Taming the Noise in Reinforcement Learning via Soft Updates

Roy Fox[*] Hebrew University

Ari Pakman[*] Columbia University

Naftali Tishby Hebrew University

Abstract

Model-free reinforcement learning algorithms, such as Q-learning, perform poorly in the early stages of learning in noisy environments, because much effort is spent unlearning biased estimates of the state-action value function. The bias results from selecting, among several noisy estimates, the apparent optimum, which may actually be suboptimal. We propose G-learning, a new off-policy learning algorithm that regularizes the value estimates by penalizing deterministic policies in the beginning of the learning process. We show that this method reduces the bias of the value-function estimation, leading to faster convergence to the optimal value and the optimal policy. Moreover, G-learning enables the natural incorporation of prior domain knowledge, when available. The stochastic nature of G-learning also makes it avoid some exploration costs, a property usually attributed only to on-policy algorithms. We illustrate these ideas in several examples, where G-learning results in significant improvements of the convergence rate and the cost of the learning process.

1 INTRODUCTION

The need to separate signals from noise stands at the center of any learning task in a noisy environment. While a rich set of tools to regularize learned parameters has been developed for supervised and unsupervised learning problems, in areas such as reinforcement learning there still exists a vital need for techniques that tame the noise and avoid overfitting and local minima.

One of the central algorithms in reinforcement learning is Q-learning [1], a model-free off-policy algorithm, which attempts to estimate the optimal value function Q, the cost-to-go of the optimal policy. To enable this estimation, a stochastic exploration policy is used by the learning agent to interact with its environment and explore the model. This approach is very successful and popular, and despite several alternative approaches developed in recent years [2, 3, 4], it is still being applied successfully in complex domains for which explicit models are lacking [5].

However, in noisy domains, in early stages of the learning process, the min (or max) operator in Q-learning brings about a bias in the estimates. This problem is akin to the "winner's curse" in auctions [6, 7, 8, 9]. With too little evidence, the biased estimates may lead to wrong decisions, which slow down the convergence of the learning process, and require subsequent unlearning of these suboptimal behaviors.

In this paper we present G-learning, a new off-policy information-theoretic approach to regularizing the state-action value function learned by an agent interacting with its environment in model-free settings.

This is achieved by adding to the cost-to-go a term that penalizes deterministic policies which diverge from a simple stochastic prior policy [10]. With only a small sample to go by, G-learning prefers a more randomized policy, and as samples accumulate, it gradually shifts to a more deterministic and exploiting policy. This transition is managed by appropriately scheduling the coefficient of the penalty term as learning proceeds.

In Section 4 we discuss the theoretical and practical aspects of scheduling this coefficient, and suggest that a simple linear schedule can perform well. We show that G-learning with this schedule reduces the value estimation bias by avoiding overfitting in its selection of the update policy. We further establish empirically the link between bias reduction and learning performance, that has been the underlying assumption in many approaches to reinforcement learning [11, 12, 13, 14]. The examples in Section 6 demonstrate the significant improvement thus obtained.

Furthermore, in domains where exploration incurs significantly higher costs than exploitation, such as the classic

[*]These authors contributed equally to this work.

cliff domain [2], G-learning with an ϵ-greedy exploration policy is exploration-aware, and chooses a less costly exploration policy, thus reducing the costs incurred during the learning process. Such awareness to the cost of exploration is usually attributed to on-policy algorithms, such as SARSA [2, 4] and Expected-SARSA [15, 16]. The remarkable finding that G-learning exhibits on-policy-like properties is illustrated in the example of Section 6.2.

In Section 2 we discuss the problem of learning in noisy environments. In Section 3 we introduce the penalty term, derive G-learning and prove its convergence. In Section 4 we determine a schedule for the coefficient of the information penalty term. In Section 5 we discuss related work. In Section 6 we illustrate the strengths of the algorithm through several examples.

2 LEARNING IN NOISY ENVIRONMENTS

2.1 NOTATION AND BACKGROUND

We consider the usual setting of a Markov Decision Process (MDP), in which an agent interacts with its environment by repeatedly observing its state $s \in S$, taking an action $a \in A$, with A and S finite, and incurring cost $c \in \mathbb{R}$. This induces a stochastic process $s_0, a_0, c_0, s_1, \ldots$, where s_0 is fixed, and where for $t \geq 0$ we have the Markov properties indicated by the conditional distributions $a_t \sim \pi_t(a_t|s_t)$, $c_t \sim \theta(c_t|s_t, a_t)$ and $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$.

The objective of the agent is to find a time-invariant policy π that minimizes the total discounted expected cost

$$V^\pi(s) = \sum_{t\geq 0} \gamma^t \, \mathrm{E}[c_t|s_0 = s], \tag{1}$$

simultaneously for any $s \in S$, for a given discount factor $0 \leq \gamma < 1$. For each t, the expectation above is over all trajectories of length t starting at $s_0 = s$. A related quantity is the state-action value function

$$\begin{aligned} Q^\pi(s,a) &= \sum_{t\geq 0} \gamma^t \, \mathrm{E}[c_t|s_0 = s, a_0 = a] \\ &= \mathrm{E}_\theta[c|s,a] + \gamma \, \mathrm{E}_p[V^\pi(s')|s,a], \end{aligned} \tag{2}$$

which equals the total discounted expected cost that follows from choosing action a in state s, and then following the policy π.

If we know the distributions p and θ (or at least $\mathrm{E}_\theta[c|s,a]$), then it is easy to find the optimal state-action value function

$$Q^*(s,a) = \min_\pi Q^\pi(s,a) \tag{3}$$

using standard techniques, such as Value Iteration [17]. Our interest is in model-free learning, where the model parameters are unknown. Instead, the agent obtains samples from $p(s_{t+1}|s_t, a_t)$ and $\theta(c_t|s_t, a_t)$ through its interaction with the environment. In this setting, the Q-learning algorithm [1] provides a method for estimating Q^*. It starts with an arbitrary Q, and in step t upon observing s_t, a_t, c_t and s_{t+1}, performs the update

$$\begin{aligned} Q(s_t,a_t) \leftarrow (1-\alpha_t) Q(s_t,a_t) \\ + \alpha_t \left(c_t + \gamma \sum_{a'} \pi(a'|s_{t+1}) Q(s_{t+1},a') \right), \end{aligned} \tag{4}$$

with some learning rate $0 \leq \alpha_t \leq 1$, and the greedy policy for Q having

$$\pi(a|s) = \delta_{a,a^*(s)}; \qquad a^*(s) = \arg\min_a Q(s,a). \tag{5}$$

$Q(s,a)$ is unchanged for any $(s,a) \neq (s_t,a_t)$. If the learning rate satisfies

$$\sum_t \alpha_t = \infty; \qquad \sum_t \alpha_t^2 < \infty, \tag{6}$$

and the interaction itself uses an exploration policy that returns to each state-action pair infinitely many times, then Q is a consistent estimator, converging to Q^* with probability 1 [1, 17]. Similarly, if the update rule (4) uses a fixed update policy $\pi = \rho$, we call this algorithm Q$^\rho$-learning, because Q converges to Q^ρ with probability 1.

2.2 BIAS AND EARLY COMMITMENT

Despite the success of Q-learning in many situations, learning can proceed extremely slowly when there is noise in the distribution, given s_t and a_t, of either of the terms of (2), namely the cost c_t and the value of the next state s_{t+1}. The source of this problem is a negative bias introduced by the min operator in the estimator $\min_{a'} Q(s_{t+1},a')$, when (5) is plugged into (4).

To illustrate this bias, assume that $Q(s,a)$ is an unbiased but noisy estimate of the optimal $Q^*(s,a)$. Then Jensen's inequality for the concave min operator implies that

$$\mathrm{E}[\min_a Q(s,a)] \leq \min_a Q^*(s,a), \tag{7}$$

with equality only when Q already reveals the optimal policy by having $\arg\min_a Q(s,a) = \arg\min_a Q^*(s,a)$ with probability 1, so that no further learning is needed. The expectation in (7) is with respect to the learning process, including any randomness in state transition, cost, exploration and internal update, given the domain.

This is an optimistic bias, causing the cost-to-go to appear lower than it is (or the reward-to-go higher). It is the well known "winner's curse" problem in economics and decision theory [6, 7, 8, 9], and in the context of Q-learning it was studied before in [3, 11, 12, 13]. A similar problem occurs when a function approximation scheme is used

for Q instead of a table, even in the absence of transition or cost noise, because the approximation itself introduces noise [18].

As the sample size increases, the variance in $Q(s,a)$ decreases, which in turn reduces the bias in (7). This makes the update policy (5) more optimal, and the update increasingly similar to Value Iteration.

2.3 THE INTERPLAY OF VALUE BIAS AND POLICY SUBOPTIMALITY

It is insightful to consider the effect of the bias not only on the estimated value function, but also on the real value V^π of the greedy policy (5), since in many cases the latter is the actual output of the learning process. The central quantity of interest here is the gap $Q^*(s,a')-V^*(s)$, in a given state s, between the value of a non-optimal action a' and that of the optimal action.

Consider first the case in which the gap is large compared to the noise in the estimation of the $Q(s,a)$ values. In this case, a' indeed appears suboptimal with high probability, as desired. Interestingly, when the gap is very small relative to the noise, the learning agent should not worry, either. Confusing such a' for the optimal action has a limited effect on the value of the greedy policy, since choosing a' is near-optimal.

We conclude that the real value V^π of the greedy policy (5) is suboptimal only in the intermediate regime, when the gap is of the order of the noise, and neither is small. The effect of the noise can be made even worse by the propagation of bias between states, through updates. Such propagation can cause large-gap suboptimal actions to nevertheless appear optimal, if they lead to a region of state-space that is highly biased.

2.4 A DYNAMIC OPTIMISM-UNCERTAINTY LOOP

The above considerations were agnostic to the exploration policy, but the bias reduction can be accelerated by an exploration policy that is close to being greedy. In this case, high-variance estimation is self-correcting: an estimated state value with optimistic bias draws exploration towards that state, leading to a decrease in the variance, which in turn reduces the optimistic bias. This is a dynamic form of optimism under uncertainty. While in the usual case the optimism is externally imposed as an initial condition [19], here it is spontaneously generated by the noise and self-corrected through exploration.

The approach we propose below to reduce the variance is motivated by electing to represent the uncertainty explicitly, and not indirectly through an optimistic bias. We notice that although *in the end* of the learning process one obtains the deterministic greedy policy from $Q(a,s)$ as in (5), *during* the learning itself the bias in Q can be ameliorated by avoiding the hard min operator, and refraining from committing to a deterministic greedy policy. This can be achieved by adding to Q, at the early learning stage, a term that penalizes deterministic policies, which we consider next.

3 LEARNING WITH SOFT UPDATES

3.1 THE FREE-ENERGY FUNCTION G AND G-LEARNING

Let us adopt, before any interaction with the environment, a simple stochastic prior policy $\rho(a|s)$. For example, we can take the uniform distribution over the possible actions. The *information cost* of a learned policy $\pi(a|s)$ is defined as

$$g^\pi(s,a) = \log \tfrac{\pi(a|s)}{\rho(a|s)}, \tag{8}$$

and its expectation over the policy π is the Kullback-Leibler (KL) divergence of $\pi_s = \pi(\cdot|s)$ from $\rho_s = \rho(\cdot|s)$,

$$\mathrm{E}_\pi[g^\pi(s,a)|s] = \mathrm{D}_{\mathrm{KL}}[\pi_s \| \rho_s]. \tag{9}$$

The term (8) penalizes deviations from the prior policy and serves to regularize the optimal policy away from a deterministic action. In the context of the MDP dynamics $p(s_{t+1}|s_t,a_t)$, similarly to (1), we consider the total discounted expected information cost

$$I^\pi(s) = \sum_{t\geq 0} \gamma^t \,\mathrm{E}[g^\pi(s_t,a_t)|s_0 = s]. \tag{10}$$

The discounting in (1) and (10) is justified by imagining a horizon $T \sim \mathrm{Geom}(1-\gamma)$, distributed geometrically with parameter $1-\gamma$. Then the cost-to-go V^π in (1) and the information-to-go I^π in (10) are the total (undiscounted) expected T-step costs.

Adding the penalty term (10) to the cost function (1) gives

$$\begin{aligned} F^\pi(s) &= V^\pi(s) + \tfrac{1}{\beta} I^\pi(s), \\ &= \sum_{t\geq 0} \gamma^t \,\mathrm{E}[\tfrac{1}{\beta} g^\pi(s_t,a_t) + c_t | s_0 = s], \end{aligned} \tag{11}$$

called the *free-energy function* by analogy with a similar quantity in statistical mechanics [10].

Here β is a parameter that sets the relative weight between the two costs. For the moment, we assume that β is fixed. In following sections, we let β grow as the learning proceeds.

In analogy with the Q^π function (2), let us define the *state-action free-energy function* $G^\pi(s,a)$ as

$$\begin{aligned} G^\pi(s,a) &= \mathrm{E}_\theta[c|s,a] + \gamma \,\mathrm{E}_p[F^\pi(s')|s,a] \\ &= \sum_{t\geq 0} \gamma^t \,\mathrm{E}[c_t + \tfrac{\gamma}{\beta} g^\pi(s_{t+1},a_{t+1}))|s_0 = s, a_0 = a], \end{aligned} \tag{12}$$

and note that it does not involve the information term at time $t = 0$, since the action $a_0 = a$ is already known. From the definitions (11) and (12) it follows that

$$F^\pi(s) = \sum_a \pi(a|s) \left[\tfrac{1}{\beta} \log \tfrac{\pi(a|s)}{\rho(a|s)} + G^\pi(s,a) \right]. \quad (13)$$

It is easy to verify that, given the G function, the above expression for F^π has gradient 0 at

$$\pi(a|s) = \frac{\rho(a|s) e^{-\beta G(s,a)}}{\sum_{a'} \rho(a'|s) e^{-\beta G(s,a')}}, \quad (14)$$

which is therefore the optimal policy.

The policy (14) is the soft-min operator applied to G, with inverse-temperature β. When β is small, the information cost is dominant, and π approaches the prior ρ. When β is large, we are willing to diverge much from the prior to reduce the external cost, and π approaches the deterministic greedy policy for G.

Evaluated at the soft-greedy policy (14), the free energy (13) is

$$F^\pi(s) = -\tfrac{1}{\beta} \log \sum_a \rho(a|s) e^{-\beta G^\pi(s,a)}, \quad (15)$$

and plugging this expression into (12), we get that the optimal G^* is a fixed point of the equation

$$\begin{aligned} G^*(s,a) = {} & \mathrm{E}_\theta[c|s,a] \quad (16) \\ & - \tfrac{\gamma}{\beta} \mathrm{E}_p \left[\log \sum_{a'} \rho(a'|s') e^{-\beta G^*(s',a')} \right] \\ \equiv {} & \mathbf{B}^*[G^*]_{(s,a)}. \quad (17) \end{aligned}$$

Based on the above expression, we introduce G-learning as an off-policy TD-learning algorithm [2], that learns the optimal G^* from the interaction with the environment by applying the update rule

$$\begin{aligned} G(s_t,a_t) \leftarrow {} & (1-\alpha_t) G(s_t,a_t) \quad (18) \\ & + \alpha_t \left(c_t - \tfrac{\gamma}{\beta} \log \left(\sum_{a'} \rho(a'|s_{t+1}) e^{-\beta G(s_{t+1},a')} \right) \right). \end{aligned}$$

3.2 THE ROLE OF THE PRIOR

Clearly the choice of the prior policy ρ is significant in the performance of the algorithm. The prior policy can encode any prior knowledge that we have about the domain, and this can improve the convergence if done correctly. However an incorrect prior policy can hinder learning. We should therefore choose a prior policy that represents all of our prior knowledge, but nothing more. This prior policy has maximal entropy given the prior knowledge [20].

In our examples in Section 6, we use the uniform prior policy, representing no prior knowledge. Both in Q-learning and in G-learning, we could utilize the prior knowledge that moving into a wall is never a good action, by eliminating those actions. One advantage of G-learning is that it can utilize softer prior knowledge. For example, a prior policy that gives lower probability for moving into a wall represent the prior knowledge that such an action is usually (but not always) harmful, a type of knowledge that cannot be utilized in Q-learning.

We have presented G-learning in a fully parameterized formulation, where the function G is stored in a lookup table. Practical applications of Q-learning often resort to approximating the function Q through function approximations, such as linear expansions or neural networks [2, 3, 4, 21, 5]. Such an approximation generates inductive bias, which is another form of implicit prior knowledge. While G-learning is introduced here in its table form, preliminary results indicate that its benefits carry over to function approximations, despite the challenges posed by this extension.

3.3 CONVERGENCE

In this section we study the convergence of G under the update rule (18). Recall that the supremum norm is defined as $|x|_\infty = \max_i |x_i|$. We need the following Lemma, proved in the Supplementary Material.

Lemma 1. *The operator* $\mathbf{B}^*[G]_{(s,a)}$ *defined in* (17) *is a contraction in the supremum norm,*

$$\left| \mathbf{B}^*[G_1] - \mathbf{B}^*[G_2] \right|_\infty \le \gamma \left| G_1 - G_2 \right|_\infty. \quad (19)$$

The update equation (18) of the algorithm can be written as a stochastic iteration equation

$$\begin{aligned} G_{t+1}(s_t,a_t) = {} & (1-\alpha_t) G_t(s_t,a_t) \quad (20) \\ & + \alpha_t (\mathbf{B}^*[G_t]_{(s_t,a_t)} + z_t(c_t, s_{t+1})) \end{aligned}$$

where the random variable z_t is

$$\begin{aligned} z_t(c_t,s_{t+1}) = {} & -\mathbf{B}^*[G_t]_{(s_t,a_t)} \quad (21) \\ & + c_t - \tfrac{\gamma}{\beta} \log \sum_{a'} \rho(a'|s_{t+1}) e^{-\beta G_t(s_{t+1},a')}. \end{aligned}$$

Note that z_t has expectation 0. Many results exist for iterative equations of the type (20). In particular, given conditions (6) for α_t, the contractive nature of $\mathbf{B}^*$, infinite visits to each pair (s_t, a_t) and assuming that $|z_t| < \infty$, G_t is guaranteed to converge to the optimal G^* with probability 1 [17, 22].

4 SCHEDULING β

In the previous section, we showed that running G-learning with a fixed β converges, with probability 1, to the optimal G^* for that β, given by the recursion in (12)–(14).

When $\beta = \infty$, the equations for G^* and F^* degenerate into the equations for Q^* and V^*, and G-learning becomes Q-learning. When $\beta = 0$, the update policy π in (14) is equal to the prior ρ. This case, denoted Q$^\rho$-learning, converges to Q^ρ.

In an early stage of learning, Q$^\rho$-learning has an advantage over Q-learning, because it avoids committing to a deterministic policy based on a noisy Q function. In a later stage of learning, when Q is a more precise estimate of Q^*, Q-learning gains the advantage by updating with a better policy than the prior. This is demonstrated in section 6.1.

We would therefore like to schedule β so that G-learning makes a smooth transition from Q$^\rho$-learning to Q-learning, just at the right pace to enjoy the early advantage of the former and the late advantage of the latter. As we argue below, such a β always exists.

4.1 ORACLE SCHEDULING

To consider the effect of the β scheduling on the correction of the bias (7), suppose that during learning we reach some G that is an unbiased estimate of G^*. $G(s_t, a_t)$ would remain unbiased if we update it towards

$$c_t + \gamma G(s_{t+1}, a^*) \tag{22}$$

with

$$a^* = \arg\min_{a'} G^*(s_{t+1}, a'), \tag{23}$$

but we do not have access to this optimal action. If we use the update rule (18) with $\beta = 0$, we update $G(s_t, a_t)$ towards

$$c_t + \gamma \sum_{a'} \rho(a'|s_{t+1}) G(s_{t+1}, a'), \tag{24}$$

which is always at least as large as (22), creating a positive bias. If we use $\beta = \infty$, we update $G(s_t, a_t)$ towards

$$c_t + \gamma \min_{a'} G(s_{t+1}, a'), \tag{25}$$

which creates a negative bias, as explained in Section 2.2. Since the right-hand side of (18) is continuous and monotonic in β, there must be some β for which this update rule is unbiased.

This is a non-constructive proof for the existence of a β schedule that keeps the value estimators unbiased (or at least does not accumulate additional bias). We can imagine a scheduling oracle, and a protocol for the agent by which to consult the oracle and obtain the β for its soft updates. At the very least, the oracle must be told the iteration index t, but it can also be useful to let β depend on any other aspect of the learning process, particularly the current world state s_t.

4.2 PRACTICAL SCHEDULING

A good schedule should increase β as learning proceeds, because as more samples are gathered the variance of G decreases, allowing more deterministic policies. In the examples of Section 6 we adopted the linear schedule

$$\beta_t = kt, \tag{26}$$

with some constant $k > 0$. Another possibility that we explored was to make β inversely proportional to a running average of the Bellman error, which decreases as learning progresses. The results were similar to the linear schedule.

The optimal parameter k can be obtained by performing initial runs with different values of k and picking the value whose learned policy gives empirically the lower cost-to-go. Although this exploration would seem costly compared to other algorithms for which no parameter tuning is needed, these initial runs do not need to be carried for many iterations. Moreover, in many situations the agent is confronted with a class of similar domains, and tuning k in a few initial domains leads to an improved learning for the whole class. This is the case in the domain-generator example in Section 6.1.

5 RELATED WORK

The connection between domain noise or function approximation, and the statistical bias in the Q function, was first discussed in [18, 3]. An interesting modification of Q-learning to address this problem is Double-Q-learning [11, 14], which uses two estimators for the Q function to alleviate the bias. Other modifications of Q-learning that attempt to reduce or correct the bias are suggested in [12, 13].

An early approach to Q-learning in continuous noisy domains was to learn, instead of the value function, the advantage function $A(s, a) = Q(s, a) - V(s)$ [23]. The algorithm represents A and V separately, and the optimal action is determined from $A(s, a)$ as $a^*(s) = \arg\min_a A(s, a)$. In noisy environments, learning A is shown in some examples to be faster than learning Q [23, 24].

More recently, it was shown that the advantage learning algorithm is a gap-increasing operator [25]. As discussed in Section 2.2, the action gap is a central factor in the generation of bias, and increasing the gap should also help reduce the bias. In Section 6.1 we compare our algorithm to the consistent Bellman operator $\mathcal{T}_C$, one of the gap-increasing algorithms introduced in [25].

For other works that study the effect of noise in Q-learning, although without identifying the bias (7), see [26, 27, 28].

Information considerations have received attention in recent years in various machine learning settings, with the free energy F^π and similar quantities used as a design

principle for policies in known MDPs [10, 29, 30]. Other works have used related methods for reinforcement learning [31, 32, 33, 34, 35]. A KL penalty similar to ours is used in [35], in settings with known reward and transition functions, to encourage "curiosity".

Soft-greedy policies have been used before for exploration [2, 36], but to our knowledge G-learning is the first TD-learning algorithm to explicitly use soft-greedy policies in its updates.

Particularly relevant to our work is the approach studied in [32]. There the policy is iteratively improved by optimizing it in each iteration under the constraint that it only diverges slightly, in terms of KL-divergence, from the empirical distribution generated by the previous policy. In contrast, in G-learning we measure the KL-divergence from a fixed prior policy, and in each iteration allow the divergence to grow larger by increasing β. Thus the two methods follow different information-geodesics from the stochastic prior policy to more and more deterministic policies.

This distinction is best demonstrated by considering the Ψ-learning algorithm presented in [33, 34], based on the same approach as [32]. It employs the update rule

$$\Psi(s_t, a_t) \leftarrow \Psi(s_t, a_t) + \alpha_t(c_t + \gamma\bar{\Psi}(s_{t+1}) - \bar{\Psi}(s_t)), \tag{27}$$

with

$$\bar{\Psi}(s) = -\log \sum_a \rho(a|s) e^{-\Psi(s,a)}, \tag{28}$$

which is closely related to our update of G in (18).

Apart from lacking a β parameter, the most important difference is that the update of Ψ involves subtracting $\alpha_t\bar{\Psi}(s_t)$, whereas the update of G involves subtracting $\alpha_t G(s_t, a_t)$. This seemingly minor modification has a large impact on the behavior of the two algorithms. The update of G is designed to pull it towards the optimal state-action free energy G^*, for all state-action pairs. In contrast, subtracting the log-partition $\bar{\Psi}(s_t)$, in the long run pulls only $\Psi(s_t, a^*)$, with a^* the optimal action, towards its true value, while for the other actions the values grow to infinity. In this sense, the Ψ-learning update (27) is an information-theoretic gap-increasing Bellman operator [25].

The growth to infinity of suboptimal values separates them from the optimal value, and drives the algorithm to convergence. In G-learning, this parallels the increase in β with the accumulation of samples. However, there is a major benefit to keeping G reliable in all its parameters, and controlling it with a separate β parameter. In Ψ-learning, the Ψ function penalizes actions it deems suboptimal. If early noise causes an error in this penalty, the algorithm needs to unlearn it - a similar drawback to that of Q-learning. In Section 6, we demonstrate the improvement offered by G-learning.

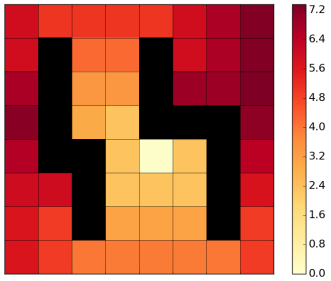

Figure 1: Gridworld domain. The agent can choose an adjacent square as the target to move to, and then may end up stochastically in a square adjacent to that target. The color scale indicates the optimal values V^* with a fixed cost of 1 per step.

6 EXAMPLES

This section illustrates how G-learning improves on existing model-free learning algorithms in several settings. The domains we use are clean and simple, to demonstrate that the advantages of G-learning are inherent to the algorithm itself.

We schedule the learning rate α_t as

$$\alpha_t = n_t(s_t, a_t)^{-\omega}, \tag{29}$$

where $n_t(s_t, a_t)$ is the number of times the pair (s_t, a_t) was visited. This scheme is widely used, and is consistent with (6) for $\omega \in (1/2, 1]$. We choose $\omega = 0.8$, which is within the range suggested in [37].

We schedule β linearly, as discussed in Section 4.2. In each case, we start with 5 preliminary runs of G-learning with various linear coefficients, and pick the coefficient with the lowest empirical cost. This coefficient is used in the subsequent test runs, whose results are plotted in Figure 2.

In all cases, we use a uniform prior policy ρ, a discount factor $\gamma = 0.95$, and 0 for the initial values ($Q_0 = 0$ in Q-learning, and similarly in the other algorithms). Except when mentioned otherwise, we employ random exploration, where s_t and a_t are chosen uniformly at the beginning of each time step, independently of any previous sample. This exploration technique is useful when comparing update rules, while controlling for the exploration process.

6.1 GRIDWORLD

Our first set of examples occurs in a gridworld of 8×8 squares, with some unavailable squares occupied by walls shown in black (Figure 1). The lightest square is the goal, and reaching it ends the episode.

At each time step, the agent can choose to move one square in any of the 8 directions (including diagonally), or stay in place. If the move is blocked by a wall or the edge of the

board, it effectively attempts to stay in place. With some probability, the action performed by the agent is further followed by an additional random slide: with probability 0.15 to each vertically or horizontally adjacent available position, and with probability 0.05 to each diagonally adjacent available position.

The noise associated with these random transitions can be enhanced further by the possible variability in the costs incurred along the way. We consider three cases. In the first case, the cost in each step is fixed at 1. In the second case, the cost in each step is distributed normally i.i.d, with mean 1 and standard deviation 2. In the third case we define a distribution over domains, such that at the time of domain-generation the mean cost for each state-action is distributed uniformly i.i.d over $[1, 3]$. Once the domain has been generated and interaction begins, the cost itself in each step is again distributed normally i.i.d, with the generated mean and standard deviation 4.

We attempt to learn these domains using various algorithms. Figure 2 summarizes the results for Q-learning, G-learning, Double-Q-learning [11], Ψ-learning [33, 34] and the consistent Bellman operator $\mathcal{T}_C$ of [25]. We also include Q$^\rho$-learning, which performs updates as in (4) towards the prior policy ρ. Comparison with Speedy-Q-learning [12] is omitted, since it showed no improvement over vanilla Q-learning in these settings. In our experiments, these algorithms had comparable running times.

The β scheduling used in G-learning is linear, with the coefficient k equal to 10^{-3}, 10^{-4}, $5 \cdot 10^{-5}$ and 10^{-6}, respectively for the fixed-cost, noisy-cost, domain-generator and cliff domains (see Section 6.2).

For each case, Figure 2 shows the evolution over 250,000 algorithm iterations of the following three measures, averaged over $N = 100$ runs:

1. Empirical bias, defined as

$$\frac{1}{Nn} \sum_{i=1}^{N} \sum_{s=1}^{n} (V_{i,t}(s) - V_i^*(s)), \tag{30}$$

   where i indexes the N runs and s the n states. Here $V_{i,t}$ is the greedy value based on the estimate obtained by each algorithm (Q, G, etc.), in iteration t of run i. The optimal value V_i^*, computed via Value Iteration, varies between runs in the domain-generator case.

2. Mean absolute error in V

$$\frac{1}{Nn} \sum_{i=1}^{N} \sum_{s=1}^{n} |V_{i,t}(s) - V_i^*(s)|. \tag{31}$$

   A low bias could result from the cancellation of terms with high positive and negative biases. A convergence in the absolute error is more indicative of the actual convergence of the value estimates.

3. Increase in cost-to-go, relative to the optimal policy

$$\frac{1}{Nn} \sum_{i=1}^{N} \sum_{s=1}^{n} (V^{\pi_{i,t}}(s) - V_i^*(s)). \tag{32}$$

   This measures the quality of the learned policy. Here $\pi_{i,t}$ is the greedy policy based on the state-action value estimates, and $V^{\pi_{i,t}}$ is its value in the model, computed via Value Iteration.

An algorithm is better when these measures reach zero faster. As is clear in Figure 2, in the domains with noisy cost (Rows 2 and 3), G-learning dominates over all the other competing algorithms by the three measures. The results are statistically significant, but plotting confidence intervals would clutter the figure.

An important and surprising point of Figure 2 is that Q$^\rho$-learning always outperforms Q-learning initially, before degrading. The reason is that the Q-learning updates initially rely on very few samples, so these harmful updates need to be undone by later updates. Q$^\rho$-learning, on the other hand, updates in the direction of a uniform prior. This gives an early advantage in mapping out the local topology of the problem, before long-range effects start pulling the learning towards the suboptimal Q^ρ.

The power of G-learning is that it enjoys the early advantage of Q$^\rho$-learning, and smoothly transitions to the convergence advantage of Q-learning. When β is small, the information cost g_t (8) outweighs the external costs c_t, and we update towards ρ. As samples keep coming in, and our estimates improve, β increases, and the updates gradually lean more towards a cost-optimizing policy. Unlike early stages in Q-learning, at this point G_t is already a good estimate, and we avoid overfitting. As mentioned above, Figure 2 shows that this effect is more manifest in noisier scenarios.

Finally, Figure 3 shows running averages of the Bellman error for the different algorithms considered. The Bellman error in G-learning is the coefficient multiplying α_t in (18),

$$\Delta G_t \equiv c_t - \frac{\gamma}{\beta} \log \left(\sum_{a'} \rho(a'|s_{t+1}) e^{-\beta G_t(s_{t+1}, a')} \right) - G_t(s_t, a). \tag{33}$$

When learning ends and $G = G^*$, the expectation of ΔG_t is zero (see (16)). Similar definitions hold for the other learning algorithms we compare with. As is clear from Figure 3, G-learning reaches zero average Bellman error faster than the competing methods, even while β is still increasing in order to make G^* converge to Q^*.

6.2 CLIFF WALKING

Cliff walking is a standard example in reinforcement learning [2], that demonstrates an advantage of on-policy algorithms such as SARSA [2, 4] and Expected-SARSA [15,

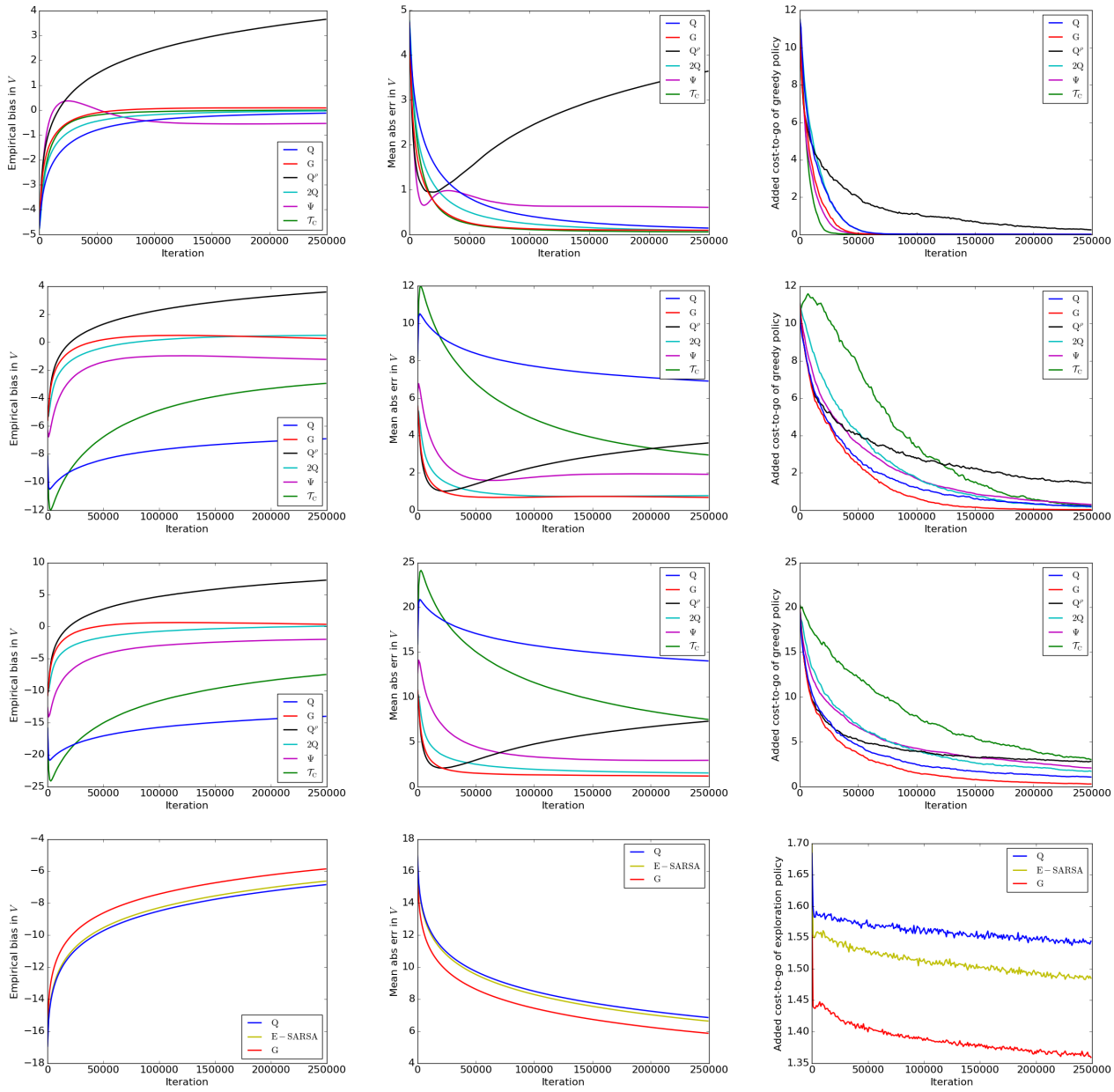

Figure 2: **Gridworld (Rows 1-3):** Comparison of Q-, G-, Q^ρ-, Double-Q-, Ψ- and $\mathcal{T}_C$-learning. **Row 1:** The cost in each step is fixed at 1. **Row 2:** The cost in each step is distributed as $\mathcal{N}(1, 2^2)$. **Row 3:** In each run, the domain is generated by drawing each $\mathrm{E}[c|s,a]$ uniformly over $[1,3]$. The cost in each step is distributed as $\mathcal{N}(\mathrm{E}[c|s,a], 4^2)$. Note that in the noisy domains (Rows 2 and 3), G-learning dominates over all the other algorithms by the three measures. **Cliff (Row 4):** Comparison of Q- and G-learning, and Expected-SARSA. The cost in each step is 1, and falling off the cliff costs 5. **Left:** Empirical bias of V, relative to V^* (30). **Middle:** Mean absolute error between V and V^* (31). **Right:** Value of greedy policy, with the baseline V^* subtracted (32); except in Row 4, which shows the value of the exploration policy.

16] over off-policy learning approaches such as Q-learning. We use it to show another interesting strength of G-learning.

In this example, the agent can walk on the grid in Figure 4 horizontally or vertically, with deterministic transitions. Each step costs 1, except when the agent walks off the cliff (the bottom row), which costs 5, or reaches the goal (lower right corner), which costs 0. In either of these cases, the position resets to the lower left corner.

Exploration is now on-line, with s_t taken from the end of the previous step. The exploration policy in our simulations is ϵ-greedy with $\epsilon = 0.1$, i.e. with probability ϵ the agent

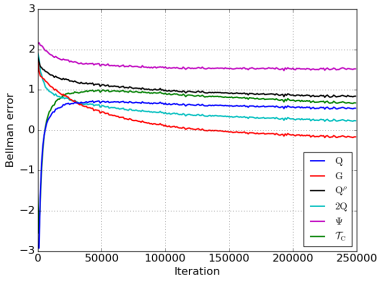

Figure 3: Running average of the Bellman error in the grid-world domain-generator example for Q-, G-, Q^ρ-, Double-Q-, Ψ- and $\mathcal{T}_C$-learning. The results for the other two grid-worlds of Figure 2 are similar.

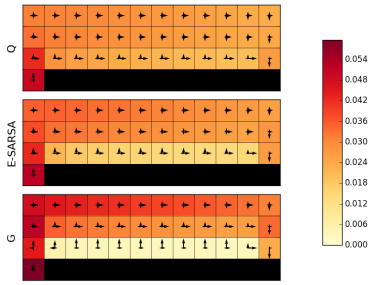

Figure 4: Cliff domain. The agent can choose a horizontally or vertically adjacent square, and moves there deterministically. The color scale and the arrow lengths indicate, respectively, the frequency of visiting each state and of making each transition, in the first 250,000 iterations of Q-learning, Expected-SARSA and G-learning. The near-greedy exploration policy of Q-learning has higher chance of taking the shortest path near the edge of the cliff at the bottom, than that of G-learning. As an off-policy algorithm, Q-learning fails to optimize for the exploration policy, whereas G-learning succeeds.

chooses a random action, and otherwise it takes deterministically the one that seems optimal. In practice, ϵ can be decreased after the learning phase, however it is also common to keep ϵ fixed for continued exploration [2].

In this setting, as shown in the bottom row of Figure 2, an off-policy algorithm like Q-learning performs poorly in terms of the value of its exploration policy, and the empirical cost it incurs. It learns a rough estimate of Q^* quickly, and then tends to use it and walk on the edge of the cliff. This leads to the agent occasionally exploring the possibility of falling off the cliff. In contrast, an on-policy algorithm like Expected-SARSA [15, 16] learns the value of its exploration policy, and quickly manages to avoid the cliff.

Figure 4 compares Q-learning, G-learning and Expected-SARSA in this domain, and shows that G-learning learns to avoid the cliff even better than an on-policy algorithm, although for a different reason. As an off-policy algorithm, G-learning does learn the value of the update policy, which prefers trajectories far from the cliff in the early stages of learning. This occurs because near the cliff, avoiding the cost of falling requires ruling out downward moves, which has a high information cost. On the other hand, trajectories far from the cliff, while paying a higher cost in overall distance to the goal, enjoy lower information cost because acting randomly is not costly for them.

As shown in the bottom row of Figure 2, by using a greedy policy for G as the basis of the ϵ-greedy exploration, we enjoy the benefits of being aware of the value of the exploration policy during the learning stage. At the same time, G-learning converges faster than either Q-learning or Expected-SARSA to the correct value function. In this case the "noise" that G-learning mitigates is related to the variability associated with the exploration.

7 CONCLUSIONS

The algorithm we have introduced successfully mitigates the slow learning problem of early stage Q-learning in noisy environments, that is caused by the bias generated by the hard optimization of the policy.

Although we have focused on Q-learning as a baseline, we believe that early-stage information penalties can also be applied to advantage in more sophisticated model-free settings, such as TD(λ), and combined with other incremental learning techniques, such as function approximation, experience replay and actor-critic methods.

G-learning takes a Frequentist approach to estimating the optimal Q function. This is in contrast to Bayesian Q-learning [38], which explicitly models the uncertainty about the Q function as a posterior distribution. It would be interesting to study the bias that hard optimization causes in the mean of this posterior, and to consider its reduction using methods similar to G-learning.

An important next step is to apply G-learning to more challenging domains, where an approximation of the G function is necessary. The simplicity of our linear β schedule (26) should facilitate such extensions, and allow G-learning to be combined with other schemes and algorithms. Further study should also address the optimal schedule for β. We leave these important questions for future work.

Acknowledgments

AP is supported by ONR grant N00014-14-1-0243 and IARPA via DoI/IBC contract number D16PC00003. RF and NT are supported by the DARPA MSEE Program, the Gatsby Charitable Foundation, the Israel Science Foundation and the Intel ICRI-CI Institute.

References

[1] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[2] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

[3] Warren B Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

[4] Csaba Szepesvári. Algorithms for reinforcement learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 4(1):1–103, 2010.

[5] Volodymyr Mnih et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[6] Edward C Capen, Robert V Clapp, William M Campbell, et al. Competitive bidding in high-risk situations. *Journal of petroleum technology*, 23(06):641–653, 1971.

[7] Richard H Thaler. Anomalies: The winner's curse. *The Journal of Economic Perspectives*, pages 191–202, 1988.

[8] Eric Van den Steen. Rational overoptimism (and other biases). *American Economic Review*, pages 1141–1151, 2004.

[9] James E Smith and Robert L Winkler. The optimizer's curse: Skepticism and postdecision surprise in decision analysis. *Management Science*, 52(3):311–322, 2006.

[10] Jonathan Rubin, Ohad Shamir, and Naftali Tishby. Trading value and information in MDPs. In *Decision Making with Imperfect Decision Makers*, pages 57–74. Springer, 2012.

[11] Hado V Hasselt. Double Q-learning. In *NIPS*, 2010.

[12] Mohammad Ghavamzadeh, Hilbert J Kappen, Mohammad G Azar, and Rémi Munos. Speedy Q-learning. In *NIPS*, pages 2411–2419, 2011.

[13] Donghun Lee and Warren B Powell. An intelligent battery controller using bias-corrected Q-learning. In *AAAI*, 2012.

[14] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with Double Q-learning . In *AAAI*, 2016.

[15] Harm Van Seijen, Hado Van Hasselt, Shimon Whiteson, and Marco Wiering. A theoretical and empirical analysis of Expected Sarsa. In *ADPRL*, pages 177–184. IEEE, 2009.

[16] George H John. When the best move isn't optimal: Q-learning with exploration. In *AAAI*, page 1464, 1994.

[17] Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1,2. Athena Scientific Belmont, MA, 1995.

[18] Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the Fourth Connectionist Models Summer School*. Lawrence Erlbaum Publisher, Hillsdale, NJ, 1993.

[19] Ronen I Brafman and Moshe Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *JMLR*, 3:213–231, 2003.

[20] Edwin T Jaynes. *Probability theory: The logic of science*. Cambridge University Press, 2003.

[21] Lucian Busoniu, Robert Babuska, Bart De Schutter, and Damien Ernst. *Reinforcement learning and dynamic programming using function approximators*, volume 39. CRC press, 2010.

[22] Vivek S Borkar. Stochastic approximation. *Cambridge Books*, 2008.

[23] Leemon C Baird III. Reinforcement learning in continuous time: Advantage updating. In *IEEE International Conference on Neural Networks*, volume 4, pages 2448–2453. IEEE, 1994.

[24] Mance E Harmon, Leemon C Baird III, and A Harry Klopf. Advantage updating applied to a differential game. *NIPS 7*, 7:353, 1995.

[25] Marc G Bellemare, Georg Ostrovski, Arthur Guez, Philip S Thomas, and Rémi Munos. Increasing the action gap: New operators for reinforcement learning. In *AAAI*, 2016.

[26] Mark D Pendrith and C Sammut. On reinforcement learning of control actions in noisy and non-Markovian domains. Technical report, 1994.

[27] Mark D Pendrith and Malcolm RK Ryan. Estimator variance in reinforcement learning: Theoretical problems and practical solutions. In *AAAI Workshop on On-Line Search*, 1997.

[28] Álvaro Moreno, José D Martín, Emilio Soria, Rafael Magdalena, and Marcelino Martínez. Noisy reinforcements in reinforcement learning: some case studies based on gridworlds. In *Proceedings of the 6th WSEAS international conference on applied computer science*, pages 296–300, 2006.

[29] Emanuel Todorov. Linearly-solvable Markov decision problems. In *NIPS*, pages 1369–1376, 2006.

[30] Hilbert J Kappen, Vicenç Gómez, and Manfred Opper. Optimal control as a graphical model inference problem. *Machine learning*, 87(2):159–182, 2012.

[31] Emanuel Todorov. Efficient computation of optimal actions. *Proceedings of the national academy of sciences*, 106(28):11478–11483, 2009.

[32] Jan Peters, Katharina Mülling, and Yasemin Altun. Relative entropy policy search. In *AAAI*, 2010.

[33] Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. Approximate inference and stochastic optimal control. *arXiv preprint arXiv:1009.3958*, 2010.

[34] Mohammad Gheshlaghi Azar, Vicenç Gómez, and Hilbert J Kappen. Dynamic policy programming. *JMLR*, 13(1):3207–3245, 2012.

[35] Susanne Still and Doina Precup. An information-theoretic approach to curiosity-driven reinforcement learning. *Theory in Biosciences*, 131(3):139–148, 2012.

[36] Michel Tokic and Günther Palm. Value-difference based exploration: adaptive control between epsilon-greedy and softmax. In *KI 2011: Advances in Artificial Intelligence*, pages 335–346. Springer, 2011.

[37] Eyal Even-Dar and Yishay Mansour. Learning rates for Q-learning. *JMLR*, 5:1–25, 2004.

[38] Richard Dearden, Nir Friedman, and Stuart Russell. Bayesian Q-learning. In *AAAI/IAAI*, pages 761–768, 1998.

Taming the Noise in Reinforcement Learning via Soft Updates — Supplementary Material —

Roy Fox[*] Hebrew University

Ari Pakman[*] Columbia University

Naftali Tishby Hebrew University

CONVERGENCE OF G-LEARNING

In this section we prove the convergence of G to the optimal G^*, with probability 1, under the G-learning update rule

$$G(s_t,a_t) \leftarrow (1-\alpha_t)G(s_t,a_t) + \alpha_t\left(c_t - \tfrac{\gamma}{\beta}\log\left(\sum_{a'}\rho(a'|s_{t+1})e^{-\beta G(s_{t+1},a')}\right)\right). \tag{1}$$

Recall that the supremum norm is defined as $|x|_\infty = \max_i |x_i|$, and that the optimal G function satisfies

$$\begin{aligned} G^*(s,a) &= \mathrm{E}_\theta[c|s,a] - \tfrac{\gamma}{\beta}\,\mathrm{E}_p\left[\log\sum_{a'}\rho(a'|s')e^{-\beta G^*(s',a')}\right] \qquad (2)\\ &\equiv \mathbf{B}^*[G^*]_{(s,a)}. \qquad (3)\end{aligned}$$

The convergence proof relies on the following Lemma.

Lemma 1. *The operator* $\mathbf{B}^*[G]_{(s,a)}$ *defined in* (3) *is a contraction in the supremum norm.*

Proof. Let us define

$$\mathbf{B}^\pi[G]_{(s,a)} = k^\pi(s,a) + \gamma\sum_{s',a'}p(s'|s,a)\pi(a'|s')G(s',a'), \tag{4}$$

where

$$k^\pi(s,a) = \mathrm{E}_\theta[c|s,a] + \tfrac{\gamma}{\beta}\sum_{s',a'}p(s'|s,a)\pi(a'|s')\log\tfrac{\pi(a'|s')}{\rho(a'|s')}. \tag{5}$$

Now, for any policy π, the operator (4) is a contraction under the supremum norm [1], i.e. for any G_1 and G_2

$$|\mathbf{B}^\pi[G_1] - \mathbf{B}^\pi[G_2]|_\infty \le \gamma|G_1 - G_2|_\infty. \tag{6}$$

[*]These authors contributed equally to this work.

Also note that

$$\mathbf{B}^*[G_i]_{(s,a)} = \min_\pi \mathbf{B}^\pi[G_i]_{(s,a)}, \tag{7}$$

and that the optimum is achieved for

$$\pi_{G_i}(a|s) = \frac{\rho(a|s)e^{-\beta G_i(s,a)}}{\sum_{a'}\rho(a'|s)e^{-\beta G_i(s,a')}}. \tag{8}$$

The Lemma now follows from

$$\begin{aligned} &\left|\mathbf{B}^*[G_1] - \mathbf{B}^*[G_2]\right|_\infty \qquad (9)\\ &\quad= \max_{(s,a)}\left|\mathbf{B}^*[G_1]_{(s,a)} - \mathbf{B}^*[G_2]_{(s,a)}\right| \\ &\quad= \max_{(s,a)}\left|\mathbf{B}^{\pi_{G_1}}[G_1]_{(s,a)} - \mathbf{B}^{\pi_{G_2}}[G_2]_{(s,a)}\right| \\ &(\text{choose } i = \arg\min \mathbf{B}^{\pi_{G_i}}[G_i]_{(s,a)}) \\ &\quad\le \max_{(s,a)}\max_{i=1,2}\left|\mathbf{B}^{\pi_{G_i}}[G_1]_{(s,a)} - \mathbf{B}^{\pi_{G_i}}[G_2]_{(s,a)}\right| \\ &\quad= \max_{i=1,2}\left|\mathbf{B}^{\pi_{G_i}}[G_1] - \mathbf{B}^{\pi_{G_i}}[G_2]\right|_\infty \\ &\quad\le \gamma\left|G_1 - G_2\right|_\infty. \end{aligned}$$

□

The update equation (1) of the algorithm can be written as a stochastic iteration equation

$$G_{t+1}(s_t,a_t) = (1-\alpha_t)G_t(s_t,a_t) + \alpha_t(\mathbf{B}^*[G_t]_{(s_t,a_t)} + z_t(c_t,s_{t+1})) \tag{10}$$

where the random variable z_t is

$$z_t(c_t,s_{t+1}) \equiv -\mathbf{B}^*[G_t]_{(s_t,a_t)} + c_t - \tfrac{\gamma}{\beta}\log\sum_{a'}\rho(a'|s_{t+1})e^{-\beta G_t(s_{t+1},a')}. \tag{11}$$

Note that z_t has expectation 0. Many results exist for iterative equations of the type (10). In particular, given conditions

$$\sum_t \alpha_t = \infty; \qquad \sum_t \alpha_t^2 < \infty, \tag{12}$$

the contractive nature of $\mathbf{B}^*$, infinite visits to each pair (s_t,a_t) and assuming that $|z_t| < \infty$, G_t is guaranteed to converge to the optimal G^* with probability 1 [1, 2].

References

[1] Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1,2. Athena Scientific Belmont, MA, 1995.

[2] Vivek S Borkar. Stochastic approximation. *Cambridge Books*, 2008.