Astro732 - Computational Methods for Physics and Astronomy
CLASS #19

19 Introduction to Monte Carlo Markov Chains

Question: What were the ideas behind Monte Carlo integration?

- No need to sample all the M-dimensional volume to do the integral.
- Acceptance/rejection
- Drawing from a random distribution
- Importance sampling to aid efficiency

Can we use some of these ideas to more optimally find our best-fit values of a set of model parameters $\mathbf{a}$ along with their probability distribution?

Let's take a moment to remind ourselves of the problem and our notation. Our goal is, given a model function, $f(\mathbf{x}|\mathbf{a})$ for our N-dimensional data $\mathbf{y}$, we want to find the set of M parameters, $\mathbf{a}$, for which $P(\mathbf{a}|\mathbf{y})$ is maximized.

- According to Bayes
$$P(\mathbf{a}|\mathbf{y}) \propto P(\mathbf{y}|\mathbf{a})P(\mathbf{a}) \tag{1}$$
where $P(\mathbf{a})$ encapsulates our prior knowledge of the parameters $\mathbf{a}$ and $P(\mathbf{y}|\mathbf{a})$ is a calculable probability (or "likelihood") of the measured data set given the parameters.

- Recall, for example, that for independent data points with normally distributed uncertainties with variance σ_i^2 we have
$$P(\mathbf{y}|\mathbf{a}) = \prod_{i=1}^{N} \exp - \left(\frac{(y_i - f(x_i|\mathbf{a}))^2}{2\sigma_i^2} \right). \tag{2}$$
That is, the probability of the data set given the parameters is equal to the product of the probabilities of each data point given the parameters.

- But note that in our formalism so far these things don't have to be strict probabilities. Equation 1 is a proportionality, not an equality. For it to be an equality we would need to include our normalization factor. This gives us some leverage in defining our stand-in for $P(\mathbf{y}|\mathbf{a})$.

- To save typing, define the *posterior* probability of the model as
$$\pi(\mathbf{a}) = P(\mathbf{y}|\mathbf{a})P(\mathbf{a}) \tag{3}$$
so that
$$P(\mathbf{a}|\mathbf{y}) \propto \pi(\mathbf{a}). \tag{4}$$
Again, because it is unnormalized, $\pi(\mathbf{a})$ is not a probability density with unit integral.

- But do we care? What we're really after is the *shape* of $P(\mathbf{a}|\mathbf{y})$ after all. We want to know where it peaks, what it's variance is, etc. All that is encoded in the shape of $\pi(\mathbf{a})$.
- Now all we need to do is to find a way to sample from $\pi(\mathbf{a})$ so that we can map out the probability distribution of our parameters. Enter Monte Carlo Markov Chains.

19.1 Metropolis-Hastings MCMC

There are two main ideas behind all functioning MCMC algorithms, these are originally due to Metropolis (Metropolis, Rosenbluth, Rosenbluth, Teller, & Teller, 1953, J. Chem. Phys., 21, 1987) and later generalized by Hastings (1970).

- **Ergodic Chains:** Now there is a large body of mathematical and statistical descriptions of ergodicity and ergodic chains. Here's what I surmise are the key properties. An ergodic Markov chain is a sequence of positions in parameter space $\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2 \ldots$ that
  1. is completely defined by a transition probability $p(\mathbf{a}_i|\mathbf{a}_{i-1})$ that depends *only* on the previous point;
  2. has the *potential* to visit every point in the parameter space of interest.

  Of course, (1) is a definition while (2) is a constraint on the definition, but I include it for reasons that will be obvious below. The key point here is the introduction of the *transition probability* from one state (set of parameters) to another.
- **Detailed Balance:** If we choose $p(\mathbf{a}_i|\mathbf{a}_{i-1})$ such that it satisfies the detailed balance equation
$$\pi(\mathbf{a}_i)p(\mathbf{a}_{i-1}|a_i) = \pi(\mathbf{a}_{i-1})p(\mathbf{a}_i|\mathbf{a}_{i-1}) \tag{5}$$
then the Markov chain will sample $\pi(\mathbf{a})$ ergodically.

Now let's try to add some insight to the ideas above.

- The ergodic chain idea above is just a method of sampling parameter space. The fact that you use a simple transitional probability does two major things:
  1. It makes *all* of parameter space (at least in principle) available to be sampled; and
  2. it limits the correlation length in the data so that the end state is only weakly correlated with the starting state.
- The detailed balance idea above is what gets the Markov chain to sample $\pi(\mathbf{a})$ ergodically. Integrate the detailed balance equation with respect to $\mathbf{a}_{i-1}$:
$$\begin{aligned}\int \mathrm{d}\mathbf{a}_{i-1} p(\mathbf{a}_i|\mathbf{a}_{i-1})\pi(\mathbf{a}_{i-1}) &= \pi(\mathbf{a}_i)\int \mathrm{d}\mathbf{a}_{i-1} p(\mathbf{a}_{i-1}|\mathbf{a}_i) \\ &= \pi(\mathbf{a}_i).\end{aligned}$$
This equations makes the profound statement that if $\mathbf{a}_{i-1}$ is drawn from the distribution $\pi(\mathbf{a})$,then so is $\mathbf{a}_i$!

- This is great. All we need to do is to pick a starting point $\mathbf{a}_0$ and then random walk (according to $p(\mathbf{a}_i|\mathbf{a}_{i-1})$) our way through parameter space. Ergodicity guarantees that we will eventually come to equilibrium.

- Note that this is all independent of what $p(\mathbf{a}_i|\mathbf{a}_{i-1})$ is as long as it satisfies the detailed balance equation.

Implementing $p(\mathbf{a}_i|\mathbf{a}_{i-1})$ with Metropolis-Hastings

Here's a recipe for implementing MCMC according to Metropolis-Hastings and with my commentary added. I particularly like the discussion about this method in Verde *et al* 2003ApJS..148..195V.

1. Choose a *proposal distribution* $q(\mathbf{a}_i|\mathbf{a}_{i-1})$. This can be anything you want as long as it can allow, in principle, steps from $\mathbf{a}_{i-1}$ to anywhere else in parameter space. The example *NR* uses is a multivariate normal distribution centered on $\mathbf{a}_{i-1}$.

2. Now draw a candidate value, $\mathbf{a}_{ic}$ from $q(\mathbf{a}_i|\mathbf{a}_{i-1})$. This is just a candidate step for the time being.

3. Now calculate an acceptance probability $\alpha(\mathbf{a}_{i-1}, \mathbf{a}_{ic})$ according to

$$\alpha(\mathbf{a}_{i-1}, \mathbf{a}_{ic}) = \min\left(1, \frac{\pi(\mathbf{a}_{ic})q(\mathbf{a}_{i-1}, \mathbf{a}_{ic})}{\pi(\mathbf{a}_{i-1})q(\mathbf{a}_{ic}|\mathbf{a}_{i-1})}\right) \tag{6}$$

4. Draw a uniformly distributed random number, u, between 0 and 1. If $u < \alpha(\mathbf{a}_{i-1}, \mathbf{a}_{ic})$ then accept the candidate step: $\mathbf{a}_i \to \mathbf{a}_{ic}$. If not, then set $\mathbf{a}_i = \mathbf{a}_{i-1}$.

5. Go back to step 2 and take the next step.

6. The net result of the algorithm in step 4 is the transition probability

$$p(\mathbf{a}_i|\mathbf{a}_{i-1}) = q(\mathbf{a}_i|\mathbf{a}_{i-1})\alpha(\mathbf{a}_{i-1}, \mathbf{a}_i) \tag{7}$$

19.2 The Art of MCMC

Okay, so as I've just demonstrated, any monkey can implement the MCMC algorithm. What are the aspects of this that require thought?

- **Choice of Transition Probability** There is a lot of literature on how to optimally choose your transition probability function. The only guidance I am willing to give in these notes is the following:

    1. you may consider something that is symmetric in the transition between states: ie, a function that only depends on $\mathbf{r} = |\mathbf{a}_i - \mathbf{a}_{i-1}|$. Doing this gives you the simplification of the calculation of α as shown in the example above.

2. Also notice that the breadth of the transition probability strongly affects what fraction of the trial choices are accepted. Too wide and you spend all your time calculating likelihoods for low-likelihood parameter sets. Too narrow and your chain becomes strongly correlated. Again, there is literature on how to make this choice (and even on how to do this with feedback). In general, it has been shown that a good (optimal?) acceptance rate is 23.4%.

- **Burn In** Depending on your initial guess of parameters, it takes some time for the Markov Chain to reach equilibrium (find the minimum in parameter space). This is classically called the "burn in" period for the chain. The worse your initial guess, the longer the burn in time.

- **Multiple Chains** One of the real powers of this approach is that you can always (and often shoud) run multiple chains with different starting positions. This does a few things

    1. It allows you to explore more of parameter space outside of the equilibrium region (good if there are multiple minima in the likelihood).
    2. It allows you to parallelize the code easily.