The Other Monte Carlo Method

Isabel Beichl and Francis Sullivan

1. Introduction

In the past few years there has been a surge in research activity concerning computational methods that use randomness in ways quite different from its use in the traditional Metropolis algorithm or its use in any of the many relatives and offspring of the Metropolis algorithm. The community seems to have settled on the term "sequential importance sampling" (SIS) for one such class of *other* Monte Carlo methods. What the term SIS means will become clear as we go along. For now, just think of SIS as sampling according to a non-uniform probability distribution that is generated as the choice is being made and then dividing by the observed probability in order to remove bias.

Sequential importance sampling has been used with remarkable success in fields including biology, astronomy, simulation of self-avoiding random walks, signal processing and evolutionary theory [6]. The present authors have used it to estimate the permanent and thus estimate the dimer covering constant [2]. In an early paper [5], Knuth used it to estimate the amount of work needed to solve a puzzle by a backtrack algorithm. Interest in SIS comes from the fact that it has sometimes succeeded on problems where the Monte Carlo Markov Chain method (MCMC) has failed or where it is far from obvious how to apply MCMC. Naturally, there are also cases in which SIS, or rather the precise version of SIS used, simply does not work. Part of the difficulty in understanding exactly when and how SIS should be tried is a consequence of the almost total lack of theory for this method, in contrast to MCMC, which now has a robust theoretical foundation and a well-developed set of criteria for determining rate of convergence. The other articles in this issue of CiSE provide background on some of the theory. But because of the increasing number of applications, constructing a theory for SIS is an important challenge facing the statistics research community.

2. The basic idea

We begin by introducing importance sampling in a way that's slightly non-standard but which gets us quickly to some of the central ideas in SIS. After

importance sampling is illustrated, we'll give examples of how to use it sequentially.

Suppose we are working with the union of two sets A and B

$$S = A \cup B$$

and we know the size of A, $|A|$, the number of elements in the set A, and the size of B, $|B|$. We also assume that we can sample uniformly from set A or B and that we can easily determine if an element is in A or B or both. What we would like to do is to estimate the size of S, $|S|$, that is count the number of elements in S. We always have that

$$A \cup B = (A \Delta B) \cup (A \cap B)$$

where $A \Delta B$ is the symmetric difference of A and B, that is the elements of A that are not in B plus the elements of B that are not in A. So, because the two sets on the right side of the above equation are disjoint, we get

$$|A \cup B| = |A| - |A \cap B| + |B| - |A \cap B| + |A \cap B| \leq |A| + |B|.$$

The right side of this inequality is the simplest instance of the so-called Bonferroni bounds. In this case, it restates the obvious fact that $A \cup B$ may have fewer elements than the total of $|A|$ and $|B|$, because of double counting of the elements of $|A \cap B|$. The simple sum is not a very good estimate of $|A \cup B|$ because the intersection $|A \cap B|$ can be considerable. But we can use importance sampling to compute a number, $\rho \leq 1$, so that

$$|A \cup B| \approx \rho * (|A| + |B|).$$

gives a much better estimate.
We define two probabilities, P_A and P_B,

$$P_A = \frac{|A|}{|A| + |B|}, P_B = \frac{|B|}{|A| + |B|}$$

so that

$$P_A + P_B = 1.$$

Here is the algorithm for computing ρ. Choose one of set A or B, A with probability P_A, set B with probability P_B. If A was chosen, then choose

an element of A uniformly. The probability of choosing any one element of A will be $\frac{1}{|A|}$. Similarly if B was chosen. What is the probability in this procedure for choosing one particular element of $S = A \cup B$? There are three disjoint possibilities, the element is in $A \setminus B$, $B \setminus A$ or $A \cap B$. If the element is in the first case, A but not B, then the probability of selecting A was

$$\frac{|A|}{|A| + |B|}$$

Next, having chosen A, we choose an element uniformly. So, if A had been chosen, any particular element in it is chosen with probability

$$\frac{1}{|A|}$$

and similarly for B. In either case, multiplying the "choose a set" probability by the "choose an element" probability we get

$$\frac{1}{|A| + |B|}.$$

Now, let's calculate the *true* total probability of choosing some particular element, say f, from $A \cap B$ by this process. First we'd have to choose, say, B and then in B we'd have to choose f. Therefore, the probability of choosing f is

$$\frac{|B|}{|A| + |B|} * \frac{1}{|B|} = \frac{1}{|A| + |B|}.$$

But, the *true* total probability of choosing f, is

$$\left(\frac{|A|}{|A| + |B|} * \frac{1}{|A|}\right) + \left(\frac{|B|}{|A| + |B|} * \frac{1}{|B|}\right) = \frac{2}{|A| + |B|}.$$

To compute ρ we'll choose elements from A or B by the procedure described above and record the inverse of the *true* probability of selecting an element as just described. We do this by first choosing and then noting if the element chosen is in one or two sets. (Recall we said that it is assumed to be easy to do this check.) So each sample will be either

$$\frac{|A| + |B|}{1}$$

or else

$$\frac{|A| + |B|}{2}.$$

What will the mean of the samples be? Clearly something smaller than $|A| + |B|$, because every term has this factor multiplied by either 1 or 1/2. So the mean will be

$$(|A| + |B|) * \rho$$

where ρ is the mean of the 1's and 1/2's. So we have corrected our simple Bonferroni bound by dividing by the probability of choosing each particular element. By convention, the inverse of the probability is called the 'importance', hence the term *importance sampling*. More details on this method for correcting the Bonferroni bound can be found in [3].

3. Counting via importance sampling

Suppose now, more generally, we want to estimate a sum of T terms:

$$F = \sum_{t \in T} f_t$$

by sampling.

One strategy is to choose M samples f_s uniformly at random from among all $|T|$ values and then average to give the estimate, $|T|\langle f \rangle$. Note that this can be written:

$$\frac{1}{M} \sum_{s=1}^{M} \frac{f_s}{1/|T|}$$

That is, for each sample we divide by the probability of choosing that sample. Now an important point: we could, in fact, use *any* probability instead of $1/|T|$ giving

$$\langle f/p(f) \rangle = \sum_{s \in T} \frac{f_s}{p(f_s)} p(f_s) = \sum_{t \in T} f_t$$

The ideal choice, of course, is $p(f) = f/F$, but this presumes we already know the answer! However, there are situations in which one needs to design a good but not perfect $p(f)$. Suppose, for example, we don't know $|T|$. Notice that this idea still works so long as we can sample f and so can be used to estimate $|T|$ itself. All that is needed is a probability distribution on the set T for sampling the number 1. The distribution must cover all possible values from T and we don't know $|T|$. This is the genesis of "sequential" methods.

4. Sequential importance sampling

A number of years ago, Knuth developed an approximate counting method in an attempt to estimate the running time of a back-track program without actually performing the entire backtrack [5]. The underlying idea is simple. Recall that any backtrack can be thought of as a search of a tree. The search backs up to the first ancestor node having an available choice whenever it is blocked, and continues doing this until an "answer" is found or all nodes have been examined. If we imagine that the backtrack is a tree (not necessarily balanced) then this amounts to a depth first traversal of this tree that stops at a node satisfying some pre-specified condition. In many situations, it is useful to be able to estimate how much work will be done before the answer is found, i.e. to estimate how large the tree is without actually traversing all of it.

If the backtrack search generated a perfect binary tree, the size of the search is easy to estimate. Just determine d, the depth of the tree and assume that the whole tree must be traversed before finding the answer. The amount of work is then 2^{d+1}, the number of nodes in the tree. To determine d, just walk down one branch of the tree and count the number of steps to reach the leaf.

Amazingly, a simple and obvious-seeming generalization of this idea works in much more general situations where the tree is not binary or even fixed degree and the depth is not uniform. A "sample" is a traversal of any path of the tree, stopping when a leaf is reached. The dashed lines in Figure 1 are an example of such a sample. At each step k choose at random among the n_k children of the current node and record n_k. After traversing a path, the estimate obtained for the number of nodes is given by:

$$c_{tot} = n_0(1 + n_1(1 + n_2(1 + \ldots)))$$

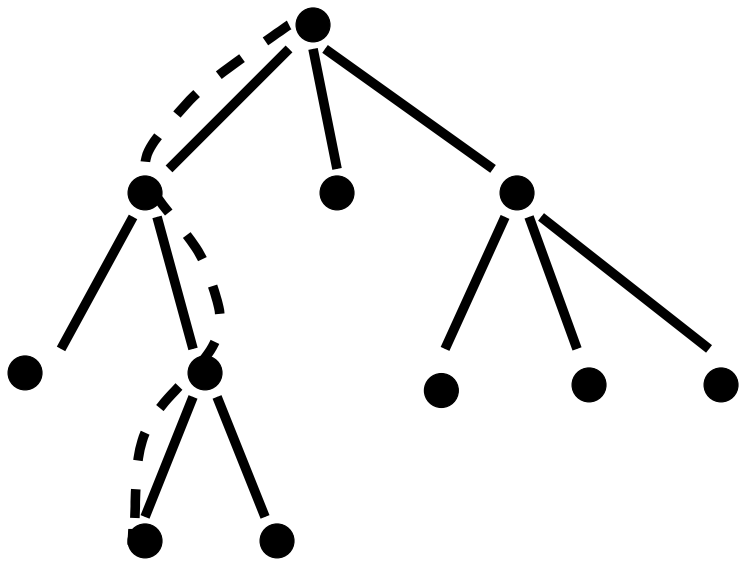

Figure 1: The Knuth method selects one path and estimates the number of nodes in the whole tree.

Averaging values of c_{tot} over sufficiently many samples gives the estimate. Note that this estimates the number of nodes in the tree, a quantity *we did not know before getting the estimate*!

Knuth's method can be thought of as an application of importance sampling. For one sample, the number of possible choices for the first k levels of the tree is $c_{k-1} = n_0 n_1 \dots n_{k-1}$ and, because of the uniform choice at each level, the probability of making that particular sequence of choices is $1/c_n = 1/(n_0 n_1 \dots n_{k-1})$. If the choices were made using some other, non-uniform probabilities p_i, then the estimate for number of leaves at the k^{th} step would be $c_{k-1} = 1/(p_0 p_1 \dots p_{k-1})$

The novel idea here is the use of probabilities that are determined sequentially at each level of the tree as the computation proceeds. In other words,

the importance sampling is done sequentially, hence sequential importance sampling, now called SIS. As was mentioned earlier, when SIS works, it works amazingly well. Naturally, the better the 'invented' probability p, the better the result. In fact, one can show that the whole role of p is to reduce the variance.

The idea can be used in a variety of situations where one wishes to estimate a hard-to-compute sum. For example, one standard algorithm for topological sorting of an acyclic directed graph looks for vertices having no predecessors, eliminates them and the edges connected to them producing more vertices having no predecessors, etc. Counting the number of possible topological sort orders of an acyclic graph is known to be an NP-hard problem. But obtaining an approximate count is easy via SIS. At each stage, simply record the number of having no predecessors.

5. Discussion

Call the sum we wish to approximate A, and the approximation $\hat{A}$. Then because the sample $1/p$ is chosen with probability p, we have that:

$$var = \langle 1/p^2 \rangle - \langle 1/p \rangle^2 = \sum_a 1/p_a - \hat{A}^2$$

From this it is clear that the critical question is the size of the difference:

$$\frac{\langle 1/p \rangle}{\hat{A}} - 1$$

where this time the average $\langle . \rangle$ is uniform instead of the p-weighted mean.

If one could design an algorithm so that p actually *is* the true probability $1/A$, then the result would be perfect (and would require only one sample!). Naturally, this does not happen in real-world applications. One interesting question is how to understand the trade-off between amount of effort devoted to generating p on the one hand, and, on the other, the number of samples needed. We'll give one example from our own research.

In our work on approximating the permanent of a zero-one matrix we used a technique called Sinkhorn balancing [8], [1] to get a better p. This increases the work per sample and decreases the number of samples needed because it lowers the variance. Recall that for an $n \times n$ matrix $\mathcal{A}$, the permanent is like

the determinant, except there are no sign changes, i.e. it is defined to be the sum over S_n the set of all permutations on n letters:

$$per(\mathcal{A}) = \sum_{\sigma \in \mathcal{S}_\backslash} \prod_{\rangle} \dashv_{\rangle \sigma(\rangle)}.$$

For a zero-one matrix, each non-zero term in the sum is equal to one, so to estimate the permanent means to estimate the number of permutations σ such that $\prod_i a_{i\sigma(i)} = 1$. We can do this by choosing a non-zero from row one with some probability $p(1)$, eliminating row one and the appropriate column to give an $(n-1) \times (n-1)$ matrix and then repeat the process on this smaller matrix. The final probability for a sample is $p(1)p(2) \ldots p(n)$.

In the problem of interest, the matrix $\mathcal{A}$ is a zero-one matrix and Sinkhorn balancing is used to generate a probability for choosing elements row by row. The idea is to generate diagonal matrices $\mathcal{D}$ and $\mathcal{E}$ so that $\mathcal{B} = \mathcal{D}\mathcal{A}\mathcal{E}$ is a *doubly stochastic* matrix - every row and column sum to one, so that the entries of a row are probabilities. The balanced matrix $\mathcal{B}$ is generated by first dividing each row by its sum, then each column, then each row, etc. This is a simple algorithm, but the theory is deep. It's not obvious it converges and if it does converge, how long it takes. There are cases in which it converges and generates $\mathcal{B}$ but doesn't produce diagonal matrices $\mathcal{D}$ and $\mathcal{E}$. Not every pattern of zeros and ones has a corresponding $\mathcal{B}$. The matrix

$$\mathcal{A} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

has no corresponding B with the same pattern of non-zeros. However, Sinkhorn balancing will converge (slowly) to the identity matrix.

Recent work by Chen and Liu [6] uses a much simpler method to generate p, so that sampling is quicker but more samples are required. In effect, one iteration of Sinkhorn balancing is done and convergence is not demanded. In computational experiments we find that Sinkhorn balancing is worth doing - the total running time of codes is lower than the method of Chen and Liu. On examples tested, Sinkhorn balancing takes about half the time of the Chen-Liu method and achieves a standard deviation a factor of more than 40 lower. However, in his recent book, Liu reports exactly the opposite - the simpler method is claimed to be better than the balancing method. The explanation

of the difference comes from an important but subtle point about Sinkhorn balancing. In our application Sinkhorn balancing is used to generate a doubly stochastic matrix from a zero-one matrix. Balancing is an iterative method that converges quadratically [9] *assuming* that the so-called "unsupported" elements are removed from the input matrix. The unsupported elements are those that would converge to zero under Sinkhorn balancing. They are easy to detect using some ideas from graph theory[4]. But if the unsupported elements are not removed, convergence is sub-linear. In Figure 2 we compare simple SIS with Sinkhorn balancing for a 20×20 Fibonacci matrix, i.e. a tridiagonal zero-one matrix whose permanent is the 21st Fibonacci number. We show running average vs error, meaning difference from the true mean.

The results for Sinkhorn balancing look much better and the variance is much smaller. Of course it is possible that there is an important and subtle aspect of the Chen-Liu method that we have overlooked. From this it is obvious that a set of rigorous and usable mathematical tools for evaluating SIS algorithms is needed.

A final note on the history of SIS. The standard Metropolis algorithm is usually traced to the 1953 paper by Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller. Of course, the idea is older and, like all really important ideas in science, one can argue about its true origin. Interestingly, one of the earliest papers on SIS is by Rosenbluth and Rosenbluth [7] and dates back to 1955.

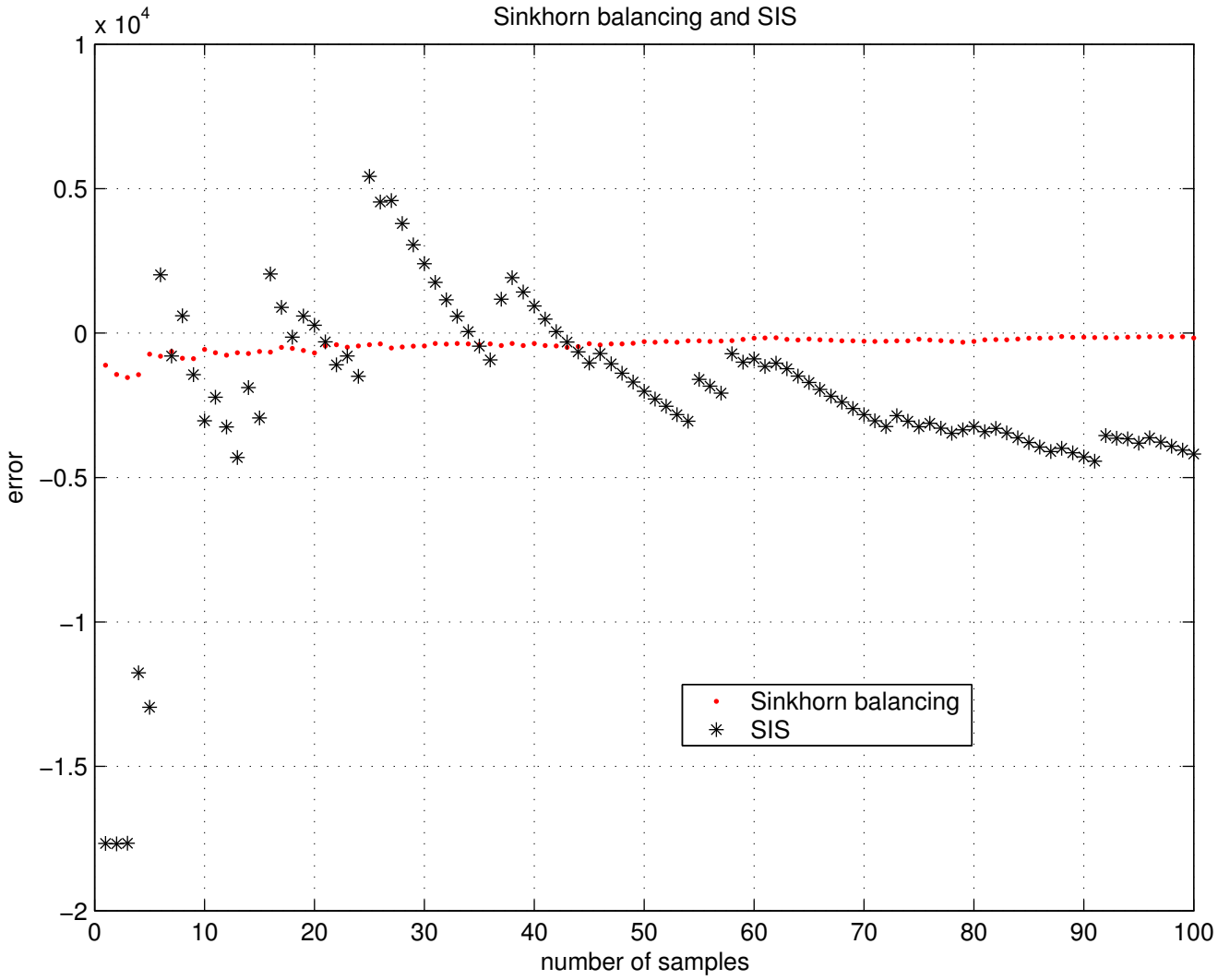

Figure 2: Comparison between simple SIS and Sinkhorn balancing

References

[1] Ando, T. "Majorization, Doubly Stochastic Matrices and Comparison of Eigenvalues" *Linear Algebra & Applic.* **118**, 163 (1989).

[2] Beichl, I. and Sullivan, F. "Approximating the Permanent via Importance Sampling with Applications to the Dimer Covering Problem," J. Comp. Phys. **149**, (1999) pp. 128-147.

[3] Beichl, I. and Sullivan F., "It's bound to be right", IEEE Computing in Science & Engineering, **4** (2002) no. 2, pp 86-89.

[4] Dulmage, A. and Mendelsohn, N. "Coverings of bipartite graphs", Can. J. Math. **10** (1958) pp. 517-534.

[5] Knuth, Donald E., "Estimating the Efficiency of Backtrack Programs", *Selected Papers on Analysis of Algorithms*, CSLI Publications, Stanford, California, (2000).

[6] Liu, Jun S., *Monte Carlo Strategies in Scientific Computing* (2001) Springer Series in Statistics, Springer Verlag.

[7] Rosenbluth, M. and Rosenbluth, A. "Monte Carlo calculation of the average extension of molecular chains", *Journal of Chemical Physics* **23**, (1955) pp. 356-359.

[8] Sinkhorn, R., "A relationship between arbitrary positive matrices and double stochastic matrices" *Annals of Math. Stat.* **35**, (1964) pp. 876-879.

[9] Soules, G. W., "The rate of convergence of Sinkhorn balancing" *Linear Algebra & Applic.* **150**, (1991) pp. 3-40.