Package ‘fabMix’

January 27, 2017

Type Package

Title Overfitting Bayesian Mixtures of Factor Analyzers with an Unknown Number of Components

Version 1.0

Date 2017-01-25

Author Panagiotis Papastamoulis

Maintainer Panagiotis Papastamoulis <papapast@yahoo.gr>

Description
Model-based clustering of multivariate continuous data with possibly complex covariance structure. The underlying model is a Bayesian mixture of factor analyzers with a large number of components (overfitting mixture). Suitable prior assumptions ensure that asymptotically the extra components will have zero posterior weight (empty), therefore, the inference on the number of clusters is based on the ``alive'' components. The number of factors is considered fixed, and the optimal one can be estimated using information criteria. Identifiability issues related to label switching are dealt by post-processing the simulated output with the ECR algorithm.

Imports MASS, doParallel, foreach, label.switching, mvtnorm

License GPL-2

NeedsCompilation no

R topics documented:

fabMix-package 2
complete.log.likelihood 3
compute_A_B_G_D_and_simulate_mu_Lambda 4
compute_sufficient_statistics 5
dealWithLabelSwitching_same_sigma 5
fabMix 6
getStuffForDIC 8
log_dirichlet_pdf 9
myDirichlet 10
observed.log.likelihood0 10
overfittingMFA 11
simData 12

update_all_y 13
update_OmegaINV 13
update_SigmaINV_faster 14
update_z4 15
update_z_b 15
waveDataset1500 16

Index 17

`fabMix-package` *Overfitting Bayesian Mixtures of Factor Analyzers with an Unknown Number of Components*

Description

Model-based clustering of multivariate continuous data with possibly complex covariance structure. The underlying model is a Bayesian mixture of factor analyzers with a large number of components (overfitting mixture). Suitable prior assumptions ensure that asymptotically the extra components will have zero posterior weight (empty), therefore, the inference on the number of clusters is based on the “alive” components. The number of factors is considered fixed, and the optimal one can be estimated using information criteria. Identifiability issues related to label switching are dealt by post-processing the simulated output with the ECR algorithm.

Author(s)

Panagiotis Papastamoulis

Maintainer: Panagiotis Papastamoulis <papapast@yahoo.gr>

References

Fokoue, E. and Titterington, D.M. (2003). Mixtures of Factor Analysers: Bayesian Estimation and Inference by Stochastic Simulation. Machine Learing, 50(1): 73-94.

Papastamoulis P. and Iliopoulos G. (2010). An artificial allocations based solution to the label switching problem in Bayesian analysis of mixtures of distributions. Journal of Computational and Graphical Statistics, 19: 313-331.

van Havre, Z., White, N., Rousseau, J. and Mengersen, K. (2015). Overfitting Bayesian Mixture Models with an Unknown Number of Components. PLOS ONE, 10(7): 1-27.

Papastamoulis, P. (2016). `label.switching`: An R Package for Dealing with the Label Switching Problem in MCMC Outputs. Journal of Statistical Software, 69(1), 1-24.

Papastamoulis, P. (2017). Overfitting Bayesian mixtures of factor analyzers with an unknown number of components. arXiv:1701.04605 [stat.ME]

See Also

`fabMix`, `dealWithLabelSwitching_same_sigma`, `getStuffForDIC`

Examples

```
# simulate a synthetic dataset along the lines of the paper:
n = 1000             # sample size
p = 40               # number of variables
q = 4                # number of factors
K = 10               # number of clusters
sINV_diag = 1/((1:p)) # diagonal of inverse variance of errors
set.seed(10)
syntheticDataset <- simData(K.true = K, n = n, q = q, p = p, sINV_values = sINV_diag )

## Not run:

# define parameters
Kmax <- 20    # number of overfitted mixture components
nChains <- 8  # number of parallel chains
dN <- 1
# Dirichlet prior of mixture weights per chain.
#   The target chain corresponds to the first entry.
dirPriorAlphas <- c(1, 1 + dN * (2:nChains - 1))/Kmax
outputFolder <- "fabMixExample"
# Run algorithm
fabMix( dirPriorAlphas = dirPriorAlphas,
        rawData = syntheticDataset$data,
        outDir = outputFolder, Kmax = Kmax, mCycles = 1200,
        burnCycles = 200, q = q)

# Compute information criteria:
getStuffForDIC(x_data = syntheticDataset$data, outputFolder = outputFolder, q = q)

# Deal with label switching:
dealWithLabelSwitching_same_sigma(x_data = syntheticDataset$data,
        outputFolder = outputFolder, q = q,
        compute_regularized_expression = TRUE, Km = Kmax)

## End(Not run)
```

---

complete.log.likelihood
Complete log-likelihood function

Description

Complete log-likelihood function

Usage

```
complete.log.likelihood(x_data, w, mu, Lambda, SigmaINV, z)
```

Arguments

`x_data`	Data
`w`	Mixture weights
`mu`	Marginal means
`Lambda`	Factor loadings
`SigmaINV`	Precision matrix (inverse covariance)
`z`	Allocation vector of the data to the mixture components

Value

complete log-likelihood value

Author(s)

Panagiotis Papastamoulis

`compute_A_B_G_D_and_simulate_mu_Lambda` *Computation and simulations*

Description

This function simulates μ and Λ.

Usage

```
compute_A_B_G_D_and_simulate_mu_Lambda(SigmaINV,
suff_statistics, OmegaINV, K, priorConst1, T_INV, v_r)
```

Arguments

`SigmaINV`	Precision matrix Σ^{-1}
`suff_statistics`	Sufficient statistics
`OmegaINV`	Prior parameter: Ω^{-1}
`K`	Number of overfitting mixture components
`priorConst1`	Prior constant: $T^{-1}\xi$
`T_INV`	Prior parameter: $T^{-1}\xi$
`v_r`	This vector is used to set to zero the upper right $(q-1)\times(q-1)$ diagonal block of factor loadings for identifiability purposes.

Value

A list containing A, B, Γ, Δ and a draw from the conditional distributions of μ and Λ.

Author(s)

Panagiotis Papastamoulis

`compute_sufficient_statistics` *Compute sufficient statistics*

Description

Compute sufficient statistics given y and z.

Usage

```
compute_sufficient_statistics(y, z, K, x_data)
```

Arguments

`y`	Matrix of factors
`z`	Allocation vector
`K`	Number of components
`x_data`	Data

Value

A list with six entries of sufficient statistics.

Author(s)

Panagiotis Papastamoulis

`dealWithLabelSwitching_same_sigma` *Apply label switching algorithms for the Σ model*

Description

This functions is a wrapper for the `label.switching` package and applies the `ECR` and `ECR.ITERATIVE.1` algorithms. The model should have the same variance of error terms per cluster.

Usage

```
dealWithLabelSwitching_same_sigma(x_data, outputFolder, q, burn,
z.true, compute_regularized_expression, Km)
```

Arguments

`x_data`	Data
`outputFolder`	Name of the folder where the `fabMix` function has saved its output
`q`	Number of factors
`burn`	Discard observations as burn-in period (optional).
`z.true`	An (optional) vector of cluster assignments which is considered as the groun-truth clustering of the data. Useful for direct comparisons against the real parameter values in simulated data.
`compute_regularized_expression`	Logical. Should regularized expression be computed?
`Km`	Number of components in the overfitted mixture model.

Value

The following files are produced in the output folder:

Author(s)

Panagiotis Papastamoulis

References

Papastamoulis, P. (2016). `label.switching`: An R Package for Dealing with the Label Switching Problem in MCMC Outputs. Journal of Statistical Software, 69(1), 1-24.

`fabMix` *Main function of the package*

Description

This function runs parallel chains under a prior tempering scheme of the Dirichlet prior distribution of mixture weights.

Usage

```
fabMix(dirPriorAlphas, rawData, outDir, Kmax, mCycles, burnCycles,
g, h, alpha_sigma, beta_sigma, q, normalize, thinning, zStart, nIterPerCycle)
```

Arguments

`dirPriorAlphas`	The prior Dirichlet parameters for each chain.
`rawData`	The observed data as an $n \times p$ matrix. Clustering is performed on the rows of the matrix.
`outDir`	Name of the output folder.
`Kmax`	Number of components in the overfitted mixture. Default: 20.

`mCycles`	Number of MCMC cycles. Each cycle consists of `nIterPerCycle` MCMC iterations. At the end of each cycle a swap of the state of two randomly chosen adjacent chains is attempted.
`burnCycles`	Number of cycles that will be discarded as burn-in period.
`g`	Prior parameter g. Default value: $g = 2$.
`h`	Prior parameter h. Default value: $h = 1$.
`alpha_sigma`	Prior parameter α. Default value: $\alpha = 2$.
`beta_sigma`	Prior parameter β. Default value: $\beta = 1$.
`q`	Number of factors q, where $1 \leq q \leq L$. An error is thrown if the Ledermann bound (L) is exceeded.
`normalize`	Should the observed data be normalized? Default value: TRUE.
`thinning`	Optional integer denoting the thinning of the keeped MCMC cycles.
`zStart`	Optional starting value for the allocation vector.
`nIterPerCycle`	Number of iteration per MCMC cycle. Default value: 10.

Value

List of files written to `outDir`

Note

It is recommended to always use: `normalize = TRUE` (default). Tuning of `dirPriorAlphas` may be necessary to achieve reasonable acceptance rates of chain swaps. Also note that the output is not identifiable due to label switching and the user has to subsequently call the `dealWithLabelSwitching_same_sigma` function.

Author(s)

Panagiotis Papastamoulis

References

Papastamoulis, P. (2017). Overfitting Bayesian mixtures of factor analyzers with an unknown number of components. arXiv:1701.04605 [stat.ME]

See Also

`dealWithLabelSwitching_same_sigma`

Examples

```
## Not run:
# simulate a synthetic dataset along the lines of the paper:
n = 1000              # sample size
p = 40                # number of variables
q = 4                 # number of factors
K = 10                # number of clusters
```

```
sINV_diag = 1/((1:p)) # diagonal of inverse variance of errors
set.seed(10)
syntheticDataset <- simData(K.true = K, n = n, q = q, p = p, sINV_values = sINV_diag )

# define parameters
Kmax <- 20    # number of overfitted mixture components
nChains <- 8  # number of parallel chains
dN <- 1
# Dirichlet prior of mixture weights per chain.
#   The target chain corresponds to the first entry.
dirPriorAlphas <- c(1, 1 + dN * (2:nChains - 1))/Kmax
outputFolder <- "fabMixExample"
# Run algorithm
fabMix( dirPriorAlphas = dirPriorAlphas,
        rawData = syntheticDataset$data,
        outDir = outputFolder, Kmax = Kmax, mCycles = 1200,
        burnCycles = 200, q = q)

# Compute information criteria:
getStuffForDIC(x_data = syntheticDataset$data, outputFolder = outputFolder, q = q)

# Deal with label switching:
dealWithLabelSwitching_same_sigma(x_data = syntheticDataset$data,
        outputFolder = outputFolder, q = q,
        compute_regularized_expression = TRUE, Km = Kmax)

## End(Not run)
```

getStuffForDIC *Compute information criteria*

Description

This function computes four information criteria for a given run of the `fabMix` algorithm, namely: AIC, BIC, DIC and DIC_2. Given various runs with different number of factors, the selected model corresponds to the one with the smalled value of the selected criterion.

Usage

```
getStuffForDIC(x_data, outputFolder, q, burn, Km, normalize, discardLower)
```

Arguments

`x_data`	Observed data.
`outputFolder`	Name of the folder where the `fabMix` function has saved its output.
`q`	Number of factors. Note that this should coincide with the number of factors in the `fabMix` run.

`burn`	Discard observations as burn-in period (optional).
`Km`	Number of components in the overfitted mixture model. Note that this should coincide with the same entry in the `fabMix` run.
`normalize`	Should the observed data be normalized? Note that this should coincide with the same entry in the `fabMix` run. Default value: TRUE.
`discardLower`	Discard draws with log-likelihood values lower than the specific quantile. This applied only for the DIC computation.

Details

If necessary, more details than the description above

Value

The information criteria are saved to the `informationCriteria_map_model.txt` file in the codeoutputFolder.

Note

It is well known that DIC tends to overfit, so it advised to compare models with different number of factors using AIC or BIC.

Author(s)

Panagiotis Papastamoulis

`log_dirichlet_pdf` *Log-density function of the Dirichlet distribution*

Description

Log-density function of the Dirichlet distribution

Usage

```
log_dirichlet_pdf(alpha, weights)
```

Arguments

`alpha`	Parameter vector
`weights`	Vector of weights

Value

Log-density of the $D(alpha_1, \ldots, \alpha_k)$ evaluated at $w_1, \ldots, w_k$.

Author(s)

Panagiotis Papastamoulis

`myDirichlet` *Simulate from the Dirichlet distribution*

Description

Generate a random draw from the Dirichlet distribution $D(\alpha_1, \ldots, \alpha_k)$.

Usage

```
myDirichlet(alpha)
```

Arguments

`alpha`	Parameter vector

Value

Simulated vector

Author(s)

Panagiotis Papastamoulis

`observed.log.likelihood0` *Log-likelihood of the mixture model*

Description

Log-likelihood of the mixture model evaluated only at the alive components.

Usage

```
observed.log.likelihood0(x_data, w, mu, Lambda, Sigma, z)
```

Arguments

`x_data`	The observed data
`w`	Vector of mixture weights
`mu`	Vector of marginal means
`Lambda`	Factor loadings
`Sigma`	Common covariance matrix of the errors per cluster
`z`	Allocation vector

Value

Log-likelihood value

Author(s)

Panagiotis Papastamoulis

`overfittingMFA` *Basic MCMC sampler*

Description

Gibbs sampling for fitting a mixture model of factor analyzers.

Usage

```
overfittingMFA(x_data, originalX, outputDirectory, Kmax, m, thinning, burn,
g, h, alpha_prior, alpha_sigma, beta_sigma, start_values, q, zStart, gibbs_z)
```

Arguments

`x_data`	normalized data
`originalX`	observed raw data (only for plotting purpose)
`outputDirectory`	Name of the output folder
`Kmax`	Number of mixture components
`m`	Number of iterations
`thinning`	Thinning of chain
`burn`	Burn-in period
`g`	Prior parameter g. Default value: $g = 2$.
`h`	Prior parameter h. Default value: $h = 1$.
`alpha_prior`	Parameters of the Dirichlet prior distribution of mixture weights.
`alpha_sigma`	Prior parameter α. Default value: $\alpha = 2$.
`beta_sigma`	Prior parameter β. Default value: $\beta = 1$.
`start_values`	Optional (not used)
`q`	Number of factors.
`zStart`	Optional (not used)
`gibbs_z`	Optional

Value

List of files

Author(s)

Panagiotis Papastamoulis

simData *Synthetic data generator*

Description

Simulate data from a multivariate normal mixture using a mixture of factor analyzers mechanism.

Usage

```
simData(p, q, K.true, n, loading_means, loading_sd, sINV_values)
```

Arguments

`p`	The dimension of the multivariate normal distribution ($p > 1$).
`q`	Number of factors. It should be strictly smaller than p.
`K.true`	The number of mixture components (clusters).
`n`	Sample size.
`loading_means`	A vector which contains the means of blocks of factor loadings. Default: `loading_means = c(-30,-20,-10,10, 20, 30)`.
`loading_sd`	A vector which contains the standard deviations of blocks of factor loadings. Default: `loading_sd <- rep(2, length(loading_means))`.
`sINV_values`	A vector which contains the values of the diagonal of the inverse covariance matrix. Default: `sINV_values = rgamma(p, shape = 1, rate = 1)`.

Value

A list with the following entries:

`data`	$n \times p$ array containing the simulated data.
`class`	n-dimensional vector containing the class of each observation.
`factorLoadings`	$K.true \times p \times q$-array containing the factor loadings Λ_{krj} per cluster k, feature r and factor j, where $k = 1, \ldots, K$; $r = 1, \ldots, p$; $j = 1, \ldots, q$.
`means`	$K.true \times p$ matrix containing the marginal means μ_{kr}, $k = 1, \ldots, K$; $r = 1, \ldots, p$.
`variance`	$p \times p$ diagonal matrix containing the variance of errors σ_{rr}, $r = 1, \ldots, p$. Note that the same variance of errors is assumed for each cluster.
`factors`	$n \times q$ matrix containing the simulated factor values.
`weights`	$K.true$-dimensional vector containing the weight of each cluster.

Note

The marginal variance for cluster k is equal to $\Lambda_k \Lambda_k^T + \Sigma$.

Author(s)

Panagiotis Papastamoulis

`update_all_y` *Gibbs sampling for y*

Description

Gibbs sampling for y

Usage

```
update_all_y(x_data, mu, SigmaINV, Lambda, z)
```

Arguments

`x_data`	Data
`mu`	Marginal means
`SigmaINV`	Precision matrix
`Lambda`	Factor loadings
`z`	Allocation vector

Value

A matrix with generated factors

Author(s)

Panagiotis Papastamoulis

`update_OmegaINV` *Gibbs sampling for $\Omega\hat{}-1$*

Description

Gibbs sampling for Ω^{-1}

Usage

```
update_OmegaINV(Lambda, K, g, h)
```

Arguments

`Lambda`	Factor loadings
`K`	Number of components
`g`	Prior parameter
`h`	Prior parameter

Value

Ω^{-1}

Author(s)

Panagiotis Papastamoulis

`update_SigmaINV_faster` *Gibbs sampling for* $\Sigma\hat{}-1$

Description

Gibbs sampling for Σ^{-1}

Usage

```
update_SigmaINV_faster(x_data, z, y, Lambda, mu, K, alpha_sigma, beta_sigma)
```

Arguments

`x_data`	Data
`z`	Allocation vector
`y`	Factors
`Lambda`	Factor loadings
`mu`	Marginal means
`K`	Number of components
`alpha_sigma`	Prior parameter
`beta_sigma`	Prior parameter

Value

Σ^{-1}

Author(s)

Panagiotis Papastamoulis

update_z4 *Collapsed Gibbs for z*

Description

Collapsed Gibbs for z

Usage

```
update_z4(w, mu, Lambda, SigmaINV, K, x_data)
```

Arguments

w	Mixture weights
mu	Marginal means
Lambda	Factor loadings
SigmaINV	Precision matrix
K	Number of components
x_data	Data

Value

Allocation vector

Author(s)

Panagiotis Papastamoulis

update_z_b *Gibbs sampling for z*

Description

Gibbs sampling for z

Usage

```
update_z_b(w, mu, Lambda, y, SigmaINV, K, x_data)
```

Arguments

`w`	Mixture weights
`mu`	Marginal means
`Lambda`	Factor loadings
`y`	Matrix of factors
`SigmaINV`	Precision matrix
`K`	Number of components
`x_data`	Data

Value

Allocation vector

Author(s)

Panagiotis Papastamoulis

`waveDataset1500` *Wave dataset*

Description

A subset of 1500 randomly sampled observations from the wave dataset (version 1), available from the UCI machine learning repository. It contains 3 classes of waves (variable `class` with values "1", "2" and "3") and 21 attributes. Each class is generated from a combination of 2 of 3 base waves with noise.

Usage

```
waveDataset1500
```

Format

A data frame with 1500 rows and 22 columns. The first column denotes the class of each observation.

Source

`https://archive.ics.uci.edu/ml/datasets/Waveform+Database+Generator+(Version+1)`

References

Lichman, M. (2013). UCI Machine Learning Repository `http://archive.ics.uci.edu/ml`. Irvine, CA: University of California, School of Information and Computer Science.

Breiman,L., Friedman,J.H., Olshen,R.A. and Stone,C.J. (1984). Classification and Regression Trees. Wadsworth International Group: Belmont, California.

Index

∗Topic **datasets**
    waveDataset1500, 16
∗Topic **package**
    fabMix-package, 2

complete.log.likelihood, 3
compute_A_B_G_D_and_simulate_mu_Lambda, 4
compute_sufficient_statistics, 5

dealWithLabelSwitching_same_sigma, *2*, 5, *7*

fabMix, *2*, 6
fabMix-package, 2

getStuffForDIC, *2*, 8

log_dirichlet_pdf, 9

myDirichlet, 10

observed.log.likelihood0, 10
overfittingMFA, 11

simData, 12

update_all_y, 13
update_OmegaINV, 13
update_SigmaINV_faster, 14
update_z4, 15
update_z_b, 15

waveDataset1500, 16