Learning from data: Markov Chain Monte Carlo sampling

Christian Forssén

Department of Physics, Chalmers University of Technology, Sweden

Sep 20, 2020

1 Why MCMC

We have been emphasizing that everything is a pdf in the Bayesian approach. In particular, we studied parameter estimation for which we were interested in the posterior pdf of parameters $\boldsymbol{\theta}$ in a model M given data D and other information I

$$p(\boldsymbol{\theta}|D, I) \equiv p(\boldsymbol{\theta}).$$

Suppose that this parametrized model can make predictions for some quantity $y = f(\boldsymbol{\theta})$ and that we would like to compute the expectation value of this prediction given our knowledge of the model parameters

$$\langle f(\boldsymbol{\theta}) \rangle = \int f(\boldsymbol{\theta}) p(\boldsymbol{\theta}|D, I) d\boldsymbol{\theta} \equiv \int g(\boldsymbol{\theta}) d\boldsymbol{\theta}.$$

The function $f(\boldsymbol{\theta})$ might represent the prediction of some new data that was not part of the original data set D used to constrain the model.

Note that this is more involved than traditional calculations in which we would use a single vector of parameters, e.g. denoted $\boldsymbol{\theta}^*$, that we might have found by maximizing a likelihood (minimizing a chi-squared function). Instead, $\langle f(\boldsymbol{\theta}) \rangle$ means that we do a multidimensional integral over the full range of possible $\boldsymbol{\theta}$ values, weighted by the probability density function, $p(\boldsymbol{\theta}|D, I)$ that we have worked out.

- This is a lot more work!
- The same sort of multidimensional integrals appear when we want to marginalize over a subset of parameters $\boldsymbol{\theta}_B$ to find a pdf for the rest, $\boldsymbol{\theta}_A$. E.g., when extracting the masses of binary black holes from gravitational-wave signals there are many (nuisance) parameters that characterize, e.g., background noise that should be integrated out.

- Therefore, in the Bayesian approach we will frequently encounter these multidimensional integrals. However, conventional methods for low dimensions (Gaussian quadrature or Simpson's rule) become inadequate rapidly with the increase of dimension.
- In particular, the integrals are problematic because the posterior pdfs are usually very small in much of the integration volume so that the relevant region has a very complicated shape.

1.1 Monte Carlo integration

To approximate such integrals one turns to Monte Carlo (MC) methods. The straight and naive version of MC integration evaluates the integral by randomly distributing n points in the multidimensional volume V of possible parameter values $\boldsymbol{\theta}$. These points have to cover the regions where $p(\boldsymbol{\theta}|D, I)$ is significantly different from zero. Then

$$\langle f(\boldsymbol{\theta})\rangle = \int_V g(\boldsymbol{\theta}) d\boldsymbol{\theta} \approx V\langle g(\boldsymbol{\theta})\rangle \pm V\sqrt{\frac{\langle g^2(\boldsymbol{\theta})\rangle - \langle g(\boldsymbol{\theta})\rangle^2}{n}},$$

where

$$\langle g(\boldsymbol{\theta})\rangle = \frac{1}{n}\sum_{i=0}^{n-1} g(\boldsymbol{\theta}_i)$$

$$\langle g^2(\boldsymbol{\theta})\rangle = \frac{1}{n}\sum_{i=0}^{n-1} g^2(\boldsymbol{\theta}_i)$$

Example: One-dimensional integration. The average of a function $g(\theta)$ on $\theta \in [a, b]$ is

$$\overline{g(\theta)} = \frac{1}{b-a}\int_a^b g(\theta) d\theta,$$

from calculus. However, we can estimate $g(\bar{\theta})$ by averaging over a set of random samples

$$\overline{g(\theta)} \approx \frac{1}{n}\sum_{i=0}^{n-1} g(\theta_i).$$

Let us consider the integral

$$\langle f(\theta)\rangle = \int_a^b g(\theta) d\theta \approx \frac{b-a}{n}\sum_{i=0}^{n-1} g(\theta_i),$$

where $b - a$ is the volume V.

Slow convergence. The main uncertainty lies in assuming that a Gaussian approximation is valid. Note the dependence on $a/\sqrt{n}$, which means that you can get a more precise answer by increasing n. However, the result only gets better very slowly. Each additional decimal point accuracy costs you a factor of 100 in n.

The key problem is that too much time is wasted in sampling regions where $p(\boldsymbol{\theta}|D,I)$ is very small. Consider the situation where the significant fraction for one parameter is 10^{-1}. For a m-parameter problem the significant fraction of the volume is 10^{-m}! This necessitates *importance sampling* which reweights the integrand to more appropriately distribute points (e.g. the VEGAS algorithm), but this is difficult to accomplish.

The bottom line is that its not feasible to draw a series of independent random samples from $p(\boldsymbol{\theta}|D,I)$ from large $\boldsymbol{\theta}$ dimensions. Independence means if $\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \ldots$ is the series, knowing $\boldsymbol{\theta}_1$ doesn't tell us anything about $\boldsymbol{\theta}_2$.

However, the samples don't actually need to be independent. they just need to generate a distribution that is proportional to $p(\boldsymbol{\theta}|D,I)$. E.g., a histogram of the samples should approximate the true distribution.

1.2 Markov Chain Monte Carlo

A solution is therefore to do a *random walk* in the parameter space of $\boldsymbol{\theta}$ so that the probability for being in a region is proportional to $p(\boldsymbol{\theta}|D,I)$ in that region.

- The position $\boldsymbol{\theta}_{i+1}$ follows from $\boldsymbol{\theta}_i$ by a transition probability (kernel) $t(\boldsymbol{\theta}_{i+1}|\boldsymbol{\theta}_i)$.
- The transition probability is *time independent*, which means that $t(\boldsymbol{\theta}_{i+1}|\boldsymbol{\theta}_i)$ is always the same.

A sequence of points generated according to these rules is called a *Markov Chain* and the method is called Markov Chain Monte Carlo (MCMC).

Before describing the most basic implementation of the MCMC, namely the Metropolis and Metropolis-Hastings algorithms, let us list a few state-of-the-art implementations and packages that are available in Python (and often other languages)

emcee: emcee is an MIT licensed pure-Python implementation of Goodman & Weare's Affine Invariant Markov chain Monte Carlo (MCMC) Ensemble sampler

PyMC3: PyMC3 is a Python package for Bayesian statistical modeling and probabilistic machine learning which focuses on advanced Markov chain Monte Carlo and variational fitting algorithms.

PyStan: PyStan provides an interface to Stan, a package for Bayesian inference using the No-U-Turn sampler, a variant of Hamiltonian Monte Carlo.

PyMultiNest: PyMultiNest interacts with MultiNest, a Nested Sampling Monte Carlo library.

We have been using emcee extensively in this course. It is based on ensamble samplers (many MCMC walkers) with affine-invariance. For more details, there is the paper (see above) and some lecture notes

1.3 The Metropolis Hastings algorithm

The basic structure of the Metropolis (and Metropolis-Hastings) algorithm is the following:

1. Initialize the sampling by choosing a starting point $\boldsymbol{\theta}_0$.
2. Collect samples by repeating the following:
   (a) Given $\boldsymbol{\theta}_i$, *propose* a new point $\boldsymbol{\theta}_{i+1}$, call it $\boldsymbol{\phi}$, sampled from a proposal distribution $q(\boldsymbol{\phi}|\boldsymbol{\theta}_i)$. This proposal distribution could take many forms. However, for concreteness you can imagine it as a multivariate normal with mean given by $\boldsymbol{\theta}_i$ and variance $\boldsymbol{\sigma}^2$.
      - The transition density will (usually) give a smaller probability for visiting positions that are far from the current position.
      - The width $\boldsymbol{\sigma}$ determines the average step size and is known as the proposal width.

   (b) Compute the Metropolis(-Hastings) ratio r (defined below). Note that the second ratio is equal to one if the proposal distribution is symmetric. It is then known as the Metropolis algorithm.

   (c) Decide whether or not to accept candidate $\boldsymbol{\phi}$ for $\boldsymbol{\theta}_{i+1}$.
      - If $r \geq 1$: accept the proposal position and set $\boldsymbol{\theta}_{i+1} = \boldsymbol{\phi}$.
      - If $r < 1$: accept the position with probability r (remember that now we have $0 \leq r < 1$) by sampling a uniform $\mathrm{U}(0,1)$ distribution. If $u \sim \mathrm{U}(0,1) \leq r$, then $\boldsymbol{\theta}_{i+1} = \boldsymbol{\phi}$ (accept); else $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i$ (reject).

      Note that the chain always grows (even if the proposed step is rejected in which case you add the current position again.

   (d) Loop until the chain has reached a predetermined length.

The Metropolis(-Hastings) ratio is

$$r = \frac{p(\boldsymbol{\phi}|D,I)}{p(\boldsymbol{\theta}_i|D,I)} \frac{q(\boldsymbol{\theta}_i|\boldsymbol{\phi})}{q(\boldsymbol{\phi}|\boldsymbol{\theta})}.$$

- The Metropolis algorithm dates back to the 1950s in physics, but didn't become widespread in statistics until almost 1980.
- It enabled Bayesian methods to become feasible.
- Note, however, that novadays there are much more sophisticated samplers than the original Metropolis one.

1.4 Visualizations of MCMC

- There are excellent javascript visualizations of MCMC sampling on the internet.
- A particularly useful set of interactive demos was created by Chi Feng, and is available on the github page: The Markov-chain Monte Carlo Interactive Gallery
- An accessible introduction to MCMC, with simplified versions of Feng's visualizations, was created by Richard McElreath. It promotes Hamiltonian Monte Carlo and is available in a blog entry called Markov Chains: Why Walk When You Can Flow?

2 Challenges in MCMC sampling

There is much to be written about challenges in performing MCMC sampling and diagnostics that should be made to ascertain that your Markov chain has converged (although it is not really possible to be 100% certain except in special cases.)

We will not focus on these issues here, but just list a few problematic pdfs:

- Correlated distributions that are very narrow in certain directions. (scaled parameters needed)
- Donut or banana shapes. (very low acceptance ratios)
- Multimodal distributions. (might easily get stuck in local region of high probability and completely miss other regions.)