Full Gradient DQN Reinforcement Learning: A Provably Convergent Scheme

Konstantin Avrachenkov, Vivek S Borkar, Harsh P Dolhare, Kishor Patil

▶ **To cite this version:**

Konstantin Avrachenkov, Vivek S Borkar, Harsh P Dolhare, Kishor Patil. Full Gradient DQN Reinforcement Learning: A Provably Convergent Scheme. Alexey Piunovskiy; Yi Zhang. Modern Trends in Controlled Stochastic Processes: Theory and Applications, V.III, 41, Springer International Publishing, pp.192-220, 2021, Emergence, Complexity and Computation, 978-3-030-76928-4. 10.1007/978-3-030-76928-4_10 . hal-03462350

HAL Id: hal-03462350

https://inria.hal.science/hal-03462350

Submitted on 1 Dec 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Full Gradient DQN Reinforcement Learning: A Provably Convergent Scheme

K. Avrachenkov [1], V. S. Borkar[2], H. P. Dolhare[2], and K. Patil[1]

[1]INRIA Sophia Antipolis, France 06902*
[2]Indian Institute of Technology, Bombay, India 400076

Abstract

We analyze the DQN reinforcement learning algorithm as a stochastic approximation scheme using the o.d.e. (for *'ordinary differential equation'*) approach and point out certain theoretical issues. We then propose a modified scheme called Full Gradient DQN (FG-DQN, for short) that has a sound theoretical basis and compare it with the original scheme on sample problems. We observe a better performance for FG-DQN.

Keywords: Markov decision process (MDP); approximate dynamic programming; deep reinforcement learning (DRL); stochastic approximation; Deep Q-Network (DQN); Full Gradient DQN; Bellman error minimization.

AMS 2000 subject classification: 93E35; 68T05

1 Introduction

Recently we have witnessed tremendous success of Deep Reinforcement Learning algorithms in various application domains. Just to name a few examples, DRL has achieved superhuman performance in playing Go [37], Chess [38] and many Atari video games [29, 30]. In Chess, DRL algorithms have also beaten the state of the art computer programs, which are based on more or less brute-force enumeration of moves. Moreover, playing Go and Chess, DRL surprised experts with new insights and beautiful strategies [37, 38]. We would also like to mention the impressive progress of DRL applications in robotics [21, 22, 31], telecommunications [27, 33, 47] and medicine [24, 32].

The use of Deep Neural Networks is of course an essential part of DRL. However, there are other paramount elements that contributed to the success of DRL. A starting point for DRL was the Q-learning algorithm of Watkins [45], which in its original form can suffer from the proverbial curse of dimensionality. In [23], [41] the convergence of Q-learning has been rigorously established. Then, in [19, 20] Gordon has proposed and analyzed fitted Q-learning using a novel architecture based on what he calls 'averager' maps. In [35] Riedmiller has proposed using a neural network for approximating Q-values. There he has also suggested that we treat the

*email: k.avrachenkov@inria.fr, borkar.vs@gmail.com, harshdolhare99@gmail.com, kishor88k@gmail.com.

right hand side of the dynamic programming equation for Q-values (see equation (5) below) as the 'target' to be chased by the left hand side, i.e., the Q-value itself, and then seek to minimize the mean squared error between the two. The right hand side in question also involves the Q-value approximation and *ipso facto* the parameter itself, which is treated as a 'given' for this purpose, as a part of the target, and the minimization is carried out only over the same parameter appearing in the left hand side. This leads to a scheme reminiscent of temporal difference learning, albeit a nonlinear variant of it. The parameter dependence of the target leads to some difficulties because of the permanent shifting of the target itself what one might call the 'dog chasing its own tail' phenomenon. Already in [35], frequent instability of the algorithm has been reported.

The next big step in improvement of DRL performance was carried out by DeepMind researchers, who elaborated the Deep Q-Network (DQN) scheme [29], [30]. Firstly, to improve the stability of the algortihm in [35], they suggested freezing the parameter value in the target network for several iterates. Thus in DQN, the target network evolves on a slower timescale. The second successful tweak for DQN has been the use of 'experience replay', or averaging over some relevant traces from the past, a notion introduced in [25, 26]. Then, in [43, 44] it was suggested that we introduce a separation of policy estimation and evaluation to further improve stability. The latter scheme is called Double DQN. While various success stories of DQN and Double DQN schemes have been reported, this does not completely fix the theoretical and practical issues.

Let us mention that apart from Q-value based methods in DRL, there is another large family of methods based on policy gradient. Each family has its own positive and negative features (for background on RL and DRL methods we recommend the texts [39, 6, 18]). While there has been a notable progress in the theoretical analysis of the policy gradient methods [28, 40, 7, 11, 1, 2], there are no works establishing convergence of the neural Q-value based methods to the best of our knowledge.

In this work, we revisit DQN and scrutinize it as a stochastic approximation algorithm, using the 'o.d.e.' (for 'ordinary differential equation') approach for its convergence analysis (see [9] for a textbook treatment). In fact, we go beyond the basic o.d.e. approach to its generalization based on differential inclusions, involving in particular non-smooth analysis. This clarifies the underlying difficulties regarding theoretical guarantees of convergence and also suggests a modification, which we call the Full Gradient DQN, or FG-DQN. We establish theoretical convergence guarantees for FG-DQN and compare it empirically with DQN on sample problems (forest management [12, 14] and cartpole [4, 17]), where it gives better performance at the expense of some additional computational overhead per iteration.

As was noticed above, another successful tweak for DQN has been the use of 'experience replay'. We too incorporate this in our scheme. Many advantages of experience replay have been cited in literature, which we review later in this article. We also unearth an interesting additional advantage of 'experience replay' for Bellman error minimization using gradient descent.

2 DQN reinforcement learning

2.1 Q-learning

We begin by recalling the derivation of the original Q-learning scheme [45] to set up the context. Consider a Markov chain $\{X_n\}$ on a finite state space $S := \{1, 2, \cdots, s\}$, controlled by a control process $\{U_n\}$ taking values in a finite action space $A = \{1, 2, \cdots, a\}$. Its transition probability function is denoted by $(x, y, u) \in S^2 \times A \mapsto p(y|x, u) \in [0, 1]$ such that $\sum_y p(y|x, u) = 1 \ \forall \ x, u$. The controlled Markov property then is

$$P(X_{n+1} = y|X_m, U_m, m \leq n) = p(y|X_n, U_n) \quad \forall \ n \geq 0, \ y \in S.$$

We call $\{U_n\}$ an admissible control policy. It is called a stationary policy if $U_n = v(X_n) \ \forall n$ for some $v : S \to A$. A more general notion is that of a stationary randomized policy wherein one chooses the control U_n at time n probabilistically with a conditional law given the σ-field $\mathcal{F}_n := \sigma(X_m, U_m, m < n; X_n)$ that depends only on X_n. That is,

$$\varphi(u|X_n) := P(U_n = u|\mathcal{F}_n) = P(U_n = u|X_n)$$

for a prescribed map $x \in S \mapsto \varphi(\cdot|x) \in \mathcal{P}(A) :=$ the simplex of probability vectors on A. One identifies such a policy with the map φ. Denote the set of stationary randomized policies by $\mathcal{U}_{SR}$. In anticipation of the learning schemes we discuss, we impose the 'frequent updates' or 'sufficient exploration' condition

$$\liminf_{n\uparrow\infty} \frac{1}{n} \sum_{m=0}^{n-1} I\{X_m = x, U_m = u\} > 0 \quad \text{a.s.} \quad \forall x, u. \tag{1}$$

Given a per stage reward $(x, u) \mapsto r(x, u)$ and a discount factor $\gamma \in (0, 1)$, the objective is to maximize the infinite horizon expected discounted reward

$$E\left[\sum_{m=0}^{\infty} \gamma^m r(X_m, U_m)\right].$$

The 'value function' $V : S \to \mathcal{R}$ defined as

$$V(x) = \max E\left[\sum_{m=0}^{\infty} \gamma^m r(X_m, U_m)\Big|X_0 = x\right], \quad x \in S, \tag{2}$$

then satisfies the dynamic programming equation

$$V(x) = \max_u \left[r(x, u) + \gamma \sum_y p(y|x, u)V(y)\right], \quad x \in S. \tag{3}$$

Furthermore, the maximizer $v^*(x)$ on the right (chosen arbitrarily if not unique) defines a stationary policy $v^* : S \to A$ that is optimal, i.e., achieves the maximum in (2). Equation (3) is a fixed point equation of the form $V = F(V)$ (which defines the map $F : \mathcal{R}^s \to \mathcal{R}^s$) and can be solved by the 'value iteration' algorithm

$$V_{n+1}(x) = \max_u \left[r(x, u) + \gamma \sum_y p(y|x, u)V_n(y)\right], \quad n \geq 0, \tag{4}$$

beginning with any $V_0 \in \mathcal{R}^s$. F can be shown to satisfy

$$\|F(x) - F(y)\|_\infty \leq \gamma \|x - y\|_\infty,$$

i.e., it is an $\|\cdot\|_\infty$-norm contraction. Then (4) is a standard fixed point iteration of a contraction map and converges exponentially to its unique fixed point V.

Now define Q-values as the expression in square brackets in (3), i.e.,

$$Q(x,u) = r(x,u) + \gamma \sum_y p(y|x,u) V(y), \quad x \in S,\ u \in A.$$

If the function $Q(\cdot,\cdot)$ is known, then the optimal control at state x is found by simply minimizing $Q(x,\cdot)$ without requiring the knowldge of reward or transition probabilities. This makes it suitable for data-driven algorithms of reinforcement learning. By (3), $V(x) = \max_u Q(x,u)$. The Q-values then satisfy their own dynamic programming equation

$$Q(x,u) = r(x,u) + \gamma \sum_y p(y|x,u) \max_v Q(y,v), \tag{5}$$

which in turn can be solved by the 'Q-value iteration'

$$Q_{n+1}(x,u) = r(x,u) + \gamma \sum_y p(y|x,u) \max_v Q_n(y,v), \quad x \in S,\ u \in A. \tag{6}$$

What we have gained at the expense of increased dimensionality is that the nonlinearity is now inside the conditional expectation w.r.t. the transition probability function. This facilitates a stochastic approximation algorithm [9] where we first replace this conditional expectation by actual evaluation at a real or simulated random variable $\zeta_{n+1}(x,u)$ with law $p(\cdot|x,u)$, and then make an incremental correction to the current guess based on it. That is, replace (6) by

$$Q_{n+1}(x,u) = (1 - a(n)) Q_n(x,u) + a(n) \left(r(x,u) + \gamma \max_v Q_n(\zeta_{n+1}(x,v), v) \right) \tag{7}$$

for some $a(n) > 0$. The Q-learning algorithm does so using a single run of a real or simulated controlled Markov chain $(X_n, U_n), n \geq 0$, so that:

- at each time instant n, (X_n, U_n) are observed and the (X_n, U_n)th component of Q is updated, leaving other components of $Q_n(\cdot,\cdot)$ unchanged,
- this update follows (7) where $\zeta_{n+1}(x,u)$ with $x = X_n, u = U_n$, gets replaced by X_{n+1}, which indeed has the conditional law $p(\cdot|X_n, U_n)$ as required,
- $\{a(n)\}$ are positive scalars in $(0,1)$ chosen to satisfy the standard Robbins-Monro conditions of stochastic approximation [9], i.e.,

$$\sum_n a(n) = \infty, \quad \sum_n a(n)^2 < \infty. \tag{8}$$

It is more convenient to write the resulting Q-learning algorithm as

$$\begin{aligned} Q_{n+1}(x,u) \quad = \quad & Q_n(x,u) + a(n) I\{X_n = x, U_n = u\} \Big(r(x,u) + \\ & \gamma \max_v Q_n(X_{n+1}, v) - Q_n(x,u) \Big) \quad \forall\ x, u, \end{aligned} \tag{9}$$

where $I\{\cdots\}$:= the indicator random variable that equals 1 if '$\cdots$' holds and 0 if not. The fact that only one component is being updated at a time makes this an asynchronous stochastic approximation. Nevertheless, it exhibits the well known 'averaging effect' of stochastic approximation whereby it is a data-driven scheme that emulates (6) and exhibits convergence a.s. to the same limit, viz., Q. For formal proofs, see [23, 41, 46].

2.2 DQN learning

The raw Q-learning scheme (9), however, does inherit the 'curse of dimensionality' of MDPs. One common fix is to replace Q by a parametrized family $(x, u, \theta) \mapsto Q(x, u; \theta)$ (where we again use the notation $Q(\cdot, \cdot; \cdot)$ by abuse of terminology so as to match standard usage). Here $\theta \in \Theta \subset \mathcal{R}^d$ for a moderate $d \geq 1$ and the objective is to learn the 'optimal' approximation $Q(\cdot, \cdot; \theta^*)$ by iterating in Θ. For simplicity, we take $\Theta = \mathcal{R}^d$. One natural performance measure is the 'empirical Bellman error'

$$\mathcal{E}(\theta) := E\left[\left(Z_n - Q(X_n, U_n; \theta)\right)^2\right], \tag{10}$$

where

$$Z_n := r(X_n, U_n) + \gamma \max_v Q(X_{n+1}, v; \theta_n)$$

is the 'target' that is taken as a *given quantity* and expectation is w.r.t. the stationary law of (X_n, U_n). For later reference, note that this is different from the 'true Bellman error'

$$\bar{\mathcal{E}}(\theta) := E\left[\left(r(X_n, U_n) + \gamma \sum\nolimits_y p(y|X_n, U_n) \max_v Q(y, v; \theta_n) - Q(X_n, U_n; \theta)\right)^2\right]. \tag{11}$$

The stochastic gradient type scheme based on the empirical semi-gradient of $\mathcal{E}(\cdot)$ then becomes

$$\theta_{n+1} = \theta_n + a(n)(Z_n - Q(X_n, U_n; \theta_n))\nabla_\theta Q(X_n, U_n; \theta_n), \quad n \geq 0. \tag{12}$$

2.3 Experience replay

An important modification of the DQN scheme has been the incorporation of 'experience replay'. The idea is to replace the term multiplying $a(n)$ on the right hand side of (12) by an empirical average over traces of transitions from past that are stored in memory. The algorithm then becomes

$$\begin{aligned}\theta_{n+1} = \theta_n + \frac{a(n)}{M} \times \\ \sum_{m=1}^{M} \left((Z_{n(m)} - Q(X_{n(m)}, U_{n(m)}))\nabla_\theta Q(X_{n(m)}, U_{n(m)}; \theta_{n(m)}) \right), \ n \geq 0,\end{aligned} \tag{13}$$

where $(X_{n(m)}, U_{n(m)}), 1 \leq m \leq N$, are samples from past. This has multiple advantages. Some that have been cited in literature are as follows.

1. As in the mini-batch stochastic gradient descent for empirical risk minimization in machine learning, it helps reduce variance. It also diminishes effects of anomalous transitions.

2. Training based on only the immediate experiences ($\approx$ samples) tends to overfit the model to current data. This is prevented by experience replay. In particular, if past samples are randomly picked, they are less correlated.

3. The re-use of data leads to data efficiency.

4. Experience replay is better suited for delayed rewards or costs, e.g., when the latter are realized only at the end of a long episode or epoch.

There are also variants of basic experience replay, e.g., [36], which replaces purely random sampling from past by a non-uniform sampling which picks a sample with probability proportional to its absolute Bellman error.

We shall be implementing experience replay a little differently in the variant we describe next, which has yet another major advantage from a theoretical standpoint in the specific context of our scheme.

2.4 Double DQN learning

One more modification of the vanilla DQN scheme is doing the policy selection according the local network [43, 44]. The target network is still used in Z_n and is updated on a slower time scale. The latter can be represented with another set of parameters $\bar{\theta}_n$. Thus, the iterate for the Double DQN scheme can be written as follows:

$$\theta_{n+1} = \theta_n + a(n)(Z_n - Q(X_n, U_n; \theta_n))\nabla_\theta Q(X_n, U_n; \theta_n), \quad n \geq 0, \tag{14}$$

with

$$Z_n := r(X_n, U_n) + \gamma Q(X_{n+1}, v; \bar{\theta}_n)\Big|_{v=\text{argmax}_{v'} Q(X_{n+1}, v'; \theta_n)}.$$

For the sake of comparison, in the vanilla DQN one has:

$$Z_n := r(X_n, U_n) + \gamma Q(X_{n+1}, v; \bar{\theta}_n)\Big|_{v=\text{argmax}_{v'} Q(X_{n+1}, v'; \bar{\theta}_n)}.$$

Note that in Double DQN, the selection and evaluation of the policy is done separately. According to [43, 44] this modification improves the stability of the DQN learning. One can also combine Double DQN with experience replay [44].

3 The issues with DQN learning

The expression for DQN learning scheme is appealing because of its apparent similarity with the very successful temporal difference learning for policy evaluation, not to mention its empirical successes, including some high profile ones such as [30]. Nevertheless, a good theoretical justification seems lacking. The difficulty arises from the fact that the 'target' Z_n is not something extraneous, but also a function of the operative parameter θ_n. In fact, this becomes apparent once we expand Z_n in (12) to write

$$\begin{aligned} \theta_{n+1} &= \theta_n + a(n)(r(X_n, U_n) + \gamma \max_v Q(X_{n+1}, v; \theta_n) - Q(X_n, U_n; \theta_n)) \times \\ &\quad \nabla_\theta Q(X_n, U_n; \theta_n), \ n \geq 0. \end{aligned} \tag{15}$$

Write

$$\tilde{\mathcal{E}}(\theta,\bar{\theta}) := E\left[\left(r(X_n,U_n)+\gamma\max_v Q(X_{n+1},v;\bar{\theta})-Q(X_n,U_n;\theta)\right)^2\right], \tag{16}$$

where $E[\cdot]$ is the stationary expectation as before. Consider the 'off-policy' case, i.e., $\{(X_n,U_n)\}$ is the state-action sequence of a controlled Markov chain satisfying (1) with a pre-specified stationary randomized policy that does not depend on the iterates. (As we point out later, the 'on-policy' version, which allows for the latter adaptation, has additional issues.) If we apply the 'o.d.e. approach' for analysis of stochastic approximation (see, e.g., [9] for a textbook treatment), we get the limiting o.d.e. as

$$\dot{\theta}(t) = -\nabla_1\tilde{\mathcal{E}}(\theta(t),\theta(t)),$$

where ∇_i denotes gradient with respect to the ith argument of $\tilde{\mathcal{E}}(\cdot,\cdot)$ for $i=1,2$. Thus it is a partial stochastic gradient descent wherein only the gradient with respect to the first occurrence of the variable is used. Unlike gradient dynamics, there is no reason why such dynamics should converge. It was already mentioned that in case of linear function approximation, the DQN iteration bears a similarity with TD(0), except for the nonlinear 'max' term. The o.d.e. proof of convergence for TD(0) does not carry over to DQN precisely because the stochastic approximation version leads to the interchange of the conditional expectation and max operators. The other issue is that in TD(0), the linear operator in question is a contraction w.r.t. the weighted L_2-norm weighted by the stationary distribution. That argument also fails for DQN because of presence of the max operator.

That said, there is already a tweak that treats the first occurrence of θ on the RHS, i.e., that inside the maximizer, as the 'target' being followed, and updates it only after several (say, K) iterates. In principle, this implies a delay in the corresponding input to the iteration and with decreasing stepsizes, introduces only an asymptotically negligible additional error, so that the limiting o.d.e. remains the same ([9], Chapter 6). This is also the case for Double DQN.

Suppose on the other hand that in DQN or Double DQN we consider a small constant stepsize $a(n)\equiv a>0$ and let K be large, so that with a fixed target value, the algorithm nearly minimizes the Bellman error before the target is updated. Then, assuming the simpler 'off-policy' case again, the limiting o.d.e. *for the target*, treating the multiple iterates between its successive iterates as a subroutine, is

$$\dot{\theta}(t) = -\nabla_1\tilde{\mathcal{E}}(x,\theta(t))\Big|_{x=\text{argmax}(\tilde{\mathcal{E}}(\cdot,\theta(t)))}. \tag{17}$$

There is no obvious reason why this should converge either. In fact the right hand side would be $\approx$ the zero vector near the current maximizer and the evolution of the o.d.e. and the iteration would be very slow. Of course, this is a limiting case of academic interest only, stated to underscore the fact that it is difficult to get convergent dynamics out of the DQN learning scheme. This motivates our modification, which we state in the next subsection.

4 Full Gradient DQN

We propose the obvious, viz., to treat both occurrences of the variable θ on equal footing, i.e., treat it as a single variable, and then take the full gradient with respect to it. The iteration now

is

$$\begin{aligned}\theta_{n+1} &= \theta_n - a(n)\left(r(X_n,U_n)+\gamma\max_v Q(X_{n+1},v;\theta_n)-Q(X_n,U_n;\theta_n)\right)\times \\ &\quad (\gamma\nabla_\theta Q(X_{n+1},v_n;\theta_n)-\nabla_\theta Q(X_n,U_n;\theta_n)) \end{aligned} \tag{18}$$

for $n \geq 0$, where $v_n \in \text{Argmax}Q(X_{n+1},\cdot;\theta_n)$ chosen according to some tie-breaking rule when necessary. Note that when the maximizer in the term involving the max operator is not unique, one may lose its differentiability, but the expression above still makes sense in terms of the Frechet sub-differential, see Appendix. We assume throughout that $\{X_n\}$ is a Markov chain controlled by the control process $\{U_n\}$ generated according to a fixed stationary randomized policy $\varphi \in \mathcal{U}_{SR}$. Other simulation scenarios are possible for the off-policy set-up. For example, we may replace the triplets (X_n,U_n,X_{n+1}) on the right hand side by triplets (X'_n,U'_n,Y'_n) where $\{X'_n\}$ are generated i.i.d. according to some distribution with full support and (U'_n,Y'_n) are generated with conditional law $P(U'_n=u,Y'_n=y|X'_n=x)=\varphi(u|x)p(y|x,u)$, conditionally independent of all other random variables generated till n given X'_n. The analysis will be similar. Yet another possibility is that of going through the relevant triplets (x,y,u) in a round robin fashion.

We modify (18) further by replacing the right hand side as follows:

$$\begin{aligned}\theta_{n+1} &= \theta_n - a(n)\Bigg(\overline{(r(X_n,U_n)+\gamma\max_v Q(X_{n+1},v;\theta_n)-Q(X_n,U_n;\theta_n))}\times \\ &\quad (\gamma\nabla_\theta Q(X_{n+1},v_n;\theta_n)-\nabla_\theta Q(X_n,U_n;\theta_n))+\xi_{n+1}\Bigg)\end{aligned} \tag{19}$$

for $n \geq 0$, where $\{\xi_n\}$ is extraneous i.i.d. noise componentwise distributed independently and uniformly on $[-1,1]$, and the overline stands for a modified form of experience replay which comprises of averaging at time n over past traces sampled from $(X_k,U_k,X_{k+1}), k \leq n$, for which $X_k = X_n, U_k = U_n$. We analyze the asymptotic behavior of this scheme in the remainder of this section in the 'off-policy' case, i.e., we use a prescribed stationary randomized policy $\varphi \in \mathcal{U}_{SR}$.

We make the following key assumptions:

(C1) (Assumptions regarding the function $Q(\cdot,\cdot;\cdot)$)

1. The map $(x,u;\theta) \mapsto Q(x,u;\theta)$ is bounded and twice continuously differentiable in θ with bounded first and second derivatives;
2. For each choice of $x \in S$, the set of θ for which the maximizer of $Q(x,\cdot;\theta)$ is not unique, is open and dense, in particular has Lebesgue measure zero;
3. Call $\hat{\theta}$ a critical point of $\mathcal{E}(\cdot)$ (which is defined in terms of Q) if the zero vector is contained in the (Frechet) subdifferential $\partial^-\mathcal{E}(\hat{\theta})$ (see the Appendix for a definition). We assume that there are at most finitely many such points.

We also assume:

(C2) (Stability assumption)

The iterates remain a.s. bounded, i.e.,

$$\sup_n \|\theta_n\| < \infty \text{ a.s.} \tag{20}$$

Our final assumption is a bit more technical. Rewrite the term

$$\overline{(r(X_n, U_n) + \gamma \max_v Q(X_{n+1}, v; \theta_n) - Q(X_n, U_n; \theta_n))}$$

as

$$\sum_y p(y|X_n, U_n) \left(r(X_n, U_n) + \gamma \max_v Q(y, v; \theta_n) - Q(X_n, U_n; \theta_n) \right) + \varepsilon(X_n, U_n, \theta_n)$$

where the error term $\varepsilon(X_n, U_n, \theta_n)$ captures the difference between the empirical conditional expectation using experience replay and the actual conditional expectation. We assume that:

(C3) (Assumption regarding the residual error in experience replay)

The error terms $\{\varepsilon(X_n, U_n, \theta_n)\}$ satisfy

$$\varepsilon(X_n, U_n, \theta_n) \to 0 \text{ a.s. and } \sum_n a(n) E[|\varepsilon(X_n, U_n, \theta)|]|_{\theta=\theta_n} < \infty \text{ a.s.},$$

where the expectation is taken w.r.t. the stationary distribution of the state-action pairs.

We comment on these assumptions later. Recall the true Bellman error $\bar{\mathcal{E}}(\cdot)$ defined in (11).

Theorem 1 The sequence $\{\theta_n\}$ generated by FG-DQN converges a.s. to a sample path dependent critical point of $\bar{\mathcal{E}}(\cdot)$.

Proof: For notational ease, write

$$\epsilon(n) := -\varepsilon(X_n, U_n, \theta_n) \left(\gamma \nabla_\theta Q(X_{n+1}, v_n; \theta_n) - \nabla_\theta Q(X_n, U_n; \theta_n) \right),$$

where v_n is chosen from Argmax $Q(X_{n+1}, \cdot; \theta_n)$ as described earlier. Consider the iteration

$$\begin{aligned} \theta_{n+1} &= \theta_n - a(n) \times \\ &\left(\left(\sum_y p(y|X_n, U_n)(r(X_n, U_n) + \gamma \max_v Q(y, v; \theta_n) - Q(X_n, U_n; \theta_n)) \right) \times \right. \\ &\left. (\gamma \nabla_\theta Q(X_{n+1}, v_n; \theta_n) - \nabla_\theta Q(X_n, U_n; \theta_n)) + \epsilon(n) + \xi_{n+1} \right) \end{aligned} \tag{21}$$

for $n \geq 0$. Adding and subtracting the one step conditional expectation of the RHS with respect to $\mathcal{F}_n$, we have

$$\begin{aligned}\theta_{n+1} &= \theta_n - a(n) \times \\ &\left(\sum_y p(y|X_n,U_n)(r(X_n,U_n) + \gamma \max_v Q(y,v;\theta_n) - Q(X_n,U_n;\theta_n))\right) \\ &\times \left(\sum_y p(y|X_n,U_n)\left(\gamma \nabla_\theta Q(y,u_n(y);\theta_n) - \nabla_\theta Q(X_n,U_n;\theta_n)\right)\right) \\ &+ a(n)\epsilon(n) + \ a(n) M_{n+1}(\theta_n) \end{aligned} \tag{22}$$

where $u_n(y) \in \text{Argmax}\ Q(y,\cdot;\theta_n)$ is chosen as described earlier, and $\{M_n(\theta_{n-1})\}$ is a martingale difference sequence w.r.t. the sigma fields $\{\mathcal{F}'_n := \sigma(X_m, U_m, m \leq n), n \geq 0\}$, given by

$$\begin{aligned}&M_{n+1}(\theta_n) = \\ &\left(\Big(\sum_y p(y|X_n,U_n)(r(X_n,U_n) + \gamma \max_v Q(y,v;\theta_n) - Q(X_n,U_n;\theta_n))\Big)\right. \\ &\left.\times \left(\gamma \nabla_\theta Q(X_{n+1},v_n;\theta_n) - \sum_y p(y|X_n,U_n)\gamma \nabla_\theta Q(y,u_n(y);\theta_n)\right) + \xi_{n+1}\right).\end{aligned}$$

Because of our assumptions on $Q(\cdot,\cdot;\cdot)$ and $\{\xi_n\}$, $M_n(\cdot)$ will have derivatives uniformly bounded in n and therefore a uniform linear growth w.r.t. θ. The same holds for the expression multiplying $a(n)$ in the first term on the right. We shall analyze this iteration as a stochastic approximation with Markov noise $(X_n, U_n), n \geq 0$, and martingale difference noise $M_{n+1}, n \geq 0$ ([9], Chapter 6).

The difficult terms are those of the form $\gamma \nabla_\theta Q(y,u;\theta)$ above, because all we can say about them is that :

$$\begin{aligned}&\nabla_\theta Q(y,u;\theta) \ \in \ G(y,\theta) := \\ &\left\{\sum_v \psi(v|y) \nabla_\theta Q(y,v;\theta) : \psi(\cdot|y) \in \ \text{Argmax}_{\phi(\cdot|y)} \left(\sum_u \phi(u|y) Q(y,u;\theta)\right)\right\}.\end{aligned}$$

Define correspondingly the set-valued map

$$(x,u,\theta) \mapsto H(x,u,\theta)$$

by

$$\begin{aligned}H(x,u;\theta) &:= \overline{co}\left(\left\{\Big(\sum_y p(y|x,u)(r(x,u) + \gamma \max_v Q(y,v;\theta) - Q(x,u;\theta))\Big)\right.\right. \\ &\left.\left.\times \sum_y p(y|x,u)\left(\gamma \nabla_\theta Q(y,v_j;\theta) - \nabla_\theta Q(x,u;\theta)\right) : v_j \in \ \text{Argmax} Q(y,\cdot;\theta)\right\}\right) \\ &= \left\{\Big(\sum_y p(y|x,u)(r(x,u) + \gamma \max_v Q(y,v;\theta) - Q(x,u;\theta))\Big)\right.\end{aligned}$$

$$\times \sum_y p(y|x,u)\left(\gamma \nabla_\theta \bar{Q}(y,\pi_y;\theta) - \nabla_\theta Q(x,u;\theta)\right) : \pi_y \in \ \mathrm{Argmax} \bar{Q}(y,\cdot;\theta)\Bigg\}$$

where $\bar{Q}(y,\psi;\theta) := \sum_u \psi(u|y) Q(y,u;\theta)$ for $\psi \in \mathcal{U}_{SR}$. Then (22) can be written in the more convenient form as the stochastic recursive inclusion ([9], Chapter 5) given by

$$\theta_{n+1} \ \in \ \theta_n - a(n)\Bigg(H(X_n,U_n;\theta_n) + \epsilon(n) + M_{n+1}(\theta_n)\Bigg). \tag{23}$$

We shall now use Theorem 7.1 of [48], pp. 355, for which we need to verify the assumptions (A1)-(A5), pp. 331-2, therein. We do this next.

- (A1) requires $H(y,\phi,\theta)$ to be nonempty convex compact valued and upper semicontinuous, which is easily verified. It is also bounded by our assumptions on $Q(\cdot,\cdot;\cdot)$.
- S_n of [48] corresponds to our (X_n,U_n) and (A2) can be verified easily.
- (A3) are the standard conditions on $\{a(n)\}$ also used here.
- $M_{n+1}(\theta_n), n \geq 0$, defined above, has linear growth in $\|\theta_n\|$ as observed above. Thus (20) implies that

  $$\sum_{m=0}^{n} a(m)^2 E\left[\|M_{m+1}(\theta_m)\|^2 | \mathcal{F}_m\right] \leq K(1+\sup_m \|\theta_m\|^2) \sum_m a(n)^2 < \infty \text{ a.s.}$$

  This implies that $\sum_{m=0}^{n-1} a(m) M_{m+1}(\theta_m)$ is an a.s. convergent martingale by Theorem 3.3.4, pp. 53-4, [8]. This verifies (A4).
- (A5) is the same as (20) above.

Let $\mu(x,u) :=$ the stationary probability $P(X_n = x, U_n = u)$ under φ. Then Theorem 7.1 of [48] applies and allows us to conclude that the iterates will track the asymptotic behavior of the differential inclusion

$$\dot{\theta}(t) \in -\sum_{x,u} \mu(x,u) H(x,u,\theta(t)). \tag{24}$$

Now we make the important observation that under our hypotheses on the function $Q(\cdot,\cdot;\cdot)$ (see 3. of (C1)), for all x,u and Lebesgue-a.e. θ belonging to some open dense set O, $H(x,u,\theta)$ is the singleton corresponding to Argmax $Q(x,\cdot;\theta) = \{u\}$ for some $u \in A$. Furthermore, in this case, the RHS of (24) reduces to $-\nabla \mathcal{E}(\theta(t))$. Since $\{\xi_n\}$ has density w.r.t. the Lebesgue measure, so will $\{\theta_n\}$ and therefore by (C1), $\theta_n \in O \ \forall n$, a.s. Let

$$L(x,u;\theta) := \frac{1}{2}\left(r(x,u) + \gamma \sum_y p(y|x,u) \max_v Q(y,v;\theta) - Q(x,u;\theta)\right)^2$$

denote the instantaneous Bellman error. Then

$$\bar{\mathcal{E}}(\theta) = \sum_{x,u} \mu(x,u) L(x,u;\theta).$$

Write $\hat{\mathcal{E}}(\theta')$ for $\bar{\mathcal{E}}(\theta)$ evaluated at a possibly random θ', in order to emphasize the fact thet while $\bar{\mathcal{E}}(\cdot)$ is defined in terms of an expectation, a random argument of $\hat{\mathcal{E}}(\cdot)$ is not being averaged over. We use an analogous notation for other quantities in what follows. Applying the Taylor formula to $\bar{\mathcal{E}}(\cdot)$, we have,

$$\hat{\mathcal{E}}(\theta_{n+1}) = \hat{\mathcal{E}}(\theta_n) + \sum_{x,u} \mu(x,u)\langle \nabla_\theta L(x,u;\theta), \theta_{n+1} - \theta_n \rangle + O(a(n)^2).$$

But by (22), a.s.,

$$\begin{aligned}
\theta_{n+1} - \theta_n &= a(n)\Big(- \nabla_\theta L(X_n, U_n; \theta_n) + \epsilon(n) + M_{n+1}(\theta_n) \Big) \\
&= a(n)\Bigg(- \sum_{x,u} \mu(x,u) \nabla_\theta L(x,u;\theta_n) + \epsilon(n) + \widetilde{M}_{n+1}(\theta_n) + O(a(n)^2) \Bigg),
\end{aligned}$$

where we have replaced $\nabla_\theta L(X_n, U_n; \theta_n)$ with $\sum_{i,u} \mu(i,u) \nabla_\theta L(i,u;\theta_n)$, i.e., with the state-action process (X_n, Z_n) averaged w.r.t. its stationary distribution (recall that under our randomized stationary Markov policy, it is a Markov chain). This uses a standard (though lengthy) argument for stochastic approximation with Markov noise that converts it to a stochastic approximation with martingale difference noise using the associated parametrized Poisson equation, at the expense of: (i) adding an additional martingale difference noise term that we have added to $M_{n+1}(\theta_n)$ to obtain the combined martingale difference noise $\widetilde{M}_{n+1}(\theta_n)$, and, (ii) another $O(a(n)^2)$ term that comes from the difference of the solution of the Poisson equation evaluated at θ_n and θ_{n+1}, which is $O(\|\theta_{n+1} - \theta_n\|) = O(a(n))$, multiplied further by an additional $a(n)$ from (22) to give a net error that is $O(a(n)^2)$. See [5] for a classical treatment of this passage.

Hence for suitable constants $0 < K_1, K_1' < \infty$,

$$\begin{aligned}
E[\hat{\mathcal{E}}(\theta_{n+1}) | \mathcal{F}_n'] &\leq \hat{\mathcal{E}}(\theta_n) + a(n)\Bigg(- \| \sum_{x,u} \mu(x,u) \nabla_\theta L(x,u;\theta_n) \|^2 \\
&\qquad + K_1 \sum_{x,u} \mu(x,u) |\varepsilon(x,u,\theta_n)| + K_2 a(n)^2 \Bigg) \\
&\leq \hat{\mathcal{E}}(\theta_n) + a(n)\Bigg(K_1 \sum_{x,u} \mu(x,u) |\varepsilon(x,u,\theta_n)| + K_2 a(n)^2 \Bigg), \qquad (25)
\end{aligned}$$

where we have used (C1). In view of (C3) and the fact $\sum_n a(n)^2 < \infty$, the 'almost supermartingale' convergence theorem (Theorem 3.3.6, p. 54, [8]) implies that $\hat{\mathcal{E}}(\theta_n)$ converges a.s. This is possible only if

$$\begin{aligned}
\theta_n &\to \left\{ \theta : \text{the zero vector is in } \sum_{x,u} \mu(x,u) H(x,u;\theta) \right\} \\
&= \left\{ \theta : \theta \text{ is a critical point of } \sum_{x,u} \mu(x,u) H(x,u;\theta) \right\}.
\end{aligned}$$

By property (P4) of the Appendix, it follows that $H(i,u;\theta) \subset \partial^- L(i,u;\theta)$. By property (P3) of the Appendix, it then follows that $\sum_{i,u} \mu(i,u) H(i,u;\theta) \subset \partial^- \bar{\mathcal{E}}(\theta)$. The claim follows from

item 3. in (C1) given that any limit point of θ_n as $n \uparrow \infty$ must be a critical point of $\partial^- \bar{\mathcal{E}}(\cdot)$ in view of the foregoing. □

Some comments regarding our assumptions are in order.

1. The vanilla Q-learning iterates, being convex combinations of previous iterates with a bounded quantity, remain bounded. Thus the boundedness assumption on Q in (C1) is reasonable. The twice continuous differentiability of Q in θ is reasonable when the neural network uses a smooth nonlinearity such as SmoothReLU, GELU or a sigmoid function. As we point out later, using standard ReLU adds another layer of non-smooth analysis which we avoid here for the sake of simplicity of exposition. The last condition in (C1) is also reasonable, e.g., when the graphs of $Q(x,u;\cdot), Q(x,u';\cdot)$ cross along a finite union of lower dimensional submanifolds.

2. (C2) assumes stability of iterates, i.e., $\sup_n \|\theta_n\| < \infty$ a.s. There is an assortment of tests to verify this. See, e.g., [9], Chapter 3. Also, one can enforce this condition by projection onto a convenient large convex set every time the iterates exit this set, see *ibid.*, Chapter 7.

3. (C3) entails that we perform successive experience replays over larger and larger batches of past samples so that the error in applying the strong law of large numbers decreases sufficiently fast. While this is possible in principle because of the increasing pool of past traces with time, this will be an idealization in practice. It seems possible that the additional error in absence of this can be analyzed as in [34]. Note also that for deterministic control problems, experience replay is not needed for our purposes. The cartpole model studied in the next section is an example of this.

It is worth noting that bulk of the argument above is indeed the classical argument for convergence of stochastic gradient descent with both Markov and martingale difference noise, except that our iteration fits this paradigm only 'a.s.'. The missing piece is that the (possibly random) point it converges to need not be a point of differentiability of $\bar{\mathcal{E}}(\cdot)$, and therefore not a classical critical point thereof. This is what calls for the back and forth between the classical proof and the differential inclusion limit for stochastic gradient descent to minimize a non-smooth objective function.

Before we proceed, we would like to underscore a subtle point, viz., the role of experience replay here. Consider the scheme without the experience replay as above, given by

$$\begin{aligned}\theta_{n+1} &= \theta_n - a(n)(r(X_n,U_n) + \gamma \max_v Q(X_{n+1},v;\theta_n) - Q(X_n,U_n;\theta_n)) \times \\ &\quad \left(\gamma \nabla_\theta Q(X_{n+1},v;\theta_n)\Big|_{v=\text{argmax}_{Q(X_{n+1},\cdot;\theta_n)}} - \nabla_\theta Q(X_n,U_n;\theta_n)\right)\end{aligned} \tag{26}$$

The limiting o.d.e. for this is

$$\begin{aligned}\dot{\theta}(t) &= E\Bigg[\sum_y p(y|X_n,U_n)\Bigg((r(X_n,U_n)+\gamma\max_v Q(y,v;\theta(t))-Q(X_n,U_n;\theta_n))\times \\ &\quad \Bigg(\gamma\nabla_\theta Q(y,v;\theta(t))\Big|_{v=\text{argmax }Q(y,\cdot;\theta(t))}-\nabla_\theta Q(X_n,U_n;\theta(t))\Bigg)\Bigg)\Bigg],\end{aligned} \tag{27}$$

where $E[\cdot]$ denotes the stationary expectation. This is again not in a form where the convergence is apparent. The problem, typical of naive Bellman error gradient methods, is that we have a conditional expectation (w.r.t. $p(\cdot|X_n,U_n)$) of a product instead of a product of conditional expectations, as warranted by the actual Bellman error formula. The experience replay suggested above does one of the conditional expectations ahead of time, albeit approximately, and therefore renders (approximately) the expression a product of conditional expectations. Observe that this is so because we average over past traces (X_m,U_m,X_{m+1}) where X_m,U_m are fixed at the current X_n,U_n, so that it is truly a Monte Carlo evaluation of a conditional expectation. If we were to average over such traces without fixing X_n,U_n, we would get the o.d.e.

$$\begin{aligned}\dot{\theta}(t) &= E\Big[r(X_n,U_n)+\gamma\max_v Q(X_{n+1},v;\theta(t))-Q(X_n,U_n;\theta_n)\Big]\times \\ &\quad E\Big[\gamma\nabla_\theta Q(X_{n+1},v;\theta(t))\Big|_{v=\text{argmax}Q(X_{n+1},\cdot;\theta(t))}-\nabla_\theta Q(X_n,U_n;\theta(t))\Big],\end{aligned} \tag{28}$$

where $E[\cdot]$ denotes the stationary expectation. Here the problem is that the desired 'expectation of a product of conditional expectations' has been split into a product of expectations, which too is wrong. This discussion underscores an additional advantage of experience replay in the context of Bellman error gradient methods, over and above its traditional advantages listed earlier.

4.1 Comments about 'on-policy' schemes

An 'on-policy' scheme has an additional complication, viz., the expectation operator in the definition of $\bar{\mathcal{E}}(\cdot)$ itself depends on the parameter θ. This is because the policy with which the state-action pairs (X_n,U_n) are being sampled depends at time n on the current iterate θ_n. Therefore there is explicit θ dependence for the probability measure $\mu(\cdot,\cdot)$, now written as $\mu_\theta(\cdot,\cdot)$. The framework of [48] is broad enough to allow this 'iterate dependence' and we get the counterpart of (24) with $\mu(\cdot,\cdot)$ replaced by $\mu_{\theta(t)}(\cdot,\cdot)$, leading to the limiting differential inclusion

$$\dot{\theta}(t)\in-\nabla^*\bar{\mathcal{E}}_{\theta(t)}(\theta(t)). \tag{29}$$

Here ∇^* denotes the Frechet subdifferential with respect to only the argument in parentheses, not the subscript. Hence it is not the full subdifferential and the theoretical issues we pointed out regarding DQN come back to haunt us. This is true, e.g., when you use the ϵ-greedy policy that picks the control $\text{argmax}(Q(X_n,\cdot;\theta_n))$ with probability $1-\epsilon$, and chooses a control independently and with uniform probability from A, with probability ϵ.

Clearly, a scheme such as (29) that performs gradient descent for the stationary expectation of a parametrized cost function w.r.t. the parameter, but ignoring the parameter dependence of the stationary law itself on the parameter, is not guaranteed to converge. There are special situations such as the EM algorithm [16] where additional structure of the problem makes it work. In general, policy gradient methods based on suitable sensitivity formulas for Markov decision processes seem to provide the most flexible approach in such situations, see, e.g., [28].

5 Numerical results

In this section, we compare on two realistic examples the performance of FG-DQN with respect to that of the standard DQN scheme [30]. In particular, we investigate the behaviour of Bellman error, Hamming distance from the optimal policy (if the optimal policy is known) and the average reward. The pseudo-code for FG-DQN is described in Algorithm 1.

5.1 Forest management problem

Consider a Markov decision process framework for a simple forest management problem [12, 14]. The objective is to maintain an old forest for wildlife and make money by selling the cut wood. We consider discounted infinite horizon discrete-time problem. The state of the forest at time n is represented by $X_n \in \{0, 1, 2, 3, \cdots, M\}$ where the value of the state represents the age of the forest; 0 being the youngest and M being the oldest. The forest is managed by two actions: 'Wait' and 'Cut'. An action is applied at each time at the beginning of the time slot. If we apply the action 'Cut' at any state, the forest will return to its youngest age, i.e., state 0. On the other hand, when the action 'Wait' is applied, the forest will grow and move to the next state if no fire occurred. Otherwise, with probability p, the fire burns the forest after applying the 'Wait' action, leaving it at its youngest age (state 0). Note that if the forest reaches its maximum age, it will remain there unless there is a fire or action 'Cut' is performed. Lastly, we only get a reward when the 'Cut' action is performed. In this case, the reward is equal to the age of the forest. There is no reward for the action 'Wait'.

Algorithm 1: Full Gradient DQN (FG-DQN)

Input: replay memory $\mathcal{D}$ of size M, minibatch size B, number of episodes N, maximal length of an episode T, discount factor γ, exploration probability ϵ.

Initialise the weights θ randomly for the Q-Network.

for *Episode* $= 1$ *to* N **do**

  Receive initial observation s_1.

  for $n = 1$ *to* T **do**

    if *Uni[0,1]* $< \epsilon$ **then**

      Select action U_n at random.

    else

      $U_n = Argmax_u Q(X_n, u; \theta)$

    end

    Execute the action and take a step in the RL environment.

    Observe the reward R_n and obtain next state X_{n+1}.

    Store the tuple (X_n, U_n, R_n, X_{n+1}) in $\mathcal{D}$.

    Sample random minibatch of B tuples from $\mathcal{D}$.

    for $k = 1$ *to* B **do**

      Sample all tuples (X_j, U_j, R_j, X_{j+1}) with a fix state-action pair $(X_j = X_k, U_j = U_k)$ from $\mathcal{D}$

      $$Set\ Z_j = \begin{cases} R_j, & \text{for terminal state,} \\ R_j + \gamma \max_u Q(X_{j+1}, u; \theta), & \text{otherwise.} \end{cases}$$

      Compute gradients and using Eq. (19) update parameters θ.

    end

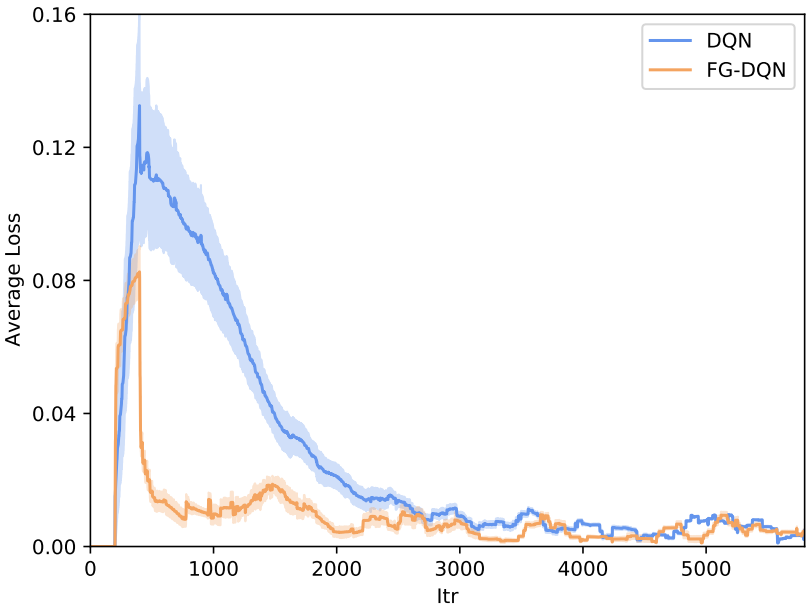
  end

end

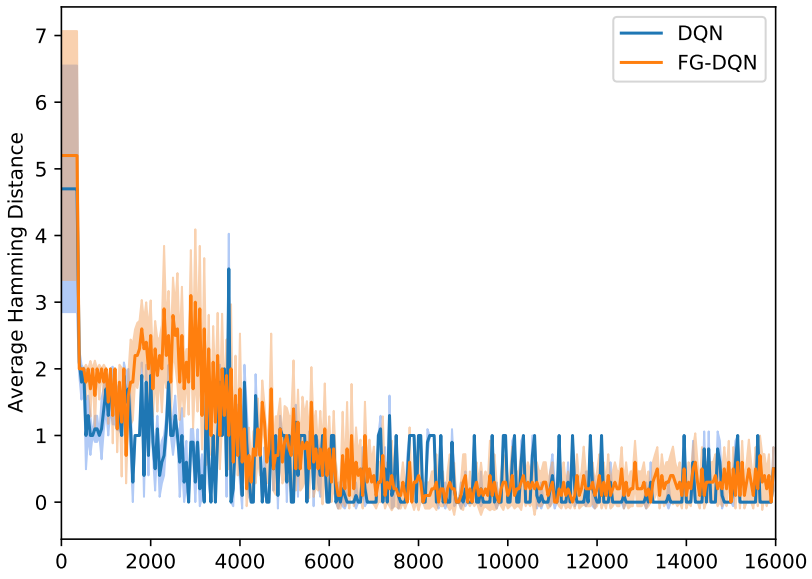
Since the objective is to maximize the discounted profit obtained by selling wood, we may want to keep waiting to get the maximum possible reward, but there is an increasing chance that the forest will get burned down.

For numerical simulations, we assume that the maximum age of the forest is $M = 10$. Then, we implement standard DQN and FG-DQN to analyse the policy obtained from the algorithm and the Bellman error. We use a neural network with one hidden layer to approximate the Q-value. The number of neurons for this hidden layer is 2000, and we use ReLU for nonlinear activation. It has been recently advocated to use a neural network with one but very wide hidden layer [2, 13]. The input to the neural network is the state of the forest and the action. Furthermore, the batch size to draw the samples for the experience replay is fixed to 25. We test both the algorithms for the off-policy scheme, i.e., we run through all possible state-action pairs in round-robin fashion to train the neural network.

We run two different simulations - i) with low discounting factor $\gamma = 0.8$ and ii) with high discounting factor $\gamma = 0.95$. Fig. 1 depicts the simulation results for case i) with forest fire probability $p = 0.05$. We run the experiment 10 times and plot the running average of Bellman error across iterations in Fig. 1(a). We also calculate the standard deviation of the Bellman

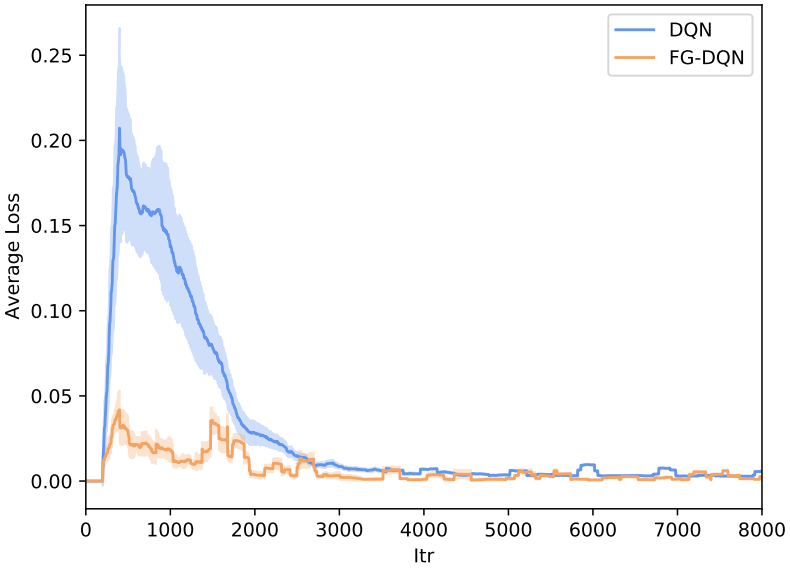


(a) Average loss and 95% confidence interval

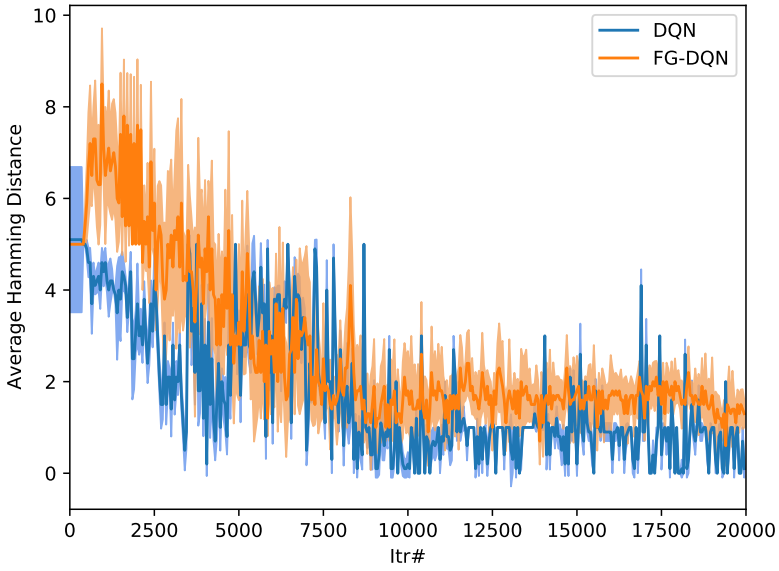


(b) Average Hamming distance from the optimal policy

Figure 1: Forest management problem with $\gamma = 0.8$ and $p = 0.05$



(a) Average loss and 95% confidence interval



(b) Average Hamming distance from the optimal policy

Figure 2: Forest management problem with $\gamma = 0.95$ and $p = 0.01$

error. The shaded region in the plot denotes the 95% confidence interval. We observe that FG-DQN converges much faster than DQN. Furthermore, the variance for FG-DQN is relatively low.

We now analyse how far the answer of each algorithm is from the optimal policy. To do this, we first find the optimal policy for this setting by policy iteration algorithm. The optimal policy has a threshold structure as follows: $\pi^* = [0, 0, 1, 1, 1, 1, 1, 1, 1, 1]$ for $\gamma = 0.8$ and $p = 0.05$.

After each iteration, we now evaluate the Q-network and calculate the Hamming distance between the current policy and the optimal policy π^*, which gives us the count of the number of states where optimal action is not taken. We run the simulations 10 times and plot the average Hamming distance for DQN and FG-DQN in Fig. 1(b). Note that we plot every $50th$ value of the average Hamming distance in the figure. It is to avoid the squeezing of rare spikes obtained at later time steps of the simulations. The shaded region denotes the 95% confidence interval for the averaged Hamming distance. We observe from the figure that the policy obtained by FG-DQN starts converging to the optimal policy at around 8000 iterations. In comparison, for DQN, we observe a lot of spikes during later iterations. The occurrence of these spikes means that there is a one-bit error in the policy obtained by DQN. Further analysis shows that the DQN policy in this case which has one bit error resembles to myopic policy $[0, 1, 1, 1, 1, 1, 1, 1, 1, 1]$.

We next observe the impact of a high discounting factor on the performance of our algorithm and how well it performs as compared to the standard DQN scheme. We set $\gamma = 0.95$ and forest fire probability $p = 0.01$. The optimal policy obtained by exact policy iteration for this case is $\pi^* = [0, 0, 0, 0, 0, 1, 1, 1, 1, 1]$. Fig 2(a) shows the mean loss for 10 simulations and the corresponding 95% confidence interval. We observe similar behaviour as before, i.e., the variance for FG-DQN is low. Fig 2(b) shows the averaged Hamming distance between the policy obtained by the algorithm and the optimal policy. It is clear from the figure that the variance for DQN is very high throughout the simulation. It means we may end up with a policy that can have a 3 or 4 bits error at the end of our simulation runs. On the other hand, FG-DQN is more stable since it shows fewer variations with the increasing number of iterations. Thus, we are more likely to get the policy with a 2 bits error on average. The shaded region in the plot shows the 95% confidence interval for 10 simulations which demonstrates that the behaviour is consistent across the multiple simulations.

5.2 Cartpole - OpenAI Gym model

We now test our algorithm for a more complex example, the Cartpole-v0 model from OpenAI gym [10]. The environment description is as follows. The state of the system is defined by a four dimensional tuple that represents cart position x, cart velocity $\dot{x}$, pole angle α and angular velocity $\dot{\alpha}$ (See Fig. 3). The pole starts upright and the aim is to prevent it from falling over by pushing the cart to the left or to the right (binary action space). The cart moves without friction along the x-axis.

We run multiple simulations, each with 1500 episodes for DQN and FG-DQN. For every time-step while an episode is running, we get the reward of +1. The episode ends if any of the following conditions holds: the pole is more than 12° degrees from the vertical axis, the cart moves more than 2.4 units from the centre, or the episode length is more than 200. The model

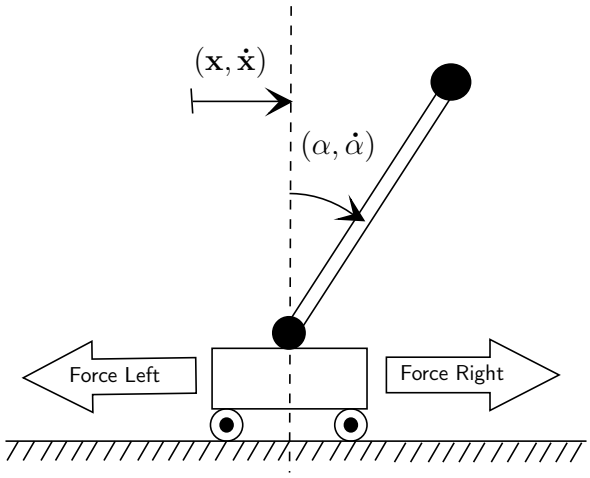


Figure 3: Cartpole system

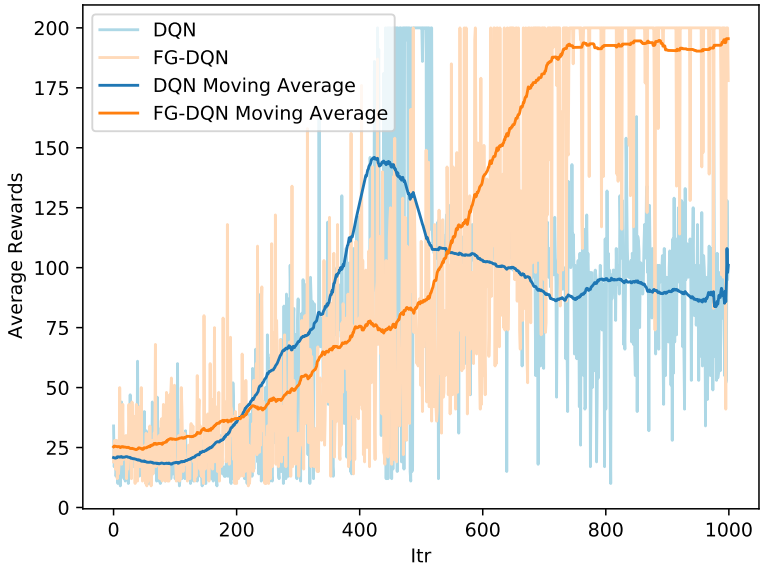
is considered to be trained well when the discounted reward is greater than or equal to 195.0 over 100 consecutive trials.

In this example, we used the 'on-policy' version with the popular 'ϵ-greedy' scheme which picks the current guess for the optimal (i.e., the control that maximizes $Q(X_n, \cdot; \theta_n)$) with probability $1 - \epsilon$ and chooses a control uniformly with probability ϵ for a prescribed $\epsilon > 0$. We use $\epsilon = 0.1$. As we see below, FG-DQN continues to do much better than DQN even in this on-policy scheme for which we do not have a convergence proof as yet.

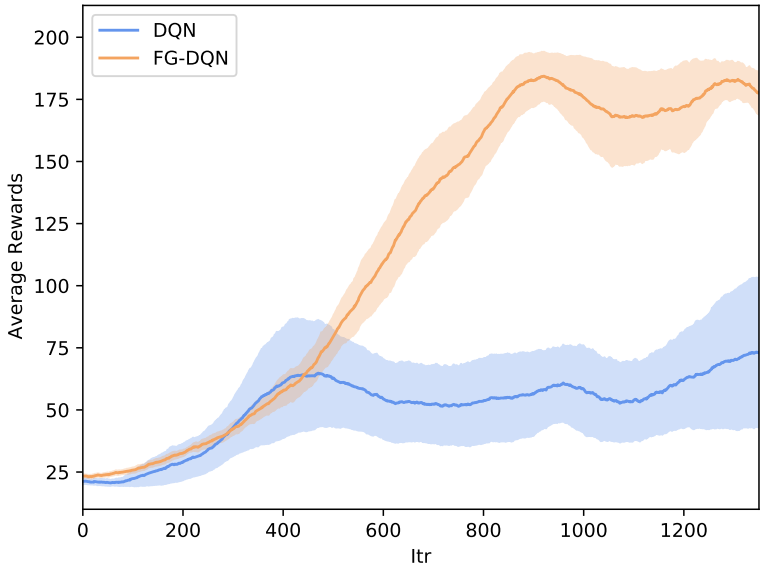
We use a neural network with three hidden layers. The number of nodes for the hidden layers are $16, 32$, and 32, respectively. For non-linearity, we use ReLU activation after each hidden layer.

We now compare the performances of FG-DQN and DQN for a very high discounting factor of 0.99. Note that the Cartpole example is deterministic, meaning that for a fixed state-action pair (X_n, U_n), the pole moves to state X'_n with probability 1. As a result, there will be no averaging in eq. (18) and no need for 'experience replay'. Since this example is complex with significant non-linearity, we use the batch size of 128 for both DQN and FG-DQN to update the parameters of the neural network inside one iteration.

Fig.4(a) depicts the reward behaviour for a single typical run of DQN and FG-DQN. We see that the fluctuations for reward per episode for both the algorithms are high, and thus, we also plot the moving average of rewards with a window of 100 episodes. It is clear from the figure that FG-DQN starts achieving the maximum reward of 200 after 800 episodes regularly, however, DQN hardly attains the maximum reward during 1000 episodes. To check the consistency of the behaviour of our algorithm, we run the experiment 10 times and plot the average reward and 95% confidence interval in Fig. 4(b). We see that FG-DQN performs much better than DQN with an average reward after 1000 episodes lying around 175. In comparison, the average reward for DQN is between 50 and 75.



(a) Average rewards for a typical single simulation run



(b) Rewards averaged over simulations for Cartpole and 95% confidence intervals

Figure 4: Cartpole example with $\gamma = 0.99$

6 Conclusions and future directions

We proposed and analyzed a variant of the popular DQN algorithm that we call Full Gradient DQN or FG-DQN wherein we also include the parametric gradient (in a generalized sense) of the target. This leads to a provably convergent scheme with sound theoretical basis which also shows improved performance over test cases. There is ample opportunity for further research in this direction, both theoretically and in terms of actual implementations. To highlight opportunities, we state here some additional remarks, which also contain a few pointers to future research directions.

1. Since the critical points are isolated, we get a.s. convergence to a single sample path dependent critical point. This situation is generic in the sense that it holds true for the problem parameters in an open dense set thereof, by a standard fact from Morse theory in the smooth case. However, connected sets of non-isolated equilibria can occur due to overparametrization and it will be interesting to develop sufficient conditions for point convergence.

2. Thanks to the addition of $\{\xi_n\}$, the noise in FG-DQN is 'rich enough' in all directions in a certain sense. One then expects it to ensure that under reasonable assumptions, the unstable equilibria, here the critical points other than local minima of the Bellman error, will be avoided with probability one. That is, a.s. convergence to a local minimum can be claimed. See section 4.3 of [9] for a result of this flavor under suitable technical conditions. We expect a similar result to hold here. In practice, the extraneous noise $\{\xi_n\}$ is usually unnecessary and the inherent numerical errors and noise of the iterations suffice.

3. We can also use approximation of the 'max' operator by 'softmax', i.e., by picking the control with a probability distribution that concentrates on argmax and depends smoothly on the parameter θ. Then we can work with a legitimate gradient in place of a set-valued map in the o.d.e. limit, at the expense of picking up an additional bounded error term. Then the convergence to a small neighborhood of an equilibrium may be expected, the size of which will be dictated in turn by the bound on this error, see, e.g., [34]. There is a similar issue if we drop (C3) and let a persistent small error due to the use of approximate conditional expectation by experience replay remain.

4. Working with nondifferentiable nonlinearities such as ReLU raises further technical issues in analysis that need to be explored. This will require further use of non-smooth analysis.

5. As we have pointed out while describing our numerical experiments on the cartpole example, FG-DQN gives a significantly better performance than DQN, in an 'on-policy' scenario for which we do not have rigorous theory yet. This is another promising and important research direction for the future.

7 Acknowledgement

The authors are greatly obliged to Prof. K. S. Mallikarjuna Rao for pointers to the relevant literature on non-smooth analysis. The work of VSB was supported in part by an S. S. Bhatnagar Fellowship from the Council of Scientific and Industrial Research, Government of India. The work of KP and KA is partly supported by ANSWER project PIA FSN2 (P15 9564-266178

/ DOS0060094) and the project of Inria - Nokia Bell Labs "Distributed Learning and Control for Network Analysis". This work is also partly supported by the project IFC/DST-Inria-2016-01/448 "Machine Learning for Network Analytics".

This is the author's version of the work for the Springer book: A.B. Piunovskiy and Y. Zhang (eds.), *Modern Trends in Controlled Stochastic Processes: Theory and Applications, Volume III*, 2021.

Appendix: Elements of non-smooth analysis

The (Frechet) sub/super-differentials of a map $f : \mathcal{R}^d \mapsto \mathcal{R}$ are defined by

$$\begin{aligned} \partial^- f(x) &:= \left\{ z \in \mathcal{R}^d : \liminf_{y \to x} \frac{f(y) - f(x) - \langle z, y - x \rangle}{|x - y|} \geq 0 \right\}, \\ \partial^+ f(x) &:= \left\{ z \in \mathcal{R}^d : \limsup_{y \to x} \frac{f(y) - f(x) - \langle z, y - x \rangle}{|x - y|} \leq 0 \right\}, \end{aligned}$$

respectively. Assume f, g is Lipschitz. Some of the properties of $\partial^{\pm} f$ are as follows.

- **(P1)** Both $\partial^- f(x), \partial^+ f(x)$ are closed convex and are nonempty on dense sets.
- **(P2)** If f is differentiable at x, both equal the singleton $\{\nabla f(x)\}$. Conversely, if both are nonempty at x, f is differentiable at x and they equal $\{\nabla f(x)\}$.
- **(P3)** $\partial^- f + \partial^- g \subset \partial^-(f + g),\ \partial^+ f + \partial^+ g \subset \partial^+(f + g)$.

The first two are proved in [3], pp. 30-1. The third follows from the definition. Next consider a continuous function $f : \mathcal{R}^d \times B \mapsto \mathcal{R}$ where B is a compact metric space. Suppose $f(\cdot, y)$ is continuously differentiable uniformly w.r.t. y. Let $\nabla_x f(x, y)$ denote the gradient of $f(\cdot, y)$ at x. Let $g(x) := \max_y f(x, y), h(x) := \min_y f(x, y)$ with

$$M(x) := \{\nabla_x f(x, y), y \in \text{Argmax} f(x, \cdot)\}$$

and

$$N(x) := \{\nabla_x f(x, y), y \in \text{Argmin} f(x, \cdot)\}.$$

Then $N(x), M(x)$ are compact nonempty subsets of B which are upper semicontinous in x as set-valued maps. We then have the following general version of Danskin's theorem [15]:

- **(P4)** $\partial^- g(x) = \overline{\text{co}}(M(x)), \partial^+ g(x) = y$ if $M(x) = \{y\}$, $= \phi$ otherwise, and g has a directional derivative in any direction z given by $\max_{y \in M(x)} \langle y, z \rangle$.
- **(P5)** $\partial^+ h(x) = \overline{\text{co}}(N(x)), \partial^- h(x) = y$ if $N(x) = \{y\}$, $= \phi$ otherwise, and h has a directional derivative in any direction z given by $\min_{y \in N(x)} \langle y, z \rangle$.

The latter is proved in [3], pp. 44-6, the former follows by a symmetric argument.

References

[1] Agarwal, A., Kakade, S.M., Jason D. L., Mahajan, G.: Optimality and approximation with policy gradient methods in Markov decision processes. In Conference on Learning Theory, PMLR, 64-66 (2020)

[2] Agazzi, A., Lu, J.: Global optimality of softmax policy gradient with single hidden layer neural networks in the mean-field regime. arXiv preprint arXiv:2010.11858, (2020)

[3] Bardi, M., Capuzzo-Dolcetta, I.: Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations. Birkhäuser, Boston (2018)

[4] Barto, A. G., Sutton, R. S., Anderson, C. W.: Neuronlike adaptive elements that can solve difficult learning control problems. IEEE Trans. Syst. Man Cybern. Syst., **5**, 834-846 (1983)

[5] Benveniste, A., Metivier, M., Priouret, P.: Adaptive Algorithms and Stochastic Approximations. Springer Verlag, Berlin Heidelberg (1991)

[6] Bertsekas, D. P.: Reinforcement Learning and Optimal Control. Athena Scientific. (2019)

[7] Bhandari, J., Russo, D.: Global optimality guarantees for policy gradient methods. arXiv preprint arXiv:1906.01786. (2019)

[8] Borkar, V. S.: Probability Theory: An Advanced Course, Springer Verlag, New York. (1995)

[9] Borkar, V. S.: Stochastic Approximation: A Dynamical Systems Viewpoint. Hindustan Publishing Agency, New Delhi, and Cambridge University Press, Cambridge, UK (2008)

[10] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W.: OpenAI Gym. ArXiv preprint arXiv:1606.01540 (2016)

[11] Cai, Q., Yang, Z., Lee, J. D., Wang, Z.: Neural temporal-difference learning converges to global optima. Advances in Neural Information Processing Systems **32**, (2019)

[12] Chadès, I., Chapron, G., Cros, M. J., Garcia, F., Sabbadin, R.: MDPtoolbox: A multi-platform toolbox to solve stochastic dynamic programming problems. Ecography, **37**, 916-920 (2014)

[13] Chizat, L., Bach, F.: On the global convergence of gradient descent for over-parameterized models using optimal transport. In Proceedings of Neural Information Processing Systems, 3040-3050 (2018)

[14] Couture, S., Cros, M. J., and Sabbadin R.: Risk aversion and optimal management of an uneven-aged forest under risk of windthrow: A Markov decision process approach. J. For. Econ. **25**, 94-114 (2016)

[15] Danskin, J. M.: The theory of max-min, with applications. SIAM J. Appl. Math. **14**, 641-664 (1966)

[16] Delyon, B., Lavielle, M. and Moulines, E.: Convergence of a stochastic approximation version of the EM algorithm. Ann. Stat. **27**, pp. 94-128 (1999)

[17] Florian, R. V.: Correct equations for the dynamics of the cart-pole system. Center for Cognitive and Neural Studies (Coneural), Romania (2007)

[18] François-Lavet, V., Henderson, P., Islam, R., Bellemare, M. G., Pineau, J.: An Introduction to Deep Reinforcement Learning. Found. Trends Mach. Learn. **11** 3-4 pp. 219-354 (2018)

[19] Gordon, G. J.: Stable fitted reinforcement learning. Advances in Neural Information Processing Systems, pp.1052-1058 (1996)

[20] Gordon, G. J.: Approximate Solutions to Markov Decision Processes. PhD Thesis, Carnegie-Mellon University (1999)

[21] Gu, S., Holly, E., Lillicrap, T., Levine, S.: Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In Proceedings of IEEE International Conference on Robotics and Automation, 3389-3396 (2017)

[22] Haarnoja, T., Ha, S., Zhou, A., Tan, J., Tucker, G., Levine, S.: Learning to walk via deep reinforcement learning. ArXiv preprint arXiv:1812.11103 (2018)

[23] Jaakola, T., Jordan, M. I., Singh, S. P.: On the convergence of stochastic iterative dynamic programming algorithms. Neural Computation. **6**, 1185–1201 (1994)

[24] Jonsson, A.: Deep reinforcement learning in medicine. Kidney Diseases **5**, 18-22 (2019)

[25] Lin, L.J.: Self-improving reactive agents based on reinforcement learning, planning and teaching. Machine learning **8** 3-4, 293-321 (1992)

[26] Lin, L.-J.: Reinforcement Learning for Robots Using Neural Networks., Ph.D. Thesis School of Computer Science, Carnegie-Mellon University, Pittsburgh (1993)

[27] Luong, N. C., Hoang, D. T., Gong, S., Niyato, D., Wang, P., Liang, Y. C., Kim, D. I.: Applications of deep reinforcement learning in communications and networking: A survey. IEEE Commun. Surv. Tutor. **21**, 3133-3174 (2019)

[28] Marbach, P., Tsitsiklis, J. N.: Simulation-based optimization of Markov reward processes. IEEE Trans. Automat. Contr. **46**, 191-209 (2001)

[29] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)

[30] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S.: Human-level control through deep reinforcement learning. Nature **518**, 529-533 (2015)

[31] Peng, X. B., Berseth, G., Yin, K., van de Panne, M. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. ACM Trans. Graph. **36**, 1-13 (2017)

[32] Popova, M., Isayev, O., and Tropsha, A.: Deep reinforcement learning for de novo drug design. Sci. Adv. 4(7), eaap7885 (2018)

[33] Qian, Y., Wu, J., Wang, R., Zhu, F. and Zhang, W.: Survey on reinforcement learning applications in communication networks. J. Commun. Netw. **4**, 30-39 (2019)

[34] Ramaswamy, A., Bhatnagar, S.: Analysis of gradient descent methods with nondiminishing bounded errors. IEEE Trans. Automat. Contr. **63**, 1465-1471 (2018)

[35] Riedmiller, M.: Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method. Machine Learning: ECML, pp. 317-328 (2005)

[36] Schaul, T., Quan, J., Antonoglou, I., Silver, D.: Prioritized experience replay. arXiv preprint arXiv:1511.05952 (2015)

[37] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Hassabis, D.: Mastering the game of Go with deep neural networks and tree search. Nature, **529**, pp.484-489 (2016)

[38] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Hassabis, D.: A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. Science, **362**, 1140-1144 (2018)

[39] Sutton, R. S. and Barto, A. G.: Reinforcement Learning: An Introduction. 2nd edition, MIT Press, (2018)

[40] Sutton, R. S., McAllester, D. A., Singh, S. P., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In Neural Information Processing Systems Proceedings, pp.1057-1063 (1999)

[41] Tsitsiklis, J. N.: Asynchronous stochastic approximation and Q-learning. Machine learning **16**, 185-202 (1994)

[42] Tsitsiklis, J. N., Van Roy, B.: An analysis of temporal-difference learning with function approximation. IEEE Trans. Automat. Contr. **42**, 674-690 (1997)

[43] van Hasselt, H.: Double Q-learning. Advances in Neural Information Processing Systems **23**, 2613-2621 (2010)

[44] van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double Q-learning. In Proceedings of the 30th AAAI Conference on Artificial Intelligence, **30**, 2094-2100 (2016)

[45] Watkins, C. J. C. H.: Learning from Delayed Rewards. Ph.D. Thesis, King's College, University of Cambridge, UK (1989)

[46] Watkins, C. J., Dayan, P.: Q-learning. Machine learning **8** 3-4, 279-292 (1992)

[47] Xiong, Z., Zhang, Y., Niyato, D., Deng, R., Wang, P., Wang, L. C.: Deep reinforcement learning for mobile 5G and beyond: Fundamentals, applications, and challenges. IEEE Veh. Technol. Mag. **14**, 44-52 (2019)

[48] Yaji, V.G., Bhatnagar, S.: Stochastic recursive inclusions with non-additive iterate-dependent Markov noise. Stochastics, **90**, 330-363 (2018)