icamix

July 1, 2014

ATRANSDENSITY *ATRANSDENSITY*

Description

A function evaluates density of linearly transformed random vector on a given grid. It is used in processing EMFASTICAALG object to obtain density estimation of the mixture components.

Usage

```
ATRANSDENSITY(grid, A, f)
```

Arguments

grid	A matrix whose columns store the grid points.
A	Matrix for the linear transformation.
f	Density function before the linear transformation.

Value

answer	Matrix of the same size as grid, with each element being the evaluated linear transformed density at the corresponding grid point.

Examples

```
## An example that evaluates the 2-D uniform distribution on a linear transformation of [1,3]x[1,3]
## f1ind is the density of the uniform distribution on [1,3]^r
f1ind <- function(grid){# mixture component 1 original signal density function
  n <- ncol(grid)
  r <- nrow(grid)
  answer <- rep(1,n)
  for(i in 1:n){
    for(j in 1:r){
      answer[i] <- answer[i] * (grid[j,i] >= 1 & grid[j,i] <= 3) / 2
    }
  }
  answer
}

A <- matrix(c(6, 9, -12, 15), 2, 2, byrow = FALSE)
```

```
gridpoints <- t(as.matrix(expand.grid(seq(-32,12,2),seq(18,80,2))))

f1trans <- ATRANSDENSITY(gridpoints, A, f1ind)

plot(t(gridpoints),col=(f1trans>0))
```

CLASSDIFFRATE *CLASSDIFFRATE*

Description

A function calculates classification difference rate between two factors. It is used in interpreting info stored in EMFASTICA object.

Usage

```
CLASSDIFFRATE(factor1, factor2)
```

Arguments

factor1	First factor.
factor2	Second factor of the same length as the First factor.

Value

answer	The percentage of instances when factor1[i] is not equal factor2[i].

Examples

```
## An example evaluates the classification difference rate
## between two classification results in the form of factors
fac1<-factor(c(1,4,2,3,1,1,3,3,1,2,2,1))
fac2<-factor(c(3,1,2,2,1,2,4,3,2,3,1,1))
CLASSDIFFRATE(fac1, fac2)
```

EMFASTICAALG *EMFASTICAALG*

Description

An R wrapper for carrying out NSMM-ICA on nonparametric multivariate ICA mixture data.

Usage

```
EMFASTICAALG(DataMatrix, numCluster, h = 0, maxiter = 300,
icaiter = 150, tol = 1e-06, verb = TRUE, combine = TRUE,
seednum = 82196, ...)
```

Arguments

DataMatrix	A matrix of which the rows are data entries. Its dimension is n by r.
numCluster	Predetermined number of mixing components m.
h	Bandwidth. If h is set equal zero (default), iterative bandwidth selection will be used.
maxiter	Maximum number of iterations. Default is 300.
icaiter	Maximum number of ICA iterations in each step. Default is 150.
tol	Threshold that defines convergence (of the outer loop). Default is 1e-6.
verb	TRUE (default) or FALSE indicating whether to print out info at each iteration.
combine	TRUE (default) or FALSE indicating whether to implement the ICA step.
seednum	Seed number (default is 82196) used in kmeans before 1st iteration.
...	

Value

The returned value is an EMFASTICAALG object which consists of a list of items:

$InputData	A matrix of which the columns are data entries. Its dimension is r by n.
$Lambdas	A matrix where rows store estimated mixing weights from each iteration.
$WMtrs	List of r by r unmixing matrices for each of the m clusters.
$WUnmixZ	List of unmixing matrices for whitened data for each of the m clusters.
$OriginalSignals	List of Recovered ICA components for each of the m clusters.
$ProductDensity	m by n matrix where each row stores the estimated density value of the observed data points for each of the m clusters.
$MembershipProbs	n by m matrix where each row stores the component membership probabilities of the corresponding data point.
$ObjValue	Vector holding values of data loglikelihood.
$ICABandWidth	Matrix holding choices of bandwidth for original signals.
$call	The function call that results in the returned object.
$time	Computing time elapsed in second.

Examples

```
## An Example that runs the NSMM-ICA algorithm on Cohens tone data
## and compares the results with that obtained by npEM of the "mixtools" package
library(mixtools)
data(tonedata)

set.seed(100)
a <- npEM(tonedata, mu0=2, samebw=FALSE)
b <- EMFASTICAALG(tonedata, 2, h=0, tol=1e-8)

postmembership_a <- a$posteriors == apply(a$posteriors,1,max) # estimated membership matrix for a
estimatedmember_a <- factor(rep((1:2),
nrow(tonedata))[as.vector(t(postmembership_a))]) # estimated membership info for a
```

```
levels(estimatedmember_a) <- c(1,2) # numbering the estimated membership info for a

estimatedmember_b <- ESTIMATEDMEMBER(b) # estimated membership info for b
levels(estimatedmember_b) <- c(1,2) # numbering the estimated membership info for b

par(mfrow=c(1,2))
plot(tonedata, col = as.numeric(levels(estimatedmember_a))[estimatedmember_a]+1,
pch = as.numeric(levels(estimatedmember_a))[estimatedmember_a],
main = "npEM Classification Scatter Plot",asp=1)
plot(tonedata, col = as.numeric(levels(estimatedmember_b))[estimatedmember_b]+1,
pch = as.numeric(levels(estimatedmember_b))[estimatedmember_b],
main = "NSMM-ICA Classification Scatter Plot",asp=1)
par(mfrow=c(1,1)) # reset par
```

ESTIMATEDMEMBER *ESTIMATEDMEMBER*

Description

A function calculates estimated class membership from an EMFASTICAALG object.

Usage

```
ESTIMATEDMEMBER(rstICAMIX)
```

Arguments

rstICAMIX	An EMFASTICAALG object.

Value

estimatedmember

A factor with levels representing the estimated classes.

Examples

```
## An example running NSMM-ICA algorithm on Andersons iris data set
data(iris)
irisdata <- as.matrix(iris[,1:4])
iris1 <- EMFASTICAALG(irisdata, 3) # with iterative bandwidth selection
estimatedspecies <- ESTIMATEDMEMBER(iris1) # estimated species info
estimatedspecies
plot(irisdata[,1:2], asp = 1, col=as.numeric(levels(estimatedspecies))[estimatedspecies]+1,
pch = as.numeric(levels(estimatedspecies))[estimatedspecies],
main = "NSMM-ICA Classification",cex.lab=1.1)
```

icamix-package *icamix: Estimation for ICA Mixture Model*

Description

The icamix package provides R functions as well as a C++ library which facilitate the estimation of ICA mixture models.

Details

Package:	icamix
Type:	Package
Version:	1.0
Date:	2014-06-30
License:	GPL-2 | GPL-3

We have developed the NSMM-ICA algorithm which is implemented in the main function EMFASTICAALG that currently integrates npEM and Fast-ICA for non-parametric estimation of ICA mixture models.

Author(s)

Xiaotian Zhu, David R. Hunter

Maintainer: Xiaotian Zhu <xxz131@psu.edu>

References

http://sites.stat.psu.edu/~xxz131/XTZ_Dissertation.pdf

plot.EMFASTICAALG *plot.EMFASTICAALG*

Description

`plot` method for class `EMFASTICAALG`.

Usage

```
## S3 method for class EMFASTICAALG
plot(x, vec1 = c(1:2), vec2 = c(1:2), ...)
```

Arguments

`x`	An `EMFASTICAALG` object.
`vec1`	An integer vector of length two specifying the coordinates with respect to which the data is scatter plotted.
`vec2`	An integer vector of length two specifying the coordinates with respect to which the original signal for each mixture component is scatter plotted.
`...`	

Value

Returned (invisibly) is the full value of x itself.

Examples

```
## An example running NSMM-ICA algorithm on Andersons iris data set
data(iris)
irisdata <- as.matrix(iris[,1:4])
iris1 <- EMFASTICAALG(irisdata, 3) # with iterative bandwidth selection
plot(iris1)
```

`plot.summary.EMFASTICAALG`

plot.summary.EMFASTICAALG

Description

`plot` method for class `summary.EMFASTICAALG`.

Usage

```
## S3 method for class summary.EMFASTICAALG
plot(x, vec1 = c(1:2), vec2 = c(1:2), ...)
```

Arguments

`x`	An `summary.EMFASTICAALG` object.
`vec1`	An integer vector of length two specifying the coordinates with respect to which the data is scatter plotted.
`vec2`	An integer vector of length two specifying the coordinates with respect to which the original signal for each mixture component is scatter plotted.
`...`	

Value

Returned (invisibly) is the full value of x itself.

Examples

```
## An example running NSMM-ICA algorithm on Andersons iris data set
data(iris)
irisdata <- as.matrix(iris[,1:4])
iris1 <- EMFASTICAALG(irisdata, 3) # with iterative bandwidth selection
plot(summary(iris1))
```

print.EMFASTICAALG *print.EMFASTICAALG*

Description

print method for class EMFASTICAALG.

Usage

```
## S3 method for class EMFASTICAALG
print(x, ...)
```

Arguments

x	An EMFASTICA object.
...	

Value

Returned (invisibly) is the full value of x itself.

Examples

```
## An example running NSMM-ICA algorithm on Andersons iris data set
data(iris)
irisdata <- as.matrix(iris[,1:4])
iris1 <- EMFASTICAALG(irisdata, 3) # with iterative bandwidth selection
print(iris1)
```

---

print.summary.EMFASTICAALG *print.summary.EMFASTICAALG*

Description

print method for class summary.EMFASTICAALG.

Usage

```
## S3 method for class summary.EMFASTICAALG
print(x, ...)
```

Arguments

x	An summary.EMFASTICA object.
...	

Value

Returned (invisibly) is the full value of x itself.

Examples

```
## An example running NSMM-ICA algorithm on Andersons iris data set
data(iris)
irisdata <- as.matrix(iris[,1:4])
iris1 <- EMFASTICAALG(irisdata, 3) # with iterative bandwidth selection
print(summary(iris1))
```

---

summary.EMFASTICAALG *summary.EMFASTICAALG*

Description

summary method for class EMFASTICAALG.

Usage

```
## S3 method for class EMFASTICAALG
summary(object, ...)
```

Arguments

object	An EMFASTICAALG object.
...	

Value

The returned value is a "summary.EMFASTICAALG" object which consists a list:

$inputData	A matrix of which the columns are data entries. Its dimension is r by n.
$originSig	List of Recovered ICA components for each of the m clusters.
$call	The function call which results in the corresponding EMFASTICAALG object.
$time	Computing time elaped in second.
$numIter	Total number of iterations.
$lastObj	Objective function value from the last iteration.
$compMeans	Means of each mixture component.
$compVars	Covariances of each mixture component.
$numObs	Total number of observations.
$numAtr	Dimension of data points.
$numCls	Number of mixture components.
$estWts	Estimated mixing weights.
$estCls	A factor whose levels represent estimated class membership.
$dataLklhd	A vector storing data loglikelihood from each iteration.

Examples

```
## An example running NSMM-ICA algorithm on Andersons iris data set
data(iris)
irisdata <- as.matrix(iris[,1:4])
iris1 <- EMFASTICAALG(irisdata, 3) # with iterative bandwidth selection
summary(iris1)
```

Index