Package 'GFA'

August 4, 2017

Type Package

Title Group Factor Analysis

Version 1.0.3

Date 2017-08-04

Author Eemeli Leppäaho [aut, cre],
Seppo Virtanen [aut],
Muhammad Ammad-ud-din [ctb],
Suleiman A Khan [ctb],
Tommi Suvitaival [ctb],
Inka Saarinen [ctb],
Samuel Kaski [ctb]

Maintainer Eemeli Leppäaho <eemeli.leppaaho@gmail.com>

Description Factor analysis implementation for multiple data sources, i.e., for groups of variables. The whole data analysis pipeline is provided, including functions and recommendations for data normalization and model definition, as well as missing value prediction and model visualization. The model group factor analysis (GFA) is inferred with Gibbs sampling, and it has been presented originally by Virtanen et al. (2012), and extended in Klami et al. (2015) <DOI:10.1109/TNNLS.2014.2376974> and Bunte et al. (2016) <DOI:10.1093/bioinformatics/ tails, see the citation info.

License MIT + file LICENSE

Encoding latin1

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-08-04 08:53:59 UTC

R topics documented:

GFA-package *Group factor analysis.*

Description

GFA does factor analysis for multiple data sets having matched observations, for exploratory or predictive data analysis.

Details

The posterior distribution of GFA model parameters can be inferred with function `gfa`, once the priors have been defined with `getDefaultOpts`. The priors are widely customizable, with two recommended setups: (i) dense group-sparse components (default; similar to package CCAGFA that provides variational Bayesian inference for the same model) and (ii) components interpretable as biclusters. It is recommended to preprocess the data with function `normalizeData`. Functions are provided for predicting missing data, choosing a prior for the residual noise, identifying robust components and visualizing the inferred model. A simple toy example of the pipeline is provided as demo(GFApipeline), and a more elaborate one as demo(GFAexample). Finally, the experiment presented in (Bunte, Leppaaho, Saarinen and Kaski: Sparse group factor analysis for biclustering of multiple data sources, Bioinformatics, 32(16):2457–2463, 2016) can be replicated with demo(GFAdream). Most of the computational complexity of the package is related to matrix operations, which can be parallelized inherently by using e.g. OpenBLAS libraries.

Examples

```
#Data generation
X <- matrix(rnorm(20*3),20,3)                    #Latent variables
W <- matrix(rnorm(30*3),30,3)                    #Projection matrix
Y <- tcrossprod(X,W) + matrix(rnorm(20*30),20,30) #Observations
Y <- sweep(Y, MARGIN=2, runif(30), "+")          #Feature means
Y <- list(Y[,1:10], Y[,11:30])                   #Data grouping
#Model inference and visualization
norm <- normalizeData(Y, type="center")          #Centering
opts <- getDefaultOpts()                         #Model options
#Fast runs for the demo, default options recommended in general
opts[c("iter.burnin", "iter.max")] <- c(500, 1000)
res <- gfa(norm$train, K=5, opts=opts)           #Model inference
rec <- reconstruction(res)                       #Reconstruction
recOrig <- undoNormalizeData(rec, norm)          #... to original space
vis <- visualizeComponents(res, Y, norm)         #Visualization
```

getDefaultOpts *A function for generating the default priors of GFA model*

Description

getDefaultOpts returns the priors of GFA

Usage

```
getDefaultOpts(bicluster = FALSE)
```

Arguments

bicluster	Use binary sparsity priors in both the data modes? If FALSE (default), the components will be dense in the data sources, but group-sparse, i.e., each component is active in a (potentially different) subset of the data sources. If TRUE, binary sparsity is inferred for each data sample and feature, resulting in each component to be interpretable as a multi-source bicluster.

Details

This function returns options defining the model's high-level structure (sparsity priors) and the model's hyperparameters, defining uninformative priors. We recommend keeping these as provided, with one exception: if the uninformative prior of the noise residual (tau) seems to result in an overly complex model (no components become shut down even if the initial K is set high), risking overfitting, we recommend using function informativeNoisePrior to adjust the priors.

Value

A list with the following model options:

tauGrouped	If TRUE (default), data views have separate noise precisions, otherwise each feature has.
normalLatents	If TRUE, X will have a normal prior; if FALSE X, will have a spike-and-slab prior.
spikeW	Sparsity prior of W. "group"=group sparsity, "element"= element-wise sparsity with shared hyperparameter across views, "shared"= element-wise sparsity with no grouping.
ARDW	ARD prior type for W, determining the scale of the inferred components. "shared"=same scale for all the data sources, "grouped" (default)= separate scale for each data source, "element"=separate scale for each feature.
ARDLatent	ARD prior type for X: "shared" (default)=shared scale for all the samples, "element"=separate scale for each sample.

imputation	Missing value imputation type: "Bayesian" (default)=proper Bayesian handling of missing values. "conservative"=missing values result in smaller parameter scale, which can be useful if tricky missing value structure causes exaggerated imputed values with the default setting (which can also be dealt with informative priors for alpha and beta).
iter.max	The total number of Gibbs sampling steps (default 5000).
iter.saved	The number of saved posterior samples (default 100).
iter.burnin	The number of burn-in samples (default 2500).
init.tau	The initial noise precision. High values imply initializing the model with an adequate number of components. Default 1000.
sampleZ	When to start sampling spike and slab parameters (default: Gibbs sample 1).
prior.alpha_0t	The shape parameter of tau's prior (default 10).
prior.beta_0t	The rate parameter of tau's prior (default 10).
prior.alpha_0	The shape parameter of alpha's prior (default 10).
prior.beta_0	The rate parameter of alpha's prior (default 01).
prior.alpha_0X	The shape parameter of beta's prior (default 10).
prior.beta_0X	The rate parameter of beta's prior (default 1).
prior.beta	Bernoulli prior for the spike-and-slab prior of W (counts for 1s and 0s; default c(1,1)).
prior.betaX	Bernoulli prior for the possible spike-and-slab prior of X (default c(1,1)).
verbose	The verbosity level. 0=no printing, 1=moderate printing, 2=maximal printing (default 1).
convergenceCheck	Check for the convergence of the data reconstruction, based on the Geweke diagnostic (default FALSE).
save.posterior	A list determining which parameters' posterior samples are saved (default: X, W and tau).

Examples

```
#Given pre-specified data collection Y and component number K
opts <- getDefaultOpts(bicluster=FALSE)
opts$normalLatents <- FALSE #Binary sparsity for each sample and data source
 ## Not run: model <- gfa(Y,opts,K)
```

---

gfa *Gibbs sampling for group factor analysis*

Description

gfa returns posterior samples of group factor analysis model.

Usage

```
gfa(Y, opts, K = NULL, projection = NULL, filename = "")
```

Arguments

Y	Either 1. Data sources with co-occuring samples: a list of data matrices, where Y[[m]] is a numeric $N \times D_m$ matrix, or 2. Data sources paired in two modes (some data sources share the samples of the first data source, and some share its features): A list with two elements structured as 1. The data collections Y[[1]] and Y[[2]] should be connected by sharing their first data source, i.e. Y[[1]][[1]] should equal the transpose of Y[[2]][[1]]. NOTE: The data features should have roughly zero mean and unit variance. If this is not the case, preprocessing with function `normalizeData` is recommended.
opts	List of model options; see function `getDefaultOpts`.
K	The number of components (i.e. latent variables). Recommended to be set somewhat higher than the expected component number, so that the sampler can determine the model complexity by shutting down excessive components. High values result in high CPU time. Default: half of the minimum of the sample size and total data dimensionality.
projection	Fixed projections. Only intended for sequential prediction use via function `sequentialGfaPrediction`. Default: NULL.
filename	A string. If provided, will save the sampling chain to this file every 100 iterations. Default "", inducing no saving.

Details

GFA allows factor analysis of multiple data sources (i.e. data sets). The priors of the model can be set to infer bicluster structure from the data sources; see `getDefaultOpts`. Missing values (NAs) are inherently supported. They will not affect the model parameters, but can be predicted with function `reconstruction`, based on the observed values of the corresponding sample and feature. The association of a data source to each component is inferred based on the data. Letting only a subset of the components to explain a data source results in the posterior identifying relationships between any subset of the data sources. In the extreme cases, a component can explain relationships within a single data source only ("structured noise"), or across all the data sources.

Value

A list containing the model parameters - in case of pairing in two modes, each element is a list of length 2; one element for each mode. For most parameters, the final posterior sample is provided to aid in initial checks; all the posterior samples should be used for model analysis. The list elements are:

W	The loading matrix (final posterior sample); $D \times K$ matrix.
X	The latent variables (final sample); $N \times K$ matrix.

Z	The spike-and-slab parameters (final sample); $D \times K$ matrix.
r	The probability of slab in Z (final sample).
rz	The probability of slab in the spike-and-slab prior of X (final sample).
tau	The noise precisions (final sample); D-element vector.
alpha	The precisions of the projection weights W (final sample); $D \times K$ matrix.
beta	The precisions of the latent variables X (final sample); $N \times K$ matrix.
groups	A list denoting which features belong to each data source.
D	Data dimensionalities; M-element vector.
K	The number of components inferred. May be less than the initial K.

and the following elements:

posterior	the posterior samples of, by default, X, W and tau.
cost	The likelihood of all the posterior samples.
aic	The Akaike information criterion of all the posterior samples.
opts	The options used for the GFA model.
conv	An estimate of the convergence of the model's reconstruction based on Geweke diagnostic. Values significantly above 0.05 imply a non-converged model, and hence the need for a longer sampling chain.
time	The CPU time (in seconds) used to sample the model.

`informativeNoisePrior` *Informative noise residual prior*

Description

`informativeNoisePrior` returns an informative noise prior for GFA, for a given data collection. The function sets the noise residual parameters such that the expected proportion of variance explained is for all variables and groups (in contrast to being proportional to their original scale). Recommended e.g. when the data is 'small n, large p', and the standard prior from `getDefaultOpts` seems to overfit the model by not shutting off any component with high initial K.

Usage

```
informativeNoisePrior(Y, opts, noiseProportion = 0.5, conf = 1)
```

Arguments

Y	The data. For details, see function `gfa`.
opts	Model options. See function `getDefaultOpts` for details. If option tauGrouped is TRUE (default), each data source is given equal importance (feature importance may vary within each source). If it is FALSE, each feature is given equal importance.

noiseProportion
: proportion of total variance to be explained by noise. Suggested to lie between 0.01 and 0.99.

conf
: Confidence in the prior, relative to confidence in the data. Suggested to lie between 0.01 and 100.

Value

The input model options (opts) with an updated residual noise prior, corresponding to the elements prior.alpha_0t and prior.beta_0t.

Examples

```
#Given data collection Y
opts <- getDefaultOpts()
## Not run: opts <- informativeNoisePrior(Y,opts,0.2,1)
## Not run: res <- gfa(Y,opts=opts)
```

---

normalizeData *Normalize data to be used by GFA*

Description

normalizeData is used to transform a data collection into a normalized form suitable for GFA. This function does two things: 1. It centers each variable. GFA assumes zero-mean data, as it models variances. 2. It normalizes the scales of variables and/or variable groups. Features with higher variance will affect the model structure more; if this is not desired, the normalization should be done. In GFA it is additionally possible to normalize the importance of variable groups (data sources), in addition or instead of individual variables. Finally, the total variance of data is normalized for numerical reasons. This is particularly important if no other normalization is done. NOTE: the function assumes continuous-valued data. If some features are e.g. binary with only a small portion of 1s, we do not recommend centering them.

Usage

```
normalizeData(train, test = NULL, type = "scaleOverAll")
```

Arguments

train
: a training data set. For a detailed description, see parameter Y in gfa.

test
: a test dataset. Should be provided if sequential prediction is used later.

type
: Specifies the type of normalization to do. Mean-centering of the features is performed in all the cases, and option "center" does not perform any scaling. Option "scaleOverall" (default) uses a single parameter to scale the variance of the whole data collection to 1, while "scaleSources" scales each data source to have variance 1. Finally, "scaleFeatures" performs z-normalization, i.e. assigns the variance of each feature to 1.

Value

A list containing the following elements:

`train`	Normalized training data.
`test`	Normalized test data for sequential prediction (if provided as input).
`trainMean`	Feature-wise means of the training data sources.
`trainSd`	Feature-wise/overall standard deviations of the training data sources.

`reconstruction` *Full data reconstruction based on posterior samples*

Description

`reconstruction` returns the full data reconstruction based on given posterior samples.

Usage

```
reconstruction(res, average = TRUE)
```

Arguments

`res`	The sampled model from function `gfa`
`average`	If TRUE (default), averages the reconstruction over the posterior predictive samples. If set to FALSE, the output may require a large amount of memory. In case of large input data, we recommend acquiring the posterior predictive samples for subsets of data at a time, based on this implementation.

Value

The data reconstruction, a numeric $N \times \sum_{m=1}^{M} D_m$ matrix, if average is TRUE (default). Otherwise, the reconstruction is a $N \times \sum_{m=1}^{M} D_m \times Npost$ array, with posterior samples in the third dimension. If the input data has been paired in two modes, the output will be a list of length 2, one element corresponding to each mode.

`robustComponents` *Robust GFA components*

Description

`robustComponents` analyzes a collection of sampling chains and returns robust components.

Usage

```
robustComponents(models, corThr = 0.9, matchThr = 0.5)
```

Arguments

`models`	Either a vector containing the file names, where the models are saved as 'res', or a list containing the models.
`corThr`	How close two components are required to be, in terms of correlation, in order to match them.
`matchThr`	How big proportion of the chains need to contain the component to include it in the robust components.

Details

The function returns the effects (i.e. reconstructions) of robust components to the data level. It is useful for a thorough model interpretation, accumulating power over several sampling chains by comparing them in the observation space (as opposed to the latent space). The function is needed for this task, as the extreme multi-modality of factor analysis prohibits efficient sampling techniques that would result in a posterior estimate converging to the true posterior in practice. The function uses a heuristic correlation-based procedure to analyze which components occur frequently in GFA sampling chains.

Value

A list with the following elements (when input data are paired in two modes, the returned list is of length 2, containing the following elements for each mode):

`Krobust`	The number of robust components found with the given thresholds.
`effect`	The component effect in the data space; and array of size $N \times \sum_{m=1}^{M} D_m \times Krobust$.
`indices`	The corresponding component indices; a $length(models) \times Krobust$ matrix. Negative indices denote the closest component in the corresponding repetition, having no components above the threshold.
`cor`	The correlations of the components matched to this robust component; a matrix of size $length(models) \times Krobust$. The correlations are reported relative to the first repetition with this component observed.

Examples

```
X <- matrix(rnorm(10*2),10,2)
W <- matrix(rnorm(15*2),15,2)
Y <- tcrossprod(X,W) + matrix(rnorm(10*15),10,15)
opts <- getDefaultOpts() #Default options
#Fast runs for the demo, default options recommended in general
opts[c("iter.burnin", "iter.max")] <- c(500, 1000)
res <- list()
for(i in 1:4) res[[i]] <- gfa(list(Y[,1:6],Y[,7:15]),opts=opts,K=3)
rob <- robustComponents(res)
```

`sequentialGfaPrediction` *Sequential prediction of new samples from observed data views to unobserved*

Description

`sequentialGfaPrediction` returns predictions for unobserved data sources of a new set of data samples.

Usage

```
sequentialGfaPrediction(Y, model)
```

Arguments

`Y`	The new data samples, in a similar format as in function `gfa`
`model`	The sampled model from function `gfa`, with model$opts$predict being a logical vector with the length matching the length of Y, describing which data views will be predicted (TRUE), and which have been observed (FALSE).

Value

A list containing the predictions, with the observed data sources empty. Additionally, sampling MSE is given as element 'mse', and likelihood as 'cost'.

undoNormalizeData *A function for returning predictions into the original data space*

Description

undoNormalizeData returns the predictions on normalized data acquired from normalizeData into the original data space.

Usage

```
undoNormalizeData(pred, normalization)
```

Arguments

pred	The predictions acquired from reconstruction.
normalization	The output list obtained from normalizeData.

Value

The predictions in the original data space.

visualizeComponents *Visualize GFA components*

Description

visualizeComponents illustrates the factorization inferred by GFA, averaging over the posteriors of the parameters, if they have been stored.

Usage

```
visualizeComponents(model, Y = NULL, norm = NULL, mode = 1,
  showAll = TRUE, hclust = FALSE, topK = 3, topFeatures = NA,
  topSamples = NA)
```

Arguments

model	The learned GFA model.
Y	The used input data to be plotted, if supplied. Default NULL.
norm	The normalization acquired from normalizeData, if applied. If provided, the reconstruction is shown in the original data space. Default NULL.
mode	Determines which mode to visualize in case of pairing in two modes (default: 1).

showAll	Show the full predictions and factorizations? May be cumbersome for large data. Default TRUE.
hclust	Order features and samples based on hierarchical clustering? Default FALSE.
topK	Number of strongest components visualized in the data space. Default 3.
topFeatures	How many most relevant features to show for the data space visualizations? Default NA, showing all the features.
topSamples	How many most relevant samples to show for the data space visualizations? Default NA, showing all the samples.

Value

A list containing the matrices that have been visualized.

Index