Package 'coloredICA'

February 24, 2015

Type Package

Title Implementation of Colored Independent Component Analysis and Spatial Colored Independent Component Analysis

Version 1.0.0

Date 2014-03-04

Author Lee, S., Shen, H., Truong, Y. and Zanini, P.

Maintainer Paolo Zanini <paolo.zanini@polimi.it>

Depends MASS

Suggests fastICA

Description It implements colored Independent Component Analysis (Lee et al., 2011) and spatial colored Independent Component Analysis (Shen et al., 2014). They are two algorithms to perform ICA when sources are assumed to be temporal or spatial stochastic processes, respectively.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2015-02-24 11:01:51

R topics documented:

amari_distance *Amari error*

Description

This function measures the Amari error between two matrices.

Usage

```
amari_distance(Q1, Q2)
```

Arguments

Q1	First matrix.
Q2	Second matrix.

Details

The Amari error $D(Q1|Q2)$ between two M x M matrices $Q1$ and $Q2$ is evaluated through

$$D(Q1|Q2) = \frac{1}{2M(M-1)} \sum_{j=1}^{M} \left(\frac{\sum_i |a_{ij}|}{max_i |a_{ij}|} - 1 \right) + \frac{1}{2M(M-1)} \sum_{i=1}^{M} \left(\frac{\sum_j |a_{ij}|}{max_j |a_{ij}|} - 1 \right),$$

where $Q2$ is invertible and a_{ij} is the ijth element of $Q1Q2^{-1}$.

Value

It returns the Amari error between two matrices $Q1$ and $Q2$.

Author(s)

Lee, S., Shen, H., Truong, Y. and Zanini, P.

References

Amari, S., Cichocki, A., Yang, H. et al. (1996). A New Learning Algorithm for Blind Signal Separation. *Advances in Neural Information Processing Systems*, **8**, 757–763

Bach, F., Jordan, M. (2003). Kernel Independent Component Analysis. *Journal of Machine Learning Research*, **3**, 1–48

See Also

cICA, scICA

Examples

```
M <- 4
A <- matrix(rnorm(M*M),M,M)
B <- matrix(rnorm(M*M),M,M)

amari_distance(A,B)
```

---

cICA *colored Independent Component Analysis*

Description

This function implements the colored Independent Component Analysis (cICA) algorithm, where sources are treated as temporal stochastic processes.

Usage

```
cICA(Xin, M = dim(Xin)[1], Win = diag(M), tol = 1e-04, maxit = 20, nmaxit = 1,
unmixing.estimate = "eigenvector", maxnmodels = 100)
```

Arguments

`Xin`	Data matrix with `p` rows (representing variables) and `n` columns (representing observations).
`M`	Number of components to be extracted.
`Win`	Initial guess for the unmixing matrix `W`. Dimensions need to be `M x M`.
`tol`	Tolerance used to establish the convergence of the algorithm.
`maxit`	Maximum number of iterations.
`nmaxit`	If the algorithm does not converge, it is run again with a new initial guess for the unmixing matrix `W`. This operation is done `nmaxit` times.
`unmixing.estimate`	The method used in the unmixing matrix estimation step. The two allowed choices are `"eigenvector"` and `"newton"` (see Details).
`maxnmodels`	Maximum number of models tested in the spectral density estimation step of the algorithm (see Details).

Details

In the Independent Component Analysis approach, the data matrix X is considered to be a linear combination of independent components, i.e. $X = AS$, where rows of S contain the unobserved realizations of the independent components and A is a linear mixing matrix. According to classical ICA procedures data matrix X is centered and, then, whitened by projecting the data onto its principal component directions, i.e. $X \rightarrow KX = \widetilde{X}$ where K is a `M x p` pre-whitening matrix. The cICA algorithm then estimates the unmixing matrix W, with $W\widetilde{X} = S$, according to the

procedure described below. Then, defining $\widetilde{W} = WK$, the mixing matrix A is recovered through $A = \widetilde{W}^T(\widetilde{W}\widetilde{W}^T)^{-1}$.

Colored Independent Component Analysis assumes that the independent sources are temporal stochastic processes. To perform ICA, the Whittle log-likelihood is exploited. In particular the log-likelihood is written in function of the unmixing matrix W and the spectral densities f_{S_j} of the autocorrelated sources as follows:

$$l(W, \boldsymbol{f_S}; \widetilde{X}) = \sum_{j=1}^{p}\sum_{k=1}^{n}\left(\frac{\boldsymbol{e}_j^T W \widetilde{\boldsymbol{f}}(r_k, \widetilde{X}) W^T \boldsymbol{e}_j}{f_{S_j}(r_k)} + \ln f_{S_j}(r_k)\right) + n \ln|\det(W)|.$$

Due to whitening, W is orthogonal and the last term of the objective function can be dropped. The orthogonality of the unmixing matrix W can be imposed in two different ways, setting the argument `unmixing.estimate`. In this way the estimate of the unmixing matrix W can be found according two different procedures:

- as described in Shen et al. (2014). A penalty term is added to the objective function. In particular $\tau \boldsymbol{w}_j' C_j \boldsymbol{w}_j$, where $\boldsymbol{w}_j'$ is the jth column of W, $C_j = \sum_{k \neq j} \boldsymbol{w}_k \boldsymbol{w}_k'$ and τ is a tuning parameter. The matrix C_j provides an orthogonality constraint in the sense that $\boldsymbol{w}_j' C_j \boldsymbol{w}_j = \sum_{k \neq j}$. In this way the objective function assumes a symmetric and positive-definite form and the argmin correspond to the lower eigenvalue. This choice is obtained setting `unmixing.matrix = "eigenvector"`.
- as described in Lee et al. (2011). The orthogonality constraint is considered performing the minimization of the objective function according a Newton-Raphson method with Lagrange multiplier. This choice is obtained setting `unmixing.matrix = "newton"`.

Independently from the choice of the technique to minimize the objective function, the cICA algorithm is based on an iterative procedure. While the Amari error is greater than `tol` and the number of iteration is less or equal than `maxit`, the two following steps are repeated:

- parametric estimation of the sources spectral densities using the Yule-Walker method, evaluating `maxmodels` models.
- estimate the unmixing matrix W according the method selected in `unmixing.estimate`.

Value

A list containing the following components:

`W`	Estimate of the `M x M` unmixing matrix in the whitened space.
`K`	pre-whitening matrix that projects data onto the first `M` principal components. Dimensions are `M x p`.
`A`	Estimate of the `p x M` mixing matrix.
`S`	Estimate of the `M x n` source matrix.
`X`	Original `p x n` data matrix.
`iter`	number of iterations.
`NInv`	number of times the algorithm is rerun after it does not achieve convergence.
`den`	Estimate of the spectral density of the sources. Dimensions are `M x n`.

Author(s)

Lee, S., Shen, H., Truong, Y. and Zanini, P.

References

Lee, S., Shen, H., Truong, Y., Lewis, M., Huang, X. (2011). Independent Component Analysis Involving Autocorrelated Sources With an Application to Funcional Magnetic Resonance Imaging. *Journal of the American Statistical Association*, **106**, 1009–1024.

Shen, H., Truong, Y., Zanini, P. (2014). Independent Component Analysis for Spatial Processes on a Lattice. *MOX report 38/2014*, Department of Mathematics, Politecnico di Milano.

See Also

scICA

Examples

```
## Not run:

require(fastICA)

T=256
n1=16
n2=16
M=2

S1 = arima.sim(list(order=c(0,0,2),ma=c(1,0.25)),T)
S2 = arima.sim(list(order=c(1,0,0), ar=-0.5),T,rand.gen = function(n, ...) (runif(n)-0.5)*sqrt(3))

A = rerow(matrix(runif(M^2)-0.5,M,M))
W = solve(A)
S=rbind(S1,S2)
X = A %*% S

cica = cICA(X,tol=0.001)
## scica = scICA(X,n1=n1,n2=n2,h=0.8,tol=0.001)
fica = fastICA(t(X),2)

amari_distance(t(A),t(cica$A))
## amari_distance(t(A),t(scica$A))
amari_distance(t(A),fica$A)

Shat1=cica$S
## Shat2=scica$S
Shat3=t(fica$S)

par(mfrow=c(2,2))
plot(S[1,],type="l",lwd=2)
plot(S[2,],type="l",lwd=2)
plot(Shat1[1,],type="l",lwd=2,col="red")
plot(Shat1[2,],type="l",lwd=2,col="red")
```

```
## par(mfrow=c(2,2))
## plot(S[1,],type="l",lwd=2)
## plot(S[2,],type="l",lwd=2)
## plot(Shat2[1,],type="l",lwd=2,col="green")
## plot(Shat2[2,],type="l",lwd=2,col="green")

par(mfrow=c(2,2))
plot(S[1,],type="l",lwd=2)
plot(S[2,],type="l",lwd=2)
plot(Shat3[1,],type="l",lwd=2,col="blue")
plot(Shat3[2,],type="l",lwd=2,col="blue")

## End (Not run)
```

`grad` *Gradient*

Description

This function evaluates the gradient of the objective function for the spectral density local maximum likelihood estimator.

Usage

```
grad(x, omega, l_period, n, freq, h)
```

Arguments

`x`	Current estimate
`omega`	Frequency at which the spectral density estimate is evaluated.
`l_period`	Vector of length `n` with the log-periodogram evaluations at the `n` Fourier frequencies.
`n`	Number of points in the analyzed lattice.
`freq`	`n` $\times$ 2 matrix with the `n` Fourier frequencies.
`h`	Kernel bandwidth.

Details

In `scICA` function the maximization for the spectral density local maximum likelihood estimator is obtained through the Newton-Raphson algorithm. This function returns the gradient needed in the optimization method. See `locmulti` for further details.

Value

It returns a gradient vector of length 3.

Note

It is auxiliary for scICA function.

Author(s)

Lee, S., Shen, H., Truong, Y. and Zanini, P.

See Also

scICA, locmulti, kern.

---

hess *Hessian*

Description

This function evaluates the Hessian matrix of the objective function for the spectral density local maximum likelihood estimator.

Usage

```
hess(x, omega, l_period, n, freq, h)
```

Arguments

x	Current estimate
omega	Frequency at which the spectral density estimate is evaluated.
l_period	Vector of length n with the log-periodogram evaluations at the n Fourier frequencies.
n	Number of points in the analyzed lattice.
freq	$n \times 2$ matrix with the n Fourier frequencies.
h	Kernel bandwidth.

Details

In scICA function the maximization for the spectral density local maximum likelihood estimator is obtained through the Newton-Raphson algorithm. This function returns the Hessian matrix needed in the optimization method. See locmulti for further details.

Value

It returns a 3×3 Hessian matrix.

Note

It is auxiliary for scICA function.

Author(s)

Lee, S., Shen, H., Truong, Y. and Zanini, P.

See Also

scICA, locmulti, kern.

`kern` *Kernel evaluation*

Description

This function returns a vector of length n with the evaluation of an exponential Kernel at a point `x0` $\in R^2$ for the n Fourier frequencies.

Usage

```
kern(x0, h, freq)
```

Arguments

`x0`	Point $\in R^2$ at which the Kernel is evaluated.
`h`	Kernel bandwidth.
`freq`	`n` $\times$ 2 matrix with the `n` Fourier frequencies, where `n` is the number of points in the analyzed lattice.

Details

This function returns a vector of length n with the evaluation of an exponential Kernel at a point `x0` $\in R^2$ for the n Fourier frequencies. In particular the k-th vector element is

$$e^{-\frac{(\boldsymbol{\omega}_k - x_0)^T(\boldsymbol{\omega}_k - x_0)}{h^2}}.$$

Value

It returns a list containing the following component:

`v`	vector containing the n exponential Kernel evaluations.

Note

It is auxiliary for locmulti, grad and hess functions.

Author(s)

Lee, S., Shen, H., Truong, Y. and Zanini, P.

See Also

scICA, locmulti, grad, hess.

`locmulti` *Local polynomial spectral density estimation*

Description

This function implements a local polinomial estimation for the log spectral density at a point `x0` $\in R^2$.

Usage

```
locmulti(x0, l_period, n, freq, h)
```

Arguments

`x0`	Point $\in R^2$ at which the spectral density estimate is evaluated.
`l_period`	Vector of length `n` with the log-periodogram evaluations at the `n` Fourier frequencies.
`n`	Number of points in the analyzed lattice.
`freq`	`n` $\times$ 2 matrix with the `n` Fourier frequencies.
`h`	Kernel bandwidth.

Details

`locmulti` function is auxiliary for the nonparametric estimation of the sources spectral density step of the `scICA` function. `locmulti` function implements the initial estimates for the local maximum likelihood estimator of the log spectral density $m(\texttt{x0})$ at a point `x0` $\in R^2$. To obtain an estimate of $m(\texttt{x0})$ the local likelihood function

$$\sum_k \left(Y_k - a - \mathbf{b}'(\boldsymbol{\omega}_k - x0) - e^{Y_k - a - \mathbf{b}'(\boldsymbol{\omega}_k - x0)} \right) K_H(\boldsymbol{\omega}_k - x0)$$

is constructed, where Y_k denotes the log-periodogram value at the Fourier frequency $\boldsymbol{\omega}_k$, K_H a surface kernel and $H = (h, h)$. The local maximum estimator $\widehat{m}_{LK}(\texttt{x0})$ is $\widehat{a}$ in the maximizer $(\widehat{a}, \widehat{\mathbf{b}})$. The estimate is implemented directly in the `scICA` function through a Newton-Rapshon algorithm. The initialization for the Newton-Rapshon algorithm is derived through a local polynomial approximation implemented in this `locmulti` function. In particular the following function is minimized to find a local polynomial approximation for $m(\texttt{x0})$

$$\sum_k \left(Y_k - a - \mathbf{b}'(\boldsymbol{\omega}_k - x0) \right)^2 K_H(\boldsymbol{\omega}_k - x0)$$

and the minimizer $\widehat{a}$ is used as an initial value in order to obtain the local maximum likelihood estimator $\widehat{m}_{LK}(\texttt{x0})$.

Value

It returns a list containing the following component:

`ahat`	local polynomial estimate of the log spectral density at `x0`.

Note

It is auxiliary for scICA function.

Author(s)

Lee, S., Shen, H., Truong, Y. and Zanini, P.

References

Shen, H., Truong, Y., Zanini, P. (2014). Independent Component Analysis for Spatial Processes on a Lattice. *MOX report 38/2014*, Department of Mathematics, Politecnico di Milano.

Fan, J., Kreutzberger, E. (1998). Automatic Local Smoothing for Spectral Density Estimation. *Scandinavian Journal of Statistics*, **25**, 359–369.

See Also

scICA, kern

`rerow` *Standardization of a matrix reordering the rows*

Description

This function reorders and standardizes the rows of a matrix.

Usage

```
rerow(w)
```

Arguments

w	Matrix to standardize.

Details

The standardization is done in such a way that every row has length 1, the largest absolute value of the row has a positive sign and the rows are ordered decreasingly according to their largest value.

Value

It returns the standardized matrix.

Note

It is auxiliary for cICA and scICA functions.

Author(s)

Lee, S., Shen, H., Truong, Y. and Zanini, P.

See Also

amari_distance, cICA, scICA

---

scICA *spatial colored Independent Component Analysis*

Description

This function implements the spatial colored Independent Component Analysis (scICA) algorithm, where sources are treated as spatial stochastic processes on a lattice.

Usage

```
scICA(Xin, M = dim(Xin)[1], Win = diag(M), tol = 1e-04, maxit = 20, nmaxit = 1,
unmixing.estimate = "eigenvector", n1, n2, nx01 = n1, nx02 = n2, h)
```

Arguments

Xin	Data matrix with p rows (representing variables) and n columns (representing observations).
M	Number of components to be extracted.
Win	Initial guess for the unmixing matrix W. Dimensions need to be M x M.
tol	Tolerance used to establish the convergence of the algorithm.
maxit	Maximum number of iterations.
nmaxit	If the algorithm does not converge, it is run again with a new initial guess for the unmixing matrix W. This operation is done nmaxit times.
unmixing.estimate	The method used in the unmixing matrix estimation step. The two allowed choices are "eigenvector" and "newton" (see Details).
n1	Number of rows of the lattice.
n2	Number of columns of the lattice.
nx01	Number of rows of the lattice where the spectral density is evaluate. Default value is n1.
nx02	Number of columns of the lattice where the spectral density is evaluate. Default value is n2.
h	Kernel bandwidth used for the nonparametric estimation of the sources spectral densities.

Details

In the Independent Component Analysis approach, the data matrix X is considered to be a linear combination of independent components, i.e. $X = AS$, where rows of S contain the unobserved realizations of the independent components and A is a linear mixing matrix. According to classical ICA procedures data matrix X is centered and, then, whitened by projecting the data onto its principal component directions, i.e. $X \to KX = \widetilde{X}$ where K is a `M x p` pre-whitening matrix. The scICA algorithm then estimates the unmixing matrix W, with $W\widetilde{X} = S$, according to the procedure described below. Then, defining $\widetilde{W} = WK$, the mixing matrix A is recovered through $A = \widetilde{W}^T(\widetilde{W}\widetilde{W}^T)^{-1}$.

Spatial colored Independent Component Analysis assumes that the independent sources are spatial stochastic processes on a lattice. To perform ICA, the Whittle log-likelihood is exploited. In particular the log-likelihood is written in function of the unmixing matrix W and the spectral densities f_{S_j} of the spatial autocorrelated sources as follows:

$$l(W, \boldsymbol{f_S}; \widetilde{X}) = \sum_{j=1}^{p}\sum_{k=1}^{n}\left(\frac{\boldsymbol{e}_j^T W \widetilde{\boldsymbol{f}}(r_k, \widetilde{X}) W^T \boldsymbol{e}_j}{f_{S_j}(r_k)} + \ln f_{S_j}(r_k)\right) + n \ln|\det(W)|.$$

Due to whitening, W is orthogonal and the last term of the objective function can be dropped. The orthogonality of the unmixing matrix W can be imposed in two different ways, setting the argument `unmixing.estimate`. In this way the estimate of the unmixing matrix W can be found according two different procedures:

- as described in Shen et al. (2014). A penalty term is added to the objective function. In particular $\tau \boldsymbol{w}_j' C_j \boldsymbol{w}_j$, where $\boldsymbol{w}_j'$ is the jth column of W, $C_j = \sum_{k \neq j} \boldsymbol{w}_k \boldsymbol{w}_k'$ and τ is a tuning parameter. The matrix C_j provides an orthogonality constraint in the sense that $\boldsymbol{w}_j' C_j \boldsymbol{w}_j = \sum_{k \neq j}$. In this way the objective function assumes a symmetric and positive-definite form and the argmin correspond to the lower eigenvalue. This choice is obtained setting `unmixing.matrix = "eigenvector"`.
- as described in Lee et al. (2011). The orthogonality constraint is considered performing the minimization of the objective function according a Newton-Raphson method with Lagrange multiplier. This choice is obtained setting `unmixing.matrix = "newton"`.

Independently from the choice of the technique to minimize the objective function, the scICA algorithm is based on an iterative procedure. While the Amari error is greater than `tol` and the number of iteration is less or equal than `maxit`, the two following steps are repeated:

- nonparametric estimation of the sources spectral density through a multidimensional local linear kernel estimator $\widehat{m}_{LK}$ (see Shen et al. (2014) for further details).
- estimate the unmixing matrix W according the method selected in `unmixing.estimate`.

Value

A list containing the following components:

`W`	Estimate of the `M x M` unmixing matrix in the whitened space.
`K`	pre-whitening matrix that projects data onto the first `M` principal components. Dimensions are `M x p`.
`A`	Estimate of the `p x M` mixing matrix.

S	Estimate of the M x n source matrix.
X	Original p x n data matrix.
iter	number of iterations.
NInv	number of times the algorithm is rerun after it does not achieve convergence.
den	Estimate of the spectral density of the sources. Dimensions are M x n.

Note

Note that source matrix S and spectral density matrix den are n x M matrices. Every row, that should be in a n1 x n2 grid, has been vectorized in a n vector by column, with n = n1 x n2.

Author(s)

Lee, S., Shen, H., Truong, Y. and Zanini, P.

References

Shen, H., Truong, Y., Zanini, P. (2014). Independent Component Analysis for Spatial Processes on a Lattice. *MOX report 38/2014*, Department of Mathematics, Politecnico di Milano.

Lee, S., Shen, H., Truong, Y., Lewis, M., Huang, X. (2011). Independent Component Analysis Involving Autocorrelated Sources With an Application to Funcional Magnetic Resonance Imaging. *Journal of the American Statistical Association*, **106**, 1009–1024.

See Also

cICA

Examples

```
## Not run:

require(fastICA)

n1=20
n2=20
M=2

# Fist source

sigma1=2
S1=matrix(0,n1,n2)
for (i in 1:n1){
S1[i,]=rnorm(n2,i*2,0.2)
}
for (j in 1:n2){
S1[,j]=S1[,j]+rnorm(n1,j*2,0.2)
}
S1=S1+matrix(rnorm(n1*n2,0,sigma1),n1,n2)

image(1:n2,1:n1,t(S1[n1:1,]),xlab="",ylab="",main="Source 1")
```

```
contour(1:n2,1:n1,t(S1[n1:1,]),add=TRUE)

# Second source

val1=1
val2=1.2
val3=1.5
val4=2
sigma2=0.1

S2=matrix(0,n1,n2)
S2[2:5,4:10]=val1
S2[3:4,5:9]=val3
S2[13:18,16:19]=val2
S2[14:17,17:18]=val4
S2=S2+matrix(rnorm(n1*n2,0,sigma2),n1,n2)

image(1:n2,1:n1,t(S2[n1:1,]),xlab="",ylab="",main="Source 2")
contour(1:n2,1:n1,t(S2[n1:1,]),add=TRUE)

# Generating data matrix X

A = rerow(matrix(runif(M^2)-0.5,M,M))
W = solve(A)
S=rbind(as.vector(S1),as.vector(S2))
X = A %*% S

# Solving Blind Source Separation problem with three different methods

cica = cICA(X,tol=0.001)
## scica = scICA(X,n1=n1,n2=n2,h=(2*pi/10),tol=0.001)
fica = fastICA(t(X),2)

amari_distance(t(A),t(cica$A))
## amari_distance(t(A),t(scica$A))
amari_distance(t(A),fica$A)

Shat1=cica$S
## Shat2=scica$S
Shat3=t(fica$S)

par(mfrow=c(2,2))
image(t(S1[n1:1,]),xlab="",ylab="")
contour(t(S1[n1:1,]),add=TRUE)
image(t(S2[n1:1,]),xlab="",ylab="")
contour(t(S2[n1:1,]),add=TRUE)
image(t(matrix(Shat1[1,],n1,n2)[n1:1,]),xlab="",ylab="")
contour(t(matrix(Shat1[1,],n1,n2)[n1:1,]),add=TRUE)
image(t(matrix(Shat1[2,],n1,n2)[n1:1,]),xlab="",ylab="")
contour(t(matrix(Shat1[2,],n1,n2)[n1:1,]),add=TRUE)

## par(mfrow=c(2,2))
## image(t(S1[n1:1,]),xlab="",ylab="")
```

```
## contour(t(S1[n1:1,]),add=TRUE)
## image(t(S2[n1:1,]),xlab="",ylab="")
## contour(t(S2[n1:1,]),add=TRUE)
## image(t(matrix(Shat2[1,],n1,n2)[n1:1,]),xlab="",ylab="")
## contour(t(matrix(Shat2[1,],n1,n2)[n1:1,]),add=TRUE)
## image(t(matrix(Shat2[2,],n1,n2)[n1:1,]),xlab="",ylab="")
## contour(t(matrix(Shat2[2,],n1,n2)[n1:1,]),add=TRUE)

par(mfrow=c(2,2))
image(t(S1[n1:1,]),xlab="",ylab="")
contour(t(S1[n1:1,]),add=TRUE)
image(t(S2[n1:1,]),xlab="",ylab="")
contour(t(S2[n1:1,]),add=TRUE)
image(t(matrix(Shat3[1,],n1,n2)[n1:1,]),xlab="",ylab="")
contour(t(matrix(Shat3[1,],n1,n2)[n1:1,]),add=TRUE)
image(t(matrix(Shat3[2,],n1,n2)[n1:1,]),xlab="",ylab="")
contour(t(matrix(Shat3[2,],n1,n2)[n1:1,]),add=TRUE)

## End (Not run)
```

Index