Package ‘fastICA’

December 11, 2024

Version 1.2-7

Date 2024-12-10

Title FastICA Algorithms to Perform ICA and Projection Pursuit

Depends R (>= 4.0.0)

Suggests MASS

Description Implementation of FastICA algorithm to perform Independent
Component Analysis (ICA) and Projection Pursuit.

License GPL-2 | GPL-3

NeedsCompilation yes

Author Jonathan L Marchini [aut],
Chris Heaton [aut],
Brian Ripley [aut, cre]

Maintainer Brian Ripley <Brian.Ripley@R-project.org>

Repository CRAN

Date/Publication 2024-12-11 10:04:09 UTC

Contents

`fastICA` *FastICA algorithm*

Description

This is an R and C code implementation of the FastICA algorithm of Aapo Hyvarinen et al. (`https://www.cs.helsinki.fi/u/ahyvarin/`) to perform Independent Component Analysis (ICA) and Projection Pursuit.

Usage

```
fastICA(X, n.comp, alg.typ = c("parallel","deflation"),
        fun = c("logcosh","exp"), alpha = 1.0, method = c("R","C"),
        row.norm = FALSE, maxit = 200, tol = 1e-04, verbose = FALSE,
        w.init = NULL)
```

Arguments

`X`	a data matrix with `n` rows representing observations and `p` columns representing variables.
`n.comp`	number of components to be extracted
`alg.typ`	if `alg.typ == "parallel"` the components are extracted simultaneously (the default). if `alg.typ == "deflation"` the components are extracted one at a time.
`fun`	the functional form of the G function used in the approximation to neg-entropy (see 'details').
`alpha`	constant in range [1, 2] used in approximation to neg-entropy when `fun == "logcosh"`
`method`	if `method == "R"` then computations are done exclusively in R (default). The code allows the interested R user to see exactly what the algorithm does. if `method == "C"` then C code is used to perform most of the computations, which makes the algorithm run faster. During compilation the C code is linked to an optimized BLAS library if present, otherwise stand-alone BLAS routines are compiled.
`row.norm`	a logical value indicating whether rows of the data matrix `X` should be standardized beforehand.
`maxit`	maximum number of iterations to perform.
`tol`	a positive scalar giving the tolerance at which the un-mixing matrix is considered to have converged.
`verbose`	a logical value indicating the level of output as the algorithm runs.
`w.init`	Initial un-mixing matrix of dimension `c(n.comp, n.comp)`. If `NULL` (default) then a matrix of normal r.v.'s is used.

Details

Independent Component Analysis (ICA)

The data matrix X is considered to be a linear combination of non-Gaussian (independent) components i.e. X = SA where columns of S contain the independent components and A is a linear mixing matrix. In short ICA attempts to 'un-mix' the data by estimating an un-mixing matrix W where XW = S.

Under this generative model the measured 'signals' in X will tend to be 'more Gaussian' than the source components (in S) due to the Central Limit Theorem. Thus, in order to extract the independent components/sources we search for an un-mixing matrix W that maximizes the non-gaussianity of the sources.

In FastICA, non-gaussianity is measured using approximations to neg-entropy (J) which are more robust than kurtosis-based measures and fast to compute.

The approximation takes the form

$$J(y) = [E\{G(y)\} - E\{G(v)\}]^2$$

where v is a N(0,1) r.v.

The following choices of G are included as options $G(u) = \frac{1}{\alpha} \log \cosh(\alpha u)$ and $G(u) = -\exp(u^2/2)$.

Algorithm

First, the data are centered by subtracting the mean of each column of the data matrix X.

The data matrix is then ‘whitened’ by projecting the data onto its principal component directions i.e. X -> XK where K is a pre-whitening matrix. The number of components can be specified by the user.

The ICA algorithm then estimates a matrix W s.t XKW = S . W is chosen to maximize the neg-entropy approximation under the constraints that W is an orthonormal matrix. This constraint ensures that the estimated components are uncorrelated. The algorithm is based on a fixed-point iteration scheme for maximizing the neg-entropy.

Projection Pursuit

In the absence of a generative model for the data the algorithm can be used to find the projection pursuit directions. Projection pursuit is a technique for finding ‘interesting’ directions in multi-dimensional datasets. These projections and are useful for visualizing the dataset and in density estimation and regression. Interesting directions are those which show the least Gaussian distribution, which is what the FastICA algorithm does.

Value

A list containing the following components

`X`	pre-processed data matrix
`K`	pre-whitening matrix that projects data onto the first `n.comp` principal components.
`W`	estimated un-mixing matrix (see definition in details)
`A`	estimated mixing matrix
`S`	estimated source matrix

Author(s)

J L Marchini and C Heaton

References

A. Hyvarinen and E. Oja (2000) Independent Component Analysis: Algorithms and Applications, *Neural Networks*, **13(4-5)**:411-430

See Also

`ica.R.def`, `ica.R.par`

Examples

```
#---------------------------------------------------
#Example 1: un-mixing two mixed independent uniforms
#---------------------------------------------------

S <- matrix(runif(10000), 5000, 2)
A <- matrix(c(1, 1, -1, 3), 2, 2, byrow = TRUE)
X <- S %*% A

a <- fastICA(X, 2, alg.typ = "parallel", fun = "logcosh", alpha = 1,
             method = "C", row.norm = FALSE, maxit = 200,
             tol = 0.0001, verbose = TRUE)

par(mfrow = c(1, 3))
plot(a$X, main = "Pre-processed data")
plot(a$X %*% a$K, main = "PCA components")
plot(a$S, main = "ICA components")

#--------------------------------------------
#Example 2: un-mixing two independent signals
#--------------------------------------------

S <- cbind(sin((1:1000)/20), rep((((1:200)-100)/100), 5))
A <- matrix(c(0.291, 0.6557, -0.5439, 0.5572), 2, 2)
X <- S %*% A

a <- fastICA(X, 2, alg.typ = "parallel", fun = "logcosh", alpha = 1,
             method = "R", row.norm = FALSE, maxit = 200,
             tol = 0.0001, verbose = TRUE)

par(mfcol = c(2, 3))
plot(1:1000, S[,1 ], type = "l", main = "Original Signals",
     xlab = "", ylab = "")
plot(1:1000, S[,2 ], type = "l", xlab = "", ylab = "")
plot(1:1000, X[,1 ], type = "l", main = "Mixed Signals",
     xlab = "", ylab = "")
plot(1:1000, X[,2 ], type = "l", xlab = "", ylab = "")
plot(1:1000, a$S[,1 ], type = "l", main = "ICA source estimates",
     xlab = "", ylab = "")
plot(1:1000, a$S[, 2], type = "l", xlab = "", ylab = "")

#-----------------------------------------------------------
#Example 3: using FastICA to perform projection pursuit on a
#           mixture of bivariate normal distributions
#-----------------------------------------------------------

if(require(MASS)){
x <- mvrnorm(n = 1000, mu = c(0, 0), Sigma = matrix(c(10, 3, 3, 1), 2, 2))
x1 <- mvrnorm(n = 1000, mu = c(-1, 2), Sigma = matrix(c(10, 3, 3, 1), 2, 2))
X <- rbind(x, x1)

a <- fastICA(X, 2, alg.typ = "deflation", fun = "logcosh", alpha = 1,
              method = "R", row.norm = FALSE, maxit = 200,
              tol = 0.0001, verbose = TRUE)

par(mfrow = c(1, 3))
plot(a$X, main = "Pre-processed data")
plot(a$X %*% a$K, main = "PCA components")
plot(a$S, main = "ICA components")
}
```

`ica.R.def` *R code for FastICA using a deflation scheme*

Description

R code for FastICA using a deflation scheme in which the components are estimated one by one. This function is called by the fastICA function.

Usage

```
ica.R.def(X, n.comp, tol, fun, alpha, maxit, verbose, w.init)
```

Arguments

`X`	data matrix
`n.comp`	number of components to be extracted
`tol`	a positive scalar giving the tolerance at which the un-mixing matrix is considered to have converged.
`fun`	the functional form of the G function used in the approximation to negentropy.
`alpha`	constant in range [1,2] used in approximation to negentropy when `fun == "logcosh"`
`maxit`	maximum number of iterations to perform
`verbose`	a logical value indicating the level of output as the algorithm runs.
`w.init`	Initial value of un-mixing matrix.

Details

See the help on `fastICA` for details.

Value

The estimated un-mixing matrix W.

Author(s)

J L Marchini and C Heaton

See Also

`fastICA`, `ica.R.par`

ica.R.par *R code for FastICA using a parallel scheme*

Description

R code for FastICA using a parallel scheme in which the components are estimated simultaneously. This function is called by the fastICA function.

Usage

```
ica.R.par(X, n.comp, tol, fun, alpha, maxit, verbose, w.init)
```

Arguments

`X`	data matrix.
`n.comp`	number of components to be extracted.
`tol`	a positive scalar giving the tolerance at which the un-mixing matrix is considered to have converged.
`fun`	the functional form of the G function used in the approximation to negentropy.
`alpha`	constant in range [1,2] used in approximation to negentropy when `fun == "logcosh"`.
`maxit`	maximum number of iterations to perform.
`verbose`	a logical value indicating the level of output as the algorithm runs.
`w.init`	Initial value of un-mixing matrix.

Details

See the help on `fastICA` for details.

Value

The estimated un-mixing matrix W.

Author(s)

J L Marchini and C Heaton

See Also

`fastICA`, `ica.R.def`

Index