Package 'support'

July 22, 2018

Type Package

Title Support Points

Version 0.1.2

Author Simon Mak

Maintainer Simon Mak <smak6@gatech.edu>

Description

Provides the function sp() for generating the support points proposed in Mak and Joseph (2017) <arXiv:1609.01811>. Support points are representative points of a possibly non-uniform distribution, and can be used as optimal sampling or integration points for a distribution of choice. The provided function sp() can be used to generate support points for standard distributions or for reducing big data (e.g., from Markov-chain Monte Carlo methods). This work was supported by USARO grant W911NF-14-1-0024 and NSF DMS grant 1712642.

License GPL (>= 2)

LazyData TRUE

Imports Rcpp (>= 0.12.4), randtoolbox, matrixStats, doParallel, nloptr, MHadaptive

LinkingTo Rcpp, RcppArmadillo, BH

RoxygenNote 6.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2018-07-22 20:50:02 UTC

R topics documented:

support-package *Support Points*

Description

The 'support' package provides functions for generating support points.

Details

Package:	support
Type:	Package
Version:	1.0
Date:	2017-08-19
License:	GPL (>= 2)

The 'support' package provides the function `sp()` for generating the support points proposed in Mak and Joseph (2017) <arXiv:1609.01811>. Support points are representative points of a possibly non-uniform distribution, and can be used as optimal sampling or integration points for a distribution of choice. The provided function `sp()` can be used to generate support points for standard distributions or for reducing big data (e.g., from Markov-chain Monte Carlo methods). This work was supported by USARO grant W911NF-14-1-0024 and NSF DMS grant 1712642.

Author(s)

Simon Mak

Maintainer: Simon Mak <smak6@gatech.edu>

References

Mak, S. and Joseph, V. R. (2017). Support points. *Annals of Statistics*. To appear.

sp *Generating support points using difference-of-convex programming*

Description

`sp` is the main function for generating the support points proposed in Mak and Joseph (2017) <https://arxiv.org/abs/1609.01811>. To compute support points on standard distributions, the parameters `dist.str` and `dist.param` should be specified for the desired distribution and their parameters. To compute support points for big data reduction (e.g., Markov-chain Monte Carlo chains or large datasets), the data should be provided in the parameter `dist.samp`.

Usage

```
sp(n, p, ini=NA, std.flg=T,
  dist.str=rep("normal",p), dist.param=vector("list",p),
  dist.samp=NA, scale.ind=T, bd=NA,
  num_subsamp=max(10000,50*n), max_iter=200, min_iter=50,
  tol_out=min(1e-2/n,1e-4)*p, warm.cl=F)
```

Arguments

n	Number of support points.
p	Dimension of sample space.
ini	An n x p matrix for the initial point set.
std.flg	TRUE if standard distribution to be reduced; FALSE if (finite) big data.
dist.str	A p-length vector of strings for desired standard distributions (independence assumed). Current choices include uniform, normal, exponential, gamma, lognormal, student-t, weibull, cauchy and beta. Specify *only* for standard distributions.
dist.param	A p-length list for desired parameters of standard distributions. Parameter settings are as follows: • **Uniform**: Minimum, maximum; • **Normal**: Mean, standard deviation; • **Exponential**: Rate parameter; • **Gamma**: Shape parameter, scale parameter; • **Lognormal**: Log-mean, Log-standard deviation; • **Student-t**: Degree-of-freedom; • **Weibull**: Shape parameter, scale parameter; • **Cauchy**: Location parameter, scale parameter; • **Beta**: Shape parameter 1, shape parameter 2. Specify *only* for standard distributions.
dist.samp	An N x p matrix for the big data (e.g., MCMC chain) to reduce. Specify *only* for big data reduction.
scale.ind	Should the data be normalized to unit variance before processing? Specify *only* for big data reduction.
bd	A p x 2 matrix for the lower and upper bounds of each distribution.
num_subsamp	Batch sample size for resampling.
max_iter	Maximum number of iterations for optimization.
min_iter	Minimum number of iterations for optimization.
tol_out	Error tolerance for terminating optimization.
warm.cl	Still in development.

Value

sp	An n x p matrix for support points.
ini	An n x p matrix for the initial point set.

References

Mak, S. and Joseph, V. R. (2017). Support points. *Annals of Statistics*. To appear.

Examples

```
## Support points on standard distributions

#Generate 25 support points for the 2-d i.i.d. N(0,1) distribution
n <- 25 #number of points
p <- 2 #dimension
des <- sp(n,p,std.flg=TRUE,dist.str=rep("normal",p))

x1 <- seq(-3.5,3.5,length.out=100) #Plot contours
x2 <- seq(-3.5,3.5,length.out=100)
z <- exp(-outer(x1^2,x2^2,FUN="+")/2)
contour.default(x=x1,y=x2,z=z,drawlabels=FALSE,nlevels=10)
points(des$sp,pch=16,cex=1.25,col="red")

#Generate 50 support points for the 2-d i.i.d. Beta(2,4) distribution
n <- 50
p <- 2
dist.param <- vector("list",p)
for (l in 1:p){
  dist.param[[l]] <- c(2,4)
}
des <- sp(n,p,std.flg=TRUE,dist.str=rep("beta",p),dist.param=dist.param)

x1 <- seq(0,1,length.out=100) #Plot contours
x2 <- seq(0,1,length.out=100)
z <- matrix(NA,nrow=100,ncol=100)
for (i in 1:100){
  for (j in 1:100){
    z[i,j] <- dbeta(x1[i],2,4) * dbeta(x2[j],2,4)
  }
}
contour.default(x=x1,y=x2,z=z,drawlabels=FALSE,nlevels=10 )
points(des$sp,pch=16,cex=1.25,col="red")

#Generate 100 support points for the 5-d i.i.d. Exp(1) distribution
n <- 100
p <- 5
des <- sp(n,p,std.flg=TRUE,dist.str=rep("exponential",p))
pairs(des$sp,xlim=c(0,5),ylim=c(0,5),pch=16)

## Not run:
  ## Support points for big data reduction
  library(MHadaptive)

  #Use modified Franke's function as posterior
  franke2d <- function(xx){
```

```
  if ((xx[1]>1)||(xx[1]<0)||(xx[2]>1)||(xx[2]<0)){
    return(-Inf)
  }
  else{
    x1 <- xx[1]
    x2 <- xx[2]

    term1 <- 0.75 * exp(-(9*x1-2)^2/4 - (9*x2-2)^2/4)
    term2 <- 0.75 * exp(-(9*x1+1)^2/49 - (9*x2+1)/10)
    term3 <- 0.5 * exp(-(9*x1-7)^2/4 - (9*x2-3)^2/4)
    term4 <- -0.2 * exp(-(9*x1-4)^2 - (9*x2-7)^2)

    y <- term1 + term2 + term3 + term4
    return(2*log(y))
  }
}

#Generate MCMC samples
li_func <- franke2d #Desired log-posterior
ini <- c(0.5,0.5) #Initial point for MCMc
NN <- 1e5 #Number of MCMC samples desired
burnin <- NN/2 #Number of burn-in runs
mcmc_r <- Metro_Hastings(li_func, pars=ini, prop_sigma=0.05*diag(2),
              iterations=NN, burn_in=burnin)

#Compute 100 support points
n <- 100
des <- sp(n,2,std.flg=FALSE,dist.samp=mcmc_r$trace)

#Plot support points
x1 <- seq(0,1,length.out=100) #contours
x2 <- seq(0,1,length.out=100)
z <- matrix(NA,nrow=100,ncol=100)
for (i in 1:100){
  for (j in 1:100){
    z[i,j] <- franke2d(c(x1[i],x2[j]))
  }
}
contour.default(x=x1,y=x2,z=z,drawlabels=TRUE,nlevels=10 )
points(des$sp,pch=16,cex=1.25,col="red")


## End(Not run)
```

Index