There's No Such Thing as a Free Lunch

The Bias-Variance Dilemma

Vivek S Borkar

Vivek S Borkar is with the Department of Computer Science and Automation, Indian Institute of Science, Bangalore. He is one of those who do not know much statistics, but wish they did.

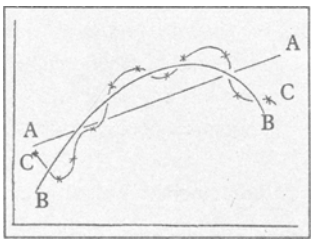
A much publicised (but rarely explained) dilemma in empirical model building is described. Techniques for 'balancing on its horns' are outlined. In particular, it is argued that one can get an edge over it by not going over the edge.

The Razor's Edge

Many years ago when I was with an institution inhabited by mathematicians and astronomers, a story was making the rounds of the place. Once an astronomer and a mathematician went hiking in the western ghats and saw a black cow in profile at a distance. "All cows in the western ghats are black", exclaimed the astronomer. "No", said the mathematician, "on date such and such and time so and so, there existed in the Western Ghats a cow the left half of whose body appeared black at that time".

Clearly, both are missing something. The astronomer makes a simplistic but flawed generalisation. The mathematician is accurate to a fault, but won't make even the obvious extrapolations. These might be extreme positions, but the underlying choice between the simple and the complex is something most model-builders have to contend with all the time. The problem is always the same. One has a finite amount of observations (data), based on which one has to infer something about the underlying reality relatively quickly. The crux of the matter then is how to avoid reading either too much or too little into the data.

Figure 1.



Let's take a more serious example. Given a finite sample $(x_1, y_1), \ldots, (x_n, y_n)$ of pairs of real numbers with a hypothesised relationship $y_i = f(x_i)$ + 'noise', one wants to 'learn' $f(\cdot)$. *Figure 1* shows one such situation where curves A, B, C are candidate f's fitted from the class of straight lines, quadratic functions and all

polynomials, respectively, by minimising over these classes the 'least squares' error :

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2 . \qquad (1)$$

One look at the picture and our intuition tells us that B is about right. A is simple, but too crude a fit. C is accurate but too complicated. How does one justify this intuition?

Given such a multiple choice situation, a computer scientist would typically invoke Occam's razor. Named after William of Occam, this principle states that one should pick the simplest of all valid explanations. But this razor doesn't quite cut here, because unlike in the Boolean world of computer scientists, here there is no clear-cut dichotomy between valid and invalid. There are only degrees of validity.

Naive error minimisation won't do either, because then C is better than B. The fallacy lies in our use of the 'training data' $\{(x_i, y_i), 1 \le i \le n\}$, used for fitting these f's, also for comparing them. The data cannot yield more information than what it already has. What is needed is to take fresh 'out of sample data' or 'test data' for comparing the fits. (In practice, one usually has a single data set and splits it artificially into a largish chunk – say, 70%, of 'training data', and the remainder as 'test data'.) No matter what we do, the error on test data won't in general be zero due to the noise. In fact one cannot rule out the risk of a freak streak of pathological test data that gives a large error for even the true $f(\cdot)$. Typically, such pathologies are rare, i.e., of low probability. Thus at best, one aspires for a good approximation with high probability, i.e., to be 'probably, approximately correct.'

The error on test data $(\bar{x}_1, \bar{y}_1), \ldots, (\bar{x}_m, \bar{y}_m)$ is given by

$$\frac{1}{m}\sum_{i=1}^{m}(\bar{y}_i - f(\bar{x}_i))^2 \qquad (2)$$

and is called the generalisation error, to contrast it with the 'training error' given by (1). If we plot the two against a suitable

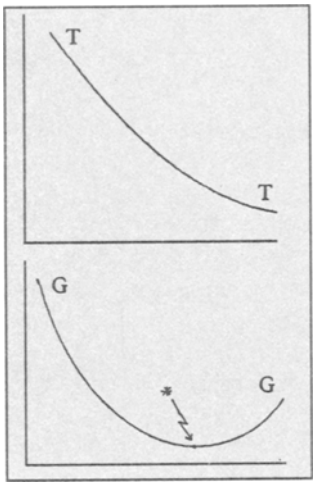


Figure 2.

complexity parameter (say, the degree of the polynomial being fitted), one typically gets something like *Figure 2*. The training error (T) decreases monotonically as it should, since the error minimisation is being performed over successively larger sets of functions. The generalisation error (G), however, first takes a dip and then rises again. The lowest point (marked by a '*' in *Figure 2*) sits on the edge of underfitting (simplistic models leading to error) and overfitting (complex models that read too much into the data, leading to error). It is this 'point on the edge' that we are after.

Following Occam's razor, we may dub this principle of seeking the point on the edge of simplicity and complexity as 'Einstein's razor', after the following quotation attributed to Einstein: "Everything should be made as simple as possible, but not simpler".

Lies, Damn Lies and Statistics

This, apparently, was the classification of lies according to Disraeli. But despite the misgivings of him and his ilk, statisticians have been plying their trade with a not inconsiderable success (ironically, notably so in his own country). It is this much maligned subject that we have to fall back upon in order to make sense of the foregoing.

Ideally, we want to minimise over all f the 'risk'

$$E[(Y-f(X))^2]\,, \qquad (3)$$

where $E[\cdot]$ denotes the mathematical expectation (i.e., the probabilistic average) with respect to the joint distribution of (X, Y). The optimum is attained by $f^*(X)=E[Y/X]$, the conditional expectation of Y given X. Thus $E[(Y-E[Y/X])^2]$ is the 'unavoidable error'. In practice, there are two additional sources of error. We don't usually know the joint distribution in question, so that the conditional expectation cannot be evaluated. We then replace (3) by the 'empirical risk' (1). For each fixed f, (1) approximates (3) well in the large n limit, by the law of large

numbers. But in order that the minimisation of (3) in place of (1) be justified, one needs this approximation to be uniformly good for all f under consideration, i.e., a 'uniform law of large numbers' must hold. This forces us to consider restricted classes of f for which such a result is available. Other considerations (such as computational, smoothness constraints on f, physical or biological analogies, . . .) may restrict the class F of candidate f's even further, e.g., to splines or feedforward neural networks. The two sources of error then are the finite sample size n and the restricted search domain F. We shall now try to quantify the error contribution of each.

Let D denote the training data vector $[x_1, y_1, x_2, y_2, \ldots, x_n, y_n]$ and f_D the element of F that minimises (1). The net error then is $E[(\bar{y}_i - f_D(\bar{x}_i))^2]$, which can be written as a sum of three terms:

$$E[(\bar{y}_i - f_D(\bar{x}_i))^2] = E[(\bar{y}_i - E[\bar{y}_i/\bar{x}_i])^2] + E[(E[\bar{y}_i/\bar{x}_i] - E_D[f_D(\bar{x}_i)])^2] + E[(E_D[f_D(\bar{x}_i)] - f_D(\bar{x}_i))^2], \quad (4)$$

where $E_D[f_D(\bar{x}_i)]$ is our compact notation for $E[f_D(\bar{x}_i)/\bar{x}_i]$ (i.e., one is averaging over all possible training vectors D, keeping $\bar{x}_i$ fixed).

The proof of this pudding lies in splitting. Split $(\bar{y}_i - f_D(\bar{x}_i))$ as follows, $(\bar{y}_i - E[\bar{y}_i/\bar{x}_i] + (E[\bar{y}_i/\bar{x}_i] - E_D[f_D(\bar{x}_i)]) + (E_D[f_D(\bar{x}_i)] - f_D(\bar{x}_i))$, complete the square, take expectation, and watch the cross terms drop out by virtue of being uncorrelated. (This takes a simple conditioning argument, exactly along the lines of the proof of the fact that (3) is minimised at $E[Y/X]$. This is standard textbook material.)

The first term on the right in (4) is the unavoidable error. The second and the third are called respectively the bias and the variance terms. There is a trade-off between these that leads to our 'Einstein's razor'. The popular tag for this trade-off is the 'bias-variance dilemma', called thus because beyond a point, one can reduce one only at the expense of increasing the other [3–5]. How this comes about is the main theme of this article.

Suppose we had access to independent and identically distributed 'copies' $\{D_i\}$ of the training data vector D. The $E_D[f_D(\bar{x}_i)]$ could be obtained as the large N limit of $1/N\sum_{j=1}^{N} f_{D_j}(\bar{x}_i)$. Thus the passage from $f_D(\bar{x}_i)$ to $E_D[f_D(\bar{x}_i)]$ washes out the effect of finite sample size in some sense, leaving an error primarily due to a restrictive choice of F. This is the bias term, which decreases as one admits larger, more complex F.

The variance term is indeed the averaged conditional variance $E[E[f_D(\bar{x}_i) - E[f_D(\bar{x}_i)/\bar{x}_i])^2/\bar{x}_i]]$. It makes no explicit reference to $E[\bar{y}_i/\bar{x}_i]$. Rather, it measures the extent to which $f_D(\bar{x}_i)$ as a function of D fluctuates from $E_D[f_D(\bar{x}_i)]$, its average over D, thereby pinning down the finite sample effect. Not surprisingly, under moderate conditions this term goes to zero as $n \to \infty$ for most traditional statistical schemes. We shall, however, eschew such 'asymptopia' and work with finite, fixed n. The claim then is that this error tends to grow as F becomes more complex.

A Cursed Affair

The culprit here is a familiar foe, the curse of dimensionality. To get a feel of this, consider planting points about 1 cm apart in an interval of length 1 m. You need about 100. Do the same in a 1m× 1m square, you need about 10^4. For 1m×1m×1m cube, around 10^6 and so on. The message is: The number of points needed to 'sample' the unit hypercube to a given accuracy grows exponentially with the dimension of the underlying space.

Now consider minimising (1) over function classes $F_1, F_2, \ldots$ where each F_i is a family of functions parametrised by some parameter vector $\beta(i)$ belonging to a subset $A(i)$ of, say, $m(i)$-dimensional Euclidean space. Suppose that $m(i)$ increases with i, corresponding to increasing complexity of the function class. Furthermore, we suppose that (1) is minimised by a recursive algorithm (such as recursive least squares), which is run for (say) k steps. The operation $E_D[\cdot]$ implies averaging over all such trajectories of the algorithm.

The 'curse' operates in more than one way. First and foremost, recall the problem of packing points in $A(i)$ with prescribed maximum separation between neighbours. As already seen, this grows exponentially with $m(i)$. A learning algorithm typically makes moves of a certain order of magnitude. The above then suggests that it will take exponentially longer to explore the parameter space in higher dimensions. In other words, a fixed run length thereof will explore the parameter space much less efficiently with increasing dimension.

In addition, the performance of the algorithm itself can deteriorate with dimension. A typical algorithm moves from one point to a nearby point 'incrementally'. In higher dimensions, there are more degrees of freedom and more ways in which an algorithm can 'curl up'. (Recall that the algorithm is 'stochastic'.) Thus even when it is asymptotically convergent to the 'ideal' point, its actual net movement in k steps will tend to be better in low dimensions for fixed k.

One may think of getting around this problem by choosing a bigger stepsize for the algorithm. But this only reintroduces the bias-variance dilemma through the backdoor (see *Box 1*).

The reader is invited to ponder over both these points in the context of the passage from f_D to $E_D[f_D]$ described earlier and convince himself that both work against lower variance if the dimension is higher.

To illustrate the point further, consider the scheme of *Figure 3*.

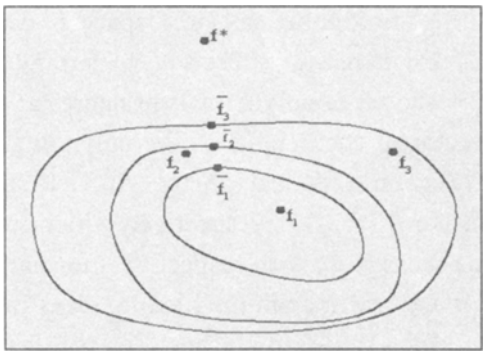


Figure 3.

Box 1. The Hare vs. the Tortoise

A typical 'learning' algorithm has the form

$$X(n+1) = X(n) + a(n)\,(h(X(n)) + M(n+1)),$$

where $\{a(n)\}$ is the stepsize sequence and $\{M(n)\}$ the 'noise'. For simplicity, let $a(n)$= a constant ('a') for all n. View 'a' as a small time step and the algorithm itself as a noisy discretisation of the differential equation

$$dx(t)/dt = h(x(t)).$$

Suppose $x(t)$ converges to x^* for any $x(0)$. Then $X(n)$ may be expected to remain in a neighbourhood of x^* with high probability for large n. This, roughly, is what most of these algorithms do.

Keeping in mind our picture of 'a' as a time step, a fixed run length of, say, k, of the algorithm will 'simulate' $x(.)$ over a time interval of length ka, which increases with a. Thus it moves further towards its destination with larger a, suggesting lower bias.

But the flip side of this is that the approximation error, even without noise, grows rapidly with a and worse, the total noise variance in simulating $x(.)$ over a time interval of length T goes as $(T/a).a^2$, which increases with a.

Thus these 'great leaps forward' do take us further, but that could be farther from the goal because they are so erratic. This is the bias-variance dilemma by any other name.

There are other avatars of this phenomenon. For example there are variance reduction techniques based on averaging, either explicitly averaging the iterates or implicitly averaging the right hand side of the recursion above by introducing a 'momentum' term. These introduce a kind of 'inertia' in the dynamics that makes it more sluggish, increasing bias.

You simply can't eat your dosa and have it too!

Here f^* is the ideal point in some space of functions and $F_1, F_2, F_3, \ldots$ are increasing subsets of the latter. (For example, consider F_i = the set of polynomials of degree at most i. Then $\beta(i)$ is the vector of coefficients of the polynomials, sitting in $A(i) = R^{i+1}$.) A good statistical scheme will yield, in the $n \to \infty$ limit, points like $\bar{f}_1, \bar{f}_2, \bar{f}_3, \ldots$ respectively which are the 'closest' to f^* in some sense (e.g., with respect to information theoretic divergence if we use maximum likelihood estimation). Our inference based on a finite sample, however, may lead us to points

Box 2. Portrait of the Artist as an Young Edge-detector

Cognitive scientists recognise a kind of bias-variance dilemma in human learning. Point out a rabbit sitting on a lawn to a child and say 'rabbit'. The child does not interpret it as 'a white object of certain dimensions' and refer to a folded white towel as a rabbit. Nor does it think of it as 'a certain stationary arrangement of two long ears, whiskers, two eyes,. . . on a green backdrop', and fail to recognise a rabbit when it is running, chewing a carrot, or doing other things that rabbits do. It pitches its interpretation of the word 'rabbit' at the correct edge of simplicity and complexity and realises, as Gertrude Stein might have put it, that 'a rabbit is a rabbit is a rabbit is a rabbit is a rabbit'.

How did this innate ability for 'edge-detection' come by? The answer lies in evolution. Only the brains that could 'get the edge' got an edge in the battle for survival.

like f_1, f_2, f_3, . . . respectively. Note that there is a progressive improvement from $\bar{f}_1$ to $\bar{f}_2$ to $\bar{f}_3$, but though f_2 improves upon f_1, f_3 does not improve upon f_2. Our 'Einstein's razor' would then pick f_2 as the point on the edge of the bias-variance dilemma. As an aside, see *Box 2* for a different (but not quite) kind of dilemma.

I close this section with the caveat that the above generalisations about generalisation have a large dose of heuristics, to an extent that one might be tempted to say, 'Lies, damn lies and heuristics!' Take these as aids to intuition, not as mathematical theorems (though there are a few of those lurking underneath. [4]). As Alexandre Dumas said, "All generalisations are dangerous, even this one".

Putting Expensive Bits on Posteriors

This still leaves out the edgy issue of how to 'get the edge' (or rather, how not to go over it). One way to aim for the 'point on the edge' is by rigging the error criterion (1) to have a minimum thereabouts. Here we take a cue from the penalty function method of constrained optimisation, in which the constraints are accounted for by adding a 'penalty' term to the function being minimised, with the provision that this term takes very high values at points that don't meet the constraints. In the same spirit, one adds to (1) a 'penalty term' that depends on the model class and increases with its complexity. A

prominent scheme in this vein is the 'minimum description length' (MDL) principle of Rissanen, where one minimises the total code length, i.e., the number of binary digits (bits) required to first encode the model and then the data. Related schemes are the older 'Akaike information criterion' and its antecedent by Gideon Schwartz, the 'Bayesian information criterion' and the recent work of Andrew Barron on complexity regularisation [2]. We shall try to get a feel for the spirit of such schemes in this section.

Specifically, consider model classes $F_1, F_2, F_3, \ldots$ of increasing complexity with associated increasing 'complexity cost' $L(1), L(2), L(3), \ldots$. One then minimises jointly over $i = 1, 2, \ldots$ and $f \in F_i$ the expression

$$\frac{1}{2}\sum_{i=1}^{n}(y_i - f(x_i))^2 + \lambda L(i) \qquad (5)$$

Box 3. Some Vital Statistics

Suppose you observe the value (say, y) of a random variable Y whose distribution you want to estimate. For simplicity, let Y be discrete-valued. One may hypothesise a parametrised family P_θ of probabilities on the range S of Y, with the parameter θ in some parameter space A (assumed discrete once again). It is implicit that some θ_0 in A is the 'true' parameter, i.e., probability law of Y is P_{θ_0}. One may estimate θ_0 by $\hat{\theta}$ = the θ that maximises $P_\theta(y)$. This is the maximum likelihood estimate, called non-Bayesian because no 'prior' probability was imposed on A as in the Bayesian framework we discuss next.

In the Bayesian framework, θ_0 itself is assumed to be a realisation of some A-valued random variable Z with probability law μ (the 'prior'). Bayes rule then leads to

$$P(Z = \theta / Y = y) = P_\theta(y)\,\mu(\theta) \Big/ \left(\sum_{\eta \in A} P_\eta(y)\,\mu(\eta)\right)$$

the posterior probability on A given the observation y. Maximising this over θ yields the maximum a posteriori (MAP) estimate of θ_0. In the context of our discussion in section 4, it is worth noting that this is equivalent to maximising the logarithm of the same, which is the sum of the usual 'log-likelihood function' of maximum likelihood scheme given by $\log(P_\theta(y))$, plus a term involving the prior that can be viewed as a penalty term.

Both fall in the category of parametric statistics, because we started with a parametrised family P_θ, $\theta \in A$, of candidate probability laws. 'Nonparametric statistics' is another, altogether different ball game.

for a prescribed $\lambda > 0$. (For example, in MDL, $L(i) = (1/2)m(i)ln(n)$ with $m(i)$ as before. This is related to the information theoretic 'redundancy' of a code.)

To put all this in the familiar perspective of traditional statistical paradigms (*See Box 3*), note that minimising (1) is equivalent to maximising

$$\prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-(y_i - f(x_i))^2/2} \qquad (6)$$

In other words, we hypothesise that $y_i = f(x_i) + \text{noise}$, with the latter being an independent zero mean, unit variance, Gaussian random variable. Then (6) is the likelihood function and f_D the maximum likelihood estimate of f. Correspondingly, (5) amounts to maximising

$$\prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}} e^{-(y_j - f(x_j))^2/2} \times C(\lambda) e^{-\lambda L(i)} \qquad (7)$$

where $C(\lambda) = (\Sigma_i e^{-\lambda L(i)})^{-1}$ is the normalising factor. This does not have to be finite, but we assume it is. It often is, in particular if $L(i)$ has interpretation as the codelength of some 'instantaneous' binary code (i.e., a code that can be decoded without reference to future bits).If so, Kraft's inequality of information theory ensures the finiteness of $C(\lambda)$: It says that $\Sigma 2^{-L(i)}$ cannot exceed 1. The term $C(\lambda)e^{-\lambda L(i)}$ can be viewed as the prior probability we have put on F_i. Then (7) is 'sort of' proportional to the posterior probability. (The qualification 'sort of' is inserted because we haven't put a prior probability on $A(i)$ given i. This can also be done, e.g., in order to build in any additional prior knowledge or to penalise further those $\beta(i)$ with, say, large $||\beta(i)||$. Thus the maximisation of (5) is 'sort of' like the maximum a posteriori estimate, establishing a link with Bayesian statistics [7].

There are also other approaches for finding the edge. Many of these first overfit a very complex model, which is then pruned by discarding apparently unimportant parameters using an appropriate heuristic. One such scheme in the context of neural networks goes by the quaint name of 'optimum brain damage',

Box 4

'BAGGing' is an acronym for 'Bootstrapped AGGregation', which gives away its antecedents from the bootstrap method in statistics. 'Bootstrap', in statistics, refers to schemes which resample from a given data set to generate additional data sets and use them for inference [6]. (The terminology apparently comes from a certain Baron Munchausen, a fictional character who is supposed to have pulled himself up from the bottom of a lake by pulling at his bootstraps.)

stretching their biological analogy perhaps a bit too far.

In the neural network context, there are also schemes based on 'weight sharing'. These impose a priori relationships between parameters, thereby bringing down the effective dimension of the parameter space.

Baggers Can be Choosers

The reader needn't go away with the impression that there is no way to beat down variance without increasing the bias. At the algorithmic level, a good compendium of variance reduction techniques is the book by A Davison and D Hinkley [6]. Here, however, we shall concentrate on a different bag of tricks, viz., bagging (see *Box 4*) and related techniques [5].

In bagging, one samples from $D=[(x_1,y_1), (x_2,y_2), \ldots, (x_n,y_n)]$ independently, with uniform probability and with replacement to form another data string $D_1 = [(\tilde{x}_1, \tilde{y}_1), (\tilde{x}_2, \tilde{y}_2), \ldots, (\tilde{x}_n, \tilde{y}_n)]$. In particular, repetitions are possible. Repeat this procedure to form further strings $D_2, \ldots, D_N$. Setting $D_0 = D$, find the best fit f_{D_i} for each i and then let $\bar{f}_D = (1/(N+1))\sum_{i=0}^{N} f_{D_i}$. The idea is clear: Limited by a single data string D, we 'simulate' additional data strings D_i through resampling and combine the 'findings' $\{f_{D_i}\}$ into an averaged $\bar{f}_D$, which then may be hoped to be closer to $E_D[f_D]$ than the original f_D is, with a high probability. If so, the variance term is reduced without affecting the bias significantly. Empirical work supports this intuition.

There is a further refinement which often does better. This is a scheme called 'arcing', once more an acronym (this time for 'Adaptive Resampling and Combining') [8]. The main difference here is that the sampling distribution, initially uniform, is adaptively modified to favour data points that give larger error. One may also play around with the weights used for forming $\bar{f}_D$ from $\{f_{D_i}\}$ through a weighted sum. (These were uniform $(=1/n)$ in bagging.)

A word of caution: Bagging and related techniques work well

when the variance term is initially very high. But if it is low to start with, f_D may turn out to be a worse estimate than f_D! Intuitively, these techniques correct for sensitive (or 'non-robust') dependence on data of the algorithm. This roughly reflects the second manifestation of the curse of dimensionality mentioned above, viz., the data-dependent 'wandering' of the algorithm. The first and the more fundamental issue – that it simply takes many more points to sample the space in higher dimensions – will not be adequately handled by these schemes. Other caveats regarding bootstrap schemes also apply, e.g., their problems with incomplete or dependent data, outliers etc. [6] for the 'when and how' of bootstrap.

After having waxed eloquent about the bias-variance dilemma in earlier sections, it may come as an anti-climax that one can escape it to some extent through 'boosting techniques' like bagging and arcing that improve the accuracy of the training procedure through its repeated application. But in reality, I have only replaced one dilemma by another. These techniques are computationally intensive and we have escaped (albeit partially) the bias-variance dilemma only to find ourselves in the dilemma of variance vs. computational resources.

But then, there's no such thing as a free lunch.

Suggested Reading

[1] R Rubinstein. *Simulation and the Monte Carlo Method.* John Wiley. New York, 1981.

[2] A Barron. Complexity regularisation with application to artificial neural networks, in *Nonparametric Functional Estimation and Related Topics*).G. Roussas (ed.). Kluwer Academic Publishers. 1991, pp. 561–576.

[3] S Geman, E Bienenstock and R Doursat. Neural networks and the bias-variance dilemma. *Neural Computation.* 2, 1–58, 1992.

[4] V Vapnik. *The Nature of Statistical Learning Theory.* Springer Verlag. New York, 1995.

[5] L Breiman. *Bagging predictors, Machine Learning.* 26(2). 123–140, 1996.

[6] A Davison and D Hinkley. *Bootstrap methods and their applications.* Cambridge Uni. Press. Cambridge, 1997.

[7] D Lindley. *Bayesian Statistics: A Review.* SIAM. Philadelphia, 1997.

[8] L Breiman. Arcing classifiers. *Annals of Statistics*, 1998.

Address for Correspondence
Vivek S Borkar
Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560 012, India
Email:
borkar@csa.iisc.ernet.in