Proceedings of the 39th IEEE
Conference on Decision and Control
Sydney, Australia • December, 2000

A LEARNING ALGORITHM FOR MARKOV DECISION PROCESSES WITH ADAPTIVE STATE AGGREGATION [1]

J.S. Baras
Institute for Systems Research
University of Maryland
College Park, MD 20742, USA
email: baras@isr.umd.edu.

V.S. Borkar
School of Technology and Computer Science
Tata Institute of Fundamental Research
Mumbai 400005, INDIA
email: borkar@tifr.res.in.

Abstract

We propose a simulation-based algorithm for learning good policies for a Markov decision process with unknown transition law, with aggregated states. The state aggregation itself can be adapted on a slower time scale by an auxiliary learning algorithm. Rigorous justifications are provided for both algorithms.
KEY WORDS: reinforcement learning, Markov decision processes, learning vector quantization, stochastic approximation, dynamic programming.

1 INTRODUCTION

In classical stochastic control, it is assumed that the state process or an associated observation process is exactly known to the controller. In many realistic situations, however, this is not quite true. This is because this information needs to be transmitted across a communication channel with rate constraints, calling for explicit data compression. At one level, one can view compressed observations as further 'partial observations'. But there is one important difference: The data compression scheme itself is a decision variable and should be optimized. Many recent works have addressed these issues [8], [10], [12], [13], [27].

Our aim here is to address this issue in the context of yet another important recent development, viz., reinforcement learning algorithms for learning near-optimal policies. (See [3], [22] and the references therein.) These are computationally intensive simulation-based algorithms. They are particularly appealing for large complex systems for which exact analytic modelling and accurate model estimation are unfeasible or expensive, but simulating typical system transitions is relatively easy (e.g., based on local update rules for interconnected systems). Even so, in their theoretically exact form, these algorithms suffer from the curse of dimensionality, in fact much more so than their classical antecedents like value and policy iteration. Hence there is a need to interface them with a suitable approximation architecture.

The two standard paradigms for approximation are state aggregation and the use of parametrized families of functions (with a low dimensional parameter space) for approximating the value function [3], [24], [25]. The latter, though it makes eminent sense as an approximation scheme, is not much use from a data compression perspective, whereas state aggregation fits it naturally. This motivates our analysis of a learning algorithm with state aggregation.

To be specific, we consider an 'actor-critic' type learning algorithm. These were introduced in [1] and studied extensively in [14]. We interface this algorithm with state aggregation as proposed in [24], except that this is an 'on line' learning algorithm based on a single simulation run, eschewing in particular the 'independent sampling' over each aggregated state as in [24] which simplifies the analysis of [24] somewhat. Our analysis rigorously justifies aggregating states in terms of the value function. (This was also suggested in [10].) We then propose 'learning' schemes to achieve this aggregation. Taken together, this provides a combined scheme for data compression and optimization.

The paper is organized as follows. The next section formulates our control problem as a Markov decision process on a large but finite state space. It then considers a fixed aggregation of states and formulates the basic algorithm. Section 3 states the main theoretical results in this context, notably the key error estimates. These estimates justify state aggregation on the basis of clustering in the range of the value function. This motivates the clustering algorithm proposed in section 4.

[1] This work was initiated when both authors were visiting the Lab. for Information and Decision Systems, M.I.T. Work supported by Grant No. III 5(12)/96-ET from the Department of Science and Technology, Government of India, and U.S. Army Grant PAAL03-92-G-0115, Center for Intelligent Control Systems, M.I.T.

The proofs are quite technical, but a large part of them is standard and is adapted from analogous arguments in [5], [6], [14]. Hence in the interest of brevity we refer the reader to these references for detail, focusing here only on the few points of departure from these references. A complete version of the paper with detailed proofs will appear elsewhere.

2 THE LEARNING PROBLEM

We consider a controlled Markov chain $X(n), n \geq 0$, on a large finite state space $S = \{1, 2, \ldots, s\}$, controlled by a control process $Z(n), n \geq 0$, taking values in a finite action space $A = \{a_1, \ldots, a_r\}$. ('A' could be a discretization of a compact metric space.) The transition probability is given by $p : S \times S \times A \to [0, 1]$. That is,

$$P(X(n+1) = i / X(m), Z(m), m \leq n) = p(X(n), i, Z(n)), n \geq 0.$$

$\{Z(n)\}$ is said to be a stationary policy if $Z(n) = v(X(n))$ $\forall$ n for some $v : S \to A$, and a stationary randomized policy if the conditional law of $Z(n)$ given $X(m), Z(m-1), m \leq n$, is $\varphi(X(n))$ $\forall$ n for a $\varphi : S \to \mathcal{P}(A)$. (Here and later, $\mathcal{P}(\ldots)$ stands for the space of probability measures on '$\ldots$' with topology of weak convergence.) By abuse of terminology, we may refer to the map $v(\cdot)$ (resp., $\varphi(\cdot)$) itself as a stationary (resp. stationary randomized) policy.

Our aim is to minimize over all admissible $\{Z(n)\}$ the cost

$$E\left[\sum_{m=0}^{\infty} \alpha^m g(X(m), Z(m))\right]$$

for a prescribed running cost $g : S \times R$ and discount factor $\alpha \in (0, 1)$. The value function $V^* : S \to R$ associated with this problem is defined as

$$V^*(i) = \min_{\{Z(n)\}} E\left[\sum_{m=0}^{\infty} \alpha^m g(X(m), Z(m)) / X(0) = i\right], \quad i \in S,$$

and is the unique solution to the dynamic programming equations

$$V^*(i) = \min_a \left[g(i, a) + \alpha \sum_j p(i, j, a) V^*(j)\right], \quad i \in S. \tag{2.1}$$

In fact, $\varphi : S \to \mathcal{P}(A)$ is an optimal stationary randomized policy if and only if for each i, support $(\varphi(i)) \subset$ Argmin (r.h.s. of (2.1). A stationary optimal policy always exists, obtainable by performing the minimization on the r.h.s. of (2.1). For these and other standard facts concerning Markov decision processes, see [21]. We assume that the chain is irreducible under every stationary randomized policy.

We wish to learn near-optimal policies for an aggregated state space based on a simulation run. The state aggregation is achieved by a partition of the state space into disjoint subsets:

$$S = S_1 \cup S_2 \cup \ldots \cup S_m, S_i \cap S_j = \phi \text{ for } i \neq j,$$

where typically $m << s$. We view the S_i's as aggregated 'meta-states'. Correspondingly, define $\overline{p}(i, l, a)$, $i \in S$, $1 \leq l \leq m$, $a \in A$, by

$$\overline{p}(i, l, a) = \sum_{j \in S_l} p(i, j, a).$$

Let $\mathcal{S} = \{S_1, S_2, \ldots, S_m\}$ and $\tilde{S}(n), n \geq 0$, the $\mathcal{S}$-valued process given by $\tilde{S}(n) = S_j$ if $X(n) \in S_j, n \geq 0$. Finally, let $\{a(n)\}, \{b(n)\}$ be prescribed sequences in $(0, 1)$ such that

$$\sum_{n=0}^{\infty} a(n) = \sum_{n=0}^{\infty} b(n) = \infty,$$
$$\sum_{n=0}^{\infty} a(n)^2, \quad \sum_{n=0}^{\infty} b(n)^2 < \infty,$$
$$a(n+1)/a(n) \to 1,$$
$$b(n)/a(n) \to 0.$$

The proposed learning algorithm is as follows: Let $\{X(n)\}$ be a simulated trajectory of the chain with control sequence $\{Z(n)\}$ soon to be specified. Let $\beta_0 > 0$ and $\Gamma : R \to [-\beta_0, \beta_0]$ the projection map. The algorithm is a pair of coupled iterations on two different time scales which are simulated by two different choices of stepsizes [4]. The 'fast' component is as follows: For $1 \leq i \leq m$ and $n \geq 0$,

$$W_i(n+1) = W_i(n) + a(n) I\{\tilde{S}(n) = i\}[g(X(n), Z(n)) + \alpha W_{\tilde{S}(n+1)}(n) - W_i(n)]. \tag{2.2}$$

Here $Z(n) = a$ with probability $\psi_{\beta(n)}(\tilde{S}(n), a), a \in A$, where

$$\psi_\beta(i, a) \triangleq e^{\beta(i,a)} / \left(\sum_b e^{\beta(i,b)}\right), \quad 1 \leq i \leq m, a \in A,$$

and $\beta(n) \triangleq [[\beta_{ia}(n)]]$ is given recursively by the 'slow' iteration.

$$\beta_{ia}(n+1) = \Gamma(\beta_{ia}(n) + b(n) I\{\tilde{S}(n) = S_i, Z(n) = a\} [W_i(n) - g(X(n), a) - \alpha W_{\tilde{S}(n+1)}(n)]) \tag{2.3}$$

To get an intuitive feel for this scheme, consider $s = m, S_i = \{i\}$ $\forall$ i (i.e., no aggregation), whence it reduces to the 'algorithm 3' of [14]. As per the rationale

for two time scale stochastic approximation developed in [4], the fast iteration (2.2) views the slow iteration, i.e. $\{\beta(n)\}$, as quasi-static. Thus (2.2) can be 'approximated' by the corresponding iteration with fixed $\beta(n)(\approx \beta$, say). But then it is the stochastic approximation version of fixed policy value iteration, the fixed policy in question being the stationary randomized policy $\varphi_\beta(\cdot)$ given by $\varphi_\beta(i) = \pi_\beta(i,\cdot), i \in S$. Thus (2.2) iteratively tracks the fixed policy value function $V_\beta(\cdot)$ for the policy $\varphi_\beta(\cdot)$, given by

$$V_\beta(i) = E\left[\sum_{m=0}^{\infty} \alpha^m k(X(m), \tau(m))/X(0) = i\right], i \in S,$$

where the expectation is w.r.t. φ_β. Since $\beta \sim \beta(n)$ varies slowly, it can then be expected to track $V_{\beta(n)}$. Plugging this into (2.3) in place of $W(n)$, (2.3) is seen to be a search scheme for minimizing

$$V_\beta(i,\cdot) + \alpha \sum_j p(i,j,\cdot) V_\beta(j), i \in S,$$

thereby simulating the minimization step of policy iteration. (See [14] for further discussion.) For $m \leq s$ as here, our algorithm is then nothing but the algorithm 3 of [14] suitably interfaced with state aggregation.

3 CONVERGENCE ANALYSIS

In this section, we state the main theoretical results concerning the algorithm (2.2)-(2.3). As is always the case with stochastic approximation algorithms, their convergence is contingent on their stability, i.e., on their iterates remaining bounded a.s. This is no problem for (2.3) where the projection map Γ ensures the boundedness for free. For (2.2), we need:

Lemma 3.1 The iterates of (2.2) remain bounded a.s.

The proof of this lemma is sketched in the appendix. Let $\Phi_\beta(\cdot) \in \mathcal{P}(S)$ denote the unique stationary distribution under the stationary randomized policy defined by

$$\varphi_\beta(i) = \pi_\beta(j,\cdot) \in \mathcal{P}(A) \text{ for } i \in S_j.$$

Define

$$\phi_{\beta i}(j) = \Phi_\beta(j) / \left(\sum_{l \in S_i} \Phi_\beta(l)\right), j \in S_i, 1 \leq i \leq m.$$

Lemma 3.2 If we consider (2.2) with the following change: $\beta(n) = \beta$, a constant in $[-\beta_0, \beta_0]$ $\forall$ n, then $W(n) \rightarrow \tilde{W}_\beta$ a.s., where $\tilde{W}_\beta$ is the unique solution to

$$\tilde{W}_\beta(i) = \sum_{j \in S_i} \phi_{\beta i}(j) \left(\sum_a \pi_\beta(i,a)\left[g(j,a) + \alpha \sum_l \overline{p}(j,l,a)\tilde{W}_\beta(l)\right]\right), \quad 1 \leq i \leq m.$$

The complete proof of this lemma will appear elsewhere. Here we confine ourselves to some important observations concerning (3.1). (3.1) is of the form

$$\tilde{W}_\beta = G_\beta(\tilde{W}_\beta) \tag{3.4}$$

for an appropriately defined $G_\beta : R^m \rightarrow R^m$ satisfying

$$||G_\beta(x) - G_\beta(y)||_\infty \leq \alpha ||x - y||_\infty, x, y \in R^m,$$

Here $||x||_\infty \triangleq \sup_i |x_i|$ is the 'max-norm' on R^m. That (3.2) (equivalently, (3.1)) has a unique solution then follows from the contraction mapping theorem.

Returning to the original set-up, we have:

Corollary 3.1 $||W(n) - \tilde{W}_{\beta(n)}|| \rightarrow 0$ a.s.

Proof This follows from the above lemma and the two time scale argument of [4]. (See, e.g., Lemma 2.3 of [4].)

The two time scale argument of [4] then justifies analysing (2.3) with $\tilde{W}_{\beta(n)}$ replacing $W(n)$ on the r.h.s. Let W^*_β denote the unique solution to

$$W^*_\beta(i) = \min_a \left[\sum_j \phi_{\beta i}(j)\left[k(j,a) + \alpha \sum_l \overline{p}(j,l,a) W^*_\beta(l)\right]\right], \quad 1 \leq i \leq m$$

where the existence and uniqueness is ensured as before by the contraction mapping theorem. Then we have

Lemma 3.3 $\beta(n) \rightarrow \beta^* \in [-\beta_0, \beta_0]$ a.s., where β^* satisfies:

$$W^*_{\beta^*}(i) \leq \tilde{W}_{\beta^*}(i) \leq W^*_{\beta^*}(i) + \epsilon(\beta_0), \tag{3.5}$$

where $\epsilon(\beta_0) \downarrow 0$ as $\beta_0 \uparrow \infty$.

This follows along the lines of section 5.4 of [14].

Let $\hat{V}(j) = W^*_{\beta^*}(i)$, when $j \in S_i, 1 \leq i \leq m, j \in S$. Let $e = [e_1, \ldots, e_m]$ with

$$e_i = \max_{j,l \in S_i} |V^*(j) - V^*(l)|, \tag{3.6}$$

Then we have:

Lemma 3.4 $||\hat{V} - V^*||_\infty \leq ||e||_\infty/(1-\alpha)$.

Proof This follows exactly as in Theorem 1 of [24].

Finally, define $V(j) = \tilde{W}_{\beta^*}(i)$ when $j \in S_i, 1 \leq i \leq m, j \in S$. The following is then immediate:

Corollary 3.2 $||V - V^*||_\infty \leq ||e||_\infty/(1-\alpha) + \epsilon(\beta_0)$.

The second term can be made arbitrarily small by making the parameter β_0 sufficiently large (though one expects a trade-off with the convergence rate of (2.2)-(2.3)). The first term involves $||e||_\infty$, which depends

only on the partition $\{S_i\}$. Given the definition (3.5) of e, it then makes sense to aggregate states based on the value of the value function. The next section proposes a scheme for achieving this.

4 CLUSTERING ALGORITHMS

Having concluded that one should aggregate by clustering (or 'vector quantization') in the range of the value function, one important problem remains: The value function is not known. Running a learning algorithm to estimate the value function for sake of state aggregation defeats the purpose, as it brings back into picture the very 'curse of dimensionality' that (2.2)-(2.3) were supposed to work around. Thus a suitable approximation to the value function is called for. Furthermore, we want to come up with a good state aggregation scheme concurrently with (2.2)-(2.3) based on a single simulation run. Hence we are not in a position to first estimate the value function and then aggregate on the basis of it using a batch-mode algorithm like Lloyd's [17]. Once again, two time scale stochastic approximation comes to the rescue: One can run a clustering algorithm on estimated values concurrently, albeit on a slower time scale so as to achieve the desired effect. Implicit in this is the requirement that the algorithm be sequential and based on 'on line' estimates of the value function as they arrive. Thus one has to resort to one of the 'Learning Vector Quantization' (LVQ) schemes. We briefly review them below.

Recall that in vector quantization, one partitions the space into finitely many regions, the 'Voronoi regions', each identified with a point called the 'centroid' therein, as follows. The i-th Voronoi region is the set of all points that are closer to the i-th centroid w.r.t. a suitable metric (usually Euclidean) than any other, any tie being resolved as per some fixed convention. Thus specifying the centroids specifies the Voronoi regions. The connection with clustering is that all points which fall in a single Voronoi region are deemed to form a single cluster. In order that the partition be optimal w.r.t. the mean square error, there is a further requirement. The 'centroids' must in fact be the centroids of the Voronoi regions associated with them w.r.t. the underlying sampling distribution. This, however, is a necessary condition and not sufficient, as there can be local minima of the error function that meet this requirement.

The simplest LVQ algorithm is the 'winner take all' competitive learning algorithm described as follows ([11], Ch.9): Let $Q_i(n)$ denote the position of the i-th centroid at time n. Given a new observation $x(n)$, one first locates the centroid that is closest to it (the 'winner'), say $q_j(n)$, and then moves it a little towards $x(n)$ according to the update

$$q_j(n+1) = (1 - c(n))q_j(n) + c(n)x(n).$$

Here, $c(n) \in (0,1), n \geq 0$, is the usual stochastic approximation-type stepsize scheme satisfying

$$\sum_n c(n) = \infty, \quad \sum_n c(n)^2 < \infty.$$

This algorithm can converge to a local minimum of the error function that may be far from the optimal. There are schemes which, while not guarranteeing convergence to the global minimum, do generally improve upon the above algorithm. One such scheme is the Kohonen LVQ algorithm ([11], Ch.9) wherein one updates not only $q_j(n)$, but also the centroids that are its neighbours as per some prespecified neighbourhood scheme. Various 'weighting' schemes have also been proposed to modulate the extent to which different centroids should be moved [18], [20]. A general formalism for such algorithms and its convergence analysis have been presented in [15]. We take this viewpoint here.

Let $q(n) = [q_1(n), \ldots, q_m(n)]$ denote the positions of the m centroids at the n-th iteration. The general algorithm is:

$$q_i(n+1) = q_i(n) + c(n)I_i(x(n), q(n))(x(n) - q(n)), \quad 1 \leq i \leq m.$$

Here $I_i(\cdot,\cdot), 1 \leq i \leq m$, is the 'winner activation function'. For the winner-take all LVQ, $I_i(x,y) = 1$ if $||x - y_i|| \leq ||x - y_j||, j \neq i, 0$ otherwise. It can be correspondingly modified for the other LVQ algorithms. The analysis of [15] shows that

$$q(n) \rightarrow q^* = [q_1^*, \ldots q_m^*]$$

with probability one, where each q_i^* is indeed the centroid of the associated Voronoi region w.r.t. the underlying sampling distribution.

Based on this, we propose below a scheme for adaptive state aggregation to go with (2.2) - (2.3). Complete theoretical analysis is not presented, as it would be quite lengthy, while much of it is standard fare in the analysis of stochastic approximation and related algorithms. This is not to say that there is nothing novel: Once again, it is these points of departure that we shall highlight.

Our scheme is based on the algorithm of Tsitsiklis and Van Roy [25] for learning approximate value function for a fixed policy, based on linear parametrizations. Thus we seek for an approximation of the form

$$V_0(i) = \sum_{k=1}^{d} Q(i,k)h_k, \quad i \in S, \tag{4.7}$$

where $Q = [[Q(i,k)]]$ is a prescribed $s \times d$ matrix and $h = [h_1, \ldots, h_d]^T$ is the variable parameter vector which we seek to optimize. One chooses $s >> d$ so that the parameter search is in a space of dimension much lower than $|S| = s$. As before, let $\varphi_\beta(\cdot) = \pi_\beta(\cdot,\cdot)$ be a prescribed stationary randomized policy, and $V_\beta(\cdot)$ the corresponding fixed policy value function. The algorithm of [25] for learning an approximation of $V_\beta(\cdot)$ of form (4.2) is (for prescribed $\lambda \in [0,1]$):

$$\begin{aligned} h(n+1) &= h(n) + c(n)(g(X(n),Z(n)) \\ &+ \alpha \sum_k Q(X(n+1),k)h_k(n) \\ &- \sum_k Q(X(n),k)h_k(n))q(n) \\ q(n+1) &= \alpha\lambda q(n) + Q(X(n+1),\cdot), \end{aligned}$$

where $\sum c(n) = \infty, \sum c(n)^2 < \infty$.

It is proved in [25] that $h(n) \to h^\beta$ with probability one, where h^β is the unique solution to the equations

$$\prod T^{(\lambda)} Q h^\beta = Q h^\beta,$$

where: $\prod$ is the projection operator from R^s onto its d-dimensional subspace spanned by vectors of the form (4.2), and the operator $T^{(\lambda)} : R^S \to R^S$ is defined for $0 \leq \lambda < 1$ by

$$(T^{(\lambda)}J)(i) = (1-\lambda)\sum_{m=0}^{\infty} \lambda^m E$$

$$\left[\sum_{n=0}^{m} \alpha^n g(X(n),Z(n)) + \alpha^{m+1} J(X(m+1))/X(0) = i\right],$$

and for $\lambda = 1$ by

$$(T^{(1)}J)(i) = V_\beta(i),$$

$i \in S$. The control sequence $\{Z(n)\}$ is realized as per φ_β.

The aggregation scheme we propose appends (4.3)-(4.4) to (2.2)-(2.3), with one difference: $Z(n)$ is now generated with conditional law $\varphi_{\beta(n)}(X(n))$, conditional on $X(m), Z(m-1), m \leq n$. In view of the results of [15] recalled above and the two time scale analysis of [4], it is not hard to show that

$$h(n) - h^{\beta(n)} \to 0 \quad \text{a.s.}$$

Viewing

$$\tilde{V}_{\beta(n)} \triangleq Q h^{\beta(n)}$$

as an approximation to $V_{\beta(n)}$, one then aggregates states based on it by algorithm (4.1), with

$$x(n) \triangleq \sum_k Q(X(n),k) h_k^{\beta(n)}, n \geq 0.$$

Of course, the aggregation needs to be done on a much slower time scale than the rest of the algorithm. This is ensured by making $c(n)/b(n) \to 0$. To make a correspondence with our earlier notation: Let $\mathcal{S}^n = \{S_1^n, \ldots, S_m^n\}$ be the partition operative at time n. Then S_i^n = the Voronoi region associated with the centroid $y_i(n)$ for $1 \leq i \leq m$. Invoking the two time scale analysis of [4] in conjunction with the results of the preceding section and of [15], we conclude that

$$q(n) \to q^* \quad \text{a.s.}$$

where q^* satisfies the 'centroid' condition w.r.t. Φ_{β^*} for β^* as in (3.4).

It may be recalled that both state aggregation and use of linearly parametrized families were originally advocated as approximation schemes to soften the curse of dimensionality. We combine both, which may seem a bit redundant. The reason for doing so is, however, that our concern is not just ameliorating the curse, but also explicit data compression for purposes of transmitting information from the plant to the controller across a communication channel with limited capacity. The state aggregation approach is ideally suited for this, as it replaces an S-valued state taking s distinct values by a meta-state that takes only $m << s$ distinct values, thereby reducing per step bit requirement from $\log_2 s$ to $\log_2 m$. This data compression aspect is not as apparent in the linear parameterization approach, which is more approximation-theoretic in its flavour. Our use thereof is for doing a secondary optimization, viz., that over possible aggregation schemes. This is done on a much slower time scale and can be rendered computationally inexpensive by choosing a small value of d. In our formulation, this is of secondary importance to the optimization over policies implicit in (2.2)-(2.3), hence some laxity in convergence speed and accuracy can be tolerated in the interests of lowering computational overheads.

In conclusion, we have presented a simulation based learning algorithm for Markov decision processes, based on the 'actor-critic' paradigm, that explicitly takes into account state aggregation motivated by data compression. A parallel treatment is possible for the alternative learning paradigm of 'Q-learning' [23], [26].

References

[1] BARTO A., SUTTON R., ANDERSON C., Neuron-like elements that can solve difficult learning control problems, IEEE Trans. on Systems, Man and Cybernetics 13 (1983), 835-846.

[2] BENVENISTE A., METIVIER M., PRIOURET P., Adaptive Algorithms and Stochastic Approximation, Springer Verlag, Berlin-Heidelberg, 1990.

[3] BERTSEKAS D., TSITSIKLIS J., Neurodynamic Programming, Athena Scientific, Belmont, MA, 1996.

[4] BORKAR V., Stochastic approximation with two time scales, Systems and Control Letters 29 (1996), 291-294.

[5] BORKAR V., Distributed computation of fixed points of ∞-nonexpansive maps, Proc. of the Indian Academy of Sciences (Mathematical Sciences) 106 (1996), 289-300.

[6] BORKAR V., Asynchronous stochastic approximation, SIAM J. Control and Optim. 36 (1998), 168-188.

[7] BORKAR V., MEYN S., The O.D.E. method for convergence of stochastic approximation and reinforcement learning algorithms, to appear in SIAM J. Control and Optim..

[8] BORKAR, V., MITTER S., LQG control with communication constraints, in 'Communication, Computation, Control and Signal Processing: A Tribute to Thomas Kailath', A. Paulraj, V. Roychowdhury, C. Schaper (eds.), Kluwer Academic Publ., Norwell, MA, 1997, 365-373.

[9] BORKAR V., SOUMYANATH K., A new analog parallel scheme for fixed point computation, part I: Theory, IEEE Trans. on Circuits and Systems I: Theory and Applications 44 (1997), 351-355.

[10] BORKAR V., TATIKONDA S., MITTER S., Markov control problems under communication constraints, preprint.

[11] BOSE N.K., LIANG P., Neural Network Fundamentals with Graphs, Algorithms and Applications, McGraw-Hill, Inc., New York, 1996.

[12] DELCHAMPS D., Stabilizing a linear system with quantized state feedback, IEEE Trans. on Automatic Control 35 (1990), 916-924.

[13] FENG X., LOPARO K., Active probing in control systems with quantized state measurements: a minimum entropy approach, IEEE Trans. Automatic Control 42 (1997), 216-238.

[14] KONDA V.R., BORKAR V., Actor-critic like learning algorithms for Markov decision processes, to appear in SIAM J. Control and Optim.

[15] KOSMATOPOULOS E., CHRISTODOULOU M., Convergence properties of a class of Learning Vector Quantization algorithms, IEEE Trans. on Image Processing 5 (1996), 361-368.

[16] KUSHNER H., YIN G., Stochastic Approximation Algorithms and Applications, Springer Verlag, New York, 1997.

[17] LLOYD S., Least squares quantization in PCM, IEEE Trans. on Info. Theory 28 (1982), 129-137.

[18] MARTINEZ T., BERKOVICH S., SCHULTEN K., "Neural-gas" network for vector quantization and its application to time-series prediction, IEEE Trans on Neural Networks 4(1993), 553-569.

[19] NEVEU, J., Discrete Parameter Martingales, North Holland, Amsterdam, 1975.

[20] PAL N., BEZDEK J., TSAO, E., Generalized clustering networks and Kohonen's self-organizing scheme, IEEE Trans. on Neural Netwroks 4 (1993), 549-557.

[21] PUTERMAN M., Markov Decision Processes, John Wiley, New York, 1994.

[22] SUTTON R., BARTO A., Reinforcement Learning, MIT Press, Cambridge, MA, 1998.

[23] TSITSIKLIS J., Asynchronous stochastic approximation and Q-learning, Machine Learning, 16 (1994), 185-202.

[24] TSITSIKLIS J., VAN ROY, B., Feature-based methods for large scale dynamic programming, Machine Learning 22 (1996), 185-202.

[25] TSITSIKLIS J., VAN ROY B., An analysis of temporal difference learning with function approximation, IEEE Trans on Automatic control 42 (1997), 674-690.

[26] WATKINS, C., DAYAN, P., Q-learning, Machine Learning 8(1992), 279-292.

[27] WONG W., BROCKETT R., Systems with finite communication bandwidth constraints II: Feedback control problems, submitted.