In a sampling problem, the objective is to produce a sample from some specified distribution. This could be one of the distributions that you learn about in probability class like Gaussian or Poisson, or something more complicated like a random image with a cow in it coming from some model.

Sampling algorithms assume access to some "pure" source of randomness like independent random coin flips or uniformly random numbers in some (possibly continuous) range. The job of the algorithm is to convert this randomness into a sample from the desired distribution.

When the distribution is supported on a few values and their probabilities are explicitly provided there are several natural algorithms. One way is to output the preimage of the cumulative distribution function evaluated at a random point U between 0 and 1 (see Figure 1). If you are bothered by infinitely precise sources of randomness there is the Knuth-Yao algorithm. For common random variables like Gaussians there are tailored samplers like the Box-Muller transform.

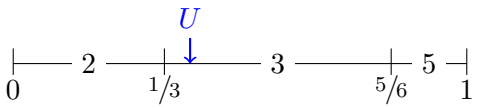


FIGURE 1: Cumulative distribution sampling for the distribution with probability mass function $p(2) = 1/3, p(3) = 1/2, p(5) = 1/6$. If $U = 0.39$ the sampler outputs 3.

1 Random walks on large graphs

Markov Chain Monte Carlo is a class of sampling algorithms for distributions over "exponentially" large sets. The ordering of a deck of cards is one example. A deck of 52 cards has 52! possible orderings. The job of the dealer in the casino is to pick one of these 52! possible orderings at random. How does he do it? By shuffling the deck.

A card shuffle is a complex process. The idea of Markov Chain Monte Carlo is to break it up into simple small independent random steps. One type of step is a random transposition: Pick two cards from the deck at random and swap (transpose) their positions. Then repeat. This is not how dealers shuffle but a computer could. Why should we expect it to end up with a random ordering?

The key to understanding a stochastic process like shuffling is good representation. Graphs come in handy. Imagine a humongous graph G with 52! vertices. Each vertex represents a configuration (ordering) of the deck. Two configurations form an edge if they differ by a swap of some two cards. The analogous graph for a three-card deck is shown in Figure 2.

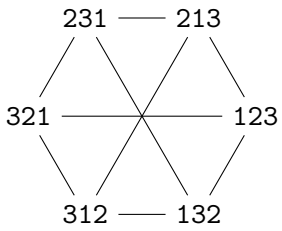


FIGURE 2: The random walk graph G for shuffling three cards using random transpositions.

Markov Chain Monte Carlo performs a *random walk* on this graph.

Algorithm *RandomWalk*
Input: A graph G represented implicitly.
1 Choose a starting vertex v of G.
2 Until you are happy,
3 Move v to a random neighbor of v.

In our example the starting vertex is the ordered deck 123. It moves to one of its three neighbors at random, and so on. After say 100 steps we would expect the algorithm to be more or less equally likely to be present at any of the six vertices. Why is that?

To answer this question we need to understand how the probability mass function of the random walk state evolves over time. Initially (at time zero) the state is 123 (with probability 1). At time 1 all three of its neighbors' are equally likely and the state has probability distribution

$$p_1(132) = 1/3 \quad p_1(213) = 1/3 \quad p_1(321) = 1/3.$$

The state in step 2 depends on its state in step 1. Conditioned on it being at 132 in step 1, all of the neighbors of 132 are equally likely:

$$T(123|132) = 1/3 \quad T(231|132) = 1/3 \quad T(312|132) = 1/3.$$

These *transition probabilities* are time-independent. A transition from state u to state v always happens with probability

$$T(v|u) = \frac{1}{\text{degree of } u}, \qquad \text{whenever } v \text{ is a neighbor of } u \tag{1}$$

because all neighbors of u are equally likely. The probability $p'(v)$ of reaching state v at the next step is calculated using the law of total probability. It is the average of the probabilities among v's neighbors.

$$p'(v) = \sum_u T(v|u) \cdot p(u) \tag{2}$$

For example, the probability of reaching state 123 in step 2 is

$$p_2(123) = \frac{1}{3} \cdot p_1(132) + \frac{1}{3} \cdot p_1(213) + \frac{1}{3} \cdot p_1(321) = \frac{1}{3}$$

On the other hand, the probability of reaching state 132 is

$$p_2(132) = \frac{1}{3} \cdot p_1(123) + \frac{1}{3} \cdot p_1(231) + \frac{1}{3} \cdot p_1(312) = 0.$$

Completing this calculation reveals that the probability distribution at step 2 is

$$p_2(123) = 1/3 \quad p_2(231) = 1/3 \quad p_2(312) = 1/3.$$

In matrix-vector notation, equation (2) becomes

$$\mathbf{p}' = T\mathbf{p}. \tag{3}$$

To figure out the distribution $\mathbf{p}_t$ after t steps we iterate (3) to obtain

$$\mathbf{p}_t = T^t \mathbf{p}_0.$$

For large t, the action of T^t is governed by the dominant right eigenvector(s) of T. (We distinguish between left and right eigenvectors because T might not be symmetric.) What are the corresponding eigenvalues? We can answer this question almost completely without knowing much about T. Because $\mathbf{p}_0$ and $\mathbf{p}_t$ are

both probability distributions, we know that T cannot grow its input. Applying it many times would result in "probabilities" that exceed 1. It cannot shrink its input either because after many steps the "probabilities" would be adding up to something close to zero. The only remaining possibility is that T has spectral norm 1.

A matrix like T whose columns are (conditional) probability distributions is called a *stochastic matrix*.[1] We just argued that

Proposition 1. *Every stochastic matrix has spectral norm* 1.

Equation (3) takes on a particularly nice form when the graph is regular, namely all vertices have the same degree d. In that case, the transition probabilities are either $1/d$ when uv is an edge, or zero when it isn't. The transition matrix equals the adjacency matrix of G scaled by $1/d$:

$$T = \frac{1}{d}G, \qquad \text{if } G \text{ is } d\text{-regular.}$$

The *uniform distribution* $\mathbf{u} = (1/N, \ldots, 1/N) = (1/N)(1, \cdots, 1)$, where N is the number of vertices, is an eigenvector of G with eigenvalue d. This means it is preserved by the transition matrix: $T\mathbf{u} = \mathbf{u}$. As $\mathbf{u}$ is associated with the top eigenvalue 1 of T we might expect the random walk distribution $\mathbf{p}_t$ to eventually approach the uniform distribution $\mathbf{u}$. In other words, the state becomes uniformly random in the long run. Is that so?

In general, no. For example, in the 3-card shuffle the distribution keeps shifting between the $\mathbf{p}_1$ and $\mathbf{p}_2$ that we calculated, never settling onto $\mathbf{u}$ (see Figure 3 (a)). The reason is that T has the spurious eigenvalue -1.

This -1 eigenvalue can be eliminated by adding loops at the vertices of G. The effect is that at each step the random walk stays where it is with some probability. These loops adjust the transition matrix to a mixture of itself and the identity matrix

$$\text{new } T = (1-\rho)T + \rho I$$

thereby ensuring no eigenvalue smaller than $-1+2\rho$ survives. This transformation preserves the stochasticity of T. Applying it to the 3-card shuffle (with $\rho = 1/4$) results in convergence to a uniform configuration (see Figure 3 (b)).

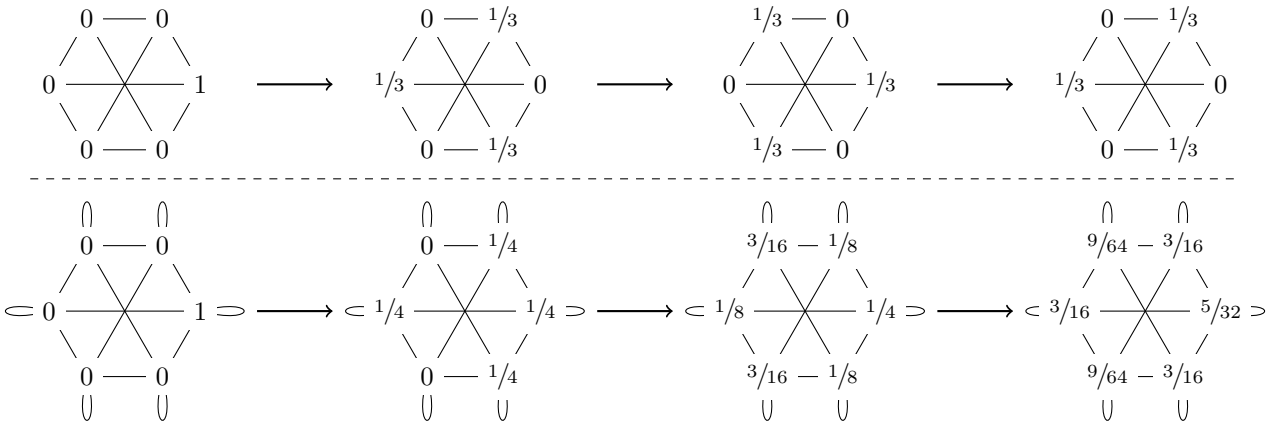


FIGURE 3: Evolution of the random walk on G (a) without loops and (b) with loops.

[1] Probability books usually work with the transpose of T, in which case the probability distributions are the rows of the matrix and equation (3) reads $\mathbf{p}' = \mathbf{p}T^\top$.

2 The Metropolis-Hastings algorithm

There are many ways to shuffle a deck of cards on the computer. Markov Chains do not have any particular advantage in this context. To demonstrate their worth we have to complicate the setup a little.

Suppose that we want the shuffle to result with not just any configuration, but one that satisfies a particular constraint. Here is one example: Consecutive cards should never have consecutive labels. In a 6-card deck this disallows configurations like `136524` because cards `5` and `6` occupy consecutive positions.

This is an example of a *conditional sampling* problem. You want to sample from a distribution conditioned on it satisfying some constraint. Conditional sampling is a common problem in causal statistics. There one starts with a model that describes how a (probabilistic) cause results in some effect, e.g., what are the chances that a person that has COVID tests positive for it. One then observes the effect and wants to figure out the conditional distribution of the cause given the effect. The ability to sample from this distribution is a good starting point.

How would we shuffle a deck of n cards but avoid consecutive labels? The idea of Markov Chain Monte Carlo is to set up and perform a random walk, but only on the subset of those valid configurations that satisfy the constraint. What is the underlying graph? The simplest way to construct it is to start with the one we already have, namely the transposition shuffle graph, and remove all the invalid vertices and their incident edges from it. Let's name this graph V_n. The graph V_5 is shown in Figure 4 (a).

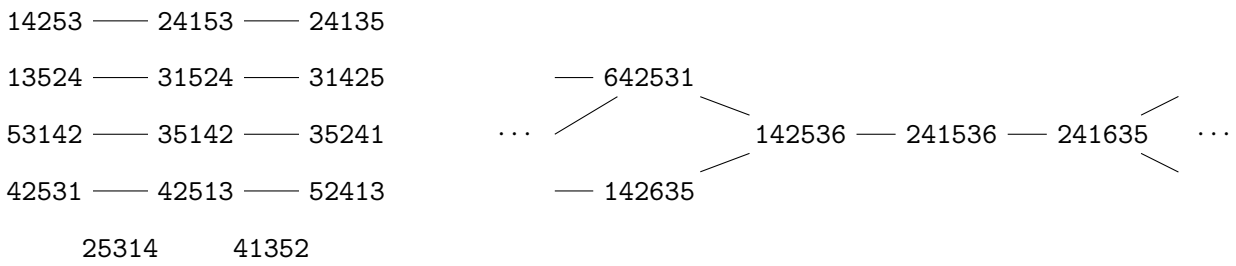


FIGURE 4: Subgraphs induced by the subset of valid configurations for (a) the 5-card shuffle and (b) the 6-card shuffle (the portion incident to vertices `142536` and `241536`).

There is a problem. V_5 is not connected. If the random walk starts at `14253` it can never reach `13524`. Even if we add in loops the distribution after many steps will end up settling on its initial connected component, resulting in a non-uniform outcome. The algebraic reason for this is the multiplicity of the eigenvalue 1 in the matrix T.

A transition matrix is *irreducible* if the underlying graph is (strongly) connected: It is possible to get from any state to any other state with nonzero probability. Reducibility turns out to be the only essential obstacle to convergence.

Theorem 2 (Perron-Frobenius Theorem)**.** *If T is stochastic, is irreducible and aperiodic then it has a unique right eigenvector with eigenvalue* 1*.*

By scaling this eigenvector we can ensure its entries add up to one. It then represents a probability distribution called the *stationary distribution*. In summary, as long as we ensure connectivity (and aperiodicity by adding loops), the Perron-Frobenius Theorem guarantees that the random walk will converge to the distribution described by the unique top right eigenvector of T.

Unlike V_5, V_6 happens to be connected. A random walk on V_6 is guaranteed to converge. Is this a good way to shuffle?

The answer is no. It is not enough that the random walk converges to a unique stationary distribution. We also have to ensure that the stationary distribution is the one we want, namely the uniform distribution over the subset of valid configurations. This is not the case for the random walk on V_6 because its vertices have uneven degrees. For example, vertex `142536` has three neighbors, but vertex `241536` has only two (see

Figure 4 (b)). If the uniform distribution (over valid configurations) is ever reached, in the next step the odds of reaching $u = \texttt{142536}$ versus $v = \texttt{241536}$ are $3:2$. The uniform distribution cannot be stationary.

Famed physicist Nicholas Metropolis came up with an ingenious solution to this problem. Suppose the random walk is at state u and is thinking of transitioning to state v. If the ratio of probabilities $T(u|v)/T(v|u)$ exceeds 1, move to v. Otherwise, move to v with probability $T(u|v)/T(v|u)$ only, and stay at u with the remaining probability $1 - T(u|v)/T(v|u)$. Altogether, the transition probabilities resulting from Metropolis's rule are

$$T'(v|u) = T(v|u) \cdot \min\{T(u|v)/T(v|u), 1\} = \min\{T(u|v), T(v|u)\} \qquad \text{whenever } u \neq v. \tag{4}$$

The transition probabilities in T' are smaller than those in T. How come they add up to 1? The difference is accounted for by the loop probabilities $T'(v|v)$.

The beauty of equation (4) is that T' is now symmetric: $T'(v|u)$ equals $T'(u|v)$. This means that not only are the columns of T' probability distributions, but so are its rows. So the entries in each row add to 1. But then the all-ones vector is an eigenvector of T' with eigenvalue 1. By Theorem 2 the uniform distribution, which is a multiple of this vector, must be the stationary one!

In summary, Metropolis's rule allows us to turn *any* connected graph over the set of valid configurations (with bidirectional edges) into a Markov Chain that converges to the uniform distribution on its vertices. The rule can be further modified to allow convergence to any desired stationary distribution $\mathbf{s}$, not necessarily the uniform one.

Algorithm MH (the Metropolis-Hastings algorithm)
Inputs: A stochastic matrix T and a target distribution $\mathbf{s}$, represented implicitly.
1 Choose a starting vertex u.
2 Until you are happy,
3 Sample a neighbor v of u with probability $T(v|u)$.
4 Calculate the ratio $\rho = \min\{s(v)T(u|v)/s(u)T(v|u), 1\}$.
5 With probability ρ, replace u by v.

The Metropolis-Hastings' rule ensures that the modified stochastic matrix satisfies the *detailed balance condition*

$$T'(v|u)s(u) = T'(u|v)s(v) \tag{5}$$

for all u and v. This in turn guarantees that $\mathbf{s}$ is the top right eigenvector of T'. By Theorem 2 it must be the unique stationary distribution of the Markov Chain.

How might we go about implementing the MH algorithm for a specific problem like sampling valid card orderings? First, we need a starting vertex. This is not a given. Coming up with a starting vertex means that we need to know at least one solution that satisfies the constraint. Such a solution can be hard to find for some problems.[2] One tactic that ensures an easy starting vertex is to replace a hard constraint like "the permutation has no consecutive numbers" with a softer one that allows such permutations but assigns a relatively small probability to them, making it unlikely that they are ever sampled. We'll come back to this idea in Section 4. For the valid card shuffling example V_6 we don't have to go there. We can pick a specific valid ordering like $\texttt{142536}$ to start with.

Next we need to implement step 3. MH asks us to sample a random neighbor of any given vertex. A common strategy for this is to keep the graph *sparse*, ensuring that the set of neighbors of each state is much smaller than the number of states. This is the case for the transposition shuffle. There are $n! = n^{\Theta(n \log n)}$ ways to order an n-card deck, but only $\binom{n}{2} = O(n^2)$ ways to transpose a pair of cards. Asymptotically, the "degree" of a configuration is polynomial in the length of the string that describes it (a permutation of n numbers). Moreover, the transitioning rule is simple to implement: Swap two numbers.

[2] One such problem is finding a preimage of a random output in a cryptographic one-way function like SHA3.

We now come to step 4. To implement it we seem to need four pieces of information: The transition probabilities $T(u|v), T(v|u)$ and the stationary probabilities $s(u), s(v)$. In a transition matrix arising from a graph all transitions out of a vertex are equally likely, and $T(v|u)$ is the reciprocal of the degree of u. If the neighborhoods are small then the degree of a vertex should be easy to calculate. For example, in the Markov Chain on V_n we can enumerate all potential $\binom{n}{2}$ neighbors of v and check how many of them represent valid configurations.

How about the stationary probabilities $s(u)$ and $s(v)$? In our example the stationary distribution is uniform over the set of valid configurations, so $s(u)$ is the reciprocal of the number of such configurations. But this number is not easy to find. Counting configurations is typically even harder than sampling a random one. Yet another amazing feature of the Metropolis-Hastings rule is that we do not need to know $s(u)$ and $s(v)$ but merely their ratio $s(u)/s(v)$. When the stationary distribution is uniform over some subset of configurations this ratio is always one! Even when it is not, there are many interesting examples in which the *likelihood ratio* $s(u)/s(v)$ is easy to find even when $s(u)$ and $s(v)$ are not. We'll see a less obvious example in Section 4.

3 Mixing times

You can now start walking the random walk. When can you expect to arrive in stationary heaven? If you are a perfectionist, never. The best you can aspire is to get sufficiently close.

The tools we learned in lectures 2 and 3 should give us a sense of the rate of convergence. For simplicity let's assume T is symmetric already and the unique stationary distribution is uniform. (The general theory is similar but the notation is more cumbersome.) The initial distribution $\mathbf{p}_0$ can be written as a sum of the stationary $\mathbf{u}$ plus some error vector $\mathbf{v}_0$ (which is not a probability distribution):

$$\mathbf{p}_0 = \mathbf{u} + \mathbf{v}_0$$

This decomposition is orthogonal: $\mathbf{u} \cdot \mathbf{v}_0 = 0$ because

$$\mathbf{u} \cdot \mathbf{p}_0 = \frac{1}{N}(\mathbf{p}_0(1) + \cdots + \mathbf{p}_0(N)) = \frac{1}{N} \cdot 1 = \frac{1}{N}(\mathbf{u}(1) + \cdots + \mathbf{u}(N)) = \mathbf{u} \cdot \mathbf{u}.$$

After the random walk makes a step, $\mathbf{p}_0$ becomes $T\mathbf{p}_0 = \mathbf{p}_1$, while $\mathbf{u}$ remains invariant: $T\mathbf{u} = \mathbf{u}$. Therefore

$$\mathbf{p}_1 - \mathbf{u} = T\mathbf{p}_0 - T\mathbf{u} = T(\mathbf{p}_0 - \mathbf{u}) = T\mathbf{v}_0.$$

As $\mathbf{v_0}$ is orthogonal to the top eigenvector $\mathbf{u}$, T shrinks $\mathbf{v}_0$ by at least the *second* largest eigenvalue λ_2 of T (in absolute value):

$$\|\mathbf{p}_1 - \mathbf{u}\| = \|T\mathbf{v}_0\| \leq |\lambda_2| \cdot \|\mathbf{v}_0\| = |\lambda_2| \cdot \|\mathbf{p}_0 - \mathbf{u}\|.$$

After t iterations,

$$\|\mathbf{p}_t - \mathbf{u}\| \leq |\lambda_2|^t \cdot \|\mathbf{p}_0 - \mathbf{u}\|. \tag{6}$$

Equation (6) tells us two things. First, the rate of convergence is determined by the *spectral gap* $1 - |\lambda_2|$. The farther $|\lambda_2|$ is from 1, the faster we get to stationary. Second, the convergence rate is exponential. Once $|\lambda_2|^t$ becomes smaller than some fixed constant, say $1/e$, then every subsequent t steps get us down by another factor of $1/e$. After running it for kt steps the state of the Markov Chain is e^{-k}-close to uniform in Euclidean distance. The "hard work" is getting $|\lambda_2|^t$ down to a constant, say $1/e$.

Definition 3. The *geometric mixing time* of a Markov Chain is the smallest value of t for which $|\lambda_2|^t$ drops below $1/e$, namely $t = -1/\ln|\lambda_2|$.

When $|\lambda_2|$ is close to 1, $-\ln|\lambda_2|$ is approximately equal to $1 - |\lambda_2|$ so the Euclidean mixing time is about the inverse of the spectral gap.

As the underlying graph is very large we have no chance of calculating the spectral gap even on a computer. For some "structured" Markov Chains like permuting with unrestricted transpositions it can be calculated, but examples like that are rather rare.

There is an appealing alternative measure called the *(stochastic) mixing time*. This is the first step t at which the random walk distribution $\mathbf{p}_t$ becomes $1/2e$-statistically indistinguishable from the stationary one. Two distributions are ϵ-statistically indistinguishable if no hypothesis can tell a sample of one versus the other with advantage more than ϵ.

Coupling is a fairly intuitive method for bounding the mixing time. The idea is to compare two executions of the Markov Chain starting from different states. A *coupling* is a joint distribution of the state sequences of both executions in which the marginal distribution of each is identical to that of the Markov Chain. If we can find a way to couple the executions in a way that they are likely to converge to the same state at time t (except with failure probability $1/2e$) then the Markov Chain is guaranteed to have mixed by time t.

It would take some time to unwrap this concept. If you are curious there are several good examples in Chapter 8 of the Markov Chain textbook by Levin and Peres. One of their examples is precisely the transposition shuffle. It is advanced but let me try to reproduce it to give you a flavor of how mixing is argued.

Proposition 4. *The mixing time of the unconstrained transposition shuffle for an n-card deck (with loops) is $O(n^2)$.*

Proof (assumes knowledge of probability). Here is a somewhat unintuitive way to implement the transposition shuffle. At each step, choose two random numbers X and Y between 1 and n. Swap the card labeled X with the card *in position* Y. For example, if the current state is `326541` and $X = 5, Y = 3$ are chosen, then the `5`-card is swapped with the card in position 3, which is `6`. The Markov Chain will transition to `325641`.

Let's try to understand what happens if two executions of the Markov Chain with different start states carry out the same sequence of transitions $X_1Y_1, X_2Y_2, \ldots$. As an example, suppose that at some point in time they have reached states $u =$ `54236`$\underline{\texttt{1}}$ and $v =$ `61435`$\underline{\texttt{2}}$. The common transition is $X =$ `4` (the red card) and $Y = 5$ (the underlined card). After the swap the two cards in position Y (the underlined position) are guaranteed to match: They both equal X. In this example the move will create a new match (in position 5) without destroying any old ones (such as the one in position 4). Indeed, the new states $u' =$ `562341` and $v' =$ `615342` now match in both position 4 and position 5. The common transition produced an extra match.

An extra match does not always arise: If the X-card happened to already match in u and v the transition would not have earned us any extra matches. Neither would it have if the cards at position Y happened to be identical. In either case, however, the number of matches would not decrease (even though the set of matching cards might change). The additional match is created whenever the X-cards do not match and the cards in position Y are different. If there are m existing matches between u and v, the probability of this event is $(n-m)^2/n^2$ because X and Y are random and independent.

In summary, after the mth match has been created, each random walk step yields the next match with probability $p = (n-m)^2/n^2$. The number of steps that it takes to obtain this match is a geometric random variable with mean p. Its average value is $1/p = n^2/(n-m)^2$. The coupled Markov Chains have take an average of $n^2/(n-m)^2$ steps to produce the extra match.

In the worst case, the initial states of the two Markov Chains will have zero matches. It will take a single step to go to one match, an average of $n^2/(n-1)^2$ steps to go to two matches, and so on. The total average number of steps to go all the way up to n matches is the sum of these averages

$$1 + \frac{n^2}{(n-1)^2} + \frac{n^2}{(n-2)^2} + \cdots + \frac{n^2}{1^2} = n^2\Big(1 + \frac{1}{2^2} + \cdots + \frac{1}{n^2}\Big) = O(n^2).$$

Markov's inequality tells us that $O(n^2)$ steps are also sufficient for the states to coalesce with probability $1/2e$ (or any other constant). □

Even if you didn't follow this argument two points should stand out. First, the mixing time is polynomial in n while the number of states is exponential. For a theorist polynomial-time algorithms indicate efficiency. Once we know that polynomial mixing time is within reach we can try to optimize further. In this example the actual mixing time turns out to be not quadratic but almost linear—$\Theta(n \log n)$ to be precise.

The second lesson is that even for a Markov Chain as simple as shuffling cards by transposition bounding the mixing time takes some ingenuity. Bounding mixing times is a popular sport for mathematicians, statisticians, physicists, and computer scientists of all stripes. This endeavor keeps some employed and others entertained, but the downside is that there is no truly systematic method for telling when *your* Markov Chain will mix. I have no idea, for example, what the mixing time of the constrained transposition shuffle is. (This would have made for an amazing final project!)

Bounding mixing times is more than just an intellectual exercise. Suppose you have designed a Markov Chain for your favorite distribution and have been running it for a while. How do you know when to stop? In the end all the chain produces is some configuration. How can you tell whether it is truly random? Randomness is notoriously difficult to test, especially for complex distributions like the ones we need Markov Chains to sample from in the first place.

4 The Ising model

The Ising model is a telling example that illustrates both the power and the challenges of Markov Chain Monte Carlo. It is a statistical model of a ferromagnet. There are n particles occupying the vertices of some interaction graph. Each particle can be have positive ($x_i = \texttt{+}$) or negative ($x_i = \texttt{-}$) *spin*. The *energy* $H(x_1, \ldots, x_n)$ of the configuration is calculated by taking multiplying the spins along the edges and adding everything up:

$$H(x_1, \ldots, x_n) = - \sum_{ij \text{ is an edge}} x_i x_j.$$

For example, if the interaction graph is the cycle C_n with the vertices $1, \ldots, n$ in order, the energy is $-(x_1x_2 + x_2x_3 + \cdots + x_nx_1)$.

The effect of the minus sign is that configurations in which the spins align have lower energy. We would expect a ferromagnet to favor such alignment. There are two configurations of lowest energy, the ones with all-$+$ and all-$-$ spins, respectively. We will denote them by $\oplus$ and $\ominus$. Statistical physics predicts that what we will typically see is not necessarily one of the lowest energy configurations, but that the low-energy configurations are more likely than the others.

In the *Gibbs distribution* each of the 2^n configurations $\{\texttt{+}, \texttt{-}\}^n$ may occur with some probability. The probability of configuration $\mathbf{x} = (x_1, \ldots, x_n)$ is

$$g(\mathbf{x}) = \frac{e^{-\beta H(\mathbf{x})}}{Z}. \tag{7}$$

Z is a proportionality constant (called the partition function) that makes the probabilities add up to 1, namely $Z = \sum_{\mathbf{x}} e^{-\beta H(\mathbf{x})}$. The number $\beta > 0$ is a parameter called the *inverse temperature*. As β approaches zero (infinite temperature), the numerator approaches 1 and the Gibbs distribution becomes uniform over all possible spins. The ferromagnet does not favor one state over another. When β approaches infinity (zero temperature), the effect of the exponent is that the lowest energy configurations tower above all others. The Gibbs distribution becomes uniform over $\oplus$ and $\ominus$.

For other values of β, any of the 2^n configurations is a possible outcome, but their probabilities are different. How different? To get a sense let's calculate it for the 3-cycle C_3. There the extreme configurations $\oplus = \texttt{+++}$ and $\ominus = \texttt{---}$ have energy -3 and the remaining six all have energy 1. The Gibbs distribution is

$$g(\oplus) = g(\ominus) = \frac{e^{3\beta}}{Z} \qquad g(\text{any other}) = \frac{e^{-\beta}}{Z}.$$

The numbers are not important. The extreme configurations are more likely than the others, with β controlling the difference in probabilities.

Physicists are interested in what happens when n is large. In a typical configuration do the spins tend to align or balance out? Apart from some special cases direct calculations are difficult. An appealing alternative is to look at random samples from the Gibbs distribution. This is where Markov Chain Monte Carlo comes in handy.

It is not too difficult to set up a Markov Chain for the Ising model via Metropolis-Hastings. For Ising-like models there is an alternative that describes the physical process of reaching steady state more faithfully.

Glauber dynamics

Algorithm *Glauber* (Glauber dynamics for the Ising model)
1 Choose an initial configuration $\mathbf{x}$.
2 Until you are happy,
3 Choose a random index i.
4 Erase the spin x_i.
5 Resample x_i from the conditional Gibbs distribution (7) given all spins except x_i.

Glauber dynamics is set up to satisfy the detailed balance condition (5). Here is why. For concreteness suppose $i = 1$ and we are considering a transition from state $\mathbf{x} = \texttt{-a}$ to state $\mathbf{x}' = \texttt{+a}$. The transition probability $T(\mathbf{x}'|\mathbf{x})$ is the conditional probability $g(\texttt{+a}|\texttt{?a})$, so the left-hand side of (5) is

$$T(\mathbf{x}'|\mathbf{x})g(\mathbf{x}) = g(\texttt{+a}|\texttt{?a})g(\texttt{-a}) = g(\texttt{+a}|\texttt{?a})g(\texttt{-a}|\texttt{?a})g(\texttt{?a}).$$

Here $\texttt{?a}$ is the event that spins 2 to n equal $\mathbf{a}$, and $g(\texttt{?a}) = g(\texttt{+a}) + g(\texttt{-a})$ is its marginal probability. By symmetry $T(\mathbf{x}|\mathbf{x}')g(\mathbf{x}')$ is exactly the same.

To implement Glauber dynamics we must figure out how to sample from the conditional Gibbs distribution given all but one spin. That is where the Gibbs distribution shows its worth. To be specific, suppose the interaction graph is C_6 and the current configuration is $\texttt{++-+-+}$. Suppose *Glauber* picks $i = 4$ in step 3. In step 4 it erases $x_4 = \texttt{+}$ to obtain the partial configuration $\mathbf{a} = \texttt{++-?-+}$.

To pick the next move we need the conditional probabilities of $x_4 = \texttt{+}$ and $x_4 = \texttt{-}$ given $\mathbf{x}$ is of the form $\texttt{++-?-+}$. The energies $-(x_1x_2 + x_2x_3 + x_3x_4 + x_4x_5 + x_5x_6 + x_6x_1)$ of the two candidate configurations $\texttt{+a} = \texttt{++-+-+}$ and $\texttt{-a} = \texttt{++--+}$ for the next state differ only in the red terms indexed by the two edges incident to x_4. Specifically,

$$H(x_1x_2x_3\texttt{+}x_5x_6) - H(x_1x_2x_3\texttt{-}x_5x_6) = -(x_3 + x_5) + (-x_3 - x_5) = -2(x_3 + x_5).$$

In particular,

$$H(\texttt{+a}) - H(\texttt{-a}) = H(\texttt{++-+-+}) - H(\texttt{++--+}) = -2 \cdot ((-1) + (-1)) = 4. \tag{8}$$

We now have everything we need to find the conditional probabilities $g(\texttt{+a}|\texttt{?a})$ and $g(\texttt{-a}|\texttt{?a})$. Their ratio equals the ratio of the unconditional probabilities $g(\texttt{+a})/g(\texttt{-a})$. By the Gibbs formula (7) this equals $e^{-\beta(H(\texttt{+a})-H(\texttt{-a}))}$, and by (8) this is $e^{-4\beta}$. Thus $g(\texttt{+a}|\texttt{?a})/g(\texttt{-a}|\texttt{?a}) = e^{-4\beta}$, but also $g(\texttt{+a}|\texttt{?a}) + g(\texttt{-a}|\texttt{?a}) = 1$ because probablilities must add up to 1. We solve these equations and get

$$g(\texttt{+a}|\texttt{?a}) = \frac{1}{1 + e^{4\beta}} \qquad g(\texttt{-a}|\texttt{?a}) = \frac{1}{1 + e^{-4\beta}}.$$

This transition is illustrated in Figure 5. The general update rule for the spin of i is

$$x_i' = b \text{ with probability } \frac{1}{1 + e^{bs\beta}}, \text{ where } s \text{ is the total spin of the neighbors of } i.$$

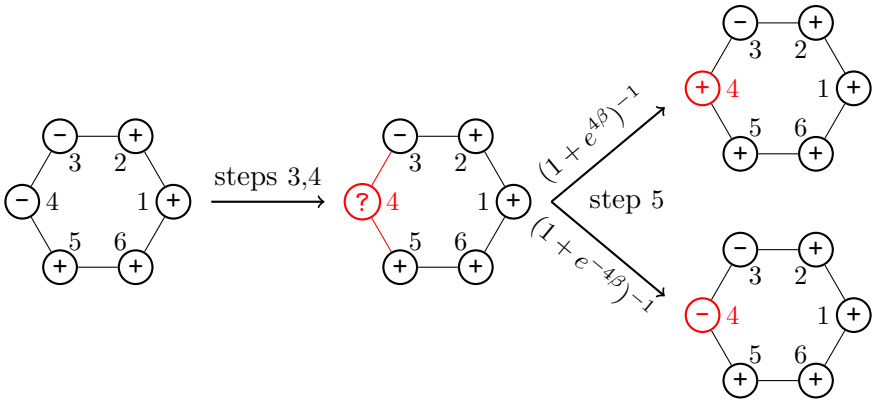


FIGURE 5: A transition of the Glauber dynamics on interaction graph C_6.

There is an important point about this formula. The only relevant information is the difference in energies (8). This number, in turn, is extremely easy to calculate. It only depends on the spins of the *neighbors* of x_i. The reason is that x_i is conditionally independent of the rest of the state given the spins at its neighbors. Probability distributions of this type are called *graphical models*. The complexity of computing a transition in a graphical model of degree d can be at most 2^d, a far cry from 2^n. The complexity is linear in d for the Ising model.

Phase transitions

Mixing visualizes nicely in the Glauber dynamics for the Ising model. In Figure 6 I plotted the state obtained by running 1000 steps of Glauber dynamics on the 16×16 grid. The vertices of the interaction graph are pairs of integers (x, y) modulo 16 and the edges are those pairs whose x or y values (not both) differ by one. The initial state is the extreme $\ominus$ configuration. At low temperature there is hardly any movement. Only a handful of spins turn positive. As the temperature increases regions of + spins start forming, but the − spins still dominate. In the high temperature regime the + and − spins roughly balance. The average spin is close to zero.

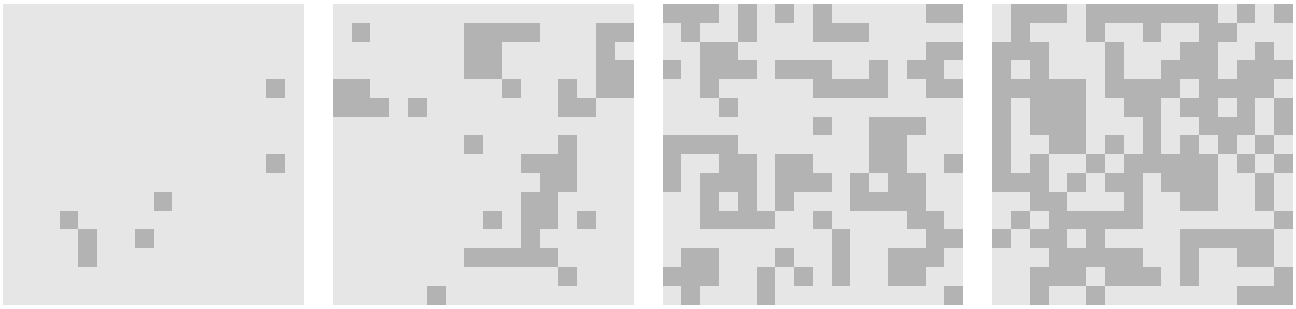


FIGURE 6: State of Glauber dynamics on the 16×16 interaction grid at step 1000 at inverse temperatures (a) 1 (b) 0.7 (c) 0.3 (d) 0.05. Light and dark squares represent − and + spins. The initial configuration is $\ominus$.

The Ising model on the grid exhibits a *phase transition*: There is a specific value of β^* (around 0.44) at which the average spin drops to zero (for large n). This is presumably the temperature at which the system demagnetizes. Interestingly, the Markov Chain starts mixing rapidly at exactly the same β^*.

What does this physics mumbo-jumbo have to do with Great Algorithms? Energy is nothing but the loss function. You can imagine cooking up a loss function over songs that favors the type of music you like. A sample from the Gibbs distribution would be the next item on your playlist.