2019 Fifth Indian Control Conference (ICC)
IIT Delhi, India, January 9-11, 2019

Zap Q-Learning – A User's Guide

Adithya M. Devraj[1], Ana Bušić[2], and Sean Meyn[1]

Abstract— **There are two well known Stochastic Approximation techniques that are known to have optimal rate of convergence (measured in terms of asymptotic variance): the Stochastic Newton-Raphson (SNR) algorithm (a matrix gain algorithm that resembles the deterministic Newton-Raphson method), and the Ruppert-Polyak averaging technique. This paper surveys new applications of these concepts for Q-learning:**

(i) The *Zap Q-Learning* algorithm was introduced by the authors in a NIPS 2017 paper. It is based on a variant of SNR, designed to more closely mimic its deterministic cousin. The algorithm has optimal rate of convergence under general assumptions, and showed astonishingly quick convergence in numerical examples. These algorithms are surveyed and illustrated with numerical examples. A potential difficulty in implementation of the Zap-Q-Learning algorithm is the matrix inversion required in each iteration.

(ii) Remedies are proposed based on stochastic approximation variants of two general deterministic techniques: Polyak's momentum algorithms and Nesterov's acceleration technique. Provided the hyper-parameters are chosen with care, the performance of these algorithms can be comparable to the Zap algorithm, while computational complexity per iteration is far lower.

I. Introduction

The goal of this paper is to survey new techniques for approximating the value function that arises in Markov Decision Process models (MDPs). To ease exposition it is assumed that the state and action spaces are finite, and time is discrete. Of particular interest are on-line algorithms that require only input-output measurements for implementation. The celebrated Q-Learning algorithm of Watkins [1] is an important example.

Consider the discounted-cost control problem with state-action process denoted $(\boldsymbol{X},\boldsymbol{U}) = \{(X_n,U_n) : n \ge 0\}$ evolving on the finite set $\mathsf{X}\times\mathsf{U}$, cost function $c\colon \mathsf{X}\times\mathsf{U}\to\mathbb{R}$, and discount factor $0<\beta<1$. The Q-function solves the Bellman equation:

$$Q^*(x,u) = c(x,u) + \beta \mathsf{E}_{x,u}[\underline{Q}^*(X_1)] \tag{1}$$

where the subscript in the expectation indicates condition that $(X_0,U_0)=(x,u)$, and the underbar denotes the minimum: $\underline{Q}(x) := \min_{u'} Q(x,u')$ for any function Q on $\mathsf{X}\times\mathsf{U}$. The optimal policy can be obtained as the minimizer:

$$\phi^*(x) = \arg\min_u Q^*(x,u) \tag{2}$$

Funding from the National Science Foundation award EPCN 1609131.

[1]Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611

[2]Inria and the Computer Science Department of École Normale Supérieure, 75005 Paris, France

The paper will focus on approximation within a finite-dimensional linear function class. Given d basis functions $\{\psi_k : 1\le k\le d\}$, with each $\psi_k : \mathsf{X}\times\mathsf{U}\to\mathbb{R}$, and a parameter vector $\theta\in\mathbb{R}^d$, the corresponding Q-function estimate is defined by

$$Q^\theta(x,u) = \sum_k \theta_k\psi_k(x,u)\,.$$

The Bellman error is denoted

$$\mathcal{B}^\theta(x,u) = -Q^\theta(x,u) + c(x,u) + \beta\mathsf{E}_{x,u}[\underline{Q}^\theta(X_1)] \tag{3}$$

If this is identically zero for some θ^* and all (x,u), then we have solved the dynamic programming equation (1).

A *Galerkin relaxation* is adopted as a quality of fit in most formulations of reinforcement learning. In the Q-learning problem considered here, it is assumed that $\boldsymbol{U}$ is obtained as a stationary randomized policy, defined so that $(\boldsymbol{X},\boldsymbol{U})$ is an irreducible Markov chain. A d-dimensional stochastic process $\boldsymbol{z} = \{z_n : n\ge 0\}$ is constructed that is adapted to $(\boldsymbol{X},\boldsymbol{U})$. This is known as the sequence of *eligibility vectors* in the reinforcement learning literature. For a stationary realization of the triple $(\boldsymbol{X},\boldsymbol{U},\boldsymbol{z})$, a relaxation of the Bellman error is denoted

$$\overline{\mathcal{B}}(\theta) = \mathsf{E}[\mathcal{B}^\theta(X_n,U_n)z_n] \tag{4}$$

A solution to the Galerkin relaxation is defined as any vector $\theta^*\in\mathbb{R}^d$ solving $\overline{\mathcal{B}}(\theta^*)=0\in\mathbb{R}^d$.

The solution to the Galerkin relaxation can be obtained through the techniques of *stochastic approximation* (SA). In this approach, a recursive algorithm to compute θ^* is constructed based on an approximation of the ODE

$$\tfrac{d}{dt}x(t) = G\overline{\mathcal{B}}(x(t)) \tag{5}$$

where $G\in\mathbb{R}^{d\times d}$ is a matrix gain.

Watkins' Q-learning algorithm can be constructed in this way – details are provided in the next section. This algorithm is designed to compute Q^* exactly [2], [1]. The basis is taken to be the set of indicator variables:

$$\psi_k(x,u) = \mathbb{I}\{(x,u)=(x^k,u^k)\}\,, \tag{6}$$

where $\{(x^k,u^k) : 1\le k\le d\}$, are ordered state-action pairs, with $d=|\mathsf{X}\times\mathsf{U}|$. The sequence of eligibility vectors is taken to be $z_n=\psi(X_n,U_n)\in\mathbb{R}^d$ for $n\ge 0$.

The rich theory of SA can give enormous insight on algorithm design, such as the optimal choice of the matrix G appearing in (5).

The literature on SA provides conditions under which the Central Limit Theorem and Law of the Iterated Logarithm hold for the error sequence $\{\tilde\theta_n := \theta_n-\theta^* : n\ge 0\}$; these

limit theorems will serve as a guide to algorithm design in the present paper. The *asymptotic covariance* appearing in these results can be expressed as the limit

$$\Sigma^\theta := \lim_{n\to\infty} \Sigma_n^\theta := \lim_{n\to\infty} n\mathsf{E}[\tilde{\theta}_n \tilde{\theta}_n^{\scriptscriptstyle T}] \,. \tag{7}$$

The LIL is most interesting in terms of bounds:

$$|\tilde{\theta}_n(i)| \le \Big(\sigma(i) + o(1)\Big)\sqrt{\frac{2\log\log(n)}{n}} \tag{8}$$

where $o(1) \to 0$ as $n \to \infty$, and $\sigma^2(i) = \Sigma^\theta(i,i)$.

An approach to algorithm design is to minimize $\sigma(i)$ for each i. This optimization problem has an attractive solution: denote the matrix defining the linearized dynamics by

$$A := \partial_\theta \overline{\mathcal{B}}\,(\theta^*)$$

Under general conditions, the optimal gain is obtained as $G^* := -A^{-1}$; it is optimal in the sense that the difference $\Sigma_\theta^G - \Sigma^*$ is positive semi-definite for any G [3], [4], [5]. This observation is the starting point of the Zap algorithm introduced in [6], [7].

This paper provides a user's guide for these new algorithms, as well as more recent algorithms based on a synthesis of techniques from classical SA theory, combined with variants of momentum algorithms pioneered by Polyak [8], [9], and Nesterov's acceleration methods [10], [11].

Literature survey: This paper is built on a vast literature on optimization [10], [8], [9], [11] and stochastic approximation [13], [4], [5], [14], [15], [16].

An emphasis in much of the literature is computation of finite-time PAC (probably almost correct) bounds as a metric for performance. Explicit bounds were obtained in [17] for Watkins' algorithm, and in [18] for the "speedy" Q-learning algorithm. A general theory is presented in [19] for stochastic approximation algorithms. The assumption in prior work that the noise terms appearing in SA recursions form a martingale-difference sequence does not hold for the algorithms considered in this paper. Hence it is unlikely that the techniques used in this prior work can be extended to the algorithms considered in this paper.

Empirical value iteration is the topic of the recent work [20], [21] in the context of off-line (simulation based) value function approximation. The numerical results in this work are impressive. It will be of interest to investigate rates of convergence and computation complexity for these algorithms.

The most relevant related research concerns ERM (empirical risk minimization) [22], [23], [24] rather than Q-learning. Under general conditions it can be shown that the sequence of ERM optimizers $\{\theta_n^*\}$ is convergent, and has optimal asymptotic covariance (a survey and further discussion is presented in [24]). The papers [19], [25], [26], [27], [24] establish the optimal convergence rate of $O(1/\sqrt{n})$ for various algorithms.

The recent paper [24] is most closely related to this survey, considering the shared goal of optimization of the asymptotic covariance, along with rapidly vanishing transients through algorithm design. The paper restricts to nonlinear optimization; it is not yet clear if the techniques can be extended to reinforcement learning.

II. Zap Q-learning

It is first necessary to explain how to construct an SA algorithm based on the ODE (5). The crucial step is to remove the conditional expectation arising in the definition of the Bellman error. The smoothing property of conditional expectation gives

$$\overline{\mathcal{B}}(\theta) = \mathsf{E}[\widetilde{\mathcal{B}}_{n+1}^\theta z_n] \tag{9}$$

where for each $n \ge 0$,

$$\widetilde{\mathcal{B}}_{n+1}^\theta = -Q^\theta(X_n, U_n) + c(X_n, U_n) + \beta \underline{Q}^\theta(X_{n+1}) \tag{10}$$

and the expectation is with respect to the steady state distribution of the Markov chain $(\boldsymbol{X}, \boldsymbol{U})$. The stochastic approximation for the ODE (5) is defined by the recursion

$$\theta_{n+1} = \theta_n + \alpha_n G_{n+1} d_{n+1} z_n \,, \quad n \ge 0 \,, \tag{11}$$

where the fixed matrix gain is replaced with a sequence $\boldsymbol{G} = \{G_n\}$, and the "temporal difference" is

$$d_{n+1} = \widetilde{\mathcal{B}}_{n+1}^\theta \,|_{\theta=\theta_n} \,.$$

Throughout this paper it is assumed that $\alpha_n = 1/(n+1)$.

In the standard version of Watkins' algorithm, the gain sequence is diagonal, with

$$G_n(i,i) = \frac{1}{p_n(i)} \tag{12}$$

where $p_n(i)$ denotes the fraction of times that the pair (x^i, u^i) is visited up to time n; $G_n(i,i)$ is set to zero up until the first time the corresponding state-action pair is visited. The algorithm is typically described differently, but it is useful to have a common representation for the algorithms discussed in this paper. We will denote this special diagonal matrix gain sequence with $\{\widehat{D}_n\}$.

Proposition 2.1: For Watkins' algorithm:

(i) The asymptotic covariance is not finite if $\beta > 1/2$.

(ii) The asymptotic covariance is finite for any $\beta \in (0,1)$ if the gain is increased to:

$$G_n(i,i) = \widehat{D}_n(i,i) := \frac{1}{1-\beta}\frac{1}{p_n(i)} \tag{13}$$

□

Part (i) is established in [6], [7]; a similar proof can be used to establish (ii). The algorithm with matrix gain (13) works well in some cases, but we do not know of any example for which the resulting variance is optimal.

A particular gain choice based on stochastic Newton-Raphson, and a particular choice of eligibility vectors, results in the ***Zap-Q(λ) algorithm:***

$$\begin{aligned} \theta_{n+1} &= \theta_n - \alpha_n \widehat{A}_{n+1}^{-1} d_{n+1} z_n \\ d_{n+1} &= \widetilde{\mathcal{B}}_{n+1}^{\theta_n} \\ z_{n+1} &= \lambda\beta z_n + \psi(X_n, U_n), \end{aligned} \tag{14}$$

where $\{\widehat{A}_n\}$ is calculated using the recursion:

$$\begin{aligned}\widehat{A}_{n+1} &= \widehat{A}_n + \gamma_n\big[A_{n+1} - \widehat{A}_n\big]\\ A_{n+1} &= z_n\big[\beta\psi(X_{n+1},\phi_n(X_{n+1})) - \psi(X_n,U_n)\big]^{\intercal}\end{aligned} \tag{15}$$

and

$$\phi_n(x) \in \arg\min_u Q^{\theta_n}(x,u), \quad x \in \mathsf{X}\,. \tag{16}$$

The step-size sequence $\{\gamma_n\}$ must satisfy:

$$\lim_{n\to\infty} \frac{\gamma_n}{\alpha_n} \to \infty.$$

In experiments the gain sequences are assumed to be of the following form: for some $\rho \in (\frac{1}{2}, 1)$,

$$\alpha_n = \frac{1}{n+1}\,, \quad \gamma_n = \frac{1}{(n+1)^\rho}\,, \qquad n \ge 1\,. \tag{17}$$

This algorithm is of the form (11) with $G_n = -\widehat{A}_n^{-1}$. It is argued in [6], [7] that $\{\widehat{A}_n\}$ are approximations of the matrices used in the deterministic Newton-Raphson algorithm:

$$\widehat{A}_n \approx \partial_\theta \overline{\mathcal{B}}\,(\theta_n)$$

Properties of this algorithm are summarized in the next result in the special case of *tabular Q-learning*, in which the basis functions coincide with Watkins' algorithm, so we can identify $Q^\theta \equiv \theta$.

First a few preliminaries: Given any function $\varsigma\colon \mathsf{X}\times\mathsf{U} \to \mathbb{R}$, let $\mathcal{Q}^*(\varsigma)$ denote the corresponding solution to the fixed point equation (1), with c replaced by ς. That is, the function $q = \mathcal{Q}^*(\varsigma)$ is the solution to the fixed point equation,

$$q(x,u) = \varsigma(x,u) + \beta\sum_{x'} P_u(x,x')\min_{u'} q(x',u')\,. \tag{18}$$

The following result implies that $Q^{\theta_n} \approx \mathcal{Q}^*(\widehat{C}_n)$, with

$$\widehat{C}_n = -\Pi^{-1}\widehat{A}_n Q^{\theta_n}\,, \quad n \ge 1\,, \tag{19}$$

where Π is the $d\times d$ diagonal matrix with entries ϖ (the steady-state distribution of $(\boldsymbol{X},\boldsymbol{U})$).

The proof of Theorem 2.2 can be found in [6], [7].

Theorem 2.2: Suppose the following assumptions hold for the Zap Q-learning algorithm with basis (6):

(i) $(\boldsymbol{X},\boldsymbol{U})$ is an irreducible Markov chain

(ii) The optimal policy ϕ^* defined in (2) is unique

(iii) The step-size sequences $\{\alpha_n\}$ and $\{\gamma_n\}$ satisfy (17)

(iv) The sequence of policies $\{\phi_n\}$ satisfy

$$\sum_{n=1}^{\infty} \gamma_n \mathbb{I}\{\phi_{n+1} \ne \phi_n\} < \infty\,, \quad a.s..$$

Then,

(i) The sequence $\{Q^{\theta_n}\}$ obtained using the Zap-Q algorithm converges to Q^* a.s..

(ii) The resulting asymptotic covariance is optimal: the matrix inequality $\Sigma^{\text{Zap}} \ge \Sigma$ holds for any other consistent algorithm with finite asymptotic covariance Σ.

(iii) An ODE approximation holds for the sequence $\{Q^{\theta_n}, \widehat{C}_n\}$, by continuous functions $(\boldsymbol{q},\boldsymbol{\varsigma})$ satisfying

$$q_t = \mathcal{Q}(\varsigma_t)\,, \quad \tfrac{d}{dt}\varsigma_t = -\varsigma_t + c \tag{20}$$

This ODE approximation is exponentially asymptotically stable, with $\lim_{t\to\infty} q_t = Q^*$. □

III. Momentum and Acceleration

In this section, we survey the momentum based techniques that are introduced in [12], with applications to Q-learning.

To simplify discussion it is helpful to consider first an abstract setting. Let $\{f_n(\,\cdot\,) : n \ge 0\}$ be a sequence of random functions from $\mathbb{R}^d$ to itself. It is assumed that there is a well defined average: for each $\theta \in \mathbb{R}^d$,

$$\bar{f}(\theta) := \lim_{n\to\infty} \frac{1}{n}\sum_{k=0}^{n-1} f_k(\theta) = \lim_{n\to\infty} \mathsf{E}[f_n(\theta)]\,. \tag{21}$$

where the first limit is in the a.s. sense. The goal of SA: find the vector θ^* solving $\bar{f}(\theta^*) = 0$.

Each of the algorithms described here is defined in terms of the difference sequence $\Delta\theta_n := \theta_n - \theta_{n-1}$, $n \ge 0$, with given initial condition $\theta_0 = \theta_{-1}$. The matrix gain algorithm of Robbins and Monro is expressed as follows:

$$\Delta\theta_{n+1} = \alpha_n G_{n+1} f_{n+1}(\theta_n)\,, \quad n \ge 0. \tag{22}$$

The Q-learning recursion (11) is thus a special case. The Zap stochastic approximation algorithm is obtained with

$$G_n = -\widehat{A}_n^{-1} \tag{23}$$

where $\{\widehat{A}_n\}$ are estimates of $\partial_\theta \bar{f}(\theta_n)$, obtained as in Zap Q-learning.

The two new algorithms surveyed here fall outside of the class of SA recursions [12]:

***Matrix Heavy-Ball Stochastic approximation (PolSA)*:** For a sequence of matrices $\{M_n : n \ge 1\}$, and a fixed scalar $\zeta > 0$,

$$\Delta\theta_{n+1} = M_{n+1}\Delta\theta_n + \alpha_n\zeta f_{n+1}(\theta_n) \tag{24}$$

***Nesterov Stochastic approximation (NeSA)*:** For a fixed scalar $\zeta > 0$,

$$\begin{aligned}\Delta\theta_{n+1} = \Delta\theta_n &+ \zeta[f_{n+1}(\theta_n) - f_{n+1}(\theta_{n-1})]\\ &+ \zeta\alpha_n f_{n+1}(\theta_n)\end{aligned} \tag{25}$$

The PolSA algorithm coincides with Polyak's heavy-ball method when $\{M_n\}$ is a sequence of *scalars* [8], [9], [28]. The recursion (25) is inspired by Nesterov's acceleration method. Motivation is provided through the following arguments.

Gain selection: Assume that for an effective algorithm, the difference sequence $\{\Delta\theta_n\}$ vanishes at a rate much faster than the error sequence $\{\tilde{\theta}_n\}$. This is true for standard SA algorithms provided the asymptotic covariance is finite. Replacing $M_{n+1}\Delta\theta_n$ by $M_{n+1}\Delta\theta_{n+1}$ in (24) results in the recursion

$$\Delta\theta_{n+1} = \alpha_n\zeta[I - M_{n+1}]^{-1} f_{n+1}(\theta_n)$$

This recursion coincides with Zap SA, provided the matrix gain sequence $\{M_n\}$ is chosen to solve

$$\zeta[I - M_{n+1}]^{-1} = -\widehat{A}_{n+1}^{-1}$$

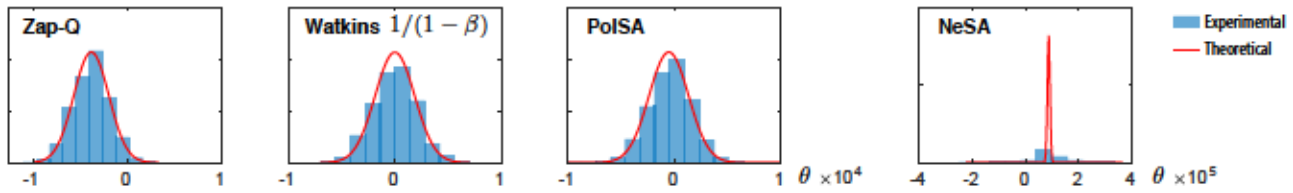

Fig. 1. Histograms for entry 18 of $\{\sqrt{n}\tilde{\theta}_n\}$ for the 6 node example at iteration 10^6.

and therefore, a solution is

$$M_{n+1} = I + \zeta \widehat{A}_{n+1} \tag{26}$$

This "Zap" momentum gain will be used in the PolSA recursion in the numerical results that follow, resulting in

$$\Delta\theta_{n+1} = [I + \zeta \widehat{A}_{n+1}]\Delta\theta_n + \alpha_n \zeta f_{n+1}(\theta_n) \tag{27}$$

The special form (25) is motivated by a Taylor series approximation:

$$f_{n+1}(\theta_n) - f_{n+1}(\theta_{n-1}) \approx A_{n+1}\Delta\theta_n$$

where $A_{n+1} = \partial_\theta f_{n+1}(\theta_n)$. Hence the NeSA algorithm is approximated by the following "noisy" version of (27):

$$\Delta\theta_{n+1} = [I + \zeta A_{n+1}]\Delta\theta_n + \alpha_n \zeta f_{n+1}(\theta_n) \tag{28}$$

Theoretical analysis of these algorithms is currently under study. It is shown in [12] that the asymptotic covariance is finite for linear recursions (such as applications to TD-learning), under suitable conditions on the model. The analysis requires replacing the estimates of A in (23) and (26) with its limit, resulting in

$$G_n \equiv -A^{-1} \tag{29}$$
$$M_n \equiv I + \zeta A \tag{30}$$

Under these conditions, the PolSA algorithm is shown to couple with the Zap SA algorithm, and therefore have optimal asymptotic covariance [12]:

Proposition 3.1: Consider the linear model of the form

$$f_k(\theta) = A(\theta - \theta^*) + \Delta_k$$

where $\boldsymbol{\Delta}$ is a bounded martingale difference sequence. Let $\{\theta_n^*\}$ denote the iterates using the Zap SA algorithm ((22), with gain (29)), and $\{\theta_n\}$ the iterates obtained using the PolSA algorithm (24) with gain (30). Assume moreover that A is Hurwitz, and the eigenvalues of ζA lie within the open unit disc in $\mathbb{C}$. Then, there is a square-integrable random variable $\tilde{b}$ such that

$$\|\theta_n - \theta_n^*\| \le \tilde{b} n^{-1}, \quad n \ge 1. \tag{31}$$

Consequently, the PolSA algorithm has optimal asymptotic covariance. □

IV. Numerical Examples

A. User's guide

The following is a sampling of warnings to be kept in mind in application of any of these methods:

(i) Avoid inversion of matrices. In Matlab, replace the operation $A^{-1}b$ by $A\backslash b$.

(ii) Batch updates can reduce complexity substantially. For example, with batch length N, for each k,

$$\theta_{(k+1)N} = \theta_{kN} - \widehat{A}_{kN}^{-1} \sum_{n=kN}^{kN-1} \alpha_n d_{n+1} z_n \,. \tag{32}$$

and $\theta_{kN+i} = \theta_{kN}$ for each $1 \le i < N$. The choice $N = d$ reduces overall complexity by a factor of $1/d$, and does not appear to impact performance.

(iii) The following approximation of the Moore–Penrose pseudo inverse may be used as a substitute for $\widehat{A}_n^{-1}$ during the transient phase of the algorithm (though this has not been needed in our own applications):

$$\widehat{A}_n^\dagger = [n^{-1}I + \widehat{A}_n^\intercal \widehat{A}_n]^{-1} \widehat{A}_n^\intercal$$

(iv) Re-consider the use of a randomized policy in off-line applications (using simulated data). There is no reason to introduce randomness which will increase variance. The simplest deterministic scheme is *clock sampling*, in which state-action pairs (x^t, u^t) are chosen sequentially. At stage n, if (x, u) is the current pair, a random variable X'_{n+1} is chosen according to the distribution $P_u(x, \cdot)$, and the (x, u) entry of the Q-function is updated according to the particular algorithm using the triple (x, u, X'_{n+1}).
A significant change to Watkins' iteration is that the matrix gain (12) is replaced by $d^{-1}I$ in this case. This combined with deterministic sampling results in significant reduction in variance in many cases.

B. Comparing the algorithms

The algorithms compared here are all in the setting of tabular Q-learning, in which the basis functions coincide with Watkins' algorithm. Zap Q-learning has been tested in the general linear function approximation setting, with some results summarized in [6], [7].

The performance of several algorithms are compared here: Watkins' algorithm, the Zap-Q(λ) algorithm defined in (14, 15), along with PolSA (24) and the linear approximation of NeSA (28):

$$\begin{aligned} \text{PolSA:} \quad & \Delta\theta_{n+1} = [I + \zeta \widehat{A}_{n+1}]\Delta\theta_n + \zeta\alpha_n d_{n+1} z_n \\ \text{NeSA:} \quad & \Delta\theta_{n+1} = [I + \zeta A_{n+1}]\Delta\theta_n + \zeta\alpha_n d_{n+1} z_n \end{aligned} \tag{33}$$

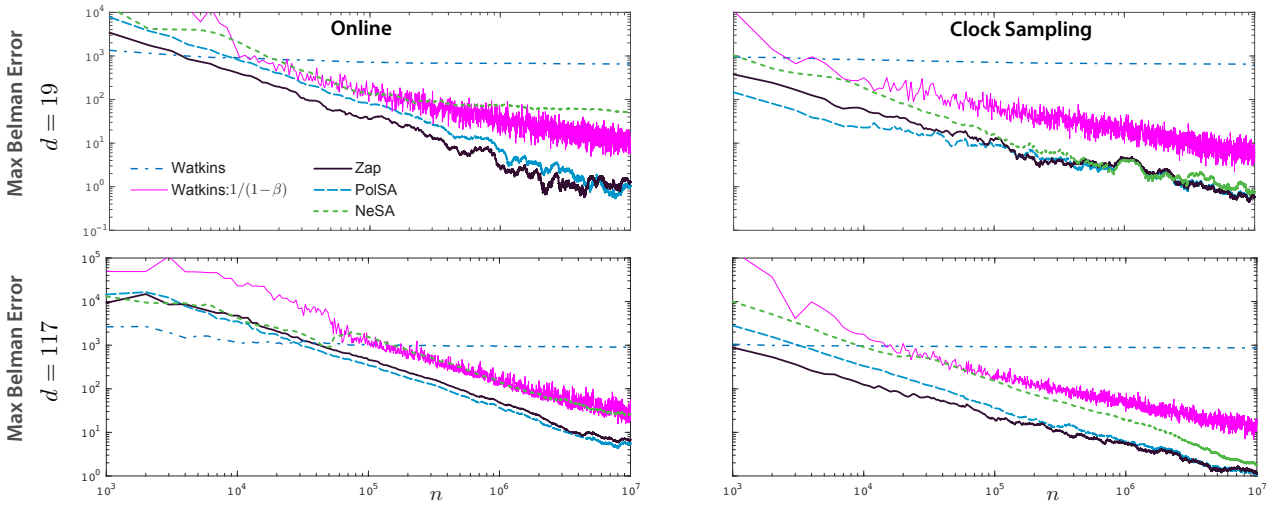

Fig. 2. Bellman error for $n \leq 10^7$ in the shortest path problem. The first row corresponds to a graph with $N = 10$ nodes, and 19 state-action pairs; the second $N = 20$ nodes, and 117 state-action pairs. Coupling of PolSA and Zap is evident in each of the four experiments. Deterministic exploration leads to much faster convergence for the larger graph.

The matrices $\widehat{A}_{n+1}$ and A_{n+1} are defined in (15), and the pair (d_{n+1}, z_n) is defined in (14). As in the classical setting, we let $\lambda = 0$ (performance for non-zero λ has not yet been explored).

The linearized NeSA algorithm is adopted here mainly to remind the reader that the recursion is very similar to the PolSA algorithm. Algorithms (25) and (28) nearly coincide in Q-learning: the updates differ only when $\phi_n \neq \phi_{n-1}$.

The *synchronous speedy Q-learning* recursion of [18] appears similar to the NeSA algorithm with clock sampling, following the heuristic swapping argument used to motivate the PolSA algorithm in the form (33).

The 6-node example of [6], [7] was revisited to investigate the rate of convergence in the CLT. The value $\zeta = 1$ was chosen in each of the two algorithms displayed in (33). Fig. 1 shows histograms for $\{\sqrt{n}\tilde{\theta}_n\}$, $n = 10^6$, based on independent repeated experiments for four algorithms. Those for PolSA and Zap nearly coincide. Watkins' Q-learning algorithm has excellent performance in this small example, when the gain (13) is used. The histogram for NeSA shows much higher variance.

Experiments were repeated for larger examples. It was useful to reduce the gain slightly in the new algorithms: $\zeta = 0.8$ for PolSA and $\zeta = 0.9$ for NeSA.

As in the 6-node example, the MDP model is a stochastic shortest path problem. The model construction was based on the creation of a graph with N nodes, in which the probability of an edge between a pair of nodes is i.i.d. with probability p. Additional edges $(i, i+1)$ are added, for each $i < N$, to ensure the resulting graph is strongly connected.

The state space X coincides with the set of nodes, and the action space U the set of edges.

The transition law is similar to the 6-state example: with probability 0.8 the agent moves in the desired direction (indicated by the input), and with remaining probability any of the neighboring nodes is arrived at with uniform probability, for each state $i < N$. From state N, the chain regenerates with uniform distribution over $\{1, \ldots, N-1\}$.

Two exploration rules were considered: the "online" (or asynchronous) version that has been the focus of this paper, and the offline "clock sampling" (or synchronous) approach in which state-action pairs (x^i, u^i) are chosen sequentially. The cost function was chosen to be independent of u, with $c(i) = 10^3$ for all nodes except the terminal node, for which $c(N) = 1$.

A metric for success is the maximal Bellman error, $\mathcal{E}_B(\theta) = \max_{x,u} |\mathcal{B}^\theta(x,u)|$. A plot of $\mathcal{E}_B(\theta_n)$ vs. n is shown in Fig. 2 for several algorithms, and two different models. In each case, Zap and PolSA are quickest to converge, by far, but the other two algorithms require much lower computation per iteration. While Theorem 2.2 has been proven only for a special linear setting, the coupling of the Zap and PolSA trajectories is evident in these experiments.

V. Conclusions

It is fascinating to see how the asymptotic theory of stochastic approximation can be used to obtain new optimized reinforcement-learning algorithms with super-fast convergence rate. Moreover, in experiments we observe that the asymptotic theory holds in practice after a reasonable number of iterations.

It is important to recall the crucial role of the constants multiplying the step-size sequence $\{\alpha_n\}$. It is common to ignore this because they are diminishing, but using a constant multiplier $\frac{1}{2}(1-\beta)^{-1}$ instead of $(1-\beta)^{-1}$ in (13) can make a difference between infinite and finite asymptotic variance in Watkins' Q-Learning. How these constants can be found for a general stochastic approximation algorithm is a work-in-progress.

It is also exciting to see how the intuitive transformation from stochastic Newton-Raphson to the momentum based methods of PolSA and NeSA can be justified theoretically and in simulations. Other than the known algorithms of Ruppert-Polyak averaging and SNR/Zap, PolSA is the

third algorithm that achieves optimal asymptotic variance [12]. Moreover, its transient performance is far better than Ruppert-Polyak averaging, and computational complexity much lower compared to Zap, since there is no matrix inversion step.

While the covariance of NeSA is not optimal, it is extremely simple to implement with low computational cost, and it also performs great in most of our experiments. Its transient performance was found to be much better than Polyak-Ruppert averaging. Why the averaging technique has such bad transients is an open question. The answer *may* lie in the analysis of rates of convergence of CLT for the algorithm.

An important next step is to formulate adaptive techniques to ensure fast convergence in terms of both asymptotics and transient response. One option is to optimize the gain ζ appearing in in (24) and (25), and then formulating algorithms to estimate this value on-line. It is possible that techniques in [24] may be adapted. We do not have a natural notion of expected loss in the general root finding problem, but it will be of interest to pursue analysis in the special case of nonlinear optimization.

REFERENCES

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

[2] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, King's College, Cambridge, Cambridge, UK, 1989.

[3] A. Benveniste, M. Métivier, and P. Priouret, *Adaptive algorithms and stochastic approximations*, ser. Applications of Mathematics (New York). Berlin: Springer-Verlag, 1990, vol. 22, translated from the French by Stephen S. Wilson.

[4] H. J. Kushner and G. G. Yin, *Stochastic approximation algorithms and applications*, ser. Applications of Mathematics (New York). New York: Springer-Verlag, 1997, vol. 35.

[5] V. S. Borkar, *Stochastic Approximation: A Dynamical Systems Viewpoint*. Delhi, India and Cambridge, UK: Hindustan Book Agency and Cambridge University Press (jointly), 2008.

[6] A. M. Devraj and S. P. Meyn, "Fastest convergence for Q-learning," *ArXiv e-prints*, Jul. 2017.

[7] A. M. Devraj and S. Meyn, "Zap Q-Learning," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 2235–2244.

[8] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.

[9] ——, *Introduction to Optimization*. New York: Optimization Software Inc, 1987.

[10] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $O(1/k^2)$," in *Soviet Mathematics Doklady*, 1983.

[11] ——, "Efficiency of coordinate descent methods on huge-scale optimization problems," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 341–362, 2012.

[12] A. M. Devraj, A. Bušić, and S. Meyn, "Zap meets momentum: Stochastic approximation algorithms with optimal convergence rate," *arXiv preprint arXiv:1809.06277*, 2018.

[13] V. R. Konda and J. N. Tsitsiklis, "Convergence rate of linear two-time-scale stochastic approximation," *Ann. Appl. Probab.*, vol. 14, no. 2, pp. 796–819, 2004.

[14] D. Ruppert, "A Newton-Raphson version of the multivariate Robbins-Monro procedure," *The Annals of Statistics*, vol. 13, no. 1, pp. 236–245, 1985.

[15] B. T. Polyak, "A new method of stochastic approximation type," *Avtomatika i telemekhanika (in Russian). translated in Automat. Remote Control, 51 (1991)*, pp. 98–107, 1990.

[16] B. T. Polyak and A. B. Juditsky, "Acceleration of stochastic approximation by averaging," *SIAM J. Control Optim.*, vol. 30, no. 4, pp. 838–855, 1992.

[17] C. Szepesvári, "The asymptotic convergence-rate of Q-learning," in *Proceedings of the 10th International Conference on Neural Information Processing Systems*. MIT Press, 1997, pp. 1064–1070.

[18] M. G. Azar, R. Munos, M. Ghavamzadeh, and H. Kappen, "Speedy Q-learning," in *Advances in Neural Information Processing Systems*, 2011.

[19] E. Moulines and F. R. Bach, "Non-asymptotic analysis of stochastic approximation algorithms for machine learning," in *Advances in Neural Information Processing Systems 24*. Curran Associates, Inc., 2011, pp. 451–459.

[20] D. Kalathil, V. S. Borkar, and R. Jain, "Empirical q-value iteration," *arXiv preprint arXiv:1412.0180*, 2014.

[21] W. B. Haskell, R. Jain, and D. Kalathil, "Empirical dynamic programming," *Mathematics of Operations Research*, vol. 41, no. 2, pp. 402–429, 2016.

[22] Z. Allen-Zhu, "Katyusha: The first direct acceleration of stochastic gradient methods," *ArXiv e-prints*, Mar. 2016.

[23] A. Defazio, F. Bach, and S. Lacoste-Julien, "Saga: A fast incremental gradient method with support for non-strongly convex composite objectives," in *Advances in neural information processing systems*, 2014, pp. 1646–1654.

[24] P. Jain, S. M. Kakade, R. Kidambi, P. Netrapalli, and A. Sidford, "Accelerating Stochastic Gradient Descent," *ArXiv e-prints (and to appear, COLT 2018)*, Apr. 2017.

[25] F. Bach and E. Moulines, "Non-strongly-convex smooth stochastic approximation with convergence rate $o(1/n)$," in *Advances in Neural Information Processing Systems 26*. Curran Associates, Inc., 2013, pp. 773–781.

[26] S. Gadat, F. Panloup, and S. Saadane, "Stochastic heavy ball," *Electron. J. Statist.*, vol. 12, no. 1, pp. 461–529, 2018.

[27] J. Duchi, "Introductory lectures on stochastic optimization," *Stanford Lecture Series*, 2016.

[28] N. Loizou and P. Richtárik, "Momentum and Stochastic Momentum for Stochastic Gradient, Newton, Proximal Point and Subspace Descent Methods," *ArXiv e-prints*, Dec. 2017.