Package 'bgmm'

March 3, 2020

Type Package

Title Gaussian Mixture Modeling Algorithms and the Belief-Based Mixture Modeling

Version 1.8.4

Date 2020-03-03

Author Przemyslaw Biecek \{ }& Ewa Szczurek

Description Two partially supervised mixture modeling methods: soft-label and belief-based modeling are implemented. For completeness, we equipped the package also with the functionality of unsupervised, semi- and fully supervised mixture modeling. The package can be applied also to selection of the best-fitting from a set of models with different component numbers or constraints on their structures. For detailed introduction see: Przemyslaw Biecek, Ewa Szczurek, Martin Vingron, Jerzy Tiuryn (2012), The R Package bgmm: Mixture Modeling with Uncertain Knowledge, Journal of Statistical Software <doi:10.18637/jss.v047.i03>.

Maintainer Przemyslaw Biecek <Przemyslaw.Biecek@gmail.com>

Depends R (>= 2.0), mvtnorm, car, lattice, combinat

Suggests testthat

URL http://bgmm.molgen.mpg.de/

License GPL-3

Repository CRAN

NeedsCompilation no

Date/Publication 2020-03-03 13:00:02 UTC

R topics documented:

bgmm-package 2
CellCycle 3

chooseModels 4
crossval 6
DEprobs 7
genotypes 8
getModelStructure 9
init.model.params 10
miRNA 12
mModel 13
mModelList 16
plot.mModel 19
plot.mModelList 20
plotGIC 22
predict.mModel 23
simulateData 25
Ste12 27
Supplementary functions 28

Index **29**

`bgmm-package` *Belief-Based Gaussian Mixture Modeling*

Description

This package implements partially supervised mixture modeling methods: soft-label and belief-based modeling, the semi-supervised methods and for completeness also unsupervised and fully supervised methods for mixture modeling.

Details

Package:	bgmm
Type:	Package
Version:	1.8
Date:	2017-02-22
License:	GPL-3
LazyLoad:	yes

For short overview see the webpage http://bgmm.molgen.mpg.de/rapBGMM/.

Author(s)

Przemyslaw Biecek \& Ewa Szczurek

Maintainer: Przemyslaw Biecek <P.Biecek@mimuw.edu.pl>

References

Przemyslaw Biecek, Ewa Szczurek, Martin Vingron, Jerzy Tiuryn (2012), The R Package bgmm: Mixture Modeling with Uncertain Knowledge, Journal of Statistical Software.

See Also

Package for unsupervised learning of Gaussian mixture model `link{mclust}`, methods for supervised learning `link{MASS::lda()}`, `link{MASS::qda()}`.

Examples

```
## Do not run
## It could take more than one minute
#demo(bgmm)
```

`CellCycle` *Data for clustering of 384 cell cycle genes into five clusters corresponding to cell cycle phases*

Description

Time course expression data for 384 cell cycle genes (Cho et al., 1998). Literature examples of genes that should, and of genes that should not peak at each time point are given. For each cycle phase, there is a characteristic binary profile, stating when the phase occurs.

Usage

```
data(CellCycle)
```

Format

CellCycleData list: 17x384 CellCycleBeliefs list: 17x (35x2) CellCycleCenters matrix: 5x17 CellCycleClass vector: 384

Details

`CellCycleData`: A list, where each entry corresponds to one time-point. A given time point entry contains a vector with expression ratios for 384 cell cycle genes measured in this time point. `CellCycleBeliefs`: A list, where each entry corresponds to one time-point. A given time point entry gives the certainty (belief/plausibility) for each out of 35 example genes. Out of the genes, seven are known to peak in this time point and the remaining 28 are known to peak in other cycle phases. `CellCycleCenters`: A matrix, where the columns are the 17 time-points and the rows to the five cell phase clusters. A given entry in the matrix is equal to 1 if the genes from the cluster should peak in the time point, and 0 otherwise. `CellCycleClass`:Gives the true cluster for each gene. Each cluster corresponds to a cell cycle phase.

Author(s)

Ewa Szczurek

References

Cho, R., Campbell, M., Winzeler, E., Steinmetz, L., Conway, A., Wodicka, L., Wolfsberg, T., Gabrielian, A., Landsman, D., Lockhart, D., and Davis, R. (1998). A genome-wide transcriptional analysis of the mitotic cell cycle. Molecular Cell, 2(1), 65–73.

See Also

miRNA,Ste12

Examples

```
library(bgmm)
data(CellCycle)
print(CellCycleData)
print(CellCycleBeliefs)
print(CellCycleCenters)
print(CellCycleClass)
```

---

chooseModels *Selecting a subset of fitted models*

Description

The function chooseModels extracts a sublist of models that match constraints on the number of components or on the model structure. The function chooseOptimal returns the model which is the best according the given model selection criteria.

Usage

```
chooseModels(models, kList = NULL, struct = NULL)

chooseOptimal(models, penalty=2, ...)
```

Arguments

models	an object of the class mModelList which represents a list of fitted models.
kList	a vector which specifies the requested numbers of Gaussian components (constraints on the number of components).
struct	a vector which specifies four letter abbreviations of names of the requested model structures (constraints on the model structure).
penalty	a penalty parameter in the GIC criteria. This parameter can be a single number or a string, either "BIC" or "AIC".
...	other arguments that will be passed to the getGIC function.

Details

The function chooseModels() extracts a sublist of models from the `models` argument. The returned sublist is also an object of the class `mModelList` and is composed of models that simultaneously satisfy the restrictions of the number of Gaussian components defined by `kList` and restrictions of the model structure defined by `struct`. If the argument `kList` is set to NULL then no restrictions of the number of components are applied, same with the argument `struct`.

The function chooseOptimal() returns an object of the class `mModel` which is the single model that has the best (smallest) GIC score.

Value

An object of the class `mModelList` or `mModel`.

Author(s)

Przemyslaw Biecek

References

Przemyslaw Biecek, Ewa Szczurek, Martin Vingron, Jerzy Tiuryn (2012), The R Package bgmm: Mixture Modeling with Uncertain Knowledge, Journal of Statistical Software.

Examples

```
 simulated = simulateData(d=2, k=3, n=100, m=50, cov="0", within="E", n.labels=2)

 models3 = beliefList(X=simulated$X, knowns=simulated$knowns, B=simulated$B,
              kList=2:4, mean="D", within="D")
 plotGIC(models3, penalty="BIC")

## Do not run
## It could take more than one minute
# simulated = simulateData(d=2, k=3, n=300, m=60, cov="0", within="E", n.labels=2)
#
# models3 = beliefList(X=simulated$X, knowns=simulated$knowns, B=simulated$B,
#               kList=2:7, mean="D")
# plotGIC(models3, penalty="BIC")
#
# models4 = chooseModels(models3, kList=2:5, struct=c("DDDD","DDED","DDE0"))
# plot(models4)
# plotGIC(models4, penalty="BIC")
#
# model4 = chooseOptimal(models3, "BIC")
# plot(model4)
```

crossval *k-fold cross-validation for the specified model*

Description

The function `crossval()` performes k-fold cross-validation.

Usage

```
crossval(model = NULL, X = NULL, knowns = NULL, class = NULL,
   k = length(unique(class)), B = NULL, P = NULL, model.structure = getModelStructure(),
     ..., folds = 2, fun = belief)
```

Arguments

`model`	an object of the class `mModel`.
`X`	a data.frame with unknown realizations. If not supplied `X` is extracted from the `model` argument.
`knowns`	a data.frame with labeled realizations. If not supplied `knowns` is extracted from the `model` argument.
`class, B, P`	a vector of classes, beliefs and plausibilities. If not supplied they will be extracted from the `model` argument.
`fun`	function that will be used for modeling, one of `supervised`, `unsupervised`, `belief`, `soft`, `semisupervised`.
`model.structure, k, ...`	arguments that will be passed to `fun` function,
`folds`	number of folds in k-fold cross validation. Cannot be grated that number of labeled samples.

Details

The function `crossval()` divides the dataset into `k` equal subsets, the number of labeled cases versus number of unlabeled cases is keep as close to constant as possible (the subset are generated with stratification). Then each subset is used as test set against a train set build from all remaining sets. In total `k` new models are estimated thus this procedure is time consuming.

For each model the error is calculated as average absolute differences between the distribution of estimated posteriors and distribution of beliefs/plausibilities for labeled cases.

Value

The list with three vectors: errors calculated as mean absolute differences between estimated posteriors and initial beliefs for known cases, indexes of folds for both labeled and unlabeled cases.

Author(s)

Przemyslaw Biecek

References

Przemyslaw Biecek, Ewa Szczurek, Martin Vingron, Jerzy Tiuryn (2012), The R Package bgmm: Mixture Modeling with Uncertain Knowledge, Journal of Statistical Software.

Examples

```
set.seed(1313)
simulated = simulateData(d=2, k=3, n=300, m=60, cov="0", within="E", n.labels=2)
amodel = belief(X=simulated$X, knowns=simulated$knowns, B=simulated$B, k=4)
str(crossval(model=amodel, folds=6))

amodel = supervised(knowns=rbind(simulated$X, simulated$knowns), class=simulated$Ytrue)
str(crossval(model=amodel, folds=6, fun=supervised))
```

DEprobs *Signed probabilities of differential expression*

Description

The DEprobs function is an application of mixture modeling to differential gene expression analysis. The function takes as input a two- or three-component model of one-dimensional gene expression data. The data is assumed to represent log fold change expression values and be negative when the corresponding genes are down-regulated. The function calculates probabilities of differential expression for the data and gives them a sign according to the sign of the data.

Usage

```
DEprobs(model, verbose=FALSE)
```

Arguments

model	an object of the class mModel,
verbose	indicates whether log messeges should be prited out.

Details

Given the input model, the function identifies the component which corresponds to the differentially expressed genes as the one which looks differential according to the posterior probabilities.

For input models with two Gaussian components the differential component should be the one with a broader range (encompassing the other), or the one with higher deviation from 0 (we assume the data are centered around 0).

For input models with three Gaussian components there are two differential components: one corresponding to the down-regulated genes, and one corresponding to the up-regulated genes. Those components are identified as the ones with the lowest and the highest mean, respectively.

For `verbose=TRUE` the index of the differential component is printed out.

Value

An list with the following elements:

`diff.p.X`	a vector with the calculated signed differential expression probabilities for the unlabeled observations in the dataset `model$X`.
`diff.p.knowns`	a vector with the calculated signed differential expression probabilities for the unlabeled observations in the dataset `model$knowns`. For `model$knowns=NULL` `diff.p.knowns` is also NULL(null).
`diff.c`	the index (or two indexes, in case of a three-component input model) of the identified differential component.

Author(s)

Przemyslaw Biecek

References

Przemyslaw Biecek, Ewa Szczurek, Martin Vingron, Jerzy Tiuryn (2012), The R Package bgmm: Mixture Modeling with Uncertain Knowledge, Journal of Statistical Software.

Examples

```
data(Ste12)
X = Ste12Data[ match(names(Ste12Data), rownames(Ste12Beliefs), nomatch = 0) ==0 ]
knowns = Ste12Data[rownames(Ste12Beliefs)]
model = belief(X=X, knowns=knowns, B=Ste12Beliefs)
dep=DEprobs(model)
str(dep)
```

---

genotypes *Fluorescence signals corresponding to a given allele for 333 SNPs*

Description

The genotypes dataset describes 333 SNPs. Each SNP is characterized by the presence of one of its two possible alleles (or the presence of both of them). Therefore, the SNPs can be divided into three types. The first type corresponds to the SNPs with the first possible allele, the second type with the second allele, and the third with both alleles. The presence of the alleles is measured experimentally with fluorescence intensities. The dataset contains the intensities in the slots `X` and `knowns`.

15 SNPs in the dataset are given their correct type. These 'known' SNPs can be used as the input for the semi-, partially and fully superised modeling methods. They were selected by taking at random

five SNPs per each type. Their intensities are contained in the slot knowns. Their belief/plausibility values (given in the slot B) of the most probable type (the slot labels) are set to 0.95, and of the other two types are equal 0.025. The remaining 318 SNPs are kept unlabeled.

Usage

```
data(genotypes)
```

Format

X : a matrix with 318 rows (unlabeled SNPs) and 2 columns (alleles) knowns : a matrix with 15 rows (known SNPs) and 2 columns (alleles) B : a matrix with 15 rows (known SNPs) and 3 columns (types) labels : a vector of length 15 (types of the known SNPs)

Details

The rows of both the slots X and knowns correspond to the SNPs. For each SNP, the values in the columns represent the intensities of the fluorescence signal corresponding to the alleles of the SNP. The slot B corresponds to the belief matrix while labels contains the true types for the labeled SNPs.

References

Takitoh, S. Fujii, S. Mase, Y. Takasaki, J. Yamazaki, T. Ohnishi, Y. Yanagisawa, M. Nakamura, Y. Kamatani, N., Accurate automated clustering of two-dimensional data for single-nucleotide polymorphism genotyping by a combination of clustering methods: evaluation by large-scale real data, Bioinformatics (2007) Vol. 23, 408–413.

Examples

```
library(bgmm)
data(gnotypes)
```

---

getModelStructure *Model structure*

Description

This function creates an object which describes constraints over the model parameters.

Usage

```
getModelStructure(mean = "D", between = "D", within = "D", cov = "D")
```

Arguments

	Each argument is a single character, by default equal "D" (Different). If argument is set to "E" (or "0" for the argument cov) then the given parameter is constrained. By default all arguments are set to "D".
	mean="E" forces equality of the means between the components,
between	between="E" forces equality of the covariance matrices among the components,
within	within="E" forces equality of variances within each covariance matrix i to some constant vi and equality of covariances to some constant wi
cov	cov="0" forces equality of covariances within each covariance matrix i to "0",

Value

List of four elements specifying the constraints on 1) relations between the component means, 2) relations between the covariance matrices of the model components, 3) relations within each covariance matrix and 4) the covariances within each matrix. By default, the function returns an unconstrained structure.

Author(s)

Ewa Szczurek

References

Przemyslaw Biecek, Ewa Szczurek, Martin Vingron, Jerzy Tiuryn (2012), The R Package bgmm: Mixture Modeling with Uncertain Knowledge, Journal of Statistical Software.

Examples

```
getModelStructure()
getModelStructure(mean="E")
```

---

init.model.params *Initiation of model parameters*

Description

Methods for the initiation of model parameters for the EM algorithm. Two initiation procedures are implemented. The first procedure is available by setting the argument method='knowns'. It takes into account only labeled observations and is thus suitable for datasets with a high percentage of labeled cases. The second is available by setting method='all' and does not take the labeling into account.

Usage

```
init.model.params(X = NULL, knowns = NULL, class = NULL,
    k = length(unique(class)), method = "all", B = P, P = NULL)
```

Arguments

X	a data.frame with the unlabeled observations, its rows correspond to the observations while the columns correspond to variables/data dimensions.
knowns	a data.frame with the labeled observations, rows correspond to the observations while the columns correspond to variables/data dimensions.
B	a beliefs matrix with the distribution of beliefs for the labeled observations. If not specified and the argument P is given, the beliefs matrix is set to the value of P.
P	a matrix of plausibilities, specified only for the labeled observations. The function assumes that the remaining observations are unlabeled and gives them uniformly distributed plausibilities by default. If not specified and the argument B is given, the plausibilities matrix is set to the value of B.
class	class is a vector of labels for the known observations. If not specified, it is derived from eithter the argument B or P with the use of the MAP rule.
k	the desired number of model components.
method	a method for parameter initialization, one of following c("knowns","all"), see the section Details.

Details

For method='knowns', the initialization is based only on the labeled observations. i.e. those observations which have certain or probable components assigned. The initial model parameters for each component are estimated in one step from the observations that are assigned to this component (as in fully supervised learning).

If method='all' (default), the initialization is based on all observations. In this case, to obtain the initial set of model components, we start by clustering the data using the k-means algorithm (repeated 10 times to get stable results). The only exception is for one dimensional data. In such a case the clusters are identified by dividing the data into k equal subsets of observations, where the subsets are separated by empirical quantiles c(1/2k, 3/2k, 5/2k, ..., (2k-1)/2k). After this initial clustering each cluster is linked to one model component and initial values for the model parameters are derived from the clustered observations.

For the partially and semi-supervised methods, correspondence of labels from the initial clustering algorithm and labels for the observations in the knowns dataset rises a technical problem. The cluster corresponding to component y should be as close as possible to the set of labeled observations with label y.

Note that for the unsupervised modeling this problem is irrelevant and any cluster may be used to initialize any component.

To mach the cluster labels with the labels of model components a greedy heuristic is used. The heuristic calculates weighted distances between all possible pairs of cluster centers and sets of observations grouped by their labels. In each step, the pair with a minimal distance is chosen (the pair: a group of observations with a common label and a cluster, for which the center of the group is the closest to the center of the cluster). For the chosen pair, the cluster is labeled with the same label as the group of observations. Then, this pair is removed and the heuristic repeats for the reduced set of pairs.

Value

A list with the following elements:

pi	a vector of length k with the initial values for the mixing proportions.
mu	a matrix with the means' vectors with the initial values for k components.
cvar	a three-dimensional matrix with the covariance matrices with the initial values for k components.

Author(s)

Przemyslaw Biecek

References

Przemyslaw Biecek, Ewa Szczurek, Martin Vingron, Jerzy Tiuryn (2012), The R Package bgmm: Mixture Modeling with Uncertain Knowledge, Journal of Statistical Software.

Examples

```
data(genotypes)
initial.params = init.model.params(X=genotypes$X, knowns=genotypes$knowns,
class = genotypes$labels)
str(initial.params)
```

---

miRNA *miRNA transfection data for miR1 and miR124 target genes*

Description

miRNA transfection data (Lim et al., 2005) and knowledge from computational miRNA target predictions.

Usage

```
data(miRNA)
```

Format

miR1Data vector: 117, miR124Data vector: 117, miRNABeliefs matrix of example certainty: 26 x 2, miRNAClass vector: 117

Details

`miR1Data` Log2 expression ratios of miR1 transfection versus wild type, for 117 genes. `miR124Data` Log2 expression ratios of miR124 transfection versus wild type, for 117 genes. `miRNABeliefs` Gives the certainty (belief/plausibility) for each out of 26 example miRNA targets to belong to their cluster. `miRNAClass` Gives the true cluster for each gene. Cluster 1 corresponds to the experimentally verified targets of miR1. Cluster 2 corresponds to the targets of miR124.

Author(s)

Ewa Szczurek

References

Lim, L. P., Lau, N. C., Garrett-Engele, P., Grimson, A., Schelter, J. M., Castle, J., Bartel, D. P., Linsley, P. S., and Johnson, J. M. (2005). Microarray analysis shows that some microRNAs down-regulate large numbers of target mRNAs. Nature, 433(7027).

See Also

Ste12,CellCycle

Examples

```
library(bgmm)
data(miRNA)
print(miR1Data)
print(miR124Data)
print(miRNABeliefs)
print(miRNAClass)
```

---

mModel *Fitting Gaussian Mixture Model*

Description

These functions fit different variants of Gaussian mixture models. These variants differ in the fraction of knowledge utilized into the the fitting procedure.

Usage

```
belief(X, knowns, B = NULL, k = ifelse(!is.null(B), ncol(B),
    ifelse(!is.null(P), ncol(P), length(unique(class)))), P = NULL,
    class = map(B), init.params = init.model.params(X, knowns,
      B = B, P = P, class = class, k = k), model.structure = getModelStructure(),
    stop.likelihood.change = 10^-5, stop.max.nsteps = 100, trace = FALSE,
    b.min = 0.025,
    all.possible.permutations=FALSE, pca.dim.reduction = NA)

soft(X, knowns, P = NULL, k = ifelse(!is.null(P), ncol(P),
    ifelse(!is.null(B), ncol(B), length(unique(class)))), B = NULL,
    class = NULL, init.params = init.model.params(X, knowns,
        class = class, B = P, k = k),
    model.structure = getModelStructure(), stop.likelihood.change = 10^-5,
    stop.max.nsteps = 100, trace = FALSE, b.min = 0.025,
all.possible.permutations=FALSE, pca.dim.reduction = NA, ...)
```

```
semisupervised(X, knowns, class = NULL, k = ifelse(!is.null(class),
    length(unique(class)), ifelse(!is.null(B), ncol(B), ncol(P))),
    B = NULL, P = NULL, ..., init.params = NULL,
all.possible.permutations=FALSE, pca.dim.reduction = NA)

supervised(knowns, class = NULL, k = length(unique(class)), B = NULL, P = NULL,
    model.structure = getModelStructure(), ...)

unsupervised(X, k, init.params=init.model.params(X, knowns=NULL, k=k),
      model.structure=getModelStructure(), stop.likelihood.change=10^-5,
      stop.max.nsteps=100, trace=FALSE, ...)
```

Arguments

X	a data.frame with the unlabeled observations. The rows correspond to the observations while the columns to variables/dimensions of the data.
knowns	a data.frame with the labeled observations. The rows correspond to the observations while the columns to variables/dimensions of the data.
B	a beliefs matrix which specifies the distribution of beliefs for the labeled observations. The number of rows in B should equal the number of rows in the data.frame knowns. It is assumed that both the observations in B and in knowns are given in the same order. Columns correspond to the model components. If matrix B is provided, the number of columns has to be less or equal k. Internally, the matrix B is completed to k columns.
P	a matrix of plausibilities, i.e., weights of the prior probabilities for the labeled observations. If matrix P is provided, the number of columns has to be less or equal k. The came conditions as for B apply.
class	a vector of classes/labels for the labeled observations. The number of its unique values has to be less or equal k.
k	a number of components, by default equal to the number of columns of B.
init.params	initial values for the estimates of the model parameters (means, variances and mixing proportions), by default derived with the use of the init.model.params function.
stop.likelihood.change, stop.max.nsteps	the parameters for the EM algorithms defining the stop criteria, i.e., the minimum required improvement of loglikelihood and the maximum number of steps.
trace	if trace=TRUE the loglikelihoods for every step of EM algorithm are printed out.
model.structure	an object returned by the getModelStructure function, which specifies constraints for the parameters of the model to be fitted.
b.min	this argument is passed to the init.model.params function.
...	these arguments will be passed tothe init.model.params function.
all.possible.permutations	If equal TRUE, all possible initial parameters' permutations of components are considered. Since there is kList! permutations, model fitting is repeated kList! times. As a result, only the model with the highest likelihood is returned.

`pca.dim.reduction`
: Since the fitting for high dimensional space is numerically a bad idea an attempt to PCA will be performed if `pca.dim.reduction !-FALSE`. If equal `NA` then the target dimension is data driven, if it's a number then this will be the target dimension.

Details

In the `belief()` function, if the argument `B` is not provided, it is by default initialized from the argument `P`. If the argument `P` is not provided, `B` is derived from the `class` argument with the use of the function `get.simple.beliefs()` which assigns `1-(k-1)*b.min` to the component given by `class` and `b.min` to all remaining components.

In the `soft()` function, if the argument `P` is not provided, it is by default initialized from the argument `B`. If the argument `B` is not provided, `P` is derived from the `class` argument as in the `belief()` function.

In the `supervised()` function, if the argument `class` is not provided, it is by default initialized from argument `B` or `P`, taking the label of each observation as its most believed or plausible component (by the MAP rule).

The number of columns in the beliefs matrix `B` or in the matrix of plausibilities `P` may be smaller than the number of model components defined by the argument `k`. Such situation corresponds to the scenario when the user does not know any examples for some component. In other words, this component is not used as a label for any observation, and thus can be omitted from the beliefs matrix. An equivalent would be to include a column for this component and fill it with beliefs/plausibilities equal 0.

Slots in the returned object are listed in section Value. The returned object differs slighty with respect to the used function. Namely, the `belief()` function returns an object with the slot `B`. The function `soft()` returns an object with a slot `P`, while the functions `supervised()` and `semisupervised()` return objects with a slot `class` instead.

The object returned by the function `supervised()` does not have the slot `X`.

Value

An object of the class `mModel`, with the following slots:

`pi`	a vector with the fitted mixing proportions
`mu`	a matrix with the means' vectors, fitted for all components
`cvar`	a three-dimensional matrix with the covariance matrices, fitted for all components
`X`	the unlabeled observations
`knowns`	the labeled observations
`B`	the beliefs matrix
`n`	the number of all observations
`m`	the number of the unlabeled observations
`k`	the number of fitted model components
`d`	the data dimension

likelihood	the log-likelihood of the fitted model
n.steps	the number of steps performed by the EM algorithm
model.structure	the set of constraints kept during the fitting process.

Author(s)

Przemyslaw Biecek

References

Przemyslaw Biecek, Ewa Szczurek, Martin Vingron, Jerzy Tiuryn (2012), The R Package bgmm: Mixture Modeling with Uncertain Knowledge, Journal of Statistical Software.

Examples

```
data(genotypes)

modelSupervised = supervised(knowns=genotypes$knowns,
            class=genotypes$labels)
plot(modelSupervised)

modelSemiSupervised = semisupervised(X=genotypes$X,
            knowns=genotypes$knowns, class = genotypes$labels)
plot(modelSemiSupervised)

modelBelief = belief(X=genotypes$X,
            knowns=genotypes$knowns, B=genotypes$B)
plot(modelBelief)

modelSoft = soft(X=genotypes$X,
            knowns=genotypes$knowns, P=genotypes$B)
plot(modelSoft)

modelUnSupervised = unsupervised(X=genotypes$X, k=3)
plot(modelUnSupervised)
```

---

mModelList *Fitting Gaussian mixture model or collection of models*

Description

These functions fit collection of models of one particular variant/class. Models to be fitted may differ in the requested number of Gaussian components or in the requested model structure.

Usage

```
mModelList(X, knowns, B = NULL, P = NULL, class = NULL, kList = ncol(B),
    init.params = NULL, stop.likelihood.change = 10^-5, stop.max.nsteps = 100,
    trace = FALSE, mean = c("D", "E"), between = c("D", "E"), within = c("D",
     "E"), cov = c("D", "0"), funct = belief, all.possible.permutations = FALSE, ...)

beliefList(..., funct=belief)

softList(..., funct=soft)

semisupervisedList(..., funct=semisupervised)

unsupervisedList(X, kList = 2, ...)
```

Arguments

X	a data.frame with the unlabeled observations. The rows correspond to the observations while the columns to variables/dimensions of the data.
knowns	a data.frame with the labeled observations. The rows correspond to the observations while the columns to variables/dimensions of the data.
B	a beliefs matrix which specifies the distribution of beliefs for the labeled observations. The number of rows in B should equal the number of rows in the data.frame knowns. It is assumed that both the observations in B and in knowns are given in the same order. Columns correspond to the model components. If matrix B is provided, the number of columns has to be less or equal k. Internally, the matrix B is completed to k columns.
P	a matrix of plausibilities, i.e., weights of the prior probabilities for the labeled observations. If matrix P is provided, the number of columns has to be less or equal k. The came conditions as for B apply.
class	a vector of classes/labels for the labeled observations. The number of its unique values has to be less or equal min(kList).
kList	a vector or a list with numbers of Gaussian components to fit. By default it is one number equal to the number of columns of B.
init.params	initial values for the estimates of the model parameters (means, variances and mixing proportions). The initial parameters are internally passed to the funct function.
stop.likelihood.change, stop.max.nsteps, trace	the parameters for the EM algorithm. Internally, these parameters are passed to the funct function.
mean, between, within, cov	four vectors which define the model structures for models to be fitted. For example, if mean="E", only models with constrained means are considered (means of Gaussian components are forced to be equal). On the other hand if mean=c("E","D"), both models with constrained means and models without constraint on the means are fitted.
funct	a function which fits a variant of Gaussian mixture model, one of the: belief, soft, semisupervised or unsupervised functions.

`...` arguments that are passed to function `funct`.

`all.possible.permutations`
If equal `TRUE`, all possible initial parameters' permutations of components are considered. Since there is kList! permutations, model fitting is repeated kList! times. As a result only the model with the highest likelihood is returned.

Details

Arguments `kList`, as well as `mean`, `between`, `within`, and `cov` define the list of models to be fitted. All combinations of specified model sizes and model structures are considered. List of fitted models is returned as a result.

The argument `funct` defines which variant of Gaussian mixture models should be used for model fitting. One can use the wrappers `beliefList()`, `softList()`, `semisupervisedList()`, `unsupervisedList()` which call the `mModelList()` function and have a prespecified argument `funct`.

Value

An object of the class mModelList, with the following slots:

`models`	a list of models, each of the class `mModel`
`loglikelihoods`	a vector with log likelihoods of the models from list `models`
`names`	a vector with names of the models from list `models`
`params`	a vector with the number of parameters of models from list `models`
`kList`	equals the input argument `kList`

Author(s)

Przemyslaw Biecek

References

Przemyslaw Biecek, Ewa Szczurek, Martin Vingron, Jerzy Tiuryn (2012), The R Package bgmm: Mixture Modeling with Uncertain Knowledge, Journal of Statistical Software.

See Also

`mModel`, `getModelStructure`

Examples

```
simulated = simulateData(d=2, k=3, n=100, m=60, cov="0", within="E", n.labels=2)

models1=mModelList(X=simulated$X, knowns=simulated$knowns, B=simulated$B,
            kList=3:4, mean=c("D","E"), between="D", within="D",
            cov="0", funct=belief)

plot(models1)
plotGIC(models1, penalty="BIC")
```

```
## Do not run
## It could take more than one minute
# simulated = simulateData(d=2, k=3, n=300, m=60, cov="0", within="E", n.labels=2)
#
# models1=mModelList(X=simulated$X, knowns=simulated$knowns, B=simulated$B,
#             kList=3, mean=c("D","E"), between=c("D","E"), within=c("D","E"),
#             cov=c("D","0"), funct=belief)
# plot(models1)
# plotGIC(models1, penalty="BIC")
#
# models2 = beliefList(X=simulated$X, knowns=simulated$knowns, B=simulated$B,
#                kList=2:7, mean="D", between="D", within="E", cov="0")
# plot(models2)
# plotGIC(models2, penalty="BIC")
#
# models3 = beliefList(X=simulated$X, knowns=simulated$knowns, B=simulated$B,
#               kList=2:7, mean="D")
# plotGIC(models3, penalty="BIC")
```

`plot.mModel` *Plotting a Graphical Visualization of a Gaussian Model or a List of Models*

Description

The generic function `plot` is used to visualize the data set and Gaussian model components fitted to this data. On the resulting plot the observations without labels are presented with black points, whereas the labeled observations are marked by different colors and different symbols. The fitted Gaussian components are represented by ellipses into the two-dimensional case and by densities in the one dimensional case. If data has more than two dimensions thus graphs are presented on the subspace generated by first two PCA components. Note that the estimation is done in higher dimension and the reduction to 2D is done only for illustration. That gives different results than data reduction prior to modeling process.

Usage

```
## S3 method for class 'mModel'
plot(x, ...)
```

Arguments

`x`	an object of the class `mModel`.
`...`	graphical arguments that are passed to the underlying `plot()` function.

Details

For one dimensional data the width of the density corresponds to standard deviation of the fitted Gaussian component. Fitted means are marked by vertical dashed lines.

For two dimensional data ellipses represents covariances for the corresponding model components.

For more dimensional points and ellipses are projected into 2D subspace spanned by first two PCA components.

Author(s)

Przemyslaw Biecek

References

Przemyslaw Biecek, Ewa Szczurek, Martin Vingron, Jerzy Tiuryn (2012), The R Package bgmm: Mixture Modeling with Uncertain Knowledge, Journal of Statistical Software.

Examples

```
data(genotypes)
modelSupervised = supervised(knowns=genotypes$knowns, class=genotypes$labels)
plot(modelSupervised)

# semi-supervised modeling
modelSemiSupervised = semisupervised(X=genotypes$X, knowns=genotypes$knowns,
                          class = genotypes$labels)
plot(modelSemiSupervised)

# belief-based modeling
modelBelief = belief(X=genotypes$X, knowns=genotypes$knowns, B=genotypes$B)
plot(modelBelief)

# soft-label modeling
modelSoft = soft(X=genotypes$X, knowns=genotypes$knowns, P=genotypes$B)
plot(modelSoft)

# unsupervised modeling
modelUnSupervised = unsupervised(X=genotypes$X, k=3)
plot(modelUnSupervised)
```

`plot.mModelList` *Plotting a graphical visualization of a model or a list of models*

Description

The function `plot.mModelList()` creates a grid of panels and then plots a set of input fitted models in the consecutive panels. The `plot.mModel()` function is used to plot each single model.

Usage

```
## S3 method for class 'mModelList'
plot(x, ...)
```

Arguments

x	an object of the class mModelList.
...	graphical arguments that are passed to underlying plot() function.

Details

The argument x is a list of models. If these models differ both by component numbers and by the model structures, in the resulting grid of panels columns correspond to the different model structures while rows correspond to the different component numbers.

If considered models differ only by component numbers or only by the model structures, the grid of panels is as close to square as possible and consecutive panels contain consecutive models from the list of models x.

Author(s)

Przemyslaw Biecek

References

Przemyslaw Biecek, Ewa Szczurek, Martin Vingron, Jerzy Tiuryn (2012), The R Package bgmm: Mixture Modeling with Uncertain Knowledge, Journal of Statistical Software.

See Also

plot.mModel

Examples

```
 simulated = simulateData(d=2, k=3, n=100, m=60, cov="0", within="E", n.labels=2)
 models1=mModelList(X=simulated$X, knowns=simulated$knowns, B=simulated$B,
             kList=3:4, mean=c("D","E"), between="D", within="D",
             cov="0", funct=belief)
 plot(models1)

## Do not run
## It could take more than one minute
# simulated = simulateData(d=2, k=3, n=300, m=60, cov="0", within="E", n.labels=2)
#
# models1=mModelList(X=simulated$X, knowns=simulated$knowns, B=simulated$B,
#             kList=3, mean=c("D","E"), between=c("D","E"), within=c("D","E"),
#             cov=c("D","0"), funct=belief)
# plot(models1)
#
# models2 = beliefList(X=simulated$X, knowns=simulated$knowns, B=simulated$B,
#                kList=2:7, mean="D", between="D", within="E", cov="0")
# plot(models2)
#
# models3 = beliefList(X=simulated$X, knowns=simulated$knowns, B=simulated$B,
#               kList=2:7, mean="D")
# plot(models3)
```

plotGIC *Plotting GIC scores*

Description

The function plotGIC() plots the GIC scores for an input collection of models. The function getGIC() extracts GIC for given model and penalty function. The function getDF() extracts the number of degree of freedom for model parameters.

Usage

```
plotGIC(models, penalty = 2, plot.it = TRUE, ...)

getGIC(model, p = 2, whichobs="unlabeled")

getDF(model)
```

Arguments

models	an object of the class mModelList or a matrix of GIC scores.
model	an object of the class mModel.
penalty	a penalty for the GIC criteria. This parameter can be a single number or a string, on of the "BIC", "AIC", "AIC3", "AIC4", "AICc", "AICu", "CAIC", "BIC", "MDL", "CLC", "ICL-BIC", "AWE".
p	same as penalty,
whichobs	one of "unlabeled", "labeled", "all". This parameter specify which observations should be used in the likelihood and gic score calculation,
plot.it	a logical value, if TRUE then the chart with the GIC scores will be plotted.
...	other arguments that will be passed to the getGIC function.

Details

The function plotGIC() calculates the GIC scores for each model from the models list and, given plot.it=TRUE, plots a dotchart with the calculated GIC scores.

As a result the function plotGIC() returns a matrix with the calculated GIC scores. This matrix or its submatrix can be used in next call of the plotGIC() function as models argument. The columns of the matrix correspond to different component numbers of the models, while the rows correspond to their structures. The structures are coded with four-letter strings. The letters refer, in order from left to right: first, the relation between the means' vectors of the components, which can either be equal (letter "E") or unconstrained ("D"). Second, the relation between covariance matrices, which can all either be equal ("E"), or unconstrained ("D"). Third, the relation between the data vector components (corresponding to data dimensions) within each covariance matrix, i.e. each covariance matrix can either have all variances equal to some constant and all covariances equal to some constant ("E") or can be unconstrained ("D"). Fourth, the covariances in each covariance matrix, which can either all be forced to equal 0 ("0") or be unconstrained ("D").

The best model, i.e. model with the smallest GIC score is marked with a star on the plotted chart.

Value

The matrix with GIC scores calculated for the list of models specified by the `models` argument.

Author(s)

Przemyslaw Biecek

References

Przemyslaw Biecek, Ewa Szczurek, Martin Vingron, Jerzy Tiuryn (2012), The R Package bgmm: Mixture Modeling with Uncertain Knowledge, Journal of Statistical Software.

Examples

```
simulated = simulateData(d=2, k=3, n=100, m=60, cov="0", within="E", n.labels=2)
models1   = mModelList(X=simulated$X, knowns=simulated$knowns, B=simulated$B,
            kList=3:4, mean=c("D","E"), between="D", within="D",
            cov="0", funct=belief)
plotGIC(models1, penalty="BIC")

## Do not run
## It could take more than one minute
# simulated = simulateData(d=2, k=3, n=300, m=60, cov="0", within="E", n.labels=2)
#
# models1=mModelList(X=simulated$X, knowns=simulated$knowns, B=simulated$B,
#             kList=3, mean=c("D","E"), between=c("D","E"), within=c("D","E"),
#             cov=c("D","0"), funct=belief)
# plotGIC(models1, penalty="BIC")
#
# models2 = beliefList(X=simulated$X, knowns=simulated$knowns, B=simulated$B,
#                kList=2:7, mean="D", between="D", within="E", cov="0")
# plotGIC(models2, penalty="BIC")
#
# models3 = beliefList(X=simulated$X, knowns=simulated$knowns, B=simulated$B,
#               kList=2:7, mean="D")
# plotGIC(models3, penalty="BIC")
```

`predict.mModel` *Predictions for fitted Gaussian component model*

Description

For every row in the matrix `X` the posterior probability of belonging to class `i` is calculated.

Usage

```
## S3 method for class 'mModel'
predict(object, X, knowns = NULL, B = NULL, P = NULL, ...)
```

Arguments

object	an object of the class mModel,
X	a matrix or data.frame in which number of columns is equal to object$d.
knowns	a data.frame or matrix with the labeled observations. If the argument knowns is specified then eighter B or P need to be specified.
P	a matrix with plausibilities for object knowns.
B	a matrix with beliefs for object knowns.
...	all other arguments will be neglected.

Details

The matrix tij of posterior probabilities is calculated as normalized products of priors pi's and density of model components in values specified by rows of the matrix X.

If arguments knowns and B are specified then the priors's for objects in knowns are replaced by belief matrix B. If arguments knowns and P are specified then the priors's for objects in knowns are multiplied by plausibility matrix P.

Value

An list with the following elements:

tij.X, tij.knowns
: the matrix tij.X is a matrix with number of rows equal to number of rows in the matrix X and the number of columns equal to the number of components in model defined by argument object. Values in this matrix are posterior probabilities that observation i belongs to component j. The slot tij.knowns is equal to NULL if neither B nor P are specified, otherwise it is a matrix with number of rows equal to number of rows in the matrix knowns and contains posterior probabilities for observarions with specified belief or plausibilities matrix

class.X, class.knowns
: vactors of labels/classes obtained with the MAP rule. The vector class.X corresponds to observations in X while the vector class.knowns corresponds to observations in knowns.

Author(s)

Przemyslaw Biecek, Ewa Szczurek, Martin Vingron, Jerzy Tiuryn (2012), The R Package bgmm: Mixture Modeling with Uncertain Knowledge, Journal of Statistical Software.

References

http://bgmm.molgen.mpg.de

See Also

belief

Examples

```
data(genotypes)

modelSoft = soft(X=genotypes$X, knowns=genotypes$knowns, P=genotypes$B)

preds = predict(modelSoft, X = genotypes$X)
str(preds)
```

`simulateData` *Dataset generation*

Description

The function `simulateData` generates an artificial dataset from a mixture of Gaussian components with a given set of parameters.

Usage

```
simulateData(d = 2, k = 4, n = 100, m = 10, mu = NULL, cvar = NULL,
   s.pi = rep(1/k, k), b.min = 0.02, mean = "D", between = "D",
   within = "D", cov = "D", n.labels = k)
```

Arguments

`d`	the dimension of the data set,
`k`	the number of the model components,
`n`	the total number of observations, both labeled and unlabeled,
`mu`	a matrix with `k` rows and `d` columns, which defines the means' vectors for the corresponding model components. If not specified, by default its values are generated from a normal distribution N(0,49),
`cvar`	a three-dimensional array with the dimensions (`k`, `d`, `d`). If not specified, each covariance matrix is generated in three steps: first, 2*`d` samples from a `d`-dimensional normal distribution N(0, Id) are generated. Next, a covariance matrix d x d for these samples is calculated. Finally, the resulting sample covariance matrix is scaled by a factor generated from an exponential distribution Exp(1),
`s.pi`	a vector of `k` probabilities, i.e. the mixing proportions of the model. The mixing proportions specify a multinomial distribution over the components, from which the numbers of observations in each cluster are generated. By default a uniform distribution is used.
`mean, between, within, cov`	constraints on the model structure. By default all are equal to "D". If other values are set, the parameters `mu` and `cvar` are adjusted to match the specified constraints,
`m`	the number of the observations, for which the beliefs are to be calculated,

b.min	the belief that an observation does not belong to a component. Formally, the belief bij for the observation i to belong to component j is equal b.min if i is not generated from component j. Thus, the belief that i belongs to its true component is set to 1-b.min*(n.labels-1), and b.min is constrained that b.min$<1/$n.labels. By default b.min=0.02,
n.labels	the number of components used as labels, defining the number of columns in the resulting beliefs matrix. By default n.labels equals k, but the user can specify a smaller number. Using this argument the user can define a scenario in which the data are generated from a mixture of three components, but only two of them are used as labels in the beliefs matrix (applied in the example below).

Value

An list with the following elements:

X	the matrix of size n-m rows and d columns with generated values of unlabeled observations,
knowns	the matrix of size m rows and d columns with generated values of labeled observations,
B	the belief matrix of the size m rows and k columns derived for knowns matrix,
model.params	the list of model parameters,
Ytrue	indexes of the true Gaussian components from which each observation was generated. Lables for knowns go first.

Author(s)

Przemyslaw Biecek

References

Przemyslaw Biecek, Ewa Szczurek, Martin Vingron, Jerzy Tiuryn (2012), The R Package bgmm: Mixture Modeling with Uncertain Knowledge, Journal of Statistical Software.

Examples

```
simulated = simulateData(d=2, k=3, n=300, m=60, cov="0", within="E", n.labels=2)
model = belief(X = simulated$X, knowns = simulated$knowns, B=simulated$B)
plot(model)

simulated = simulateData(d=1, k=2, n=300, m=60, n.labels=2)
model = belief(X = simulated$X, knowns = simulated$knowns, B=simulated$B)
plot(model)
```

Ste12 *Ste12 knockout data under pheromone treatment versus wild type; Examples of Ste12 targets; Binding p-values of Ste12 to those targets.*

Description

Ste12 knockout expression data (Roberts et al., 2002) and knowledge from a Ste12 binding experiment (Harbison et al., 2004) used for identifying Ste12 target genes under pheromone treatment.

Usage

```
data(Ste12)
```

Format

Ste12Data vector: 601 Ste12Beliefs matrix of example certainty: 42 x 2 Ste12Binding vector: 42

Details

`Ste12Data` Log2 expression ratios of Ste12 knockout versus wild type, both under 50nM alpha-factor treatment for 30min. This data is for 601 genes that had more than 1.5 fold change in expression after pheromone treatment versus wild type. `Ste12Beliefs`: Gives the certainty (belief/plausibility) for each out of 42 example Ste12 targets to belong to their cluster. `Ste12Beliefs`: Gives the certainty (belief/plausibility) for each out of 42 example Ste12 targets to belong to their cluster. The 42 examples were chosen to meet two criteria: (1) Had a binding p-value <0.0001 (see `Ste12Binding`), and (2) Had a 2-fold change in response to pheromone treatment (versus wild-type) `Ste12Binding`: Gives the binding p-value for each example Ste12 target (see `Ste12Belief`).

Author(s)

Ewa Szczurek

References

Roberts, C. J., Nelson, B., Marton, M. J., Stoughton, R., Meyer, M. R., Bennett, H. A., He, Y. D., Dai, H., Walker, W. L., Hughes, T. R., Tyers, M., Boone, C., and Friend, S. H. (2000). Signaling and Circuitry of Multiple MAPK Pathways Revealed by a Matrix of Global Gene Expression Profiles. Science, 287(5454), 873–880.

Harbison, C. T., Gordon, D. B., Lee, T. I., Rinaldi, N. J., Macisaac, K. D., Danford, T. W., Hannett, N. M., Tagne, J.-B., Reynolds, D. B., Yoo, J., Jennings, E. G., Zeitlinger, J., Pokholok, D. K., Kellis, M., Rolfe, P. A., Takusagawa, K. T., Lander, E. S., Gifford, D. K., Fraenkel, E., and Young, R. A. (2004). Transcriptional regulatory code of a eukaryotic genome. Nature, 431(7004), 99–104.

See Also

miRNA,CellCycle

Examples

```
data("Ste12")
print(Ste12Data)
print(Ste12Beliefs)
print(Ste12Binding)
```

`Supplementary functions`

Set of supplementary functions for bgmm package

Description

Set of supplementary functions for bgmm package.

Usage

```
## S3 method for class 'numeric'
determinant(x, logarithm = TRUE, ...)

 map(B)

 loglikelihood.mModel(model, X)
```

Arguments

x	a single number.
X	a data.frame with the unlabeled observations, the rows correspond to the observations and the columns to the dimensions of the data.
B	a beliefs matrix with the distribution of beliefs for the labeled observations.
model	an object of the class mModel.
logarithm, ...	these arguments are ignored.

Author(s)

Przemyslaw Biecek

References

Przemyslaw Biecek, Ewa Szczurek, Martin Vingron, Jerzy Tiuryn (2012), The R Package bgmm: Mixture Modeling with Uncertain Knowledge, Journal of Statistical Software.

Examples

```
data(genotypes)

map(genotypes$B)
```

Index

∗Topic **datasets**
CellCycle, 3
genotypes, 8
miRNA, 12
Ste12, 27
∗Topic **package**
bgmm-package, 2

belief, *24*
belief *(*mModel*)*, 13
beliefList *(*mModelList*)*, 16
bgmm *(*bgmm-package*)*, 2
bgmm-package, 2

CellCycle, 3, *13*, *27*
CellCycleBeliefs *(*CellCycle*)*, 3
CellCycleCenters *(*CellCycle*)*, 3
CellCycleClass *(*CellCycle*)*, 3
CellCycleData *(*CellCycle*)*, 3
chooseModels, 4
chooseOptimal *(*chooseModels*)*, 4
crossval, 6

DEprobs, 7
determinant.numeric *(*Supplementary functions*)*, 28

genotypes, 8
getDF *(*plotGIC*)*, 22
getGIC *(*plotGIC*)*, 22
getModelStructure, 9, *14*, *18*

init.model.params, 10, *14*

loglikelihood.mModel *(*Supplementary functions*)*, 28

map *(*Supplementary functions*)*, 28
miR124Data *(*miRNA*)*, 12
miR1Data *(*miRNA*)*, 12
miRNA, *4*, 12, *27*
miRNABeliefs *(*miRNA*)*, 12
miRNAClass *(*miRNA*)*, 12
mModel, *5*, 13, *18*
mModelList, *4*, *5*, 16

plot.mModel, 19, *21*
plot.mModelList, 20
plotGIC, 22
predict.mModel, 23

semisupervised *(*mModel*)*, 13
semisupervisedList *(*mModelList*)*, 16
simulateData, 25
soft *(*mModel*)*, 13
softList *(*mModelList*)*, 16
Ste12, *4*, *13*, 27
Ste12Beliefs *(*Ste12*)*, 27
Ste12Binding *(*Ste12*)*, 27
Ste12Data *(*Ste12*)*, 27
supervised *(*mModel*)*, 13
supervisedList *(*mModelList*)*, 16
Supplementary functions, 28

unsupervised *(*mModel*)*, 13
unsupervisedList *(*mModelList*)*, 16