Part 4

Bayesian computing

ST740

North Carolina State University

Bayesian computing

- Given the prior and data, the posterior is fixed and a Bayesian analysis boils down to summarizing the posterior
- We need point estimates, credible sets, etc
- Summarizing a p-dimensional posterior distribution is challenging for large p
- In the 80's, Bayesian computing was unable to do this for more than a few parameters
- In the 90's, new algorithms were developed that revolutionized Bayesian statistics
- Understanding these algorithms is obviously important

Outline

- **Deterministic methods**
  - MAP estimation
  - Numerical integration
  - Bayesian CLT
  - INLA
- Markov Chain Monte Carlo
  - Gibbs sampling
  - Slice sampling
  - Metropolis-Hastings sampling
  - Hamiltonian Monte Carlo
  - JAGS
  - Convergence diagnostics
- ABC

MAP estimation

- Sometimes you don't need an entire posterior distribution and a single point estimate will do

- Example: prediction in machine learning

- The Maximum a Posteriori (MAP) estimate is the posterior mode

$$\hat{\theta}_{MAP} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ p(\boldsymbol{\theta}|\mathbf{Y}) = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\ \log[f(\mathbf{Y}|\boldsymbol{\theta})] + \log[\pi(\boldsymbol{\theta})]$$

- This is similar to the maximum likelihood estimation but includes the prior (penalty)

Univariate example

Say $Y|\theta \sim \textit{Binomial}(n, \theta)$ *and* $\theta \sim \textit{Beta}(0.5, 0.5)$*, find* $\hat{\theta}_{MAP}$

- The likelihood is $f(Y|\theta) \propto \theta^{Y}(1-\theta)^{n-Y}$
- The log likelihood is[1]

$$\log[f(Y|\theta)] = Y\log(\theta) + (n-Y)\log(1-\theta)$$

- The prior is $\pi(\theta) \propto \theta^{0.5-1}(\theta)^{0.5-1}$
- The log prior[1] is $\log[\pi(\theta)] = -0.5\log(\theta) - 0.5\log(1-\theta)$
- Therefore, the MAP estimator is

$$\hat{\theta} = \arg\max_{\theta}(Y-0.5)\log(\theta) + (n-Y-0.5)\log(1-\theta)$$

[1] ignoring constants that don't depend on θ

Univariate example

Say $Y|\theta \sim$ Binomial(n, θ) and $\theta \sim$ Beta$(0.5, 0.5)$, find $\hat{\theta}_{MAP}$

- The MAP estimator is

$$\hat{\theta} = \arg\max_{\theta} (Y - 0.5)\log(\theta) + (n - Y - 0.5)\log(1 - \theta)$$

- Taking the derivative and setting to zero gives

$$\frac{Y - 0.5}{\theta} - \frac{n - Y - 0.5}{1 - \theta} = 0$$

- The solution (assuming $Y, n - Y \geq 1$) is

$$\hat{\theta} = \frac{Y - 0.5}{n - 1}$$

Bayesian central limit theorem

- Another simplification is to approximate the posterior as Gaussian
- Berstein-Von Mises Theorem: As the sample size grows the posterior doesn't depend on the prior
- Frequentist result: As the sample size grows the likelihood function is approximately normal
- Bayesian CLT: For large *n* and some other conditions $\theta|\mathbf{Y} \approx$ Normal

Bayesian central limit theorem

- Bayesian CLT: For large n and some other conditions

$$\boldsymbol{\theta} \sim \text{Normal}[\hat{\boldsymbol{\theta}}_{MAP}, \mathcal{I}(\hat{\boldsymbol{\theta}}_{MAP})^{-1}]$$

- $\mathcal{I}$ is Fisher's information matrix

- The (j,k) element of $\mathcal{I}$ is

$$-\frac{\partial^2}{\partial\theta_j\partial\theta_k}\log[p(\boldsymbol{\theta}|\mathbf{Y})]$$

  evaluated at $\hat{\boldsymbol{\theta}}_{MAP}$

- We have marginal and conditional means, standard deviations and intervals for the normal distribution

Univariate example

Say $Y|\theta \sim Binomial(n, \theta)$ and $\theta \sim Beta(0.5, 0.5)$, find the Gaussian approximation for $p(\theta|\mathbf{Y})$

- We have seen that (assuming $Y, n - Y \geq 1$),

$$\hat{\theta}_{MAP} = \frac{Y - 0.5}{n - 1}$$

- We have also seen (Jeffreys lecture) that

$$I(\theta) = n\theta^{-1}(1 - \theta)^{-1}$$

- Therefore,

$$\begin{aligned} \theta|Y &\approx \text{Normal}\left[\hat{\theta}_{MAP}, I(\hat{\theta}_{MAP})^{-1}\right] \\ &\approx \text{Normal}\left[\hat{\theta}_{MAP}, \hat{\theta}_{MAP}(1 - \hat{\theta}_{MAP})/n\right] \end{aligned}$$

Illustration of the Bayesian CLT

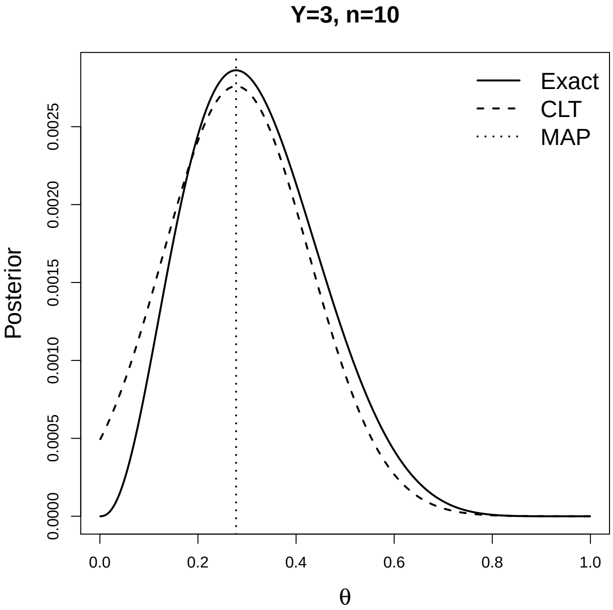


Illustration of the Bayesian CLT

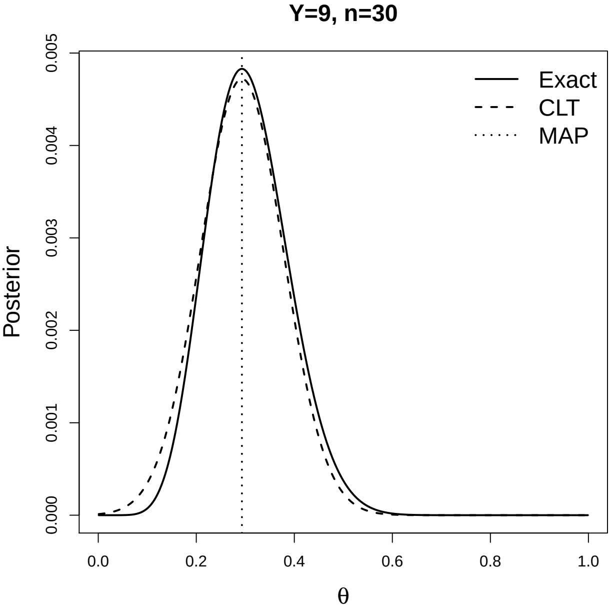
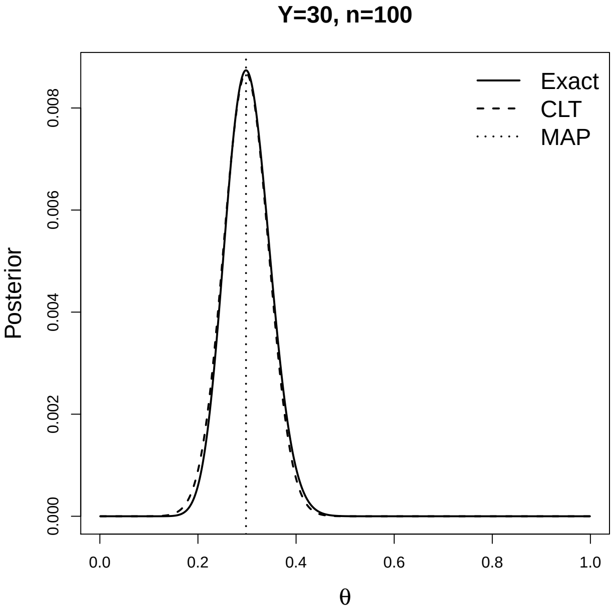


Illustration of the Bayesian CLT



Bayesian central limit theorem

- For large datasets with a small number of parameters evoking the Bayes CLT is probably the best approach
- The approximate posterior can be computing using standard software (e.g., `glm` in `R`)
- The numerical values (e.g., intervals) will equal the frequentist values, but the interpretation remains Bayesian
- Why not just do a frequentist analysis? Well, why not just do a Bayesian analysis?

Numerical integration

- Many posterior summaries of interest are integrals over the posterior
- Ex: $\text{E}(\theta_j|\mathbf{Y}) = \int \theta_j p(\boldsymbol{\theta}) d\boldsymbol{\theta}$
- Ex: $\text{V}(\theta_j|\mathbf{Y}) = \int [\theta_j - \text{E}(\theta|\mathbf{Y})]^2 p(\boldsymbol{\theta}) d\boldsymbol{\theta}$
- These are *p* dimensional integrals that we usually can't solve analytically
- A grid approximation is a crude approach
- Gaussian quadrature is better

Numerical integration

- Numerical integration is only feasible for small p
- The Iteratively Nested Laplace Approximation (INLA) is an even more sophisticated method
- INLA combines Gaussian approximations with numerical integration
- This works well if most of the parameters are approximately normal and only a few are non-Gaussian and require numerical integration

Outline

- Deterministic methods
    - MAP estimation
    - Numerical integration
    - Bayesian CLT
    - INLA
- **Markov Chain Monte Carlo**
    - Gibbs sampling
    - Slice sampling
    - Metropolis-Hastings sampling
    - Hamiltonian Monte Carlo
    - JAGS
    - Convergence diagnostics
- ABC

Monte Carlo sampling

- Monte Carlo (MC) sampling is the predominant method of Bayesian inference because it can be used for high-dimensional models (i.e., with many parameters)

- The main idea is to approximate posterior summaries by drawing samples from the posterior distribution, and then using these samples to approximate posterior summaries of interest

- This requires drawing samples from non-standard distributions

- It also requires careful analysis to be sure the approximation is sufficiently accurate

Monte Carlo sampling

- Notation: Let $\boldsymbol{\theta} = (\theta_1, ..., \theta_p)$ be the collection of all parameters in the model

- Notation: Let $\mathbf{Y} = (Y_1, ..., Y_n)$ be the entire dataset

- The posterior $f(\boldsymbol{\theta}|\mathbf{Y})$ is a distribution

- If $\theta^{(1)}, ..., \theta^{(S)}$ are samples from $f(\boldsymbol{\theta}|\mathbf{Y})$, then the mean of the S samples approximates the posterior mean

- This only provides approximations of the posterior summaries of interest.

- But how to draw samples from some arbitrary distribution $p(\boldsymbol{\theta}|\mathbf{Y})$?

Software optioms

- There are now many software options for performing MC sampling

- There are SAS procs and R functions for particular analyses (e.g., the function `BLR` for linear regression)

- There are also all-purpose programs that work for virtually any user-specified model: OpenBUGS; JAGS; Proc MCMC; STAN; INLA (not MC)

- We will use JAGS, but they are all similar

Gibbs sampling

- Gibbs sampling is attractive because it can sample from high-dimensional posteriors
- The main idea is to break the problem of sampling from the high-dimensional joint distribution into a series of samples from low-dimensional conditional distributions
- Updates can also be done in blocks (groups of parameters)
- Because the low-dimensional updates are done in a loop, samples are not independent
- The dependence turns out to be a Markov distribution, leading to the name Markov chain Monte Carlo (MCMC)

MCMC for the Bayesian t test

- Say $Y_i \sim \text{Normal}(\mu, \sigma^2)$ with $\mu \sim \text{Normal}(0, \sigma_0^2)$ and $\sigma^2 \sim \text{InvGamma}(a, b)$

- We saw that if we knew either μ or σ^2, we can sample from the other parameter

- $\mu|\sigma^2, \mathbf{Y} \sim \text{Normal}\left[\frac{n\bar{Y}\sigma^{-2}+\mu_0\sigma_0^{-2}}{n\sigma^{-2}+\sigma_0^{-2}}, \frac{1}{n\sigma^{-2}+\sigma_0^{-2}}\right]$

- $\sigma^2|\mu, \mathbf{Y} \sim \text{InvGamma}\left[\frac{n}{2}+a, \frac{1}{2}\sum_{i=1}^{n}(Y_i-\mu)^2+b\right]$

- But how to draw from the joint distribution?

Gibbs sampling for the Gaussian model

- The full conditional (FC) distribution is the distribution of one parameter taking all other as fixed and known

- FC1: $\mu|\sigma^2, \mathbf{Y} \sim$ Normal $\left[\frac{n\bar{Y}\sigma^{-2}+\mu_0\sigma_0^{-2}}{n\sigma^{-2}+\sigma_0^{-2}}, \frac{1}{n\sigma^{-2}+\sigma_0^{-2}}\right]$

- FC2: $\sigma^2|\mu, \mathbf{Y} \sim$ InvGamma $\left[\frac{n}{2}+a, \frac{1}{2}\sum_{i-1}^{n}(Y_i-\mu)^2+b\right]$

Gibbs sampling

- In the Gaussian model $\boldsymbol{\theta} = (\mu, \sigma^2)$ so $\theta_1 = \mu$ and $\theta_2 = \sigma^2$
- The algorithm begins by setting initial values for all parameters, $\boldsymbol{\theta}^{(0)} = (\theta_1^{(0)}, ..., \theta_p^{(0)})$.
- Variables are then sampled one at a time from their full conditional distributions,

$$p(\theta_j | \theta_1, ..., \theta_{j-1}, \theta_{j+1}, ..., \theta_p, \mathbf{Y})$$

- Rather than 1 *p*-dimensional joint sample, we make *p* 1-dimensional samples.
- The process is repeated until the required number of samples have been generated.

Gibbs sampling

A Set initial value $\boldsymbol{\theta}^{(0)} = (\theta_1^{(0)}, ..., \theta_p^{(0)})$

B For iteration t,

FC1 Draw $\theta_1^{(t)} | \theta_2^{(t-1)}, ..., \theta_p^{(t-1)}, \mathbf{Y}$

FC2 Draw $\theta_2^{(t)} | \theta_1^{(t)}, \theta_3^{(t-1)}, ..., \theta_p^{(t-1)}, \mathbf{Y}$

...

FCp Draw $\theta_p^{(t)} | \theta_1^{(t)}, ..., \theta_{p-1}^{(t)}, \mathbf{Y}$

We repeat step B S times giving posterior draws

$$\boldsymbol{\theta}^{(1)}, ..., \boldsymbol{\theta}^{(S)}$$

Why does this work?

- $\theta^{(0)}$ isn't a sample from the posterior, it is an arbitrarily chosen initial value

- $\theta^{(1)}$ likely isn't from the posterior either. Its distribution depends on $\theta^{(0)}$

- $\theta^{(2)}$ likely isn't from the posterior either. Its distribution depends on $\theta^{(0)}$ and $\theta^{(1)}$

- **Theorem**: For any initial values, the chain will eventually converge to the posterior

- **Theorem**: If $\theta^{(s)}$ is a sample from the posterior, then $\theta^{(s+1)}$ is too

Proof

Convergence

- We need to decide:
  1. When has it converged?
  2. When have we taken enough samples to approximate the posterior?
- Once we decide the chain has converged at iteration T, we discard the first T samples as “burn-in”
- We use the remaining $S - T$ to approximate the posterior
- For example, the posterior mean (marginal over all other parameters) of θ_j is

$$\mathrm{E}(\theta_j|\mathbf{Y}) \approx \frac{1}{S - T} \sum_{s=S-T+1}^{S} \theta_j^{(s)}$$

Practice problem

- Implementing Gibbs sampling requires deriving the full conditional distribution of each parameter

- Work out the full conditionals for λ and b for the following model:

$$\begin{aligned} &Y|\lambda, b \sim \text{Poisson}(\lambda)\\ &\lambda|b \sim \text{Gamma}(1, b)\\ &b \sim \text{Gamma}(1, 1) \end{aligned}$$

Practice problem

$Y|\lambda, b \sim \text{Poisson}(\lambda)$, $\lambda|b \sim \text{Gamma}(1, b)$, $b \sim \text{Gamma}(1, 1)$

- The full conditional for λ is

$$\begin{aligned}
p(\lambda|b, Y) &\propto \frac{f(Y, \lambda, b)}{f(Y, b)} \propto f(Y, \lambda, b) \\
&\propto f(Y|\lambda, b)\pi(\lambda|b)\pi(b) \\
&\propto f(Y|\lambda)\pi(\lambda|b) \\
&\propto \left[\exp(-\lambda)\lambda^{Y}\right]\left[\exp(-b\lambda)\lambda^{1-1}\right] \\
&\propto \exp[-(b+1)\lambda]\lambda^{(Y+1-1)}
\end{aligned}$$

- Therefore, $\lambda|b, Y \sim \text{Gamma}(Y+1, b+1)$

Practice problem

$Y|\lambda, b \sim$ Poisson(λ), $\lambda|b \sim$ Gamma$(1, b)$, $b \sim$ Gamma$(1, 1)$

- The full conditional for b is

$$\begin{aligned} p(\lambda|b, Y) &\propto \frac{f(Y, \lambda, b)}{f(Y, \lambda)} \propto f(Y, \lambda, b) \\ &\propto f(Y|\lambda)\pi(\lambda|b)\pi(b) \\ &\propto \pi(\lambda|b)\pi(b) \\ &\propto \left[b^1 \exp(-b\lambda)\right] \left[\exp(-b)b^{1-1}\right] \\ &\propto \exp[-(\lambda+1)b]b^{(2-1)} \end{aligned}$$

- Therefore, $b|\lambda, Y \sim$ Gamma$(2, \lambda + 1)$

Non-conjugate priors sampling

- In Gibbs sampling each parameter is updated by sampling from its full conditional distribution
- This is possible with conjugate priors
- However, if the prior is not conjugate it is not obvious how to make a draw from the full conditional
- For example, if $Y \sim \text{Normal}(\mu, 1)$ and $\mu \sim \text{Beta}(a, b)$ then

$$p(\mu|Y) \propto \exp\left[-\frac{1}{2}(Y-\mu)^2\right]\mu^{(a-1)}(1-\mu)^{b-1}$$

- For some likelihoods there is no known conjugate prior, e.g., logistic regression
- In these cases we can use slice or Metropolis sampling

Slice sampling

- Slice sampling introduces an auxiliary variable to apply Gibbs sampling to non-conjugate priors

- Say θ is univariate and u is the auxiliary variable

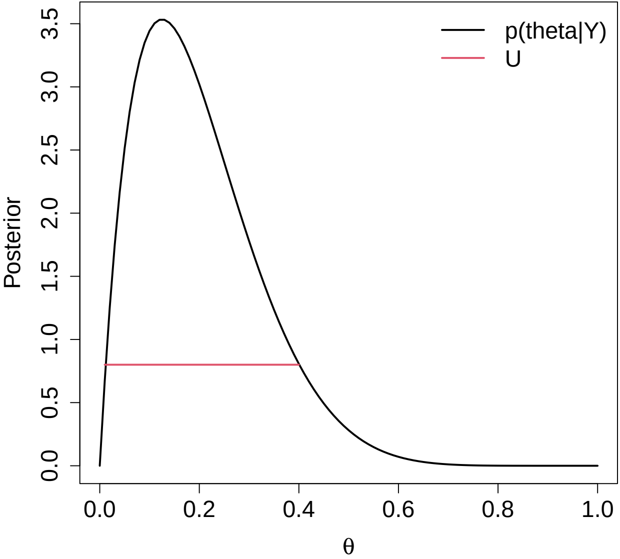
- Consider the joint density function

$$g(\theta, U) = I[0 < U < p(\theta|\mathbf{Y})]$$

- The marginal density of θ is $p(\theta|\mathbf{Y})$

- So if we make draws from (θ, U) and discard U, the draws of θ will be draws from the desired posterior

Slice sampling



Slice sampling

- Slice sampling is Gibbs sampling for (U, θ)
- The full conditional distribution of U is

$$U|\theta, \mathbf{Y} \sim \text{Uniform}(0, f(\theta|Y))$$

- The full conditional distribution of θ is

$$\theta|U, \mathbf{Y} \sim \text{Uniform on } \mathcal{D}_U = \{\theta; f(\theta|\mathbf{Y}) > U\}$$

- Updating θ requires solving for or approximating the excursion set $\mathcal{D}_U$

Metropolis sampling

- Metropolis sampling is a version of rejection sampling
- Let θ_j^* be the current value of the parameter being updated and $\theta_{(j)}$ be the current value of all other parameters
- You propose a random candidate based on the current value, e.g.,
$$\theta_j^c \sim \text{Normal}(\theta_j^*, s_j^2)$$
- The candidate is accepted with probability
$$R = \min\left\{1, \frac{p(\theta_j^c|\theta_{(j)}, \mathbf{Y})}{p(\theta_j^*|\theta_{(j)}, \mathbf{Y})}\right\}$$
- If the candidate is not accepted then you simply retain the previous value and move to the next step

Metropolis sampling

- The candidate standard deviation s_j is a tuning parameter
- Ideally s_j is tuned to give acceptance probability around 0.3-0.4
- If s_j is too small:
- If s_j is too large:
- Off-the-shelf programs have default values, and many allow you to change the value if the results are unsatisfactory

Metropolis-Hastings sampling

- Denote $\theta_j^c \sim q(\theta|\theta^*)$ as the candidate distribution

- The candidate distribution is symmetric if

$$q(\theta^*|\theta_j^c) = q(\theta_j^c|\theta^*)$$

- For example, if $\theta_j^c \sim \text{Normal}(\theta_j^*, s_j^2)$ then

$$q(\theta_j^c|\theta^*) = \frac{1}{\sqrt{2\pi}s_j} \exp\left[-\frac{(\theta_j^c - \theta_j^*)^2}{2s_j^2}\right] = q(\theta^*|\theta_j^c).$$

Metropolis-Hastings sampling

- Metropolis-Hastings (MH) sampling generalizes Metropolis sampling to allow for asymmetric candidate distributions
- For example, if $\theta_j \in [0,1]$ then a reasonable candidate is

$$\theta_j^c | \theta_j^* \sim \text{Beta}[10\theta_j^*, 10(1-\theta_j^*)]$$

- Then $q(\theta_j^* | \theta_j^c)$ and $q(\theta_j^c | \theta^*)$ are both beta PDFs
- MH proceeds exactly like Metropolis except the acceptance probability is

$$R = \min\left\{1, \frac{p(\theta_j^c | \theta_{(j)}, \mathbf{Y}) q(\theta_j^* | \theta_j^c)}{p(\theta_j^* | \theta_{(j)}, \mathbf{Y}) q(\theta_j^c | \theta_j^*)}\right\}$$

Metropolis-Hastings sampling

- What if we take the candidate distribution to be the full conditional distribution

$$\theta_j^c \sim p(\theta_j^c|\theta_{(j)}, \mathbf{Y})$$

- What is the acceptance ratio?

$$\frac{p(\theta_j^c|\theta_{(j)}, \mathbf{Y})q(\theta_j^*|\theta_j^c)}{p(\theta_j^*|\theta_{(j)}, \mathbf{Y})q(\theta_j^c|\theta_j^*)} = \frac{p(\theta_j^c|\theta_{(j)}, \mathbf{Y})p(\theta_j^*|\theta_{(j)}, \mathbf{Y})}{p(\theta_j^*|\theta_{(j)}, \mathbf{Y})p(\theta_j^c|\theta_{(j)}, \mathbf{Y})} = 1$$

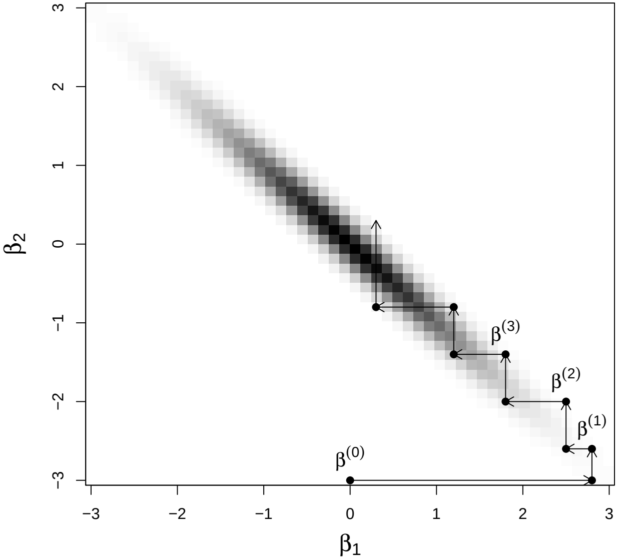
- What does this say about the relationship between Gibbs and Metropolis Hastings sampling?

- Gibbs is a special case of MH with the full conditional as the candidate

Variants

- You can combine Gibbs and Metropolis in the obvious way, sampling directly from full conditional when possible and Metropolis otherwise
- Adaptive MCMC varies the candidate distribution throughout the chain
- If a group of parameters are highly correlated convergence can be slow
- One way to improve Gibbs sampling is a block update
- For example, in linear regression might iterate between sampling the block $(\beta_1, ..., \beta_p)$ and σ^2
- Blocked Metropolis is possible too
- For example, the candidate for $(\beta_1, ..., \beta_p)$ could be a multivariate normal

Posterior correlation leads to slow convergence



Metropolis-adjusted Langevin algorithm (MALA)

- MALA sampling improves convergence by using the posterior's gradient $g(\boldsymbol{\theta}) = \nabla \log\{p(\boldsymbol{\theta}|\mathbf{Y})\}$ with j^{th} element

$$g_j(\boldsymbol{\theta}) = \frac{\partial}{\partial\theta_j}\log\{p(\boldsymbol{\theta}|\mathbf{Y})\} = \frac{\partial}{\partial\theta_j}\log\{f(\mathbf{Y}|\boldsymbol{\theta})\pi(\boldsymbol{\theta})\}$$

- It is a special case of Metropolis-Hastings sampling and approximates Langevin dynamics
- The candidate distribution is

$$\boldsymbol{\theta}^* \sim \text{Normal}\left(\boldsymbol{\theta} + \tau g(\boldsymbol{\theta}), 2\tau\Sigma\right)$$

- The tuning parameter is $\tau \in (0,1)$
- The candidate covariance matrix Σ could be approximately the posterior covariance

Hamiltonian Monte Carlo (HMC)

- HMC is (sort of) a multi-step extension of MALA

- HMC is a discrete approximation to Hamiltonian dynamics

- The algorithm has two tuning parameters, the number of steps L and the step size τ

- It also introduces momentum variable $\mathbf{z} = (z_1, ..., z_p)$

HMC proposal distribution

- For MCMC iteration s, set $\boldsymbol{\theta}^* = \boldsymbol{\theta}^{(s-1)}$ and sample $\mathbf{z} \sim \text{Normal}(\mathbf{0}, \mathbf{I}_p)$ and set $\mathbf{z}^* = \mathbf{z}$

- Repeat the following steps L times
  1. Set $\mathbf{z}^* = \mathbf{z}^* + \tau g(\boldsymbol{\theta}^*)/2$
  2. Set $\boldsymbol{\theta}^* = \boldsymbol{\theta}^* + \tau \mathbf{z}^*$
  3. Set $\mathbf{z}^* = \mathbf{z}^* + \tau g(\boldsymbol{\theta}^*)/2$

- The final of $\boldsymbol{\theta}^*$ is the candidate for the Metropolis step

- The MH acceptance probability is $\min\{1, R\}$ for

$$R = \frac{p(\boldsymbol{\theta}^*|\mathbf{Y})}{p(\boldsymbol{\theta}^{(s-1)}|\mathbf{Y})} \frac{\exp(-\sum_{j=1}^p z_j^{*2}/2)}{\exp(-\sum_{j=1}^p z_j^2/2)}.$$

HMC proposal distribution

- One option is to set L at a moderate value, say $L = 20$, and turn τ to give acceptance rate ≈ 0.8
- Alternatively, the No-U-Turns Sampler (NUTS) can be used to select L automatically
- Very loosely speaking, if you run HMC with huge L, it will eventually start doing loops around the posterior's support
- NUTS uses a criteria to stop sampling when the chain goes downhill, and then takes a random sample from the path
- This is implemented in STAN

Reversible jump MCMC

- Say there are J possible models: $\mathcal{M}_1, ..., \mathcal{M}_J$

- Example, $\mathcal{M}_1$ is a multiple linear regression model and $\mathcal{M}_2$ is a neural network

- Let θ_j denote the collection of parameters in $\mathcal{M}_j$

- The θ_j need not have the same dimension or interpretation across models

- RJMCMC computes posterior draws of the model $\mathcal{M} \in \{\mathcal{M}_1, ..., \mathcal{M}_J\}$ and the model parameters

Reversible jump MCMC

- It alternates between updating the parameters within a model and the model
- The complicated step is updating the model, say $j \in \{1, ..., J\}$
- In the Metropolis-Hastings step, you propose to move from model j to model k
- You have the current value of θ_j, but you need to propose a candidate for θ_k
- This step is difficult to tune, and the acceptance probability has a complicated form

Summary

- With the combination of Gibbs and Metropolis-Hastings sampling we can fit virtually any model
- In some cases Bayesian computing is actually preferable to maximum likelihood analysis
- In most cases Bayesian computing is slower
- However, in the opinion of many it is worth the wait for improved uncertainty quantification and interpretability
- In all cases it is important to carefully monitor convergence

Options for coding MCMC

- Writing your own code
- Bayesian options in SAS procedures
- R packages for specific models
- All-purpose software like JAGS, BUGS, PROC MCMC, and STAN

Bayes in SAS procedures and R functions

- Here is a SAS proc

```
proc phreg data=VALung;
    class PTherapy(ref=‘no‘) Cell(ref=‘large‘)
    Therapy(ref=‘standard‘);
    model Time*Status(0) = KPS Duration;
    bayes seed=1 outpost=cout coeffprior=uniform
    plots=density;
run;
```

- In `R` you can use `BLR` for linear regression, `MCMClogit` for logistic regression, etc.

Why Just Another Gibbs Sampler (JAGS)?

- You can fit virtually any model

- You can call JAGS from `R` which allows for plotting and data manipulation in `R`

- It runs on all platforms: LINUX, Mac, Windows

- There is a lot of help online

- R has many built in packages for convergence diagnostics

How does JAGS work?

- You specify the model by declaring the likelihood and priors
- JAGS then sets up the MCMC sampler, e.g., works out the full conditional distributions for all parameters
- It returns MCMC samples in a matrix or array
- It also automatically produces posterior summaries like means, credible sets, and convergence diagnostics
- User's manual: `http://blue.for.msu.edu/CSTAT_13/jags_user_manual.pdf`

Running JAGS from `R` has the following steps

1. Install JAGS: `https://sourceforge.net/projects/mcmc-jags/files/JAGS/4.x/Windows/`
2. Download `rjags` from CRAN and load the library
3. Specify the model as a string
4. Compile the model using the function `jags.model`
5. Draw burn-in samples using the function `update`
6. Draw posterior samples using the function `coda.samples`
7. Inspect the results using the `plot` and `summary` functions

Examples

- The course website has many example of Bayesian analyses using JAGS
- There are also comparisons with other software
- For moderately-sized problems JAGS is competitive with these methods
- For really big and/or complex analyses STAN is preferred
- JAGS is easier to code and so we will use it through the course, but you should be familiar with other software
- Once you understand JAGS, switching to the others is straightforward

Tuning the MCMC algoritm

- MCMC is beautiful because it can handle virtually any statistical model and it is usually pretty easy to write functional code
- However, for hard problems great care must be taken to ensure that the algorithm has converged
- There are three main decisions:
  - Selecting the initial values
  - Determining if/when the chain(s) has converged
  - Selecting the number of samples needed to approximate the posterior

Initial values

- The algorithm will eventually converge no matter what initial values you select
- However taking time to select good initial values will speed up convergence
- It is important to try a few initial values to verify they all give the same result
- Usually 3-5 separate chains is sufficient
- **Option 1**: Select good initial values using method of moments or MLE
- **Option 2**: Purposely pick bad but different initial values for each chain to check convergence

Convergence

- The first few samples are probably not draws from the posterior distribution

- It can take hundreds or even thousands of iterations to move from the initial values to the posterior

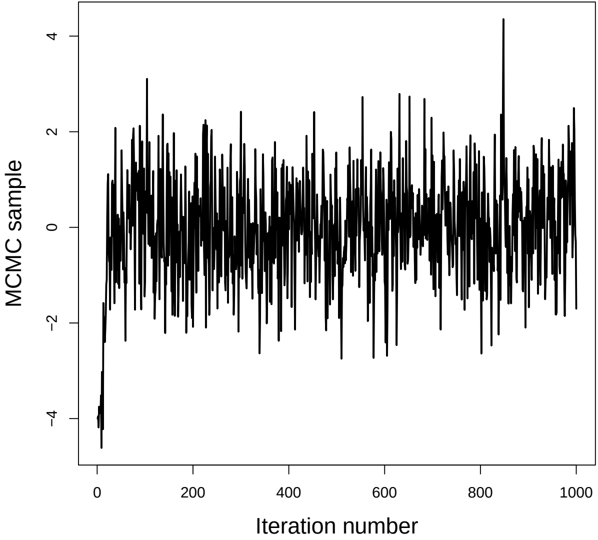
- When the sampler reaches the posterior this is called convergence

- Samples before convergence are discard as **burn-in**

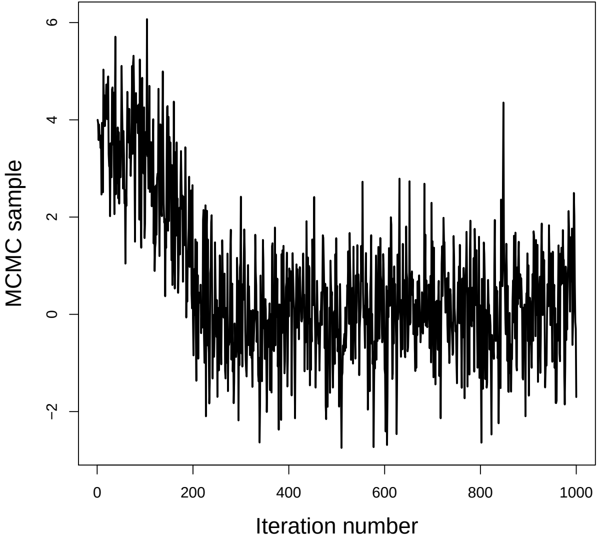
- After convergence the samples should not converge to a single point!

- They should be draws from the posterior, and ideally look like a caterpillar or bar code

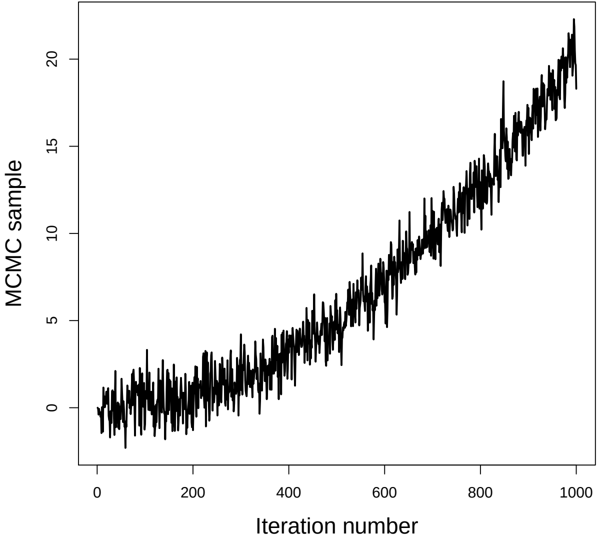
Convergence in a few iterations



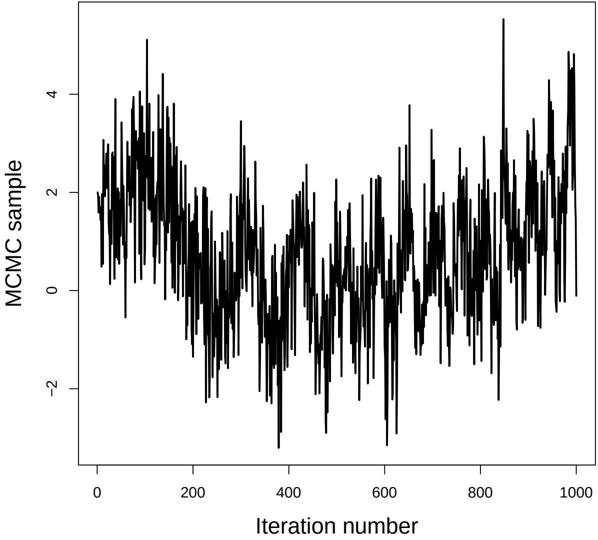
Convergence in a few hundred iterations



This one never converged



Convergence is questionable



Convergence diagnostics

- So far we have visually inspected the chains for convergence
- There are many formal diagnostics
- The `CODA` package in `R` has dozens of diagnostics
- Most give a measure of convergence for each parameter
- Checking convergence using these one-number summaries is more efficient and objective than visual inspection

Convergence diagnostics

- Did my chains converge?
  - Geweke
  - Gelman-Rubin
- Did I run the sampler long enough after convergence?
  - Effective sample size
  - Standard errors for the posterior mean estimate

Examples

- The JAGS function `coda.samples` returns sample is the format that can be passed to the `CODA` function which actually computes the diagnostics

- The course website uses CODA to access convergence for a best-case and a worst-case scenario

Geweke diagnostic

- Compares the mean in the beginning of the chain with the mean at the end of the chain
- Can we used for a single chain
- Done separately for each parameter
- The JAGS default is to compare the first 10% with the last 50%
- The test accounts for autocorrelation
- The test statistic is a z-score, so $|Z| > 2$ indicates poor convergence

Gelman-Rubin statistic

- If we run multiple chains, we hope that all chains give same result
- The Gelman-Rubin statistics measures agreement between chains
- Is it essentially an ANOVA test of whether the chains have the same mean
- It is scaled so that 1 is perfect and 1.1 is decent but not great convergence
- JAGS plots the statistic over iteration
- When the statistic reaches one this indicates convergence

Autocorrelation

- Ideally the samples would be independent across iteration
- The autocorrelation function $\rho(h)$ is the correlation between samples h iterations apart
- JAGS plots the autocorrelation as a function of h
- Lower values are better, but if the chains are long enough even large values can be OK
- **Thinning**: If autocorrelation is zero after lag h you can thin the samples by h to achieve independence
- This is always less efficient than using all samples, but can save memory

Effective sample size

- Highly correlated samples have less information than independent samples
- Say S is the actual number of MCMC samples
- The **effective samples size** is

$$ESS = \frac{S}{1 + 2\sum_{h=1}^{\infty} \rho(h)}$$

- The correlated MCMC sample of length S has the same information as ESS independent samples
- ESS should be at least a few thousand for all parameters

Standard errors of posterior mean estimates

- The sample mean of the MCMC draws is an estimate of the posterior mean

- The standard error of this estimate as another diagnostic

- Assuming independence the standard error is

  $$\text{Naive SE} = \frac{s}{\sqrt{S}}$$

  where s is the sample SD and S is the number of samples

- A more realistic standard error is

  $$\text{Times-series SE} = \frac{s}{\sqrt{ESS}}$$

What to do if the chains haven’t converged?

- Determining if chains have converged is not that difficult
- Improving converge is challenging
- We will discuss options in lab
- Hopefully we can get a list of 10 or so

Outline

- Deterministic methods
  - MAP estimation
  - Numerical integration
  - Bayesian CLT
  - INLA
- Markov Chain Monte Carlo
  - Gibbs sampling
  - Slice sampling
  - Metropolis-Hastings sampling
  - Hamiltonian Monte Carlo
  - JAGS
  - Convergence diagnostics
- **ABC**

Approximate Bayesian Computing (ABC)

- ABC is a clever trick for models from which it is easy to simulate data but the likelihood is cumbersome
- For example, the SIR compartmental model involves differential equation and so the likelihood is complicated
- ABC provides an approximate solution in this case
- It generally works well when model is easy to simulate from and has a small number of parameters

Approximate Bayesian Computing (ABC)

Here is an exact way to sample from the posterior:

1. Sample candidate θ^* from the prior

2. Simulate a dataset $\mathbf{Y}^*$ given θ^* of the same dimension of $\mathbf{Y}$

3. If $\mathbf{Y}^* = \mathbf{Y}$, retain the draw of θ, otherwise return to 1.

4. Repeat until the desired number of sample have been collected

Approximate Bayesian Computing (ABC)

Proof:

Approximate Bayesian Computing (ABC)

- If $\mathbf{Y}$ is continuous, then $\mathbf{Y}^*$ will never equal $\mathbf{Y}$
- Instead you retain the sample if the discrepancy $d(\mathbf{Y}^*, \mathbf{Y})$ is small
- Example: $d(\mathbf{Y}^*, \mathbf{Y}) = \sum_{i=1}^{n}(Y_i^* - Y_i)^2/n$
- Often the discrepancy is a function of sufficient statistics
- Example: $d(\mathbf{Y}^*, \mathbf{Y}) = |\bar{Y}^* - \bar{Y}|$
- Example: $d(\mathbf{Y}^*, \mathbf{Y}) = |\bar{Y}^* - \bar{Y}| + |s^* - s|$

Approximate Bayesian Computing (ABC)

- The proportion of samples retained is small if the discrepancy threshold is small or the prior is diffuse

- This make the method inefficient

- There are adaptive procedures to circumvent this

- You can also combine ABC and MCMC, although this is complicated