Package 'icapca'

February 20, 2015

Version 1.1

Date 2014-10-19

Title Mixed ICA/PCA

Author Roger Woods <rwoods@ucla.edu>

Maintainer Roger Woods <rwoods@ucla.edu>

Description Implements mixed ICA/PCA model for blind source separation, potentially with inclusion of Gaussian sources

License Unlimited

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-10-20 19:18:09

R topics documented:

`ica_pca` *Performs mixed ICA/PCA on the input matrix*

Description

This function will decompose a matrix into the matrix product A*S such that the non-Gaussian rows of S are maximally statistically independent. Unlike traditional ICA, a Gaussian sub-space can be included in the model. Gaussian components cannot be separated from one another and are decomposed using PCA criteria.

Usage

```
ica_pca(x, inf_crit = "unc", components = 0, center = TRUE,
 subgaussian_range = NULL, supergaussian_range = NULL,
 gaussian_range = NULL,
 hinted_subgaussian_sources = NULL,
 hinted_supergaussian_sources = NULL,
 hinted_unspecified_sources = NULL,
 seed = 0, offset_random = 0, fold = 0, xval_epsilon = 0.0,
 desired_initialization = 0,
 sample_order = NULL, sample_offset = 0, samples = 0)
```

Arguments

x	the matrix to decompose using mixed ICA/PCA
inf_crit	the bias correction to apply to loglikelihoods, one of: 'unc', 'aic', 'bic', 'xval' or 'caicj'
components	if less than the number of rows of x, x will be reduced to components rows by singular value decomposition before mixed ICA/PCA
center	if TRUE, x will be centered by subtracting the mean of each row
subgaussian_range	the number of sub-Gaussian sources to model (single integer) or the minimum and maximum number of sub-Gaussian sources to model (two integers)
supergaussian_range	the number of super-Gaussian sources to model (single integer) or the minimum and maximum number of super-Gaussian sources to model (two integers)
gaussian_range	the number of Gaussian sources to model (single integer) or the minimum and maximum number of Gaussian sources to model (two integers)
hinted_subgaussian_sources	a matrix of initializing sub-Gaussian sources to include in model (number of columns must match x)
hinted_supergaussian_sources	a matrix of initializing super-Gaussian sources to include in model (number of columns must match x)
hinted_unspecified_sources	a matrix of non-Gaussian sources to include in model (number of columns must match x)
seed	if >0, seeds the R default random number generator to randomly rotate inputs after preprocessing PCA
offset_random	if seed>0, specifies the number or random rotation matrices to skip before selecting the one to apply
fold	if inf_crit == 'xval' and fold>0, specifies k for k-fold cross-validation; fold==0 defaults to leave-one-out cross validation
xval_epsilon	if inf_crit == 'xval' and xval_epsilon != 0.0, omits initializations that produce identical loglikelihood and det(W) values

`desired_initialization`
: if desired_initialization > 0 only the specified initialization is used

`sample_order`
: if inf_crit = 'xval', an integer vector of permuted column vectors of length dim(x)[2]

`sample_offset`
: if samples > 0 and inf_crit = 'xval', an integer specifying the offset into sample_order

`samples`
: if samples > 0 and inf_crit = 'xval', an integer specifying the number of samples to use from sample_order

Details

The recommended value for inf_crit is 'xval' which uses cross-validation. However, this option is computationally intensive. The default value 'unc' provides no correction. The 'aic' option uses the Akaike Information Criterion (AIC) but is typically an unacceptably biased estimate, especially when Gaussian sources are modeled as non-Gaussian. The 'bic' option uses the Bayes Information Criterion (BIC) which is also dubious. If all sources are Gaussian, the 'caicj' option will provide a good estimate, but this correction is not applicable if any non-Gaussian sources are modeled. If a non-zero value is specified for xval_epsilon, initializations that produce very similar uncorrrected log likelihoods are assumed identical and only evaluated once through cross-validation.

If inf_crit is 'xval', fold == 0 and samples > 0, only a subset of all possible leave-one-out cross validations will be performed and the results will be rescaled to estimate the result that would have been obtained if all possible leave-one-out cross validations had been performed. In this case, sample_order should be a vector containing a permutation of the integers from 1 to the number of columns in x and sample_offset should be an integer indexing an offset into this vector. The specified number of samples will be taken from sample_order after skipping sample_offset entries. The result will be used as the indices of the samples left out, one-by-one, for cross-validation estimation. Parallelization can be achieved by running multiple instances with differing sample_offset values as needed to cover all samples and then computing a weighted average (weighted based on the number of samples) of the result.

If inf_crit is 'xval' and fold > 0, k-fold cross validation will be performed. Unless the ordering of samples is known to be random, a random sample_order should be provided since the estimated result will be order dependent and the validity of k-fold cross validation depends on the order being random. The values of fold and samples cannot both be non-zero. The value of fold must be an exact divisor of the number of columns in x and cannot be equal to one. Smaller values of fold will run more quickly, but will likely provide less accurat results relative to larger values since bias is calculated using a smaller number of sample. When fold is equal to the number of columns in x, the result is leave-one-out cross validation, which is more efficiently computed using fold = 0.

If seed is specified, R's random number generator is set to the default and seeded with the specified value. If the state of the random number generator needs to be restored to its initial value that preceeded the call to ica_pca, save the value of .Random.seed before calling and restore it afterwards.

Value

`s`
: the matrix of sources

`loglikelihood`
: the adjusted log-likelihood of the best fitting model

`distribution`
: the type of source used to model each row of s (0=subgaussian, 1=supergaussian, 2=gaussian)

variance	the variance associated with each row of s
probability	the relative probability of the best model containing a given number of Gaussian sources, starting with zero Gaussian sources in the first element
subgaussian_range	the range of subgaussian sources actually modeled
supergaussian_ranges	the range of supergaussian sources actually modeled
gaussian_range	the range of Gaussian sources actually modeled

Author(s)

Roger P. Woods, M.D.

References

Woods RP, Hansen LK, Strother S. How many separable sources? Model selection in independent components analysis (in preparation)

Examples

```
x<-matrix(nrow=4, ncol=150)
x[1,]<-iris[[1]]
x[2,]<-iris[[2]]
x[3,]<-iris[[3]]
x[4,]<-iris[[4]]
result<-ica_pca(x,inf_crit='xval', subgaussian_range=c(1,1), supergaussian_range=c(3,3))
area<-as.double(dim(x)[2])

subg<-function(x){return ((area*plot_width/sqrt(pi*exp(1)))*exp(-x*x)*cosh(sqrt(2)*x))}
superg<-function(x){return ((area*plot_width)/(2*cosh(pi*x/2.0)))}
gaussian<-function(x){return ((area*plot_width/sqrt(2*pi))*exp(-x*x))}
distributions<-list(subg, superg, gaussian)

par(mfrow=c(2,2))

plot_params<-hist(result$s[1,], ylim=c(0,45))
plot_width=plot_params$breaks[2]-plot_params$breaks[1]
par(new=TRUE)
plot(distributions[[result$distribution[1]+1]],
    plot_params$breaks[1],
    plot_params$breaks[length(plot_params$breaks)],
    ylim=c(0,45),
    xlab="",
    ylab="")
par(new=FALSE)

plot_params<-hist(result$s[2,], ylim=c(0,45))
plot_width=plot_params$breaks[2]-plot_params$breaks[1]
par(new=TRUE)
plot(distributions[[result$distribution[2]+1]],
```

```
    plot_params$breaks[1],
    plot_params$breaks[length(plot_params$breaks)],
    ylim=c(0,45),
    xlab="",
    ylab="")
par(new=FALSE)


plot_params<-hist(result$s[3,], ylim=c(0,45))
plot_width=plot_params$breaks[2]-plot_params$breaks[1]
par(new=TRUE)
plot(distributions[[result$distribution[3]+1]],
    plot_params$breaks[1],
    plot_params$breaks[length(plot_params$breaks)],
    ylim=c(0,45),
    xlab="",
    ylab="")
par(new=FALSE)

plot_params<-hist(result$s[4,], ylim=c(0,90))
plot_width=plot_params$breaks[2]-plot_params$breaks[1]
par(new=TRUE)
plot(distributions[[result$distribution[4]+1]],
    plot_params$breaks[1],
    plot_params$breaks[length(plot_params$breaks)],
    ylim=c(0,90),
    xlab="",
    ylab="")
par(new=FALSE)

x_centered<-x
for (i in 1:(dim(x))[1]) x_centered[i,]=x_centered[i,]-mean(x_centered[i,])
w=t(qr.solve(t(x_centered),t(result$s)))

residuals<-w%*%x_centered-result$s
```

`initializations` *Computes number of initializations to be performed by ica_pca*

Description

For a given number of sub-Gaussian and super-Gaussian sources and Gaussian components, the function ica_pca will initialize the model multiple times. This function will compute the number of initializations that will be performed.

Usage

```
initializations(subgaussians, supergaussians, gaussians)
```

Arguments

subgaussians	the number of sub-Gaussian sources in the model
supergaussians	the number of super-Gaussian sources in the model
gaussians	the number of Gaussian sources in the model

Details

If the number of initializations is small (less than 50 to 100), the ica_pca function may fail to identify the optimal model; models with small numbers of initializations should be run several times using different values for seed and/or offset_random. As the number of sources and components gets large (e.g., with totals more than 10) the number of initializations grows quickly. To a first approximation, computation time is proportional to the number of initializations.

Value

returns the number of initializations

Author(s)

Roger P. Woods, M.D.

Examples

```
initializations(4,1,2)
```

Index