Package 'parallelMCMCcombine'

June 23, 2021

Type Package

Title Combining Subset MCMC Samples to Estimate a Posterior Density

Version 2.0

Date 2021-06-18

Author Alexey Miroshnikov, Erin Conlon

Maintainer Erin Conlon <econlon@umass.edu>

Description See Miroshnikov and Conlon (2014) <doi:10.1371/journal.pone.0108425>. Recent Bayesian Markov chain Monto Carlo (MCMC) methods have been developed for big data sets that are too large to be analyzed using traditional statistical methods. These methods partition the data into non-overlapping subsets, and perform parallel independent Bayesian MCMC analyses on the data subsets, creating independent subposterior samples for each data subset. These independent subposterior samples are combined through four functions in this package, including averaging across subset samples, weighted averaging across subsets samples, and kernel smoothing across subset samples. The four functions assume the user has previously run the Bayesian analysis and has produced the independent subposterior samples outside of the package; the functions use as input the array of subposterior samples. The methods have been demonstrated to be useful for Bayesian MCMC models including Bayesian logistic regression, Bayesian Gaussian mixture models and Bayesian hierarchical Poisson-Gamma models. The methods are appropriate for Bayesian hierarchical models with hyperparameters, as long as data values in a single level of the hierarchy are not split into subsets.

Depends mvtnorm

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2021-06-23 07:20:02 UTC

R topics documented:

`parallelMCMCcombine-package`
parallelMCMCcombine

Description

Recent Bayesian Markov chain Monto Carlo (MCMC) methods have been developed for big data sets that are too large to be analyzed using traditional statistical methods. These methods partition the data into non-overlapping subsets, and perform parallel independent Bayesian MCMC analyses on the data subsets, creating independent subposterior samples for each data subset. These independent subposterior samples are combined through four functions in this package, including averaging across subset samples, weighted averaging across subsets samples, and kernel smoothing across subset samples. The four functions assume the user has previously run the Bayesian analysis and has produced the independent subposterior samples outside of the package; the functions use as input the array of subposterior samples. The methods have been demonstrated to be useful for Bayesian MCMC models including Bayesian logistic regression, Bayesian Gaussian mixture models and Bayesian hierarchical Poisson-Gamma models. The methods are appropriate for Bayesian hierarchical models with hyperparameters, as long as data values in a single level of the hierarchy are not split into subsets.

Details

Package:	parallelMCMCcombine
Type:	Package
Version:	1.0
Date:	2021-06-18
License:	GPL-2

The package contains the following functions:

`consensusMCcov` Consensus Monte Carlo Algorithm (for correlated parameters)

`consensusMCindep` Consensus Monte Carlo Algorithm (for independent parameters)

`sampleAvg` Sample Averaging Method

`semiparamDPE` Semiparametric Method

Author(s)

Alexey Miroshnikov, Erin Conlon

Maintainer: Erin Conlon <econlon@umass.edu>

References

Scott, S.L., Blocker, A. W., Bonassi (2013) Bayes and Big Data: The consensus Monte Carlo Algorithm. *Bayes 250*.

Neiswanger, W., Wang, C., Xing, E. (2014) Asymptotically exact, embarrassingly parallel MCMC. *arXiv:1311.4780v2*.

Silverman, B.W. (1986). Density Estimation for Statistics and Data Analysis. *Chapman & Hall/CRC. pp. 7-11*.

`consensusMCcov` *Consensus Monte Carlo Algorithm (for correlated parameters)*

Description

The function uses the Consensus Monte Carlo algorithm introduced by Scott et al. (see References) to combine the independent subset posterior samples subchains into the set of samples that estimate the posterior density given the full data set. The Consensus Monte Carlo algorithm uses a weighted average of the subset posterior samples to produce the combined posterior samples, where the weights are based on the inverse variance-covariance matrix of the subset posterior samples.

Usage

```
consensusMCcov(subchain, shuff = FALSE)
```

Arguments

`subchain`	array of subset posterior samples of the dimension '`c(d,sampT,M).`' Here '`d`' is the dimension of the parameter space, '`sampT`' is the number of samples, and '`M`' is the number of subposterior datasets.
`shuff`	shuff: logical; if TRUE, each of the '`M`' subsets of '`d`' dimensional parameters in '`subchain`' is shuffled.

Details

The array '`subchain`' must have dimension '`c(d,sampT,M)`'. Here '`d`' is the dimension of the parameter space, '`sampT`' is the number of samples, and '`M`' is the number of subposterior datasets.

Value

Returns an array of samples of dimension dim=c(d,sampT) representing an estimated (combined) full posterior density.

References

Scott, S.L., Blocker, A. W., Bonassi (2013) Bayes and Big Data: The consensus Monte Carlo Algorithm. *Bayes 250*.

Examples

```
d      <- 2     # dimension of the parameter space
sampT  <- 1000  # number of subset posterior samples
M      <- 3     # total number of subsets

## simulate Gaussian subposterior samples

theta <- array(NA,c(d,sampT,M))

norm.mean <- c(1.0, 2.0)
norm.sd   <- c(0.5, 1.0)

for (i in 1:d)
  for (s in 1:M)
    theta[i,,s] <- rnorm(sampT, mean=norm.mean[i]+runif(1,-0.01,0.01), sd=norm.sd[i])

## combine samples:

full.theta <- consensusMCcov(subchain=theta, shuff=FALSE)
```

---

consensusMCindep *Consensus Monte Carlo Algorithm (for independent parameters)*

Description

The function uses the Consensus Monte Carlo algorithm introduced by Scott et al. (see References) to combine the independent subset posterior samples subchains into the set of samples that estimate the posterior density given the full data set. The Consensus Monte Carlo algorithm uses a weighted average of the subset posterior samples to produce the combined posterior samples, where the weights are based on the inverse variance-covariance matrix of the subset posterior samples. Here, the model parameters are assumed to be independent, so that the covariance between model parameters is equal to zero.

Usage

```
consensusMCindep(subchain, shuff = FALSE)
```

Arguments

subchain	array of subset posterior samples of the dimension 'c(d,sampT,M).' Here 'd' is the dimension of the parameter space, 'sampT' is the number of samples, and 'M' is the number of subposterior datasets.
shuff	shuff: logical; if TRUE, each of the 'M' subsets of 'd' dimensional parameters in 'subchain' is shuffled.

Details

The array 'subchain' must have dimension 'c(d,sampT,M)'. Here 'd' is the dimension of the parameter space, 'sampT' is the number of samples, and 'M' is the number of subposterior datasets.

Value

Returns an array of samples of dimension dim=c(d,sampT) representing an estimated (combined) full posterior density.

References

Scott, S.L., Blocker, A. W., Bonassi (2013) Bayes and Big Data: The consensus Monte Carlo Algorithm. *Bayes 250 day*.

Examples

```
d      <- 2     # dimension of the parameter space
sampT  <- 1000  # number of subset posterior samples
M      <- 3     # total number of subsets

## simulate Gaussian subposterior samples

theta <- array(NA,c(d,sampT,M))

norm.mean <- c(1.0, 2.0)
norm.sd   <- c(0.5, 1.0)

for (i in 1:d)
  for (s in 1:M)
    theta[i,,s] <- rnorm(sampT, mean=norm.mean[i]+runif(1,-0.01,0.01), sd=norm.sd[i])

## combine samples:

full.theta <- consensusMCindep(subchain=theta, shuff=FALSE)
```

`sampleAvg` *Sample Averaging Method*

Description

The function combines the independent subset posterior samples subchains into the set of samples that estimate the posterior density given the full data set, by averaging the samples across subsets. Individual model parameters are assumed to be independent.

Usage

```
sampleAvg(subchain, shuff = FALSE)
```

Arguments

`subchain`	array of subset posterior samples of the dimension '`c(d,sampT,M).`' Here '`d`' is the dimension of the parameter space, '`sampT`' is the number of samples, and '`M`' is the number of subposterior datasets.
`shuff`	shuff: logical; if TRUE, each of the '`M`' subsets of '`d`' dimensional parameters in '`subchain`' is shuffled.

Details

The array 'subchain' must have dimension 'c(d,sampT,M)'. Here 'd' is the dimension of the parameter space, 'sampT' is the number of samples, and 'M' is the number of subposterior datasets.

Value

Returns an array of samples of dimension dim=c(d,sampT) representing an estimated (combined) full posterior density.

Examples

```
d      <- 2     # dimension of the parameter space
sampT  <- 1000  # number of subset posterior samples
M      <- 3     # total number of subsets

## simulate Gaussian subposterior samples

theta <- array(NA,c(d,sampT,M))

norm.mean <- c(1.0, 2.0)
norm.sd   <- c(0.5, 1.0)

for (i in 1:d)
  for (s in 1:M)
    theta[i,,s] <- rnorm(sampT, mean=norm.mean[i]+runif(1,-0.01,0.01), sd=norm.sd[i])

## combine samples:

full.theta <- sampleAvg(subchain=theta, shuff=FALSE)
```

---

semiparamDPE *Semiparametric Density Product Estimator Method*

Description

The function uses the Semiparametric Density Product Estimator method introduced by Neiswanger et al. (see References) to combine the independent subset posterior samples subchains into the set of samples that estimate the posterior density given the full data set. The semiparametric density product estimator method uses kernel smoothing techniques to estimate each subset posterior density; the subposterior densities are then multiplied together to approximate the posterior density based on the full data set.

Usage

```
semiparamDPE(subchain, bandw = rep(1.0, dim(subchain)[1]), anneal = TRUE, shuff = FALSE)
```

Arguments

subchain	array of subset posterior samples of the dimension 'c(d,sampT,M).' Here 'd' is the dimension of the parameter space, 'sampT' is the number of samples, and 'M' is the number of subposterior datasets.
bandw	bandwidth vector of the length 'd=dim(subchain)[1]'. It is a vector of tuning parameters used in kernel density approximation employed by the semiparametric method. When 'anneal=TRUE' then one of the choices for 'bandw' could be the vector consisting of standard deviations for each of the 'd' parameters. When 'anneal=FALSE' then one of the choices for 'bandw' could be the diagonal of the optimal bandwidth matrix obtained via Silverman's rule of thumb; see Examples. By default 'bandw=rep(1.0,d)'.
anneal	logical; if TRUE, the bandwidth 'bandw' (instead of being fixed) is annealed as 'bandw*i^(-1/(4+d))'; here 'i' is the index corresponding to a sample; see References.
shuff	logical; if TRUE, each of the 'M' subsets of 'd' dimensional parameters in 'subchain' is shuffled.

Value

Returns an array of samples of dimension dim=c(d,sampT) representing an estimated (combined) full posterior density.

References

Neiswanger, W., Wang, C., Xing E. (2014) Asymptotically exact, embarrassingly parallel MCMC. arXiv:1311.4780v2.

Silverman, B.W. (1986). Density Estimation for Statistics and Data Analysis. *Chapman & Hall/CRC. pp. 7-11.*

Examples

```
d      <- 2     # dimension of the parameter space
sampT  <- 300   # number of subset posterior samples
M      <- 3     # total number of subsets

## simulate Gaussian subposterior samples

theta <- array(NA,c(d,sampT,M))

norm.mean <- c(1.0, 2.0)
norm.sd   <- c(0.5, 1.0)

for (i in 1:d)
  for (s in 1:M)
    theta[i,,s] <- rnorm(sampT, mean=norm.mean[i]+runif(1,-0.01,0.01), sd=norm.sd[i])

## estimate (mean) standard deviations for each parameter across the subsets

norm.var.est <- rep(0,d)
```

```
for(i in 1:d)
  for(s in 1:M)
    norm.var.est[i] <- norm.var.est[i] + var(theta[i,,s])

norm.sd.est <- sqrt(norm.var.est/M)


## Compute the diagonal of the optimal bandwidth
## matrix according to Silverman's rule

h_opt1 = (4/(d+2))^(1/(4+d)) * (sampT^(-1/(4+d))) * norm.sd.est

## Combine samples. The bandwidth matrix is fixed:

full.theta1 <- semiparamDPE( subchain = theta, bandw = h_opt1 * 2, anneal = FALSE)

## Compute the diagonal of the optimal bandwidth
## matrix for the method that uses annealing

h_opt2 = (4/(d+2))^(1/(4+d)) * norm.sd.est

## Combine samples. The bandwidth matrix will be annealed:

full.theta2 <- semiparamDPE(subchain = theta, bandw = h_opt2 * 2, anneal = TRUE)
```

Index