Notes for Hogg and Foreman-Mackey "DATA ANALYSIS RECIPES: USING MARKOV CHAIN MONTE CARLO"

Malte Brinch
University of Copenhagen
Septemper 2019

1. Why do we need MCMC?

Markov Chain Monte Carlo (MCMC) methods are methods for sampling probability distribution functions or probability density functions (pdfs). They only require that you be able to compute ratios of the pdf at pairs of locations. MCMC works well for posterior pdf sampling. The posterior pdf p(θ | D) is constructed from the likelihood p(D | θ) and the prior p(θ)

$$p(\theta|D) = \frac{1}{Z} p(D|\theta)p(\theta)$$

where Z=p(D) is the evidence and is hard to calculate but this is not needed if you compute the pdf ratios for two poits in parameter space.
given a huge set of data D and a model (likelihood function and prior) we can for a huge set of parameters θ use MCMC to give us a fair sampling of the posterior pdf.

MCMC should not be used to search the parameter space for good models. MCMC should also not be used to optimizes the posterior pdf, or, for certain choices of prior pdf, optimizes the likelihood. MCMC is good at one thing, and one thing only: Sampling ill-normalized (or otherwise hard to sample) pdfs.

2. WHAT IS A SAMPLING?

If we sample a pdf p(θ) we draw a number of K values and if K is large enough we could make a histogram and it would up to a normalization constant be the same as the original pdf. A pdf is always positive and its integral over the total parameter space is equal to 1 that is there is a 100% chance the parameters have some value. There can be many parameters and so the integral can be multidimensional. If we have the pdf p(θ) we can define the expectation values for θ or for a any value that can be expressed as a function of θ, g(θ)

$$E_{p(\theta)}[\theta] = \int \theta p(\theta) d\theta$$

$$E_{p(\theta)}[g(\theta)] = \int g(\theta) p(\theta) d\theta$$

These expectation values are the mean values of θ and g(θ) under the pdf. A good sampling makes the sampling approximation to these integrals accurate. We can replace the integrals with sums over samples θ_k for a good sampling

$$E_{p(\theta)}[\theta] \approx \frac{1}{K} \sum_{k=1}^{K} \theta_k$$

$$E_{p(\theta)}[g(\theta)] \approx \frac{1}{K} \sum_{k=1}^{K} g(\theta_k)$$

Often when working with MCMC we have a badly normalized function f(θ) which is the pdf p(θ) multiplied by some unknown scalar that is hard to cumpute. we assume the that f(θ) is postive and that its integral is finite. When we sample f(θ) the sampling-approximation expectation values are the expectation values under the properly-normalized corresponding pdf, even though you might never determine what the normalization is.

$$E_{p(\theta)}[g(\theta)] = \frac{\int g(\theta) f(\theta) d\theta}{\int f(\theta) d\theta}$$

$$E_{p(\theta)}[g(\theta)] \approx \frac{1}{K} \sum_{k=1}^{K} g(\theta_k)$$

MCMC is for solving integrals!
Integrals can be used to find the mean, median and quantiles of a pdf but not the mode. if there are nuisance parameters of little interest in the parameter vector we can just drop them when we sample. This means that, the projection of the sampling to the subspace of interest produces a sampling of the marginalized pdf, marginalizing (or projecting) out the nuisance parameters.

for inference f(θ) is product of the likelihood function and prior.

3. Metropolis-Hastings

The idea to update a parameter with a function f(θ) in MH MCMC is the following:

- Draw a proposal θ' from the proposal pdf q($\theta|\theta'$).
- Draw a random number from 0 to 1 from a uniform distribution.
- If $\frac{f(\theta')}{f(\theta)}$ > r then opdate the parameter otherwise keep it the same.

This algorithm | and indeed any MCMC algorithm | produces a biased random walk through parameter space. It is a random walk because each part in the chain only depends on the last, this is also why it is a Markov chain. It is biased because of the acceptance algorithm involving the ratios of function values; this acceptance rule biases the random walk such that the amount of time spent in the neighborhood of location θ is proportional to f(θ). The proposal pdf must satisfy detailed balance $q(\theta'|\theta) = q(\theta|\theta')$ (does not have to but difficult to work with otherwise). A typical choice for q is a multi-variate Gaussian distribution for θ' centered on θ with some simple (diagonal, perhaps) variance tensor.

Write everything in the same parameterization or you will have to fuck with the jacobian. Due to the dynamic range often associated with these problems it can be easier to work with the natural logarithm so instead you compare $ln(f(\theta')) - ln(f(\theta)) > ln(r)$ This protects you from underflow, but exposes you to some (negative) infinities, if you end up taking the logarithm of a zero.

4. LIKELIHOODS AND PRIORS

pseudo code for ln-f

```
def ln-f(pars, data):
x = ln-prior(pars)
if not is-finite(x):
return -Inf
return x + ln-likelihood(data, pars)
```

This pseudo-code ensures that you only compute the likelihood when the parameters are within the prior bounds; it assumes implicitly that the prior pdf is easier to compute than the likelihood.

If you are using "flat" (improper) priors, the ln-prior() function can just return a zero no matter what the parameters. If it is flat with bounds (that is, proper), the ln-prior() should check the bounds and return -Inf values when the parameter vector is out of bounds.

5. AUTOCORRELATION and CONVERGENCE

In general, when one has a sequence (θ_1; θ_2; θ_3; $\cdots$) generated by a Markov Process in the θ space, nearby points in the sequence will be similar, but sufficiently distant points will not "know about" one another; the autocorrelation function measures this. The Monte Carlo error introduced by this approximation is proportional to $\sqrt{\tau_{int}/N}$ where τ_{int} is the integrated autocorrelation time and N is the total number of samples of p(θ). In other words, τ_{int} is the number of steps required for the chain to produce an independent sample.

6. TUNING

for a too narrow proposal distribution almost all steps are accepted but it will take a long time to move anywhere. If the proposal distribution is too wide | it proposes steps too large | the moves will cover parameter space easily, but almost no steps will be accepted. The acceptance fraction should be 0.2-0.5 with a value around 0.234 for high dimensional problems.

you can only tune during burn-in phase and not the final MCMC fun since that would break the Markov property. another thing to use for tuning is Expected Squared Jump Distance. This is the mean squared distance the walker moves, per step.

7. INITIALIZATION AND BURN-IN

Initialization is where the walker(s) start from. a optimal place to start the walkers can be done by optimizing the likelihood or the posterior pdf in advance. This is not good in too high dimensions but for for a few to tens it should be ok. Burn-in is the movement of the walker if it starts in a non typical place. For badly multi-modal, then you will have to start at multiple points in parameter space and compare the resulting chains.

8. RESULTS, ERROR BARS, AND FIGURES

We should use as our "results" outputs from the MCMC that are based on integrals computed with the sampling. This includes expectations, medians, quantiles, one-dimensional histograms, and multi-dimensional histograms. This does not include the "best" sample or a mode or optimum. To report the one sigma errorbar we use the interval that contains 68 percent of the posterior samples around the center. The two sigma error can be done in the same way to see the skewness of the posterior pdf. You should report the results as the parameter value or values for the best sample. Giving some posterior samples is a good idea.
If you want to report the best sample, instead give the parameter values found by optimizing the posterior pdf, starting at the best sample as an initialization. This is called the maximum a posteriori or MAP. It is like a maximum likelihood value but regularized by the prior.

It is important, when reporting the output of an MCMC run by giving means or medians of the posterior pdf to remind the reader or user of that output that it does not necessarily represent (collectively) a good fit to the data; these outputs only give information that is useful one parameter at a time. In principle the only output from the sampling that safely gives both probabilistic information about the result of the inference and also good-fitting models is a few | randomly chosen | example samples. We strongly recommend this; in particular this is a better thing to do than to give the "best sample", which isn't even guaranteed to be near the bulk of the posterior pdf or the samples therefrom.

there are useful figures to plot for showing MCMC results

- Trace plots: plots parameter values as a function of step number.
- Posterior predictive plots: take some K random samples from your chain, plot the prediction that each sample makes for the data and over-plot the observed data.
- Corner plots or scatterplot matrices: for a D dimensional parameter space plot the one dimensional histograms and choose-2 two-dimensional histograms (scatter plots).

9. TROUBLESHOOTING AND ADVICE

a good idea is to do functional testing for your sampler. See if it samples a known distribution right or try to sample your prior to see if your is really what you think it is. You can test this by changing your ln-likelihood function to always return 0

The likelihood function should always return the same values for the same parameters. If random numbers are used for internal integration in the likelihood function then pick some at the beginning and save them for every likelihood call. This speeds up the code and makes the likelihood function single valued. Even better is to replace internal Monte Carlo integrals with deterministic numerical (or even better analytic) integrals. Visualize slices through parameter space to check if the likelihood function is smooth this is important for Hamiltionian methods as they depend on gradients.

low or high acceptance fraction can be controlled by the step size. Lower stepsize gives higher acceptance fraction and vice versa.

Plot parameter values as a function of "step number" or iteration number. Plot the ordered chain in each parameter dimension. These plots will have long horizontal patches if the chain is getting stuck.

If you reparameterize you have to change your priors which brings in a Jacobian. Affine invariant samplers are invariant to those transformations

Try plotting residuals of the data away from the model in the space of the data and look at what parts of the data the model "explains well" and what parts of the data the model "explains badly".

10. MORE SOPHISTICATED SAMPLING METHODS

Most users of MCMC have one of two (related) problems: Either (1) it is taking too long it is taking too many steps or executing too many calls of the function f(θ) to get independent samples or (2) it is not exploring the full parameter space there are local optima (or bad local geometry around those optima) "trapping" the algorithm.

MH and ensemble samplers have problems with very large numbers or parameters. For situations where some parameters are global and affect every part of the data and some local parameters only affect some data points we can use a gibbs sampler where parameters are updated one at a time while the others are kept constant. Hamiltonian sampling uses the derivative/gradient of the function f(θ) to augment the position of the walker in parameter space with a fictitious momentum in parameter space, and use the derivatives in the accept-reject step. These samplers are the best to deal with many parameters. Nested sampling are of a censored version of the prior, censored by the value of the likelihoood function. When the likelihood censoring is strong, only the most high-likelihood parts of the prior get sampled; when the likelihood censoring is weak, almost all of the prior gets sampled. It is useful for exploring the full parameter space or searching for all of the modes of the posterior pdf.