Notes on Various MCMC Algorithms

Maya Watanabe
Christina Catlett

May 27, 2020

1 Single Chain Methods

1.1 Metropolis-Hastings

Original and simplest MCMC algorithm. Used to sequentially sample parameter space for candidate parameters, accepting or rejecting the candidates based on the evaluation of the posterior distribution. Can be univariate or multivariate.

- Requires the definition of priors for parameters, likelihood function (proposal distribution), and initial guess for the parameters, and, for multivariate models, a covariance matrix.
- Algorithm:
  1. Choose initial values for the parameter set θ_{init}, and calculate its posterior probability according to Baye's Theorem.
  2. Independently sample a random candidate value for each parameter in θ_{prop} from the proposal distribution.
  3. Compute the posterior probability of θ_{prop}, again according to Baye's Theorem.
  4. Accept or reject θ_{prop} as the new 'best guess' for the parameters according to the acceptance criteria, replacing θ_{curr}
  5. Repeat a large number of times, eventually reaching convergence
- Acceptance Criteria: θ_{prop} is accepted if it produces a higher posterior probability than θ_{curr}, meaning that it is more likely for the model's parameters to be θ_{prop} than θ_{curr}. Additionally, to avoid converging at non-global maxima, candidate θ_{prop}'s which are not found to be more likely than θ_{curr} can still be accepted with a probability proportional to the difference between the proposed and current parameter sets. Formalized,

  $$\rho = \frac{P(\theta_{prop})}{P(\theta_{curr})}$$

  where $P(\theta)$ represents the evaluation of the posterior for parameters θ
  - If $\rho \geq 1$, $\theta_{curr} = \theta_{prop}$
  - If $\rho < 1$, $\theta_{curr} = \theta_{prop}$ with a probability of ρ
- Code availability: Metropolis-Hastings (MH) can be run using built-in MATLAB command 'mhsample'. Aditionally, we wrote an MH-based linear fitting model located at `https://github.com/shtyllab/2020-HMC-REU-Codes/tree/master/subteam1/astrostats_tutorial`

1.2 Adaptive

The basis of this Adaptive MCMC algorithm is the Metropolis-Hastings algorithm but the proposal distribution is "tuned" (or updated) along the search according to the covariance calculated from a fixed number of previous states.

- Initiation of this algorithm requires the same definitions required by the MH MCMC algorithm: parameter priors, likelihood functions, a starting parameter set, and a covariance matrix (for multivariate distributions.
- We can also think of the adaptive algorithm as an MH algorithm where the proposald distribution depends on time
- Algorithm:
  1. Let q be the initial proposal distribution and assume that points $X_1, ..., X_k$ have been sampled
  2. Candidate point Y is sampled from the proposal distribution $q_k(\cdot|X_1, ..., X_k)$
     - Y has been sampled from a distribution that depends on the history of the chain
  3. Accept Y with a probability $\alpha(X_k, Y) = min(1, \frac{\pi(Y)}{\pi(X_k)})$
     - Where $\pi(\cdot)$ is the unscaled probability density of the target distribution (posterior)
- Acceptance Criteria: Y is accepted with probability 1 if the posterior distribution for Y is greater than the posterior for the current parameter set X_k: $\alpha(X_k, Y) = min(1, \frac{\pi(Y)}{\pi(X_k)})$. It is important to remember that this posterior (π) is continuously updated based on the covariance matrix computed from the previous samples.
- How does the algorithm update and how does the proposal distribution depend on the "history" of sampling?
  - Starting at time 0, we sample using the MH algorithm
  - Some time later at time t, we have sampled at least H points $\{X_1, ..., X_{t-H+1}, ..., X_t\}$
  - We then define the proposal distribution at time t, q_t for sampling proposal state Y as: $q_t(\cdot|X_1, ..., X_t) \sim N(X_t, c_d^2 R_t)$
    - R_t: $d \times d$ covariance matrix determined by points $X_{t-H+1}, ..., X_t$ (i.e. all the points before Y)
    - c_d: a scaling factor
  - The update occurs when the new covariance matrix is calculated using the previously sampled data
- Resource: `https://link.springer.com/article/10.1007/s001800050022#Sec3`
- Code availability:
  - The *DREAM toolbox* provides a function for adaptive MCMC: `http://faculty.sites.uci.edu/jasper/files/2015/03/manual_DREAM.pdf`
  - *Adaptive Metropolis Hastings and Factor Slice Sampling* file exchange also implements an adaptive MCMC algorithm: `https://www.mathworks.com/matlabcentral/fileexchange/45976-adaptive-metropolis`

2 Parallel Methods

In general it seems that parallel methods are better at sampling from "multi-modal targets" – a larger parameter space as having multiple chains allows a broader exploration of the parameter space.

2.1 Parallel Tempering

In its essence, parallel tempering consists of running many MH chains synchronously, allowing for swaps between the chains. This is beneficial because it becomes more likely that a larger portion of the potential parameter space is explored in a reasonable amount of time, in comparison to MH. This is especially beneficial in a distribution where areas of high probability are seperated by areas of low probability, making it unlikely that a traditional MH chain would explore all high-probability areas.

The idea of "tempering" comes from the idea that a "temperature" parameter can be used to flatten a distribution. As the temperatures rises, the distribution flattens, making the likely random walk performed by MH span more of the parameter space. The idea of parallelism enters when a number of MCMC chains, k, are run simultaneously at different temperatures (degrees of flattening) given by $p(x_k)$ where x_k represents the values of the chain. The chains of different temperatures are allowed to "mix" by swapping entries with a swap probability, leading to wider exploration, and potentially a more likely final parameter estimate.

- Because parallel tempering uses the MH algorithm to run its replica chains (chains at different temperatures), it relies on having the same initial information as MH. It additionally requires a set of temperatures of length M, t_k, and a defined number of sweeps, N_{sweep}. Each "sweep" of the model consists of running the chains for N_{iter} iterations and then making swaps.

- Algorithm:

  1. For each sweep, $i = 1, ..., N_{sweep}$
     (a) For each temperature, compute N_iter iterations of an MH MCMC chain, adjusting the likelihood function, $q(x_{prop}|x_{curr})$ to consider the temperature:
     $$q(x_{prop}|x_{curr}) = x_{curr} + N(0, I/t_k$$
     (b) For each chain $k = 1, ..., M-1$, swap the elements x_k^i and x_{k+1}^i according to the swap criteria; that is, switch the parallel elements in the chain at the current temperature and the temperature directly above according to the swap criteria.

- Acceptance Criteria: Same as MH algorithm (Section 1.1)

- Swap criteria: The algorithm intends to keep the "warmest" temperature chain after each sweep. This means that if performing the swap would increase the temperature, the swap will be performed. Likewise, the swap would be unlikely to occur if the temperature is lowered. This is formalized by accepting the swap with a probability α_k:

  $$\alpha = \min(1, \frac{p_k(x_{k+1})p_{k+1}(x_k)}{p_k(x_k)p_{k+1}(x_{k+1})}$$

  In the above expression, the numerator represents the swap: evaluating x_k in the temperature function for the chain $k+1$, and vice versa. The denominator represents the current temperature: evaluating x_k in p_k and x_{k+1} in p_{k+1}. If the newly proposed temperature is warmer, the fraction will have a value > 1, and the swap will occur with a probability of 1. If not, the swap will be performed with a probability of α_k.

- Code availability: MATLAB code for parameter estimation using parallel tempering was developed as part of "Evaluation of Parallel Tempering to Accelerate Bayesian Parameter Estimation in Systems Biology" by Gupta et al. (2018) `https://github.com/RuleWorld/ptempest`

- Resource: `https://www.cs.ubc.ca/~nando/540b-2011/projects/8.pdf`

2.2 Parallel Adaptive

The basis of parallel adaptive MCMC (aka inter-chain adaptation (INCA)) is to use the algorithm of the adaptive single-chain MCMC method and apply it to multiple chains. The concept of parallel chains is used to detect different regions of parameter space with high probability under the posterior distribution.

- The very foundation of parallel adaptive methods is run using the MH method. Thus, these methods require the same initial conditions.
- Inter-chain Adaptive (INCA) Algorithm:
  1. Run K different chains in parallel
     - Each chain is started independently from the same over-dispersed starting distribution (this is the initial guess for the likelihood function but will change in the future)
  2. After a burn-in period, the K kernels are simultaneously adapted using *all the samples* provided by the K chains so fair
     - This is where the adaptive algorithm is borrowed from the single-chain method above as the sampling distribution is updated based on the samples already taken
     - If our MCMC method is random walk Metropolis with Gaussian proposals, updating kernels are equivalent to setting the proposal covariance matrix to the sample covariance matrix of all the available samples
     - Note that within this step, the step of accepting and rejecting the parameter set also takes place
  3. Stop the swapping of information between chains when the chains no longer each contribute different information about the target distribution
     - Do this by using the *potential scale reduction* R (Brooks-Gelman-Rubin R), i.e. a convergence factor
- Acceptance Criteria: Same as Adaptive algorithm (Section 1.2)
- Swap Criteria:
- Tempered INCA (TINCA) Algorithm: The idea is to apply adaptive and tempering methods to multiple chains.

  Start with $T = t_{max}$ and at each temperature and at each temperature:

  1. For $T = t_j$ perform INCA for a target density until convergence
  2. Keep the simulation parameters obtained (after rejection/acceptance that occurs within INCA) and repeat 1) with the next colder temperature $T = t_{j-1}$. Stop after $T = 1$

  - We assume that the kernel (or covariance matrix) adapted/updated at temperature t_j is a reasonable starting choice for the kernel used at temperature t_{j-1}
  - The goal of TINCA is to produce a reasonable starting proposal in a high dimensional problem (something usually difficult to do)
  - TINCA significantly reduces the burn-in period (compared to INCA).
- Acceptance Criteria: Same as Adaptive algorithm (Section 1.2)
- Swap Criteria: Same as Parallel Tempering (Section 2.1)
- Code availability: adaptive_pt.m: Implementation of adaptive parallel tempering from `https://www.tandfonline.com/doi/full/10.1080/10618600.2013.778779`
- Resource: `http://probability.ca/jeff/ftpdir/chao4.pdf`