MSc HT15. Further Statistical Methods: MCMC

Lecture 7-8: convergence and mixing; Gibbs sampler; Data augmentation and Bayesian probit regression; Slice sampler; RJAGS.

Notes and Practicals available at

www.stats.ox.ac.uk/~nicholls/MScMCMC15
MH example: an equal mixture of bivariate normals

\[
\pi(\theta) = (2\pi)^{-1}\left(0.5e^{-(\theta-\mu_1)^\top\Sigma_1^{-1}(\theta-\mu_1)/2} + 0.5e^{-(\theta-\mu_2)^\top\Sigma_2^{-1}(\theta-\mu_2)/2}\right)
\]

with \(\theta = (\theta_1, \theta_2)\). Use \(\mu_1 = (1, 1)^T\), \(\mu_2 = (5, 5)^T\) and \(\Sigma_1 = \Sigma_2 = I_2\) for this illustration.

Step 1. For a proposal distribution \(q\) we want something simple to sample. The simplest thing I can think of is the same as before:

\[
\theta_i' \sim U(\theta_i - a, \theta_i + a)
\]

with \(a\) a fixed constant. Note that this time we are proposing in a box of side \(2a\). That is easy to sample, and certainly \(q(\theta'|\theta) > 0 \iff q(\theta|\theta') > 0\) since \(q(\theta'|\theta) = q(\theta|\theta') = 1/4a^2\).
Step 2. The algorithm is, given $\theta^{(n)} = \theta$,

[1] for $i = 1, 2$ simulate $\theta_i' \sim U(\theta_i - a, \theta_i + a)$

[2] with probability

$$\alpha(\theta' | \theta) = \min \left\{ 1, \frac{\pi(\theta')}{\pi(\theta)} \right\}$$

set $\theta^{(n+1)} = \theta'$ otherwise set $\theta^{(n+1)} = \theta$.

This algorithm is ergodic for any $a > 0$ but we will see that the choice of $a$ makes a difference to efficiency.
# MCMC simulate \( X_t \) according to a mixture of normals

\[
f(x, \mu_1, \mu_2, S_{1i}, S_{2i}, p_1 = 0.5) = p_1 \cdot \exp(-t(x-\mu_1)'S_{1i}(x-\mu_1)) + (1-p_1) \cdot \exp(-t(x-\mu_2)'S_{2i}(x-\mu_2))
\]

\( a = 3, \ n = 2000 \)

\( \mu_1 = (1, 1), \ \mu_2 = (5, 5), \ S = \text{diag}(2), \ S_{1i} = S_{2i} = \text{solve}(S) \)

\( X = \text{matrix}(NA, 2, n) \)

\[
\text{for } (t \text{ in } 1: (n-1)) \{
\text{y} = x + (2 \cdot \text{runif}(2) - 1) \cdot a
\text{MHR} = f(y, \mu_1, \mu_2, S_{1i}, S_{2i}) / f(x, \mu_1, \mu_2, S_{1i}, S_{2i})
\text{if } (\text{runif}(1) < \text{MHR}) \ x = y
\text{X[, t+1]} = x
\}
\]
(see the associated R-file for plotting commands)
Convergence and mixing

We want to estimate $E_p(f(X))$ using our MCMC samples $X_0, X_1, X_2, \ldots, X_n$ targeting $p(x)$ and calculate the estimate $\bar{f}_n = n^{-1} \sum_t f(X_t)$. The ergodic theorem tells us this estimate converges in probability to $E_p(f(X))$.

How large should we take $n$? There are two issues.

First, suppose $p^{(0)}(x) = p(x)$, so we start the chain in equilibrium. The variance, $\text{var}(\bar{f}_n)$, of $\bar{f}_n$ will get smaller as $n$ increases. We should choose $n$ large enough to ensure $\text{var}(\bar{f}_n)$ is small enough so that $\bar{f}_n$ has useful precision. However, calculating $\text{var}(\bar{f}_n)$ won't be completely straightforward as the MCMC samples are correlated.
Second, we don’t start the chain in equilibrium. The samples in the first part of the chain are biased by the initialization. It is common practice to drop the first part of the MCMC run (called “burn-in”) to reduce the initialization bias. We know \( p(t) \to p \) as \( t \to \infty \) and want to choose a cut-off \( T \) beyond which \( p(t) \simeq p \) to a good approximation. We need \( n \gg T \) so that most of the samples are representative of \( p \).

Note that if \( n \gg T \) then the bias from burn-in will be slight anyway. One observation here is that if you need to drop states from the start of the chain to reduce this bias, you probably haven’t run the chain long enough.

The following figures show autocorrelations for two MCMC runs of the \( \text{N}(0,1) \) sampler above, with different values of the jump size \( a = 0.5, 3 \).
MCMC variance in equilibrium

\(X_0, X_1, X_2, \ldots\) are correlated so \(\text{var}(\bar{f}_n) \neq \frac{\text{var}(f(X))}{n}\) in general.

Correlation at lag \(s\)

\[
\rho_s^{(f)} = \frac{\text{cov}(f(X_i), f(X_{i+s}))}{\text{var}(f(X_i))}
\]

(so \(\rho_0 = 1\)). Let \(\sigma^2 = \text{var}(f(X_i))\). This doesn’t depend on \(i\) because the chain is stationary, because it was started in equilibrium.

Express \(\text{var}(\bar{f}_n)\) in terms of \(\rho_s^{(f)}\). This gives insight and leads to an estimator for \(\text{var}(\bar{f}_n)\), since we can estimate \(\rho_s^{(f)}\).
\[
\text{var}(\overline{f}) = n^{-2} \sum_{i=1}^{n} \sum_{j=1}^{n} \text{cov}(f(X_i), f(X_j)) \\
= \sigma^2 n^{-2} \sum_{i=1}^{n} \sum_{j=1}^{n} \rho_{|i-j|} \\
= \sigma^2 n^{-1} \left[ 1 + 2 \sum_{s=1}^{n-1} \left( 1 - \frac{s}{n} \right) \rho_s \right] \\
\approx \sigma^2 n^{-1} \left[ 1 + 2 \sum_{s=1}^{n-1} \rho_s \right] \\
= \sigma^2 \tau_f / n,
\]

if as usual \( \rho_s \) is small when \( s \) is large. \( \tau_f \) is called the integrated autocorrelation time. The quantity \( \text{ESS} = n / \tau_f \) is called the effective sample size - the number of independent samples that would give the same precision for \( \overline{f} \) as the \( n \) correlated samples we actually have.
We can estimate $\gamma_s = \text{cov}(f(X_i), f(X_{i+s}))$ using

$$\hat{\gamma}_s = \frac{1}{n} \sum_{i=1}^{n-s} (f(X_i) - \hat{f})(f(X_{i+s}) - \hat{f}),$$

and $\gamma_0 = \text{var}(f(X_i))$ (as usual) from the sample output, and compute $\hat{\rho}_s = \hat{\gamma}_s / \hat{\gamma}_0$.

We get an estimate of $\tau_f$,

$$\hat{\tau}_f = 1 + 2 \sum_{s=1}^{M} \hat{\rho}_s,$$

with $M$ a cut-off on the sum. $\hat{\rho}_s$ goes to zero with $s$ and is dominated by noise at large $s$; beyond some value of $s$ we are actually making our estimate worse by adding more terms to form the sum of $\hat{\rho}_s$ over $s$. We have to truncate the sum over $s$, at $s = M$. 
MCMC convergence

There is no simple generic sufficient condition we can test for convergence. Here some checks we can run to detect poor mixing and identify a burn-in and run length.

[1] Make multiple runs from different start states and check marginal distributions agree.

[2] Plot the autocorrelation function. Check that it falls off to vary around zero. Calculate the ESS and check it is reasonably large.

[3] Plot MCMC traces of the variables and key functions. The chain should be stationary after burn-in.
Here is an example of some of the plots I would use for convergence checking on the $N(x; 0, 1)$ MCMC sampler.

See associated R-file for further examples.
The Gibbs sampler

The Gibbs sampler is particularly natural. It works well if we can easily sample conditional distributions. Suppose for eg $\pi(\theta_1, \theta_2)$ is a bivariate density we want to sample and we have $(\theta_1, \theta_2) \sim \pi$.

Simulate a new $\theta'_1 \sim \pi(\theta'_1|\theta_2)$ and then simulate $\theta'_2 \sim \pi(\theta'_2|\theta'_1)$ (using the new $\theta'_1$). The distribution of $(\theta'_1, \theta'_2)$ is

$$p(\theta'_1, \theta'_2) = \int \pi(\theta_1, \theta_2)\pi(\theta'_1|\theta_2)\pi(\theta'_2|\theta'_1)d\theta_1d\theta_2$$

$$= \int \pi(\theta_1, \theta_2)\frac{\pi(\theta'_1, \theta_2)}{\pi(\theta'_1)}\frac{\pi(\theta'_1, \theta'_2)}{\pi(\theta'_1)}d\theta_1d\theta_2$$

$$= \int \pi(\theta_1|\theta_2)\pi(\theta_2|\theta'_1)\pi(\theta'_1, \theta'_2)d\theta_1d\theta_2$$

$$= \pi(\theta'_1, \theta'_2)\int \pi(\theta_1|\theta_2)d\theta_1\int \pi(\theta_2|\theta'_1)d\theta_2$$

$$= \pi(\theta'_2, \theta'_1)$$ since the integrals are each equal one.
so if we start with \((\theta_1, \theta_2) \sim \pi\) then after these two steps we have a new correlated sample \((\theta'_1, \theta'_2) \sim \pi\). If \(\theta^{(0)} = (\theta_1, \theta_2)\), then \(\theta^{(1)} = (\theta'_1, \theta'_2)\) and we can iterate to simulate \(\theta^{(2)}, \theta^{(3)}\)...

The Gibbs sampler comes in several flavors. In the multivariate case \(\theta = (\theta_1, \theta_2, ..., \theta_K)\) we can sample \(\pi(\theta_i|\theta_{-i})\) for \(i = 1, 2, 3, ..., K\) in turn, or select \(i \sim U\{1, 2, 3, ..., K\}\) at each step. We need to check the ergodicity conditions when we construct a new MCMC sampler.

Exercise: consider a Metropolis Hastings algorithm targeting \(\pi(\theta_1, \theta_2)\) with proposal \(q(\theta'_1|\theta) = \pi(\theta'_1|\theta_2)\). Show that the acceptance probability is equal one. Hence show that the Gibbs sampler is a MH sampler.
Data Augmentation

Some important early applications of the Gibbs sampler arise in the context of missing data. This is also called “data augmentation”.

DA is convenient when the likelihood on the full data is much simpler than the likelihood on the observed data.

Suppose the observation process is \( z \sim p(z|\theta), \ y \sim p(y|z, \theta) \) and we observe \( y \). The posterior \( p(\theta|y) \) is awkward as the likelihood is an integral,

\[
L(\theta; y) = \int p(y|z, \theta)p(z|\theta)dz
\]
In data augmentation we work with the joint posterior density
\( p(\theta, z|y) \), thinking of the missing data as another parameter.
The likelihood is
\[
p(\theta, z|y) \propto p(y|z, \theta)p(z|\theta)p(\theta)
\]
The original DA algorithm was a Gibbs sampler

[1] \( z' \sim p(z'|\theta, y) \) with \( p(z'|\theta, y) \propto p(y|z', \theta)p(z'|\theta) \)
[2] \( \theta' \sim p(\theta'|y, z') \)
Example of DA: Probit regression

In probit regression we have covariates \( x = (x_1, ..., x_p) \), parameters \( \theta = (\theta_1, \theta_2, ..., \theta_p) \), a linear predictor \( \eta = \sum_i \theta_i x_i \), and observation model \( y \sim Bernoulli(\Phi(\eta)) \) with \( \Phi \) the cdf of a standard normal.

There is another way to represent model. Let \( z \sim N(\eta, 1) \) be a scalar normal rv. Set \( y = 1 \) if \( z > 0 \) and \( y = 0 \) if \( z \leq 0 \). Now \( \Pr(y = 1) = \Pr(z > 0) = \Pr(\eta + W > 0) \) with \( W \) a standard normal. Now \( \Pr(\eta + W > 0) = \Pr(W > -\eta) \). Since a standard normal is symmetrical, \( \Pr(W > -\eta) = \Pr(W < \eta) \) which is \( \Phi(\eta) \).

In this representation we have a latent 'propensity' score \( z \) for each observation \( y \), and we effectively observe the sign of \( z \).
Suppose we are doing Bayesian inference for $\theta$ with normal priors. The joint posterior is

$$p(\theta, z | y) \propto p(y | z, \theta) p(z | \theta) p(\theta)$$

Now $p(y | z, \theta)$ is 0/1 as $z$ agrees/disagrees with $y$ and $p(z | \theta)$ is $N(\eta, 1)$. Because $p(y | z, \theta) = p(y | z)$ here, the conditional distribution of $\theta$ is $p(\theta | y, z) \propto p(z | \theta) p(\theta)$ which is $p(\theta | z)$. If $p(\theta)$ is normal, it is conjugate to $p(z | \theta)$ and $\theta | y, z$ is also normal for each component.

Our DA/Gibbs sampler becomes

[1] simulate $z'$ from $N(\eta, 1)$ conditioned on the sign of $z$ ($+/-$ as $y = 1/0$)
[2] simulate $\theta' \sim p(\theta | z')$.

(see R code example)
Slice sampler: a last piece of theory. This is another DA sampler. With the standard Gibbs sampler and certain forms of rejection, this is one of the common generic samplers in use in code like JAGS. Here is the idea for a one-D problem.

Say we want to sample $\pi(\theta), \theta \in \Omega$. Consider the joint distribution $\pi(\theta, u)$ uniform on the set of points “under the graph” of $\pi(\theta)$ - this has area equal one ($\pi$ is a probability density) so

$$
\pi(\theta, u) = \begin{cases} 
1 & \text{if } \theta \in \Omega \text{ and } 0 < u < \pi(\theta) \\
0 & \text{otherwise.}
\end{cases}
$$

Notice that $\int \pi(\theta, u)du = \int_0^{\pi(\theta)} 1du = \pi(\theta)$ so we have not messed up the distribution of $\theta$ by introducing $u$: this Data Augmentation creates an artificial piece of missing data $u$. 
To sample $\pi(\theta, u)$ we alternate between

[1] sampling $u' \sim U(0, \pi(\theta))$ at fixed $\theta$ and then

[2] sampling $\theta \sim U(A_u)$ where $A_u = \{\theta : \pi(\theta) > u\}$.

Notice we only ever have to sample uniformly. It can be a bit tricky to sample uniformly from $A$. Rejection sampling from a set $A$ that covers $A_u$ is one simple approach. At [2] we keep trying random values from $A$ until we get one in $A_u$ (so sample $\theta \sim U(A)$ till $f(\theta) > u$).

http://www.probability.ca/jeff/java/slice.html has a nice java illustration of the slice sampler.
When we do Bayesian inference we specify a prior and likelihood. That is the modeling phase. In the inference phase we compute summary and test statistics (in order to estimate parameters, compare models, test hypotheses, summarize posterior distribution). If we had a set of samples from the posterior we could do alot of this inference straightforwardly.

JAGS (Martyn Plummer 2003, mcmc-jags.sourceforge.net/) takes a specification of the prior and likelihood and returns a set of samples from the posterior.

RJAGS is an R interface to JAGS (an R package). We specify the posterior in a file, and call JAGS to run the file from R. The
samples generated by JAGS MCMC are returned to R where we can process them.

This support for Bayesian inference was first seen BUGS. The 'language' we use to specify prior and likelihood comes from BUGS (with small variations). We will learn this language from examples. For a more structured introduction see Appendix A in Prof Ripley's lecture notes

http://www.stats.ox.ac.uk/~nicholls/MScMCMC14/MCMC.pdf

and references cited there.
Examples: Normal and Poisson regression As an example we will fit normal and Poisson models to the puffin data.

> library(LearnBayes)
> puffin[1:5,]
  Nest Grass Soil Angle Distance
  1  16   45  39.2  38    3
  2  15   65  47.0  36   12
  3  10   40  24.3  14   18
  4   7   20  30.0  16   21
  5  11   40  47.6   6   27

Consider a Bayesian analysis with response $y = \text{Nest counts}$ and the normal linear model $\text{Nest} \sim \text{Grass} + \text{Soil} + \text{Angle} + \text{Distance}$ with diffuse normal priors on regression parameters and a diffuse prior on standard deviation.
library(MCMCpack)
Bfit <- MCMCregress(Nest ~ Grass + Soil + Angle + Distance, data = puffin, burnin = 1000, mcmc = 25000, thin = 25)

will fit
\[ \eta = x\beta, \beta = (\beta_0, ..., \beta_4), \]
\[ y \sim N(\eta, \sigma^2), \]
\[ \beta_i \sim N(b0, B0^{-1}), i = 0, 2, ..., 4, \]
\[ \sigma \sim \Gamma(c0/2, d0/2). \]

MCMCregress(formula, data = NULL, burnin = 1000, mcmc=10000, thin = 1, verbose = 0, seed = NA, beta.start = NA, b0 = 0, B0 = 0, c0 = 0.001, d0 = 0.001, marginal.likelihood = c("none", "Laplace", "Chib95"), ...)

No choice of priors beyond mean and variance.
Suppose we would like to fit the normal linear model above with prior $\sigma \sim U(0, 10)$. Try JAGS. The model spec

```r
model{
    for(i in 1:38) {
        Nest[i] ~ dnorm(mu[i], sigma^-2)
        mu[i] <- beta0 + beta1*Grass[i] + beta2*Soil[i] + beta3*Angle[i] + beta4*Distance[i]
    }
    beta0 ~ dnorm(0, 0.01)
    beta1 ~ dnorm(0, 0.01); beta2 ~ dnorm(0, 0.01)
    beta3 ~ dnorm(0, 0.01); beta4 ~ dnorm(0, 0.01)
    sigma ~ dunif(0, 10)
}
```

goes in a file called `puffin.bug`, and we run this with
library(rjags)
load.module("glm")  # faster algorithm
#start state initialisation
inits <- function()
list(beta0 = 0, beta1 = 0, beta2 = 0, beta3 = 0,
     beta4 = 0, sigma = 1)
#generate the model
p.jags <- jags.model("puffin.bug", data = puffin,
                     inits = inits, n.chain = 3)
#run the MCMC, monitoring vars
vars <- c("beta1", "beta2", "beta3", "beta4", "sigma")
p1.sims <- coda.samples(p.jags, vars, n.iter = 100)
plot(p1.sims)
A denser style is useful for large models/data. In the .bugs file

```plaintext
model{
    for(i in 1:38) { Nest[i] ~ dnorm(mu[i], sigma^-2) }
    mu <- X %*% beta
    for(i in 1:5) { beta[i] ~ dnorm(0, 0.01) }
    sigma ~ dunif(0, 10)
}

and call

#denser style for big models
X <- model.matrix(~ Grass + Soil + Angle + Distance, 
                 data = puffin)
init <- list(list(beta = rep(0, 5), sigma = 1))
```
p2.jags <- jags.model("puffin2.bug",
    data = list(Nest = puffin$Nest, X = X),
    inits = inits, n.chain = 1)

#run the MCMC, monitoring vars
vars <- c("beta", "sigma")
p2.sims <- coda.samples(p2.jags, vars, n.iter = 1000)
Try Poisson regression. Frequentist is

```r
pfit <- glm(Nest ~ Grass + Soil + Angle + Distance, 
            poisson, data = puffin)
summary(pfit)
```

We see over-dispersion (try quasipoisson) and 13/38 zero counts in data (and none of one or two). Ignore that for now for illustration of method.

In order to fit

\[ y \sim \text{Poisson}(\exp(\eta)), \text{ with} \]
\[ \eta = x\beta, \quad \beta = (\beta_0, \ldots, \beta_4) \text{ and,} \]
\[ \beta_i \sim N(b_0, B_0^{-1}), \quad i = 0, 2, \ldots, 4 \]

we can use MCMCpack
Bpfit <- MCMCpoisson(Nest ~ Grass + Soil + Angle + Distance, data = puffin, burnin = 1000, mcmc = 25000, thin = 25)
summary(Bpfit)

We really need a better model (eg negative binomial). Package MCMCpack doesn't cover that, but JAGS is considerably more flexible.

For this Poisson regression in JAGS the model in puffin3.bug is

```r
model{
  for(i in 1:38) { Nest[i] ~ dpois(exp(eta[i])) }
  eta <- X %*% beta
  for(i in 1:5) { beta[i] ~ dnorm(0, 0.01) }
}
```
run by

```r
inits <- function() list(beta = c(2, rep(0, 4)))
p3.jags <- jags.model("puffin3.bug", data = list(Nest = puffin$Nest), inits = inits, n.chain = 3)
p3.sims <- coda.samples(p3.jags, "beta", n.iter = 1000)
summary(p3.sims)
```
We could move to negative binomial regression. Classically (Venables & Ripley, 2002, x7.4):

```r
library(MASS)
nbfit <- glm.nb(Nest ~ Grass + Soil + Angle + Distance, data)
maxit = 50)
summary(nbfit)
```
The JAGS model is

```r
model{
  for(i in 1:38) {
    Nest[i] ~ dnegbin(p[i], theta)
    lambda[i] <- exp(eta[i])
    p[i] <- theta/(theta + lambda[i])
  }
  eta <- X %*% beta
  for(i in 1:5) { beta[i] ~ dnorm(0, 0.01) }
  theta ~ dunif(1, 50) # a guess, the classical fit is about 9
}
```