Package ‘SBmedian’

October 12, 2022

Type Package

Title Scalable Bayes with Median of Subset Posteriors

Version 0.1.1

Description Median-of-means is a generic yet powerful framework for scalable and robust estimation. A framework for Bayesian analysis is called M-posterior, which estimates a median of subset posterior measures. For general exposition to the topic, see the paper by Minsker (2015) <doi:10.3150/14-BEJ645>.

License MIT + file LICENSE

Encoding UTF-8

Imports Rcpp, Rdpack, expm, stats, utils

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.1.1

RdMacros Rdpack

NeedsCompilation yes

Author Kisung You [aut, cre] (<https://orcid.org/0000-0002-8584-459X>)

Maintainer Kisung You <kisungyou@outlook.com>

Repository CRAN

Date/Publication 2021-08-16 07:10:05 UTC

R topics documented:

mpost.euc *Median Posterior for Subset Posterior Samples in Euclidean Space*

Description

`mpost.euc` is a general framework to *merge* multiple empirical measures $Q_1, Q_2, \ldots, Q_M \subset R^p$ from independent subset of data by finding a median

$$\hat{Q} = \mathrm{argmin}_Q \sum_{m=1}^{M} d(Q, Q_m)$$

where Q is a weighted combination and $d(P_1, P_2)$ is distance in RKHS between two empirical measures P_1 and P_2. As in the references, we use RBF kernel with bandwidth parameter σ.

Usage

```
mpost.euc(
  splist,
  sigma = 0.1,
  maxiter = 121,
  abstol = 1e-06,
  show.progress = FALSE
)
```

Arguments

`splist`	a list of length M containing vectors or matrices of univariate or multivariate subset posterior samples respectively.
`sigma`	bandwidth parameter for RBF kernel.
`maxiter`	maximum number of iterations for Weiszfeld algorithm.
`abstol`	stopping criterion for Weiszfeld algorithm.
`show.progress`	a logical; `TRUE` to show iteration mark, `FALSE` otherwise.

Value

a named list containing:

med.atoms a vector or matrix of all atoms aggregated.

med.weights a weight vector that sums to 1 corresponding to `med.atoms`.

weiszfeld.weights a weight for M subset posteriors.

weiszfeld.history updated parameter values. Each row is for iteration, while columns are weights corresponding to `weiszfeld.weights`.

References

Minsker S, Srivastava S, Lin L, Dunson DB (2014). "Scalable and Robust Bayesian Inference via the Median Posterior." In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, II–1656–II–1664. event-place: Beijing, China.

Minsker S, Srivastava S, Lin L, Dunson DB (2017). "Robust and Scalable Bayes via a Median of Subset Posterior Measures." *Journal of Machine Learning Research*, **18**(124), 1–40. https://jmlr.org/papers/v18/16-655.html.

Examples

```
## Median Posteior from 2-D Gaussian Samples
#  Step 1. let's build a list of atoms whose numbers differ
set.seed(8128)                   # for reproducible results
mydata = list()
mydata[[1]] = cbind(rnorm(96, mean= 1), rnorm(96, mean= 1))
mydata[[2]] = cbind(rnorm(78, mean=-1), rnorm(78, mean= 0))
mydata[[3]] = cbind(rnorm(65, mean=-1), rnorm(65, mean= 1))
mydata[[4]] = cbind(rnorm(77, mean= 2), rnorm(77, mean=-1))

#  Step 2. Let's run the algorithm
myrun = mpost.euc(mydata, show.progress=TRUE)

#  Step 3. Visualize
#  3-1. show subset posterior samples
opar <- par(no.readonly=TRUE)
par(mfrow=c(2,3), no.readonly=TRUE)
for (i in 1:4){
  plot(mydata[[i]], cex=0.5, col=(i+1), pch=19, xlab="", ylab="",
       main=paste("subset",i), xlim=c(-4,4), ylim=c(-3,3))
}

#  3-2. 250 median posterior samples via importance sampling
id250 = base::sample(1:nrow(myrun$med.atoms), 250, prob=myrun$med.weights, replace=TRUE)
sp250 = myrun$med.atoms[id250,]
plot(sp250, cex=0.5, pch=19, xlab="", ylab="",
     xlim=c(-4,4), ylim=c(-3,3), main="median samples")

#  3-3. convergence over iterations
matplot(myrun$weiszfeld.history, xlab="iteration", ylab="value",
        type="b", main="convergence of weights")
par(opar)
```

`mpost.spd` *Median Posterior for Subset Posterior Samples in SPD manifold*

Description

SPD manifold is a collection of matrices that are symmetric and positive-definite and it is well known that using Euclidean geometry for data on the manifold is rather inaccurate. Here, we propose a function for dealing with SPD matrices specifically where valid examples include full-rank covariance and precision matrices. Note that $N_M = \sum_{m=1}^M n_m$.

Usage

```
mpost.spd(
  splist,
  sigma = 0.1,
  maxiter = 121,
  abstol = 1e-06,
  show.progress = FALSE
)
```

Arguments

`splist`	a list of length M containing $(p \times p)$ matrix or 3d array of size $(p \times p \times n_m)$ whose slices are SPD matrices from subset posterior samples respectively.
`sigma`	bandwidth parameter for RBF kernel.
`maxiter`	maximum number of iterations for Weiszfeld algorithm.
`abstol`	stopping criterion for Weiszfeld algorithm.
`show.progress`	a logical; `TRUE` to show iteration mark, `FALSE` otherwise.

Value

a named list containing:

med.atoms a $(p \times p \times N_M)$ 3d array whose slices are atoms aggregated.

med.weights a weight vector that sums to 1 corresponding to `med.atoms`.

weiszfeld.weights a weight for M subset posteriors.

weiszfeld.history updated parameter values. Each row is for iteration, while columns are weights corresponding to `weiszfeld.weights`.

Examples

```
## Median Posteior from 5-dimension Wishart distribution
## Visualization will be performed for distribution of larget eigenvalue
## where RED is for estimated density and BLUE is density from all samples.

#  Step 1. let's build a list of atoms whose numbers differ
set.seed(8128)                   # for reproducible results
mydata = list()
mydata[[1]] = stats::rWishart(96, df=10, Sigma=diag(5))
mydata[[2]] = stats::rWishart(78, df=10, Sigma=diag(5))
mydata[[3]] = stats::rWishart(65, df=10, Sigma=diag(5))
mydata[[4]] = stats::rWishart(77, df=10, Sigma=diag(5))
```

```
#  Step 2. Let's run the algorithm
myrun = mpost.spd(mydata, show.progress=TRUE)

#  Step 3. Compute largest eigenvalues for the samples
eig4 = list()
for (i in 1:4){
  spdmats = mydata[[i]]        # SPD atoms
  spdsize = dim(spdmats)[3]    # number of atoms
  eigvals = rep(0,spdsize)     # compute largest eigenvalues
  for (j in 1:spdsize){
    eigvals[j] = max(base::eigen(spdmats[,,j])$values)
  }
  eig4[[i]] = eigvals
}
eigA   = unlist(eig4)
eiglim = c(min(eigA), max(eigA))

#  Step 4. Visualize
#  4-1. show distribution of subset posterior samples' eigenvalues
opar <- par(no.readonly=TRUE)
par(mfrow=c(2,3))
for (i in 1:4){
  hist(eig4[[i]], main=paste("subset", i), xlab="largest eigenvalues",
       prob=TRUE, xlim=eiglim, ylim=c(0,0.1))
  lines(stats::density(eig4[[i]]), lwd=1, col="red")
  lines(stats::density(eigA),      lwd=1, col="blue")
}

#  4-2. 250 median posterior samples via importance sampling
id250 = base::sample(1:length(eigA), 250, prob=myrun$med.weights, replace=TRUE)
sp250 = eigA[id250]
hist(sp250, main="median samples", xlab="largest eigenvalues",
     prob=TRUE, xlim=eiglim, ylim=c(0,0.1))
lines(stats::density(sp250), lwd=1, col="red")
lines(stats::density(eigA),  lwd=1, col="blue")

#  4-3. convergence over iterations
matplot(myrun$weiszfeld.history, xlab="iteration", ylab="value",
        type="b", main="convergence of weights")
par(opar)
```

SBmedian *Scalable Bayes with Median of Subset Posteriors*

Description

Median-of-means is a generic yet powerful framework for scalable and robust estimation. A framework for Bayesian analysis is called M-posterior, which estimates a median of subset posterior measures.

Index