Package 'mcgfa'

June 24, 2019

Version 2.2.1

Type Package

Title Mixtures of Contaminated Gaussian Factor Analyzers

Description Clustering and classification using the Mixtures of Contaminated Gaussian Factor Analyzers model. Allows for automatic detection of outliers and noise. Punzo, A, Blostein, M, McNicholas, PD (2017) <arXiv:1408.2128v2>.

Imports stats, parallel

License GPL (>= 2)

LazyData TRUE

NeedsCompilation yes

Author Martin Blostein [aut, cre],
Antonio Punzo [aut],
Paul D. McNicholas [aut, ths]

Maintainer Martin Blostein <martin.blostein@gmail.com>

Repository CRAN

Date/Publication 2019-06-24 14:30:03 UTC

R topics documented:

`mcgfa-package` *mcgfa: Model-Based Clustering and Classification with Mixtures of Contaminated Gaussian Factor Analyzers*

Description

Performs clustering and classification using the Mixtures of Contaminated Gaussian Factor Analyzers model. Allows for automatic detection of outliers and noise.

Details

Package:	mcgfa
Type:	Package
Version:	1.0
Date:	2015-06-13
License:	GPL (>=2)
LazyLoad:	yes

Author(s)

Martin Blostein, Antonio Punzo, and Paul D. McNicholas

Maintained by: Martin Blostein <martin.blostein@gmail.com>

References

...

See Also

`mcgfa` for main function, and details.

`gaussnoise` *Gaussian Clusters with White Noise*

Description

Simulated toy data set to demonstrate the MCGFA package. Generated using the `mnormt` package for R.

Usage

```
data(gaussnoise)
```

Format

A data frame with 205 observations and 6 columns. The first column gives the class number: 1 and 2 are the two Gaussian clusters. The second column indicates whether an observation arose as noise. (This is eqivalent to the first column being equal to 3.)

`mcgfa` *Model Fitting for Mixtures of Contaminated Gaussian Factor Analyzers*

Description

Performs clustering and classification using the Mixtures of Contaminated Gaussian Factor Analyzers model. This model allows for automatic detection of outliers and noise, and automatically detects outliers. It is appropriate for high-dimensional numerical data.

Usage

```
mcgfa(X, rG=1:3, rq=1:3, models="all", known=NULL, init_method="emEM",
      init_z, max_it=400, tol=1e-3, alpha_min=0.5, eta_max=1000,
      scale=T, parallel=F, cores=NULL, silent=F, ememargs =
      list(numstart = 25, iterations = 5, model = "UUUUU", q = max(rq)))
```

Arguments

`X`	The data matrix. Rows correspond to observations and columns to variables/features; thus X is an N-by-p matrix. X must be numerical.
`rG`	The set of values used for the number of components.
`rq`	The set of values used for the number of latent factors.
`models`	The set of parsimonious models used. Either the string "all", or a character vector specifying a subset of models. See Details.
`known`	If NULL, clustering or "unsupervised learning" is performed. If a vector of length equal to the number of observations, semi-supervised learning is performed. In this case, the `i`-th entry of the argument `class` is either 0, or some number in $1, 2, \ldots, G$, where G is the number of components. A value of 0 indicates that the label for observation `i` is unknown, while a nonzero value gives the known label. Note that if all values are nonzero, the model parameters are simply fit and fully supervised classification of new observations may be performed using the `predict` method of the `mcgfa` class.
`init_method`	Determines how starting values for the AECM algorithm are generated. Valid options are "emEM", "kmeans", "given" or "supervised". If `known` parameter is provided, this parameter is automatically set to "supervised". • `emEM`: initialization determined using the emEM method, where initialization candidates are randomly generated, and the AECM algorithm is applied for a small number of iterations. Whichever random initialization produces the best model in the short AECM runs is selected as the initialization for the full AECM runs.

- kmeans: begin by clustering observations using built-in kmeans R function.
- given: initialization manually specified in init_z argument
- supervised: Observations with known class labels are assigned to their known component with probability 1, while unlabelled components are initialized with equal prior probability of membership for each component

See Details for more information.

init_z — Used when init_method is set to given to manually prescribe the initialization used by the AECM algorithm. A list of row-stochastic matrices of same length as rG, where $z[i, g]$ represents the prior probability that observation i is a member of group g.

max_it — The maximum number of iterations for the AECM algorithm.

tol — The tolerance for the Aitken acceleration stopping criterion.

alpha_min — The minimum allowable value for alpha, which represents the proportion of "good" points in a given group.

eta_max — The maximum allowable value for eta, which is the covariance inflation factor for outlying points.

scale — If TRUE, the data is scaled before the algorithm is begun. Recommended.

parallel — If TRUE, computation takes place in parallel on several processors.

cores — Only relevent if parallel=TRUE. Determines the number of cores used in parallel computation. If left undefined, number of available cores is determined automatically.

silent — If TRUE, function will not print any output at completion.

ememargs — A list used to set options for the emEM initialization method:

- numstart: The number of random starting values
- iterations: The number of AECM iterations applied to each starting value
- model: The covariance model used in the emEM iterations
- q: The number of latent factors used in the emEM iterations

Details

This function implements the Mixtures of Contaminated Gaussian Factor Analyzers (MCGFA) model, for model-based clustering and classification. The approach is meant to be applied on on high-dimensional, and noisy, data. A description of the model can be found in Punzo, Blostein and McNicholas (2017). Parameter estimation is performed using an Alternating Expectation-Conditional Maximization (AECM) algorithm (Meng and Van Dyk, 1997).

Besides clustering into components, this algorithm also automatically detects outliers through the use of the contaminated Gaussian distribution. So, in the end each observation is classified in a nested fashion: first into components, then into as inliers/outliers.

To specify an individual MCGFA model, one must determine: the number of components (G), the number of latent factors (q), and the model name. The model name is one of the following thirty-two options:

- CCCCC, CCUCC, CUCCC, CUUCC, UCCCC, UCUCC, UUCCC, UUUCC
  CCCCU, CCUCU, CUCCU, CUUCU, UCCCU, UCUCU, UUCCU, UUUCU
  CCCUC, CCUUC, CUCUC, CUUUC, UCCUC, UCUUC, UUCUC, UUUUC
  CCCUU, CCUUU, CUCUU, CUUUU, UCCUU, UCUUU, UUCUU, UUUUU

Explanation of the model naming scheme can be found in the next subsection, as well as instructions on how to conveniently generate of subset of the full set of models.

The user may choose to fit a single model. However, usually many models are fit to the same data, and a model selection criterion is used to determine the best one. By multiple values for the `rG`, `rq` and `models` parameters, many models can be fit at once. The `mcgfa` function then selects the best model according to the Bayesian Information Criterion (BIC).

When fitting many models to large data, parallel computation may be employed by `mcgfa`, using the `parallel` parameter. This parallelization is provided by the `mcapply` function from the `parallel` package for R.

Model Names & Constraints: The model name indicates which constraints are to be imposed on to the covariance structure of the factor analysis model, and as well as on to the parameters η and α.

Because the MCGFA is a mixture of factor analyzers model, the covariance matrix of the g-th group, Σ_g, can be decomposed as follows:

$$\Sigma_g = \Lambda_g \Lambda_g' + \Psi_g$$

where Λ_g is a p by q factor loading matrix, and Ψ_g is a diagonal matrix that determines the noise variance for each of the p variables. The family of eight models is formed by introducing different sets of constraints on Λ and Ψ.

The five-letter model names are interpreted as follows. "C" indicates that the constraint is imposed, and "U" that it is not.

1. Λ constrained to be equal across groups ($\Lambda_g = \Lambda$)
2. Ψ constrained to be equal across groups ($\Psi_g = \Psi$)
3. Error variances constrained to be to be equal *within* groups ($\Psi_g = I\psi$)
4. α constrained to be equal across groups ($\alpha_g = \alpha$)
5. η constrained to be equal across groups ($\eta_g = \eta$)

The subset of models to fit is specified in the `models` argument as a character vector. For convenience, the user may also use the character 'X' in a model name to indicate a wildcard. For example, `'UUUXX'` is equivalent to the set `c('UUUCC','UUUCU','UUUUC','UUUUU')`. These wildcards may be combined, so for example `c('UUUUU','CCCCX')` is equivalent to `c('UUUUU','CCCCU','CCCCC')`. Any duplicate models generated by wildcards will be removed.

Initialization Methods: Because the AECM algorithm cannot guarantee convergence to a global maximum, success of the algorithm in reaching a good fit depends heavily on starting values. Starting values in this case refer to the prior probability of class membership for each observation. This is represented in an n-by-G stochastic matrix z, where n is the number of observations and G is the number of components. The element $z[i, j]$ indicates the prior probability that observation i lies in group G. Several different options are provided for the generation of this initialization matrix, through the `init_method` argument.

The default initialization method is `emEM`. In this case, several candidates for initial classification matrices are generated, and then the AECM algorithm is applied to each for a small number

of iterations. Whichever candidate achieves the best BIC value is selected as the initialization for the full AECM runs. This process occurs separately for each value of G, but only using one parsimonious model and one value of q. The number of candidates, number of iterations, parsimonious model and q used for the emEM initialization can be specified in the `ememargs` argument. Options for the emEM initialization can be provided in the `ememargs` argument.

The second option is `kmeans`, which uses the k-means clustering method to generate a "hard" initial classification. k-means is a fast, simple clustering algorithm that returns a hard classification.

Finally, using the `given` option, the user may provide a specific initialization matrix for each value of `G` in `rG`. The initialization is provided in the `init_z` argument. That argument must be a list of the same length as `rG`, where each element is a matrix with N rows and number of columns corresponding the matching value of `rG`. The $[i, j] - th$ element of each matrix indicates the initial probability that observation i lies in class j.

Classification: This function provides two methods for model-based classification. The first is usual fully-supervised classification. For this method, the model is fit to fully labelled data (by providing a `known` vector argument with no zeros). Then, after model fitting takes place, the `predict` generic function can be applied to the `mcgfa` object returned by this function, to predict the class labels of a matrix of unlabelled observations. This method has the advantage that as new observations arise, predictions can be made very quickly, without refitting the model.

The other form of classification is a form of semi-supervised learning. Semi-supervised classification makes use of both unlabelled and labelled data for training. The information that the unlabelled data provides about the structure of the dataset can improve classification. For this method, the `known` vector argument is simply provided with elements equal to zero, indicated that the corresponding observation has no known label. The MCGFA model is then fit and the unknown elements are classified as usual. The second method requires that the model be fitted again each time new data arrives. However, it is also completely possible to apply the `predict` method a model fit using semi-supervised classification.

Value

`X`	The data the model was fit to. If the data was scaled, then this matrix will be as well, with the centering and scale factors applied to it stored as attributes.
`all.bic`	An array of all of the BIC values for every model fitted, indexed by model name, number of groups and number of latent factors.
`model`	The name of the best model: determines the constraints on the covariance structure. See Details for more information.
`G`	The number of groups in the best model.
`q`	The number of latent factors for the best model.
`z`	The "soft" clustering matrix. The element in the `i`-th row and `g`-th column gives the posterior probability that observation `i` is in group `g`.
`group`	The "hard" classifications into groups. A vector of integers in the range `rG`, giving the a posteriori classification of each observation for the best model.
`isBad`	Similar, to `group`, the maximum a posteriori classification of each observation as "good" or "bad". If the `i`-th value is 1, the `i`-th observation is labelled as an outlier or noise in the best model.

isBad.soft	The "soft" classification vector of each observation as "good" or "bad". The closer the i-th value is to 1, the more likely the i-th observation is an outlier or noise.
mu	The matrix of group means (each column corresponds to a group), for the best model.
alpha	The proportion of "good" points for each group, for the best model.
eta	The contamination inflation factors for each group, for the best model.
lambda	The factor loading matrices of each group, for the best model.
psi	The error variance matrices of each group, for the best model.
sigma	The covariance matrices of each group, for the best model.
npar	The total number of free parameters in the best model.
iterations	The number of iterations of the AECM algorithm performed, for the fitting of the best model.
init_z	The initial z matrix used for initialization of the AECM algorithm, for each value of G in rG.

References

Meng, X.-L. and Van Dyk, D. (1997). The EM Algorithm - An old folksong sung to a fast new tune. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, **59(3)**, 511-567. Punzo, A., Blostein, M. and McNicholas, P. D. (2017). High-dimensional clustering with the contaminated Gaussian distribution *arXiv preprint arXiv:1408.2128v2*.

Examples

```
# TOY EXAMPLE: 2 GAUSSIAN COMPONENTS WITH NOISE
# small example for easy visualization
data(gaussnoise)
toy_classes <- gaussnoise[,1]
is_noise <- gaussnoise[,2]
X <- gaussnoise[,-c(1,2)]

set.seed(1)
toy_fit <- mcgfa(X, rG=1:3, rq=1, models="XXXCC")
# plot: outliers => triangle
plot(toy_fit)
# check clustering performance: 2 errors
table(toy_classes[1:175],toy_fit$group[1:175])
# check noise detection performance: 2 false positive, 3 false negative
table(is_noise,toy_fit$isBad)


# REAL DATA EXAMPLE: WINE DATA
data(wine)
# simulate clustering by completely hiding known classes from algorithm
X <- wine[,-1]
wine_classes <- wine[,1]
```

```
set.seed(1)
wine_fit <- mcgfa(X, rG=1:3, rq=1:4)

# check performance:
#  - correctly selected 3 groups
#  - 2 errors
table(wine_classes,wine_fit$group)

# CLASSIFICATION EXAMPLE: OLIVE DATA

# load data
data(olive)
X <- olive[,-c(1,2)]
# classes correspond to regions of Italy
regions <- olive[,1]

# take eighth of observations to form training data
set.seed(1)
train_ind <- sample.int(nrow(X),nrow(X)/8)

known <- rep(0,nrow(X))
known[train_ind] <- regions[train_ind]

# FULL SUPERVISION (only labelled data available to classifier)
# In this case, the model is formed using only the 71 labelled
# observations. Then this model is used to predict the class
# membership of the remaining 501 observations.
fit <- mcgfa(X[train_ind,], rq = 3:4, rG = 3)
pred <- predict(fit,X[-train_ind,])
# check classification performance: poor separation
table(regions[-train_ind],pred$hard)

# PARTIAL SUPERVISION (unlabelled data available to classifier)
# In this case, the model is formed using the 71 labelled
# observations, as well the the 501 unlabelled observations.
# As a part of this initial model fit, class predictions for the
# unlabelled observations are automatically generated.
fit2 <- mcgfa(X,rq=3:4,rG=3,known=known)
# check classification performance: extremely good, only 1 error
table(regions[-train_ind],fit2$group[-train_ind])
# This shows that the use of the  unlabelled observations during
# model fitting can significantly improve classification performance.
```

`olive` *Italian Olive Oil*

Description

Data on the percentage composition of eight fatty acids found by lipid fraction of 572 Italian olive oils. The data come from three regions: Southern Italy, Sardinia, and Northern Italy. Within each

region there are a number of different areas. Southern Italy comprises North Apulia, Calabria, South Apulia, and Sicily. Sardinia is divided into Inland Sardinia and Costal Sardinia. Northern Italy comprises Umbria, East Liguria, and West Liguria.

Usage

```
data(olive)
```

Format

A data frame with 572 observations and 10 columns. The first column gives the region: (1) Southern Italy, (2) Sardinia, or (3) Northern Italy. The second column gives the area: (1) North Apulia, (2) Calabria, (3) South Apulia, (4) Sicily, (5) Inland Sardinia, (6) Costal Sardinia, (7) East Liguria, (8) West Liguria, and (9) Umbria. The other eight columns contain the variables.

Source

These data are available within the GGobi software (Swayne et al., 2006).

References

Forina, M., Armanino, C., Lanteri, S. and Tiscornia, E. (1983). Classification of olive oils from their fatty acid composition. In *Food Research and Data Analysis*, pp. 189–214. Applied Science Publishers, London.

Forina, M. and Tiscornia, E. (1982). Pattern recognition methods in the prediction of Italian olive oil origin by their fatty acid content. *Annali di Chimica* **72**, 143–155.

Swayne, D.F., Cook, D., Buja, A., Lang, D.T., Wickham, H. and Lawrence, M. (2006). *GGobi Manual*. Sourced from www.ggobi.org/docs/manual.pdf.

---

wine *Italian Wine*

Description

Data on 27 chemical and physical properties of three types of wine (Barolo, Grignolino, Barbera) from the Piedmont region of Italy. The study did include one further variable but the sulphur measurements were not available.

Usage

```
data(wine)
```

Format

A data frame with 178 observations and 28 columns. The first column gives the type of wine: (1) Barolo, (2) Grignolino, or (3) Barbera. The other 27 columns contain the variables.

Source

Forina, M., Armanino, C., Castino, M. and Ubigli, M. (1986). Multivariate data analysis as a discriminating method of the origin of wines. *Vitis* **25**, 189–201.

Index