Study guide

The Metropolis-Hastings algorithm is a generic simulation that produces results that follow any probability distribution we want. With the Ising model simulation, we saw it was difficult to run the simulation long enough to produce uncorrelated samples from the simulation at a particular temperature. Metropolis-Hastings gets us a step closer to sampling from the distribution over the state space of the Ising model — as well as any other probability distribution we can think of. In this session, we describe the Metropolis-Hastings algorithm and apply it to some simple problems. In the next session, we apply the algorithm to the Ising model.

In order to be prepared for class, you need to be able to do the following.

- Understand and explain why the Metropolis-Hastings algorithm is a simulation. Think carefully about the definition of simulation and how that definition relates to the algorithm.
- Generate samples from a 1-dimensional probability density function, using a Gaussian proposal distribution (see the pre-class work).

Shonkwiler & Mendivil notes

- "The Metropolis algorithm is at the heart of most MCMC sampling methods. The basic idea behind MCMC is to construct a Markov chain whose invariant distribution is the desired sampling distribution."
- The basic setup here is that we have a probability distribution from which we would like to draw samples, called $f(x)$ in the text. In practice, $f(x)$ is a complicated function and it is difficult to generate samples from it directly.
    - We generate samples from $f(x)$ by creating a simulation.
    - Start in an arbitrary state, x_0.
    - At each step, to move from x_t to x_{t+1} —
        - Sample from a much simpler distribution (for example a normal distribution centered on the current x-value), to propose a new state, y. Note that this is not necessarily the next state — we will either accept or reject this proposal.
        - Accept the proposed state with probability
          $$\min(1,\ f(y) / f(x_t))$$
          Remember that $f(x)$ is the target density — i.e., the distribution from which we want to generate samples. So we use the target density to determine which proposed state changes get accepted and which get rejected. If the proposed state has a greater probability than the current state we always accept, otherwise we accept with the ratio of the proposed and current state probabilities.
        - If we accept $x_{t+1} = y$, otherwise if we reject $x_{t+1} = x_t$.
    - That's it. Run this simulation for a large number of steps and the final state will be a sample from the target distribution, $f(x)$.
    - Repeat the simulation to get another sample from $f(x)$.
- Remember we saw that the state of a Markov chain tends towards a sample from a particular probability distribution (or probability vector), known as the invariant distribution of the Markov chain. The Metropolis-Hastings algorithm lets us design a Markov chain such that its invariant distribution is any distribution we choose. This is a really powerful idea. If we want to generate random samples from some really complicated probability distribution, we can use the Metropolis algorithm to create a simulation of which the invariant distribution is our desired probability distribution, and as a consequence, the long-term state of that simulation will be a random sample from the desired distribution.
- Since this is a Markov chain, both the proposal and acceptance processes may depend on the current state, but not on any additional states before the current state.

Example

Look at this demo of running the Metropolis-Hastings method on a 2-dimensional distribution called the banana distribution. Check that you understand how the steps of the Metropolis-Hastings algorithm play out in this demo.

- The banana-shaped distribution is the target distribution, $f(x)$.
- The proposal distribution, $g(x_{t+1} \mid x_t)$, is a circular Gaussian (normal) distribution with mean set to the current state.
- The Gaussian is used to generate a proposal, which gets accepted (green arrow) or rejected (red arrow).
- The resulting samples (dots) tend to follow the target distribution.
- Note that when the current state has a high probability according to the target distribution, there might be many rejections before a new state is accepted — since the new state usually has a lower probability than the current state.
- When the current state is in a low-probability region of the target distribution, the next state almost always gets accepted.

Proposal distributions

There isn't just one Metropolis-Hastings algorithm for solving a particular problem, because we need to choose or design the proposal function. The same problem can be solved using multiple proposal functions. The main (and very important) difference between them is their efficiency.

- Some proposal functions result in highly correlated samples so that we need many steps before we can reasonably assume that we have uncorrelated samples from the target distribution. Proposal distributions that take larger steps — that move far away from the current state — are better for this.
- Some proposal functions result in very small acceptance rates. If we do not accept a proposed new state we stay in the current state, again resulting in highly correlated samples. This tends to be a problem with proposal functions that take steps that are too large — that try to move really far away from the current state — since it is usually unlikely that the new state will have a higher probability than the current state.
- There are some really clever, problem-specific proposal functions that allow us to take large steps with high acceptance probability. We look at one such function for the Ising model in the next session.

If you did CS146, there are more sampler demos showing the Hamiltonian Monte Carlo and No-U-Turn samplers. These samplers are much more advanced and efficient than the standard Metropolis-Hastings method, which means they produce less correlated samples using fewer steps than the basic implementation above, but they are still examples of the Metropolis-Hastings algorithm — just using better proposal distributions.