Introduction to Bayesian Data Analysis

Overview of Markov Chain Monte Carlo

Alix I. Gitelman

Statistics Department
Oregon State University
gitelman@science.oregonstate.edu

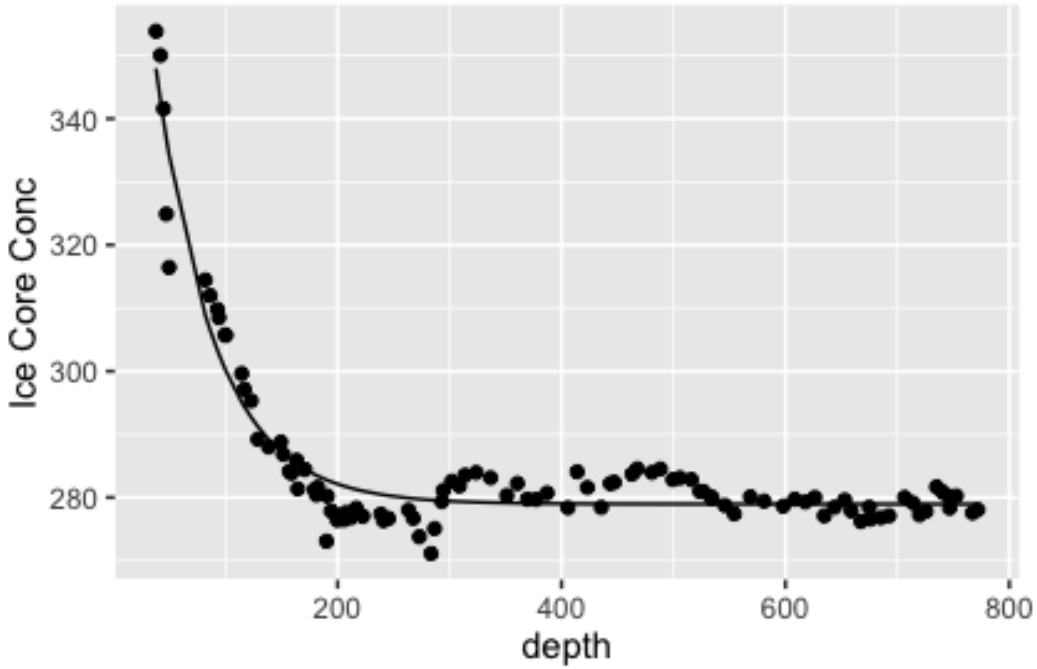
July 25, 2016

More Complicated Models

For many one-parameter problems like the binomial problem, we can obtain analytical (or exact) solutions for the posterior distribution.

- This is even true for some multi-parameter problems (e.g., Normal data with unknown mean and variance)
- But, for many interesting problems, there aren't analytical solutions, and so we must rely on numerical and computational methods for estimation.
- A popular tool of choice in Bayesian inference is MCMC—Markov Chain Monte Carlo.

CO_2 Data, Revisited



Exponential Decay

Putting aside our goal of estimating continuous atmospheric CO_2 for the moment, let's just model the ice core CO_2 measurements as a function of depth.

- For this, an exponential decay model may be appropriate:

$$y_i = \alpha_1 + \alpha_2 e^{-\alpha_3 x_i} + \epsilon_i,$$

  for $i = 1, \ldots, 101$. Here, x_i is depth; α_1, α_2 and α_3 are the three model parameters; and for simplicity, ϵ_i are independent $N(0, \sigma^2)$ errors.

- This model has four parameters, and there's no closed form solution for estimating them—a frequentist approach uses nonlinear least squares, and we'll use MCMC in a Bayesian approach.

MCMC

Markov Chain Monte Carlo (MCMC) is a method in which we generate random variables from a (usually high dimensional) posterior distribution—in our exponential decay model, there are four parameters.

- We then use these random simulations to obtain summaries of the posterior distributions of the parameters
- Our inferences won't be exact, but if we increase the number of simulations, they will be quite good
- We have to check a number of things to make sure that the simulations we obtain are from the posterior distribution of interest—this involves making sure that the chain has converged

Markov Chains

Some high-level background on Markov chains:

- A Markov chain is a **stochastic process**, which is just a family of random variables, X_t, where t runs over some index set, T
- For us $T \subset \mathbb{N}$, and $t = 1, 2, 3, \ldots$ just indexes the successive draws from the Markov chain.
- The state space of a Markov chain is the support space for the random variables, X_t (i.e., the range of all values X_t can take).
- A Markov chain is distinguished by the **Markov property** which governs how the chain moves around its state space.

The Markov Property

The Markov property: given the value of X_t, the value of X_{t+1} does not depend on the values of X_u for $u < t$.

- In words: the probability of the future behavior of a Markov chain—when its current state is known exactly—is not changed by any knowledge concerning its past behavior.
- Notationally:

$$Pr(X_{t+1} = x_{t+1} | X_t = x_t, \ldots, X_1 = x_1) = Pr(X_{t+1} = x_{t+1} | X_t = x_t).$$

Markov Chains

We're going to deal with Markov chains whose state spaces are the support spaces of the parameters we're trying to estimate.

- For example, in the non-linear regression model,

$$y_i = \alpha_1 + \alpha_2 * e^{-\alpha_3 x_i} + \epsilon_i$$

$\alpha_1 \in [0, \infty), \alpha_2 \in [0, \infty), \alpha_3 \in [0, \infty)$, and $\sigma^2 \in [0, \infty)$ so the state space for our Markov chain is:

$$[0, \infty) \times [0, \infty) \times [0, \infty) \times [0, \infty)$$

- We'll construct a Markov chain that will take draws from this state space, where each successive draw depends on the value of the most recent draw and the posterior distribution of the parameters given the data.

MCMC

Suppose that $\boldsymbol{\theta}$ is the parameter vector of interest.

- The general idea of MCMC: build a continuous state-space Markov chain whose state-space is the support space of $\boldsymbol{\theta}$, and whose stationary distribution *is* the joint posterior distribution, $[\boldsymbol{\theta}|\mathbf{y}]$.
- A stationary distribution is the long-run (or limiting) distribution of the Markov chain (i.e., it's the distribution of the random variables in the chain that the chain settles into after we run it for a long time)

MCMC

If we run an appropriately constructed Markov chain long enough, we will end up drawing samples from the posterior distribution of interest, $[\boldsymbol{\theta}|\mathbf{y}]$.

- This is a very clever idea, and surprisingly, it turns out not to be hard to create Markov chains whose stationary distribution is $[\boldsymbol{\theta}|\mathbf{y}]$.

- The key is in the construction of the "transition kernels" that are used to jump around the parameter space.

MCMC

Typically, MCMC algorithms update each parameter separately, although sometimes the algorithms run faster if highly correlated parameters are updated together. The general idea is that for each iteration of the Markov chain, we run through all of the parameters, updating them one (or a few or several) at a time.

- The updates are based on the "complete" or "full conditional" distributions of the parameters.
- For our non-linear regression example, there are four univariate complete conditional distributions:
  - $[\alpha_1 | \alpha_2, \alpha_3, \sigma^2, \mathbf{Y}]$
  - $[\alpha_2 | \alpha_1, \alpha_3, \sigma^2, \mathbf{Y}]$
  - $[\alpha_3 | \alpha_1, \alpha_2, \sigma^2, \mathbf{Y}]$
  - $[\sigma^2 | \alpha_1, \alpha_2, \alpha_3, \mathbf{Y}]$

MCMC

At each iteration of the MCMC, we loop through all of the parameters in the model and update them:

- Each update usually involves a proposal distribution, typically centered at the current value(s) of the parameter(s) being updated, call this θ_{cur}.
- A random draw is taken from this proposal distribution, call this draw θ^*. Roughly, if the posterior density is higher for θ^* than it is for θ_{cur}, the chain moves to θ^*.
- If the posterior density is lower for θ^* than it is for θ_{cur}, then the chain moves to θ^* with probabililty related to $[\theta^*|\mathbf{Y}]/[\theta_{cur}|\mathbf{Y}]$.

Proposal Distributions

One tricky part of MCMC involves specifying proposal distributions (or deciding whether to accept defaults in programs like NIMBLE, more tomorrow).

- In general, we want to ensure that the chain explores the important parts of the parameters space relatively quickly.
- This involves calibrating the tuning parameter(s) of the proposal distribution(s) so that the chain doesn't take steps that are too large or too small around the parameter space
- Good news: for many well-studied models, much of this has been worked out.

MCMC Algorithms

There are many different algorithms for MCMC, and they typically differ in the construction of the transition kernel:

- Metropolis algorithm, Metropolis et al. (1953).
- Metropolis-Hastings algorithm, Hastings (1970)
- Gibbs Sampler
- Hamiltonian Monte Carlo
- Slice sampler
- Reversible jump

Initial Values

As with most computational algorithms, a Markov Chain used in MCMC needs to be assigned an initial state. That is, we have to specify $\boldsymbol{\theta}_0$.

- Sometimes, stationarity is achieved faster depending on the starting values.
- Mostly, stationarity is achieved *regardless* of the starting values, **provided that the posterior is proper!**
- It's useful to start chains at multiple sets of initial values.

The Issue with MCMC Convergence

In many cases, prior specification and choice of initial values can go a long way to speeding convergence of the MCMC.

- If a chain does not converge it may be that the posterior distribution is **improper**. Remember that Bayes theorem is

$$[\theta|\mathbf{X}] = \frac{[\mathbf{X}|\theta][\theta]}{\int [\mathbf{X}|\theta][\theta] d\theta}$$

  The posterior, $[\theta|\mathbf{X}]$, is proper provided that the integral in the denominator is finite.

- $[\theta|\mathbf{X}]$ is a probability distribution function, a key feature of which is that it integrates to one.

MCMC: Monitoring Convergence

There are a few ways to evaluate Markov Chain convergence:

- Use conjugate priors whenever possible—this doesn't ensure fast convergence, but it can help.
- Examine the trace (or history) plots of ALL parameters in the chain: you're looking for stability in the means and the variances.
- Run multiple chains, starting at different starting values and check to see whether the posterior results are the same (within MC-error) across the different chains

CO_2 Example

Recall that the exponential decay model takes the form:

$$y_i = \alpha_1 + \alpha_2 e^{-\alpha_3 x_i} + \epsilon_i,$$

where the ϵ_i are independent $N(0, \sigma^2)$ error terms.

An equivalent way to write this model is:

$$y_i | \alpha_1, \alpha_2, \alpha_3, \sigma^2 \overset{ind}{\sim} N(\mu_i, \sigma^2),$$

where $\mu_i = \alpha_1 + \alpha_2 e^{-\alpha_3 x_i}$.

And to complete the Bayesian model specification, we need priors for $\alpha_1, \alpha_2, \alpha_3$, and σ^2.

CO_2 Example

In the exponential decay model,

$$y = \alpha_1 + \alpha_2 e^{-\alpha_3 * x}$$

α_1 is the horizontal asymptote, $(\alpha_1 + \alpha_2)$ is the y-intercept, and α_3 is the decay rate.

- Knowing these things, we have an idea of what we're shooting for in terms of estimates
- And, we'll be able to think of some reasonable initial values for the parameters in the MCMC: if you look back at the plot, perhaps $\alpha_1^{(init)} = 280$, $\alpha_2^{(init)} = 100$ and $\alpha_3^{(init)} = 0.2$

The Plan from Here

Tomorrow, you will learn about NIMBLE (Numerical Inference for statistical Models for Bayesian and Likelihood Estimation), an R package for running MCMC.

- The key component NIMBLE uses is the Bayesian model you're trying to estimate specified in a format used by BUGS (Bayesian inference Using Gibbs Sampling), one of the first automated programs for MCMC
- NIMBLE in an R package that wraps around BUGS code in which you specify your Bayesian model.

Some BUGS Conventions

Elements that you specify in BUGS are called **nodes**, of which there are three kinds:

- **stochastic nodes:** these are data or parameters that are associated with probability distributions. Examples:

```
Y ~ dnorm(mu,tau)
mu ~ dnorm(0,0.0001)
tau ~ dgamma(1,0.01)
```

- **logical nodes:** these typically functions of stochastic node. Example:

```
sig2 <- 1/tau
```

- **constant nodes:** these are like data, but not stochastic. Example: n, the number of observations.

Stochastic Nodes

These include all of the common probability distributions:

pmf	BUGS name	pdf	BUGS name
Bernoulli	dbern	Beta	dbeta
Binomial	dbin	Gamma	dgamma
Multinomial	dmulti	Normal	dnorm
Negative Bin.	dnegbin	Student's t	dt
Poisson	dpois	Uniform	dunif

Stochastic Nodes

Some less common ones:

Distribution	BUGS name	Comment
Categorical	dcat	can serve as a discrete Uniform
Chi-Square	dchisqr	special case of gamma
Exponential	dexp	exponential (special gamma)
Double Exp.	ddexp	double exponential
Log-Normal	dlnorm	log normal
Pareto	dpar	pareto distribution
Weibull	dweib	Weibull distribution

Stochastic Nodes

Some multivariate distributions:

Distribution	BUGS name	Comment
Dirichlet	ddir	multivariate version of Beta
MVNorm	dmnorm	multivariate Normal
Multivariate T	dmt	multivariate t-distribution
Wishart	dwish	multivariate Chi-square

And even an improper distribution, dflat.

Logical Nodes

Logical nodes are specified using the assignment notation R: as in:

```
sig2 <- 1/tau
```

- Since logical nodes may be based on stochastic nodes, we can make posterior inferences about them.
- The idea is that the stochasticity is attached to τ and then just inherited by σ^2.

Model Specification

Specifying a model in BUGS is not unlike writing the model down with pencil and paper:

- Identify the likelihood—i.e., a model for the data—as a function of the parameters of interest.
- Identify priors for the parameters.

```
model {
      for (i in 1:N) {
            Y[i] ~ dnorm(mu[i],tau)
            mu[i] <- alpha[1] + alpha[2]*exp(-alpha[3]*x[i])
       }
       alpha[1] ~ dgamma(1,0.01)
       alpha[2] ~ dgamma(1,0.01)
       alpha[3] ~ dgamma(1,0.01)
       tau ~ dgamma(1,0.01)
       sig2 <- 1/tau
}
```

CO_2 Example

For the CO_2 example we still have to specify initial values and decide how long to run the chain and start the MCMC.

- I already suggested some initial values for this particular model.
- If we want to run multiple chains (for the purpose of checking convergence), we'll have to specify some additional initial values, maybe just by jittering the ones I already gave.
- Deciding on how long to run the chain depends a lot on how quickly the chain converges, and we may not know this until we run the chain.
- More details to come on this tomorrow.