Package 'gibbs.met'

January 27, 2015

Version 1.1-3

Title Naive Gibbs Sampling with Metropolis Steps

Author Longhai Li <longhai@math.usask.ca>

Maintainer Longhai Li <longhai@math.usask.ca>

Depends R (>= 2.5.1)

Description This package provides two generic functions for performing Markov chain sampling in a naive way for a user-defined target distribution, which involves only continuous variables. The function ``gibbs_met'' performs Gibbs sampling with each 1-dimensional distribution sampled with Metropolis update using Gaussian proposal distribution centered at the previous state. The function ``met_gaussian'' updates the whole state with Metropolis method using independent Gaussian proposal distribution centered at the previous state. The sampling is carried out without considering any special tricks for improving efficiency. This package is aimed at only routine applications of MCMC in moderate-dimensional problems.

License GPL (>= 2)

URL \{}url{http://www.r-project.org\protect\T1\textbraceright, \{}url{http://math.usask.ca/~longhai\protect\T1\textbraceright

Repository CRAN

Date/Publication 2012-10-29 08:58:54

NeedsCompilation no

R topics documented:

gibbs-metropolis *Gibbs sampling with Metropolis steps and multivariate Metropolis sampling*

Description

The function gibbs_met performs Gibbs sampling with each 1-dimensional distribution sampled with Metropolis update using Gaussian proposal distribution centered at the previous state. The function met_gaussian updates the whole state with Metropolis method using independent Gaussian proposal distribution centered at the previous state. The sampling is carried out without considering any special tricks for improving efficiency. The functions are written for routine applications in moderate-dimensional problems.

Usage

```
gibbs_met(log_f,no_var,ini_value,
          iters,iters_per.iter=1,iters_met,stepsizes_met, ...)
met_gaussian(log_f, no_var, ini_value,
             iters,iters_per.iter=1,stepsizes_met, ...)
```

Arguments

log_f	the log of the density function from which one wants to sample.
no_var	the number of variables to be sampled.
ini_value	the initial value.
iters,iters_per.iter	Run iters super-transition, each consisting of iters_per.iter single Markov chain update (using Gibbs sampling or Metropolis sampling). Only the state at the end of each super-transition is saved and returned.
iters_met	iterations of updating each 1-dim conditional distribution with Metropolis method in Gibbs sampling using the function gibbs_met.
stepsizes_met	a vector of length no_var, with stepsizes_met[i] being the standard deviation of Gaussian proposal for updating 'i'th variable, which is used either in Gibbs sampling for sampling from the 'i'th conditional distribution, or used in multivariate Metropolis sampling (using the function met_gaussian) for 'i'th variable.
...	extra arguments needed to compute log_f.

Value

a matrix with dim (iters +1) * no_var is returned, with each row for an iteration

Examples

```
##############################################################################
##     demonstration by sampling from bivariate normal distributions
##############################################################################

## the function computing the log density function of multivariate normal
## x     --- a vector, the p.d.f at x will be computed
## mu    --- the mean vector of multivariate normal distribution
## A     --- the inverse covariance matrix of multivariate normal distribution
log_pdf_mnormal <- function(x, mu, A)
{
    0.5 * (-length(mu)*log(2*pi)+sum(log(svd(A)$d))-sum(t(A*(x-mu))*(x-mu)) )
}

## sampling from a bivariate normal distribution with correlation 0.1,
## both marginal standard deviations 1, mean vector (0,5)
A <- solve(matrix(c(1,0.1,0.1,1),2,2))
mc_mvn <- gibbs_met(log_f=log_pdf_mnormal,no_var=2,
                    ini_value=c(0,0),iters=400,iters_met=2,
                    stepsizes_met=c(0.5,0.5), mu=c(0,5), A = A)

postscript("mc_mvn_lowcor.eps",width=7,height=8,horiz=FALSE)

par(mfrow=c(2,2), oma=c(0,0,1,0))

## looking at the trace of Markov chain in the first 100 iterations
plot(mc_mvn[1:100,1],mc_mvn[1:100,2],type="b",pch=20,
     main="Markov chain trace of both variables")

## looking at the trace of Markov chain for a variable
plot(mc_mvn[,1],type="b",pch=20, main="Markov chain trace of the 1st variable")

## looking at the QQ plot of the samples for a variable
qqnorm(mc_mvn[-(1:50),1],main="Normal QQ plot of the 1st variable")

## looking at the ACF of the samples for a variable
acf(mc_mvn[,1],main="ACF plot of the 1st variable")

title(main="Gibbs sampling for a bivariate normal with correlation 0.1",
      outer=TRUE)

dev.off()

## checking the correlation of samples
cat("The sample correlation is",cor(mc_mvn[-(1:50),1],mc_mvn[-(1:50),2]),"\n")


##############################################################################
##     demonstration by sampling from a mixture bivariate normal distribution
##############################################################################
```

```
## the function computing the log density function of mixture multivariate normal
## x     --- a vector, the p.d.f at x will be computed
## mu1,mu2   --- the mean vectors of multivariate normal distributions
## A1,A2     --- the inverse covariance matrice of multivariate normal distributions
## mixture proportion is 0.5
log_pdf_twonormal <- function(x, mu1, A1, mu2, A2)
{  log_sum_exp(c(log_pdf_mnormal(x,mu1,A1),log_pdf_mnormal(x,mu2,A2))
             )
}

log_sum_exp <- function(lx)
{  ml <- max(lx)
   ml + log(sum(exp(lx-ml)))
}

## set parameters defining a mixture bivariate distribution
A1 <- solve(matrix(c(1,0.1,0.1,1),2,2))
A2 <- solve(matrix(c(1,0.1,0.1,1),2,2))
mu1 <- c(0,0)
mu2 <-c(4,4)

## performing Gibbs sampling
mc_mvn <- gibbs_met(log_f=log_pdf_twonormal,no_var=2,ini_value=c(0,0),
                  iters=400,iters_met=2,stepsizes_met=c(0.5,0.5),
                  mu1=mu1,mu2=mu2,A1=A1,A2=A2)

postscript("mc_mvn_closemix.eps",width=7,height=8,horiz=FALSE)

par(mfrow=c(2,2), oma=c(0,0,2,0))

## looking at the trace of Markov chain in the first 100 iterations
plot(mc_mvn[,1],mc_mvn[,2],type="b",pch=20,
     main="Markov chain trace of both variables")

## looking at the trace of Markov chain for a variable
plot(mc_mvn[,1],type="b",pch=20, main="Markov chain trace of the 1st variable")

## looking at the trace of Markov chain for a variable
plot(mc_mvn[,2],type="b",pch=20, main="Markov chain trace of the 2nd variable")

## looking at the ACF of the samples for a variable
acf(mc_mvn[,1],main="ACF plot of the 1st variable")

title(main="Gibbs sampling for a mixture of two bivariate normal distributions
with locations (0,0) and (4,4)", outer=TRUE)

dev.off()

## checking the correlation of samples
cat("The sample correlation is",cor(mc_mvn[-(1:50),1],mc_mvn[-(1:50),2]),"\n")


################################################################################
```

```
## Sampling from a mixture bivariate normal distribution with Metropolis method
#############################################################################

## set parameters defining a mixture bivariate distribution
A1 <- solve(matrix(c(1,0.1,0.1,1),2,2))
A2 <- solve(matrix(c(1,0.1,0.1,1),2,2))
mu1 <- c(0,0)
mu2 <-c(6,6)

## performing Gibbs sampling
mc_mvn <- met_gaussian(log_f=log_pdf_twonormal,no_var=2,ini_value=c(0,0),
                       iters=400,iters_per.iter=2,stepsizes=c(1,1),
                       mu1=mu1,mu2=mu2,A1=A1,A2=A2)

postscript("mc_mvn_farmix_met.eps",width=7,height=8,horiz=FALSE)

par(mfrow=c(2,2), oma=c(0,0,2,0))

## looking at the trace of Markov chain in the first 100 iterations
plot(mc_mvn[,1],mc_mvn[,2],type="b",pch=20,
     main="Markov chain trace of both variables")

## looking at the trace of Markov chain for a variable
plot(mc_mvn[,1],type="b",pch=20, main="Markov chain trace of the 1st variable")

## looking at the trace of Markov chain for a variable
plot(mc_mvn[,2],type="b",pch=20, main="Markov chain trace of the 2nd variable")

## looking at the ACF of the samples for a variable
acf(mc_mvn[,1],main="ACF plot of the 1st variable")

title(main="Sampling with Metropolis method for a mixture of two bivariate normal
distributions with locations (0,0) and (6,6)", outer=TRUE)

dev.off()

## checking the correlation of samples
cat("The sample correlation is",cor(mc_mvn[-(1:50),1],mc_mvn[-(1:50),2]),"\n")
```

Index