Package ‘CCAGFA’

December 17, 2015

Type Package

Title Bayesian Canonical Correlation Analysis and Group Factor Analysis

Version 1.0.8

Date 2015-10-20

Author Seppo Virtanen [aut, cre],
Eemeli Leppaaho [aut],
Arto Klami [aut]

Maintainer Seppo Virtanen <s.virtanen@warwick.ac.uk>

Description Variational Bayesian algorithms for learning canonical correlation analysis (CCA), interbattery factor analysis (IBFA), and group factor analysis (GFA). Inference with several random initializations can be run with the functions CCAexperiment() and GFAexperiment().

License GPL (>= 2)

URL http://research.ics.aalto.fi/mi/

NeedsCompilation no

Repository CRAN

Date/Publication 2015-12-17 16:08:55

R topics documented:

CCAGFA-package .................................................. 2
CCAcorr .......................................................... 3
getDefaultOpts .................................................... 4
GFA ............................................................... 5
GFApred .......................................................... 8
GFAsample ......................................................... 9
GFAtrim .......................................................... 10

Index 12
CCAGFA-package

**Description**

Variational Bayesian solution for canonical correlation analysis, inter-battery factor analysis and group factor analysis. The package contains code for learning the model and some supporting functionality for interpretation.

The Bayesian CCA model as implemented here was originally presented by Virtanen et al. (2011), but a more comprehensive treatment is found in Klami et al. (2013). The latter also explains the BIBFA model. The GFA extends CCA to multiple data sources (or groups of variables), providing interpretable linear factorizations that describe variation shared by all possible subsets of sources. It was originally presented by Virtanen et al. (2012). Later Klami et al. (2014) provide a more extensive literature review and present a novel hierarchical low-rank ARD prior for the factor loadings to better account for inter-source relationships.

We recommend that scientific publications using the code for CCA or BIBFA cite Klami et al. (2013), and publications using the code for GFA cite Virtanen et al. (2012), until Klami et al. (2014) has been published.

The package is based on the research done in the SMLB group, Helsinki Institute for Information Technology HIIT, Department of Information and Computer Science, Aalto University, http://research.ics.aalto.fi/mi/.

**Details**

- **Package**: CCAGFA
- **Type**: Package
- **Version**: 1.0.4
- **Date**: 2013-04-23
- **License**: GPL (>= 2)

**Author(s)**

Seppo Virtanen, Eemeli Leppaaho and Arto Klami. Maintainer: Seppo Virtanen <seppo.j.virtanen@aalto.fi>

**References**


CCAcorr


Examples

```r
# Load the package
# require(CCAGFA)

# demo(CCAGFAexample)
```

CCAcorrCompute correlation between the views

Description

A function for estimating the canonical correlations between two data sets. This function can only be used for models learned based on two data sources, since canonical correlation is only defined for two sets.

Usage

CCAcorr(Y, model, threshold = 0.001)

Arguments

Y The data given as a list of two $N \times D[m]$ matrices

model A list of model parameters as returned by `CCA`.

threshold Relative amount of variance explained that is needed for a component to be treated active (see `CCATrim`).

Details

The function computes the correlations for each component. The inactive ones are not suppressed away, but the variable `active` can be used for filtering them out; the correlations for the non-shared components should typically not be trusted. The estimated correlation corresponds to the correlation between the expected values of $Z|Y[1]$ and $Z|Y[2]$.

Value

r The correlations, a vector of length $K$.

active A binary indicator telling which of the components are shared.

Author(s)

Seppo Virtanen and Arto Klami
Examples

```r
# Assume we have a variable model which has been learned with
# CCAexperiment() or CCA().
# output <- CCACorr(model)
# print(output$rc)  # Print the correlations
# print(output$r[which(output$active==1)])  # Only the shared components
```

getDefaultOpts

*Get default options for BIBFA*

Description

A helper function that creates a list of options to be passed for CCA and GFA.

Usage

```r
getDefaultOpts()
```

Details

To run the code with other option values, first run this function and then directly modify the entries before passing the list to CCA and GFA.

Value

- **R**
  The rank of hierarchical low-rank ARD prior. Possible values are all integers, including zero, and "full". When R equals "full" or R equals or is larger than the minimum value of the number of data sets and the number of latent factors, that is min(M,K), the prior corresponds to ARD prior with no low-rank structure.

- **lambda**
  The regularization parameter of the low-rank ARD model.

- **rotate**
  Whether to optimize for a linear transformation to make the variational updates less correlated.

- **init.tau**
  Initial values for the noise precision.

- **iter.crit**
  The iteration is terminated when the relative change in the lower bound for the marginal likelihood drops below this threshold.

- **iter.max**
  Maximum number of iterations.

- **opt.method**
  Which method to use for optimizing the rotation; "BFGS" or "L-BFGS".

- **lbfgs.factr**
  Optimization parameter of L-BFGS.

- **bfgs.crit**
  Optimization parameter of BFGS.

- **opt.iter**
  Number of iterations for the (L-)BFGS optimization.
addednoise  A small constant used to de-correlate latent variables of inactive components.
prior.alpha_0  Gamma prior for ARD.
prior.beta_0  Gamma prior for ARD.
prior.alpha_0t  Gamma prior for tau.
prior.beta_0t  Gamma prior for tau.
dropK  Whether to prune out empty factors from the model during inference.
low.mem  Whether to store and return the covariance matrices of W.
verbose  The amount of details printed while running CCA and GFA. 0=none, 1=medium, 2=high.

Author(s)
Seppo Virtanen, Eemeli Leppaaho and Arto Klami

See Also
CCA, GFA.

Examples
# opts <- getDefaultOpts()  # Get the default options
# opts$verbose <- 1  # Change some of them
# opts$init.tau <- 10^5

# Run the model with the new options
# model <- CCAexperiment(Y,K,opts)

GFA  Estimate a Bayesian IBFA/CCA/GFA model

Description
Estimates the parameters of a Bayesian group factor analysis (GFA), canonical correlation analysis (BCCA), or inter-battery factor analysis (BIBFA).
GFA is a latent variable model for explaining relationships between multiple data matrices with co-occurring samples. The model finds linear factors that explain dependencies between these matrices, similarly to how factor analysis explains dependencies between individual variables.
BIBFA is a special case of GFA for two data matrices. It finds factors explaining the relationship between them, as well as factors explaining the residual variation in each matrix. The solution of BIBFA equals that of CCA, with additional factors for explaining the data-specific noise.

Usage
CCA(Y, K, opts)
GFA(Y, K, opts)
CCAexperiment(Y, K, opts, Nrep=10)
GFAexperiment(Y, K, opts, Nrep=10)
Arguments

\(Y\) A list containing matrices with \(N\) rows (samples) and \(D[m]\) columns (features). Must have exactly two matrices for CCA and any number of co-occurring matrices for GFA.

\(K\) The number of components.

\(\text{opts}\) A list of parameters and options to be used when learning the model. See `getDefaultOpts`.

\(\text{Nrep}\) The number of random initializations used for learning the model; only used for CCAexperiment and GFAexperiment.

Details

The recommended strategy is to use `GFAexperiment` for learning a Bayesian group factor analysis model. It simply calls `GFA Nrep` times and returns the model with the best variational lower bound for the marginal likelihood.

CCAexperiment and CCA are simple wrappers for the corresponding GFA functions, to be used for the case of \(M=2\) data sets. CCA is a special case of GFA with exactly two co-occurring matrices, and these functions are provided for convenience only.

Value

The methods return a list that contains all the model parameters and other details.

\(Z\) The mean of the latent variables; \(N\) times \(K\) matrix

\(\text{covZ}\) The covariance of the latent variables; \(K\) times \(K\) matrix

\(ZZ\) The second moments \(Z^TZ\); \(K\) times \(K\) matrix

\(W\) List of the mean projections; \(D_i\) times \(K\) matrices

\(\text{covW}\) List of the covariances of the projections; \(K\) times \(K\) matrices

\(WW\) List of the second moments \(W^TW\); \(K\) times \(K\) matrices

\(\text{tau}\) The mean precisions (inverse variance, so \(1/\text{tau}\) gives the variances denoted by \(\sigma\) in the paper); \(M\)-element vector

\(\alpha\) The mean precisions of the projection weights, used in the ARD prior; \(M\) times \(K\) matrix

\(\text{cost}\) Vector collecting the variational lower bounds for each iteration

\(D\) Data dimensionalities; \(M\)-element vector

\(K\) The number of latent factors

\(\text{datavar}\) The total variance in the data sets, needed for `GFAtrim`

\(R\) The rank of \(\alpha\)

\(U\) The group factor loadings; \(M\) times \(R\) matrix

\(V\) The latent group factors; \(K\) times \(R\) matrix

\(u.\mu\) The mean of group factor loadings \(U\); \(M\)-element vector

\(v.\mu\) The mean of latent group factors \(V\); \(K\)-element vector
Author(s)
Seppo Virtanen, Eemeli Leppaaho and Arto Klami

References

See Also
getDefaultOpts

Examples

# Create simple random data
#
N <- 50; D <- c(4,6) # 50 samples with 4 and 6 dimensions
tau <- c(3,3) # residual noise precision
K <- 3 # K real components (1 shared, 1+1 private)
Z <- matrix(rnorm(N*K,0,1),N,K) # drawn from the prior
alpha <- matrix(c(1,1,1e6,1,1e6,1),2,3)
Y <- vector("list",length=2)
W <- vector("list",length=2)
for(view in 1:2) {
  W[[view]] <- matrix(0,D[view],K)
  for(k in 1:K) {
    W[[view]][,k] <- rnorm(D[view],0,1/sqrt(alpha[view,k]))
  }
  Y[[view]] <- Z %*% t(W[[view]]) +
    matrix(rnorm(N*D[view],0,1/sqrt(tau[view])),N,D[view])
}

# Run the model
#
opts <- getDefaultOpts()
opts$iter.max <- 10 # Terminate early for fast testing
GFApred

Predict samples of one view given the other(s)

Description

Function for making predictions from some subset of views to the remaining ones. This can be used, for example, for multi-output regression and classification tasks.

Usage

```r
CCApred(pred, Y, model, sample = FALSE, nSample = 100)
GFApred(pred, Y, model, sample = FALSE, nSample = 100)
```

Arguments

- `pred`: A vector of binary indicators telling which of the views are observed (1), and which are to be predicted (0).
- `Y`: The input data as a list of M elements, N times D[m] matrices.
- `model`: A list of model parameters as returned by `GFA`.
- `sample`: Boolean indicator telling whether to also draw samples from the predictive distribution.
- `nSample`: How many samples to draw if `sample=TRUE`.

Details

Estimates the conditional distribution of Z given the observed view and then estimates the expected predictions for the unobserved view. It is also possible to draw samples from the full predictive distribution, which cannot be specified in analytic form.

Value

- `Y`: The mean predictions. Also the observed input data is returned, so that Y is in the same format as the input data for GFA.
- `Z`: The mean of the latent variables given the observed data.
- `covZ`: The covariance of the latent variables given the observed data.
- `sam`: List that contain `nSample` elements. Each is a list that contains the projection matrices (W), the latent variables (Z), and the N samples drawn from the predictive posterior.

Author(s)

Seppo Virtanen and Arto Klami
**GFAsample**

Generate data from CCA/BIBFA/GFA model

---

**Description**

Generate data from a CCA/BIBFA/GFA model that has been learned with `GFA`. The most likely use of this function is for model checking.

**Usage**

CCAsample(model, N)

GFAsample(model, N)

**Arguments**

- **model**: A list of model parameters as returned by `GFA`.
- **N**: How many samples to draw.

**Details**

The code randomly samples $Z$ from the prior and then draws $N$ observations for both views.

**Value**

- **Y**: The data, a list of $N$ times $D[m]$ matrices.
- **Z**: The latent variables, a $N$ times $K$ matrix.
GFAtrim

Author(s)
Seppo Virtanen and Arto Klami

See Also
GFA,CCA

Examples

```#
# Assume we have a variable model which has been learned with
# GFAexperimnet() or GFA().
# Then the following line would draw 100 samples from it:
#
# Y2 <- GFAAsample(model,100)
#
```

Description
Prunes out unnecessary components and determines for each of the remaining components whether
it is shared or not. In other words, the function reveals the component allocation into shared and
view-specific ones.

Usage

```R
ccatrim(model, threshold = 0.001)
gfatrim(model, threshold = 0.001)
```

Arguments

- `model`: A list of model parameters as returned by `CCA` or `GFA`.
- `threshold`: The proportion of relative variance explained that components need to exceed to
  be detected as active.

Details
This function can be used to prune out unnecessary components and to recognize which of the com-
ponents are shared. This can be useful for interpretative purposes, but it is typically not necessary
to apply this function prior to making predictions (with `GFApred` or otherwise). The inactive com-
ponents will anyway automatically cancel out for the predictive formulas. The code works well for
low-dimensional data, but for complex high-dimensional data sources one should check whether the
trimming is reasonable; in such cases it is difficult to make clear decisions on component activity.
Value

A list of parameter values as returned by GFA. The list also includes two extra elements:

- **trimmed**: A boolean variable indicating that the model has been trimmed with this function.
- **active**: A binary matrix indicating for each component (column) in which views (row) it is active.

Author(s)

Seppo Virtanen and Arto Klami

See Also

GFA, CCA

Examples

```r
# Assume we have a variable model which has been learned with
# GFAexperiment() or GFA().
# Then the following line would trim it:
#
# trimmed <- GFATrim(model)
#```
Index

*Topic package
  CCA (GFA), 5
  CCAcorr, 3
  CCAexperiment (GFA), 5
  CCAGFA-package, 2
  CCApred (GFApred), 8
  CCAsample (GFAsample), 9
  CCAtrim, 3
  CCAtrim (GFAtrim), 10
  CCGFA (CCAGFA-package), 2
  getDefaultOpts, 4, 6, 7
  GFA, 5, 5, 8–11
  GFAexperiment (GFA), 5
  GFApred, 8, 10
  GFAsample, 9
  GFAtrim, 6, 10