information are propagated to other states.

The interesting aspect of this approach is that it is possible to use any maximum expected value estimator and, moreover, it can be easily extended in an on-policy scenario. Since we believe that the choice of the estimator is not a relevant issue in our approach, we choose the maximum operator as it is the simplest one.

II. Preliminaries

A Markov Decision Process (MDP) is defined by $\mathcal{M} = \langle \mathcal{X}, \mathcal{U}, \mathcal{P}, \gamma, \mathcal{R}, D \rangle$, where $\mathcal{X}$ denotes the state space, $\mathcal{U}$ denotes the action space, $\mathcal{P}$ is a Markovian transition model where $\mathcal{P}(x'|x,u)$ defines the transition density between state

x and x' under action u, $\gamma \in [0,1)$ is the discount factor, $\mathcal{R}(r|x,u,x')$ defines the distribution of the reward, and D is the distribution of the initial states. A stochastic policy is a density distribution $\pi(\cdot|x)$ specifying the density distribution over the action space $\mathcal{U}$ for each state x.

Given a policy π, we define the action-value function as:

$$Q^{\pi}(x,u) = \mathop{\mathbb{E}}_{\substack{(x',u')\sim \\ \mathcal{P}(x'|x,u)\pi(u'|x')}} [r(x,u,x') + \gamma Q^{\pi}(x,u)(x',u')] . \quad (1)$$

The optimal policy $\pi*$ is the policy with the highest expected return in the MDP and the optimal action-value function $Q^*(x,u)$ is the action-value function of the optimal policy. It has been shown that the optimal policy in any MDP can be always a deterministic policy that at each state chooses the action with the highest $Q^*(x,u)$ value. Given that we can always write the optimal action-value function as:

$$Q^*(x,u) = \mathop{\mathbb{E}}_{x'\sim\mathcal{P}(x'|x,u)} \left[r(x,u,x') + \gamma \max_{u'} Q^*(x',u') \right] . \quad (2)$$

As the expected value is a linear operator, we can always write (2) as:

$$Q^*(x,u) = \mathop{\mathbb{E}}_{x'\sim\mathcal{P}(x'|x,u)} [r(x,u,x')] + \gamma \mathop{\mathbb{E}}_{x'\sim\mathcal{P}(x'|x,u)} \left[\max_{u'} Q^*(x',u') \right] . \quad (3)$$

We now introduce two functions, $\widetilde{R}$ and $\widetilde{Q}$, defined as:

$$\widetilde{R}(x,u) = \mathop{\mathbb{E}}_{x'\sim\mathcal{P}(x'|x,u)} [r(x,u,x')] ,$$
$$\widetilde{Q}(x,u) = \mathop{\mathbb{E}}_{x'\sim\mathcal{P}(x'|x,u)} \left[\max_{u'} Q^*(x',u') \right] . \quad (4)$$

We can give an interpretation of these two functions. $\widetilde{R}(x,u)$ is the expected immediate reward of the action u in the state x. $\widetilde{Q}(x,u)$ is the expected discounted return of the states reached after performing action u in state x, i.e. the expected gain of the reached state.

We can now write the optimal value function as:

$$Q^*(x,u) = \widetilde{R}(x,u) + \gamma \widetilde{Q}(x,u). \quad (5)$$

Our approach shifts the focus of the RL task from finding a good estimator for the optimal action-value function, to the task of finding good estimators for the $\widetilde{R}$ and $\widetilde{Q}$ functions. The main motivation is that the sources of uncertainty of the two components of the action-value function are different: the $\widetilde{R}$ function only depends on the transition and reward models, while $\widetilde{Q}$ also depends on the optimal policy.

III. The Proposed Method

In the following section we derive our method from (5). We propose the general schema, using the maximum estimator and we show the relations with the standard Q learning update. We call this method *RQ-Learning* as it decomposes the TD-Error in a reward component and an action-value component.

A. Decomposition of the TD Error

Standard Q-Learning algorithm computes the temporal difference error given the tuple (x,u,r,x') w.r.t. the current action-value estimates, and then updates such estimate proportionally to the error. The amount of correction in the direction of the new sample is measured by the learning rate: if the learning rate equals to 1, the new sample substitutes the old estimate; if the learning rate equals to 0, the new sample is discarded, and the old estimate is kept unchanged. As shown in (5) the action-value function could be decomposed in two different components. Our method is based on the idea to give separate estimates for these two components, instead of computing a TD error, we compute the error w.r.t. each component of the action-value function:

$$\widetilde{R}_{t+1}(x,u) \leftarrow \widetilde{R}_t(x,u) + \alpha_t(R(x,u,x') - \widetilde{R}_t(x,u)), \quad (6)$$
$$\widetilde{Q}_{t+1}(x,u) \leftarrow \widetilde{Q}_t(x,u) + \beta_t(\max_{u'} Q_t(x',u') - \widetilde{Q}_t(x,u)). \quad (7)$$

Separating the two components of the value function can be useful, as the two components have inherently different sources of stochasticity. Moreover, the information stored in $\widetilde{R}$ is local to each state-action pair, and does not contain the uncertainty of the estimation of others states. The information stored in $\widetilde{Q}$ instead depends only on the action-value function of the states that could be reached after performing the action u in the state x, which depends, recursively, on the other action-value functions. It is clear that the propagation of uncertain values only affects the $\widetilde{Q}$ component. As the actual action value function is the sum of the two estimates, we can write an equivalent update for the Q function:

$$\begin{aligned} Q_{t+1}(x,u) \leftarrow & \widetilde{R}_t(x,u) + \alpha_t(R(x,u,x') - \widetilde{R}_t(x,u)) \\ & + \gamma \left(\widetilde{Q}_t(x,u) + \beta_t(\max_{u'} Q_t(x',u') - \widetilde{Q}_t(x,u)) \right) \\ = & Q_t(x,u) + \alpha_t(R(x,u,x') - \widetilde{R}_t(x,u)) \\ & + \gamma\beta_t(\max_{u'} Q_t(x',u') - \widetilde{Q}_t(x,u)) \end{aligned} \quad (8)$$

notice that this update cannot be used in practice in the algorithm, as it is not keeping the current values of the single components. However (8) is useful to analyze the relations to standard Q-Learning algorithm.

B. Analysis of the Decomposed Update

We will discuss the relationship of our method with standard temporal difference methods. Let t be the learning step. As a first step of our analysis we can consider the simplest case $\alpha_t = \beta_t$, $\forall t$ by combining (8) and (5) we obtain:

$$\begin{aligned} Q_{t+1}(x,u) \leftarrow & Q_t(x,u) + \alpha_t(R(x,u,x') + \gamma \max_{u'} Q_t(x',u') \\ & - Q_t(x,u)) \end{aligned} \quad (9)$$

That is the classical Q-Learning update.

We consider now the setting $\alpha_t \geq \beta_t > 0$, $\forall t$. Let $\beta_t = \delta_t \alpha_t$, we obtain:

$$\begin{aligned} Q_{t+1}(x,u) \leftarrow & Q_t(x,u) + \alpha_t(R(x,u,x') + \gamma\delta_t \max_{u'} Q_t(x',u') \\ & - (\widetilde{R}_t(x,u) + \gamma\delta_t \widetilde{Q}_t(x,u))) \\ = & Q_t(x,u) + \alpha_t(R(x,u,x') + \gamma'_t \max_{u'} Q_t(x',u') \\ & - (\widetilde{R}_t(x,u) + \gamma'_t \widetilde{Q}_t(x,u))) \\ = & Q_t(x,u) + \alpha_t((R(x,u,x') + \gamma'_t \max_{u'} Q_t(x',u')) \\ & - Q'_t(x,u)) \end{aligned} \tag{10}$$

with $\gamma'_t = \gamma\delta_t$. Notice that $Q'_t(x,u)$ is the current Q function, but computed with a different discount factor. If the condition above is satisfied, then we can see our method as a variable discount factor learning. If we consider δ_t that increases monotonically in the interval $[0,1]$, our method works increasing the effective horizon each step, starting from trying to solve a greedy myopic problem and moving towards the real one. This approach has been used in practice to solve infinite horizon problems when the discount factor is close to 1 [8], [9], [10].

Finally, we can observe that if the reward function and the transition model are deterministic, we can use $\alpha = 1$ and consider only $\beta_t = \delta_t$.

C. Variance dependent learning rate

To improve performance on the estimation, we would like to weight the error of each sample w.r.t. the current estimate depending on how much we are sure about the current value of our estimate. Thus, we propose a learning rate that depends on the variance of the current variable estimate.

First of all we need to compute the variance of each estimator. To perform such computation we have to make the assumption that the learning rates are independent from the data. Of course, we violate this assumption, but it is needed in order to have a closed form for the variance of the estimator and works well in practice. We will suppose also that the samples X_i are i.i.d., with mean μ and variance σ^2. Consider the general form of the estimator:

$$\widetilde{X}_{n+1} = (1-\alpha_t)\widetilde{X}_n + \alpha_t X_n \tag{11}$$

We now compute the expected value and the variance of this estimator:

$$\mathbb{E}\left[\widetilde{X}_{n+1}\right] = \mu \sum_i^n \alpha_i \prod_{j=i+1}^n (1-\alpha_j) \tag{12}$$

$$\mathrm{Var}\left[\widetilde{X}_{n+1}\right] = \sigma^2 \sum_i^n \alpha_i^2 \prod_{j=i+1}^n (1-\alpha_j)^2 = \sigma^2 \omega \tag{13}$$

with $\omega = \sum_i^n \alpha_i^2 \prod_{j=i+1}^n (1-\alpha_j)^2$. Notice also that we can use the sample covariance S_{n-1} of the random variable X to estimate the real covariance σ^2. It is possible to compute in an incremental way both the sample covariance, in the traditional way, and ω:

$$\omega_{n+1} = (1-\alpha_n)^2 \omega_n + \alpha_n^2. \tag{14}$$

A weak assumption of this model is that the variables are identically distributed. While this should be true for the reward function, if the MDP is stationary, this is not true for the Q function values whose distribution is affected by the policy and by the other states current estimates. However, a good approximation, could be to consider data collected in a temporal window in which the distribution is approximately stationary: using such approach, we can compute the variance of the process in a given time window forgetting old values that can lead to a biased estimation of the current window variance. While this approach is not formally correct, as the derivation of the variance estimates makes the assumption of i.i.d. variables, this approximation leads to very good results in practice as we show in the empirical Section IV.

Finally, we can choose a learning rate that depends on the covariance. Let $\sigma_e^2(t)$ be an estimate of $\mathrm{Var}\left[\widetilde{X}_t\right]$. We propose the following learning rate for each component of the action-value function:

$$\alpha_t = \frac{\sigma_e^2(t)}{\sigma_e^2(t) + \eta} \tag{15}$$

where η is the amount of the estimator variance for which the learning rate is 0.5 (i.e. when $\sigma_e = \eta$ the learning rate is 0.5). It can be seen as a soft threshold to tune the speed of the decrease of the learning rate w.r.t. the estimator variance.

If we consider the case $\beta_t = \alpha_t \delta_t$, then we have to use a different learning rate. As we want an increase of the discount factor faster than the decrease of the general learning rate, we can use an exponentially increasing learning rate for the delta parameter:

$$\delta(t) = e^{\frac{\sigma_e^2}{\eta} \log \frac{1}{2}} \tag{16}$$

where η has the same interpretation of the previous scenario thanks to the $\log \frac{1}{2}$ factor.

D. Discussion on convergence

Convergence of the Q-Learning algorithm is guaranteed under some conditions, including some properties on the learning rates [11], [1]:

$$\begin{aligned} & 0 \leq \alpha < 1, \\ & \lim_{N\to\infty} \sum_{t=0}^N \alpha_t = \infty, \\ & \lim_{N\to\infty} \sum_{t=0}^N \alpha_t^2 < \infty. \end{aligned} \tag{17}$$

In order to give some preliminary results about the convergence of RQ-Learning and to motivate the choice of the formulas of learning rates α and δ, we show that, under some assumptions, the proposed formulas (15) (16) satisfy (17).

Suppose that the variance of the estimator $\sigma_e^2(t)$ is the variance of the sample mean. It is well known, by the central limit theorem, that the variance of the sample mean is $\sigma_\mu(t) = \frac{\sigma^2}{t}$, where σ is the variance of the process that generates the samples. This assumption does not hold in general as we can see from the formula of the learning rate, however in some cases this could be a good approximation as the sample mean is a special case of the estimator.

Now, just consider the proposed learning rate in (15), replacing the variance of the sample mean into the variance of the estimator:

$$\alpha_t = \frac{\sigma^2}{\sigma^2 + \eta t} \tag{18}$$

it can be easily shown that:

$$\lim_{N\to\infty} \sum_{t=0}^{N} \frac{\sigma^2}{\sigma^2 + \eta t} = \infty \tag{19}$$

$$\lim_{N\to\infty} \sum_{t=0}^{N} \left(\frac{\sigma^2}{\sigma^2 + \eta t}\right)^2 < \infty. \tag{20}$$

Now, consider $\beta_t = \alpha_t \delta_t$. In this scenario we propose an exponential learning rate. Let $\alpha_t = \frac{1}{t}$ for simplicity and consider the learning rate (16) replacing the sample mean into the variance of the estimator:

$$\beta(t) = \frac{1}{t} e^{\frac{\sigma^2}{\eta t} \log \frac{1}{2}}. \tag{21}$$

Then:

$$\lim_{N\to\infty} \sum_{t=0}^{N} \frac{1}{t} e^{\frac{\sigma^2}{\eta t} \log \frac{1}{2}} = \infty \tag{22}$$

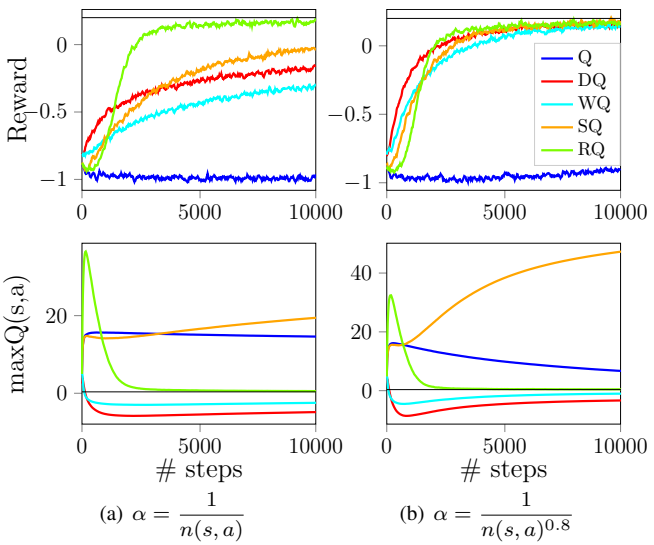
$$\lim_{N\to\infty} \sum_{t=0}^{N} \left(\frac{1}{t} e^{\frac{\sigma^2}{\eta t} \log \frac{1}{2}}\right)^2 < \infty. \tag{23}$$

This analysis considers only the estimation of the variance of the sample mean. The analysis of the generic variance estimator is more complex, but we conjecture that convergence could be guaranteed under some mild conditions on η and σ.

IV. Experimental Results

In this section, we highlight the main advantages of exploiting the structure of the TD-Error with RQ-Learning in three discrete MDPs. We compare RQ-Learning against Q-Learning [1], Double Q-Learning [4], Weighted Q-Learning [6] and Speedy Q-Learning[1] [7]. We choose this set of algorithms because we want to analyze the impact of the maximum expected value estimator in the learning process. Indeed, while [4] strongly suggests that a negatively biased estimator should be used to deal with stochastic MDPs, the following empirical results will show that positively biased estimators are able to achieve better performance in highly stochastic problems as well. We want to show that instead the main point consists in exploiting data in the best way, in particular in the estimation of the action-value, giving higher relevancy to more recent samples. We conjecture that the trade-off between keeping the old estimate and updating it with the new one is the key issue in highly stochastic environments. Both RQ-Learning and Speedy Q-Learning exploit this idea, in particular the latter uses an increasing learning rate on the difference between the target computed with the current estimation and the one computed with the previous estimation; on the other hand RQ-Learning weights the update considering the uncertainty of the current estimation.

[1]In these experiments we consider the asynchronous version of Speedy Q-Learning.



(a) $\alpha = \frac{1}{n(s,a)}$ (b) $\alpha = \frac{1}{n(s,a)^{0.8}}$

Fig. 1. Mean reward per step (top) and maximum action-value estimate in the initial state (bottom) of all the other algorithms and of the best setting of RQ-Learning for this experiment. Results are averaged over 10000 experiments.

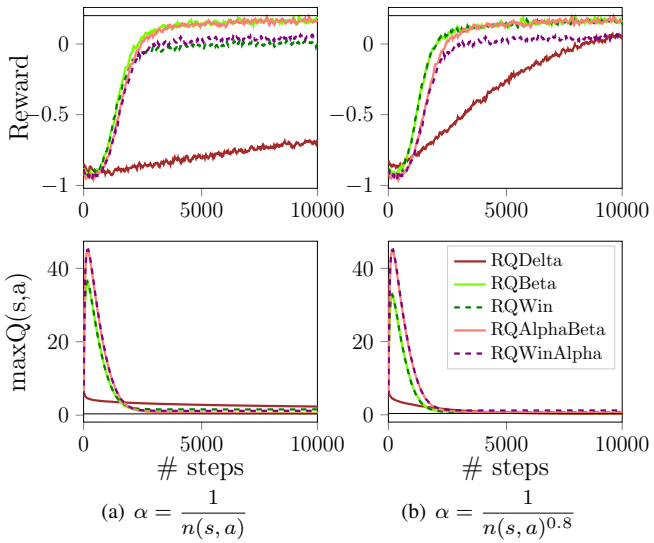
We analyze the performance of RQ-Learning with different choices of learning rates. All the other algorithms use a decaying learning rate $\alpha(s,a) = \frac{1}{n(s,a)^k}$ where $n(s,a)$ and k are respectively the number of updates for each action a in state s and a coefficient to tune the rate of decay.[2]

A. Noisy Grid World

This environment is proposed in [4] and consists in a 3×3 grid with the initial position in the lower-left cell and the goal state in the upper-right cell. Each action performed in a non-goal state obtains a reward -12 and 10 with equal probability. In the goal state, every action obtains a reward of 5 and terminates the episode. The discount factor is $\gamma = 0.95$. The policy is ε-greedy with $\varepsilon = \frac{1}{\sqrt{n(s)}}$, where $n(s)$ is the number of visits of the state s. The optimal average reward per step is 0.2 and the maximum action-value function of the initial state is $5\gamma^4 - \sum_{k=0}^{3} \gamma^k \approx 0.36$.

Figure 1 shows the mean reward per step and the approximation of the maximum action-value in the initial state

[2]Assuming that Double Q-Learning splits the action-value table in a table A and a table B, also the learning rate are split in $\alpha_A(s,a) = \frac{1}{n_A(s,a)^k}$ and $\alpha_B(s,a) = \frac{1}{n_B(s,a)^k}$ where $n_A(s,a)$ and $n_B(s,a)$ are the number of updates for each action a in state s, respectively in table A and table B.



(a) $\alpha = \frac{1}{n(s,a)}$ (b) $\alpha = \frac{1}{n(s,a)^{0.8}}$

Fig. 2. Mean reward per step (top) and maximum action-value estimate in the initial state (bottom) of the best setting of RQ-Learning for this experiment together with other less effective setting of RQ-Learning. Results are averaged over 10000 experiments.

computed by the other algorithms and RQ-Learning with α as used by the other algorithms, but with the separated variance-dependent learning rate β for the action-value estimate using $\eta = 1$, that is the most effective setting of RQ-Learning that we tried for this problem. Note how the performance of RQ-Learning w.r.t. reward are the best one (except for $k = 0.8$ where all algorithms perform similarly) and, moreover, how the estimate of the action-value is also the best one. Notice that, with $k = 1$, Speedy Q-Learning outperforms both Double Q-Learning and Weighted Q-Learning w.r.t. the mean reward per step, even with a diverging estimate of the action-value function. This is an empirical evidence of our conjecture on the non-correlation of the bias of the estimation, obtained in the experiment proposed in [4] and [6]. Moreover, as expected, the performance of RQ-Learning is not affected to the exponent used in the learning rate. The other algorithms achieve the optimal performance only in the setting with the higher learning rate, confirming the advantage of giving more importance to newer samples.

In Figure 2 we compare different variants of RQ-Learning: "RQBeta" is the same configuration used in Figure 1; "RQDelta" uses $\beta = \alpha\delta$ with $\eta = 1$; "RQWin" uses a windowed estimation of variance with a window of length 50 and $\eta = 0.5$. "RQAlphaBeta" uses a variance-dependent learning rate also for α with $\eta = 100$ and β with $\eta = 1$; "RQWinAlpha" is the same configuration of the previous one, but uses a windowed β with $\eta = 0.5$. Note that η has a larger value in configurations without windowed variance estimation because such configurations are likely to overestimate the current variance of the process. "RQDelta" configurations result in a cautious learning that leads to very slow improvements, but avoids the overestimation of the action-value slowly converging to the optimal value. While "RQDelta" performance are not comparable with other configurations of RQ-Learning, it still outperforms Q-Learning. The other configurations perform similarly to the best one.

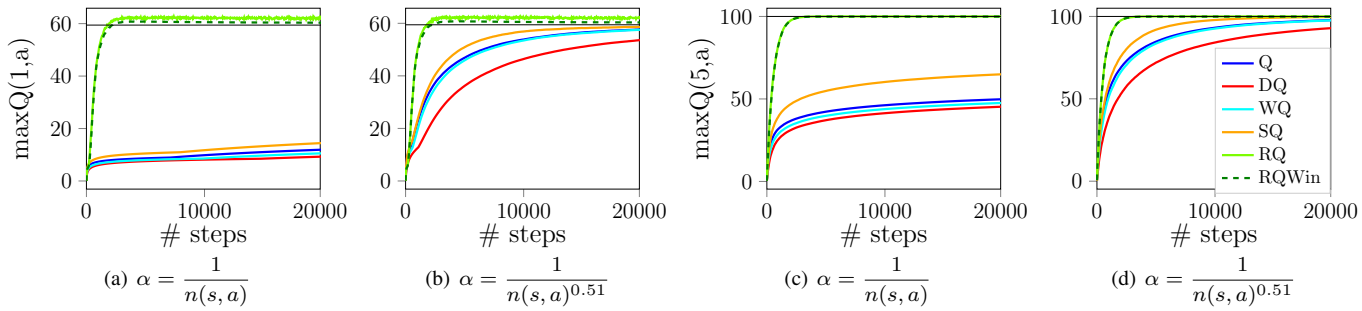
B. Double Chain

This is a problem proposed in [12] which consists in a Markov chain with two branches (Figure 6). In state 1, action 1 yields a reward of 0 and moves the agent in state 2; action 2 yields a reward of 2 and moves the agent in state 6. In all other states, action 2 moves the agent in state 1 and returns a reward of 2; action 1 moves the agent in the next state of the chain returning a reward of 0. In state 5 and 9, action 1 yields a reward of respectively 10 and 5. In all states, each action has a probability of success of 0.8 and, if the action fails, the agent remains in the current state and yields a reward of 0. The discount factor is $\gamma = 0.9$. The optimal policy is to take action 1 in state from 1 to 5 and action 2 in the other states. RQ-Learning uses $\eta = 10$. In this experiment we focus on the estimation of the action-value function, therefore we use a fully random policy to explore the environment.

Figure 3 shows the estimate of the maximum action-value in state 1 and 5. State 5 is the state with the highest maximum action-value. RQ-Learning approaches the optimal value faster than the other algorithms in both configurations. However, in state 1, only RQ-Learning with windowed variance estimation converges to the optimal value because the non-windowed approach suffers from variance overestimation due to the fact that the distribution of the next action-values changes during learning; this issue, together with the stochasticity of the transitions, causes the oscillation of the estimate and slow convergence rate. This behavior is highlighted in Figure 4 where we show the learning rates of the action-value in the considered states. While initially the learning rates are similar, the windowed learning rate converges to 0, instead in the non-windowed case the learning rates decrease slowly. Note that in state 5 the learning rate of action 2 is almost stationary because of the complexity of the double chain structure. In this cases, increasing η can be helpful to speedup the decreasing of the learning rate.

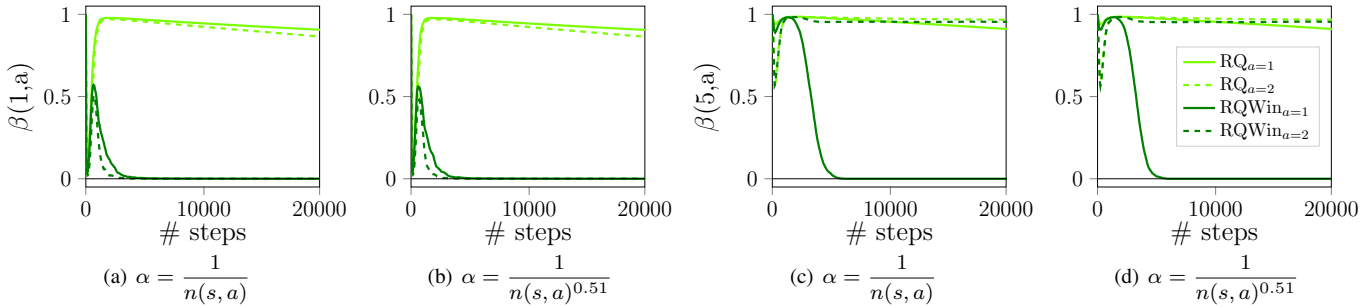
In this experiment, RQ-Learning does not only approximate the value function very well, but it is also able to converge to the optimal policy faster than the other algorithms. Figure 5 shows a comparison between Q-Learning and windowed RQ-Learning. We do not show performance of the other algorithms since they behave similarly to Q-Learning. Notice that using $k = 1$ Q-Learning (and the other algorithms, except from windowed RQ-Learning) is not able to converge to the optimal policy in state 9. Indeed, the value of action 2 in state 9 is the most difficult to estimate, considering the structure of the MDP.

Note that in this problem, where the only source of stochasticity is in the transition function, Double Q-Learning suffers the most. On the other hand Speedy Q-Learning is still the best approach compared with the others. This empirical result confirm our conjecture described above.



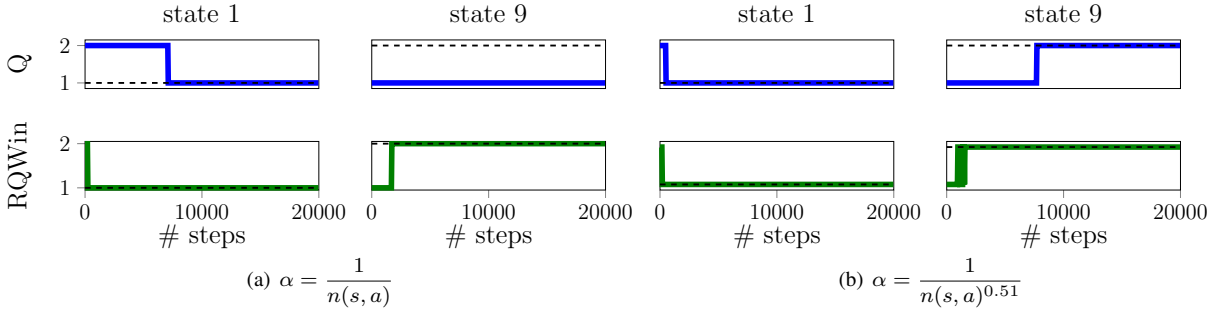
(a) $\alpha = \frac{1}{n(s,a)}$ (b) $\alpha = \frac{1}{n(s,a)^{0.51}}$ (c) $\alpha = \frac{1}{n(s,a)}$ (d) $\alpha = \frac{1}{n(s,a)^{0.51}}$

Fig. 3. Maximum action-value estimate in state 1 (3(a), 3(b)) and state 5 (3(c), 3(d)). Results are averaged over 500 experiments.



(a) $\alpha = \frac{1}{n(s,a)}$ (b) $\alpha = \frac{1}{n(s,a)^{0.51}}$ (c) $\alpha = \frac{1}{n(s,a)}$ (d) $\alpha = \frac{1}{n(s,a)^{0.51}}$

Fig. 4. Learning rate of the two actions in state 1 (4(a), 4(b)) and state 5 (4(c), 4(d)) for RQ-Learning with and without windowed variance estimation. Results are averaged over 500 experiments.



(a) $\alpha = \frac{1}{n(s,a)}$ (b) $\alpha = \frac{1}{n(s,a)^{0.51}}$

Fig. 5. Action with maximum value in state 1 and state 9 for Q-Learning and windowed RQ-Learning.

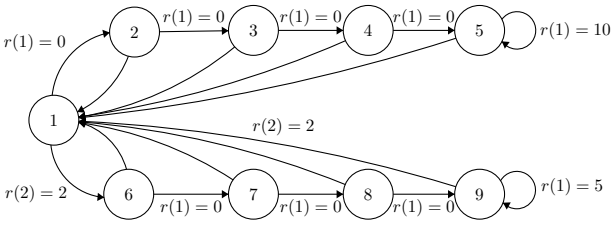


Fig. 6. Structure of the double-chain problem.

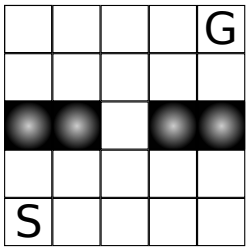


Fig. 7. Structure of the grid world with holes problem.

C. Grid World with Holes

This environment consists in a 5×5 grid with the initial position in the lower-left cell, there are 4 actions and the transition model is deterministic, the goal position in the upper-right cell and four holes in the middle row in such a way that only the cell in the middle is walkable (Figure 7). The agent receives a reward of 0 in all non-hole cells, a reward of 10 when it reaches the goal state and a reward of -10 when it reaches a cell with a hole. The episode ends when the agent reaches a cell with a hole. The discount factor is $\gamma = 0.9$.

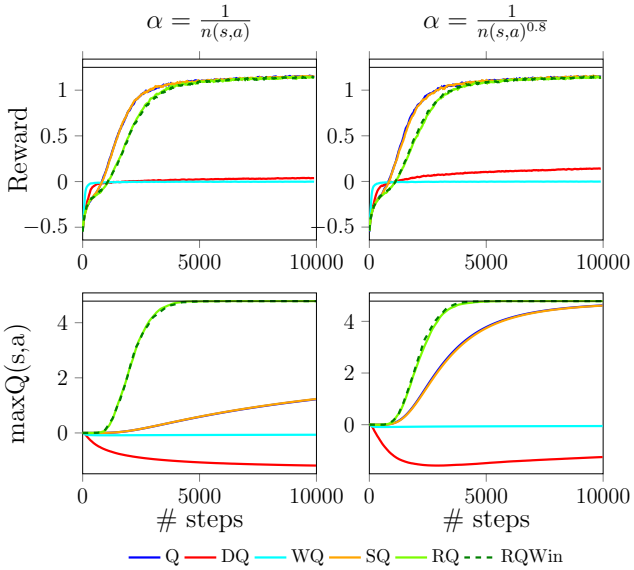


Fig. 8. Mean reward per step (top) and maximum action-value estimate in the initial state (bottom) of all the other algorithms and of the best setting of RQ-Learning for this experiment. Results are averaged over 10000 experiments.

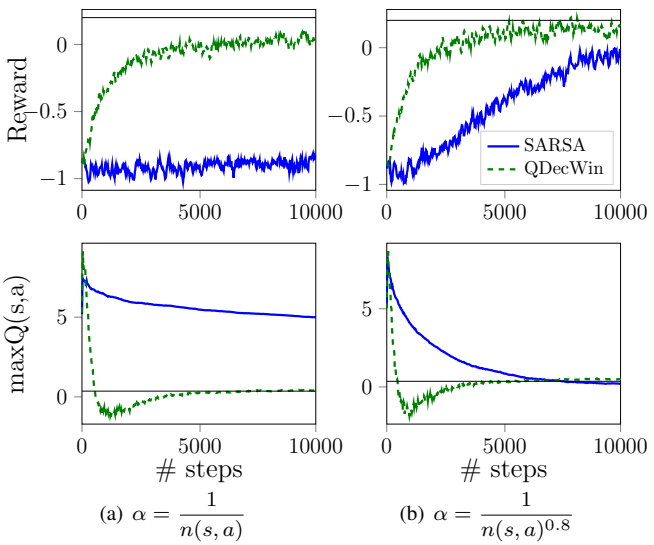


Fig. 9. Mean reward per step (top) and maximum action-value estimate in the initial state (bottom) of SARSA and of the on-policy windowed version of RQ-Learning for this experiment. Results are averaged over 1000 experiments.

RQ-Learning uses $\eta = 1$. The learning rate settings are the same of the previous problem.

We consider this simple problem to highlight the limitations of pessimistic action-value estimates. In this MDP the optimal policy consists in avoiding the hole cells stepping through the state in the middle. Notice that in this state the episode terminates with negative reward with probability $\frac{\varepsilon}{2}$ due to the ε-greedy policy used for exploration, resulting in a very low value of the state especially at the beginning of learning. Figure 8 shows that while Q-Learning, Speedy Q-Learning and RQ-Learning behave similarly well, Double Q-Learning and Weighted Q-Learning obtain very poor results due to the pessimistic estimate of the value function of the state in the middle.

D. On-policy learning

As we have discussed in the previous sections, our approach can be used also in a on-policy setting. A simple on-policy version of our algorithm can be implemented by estimating the action-value function of the current policy in the same way of the SARSA algorithm, i.e. by using the action-value function of the next action. Let u' be the next action sampled by the current policy in the current state, the on-policy update is:

$$\widetilde{R}_{t+1}(x,u) \leftarrow \widetilde{R}_t(x,u) + \alpha_t(R(x,u,x') - \widetilde{R}_t(x,u)),$$
$$\widetilde{Q}_{t+1}(x,u) \leftarrow \widetilde{Q}_t(x,u) + \beta_t(Q_t(x',u') - \widetilde{Q}_t(x,u)).$$

Figure 9 compares the windowed, on-policy version of RQ-Learning with the SARSA algorithm, in the Noisy Grid World environment. It is clear that our algorithm outperforms SARSA in this MDP. Since this is an on-policy setting, at each step the algorithm is estimating the current policy action-value function, not the optimal one. Indeed, by looking at the mean reward per step, our approach estimates the current action-value function of the policy better than the SARSA algorithm, i.e. the estimated action-value function is coherent with the performance of the policy.

V. Conclusion

In this paper we proposed a method to improve the learning process in stochastic MDPs exploiting the structure of the Bellman operator. The decomposition in two components of the Bellman error allows to consider separately the sources of uncertainty. One of these components is the expected immediate reward whose uncertainty depends only on local properties of the MDP. The other component consists in the expected value function of the next state whose uncertainty depends on the policy (in an on-line setting), on the transition model and on other action-value estimates. We showed how the proposed method obtains good results in stochastic MDPs exploiting the information on the uncertainty of the estimates by adapting the learning rate according to it. Interestingly, this method is applicable both in off-policy and in on-policy settings; moreover, it is independent from the choice of the estimator of the expected value function of the next state.

In the experimental section, we empirically show that good results in highly stochastic MDPs can also be reached by algorithms that overestimates the action-value function. These results demonstrate that there seems not to be such a strong correlation between the underestimation of the action-value function and good performance in these kind of environments, as suggested in recent literature. Indeed, while the propagation of overestimates could lead to divergent estimates, in some cases this is not an issue as the optimal policy only depends on the order of the action-values. Moreover, our method is able to converge to the optimal action-value even with a initially high overestimation. It demonstrates how overestimation allows

a better exploration given that the learning rate properly decreases. On the contrary, methods that underestimate the action-values suffer from poor exploration as we show in a simple deterministic environment.

As future work, it would be useful to derive the conditions of convergence of RQ-Learning starting from the preliminary results proposed in this paper. Moreover, it would be interesting to apply the idea of decomposing the Bellman updates, and exploiting uncertainty of its components, to continuous space problems and other types of learning settings (e.g. batch).

REFERENCES

[1] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.

[2] J. E. Smith and R. L. Winkler, "The optimizer's curse: Skepticism and postdecision surprise in decision analysis," *Management Science*, vol. 52, no. 3, pp. 311–322, 2006.

[3] E. Van den Steen, "Rational overoptimism (and other biases)," *American Economic Review*, pp. 1141–1151, 2004.

[4] H. v. Hasselt, "Double q-learning," in *Advances in Neural Information Processing Systems*, 2010, pp. 2613–2621.

[5] ——, "Estimating the maximum expected value: an analysis of (nested) cross-validation and the maximum sample average," *arXiv preprint arXiv:1302.7175*, 2013.

[6] C. D'Eramo, M. Restelli, and A. Nuara, "Estimating maximum expected value through gaussian approximation," in *International Conference on Machine Learning*, 2016, pp. 1032–1040.

[7] M. Ghavamzadeh, H. J. Kappen, M. G. Azar, and R. Munos, "Speedy q-learning," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2011, pp. 2411–2419. [Online]. Available: http://papers.nips.cc/paper/4251-speedy-q-learning.pdf

[8] R. H. Crites and A. G. Barto, "Improving elevator performance using reinforcement learning," in *Advances in neural information processing systems*, 1996, pp. 1017–1023.

[9] B.-K. Bao, B.-Q. Yin, and H.-S. Xi, "Infinite-horizon policy-gradient estimation with variable discount factor for markov decision process," in *Innovative Computing Information and Control, 2008. ICICIC'08. 3rd International Conference on*. IEEE, 2008, pp. 584–584.

[10] V. François-Lavet, R. Fonteneau, and D. Ernst, "How to discount deep reinforcement learning: Towards new dynamic strategies," *arXiv preprint arXiv:1512.02011*, 2015.

[11] E. Even-Dar and Y. Mansour, *Learning Rates for Q-Learning*. Springer Berlin Heidelberg, 2001, pp. 589–604. [Online]. Available: http://dx.doi.org/10.1007/3-540-44581-1_39

[12] J. Peters, K. Mulling, and Y. Altun, "Relative entropy policy search," in *AAAI*, 2010.