Project 3: Implementing a Metropolis walker

Technical Computation in the Earth Sciences

7th October, 2020

Introduction.1: Markov Chain Monte Carlo & the Metropolis Algorithm.

Desipte the fancy name, "Markov Chain Monte Carlo" basically just means some technique where we generate a long sequence of numbers, each of which is a random perturbation of the last one.

For this project, we will be using one particular algorithm that falls into this category, called the Metropolis algorithm, a subset of a slightly more complicated approach called the Metropolis-Hastings algorithm. This is discussed in all its fully mathey detail in the Gelman et al. (2013) Bayesian Data Analysis book, especially Chapter 11.

As the "Markov Chain" part might imply, this is an iterative approach – we don't get to the answer in one step; instead there is a procedure that we repeat many times. Breifly, this procedure is:

0. Start with some initial guess for the parameters of interest ϕ
1. Propose a variation ϕ_p by adding some random perturbation to the last ϕ. The perturbation must be symmetric (equal probability of increasing or decreasing ϕ) but otherwise can be any random number.
2. Calculate the *likelihoods* (more on that in a minute) of ϕ and ϕ_p given the data y. In math notation, that's $l(\phi|y)$ and $l(\phi_p|y)$
3. If $l(\phi_p|y) > l(\phi|y)$, *accept* the proposal ϕ_p with probability 1. Else, if $l(\phi_p|y) < l(\phi|y)$, *accept* it with probability equal to the ratio of likelihoods.

$$r = \frac{l(\phi_p|y)}{l(\phi|y)} \tag{1}$$

4. If the proposal is accepted, that means set the new ϕ equal to ϕ_p. If the proposal is rejected, ϕ_p is thrown away and ϕ stays the same. Record the current accepted ϕ (whatever that ends up being) as a step of the Markov Chain
5. Repeat steps 1-4 many many times.

Note that just as in Project 2, it will take some time for this to converge on the right answer. This is called *burn-in*. After this, the *post-burnin* parts of the chain (accepted values of ϕ) are called the *stationary distribution*, and they give us our "*posterior*" estimate for ϕ.

So what is this *likelihood*? Likelihood in a Bayesian (including MCMC) context means something that's *proportional to* a probability. Ok, great, how do we get that? While you all are probably somewhat familiar with *discrete* probability distributions (i.e., the probability distribution of a fair coin toss is 50% heads, 50% tails), for most things we'll be working with in the physical sciences we're in the realm of *continuous* probability distributions.

The continuous probability distribution we'll be working today is the *Gaussian* (aka *Normal*) distribution. The Gaussian distribution is defined by its *probability density function*

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2}$$

As an aside, the Gaussian distribution is special in something vaugely analagous to the way that the ℓ_2 norm and the arithemetic mean are special. Looking at the equation, you might even see some hint of this (notice any powers of two?). In particular, according to something called the Central Limit Theorem, combinations of other distributions tend to become Normally distributed.

Anyways, getting a probability from a *probability density function* (PDF) like Equation 2 usually means having to take an integral. For some Real number x drawn from a Normal distribution, the probability that x falls between two Real numbers a and b is equal to the integral of Equation 2 from a to b.

This has the somewhat spooky consequence that the probability of drawing any *particular* Real number from a continuous probability distribution (say 5.000000000..., or π, or etc.) is technically *zero*. Even though every time you draw from a continuous probability distribution, some particular real number is exactly what you'll get. This is at some level a consequence of the fact that the Real numbers are uncountably infinite*, and also just generally terrifying, but that's another subject.

Since we only need to worry about getting something *proportional* to the probability though, we can ignore all of that and just use the PDF directly (which is indeed proportional to the probability of observing some number in the infinitessimally small neighborhood around $x = y$). In fact, we can do even better than that, and throw out any constant terms in the PDF as well. After all, they'd only cancel when we take the ratio of the likelihoods in Equation 1, so

$$l(x|\mu,\sigma) = e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Now, because these likelihoods can be very large or very small, it actually ends up being convenient, to avoid floating-point rounding error on the computer, to actually work with *log* likelihoods. So:

$$\ell\ell(x|\mu,\sigma) = -\frac{(x-\mu)^2}{2\sigma^2} \tag{3}$$

Is this starting to remind you of anything *least-squares-ish* yet?

* *"Normal" infinity is the cardinality of the Integers, aka the number elements in the set of Integers. The cardinality of the Reals is provably larger than that. It's an entirely different kind of infinity altogether! Terrifying, I tell you.*

Introduction.2: Turning the math into code.

As you might guess, we're going to be needing a loop here. To get things started, here's a rough outline of how I might start turning the Metropolis Algorithm into actual Julia code

```
# Log-likelihood function for a Normal distribution
normpdf_ll(μ::Number,σ::Number,x::Number) = -(x-μ)*(x-μ) / (2*σ*σ)

#Set number of iterations
N = ...
lldist = Array{Float64,1}(undef, N)
ϕdist = Array{Float64,1}(undef, N) # Or possibly with more dimensions if ϕ is a vector

let # Only necessary if you're in global scope. Maybe write a function so you don't need this

    # Initial guess
    ϕ = ...
    ϕₚ = copy(ϕ)
    ll = normpdf_ll(μ, σ, ϕ) # Or something like that. Possibly a lot more complicatied, or the
sum of multiple terms

    for i=1:N
        # Prepare proposal
        ϕₚ = ϕ + ... #Some random adjustment
        ## Or if ϕ is a vector
        #copyto!(ϕₚ, ϕ) # Beware of ϕₚ = ϕ with vectors because of assignment-by-reference
        #ϕₚ += ... #Some random adjustments

        # Calculate log likelihood of proposal
        llₚ = normpdf_ll(μ, σ, ϕₚ) # Or something like that. Possibly a lot more complicatied, or
the sum of multiple terms

        # Check if proposal should be accepted
        if log(rand()) < (llₚ-ll) # Remember that subtracting the logs is equal to dividing the
likelihoods
            ll = llₚ
            ... # Probably also do something with ϕ and ϕₚ here
        end

        # Record the current accepted proposal
        lldist[i] = ll
        ϕdist[i] = ϕ # Again, possibly with more dimensions if ϕ is a vector
    end
end
```

Note again that just as in Project 2, it will take some time for this to converge on the right answer. The post-burning parts of `ϕdist` though give us our "*posterior*" estimate for ϕ. Among other things, this means that the mean of the post-burnin parts of `ϕdist` gives us or estimate for the mean of ϕ, and their standard deviation gives us our estimate for the uncertainty of ϕ.

1

Part 1 Consider a geologic scenario where you have in stratigraphic order

- A rhyolitic ash bed, with a U-Pb zircon age of 14.592 +/- 0.083 Ma, overlying...
- A critically-important fossil-bearing horizon, with an age previously approximately estimated by biostratigraphic correlation to other sections, of 15 +/- 2 Ma, overlying...
- A large basalt flow with a plagioclase Ar-Ar age of 15.89 +/- 0.29 Ma

Use the Metropolis algorithm to combine all three of these constraints to calculate a single posterior estimate for the age of the critically important horizon in the middle. What does this distribution look like? How much do the posterior mean and standard deviation for the horizon of interest differ from the prior ones?

Note that these uncertainties are reported (as most Geochronologists tend to) following the *two-sigma* convention, meaning actual σ is half of the stated uncertainty in each case. You can asume that all of these ages an uncertainties are Normally distributed; laboratory geochron analyses generally tend to be Normal thanks to the CLT.

2

Part 2

Just as you've finished your analysis, the paleontologists come back from the field. They've discovered an new fossil-bearing horizon below the last one, but still above the basalt. They also found some volcanic sanidine in this new horizon with an Ar-Ar age of 14.95 +/- 0.19 Ma (two-sigma). Use the Metropolis algorithm to combine all of the data on hand to calculate the most accurate posterior ages you can for *both* of the fossil-bearing strata.

3

Report

In addition to your code, please describe your results and their significance in the format of a brief lab report (i.e., some *paragraphs of text*) with one or more figures. While you may collaborate with others while coding (as long as this is clearly noted in comments within your code), the report should be purely your own work.

What are some of the advantages and limitations of this method? If you knew the arrival time uncertainty for the earthquake arrival times in Project 2 (say, two seconds), could you use the Metropolis algorithm to estimate the earthquake origin and uncertainty?

Some of the things you might think about for computational project reports *in general, if applicable* include:

- Code correctnes and quality
  - Code runs and produces correct result
  - Code is commented and has decent code style
  - Code is efficient enough to run in reasonable time for this problem (I.e., if small data, efficiency is not important and simplicity can take precedent; with large problems code must be efficient enough to be practical)
- Conceptual grasp of methods
  - Explanation of why method used is appropriate for this question
  - Any equations implemented are given and explained
  - Assumptions made in applying this method are discussed
  - Limitations of this method are given
- Figures
  - Any data which would be best communicated with a figure has a figure
  - Figures are readable
  - Figure captions give any context necessary to understand figures
- Understanding and communicating meaning of your result
  - What is the scientific question addressed here?
  - What was the computational result?
  - What is the scientific meaning of the computational results?
  - What questions remain unanswered by computational results, and do the results suggest any additional questions?

(N.B., you can treat this more as a checklist than a list of questions to answer!)