The Metropolis Algorithm

Bayesian Data Analysis

Steve Buyske

Markov Chain Monte Carlo (MCMC)

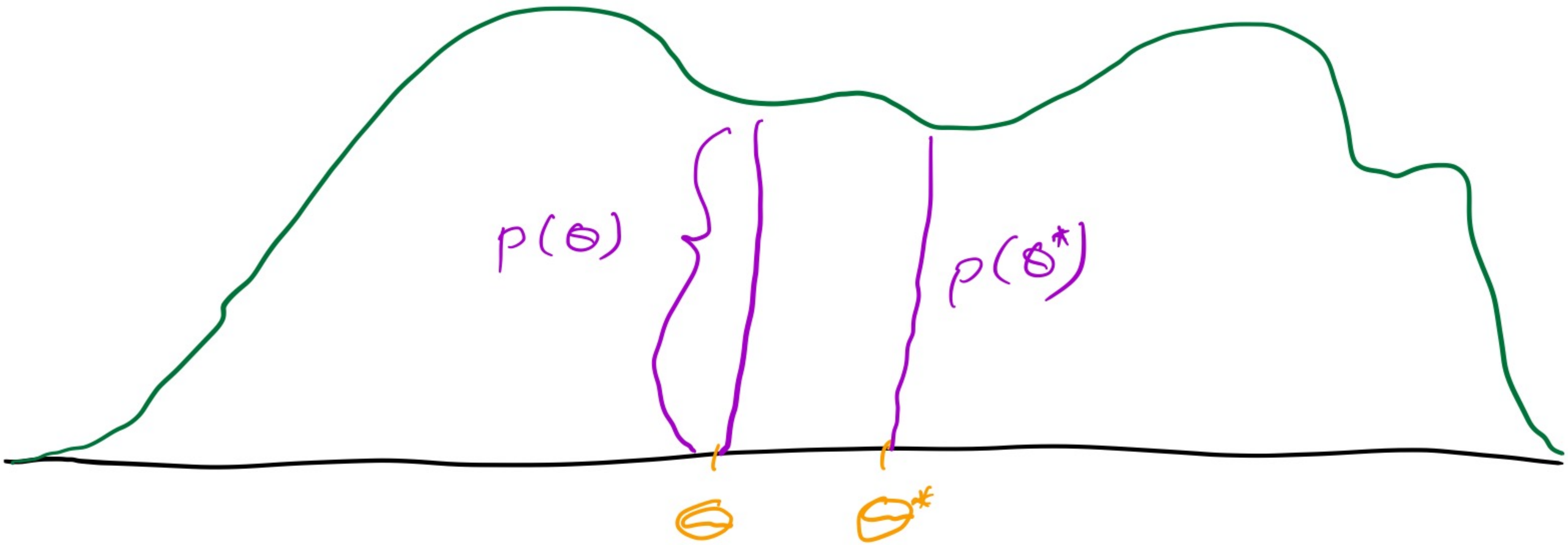
- To do Bayesian statistics, we need to be able to calculate the posterior distribution.
- How?
  - Analytic methods are too limiting
  - Quadratic approximation is inaccurate away from the mode.
  - Grid methods are impractical as the number of parameters goes up.
  - Markov Chain Monte Carlo to the rescue!
    - Idea: use random numbers in a clever way to generate a sample that looks like it came from the posterior.
    - Offers an efficient way to estimate a posterior that's too complicated to determine exactly.

The Metropolis Algorithm (aka the Metropolis Random Walk Algorithm)

- Often regarded as one of the top 10 algorithms of the 20th Century
- Although the original work was in 1953 (Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E), statisticians didn't really notice until the 1990s.
- It applies to any distribution—it doesn't have to be a posterior—and we only need a function proportional to the distribution.

The Metropolis Algorithm (aka the Metropolis Random Walk Algorithm)

- Often regarded as one of the top 10 algorithms of the 20th Century
- Although the original work was in 1953 (Metropolis N, Rosenbluth AW, Rosenbluth MN, Teller AH, Teller E), statisticians didn't really notice until the 1990s.
- It applies to any distribution—it doesn't have to be a posterior—and we only need a function proportional to the distribution.
- The basic idea:
  - From a point θ, propose—in a random way—a new point θ^*.
  - In a random way, based on the ratio $p(\theta^*) \Big/ p(\theta)$, either accept or reject the new point. If you accept it, change the label of θ^* to just θ.
  - Repeat thousands of times.



p(θ)
p(θ*)
θ
θ*

Metropolis Algorithm Details

1. Pick an initial point θ in the parameter space at random based on some distribution (typically a normal distribution).
2. Use another distribution, often called the *proposal distribution* or the *jump distribution*, to pick a new point θ^* at random.
3. Look at the ratio

$$r = \frac{\text{Prob}(\theta^* \mid \text{data})}{\text{Prob}(\theta \mid \text{data})}.$$

4. Decide whether to accept of reject the proposal

- (next slide …)

1. If we accept the proposal, record the new θ
2. Regardless, repeat from step 2.
3. The set of accepted θs will look like random draws from the original distribution.

Accept or Reject?

- There are several rules one might imagine using.
- [Don't do this] We could accept θ^* if $r \geq 1$.
  - We would always move to a higher point in the density.
- [Don't do this] We could always accept the proposal, regardless of r.
  - Then our empirical density would look like the proposal distribution, not the posterior that we are trying to find.
- [Don't do this] We could toss a coin to decide.
  - Same problem, in that we would end up getting an empirical version of the proposal distribution.

Accept or Reject? The Metropolis Way

- Instead, what we will do is accept θ^* with probability $\min(1, r)$.
- That means that when θ^* has a higher density than θ, we will always jump to the new θ, since r will be ≥ 1.
- When θ^* has a lower density than θ, though, we will jump with probability equal to the ratio of the densities.
- You can see that most of our θ's will be in high density regions, but we will also get θ's in lower density regions.

Metropolis Algorithm Notes

- Notice that nothing in the algorithm require θ to be one dimensional.
- Note that since r is a ratio, we just need to be able to compute something proportional the distribution. In particular, we don't need to calculate the denominator in Bayes' Rule.
- The sequence beginning at the original point is called a *chain*, and in practice we usually run several chains (think 4-6).

- In practice, we will typically throw away the start of the chain (usually the first half) to reduce the effect of the starting point. This is called *warm up* or *burn in*.
- We can use the behavior of the chains themselves as diagnostics, to help us judge whether the algorithm worked.
- Notice that the algorithm has a random component, that gives the *Monte Carlo* in Markov Chain Monte Carlo.
- The next step in the chain depends only on the current position—such a random sequence is called a *Markov Chain*.

More Metropolis Algorithm Notes

- Does the Metropolis algorithm converge to the right distribution (for us, $\text{Prob}(\theta| \mid \text{data})$?
- This is actually two questions: does it converge, and if so does it converge to the right distribution?
- If you are interested see Section 13.6 of the book.
- The most common proposal distribution is simply $\theta^* = \theta + \Delta\theta$, where $\Delta\theta$ is distributed as a normal, as a Student's t, or a uniform distribution.
- In this form if often known as the *Metropolis random-walk algorithm*.
- You may also come across the Metropolis Hasting algorithm. This is a version that changes the calculation of r, the jump probability, to make it asymmetric. This works better near the boundary of a distribution.