THE METROPOLIS ALGORITHM

The Metropolis Algorithm has been the most successful and influential of all the members of the computational species that used to be called the "Monte Carlo Method." Today, topics related to this algorithm constitute an entire field of computational science supported by a deep theory and having applications ranging from physical simulations to the foundations of computational complexity.

The story goes that Stan Ulam was in a Los Angeles hospital recuperating and, to stave off boredom, he tried computing the probability of getting a "perfect" solitaire hand. Before long, he hit on the idea of using random sampling: Choose a solitaire hand at random. If it is perfect, let *count* = *count* + 1; if not, let *count* = *count*. After M samples, take *count*/M as the probability. The hard part, of course, is deciding how to generate a *uniform random* hand. What's the probability distribution to draw from, and what's the algorithm for drawing a hand?

Somewhat later, John von Neumann provided part of the answer, but in a different context. He introduced the *rejection algorithm* for simulating neutron transport. In brief, if you want to sample from some specific probability distribution, simply sample from any distribution you have handy, but keep only the good samples. (Von Neumann discussed his approach in a letter to Bob Richtmeyer [11 Mar. 1947] and in a later letter to Ulam [21 May 1947]. Interestingly, the letter to Richtmeyer contains a fairly detailed program for the Eniac, while the one to Ulam gives an explanation augmented by what we would today call pseudocode.)

Since the rejection method's invention, it has been developed extensively and applied in a wide variety of settings. The Metropolis Algorithm can be formulated as an instance of the rejection method used for generating steps in a Markov chain. This is the approach we will take.

1521-9615/00/$10.00 © 2000 IEEE

ISABEL BEICHL
NIST
FRANCIS SULLIVAN
IDA/Center for Computing Science

The rejection algorithm

First, let's look at the setup for the rejection algorithm itself. We want to sample from a population (for example, solitaire hands or neutron trajectories) according to some probability distribution function, ν, that is known in theory but hard to sample from in practice. However, we can easily sample from some related probability distribution function μ.

So we do this:

1. Use μ to select a sample, x.
2. Evaluate $\nu(x)$. This should be easy, once we have x.

3. Generate a uniform random $\rho \in [0,1)$
   if $\rho < c\nu(x)/\mu(x)$
   then accept x
   else try again with another x
   Here we choose c so that $c\nu(x)/\mu(x) < 1$ for all x.

First, the probability of selecting and then accepting some x is

$$c\frac{\nu(x)}{\mu(x)}\mu(x) = c\nu(x).$$

Also, if we are collecting samples to estimate the *weighted mean*, $S(f)$, of some function $f(x)$—that is, $S(f) = \Sigma f(x)c\nu(x)$—we could merely select some large number, M, of samples by using $\mu(x)$, reject none of them, and then compute the *uniform mean*:

$$\frac{1}{M}\sum f(x)c\frac{\nu(x)}{\mu(x)}.$$

That is, if we don't reject, the ratios give us a sample whose mean converges to the mean for the limiting probability distribution function ν. This method for estimating a sum is an instance of *importance sampling*, because it attempts to choose samples according to their importance. The ideal importance function is

$$\mu(x) = \frac{f(x)\nu(x)}{\sum_y f(y)\nu(y)}.$$

The alert reader will have noticed that this $\mu(x)$ requires knowledge of the answer. However, importance sampling works for a less-than-perfect $\mu(x)$. This is because the fraction of the samples that equal any particular x will converge to $\mu(x)$, so the sample mean of $f(x)c\nu(x)/\mu(x)$ will converge to the true mean. For the special case of the constant function, $f(x) = 1$, the quantity S is the probability of a "success" on any particular trial of the rejection method. If we take f to be the function that is identically equal to one, we might know the value of S in advance. In that case, we also know the rejection rate $1/S$, which is the average number of trials before each success. As we shall see, when we use rejection in its formulation as the Metropolis Algorithm, prior knowledge of the rejection rate leads to a more efficient method called *Monte Carlo time*.[1]

Applications: The Metropolis Algorithm

We first look at two important applications of the Metropolis Algorithm—the Ising model and simulated annealing—and then we examine the problem of counting.

The Ising model

This model is one of the most extensively studied systems in statistical physics. It was developed early in the 20th century as a model of magnetization and related phenomena. The model is a 2D or 3D regular array of spins $\sigma_i \in \{-1, 1\}$ and an associated energy $E(\sigma)$ for each configuration. A configuration is any particular choice for the spins, and each configuration has the associated energy

$$E(\sigma) = -\sum_{i,j} \mathcal{J}_{i,j}\sigma_i\sigma_j - B\sum_k \sigma_k.$$

The sum is over those $\{i, j\}$ pairs that interact (usually nearest neighbors). $\mathcal{J}_{i,j}$ is the interaction coefficient (often constant), and B is another constant related to the external magnetic field.

In most applications, we want to estimate a mean of some function $f(\sigma)$ because such quantities give us a first-principles estimate of some fundamental physical quantity. In the Ising model, the mean $\mathcal{F}$ is taken over all configurations:

$$\mathcal{F} = \frac{1}{Z(T)}\sum_\sigma f(\sigma)\exp\left(-E(\sigma)/\kappa T\right).$$

But here, the weights come from the expression for the configuration's energy. The normalizing factor $Z(T)$ is the *partition function*:

$$Z(T) = \sum_\sigma \exp\left(-E(\sigma)/\kappa T\right).$$

T is the temperature and κ is the Boltzmann constant.

A natural importance-sampling approach might be to select configurations from the distribution:

$$\mu(\sigma) = \frac{\exp\left(-E(\sigma)/\kappa T\right)}{Z(T)}$$

so that the sample mean of M samples,

$$F = \frac{\sum_k f(\sigma_k)}{M}$$

will converge rapidly to the true mean, $\mathcal{F}$.

The problem, of course, is finding a way to sample from μ. In this case, sampling from the proposed "easy" distribution μ is not so simple. Nick Metropolis and his colleagues made the following brilliant observation.[2] If we change only one spin, the change in energy, ΔE, is easy to evaluate, because only a few terms of the sum change. This observation gives a way of constructing an

aperiodic symmetric Markov chain converging to the limit distribution μ. The transition probabilities, $p_{\xi,\sigma}$, are such that for each configuration, σ,

$$\mu(\sigma) = \sum_{\xi} \mu(\xi) p_{\xi,\sigma}.$$

The sum is over all configurations ξ that differ from σ by one spin, and

$$p_{\xi,\sigma} = \frac{\mu(\sigma)}{\mu(\xi)} = \exp\left(-\left(E(\sigma) - E(\xi)\right)/\kappa T\right) = \exp\left(-\Delta E(\sigma)/\kappa T\right)$$

when $\Delta(E) > 0$, and $p_{\xi,\sigma} = 1$ when $\Delta(E) < 0$.

In other words, if the move lowers the energy, do it, and if it raises the energy, do it with some probability p, meaning reject it with some probability $1 - p$. But how to choose the site for the attempted move? We use rejection yet again. If there are n sites, we use a probability distribution that looks like

$$c\nu(\sigma) = \frac{\min\left(1, \exp\left(-\Delta E(\sigma)/\kappa T\right)\right)}{n}$$

so that we take $1/n$ as the "easy" probability. That is, we select a site uniformly and accept it according to the Metropolis criterion we just described.

For this case, the expression for the success rate is

$$S = \sum_i c\nu(\sigma_i) = \sum_i \frac{\min\left(1, \exp\left(-\Delta E(\sigma_i)/\kappa T\right)\right)}{n}.$$

So, the probability of exactly k rejections followed by a success is the same as the probability that a random ρ satisfies $(1 - S)^{k+1} < \rho < (1 - S)^k$, giving this stochastic expression for the waiting time:

$$k = \frac{\log(\rho)}{\log(1 - S)}.$$

We can use this to avoid the rejection steps while still "counting" how many rejections would have occurred.[1]

In principle, this Monte Carlo-time method works with any rejection formulation. However, each stage requires explicit knowledge of all possible next steps. In other words, we need the values for the "difficult" distribution $\nu(x)$. In the Metropolis Ising case, the Markov chain formulation makes this feasible.

Simulated annealing

Suppose we wish to maximize or minimize some real-valued function defined on a finite (but large) set. The classic example is the traveling salesman's problem. The function is the tour's length, and the set is that of possible tours. One approach is *hill climbing*. That is, given a set of possible changes to a tour, such as permuting the order of some visits, choose the change that decreases the tour length as much as possible. This approach's drawback is that it can get stuck at a local minimum, if all moves from a tour increase that tour's total length.

The Metropolis Algorithm offers a possible method for jumping out of a local minimum. Let the tour's length play the same role that energy plays in the Ising model, and assign a formal "temperature," T, to the system. Then, as long as $T > 0$, there is always a nonzero probability for increasing the tour length so that you needn't get trapped in local minima.

Three questions occur:

1. Does it work?
2. How long does it take?
3. Is it better than merely using hill climbing with many different random starts?

The answers seem to be

1. Yes. A large literature covers both the theory and applications in many different settings. However, if $T > 0$, the limit probability distribution will be nonzero for nonoptimal tours. The way around this is to decrease T as the computation proceeds. Usually, T decreases like $\log(s_k)$ for some positive, decreasing sequence of "cooling schedule" values s_k, so that the acceptance probability decreases linearly until only the true minimum is accepted.
2. It depends. Designing cooling schedules to optimize the solution time is an active research topic.
3. Someone should investigate this carefully.

Counting

Let's reconsider Ulam's original question in a slightly more general form: How many members of a population P have some specific property U? We could do the counting by designing a Markov chain that walks through P and has a limit probability distribution ν that is somehow related to our interesting property U. To be more concrete, P might be the set of partial matchings of a bipartite graph G, and U might be the set of matchings that are "perfect," meaning they include every graph node.

To have our Markov chain do what we want, we define a partition function:

$$Z(\lambda) = \sum_k m_k \lambda^k .$$

The partition function is associated with the probability distribution

$$\nu(k) = \frac{m_k \lambda^k}{Z(\lambda)} .$$

Here, m_k is the number of k-matchings, and λ^k plays a role similar to that played by $\exp(-E(\sigma)/(\kappa T))$ in the Ising problem. On each step, if the move selected is from a k-matching to a $(k + 1)$-matching, the probability of doing so is λ. Mark Jerrum and Alistair Sinclair show that the fraction of the samples that are k-matchings can be used to estimate the m_k to whatever accuracy is desired and that, for fixed accuracy, the time for doing so is a polynomial in the problem's size.[3] Physicists call estimating the m_k the *monomer-dimer problem* because having a k-matching means that k pairs have been matched as dimers and the unmatched are monomers.

The limit distribution

The Metropolis Algorithm defines a convergent Markov chain whose limit is the desired probability distribution. But what is the convergence rate? Put differently, does a bound exist on the number of Metropolis steps, τ, required before the sample is close enough to the limit distribution? In some cases, τ can be bounded by a polynomial in the problem size; in other cases, we can show that no such bound exists.

Rapid mixing

Jerrum and Sinclair have provided convergence results and applications to important combinatorial problems, such as the monomer-dimer problem.[3] To obtain their results, they look for a property they call *rapid mixing* for Markov chains. Jerrum has also proved some "slow convergence" results showing that, in some situations, Metropolis sampling does not mix rapidly and so converges too slowly to be practical.[4]

Coupling from the past

The Metropolis Algorithm and its generalizations have come to be known as the Monte Carlo Markov Chain technique (MCMC) because they simulate a Markov chain in the hope of sampling from the limit distribution. For the Ising model, this limit distribution is

$$\nu(\sigma) = \frac{\exp\left(-E(\sigma)/kT\right)}{Z(T)} .$$

The big question is, when are we at the limit distribution? That is, what is the convergence rate?

In some cases, we can sample directly from the limit distribution. Jim Propp and David Wilson developed a method for this called *coupling from the past* (CFTP).[5]

Think of a single Metropolis move as a map: $f_1 : S \rightarrow S$. For example, in the Ising model, choose some site k_1 and generate a random ρ_1 for the rejection test. Depending on the particular state σ, either $f_1(\sigma) = \sigma$ or $f_1(\sigma) = \sigma'$, where σ' differs from σ at one site.

Generally, the image of the set of all states does not cover all states—that is, $f_1[S] \subset S$. And, if we now choose a second map, f_2, we get $f_1 f_2[S] \subset f_1[S] \subset S$. Continuing in this way gives

$$F_k[S] = f_1 f_2 f_3 \ldots f_k[S] \subset f_1 f_2 f_3 \ldots f_{k-1}[S] \subset \ldots \subset S.$$

The functions are composed from the inside out; to add later maps, we must save earlier ones. Because the image is getting smaller, it might become a singleton; that is, F_k is constant. Propp and Wilson show that such a singleton will have been selected from the limit distribution. So, we have a method to sample from the true limit distribution, provided that we are willing to save all the maps and that we can tell when we have enough of them. One of several methods for telling when we have converged is to look for monotonicity. For some systems, there is an order, $<$, for states, there are bottom and top states $\{\perp < \top\}$, and the maps are order-preserving. In this case, we can

apply the iteration to both $\perp$ and $\top$ and wait for the two ends to meet. This works, for example, for some instances of the Ising model.

This method's obvious advantage is that, when such a sample can be obtained, it is "perfect." The disadvantages are that not all systems are amenable to this approach and that, when it does apply, the waiting time can be long.

Progress in MCMC has been impressive and seems to be accelerating. Problems that appeared impossible have been solved. For combinatorial counting problems, recent advances have been remarkable. However, two things should be borne in mind.

The first is a famous remark attributed to von Neumann: *Anyone using Monte Carlo is in a state of sin*. We might add that anyone using MCMC is committing an especially grievous offense. Monte Carlo is a last resort, to be used only when no exact analytic method or even finite numerical algorithm is available. And, except for CFTP, the prescription for use always contains the phrase "simulate for a while," meaning until you feel as if you're at the limit distribution. As we mentioned, for the Metropolis method, there are even systems for which convergence is provably slow. The antiferromagnetic Ising model is one such case. In some situations, *no* randomized method, including MCMC, will converge rapidly.

The second thing to bear in mind is that *MCMC is only one of many possible importance-sampling techniques*. For several cases, including the dimer cover problem, the ability to approximate the limit distribution directly results in extremely efficient and accurate importance-sampling methods that are quite different from MCMC.[6] However, a solid theory for these approaches is still almost nonexistent. CiSE

References

1. I. Beichl and F. Sullivan, "(Monte-Carlo) Time after Time," *IEEE Computational Science & Eng.*, Vol. 4, No. 3, July–Sept. 1997, pp. 91–95.
2. N. Metropolis et al., "Equation of State Calculations by Fast Computing Machines," *J. Chemical Physics*, Vol. 21, 1953, pp. 1087–1092.
3. M. Jerrum and A. Sinclair, "The Markov Chain Monte Carlo Method: An Approach to Counting and Integration," *Approximation Algorithms for NP-Hard Problems*, Dorit Hochbaum, ed., PWS (Brooks/Cole Publishing), Pacific Grove, Calif., 1996, pp. 482–520.
4. V. Gore and M. Jerrum, "The Swendson-Wang Process Does Not Always Mix Rapidly," *Proc. 29th ACM Symp. Theory of Computing*, ACM Press, New York, 1997, pp. 157–165.
5. JG. Propp and D.B. Wilson, "Exact Sampling with Coupled Markov Chains and Applications to Statistical Mechanics," *Random Structures and Algorithms*, Vol. 9, Nos. 1 & 2, 1996, pp. 223–252.
6. I. Beichl and F. Sullivan, "Approximating the Permanent via Importance Sampling with Application to the Dimer Covering Problem," *J. Computational Physics*, Vol. 149, No. 1, Feb. 1999, pp. 128–147.

Isabel Beichl is a mathematician in the Information Technology Laboratory at the National Institute of Standards and Technology. Contact her at NIST, Gaithersburg, MD 20899; isabel@cam.nist.gov.

Francis Sullivan is the associate editor-in-chief of *CiSE* and director of the Institute for Defense Analyses' Center for Computing Sciences. Contact him at the IDA/Center for Computing Sciences, Bowie, MD 20715; fran@super.org.