STAT 572: Project

Chapter 10: Nonconjugate priors and Metropolis-Hastings algorithms

Geovani Bautista, Simon Lee, Shen Qu

Dec, 2019

Appendix

10.1 Generalized linear models

Example: Song sparrow reproductive success

```
#### Grid-based posterior approximation
p<-3
beta0<-rep(0,p)
S0<-diag( rep(100,3))
gs<-100
LPB<-array(0,dim=rep(gs,p))

beta1<-seq(.27-1.75,.27+1.75,length=gs)
beta2<-seq(.68-1.5,.68+1.5,length=gs)
beta3<-seq(-.13-.25,-.13+.25,length=gs)

beta1<-seq(.27-2.5,.27+2.5,length=gs)
beta2<-seq(.68-2,.68+2,length=gs)
beta3<-seq(-.13-.5,-.13+.5,length=gs)

for(i in 1:gs) { for(j in 1:gs) { for(k in 1:gs) {
  theta<-beta1[i]+beta2[j]*age+beta3[k]*age^2
  LPB[i,j,k]<-dnorm(beta1[i],beta0[1],sqrt(S0[1,1]),log=TRUE)  +
              dnorm(beta2[j],beta0[2],sqrt(S0[2,2]),log=TRUE)  +
              dnorm(beta3[k],beta0[3],sqrt(S0[3,3]),log=TRUE)  +
              sum( dpois(fledged,exp(theta),log=TRUE )  )
  }}
  }
cat(i,"\n")
```

```
## 100
```

```
PB<-exp( LPB - max(LPB) )
PB<-PB/sum(PB)

PB1<-apply(PB,1,sum)
PB2<-apply(PB,2,sum)
PB3<-apply(PB,3,sum)
PB23<-apply(PB,c(2,3),sum)

## Simulation from grid approximation
```

```
S<-50000
BETAg<-matrix(nrow=S,ncol=3)
for(s in 1:S) {
i<-sample(1:gs,1,prob=PB2)
j<-sample(1:gs,1,prob=PB23[i,] )
k<-sample(1:gs,1,prob=PB[,i,j] )
BETAg[s,]<-c(beta1[k],beta2[i],beta3[j])  }
```

10.2 The Metropolis algorithm

Algorithm 1: The Metropolis algorithm

Data: $Y \sim p(\theta)$

Result: generates $\theta^{(1)}, ..., \theta^{(s)} \sim$ iid $p(\theta|y)$

initialization;

Choose δ to make the approximation algorithm run efficiently;

while *not convergentcy* **do**

for *Symmetric:* $J(\theta_b|\theta_a) = J(\theta_a|\theta_b)$ **do**

1. Sample $\theta^\star \sim J(\theta|\theta^{(s)})$; such as ;

$$J(\theta^\star|\theta^{(s)}) = \text{uniform}(\theta^{(s)} - \delta, \theta^{(s)} + \delta)$$

OR

$$J(\theta^\star|\theta^{(s)}) = \text{normal}(\theta^{(s)}, \delta^2)$$

2. Compute the acceptance ratio

$$r = \frac{p(\theta^\star|y)}{p(\theta^{(s)}|y)} = \frac{p(y|\theta^\star)p(\theta^\star)}{p(y|\theta^{(s)})p(\theta^{(s)})}$$

3. Sampling $u \sim \text{uniform}(0, 1)$;

if *the ratio* $r > 1, (u)$ **then**

$\theta^{(s+1)} \longleftarrow \theta^\star$ with probability $\min(r, 1)$;

if *the ratio* $r < 1, (u)$ **then**

$\theta^{(s+1)} \longleftarrow \theta^{(s)}$ with probability $1 - \min(r, 1)$;

generates a value $\theta^{(s+1)}$ given $\theta^{(s)}$;

```
#### MH algorithm for one-sample normal problem with
## Setup
s2<-1
t2<-10 ; mu<-5

set.seed(1)
n<-5
y<-round(rnorm(n,10,1),2)

mu.n<-( mean(y)*n/s2 + mu/t2 )/( n/s2+1/t2)
t2.n<-1/(n/s2+1/t2)

## MCMC
s2<-1 ; t2<-10 ; mu<-5
y<-c(9.37, 10.18, 9.16, 11.60, 10.33)
```

```r
theta<-0 ; delta<-2 ; S<-10000 ; THETA<-NULL ; set.seed(1)

for(s in 1:S)
{
  theta.star<-rnorm(1,theta,sqrt(delta))

  log.r<-( sum(dnorm(y,theta.star,sqrt(s2),log=TRUE)) +
              dnorm(theta.star,mu,sqrt(t2),log=TRUE) )  -
         ( sum(dnorm(y,theta,sqrt(s2),log=TRUE)) +
              dnorm(theta,mu,sqrt(t2),log=TRUE) )
  if(log(runif(1))<log.r) { theta<-theta.star }

  THETA<-c(THETA,theta)
}
```

```r
#### MH algorithm with different proposal distributions
par(mfrow=c(2,3))
ACR<-ACF<-NULL
THETAA<-NULL
for(delta2 in 2^c(-5,-1,1,5,7) ) {
set.seed(1)
THETA<-NULL
S<-10000
theta<-0
acs<-0
delta<-2

for(s in 1:S)
{
  theta.star<-rnorm(1,theta,sqrt(delta2))
  log.r<-sum( dnorm(y,theta.star,sqrt(s2),log=TRUE)-
             dnorm(y,theta,sqrt(s2),log=TRUE)  )  +
      dnorm(theta.star,mu,sqrt(t2),log=TRUE)-dnorm(theta,mu,sqrt(t2),log=TRUE)
  if(log(runif(1))<log.r)  { theta<-theta.star ; acs<-acs+1 }
  THETA<-c(THETA,theta)

}
# plot(THETA[1:1000],col = alpha("blue", 0.1))
ACR<-c(ACR,acs/s)
ACF<-c(ACF,acf(THETA,plot=FALSE)$acf[2]  )
THETAA<-cbind(THETAA,THETA)
}
# plot(ACR,ACF) ; lines(ACR,ACF)
```

```r
THCM<-apply(THETAA,2,cumsum)
THCM<- THCM/(1:dim(THCM)[1])
#### Back to sparrow data
fit.mle<-glm(fledged~age+age2,family="poisson")
# summary(fit.mle)

y<-fledged ; X<-cbind(rep(1,length(y)),age,age^2)
yX<-cbind(y,X)
colnames(yX)<-c("fledged","intercept","age","age2")
```

```r
n<-length(y) ; p<-dim(X)[2]

pmn.beta<-rep(0,p)
psd.beta<-rep(10,p)

var.prop<- var(log(y+1/2))*solve( t(X)%*%X )
beta<-rep(0,p)
S<-10000
BETA<-matrix(0,nrow=S,ncol=p)
ac<-0
set.seed(1)

## rmvnorm function for proposals
rmvnorm<-function(n,mu,Sigma)
{ # samples from the multivariate normal distribution
  E<-matrix(rnorm(n*length(mu)),n,length(mu))
  t(  t(E%*%chol(Sigma)) +c(mu))
}

## MCMC
for(s in 1:S) {
#propose a new beta
beta.p<- t(rmvnorm(1, beta, var.prop ))

lhr<- sum(dpois(y,exp(X%*%beta.p),log=T)) -
      sum(dpois(y,exp(X%*%beta),log=T)) +
      sum(dnorm(beta.p,pmn.beta,psd.beta,log=T)) -
      sum(dnorm(beta,pmn.beta,psd.beta,log=T))

if( log(runif(1))< lhr ) { beta<-beta.p ; ac<-ac+1 }

BETA[s,]<-beta
                    }
cat(ac/S,"\n")
```

```
## 0.429
```

```r
library(coda)
apply(BETA,2,effectiveSize)
```

```
## [1] 818 778 726
```

10.4 Metropolis, Metropolis-Hastings and Gibbs

10.4.1 The Metropolis-Hastings algorithm

Algorithm 2: The Metropolis-Hastings algorithm

Data: $Y \sim p(u, v)$
Result: $p_0(u, v)$ such as $= p(\theta, \sigma^2|y)$
initialization;
while *not convergentcy* **do**
  for *J_u and J_v are separate symmetric proposal distributions for U and V* **do**
    1. update U
    a) sample $u^\star \sim J_u(u|u^{(s)})$;
    b) compute

$$r = \frac{p_0(u^\star, v^{(s)})}{p_0(u^{(s)}, v^{(s)})}$$

    c) set **if** *the ratio $r > 1$* **then**
      $u^{(s+1)} \longleftarrow u^\star$ with probability $\min(1, r)$
    else
      $u^{(s+1)} \longleftarrow u^{(s)}$ with probability $\max(0, 1 - r)$
    2. update V a) sample $v^\star \sim J_v(v|v^{(s)})$;
    b) compute

$$r = \frac{p_0(u^{(s+1)}, v^\star)}{p_0(u^{(s+1)}, v^{(s)})}$$

    c) set **if** *the ratio $r > 1$* **then**
      $v^{(s+1)} \longleftarrow v^\star$ with probability $\min(1, r)$
    else
      $v^{(s+1)} \longleftarrow v^{(s)}$ with probability $\max(0, 1 - r)$

Algorithm 3: The M-H algorithm for approximating $p_0(u, v)$

Data: $Y \sim p(u, v)$
Result: $p_0(u, v)$ such as $= p(\theta, \sigma^2|y)$
initialization;
while *not convergentcy* **do**
  for *J_u, J_v do not depend on U and V values in the sequence previous to the most current values (ensures it is a Markov chain)* **do**
    1. Update U
    a) Sample $u^\star \sim J_u(u|u^{(s)}, v^{(s)})$;
    b) Compute

$$r = \frac{p_0(u^\star, v^{(s)})}{p_0(u^{(s)}, v^{(s)})} \times \frac{J_u(u^{(s)}|u^\star, v^{(s)})}{J_u(u^\star|u^{(s)}, v^{(s)})}$$

    c) Set **if** *the ratio $r > 1$* **then**
      $u^{(s+1)} \longleftarrow u^\star$ with probability $\min(1, r)$
    else
      $u^{(s+1)} \longleftarrow u^{(s)}$ with probability $\max(0, 1 - r)$
    2. Update V a) Sample $v^\star \sim J_v(v|u^{(s+1)}, v^{(s)})$;
    b) Compute the acceptance ratio

$$r = \frac{p_0(u^{(s+1)}, v^\star)}{p_0(u^{(s+1)}, v^{(s)})} \times \frac{J_v(v^{(s)}|u^{(s+1)}, v^\star)}{J_v(v^\star|u^{(s+1)}, v^{(s)})}$$

    c) Set **if** *the ratio $r > 1$* **then**
      $v^{(s+1)} \longleftarrow v^\star$ with probability $\min(1, r)$
    else
      $v^{(s+1)} \longleftarrow v^{(s)}$ with probability $\max(0, 1 - r)$

10.5 Combining the Metropolis and Gibbs algorithms

Example: Historical CO2 and temperature data

```
lmfit<-lm(icecore$tmp~icecore$co2)
```

```
#### Starting values for MCMC
n<-dim(icecore)[1]
y<-icecore[,3]
X<-cbind(rep(1,n),icecore[,2])
DY<-abs(outer( (1:n),(1:n) ,"-"))

lmfit<-lm(y~-1+X)
beta<-lmfit$coef
s2<-summary(lmfit)$sigma^2
phi<-acf(lmfit$res,plot=FALSE)$acf[2]
nu0<-1 ; s20<-1 ; T0<-diag(1/1000,nrow=2)
```

```
## MCMC - 25000 scans saving every 25th scan
set.seed(1)
S<-25000 ; odens<-S/1000
OUT<-NULL ; ac<-0 ; par(mfrow=c(1,2))
for(s in 1:S)
{

  Cor<-phi^DY  ; iCor<-solve(Cor)
  V.beta<- solve( t(X)%*%iCor%*%X/s2 + T0)
  E.beta<- V.beta%*%( t(X)%*%iCor%*%y/s2  )
  beta<-t(rmvnorm(1,E.beta,V.beta)  )

  s2<-1/rgamma(1,(nu0+n)/2,(nu0*s20+t(y-X%*%beta)%*%iCor%*%(y-X%*%beta)) /2 )

  phi.p<-abs(runif(1,phi-.1,phi+.1))
  phi.p<- min( phi.p, 2-phi.p)
  lr<- -.5*( determinant(phi.p^DY,log=TRUE)$mod -
             determinant(phi^DY,log=TRUE)$mod  +
   sum(diag( (y-X%*%beta)%*%t(y-X%*%beta)%*%(solve(phi.p^DY) -solve(phi^DY)) ) )/s2 )

  if( log(runif(1)) < lr ) { phi<-phi.p ; ac<-ac+1 }

  if(s%%odens==0)
    {
     # cat(s,ac/s,beta,s2,phi,"\n") ;
    OUT<-rbind(OUT,c(beta,s2,phi))
    }
}
```

```
OUT.25000<-OUT
library(coda)
apply(OUT.25000,2,effectiveSize )
```

10.6 Discussion and further references