J. Indian Inst. Sci., March–April 1998, **78**, 119–130.

Robust parameter optimization of hidden Markov models

B. BHARATH* AND V. S. BORKAR**
*Department of Electrical Communication Engineering **Department of Computer Science and Automation
Indian Institute of Science, Bangalore 560 012, India.

Received on February 19, 1998.

Abstract

Several robust algorithms for parametric optimization of hidden Markov models are presented. These combine aspects of Fabian's 'sign' algorithm, two-time scale stochastic approximation and certain techniques for estimating the gradient (or related quantities) of the performance measure based on a simulation run.

Keywords: Stochastic optimization, stochastic approximation, robust algorithms, hidden Markov models, parametric optimization.

1. Introduction

The archetypical stochastic optimization algorithm is the stochastic gradient scheme, an instance of the general class of stochastic approximation algorithms.[1] This is given by

$$X(n+1) = X(n) - a(n)[\nabla f(X(n)) + \xi(n)] \tag{1}$$

where $f(\cdot)$ is the function to be minimized, $\{a(n)\}$ is a stepsize sequence satisfying

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

and $\{\xi(n)\}$ is a random process that models measurement noise, the implication being that the expression in the square brackets in (1) is a noisy measurement of the gradient. Often, this information is not available and one has to settle for a noisy measurement of an approximate gradient, e.g. a finite-difference approximation as in the so-called Kiefer–Wolfowitz scheme. A comprehensive account and analysis of these schemes appears in Polyak and Tsypkin.[2]

In many recent applications, one encounters situations where even an approximate gradient is not easily available, but has to be estimated online or based on a simulation run. This typically happens because one aims to optimize the expected value of a performance variable with respect to a parameter and the interdependence thereof is not analytically explicit. Thus, while one is still seeking the gradient of an expectation as implicit in the aforementioned schemes, one is no longer in a position to interchange it with the expectation of a gradient, as in fact has been done in (1) and its variants. This has led to a considerable body of work on estimates for the former that are either online or based on a simulation run. These techniques go under the rubric of 'infinitesimal perturbation analysis' (IPA).[3] But, these are not without their problems. To press this point further, consider, for example, a recent scheme due to

Chong and Ramadge.[4] In this, one assumes the conditional law of the performance variable given the parameter as known. The algorithm requires the derivatives of the inverse map of the conditional distribution function, with respect to the parameter. In practice, this itself will have to be estimated and will be a far from robust statistic. Furthermore, the algorithm aggregates and averages data over regeneration epochs which can be very sparse, leading to very slow convergence. In addition, there are other intrinsic limitations, such as the need for the performance variable to be real and continuous-valued, as well as insensitive to the so-called 'event order changes' and so on.

With this in mind, Bhatnagar and Borkar[5] proposed a two-time scale stochastic approximation algorithm for parametric optimization of hidden Markov models (HMMs). The idea behind the two-time scale stochastic approximation is simple.[6] It involves coupled iterations.

$$X(n+1) = X(n) + a(n)(q(X(n), Y(n)) + M(n+1)), \tag{3}$$

$$Y(n+1) = Y(n) + b(n)(g(X(n), Y(n)) + M'(n+1)), \tag{4}$$

where $\{a(n)\}$, $\{b(n)\}$ are stepsize schedules that satisfy (2) and $a(n) = o(b(n))$, $\{M(n)\}$ and $\{M'(n)\}$ are sequences of uncorrelated zero mean random variables representing measurement noise. It can be shown[6] that (3), (4) asymptotically behave like the singular o.d.e.

$$\dot{x}(t) = q(x(t), y(t)),$$
$$\varepsilon\dot{y}(t) = g(x(t), y(t)),$$

in the $\varepsilon \downarrow 0$ limit. Intuitively, the 'slow' component $x(\cdot)$ is seen as 'quasi-static' by the fast component $y(\cdot)$, while the latter is seen as 'almost equilibrated' by $x(\cdot)$. That is, from the point of view of $y(\cdot)$, $x(\cdot)$ changes very slowly so that the behaviour of $y(\cdot)$ is close to that of the o.d.e.

$$\dot{y}_x(t) = g(x, y_x(t)) \tag{5}$$

where $x \sim x(t)$ is a 'frozen' parameter. Suppose (5) has a globally asymptotically stable equilibrium $\lambda(x)$. Then $y(t) \sim \lambda(x(t))$ and as a result, $x(t)$ tracks the solution of the o.d.e.

$$\dot{x}'(t) = q(x'(t), \lambda(x'(t))). \tag{6}$$

Suppose (6) has a globally asymptotically stable equilibrium x^*. Then $(x(t), y(t))$ approximately converges to $(x^*, \lambda(x^*))$. In turn, one can show that $(X(n), Y(n)) \to (x^*, \lambda(x^*))$ with probability one[6].

The scheme comes in handy for implementing algorithms with nested 'DO' loops, where the inner loop is a subroutine that involves another algorithm whose near-convergence is required between each successive iterate of the outer loop. One can achieve the same effect by doing simultaneous updates on the two loops, albeit on different time scales simulated by different stepsize schedules (the inner one being faster). The way in which this has been used by Bhatnagar and Borkar[5] is by doing the averaging of the performance variable on the fast time scale, while performing an approximate gradient search on the said average on a slower time scale. Thus, one achieves 'gradient of expectation' directly, without having to contend with the aforementioned interchange. Also, the scheme is set up for HMMs, a much richer class of processes than those considered in earlier literature. It also allows for discrete-valued

performance variables like queue lengths and does not require conditions like insensitivity to event order changes.

It should be added that two-time scale techniques had been introduced earlier, but purely as an averaging technique for damping the oscillations of algorithm (1) above, in the spirit of 'momentum' methods.[7,8] These works do not address or resolve the issues raised above.

The workability of the proposed scheme was established by analysis and simulations in Bhatnagar and Borkar.[5] Nevertheless, the scheme shares with standard stochastic approximation its usual problem of high variance as reflected in the highly oscillatory behaviour of its trajectory, particularly in the initial stages. A large body of work on variance reduction techniques has been around for a while. We have already mentioned averaging, which can be implicit averaging of r.h.s. of (1) through a 'momentum' term[7,8] or explicit averaging of the iterates themselves.[9,10] Other techniques include importance sampling, use of common randomness or control variates, conditioning, etc.[11,12] The aim of this paper is to propose a computationally inexpensive alternative based on the Fabian algorithm[13]

$$X(n+1) = X(n) - a(n)\text{sgn}(\nabla f(X(n)) + \xi(n)) \tag{7}$$

where $\text{sgn}(\cdot)$ is the sign function given by

$$\text{sgn}(X) = \begin{cases} 1 \text{ for } X \geq 0, \\ -1 \text{ for } X < 0. \end{cases}$$

(For the vector case, $\text{sgn}(\cdot)$ is interpreted componentwise). The usual motivations for replacing (1) by (7) are high uncertainty in the gradient measurement and low computational complexity. We advocate its use as an explicit variance reduction mechanism. Thus, the algorithm we propose is

$$X(n+1) = X(n) - a(n)\text{sgn}(S(n)) \tag{8}$$

where $S(n)$ is the 'approximate gradient of expectation' obtained through the two-time scale technique. The intuition behind the expected robustness of this scheme is somewhat akin to that behind the use of median as a more robust estimator than mean—it is insensitive to freak episodes of large $|S(n)|$ due to noise spikes or numerical instability of the gradient estimation scheme. This intuition is indeed confirmed by the numerical experiments reported later in the paper.

There is also another important motivation for looking at (8). When an algorithm is implemented in a distributed manner, different processors may compute different components of the iteration and communicate these to each other over a communication channel with concomitant overheads. This makes (8) quite appealing, for it needs only one bit of information per component.

We consider two other candidates for $\{S(n)\}$ in addition to the Bhatnagar–Borkar scheme. These are described in the next section, following a brief theoretical analysis of (8). Section 4 applies these schemes to two queuing problems and concludes with some general comments on the empirical evidence and some related issues.

2. The algorithms

As in Bhatnagar and Borkar[5], the process we seek to optimize will be an 'HMM with a tunable parameter' given by the coupled iteration

$$X(n+1) = h(X(n), Y(n), \xi(n), \theta(n)),$$
$$Y(n+1) = g(X(n), Y(n), \xi'(n), \theta(n)),$$

$n \geq 0$. Here, $\{\xi(n)\}$, $\{\xi'(n)\}$ are i.i.d. sequences in $\mathbb{R}^{r(1)}$, $\mathbb{R}^{r(2)}$ resp., independent of each other, $h : \mathbb{R}^s \times \mathbb{R}^t \times \mathbb{R}^{r(1)} \times \mathbb{R}^m \to \mathbb{R}^s$ and $g : \mathbb{R}^s \times \mathbb{R}^t \times \mathbb{R}^{r(2)} \times \mathbb{R}^m \to \mathbb{R}^t$ are measurable maps, and $\{\theta(n)\}$ is the $\mathbb{R}^m$-valued parameter that is being tuned on line. The state process $\{X(n)\}$ is unobserved and $(X(n), Y(n))$ is assumed to be an ergodic Markov process for each fixed θ (i.e. when $\theta(n) = \theta$ for all n). $\{Y(n)\}$ is the observation process. Let $E_\theta[\cdot]$ denote the expectation with respect to the stationary distribution when θ is the operative parameter. Our aim is to minimize

$$J(\theta) = E_\theta[k(Y(n))]$$

for a given continuous cost function $k(\cdot)$. We assume that this expectation is well defined and that $J(\cdot)$ is continuously differentiable.

The algorithms that we consider have the form (8), rewritten as

$$\theta(n+1) = \theta(n) - a(n)\text{sgn}(\nabla J((n)) + e(n)) \tag{9}$$

where $e(n)$ is the vector error given by $S(n) - \nabla J(\theta(n))$, $S(n)$ being a suitable approximation of the gradient. Thus, $e(n)$ combines both the approximation error and the noise. We make the following additional assumptions:

(A1) $P(e_i(n) \leq x/X(m), m \leq n) = P(e_i(n) \leq x/X(n))$ and is independent of n, i.

Thus, we may write

$$F(x/y) \underline{\Delta} P(e_i(n) \leq x / X(n) = y),$$

the conditional distribution function of $e(n)$ given $X(n)$.

(A2) $(x, y) \to F(x/y)$, $x \to \nabla J(x)$ are Lipschitz maps.

(A3) For all y, $F(0/y) = 1/2$.

(A1) and (A2) are purely technical conditions that can be relaxed with some extra effort. We shall comment on these later. (A3) is the crucial condition which says that the conditional median of $e_i(n)$ given $X(n)$ is always zero. We also assume:

(A4) $\sup_n \|\theta(n)\| < \infty$ with probability one.

It should be noted that this is a highly nontrivial assumption. One is effectively 'assuming away' the stability of the scheme. There is no general scheme for verifying stability of stochastic approximation algorithms, though there are many techniques applicable to special classes thereof. Also, one can, as we do in our numerical experiments detailed later, avoid the problem altogether by projecting the iterates back to a bounded set whenever they exit from the same.

Theorem 1: Under (A1)–(A4), $\{\theta(n)\}$ converges to the set of local minima of $J(\cdot)$ with probability one.

The proof goes along standard lines, so we shall sketch only a brief outline here.

Define $h : \mathbb{R}^m \to \mathbb{R}^m$ by $h(x) = [h_1(x),..., h^m(x)]^T$ with

$$h_i(x) = 1 - 2F\left[-\frac{\partial J}{\partial x_i}(x)/x\right], 1 \le i \le m.$$

By adding and subtracting $h(\theta(n))$ from the r.h.s. of (9), it can be rewritten as

$$\theta(n+1) = \theta(n) + a(n)(h(\theta(n)) + M(n+1))$$

for an appropriately defined $\{M(n)\}$ satisfying

$$E[M(n+1)/X(i), \theta(i), i \le m] = 0.$$

Under (A4), standard 'o.d.e. analysis'[1,6] shows that with probability one, the algorithm asymptotically behaves like the differential equation

$$\dot{x}(t) = h(x(t)), \quad t \ge 0. \tag{10}$$

(A2) ensures that (10) is well-posed. From (A3), it follows that $h_i(x)$ has the sign opposite to that of $\frac{\partial J(x)}{\partial x_i}$ when the latter is nonzero. Thus,

$$dJ(x(t))/dt = \langle h(x(t)), \nabla J(x(t)) \rangle < 0$$

as long as $\|\nabla J(x(t))\| \neq 0$. The rest is routine.

This theorem forms the backdrop for all the algorithms we propose below. It should be remarked that we do not verify the above assumptions for these algorithms. Nevertheless, their empirical behaviour conforms to the foregoing intuition. (A3) in particular is not easy to verify in practice, but that seems to matter very little.

Before giving the details of the algorithms, we shall comment briefly on assumptions (A1)–(A4) above. In (A1), the dependence of the conditional distribution on the index i can be reintroduced at no extra cost except notational, one having to assume (A1)–(A3) separately for each i. Dependence on n can be reintroduced at the cost of now having to consider a family of nonautonomous o.d.e.s in place of (10). This is eminently possible[14], though tedious. Allowing the conditional distribution depends on the entire past $X(m)$, $m \le n$, explicitly, rather than just on $X(n)$ as in the first part of (A1), is a much more serious matter. Under certain (nontrivial) technical hypotheses (the 'fading memory condition') one is led to analyse in place of (10) a similar ordinary differential equation

$$\dot{x}(t) = \alpha(x(t))$$

for an appropriately defined $\alpha(\cdot)$. With this, the above analysis can be pushed through with a lot more technical complications, that too for a restrictive class of dependences.

Coming to (A2), dropping (A2) leads to possible ill-posedness of (10). This is not as bad as it seems, for one can go over to an appropriate differential inclusion and consider all possible trajectories of it for a given initial condition.

(A3), or more generally the condition that the conditional median is zero, is crucial.

(A4) can be verified by assorted techniques for checking the stability of stochastic approximation algorithms, such as those based on stochastic Lyapunov functions.[15] If $J(\theta) \to \infty$ as $\|\theta\| \to \infty$, $J(\cdot)$ itself will serve as one. For our numerical experiments reported below, (A4) is ensured for free by projecting the iterates back onto a bounded set. This is a standard trick in practice, but one has to keep in mind that one is now dealing with a 'projected' version of (10) which may develop spurious attractors on the boundary of the bounded set in question.

With this preamble, we list below the exact algorithms we have experimented with. For simplicity, only the scalar parameter case ($m = 1$) is stated, the vector case being completely analogous.

Scheme 1. The finite-difference (FD) scheme

This scheme, from Bhatnagar and Borkar[5], goes as follows: Let $\delta > 0$ be a small number and let $(X(n), Y(n))$, $(X'(n), Y'(n))$ denote simulations of the HMM corresponding to the parameter sequences $\{\theta(n)\}$, $\{\theta(n) + \delta\}$, respectively, where $\{\theta(n)\}$ is obtained as:

$$x(n+1) = x(n) + b(n)(k(Y(n)) - x(n)),$$
$$x'(n+1) = x'(n) + b(n)(k(Y'(n)) - x'(n)),$$
$$\theta(n+1) = \Gamma(\theta(n) + a(n)[(x(n) - x'(n))/\delta]),$$

with $a(n) = o(b(n))$. $\Gamma(\cdot)$ is a projection onto a prescribed bounded interval. Under appropriate stability conditions on the HMM, $x(n)$ (resp, $x'(n)$) can be shown[5] to track $E_{\theta(n)}[k(Y(n))]$ (resp., $E_{\theta(n)+\delta}[k(Y'(n))]$), this being the 'fast' scale. The iteration for $\{\theta(n)\}$ then performs an approximate gradient search based on a finite-difference approximation.

Note that this requires two parallel simulations, respectively for $\{\theta(n)\}$ and for $\{\theta(n) + \delta\}$. The computational overhead can be reduced by using common random numbers for both the simulations, which also improves the variance. For $m \geq 1$, the number of simulations increases accordingly. An alternative is to use the scheme proposed by Spall.[16]

The 'robust' version of this scheme is obtained by replacing the third recursion by

$$\theta(n+1) = \Gamma(\theta(n) + a(n)\mathrm{sgn}[(x(n) - x'(n))/\delta]).$$

Scheme 2: Smoothed functional (SF) scheme

Adapted from Katkovnik and Kulchitsky[17], the idea behind this scheme is as follows: Suppose we approximate ∇f in (1) by a 'smooth' approximation to ∇f given by Df_σ which stands for '∇f convolved with a Gaussian with zero mean and a small variance σ^2'. Denoting the Gaussian by $G_\sigma(\cdot)$, one has

$$Df_\sigma(x) = \int G_\sigma(x-y)\nabla f(y)dy \qquad \text{(componentwise integral)}$$

Integrating by parts,

$$Df_\sigma(x) = \int \nabla G_\sigma(x-y) f(y)dy.$$

Rearranging the terms in the integrand, this can be rewritten as (see section 7.6 of Rubinstein[12] for details):

$$Df_\sigma(x) = \int G_\sigma(x-y)\tilde{f}_\sigma(y)dy$$

for a suitably defined $\tilde{f}_\sigma(\cdot)$. But the right-hand side is simply the expectation of $\tilde{f}_\sigma(x+\xi)$ for an appropriate Gaussian random variable ξ, which can be estimated by a Monte Carlo scheme.

These considerations lead to the estimate[12,16] for $\nabla J(\theta(n))$ given by

$$\frac{1}{\beta}\frac{1}{N}\sum_{i=1}^{N}\eta_i J(\theta(n)+\eta_i\beta)$$

where $N > 1$, $\beta > 0$ 'small' and $\{\eta_i\}$ are i.i.d. $N(0, 1)$. This, in fact, is a Monte Carlo estimate for $DJ_\sigma(\theta(n))$, which in turn is an approximation of $\nabla J(\theta(n))$. One may achieve this averaging by the two-time scale scheme described earlier. This suggests the scheme:

$$x(n+1) = x(n) + b(n)\left[\frac{\eta_n}{\beta}k(Y(n)) - x(n)\right],$$

$$\theta(n+1) = \Gamma(\theta(n) - a(n)x(n)),$$

with $a(n) = o(b(n))$, where the HMM is now governed by $\{\theta'(n)\}$ given by

$$\theta'(n) = \theta(n) + \eta_n\beta.$$

The important feature of this scheme is that the derivative estimation is done only indirectly, thus requiring only a single simulation. The standard deviation β of the Gaussian has to be small for this estimation to be justified. But since β also appears in the denominator, very small β tends to cause numerical instability. Our 'robust' version then is

$$\theta(n+1) = \Gamma(\theta(n) - a(n)\mathrm{sgn}(x(n))).$$

Scheme 3: Correlation-based (CB) scheme

This scheme is based on the observation that $(f(x) - f(y))(x_i - y_i)$ and $(f(x) - f(y))/(x_i - y_i)$ have the same sign when $x_i \neq y_i$ (x_i, y_i being the ith components of x and y, respectively). Thus, one may consider the algorithm

$$\theta(n+1) = \Gamma(\theta(n) - a(n)(k(Y(n)) - k(Y(n-1)))(\theta(n) - \theta(n-1))).$$

This, however, turns out to be numerically ill-behaved, the reason being that $\theta(\cdot)$ changes very little over a single iterate. This suggests taking $Y(n-N)$, $\theta(n-N)$ in place of $Y(n-1)$, $\theta(n-1)$ in the above for a moderately large $N > 1$. Even better, one may average the expression multiplying $a(n)$ to stabilize it further. This averaging, in turn, can be affected by using two-time scales as explained earlier. Thus, the algorithm becomes

$$x(n+1) = x(n) + b(n)((k(Y(n)) - k(Y(n-N)))(\theta(n) - \theta(n-N))),$$

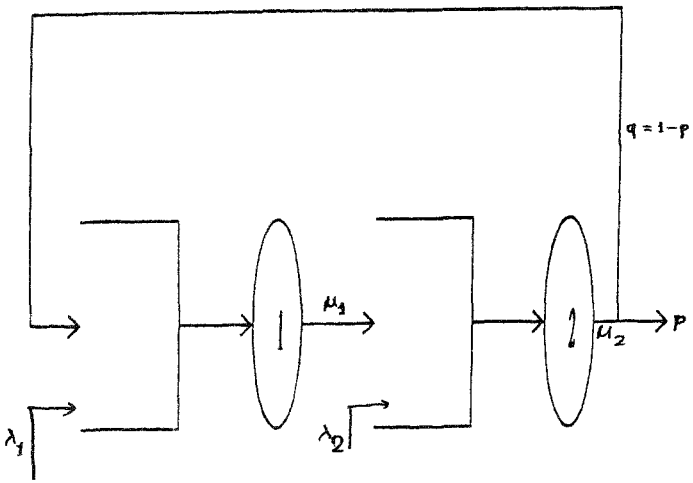

FIG. 1. Queueing network for problem 2.

$$\theta(n+1) = \Gamma(\theta(n) - a(n)\, x(n)),$$

with $a(n) = o(b(n))$. The robust version replaces the second recursion by

$$\theta(n+1) = \Gamma(\theta(n) - a(n)\mathrm{sgn}(x(n))).$$

It should be remarked parenthetically that there is some interest in correlation-based algorithms in computer vision literature because similar mechanisms are believed to operate in human vision. These considerations have led to the ALOPEX algorithm.[18,19] This is very similar to ours except that it does not use the sign of the estimated correlation, but another ± 1-valued random variable whose probability distribution is modulated by the empirical correlation through a Gibbsian mechanism.

Numerical experiments

The above algorithms were tried on the following two problems:

Problem 1

Consider a GI/G/1 queue. Let $\{\alpha(n)\}$ be the sequence of interarrival times to the queue and $\{\sigma(n)\}$ the sequence of service times. Here, we shall assume that $\sigma_n = \sigma_n(\theta)$, i.e. the service times are parameterized by a parameter θ which the server can tune. The times $\{T(n)\}$ spent by the nth customer in the system satisfy the recursion

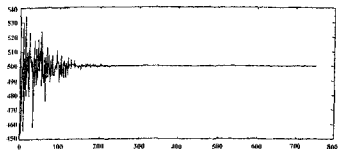

FIG. 2. Convergence of the finite-difference algorithm for problem 1.

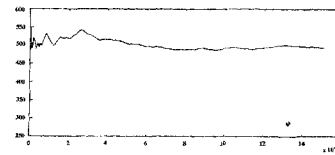

FIG. 3. Convergence of the robust finite-difference algorithm for problem 1.

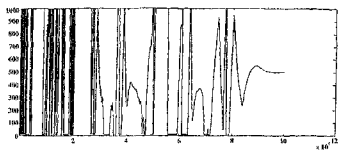

FIG. 4. Convergence of the smoothed functional algorithm for problem 1.

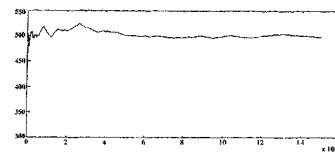

FIG. 5. Convergence of the robust smoothed functional algorithm for problem 1.

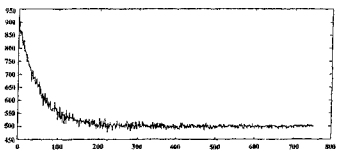

FIG. 6. Convergence of the correlation scheme for problem 1.

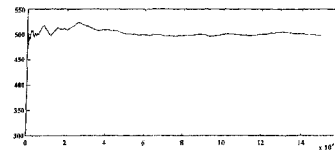

FIG. 7. Convergence of the robust correlation scheme for problem 1.

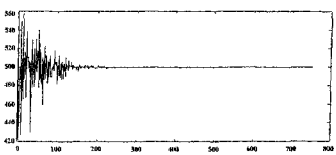

FIG. 8. Convergence of the finite-difference algorithm for queue 1 in problem 2.

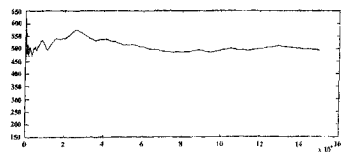

FIG. 9. Convergence of the robust finite-difference algorithm for queue 1 in problem 2.

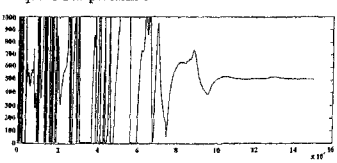

FIG. 10. Convergence of the smoothed functional algorithm for queue 1 in problem 2.

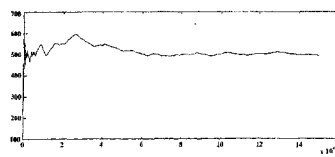

FIG. 11. Convergence of the robust smoothed functional algorithm for queue 1 in problem 2.

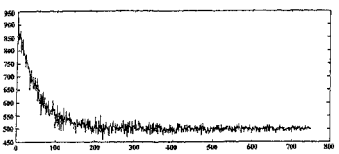

FIG. 12. Convergence of the correlation scheme for queue 1 in problem 2.

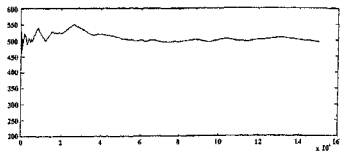

FIG. 13. Convergence of the robust correlation scheme for queue 1 in problem 2.

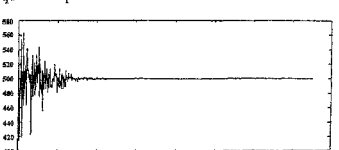

FIG. 14. Convergence of the finite-difference algorithm for queue 2 in problem 2.

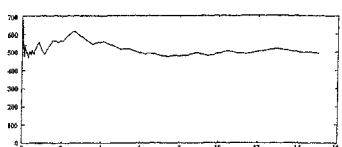

FIG. 15. Convergence of the robust finite-difference algorithm for queue 2 in problem 2.

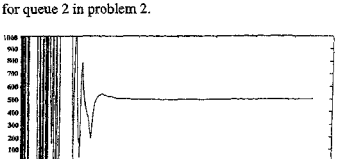

FIG. 16. Convergence of the smoothed functional algorithm for queue 2 in problem 2.

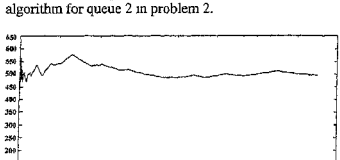

FIG. 17. Convergence of the robust smoothed functional algorithm for queue 2 in problem 2.

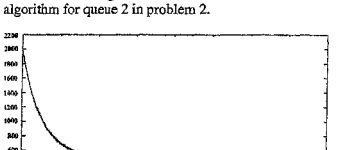

FIG. 18. Convergence of the correlation scheme for queue 2 in problem 2.

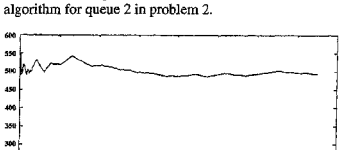

FIG. 19. Convergence of the robust correlation scheme for queue 2 in problem 2.

$$T(n+1) = (T(n) + \sigma_{n+1} - \alpha_{n+1})I\{\alpha_{n+1} \leq T(n)\} + \sigma_{n+1}I\{\alpha_{n+1} > T(n)\},$$

where $I\{\cdots\}$ denotes the indicator function. The cost we seek to minimize is $E[T(n)]$.

In the simulation results reported below, we take α_ns to be exponentially distributed with rate 0.2, and σ_ns with rate $\mu(\theta(n)) \triangleq 150/(1 + |\theta(n) - 500|)$. The initial guess is $\theta(0) = 100$ and $\Gamma(\cdot)$ is the projection onto [10,1000].

Problem 2

We consider here the network of two queues with feedback depicted in Fig. 1. Here, $\lambda_1 = 0.2$, $\lambda_2 = 0.1$ are the rates of two independent Poisson streams entering nodes 1 and 2, respectively. After getting served at node 1, a customer joins node 2. After getting served at node 2, he either leaves the system with probability p or is fed back to node 1 with probability $q = 1-p$. The service discipline is FCFS for both irrespective of where the customer comes from. The service times at the two nodes are exponential with rates $\{\mu_1(\theta_1(n))\}$, $\{\mu_2(\theta_2(n))\}$, respectively, with $\mu_i(\theta_i) = \overline{\mu}_i / (1+|\theta_i - \overline{\theta}_i|)$ where $\overline{\theta}_1 = 400, \overline{\theta}_2 = 600, \mu_1 = 360, \overline{\mu}_2 = 520, q = 0.7$. The cost function is $E_0[T_1(n) + T_2(n)]$ where $\theta = [\theta_1, \theta_2]$ and $T_i(n)$ = the time spent by the nth customer at the ith queue. The initial guesses were $\theta_1(0) = 100$, $\theta_2(0) = 200$.

The simulation results are shown in Figs 2–19. The stepsizes were $a(n) = C/n$ and $b(n) = C/n^{2/3}$ where $C = 1$ for the non-robust versions of the algorithms and $C = 50$ for the robust versions. Some broad conclusions that can be drawn from the empirical evidence are as follows.

1. In the lucky situations where the variance is already low, our variance reduction device actually worsens the performance. This is not untypical of variance reduction schemes. With this in mind, we reserve the subsequent comments for the high variance cases only.

2. For identical stepsize schedules, the robust versions are slower, as is to be expected. This can be amended by scaling up the stepsize for the robust version as above. The following comments refer to a comparison between robust and non-robust versions with the scaled stepsizes.

3. The robust versions converge more gracefully, with very little oscillatory behaviour. This is particularly pronounced in the initial stages. In the long run, however, the convergence is slower for the robust versions. (This is in conformity with the so-called bias-variance dilemma). In situations where they are used explicitly as a variance-reduction technique and not motivated by computational or communication complexity considerations, this suggests a two-tier strategy: the robust version in the initial stages till the iterations have 'settled' a little, and the original algorithm thereafter. More generally, one can consider graded schemes which switch from one 'quantized' version of the gradient to the next, more refined one, with the sign at one end of this spectrum (the coarsest) and the full gradient at the other.

4. Another advantage for robust versions is that by virtue of not having to contend with violent oscillations, the projection operation $\Gamma(.)$ is rarely called for.

5. We also tried numerical experiments where an additional noise with small variance was deliberately added to the argument of sgn(.) in the robust algorithm. This did not significantly alter their behaviour. It has also been suggested[8] that $a(n) = \varepsilon b(n)$ with a small $\varepsilon > 0$ will outperform $a(n) = o(b(n))$. Again, our simulations did not show any significant difference.

Note: In all the graphs x axis denotes θ values, y axis denote number of iterations.

Acknowledgements

The work of the second author was supported by the Department of Science and Technology, Grant No. III 5(12)/69-ET.

References

1. BENVENISTE, A., METIVIER, M. AND PRIOURET, P. — *Adaptive algorithms and stochastic approximations*, Springer–Verlag, 1990.
2. POLYAK, B. T. AND TSYPKIN, YA. Z — Pseudogradient adaptation and training algorithms, *Automation Remote Control*, 1973, **3**, 377–398.
3. HO, Y.-C. AND CAO, X.-Y. — *Perturbation analysis of discrete event dynamical systems*, Kluwer, 1991.
4. CHONG, E. K. P. AND RAMADGE, P. J. — Stochastic optimization of regenerative systems using infinitesimal perturbation analysis, *IEEE Trans.*, 1994, **AC-39**, 1400–1410.
5. BHATNAGAR, S. AND BORKAR, V. S. — A two time-scale stochastic approximation scheme for simulation based parametric optimization, *Probability Engng Inf. Sci.* (to appear).
6. BORKAR, V. S. — Stochastic approximation with two time scales, *Systems and Control Lett.*, 1996, **39**, 291–294.
7. FOGEL, E. — A fundamental approach to the convergence analysis of least squares algorithms, *IEEE Trans.*, 1981, **AC-26**, 646–655.
8. RUSZCZYNSKI, A. AND SYSKI, W. — Stochastic approximation method with gradient averaging for unconstrained problems, *IEEE Trans.*, 1983, **AC-28**, 1097–1105.
9. POLYAK, B. T — New method of stochastic approximation type, *Automation Remote Control*, 1990, **51**, 937–946.
10. YIN, G — On extensions of Polyak's averaging approach to stochastic approximation, *Stochastic Stochastic Rep.*, 1991, **36**, 245–264.
11. RIPLEY, B. D. — *Stochastic simulation*, Wiley, 1987.
12. RUBINSTEIN, R. Y. — *Simulation and the Monte Carlo method*, Wiley, 1981.
13. FABIAN, V. — Stochastic approximation methods, *Czek. Math. J.* 1960, **10**, 123–159.
14. BORKAR, V. S. — Asynchronous stochastic approximation, *SIAM J. Control Optimization*, 1998, **36**, 840–851.
15. KUSHNER, H. J. — *Approximation and weak convergence methods for random processes, with applications to stochastic systems theory*, MIT Press, 1984
16. SPALL, J. C. — Multivariate stochastic approximation using a simultaneous perturbation gradient approximation, *IEEE Trans.*, 1992, **AC-37**, 332–341.
17. KATKOVNIK, V. YA. AND KULCHITSKY, YU. — Convergence of a class of random search algorithms, *Automation Remote Control*, 1972, **8**, 1321–1326.
18. UNNIKRISHNAN, K. P. AND VENUGOPAL, K. P. — Alopex: a correlation-based learning algorithm for feedforward and recurrent neural networks, *Neural Computation*, 1994, **6**, 469–490.
19. SASTRY, P. S. AND UNNIKRISHNAN, K. P. — Analysis of Alopex optimization algorithm (preprint).