Vector Quantization for Adaptive State Aggregation in Reinforcement Learning*

Christos N. Mavridis and John S. Baras

***Abstract*— We propose an adaptive state aggregation scheme to be used along with temporal-difference reinforcement learning and value function approximation algorithms. The resulting algorithm constitutes a two-timescale stochastic approximation algorithm with: (a) a fast component that executes a temporal-difference reinforcement learning algorithm, and (b) a slow component, based on online vector quantization, that adaptively partitions the state space of a Markov Decision Process according to an appropriately defined dissimilarity measure. We study the convergence of the proposed methodology using Bregman Divergences as dissimilarity measures that can increase the efficiency and reduce the computational complexity of vector quantization algorithms. Finally, we quantify its performance on the Cart-pole (inverted pendulum) optimal control problem using Q-learning with adaptive state aggregation based on the Self-Organizing Map (SOM) algorithm.**

I. INTRODUCTION

Reinforcement learning and value function approximation algorithms have been receiving increasing attention recently and have resulted in impressive applications in multiple fields including elevator control, robot soccer [1], and game-playing, such as Backgammon, Chess and Go [2], [3].

Temporal-difference learning methods dominate current reinforcement learning algorithms due to their data efficiency. When applied to a Markov Decision Process (MDP) with finite state and action space, function approximation using lookup tables has been well established [4], [5]. Due to the exponential increase of states with respect to the dimensionality of the state space, however, parametric models have been widely studied for function approximation, and in particular linear combinations of fixed basis functions, such as artificial neural networks [6], [7]. While RL algorithms based on parametric models can deal remarkably well with the dimensionality issues, convergence properties can be difficult to establish, especially in the nonlinear case or in off-policy scenarios [5], [8], and their performance in practice heavily depends on the choice of the basis functions [9]. On the other hand, lookup tables can hardly be used in MDPs with large state spaces. For this reason, state aggregation has been proposed as a quantization scheme for large or infinite spaces, and can be viewed as a special case of linear models with the basis functions being indicator functions of a partition of the state space [7]. Although this simplicity of the feature space is often desirable, most state aggregation schemes are typically fixed and ad-hoc, and, as a result, constitute a sub-optimal representation of the state space.

Related Work. State aggregation has shown promising results and has been widely studied in Value Function Approximation (VFA) settings. George et al. in [10] propose an ad-hoc state aggregation scheme in multiple resolutions and use a linear combination of the estimates in each resolution in a VFA algorithm. Fernández et al. in [11] use a generalized Lloyd vector quantization algorithm to discretize the state space before running a Q-learning algorithm. Singh et al. in [12] use a heuristic adaptive state aggregation scheme based on soft clustering, and Lee et al. in [13] propose an adaptive state space partitioning based on vector quantization. In both cases, the methods are assessed experimentally and no formal analysis is given. Bertsekas and Castanon in [14] use an adaptive aggregation method which is applicable to offline infinite horizon dynamic programming, and Baras and Borkar in [15], first propose the use of learning vector quantization in conjunction with a policy iteration algorithm which result in a multiple-timescale stochastic approximation algorithm. Finally, Sehad et al. in [16] and Montazeri et al. in [17] make use of variants of the Self-Organizing Map algorithm ([18]) to solve reinforcement learning problems with continuous state spaces.

Contribution. In this work, we consider an MDP with infinite state space and propose a state aggregation algorithm that adaptively groups together states, according to an appropriately defined dissimilarity measure, as they are being observed by a temporal-difference reinforcement learning algorithm. We use online Vector Quantization (VQ), which can be viewed as a stochastic approximation algorithm [19], and form a two-timescale stochastic approximation algorithm [20] with: (a) a fast component corresponding to the temporal-difference learning algorithm, and (b) a slow component corresponding to the adaptive state aggregation scheme. Intuitively, the resulting algorithm approximates the value function with a linear combination of indicator functions of dynamically changing partitions of the state space. We make use of Bregman divergences [21] as dissimilarity measures that can increase the efficiency and reduce the complexity of VQ algorithms, and study the convergence properties of the proposed algorithm based on existing results in VFA [7] and stochastic approximation [22]. Finally, we quantify the performance of the proposed methodology on the inverted pendulum (Cart-pole [23]) optimal control problem using Q-learning with adaptive state aggregation based on the Self-Organizing Map (SOM) algorithm [18], a VQ variant with lattice topology [24].

The authors are with the Electrical and Computer Engineering Department and the Institute for Systems Research, University of Maryland, College Park, USA. emails:{mavridis, baras}@umd.edu

*This work was partially supported by by ONR grant N00014-17-1-2622, and by a grant from Northrop Grumman Corporation.

II. Mathematical Background and Notation

A. Reinforcement Learning

We consider a discrete-time MDP $(\mathcal{X}, \mathcal{U}, \mathcal{P}, C)$ with:

- $\mathcal{X}$ being the state space,
- $\mathcal{U}$ being the action (control) space,
- $\mathcal{P} : (x, u, x') \mapsto \mathbb{P}[x'|x, u]$ being the transition probabilities associated with a stochastic state transition function $f : (x, u) \mapsto x'$, and
- $C : \mathcal{X} \times \mathcal{U} \to \mathbb{R}_+$, being the immediate cost function, assumed deterministic.

Reinforcement Learning (RL) examines the problem of learning a control policy $u := (u_0, u_1, \ldots)$ that solves the discounted infinite-horizon optimal control problem

$$\min_u \mathbb{E}\left[\sum_{l=0}^{\infty} \gamma^l C(x_l, u_l)\right]$$

where $\gamma \in (0, 1]$.

We define the value function V^u of a policy u as

$$\begin{aligned} V^u(x_k) :&= \mathbb{E}\left[\sum_{l=k}^{\infty} \gamma^{l-k} C(x_l, u_l)\right] \\ &= C(x_k, u_k) + \gamma \mathbb{E}\left[V^u(x_{k+1}) \mid x_k\right] \\ &= Q^u(x_k, u_k) \end{aligned}$$

where Q^u represents the quality function of a policy u, i.e. the expected return for taking action u_k at time k and state x_k, and thereafter following policy u. As a result of Bellman's principle, we get the (discrete-time) Hamilton-Jacobi-Bellman (HJB) equation

$$\begin{aligned} V^*(x_k) :&= \min_u \ \mathbb{E}\left[\sum_{l=k}^{\infty} \gamma^{l-k} C(x_l, u_l)\right] \\ &\overset{(HJB)}{=} \min_u \{\ C(x_k, u_k) + \gamma \mathbb{E}\left[V^*(x_{k+1}) \mid x_k\right]\ \} \\ &= \min_{u_k} Q^*(x_k, u_k) \end{aligned} \tag{1}$$

where $V^* := V^{u^*}$ and $Q^* := Q^{u^*}$ represent the optimal value and Q functions, respectively.

Reinforcement learning algorithms consist mainly of temporal-difference learning algorithms [25] that try to approximate a solution to (1) using iterative optimization methods. The optimization is performed over a finite set of parameters which are used to describe the value (or Q) function. These parameters typically correspond to a parametric model (e.g. a neural network) used for function approximation, or to the different values of the vector $V(\mathcal{X})$ (or $Q(\mathcal{X}, \mathcal{U})$), in which case $\mathcal{X}$ and $\mathcal{U}$ are assumed finite either by definition or as a result of discretization.

In this work we will assume that the state space is a finite-dimensional vector space that has been appropriately discretized in a finite set. When $\mathcal{X}$ and $\mathcal{U}$ are finite, a widely used approach is the $Q-$learning algorithm (Alg. 1), which is a stochastic approximation algorithm [26] that asymptotically minimizes the Mean-Squared Bellman Error

$$\min_q \ \mathbb{E}\left[\|C(x, u) + \min_u \{\ \gamma Q(x', u)\ \} - q\|^2 \mid x\right].$$

$Q-$learning depends on a stochastic exploration policy $\pi_L = u'$ which decides the next action given the observed state. Given the policy π_L, the original MDP becomes a Markov Chain, and the $Q-$learning algorithm becomes an off-policy $TD(0)$ algorithm [5] for Value Function Approximation (VFA):

$$\begin{aligned} V_{j+1}^{\pi_L}(x) = V_j^{\pi_L}(x) \\ + \alpha_j \left[C(x, \pi_L(x)) + \gamma V_j^{\pi_G}(x') - V_j^{\pi_L}(x)\right] \end{aligned}$$

where $\pi_G(x) \in \arg\min_u \{V^u(f(x, u))\}$ (in the case of the on-policy counterpart of Q-learning, i.e. SARSA, we have $\pi_G = \pi_L$).

It is apparent that temporal-difference reinforcement learning and value function approximation are closely related, with the latter often being studied independently and used for policy evaluation [9]. For this reason, we will describe the proposed methodology in the general setting of temporal-difference value function approximation and test its performance using $Q-$learning.

Algorithm 1 $Q-$learning

Initialize $Q_0(x, u)$, $\forall x \in \mathcal{X},\ u \in \mathcal{U}$
repeat
  Choose $u' = \pi_L(x)$ ▷ π_L: exploration policy
  Observe $x' = f(x, u')$ ▷ f: state transition function
  Update $Q-$function:

$$\begin{aligned} Q_{j+1}(x, u') = Q_j(x, u') + \alpha_j [C(x, u') \\ + \gamma \min_u Q_j(x', u) - Q_j(x, u')] \end{aligned}$$

until Convergence criterion is satisfied
Update Policy:

$$u_\epsilon(x) = \arg\min_u \{\ Q_\epsilon(x, u)\ \}$$

B. Vector Quantization

In this section, we introduce the online Vector Quantization (VQ) algorithm to solve the unsupervised problem of prototype-based clustering using a dissimilarity measure, which can be stated as an optimization problem:

Problem 1: Let $X : \Omega \to S$ be a random variable defined in the probability space $(\Omega, \mathcal{F}, \mathbb{P})$, and $d : S \times ri(S) \to [0, \infty)$ be a divergence measure. Let $V := \{S_h\}_{h=1}^k$ be a partition of S with respect to d and $M := \{\mu_h\}_{h=1}^k$, such that $\mu_h \in ri(S_h),\ h \in K,\ K := \{1, \ldots, k\}$. A quantizer $Q : S \to S$ is defined such that $Q(X) = \sum_{h=1}^k \mu_h \mathbb{1}_{[X \in S_h]}$ and the problem is formulated as

$$\min_{M,V} \ J(Q) := \mathbb{E}\left[d\left(X, Q(X)\right)\right]$$

Assuming M is a deterministic function of X, Problem 1 becomes equivalent with

$$\min_{\{\mu_h\}_{h=1}^k} \sum_{h=1}^{k} \mathbb{E}_X\left[d\left(X, \mu_h\right) \mathbb{1}_{[X \in S_h]}\right] \tag{2}$$

for V being a Voronoi partition, i.e. for

$$S_h = \left\{ X \in S : h = \underset{\tau=1,\dots,k}{\arg\min}\ d(X, \mu_\tau) \right\},\ h \in K \quad (3)$$

It is typically the case that the actual distribution of $X \in S$ is unknown, and a set of independent realizations $\{X_i\}_{i=1}^n := \{X(\omega_i)\}_{i=1}^n$, for $\omega_i \in \Omega$, are available. In case the observations $\{X_i\}_{i=1}^n$ are available a priori, the solution of the VQ problem is traditionally approached with variants of the LBG algorithm [27], which is essentially a block optimization algorithm that solves (2) and (3) iteratively, until convergence.

When the observed data are not available a priori but are being acquired online, a stochastic (online) VQ algorithm (sVQ) can be defined as a recursive algorithm:

Definition 1 (Stochastic Vector Quantization Algorithm):

$$\begin{cases} \mu_h^{t+1} & = \mu_h^t - \alpha(v(h,t)) \mathbb{1}_{\left[X_{t+1} \in S_h^{t+1}\right]} \nabla_{\mu_h} d\left(X_{t+1}, \mu_h^t\right) \\ S_h^{t+1} & = \left\{ X \in S : h = \underset{\tau=1,\dots,k}{\arg\min}\ d(X, \mu_\tau^t) \right\},\ h \in K \end{cases} \quad (4)$$

for every $t \geq 0$, where μ_h^0 is given during initialization, and $v(h,t)$ represents the number of times the component μ_h has been updated up until time t.

It can be shown ([19]) that algorithm (5) is an asynchronous stochastic approximation algorithm of the form:

$$\mu^{t+1} = \mu^t + \alpha(t) \left[\theta^t(\mu) + M^{t+1} \right] \quad (5)$$

where $\theta_h^t(\mu) = \mathbb{E}_X \left[\Theta_h(\mu^t, X_{t+1}) \right]$ *a.s.*, $\Theta_h(\mu, X) = \left(-\mathbb{1}_{[X \in S_h]} \right) \nabla_{\mu_h} d\left(X, \mu_h\right)$, and M_h^t is the martingale difference $M_h^{t+1} := \Theta_h(\mu^t, X_{t+1}) - \theta_h^t(\mu)$. Therefore, invoking Theorem 2 and Corollary 2.1 of [19], we can show that the stochastic vector quantization algorithm (5) converges to an invariant set of the O.D.E.

$$\dot{\mu} = \theta(\mu) = -\nabla_\mu J(\mu)$$

where $J(\mu) = \mathbb{E}_X \left[d\left(X, Q(X)\right) \right]$.

C. Bregman Divergences as Dissimilarity Measures

As shown above, vector quantization algorithms depend on a dissimilarity measure. In most cases this is assumed to be a convex metric, and in particular the Euclidean distance, making the VQ algorithms applicable only to metric spaces. However, this measure can be generalized to dissimilarity measures inspired by information theory and statistical analysis, such as the Bregman divergence:

Definition 2 (Bregman Divergence): Let $\phi : H \to \mathbb{R}$, be a strictly convex function defined on a normed vector space $dom(\phi) = H$ such that ϕ is twice F-differentiable on H. The Bregman divergence $d_\phi : H \times H \to [0, \infty)$ is defined as:

$$d_\phi\left(x, \mu\right) = \phi\left(x\right) - \phi\left(\mu\right) - \frac{\partial \phi}{\partial \mu}\left(\mu\right)\left(x - \mu\right),$$

where $x, \mu \in H$, and the continuous linear map $\frac{\partial \phi}{\partial \mu}\left(\mu\right) : H \to \mathbb{R}$ is the Fréchet derivative of ϕ at μ.

In this work, we will concentrate on nonempty, compact convex sets $S \subseteq \mathbb{R}^d$ so that the derivative of d_ϕ with respect to the second argument can be written as

$$\begin{aligned} \frac{\partial d_\phi}{\partial \mu}(x, \mu) &= \frac{\partial \phi(x)}{\partial \mu} - \frac{\partial \phi(\mu)}{\partial \mu} - \frac{\partial^2 \phi(\mu)}{\partial \mu^2}(x - \mu) + \frac{\partial \phi(\mu)}{\partial \mu} \\ &= -\frac{\partial^2 \phi(\mu)}{\partial \mu^2}(x - \mu) = -\left\langle \nabla^2 \phi(\mu), (x - \mu) \right\rangle \end{aligned}$$

where $x, \mu \in S$, $\frac{\partial}{\partial \mu}$ represents differentiation with respect to the second argument of d_ϕ, and $\nabla^2 \phi(\mu)$ represents the Hessian matrix of ϕ at μ.

Example 1: As a first example, $\phi(x) = \langle x, x \rangle,\ x \in \mathbb{R}^d$, gives the squared Euclidean distance

$$d_\phi(x, \mu) = \|x - \mu\|^2$$

for which $\frac{\partial d_\phi}{\partial \mu}(x, \mu) = -2(x - \mu)$.

Example 2: A second interesting Bregman divergence that shows the connection to information theory, is the generalized I-divergence which results from $\phi(x) = \langle x, \log x \rangle,\ x \in \mathbb{R}_{++}^d$ such that

$$d_\phi(x, y) = \langle x, \log x - \log \mu \rangle - \langle \mathbb{1}, x - \mu \rangle$$

for which $\frac{\partial d_\phi}{\partial \mu}(x, \mu) = -diag^{-1}(\mu)(x - \mu)$, where $\mathbb{1} \in \mathbb{R}^d$ is the vector of ones, and $diag^{-1}(\mu) \in \mathbb{R}_{++}^{d \times d}$ is the diagonal matrix with diagonal elements the inverse elements of μ. It is easy to see that $\phi(x)$ reduces to the Kullback-Leibler divergence if $\langle \mathbb{1}, x \rangle = 1$.

A key property of Bregman divergences in vector quantization is that their use in batch algorithms based on the generalized Lloyd algorithm, is both necessary and sufficient for local convergence, which follows from the following theorem proven in [21]:

Theorem 1: Let $X : \Omega \to S$ be a random variable defined in the probability space $(\Omega, \mathcal{F}, \mathbb{P})$ such that $\mathbb{E}\left[X\right] \in ri(S)$, and let a distortion measure $d : S \times ri(S) \to [0, \infty)$, where $ri(S)$ denotes the relative interior of S. Then $\mu := \mathbb{E}\left[X\right]$ is the unique minimizer of $\mathbb{E}\left[d\left(X, s\right)\right]$ in $ri(S)$, if and only if d is a Bregman Divergence for any function ϕ that satisfies the definition.

This result allows for the optimizer μ_h of (2) to be analytically computed as the sample mean of the data inside S_h. Moreover, when $d := d_\phi$ is a Bregman divergence, and under some mild conditions on the function ϕ (see [19]), (5) can be shown to converge to an asymptotically stable equilibrium point μ^*, for some domain of attraction D^* that depends on the initial conditions μ_h^0 and the sample path X_t, which, at least locally, minimizes $J(\mu)$.

D. Self-Organizing Map

A variant of the sVQ algorithm (4) first introduced by Kohonen [18], is the Self-Organizing Map (SOM) algorithm. SOM is a competitive-learning neural network with the neurons (cluster representatives μ) arranged in a lattice topology. This introduces the notion of a neighborhood which is measured via a neighborhood function and is incorporated in the training rule. In this work, we adopt a Gaussian

neighborhood function and define the SOM algorithm as follows:

Definition 3 (Self-Organizing Map (SOM) Algorithm):

$$\begin{cases} \mu_h^{t+1} & = \mu_h^t - \alpha_t G(\mu_h^t, \mu_{h^*}^t, \Sigma(t)) \nabla_{\mu_h} d_\phi \left(X_{t+1}, \mu_h^t\right) \\ h^* & = \underset{\tau=1,\ldots,k}{\arg\min}\ d_\phi(X_{t+1}, \mu_\tau^t) \end{cases} \tag{6}$$

where

$$G(x, m, \Sigma) = \frac{1}{\sqrt{(2\pi)^k |\Sigma|}} e^{-\frac{1}{2}(x-m)^T \Sigma (x-m)}$$

is a Gaussian neighborhood function with respect to a predefined lattice topology of μ, and μ_h^0 is given during initialization.

In contrast with sVQ, where only the weight vector closest to each observation is updated, SOM updates all the representing vectors in a neighborhood of the winner vector with different weights, depending on the value of the neighborhood function, centered at the winner neuron. The neighborhood typically becomes smaller over time, and it has been shown [24] that, similar to sVQ, SOM can be viewed as a convergent stochastic approximation algorithm of the form (5).

III. State Aggregation with Vector Quantization

We consider an MDP $(S, \mathcal{U}, \mathcal{P}, C)$, where $S \subseteq \mathbb{R}^d$ is a compact convex set. We define a quantizer $Q(X) = \sum_{h=1}^k \mu(h) \mathbb{1}_{[X \in S_h]}$, where $\{S_h\}_{h=1}^k$ is a partition of S. The quantizer Q defines a state aggregation scheme with k clusters (aggregate states), each represented by $\mu(h),\ h = 1, \ldots, k$. We define the new aggregate state space as $\mathcal{X} = \{X_1, \ldots, X_k\}$ and a new MDP $(\mathcal{X}, \mathcal{U}, \mathcal{P}', C')$ on the aggregate states with transition probabilities

$$\mathbb{P}'\left[X_k | X_l, u\right] := \mathbb{P}\left[x' \in S_k | x \in S_l, u\right] \in \mathcal{P}' \tag{7}$$

as argued in [7], [12]. Under a given policy π_L, this MDP reduces to a Markov Chain. We represent the cost-to-go value function estimates for each aggregate state, given π_L, as a vector $V^{\pi_L} := (V^{\pi_L}(1), \ldots, V^{\pi_L}(k)) \in \mathbb{R}_+^k$. Notice that $V^{\pi_L}(h),\ h = 1, \ldots, k$ essentially represents the common cost-to-go value of all the states in the set S_h, which depends on the value μ_h. It is natural to seek for a partition $S_h,\ h = 1, \ldots, k$ such that the aggregate states represent the actual state space S in an optimal way according to some measure. We address this problem by running an sVQ algorithm along with a temporal-difference value iteration algorithm in order to dynamically approximate an optimal partition with respect to a dissimilarity measure expressed as a Bregman divergence. The resulting value function approximation algorithm with adaptive state aggregation constitutes a two-timescale stochastic approximation algorithm:

Definition 4: The value function approximation algorithm with adaptive state aggregation based on Vector Quantization reads as:

$$\begin{cases} V_{i+1}^{\pi_L}(h^*) & = V_i^{\pi_L}(h^*) \\ & \quad + \alpha_i \left[C(x, \pi_L(x)) + \gamma V_i^{\pi_G}(h') - V_i^{\pi_L}(h^*)\right] \\ \mu_{i+1}(h^*) & = \mu_i(h^*) - \beta_i \nabla_\mu d\left(x, \mu_i(h^*)\right) \\ h^* & = \underset{\tau=1,\ldots,k}{\arg\min}\ d(x, \mu_i(\tau)) \end{cases} \tag{8}$$

where $x \in S$ is the current state, $x' = f(x, \pi_L(x)) \in S$ is the next state with f being an unknown stochastic state transition function, $h' = \{h : x' \in S_h\}$, and $V^{\pi_G}(h) = min_u \left\{C(x, u) + \gamma V^{\pi_L}(h)\right\}$. The stepsizes α_i and β_i in (8) satisfy $\sum_i \alpha_i = \sum_i,\ \beta_i = \infty,\ \sum_i (\alpha_i^2 + \beta_i^2) < \infty$, and $\beta_i/\alpha_i \to 0$.

The condition $\beta_i/\alpha_i \to 0$ is of great importance. Intuitively, the stochastic approximation algorithm (8) consists of two components running in different timescales. The slow component $\mu(\cdot)$ corresponds to the vector quantization algorithm and is viewed as quasi-static when analyzing the behavior of the fast transient $V(\cdot)$ which corresponds to the value function approximation algorithm. As an example, the condition $\beta_n/\alpha_n \to 0$ is satisfied by stepsizes of the form $(\alpha_n, \beta_n) = (1/n, 1/1+n\log n)$, or $(\alpha_n, \beta_n) = (1/n^{2/3}, 1/n)$. Another way of achieving the two-timescale effect is to run the iterations for the slow component $\{\mu_n\}$ with stepsizes $\{\alpha_{n(k)}\}$, where $n(k)$ is a subsequence of n that becomes increasingly rare (i.e. $n(k+1) - n(k) \to \infty$), while keeping its values constant between these instants. In practice, it has been observed that a good policy is to run the slow component with slower stepsize schedule β_n and update it along a subsequence keeping its values constant in between ([22], Ch. 6).

A. Convergence Analysis

The convergence properties of the algorithm can be studied by directly applying the theory of the O.D.E. method for stochastic approximation in multiple timescales as detailed in [22]. In the case of two timescales, we have the following theorem proven by Borkar in Ch. 6 of [20]:

Theorem 2: Consider the sequences $\{x_n\} \in S \subseteq \mathbb{R}^d$ and $\{y_n\} \in \Sigma \subseteq \mathbb{R}^k$, generated by the iterative scheme:

$$\begin{aligned} x_{n+1} &= x_n + \alpha(n) \left[h(x_n, y_n) + M_{n+1}^{(x)}\right] \\ y_{n+1} &= y_n + \beta(n) \left[g(x_n, y_n) + M_{n+1}^{(y)}\right] \end{aligned} \tag{9}$$

for $n \geq 0$, and assume that

$$\sum_n \alpha(n) = \sum_n \beta(n) = \infty,$$
$$\sum_n (\alpha^2(n) + \beta^2(n)) < \infty,$$
$$\frac{\beta(n)}{\alpha(n)} \to 0.$$

with the last condition implying that the iterations for $\{y_n\}$ run on a slower timescale than those for $\{x_n\}$. If the equation

$$\dot{x}(t) = h(x(t), y),\ x(0) = x_0$$

has an asymptotically stable equilibrium $\lambda(y)$ for fixed y and some Lipschitz mapping λ, and the equation

$$\dot{y}(t) = g(\lambda(y(t)), y(t)),\ y(0) = y_0$$

has an asymptotically stable equilibrium y^*, then, almost surely, (x_n, y_n) converges to $(\lambda(y^*), y^*)$.

We have shown that the sVQ algorithm (4) is a convergent stochastic approximation algorithm [19], and it has been shown that the $Q-$learning algorithm, and, as a result, the value function approximation component of (8) is a convergent stochastic approximation algorithm as well [4], [26]. Based on the Definitions 1 and 4, and using (7), and Theorem 2, we can prove the following theorem:

Theorem 3: Algorithm (8) converges almost surely to (V^*, μ^*) where μ^* is a solution to the divergence-based vector quantization problem (Problem 1) defined in the state space S, and $V^* := (V_1^*, \ldots, V_k^*)$ is the solution of

$$V_h = \min_u\ \mathbb{E}\left[C(x,u)|x \in S_h\right] + \gamma \mathbb{E}'\left[V_{h'}|X_h\right]$$

where $S_h = \{x \in S : h = \arg\min_{\tau=1,\ldots,k}\ d(x, \mu^*(\tau))\}$ are assumed to be visited infinitely often, $h = 1, \ldots, k$, and $\mathbb{E}'$ is defined with respect to the probability measure (7).

IV. Experimental Results

A. The CartPole Problem

We assess the efficacy of the proposed methodology on the Cart-pole (inverted pendulum) problem. The state variable of the cart-pole system (Fig. 1) has four components $(x, \theta, \dot{x}, \dot{\theta})$, where x and $\dot{x}$ are the position and velocity of the cart on the track, and θ and $\dot{\theta}$ are the angle and angular velocity of the pole with the vertical. The cart is free to move within the bounds of a one-dimensional track. The pole is free to move only in the vertical plane of the cart and track.

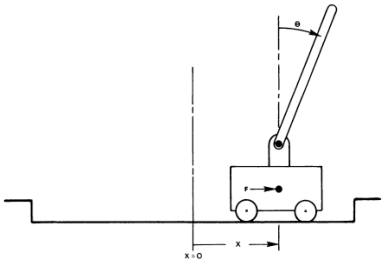

Fig. 1. Cart-pole system to be controlled ([23]).

The action space consists of an impulsive "left" or "right" force $F \in \{-10, +10\}$N of fixed magnitude to the cart at discrete time intervals. The cart-pole system is modeled by the following nonlinear system of differential equations [23]:

$$\ddot{x} = \frac{F + ml\left(\dot{\theta}^2 \sin\theta - \ddot{\theta}\cos\theta\right)\mu_c sgn(\dot{x})}{m_c + m}$$

$$\ddot{\theta} = \frac{g\sin\theta + \cos\theta\left(\frac{-F - ml\dot{\theta}^2\sin\theta + \mu_c sgn(\dot{x})}{m_c+m}\right) - \frac{\mu_p}{ml}\mu_c sgn(\dot{x})}{l\left(\frac{4}{3} - \frac{m\cos^2\theta}{m_c+m}\right)}$$

where the parameter values for $g, m_c, m, l, \mu_c, \mu_p$ can be found in [23]. The transition function for the state x is $x_{n+1} = x_n + \tau\dot{x}$, where $\tau = 0.02$s. The initial state is set to $X_0 = (u_x, u_\theta, u_{\dot{x}}, u_{\dot{\theta}})$ where u_x, u_θ, $u_{\dot{x}}$, and $u_{\dot{\theta}}$ follow a uniform distribution $U(-0.05, 0.05)$. Failure occurs when $|\theta| > 12^\circ$ or when $|x| > 2.4$m.

B. Q-Learning with Adaptive State Aggregation

While sVQ in (4) updates only the winner neuron at each iteration, in the context of state aggregation, the topological properties of SOM, which allow for updating all neurons in a given neighborhood at each iteration, can accelerate convergence, promote smoother transitions between iterations, and avoid poor local optima. Therefore, we introduce a Q-learning algorithm with Adaptive State Aggregation based on SOM, as a variant of (8). The full algorithm is shown in Alg. 2. As described above, we use Gaussian neighborhood function for the SOM algorithm with Σ being a diagonal matrix of elements that decrease over time. We train the algorithm with π_L being the $\epsilon-$greedy exploration policy, and stepsizes $\alpha_n = 1/1+(\alpha_0+n\delta_\alpha)$ and $\beta_n = 1/1+(\beta_0+n\log n\delta_\beta)$ for appropriate values of α_0, β_0, δ_α, δ_β. We ensure the two-timescale effect by progressively increasing N. We use the Euclidean distance as the Bregman divergence d_ϕ.

Algorithm 2 $Q-$learning with SOM-based state aggregation

Initialize $Q_0(h,u)$, $\forall h \in \{1,\ldots,k\},\ u \in \mathcal{U}$
repeat
  Observe x and find

$$h = \arg\min_{\tau=1,\ldots,k}\ d_\phi(x, \mu(\tau))$$

  Choose $u' = \pi_L(h)$
  Observe $x' = f(x, u')$ and find

$$h' = \arg\min_{\tau=1,\ldots,k}\ d_\phi(x', \mu(\tau))$$

  Update $Q-$function:

$$Q_{i+1}(h,u') = Q_i(h,u') + \alpha_i[C(x,u') + \gamma \min_u Q_i(h',u) - Q_i(h,u')]$$

  if $i \mod N = 0$ **then**
    Update partition:

$$\mu_{i+1}(h) = \mu_i(h) - \beta_i G(\mu_i(h), \mu_i(h^*), \Sigma(t))\nabla_\mu d_\phi(x, \mu_i(h))$$

$$h^* = \arg\min_{\tau=1,\ldots,k}\ d_\phi(x, \mu_i(\tau))$$

  end if
until Convergence criterion is satisfied
Update Policy:

$$u_\epsilon(x) = \arg\min_u \{\ Q_\epsilon(x,u)\ \}$$

In order to illustrate one of the major advantages of adaptive state aggregation, which is memory efficiency, we initialize μ by uniformly discretizing over $\hat{S} = [-1,1] \times [-4,4] \times [-1,1] \times [-4,4]$ with 625 clusters, corresponding

to a standard discretization scheme with only 5 bins for each dimension. In Fig. 2 we compare the average (over 100 episodes) number of timesteps achieved until failure using the standard $Q-$learning algorithm, and the proposed algorithms (8) and Algorithm 2.

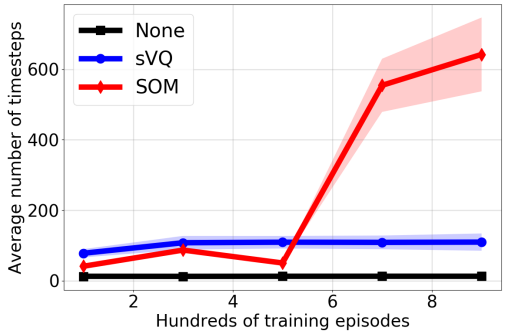

Fig. 2. Average number of timesteps (over 100 episodes) in the Cart-pole problem (Fig. 1) achieved by a Q-learning algorithm without state aggregation (black), a Q-learning algorithm with state aggregation based on sVQ (8) (blue), and a Q-learning algorithm with state aggregation based on SOM (Alg. 2) (red).

We observe that state aggregation with sVQ (8) can provide a boost in the performance of the Q-learning algorithm, but heavily depends on the initial configuration. Notably, SOM seems better suited for reinforcement learning applications, as Alg. 2 yields considerably better performance compared to the other algorithms. This is mainly due to the topological properties of SOM, which can accelerate convergence and avoid poor local optima. We expect annealing methods, such as the Online Deterministic Annealing [28], which is formulated as a stochastic approximation algorithm, to exhibit similar performance boost, with the added benefit of automatically deciding the number of clusters. We note that in our state aggregation scheme, we have assigned only 5 bins for each of the 4 dimensions of the MDP at hand, giving more weight on finding a good partition of the state-space rather than using a large number of naively chosen aggregate states. Finally, we note that the performance of the proposed reinforcement learning algorithm (Alg. 2) depends on several hyperparameters such as the the profile of the stepsizes α_i and β_i and the choice of the neighborhood function G used in the SOM algorithm.

V. Conclusion

We have shown that online vector quantization algorithms can be used as adaptive state aggregation schemes along with temporal-difference reinforcement learning algorithms. We studied the convergence of the resulting algorithm as a two-timescale stochastic approximation algorithm with: (a) a fast component that executes a temporal-difference reinforcement learning algorithm, and (b) a slow component that adaptively partitions the state space of a Markov decision process according to a dissimilarity measure based on Bregman divergences. Finally, we proposed a Q-learning algorithm with adaptive state aggregation based on self-organizing map (SOM) and solved the Cart-pole (inverted pendulum) optimal control problem with few aggregate states.

References

[1] M. Riedmiller, T. Gabel, R. Hafner, and S. Lange, "Reinforcement learning for robot soccer," *Autonomous Robots*, 2009.

[2] G. Tesauro, "Td-gammon, a self-teaching backgammon program, achieves master-level play," *Neural computation*, 1994.

[3] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[4] J. N. Tsitsiklis, "Asynchronous stochastic approximation and q-learning," *Machine learning*, vol. 16, no. 3, pp. 185–202, 1994.

[5] J. N. Tsitsiklis and B. Van Roy, "An analysis of temporal-difference learning with function approximation," *IEEE Transactions on Automatic Control*, vol. 42, no. 5, pp. 674–690, 1997.

[6] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons, 2007, vol. 703.

[7] J. N. Tsitsiklis and B. Van Roy, "Feature-based methods for large scale dynamic programming," *Machine Learning*, vol. 22, 1996.

[8] L. Baird, "Residual algorithms: Reinforcement learning with function approximation," in *Machine Learning Proceedings*. Elsevier, 1995.

[9] C. Dann, G. Neumann, J. Peters, *et al.*, "Policy evaluation with temporal differences: A survey and comparison," *Journal of Machine Learning Research*, vol. 15, pp. 809–883, 2014.

[10] A. George, W. B. Powell, and S. R. Kulkarni, "Value function approximation using multiple aggregation for multiattribute resource management," *Journal of Machine Learning Research*, 2008.

[11] F. Fernández and D. Borrajo, "Vqql. applying vector quantization to reinforcement learning," in *Robot Soccer World Cup*. Springer, 1999.

[12] S. P. Singh, T. Jaakkola, and M. I. Jordan, "Reinforcement learning with soft state aggregation," in *Advances in neural information processing systems*, 1995, pp. 361–368.

[13] I. S. Lee and H. Y. Lau, "Adaptive state space partitioning for reinforcement learning," *Engineering applications of artificial intelligence*, vol. 17, no. 6, pp. 577–588, 2004.

[14] D. P. Bertsekas and D. A. Castanon, "Adaptive aggregation methods for infinite horizon dynamic programming," *IEEE Transactions on Automatic Control*, vol. 34, no. 6, pp. 589–598, 1989.

[15] J. Baras and V. Borkar, "A learning algorithm for markov decision processes with adaptive state aggregation," in *Proceedings of the 39th IEEE Conference on Decision and Control (Cat. No. 00CH37187)*, vol. 4. IEEE, 2000, pp. 3351–3356.

[16] S. Sehad and C. Touzet, "Self-organizing map for reinforcement learning: obstacle-avoidance with khepera," in *Proceedings of PerAc'94. From Perception to Action*. IEEE, 1994, pp. 420–423.

[17] H. Montazeri, S. Moradi, and R. Safabakhsh, "Continuous state/action reinforcement learning: A growing self-organizing map approach," *Neurocomputing*, vol. 74, no. 7, pp. 1069–1082, 2011.

[18] T. Kohonen, *Learning Vector Quantization*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 175–189.

[19] C. N. Mavridis and J. S. Baras, "Convergence of stochastic vector quantization and learning vector quantization with bregman divergences," in *21st IFAC World Congress*, 2020.

[20] V. S. Borkar, "Stochastic approximation with two time scales," *Systems & Control Letters*, vol. 29, no. 5, pp. 291–294, 1997.

[21] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh, "Clustering with bregman divergences," *Journal of machine learning research*, 2005.

[22] V. S. Borkar, *Stochastic approximation: a dynamical systems viewpoint*. Springer, 2009, vol. 48.

[23] A. G. Barto, R. S. Sutton, and C. W. Anderson, "Neuronlike adaptive elements that can solve difficult learning control problems," *IEEE transactions on systems, man, and cybernetics*, pp. 834–846, 1983.

[24] M. Cottrell, J.-C. Fort, and G. Pagès, "Theoretical aspects of the som algorithm," *Neurocomputing*, vol. 21, no. 1-3, pp. 119–138, 1998.

[25] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine learning*, vol. 3, no. 1, pp. 9–44, 1988.

[26] V. S. Borkar and S. P. Meyn, "The ode method for convergence of stochastic approximation and reinforcement learning," *SIAM Journal on Control and Optimization*, vol. 38, no. 2, pp. 447–469, 2000.

[27] Y. Linde, A. Buzo, and R. Gray, "An algorithm for vector quantizer design," *IEEE Transactions on communications*, 1980.

[28] C. Mavridis and J. Baras, "Online deterministic annealing for classification and clustering," 2021.