Package 'NBLDA'

July 1, 2018

Type Package

Title Negative Binomial Linear Discriminant Analysis

Version 0.99.0

Date 2018-06-25

Description We proposed a package for classification task which uses Negative Binomial distribution within Linear Discriminant Analysis (NBLDA). It is basically an extension of 'PoiClaClu' package to Negative Binomial distribution. The classification algorithms are based on the papers Dong et al. (2016, ISSN: 1471-2105) and Witten, DM (2011, ISSN: 1932-6157) for NBLDA and PLDA respectively. Although PLDA is a sparse algorithm and can be used for variable selection, the algorithm proposed by Dong et. al. is not sparse, hence, it uses all variables in the classifier. Here, we extent Dong et. al.'s algorithm to sparse case by shrinking overdispersion towards 0 (Yu et. al., 2013, ISSN: 1367-4803) and offset parameter towards 1 (as proposed by Witten DM, 2011). We support only the classification task with this version. However, the clustering task will be included with the following versions.

Imports methods, stats, graphics, sSeq

Suggests knitr

Depends ggplot2

License GPL (>= 2)

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Collate 'FindBestTransform.R' 'all_classes.R' 'all_generics.R' 'control.R' 'generateCountData.R' 'getShrinkedDispersions.R' 'helper_functions.R' 'normalize_counts.R' 'package_and_supplementary.R' 'plot.nblda.R' 'predict.nblda.R' 'trainNBLDA.R' 'zzz_methods.R'

NeedsCompilation no

Author Dincer Goksuluk [aut, cre],
Gokmen Zararsiz [aut],
Selcuk Korkmaz [aut],
Ahmet Ergun Karaagaoglu [ths, aut]

Maintainer Dincer Goksuluk <dincer.goksuluk@hacettepe.edu.tr>

Repository CRAN

Date/Publication 2018-07-01 13:31:13 UTC

R topics documented:

`NBLDA-package` *Classifying count data using Poisson/Negative Binomial linear discriminant analysis*

Description

This package applies linear discriminant analysis using Poisson (PLDA) and Negative Binomial (NBLDA) distributions for the classification of count data, such as gene expression data from RNA-sequencing. PLDA algorithms has been proposed by Witten (2011) through an R package `PoiClaClu` which is available at CRAN. Dong et. al. (2016) proposed an extension of PLDA to negative Binomial distribution. However, the algorithm is not provided through an R package. Hence, we develop an R package `NBLDA` to make the proposed algorithm be available through CRAN. Detailed information about mathematical backgrounds can be found in the references given below.

Author(s)

Dincer Goksuluk, Gokmen Zararsiz, Selcuk Korkmaz, A. Ergun Karaagaoglu

---

Maintainers:

Dincer Goksuluk (Correspondence), <dincer.goksuluk@hacettepe.edu.tr>

Gokmen Zararsiz, <gokmenzararsiz@erciyes.edu.tr>

Selcuk Korkmaz, <selcukorkmaz@hotmail.com>

References

Witten, DM (2011). Classification and clustering of sequencing data using a Poisson model. Ann. Appl. Stat. 5(4), 2493–2518. doi:10.1214/11-AOAS493.

Dong, K., Zhao, H., Tong, T., & Wan, X. (2016). NBLDA: negative binomial linear discriminant analysis for RNA-Seq data. BMC Bioinformatics, 17(1), 369. http://doi.org/10.1186/s12859-016-1208-1

See Also

https://CRAN.R-project.org/package=PoiClaClu

Package:	NBLDA
Type:	Package
License:	GPL (>= 2)

---

cervical *Cervical cancer data*

Description

Cervical cancer data measures the expressions of 714 miRNAs of human samples. There are 29 tumor and 29 non-tumor cervical samples and these two groups are treated as two separete classes.

Format

A data frame with 58 observations and 715 variables.

Source

http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2880020/#supplementary-material-sec

References

Witten, D., et al. (2010) Ultra-high throughput sequencing-based small RNA discovery and discrete statistical biomarker analysis in a collection of cervical tumours and matched controls. BMC Biology, 8:58

Examples

```
## Not run:
data(cervical)

## End(Not run)
```

---

control *Accessors for the 'control' slot.*

Description

This slot stores control parameters for training NBLDA model.

Usage

```
## S4 method for signature 'nblda'
control(object)

## S4 method for signature 'nblda_trained'
control(object)
```

Arguments

object	an nblda or nblda_trained object.

See Also

trainNBLDA

Examples

```
set.seed(2128)
counts <- generateCountData(n = 20, p = 10, K = 2, param = 1, sdsignal = 0.5, DE = 0.8,
                            allZero.rm = FALSE, tag.samples = TRUE)
x <- t(counts$x + 1)
y <- counts$y
xte <- t(counts$xte + 1)
ctrl <- nbldaControl(folds = 2, repeats = 2)

fit <- trainNBLDA(x = x, y = y, type = "mle", tuneLength = 10,
                  metric = "accuracy", train.control = ctrl)

control(fit)
```

FindBestTransform *Find the Power Transformation Parameter.*

Description

Use this function to find a constant value of alpha to be used for transforming count data. The power transformation parameter alpha, which approximately fits transformed data to Poisson log linear model, is selected using a grid search within the interval [0, 1].

Usage

```
FindBestTransform(x, grid.length = 50)
```

Arguments

x	a n-by-p data frame or matrix of count data. Samples should be in the rows.
grid.length	how many distinct points of alpha should be searched within the interval [0, 1]? Default is 50.

Value

the value of alpha to be use within power transformation.

Note

This function is copied from PoiClaClu package and modified to control the total number of grid search.

Author(s)

Dincer Goksuluk

Examples

```
set.seed(2128)
counts <- generateCountData(n = 20, p = 10, K = 2, param = 1, sdsignal = 0.5, DE = 0.8,
                            allZero.rm = FALSE, tag.samples = TRUE)

x <- counts$x
FindBestTransform(x)
```

`generateCountData` *Generate Count Data*

Description

This function can be used to generate counts, e.g RNA-Sequencing data, for both classification and clustering purposes.

Usage

```
generateCountData(n, p, K, param, sdsignal = 1, DE = 0.3,
  allZero.rm = TRUE, tag.samples = FALSE)
```

Arguments

`n`	number of samples.
`p`	number of variables/features.
`K`	number of classes.
`param`	overdispersion parameter. This parameter is matched with the arguement `size` in `rnbinom` function. Hence, Negative Binomial distribution aproximates to Poisson distribution as `param` increases.
`sdsignal`	a nonzero numeric value. As `sdsignal` increases, the observed counts greatly differs among K classes.
`DE`	a numeric value within the interval [0, 1]. This is the proportion of total number of variables that is significantly different among K classes. The remaining part is assumed to be having no contribution to discrimination function.
`allZero.rm`	a logical. If TRUE, columns having all zero cells are dropped.
`tag.samples`	a logical. If TRUE, rownames are automatically generated. A tag for each sample such as "S1", "S2", etc.

Value

`x, xte`	count data matrix for training and test set.
`y, yte`	class labels for training and test set.
`truesf, truesfte`	true size factors for training and test set. See Witten (2011) for more information on estimating size factors.

Author(s)

Dincer Goksuluk

Examples

```
set.seed(2128)
counts <- generateCountData(n = 20, p = 10, K = 2, param = 1, sdsignal = 0.5, DE = 0.8,
                            allZero.rm = FALSE, tag.samples = TRUE)
head(counts$x)
```

`getShrinkedDispersions`

Estimate Shrinked Overdispersions

Description

Use this function to shrink initial estimates of overdispersions towards a target value.

Usage

```
getShrinkedDispersions(obs, shrinkTarget = NULL, delta = NULL)
```

Arguments

`obs`	a numeric vector. Initial dispersion estimates for each feature.
`shrinkTarget`	a numeric value. initial dispersion estimates are shrinked towards this value. If NULL, target value is estimated from initial dispersion estimates. See notes.
`delta`	a numeric value. This is the weight that is used for shrinkage algorithm. If 0, no shrinkage is performed on intiial values. If equals 1, initial values are forced to shrinked to target value. If NULL, weight are automatically estimated from initial disperson estimates.

Value

a list with elements of initial and adjusted (shrinked) dispersion estimates, shrinkage target and weight that is used to shrink towards target value. See related paper for detailed information on shrinkage algorithm (Yu et. al., 2013).

`initial`	initial estimates for dispersions estimated from method-of-momnets.
`adj`	shrinked dispersion estimates.
`cmp`	mean and variance of initial estimates.
`delta`	a weight used for shrinkage estimates. See Yu et. al. (2013) for details.
`target`	shrinkage target for initial dispersion estimates.

Note

This function is modified using source code from `getAdjustDisp`.

Author(s)

Dincer Goksuluk

References

Yu, D., Huber, W., & Vitek, O. (2013). Shrinkage estimation of dispersion in Negative Binomial models for RNA-seq experiments with small sample size. Bioinformatics, 29(10), 1275-1282.

See Also

getT, getAdjustDisp

Examples

```
set.seed(2128)
initial <- runif(10, 0, 4)

getShrinkedDispersions(initial, 0)  # shrink towards 0.
getShrinkedDispersions(initial, 0, delta = 1)  # force to shrink 0.
```

---

inputs *Accessors for the 'input' slot.*

Description

This slot stores the input data for trained model.

Usage

```
## S4 method for signature 'nblda'
inputs(object)
```

Arguments

object	an nblda object.

See Also

trainNBLDA

Examples

```
set.seed(2128)
counts <- generateCountData(n = 20, p = 10, K = 2, param = 1, sdsignal = 0.5, DE = 0.8,
                            allZero.rm = FALSE, tag.samples = TRUE)
x <- t(counts$x + 1)
y <- counts$y
xte <- t(counts$xte + 1)
ctrl <- nbldaControl(folds = 2, repeats = 2)

fit <- trainNBLDA(x = x, y = y, type = "mle", tuneLength = 10,
                  metric = "accuracy", train.control = ctrl)

inputs(fit)
```

---

nblda-class nblda *object*

Description

This object is the main class for NBLDA package. It stores inputs, results and call info for the trained model.

Details

Objects can be created by calls of the form `new("nblda", ...)`. This type of objects is created as a result of `trainNBLDA` function of `NBLDA` package. It is then used in `predict` function for predicting class labels of new samples.

Slots

`input`: an `nblda_input` object including the count matrix (or data.frame) and class labels.

`result`: an `nblda_trained` object with elements from corss-validated and final models.

`call`: a call expression.

Author(s)

Dincer Goksuluk

See Also

`nblda_trained`, `nblda_input`

nbldaControl *Control parameters for trained NBLDA model.*

Description

Define control parameters to be used within trainNBLDA function.

Usage

```
nbldaControl(folds = 5, repeats = 2, foldIdx = NULL, rhos = NULL,
  beta = 1, prior = NULL, transform = FALSE, alpha = NULL,
  truephi = NULL, target = 0, phi.epsilon = 0.15,
  normalize.target = FALSE, delta = NULL, multicore = FALSE, ...)
```

Arguments

folds	A positive integer. The number of folds for k-fold model validation.
repeats	A positive integer. This is the number of repeats for k-fold model validation. If NULL, 0 or negative, it is set to 1.
foldIdx	a list with indices of hold-out samples for each fold. It should be a list where folds are nested within repeats. If NULL, folds and repeats are used to define hold-out samples.
rhos	A vector of tuning parameters that control the amount of soft thresholding performed. If NULL, it is automatically generated within trainNBLDA using tuneLength, i.e. the length of grid search. See details.
beta	A smoothing term. A Gamma(beta,beta) prior is used to fit the Poisson model. Recommendation is to just leave it at 1, the default value. See Witten (2011) and Dong et. al. (2016) for details.
prior	A vector of length equal to the number of classes indicating the prior class probabilities. If NULL, all classes are assumed to be equally distributed.
transform	a logical. If TRUE, count data is transformed using power transformation. If alpha is not specified the power transformation parameter is automatically calculated using goodness-of-fit test. See Witten (2011) for details.
alpha	a numeric value within [0, 1] to be used for power transformation.
truephi	a vector of length equal to the number of variables representing the true overdispersion parameters for each variable. If a single value is given, it is replicated for all variables. If a vector of length unequal to the number of variables is given, the first element of this vector is used and replicated for all variables. If NULL, estimated overdispersions are used in the classifier. See details.
target	a value for the shrinkage target of dispersion estimates. If target is NULL, then then a value that is small and minimizes the average squared difference is automatically used as the target value. See getT for details.
phi.epsilon	a positive value for controlling the number of features whose dispersions are shrinked towards 0. See details.

`normalize.target`
: a logical. If TRUE and `target` is NULL, the target value is estimated using normalized dispersion estimates. See `getT` for details.

`delta`
: a weight within the interval [0, 1] that is used while shrinking dispersions towards 0. When "delta = 0", initial dispersion estimates are forced to be shrinked to 1. Similarly, if "delta = 0", no shrinkage is performed on initial estimates.

`multicore`
: a logical. If a parallel backend is loaded and available, the function runs in parallel CPUs.

`...`
: further arguements passed to `trainNBLDA`.

Details

`rhos` is used to control the level of sparsity, i.e. the number of variables (or features) used in classifier. If a variable has no contribution to discrimination function, it should be removed from the model. By setting rhos within the interval [0, Inf], it is possible control the amount of variables that is removed from the model. As the upper bound of rhos decreases towards 0, fewer variables are removed. If rhos = 0, all variables are included in classifier.

`truephi` controls how Poisson model differs from Negative Binomial model. If overdispersion is zero, Negative Binomial model converges to Poisson model. Hence, the results from `trainNBLDA` is identical to PLDA results from `Classify` when truephi = 0.

`phi.epsilon` is a value used to shring estimated overdispersions towards 0. Poisson model assumes that there is no overdispersion in the observed counts. However, this is not a valid assumption in highly overdispersed count data. `NBLDA` performs a shrinkage on estimated overdispersions. Although the amount of shrinkage is dependent on several parameters such as `delta`, `target` and `truephi`, some of the shrinked overdispersions might be very close to 0. By defining a threshold value for shrinked overdispersions, it is possible to shrink very small overdispersions towards 0. If estimated overdispersion is below `phi.epsilon`, it is shrinked to 0. If `phi.epsilon` = NULL, threshold value is set to 0. Hence, all the variables with very small overdispersion are included in the NBLDA model.

Value

a list with all the control elements.

Author(s)

Dincer Goksuluk

References

Witten, DM (2011). Classification and clustering of sequencing data using a Poisson model. Ann. Appl. Stat. 5(4), 2493–2518. doi:10.1214/11-AOAS493.

Dong, K., Zhao, H., Tong, T., & Wan, X. (2016). NBLDA: negative binomial linear discriminant analysis for RNA-Seq data. BMC Bioinformatics, 17(1), 369. http://doi.org/10.1186/s12859-016-1208-1.

Yu, D., Huber, W., & Vitek, O. (2013). Shrinkage estimation of dispersion in Negative Binomial models for RNA-seq experiments with small sample size. Bioinformatics, 29(10), 1275-1282.

See Also

getT, getAdjustDisp

Examples

```
nbldaControl()  # return default control parameters.
```

---

nbldaTrained *Accessors for the 'crossValidated' slot.*

Description

This slot stores the results for cross-validated model, e.g tuning results, optimum model parameters etc.

Usage

```
## S4 method for signature 'nblda'
nbldaTrained(object)

## S4 method for signature 'nblda_trained'
nbldaTrained(object)
```

Arguments

object	an nblda or nblda_trained object.

See Also

trainNBLDA

Examples

```
set.seed(2128)
counts <- generateCountData(n = 20, p = 10, K = 2, param = 1, sdsignal = 0.5, DE = 0.8,
                            allZero.rm = FALSE, tag.samples = TRUE)
x <- t(counts$x + 1)
y <- counts$y
xte <- t(counts$xte + 1)
ctrl <- nbldaControl(folds = 2, repeats = 2)

fit <- trainNBLDA(x = x, y = y, type = "mle", tuneLength = 10,
                  metric = "accuracy", train.control = ctrl)

nbldaTrained(fit)
```

nblda_input-class nblda_input *object*

Description

This object is the subclass for NBLDA package. It stores input objects, i.e. count data and class labels.

Slots

`x:` a data.frame or matrix. Count data input for NBLDA classifier.

`y:` a vector of length equal to number of rows of x. This is the class labels of each subject. Should be either a numeric vector or factor.

Author(s)

Dincer Goksuluk

nblda_trained-class nblda_trained *object*

Description

This object is the subclass for NBLDA package. It stores cross-validated results and the final model.

Slots

`crossValidated:` a list. Returns the results from cross-validation.

`finalModel:` a list with elements from final model using optimum model parameters from cross-validated model.

`control:` a list with controlling parameters for fitting NBLDA classifier.

Author(s)

Dincer Goksuluk

normalization *Accessors for the 'type' slot.*

Description

This slot stores the name of normalization method. Normalization is defined using type argument in trainNBLDA function.

Usage

```
## S4 method for signature 'nblda'
normalization(object)

## S4 method for signature 'nblda_trained'
normalization(object)
```

Arguments

object	an nblda or nblda_trained object.

See Also

trainNBLDA

Examples

```
set.seed(2128)
counts <- generateCountData(n = 20, p = 10, K = 2, param = 1, sdsignal = 0.5, DE = 0.8,
                            allZero.rm = FALSE, tag.samples = TRUE)
x <- t(counts$x + 1)
y <- counts$y
xte <- t(counts$xte + 1)
ctrl <- nbldaControl(folds = 2, repeats = 2)

fit <- trainNBLDA(x = x, y = y, type = "mle", tuneLength = 10,
                  metric = "accuracy", train.control = ctrl)

normalization(fit)
```

NullModel *Calculate Normalized Counts and Related Training Parameters.*

Description

Fit training set to NBLDA model and estimate normalized counts. The related model parameters which are used while normalizing training sets are also returned in order to normalize test sets using training set parameters.

Usage

```
NullModel(x, type = c("mle", "deseq", "quantile", "none", "tmm"))

NullModelTest(null.out, xte = NULL)
```

Arguments

x	a n-by-p data frame or matrix of count data. Samples should be in the rows.
type	normalization methods. See control for details.
null.out	an object returned from NullModel.
xte	a n-by-p count matrix or data frame of test set. These counts are normalized using training set parameters.

Value

a list with normalized counts and training set parameters used for normalizing raw counts.

Note

These functions are copied from PoiClaClu package and modified here to make "tmm" and "none" methods available.

Author(s)

Dincer Goksuluk

Examples

```
set.seed(2128)
counts <- generateCountData(n = 20, p = 10, K = 2, param = 1, sdsignal = 0.5, DE = 0.8,
                            allZero.rm = FALSE, tag.samples = TRUE)
x <- counts$x
xte <- counts$xte

x.out <- NullModel(x, "mle")
x.out$n ## Normalized counts using "mle" method

xte.out <- NullModelTest(x.out, xte)
```

```
xte.out$n  # Normalized counts for test set using train set parameters.
```

---

plot *Plot Method for the* nblda *and* nblda_trained *Classes*

Description

This function is used to generate model performance plots using ggplot2 functions.

Usage

```
## S3 method for class 'nblda'
plot(x, y, ..., theme = c("nblda", "default"),
  metric = c("accuracy", "error", "sparsity"), return = c("plot", "aes"))

## S3 method for class 'nblda_trained'
plot(x, y, ..., theme = c("nblda", "default"),
  metric = c("accuracy", "error", "sparsity"), return = c("plot", "aes"))

## S4 method for signature 'nblda'
plot(x, y, ..., theme = c("nblda", "default"),
  metric = c("accuracy", "error", "sparsity"), return = c("plot", "aes"))

## S4 method for signature 'nblda_trained'
plot(x, y, ..., theme = c("nblda", "default"),
  metric = c("accuracy", "error", "sparsity"), return = c("plot", "aes"))
```

Arguments

x	a nblda object returned from trainNBLDA or nblda_trained object returned from nbldaTrained.
y	same as x and not required to be defined. If x is missing or NULL, nblda or nblda_trained object is imported from y.
...	further arguments to be passed to plotting function ggplot.
theme	pre-defined plot themes. It can be defined outside plot function using ggplot's library. See examples.
metric	which metric should be used in y-axis?
return	should complete plot or a ggplot object from ggplot be returned? One may select "aes" in order to add plot layers to returned ggplot aesthetics. See examples.

Value

A list of class ggplot.

Author(s)

Dincer Goksuluk

See Also

ggplot

Examples

```
set.seed(2128)
counts <- generateCountData(n = 20, p = 10, K = 2, param = 1, sdsignal = 0.5,
                            DE = 0.8, allZero.rm = FALSE, tag.samples = TRUE)
x <- t(counts$x + 1)
y <- counts$y
xte <- t(counts$xte + 1)
ctrl <- nbldaControl(folds = 2, repeats = 2)

fit <- trainNBLDA(x = x, y = y, type = "mle", tuneLength = 10,
                  metric = "accuracy", train.control = ctrl)

plot(fit)

# Use pre-defined theme
plot(fit, theme = "nblda")

# Externally defining plot theme
plot(fit, theme = "default") + theme_dark(base_size = 14)

# Return empty ggplot object and add layers.
plot(fit, theme = "nblda", return = "aes") +
  geom_point() + geom_line(linetype = 2)
```

---

predict *Extract predictions from NBLDA model*

Description

This function predicts the class labels of test data for a given model.

Usage

```
## S3 method for class 'nblda'
predict(object, test.data, return = c("predictions",
  "everything"), ...)

## S4 method for signature 'nblda'
predict(object, test.data, return = c("predictions",
  "everything"), ...)
```

Arguments

object	a nblda object returned from trainNBLDA.
test.data	a data frame or matrix whose class labels to be predicted.
return	what should be returned? Predicted class labels or eveything?
...	further arguments to be passed to or from methods.

Value

It is possible to return only predicted class labels or a list with elements which are used within prediction process. These arguements are as follows:

xte	count data for test set.
nste	normalized count data for test set.
ds	estimates of offset parameter for each variable. See notes.
discriminant	discriminant scores of each subject.
prior	prior probabilities for each class.
ytehat	predicted class labels for test set.
alpha	power transformation parameter. If no transformation is requested, it returns NULL.
type	normalization method.
dispersions	dispersion estimates of each variable.

Note

d_kj is simply used to re-parameterize the Negative Binomial mean as s_i*g_j*d_kj where s_i is the size factor for subject i, g_j is the total count of variable j and d_kj is the offset parameter for variable j at class k.

Author(s)

Dincer Goksuluk

Examples

```
set.seed(2128)
counts <- generateCountData(n = 20, p = 10, K = 2, param = 1, sdsignal = 0.5, DE = 0.8,
                            allZero.rm = FALSE, tag.samples = TRUE)
x <- t(counts$x + 1)
y <- counts$y
xte <- t(counts$xte + 1)
ctrl <- nbldaControl(folds = 2, repeats = 2)

fit <- trainNBLDA(x = x, y = y, type = "mle", tuneLength = 10,
                  metric = "accuracy", train.control = ctrl)

predict(fit, xte)
```

selectedFeatures *Accessors for the 'selectedFeatures' slot.*

Description

This slot, if not NULL, stores the selected features/variables for sparse model.

Usage

```
## S4 method for signature 'nblda'
selectedFeatures(object)

## S4 method for signature 'nblda_trained'
selectedFeatures(object)
```

Arguments

object	an nblda or nblda_trained object.

Value

a list of selected features info including the followings:

idx	column indices of selected features/variables
names	column names of selected features/variables if input data have pre-defined column names.

Note

If return.selected.features = FALSE within nbldaControl or all features/variables are selected and used in discrimination function, idx and names are returned NULL.

See Also

trainNBLDA, nblda, nblda_trained

Examples

```
set.seed(2128)
counts <- generateCountData(n = 20, p = 50, K = 2, param = 1, sdsignal = 0.5, DE = 0.6,
                            allZero.rm = FALSE, tag.samples = TRUE)
x <- t(counts$x + 1)
y <- counts$y
xte <- t(counts$xte + 1)
ctrl <- nbldaControl(folds = 2, repeats = 2, return.selected.features = TRUE,
                     transform = TRUE, phi.epsilon = 0.10)

fit <- trainNBLDA(x = x, y = y, type = "mle", tuneLength = 10,
                  metric = "accuracy", train.control = ctrl)
```

```
selectedFeatures(fit)
```

---

show *Show Method for the S4 classes in NBLDA Package*

Description

Pretty print the objects in S4 classes on R console.

Usage

```
## S3 method for class 'nblda'
show(object)

## S4 method for signature 'nblda'
show(object)

## S3 method for class 'nblda_trained'
show(object)

## S4 method for signature 'nblda_trained'
show(object)

## S3 method for class 'nblda_input'
show(object)

## S4 method for signature 'nblda_input'
show(object)
```

Arguments

`object`	an object of class `nblda, nblda_trained` and `nblda_input` to be printed.

Author(s)

Dincer Goksuluk

Examples

```
set.seed(2128)
counts <- generateCountData(n = 20, p = 10, K = 2, param = 1, sdsignal = 0.5, DE = 0.8,
                            allZero.rm = FALSE, tag.samples = TRUE)
x <- t(counts$x + 1)
y <- counts$y
xte <- t(counts$xte + 1)
ctrl <- nbldaControl(folds = 2, repeats = 2)
```

```
fit <- trainNBLDA(x = x, y = y, type = "mle", tuneLength = 10,
                  metric = "accuracy", train.control = ctrl)

show(fit)
show(inputs(fit))
show(nbldaTrained(fit))
```

`trainNBLDA` *Train Model over Different Tuning Parameters*

Description

This function fits Negative Binomial classifier using various model parameters and finds the best model parameter using the resampling based performance measures.

Usage

```
trainNBLDA(x, y, type = c("mle", "deseq", "quantile", "tmm"),
  tuneLength = 10, metric = c("accuracy", "error"),
  train.control = nbldaControl(), ...)
```

Arguments

`x`	a n-by-p data frame or matrix. Samples should be in the rows and variables in the columns. Used to train the classifier.
`y`	a vector of length n. Each element corresponds to a class label of a sample. Integer and/or factor types are allowed.
`type`	a character string indicating the type of normalization method within the NBLDA model. See details.
`tuneLength`	a positive integer. This is the total number of levels to be used while tuning the model parameter(s).
`metric`	which criteria should be used while determining the best parameter? overall accuracu or avarage number of misclassified samples?
`train.control`	a list with control parameters to be used in NBLDA model. See nbldaControl for details.
`...`	further arguments. Deprecated.

Details

NBLDA is proposed to classify count data from any field, e.g. economics, social sciences, genomics, etc. In RNA-Seq studies, for example, normalization is used to adjust between-sample differences for downstream analysis. `type` is used to define normalization method. Available options are "mle", "deseq", "quantile" and "tmm". Since "deseq", "quantile" and "tmm" methods are originally proposed as robust methods to be used in RNA-Sequencing studies, one should carefully

define normalization types. In greater details, "deseq" estimates the size factors by dividing each sample by the geometric means of the transcript counts (Anders and Huber, 2010). "tmm" trims the lower and upper side of the data by log fold changes to minimize the log-fold changes between the samples and by absolute intensity (Robinson and Oshlack, 2010). "quantile" is quantile normalization approach of Bullard et al (2010). "mle" (less robust) divides total counts of each sample to the grand total counts (Witten, 2010). See related papers for mathematical backgrounds.

Value

an `nblda` object with following slots:

`input`	an `nblda_input` object including the raw count data and response variable. See `nblda_input` for details.
`result`	an `nblda_trained` object including the results from cross-validated and final models. See `nblda_trained` for details.
`call`	a call expression.

Author(s)

Dincer Goksuluk

References

Witten, DM (2011). Classification and clustering of sequencing data using a Poisson model. Ann. Appl. Stat. 5(4), 2493–2518. doi:10.1214/11-AOAS493.

Dong, K., Zhao, H., Tong, T., & Wan, X. (2016). NBLDA: negative binomial linear discriminant analysis for RNA-Seq data. BMC Bioinformatics, 17(1), 369. http://doi.org/10.1186/s12859-016-1208-1.

Anders S. Huber W. (2010). Differential expression analysis for sequence count data. Genome Biology, 11:R106

Witten D. et al. (2010) Ultra-high throughput sequencing-based small RNA discovery and discrete statistical biomarker analysis in a collection of cervical tumours and matched controls. BMC Biology, 8:58

Robinson MD, Oshlack A (2010). A scaling normalization method for differential expression analysis of RNA-Seq data. Genome Biology, 11:R25, doi:10.1186/gb-2010-11-3-r25

Examples

```
set.seed(2128)
counts <- generateCountData(n = 20, p = 10, K = 2, param = 1, sdsignal = 0.5, DE = 0.8,
                            allZero.rm = FALSE, tag.samples = TRUE)
x <- t(counts$x + 1)
y <- counts$y
xte <- t(counts$xte + 1)
ctrl <- nbldaControl(folds = 2, repeats = 2)

fit <- trainNBLDA(x = x, y = y, type = "mle", tuneLength = 10,
                  metric = "accuracy", train.control = ctrl)
```

```
fit
nbldaTrained(fit)  # Cross-validated model summary.
```

Index