Package: fICA (via r-universe)

September 20, 2024

Type Package

Title Classical, Reloaded and Adaptive FastICA Algorithms

Version 1.1-2

Date 2021-12-08

Imports stats, JADE, Rcpp (>= 0.11.0)

LinkingTo Rcpp, RcppArmadillo

Suggests BSSasymp

Description Algorithms for classical symmetric and deflation-based FastICA, reloaded deflation-based FastICA algorithm and an algorithm for adaptive deflation-based FastICA using multiple nonlinearities. For details, see Miettinen et al. (2014) <doi:10.1109/TSP.2014.2356442> and Miettinen et al. (2017) <doi:10.1016/j.sigpro.2016.08.028>. The package is described in Miettinen, Nordhausen and Taskinen (2018) <doi:10.32614/RJ-2018-046>.

License GPL (>= 2)

NeedsCompilation yes

Author Jari Miettinen [aut] (<https://orcid.org/0000-0002-3270-7014>), Klaus Nordhausen [aut, cre] (<https://orcid.org/0000-0002-3758-8501>), Hannu Oja [aut], Sara Taskinen [aut] (<https://orcid.org/0000-0001-9470-7258>)

Maintainer Klaus Nordhausen <klaus.k.nordhausen@jyu.fi>

Repository CRAN

Date/Publication 2021-12-08 12:50:02 UTC

Contents

`fICA-package` *Classical, Reloaded and Adaptive FastICA Algorithms*

Description

Algorithms for classical symmetric and deflation-based FastICA, reloaded deflation-based FastICA algorithm and an algorithm for adaptive deflation-based FastICA using multiple nonlinearities. For details, see Miettinen et al. (2014) <doi:10.1109/TSP.2014.2356442> and Miettinen et al. (2017) <doi:10.1016/j.sigpro.2016.08.028>. The package is described in Miettinen, Nordhausen and Taskinen (2018) <doi:10.32614/RJ-2018-046>.

Details

Package:	fICA
Type:	Package
Version:	1.1-2
Date:	2021-12-08
License:	GPL (>= 2)

Author(s)

Jari Miettinen, Klaus Nordhausen, Hannu Oja, Sara Taskinen

Maintainer: Klaus Nordhausen <klaus.k.nordhausen@jyu.fi>

References

Miettinen, J., Nordhausen, K. and Taskinen, S. (2018), fICA: FastICA Algorithms and Their Improved Variants, The R Journal, ***10***, *148–158, <doi:10.32614/RJ-2018-046>.*

`adapt_fICA` *Adaptive Deflation-based FastICA Method for Independent Component Analysis*

Description

The adaptive deflation-based FastICA method for the independent component problem. The function estimates the unmixing matrix by finding, for each component separately, the best nonlinearity from a set of nonlinearities.

Usage

```
adapt_fICA(X, gs=gf, dgs=dgf, name=gnames, kj=0, inR=TRUE,
           eps=1e-06, maxiter=100)
```

Arguments

X	a numeric data matrix. Missing values are not allowed.
gs	a list of functions containing the nonlinearities.
dgs	a list of functions containing the first derivatives of the nonlinearities.
name	a list of strings containing the names of the nonlinearities.
kj	defines the initial estimate of the unmixing matrix, see details.
inR	a logical which indicates whether R or C is used for computations. If FALSE, the default set of nonlinearities gf is used.
eps	convergence tolerance.
maxiter	maximum number of iterations.

Details

The algorithm first finds initial estimates of the sources. The method to find the estimates is decided by the choice of the argument kj. If the value of kj is an integer between 1 and number of the sources, then the method is kj-JADE, otherwise it is FOBI. For the meaning of the value kj used as kj-JADE, see the help for k_JADE.

Value

A list with class 'bss' containing the following components:

W	estimated unmixing matrix.
gs	nonlinearities that were available.
used_gs	nonlinearities, in order of appearance, that were used. The last row of the unmixing matrix follows directly from the other rows, and hence no nonlinearity is connected to it.
alphas	the statistics for the choice of the nonlinearities.
init_est	method that was used for the initial estimate (FOBI or k-JADE).
S	estimated source components standardized to have mean 0 and unit variances.

Author(s)

Jari Miettinen

References

Hyvarinen, A. and Oja, E. (1997), A fast fixed-point algorithm for independent component analysis, Neural Computation, *vol. 9, 1483–1492.*

Nordhausen, K., Ilmonen, P., Mandal, A., Oja, H. and Ollila, E. (2011), Deflation-based FastICA reloaded, in Proc. "19th European Signal Processing Conference 2011 (EUSIPCO 2011)", *Barcelona, 1854–1858.*

Miettinen, J., Nordhausen, K., Oja, H. and Taskinen, S. (2014), Deflation-based FastICA with adaptive choices of nonlinearities, IEEE Transactions on Signal Processing, *62(21), 5716–5724.*

See Also

fICA, nonlinearities, FOBI, k_JADE

Examples

```
A <- matrix(rnorm(9),3,3)
s1 <- rt(1000,6)
s2 <- rexp(1000,1)
s3 <- runif(1000)

S <- cbind(s1,s2,s3)
X <- S %*% t(A)

res1<-adapt_fICA(X, inR=FALSE)
res1
coef(res1)
plot(res1)

require(JADE)
MD(coef(res1),A)

# changing the set of candidate nonlinearities

?nonlinearities
g <- function(x){x^2}
dg <- function(x){2*x}
gf_new <- c(gf[-c(5,8,10)],g)
dgf_new <- c(dgf[-c(5,8,10)],g)
gnames_new <- c(gnames[-c(5,8,10)],"skew")

res2<-adapt_fICA(X, gs=gf_new, dgs=dgf_new, name=gnames_new)
res2
MD(coef(res2),A)

# reloaded FastICA using tanh

res3<-adapt_fICA(X, gs=gf[2], dgs=dgf[2], name=gnames[2])
res3
MD(coef(res3),A)
```

compute_alphas *Estimation of Alphas in the Asymptotic Covariance Matrix of the Deflation-based FastICA Estimator*

Description

Using the estimates of the independent components, the function computes for a given set of nonlinearities, the quantities (alphas). Alphas determine the choices of the nonlinearities and in which order the nonlinearities are used in the adaptive deflation-based FastICA method.

Usage

```
compute_alphas(Z, gs=gf, dgs=dgf, name=gnames)
```

Arguments

Z	a numeric matrix of the estimated independent components, which should be standardized so that the mean is zero and the covariance matrix is the identity matrix.
gs	a vector of functions containing the nonlinearities.
dgs	a vector of functions containing the first derivatives of the nonlinearities.
name	a vector of strings containing the names of the nonlinearities.

Details

See the references.

Value

A matrix where the ith row gives the estimates of alphas for the ith nonlinearity and the jth column corresponds to the jth component of Z.

Author(s)

Jari Miettinen

References

Hyvarinen, A. and Oja, E. (1997), A fast fixed-point algorithm for independent component analysis, Neural Computation, *vol. 9, 1483–1492.*

Nordhausen, K., Ilmonen, P., Mandal, A., Oja, H. and Ollila, E. (2011), Deflation-based FastICA reloaded, in Proc. "19th European Signal Processing Conference 2011 (EUSIPCO 2011)", *Barcelona, 1854–1858.*

Miettinen, J., Nordhausen, K., Oja, H. and Taskinen, S. (2014), Deflation-based FastICA with adaptive choices of nonlinearities, IEEE Transactions on Signal Processing, *62(21), 5716–5724.*

See Also

fICA, nonlinearities, FOBI, k_JADE

Examples

```
A <- matrix(rnorm(9),3,3)
s1 <- rt(1000,6)
s2 <- rexp(1000,1)
s3 <- runif(1000)

S <- cbind(s1,s2,s3)
X <- S %*% t(A)

Sest <- fICA(X,method="def")$S

compute_alphas(Sest, gs=gf[1:3], dgs=dgf[1:3], name=gnames[1:3])
```

`fICA` *Symmetric and Deflation-based FastICA Methods for Independent Component Analysis*

Description

The symmetric and deflation-based FastICA methods for the independent component problem. The function estimates the unmixing matrix using a single nonlinearity g.

Usage

```
fICA(X, g="tanh", dg=NULL, G=NULL, init=NULL, n.init = 1, method="sym2",
     inR=TRUE, eps=1e-06, maxiter=1000)
```

Arguments

`X`	a numeric data matrix. Missing values are not allowed.
`g`	the nonlinearity, tanh by default, see details.
`dg`	the first derivative of the nonlinearity, see details.
`G`	the integral function of the nonlinearity, see details.
`init`	a numeric matrix for the initial value of the algorithm
`n.init`	a positive integer for the number of initial values in symmetric algorithms, see details.
`method`	squared symmetric (`"sym2"`), symmetric (`"sym"`) or deflation-based (`"def"`)
.	
`inR`	a logical which indicates whether R or C is used for computations, see details.
`eps`	convergence tolerance.
`maxiter`	maximum number of iterations.

Details

The deflation-based FastICA estimate depends on the initial value of the unmixing matrix. This is due to the fact that the algorithm does not always find the global maximum of the objective function, but it may stop to local maxima as well. Hence, the deflation-based FastICA estimate is affine equivariant only if the initial value is affine equivariant. Therefore, we recommend the use of reloaded_fICA or adapt_fICA.

The standard nonlinearities can be chosen by `g="pow3"`, `g="tanh"` or `g="gaus"`. These estimates can be computed either in R or in C (except the squared symmetric). In order to use some other nonlinearity, one has to give the nonlinearity and its derivative as functions (and the integral function when squared symmetric FastICA is computed), and the computations have to be performed in R.

If the symmetric or squared symmetric algorithm does not converge, one can choose `n.init=k` for some $k > 1$. Then up to k random initial values are tried, and if none of them gives a convergent run of the algorithm, the function returns the matrix which gave the largest value of the objective function over all $k \times maxiter$ steps.

Value

A list with class 'bss' containing the following components:

`W`	estimated unmixing matrix.
`g`	nonlinearity used.
`method`	symmetric or deflation-based.
`S`	estimated source components standardized to have mean 0 and unit variances.

Author(s)

Jari Miettinen

References

Hyvarinen, A. and Oja, E. (1997), A fast fixed-point algorithm for independent component analysis, Neural Computation, *vol. 9, 1483–1492.*

Miettinen, J., Nordhausen, K., Oja, H., Taskinen, S. and Virta, J. (2015), The squared symmetric FastICA estimator, Signal Processing, *vol. 131, 402–411.*

See Also

adapt_fICA, reloaded_fICA, nonlinearities

Examples

```
# creating some toy data
A<- matrix(rnorm(9),3,3)
s1 <- rt(1000,6)
s2 <- rexp(1000,1)
s3 <- runif(1000)

S <- cbind(s1,s2,s3)
```

```
X <- S %*% t(A)

# tanh is the default nonlinearity
res1<-fICA(X,method="sym")
coef(res1)
plot(res1)
require(JADE)
MD(coef(res1),A)

# nonlinearity pow3 is chosen as follows
res2<-fICA(X,g="pow3",method="sym")
coef(res2)
require(JADE)
MD(coef(res2),A)

# nonlinearity from gf is chosen as follows
res3<-fICA(X,g=gf[[6]],dg=dgf[[6]],method="sym")
coef(res3)
require(JADE)
MD(coef(res3),A)
```

`nonlinearities` *Set of Nonlinearities for Adaptive Deflation-based FastICA Method*

Description

The default set of nonlinearities with their first derivatives and names used in `adapt_fICA`.

Usage

```
gf

dgf

Gf

gnames
```

Details

The set of nonlinearities includes both well-known functions (*pow3*, *tanh* and *gaus*) and the ones suggested in Miettinen et al. (2014).

The object `gf` contains the nonlinearities which are:

gf[[1]]	pow3	x^3
gf[[2]]	tanh	$tanh(x)$
gf[[3]]	gaus	$exp(-(x)^2/2)$

gf[[4]]	lt0.6	$(x+0.6)_-^2$
gf[[5]]	rt0.6	$(x-0.6)_+^2$
gf[[6]]	bt	$(x)_+^2-(x)_-^2$
gf[[7]]	bt0.2	$(x-0.2)_+^2-(x+0.2)_-^2$
gf[[8]]	bt0.4	$(x-0.4)_+^2-(x+0.4)_-^2$
gf[[9]]	bt0.6	$(x-0.6)_+^2-(x+0.6)_-^2$
gf[[10]]	bt0.8	$(x-0.8)_+^2-(x+0.8)_-^2$
gf[[11]]	bt1.0	$(x-1.0)_+^2-(x+1.0)_-^2$
gf[[12]]	bt1.2	$(x-1.2)_+^2-(x+1.2)_-^2$
gf[[13]]	bt1.4	$(x-1.4)_+^2-(x+1.4)_-^2$
gf[[14]]	bt1.6	$(x-1.6)_+^2-(x+1.6)_-^2$

The objects `dgf`, `Gf` and `gnames` contain the corresponding first derivatives, integrals and names in the same order.

For skew sources *lt0.6* and *rt0.6* combined are more efficient than the commonly used *skew*. The rest of the functions are useful for example for sources with multimodal density functions.

The user can easily submit a own set or modify the set suggested here. See the example below and the examples in `adapt_fICA`.

Author(s)

Jari Miettinen

References

Hyvarinen, A. and Oja, E. (1997), A fast fixed-point algorithm for independent component analysis, Neural Computation, *vol. 9, 1483–1492.*

Miettinen, J., Nordhausen, K., Oja, H. and Taskinen, S. (2014), Deflation-based FastICA with adaptive choices of nonlinearities, IEEE Transactions on Signal Processing, *62(21), 5716–5724.*

See Also

adapt_fICA

Examples

```
# leaving out functions from the default set and adding a new function
g <- function(x){x^2}
dg <- function(x){2*x}
G <- function(x){x^3/3}

gf_new <- c(gf[-c(6,8,10)],g)
dgf_new <- c(dgf[-c(6,8,10)],dg)
Gf_new <- c(Gf[-c(6,8,10)],G)
gnames_new <- c(gnames[-c(6,8,10)],"skew")
```

`reloaded_fICA` *Reloaded Deflation-based FastICA Method for Independent Component Analysis*

Description

The reloaded and the affine equivariant deflation-based FastICA method for the independent component problem. The function estimates the rows of the unmixing matrix one by one, either in asymptotically optimal order, or so that the objective function is globally maximized at each stage.

Usage

```
reloaded_fICA(X, g="tanh", dg=NULL, G=NULL, kj=0, method="alpha",
              inR=TRUE, eps=1e-06, maxiter=100)
```

Arguments

`X`	a numeric data matrix. Missing values are not allowed.
`g`	the nonlinearity, tanh by default, see details.
`dg`	the first derivative of the nonlinearity, see details.
`G`	the integral of the nonlinearity, see details.
`kj`	defines the initial estimate of the unmixing matrix, see details.
`method`	"alpha" or "G", see details.
`inR`	a logical which indicates whether R or C is used for computations, see details.
`eps`	convergence tolerance.
`maxiter`	maximum number of iterations.

Details

The algorithm first finds initial estimates of the sources. The method to find the estimates is decided by the choice of the argument `kj`. If the value of `kj` is an integer between 1 and number of the sources, then the method is `kj`-JADE, otherwise it is FOBI. For the meaning of the value `kj` used as `kj`-JADE, see the help for `k_JADE`.

The function has the argument `method`, which determines the extraction order of the components. If `method="alpha"`, the components are found in asymptotically optimal order, that is, the reloaded deflation-based FastICA estimate is computed. If `method="G"`, the objective function is globally maximized at each stage.

The standard nonlinearities can be chosen by `g="pow3"`, `g="tanh"` or `g="gaus"`. These estimates can be computed either in R or in C. In order to use some other nonlinearity, one has to give the nonlinearity and its derivative (and integral if `method="G"`) as functions, and the computations have to be performed in R.

Value

A list with class 'bss' containing the following components:

W	estimated unmixing matrix.
g	nonlinearity used.
alphas	the statistics for the choice of the nonlinearities.
init_est	method that was used for the initial estimate (FOBI or k-JADE).
S	estimated source components standardized to have mean 0 and unit variances.

Author(s)

Jari Miettinen

References

Hyvarinen, A. and Oja, E. (1997), A fast fixed-point algorithm for independent component analysis, Neural Computation, *vol. 9, 1483–1492.*

Nordhausen, K., Ilmonen, P., Mandal, A., Oja, H. and Ollila, E. (2011), Deflation-based FastICA reloaded, in Proc. "19th European Signal Processing Conference 2011 (EUSIPCO 2011)", *Barcelona, 1854–1858.*

See Also

fICA, nonlinearities, FOBI, k_JADE

Examples

```
# creating some toy data
A<- matrix(rnorm(9),3,3)
s1 <- rt(1000,6)
s2 <- rexp(1000,1)
s3 <- runif(1000)

S <- cbind(s1,s2,s3)
X <- S %*% t(A)

# tanh is the default nonlinearity
res1<-reloaded_fICA(X)
coef(res1)
plot(res1)
require(JADE)
MD(coef(res1),A)

# nonlinearity pow3 and method "G" is chosen as follows
res2<-reloaded_fICA(X,g="pow3",method="G")
coef(res2)
require(JADE)
MD(coef(res2),A)
```

Index