Bayesian Estimation of Simulation-Model Parameters via Markov Chain Monte Carlo (MCMC)

Nathaniel Osgood

University of Saskatchewan

Laboratory for Computational Epidemiology and Public Health Informatics

osgood@cs.usask.ca

Traditional Parameter Estimation via Calibration in (Dynamic) Simulation Models

- Point estimation of parameter values
- Optional: Estimation of confidence intervals surrounding estimate
- Limitations
  - Single dynamic model assumed
  - Single point estimates (doesn't make apparent how this yields estimate in certain output measures – e.g. comparison between interventions)
  - Even with estimate of intervals, assuming that unimodal
    - No understanding of global shape of likelihood
  - Error metric for estimation (e.g. square of estimate) imposes distributional assumptions on errors

Prior & Posterior Parameter Distributions

- Before MCMC, we often have some sense as to where the parameters fall
  - We can encode this with a “prior” distribution
    - This is called a “prior” because we can formulate it prior to observing the data or settling on a model
- MCMC can give us an updated distribution, which we call the “posterior” distribution
  - This takes into account not only our best guesses for the parameter values, but also the likelihood that the model(s) used could explain the observed data given certain parameter values

Notation

- θ: Parameter values (i.e., a particular mapping between parameters and values)
- y: Observed data

Posterior Distributions for Parameters (θ): Moving Beyond Point Estimates

- A posterior distribution over θ helps us understand the relative likelihood of parameter values in light of
  - The observed data (y)
  - One or more models under consideration
  - Any pre-existing expectations of the relative likelihood of different parameter values (a prior distribution over θ)
- By sampling from a posterior over θ, we can e.g., derive
  - Point estimates (MLE, mean, Minimum var, &c) of parameters & model output
  - Likelihood intervention a is “better” than intervention B
  - “Credibility intervals” for diverse model output

Markov Chain Monte Carlo Helps us Reliably Derive Parameter Posterior Distributions

- MCMC is a principled way of generating samples from θ according to posterior distribution over θ
- These samples from the posterior distribution of θ are generated over and over again $\Rightarrow$ Monte Carlo
  - For each such sample, we can e.g.
    - Accumulate a running sample mean of e.g.
      - θ
      - Model outputs
      - Intervention
    - Record the likelihood at that point
    - Record the shape of the distribution
- A "Markov Chain" is used to generate the θ samples

High-Level Operation of Using MCMC for Parameter Estimation of Simulation Model

- Elements to specify before algorithm to use in algorithm
  - Assign prior distributions (over parameters, models, etc.) may or may not be parametric (using 'hyperparameters')
  - Formulate likelihood formulae that indicate relative likelihood of seeing particular values of empirical data (y) in light of values of parameters (θ) & simulation model output
    - These could be either parametric or non-parametric
  - Conceptually (only), apply Bayes' rule to get rule for likelihood of parameter θ in terms of data (y) and priors
- A Markov Chain is used to generate samples of parameters (θ), with (asymptotic stationary) distribution proportional to posterior distribution
  - For each generated parameter sample, we sample output measures of interest & accumulate statistics, etc.

Why Do We want to Use a Markov Chain to Generate the Samples θ?

- Simple to generate samples from univariate distributions where cumulative distribution function CDF is known in advance
  - Draw random number x from uniform distribution
  - Return parameters in CDF that gives a cumulative probability at value x
- Can't typically use above to sample from posterior
  - We don't know the CDF up front (it depends on simul.!)
  - Often dealing with such a high dimensional θ space that trying to derive the CDF would involve huge work
- Markov Chain approaches lets us generate samples of the parameters (θ) from posterior without knowing it or its CDF in advance

Bayes' Rule

- $P(\theta|y)=P(y|\theta)P(\theta)/P(y)$
- Here,
  - $P(\theta|y)$ is the posterior probability of θ. We want this!
  - $P(y|\theta)$ is what we specify in our likelihood rules
  - $P(\theta)$ is the prior we specify for θ
  - $P(y)$ is just some constant that doesn't vary with θ
- Observation: The *relative value* of $P(\theta|y)$ for different θ depends only on $P(y|\theta)P(\theta)$, and not on $P(y)$
  - So to compute the relative value of $P(\theta_1|y)$ and $P(\theta_2|y)$, just need to compare $P(y|\theta_1)P(\theta_1)$ &$P(y|\theta_2)P(\theta_2)$

Rough Markov Chain Operation (MH)

- The Markov Chain generation process involves
  - Drawing a candidate value from θ (e.g., just generated from random perturbation around last generated sample
  - Assessing (relative) posterior of this candidate value of θ
    - Run the simulation model
      - Required b/c formulated likelihood statements for $P(y|\theta)$ generally depend on simulation model output (for that value of parameters θ)
    - Compute relative posterior of candidate θ using $P(y|\theta)P(\theta)$
      - $P(\theta)$ is just the value of the prior at this point
      - $P(y|\theta)$ is in generate calculated using not just θ and y, but also outputs and intermediate values of simulation model produced for value of θ
    - Accept or reject this candidate θ based on ratio between this relative posterior at θ vs. last generated sample
      - If accepted, release this as our new sample!
      - If rejected, emit existing value and go on to the next candidate!
- Run through "burn in" time before using results

Takes significant work!

Common Simple Accept/Reject Rule (e.g., Random Walk Metropolis-Hastings w/Symmetric Perturbation)

- If we can find a higher relative likelihood point than the current one, we go there automatically
- If we find a lower likelihood point than the current one, we only use that as our new sample with a certain probability (ratio of relative likelihoods involved)
  - Otherwise stay with higher likelihood point (which therefore gets more representation among sampled points – as befits its higher relative likelihood)

Schematic of Random Walk Metropolis Hastings Algorithm (1 walker)

i=0

Pick a value for θ_0 // can draw from crude dist.

Until convergence criterion is satisfied

Pick a random value for perturbation $\Delta\theta$

Create a candidate value $\tilde{\theta} = \theta_i + \Delta\theta$

if uniform(0,1)$\leq \min\left(1, \frac{Posterior(\tilde{\theta})}{Posterior(\theta_i)}\right)$ /* NB: can calc even if only know posterior up to a constant */

$\theta_{i+1} = \tilde{\theta}$ //Accept

else

$\theta_{i+1} = \theta_i$ // Reject (keep same sample for emission)

Emit θ_{i+1} as next sample // emit regardless if was accepted

i=i+1

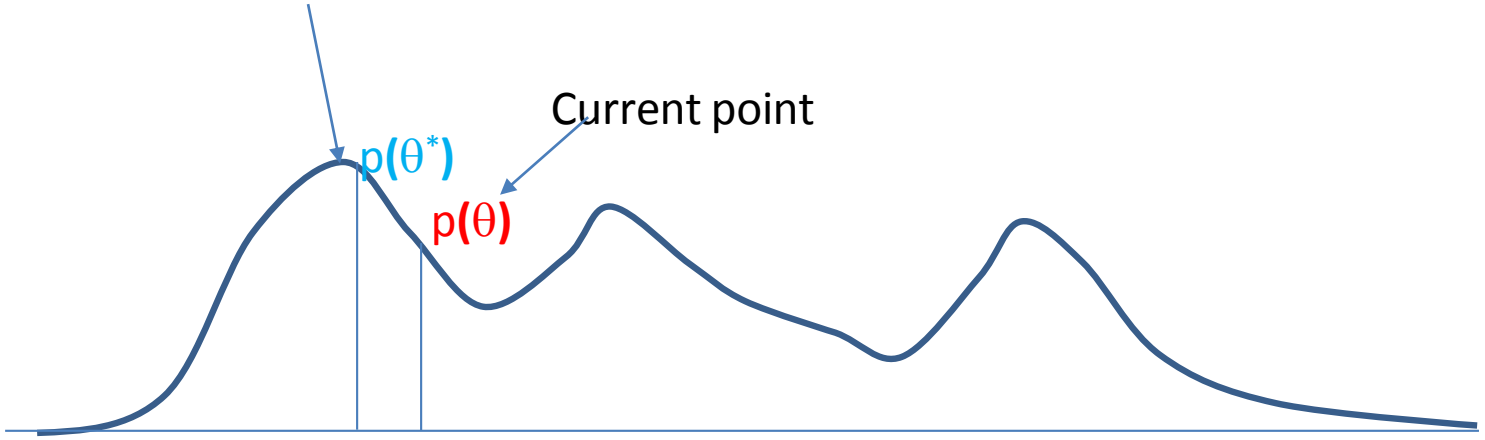
Elements of Why (Random Walk M-H) MCMC Works

For simplicity, we denote

θ as θ^*

θ_i as θ

p(x): For MCMC, posterior being sampled



p(θ*) candidate

θ (Current point)

Value of parameter(s)
In general, this is multidimensional

We move from θ to θ* with probability $\min\left(1, \frac{Posterior(\theta^*)}{Posterior(\theta)}\right)$

NB: Even if p(θ*) < p(θ), May still transition current pos. from θ to (candidate) θ* ⇒ **aids repeated explore entire distribution**

Elements of Why MCMC Works

- Only 1 stationary distribution for Markov Chain
- The actual posterior is a stationary distribution for the Markov Chain

$\Rightarrow$The posterior must be the unique stationary distribution for the Markov Chain

  - If we run it enough, we will get a distribution of values very close to posterior

Role of Simulation Model(s) in Simulation-model based parameter estimate

- In simulation-model based parameter estimation, the simulation model(s) serves as a critical mediator between parameters and observed data
  - The model helps us project behavioral consequences of parameter assumptions (in light of dynamic hypotheses captured by those model(s))
  - The estimated likelihood that the output data will be observed is typically conditional on such simulation model outputs rather than directly on the parameters θ
  - This process can also be used with stochastic simulation models (but would typically require many more Monte Carlo runs to deal with variability in sim. output given θ)
- Id of sim model can just be another parameter in θ

Likelihood Formulas

- Likelihood statements $P(y|\theta)$ must be specified by MCMC model creator to assess likelihood of seeing particular empirical data (y) values in light of parameters values (θ)
- These statements are a key factor in judging the relative likelihood of different parameters (θ) for a model
  - These must be thought through carefully

– Example: Suppose that the simulation model has a stock of diagnosed individuals with TB (D_t) at a given time *t*
  - We might express the relative likelihood that a given count of diagnosed individuals d_t was reported at time t in empirical data
  - We might assume that for each diagnosed person, there is a probability δ of reporting, and indicate that the likelihood of d_t people reported given D_t true people in the population is proportional to Binomial(d_t ; D_t, δ)

Computational Effort

- Running the simulation model often takes a large computational effort (often 100,000s of runs)
  - Typically far more than generating random values or other elements of the MCMC "core" algorithm!
- We have to run the model many times beyond the values of θ that are actually yielded as samples
  - Through burn-in period (could be 100,000s of runs)
  - In rejected samples
    - Model still has to be run to get output to assess likelihood that observed data could results from this set of parameter value
    - Not actually among generated points
- Much of this work (e.g., different walkers) can be done in parallel (concurrently on diff. machines)

Practical Concerns 1

- Burn-in time
  - Need to run for long enough that starts generating values approximately distributed as posterior
  - There are ways of judging this
- Tuning algorithm parameters (e.g. random walk perturbation size) to yield medium acceptance rate
  - Recommended: 1D: 0.44, n-D: 0.23
  - High enough to explore (don't just remain in same place)
  - Not so high that just going around in local area of optimum, without significantly improving estimate
    - If old and new ratios are close to 1, likelihood of accepting new estimate is close to 1

Practical Concerns 2

- High covariance b/t parameters lowers efficiency
  - Do iteration over residuals using regression
- Easiest if operating in unbounded space: Transform
- Sample thinning rate: To reduce correlation between successive samples, accept only 1 out of every *n* samples
  - This will help draws from posterior be closer to independent draws - which is what is being sought
- Assessing Convergence
  - Here, combine statistics between walkers with those from the same walker to get an estimate of how much would be gained by further sampling
    - If within-walker statistics are similar to those between walkers, this is a sign that may be doing good job in exploring space

Many MCMC Algorithms

- There are many variants of MCMC algorithms
- These differ in
  - Range of applicability (e.g. Gibbs sampling is associated with successive conditional formulation of model)
  - Simplicity (e.g. Random walk Metropolis-Hastings) vs. Sophistication e.g.
    - Choosing between models of different dimension – e.g. Reverse Jump MCMC
    - Allowing asymmetric random perturbations, asymmetric candidate draws from random distribution (Metropolis-Hastings vs. Metropolis)

Where is the Markov Chain?

- The algorithm uses an implicit Markov Chain that need not be represented explicitly (reified)
- Conceptually, the states of the Markov chains are associated with draws from random variables
- We are remaining in one state (with the preexisting sample) until we accept a new candidate (drawn at random from the distribution)
- When we accept a new candidate and generate a new sample, we are transitioning to a new state

Broader Context of Monte Carlo Methods

- MCMC is a batch ("offline") Monte Carlo algorithm
  - Processes all data at once to yield posterior
- There are a class of Sequential Monte Carlo ("online") methods (e.g. Particle Filtering)
  - Conceptually similar to (Extended) Kalman filtering
  - Here, at discrete points over time, take new data and formulate new estimates incorporating that data
- Rejection sampling: Simpler technique (no Markov Chain) when can bound the posterior distribution