netresponse

April 20, 2011

`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 = 20, rseed = 123, verbose =
        TRUE, prior.alpha = 1, prior.alphaKsi = 0.01, prior.betaKsi =
        0.01, update.hyperparams = 0, implicit.noise = 0, threshold =
        1.0e-5, ite = Inf)
```

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`	Binary matrix that describes pairwise interactions between the features of 'datamatrix'. This defines a network.
`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.
`rseed`	Numeric. Random seed.
`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.

`threshold`
: Minimal free energy improvement after which the algorithm is deemed converged. Used to define convergence limit.

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

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).

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)
```

`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, level = NULL)
```

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.
level	Agglomeration level to investigate. The agglomerative algorithm grows the subnetworks step-by-step. This option can be used to select a specific step during the learning process. Will be included in the next version.

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).

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 that subnetwork).

Usage

```
subnets <- get.subnets(model, level = NULL, get.names = TRUE, stat = NULL, min.s
```

Arguments

model	Output from the detect.responses function. An object of NetResponseModel class.
level	Agglomeration level to investigate. The agglomerative algorithm grows the subnetworks step-by-step. This option can be used to select a particular step during the learning process. Will be included in the next version.
get.names	Logical. Indicate whether to return subnetwork nodes using node names (TRUE) or node indices (FALSE).
stat	Subnetwork summary statistics. If this is not readily provided through this option (i.e. stat = NULL), it will be calculated. Can speed up the get.subnets function.
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).

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)
```

NetResponseModel-class *Class "NetResponseModel"*

Description

A NetResponse model.

Objects from the Class

Returned by `detect.responses` function.

Slots

moves Subnetwork merging history.

costs Cost function values at the different steps.

rseed Random seed.

last.grouping Subnetworks in the last agglomeration level.

params Parameters.

nodes Node ids from the original modelled data matrix.

samples Sample ids from the original modelled data matrix.

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

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

Methods

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

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

Author(s)

Leo Lahti <leo.lahti@iki.fi>

Examples

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

netresponse-package

NetResponse: Global modeling of transcriptional responses in interaction networks

Description

Global modeling of transcriptional responses in interaction networks.

Details

Package:	netresponse
Type:	Package
Version:	0.99.0
Date:	2010-09-23
License:	GNU GPL >=2
LazyLoad:	yes

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).

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)}))

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

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

# Retrieve model for the subnetwork with lowest cost function value
# means, standard devations and weights for the components
inds <- which(sapply(model@last.grouping, length) > 2)
subnet.id <- names(which.min(model@costs[inds]))
m <- get.model.parameters(model, subnet.id)
print(m)
```

`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, 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).

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)
```

`result.stats` *result.stats*

Description

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

Usage

```
result.stats( model, level )
```

Arguments

`model`	Result from NetResponse (detect.responses function).
`level`	Agglomeration level to investigate. The agglomerative algorithm grows the subnetworks step-by-step. This option can be used to select a specific step during the learning process. Will be included in the next version.

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, 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).

Examples

```
library(netresponse)

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

# Calculate summary statistics for the model
stat <- result.stats(model)
```

`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, 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).

Examples

```
# Load toy data
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 = 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 = FALSE)
```

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 assignments 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.

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)

Leo Lahti, Olli-Pekka Huovilainen and Antonio Gusmao. 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