netresponse

October 25, 2011

`ICMg.combined.sampler`
ICMg.combined.sampler

Description

Main function of the ICMg algorithm. ICMg.combined.sampler computes samples from the posterior of the assignments of datapoints (interactions and expression profiles) to latent components. From these we can then obtain component membership distributions and clusterings for genes.

Usage

```
ICMg.combined.sampler(L, X, C, alpha=10, beta=0.01, pm0=0, V0=1, V=0.1, B.num=8,
```

Arguments

`L`	N x 2 matrix of link endpoints (N = number of links).
`X`	M x D matrix of gene expression profiles (M = number of nodes, D = number of observations).
`C`	Number of components.
`alpha`	Hyperparameter describing the global distribution over components, larger alpha gives a more uniform distribution.
`beta`	Hyperparameter describing the component-wise distributions over nodes, larger beta gives a more uniform distribution.
`pm0`	Hyperparameter describing the prior mean of the expression profiles, should be zero.
`V0`	Hyperparameter describing the variation of the component-wise expression profiles means around pm0.
`V`	Hyperparameter describing the variation of gene-specific expression profiles around the component-wise means.
`B.num`	Number of burnin rounds.*
`B.size`	Size of one burnin round.*
`S.num`	Number of sample rounds.*
`S.size`	Size of one sample round.*
`C.boost`	Set to 1 to use faster iteration with C, set to 0 to use slower R functions.

Details

One run consists of two parts, during burnin the sampler is expected to mix, after which the samples are taken. Information about convergence (convN and convL are estimates of convergence for link and node sampling, respectively) and component sizes are printed after each burnin/sample round. For example: B.num=8, B.size=100, S.num=20, S.size=10, runs 800 burnin iterations in 8 rounds and then takes 20 samples with an interval of 10 iterations.

Value

Returns samples as a list:

`z`	S.num x N matrix of samples of component assignments for links.
`w`	S.num x M matrix of samples of component assignments for gene expression profiles.
`convl`	Vector of length (B.num + S.num) with convergence estimator values for link sampling.
`convn`	Vector of length (B.num + S.num) with convergence estimator values for node sampling.
`countsl`	(B.num + S.num) x C matrix of link component sizes.
`countsn`	(B.num + S.num) x C matrix of node component sizes.

additionally all parameters of the run are included in the list.

Author(s)

Juuso Parkkinen

References

Parkkinen, J. and Kaski, S. Searching for functional gene modules with interaction component models. BMC Systems Biology 4 (2010), 4.

See Also

`ICMg.links.sampler`

Examples

```
library(netresponse)
data(osmo) # Load data set

## Run ICMg combined sampler
res = ICMg.combined.sampler(osmo$ppi, osmo$exp, C=10)
```

`ICMg.get.comp.memberships` *ICMg.get.comp.memberships*

Description

Function for computing the component memberships for each data point from the MCMC samples.

Usage

```
ICMg.get.comp.memberships(links, samples)
```

Arguments

`links`	N x 2 matrix of link endpoints (N = number of links).
`samples`	Posterior samples, as given by either ICMg.combined.sampler or ICMg.links.sampler.

Value

A matrix containing the component memberships for each data point (node).

Author(s)

Juuso Parkkinen

References

Parkkinen, J. and Kaski, S. Searching for functional gene modules with interaction component models. BMC Systems Biology 4 (2010), 4.

See Also

`ICMg.combined.sampler`, `ICMg.links.sampler`

`ICMg.links.sampler` *ICMg.links.sampler*

Description

ICMg.links.sampler computes samples from the posterior of the assignments of datapoints (interactions) to latent components. From these we can then obtain component membership distributions and clusterings for genes.

Usage

```
ICMg.links.sampler(L, C, alpha=10, beta=0.01,  B.num=8, B.size=100, S.num=20, S.
```

Arguments

L	N x 2 matrix of link endpoints (N = number of links).
C	Number of components.
alpha	Hyperparameter describing the global distribution over components, larger alpha gives a more uniform distribution.
beta	Hyperparameter describing the component-wise distributions over nodes, larger beta gives a more uniform distribution.
B.num	Number of burnin rounds.*
B.size	Size of one burnin round.*
S.num	Number of sample rounds.*
S.size	Size of one sample round.*
C.boost	Set to 1 to use faster iteration with C, set to 0 to use slower R functions.

Details

One run consists of two parts, during burnin the sampler is expected to mix, after which the samples are taken. Information about convergence (convN and convL are estimates of convergence for link and node sampling, respectively) and component sizes are printed after each burnin/sample round. For example: B.num=8, B.size=100, S.num=20, S.size=10, runs 800 burnin iterations in 8 rounds and then takes 20 samples with an interval of 10 iterations.

Value

Returns samples as a list:

z	S.num x N matrix of samples of component assignments for links.
conv	Vector of length (B.num + S.num) with convergence estimator values for link sampling.
counts	(B.num + S.num) x C matrix of link component sizes.

additionally all parameters of the run are included in the list.

Author(s)

Juuso Parkkinen

References

Parkkinen, J. and Kaski, S. Searching for functional gene modules with interaction component models. BMC Systems Biology 4 (2010), 4.

See Also

ICMg.combined.sampler

Examples

```
library(netresponse)
data(osmo) # Load data

## Run ICMg links sampler
res = ICMg.links.sampler(osmo$ppi, C=10)
```

NetResponseModel-class *Class "NetResponseModel"*

Description

A NetResponse model.

Objects from the Class

Returned by `detect.responses` function.

Slots

moves Subnetwork merging history.

last.grouping Subnetworks in the last agglomeration level: feature indices

subnets Subnetworks in the last agglomeration level: feature names

params Input parameters.

datamatrix Original input datamatrix that was used to learn the model.

network Original network that was used to learn the model (after netresponse preprocessing), given in graphNEL format.

models Parameters for the learned subnetwork models.

Methods

[[`signature(x = "NetResponseModel")`: ...

show `signature(x = "NetResponseModel")`: ...

Author(s)

Leo Lahti `<leo.lahti@iki.fi>`

Examples

```
showClass("NetResponseModel")
```

detect.responses *detect.responses*

Description

Main function of the NetResponse algorithm. Detecting network responses across the conditions.

Usage

```
detect.responses(datamatrix, network, initial.responses = 1,
        max.responses = 10, max.subnet.size = 10, verbose =
        TRUE, prior.alpha = 1, prior.alphaKsi = 0.01, prior.betaKsi =
        0.01, update.hyperparams = 0, implicit.noise = 0, vdp.threshold =
        1.0e-5, merging.threshold = 0, ite = Inf,
        information.criterion = "BIC", speedup = TRUE,
        speedup.max.edges = 10)
```

Arguments

`datamatrix`	Matrix of samples x features. For example, gene expression matrix with conditions on the rows, and genes on the columns. The matrix contains same features than the 'network' object, characterizing the network states across the different samples.
`network`	Network describing undirected pairwise interactions between features of 'datamatrix'. The following formats are supported: binary matrix, graphNEL, igraph, graphAM, Matrix, dgCMatrix, dgeMatrix
`initial.responses`	Initial number of components for each subnetwork model. Used to initialize calculations.
`max.responses`	Maximum number of responses for each subnetwork. Can be used to limit the potential number of network states.
`max.subnet.size`	Numeric. Maximum allowed subnetwork size.
`verbose`	Logical. Verbose parameter.
`implicit.noise`	Implicit noise parameter. Add implicit noise to vdp mixture model. Can help to avoid overfitting to local optima, if this appears to be a problem.
`update.hyperparams`	Logical. Indicate whether to update hyperparameters during modeling.
`prior.alpha, prior.alphaKsi, prior.betaKsi`	Prior parameters for Gaussian mixture model that is calculated for each subnetwork (normal-inverse-Gamma prior). alpha tunes the mean; alphaKsi and betaKsi are the shape and scale parameters of the inverse Gamma function, respectively.
`vdp.threshold`	Minimal free energy improvement after which the variational Gaussian mixture algorithm is deemed converged.
`merging.threshold`	Minimal cost value improvement required for merging two subnetworks.
`ite`	Defines maximum number of iterations on posterior update (updatePosterior). Increasing this can potentially lead to more accurate results, but computation may take longer.
`information.criterion`	Information criterion for model selection. Default is BIC (Bayesian Information Criterion); other options include AIC and AICc.
`speedup`	Takes advantage of approximations to PCA, mutual information etc in various places to speed up calculations. Particularly useful with large and densely connected networks and/or large sample size.

speedup.max.edges
: Used if speedup = TRUE. Applies prefiltering of edges for calculating new joint models between subnetwork pairs when potential cost changes (delta) are updated for a newly merged subnetwork and its neighborghs. Empirical mutual information between each such subnetwork pair is calculated based on their first principal components, and joint models will be calculated only for the top candidates up to the number specified by speedup.max.edges. It is expected that the subnetwork pair that will benefit most from joint modeling will be among the top mutual infomation candidates. This way it is possible to avoid calculating exhaustive many models on the network hubs.

Value

NetResponseModel object.

Author(s)

Leo Lahti, Olli-Pekka Huovilainen and Antonio Gusmao. Maintainer: Leo Lahti <leo.lahti@iki.fi>

References

Leo Lahti et al.: Global modeling of transcriptional responses in interaction networks. Bioinformatics (2010). See citation("netresponse") for details.

Examples

```
library(netresponse)
data( toydata )         # Load toy data set
D    <- toydata$emat    # Response matrix (for example, gene expression)
netw <- toydata$netw    # Network

# Run NetReponse algorithm
model <- detect.responses(D, netw, verbose = FALSE)
```

---

dna *Dna damage data set (PPI and expression)*

Description

A combined yeast data set with protein-protein interactions and gene expression (dna damage). Gene expression profiles are transformed into links by computing a Pearson correlation for all pairs of genes and treating all correlations above 0.85 as additional links.

Usage

```
data(dna)
```

Format

List of following objects:

ppi PPI data matrix

exp gene expression profiles data matrix

gids Vector of gene ids corresponding to indices used in data matrices

obs Gene expression observation details

combined.links pooled matrix of PPI and expression links

Details

Number of genes: 1823, number of interactions: 12382, number of gene expression observations: 52, number of total links with PPI and expression links: 15547.

Source

PPI data pooled from yeast data sets of [1] and [2]. Dna damage expression set of [3].

References

Ulitsky, I. and Shamir, R. *Identification of functional modules using network topology and high-throughput data.* BMC Systems Biology 2007, 1:8.

Nariai, N., Kolaczyk, E. D. and Kasif, S. *Probabilistic Protein Function Predition from Heterogenous Genome-Wide Data*. PLoS ONE 2007, 2(3):e337.

Gasch, A., Huang, M., Metzner, S., Botstein, D. and Elledge, S. *Genomic expression responses to DNA-damaging agents and the regulatory role of the yeast ATR homolog Mex1p.* Molecular Biology of the Cell 2001, 12:2987-3003.

Examples

```
data(dna)
```

---

find.similar.features

Find similar features with a given subnetwork.

Description

Given subnetwork, orders the remaining features (genes) in the input data based on similarity with the subnetwork. Allows the identification of similar features that are not directly connected in the input network.

Usage

```
find.similar.features(model, subnet.id, datamatrix = NULL, verbose = FALSE, info
```

Arguments

model	NetResponseModel object.
subnet.id	Investigated subnetwork.
datamatrix	Optional. Can be used to compare subnetwork similarity with new data which was not used for learning the subnetworks.
verbose	Logical indicating whether progress of the algorithm should be indicated on the screen.
information.criterion	Information criterion for model selection. By default uses the same than in the 'model' object.

Details

The same similarity measure is used as when agglomerating the subnetworks: the features are ordered by delta (change) in the cost function, assuming that the feature would be merged in the subnetwork. The smaller the change, the more similar the feature is (change would minimize the new cost function value). Negative values of delta mean that the cost function would be improved by merging the new feature in the subnetwork, indicating features having coordinated response.

Value

A data frame with elements feature.names (e.g. gene IDs) and delta, which indicates similarity level. See details for details. The smaller, the more similar. The data frame is ordered such that the features are listed by decreasing similarity.

Author(s)

Leo Lahti <leo.lahti@iki.fi>

References

See citation("netresponse") for reference details.

Examples

```
data(toydata)
model <- toydata$model
subnet.id <- "Subnet-1"
g <- find.similar.features(model, subnet.id)
# List features that are similar to this subnetwork (delta < 0)
# (ordered by decreasing similarity)
subset(g, delta < 0)
```

---

get.model.parameters *get.model.parameters*

Description

Retrieve the mixture model parameters of the NetResponse algorithm for a given subnetwork.

Usage

```
get.model.parameters(model, subnet.id)
```

Arguments

model	Result from NetResponse (detect.responses function).
subnet.id	Subnet identifier. A natural number which specifies one of the subnetworks within the 'model' object.

Details

Only the non-empty components are returned. Note: the original data matrix needs to be provided for function call separately.

Value

A list with the following elements:

mu	Centroids for the mixture components. Components x nodes.
sd	Standard deviations for the mixture components. A vector over the nodes for each component, implying the diagonal covariance matrix of the model (i.e. diag(std^2)). Components x nodes
w	Vector of component weights.
nodes	List of nodes in the subnetwork.
K	Number of mixture components.

Author(s)

Leo Lahti <leo.lahti@iki.fi>

References

Leo Lahti et al.: Global modeling of transcriptional responses in interaction networks. Bioinformatics (2010). See citation("netresponse") for details.

Examples

```
# Load toy data
data( toydata )          # Load toy data set
D     <- toydata$emat    # Response matrix (for example, gene expression)
model <- toydata$model   # Pre-calculated model

# Get model parameters for a given subnet
# (Gaussian mixture: mean, covariance diagonal, mixture proportions)
get.model.parameters(model, subnet.id = 1)
```

get.subnets *get.subnets*

Description

List the detected subnetworks (each is a list of nodes in the corresponding subnetwork).

Usage

```
get.subnets(model, get.names = TRUE, min.size = 2, max.size = Inf, min.responses
```

Arguments

model	Output from the detect.responses function. An object of NetResponseModel class.
get.names	Logical. Indicate whether to return subnetwork nodes using node names (TRUE) or node indices (FALSE).
min.size, max.size	Numeric. Filter out subnetworks whose size is not within the limits specified here.
min.responses	Numeric. Filter out subnetworks with less responses (mixture components) than specified here.

Value

A list of subnetworks.

Author(s)

Leo Lahti <leo.lahti@iki.fi>

References

Leo Lahti et al.: Global modeling of transcriptional responses in interaction networks. Bioinformatics (2010). See citation("netresponse") for details.

Examples

```
library(netresponse)

# Load a pre-calculated netresponse model obtained with
# model <- detect.responses(toydata$emat, toydata$netw, verbose = FALSE)
data( toydata )
model <- toydata$model

#List the detected subnetworks
#(each is a list of nodes for the given subnetwork):
get.subnets(model)
```

model.stats *model.stats*

Description

Subnetwork statistics: size and number of distinct responses for each subnet.

Usage

```
model.stats( model )
```

Arguments

model	Result from NetResponse (detect.responses function).

Value

A 'subnetworks x properties' data frame containing the following elements.

subnet.size:
: Vector of subnetwork sizes.

subnet.responses:
: Vector giving the number of responses in each subnetwork.

Author(s)

Leo Lahti <leo.lahti@iki.fi>

References

Leo Lahti et al.: Global modeling of transcriptional responses in interaction networks. Bioinformatics (2010). See citation("netresponse") for reference details.

Examples

```
library(netresponse)

# Load a pre-calculated netresponse model obtained with
# model <- detect.responses(toydata$emat, toydata$netw, verbose = FALSE)
data( toydata )
# Calculate summary statistics for the model
stat <- model.stats(toydata$model)
```

netresponse-package *NetResponse: Global modeling of transcriptional responses in*

Description

Global modeling of transcriptional responses in interaction networks.

Details

Package:	netresponse
Type:	Package
Version:	See sessionInfo() or DESCRIPTION file
Date:	2011-02-03
License:	GNU GPL >=2
LazyLoad:	yes

Author(s)

Leo Lahti, Olli-Pekka Huovilainen, Antonio Gusmao and Juuso Parkkinen. Maintainer: Leo Lahti <leo.lahti@iki.fi>

References

Leo Lahti et al.: Global modeling of transcriptional responses in interaction networks. Bioinformatics (2010). See citation("netresponse") for details.

Examples

```
# Load the package
library(netresponse)

# Define parameters for toy data
Ns <- 200  # number of samples (conditions)
Nf <- 10   # number of features (nodes)
feature.names <- paste("feat", seq(Nf), sep="")
sample.names  <- paste("sample", seq(Ns), sep="")

# random seed
set.seed( 123 )

# Random network
netw <- pmax(array(sign(rnorm(Nf^2)), dim = c(Nf, Nf)), 0)
# in pathway analysis nodes correspond to genes
rownames(netw) <- colnames(netw) <- feature.names

# Random responses of the nodes across conditions
D <- array(rnorm(Ns*Nf), dim = c(Ns,Nf), dimnames = list(sample.names, feature.names))
D[1:100, 4:6]  <- t(sapply(1:(Ns/2),function(x){rnorm(3, mean = 1:3)}))
```

```
D[101:Ns, 4:6] <- t(sapply(1:(Ns/2),function(x){rnorm(3, mean = 7:9)}))

# Calculate the model
model <- detect.responses(D, netw)

# Subnets (each is a list of nodes)
get.subnets( model )

# Retrieve model for one subnetwork
# means, standard devations and weights for the components
inds <- which(sapply(model@last.grouping, length) > 2)
subnet.id <- names(model@subnets)[[1]]
m <- get.model.parameters(model, subnet.id)
print(m)
```

`order.responses` *order.responses*

Description

Orders the responses by association strength (enrichment score) to a given factor level.

Usage

```
order.responses(model, sample, method = "hypergeometric")
```

Arguments

`model`	NetResponseModel object.
`sample`	Measure enrichment of this sample (set) across the observed responses.
`method`	'hypergeometric' measures enrichment of factor levels in this response; 'precision' measures response purity for each factor level; 'dependency' measures logarithm of the joint density between response and factor level vs. their marginal densities: log(P(r,s)/(P(r)P(s)))

Value

A data frame with elements 'ordered.responses' which gives a data frame of responses ordered by enrichment score for the investigated sample. The subnetwork, response id and enrichment score are shown. The method field indicates the enrichment calculation method. The sample field lists the samples et for which the enrichments were calculated.

Note

Tools for analyzing end results of the model.

Author(s)

Leo Lahti `<leo.lahti@iki.fi>`

References

See citation("netresponse") for citation details.

Examples

```
# - for given sample/s (factor level), order responses (across all subnets) by associatio
#order.responses(model, sample, method  = "hypergeometric") # overrepresentation
```

`osmo` *Osmoshock data set (PPI and expression)*

Description

A combined yeast data set with protein-protein interactions and gene expression (osmotick shock response). Gene expression profiles are transformed into links by computing a Pearson correlation for all pairs of genes and treating all correlations above 0.85 as additional links.

Usage

```
data(osmo)
```

Format

List of following objects:

ppi PPI data matrix

exp gene expression profiles data matrix

gids Vector of gene ids corresponding to indices used in data matrices

obs Gene expression observation details

combined.links pooled matrix of PPI and expression links

Details

Number of genes: 1711, number of interactions: 10250, number of gene expression observations: 133, number of total links with PPI and expression links: 14256.

Source

PPI data pooled from yeast data sets of [1] and [2]. Dna damage expression set of [3].

References

Ulitsky, I. and Shamir, R. *Identification of functional modules using network topology and high-throughput data.* BMC Systems Biology 2007, 1:8.

Nariai, N., Kolaczyk, E. D. and Kasif, S. *Probabilistic Protein Function Predition from Heterogenous Genome-Wide Data*. PLoS ONE 2007, 2(3):e337.

O'Rourke, S. and Herskowitz, I. *Unique and redundant roles for Hog MAPK pathway components as revealed by whole-genome expression analysis.* Molecular Biology of the Cell 2004, 15:532-42.

Examples

```
data(osmo)
```

read.network *Reading network files*

Description

Function to read network files.

Usage

```
read.sif(sif.file, format = "graphNEL", directed = FALSE)
```

Arguments

sif.file	Name of network file in SIF format.
format	Output format: igraph or graphNEL
directed	Logical. Directed/undirected graph. Not used in the current model.

Details

Read in SIF network file, return R graph object in igraph or graphNEL format.

Value

R graph object in igraph or graphNEL format.

Author(s)

Leo Lahti <leo.lahti@iki.fi>

Examples

```
#net <- read.sif("network.sif")
```

response2sample *response2sample*

Description

List the most strongly associated response of a given subnetwork for each sample.

Usage

```
response2sample(model, subnet.id, component.list = TRUE)
```

Arguments

model	A NetResponseModel object. Result from NetResponse (detect.responses function).
subnet.id	Subnet id. A natural number which specifies one of the subnetworks within the 'model' object.
component.list	List samples separately for each mixture component (TRUE). Else list the most strongly associated component for each sample (FALSE).

Value

A list. Each element corresponds to one subnetwork response, and contains a list of samples that are associated with the response (samples for which this response has the highest probability P(response | sample)).

Author(s)

Leo Lahti <leo.lahti@iki.fi>

References

Leo Lahti et al.: Global modeling of transcriptional responses in interaction networks. Bioinformatics (2010). See citation("netresponse") for citation details.

Examples

```
library( netresponse )

# Load example data
data( toydata )         # Load toy data set
D    <- toydata$emat    # Response matrix (for example, gene expression)
model <- toydata$model  # Pre-calculated model

# Find the samples for each response (for a given subnetwork)
response2sample(model, subnet.id = 1)
```

---

sample2response *sample2response*

Description

Probabilistic sample-response assignments for given subnet.

Usage

```
sample2response(model, subnet.id)
```

Arguments

model	Result from NetResponse (detect.responses function).
subnet.id	Subnet identifier. A natural number which specifies one of the subnetworks within the 'model' object.

Value

A matrix of probabilities. Sample-response assignments for given subnet, listing the probability of each response, given a sample.

Author(s)

Leo Lahti <leo.lahti@iki.fi>

References

Leo Lahti et al.: Global modeling of transcriptional responses in interaction networks. Bioinformatics (2010). See citation("netresponse") for citation details.

Examples

```
#library(netresponse)
#data( toydata )         # Load toy data set
#D    <- toydata$emat    # Response matrix (for example, gene expression)
#netw <- toydata$netw    # Network

# Detect network responses
#model <- detect.responses(D, netw, verbose = FALSE)

# Assign samples to responses (soft, probabilistic assignments sum to 1)
#response.probabilities <- sample2response(model, subnet.id = "Subnet-1")
```

---

toydata *toydata*

Description

Toy data for NetResponse examples.

Usage

```
data(toydata)
```

Format

Toy data: a list with three elements:

emat: Data matrix (samples x features). This contains the same features that are provided in the network (toydata$netw). The matrix characterizes measurements of network states across different conditions.

netw: Binary matrix that describes pairwise interactions between features. This defines an undirected network over the features. A link between two nodes is denoted by 1.

model: A pre-calculated model. Object of NetResponseModel class, resulting from applying the netresponse algorithm on the toydata with model <- detect.responses(D, netw).

References

Leo Lahti et al.: Global modeling of transcriptional responses in interaction networks. Bioinformatics (2010).

Examples

```
data(toydata)
D    <- toydata$emat   # Response matrix (samples x features)
netw <- toydata$netw   # Network between the features
model <- toydata$model # Pre-calculated NetResponseModel obtained with
                       # model <- detect.responses(D, netw)
```

---

vdp.mixt *vdp.mixt*

Description

Accelerated variational Dirichlet process Gaussian mixture.

Usage

```
vdp.mixt(dat, prior.alpha = 1, prior.alphaKsi = 0.01, prior.betaKsi =
0.01, do.sort = TRUE, threshold = 1e-05, initial.K = 1, ite = Inf,
implicit.noise = 0, c.max = 10, speedup = TRUE, min.size = 5)
```

Arguments

dat	Data matrix (samples x features).
prior.alpha, prior.alphaKsi, prior.betaKsi	Prior parameters for Gaussian mixture model (normal-inverse-Gamma prior). alpha tunes the mean; alphaKsi and betaKsi are the shape and scale parameters of the inverse Gamma function, respectively.
do.sort	When true, qOFz will be sorted in decreasing fashion by component size, based on colSums(qOFz). The qOFz matrix describes the sample-component assigments in the mixture model.
threshold	Defines the minimal free energy improvement that stops the algorithm: used to define convergence limit.
initial.K	Initial number of mixture components.

`ite` Defines maximum number of iterations on posterior update (updatePosterior). Increasing this can potentially lead to more accurate results, but computation may take longer.

`implicit.noise` Adds implicit noise; used by vdp.mk.log.lambda.so and vdp.mk.hp.posterior.so. By adding noise (positive values), one can avoid overfitting to local optima in some cases, if this happens to be a problem.

`c.max` Maximum number of candidates to consider in find.best.splitting. During mixture model calculations new mixture components can be created until this upper limit has been reached. Defines the level of truncation for a truncated stick-breaking process.

`speedup` When learning the number of components, each component is splitted based on its first PCA component. To speed up, approximate by using only subset of data to calculate PCA.

`min.size` Minimum size for a component required for potential splitting during mixture estimation.

Details

Implementation of the Accelerated variational Dirichlet process Gaussian mixture model algorithm by Kenichi Kurihara et al., 2007.

Value

`prior` Prior parameters of the vdp-gm model.

`posterior` Posterior estimates for the model parameters and statistics.

weights: Mixture proportions, or weights, for the Gaussian mixture components.

centroids: Centroids of the mixture components.

sds: Standard deviations for the mixture model components (posterior modes of the covariance diagonals square root). Calculated as sqrt(invgam.scale/(invgam.shape + 1)).

qOFz: Sample-to-cluster assigments (soft probabilistic associations).

Nc: Component sizes

invgam.shape: Shape parameter (alpha) of the inverse Gamma distribution

invgam.scale: Scale parameter (beta) of the inverse Gamma distribution

Nparams: Number of model parameters

K: Number of components in the mixture model

`opts` Model parameters that were used.

`free.energy` Free energy of the model.

Note

This implementation is based on the Variational Dirichlet Process Gaussian Mixture Model implementation, Copyright (C) 2007 Kenichi Kurihara (all rights reserved) and the Agglomerative Independent Variable Group Analysis package (in Matlab): Copyright (C) 2001-2007 Esa Alhoniemi, Antti Honkela, Krista Lagus, Jeremias Seppa, Harri Valpola, and Paul Wagner.

Author(s)

Maintainer: Leo Lahti <leo.lahti@iki.fi>

References

Kenichi Kurihara, Max Welling and Nikos Vlassis: Accelerated Variational Dirichlet Process Mixtures. In B. Sch\"olkopf and J. Platt and T. Hoffman (eds.), Advances in Neural Information Processing Systems 19, 761–768. MIT Press, Cambridge, MA 2007.

Examples

```
set.seed(123)

# Generate toy data with two Gaussian components
dat <- rbind(array(rnorm(400), dim = c(200,2)) + 5,
             array(rnorm(400), dim = c(200,2)))

# Infinite Gaussian mixture model with
# Variational Dirichlet Process approximation
mixt <- vdp.mixt( dat )

# Centroids of the detected Gaussian components
mixt$posterior$centroids

# Hard mixture component assignments for the samples
apply(mixt$posterior$qOFz, 1, which.max)
```

Index