Learning Algorithms for Risk-Sensitive Control

Vivek S. Borkar

Abstract— **This is a survey of some reinforcement learning algorithms for risk-sensitive control on infinite horizon. Basics of the risk-sensitive control problem are recalled, notably the corresponding dynamic programming equation and the value and policy iteration methods for its solution. Basics of stochastic approximation algorithms are also sketched, in particular the 'o.d.e.' approach for its stability and convergence, and implications of asynchrony. The learning schemes give stochastic approximation versions of the traditional iterative schemes for solving dynamic programs. Two learning schemes, Q-learning and the actor-critic method, are described along with their convergence analysis. As these 'ideal' schemes suffer from 'curse of dimensionality', one needs to use function approximation as a means to beat down the dimension to manageable levels. A function approximation based scheme is described for the simpler problem of policy evaluation. Some future research directions are pointed out.**

I. Introduction

Risk-sensitive control deals with exponential of integral cost. Its infinite horizon version which we consider below seeks to minimize the asymptotic growth rate of such a cost. It has important connections with dynamic games and robust control and is popular in certain applications, particularly to finance where it offers the advantage of 'penalizing all moments', so to say, thus capturing the 'risk' in addition to mean return (hence the name). In particular, it lends itself to a multiplicative dynamic programming equation, a distinct advantage over other proposals for capturing risk such as variance-penalized cost or reward.

While there were some early efforts in this direction such as [16], the subject of risk-sensitive control got a significant boost by the work of Whittle, leading to his monograph [18]. It has seen a lot of development since, both in discrete and continuous time and / or space problems [12], [13], [14]. For representative applications to finance, see [1] and [6].

Like other cost criteria, one can propose and justify iterative algorithms for solving the dynamic programming equation. The issue we are interested in here is how to do so, even approximately, when the exact model is either unavailable or too unwieldy to afford analysis, but on the other hand simulated or real data is available easily, based on which one may hope to 'learn' the solution in an incremental fashion. This takes us into the domain of reinforcement learning. As a learning paradigm in artificial intelligence, reinforcement learning sits somewhere between the paradigms of supervised learning on one hand, where exact or at least 'good' measurement of some relevant quantity (such as gradient of the performance measure with respect to a tunable parameter) is available, and unsupervised learning on the other, where no such feedback is available and one relies on self-organization of data based on heuristic or statistical principles. In reinforcement learning, while something as immediate as a performance sensitivity measure is not available, some other relevant quantity is, which is related in some way to the performance measure, so that it can be used to differentially reinforce the advantageous decisions till eventually only the optimal ones remain.

This work was supported in part by a J. C. Bose Fellowship.

V. S. Borkar is with the School of Technology and Computer Science, Tata Institute of Fundamental Research, Homi Bhabha Road, Mumbai 400005, India. borkar@tifr.res.in

In case of approximate dynamic programming that we are interested in, the actual cost or rewards that are realized themselves serve as a reinforcement signal. At the risk of oversimplification, a typical reinforcement learning scheme for approximate dynamic programming may be described as follows. Take a standard recursive scheme of the form $x_{n+1} = F(x_n)$ (say) for solving the dynamic programming equation. The function F invariably involves a conditional expectation (because dynamic programming does). Replace this conditional expectation (which requires the knowledge of transition probabilities) by an actual evaluation at a real or simulated random variable with the desired conditional distribution. Then set x_{n+1} equal to a convex combination of x_n and this 'correction term', with a small weight $a(n)$ for the latter. Under the standard hypotheses $\sum_n a(n) = \infty, \sum_n a(n)^2 < \infty$, this becomes a stochastic approximation version of the original iterative scheme. By the well know averaging properties of stochastic approximation, the algorithm 'sees' the conditional expectation which is not explicit. This is our learning algorithm. See [5] for an extensive account of reinforcement learning algorithms for classical cost criteria such as the infinite horizon discounted cost or time averaged cost.

This article summarizes the author's work on learning algorithms for risk-sensitive control. The next section covers certain preliminaries. First it describes the risk-sensitive control problem, the associated multiplicative dynamic programming equation, and the value and policy iteration schemes for solving it, along the lines of [12]. Then it recalls the key results in the 'o.d.e.' (for 'ordinary differential equation') approach to the convergence analysis of stochastic approximation algorithms [10]. Sections 3 describes resp. the Q-learning [7] and the actor-critic [8] algorithm for learning, inspired by resp., the value iteration and the policy iteration methods for solving the dynamic program. These are 'raw' schemes in the sense that there is no further approximation involved. Since complex control problems lead to dynamic programming equations in very large dimensions ('curse of dimensionality'), one often looks for an approximate scheme. One such scheme [3] is also described in section 3, albeit for the simpler problem of policy evaluation. Section 4 concludes with some remarks.

II. Preliminaries

A. Risk-sensitive control

Consider a controlled Markov chain $\{X_n\}$ on a finite state space $\mathcal{S} := \{1, 2, \cdots, s\}$, controlled by a control process $\{Z_n\}$ taking values in a finite action space $\mathcal{A}$. (Much of what follows in this section allows for more general $\mathcal{S}$ and $\mathcal{A}$, but since we use finite state and action spaces in the next section anyway, we shall restrict to those.) The dynamics is given by

$$P(X_{n+1} = i | X_m, Z_m, m \leq n) = p(i | X_n, Z_n), \ n \geq 0,$$

for a controlled transition probability function $p(\cdot|\cdot,\cdot) : \mathcal{S}\times\mathcal{S}\times\mathcal{A} \to [0,1]$ such that $\sum_j p(j|i,a) = 1 \ \forall \ i, a$. Control process $\{Z_n\}$ of the form $Z_n = v(X_n) \ \forall n$ with $v : \mathcal{S} \to \mathcal{A}$, will be called Markov control and identified with the map $v(\cdot)$ by standard abuse of notation. We assume that under any Markov control, $\{X_n\}$, which is then a time-homogeneous Markov chain, will be irreducible and aperiodic.

Let $c : \mathcal{S} \times \mathcal{A} \to \mathcal{R}^+$ denote a prescribed 'running cost' function. Risk-sensitive control seeks to minimize over all admissible controls the 'risk-sensitive cost'

$$\lambda := \limsup_{n\uparrow\infty} \frac{1}{n} \log E\left[e^{\sum_{m=0}^{n-1} c(X_m, Z_m)}\right]. \qquad (1)$$

Our hypotheses ensure that under a Markov control, this in fact will be a limit [2].

The dynamic programming equation for this problem turns out to be the 'nonlinear eigenvalue problem'

$$\lambda^* V(i) = \min_a \left(e^{c(i,a)} \sum_j p(j|i,a) V(j)\right), \ i \in \mathcal{S}. \quad (2)$$

This is an equation in the 'value function' $V : \mathcal{S} \to \mathcal{R}$ and the scalar λ^*. It can be shown [12] that $V > 0$ is uniquely characterized up to a multiplicative scalar in $\mathcal{R}^+$ and λ^* is uniquely characterized as the optimal cost. Furthermore, if $v(i)$ for $i \in \mathcal{S}$ is the minimizer on the right hand side, then v is an optimal Markov control.

More generally one considers *randomized* Markov policies, given by $\pi(i,a) \geq 0, (i,a) \in \mathcal{S}\times\mathcal{A}$, with $\sum_a \pi(i,a) = 1 \ \forall i$. Here $\pi(i,a)$ is the probability of choosing control a when in state i. The corresponding transition probabilities are given by $\sum_a \pi(i,a) p(j|i,a)$. This contains Markov controls as a special case when $\pi(i,a)$ is 1 for $a = v(i)$ (say) and zero otherwise. The dynamic programming equation cast in terms of randomized Markov controls becomes

$$V(i) = \min_\pi \left(\sum_a \pi(i,a) \frac{e^{c(i,a)}}{\lambda^*} \sum_j p(j|i,a) V(j)\right), \ i \in \mathcal{S}. \qquad (3)$$

In particular, it is clear from (3) that the optimum will be attained at a Markov control.

It is worth noting the similarity with the dynamic programming equation for ergodic control: (2) is the multiplicative analog of the latter. Similar to the ergodic case, one can propose the relative value and policy iteration algorithms as follows.

1) *Value iteration:* Fix $i_0 \in \mathcal{S}$. Pick an arbitrary initial guess $V_0 : \mathcal{S} \to \mathcal{R}^+$ and for $n \geq 0$, do:

$$\tilde{V}_{n+1} = \min_a \left(e^{c(i,a)} \sum_j p(j|i,u) V_n(j)\right), \quad (4)$$

$$V_{n+1} = \frac{\tilde{V}_{n+1}(i)}{\tilde{V}_{n+1}(i_0)}. \qquad (5)$$

This iteration, a nonlinear analog of the 'power method' for computing principal eigenvectors, can be shown to converge to the value function V with $V(i_0) = \lambda^*$ (the latter condition renders it unique).

2) *Policy iteration:* Start with a guess v_0 for the optimal Markov control. For $n \geq 0$, do:

a) Solve the principal (Perron-Frobenius) eigenvalue problem

$$\lambda_n V_n = e^{c(i, v_n(i))} \sum_j p(j|i, v_n(i)) V_n(j), \ i \in \mathcal{S}, \qquad (6)$$

with $V_n(i_0) = 1$ (say).

b) Set

$$v_{n+1}(i) \in \ \text{Argmin} \left(e^{c(i,\cdot)} \sum_j p(j|i,\cdot) V_n(j)\right). \qquad (7)$$

Then (V_n, λ_n) will converge to (V, λ^*) with $V(i_0) = 1$, in finitely many steps.

B. Stochastic approximation

This, in its simplest form, refers to stochastic recursive algorithms in $\mathcal{R}^d, d \geq 0$, of the form

$$x_{n+1} = x_n + a(n)[h(x_n) + M_{n+1}], \ n \geq 0. \qquad (8)$$

Here,

- $h : \mathcal{R}^d \to \mathcal{R}^d$ is Lipschitz,
- $\{M_n\}$ is a martingale difference sequence, i.e., for $\mathcal{F}_n := \sigma(x_m, M_m, m \leq n), n \geq 0$,

$$E[M_{n+1}|\mathcal{F}_n] = 0 \ \forall n.$$

We also assume that for some $K > 0$,

$$E[\|M_{n+1}\|^2|\mathcal{F}_n] \leq K(1 + \|x_n\|^2) \ \forall n.$$

- $a(n) > 0$ are stepsizes satisfying $\sum_n a(n) = \infty, \ \sum_n a(n)^2 < \infty$.

Stochastic approximation has been a well known technique in statistical computation for finding zeroes of h given the 'noisy measurements' $h(x_n) + M_{n+1}, n \geq 0$. It is also used extensively in adaptive filtering and control. One of the more successful approaches for its theoretical analysis is the 'o.d.e.' approach that treats it as a noisy discretization of the o.d.e. (see, e.g., [10])

$$\dot{x}(t) = h(x(t)). \tag{9}$$

Suppose (9) has a unique globally asymptotically stable equilibrium x^*. Then under our hypotheses, if the iterates remain bounded with probability one, they will converge to x^* with probability one. One test for ensuring probability one boundedness of iterates, particularly convenient for reinforcement learning applications, is given in [11] and is as follows. Suppose

$$\frac{h(cx)}{c} \to h_\infty(x) \quad \text{as} \quad 0 < c \uparrow \infty$$

uniformly on compacts for some $h_\infty : \mathcal{R}^d \to \mathcal{R}^d$. Then h_∞ will clearly be Lipschitz. Suppose the o.d.e.

$$\dot{x}(t) = h_\infty(x(t)) \tag{10}$$

has the origin as its unique asymptotically stable equilibrium. Then $\sup_n \|x_n\| < \infty$ with probability one.

In (8), all components are being updated simultaneously. This is the syncronous version. If each component is updated according to a different clock, e.g., by a separate processor, and there are delays in transmitting the updates across processors, one has instead the following update rule. Let $\{Y_n\}, Y_n \in \{1, 2, \cdots, d\}$, denotes the set of indices updated at time n on the global clock. Then the ith component, $1 \leq i \leq d$, is updated according to

$$\begin{aligned} x_{n+1}(i) \quad &= \quad x_i(n) + a(n)I\{i \in Y_n\}[h_i(x_{n-\tau_{1i}(n)}, \\ &\quad \cdots, x_{n-\tau_{di}(n)}) + M_i(n+1)], \end{aligned} \tag{11}$$

where $I\{\cdots\}$ is the indicator function and $\tau_{ij}(n)$'s are the delays. This is still not the fully asynchronous case, because the global clock is assumed known. More generally, one replaces stepsize $a(n)$ by

$$a(\nu(i,n)) : \ \nu(i,n) := \sum_{m=0}^{n} I\{i \in Y_m\}. \tag{12}$$

$\{\nu(i,n)\}$ is the 'local' clock for i, counting the number of updates so far. In fact, the global clock may be a complete artifice as long as causal relationships are respected.

Under suitable hypotheses on delays, one can show that in either case the iterates track the o.d.e.

$$\dot{x}(t) = \Lambda(t)h(x(t)), \tag{13}$$

where $\Lambda(t)$ for each t is a diagonal matrix with nonnegative entries. These in some sense reflect the relative rates at which the components are updated. If (12) is used, then under following additional restrictions on $\{a(n)\}$:

$$\sup_n \frac{a(\lfloor xn \rfloor)}{a(n)} < \infty, \ \inf_{y \in [x,1]} \frac{\sum_{m=0}^{\lfloor yn \rfloor} a(m)}{\sum_{m=0}^{n} a(m)} \to 1 \ \ \forall \ x \in (0,1), \tag{14}$$

one has

$$\Lambda(t) \equiv \frac{1}{d} I_d, \tag{15}$$

where I_d is the $d \times d$ identity matrix. In that case (13) is simply a time-scaled version of (9) and one obtains the same conclusions as in the synchronous case. See [10], Chapter 7, for details.

III. Learning algorithms

A. Q-learning

By analogy with Q-learning for other cost criteria [5], we define the Q-factors

$$Q(i,a) := e^{c(i,a)} \sum_j p(j|i,a)V(j), \ i \in \mathcal{S}, a \in \mathcal{A}.$$

These also satisfy a 'dynamic programming equation'

$$Q(i,a) = \frac{e^{c(i,a)}}{\lambda^*} \sum_j p(j|i,a) \min_b Q(j,b), \ i \in \mathcal{S}, a \in \mathcal{A},$$

where we have used the fact $V(j) = \min_b Q(j,b)$. A natural 'Q-value iteration algorithm' then is

$$\begin{aligned} \tilde{Q}_{n+1}(i,a) \quad &= \quad e^{c(i,a)} \sum_j p(j|i,a) \min_b Q(j,b), \ \forall \ i, a, \\ Q_{n+1}(i,a) \quad &:= \quad \frac{\tilde{Q}(i,a)}{\tilde{Q}(i_0,a_0)}, \end{aligned}$$

where $i_0 \in \mathcal{S}$ and $a_0 \in \mathcal{A}$ are fixed. Following our recipe for mapping an iterative scheme into its stochastic approximation counterpart, we obtain

$$\begin{aligned} &Q_{n+1}(i,a) = \\ &\quad Q_n(i,a) + a(n)\Big(\frac{e^{c(i,a)}}{Q_n(i_0,a_0)} \min_b Q(\xi_{n+1}(i,a)) \\ &\quad - Q_n(i,a)\Big), \end{aligned} \tag{16}$$

where $\xi_{n+1}(i,a)$ is an independently generated (simulated) random variable with law $p(\cdot|i,a)$. This is our Q-learning algorithm. The o.d.e. corresponding to (16) is

$$\dot{q}_{ia}(t) = \frac{e^{c(i,a)}}{q_{i_0a_0}(t)} \sum_j p(j|i,a) \min_b q_{jb}(t) - q_{ia}(t), \ t \geq 0. \tag{17}$$

One can show that, if initiated in the interior of the positive quadrant, $q(\cdot)$ remains in the positive quadrant and converges to the Q-factor Q with $Q(i_0, a_0) = 1$. This proof depends on first proving convergence of the related o.d.e.

$$\dot{q}_{ia}(t) = \frac{e^{c(i,a)}}{\lambda^*} \sum_j p(j|i,a) \min_b q_{jb}(t) - q_{ia}(t), \ t \geq 0. \tag{18}$$

and then arguing that the trajectories of (17) can be obtained from those of (18) by some scaling transformations. To conclude from this the desired convergence of $\{Q_n\}$, we still need to argue their boundedness with probability one. The counterpart of (10) in this case turns out to be

$$\dot{q}(t) = -q(t), \tag{19}$$

which clearly has the origin as its globally asymptotically stable equilibrium. This completes the convergence analysis. Details can be found in [8].

So far we considered the synchronous version of the algorithm. If the algorithm is used on-line or is based on a single simulation run, Y_n in the terminology of the preceding section is the singleton $\{(X_n, Z_n)\}$. Then using (12) and (14), one can still have the iterates track a time-scaled version of (17) thanks to (15), and we still have the desired convergence claim.

An important observation for the on-line case is the following: While Z_n that minimizes $Q_n(X_n, \cdot)$ would be the best guess for optimal action at time n, one needs to also try all other control choices with a small probability in order to ensure enough 'exploration' for the learning schemes. (Trivially, one cannot learn what is never – or rarely – attempted.) But then the scheme can at best be near-optimal instead of optimal, and there is the usual 'exploration-exploitation' trade-off. This applies to the schemes described in the next two subsections as well.

B. Actor-critic algorithm

This is inspired by policy iteration. The idea is to solve (6) on a fast time scale and do the policy update of (7) iteratively on a slower time scale. Using standard 'two time scale' thinking, one can treat the policy as quasi-static while analyzing the former, whence it tracks the 'value' corresponding to the current guess for optimal policy. In turn, one can treat the faster iterates as quasi-equilibrated while analyzing the slower updates for policy, which then become a simple gradient scheme. At the level of (9), the analogy is with singularly perturbed differential equations. The two time scale behavior can be induced by using two different stepsize schedules for the two iterates. We make this precise below, after first deriving a sensitivity formula for parametrized policies.

It is more convenient here to work with randomized Markov policies. Let $\pi^\theta(\cdot, \cdot)$ denote a parametrized family of randomized Markov controls, parametrized by parameter θ belonging to some $\Theta \subset \mathcal{R}^m$, say, which is the closure of an open set. Consider the 'dynamic programming' equation for a fixed policy π^θ: For $i \in \mathcal{S}$,

$$V(\theta, i) = \sum_a \pi^\theta(i,a) \left(\frac{e^{c(i,a)}}{\lambda^*(\theta)} \sum_j p(j|i,a) V(\theta, j) \right). \tag{20}$$

Note that

$$\tilde{p}^\theta(j|i) := \frac{\sum_a \pi^\theta(i,a) e^{c(i,a)} p(j|i,a) V(\theta, j)}{\lambda^*(\theta) V(\theta, i)}, \ i, j \in \mathcal{S},$$

defines a transition probability for each θ, that assigns positive probabilities to precisely those transitions to which $p(\cdot|\cdot)$ assigned positive probabilities. In particular it leads to an irreducible chain with unique stationary distribution (say) η^θ. Let $\Lambda(\theta) := \ell n(\lambda^*(\theta))$ and

$$q^\theta(i,a) := \frac{e^{c(i,a)}}{V(\theta,i)\lambda^*(\theta)} \sum_j p(j|i,a) V(\theta, j), \ i, a \in \mathcal{S} \times \mathcal{A}.$$

On both sides of (20), differentiate w.r.t. θ, multiply by $\eta^\theta(i)/V(\theta, i)$ and sum over i to obtain (see [7] for details)

$$\begin{aligned} &\nabla^\theta \Lambda(\theta) \\ &= \frac{\nabla^\theta \lambda^*(\theta)}{\lambda^*(\theta)} \\ &= \sum_{i,a} \eta^\theta(i) \nabla^\theta \pi^\theta(i,a) q^\theta(i,a) \\ &= \sum_{i, a \neq a_0} \eta^\theta(i) \nabla^\theta \pi^\theta(i,a) (q^\theta(i,a) - q^\theta(i, a_0)). \end{aligned} \tag{21}$$

Let $\{a(n)\}, \{b(n)\}$ be stepsize schedules such that

$$\begin{aligned} \sum_n a(n) = \sum_n b(n) &= \infty, \\ \sum_n a(n)^2 + \sum_n b(n)^2 &< \infty, \\ \frac{b(n)}{a(n)} &\to 0. \end{aligned} \tag{22}$$

Also assume that $\{a(n)\}$ satisfy (14). The 'actor-critic' algorithm is:

$$\begin{aligned} &q_{n+1}(i,a) = \\ &\quad q_n(i,a) + a(\nu(i,a,n)) I\{X_n = i, Z_n = a\} \times \\ &\quad \Big(\frac{e^{c(X_n, Z_n)} q_n(X_{n+1}, Z_{n+1})}{q_n(i_0, a_0)} - q_n(i,a) \Big), \end{aligned} \tag{23}$$

for $(i,a) \in \mathcal{S} \times \mathcal{A}$, and,

$$\pi_{n+1}(i,a) = \Gamma\Big(\pi^n(i,a) - b(\nu(i,a,n))I\{X_n = i, Z_n = a\}(q_n(i,a) - Q(i_0,a_0))\Big) \quad (24)$$

for $i \in \mathcal{S}, a \in \mathcal{A} - \{a_0\}$, with

$$\pi_{n+1}(i,a_0) = 1 - \sum_{a \neq a_0} \pi_{n+1}(i,a). \quad (25)$$

Here $\Gamma(\cdot)$ is the projection onto the simplex $\{x = \{x_1, \cdots, x_{|\mathcal{A}|-1}\} : x_i \geq 0, \sum_i x_i \leq 1\}$. There is an abuse of notation here: the application of Γ in (24) is to be interpreted as being to the vector of its arguments as a varies over $a \neq a_0$, when the left hand side is written correspondingly as a vector. Updating all but one component of a probability vector and then setting the last one equal to one minus the sum of the rest as in (25) reduces the number of projection operations needed. Note that (24) is a projected stochastic gradient scheme based on the expression (21) for the gradient, when $\pi(\cdot,\cdot)$ themselves are treated as parameters.

The last condition in (22) ensures that (24)-(25) move on a slower time scale than (23). Using standard 'two time scale' argument ([10], Chapter 6), we can analyze (23) while treating $\pi_n \approx$ constant. Then it is the constant policy Q-learning scheme, which can be analyzed as in the preceding subsection to claim that with probability one, it tracks the Q-factors corresponding to π_n. In turn, this and a little algebra show that (24)-(25) form a *weighted* stochastic gradient descent for $\Lambda(\cdot)$. This in itself will not ensure convergence to the optimum, because a priori, local minima cannot be ruled out. But a more detailed argument along the lines of [17], pp. 111-112, establishes the convergence to the optimal θ with probability one. See [7] for a fuller description. ([7] has a sign error in (24).)

C. Function approximation

The above schemes use the full dimensionality of the state and action spaces and therefore will suffer from the 'curse of dimensionality' for large problems. This calls for a further layer of approximation to beat down the dimension. A popular device for this has been function approximation, wherein one approximates the value function or the Q-factor (as the case may be) by a parametrized family of functions, with the parameters lying in a moderate dimensional space. This family is fixed and one iterates in the parameter space to find the best parameters. A common choice that is relatively more amenable for analysis is that of linear function approximation.

We consider this for the simpler problem of policy evaluation, i.e., the problem of solving the eigenvalue problem

$$\lambda V(i) = e^{c(i)} \sum_j p(j|i,a)V(j), \ i \in \mathcal{S}. \quad (26)$$

Consider $V(i) \approx \sum_{i=1}^M r^k \phi^k(i)$, where for a moderate sized $M \geq 1$, $\phi^k, 1 \leq k \leq M$ are fixed basis functions or 'features', and $r^k, 1 \leq k \leq M$, are the weights. Let

- $\phi(i) := [\phi^1(i), \cdots, \phi^M(i)]^T$, $1 \leq i \leq M$,
- Φ is the matrix whose (i,k)th entry is $\phi^k(i)$.

We make the following important assumption:

($\dagger$) $\{\phi^k\}$ are orthogonal vectors in the positive cone and the submatrix of the transition matrix corresponding to $\cup_k\{i : \phi^k(i) > 0\}$ is irreducible.

This ensures in particular that Φ has full rank. Consider $V \approx \Phi r$ where $r := [r^1, \cdots, r^M]^T$. Plugging this into (26), we have

$$\Phi r \approx \lambda^{-1} diag\left(e^{c(\cdot)}\right) P\Phi r,$$

where P is the transition matrix and $diag(x)$ stands for the diagonal matrix with vector x along the diagonal. In turn,

$$r \approx \lambda^{-1}(\Phi^T\Phi)^{-1}\Phi^T diag\left(e^{c(\cdot)}\right) P\Phi r,$$

where P denotes the transition matrix. This suggests the algorithms (similar to the '0-LSPE' scheme of [4])

$$r_{n+1} = r_n + a(n)\left(\frac{B_n^{-1}A_n}{(\phi^T(i_0)r_n) \vee \epsilon} - I\right) r_n, \quad (27)$$

for $n \geq 0$, where

$$\begin{aligned} A_n &:= \sum_{m=0}^n e^{c(X_m)}\phi(X_m)\phi^T(X_{m+1}), \\ B_n &:= \sum_{m=0}^n \phi(X_m)\phi^T(X_m). \end{aligned}$$

Both A_n and B_n^{-1} can be recursively computed, the latter using the Sherman-Morrison-Woodbury formula [15]. Here the '$\cdot \vee \epsilon$' for $\epsilon > 0$ is artificially inserted in the denominator in order to avoid division by zero: in the 'raw' algorithms considered above, one can *prove* that a division by zero does not occur, such is not the case once function approximation is introduced.

Let μ denote the unique stationary distribution under P, $D := diag(\mu), C := diag(e^{c(\cdot)})$. (27) tracks the o.d.e.

$$\dot{r}(t) = \left(\frac{B^{-1}A}{(\phi^T(i_0)r(t)) \vee \epsilon} - I\right) r(t), \quad (28)$$

where $A := \Phi^T DCP\Phi, B := \Phi^T D\Phi$.

Let $W := \sqrt{D}\Phi(\Phi^T D\Phi)^{-1}\Phi^T DCP\sqrt{D}^{-1}$. Under ($\dagger$), this can be shown to leave the positive cone invariant and thus has a positive eigenvalue-eigenvector pair (γ^*, Y^*) with $Y^*_{i_0} = \sqrt{\mu(i_0)}\gamma^*$. If (28) is initiated so that $\sqrt{D}\Phi r_0$ is in the interior of the positive cone and $\phi^T(i_0)r(0) > 0$, then $\sqrt{D}\Phi r(t)$ and therefore $\sqrt{D}\Phi r_n$ (with probability one in the latter case) converge to Y^*. This follows along the lines of the arguments used for analyzing the Q-learning scheme above. (γ^*, Y^*) then serve as approximations for (λ^*, V) with $V(i_0) = \lambda^*$. To see this, note that for $M = d$ and $D =$ the identity matrix, W equals CP and the match would be exact. (The tweak of introducing '$\cdot \vee \epsilon$' does not affect the conclusion because it is inoperative once the iterates are sufficiently close to the desired limit.) In fact W is of the form ΠG, where $\Pi := \sqrt{D}\Phi(\Phi^T D\Phi)^{-1}\Phi^T\sqrt{D}$ is a projection w.r.t. a weighted norm (weighted by μ), and $G := \sqrt{D}CP\sqrt{D}^{-1}$ is a skewed version of CP, skewed due to the fact that we are sampling states according to the distribution μ and not uniformly. The latter problem can be cured by using stepsizes suggested by (14). Π will then be the usual projection and one is then solving the projected eigenvalue problem $\Pi CPY^* = \gamma^* Y^*$ instead of the original $CPV = \lambda^* V$. The error will depend critically on the choice of Φ for a given choice of M.

IV. Concluding remarks

One thing clear from the foregoing is that one has barely made a dent on the problem of learning algorithms for risk-sensitive control. The issues here are much harder than their counterparts for classical cost structures that lead to 'additive' dynamic programming. While the 'ideal' Q-learning and actor-critic schemes are in place, they are mainly of theoretical interest because of the curse of dimensionality, though the techniques used for their analysis can be useful otherwise. As for function approximation, one has limited success even with the simple case of policy evaluation. A lot more work is needed. This can involve better feature selection, stepsize selection, sampling schemes and so on. Regarding sampling schemes, see [9] for some proposals for sampling schemes from the point of view of accelerating convergence. Finding good error bounds for function approximation is another challenge.

References

[1] A. Bagchi and K. Suresh Kumar, Dynamic asset management: risk sensitive criteria with positive factors constraints, in '*Recent Developments in Mathematical Finance*' (J. Yong, ed.), World Scientific, Hong Kong, pp. 1-11.

[2] S. Balaji and S. P. Meyn, Multiplicative ergodicity and large deviations for an irreducible Markov chain, *Stoc. Proc. and Their Appl.* Vol. 90, 2000, pp. 123-144.

[3] A. Basu, T. Bhattacharya and V. S. Borkar, A learning algorithm for risk-sensitive cost, *Math. Op. Research* Vol. 33, 2008, pp. 880-898.

[4] D. P. Bertsekas and A. Nedic, Least squares policy evaluation algorithms with linear function approximation, *Discrete Event Dynamical Systems* Vol. 13, 2003, pp. 79-110.

[5] D. P. Bertsekas and J. N. Tsitsiklis, *Neurodynamic Programming*, Athena Scientific, Belmont, MA, 1996.

[6] T. R. Bilecki and S. R. Pliska, Risk-sensitive dynamic asset management, *Appl. Math. Optim.* Vol. 39, 1999, pp. 337-360.

[7] V. S. Borkar, A sensitivity formula for the risk-sensitive cost and the actor-critic algorithm, *Systems and Control Letters* Vol. 44, 2001, pp. 339-346.

[8] V. S. Borkar, Q-learning for risk-sensitive control, *Math. Op. Research* Vol. 27, 2002, pp. 294-311.

[9] V. S. Borkar, Reinforcement learning – a bridge between numerical methods and Markov Chain Monte Carlo', in '*Perspectives in Mathematical Sciences*' (N. S. N. Sastry, B. Rajeev, Mohan Delampady, T. S. S. R. K. Rao, eds.), World Scientific, 2009.

[10] V. S. Borkar, *Stochastic Appoximation: A Dynamical Systems View*, Hindustan Publ. Agency, New Delhi, and Cambridge Uni. Press, Cambridge, UK, 2009.

[11] V. S. Borkar and S. P. Meyn, The o.d.e. method for convergence of stochastic approximation and reinforcement learning, *SIAM J. Control and Optim.* Vol. 38, 2000, pp. 447-469.

[12] V. S. Borkar and S. P. Meyn, Risk-sensitive optimal control for Markov decision processes with monotone cost, *Math. Op. Research* Vol. 27, 2002, pp. 192-209.

[13] G. B. Di Masi and L. Stettner, Risk-sensitive control of discrete-time Markov processes with infinite horizon, *SIAM J. Control and Optimization* Vol. 38, 1999, pp. 61-78.

[14] W. H. Fleming and W. M. McEneaney, Risk-sensitive control on an infinite time horizon, *SIAM J. Control and Optimization* Vol. 33, 1995, pp. 1881-1915.

[15] G. H. Golub and C. F. Van Loan; *Matrix Computations*, Johns Hopkins Uni. Press, Baltimore, MD, 1996.

[16] D. H. Jacobson, Optimal stochastic linear systems with exponential performance criteria and their relation to deterministic differential games, *IEEE Trans. Automatic Contol* Vol. AC-18, 1973, pp. 124-131.

[17] V. R. Konda and V. S. Borkar; Actor-critic-type learning algorithms for Markov decision processes, *SIAAM J. Control and Optimization* Vol. 38, 1999, pp. 94-123.

[18] P. Whittle, *Risk-Sensitive Optimal Control*, John Wiley and Sons, Chichester, UK, 1990.