Package ‘ivaBSS’

May 19, 2022

Type Package
Title Tools for Independent Vector Analysis
Version 1.0.0
Date 2022-05-03
Imports stats, graphics, BSSprep
Suggests LaplacesDemon
Encoding UTF-8
Maintainer Mika Sipilä <mika.e.sipila@student.jyu.fi>
Description Independent vector analysis (IVA) is a blind source separation (BSS) model where several datasets are jointly unmixed. This package provides several methods for the unmixing together with some performance measures. For details, see Anderson et al. (2011) <doi:10.1109/TSP.2011.2181836> and Lee et al. (2007) <doi:10.1016/j.sigpro.2007.01.010>.
License GPL (>= 3)
RoxygenNote 7.1.2
NeedsCompilation no
Author Mika Sipilä [aut, cre], Klaus Nordhausen [aut] (<https://orcid.org/0000-0002-3758-8501>), Sara Taskinen [aut] (<https://orcid.org/0000-0001-9470-7258>)
Repository CRAN
Date/Publication 2022-05-19 17:00:02 UTC

R topics documented:

ivaBSS-package .................................................. 2
avg_ISI .......................................................... 3
coeff.iva ......................................................... 4
components.iva .................................................. 6
fastIVA .......................................................... 7
jbss_achieved .................................................... 10
joint_ISI .......................................................... 11
NewtonIVA ......................................................... 13
Description

Independent vector analysis (IVA) is a blind source separation (BSS) model where several datasets are jointly unmixed. This package provides several methods for the unmixing together with some performance measures. For details, see Anderson et al. (2011) \(<\text{doi:10.1109/TSP.2011.2181836}\>\) and Lee et al. (2007) \(<\text{doi:10.1016/j.sigpro.2007.01.010}\>\).

Details

The package contains tools for independent vector analysis. The main functions to perform IVA are "IVANewton" and "fastIVA". "NewtonIVA" performs Newton update based IVA and "fastIVA" performs fixed-point iteration based IVA. Both of the algorithms have multiple options for source density models.

Author(s)

Authors: Mika Sipilä, Klaus Nordhausen, Sara Taskinen

Maintainer: Mika Sipilä

References


**avg_ISI**

---

**avg_ISI**  
**Average Intersymbol Inference**

---

**Description**

Calculates the average intersymbol inference for two sets of matrices.

**Usage**

`avg_ISI(W, A)`

**Arguments**

- `W`  

- `A`  

**Details**

The function returns the average intersymbol inference for the set of estimated unmixing matrices and the set of true mixing matrices. The average ISI gets the value between 0 and 1, where 0 is the optimal result. The average ISI is calculated as the mean ISI over each dataset separately. The average ISI does not take the permutation of the estimated sources into account.

**Value**

Numeric value between 0 and 1, where 0 is the optimal result indicating that the sources are separated perfectly in each dataset.

**Author(s)**

Mika Sipilä

**References**


**See Also**

`joint_ISI`, `jbss_achieved`
Examples

```r
# Mixing matrices and unmixing matrices generated
# from standard normal distribution
P <- 4; D <- 4;
W <- array(rnorm(P * P * D), c(P, P, D))
A <- array(rnorm(P * P * D), c(P, P, D))

avg.ISI(W, A)

if (require("LaplacesDemon")) {
  # Generate sources from multivariate Laplace distribution
  P <- 4; N <- 1000; D <- 4;
  S <- array(NA, c(P, N, D))

  for (i in 1:P) {
    U <- array(rnorm(D * D), c(D, D))
    Sigma <- crossprod(U)
    S[i, , ] <- rmvl(N, rep(0, D), Sigma)
  }

  # Generate mixing matrices from standard normal distribution
  A <- array(rnorm(P * P * D), c(P, P, D))

  # Generate mixtures
  X <- array(NA, c(P, N, D))
  for (d in 1:D) {
    X[, , d] <- A[, , d] %*% S[, , d]
  }

  # Estimate sources and unmixing matrices
  res_G <- NewtonIVA(X, source_density = "gaussian")
  avg.ISI(coef(res_G), A)
}
```

---

**coef.iva**

*Coefficient of the Object of Class iva*

**Description**

`coef` method for class "iva".

**Usage**

```r
# S3 method for class 'iva'
coef(object, which.dataset = NA, ...)
```

**Arguments**

- `object` - an object of class "iva", usually the result of a call to `NewtonIVA` or `fastIVA`.  

which.dataset  positive integer. Provides the index in case the unmixing matrix only for a specific data set is desired. Default is to return all unmixing matrices.

... further arguments are not used.

Details

Returns the unmixing matrices for all datasets or only for the requested dataset.

Value

Unmixing matrix or all unmixing matrices of the object of class "iva". If a single unmixing matrix is requested, it is an array with dimension [P, P] and if all unmixing matrices are requested, it is an array with dimension [P, P, D].

Author(s)

Mika Sipilä

See Also

NewtonIVA, fastIVA

Examples

```r
if (require("LaplacesDemon")) {
  # Generate sources from multivariate Laplace distribution
  P <- 4; N <- 1000; D <- 4;
  S <- array(NA, c(P, N, D))

  for (i in 1:P) {
    U <- array(rnorm(D * D), c(D, D))
    Sigma <- crossprod(U)
    S[i, , ] <- rmvl(N, rep(0, D), Sigma)
  }

  # Generate mixing matrices from standard normal distribution
  A <- array(rnorm(P * P * D), c(P, P, D))

  # Generate mixtures
  X <- array(NaN, c(P, N, D))
  for (d in 1:D) {
    X[, , d] <- A[, , d] %*% S[, , d]
  }

  # Estimate sources and unmixing matrices
  res_G <- NewtonIVA(X, source_density = "gaussian")

  # All D unmixing matrices
  coef(res_G)

  # The unmixing matrix for the second dataset
  coef(res_G)[, , 2]
}
```
components.iva  

Components of the Object of Class iva

Description

Returns the estimated source components of object of class "iva".

Usage

components.iva(object, which.dataset = NA, ...)

Arguments

- **object**: an object of class "iva", usually the result of a call to `NewtonIVA` or `fastIVA`.
- **which.dataset**: positive integer. Provides the index in case the unmixing matrix only for a specific data set is desired. Default is to return all unmixing matrices.
- **...**: further arguments are not used.

Details

Returns the estimated source components for all datasets or only for the requested dataset.

Value

Estimated source components for requested dataset or for all datasets of the object of class "iva". If a single dataset is requested, it is an array with dimension \([P, N]\) and if all datasets are requested, it is an array with dimension \([P, N, D]\).

Author(s)

Mika Sipilä

See Also

`NewtonIVA`, `fastIVA`

Examples

```r
if (require("LaplacesDemon")) {
  # Generate sources from multivariate Laplace distribution
  P <- 4; N <- 1000; D <- 4;
  S <- array(NA, c(P, N, D))
  for (i in 1:P) {
    U <- array(rnorm(D * D), c(D, D))
    ```
fastIVA

---

**Fast Fixed-point IVA Algorithm**

**Description**

The algorithm estimates the sources from multiple dependent datasets jointly using their observed mixtures. The estimation is done by maximizing the independence between the sources, when the estimated unmixing matrices are restricted to be orthogonal. The options for different source densities are provided.

**Usage**

```r
fastIVA(X, source_density="laplace_diag", student_df=1,
       max_iter = 1024, eps = 1e-6, W_init = NA, verbose = FALSE)
```

**Arguments**

- **X** numeric data array containing the observed mixtures with dimension \([P, N, D]\), where \(P\) is the dimension of the observed dataset, \(N\) is the number of the observations and \(D\) is the number of the datasets. The number of datasets \(D\) should be at least 2. Missing values are not allowed.
- **source_density** string to determine which source density model should be used. The options are "laplace_diag", "student" or "entropic". For more information see the details section.
student_df integer. The degree of freedom for multivariate Student's distribution. Used only if source_density = "student".

max_iter positive integer, used to define the maximum number of iterations for algorithm to run. If max_iter is reached, the unmixing matrices of the last iteration are used.

eps convergence tolerance, when the convergence measure is smaller than eps, the algorithm stops.


verbose logical. If TRUE the convergence measure is printed during the learning process.

Details
The algorithm uses fixed-point iteration to estimate the multivariate source signals from their observed mixtures. The elements of the source signals, or the datasets, should be dependent of each other to achieve the estimates where the sources are aligned in same order for each dataset. If the datasets are not dependent, the sources can still be separated but not necessarily aligned. This algorithm restricts the estimates unmixing matrices to be orthogonal. For more of the fast fixed-point IVA algorithm, see Lee, I. et al (2007).

The source density model should be selected to match the density of the true source signals. When source_density = "laplace_diag", the multivariate Laplace source density model with diagonal covariance structure is used. When source_density = "entropic", the approximated entropy based source density model is used. For more about multivariate Laplace and entropic source density models, see Lee, I. et al (2007). When source_density = "student" the multivariate Student's source density model is used, for more see Liang, Y. et al (2013).

The algorithm assumes that observed signals are multivariate, i.e. the number of datasets $D >= 2$. The estimated signals are zero mean and scaled to unit variance.

Value
An object of class "iva".

S The estimated source signals with dimension $[P, N, D]$. The estimated source signals are zero mean with unit variance.

W The estimated unmixing matrices with dimension $[P, P, D]$.

W_whitened The estimated unmixing matrices with dimension $[P, P, D]$ for whitened data.


X_means The means for each observed mixture with dimension $[P, D]$.

niter The number of iterations that the algorithm did run.

converged Logical value which tells if the algorithm converged.

source_density The source density model used.

N The number of observations.

D The number of datasets.

P The number of sources.
The degree of freedom for Student’s source density model.

The function call.

The name of the variable containing the observed mixtures.

Author(s)

Mika Sipilä

References


See Also

NewtonIVA

Examples

```r
if (require("LaplacesDemon")) {
  # Generate sources from multivariate Laplace distribution
  P <- 2; N <- 1000; D <- 5;
  S <- array(NA, c(P, N, D))
  for (i in 1:P) {
    S[i, , ] <- rmvl(N, rep(0, D), diag(D))
  }

  # Generate mixing matrices from standard normal distribution
  A <- array(rnorm(P * P * D), c(P, P, D))

  # Generate mixtures
  X <- array(NaN, c(P, N, D))
  for (d in 1:D) {
    X[, , d] <- A[, , d] %*% S[, , d]
  }

  # Estimate sources and unmixing matrices
  res <- fastIVA(X)
}
```
**Description**

The function calculates if the joint blind source separation (JBSS) is achieved.

**Usage**

```r
jbss_achieved(W, A)
```

**Arguments**


**Details**

The function calculates if the joint blind source separation is achieved. JBSS is considered achieved when the location of maximum absolute values of each row of gain matrix \( G[,,d] = W[,,d] \times A[,,d] \) is unique within the dataset, but shared between the datasets 1, ..., \( D \). The first indicates that the sources are separated within dataset and the second indicates that the estimated sources are aligned in same order for each dataset.

**Value**

Logical. If `TRUE` the JBSS is considered achieved.

**Author(s)**

Mika Sipilä

**References**


**See Also**

`joint_ISI`, `avg_ISI`
Examples

# Mixing matrices and unmixing matrices generated
# from standard normal distribution
P <- 4; D <- 4;
W <- array(rnorm(P * P * D), c(P, P, D))
A <- array(rnorm(P * P * D), c(P, P, D))

jbss_achieved(W, A)

if (require("LaplacesDemon")) {
  # Generate sources from multivariate Laplace distribution
  P <- 4; N <- 1000; D <- 4;
  S <- array(NA, c(P, N, D))

  for (i in 1:P) {
    U <- array(rnorm(D * D), c(D, D))
    Sigma <- crossprod(U)
    S[i, , ] <- rmvl(N, rep(0, D), Sigma)
  }

  # Generate mixing matrices from standard normal distribution
  A <- array(rnorm(P * P * D), c(P, P, D))

  # Generate mixtures
  X <- array(NaN, c(P, N, D))
  for (d in 1:D) {
    X[, , d] <- A[, , d] %*% S[, , d]
  }

  # Estimate sources and unmixing matrices
  res_G <- NewtonIVA(X, source_density = "gaussian")
  jbss_achieved(coef(res_G), A)
}

---

joint_ISI  

Joint Intersymbol Inference

Description

Calculates the joint intersymbol inference for two sets of matrices.

Usage

joint_ISI(W, A)

Arguments

W  Array of unmixing matrices with dimension [P, P, D].
A  Array of true mixing matrices with dimension [P, P, D].
Details

The function returns the joint intersymbol inference for the set of estimated unmixing matrices and the set of true mixing matrices. The joint ISI gets the value between 0 and 1, where 0 is the optimal result. The joint ISI calculates the average intersymbol inference over each dataset as well as penalizes if the sources are not aligned in same order for each dataset.

Value

Numeric value between 0 and 1, where 0 is the optimal result indicating that the sources are separated perfectly and aligned in same order in each dataset.

Author(s)

Mika Sipilä

References


See Also

avg_ISI, jbss_achieved

Examples

# Mixing matrices and unmixing matrices generated
# from standard normal distribution
P <- 4; D <- 4;
W <- array(rnorm(P * P * D), c(P, P, D))
A <- array(rnorm(P * P * D), c(P, P, D))

joint_ISI(W, A)

if (require("LaplacesDemon")) {
  # Generate sources from multivariate Laplace distribution
  P <- 4; N <- 1000; D <- 4;
  S <- array(NA, c(P, N, D))
  for (i in 1:P) {
    U <- array(rnorm(D * D), c(D, D))
    Sigma <- crossprod(U)
    S[i, , ] <- rmvl(N, rep(0, D), Sigma)
  }

  # Generate mixing matrices from standard normal distribution
  A <- array(rnorm(P * P * D), c(P, P, D))

  # Generate mixtures
  X <- array(NaN, c(P, N, D))
  for (d in 1:D) {
X[, , d] <- A[, , d] %*% S[, , d]
}

# Estimate sources and unmixing matrices
res_G <- NewtonIVA(X, source_density = "gaussian")
joint_ISI(coef(res_G), A)
}

---

**NewtonIVA**  
*Newton Update Based IVA Algorithm*

**Description**

The algorithm estimates the sources from multiple dependent datasets jointly using their observed mixtures. The estimation is done by maximizing the independence between the sources. The options for different source densities are provided.

**Usage**

```
NewtonIVA(X, source_density="laplace", student_df=1,  
init = "default", max_iter = 1024, eps = 1e-6, W_init = NA,  
step_size=1, step_size_min = 0.1, alpha = 0.9, verbose = FALSE)
```

**Arguments**

- **X**
  numeric data array containing the observed mixtures with dimension \([P, N, D]\), where \(P\) is the dimension of the observed dataset, \(N\) is the number of the observations and \(D\) is the number of the datasets. The number of datasets \(D\) should be at least 2. Missing values are not allowed.

- **source_density**
  string to determine which source density model should be used. The options are "laplace", "laplace_diag", "gaussian" or "student". For more information see the details section.

- **student_df**
  integer. The degree of freedom for multivariate Student's distribution. Used only if `source_density = "student"`.

- **init**
  string, to determine how to initialize the algorithm. The options are "default", "IVA-G+fastIVA", "IVA-G", "fastIVA" or "none". For more information see the details section.

- **max_iter**
  positive integer, used to define the maximum number of iterations for algorithm to run. If `max_iter` is reached, the unmixing matrices of the last iteration are used.

- **eps**
  convergence tolerance, when the convergence measure is smaller than `eps`, the algorithm stops.

- **W_init**
  numeric array of dimension \([P, P, D]\) containing initial unmixing matrices. If not set, initialized with identity matrices.

- **step_size**
  initial step size for Newton step, should be between 0 and 1, default is 1.
step_size_min  the minimum step size.
alpha         multiplier for how much to decrease step size when convergence is not getting smaller.
verbose       logical. If TRUE the convergence measure is printed during the learning process.

Details

The algorithm uses Newton update together with decoupling trick to estimate the multivariate source signals from their observed mixtures. The elements of the source signals, or the datasets, should be dependent of each other to achieve the estimates where the sources are aligned in same order for each dataset. If the datasets are not dependent, the sources can still be separated but not necessarily aligned. The algorithm does not assume the unmixing matrices to be orthogonal. For more of the nonorthogonal Newton update based IVA algorithm, see Anderson, M. et al (2011) and Anderson, M. (2013).

The source density model should be selected to match the density of the true source signals. When source_density = "laplace", the multivariate Laplace source density model is used. This is the most flexible choice as it takes both second-order and higher-order dependence into account.

When source_density = "laplace_diag", the multivariate Laplace source density model with diagonal covariance structure is used. Multivariate diagonal Laplace source density model should be considered only when the sources are mainly higher-order dependent. It works best when the number of sources is significantly less than the number of datasets.

When source_density = "gaussian" the multivariate Gaussian source density model is used. This is the superior choice in terms of computation power and should be used when the sources are mostly second-order dependent.

When source_density = "student" the multivariate Student's source density model is used. Multivariate Student's source density model should be considered only when the sources are mainly higher-order dependent. It works best when the number of sources is significantly less than the number of datasets.

The init parameter defines how the algorithm is initialized. When init = "default", the default initialization is used. As default the algorithm is initialized using init = "IVA-G+fastIVA" when source_density is "laplace", "laplace_diag" or "student", and using init = "none" when source_density = "gaussian".

When init = "IVA-G+fastIVA", the algorithm is initialized using first the estimated unmixing matrices of IVA-G, which is NewtonIVA with source_density = "gaussian", to initialize fastIVA algorithm. Then the estimated unmixing matrices W of fastIVA are used as initial unmixing matrices for NewtonIVA. IVA-G is used to solve the permutation problem of aligning the source estimates when ever the true sources are second-order dependent. If the true sources are not second-order dependent, fastIVA is used as backup as it solves the permutation problem more regularly than NewtonIVA when the sources are purely higher-order dependent. When the sources possess any second-order dependence, IVA-G also speeds the computation time up a lot. This option should be used whenever there is no prior information about the sources and source_density is either "laplace", "laplace_diag" or "student".

When init = "IVA-G", the estimated unmixing matrices of IVA-G are used to initialize this algorithm. This option should be used if the true sources are expected to possess any second-order dependence and source_density is not "gaussian".
When `init = "fastIVA"`, the estimated unmixing matrices of fastIVA algorithm is used to initialize this algorithm. This option should be used if the true sources are expected to possess only higher-order dependence. For more details, see fastIVA.

When `init = "none"`, the unmixing matrices are initialized randomly from standard normal distribution.

The algorithm assumes that observed signals are multivariate, i.e. the number of datasets $D \geq 2$. The estimated signals are zero mean and scaled to unit variance.

**Value**

An object of class "iva".

- `S` The estimated source signals with dimension `[P, N, D]`. The estimated source signals are zero mean with unit variance.
- `W_whitened` The estimated unmixing matrices with dimension `[P, P, D]` for whitened data.
- `X_means` The means for each observed mixture with dimension `[P, D]`.
- `niter` The number of iterations that the algorithm did run.
- `converged` Logical value which tells if the algorithm converged.
- `source_density` The source density model used.
- `N` The number of observations.
- `D` The number of datasets.
- `P` The number of sources.
- `student_df` The degree of freedom for Student’s source density model.
- `call` The function call.
- `DNAME` The name of the variable containing the observed mixtures.

**Author(s)**

Mika Sipilä

**References**


Examples

if (require("LaplacesDemon")) {
  # Generate sources from multivariate Laplace distribution
  P <- 4; N <- 1000; D <- 4;
  S <- array(NA, c(P, N, D))

  for (i in 1:P) {
    U <- array(rnorm(D * D), c(D, D))
    Sigma <- crossprod(U)
    S[i, , ] <- rmvl(N, rep(0, D), Sigma)
  }

  # Generate mixing matrices from standard normal distribution
  A <- array(rnorm(P * P * D), c(P, P, D))

  # Generate mixtures
  X <- array(NaN, c(P, N, D))
  for (d in 1:D) {
    X[, , d] <- A[, , d] %*% S[, , d]
  }

  # Estimate sources and unmixing matrices
  res_G <- NewtonIVA(X, source_density = "gaussian")
}

plot.iva

Plotting an Object of Class iva

Description

plot method for the class "iva".

Usage

## S3 method for class 'iva'
plot(x, which.dataset = NA, which.source = NA,
     type = "l", xlabs = c(), ylabs = c(), colors = c(),
     oma = c(1, 1, 0, 0), mar = c(2, 2, 1, 1), ...)}

Arguments

x An object of class "iva", usually the result of a call to NewtonIVA or fastIVA.
which.dataset Positive integer to determine which dataset is returned. If not set, returns all datasets.
which.source  Positive integer to determine which dataset is returned. If not set, returns all datasets.

type  1-character string giving the type of plot desired. For details, see plot.

xlabs  Vector containing the labels for x-axis.

ylabs  Vector containing the labels for y-axis.

colors  Vector containing the colors for each plot.

oma  A vector of the form c(bottom, left, top, right) giving the size of the outer margins in lines of text. For more details, see par.

mar  A numerical vector of the form c(bottom, left, top, right) which gives the number of lines of margin to be specified on the four sides of the plot. For more details, see par.

...  Further arguments passed to plot function.

Details

Plots either all estimated sources of the object of class "iva" or the estimates for specific dataset and/or source.

Value

No return value, called for plotting the estimated sources of the object of class "iva".

Author(s)

Mika Sipilä

See Also

NewtonIVA, fastIVA

Examples

if (require("LaplacesDemon")) {
  # Generate sources from multivariate Laplace distribution
  P <- 4; N <- 1000; D <- 4;
  S <- array(NA, c(P, N, D))
  for (i in 1:P) {
    U <- array(rnorm(D * D), c(D, D))
    Sigma <- crossprod(U)
    S[i, , ] <- rmvl(N, rep(0, D), Sigma)
  }

  # Generate mixing matrices from standard normal distribution
  A <- array(rnorm(P * P * D), c(P, P, D))

  # Generate mixtures
  X <- array(NA, c(P, N, D))
  for (d in 1:D) {

# Estimate sources and unmixing matrices
res_G <- NewtonIVA(X, source_density = "gaussian")

# Plot all estimated sources
plot(res_G)

# Plot the source estimates for the first dataset only
plot(res_G, which.dataset = 1)

# Plot the source estimates for the second source only
plot(res_G, which.source = 2)

# Plot the source estimate of the second dataset and third source
plot(res_G, which.dataset = 2, which.source = 3, type = "p")

# Plot all source estimates with custom colors and labels
plot(res_G, col=c(rep(1, 4), rep(2, 4), rep(3, 4), rep(4, 4)),
     xlab = c("Subject 1", "Subject 2", "Subject 3", "Subject 4"),
     ylab = c("Channel 1", "Channel 2", "Channel 3", "Channel 4"))

---

**predict.iva**  
*Predict Method for Object of Class iva*

**Description**

Predict the new source estimates best on fitted object of "iva" class.

**Usage**

```r
## S3 method for class 'iva'
predict(object, newdata, which.dataset = NA, ...)
```

**Arguments**

- **object**: An object of class "iva", usually the result of a call to `NewtonIVA` or `fastIVA`.
- **newdata**: A numeric data array containing new observed mixtures. Either with dimension \([P, N, D]\) (if `which.dataset = NA`) or \([P, N]\), where \(P\) is the number of sources, \(N\) is the number of observations and \(D\) is the number of datasets.
- **which.dataset**: Positive integer to determine which dataset is returned. If not set, returns all datasets.
- **...**: Further arguments are not used.

**Details**

The function calculates the source estimates for new observed mixtures based on the model fitted originally. The estimates are zero mean and scaled to unit variance.
Value

Numeric array containing the estimated sources with dimension \([P, N]\) if \texttt{which.dataset} is provided and with dimension \([P, N, D]\) if \texttt{which.dataset} is not provided.

Author(s)

Mika Sipilä

See Also

\texttt{NewtonIVA, fastIVA}

Examples

```r
if (require("LaplacesDemon")) {
  # Generate sources from multivariate Laplace distribution
  P <- 4; N <- 1000; D <- 4;
  S <- array(NA, c(P, N, D))
  sigmas <- list()

  for (i in 1:P) {
    U <- array(rnorm(D * D), c(D, D))
    sigmas[[i]] <- crossprod(U)
    S[i, , ] <- rmvl(N, rep(0, D), sigmas[[i]])
  }

  # Generate mixing matrices from standard normal distribution
  A <- array(rnorm(P * P * D), c(P, P, D))

  # Generate mixtures
  X <- array(NaN, c(P, N, D))
  for (d in 1:D) {
    X[, , d] <- A[, , d] %*% S[, , d]
  }

  # Estimate sources and unmixing matrices
  res_G <- NewtonIVA(X, source_density = "gaussian")

  # Generate new observations
  N_new <- 10
  S_new <- array(NA, c(P, N_new, D))
  for (i in 1:P) {
    S_new[i, , ] <- rmvl(N_new, rep(0, D), sigmas[[i]])
  }

  X_new <- array(NaN, c(P, N_new, D))
  for (d in 1:D) {
    X_new[, , d] <- A[, , d] %*% S_new[, , d]
  }

  # Get source estimates for the new observations
  pred <- predict(res_G, X_new)
}
# Get source estimates for only the second dataset
pred2 <- predict(res_G, X_new[, , 2], which.dataset = 2)
}

---

## S3 method for class 'iva'
print(x, ...)  

### Arguments

- **x**: An object of class "iva", usually the result of a call to \texttt{NewtonIVA} or \texttt{fastIVA}.
- **...**: Further arguments are not used.

### Details

The function prints all information of "iva" object, except the estimated source signals.

### Value

No return value, called for printing information of the object of class "iva".

### Author(s)

Mika Sipilä

### See Also

\texttt{NewtonIVA, fastIVA}

### Examples

```r
if (require("LaplacesDemon")) {
  # Generate sources from multivariate Laplace distribution
  P <- 4; N <- 1000; D <- 4;
  S <- array(NA, c(P, N, D))

  for (i in 1:P) {
    U <- array(rnorm(D * D), c(D, D))
    Sigma <- crossprod(U)
    S[i, , ] <- rmvl(N, rep(0, D), Sigma)
  }
  # Get source estimates for only the second dataset
  pred2 <- predict(res_G, X_new[, , 2], which.dataset = 2)
}
```
# Generate mixing matrices from standard normal distribution
A <- array(rnorm(P * P * D), c(P, P, D))

# Generate mixtures
X <- array(NaN, c(P, N, D))
for (d in 1:D) {
  X[,, d] <- A[,, d] %*% S[,, d]
}

# Estimate sources and unmixing matrices
res_G <- NewtonIVA(X, source_density = "gaussian")
print(res_G)

summary.iva

Summarize an Object of Class iva

Description

summary method for the class "iva".

Usage

## S3 method for class 'iva'
summary(object, ...)

Arguments

object An object of class "iva", usually the result of a call to NewtonIVA or fastIVA.
...
Further arguments are not used.

Details

The function print all the information of the "iva" object except the estimated sources and the estimated unmixing matrices.

Value

No return value, called for summarizing the object of class "iva".

Author(s)

Mika Sipilä

See Also

NewtonIVA, fastIVA
Examples

```r
if (require("LaplacesDemon")) {
  # Generate sources from multivariate Laplace distribution
  P <- 4; N <- 1000; D <- 4;
  S <- array(NA, c(P, N, D))

  for (i in 1:P) {
    U <- array(rnorm(D * D), c(D, D))
    Sigma <- crossprod(U)
    S[i, , ] <- rmvl(N, rep(0, D), Sigma)
  }

  # Generate mixing matrices from standard normal distribution
  A <- array(rnorm(P * P * D), c(P, P, D))

  # Generate mixtures
  X <- array(NaN, c(P, N, D))
  for (d in 1:D) {
    X[, , d] <- A[, , d] %*% S[, , d]
  }

  # Estimate sources and unmixing matrices
  res_G <- NewtonIVA(X, source_density = "gaussian")
  summary(res_G)
}
```
Index

* methods
  coef.iva, 4
  components.iva, 6
  plot.iva, 16
  predict.iva, 18
  print.iva, 20
  summary.iva, 21

* multivariate
  avg_ISI, 3
  fastIVA, 7
  jbss_achieved, 10
  joint_ISI, 11
  NewtonIVA, 13

* package
  ivaBSS-package, 2

* print
  print.iva, 20

avg_ISI, 3, 10, 12

coef.iva, 4
components.iva, 6

fastIVA, 4–6, 7, 15–21

ivaBSS(ivaBSS-package), 2
ivaBSS-package, 2

jbss_achieved, 3, 10, 12
joint_ISI, 3, 10, 11

NewtonIVA, 4–6, 9, 13, 16–21

par, 17
plot, 17
plot.iva, 16
predict.iva, 18
print.iva, 20

summary.iva, 21