CMA

April 19, 2010

best

Show best hyperparameter settings

Description

In this package hyperparameter tuning is performed by an inner cross-validation step for each learningset. A grid of values is tried and evaluated in terms of the misclassification rate, the results are saved in an object of class tuningresult. This method displays (separately for each learningset) the hyperparameter/ hyperparameter combination that showed the best results. Note that this must not be unique; in this case, only one combination is displayed.

Usage

best(object, ...)

Arguments

| object | An object of class tuningresult. |
|--------|----------------------------------|
| | Currently unused argument. |

Value

A list with elements equal to the number of different learningsets. Each element contains the best hyperparameter combination and the corresponding misclassification rate.

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

See Also

tune

boxplot

Description

This method displays the slot scores of performance scores of an object of class evaloutput.

Arguments

| Х | An object of class evaloutput. |
|---|--|
| | Further graphical parameters passed to the classical ${\tt boxplot}$ function. |

Value

The only return is a boxplot.

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

References

Slawski, M. Daumer, M. Boulesteix, A.-L. (2008) CMA - A comprehensive Bioconductor package for supervised classification with high dimensional data. *BMC Bioinformatics 9: 439*

See Also

evaluation

```
classification-methods
```

General method for classification with various methods

Description

Perform classification for the following signatures:

Methods

X = "matrix", y = "numeric", f = "missing" signature 1

X = "matrix", **y** = "factor", **f** = "missing" signature 2

X = "data.frame", y = "missing", f = "formula" signature 3

X = "ExpressionSet", y = "character", f = "missing" signature 4

For further argument and output information, consult classification.

classification General method for classification with various methods

Description

Most general function in the package, providing an interface to perform variable selection, hyperparameter tuning and classification in one step. Alternatively, the first two steps can be performed separately and can then be plugged into this function.

For S4 method information, s. classification-methods.

Usage

classification(X, y, f, learningsets, genesel, genesellist = list(), nbgene, cla

Arguments

| Х | Gene expression data. Can be one of the following: |
|--------------|--|
| | • A matrix. Rows correspond to observations, columns to variables. |
| | • A data.frame, when f is not missing (s. below). |
| | • An object of class ExpressionSet. |
| У | Class labels. Can be one of the following: |
| | • A numeric vector. |
| | • A factor. |
| | • A character if X is an ExpressionSet that specifies the phenotype variable. |
| | • missing, if X is a data.frame and a proper formula f is provided. |
| | WARNING: The class labels will be re-coded to range from 0 to $K-1$, where K is the total number of different classes in the learning set. |
| f | A two-sided formula, if X is a data.frame. The left part correspond to class labels, the right to variables. |
| learningsets | An object of class learningsets. May be missing, then the complete datasets is used as learning set. |
| genesel | Optional (but usually recommended) object of class genesel containing vari- able importance information for the argument learningsets |
| genesellist | In the case that the argument genesel is missing, this is an argument list passed to GeneSelection. If both genesel and genesellist are missing, no variable selection is performed. |
| nbgene | Number of best genes to be kept for classification, based on either genesel or the call to GeneSelection using genesellist. In the case that both are missing, this argument is not necessary. note : |
| | • If the gene selection method has been one of "lasso", "elasticnet", "boosting", nbgene will be reset to min(s, nbgene) where s is the number of nonzero coefficients. |
| | • if the gene selection scheme has been "one-vs-all", "pairwise" for the multiclass case, there exist several rankings. The top nbgene will be kept of <i>each</i> of them, so the number of effective used genes will sometimes be much larger. |

```
classification
```

| classifier | Name of function ending with CMA indicating the classifier to be used. |
|------------|---|
| tuneres | Analogous to the argument genesel - object of class tuningresult con- taining information about the best hyperparameter choice for the argument learningsets. |
| tuninglist | Analogous to the argument genesellist. In the case that the argument tuneres is missing, this in argument list passed to tune. If both tuneres and tuninglist are missing, no variable selection is performed. warning : Note that if a user-defined hyperparameter grid is passed, this will result in a list within a list: tuninglist = list(grids=list(argname = c()), s. example. warning : Contrary to tune, if tuninglist is an empty list (de- fault), <i>no</i> hyperparameter tuning will be performed at all. To use pre-defined hy- perparameter grids, the argument is tuninglist = list(grids = list()). |
| trace | Should progress be traced ? Default is TRUE. |
| | Further arguments passed to the function classifier. |

Details

For details about hyperparameter tuning, consult tune.

Value

A list of objects of class cloutput and clvarseloutput, respectively; its length equals the number of different learningsets. The single elements of the list can convenienly be combined using the join function. The results can be analyzed and evaluated by various measures using the method evaluation.

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

Christoph Bernau
dernau@ibe.med.uni-muenchen.de>

References

Slawski, M. Daumer, M. Boulesteix, A.-L. (2008) CMA - A comprehensive Bioconductor package for supervised classification with high dimensional data. *BMC Bioinformatics 9: 439*

See Also

```
GeneSelection, tune, evaluation, compBoostCMA, dldaCMA, ElasticNetCMA, fdaCMA, flexdaCMA, gbmCMA, knnCMA, ldaCMA, LassoCMA, nnetCMA, pknnCMA, pls_ldaCMA, pls_lrCMA, pls_rfCMA, pnnCMA, qdaCMA, rfCMA, scdaCMA, shrinkldaCMA, svmCMA
```

Examples

```
### a simple k-nearest neighbour example
### datasets
## Not run: plot(x)
data(golub)
golubY <- golub[,1]
golubX <- as.matrix(golub[,-1])
### learningsets
set.seed(111)
lset <- GenerateLearningsets(y=golubY, method = "CV", fold=5, strat =TRUE)</pre>
```

cloutput-class

```
### 1. GeneSelection
selttest <- GeneSelection(golubX, golubY, learningsets = lset, method = "t.test")</pre>
### 2. tuning
tunek <- tune (golubX, golubY, learningsets = lset, genesel = selttest, nbgene = 20, class
### 3. classification
knn1 <- classification(golubX, golubY, learningsets = lset, genesel = selttest,</pre>
                        tuneres = tunek, nbgene = 20, classifier = knnCMA)
### steps 1.-3. combined into one step:
knn2 <- classification(golubX, golubY, learningsets = lset,</pre>
                        genesellist = list(method = "t.test"), classifier = knnCMA,
                        tuninglist = list(grids = list(k = c(1:8))), nbgene = 20)
### show and analyze results:
knnjoin <- join(knn2)</pre>
show(knn2)
eval <- evaluation(knn2, measure = "misclassification")</pre>
show(eval)
summary (eval)
boxplot (eval)
## End(Not run)
```

cloutput-class "cloutput"

Description

Object returned by one of the classifiers (functions ending with CMA)

Slots

learnind: Vector of indices that indicates which observations where used in the learning set.

y: Actual (true) class labels of predicted observations.

yhat: Predicted class labels by the classifier.

prob: A numeric matrix whose rows equals the number of predicted observations (length of y/yhat) and whose columns equal the number of different classes in the learning set. Rows add up to one. Entry j, k of this matrix contains the probability for the j-th predicted observation to belong to class k. Can be a matrix of NAs, if the classifier used does not provide any probabilities

method: Name of the classifer used.

mode: character, one of "binary" (if the number of classes in the learning set is two) or multiclass (if it is more than two).

Methods

show Use show (cloutput-object) for brief information

- ftable Use ftable(cloutput-object) to obtain a confusion matrix/cross-tabulation of y
 vs. yhat, s. ftable, cloutput-method.
- plot Use plot(cloutput-object) to generate a probability plot of the matrix prob described above, s. plot, cloutput-method
- roc Use roc(cloutput-object) to compute the empirical ROC curve and the Area Under the Curve (AUC) based on the predicted probabilities, s.roc, cloutput-method

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

See Also

```
clvarseloutput compBoostCMA, dldaCMA, ElasticNetCMA, fdaCMA, flexdaCMA,
gbmCMA, knnCMA, ldaCMA, LassoCMA, nnetCMA, pknnCMA, plrCMA, pls_ldaCMA, pls_lrCMA,
pls_rfCMA, pnnCMA, qdaCMA, rfCMA, scdaCMA, shrinkldaCMA, svmCMA
```

clvarseloutput-class "clvarseloutput"

Description

Object returned by all classifiers that can peform variable selection or compute variable importance. These are:

- Random Forest, s. rfCMA,
- Componentwise Boosting, s. compBoostCMA,
- LASSO-logistic regression, s. LassoCMA,
- ElasticNet-logistic regression, s. ElasticNetCMA

. Objects of class clvarseloutput extend both the class cloutuput and varsel, s. below.

Slots

learnind: Vector of indices that indicates which observations where used in the learning set.

- y: Actual (true) class labels of predicted observations.
- yhat: Predicted class labels by the classifier.
- prob: A numeric matrix whose rows equals the number of predicted observations (length of y/yhat) and whose columns equal the number of different classes in the learning set. Rows add up to one. Entry j, k of this matrix contains the probability for the j-th predicted observation to belong to class k. Can be a matrix of NAs, if the classifier used does not provide any probabilities
- method: Name of the classifer used.
- mode: character, one of "binary" (if the number of classes in the learning set is two) or multiclass (if it is more than two).
- varsel: numeric vector of variable importance measures (for Random Forest) or absolute values of regression coefficients (for the other three methods mentionned above) (from which the majority will be zero).

Extends

Class "cloutput", directly. Class "varseloutput", directly.

6

CMA-package

Methods

show Use show (cloutput-object) for brief information

- ftable Use ftable(cloutput-object) to obtain a confusion matrix/cross-tabulation of y
 vs. yhat, s. ftable, cloutput-method.
- plot Use plot(cloutput-object) to generate a probability plot of the matrix prob described above, s. plot, cloutput-method
- roc Use roc(cloutput-object) to compute the empirical ROC curve and the Area Under the Curve (AUC) based on the predicted probabilities, s.roc, cloutput-method

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

See Also

rfCMA, compBoostCMA, LassoCMA, ElasticNetCMA

CMA-package

Synthesis of microarray-based classification

Description

The aim of the package is to provide a user-friendly environment for the evaluation of classification methods using gene expression data. A strong focus is on combined variable selection, hyperparameter tuning, evaluation, visualization and comparison of (up to now) 21 classification methods from three main fields: Discriminant Analysis, Neural Networks and Machine Learning. Although the package has been created with the intention to be used for Microarray data, it can as well be used in various (p > n)-scenarios.

Details

| Package: | CMA |
|----------|--------------------------|
| Type: | Package |
| Version: | 1.3.3 |
| Date: | 2009-9-14 |
| License: | GPL (version 2 or later) |

Most Important Steps for the workflow are:

- 1. Generate evaluation datasets using GenerateLearningsets
- 2. (Optionally): Perform variable selection using GeneSelection
- 3. (Optionally): Peform hyperparameter tuning using tune
- **4.** Perform classification using 1.-3.
- 5. Repeat 2.-4. based on 1. for several methods: compBoostCMA, dldaCMA, ElasticNetCMA, fdaCMA, flexdaCMA, gbmCMA, knnCMA, ldaCMA, LassoCMA, nnetCMA, pknnCMA,

plrCMA,pls_ldaCMA,pls_lrCMA,pls_rfCMA,pnnCMA,qdaCMA,rfCMA,scdaCMA, shrinkldaCMA,svmCMA

6. Evaluate the results from 5. using evaluation and make a comparison by calling compare

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

Christoph Bernau

bernau@ibe.med.uni-muenchen.de>

Maintainer: Christoph Bernau

bernau@ibe.med.uni-muenchen.de>.

References

Slawski, M. Daumer, M. Boulesteix, A.-L. (2008) CMA - A comprehensive Bioconductor package for supervised classification with high dimensional data. *BMC Bioinformatics 9: 439*

compare-methods Compare different classifiers

Description

Compare different classifiers for the following signatures:

Methods

clresultlist = "list" signature 1

For further argument and output information, consult compare

compare

Compare different classifiers

Description

Classifiers can be evaluated separately using the method evaluation. Normally, several classifiers are used for the same dataset and their performance is compared. This comparison procedure is essentially facilitated by this method. For S4 method information, s. compare-methods

Usage

```
compare(clresultlist, measure = c("misclassification", "sensitivity",
   "specificity", "average probability", "brier score", "auc"), aggfun =
   meanrm, plot = FALSE, ...)
```

compare

Arguments

| clresultlist | A list of lists (!) of objects of class cloutput or clvarseloutput. Each inner list is usually returned by classification. Additionally, the different list elements of the outer list should have been created by different classifiers, s. also example below. |
|--------------|---|
| measure | A character vector containing one or more of the elements listed below. By de- fault, all measures are computed, using evaluation with scheme = "iterationwise". Note that "sensitivity", "specificity", "auc" cannot be com- puted for the multiclass case. |
| | "misclassification" The missclassification rate. |
| | "sensitivity" The sensitivity or 1-false negative rate. Can only be com- puted for binary classification. |
| | "specificity" The specificity or 1-false positive rate. Can only be com- puted for binary classification. |
| | "average probability" The average probability assigned to the correct class. Requirement is that the used classifier provides probability estima- tions. The optimum performance is 1. |
| | <pre>"brier score" The Brier Score is generally defined as <sum all<br="" over="">observation i> <sum all="" classes="" k="" over=""> (I(y_i=k)- P(k))^2, with I() denoting the indicator function and P(k) the esti- mated probability for class k. The optimum performance is 0.</sum></sum></pre> |
| | <pre>"auc" The Area under the Curve (AUC) belonging to the empirical ROC curve computed from the estimated probabilities and the true class labels. Can only be computed for binary classification and if "scheme = iterationwise", s. below. S. also roc, cloutput-method.</pre> |
| aggfun | Function that determines how performance among different iterations are aggre- gared. Default is meanrm, which computes the mean using na.rm=T. Other possible choices are quantiles. |
| plot | Should the performance of different classifiers be visualized by a joint boxplot ? Default is FALSE. |
| | Further arguments passed to boxplot in the case that plot = TRUE. |

Value

A data.frame with rows corresponding to the compared classifiers and columns to the performance measures, aggregated by aggfun, s. above.

Note

If more than one measure is computed and plot = TRUE, one separate plot is created for each of them.

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

Christoph Bernau
dernau@ibe.med.uni-muenchen.de>

References

Dudoit, S., Fridlyand, J., Speed, T. P. (2002) Comparison of discrimination methods for the classification of tumors using gene expression data. *Journal of the American Statistical Association 97*, 77-87

Slawski, M. Daumer, M. Boulesteix, A.-L. (2008) CMA - A comprehensive Bioconductor package for supervised classification with high dimensional data. *BMC Bioinformatics 9: 439*

See Also

classification, evaluation

Examples

```
## Not run:
### compare the performance of several discriminant analysis methods
### for the Khan dataset:
data(khan)
khanX <- as.matrix(khan[,-1])</pre>
khanY <- khan[,1]</pre>
set.seed(27611)
fiveCV10iter <- GenerateLearningsets(y=khanY, method = "CV", fold = 5, niter = 2, strat =
### candidate methods: DLDA, LDA, QDA, pls_LDA, sclda
class_dlda <- classification(X = khanX, y=khanY, learningsets = fiveCV10iter, classifier</pre>
### peform GeneSlection for LDA, FDA, QDA (using F-Tests):
genesel_da <- GeneSelection(X=khanX, y=khanY, learningsets = fiveCV10iter, method = "f.te</pre>
###
class_lda <- classification(X = khanX, y=khanY, learningsets = fiveCV10iter, classifier =</pre>
class_qda <- classification(X = khanX, y=khanY, learningsets = fiveCV10iter, classifier =</pre>
### We now make a comparison concerning the performance (sev. measures):
### first, collect in a list:
dalike <- list(class_dlda, class_lda, class_qda)
### use pre-defined compare function:
comparison <- compare(dalike, plot = TRUE, measure = c("misclassification", "brier score"
print(comparison)
```

End(Not run)

compBoostCMA-methods

Componentwise Boosting

Description

Roughly speaking, Boosting combines 'weak learners' in a weighted manner in a stronger ensemble.

'Weak learners' here consist of linear functions in one component (variable), as proposed by Buehlmann and Yu (2003).

It also generates sparsity and can as well be as used for variable selection alone. (s. GeneSelection.)

10

compBoostCMA

Methods

X = "matrix", y = "numeric", f = "missing" signature 1 X = "matrix", y = "factor", f = "missing" signature 2 X = "data.frame", y = "missing", f = "formula" signature 3 X = "ExpressionSet", y = "character", f = "missing" signature 4 For further argument and output information, consult compBoostCMA.

compBoostCMA Componentwise Boosting

Description

Roughly speaking, Boosting combines 'weak learners' in a weighted manner in a stronger ensemble.

'Weak learners' here consist of linear functions in one component (variable), as proposed by Buehlmann and Yu (2003).

It also generates sparsity and can as well be as used for variable selection alone. (s. GeneSelection).

For S4 method information, see compBoostCMA-methods.

Usage

compBoostCMA(X, y, f, learnind, loss = c("binomial", "exp", "quadratic"), mstop

Arguments

| Х | Gene expression data. Can be one of the following: |
|----------|--|
| | • A matrix. Rows correspond to observations, columns to variables. |
| | • A data.frame, when f is not missing (s. below). |
| | • An object of class ExpressionSet. |
| У | Class labels. Can be one of the following: |
| | • A numeric vector. |
| | • A factor. |
| | • A character if X is an ExpressionSet that specifies the phenotype variable. |
| | • missing, if X is a data.frame and a proper formula f is provided. |
| | WARNING : The class labels will be re-coded to range from 0 to $K-1$, where K is the total number of different classes in the learning set. |
| f | A two-sided formula, if X is a data.frame. The left part correspond to class labels, the right to variables. |
| learnind | An index vector specifying the observations that belong to the learning set. May be missing; in that case, the learning set consists of all observations and predictions are made on the learning set. |
| loss | Character specifying the loss function - one of "binomial" (LogitBoost), "exp" (AdaBoost), "quadratic"(L2Boost). |

| mstop | Number of boosting iterations, i.e. number of updates to perform. The default (100) does not necessarily produce good results, therefore usage of tune for this argument is highly recommended. |
|-------|---|
| nu | Shrinkage factor applied to the update steps, defaults to 0.1. In most cases, it suffices to set nu to a very low value and to concentrate on the optimization of mstop. |
| | Currently unused arguments. |

Details

The method is partly based on code from the package mboost from T. Hothorn and P. Buehlmann. The algorithm for the multiclass case is described in Lutz and Buehlmann (2006) as 'rowwise updating'.

Value

An object of class clvarseloutput.

Author(s)

Martin Slawski <ms@cs.uni-sb.de> Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

References

Buelmann, P., Yu, B. (2003).

Boosting with the L2 loss: Regression and Classification.
Journal of the American Statistical Association, 98, 324-339
Buehlmann, P., Hothorn, T.
Boosting: A statistical perspective.
Statistical Science (to appear)
Lutz, R., Buehlmann, P. (2006).
Boosting for high-multivariate responses in high-dimensional linear regression.
Statistica Sinica 16, 471-494.

See Also

```
dldaCMA, ElasticNetCMA, fdaCMA, flexdaCMA, gbmCMA, knnCMA, ldaCMA, LassoCMA, nnetCMA, pknnCMA, plrCMA, pls_ldaCMA, pls_lrCMA, pls_rfCMA, pnnCMA, qdaCMA, rfCMA, scdaCMA, shrinkldaCMA, svmCMA
```

Examples

```
### load Golub AML/ALL data
data(golub)
### extract class labels
golubY <- golub[,1]
### extract gene expression
golubX <- as.matrix(golub[,-1])
### select learningset
ratio <- 2/3</pre>
```

```
set.seed(111)
learnind <- sample(length(golubY), size=floor(ratio*length(golubY)))</pre>
### run componentwise (logit)-boosting (not tuned)
result <- compBoostCMA(X=golubX, y=golubY, learnind=learnind, mstop = 500)</pre>
### show results
show(result)
ftable(result)
plot(result)
### multiclass example:
### load Khan data
data(khan)
### extract class labels
khanY <- khan[,1]</pre>
### extract gene expression
khanX <- as.matrix(khan[,-1])</pre>
### select learningset
set.seed(111)
learnind <- sample(length(khanY), size=floor(ratio*length(khanY)))</pre>
### run componentwise multivariate (logit)-boosting (not tuned)
result <- compBoostCMA(X=khanX, y=khanY, learnind=learnind, mstop = 1000)
### show results
show(result)
ftable(result)
plot(result)
```

dldaCMA-methods Diagonal Discriminant Analysis

Description

Performs a diagonal discriminant analysis under the assumption of a multivariate normal distribution in each classes (with equal, diagonally structured) covariance matrices. The method is also known under the name 'naive Bayes' classifier.

Methods

X = "matrix", y = "numeric", f = "missing" signature 1 X = "matrix", y = "factor", f = "missing" signature 2 X = "data.frame", y = "missing", f = "formula" signature 3 X = "ExpressionSet", y = "character", f = "missing" signature 4 For further argument and output information, consult dldaCMA.

dldaCMA

Diagonal Discriminant Analysis

Description

Performs a diagonal discriminant analysis under the assumption of a multivariate normal distribution in each classes (with equal, diagonally structured) covariance matrices. The method is also known under the name 'naive Bayes' classifier.

For S4 method information, see dldaCMA-methods.

Usage

dldaCMA(X, y, f, learnind, ...)

Arguments

| Х | Gene expression data. Can be one of the following: |
|----------|--|
| | • A matrix. Rows correspond to observations, columns to variables. |
| | • A data.frame, when f is <i>not</i> missing (s. below). |
| | • An object of class ExpressionSet. |
| У | Class labels. Can be one of the following: |
| | • A numeric vector. |
| | • A factor. |
| | • A character if X is an ExpressionSet that specifies the phenotype variable. |
| | • missing, if X is a data.frame and a proper formula f is provided. |
| | WARNING: The class labels will be re-coded to range from 0 to $K-1$, where K is the total number of different classes in the learning set. |
| f | A two-sided formula, if X is a data.frame. The left part correspond to class labels, the right to variables. |
| learnind | An index vector specifying the observations that belong to the learning set. May be missing; in that case, the learning set consists of all observations and predictions are made on the learning set. |
| | Currently unused argument. |

Value

An object of class cloutput.

Note

As opposed to linear or quadratic discriminant analysis, variable selection is not strictly necessary.

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

References

McLachlan, G.J. (1992).

Discriminant Analysis and Statistical Pattern Recognition. Wiley, New York

See Also

```
compBoostCMA,ElasticNetCMA,fdaCMA,flexdaCMA,gbmCMA,knnCMA,ldaCMA,LassoCMA,
nnetCMA,pknnCMA,plrCMA,pls_ldaCMA,pls_lrCMA,pls_rfCMA,pnnCMA,qdaCMA,
rfCMA,scdaCMA,shrinkldaCMA,svmCMA
```

14

ElasticNetCMA-methods

Examples

```
### load Golub AML/ALL data
data(golub)
### extract class labels
golubY <- golub[,1]</pre>
### extract gene expression
golubX <- as.matrix(golub[,-1])</pre>
### select learningset
ratio <- 2/3
set.seed(111)
learnind <- sample(length(golubY), size=floor(ratio*length(golubY)))</pre>
### run DLDA
dldaresult <- dldaCMA(X=golubX, y=golubY, learnind=learnind)</pre>
### show results
show(dldaresult)
ftable(dldaresult)
plot(dldaresult)
### multiclass example:
### load Khan data
data(khan)
### extract class labels
khanY <- khan[,1]</pre>
### extract gene expression
khanX <- as.matrix(khan[,-1])</pre>
### select learningset
set.seed(111)
learnind <- sample(length(khanY), size=floor(ratio*length(khanY)))</pre>
### run LDA
ldaresult <- dldaCMA(X=khanX, y=khanY, learnind=learnind)</pre>
### show results
show(dldaresult)
ftable(dldaresult)
plot (dldaresult)
```

ElasticNetCMA-methods

Classfication and variable selection by the ElasticNet

Description

Zou and Hastie (2004) proposed a combined L1/L2 penalty for regularization and variable selection. The Elastic Net penalty encourages a grouping effect, where strongly correlated predictors tend to be in or out of the model together. The computation is done with the function glmpath from the package of the same name.

Methods

X = "matrix", y = "numeric", f = "missing" signature 1

X = "matrix", y = "factor", f = "missing" signature 2

X = "data.frame", **y** = "missing", **f** = "formula" signature 3

X = "ExpressionSet", y = "character", f = "missing" signature 4

For references, further argument and output information, consult ElasticNetCMA

ElasticNetCMA

Description

Zou and Hastie (2004) proposed a combined L1/L2 penalty for regularization and variable selection. The Elastic Net penalty encourages a grouping effect, where strongly correlated predictors tend to be in or out of the model together. The computation is done with the function glmpath from the package of the same name.

The method can be used for variable selection alone, s. ${\tt GeneSelection}.$

For S4 method information, see ElasticNetCMA-methods.

Usage

ElasticNetCMA(X, y, f, learnind, norm.fraction = 0.1, alpha=0.5, ...)

Arguments

| Х | Gene expression data. Can be one of the following: |
|-------------|--|
| | • A matrix. Rows correspond to observations, columns to variables. |
| | A data.frame, when f is <i>not</i> missing (s. below). |
| | An object of class ExpressionSet. note: by default, the predictors are scaled to have unit variance and zero mean. Can be changed by passing standardize = FALSE via the argument. |
| У | Class labels. Can be one of the following: |
| | • A numeric vector. |
| | • A factor. |
| | • A character if X is an ExpressionSet that specifies the phenotype variable. |
| | • missing, if X is a data.frame and a proper formula f is provided. |
| | WARNING: The class labels will be re-coded to range from 0 to $K-1$, where K is the total number of different classes in the learning set. |
| f | A two-sided formula, if X is a data.frame. The left part correspond to class labels, the right to variables. |
| learnind | An index vector specifying the observations that belong to the learning set. May be missing; in that case, the learning set consists of all observations and predictions are made on the learning set. |
| norm.fract: | - |
| | L1 Shrinkage intensity, expressed as the fraction of the coefficient L1 norm compared to the maximum possible L1 norm (corresponds to fraction = 1). Lower values correspond to higher shrinkage. Note that the default (0.1) need not produce good results, i.e. tuning of this parameter is recommended. |
| alpha | The elasticnet mixing parameter, with $0 < alpha <= 1$. The penalty is defined as $(1-alpha)/2 beta _2^2+alpha beta _1$. |
| | alpha=1 is the lasso penalty; Currently 'alpha<0.01' not reliable, unless you supply your own lambda sequence |
| •••• | Further arguments passed to the function glmpath from the package of the same name. |

ElasticNetCMA

Value

An object of class clvarseloutput.

Note

For a strongly related method, s. LassoCMA. Up to now, this method can only be applied to binary classification.

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

Christoph Bernau

bernau@ibe.med.uni-muenchen.de>

References

Zhou, H., Hastie, T. (2004). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society B*, 67(2),301-320

Young-Park, M., Hastie, T. (2007) L1-regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society B*, *69*(*4*), *659-677*

See Also

```
compBoostCMA, dldaCMA, fdaCMA, flexdaCMA, gbmCMA, knnCMA, ldaCMA, LassoCMA,
nnetCMA, pknnCMA, plrCMA, pls_ldaCMA, pls_lrCMA, pls_rfCMA, pnnCMA, qdaCMA,
rfCMA, scdaCMA, shrinkldaCMA, svmCMA
```

Examples

```
### load Golub AML/ALL data
data(golub)
### extract class labels
golubY <- golub[,1]
### extract gene expression
golubX <- as.matrix(golub[,-1])
### select learningset
ratio <- 2/3
set.seed(111)
learnind <- sample(length(golubY), size=floor(ratio*length(golubY)))
### run ElasticNet - penalized logistic regression (no tuning)
result <- ElasticNetCMA(X=golubX, y=golubY, learnind=learnind, norm.fraction = 0.2, alpha
show(result)
ftable(result)
plot(result)</pre>
```

evaloutput-class "evaloutput"

Description

Object returned by the method evaluation.

Slots

score: A numeric vector of performance scores whose length depends on "scheme", s.below. It
equals the number of iterations (number of different datasets) if "scheme = iterationwise"
and the number of all observations in the complete dataset otherwise. As not necessarily all
observation must be predicted at least one time, score can also contain NAs for those observations not classified at all.

measure: performance measure used, s. evaluation.

scheme: scheme used, s. evaluation

method: name of the classifier that has been evaluated.

Methods

show Use show(evaloutput-object) for brief information.

- summary Use summary(evaloutput-object) to apply the classic summary() function
 to the slot score, s. summary, evaloutput-method
- **boxplot** Use boxplot (evaloutput-object) to display a boxplot of the slot score, s. boxplot, evaloutput-method.
- obsinfo Use obsinfo (evaloutput-object, threshold) to display all observations consistenly correctly or incorrectly classified (depending on the value of the argument threshold), s. obsinfo.

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

See Also

evaluation

evaluation-methods Evaluation of classifiers

Description

Evaluate classifiers for the following signatures:

Methods

clresult = "list" signature 1

For further argument and output information, consult evaluation.

evaluation

Description

The performance of classifiers can be evaluted by six different measures and two different schemes that are described more precisely below.

For S4 method information, s. evaluation-methods.

Usage

Arguments

| clresult | A list of objects of class cloutput or clvarseloutput |
|----------|---|
| cltrain | An object of class cloutput in which the <i>whole</i> dataset was used as learning set. Only used if method = " 0.632 " or method = " 0.632 +" in order to obtain an estimation for the resubstitution error rate. |
| cost | An optional cost matrix used if measure = "misclassification". If it is not specified (default), the cost is the usual indicator loss. Otherwise, entry i, j of cost quantifies the loss when the true class is class $i-1$ and the pre- dicted class is $j-1$, provided the conventional coding $0, \ldots, K-1$ in the case of K classes is used. Usually, the matrix contains only non-negative entries with zeros on the diagonal, but this is not obligatory. Make sure that the dimension of the matrix matches the number of classes. |
| У | A vector containing the true class labels. Only needed if scheme = "classwise". |
| measure | Peformance measure to be used: |
| | "misclassification" The missclassifcation rate. |
| | "sensitivity" The sensitivity or 1-false negative rate. Can only be com- puted for binary classification. |
| | "specificity" The specificity or 1-false positive rate. Can only be com- puted for binary classification. |
| | "average probability" The average probability assigned to the correct class. Requirement is that the used classifier provides probability estima- tions. The optimum performance is 1. |
| | <pre>"brier score" The Brier Score is generally defined as <sum all<br="" over="">observation i> <sum all="" classes="" k="" over=""> (I(y_i=k) - P(k))^2, with I() denoting the indicator function and P(k) the esti- mated probability for class k. The optimum performance is 0.</sum></sum></pre> |
| | <pre>"auc" The Area under the Curve (AUC) belonging to the empirical ROC curve computed from the estimated probabilities and the true class labels. Can only be computed for binary classification and if "scheme = iterationwise", s. below. S. also roc, cloutput-method.</pre> |
| | "0.632" The 0.632 estimator (s. reference) for the misclassification rate (applied iteration- or) observationwise, if bootstrap learning sets have been used. Note that cltrain must be provided. |

| | "0.632+" The 0.632+ estimator (s. reference) for the misclassification rate (applied iteration- or) observationwise, if bootstrap learning sets have been used. Note that cltrain must be provided. |
|--------|---|
| scheme | "iterationwise" The performance measures listed above are computed for each different iteration, i.e. each different learningset |
| | "observationwise" The performance measures listed above (except for "auc") are computed separately for each observation classified one or several times, depending on the learningset scheme. |
| | "classwise" The performance measures (exceptions: "auc", "0.632", "0.632+") are computed separately for each class, averaged over both it- erations and observations. |

Value

An object of class evaloutput.

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

Christoph Bernau
dernau@ibe.med.uni-muenchen.de>

References

Efron, B. and Tibshirani, R. (1997). Improvements on cross-validation: The .632+ bootstrap method.

Journal of the American Statistical Association, 92, 548-560.

Slawski, M. Daumer, M. Boulesteix, A.-L. (2008) CMA - A comprehensive Bioconductor package for supervised classification with high dimensional data. *BMC Bioinformatics 9: 439*

See Also

evaloutput, classification, compare

Examples

```
### simple linear discriminant analysis example using bootstrap datasets:
### datasets:
data(golub)
golubY <- golub[,1]</pre>
### extract gene expression from first 10 genes
golubX <- as.matrix(golub[,2:11])</pre>
### generate 25 bootstrap datasets
set.seed(333)
bootds <- GenerateLearningsets(y = golubY, method = "bootstrap", ntrain = 30, niter = 10,
### run classification()
ldalist <- classification(X=golubX, y=golubY, learningsets = bootds, classifier=ldaCMA)</pre>
### Evaluation:
eval_iter <- evaluation(ldalist, scheme = "iter")</pre>
eval_obs <- evaluation(ldalist, scheme = "obs")</pre>
show(eval_iter)
show(eval_obs)
summary(eval_iter)
summary(eval_obs)
```

fdaCMA-methods

```
### auc with boxplot
eval_auc <- evaluation(ldalist, scheme = "iter", measure = "auc")
boxplot(eval_auc)
### which observations have often been misclassified ?
obsinfo(eval_obs, threshold = 0.75)</pre>
```

fdaCMA-methods Fisher's Linear Discriminant Analysis

Description

Fisher's Linear Discriminant Analysis constructs a subspace of 'optimal projections' in which classification is performed. The directions of optimal projections are computed by the function cancor from the package stats. For an exhaustive treatment, see e.g. Ripley (1996).

Methods

X = "matrix", y = "numeric", f = "missing" signature 1 X = "matrix", y = "factor", f = "missing" signature 2 X = "data.frame", y = "missing", f = "formula" signature 3 X = "ExpressionSet", y = "character", f = "missing" signature 4

For references, further argument and output information, consult fdaCMA.

fdaCMA

Fisher's Linear Discriminant Analysis

Description

Fisher's Linear Discriminant Analysis constructs a subspace of 'optimal projections' in which classification is performed. The directions of optimal projections are computed by the function cancor from the package stats. For an exhaustive treatment, see e.g. Ripley (1996).

For S4 method information, see fdaCMA-methods.

Usage

```
fdaCMA(X, y, f, learnind, comp = 1, plot = FALSE)
```

Arguments

Х

Gene expression data. Can be one of the following:

- A matrix. Rows correspond to observations, columns to variables.
- A data.frame, when f is not missing (s. below).
- An object of class ExpressionSet. Class labels. Can be one of the following:

У

- A numeric vector.
- A factor.

| | • A character if X is an ExpressionSet that specifies the phenotype variable. |
|----------|--|
| | • missing, if X is a data.frame and a proper formula f is provided. WARNING: The class labels will be re-coded to range from 0 to K-1, where K is the total number of different classes in the learning set. |
| f | A two-sided formula, if X is a data.frame. The left part correspond to class labels, the right to variables. |
| learnind | An index vector specifying the observations that belong to the learning set. May be missing; in that case, the learning set consists of all observations and predictions are made on the learning set. |
| comp | Number of discriminant coordinates (projections) to compute. Default is one, must be smaller than or equal to $K-1$, where K is the number of classes. |
| plot | Should the projections onto the space spanned by the optimal projection direc- tions be plotted ? Default is FALSE. |

Value

An object of class cloutput.

Note

Excessive variable selection has usually to performed before fdaCMA can be applied in the p > n setting. Not reducing the number of variables can result in an error message.

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

References

Ripley, B.D. (1996)

Pattern Recognition and Neural Networks.

Cambridge University Press

See Also

```
compBoostCMA, dldaCMA, ElasticNetCMA, fdaCMA, flexdaCMA, gbmCMA, knnCMA,
ldaCMA,LassoCMA,nnetCMA,pknnCMA,plrCMA,pls_ldaCMA,pls_lrCMA,pls_rfCMA,
pnnCMA,qdaCMA,rfCMA,scdaCMA,shrinkldaCMA,svmCMA
```

Examples

```
### load Golub AML/ALL data
data(golub)
### extract class labels
golubY <- golub[,1]
### extract gene expression from first 10 genes
golubX <- as.matrix(golub[,2:11])
### select learningset
ratio <- 2/3
set.seed(111)</pre>
```

filter

```
learnind <- sample(length(golubY), size=floor(ratio*length(golubY)))</pre>
### run FDA
fdaresult <- fdaCMA(X=golubX, y=golubY, learnind=learnind, comp = 1, plot = TRUE)</pre>
### show results
show(fdaresult)
ftable(fdaresult)
plot(fdaresult)
### multiclass example:
### load Khan data
data(khan)
### extract class labels
khanY <- khan[,1]</pre>
### extract gene expression from first 10 genes
khanX <- as.matrix(khan[,2:11])</pre>
### select learningset
set.seed(111)
learnind <- sample(length(khanY), size=floor(ratio*length(khanY)))</pre>
### run FDA
fdaresult <- fdaCMA(X=khanX, y=khanY, learnind=learnind, comp = 2, plot = TRUE)</pre>
### show results
show(fdaresult)
ftable(fdaresult)
plot(fdaresult)
```

```
filter
```

Filter functions for Gene Selection

Description

The functions listed above are usually not called by the user but via GeneSelection.

Usage

```
ttest(X, y, learnind, ...)
welchtest(X, y, learnind, ...)
ftest(X, y, learnind, ...)
kruskaltest(X, y, learnind, ...)
limmatest(X, y, learnind, ...)
golubcrit(X, y, learnind, ...)
rfe(X, y, learnind, ...)
```

Arguments

| Х | A numeric matrix of gene expression values. |
|----------|--|
| У | A numeric vector of class labels. |
| learnind | An index vector specifying the observations that belong to the learning set. |
| | Currently unused argument. |
| | |

Value

An object of class varseloutput.

References

Slawski, M. Daumer, M. Boulesteix, A.-L. (2008) CMA - A comprehensive Bioconductor package for supervised classification with high dimensional data. *BMC Bioinformatics 9: 439*

flexdaCMA-methods Flexible Discriminant Analysis

Description

This method is experimental.

It is easy to show that, after appropriate scaling of the predictor matrix X, Fisher's Linear Discriminant Analysis is equivalent to Discriminant Analysis in the space of the fitted values from the linear regression of the nlearn x K indicator matrix of the class labels on X. This gives rise to 'nonlinear discrimant analysis' methods that expand X in a suitable, more flexible basis. In order to avoid overfitting, penalization is used. In the implemented version, the linear model is replaced by a generalized additive one, using the package mgcv.

Methods

X = "matrix", y = "numeric", f = "missing" signature 1 X = "matrix", y = "factor", f = "missing" signature 2 X = "data.frame", y = "missing", f = "formula" signature 3 X = "ExpressionSet", y = "character", f = "missing" signature 4 For further argument and output information, consult flexdaCMA.

flexdaCMA

Flexible Discriminant Analysis

Description

This method is experimental.

It is easy to show that, after appropriate scaling of the predictor matrix X, Fisher's Linear Discriminant Analysis is equivalent to Discriminant Analysis in the space of the fitted values from the linear regression of the nlearn x K indicator matrix of the class labels on X. This gives rise to 'nonlinear discrimant analysis' methods that expand X in a suitable, more flexible basis. In order to avoid overfitting, penalization is used. In the implemented version, the linear model is replaced by a generalized additive one, using the package mgcv.

For S4 method information, s. flexdaCMA-methods.

Usage

```
flexdaCMA(X, y, f, learnind, comp = 1, plot = FALSE, ...)
```

flexdaCMA

| Arguments | |
|-----------|--|
| Х | Gene expression data. Can be one of the following: |
| | • A matrix. Rows correspond to observations, columns to variables. |
| | • A data.frame, when f is <i>not</i> missing (s. below). |
| | • An object of class ExpressionSet. |
| У | Class labels. Can be one of the following: |
| | • A numeric vector. |
| | • A factor. |
| | • A character if X is an ExpressionSet that specifies the phenotype variable. |
| | • missing, if X is a data.frame and a proper formula f is provided. |
| | WARNING: The class labels will be re-coded to range from 0 to K-1, where K is the total number of different classes in the learning set. |
| f | A two-sided formula, if X is a data.frame. The left part correspond to class labels, the right to variables. |
| learnind | An index vector specifying the observations that belong to the learning set. May be missing; in that case, the learning set consists of all observations and predictions are made on the learning set. |
| comp | Number of discriminant coordinates (projections) to compute. Default is one, must be smaller than or equal to K-1, where K is the number of classes. |
| plot | Should the projections onto the space spanned by the optimal projection direc- tions be plotted ? Default is FALSE. |
| | Further arguments passed to the function gam from the package mgcv. |

Value

An object of class cloutput.

Note

Excessive variable selection has usually to performed before flexdaCMA can be applied in the p > n setting. Recall that the original predictor dimension is even enlarged, therefore, it should be applied only with very few variables.

Author(s)

Martin Slawski <ms@cs.uni-sb.de> Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

References

Ripley, B.D. (1996) Pattern Recognition and Neural Networks. *Cambridge University Press*

See Also

```
compBoostCMA, dldaCMA, ElasticNetCMA, fdaCMA, gbmCMA, knnCMA, ldaCMA, LassoCMA,
nnetCMA, pknnCMA, plrCMA, pls_ldaCMA, pls_lrCMA, pls_rfCMA, pnnCMA, qdaCMA,
rfCMA, scdaCMA, shrinkldaCMA, svmCMA
```

Examples

```
### load Golub AML/ALL data
data(golub)
### extract class labels
golubY <- golub[,1]</pre>
### extract gene expression from first 5 genes
golubX <- as.matrix(golub[,2:6])</pre>
### select learningset
ratio <- 2/3
set.seed(111)
learnind <- sample(length(golubY), size=floor(ratio*length(golubY)))</pre>
### run flexible Discriminant Analysis
result <- flexdaCMA(X=golubX, y=golubY, learnind=learnind, comp = 1)</pre>
### show results
show(result)
ftable(result)
plot(result)
```

ftable

Cross-tabulation of predicted and true class labels

Description

An object of class cloutput contains (among others) the slot y and yhat. The former contains the true, the last the predicted class labels. Both are cross-tabulated in order to obtain a so-called confusion matrix. Counts out of the diagonal are misclassifications.

Arguments

| X | An object of class cloutput |
|---|-----------------------------|
| | Currently unused argument. |

Value

No return.

Author(s)

Martin Slawski <martin.slawski@campus.lmu.de>
Anne-Laure Boulesteix http://www.slcmsr.net/boulesteix

See Also

For more advanced evaluation: evaluation

26

gbmCMA-methods Tree-based Gradient Boosting

Description

Roughly speaking, Boosting combines 'weak learners' in a weighted manner in a stronger ensemble. This method calls the function gbm.fit from the package gbm. The 'weak learners' are simple trees that need only very few splits (default: 1).

Methods

X = "matrix", **y** = "numeric", **f** = "missing" signature 1

X = "matrix", **y** = "factor", **f** = "missing" signature 2

X = "data.frame", **y** = "missing", **f** = "formula" signature 3

X = "ExpressionSet", y = "character", f = "missing" signature 4

For further argument and output information, consult gbmCMA.

gbmCMA

Tree-based Gradient Boosting

Description

Roughly speaking, Boosting combines 'weak learners' in a weighted manner in a stronger ensemble. This method calls the function gbm.fit from the package gbm. The 'weak learners' are simple trees that need only very few splits (default: 1).

For S4 method information, see gbmCMA-methods.

Usage

```
gbmCMA(X, y, f, learnind, ...)
```

Arguments

Х

Gene expression data. Can be one of the following:

- A matrix. Rows correspond to observations, columns to variables.
- A data.frame, when f is not missing (s. below).
- An object of class ExpressionSet.

Class labels. Can be one of the following:

У

- A numeric vector.
- A factor.
- A character if X is an ExpressionSet that specifies the phenotype variable.
- missing, if X is a data.frame and a proper formula f is provided.

WARNING: The class labels will be re-coded to range from 0 to K-1, where K is the total number of different classes in the learning set.

| f | A two-sided formula, if ${\tt X}$ is a data.frame. The left part correspond to class labels, the right to variables. |
|----------|--|
| learnind | An index vector specifying the observations that belong to the learning set. May be missing; in that case, the learning set consists of all observations and predictions are made on the learning set. |
| | Further arguments passed to the function ${\tt gbm.fit}$ from the package of the same name. Worth mentionning are |
| | ntrees Number of trees to fit (size of the ensemble), defaults to 100. This parameter should be optimized using tune. |
| | shrinkage The learning rate (default is 0.001). Usually fixed to a very low value. |
| | distribution Loss function to be used. Default is "bernoulli", i.e. LogitBoost, a (less robust) alternative is "adaboost". |
| | <pre>interaction.depth Number of splits used by the 'weak learner' (single</pre> |

Value

An onject of class cloutput.

Note

Up to now, this method can only be applied to binary classification.

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

References

Ridgeway, G. (1999).
The state of boosting. *Computing Science and Statistics*, 31:172-181
Friedman, J. (2001).
Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics 29(5):1189-1232.*

See Also

```
compBoostCMA, dldaCMA, ElasticNetCMA, fdaCMA, flexdaCMA, knnCMA, ldaCMA,
LassoCMA, nnetCMA, pknnCMA, plrCMA, pls_ldaCMA, pls_lrCMA, pls_rfCMA, pnnCMA,
qdaCMA, rfCMA, scdaCMA, shrinkldaCMA, svmCMA
```

Examples

```
### load Golub AML/ALL data
data(golub)
### extract class labels
golubY <- golub[,1]
### extract gene expression</pre>
```

GenerateLearningsets

```
golubX <- as.matrix(golub[,-1])
### select learningset
ratio <- 2/3
set.seed(111)
learnind <- sample(length(golubY), size=floor(ratio*length(golubY)))
### run tree-based gradient boosting (no tuning)
gbmresult <- gbmCMA(X=golubX, y=golubY, learnind=learnind, n.trees = 500)
show(gbmresult)
ftable(gbmresult)
plot(gbmresult)</pre>
```

GenerateLearningsets

Repeated Divisions into learn- and tets sets

Description

Due to very small sample sizes, the classical division learnset/testset does not give accurate information about the classification performance. Therefore, several different divisions should be used and aggregated. The implemented methods are discussed in Braga-Neto and Dougherty (2003) and Molinaro et al. (2005) whose terminology is adopted.

This function is usually the basis for all deeper analyses.

Usage

Arguments

| n | The total number of observations in the available data set. May be missing if y is provided instead. |
|--------|---|
| У | A vector of class labels, either numeric or a factor. <i>Must</i> be given if strat=TRUE or n is not specified. |
| method | Which kind of scheme should be used to generate divisions into learning sets and test sets ? Can be one of the following: |
| | "LOOCV" Leaving-One-Out Cross Validation. |
| | "CV" (Ordinary) Cross-Validation. Note that fold must as well be specified. |
| | "MCCV" Monte-Carlo Cross Validation, i.e. random divisions into learning |
| | sets with ntrain(s.below) observations and tests sets with ntrain ob- servations. |
| | "bootstrap" Learning sets are generated by drawing ntrain times with re- placement from all observations. Those not drawn not all form the test set. |
| fold | Gives the number of CV-groups. Used only when method="CV" |
| niter | Number of iterations (s.details). |
| ntrain | Number of observations in the learning sets. Used only when method="MCCV" or method="bootstrap". |
| strat | Logical. Should stratified sampling be performed, i.e. the proportion of observations from each class in the learning sets be the same as in the whole data set ? |
| | Does not apply for method = "LOOCV". |

Details

- When method="CV", niter gives the number of times the whole CV-procedure is repeated. The output matrix has then foldxniter rows. When method="MCCV" or method="bootstrap", niter is simply the number of considered learning sets.
- Note that method="CV", fold=n is equivalent to method="LOOCV".

Value

An object of class learningsets

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

Christoph Bernau
dernau@ibe.med.uni-muenchen.de>

References

Braga-Neto, U.M., Dougherty, E.R. (2003).

Is cross-validation valid for small-sample microarray classification ?

Bioinformatics, 20(3), 374-380

Molinaro, A.M., Simon, R., Pfeiffer, R.M. (2005).

Prediction error estimation: a comparison of resampling methods.

Bioinformatics, 21(15), 3301-3307

Slawski, M. Daumer, M. Boulesteix, A.-L. (2008) CMA - A comprehensive Bioconductor package for supervised classification with high dimensional data. *BMC Bioinformatics 9: 439*

See Also

learningsets, GeneSelection, tune, classification

Examples

```
# LOOCV
loo <- GenerateLearningsets(n=40, method="LOOCV")</pre>
show(loo)
# five-fold-CV
CV5 <- GenerateLearningsets (n=40, method="CV", fold=5)
show(loo)
# MCCV
mccv <- GenerateLearningsets(n=40, method = "MCCV", niter=3, ntrain=30)</pre>
show(mccv)
# Bootstrap
boot <- GenerateLearningsets(n=40, method="bootstrap", niter=3)</pre>
# stratified five-fold-CV
set.seed(113)
classlabels <- sample(1:3, size = 50, replace = TRUE, prob = c(0.3, 0.5, 0.2))
CV5strat <- GenerateLearningsets(y = classlabels, method="CV", fold=5, strat = TRUE)
show(CV5strat)
```

30

genesel-class

"genesel"

Description

Object returned from a call to GeneSelection

Slots

- rankings: A list of matrices. For the two-class case and the multi-class case where a genuine multi-class method has been used for variable selection, the length of the list is one. Otherwise, it is named according to the different binary scenarios (e.g. 1 vs 3). Each list element is a matrix with rows corresponding to iterations (different learningsets) and columns to variables. Each row thus contains an index vector representing the order of the variables with respect to their variable importance (s. slot importance)
- importance: A list of matrices, with the same structure as described for the slot rankings. Each row of these matrices are ordered according to rankings and contain the variable importance measure (absolute value of test statistic or regression coefficient).
- method: Name of the method used for variable selection, s. GeneSelection.
- scheme: The scheme used in the case of a non-binary response, one of "pairwise", "onevs-all" or "multiclass".

Methods

show Use show (genesel-object) for brief information

- **toplist** Use toplist (genesel-object, k=10, iter = 1) to display the top first 10 variables and their variable importance for the first iteration (first learningset), s.toplist.
- plot Use plot (genesel-object, k=10, iter=1) to display a barplot of the variable importance of the top first 10 variables, s. plot, genesel-method

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

See Also

GeneSelection

```
GeneSelection-methods
```

General method for variable selection with various methods

Description

Performs gene selection for the following signatures:

Methods

X = "matrix", y = "numeric", f = "missing" signature 1 X = "matrix", y = "factor", f = "missing" signature 2 X = "data.frame", y = "missing", f = "formula" signature 3 X = "ExpressionSet", y = "character", f = "missing" signature 4 For further argument and output information, consult GeneSelection.

GeneSelection General method for variable selection with various methods

Description

For different learning data sets as defined by the argument learningsets, this method ranks the genes from the most relevant to the less relevant using one of various 'filter' criteria or provides a sparse collection of variables (Lasso, ElasticNet, Boosting). The results are typically used for variable selection for the classification procedure that follows.

For S4 class information, s. ${\tt GeneSelection-methods}.$

Usage

```
GeneSelection(X, y, f, learningsets, method = c("t.test", "welch.test", "wilcox.
```

Arguments

| Х | Gene expression data. Can be one of the following: |
|--------------|--|
| | • A matrix. Rows correspond to observations, columns to variables. |
| | • A data.frame, when f is <i>not</i> missing (s. below). |
| | • An object of class ExpressionSet. |
| У | Class labels. Can be one of the following: |
| | • A numeric vector. |
| | • A factor. |
| | • A character if X is an ExpressionSet. |
| | • missing, if X is a data.frame and a proper formula f is provided. |
| f | A two-sided formula, if X is a data.frame. The left part correspond to class labels, the right to variables. |
| learningsets | An object of class learningsets. May be missing, then the complete datasets is used as learning set. |
| | |

GeneSelection

| method | A character specifying the method to be used: |
|--------|---|
| | t.test two-sample t.test (equal variances for both classes assumed). |
| | welch.test Welch modification of the t.test (unequal variances for both classes). |
| | wilcox.test Wilcoxon rank sum test. |
| | <pre>f.test F test belonging to the linear hypothesis that the mean is the same for all classes. Usually used for the multiclass scheme, is equivalent to method = t.test in the two-class case.</pre> |
| | kruskal.test Multi-class generalization of the Wilcoxon rank sum test and the nonparametric pendant to the F test, respectively. |
| | <pre>limma 'Moderated t' statistic for the two-class case and 'moderated F' statistic for the multiclass case, described in Smyth (2003). Requires the package limma.</pre> |
| | rfe One-step Recursive Feature Elimination, based on the Support Vector Ma- chine. The method is decribed in Guyon et al. (2002). Requires the package e1071. Take care that appropriate hyperparameters are passed by the argument. |
| | rf Random Forest Variable Importance Measure. Requires the package randomForest |
| | lasso L1 penalized logistic regression leads to sparsity with respect to the variables used. Calls the function LassoCMA, which requires the package glmpath. warning: Take care that appropriate hyperparameters are |
| | <pre>passed by the argument. elasticnet Penalized logistic regression with both L1 and L2 penalty, claimed by Zhou and Hastie (2004) to select 'variable groups'. Calls the function ElasticNetCMA, which requires the package glmpath. warning: Take care that appropriate hyperparameters are passed by the argument.</pre> |
| | boosting Componentwise boosting (Buehlmann and Yu, 2003) has been shown to mimic the LASSO (Efron et al., 2004; Buehlmann and Yu, 2006). Calls the function compBoostCMA Take care that appropriate hyperparameters are passed by the argument. |
| | golub The (theoretically unfounded) variable selection criterion used by Golub et al. (1999), s. golub. |
| scheme | The scheme to be used in the case of a non-binary response. Must be one of "pairwise", "one-vs-all" or "multiclass". The last case only makes sense if method is one of f.test, limma, rf, boosting, which can directly be applied to the multi class case. |
| trace | Should the progress be traced ? Default is TRUE. |
| | Further arguments passed to the function performing variable selection, s. method. |

Value

An object of class genesel.

Note

most of the methods described above are only apt for the binary classification case. The only ones that can be used without restriction in the multiclass case are

- f.test
- kruskal.test
- rf

GeneSelection

• boosting

For the rest, pairwise or one-vs-all schemes are used.

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

Christoph Bernau
dernau@ibe.med.uni-muenchen.de>

References

Smyth, G. K., Yang, Y.-H., Speed, T. P. (2003). Statistical issues in microarray data analysis. *Methods in Molecular Biology 224, 111-136.*

Guyon, I., Weston, J., Barnhill, S., Vapnik, V. (2002). Gene Selection for Cancer Classification using support vector machines. *Journal of Machine Learning Research, 46, 389-422*

Zhou, H., Hastie, T. (2004). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society B*, 67(2),301-320

Buelmann, P., Yu, B. (2003). Boosting with the L2 loss: Regression and Classification. *Journal of the American Statistical Association, 98, 324-339*

Efron, B., Hastie, T., Johnstone, I., Tibshirani, R. (2004). Least Angle Regression. *Annals of Statistics*, *32:407-499*

Buehlmann, P., Yu, B. (2006). Sparse Boosting. Journal of Machine Learning Research, 7- 1001:1024

Slawski, M. Daumer, M. Boulesteix, A.-L. (2008) CMA - A comprehensive Bioconductor package for supervised classification with high dimensional data. *BMC Bioinformatics 9: 439*

See Also

filter, GenerateLearningsets, tune, classification

Examples

```
# load Golub AML/ALL data
data(golub)
### extract class labels
golubY <- golub[,1]
### extract gene expression from first 10 genes
golubX <- as.matrix(golub[,-1])
### Generate five different learningsets
set.seed(11)
five <- GenerateLearningsets(y=golubY, method = "CV", fold = 5, strat = TRUE)
### simple t-test:
selttest <- GeneSelection(golubX, golubY, learningsets = five, method = "t.test")
### show result:
show(selttest)</pre>
```

34

golub

```
toplist(selttest, k = 10, iter = 1)
plot(selttest, iter = 1)
```

golub

ALL/AML dataset of Golub et al. (1999)

Description

s. below

Usage

data(golub)

Format

A data frame with 38 observations and 3052 variables. The first column (named golub.cl) contains the tumor classes (ALL = acute lymphatic leukaemia, AML = acute myeloid leukaemia).\golub.cl: a factor with levels ALL AML.\X2-X3051: Gene expression values.

Source

Adopted from the dataset in the package multtest.

References

Golub, T., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P.,

Coller, H., Loh, M. L., Downing, J., Caligiuri, M. A., Bloomfeld, C. D., Lander, E. S. (1999).

Molecular classification of cancer: class discovery and class prediction by gene expression monitoring.

Science 286, 531-537.

Examples

data(golub)

internals

Internal functions

Description

Not intended to be called directly by the user.

join-methods

Description

The list of objects of class cloutput can be unified into one object for the following signatures:

Methods

cloutputlist = "list" signature 1

For further argument and output information, consult join.

join

Combine list elements returned by the method classification

Description

The method classification returns a list of class cloutput or clvarseloutput. It is often more convenient to work with an object of class cloutput instead with a whole list, e.g. because the convenience method defined for that class can be used.

For S4 method information, s. join-methods

Usage

```
join(cloutputlist)
```

Arguments

cloutputlist A list of objects of classes cloutput or clvarseloutput, usually that returned by a call to the method classification. The only requirement for a succesful join is that the used dataset and classfier are the same for each list element.

Value

An object of class cloutput. warning: If the elements of cloutputlist have originally been of class clvarseloutput, the slot varsel will be dropped !

Note

The result of the join method is incompatible with the methods evaluation, compare. These require the lists returned by classification.

See Also

classification, evaluation

khan

Description

s. below

Usage

data(khan)

Format

A data frame with 63 observations on the following 2309 variables. The first column (named khanY) contains the tumor classes (BL = Burkitt Lymphoma, EWS = Ewing Sarcoma, NB = Neuro Blastoma, RMS = Rhabdomyosarcoma).

khanY: a factor with levels BL EWS NB RMS \ X2-X2309: Gene expression values.

Source

Adopted from the dataset in the package pamr.

References

Khan, J., Wei, J. S., Ringner, M., Saal, L. H., Ladanyi, M., Westermann, F., Berthold, F., Schwab, M., Antonescu, C. R., Peterson, C., Meltzer, P. S., (2001).

Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks.

Nature Medicine 7, 673-679.

Examples

data(khan)

knnCMA-methods Nearest Neighbours

Description

Ordinary k nearest neighbours algorithm from the very fast implementation in the package class

Methods

X = "matrix", y = "numeric", f = "missing" signature 1 X = "matrix", y = "factor", f = "missing" signature 2 X = "data.frame", y = "missing", f = "formula" signature 3 X = "ExpressionSet", y = "character", f = "missing" signature 4

For further argument and output information, consult knnCMA.

knnCMA

Description

Ordinary k nearest neighbours algorithm from the very fast implementation in the package class. For S4 method information, see knnCMA-methods.

Usage

knnCMA(X, y, f, learnind, ...)

Arguments

| Х | Gene expression data. Can be one of the following: |
|----------|---|
| | • A matrix. Rows correspond to observations, columns to variables. |
| | • A data.frame, when f is <i>not</i> missing (s. below). |
| | • An object of class ExpressionSet. |
| У | Class labels. Can be one of the following: |
| | • A numeric vector. |
| | • A factor. |
| | • A character if X is an ExpressionSet that specifies the phenotype variable. |
| | • missing, if X is a data.frame and a proper formula f is provided. |
| | WARNING : The class labels will be re-coded to range from 0 to $K-1$, where K is the total number of different classes in the learning set. |
| f | A two-sided formula, if X is a data.frame. The left part correspond to class labels, the right to variables. |
| learnind | An index vector specifying the observations that belong to the learning set. Must not be missing for this method. |
| ••• | Further arguments to be passed to knn from the package class, in particular the number of nearest neighbours to use (argument k). |

Value

An object of class cloutput.

Note

Class probabilities are not returned. For a probabilistic variant of knn, s. pknnCMA.

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

LassoCMA-methods

References

Ripley, B.D. (1996)

Pattern Recognition and Neural Networks.

Cambridge University Press

See Also

```
compBoostCMA, dldaCMA, ElasticNetCMA, fdaCMA, flexdaCMA, gbmCMA, ldaCMA,
LassoCMA, nnetCMA, pknnCMA, plrCMA, pls_ldaCMA, pls_lrCMA, pls_rfCMA, pnnCMA,
qdaCMA, rfCMA, scdaCMA, shrinkldaCMA, svmCMA
```

Examples

```
### load Golub AML/ALL data
data(golub)
### extract class labels
golubY <- golub[,1]</pre>
### extract gene expression from first 10 genes
golubX <- as.matrix(golub[,-1])</pre>
### select learningset
ratio <- 2/3
set.seed(111)
learnind <- sample(length(golubY), size=floor(ratio*length(golubY)))</pre>
### run k-nearest neighbours
result <- knnCMA(X=golubX, y=golubY, learnind=learnind, k = 3)
### show results
show(result)
ftable(result)
### multiclass example:
### load Khan data
data(khan)
### extract class labels
khanY <- khan[,1]</pre>
### extract gene expression
khanX <- as.matrix(khan[,-1])</pre>
### select learningset
set.seed(111)
learnind <- sample(length(khanY), size=floor(ratio*length(khanY)))</pre>
### run knn
result <- knnCMA(X=khanX, y=khanY, learnind=learnind, k = 5)</pre>
### show results
show(result)
ftable(result)
```

LassoCMA-methods L1 penalized logistic regression

Description

The Lasso (Tibshirani, 1996) is one of the most popular tools for simultaneous shrinkage and variable selection. Recently, Friedman, Hastie and Tibshirani (2008) have developped and algorithm to compute the entire solution path of the Lasso for an arbitrary generalized linear model, implemented in the package glmnet. The method can be used for variable selection alone, s. GeneSelection

Methods

X = "matrix", y = "numeric", f = "missing" signature 1
X = "matrix", y = "factor", f = "missing" signature 2
X = "data.frame", y = "missing", f = "formula" signature 3
X = "ExpressionSet", y = "character", f = "missing" signature 4
For references, further argument and output information, consult LassoCMA.

LassoCMA

L1 penalized logistic regression

Description

The Lasso (Tibshirani, 1996) is one of the most popular tools for simultaneous shrinkage and variable selection. Recently, Friedman, Hastie and Tibshirani (2008) have developped and algorithm to compute the entire solution path of the Lasso for an arbitrary generalized linear model, implemented in the package glmnet. The method can be used for variable selection alone, s. GeneSelection.

For S4 method information, see LassoCMA-methods.

Usage

```
LassoCMA(X, y, f, learnind, norm.fraction = 0.1, ...)
```

Arguments

| Х | Gene expression data. Can be one of the following: |
|----------|--|
| | • A matrix. Rows correspond to observations, columns to variables. |
| | • A data.frame, when f is <i>not</i> missing (s. below). |
| | • An object of class ExpressionSet. note: by default, the predictors are scaled to have unit variance and zero mean. Can be changed by passing standardize = FALSE via the argument. |
| У | Class labels. Can be one of the following: |
| | • A numeric vector. |
| | • A factor. |
| | • A character if X is an ExpressionSet that specifies the phenotype variable. |
| | • missing, if X is a data.frame and a proper formula f is provided. |
| | WARNING : The class labels will be re-coded to range from 0 to $K-1$, where K is the total number of different classes in the learning set. |
| f | A two-sided formula, if X is a data.frame. The left part correspond to class labels, the right to variables. |
| learnind | An index vector specifying the observations that belong to the learning set. May be missing; in that case, the learning set consists of all observations and predictions are made on the learning set. |

40

LassoCMA

```
norm.fraction
```

L1 Shrinkage intensity, expressed as the fraction of the coefficient L1 norm compared to the maximum possible L1 norm (corresponds to fraction = 1). Lower values correspond to higher shrinkage. Note that the default (0.1) need not produce good results, i.e. tuning of this parameter is recommended. Further arguments passed to the function glmpath from the package of the

same name.

Value

. . .

An object of class clvarseloutput.

Note

For a strongly related method, s. ElasticNetCMA. Up to now, this method can only be applied to binary classification.

Author(s)

Martin Slawski <ms@cs.uni-sb.de> Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de> Christoph Bernau <bernau@ibe.med.uni-muenchen.de>

References

Tibshirani, R. (1996) Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B, 58(1), 267-288* Friedman, J., Hastie, T. and Tibshirani, R. (2008) Regularization

Paths for Generalized Linear Models via Coordinate Descent http://www-stat.stanford.edu/~hastie/Papers/glmnet.pdf

See Also

```
compBoostCMA, dldaCMA, ElasticNetCMA, fdaCMA, flexdaCMA, gbmCMA, knnCMA,
ldaCMA, nnetCMA, pknnCMA, plrCMA, pls_ldaCMA, pls_lrCMA, pls_rfCMA, pnnCMA,
qdaCMA, rfCMA, scdaCMA, shrinkldaCMA, svmCMA
```

Examples

```
### load Golub AML/ALL data
data(golub)
### extract class labels
golubY <- golub[,1]
### extract gene expression
golubX <- as.matrix(golub[,-1])
### select learningset
ratio <- 2/3
set.seed(111)
learnind <- sample(length(golubY), size=floor(ratio*length(golubY)))
### run L1 penalized logistic regression (no tuning)
lassoresult <- LassoCMA(X=golubX, y=golubY, learnind=learnind, norm.fraction = 0.2)
show(lassoresult)
ftable(lassoresult)
plot(lassoresult)</pre>
```

ldaCMA-methods Linear Discriminant Analysis

Description

Performs a linear discriminant analysis for the following signatures:

Methods

X = "matrix", y = "numeric", f = "missing" signature 1 X = "matrix", y = "factor", f = "missing" signature 2 X = "data.frame", y = "missing", f = "formula" signature 3 X = "ExpressionSet", y = "character", f = "missing" signature 4 For further argument and output information, consult ldaCMA.

ldaCMA

Linear Discriminant Analysis

Description

Performs a linear discriminant analysis under the assumption of a multivariate normal distribution in each classes (with equal, but generally structured) covariance matrices. The function lda from the package MASS is called for computation.

For S4 method information, see ldaCMA-methods.

Usage

ldaCMA(X, y, f, learnind, ...)

Arguments

| Х | Gene expression data. Can be one of the following: |
|----------|--|
| | • A matrix. Rows correspond to observations, columns to variables. |
| | • A data.frame, when f is not missing (s. below). |
| | • An object of class ExpressionSet. |
| У | Class labels. Can be one of the following: |
| | • A numeric vector. |
| | • A factor. |
| | • A character if X is an ExpressionSet that specifies the phenotype variable. |
| | • missing, if X is a data.frame and a proper formula f is provided. |
| | WARNING: The class labels will be re-coded to range from 0 to K-1, where K is the total number of different classes in the learning set. |
| f | A two-sided formula, if X is a data.frame. The left part correspond to class labels, the right to variables. |
| learnind | An index vector specifying the observations that belong to the learning set. May be missing; in that case, the learning set consists of all observations and predictions are made on the learning set. |
| | Further arguments to be passed to lda from the package MASS |

ldaCMA

Value

An object of class cloutput.

Note

Excessive variable selection has usually to performed before ldaCMA can be applied in the p > n setting. Not reducing the number of variables can result in an error message.

Author(s)

```
Martin Slawski <ms@cs.uni-sb.de>
```

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

References

McLachlan, G.J. (1992).

Discriminant Analysis and Statistical Pattern Recognition. *Wiley, New York*

See Also

```
compBoostCMA, dldaCMA, ElasticNetCMA, fdaCMA, flexdaCMA, gbmCMA, knnCMA,
LassoCMA, nnetCMA, pknnCMA, plrCMA, pls_ldaCMA, pls_lrCMA, pls_rfCMA, pnnCMA,
qdaCMA, rfCMA, scdaCMA, shrinkldaCMA, svmCMA
```

Examples

```
### load Golub AML/ALL data
data(golub)
### extract class labels
golubY <- golub[,1]</pre>
### extract gene expression from first 10 genes
golubX <- as.matrix(golub[,2:11])</pre>
### select learningset
ratio <- 2/3
set.seed(111)
learnind <- sample(length(golubY), size=floor(ratio*length(golubY)))</pre>
### run LDA
ldaresult <- ldaCMA(X=golubX, y=golubY, learnind=learnind)</pre>
### show results
show(ldaresult)
ftable(ldaresult)
plot(ldaresult)
### multiclass example:
### load Khan data
data(khan)
### extract class labels
khanY <- khan[,1]</pre>
### extract gene expression from first 10 genes
khanX <- as.matrix(khan[,2:11])</pre>
### select learningset
set.seed(111)
learnind <- sample(length(khanY), size=floor(ratio*length(khanY)))</pre>
### run LDA
```

```
ldaresult <- ldaCMA(X=khanX, y=khanY, learnind=learnind)
### show results
show(ldaresult)
ftable(ldaresult)
plot(ldaresult)</pre>
```

learningsets-class "learningsets"

Description

An object returned from GenerateLearningsets which is usually passed as arguments to GeneSelection, tune and classification.

Slots

- learnmatrix: A matrix of dimension niter x ntrain. Each row contains the indices of those observations representing the learningset for one iteration. If method = CV, zeros appear due to rounding issues.
- method: The method used to generate the learnmatrix, s.GenerateLearningsets
- ntrain: Number of observations in one learning set. If method = CV, this number is not attained for all iterations, due to rounding issues.
- iter: Number of iterations (different learningsets) that are stored in learnmatrix.

Methods

• showUse show (learningsets-object) for brief information.

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

Christoph Bernau
dernau@ibe.med.uni-muenchen.de>

See Also

GenerateLearningsets, GeneSelection, tune, classification

44

nnetCMA-methods Feed-Forward Neural Networks

Description

This method provides access to the function nnet in the package of the same name that trains Feed-forward Neural Networks with one hidden layer.

Methods

X = "matrix", y = "numeric", f = "missing" signature 1 X = "matrix", y = "factor", f = "missing" signature 2 X = "data.frame", y = "missing", f = "formula" signature 3 X = "ExpressionSet", y = "character", f = "missing" signature 4 For further argument and output information, consult nnetCMA.

nnetCMA

Feed-forward Neural Networks

Description

This method provides access to the function nnet in the package of the same name that trains Feed-forward Neural Networks with one hidden layer. For S4 method information, see nnetCMA-methods

Usage

```
nnetCMA(X, y, f, learnind, eigengenes = FALSE, ...)
```

Arguments

Х

Gene expression data. Can be one of the following:

- A matrix. Rows correspond to observations, columns to variables.
- A data.frame, when f is not missing (s. below).
- An object of class ExpressionSet.

Class labels. Can be one of the following:

У

| • A numeric vector. | |
|---------------------|--|
|---------------------|--|

- A factor.
- A character if X is an ExpressionSet that specifies the phenotype variable.
- missing, if X is a data.frame and a proper formula f is provided.

WARNING: The class labels will be re-coded to range from 0 to K-1, where K is the total number of different classes in the learning set.

f A two-sided formula, if X is a data.frame. The left part correspond to class labels, the right to variables.

| learnind | An index vector specifying the observations that belong to the learning set. May be missing; in that case, the learning set consists of all observations and predictions are made on the learning set. |
|------------|---|
| eigengenes | Should the training be performed be in the space of eigengenes obtained from a singular value decomposition of the Gene expression data matrix ? Default is FALSE; in this case, variable selection is necessary to reduce the number of weights that have to be optimized. |
| | Further arguments passed to the function nnet from the package of the same name. Important parameters are: |
| | • "size", i.e. the number of units in the hidden layer |
| | |

• "decay" for weight decay.

Value

An object of class cloutput.

Note

- Excessive variable selection is usually necessary if eigengenes = FALSE
- Different runs of this method on the same dataset not necessarily produce the same results due to the fact that optimization for Feed-Forward Neural Networks is rather difficult and depends on the choice of (normally randomly chosen) starting values for the network weights.

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

Christoph Bernau
dernau@ibe.med.uni-muenchen.de>

References

Ripley, B.D. (1996) Pattern Recognition and Neural Networks. *Cambridge University Press*

See Also

```
compBoostCMA, dldaCMA, ElasticNetCMA, fdaCMA, flexdaCMA, gbmCMA, knnCMA,
ldaCMA,LassoCMA,nnetCMA,pknnCMA,plrCMA,pls_ldaCMA,pls_lrCMA,pls_rfCMA,
pnnCMA,qdaCMA,rfCMA,scdaCMA,shrinkldaCMA,svmCMA
```

Examples

```
### load Golub AML/ALL data
data(golub)
### extract class labels
golubY <- golub[,1]
### extract gene expression from first 10 genes
golubX <- as.matrix(golub[,2:11])
### select learningset
ratio <- 2/3</pre>
```

obsinfo

```
set.seed(111)
learnind <- sample(length(golubY), size=floor(ratio*length(golubY)))</pre>
### run nnet (not tuned)
nnetresult <- nnetCMA(X=golubX, y=golubY, learnind=learnind, size = 3, decay = 0.01)</pre>
### show results
show(nnetresult)
ftable(nnetresult)
plot(nnetresult)
### in the space of eigengenes (not tuned)
golubXfull <- as.matrix(golubX[,-1])</pre>
nnetresult <- nnetCMA(X=golubXfull, y=golubY, learnind = learnind, eigengenes = TRUE,</pre>
                       size = 3, decay = 0.01)
### show results
show(nnetresult)
ftable(nnetresult)
plot(nnetresult)
```

obsinfo

Classifiability of observations

Description

Some observations are harder to classify than others. It is frequently of interest to know which observations are consistenly misclassified; these are candiates for outliers or wrong class labels.

Arguments

| object | An object of class evaluation, generated with scheme = "observationwise" |
|-----------|--|
| threshold | threshold value of (observation-wise) performance measure, s. evaluation that has to be exceeded in order to speak of consistent misclassification. If measure = "average probability", then values <i>below</i> threshold are regarded as consistent misclassification. Note that the default values 1 is not sensible in that case |
| show | Should the information be printed ? Default is TRUE. |

Details

As not all observation must have been classified at least once, observations not classified at all are also shown.

Value

A list with two components

```
misclassification
```

A data.frame containing the indices of consistenly misclassfied observations and the corresponding performance measure.

notclassified

The indices of those observations not classfied at all, s. details.

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

References

Slawski, M. Daumer, M. Boulesteix, A.-L. (2008) CMA - A comprehensive Bioconductor package for supervised classification with high dimensional data. *BMC Bioinformatics 9: 439*

See Also

evaluation

pknnCMA-methods Probabilistic nearest neighbours

Description

Nearest neighbour variant that replaces the simple voting scheme by a weighted one (based on euclidean distances). This is also used to compute class probabilities.

Methods

X = "matrix", y = "numeric", f = "missing" signature 1 X = "matrix", y = "factor", f = "missing" signature 2 X = "data.frame", y = "missing", f = "formula" signature 3 X = "ExpressionSet", y = "character", f = "missing" signature 4 For further argument and output information, consult pknnCMA.

pknnCMA

Probabilistic Nearest Neighbours

Description

Nearest neighbour variant that replaces the simple voting scheme by a weighted one (based on euclidean distances). This is also used to compute class probabilities.

For S4 class information, see pknnCMA-methods.

Usage

```
pknnCMA(X, y, f, learnind, beta = 1, k = 1, \ldots)
```

48

pknnCMA

| Arguments | |
|-----------|---|
| Х | Gene expression data. Can be one of the following: |
| | • A matrix. Rows correspond to observations, columns to variables. |
| | • A data.frame, when f is not missing (s. below). |
| | • An object of class ExpressionSet. |
| У | Class labels. Can be one of the following: |
| | • A numeric vector. |
| | • A factor. |
| | • A character if X is an ExpressionSet that specifies the phenotype variable. |
| | • missing, if X is a data.frame and a proper formula f is provided. |
| | WARNING : The class labels will be re-coded to range from 0 to K-1, where K is the total number of different classes in the learning set. |
| f | A two-sided formula, if X is a data.frame. The left part correspond to class labels, the right to variables. |
| learnind | An index vector specifying the observations that belong to the learning set. Must not be missing for this method. |
| beta | Slope parameter for the logistic function which is used for the computation of class probabilities. The default value (1) need not produce reasonable results and can produce warnings. |
| k | Number of nearest neighbours to use. |
| | Currently unused argument. |

Details

The algorithm is as follows:

- Determine the k nearest neighbours
- For each class represented among these, compute the average euclidean distance.
- The negative distances are plugged into the logistic function with parameter beta.
- Classify into the class with highest probability.

Value

An object of class cloutput.

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

See Also

```
compBoostCMA, dldaCMA, ElasticNetCMA, fdaCMA, flexdaCMA, gbmCMA, knnCMA,
ldaCMA,LassoCMA,nnetCMA,plrCMA,pls_ldaCMA,pls_lrCMA,pls_rfCMA,pnnCMA,
qdaCMA,rfCMA,scdaCMA,shrinkldaCMA,svmCMA
```

Planarplot

Examples

```
### load Golub AML/ALL data
data(golub)
### extract class labels
golubY <- golub[,1]</pre>
### extract gene expression from first 10 genes
golubX <- as.matrix(golub[,-1])</pre>
### select learningset
ratio <- 2/3
set.seed(111)
learnind <- sample(length(golubY), size=floor(ratio*length(golubY)))</pre>
### run probabilistic k-nearest neighbours
result <- pknnCMA(X=golubX, y=golubY, learnind=learnind, k = 3)</pre>
### show results
show(result)
ftable(result)
plot(result)
```

Planarplot-methods Visualize Separability of different classes

Description

Given two variables, the methods trains a classifier (argument classifier) based on these two variables and plots the resulting class regions, learning- and test observations in the plane.

Appropriate variables are usually found by GeneSelection.

Methods

X = "matrix", y = "numeric", f = "missing" signature 1 X = "matrix", y = "factor", f = "missing" signature 2 X = "data.frame", y = "missing", f = "formula" signature 3 X = "ExpressionSet", y = "character", f = "missing" signature 4 For further argument and output information, consult Planarplot.

PlanarplotVisualize Separability of different classes

Description

Given two variables, the methods trains a classifier (argument classifier) based on these two variables and plots the resulting class regions, learning- and test observations in the plane.

Appropriate variables are usually found by GeneSelection.

For S4 method information, s. Planarplot-methods.

Usage

Planarplot(X, y, f, learnind, predind, classifier, gridsize = 100, ...)

50

Planarplot

| Arguments | |
|------------|--|
| Х | Gene expression data. Can be one of the following: |
| | • A matrix. Rows correspond to observations, columns to variables. |
| | • A data.frame, when f is not missing (s. below). |
| | • An object of class ExpressionSet. |
| У | Class labels. Can be one of the following: |
| | • A numeric vector. |
| | • A factor. |
| | • A character if X is an ExpressionSet that specifies the phenotype variable. |
| | • missing, if X is a data.frame and a proper formula f is provided. |
| f | A two-sided formula, if X is a data.frame. The left part correspond to class labels, the right to variables. |
| learnind | An index vector specifying the observations that belong to the learning set. May be missing; in that case, the learning set consists of all observations and predictions are made on the learning set. |
| predind | A vector containing <i>exactly</i> two indices that denote the two variables used for classification. |
| classifier | Name of function ending with CMA indicating the classifier to be used. |
| gridsize | The gridsize used for two-dimensional plotting. |
| | For both variables specified in predind, an equidistant grid of size gridsize is created. The resulting two grids are then combined to obtain gridsize^2 points in the real plane which are used to draw the class regions. Defaults to 100 which is usually a reasonable choice, but takes some time. |
| ••• | Further argument passed to classifier. |

Value

No return.

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>. Idea is from the MLInterfaces package, contributed by Jess Mar, Robert Gentleman and Vince Carey.

See Also

GeneSelection, compBoostCMA, dldaCMA, ElasticNetCMA, fdaCMA, flexdaCMA, gbmCMA, knnCMA, ldaCMA, LassoCMA, nnetCMA, pknnCMA, plrCMA, pls_ldaCMA, pls_lrCMA, pls_rfCMA, pnnCMA, qdaCMA, rfCMA, scdaCMA, shrinkldaCMA, svmCMA

Examples

```
### simple linear discrimination for the golub data:
data(golub)
golubY <- golub[,1]
golubX <- as.matrix(golub[,-1])
golubn <- nrow(golubX)</pre>
```

plot

Probability plot

Description

A popular way of visualizing the output of classifier is to plot, separately for each class, the predicted probability of each predicted observations for the respective class. For this purpose, the plot area is divided into K parts, where K is the number of classes. Predicted observations are assigned, according to their true class, to one of those parts. Then, for each part and each predicted observation, the predicted probabilities are plotted, displayed by coloured dots, where each colour corresponds to one class.

Arguments

| Х | An object of class cloutput whose slot probmatrix does not contain any |
|------|---|
| | missing value, i.e. probability estimations are provided by the classifier. |
| main | A title for the plot (character). |

Value

No return.

Note

The plot usually only makes sense if a sufficiently large numbers of observations has been classified. This is usually achieved by running the classifier on several learningsets with the method classification. The output can then be processed via join to obtain an object of class cloutput to which this method can be applied.

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

References

Slawski, M. Daumer, M. Boulesteix, A.-L. (2008) CMA - A comprehensive Bioconductor package for supervised classification with high dimensional data. *BMC Bioinformatics 9: 439*

See Also

cloutput

52

Barplot

Description

This method can be seen as a visual pendant to toplist. The plot visualizes variable importance by a barplot. The height of the barplots correspond to variable importance. What variable importance exactly means depends on the method chosen when calling GeneSelection, s. genesel.

Arguments

| х | An object of class genesel |
|------|--|
| top | Number of top genes whose variable importance should be displayed. Defaults to 10. |
| iter | Iteration number (learningset) for which variable importance should be displayed. |
| | Further graphical options passed to barplot. |

Value

No return.

Note

Note the following

- If scheme = "multiclass", only one plot will be made. Otherwise, one plot will be made for each binary scenario (depending on whether "scheme" is "one-vs-all" or "pairwise").
- Variable importance do not make sense for variable selection (ranking) methods that are essentially discrete, such as the Wilcoxon-Rank sum statistic or the Kruskal-Wallis statistic.
- For the methods "lasso", "elasticnet", "boosting" the number of nonzero coefficients can be very small, resulting in bars of height zero if top has been chosen too large.

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

References

Slawski, M. Daumer, M. Boulesteix, A.-L. (2008) CMA - A comprehensive Bioconductor package for supervised classification with high dimensional data. *BMC Bioinformatics 9: 439*

See Also

genesel, GeneSelection, toplist

plot tuningresult Visualize results of tuning

Description

After hyperparameter tuning using tune it is useful to see which choice of hyperparameters is suitable and how good the performance is.

Arguments

| Х | An object of class tuningresult. |
|-------|---|
| iter | Iteration number (learningset) for which tuning results should be displayed. |
| which | Character vector (maximum length is two) naming the arguments for which tun- ing results should be display. Default is NULL; if the number of tuned hyperpa- rameter is less or equal than two, then the results for these hyperparameters will be plotted. If this number is two, then a contour plot will be made, otherwise a simple line segment plot. If the number of tuned hyperparameters exceeds two, then which may not be NULL. |
| | Further graphical options passed either to plot or contour. |

Value

no return.

Note

Frequently, several hyperparameter (combinations) perform "best", s. also the remark in best.

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

References

Slawski, M. Daumer, M. Boulesteix, A.-L. (2008) CMA - A comprehensive Bioconductor package for supervised classification with high dimensional data. *BMC Bioinformatics 9: 439*

See Also

tune, tuningresult

plrCMA-methods L2 penalized logistic regression

Description

High dimensional logistic regression combined with an L2-type (Ridge-)penalty. Multiclass case is also possible.

Methods

X = "matrix", y = "numeric", f = "missing" signature 1
X = "matrix", y = "factor", f = "missing" signature 2
X = "data.frame", y = "missing", f = "formula" signature 3
X = "ExpressionSet", y = "character", f = "missing" signature 4
For further argument and output information, consult plrCMA.

plrCMA

L2 penalized logistic regression

Description

High dimensional logistic regression combined with an L2-type (Ridge-)penalty. Multiclass case is also possible. For S4 method information, see plrCMA-methods

Usage

plrCMA(X, y, f, learnind, lambda = 0.01, scale = TRUE, ...)

Arguments

Х

Gene expression data. Can be one of the following:

- A matrix. Rows correspond to observations, columns to variables.
- A data.frame, when f is not missing (s. below).
- An object of class ExpressionSet.

Class labels. Can be one of the following:

У

| ٠ | А | nume | ri | С | vector. |
|---|---|------|----|---|---------|
| ٠ | А | nume | ri | С | vector. |

- A factor.
- A character if X is an ExpressionSet that specifies the phenotype variable.
- missing, if X is a data.frame and a proper formula f is provided.

WARNING: The class labels will be re-coded to range from 0 to K-1, where K is the total number of different classes in the learning set.

f A two-sided formula, if X is a data.frame. The left part correspond to class labels, the right to variables.

| learnind | An index vector specifying the observations that belong to the learning set. May be missing; in that case, the learning set consists of all observations and predictions are made on the learning set. |
|----------|--|
| lambda | Parameter governing the amount of penalization. This hyperparameter should be tuned. |
| scale | Scale the predictors as specified by X to have unit variance and zero mean. |
| | Currently unused argument. |

Value

An object of class cloutput.

Author(s)

Special thanks go to

Ji Zhu (University of Ann Arbor, Michigan)

Trevor Hastie (Stanford University)

who provided the basic code that was then adapted by

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>.

References

Zhu, J., Hastie, T. (2004). Classification of gene microarrays by penalized logistic regression. Biostatistics 5:427-443.

See Also

compBoostCMA, dldaCMA, ElasticNetCMA, fdaCMA, flexdaCMA, gbmCMA, knnCMA, ldaCMA, LassoCMA, nnetCMA, pknnCMA, pls_ldaCMA, pls_lrCMA, pls_rfCMA, pnnCMA, qdaCMA, rfCMA, scdaCMA, shrinkldaCMA, svmCMA

Examples

```
### load Golub AML/ALL data
data(golub)
### extract class labels
golubY <- golub[,1]</pre>
### extract gene expression from first 10 genes
golubX <- as.matrix(golub[,-1])</pre>
### select learningset
ratio <- 2/3
set.seed(111)
learnind <- sample(length(golubY), size=floor(ratio*length(golubY)))</pre>
### run penalized logistic regression (no tuning)
plrresult <- plrCMA(X=golubX, y=golubY, learnind=learnind)</pre>
### show results
show(plrresult)
ftable(plrresult)
plot (plrresult)
### multiclass example:
### load Khan data
```

pls_ldaCMA-methods

```
data(khan)
### extract class labels
khanY <- khan[,1]
### extract gene expression from first 10 genes
khanX <- as.matrix(khan[,-1])
### select learningset
set.seed(111)
learnind <- sample(length(khanY), size=floor(ratio*length(khanY)))
### run penalized logistic regression (no tuning)
plrresult <- plrCMA(X=khanX, y=khanY, learnind=learnind)
### show results
show(plrresult)
ftable(plrresult)
plot(plrresult)</pre>
```

pls_ldaCMA-methods Partial Least Squares combined with Linear Discriminant Analysis

Description

-This method constructs a classifier that extracts Partial Least Squares components that are plugged into Linear Discriminant Analysis. The Partial Least Squares components are computed by the package plsgenomics.

Methods

X = "matrix", y = "numeric", f = "missing" signature 1 X = "matrix", y = "factor", f = "missing" signature 2 X = "data.frame", y = "missing", f = "formula" signature 3 X = "ExpressionSet", y = "character", f = "missing" signature 4 For further argument and output information, consult pls_ldaCMA.

pls_ldaCMA Partial Least Squares combined with Linear Discriminant Analysis

Description

This method constructs a classifier that extracts Partial Least Squares components that are plugged into Linear Discriminant Analysis. The Partial Least Squares components are computed by the package plsgenomics.

For S4 method information, see pls_ldaCMA-methods.

Usage

```
pls_ldaCMA(X, y, f, learnind, comp = 2, plot = FALSE)
```

| Arguments | |
|-----------|--|
| Х | Gene expression data. Can be one of the following: |
| | • A matrix. Rows correspond to observations, columns to variables. |
| | • A data.frame, when f is <i>not</i> missing (s. below). |
| | • An object of class ExpressionSet. |
| У | Class labels. Can be one of the following: |
| | • A numeric vector. |
| | • A factor. |
| | • A character if X is an ExpressionSet that specifies the phenotype variable. |
| | • missing, if X is a data.frame and a proper formula f is provided. |
| | WARNING : The class labels will be re-coded to range from 0 to $K-1$, where K is the total number of different classes in the learning set. |
| f | A two-sided formula, if X is a data.frame. The left part correspond to class labels, the right to variables. |
| learnind | An index vector specifying the observations that belong to the learning set. May be missing; in that case, the learning set consists of all observations and predictions are made on the learning set. |
| comp | Number of Partial Least Squares components to extract. Default is 2 which can be suboptimal, depending on the particular dataset. Can be optimized using tune. |
| plot | If comp <= 2, should the classification space of the Partial Least Squares components be plotted ? Default is FALSE. |

Value

An object of class cloutput.

Author(s)

Martin Slawski <ms@cs.uni-sb.de> Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

References

Nguyen, D., Rocke, D. M., (2002).

Tumor classification by partial least squares using microarray gene expression data.

Bioinformatics 18, 39-50

Boulesteix, A.L., Strimmer, K. (2007).

Partial least squares: a versatile tool for the analysis of high-dimensional genomic data. Briefings in Bioinformatics 7:32-44.

See Also

```
compBoostCMA, dldaCMA, ElasticNetCMA, fdaCMA, flexdaCMA, gbmCMA, knnCMA,
ldaCMA,LassoCMA,nnetCMA,pknnCMA,plrCMA,pls_ldaCMA,pls_lrCMA,pls_rfCMA,
pnnCMA, qdaCMA, rfCMA, scdaCMA, shrinkldaCMA, svmCMA
```

onte A

pls_lrCMA-methods

Examples

```
### load Khan data
data(khan)
### extract class labels
khanY <- khan[,1]
### extract gene expression
khanX <- as.matrix(khan[,-1])
### select learningset
set.seed(111)
learnind <- sample(length(khanY), size=floor(2/3*length(khanY)))
### run Shrunken Centroids classfier, without tuning
plsresult <- pls_ldaCMA(X=khanX, y=khanY, learnind=learnind, comp = 4)
### show results
show(plsresult)
ftable(plsresult)
plot(plsresult)</pre>
```

pls_lrCMA-methods Partial Least Squares followed by logistic regression

Description

This method constructs a classifier that extracts Partial Least Squares components that form the the covariates in a binary logistic regression model. The Partial Least Squares components are computed by the package plsgenomics.

Methods

X = "matrix", y = "numeric", f = "missing" signature 1 X = "matrix", y = "factor", f = "missing" signature 2 X = "data.frame", y = "missing", f = "formula" signature 3 X = "ExpressionSet", y = "character", f = "missing" signature 4 For further argument and output information, consult pls_lrCMA

pls_lrCMA Partial Least Squares followed by logistic regression

Description

This method constructs a classifier that extracts Partial Least Squares components that form the the covariates in a binary logistic regression model. The Partial Least Squares components are computed by the package plsgenomics.

For S4 method information, see pls_lrCMA-methods.

Usage

pls_lrCMA(X, y, f, learnind, comp = 2, lambda = 1e-4, plot = FALSE)

Arguments

| Х | Gene expression data. Can be one of the following: |
|----------|--|
| | A matrix. Rows correspond to observations, columns to variables. A data.frame, when f is <i>not</i> missing (s. below). An object of class ExpressionSet. |
| У | Class labels. Can be one of the following: |
| 1 | A numeric vector. A factor. A character if X is an ExpressionSet that specifies the phenotype variable. |
| | • missing, if X is a data.frame and a proper formula f is provided. |
| | WARNING : The class labels will be re-coded to range from 0 to $K-1$, where K is the total number of different classes in the learning set. |
| f | A two-sided formula, if X is a data.frame. The left part correspond to class labels, the right to variables. |
| learnind | An index vector specifying the observations that belong to the learning set. May be missing; in that case, the learning set consists of all observations and predictions are made on the learning set. |
| comp | Number of Partial Least Squares components to extract. Default is 2 which can be suboptimal, depending on the particular dataset. Can be optimized using tune. |
| lambda | Parameter controlling the amount of L2 penalization for logistic regression, usu- ally taken to be a small value in order to stabilize estimation in the case of sepa- rable data. |
| plot | If comp <= 2, should the classification space of the Partial Least Squares components be plotted ? Default is FALSE. |

Value

An object of class cloutput.

Note

Up to now, only the two-class case is supported.

Author(s)

```
Martin Slawski <ms@cs.uni-sb.de>
```

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

References

Boulesteix, A.L., Strimmer, K. (2007).

Partial least squares: a versatile tool for the analysis of high-dimensional genomic data. *Briefings in Bioinformatics* 7:32-44.

pls_rfCMA-methods

See Also

```
compBoostCMA, dldaCMA, ElasticNetCMA, fdaCMA, flexdaCMA, gbmCMA, knnCMA, ldaCMA, LassoCMA, nnetCMA, pknnCMA, plrCMA, pls_ldaCMA, pls_rfCMA, pnnCMA, qdaCMA, rfCMA, scdaCMA, shrinkldaCMA, svmCMA
```

Examples

```
### load Golub AML/ALL data
data(golub)
### extract class labels
golubY <- golub[,1]</pre>
### extract gene expression
golubX <- as.matrix(golub[,-1])</pre>
### select learningset
ratio <- 2/3
set.seed(111)
learnind <- sample(length(golubY), size=floor(ratio*length(golubY)))</pre>
### run PLS, combined with logistic regression
result <- pls_lrCMA(X=golubX, y=golubY, learnind=learnind)</pre>
### show results
show(result)
ftable(result)
plot(result)
```

pls_rfCMA-methods Partial Least Squares followed by random forests

Description

This method constructs a classifier that extracts Partial Least Squares components used to generate Random Forests, s. rfCMA. The Partial Least Squares components are computed by the package plsgenomics.

Methods

X = "matrix", **y** = "numeric", **f** = "missing" signature 1

X = "matrix", y = "factor", f = "missing" signature 2

X = "data.frame", **y** = "missing", **f** = "formula" signature 3

X = "ExpressionSet", y = "character", f = "missing" signature 4

For further argument and output information, consult pls_rfCMA.

pls_rfCMA

Description

This method constructs a classifier that extracts Partial Least Squares components used to generate Random Forests, s. rfCMA.

For S4 method information, see pls_rfCMA-methods.

Usage

pls_rfCMA(X, y, f, learnind, comp = 2 * nlevels(as.factor(y)), seed = 111, ...)

Arguments

| Х | Gene expression data. Can be one of the following: |
|----------|--|
| | A matrix. Rows correspond to observations, columns to variables. A data.frame, when f is <i>not</i> missing (s. below). An object of class ExpressionSet. |
| У | Class labels. Can be one of the following: |
| | • A numeric vector. |
| | • A factor. |
| | • A character if X is an ExpressionSet that specifies the phenotype variable. |
| | • missing, if X is a data.frame and a proper formula f is provided. |
| | WARNING : The class labels will be re-coded to range from 0 to $K-1$, where K is the total number of different classes in the learning set. |
| f | A two-sided formula, if X is a data.frame. The left part correspond to class labels, the right to variables. |
| learnind | An index vector specifying the observations that belong to the learning set. May be missing; in that case, the learning set consists of all observations and predictions are made on the learning set. |
| comp | Number of Partial Least Squares components to extract. Default ist two times the number of different classes. |
| seed | Fix Random number generator seed to seed. This is useful to guarantee repro- ducibility of the results, due to the random component in the random Forest. |
| •••• | Further arguments to be passed to randomForests from the package of the same name. |

Value

An object of class cloutput.

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

References

Boulesteix, A.L., Strimmer, K. (2007).

Partial least squares: a versatile tool for the analysis of high-dimensional genomic data.

Briefings in Bioinformatics 7:32-44.

See Also

```
compBoostCMA, dldaCMA, ElasticNetCMA, fdaCMA, flexdaCMA, gbmCMA, knnCMA, ldaCMA, LassoCMA, nnetCMA, pknnCMA, plrCMA, pls_ldaCMA, pls_lrCMA, pnnCMA, qdaCMA, rfCMA, scdaCMA, shrinkldaCMA, svmCMA
```

Examples

```
### load Golub AML/ALL data
data(golub)
### extract class labels
golubY <- golub[,1]</pre>
### extract gene expression
golubX <- as.matrix(golub[,-1])</pre>
### select learningset
ratio <- 2/3
set.seed(111)
learnind <- sample(length(golubY), size=floor(ratio*length(golubY)))</pre>
### run PLS, combined with Random Forest
result <- pls_rfCMA(X=golubX, y=golubY, learnind=learnind)</pre>
### show results
show(result)
ftable(result)
plot(result)
```

pnnCMA-methods Probabilistic Neural Networks

Description

Probabilistic Neural Networks is the term Specht (1990) used for a Gaussian kernel estimator for the conditional class densities.

Methods

X = "matrix", **y** = "numeric", **f** = "missing" signature 1

X = "matrix", y = "factor", f = "missing" signature 2

X = "data.frame", **y** = "missing", **f** = "formula" signature 3

X = "ExpressionSet", y = "character", f = "missing" signature 4

For references, further argument and output information, consult pnnCMA.

pnnCMA

Description

Probabilistic Neural Networks is the term Specht (1990) used for a Gaussian kernel estimator for the conditional class densities.

For S4 method information, see pnnCMA-methods.

Usage

pnnCMA(X, y, f, learnind, sigma = 1)

Arguments

| Х | Gene expression data. Can be one of the following: |
|----------|--|
| | • A matrix. Rows correspond to observations, columns to variables. |
| | • A data.frame, when f is <i>not</i> missing (s. below). |
| | • An object of class ExpressionSet. |
| | Each variable (gene) will be scaled for unit variance and zero mean. |
| У | Class labels. Can be one of the following: |
| | • A numeric vector. |
| | • A factor. |
| | • A character if X is an ExpressionSet that specifies the phenotype variable. |
| | • missing, if X is a data.frame and a proper formula f is provided. |
| | WARNING : The class labels will be re-coded to range from 0 to $K-1$, where K is the total number of different classes in the learning set. |
| f | A two-sided formula, if X is a data.frame. The left part correspond to class labels, the right to variables. |
| learnind | An index vector specifying the observations that belong to the learning set. For this method, this must <i>not</i> be missing. |
| sigma | Standard deviation of the Gaussian Kernel used. |
| | This hyperparameter should be tuned, s. tune. The default is 1, but this gener- ally does not lead to good results. Actually, this method reacts very sensitively to the value of sigma. Take care if warnings appear related to the particular choice. |

Value

An object of class cloutput.

Note

There is actually no strong relation of this method to Feed-Forward Neural Networks, s. nnetCMA.

qdaCMA-methods

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

References

Specht, D.F. (1990).

Probabilistic Neural Networks. Neural Networks, 3, 109-118.

See Also

```
compBoostCMA, dldaCMA, ElasticNetCMA, fdaCMA, flexdaCMA, gbmCMA, knnCMA,
ldaCMA,LassoCMA,nnetCMA,pknnCMA,plrCMA,pls_ldaCMA,pls_lrCMA,pls_rfCMA,
qdaCMA,rfCMA,scdaCMA,shrinkldaCMA,svmCMA
```

Examples

```
### load Golub AML/ALL data
data(golub)
### extract class labels
golubY <- golub[,1]</pre>
### extract gene expression from first 10 genes
golubX <- as.matrix(golub[,2:11])</pre>
### select learningset
ratio <- 2/3
set.seed(111)
learnind <- sample(length(golubY), size=floor(ratio*length(golubY)))</pre>
### run PNN
pnnresult <- pnnCMA(X=golubX, y=golubY, learnind=learnind, sigma = 3)</pre>
### show results
show(pnnresult)
ftable (pnnresult)
plot (pnnresult)
```

qdaCMA-methods Quadratic Discriminant Analysis

Description

Performs a quadratic discriminant analysis under the assumption of a multivariate normal distribution in each classes without restriction concerning the covariance matrices. The function qda from the package MASS is called for computation.

Methods

X = "matrix", y = "numeric", f = "missing" signature 1

X = "matrix", y = "factor", f = "missing" signature 2

X = "data.frame", **y** = "missing", **f** = "formula" signature 3

X = "ExpressionSet", y = "character", f = "missing" signature 4

For further argument and output information, consult qdaCMA.

qdaCMA

Description

Performs a quadratic discriminant analysis under the assumption of a multivariate normal distribution in each classes without restriction concerning the covariance matrices. The function qda from the package MASS is called for computation.

For S4 method information, see qdaCMA-methods.

Usage

qdaCMA(X, y, f, learnind, ...)

Arguments

| Х | Gene expression data. Can be one of the following: |
|----------|---|
| | • A matrix. Rows correspond to observations, columns to variables. |
| | • A data.frame, when f is <i>not</i> missing (s. below). |
| | • An object of class ExpressionSet. |
| У | Class labels. Can be one of the following: |
| | • A numeric vector. |
| | • A factor. |
| | • A character if X is an ExpressionSet that specifies the phenotype variable. |
| | • missing, if X is a data.frame and a proper formula f is provided. |
| | WARNING: The class labels will be re-coded to range from 0 to $K-1$, where K is the total number of different classes in the learning set. |
| f | A two-sided formula, if X is a data.frame. The left part correspond to class labels, the right to variables. |
| learnind | An index vector specifying the observations that belong to the learning set. May be missing; in that case, the learning set consists of all observations and pre- dictions are made on the learning set. |
| •••• | Further arguments to be passed to qda from the package <code>MASS</code> |

Value

An object of class cloutput.

Note

Excessive variable selection has usually to performed before qdaCMA can be applied in the p > n setting. Not reducing the number of variables can result in an error message.

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

rfCMA-methods

References

McLachlan, G.J. (1992).

Discriminant Analysis and Statistical Pattern Recognition. Wiley, New York

See Also

```
compBoostCMA, dldaCMA, ElasticNetCMA, fdaCMA, flexdaCMA, gbmCMA, knnCMA,
ldaCMA,LassoCMA,nnetCMA,pknnCMA,plrCMA,pls_ldaCMA,pls_lrCMA,pls_rfCMA,
pnnCMA,rfCMA,scdaCMA,shrinkldaCMA,svmCMA
```

Examples

```
### load Golub AML/ALL data
data(golub)
### extract class labels
golubY <- golub[,1]</pre>
### extract gene expression from first 3 genes
golubX <- as.matrix(golub[,2:4])</pre>
### select learningset
ratio <- 2/3
set.seed(112)
learnind <- sample(length(golubY), size=floor(ratio*length(golubY)))</pre>
### run QDA
qdaresult <- qdaCMA(X=golubX, y=golubY, learnind=learnind)</pre>
### show results
show(qdaresult)
ftable(qdaresult)
plot(qdaresult)
### multiclass example:
### load Khan data
data(khan)
### extract class labels
khanY <- khan[,1]</pre>
### extract gene expression from first 4 genes
khanX <- as.matrix(khan[,2:5])</pre>
### select learningset
set.seed(111)
learnind <- sample(length(khanY), size=floor(ratio*length(khanY)))</pre>
### run ODA
qdaresult <- qdaCMA(X=khanX, y=khanY, learnind=learnind)</pre>
### show results
show(qdaresult)
ftable(qdaresult)
plot(qdaresult)
```

rfCMA-methods Classification based on Random Forests

Description

Random Forests were proposed by Breiman (2001) and are implemented in the package randomForest. In this package, they can as well be used to rank variables according to their importance, s. GeneSelection.

Methods

X = "matrix", y = "numeric", f = "missing" signature 1
X = "matrix", y = "factor", f = "missing" signature 2
X = "data.frame", y = "missing", f = "formula" signature 3
X = "ExpressionSet", y = "character", f = "missing" signature 4
For references, further argument and output information, consult rfCMA

```
rfCMA
```

Classification based on Random Forests

Description

Random Forests were proposed by Breiman (2001) and are implemented in the package randomForest. In this package, they can as well be used to rank variables according to their importance, s. GeneSelection. For S4 method information, see rfCMA-methods

Usage

rfCMA(X, y, f, learnind, varimp = TRUE, seed = 111, ...)

Arguments

| Х | Gene expression data. Can be one of the following: |
|----------|--|
| | • A matrix. Rows correspond to observations, columns to variables. |
| | • A data.frame, when f is <i>not</i> missing (s. below). |
| | • An object of class ExpressionSet. |
| У | Class labels. Can be one of the following: |
| | • A numeric vector. |
| | • A factor. |
| | • A character if X is an ExpressionSet that specifies the phenotype variable. |
| | • missing, if X is a data.frame and a proper formula f is provided. |
| | WARNING : The class labels will be re-coded to range from 0 to $K-1$, where K is the total number of different classes in the learning set. |
| f | A two-sided formula, if X is a data.frame. The left part correspond to class labels, the right to variables. |
| learnind | An index vector specifying the observations that belong to the learning set. May be missing; in that case, the learning set consists of all observations and predictions are made on the learning set. |
| varimp | Should variable importance measures be computed ? Defauls to TRUE. |
| seed | Fix Random number generator seed to seed. This is useful to guarantee repro- ducibility of the results. |
| | Further arguments to be passed to randomForest from the package of the same name. |

68

roc

Value

If varimp, then an object of class clvarseloutput is returned, otherwise an object of class cloutput

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

References

Breiman, L. (2001) Random Forest. *Machine Learning, 45:5-32.*

See Also

compBoostCMA, dldaCMA, ElasticNetCMA, fdaCMA, flexdaCMA, gbmCMA, knnCMA, ldaCMA,LassoCMA,nnetCMA,pknnCMA,plrCMA,pls_ldaCMA,pls_lrCMA,pls_rfCMA, pnnCMA,qdaCMA,scdaCMA,shrinkldaCMA,svmCMA

Examples

```
### load Khan data
data(khan)
### extract class labels
khanY <- khan[,1]
### extract gene expression
khanX <- as.matrix(khan[,-1])
### select learningset
set.seed(111)
learnind <- sample(length(khanY), size=floor(2/3*length(khanY)))
### run random Forest
rfresult <- rfCMA(X=khanX, y=khanY, learnind=learnind, varimp = FALSE)
### show results
show(rfresult)
ftable(rfresult)
plot(rfresult)</pre>
```

roc

Receiver Operator Characteristic

Description

The empirical Receiver Operator Characteristic (ROC) is widely used for the evaluation of diagnostic tests, but also for the evaluation of classfiers. In this implementation, it can only be used for the binary classification case. The input are a numeric vector of class probabilities (which play the role of a test result) and the true class labels. Note that misclassification performance can (partly widely) differ from the Area under the ROC (AUC). This is due to the fact that misclassification rates are always computed for the threshold 'probability = 0.5'.

Arguments

| object | An object of cloutput. |
|--------|--|
| plot | Should the ROC curve be plotted ? Default is TRUE. |
| | Argument to specifiy further graphical options. |

Value

The empirical area under the curve (AUC).

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

References

Slawski, M. Daumer, M. Boulesteix, A.-L. (2008) CMA - A comprehensive Bioconductor package for supervised classification with high dimensional data. *BMC Bioinformatics 9: 439*

See Also

evaluation

scdaCMA-methods Shrunken Centroids Discriminant Analysis

Description

The nearest shrunken centroid classification algorithm is detailly described in Tibshirani et al. (2002).

It is widely known under the name PAM (prediction analysis for microarrays), which can also be found in the package pamr.

Methods

X = "matrix", y = "numeric", f = "missing" signature 1

X = "matrix", y = "factor", f = "missing" signature 2

X = "data.frame", **y** = "missing", **f** = "formula" signature 3

X = "ExpressionSet", y = "character", f = "missing" signature 4

For references, further argument and output information, consult scdaCMA.

scdaCMA

Description

The nearest shrunken centroid classification algorithm is detailly described in Tibshirani et al. (2002).

It is widely known under the name PAM (prediction analysis for microarrays), which can also be found in the package pamr.

For S4 method information, see scdaCMA-methods.

Usage

scdaCMA(X, y, f, learnind, delta = 0.5, ...)

Arguments

| Х | Gene expression data. Can be one of the following: |
|----------|--|
| | • A matrix. Rows correspond to observations, columns to variables. |
| | • A data.frame, when f is <i>not</i> missing (s. below). |
| | • An object of class ExpressionSet. |
| У | Class labels. Can be one of the following: |
| | • A numeric vector. |
| | • A factor. |
| | • A character if X is an ExpressionSet that specifies the phenotype variable. |
| | • missing, if X is a data.frame and a proper formula f is provided. |
| | WARNING: The class labels will be re-coded to range from 0 to $K-1$, where K is the total number of different classes in the learning set. |
| f | A two-sided formula, if X is a data.frame. The left part correspond to class labels, the right to variables. |
| learnind | An index vector specifying the observations that belong to the learning set. May be missing; in that case, the learning set consists of all observations and predictions are made on the learning set. |
| delta | The shrinkage intensity for the class centroids - a hyperparameter that must be tuned. The default 0.5 not necessarily produces good results. |
| | Currently unused argument. |

Value

An object of class cloutput.

Note

The results can differ from those obtained by using the package pamr.

Author(s)

Martin Slawski <ms@cs.uni-sb.de> Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

References

Tibshirani, R., Hastie, T., Narasimhan, B., and Chu, G., (2003). Class prediction by nearest shrunken centroids with applications to DNA microarrays. *Statistical Science*, *18*, *104-117*

See Also

```
compBoostCMA, dldaCMA, ElasticNetCMA, fdaCMA, flexdaCMA, gbmCMA, knnCMA,
ldaCMA,LassoCMA,nnetCMA,pknnCMA,plrCMA,pls_ldaCMA,pls_lrCMA,pls_rfCMA,
pnnCMA,qdaCMA,rfCMA,shrinkldaCMA,svmCMA
```

Examples

```
### load Khan data
data(khan)
### extract class labels
khanY <- khan[,1]
### extract gene expression
khanX <- as.matrix(khan[,-1])
### select learningset
set.seed(111)
learnind <- sample(length(khanY), size=floor(2/3*length(khanY)))
### run Shrunken Centroids classfier, without tuning
scdaresult <- scdaCMA(X=khanX, y=khanY, learnind=learnind)
### show results
show(scdaresult)
ftable(scdaresult)
plot(scdaresult)</pre>
```

shrinkldaCMA-methods

Shrinkage linear discriminant analysis

Description

Linear Discriminant Analysis combined with the James-Stein-Shrinkage approach of Schaefer and Strimmer (2005) for the covariance matrix.

Currently still an experimental version. For S4 method information, see shrinkldaCMA-methods

Methods

X = "matrix", y = "numeric", f = "missing" signature 1

X = "matrix", y = "factor", f = "missing" signature 2

X = "data.frame", **y** = "missing", **f** = "formula" signature 3

X = "ExpressionSet", y = "character", f = "missing" signature 4

For further argument and output information, consult shrinkldaCMA.

72

shrinkldaCMA

Description

Linear Discriminant Analysis combined with the James-Stein-Shrinkage approach of Schaefer and Strimmer (2005) for the covariance matrix.

Currently still an experimental version.

For S4 method information, see shrinkldaCMA-methods

Usage

shrinkldaCMA(X, y, f, learnind, ...)

Arguments

| Х | Gene expression data. Can be one of the following: |
|----------|--|
| | A matrix. Rows correspond to observations, columns to variables. A data.frame, when f is <i>not</i> missing (s. below). |
| | • An object of class ExpressionSet. |
| У | Class labels. Can be one of the following: |
| | • A numeric vector. |
| | • A factor. |
| | • A character if X is an ExpressionSet that specifies the phenotype variable. |
| | • missing, if X is a data.frame and a proper formula f is provided. |
| | WARNING : The class labels will be re-coded to range from 0 to $K-1$, where K is the total number of different classes in the learning set. |
| f | A two-sided formula, if X is a data.frame. The left part correspond to class labels, the right to variables. |
| learnind | An index vector specifying the observations that belong to the learning set. May be missing; in that case, the learning set consists of all observations and predictions are made on the learning set. |
| | Further arguments to be passed to cov.shrink from the package corpcor |

Value

An object of class cloutput.

Note

This is still an experimental version.

Covariance shrinkage is performed by calling functions from the package corpcor.

Variable selection is not necessary.

summary

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

References

Schaefer, J., Strimmer, K. (2005).

A shrinkage approach to large-scale covariance estimation and implications for functional genomics.

Statististical Applications in Genetics and Molecular Biology, 4:32.

See Also

```
compBoostCMA, dldaCMA, ElasticNetCMA, fdaCMA, flexdaCMA, gbmCMA, knnCMA, ldaCMA, LassoCMA, nnetCMA, pknnCMA, pls_ldaCMA, pls_lrCMA, pls_rfCMA, pnnCMA, qdaCMA, rfCMA, scdaCMA, svmCMA.
```

Examples

```
### load Golub AML/ALL data
data(golub)
### extract class labels
golubY <- golub[,1]</pre>
### extract gene expression
golubX <- as.matrix(golub[,-1])</pre>
### select learningset
ratio <- 2/3
set.seed(111)
learnind <- sample(length(golubY), size=floor(ratio*length(golubY)))</pre>
### run shrinkage-LDA
result <- shrinkldaCMA(X=golubX, y=golubY, learnind=learnind)</pre>
### show results
show(result)
ftable(result)
plot(result)
```

summary

Summarize classifier evaluation

Description

This method principally does nothing more than applying the pre-implemented summary () function to the slot score of an object of class evaloutput. One then obtains the usual five-pointsummary, consisting of minimum and maximum, lower and upper quartile and the median. Additionally, the mean is also shown.

Arguments

| object | An object of class evaloutput. |
|--------|---|
| | Further arguments passed to the pre-implemented summary function. |

svmCMA-methods

Value

No return.

Note

That the results normally differ for different evaluation schemes ("iterationwise" or "observationwise").

Author(s)

```
Martin Slawski <ms@cs.uni-sb.de>
```

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

See Also

evaluation, compare, obsinfo.

svmCMA-methods Support Vector Machine

Description

Calls the function svm from the package e1071 that provides an interface to the award-winning LIBSVM routines.

Methods

X = "matrix", y = "numeric", f = "missing" signature 1 X = "matrix", y = "factor", f = "missing" signature 2 X = "data.frame", y = "missing", f = "formula" signature 3 X = "ExpressionSet", y = "character", f = "missing" signature 4 For further argument and output information, consult svmCMA.

svmCMA

Support Vector Machine

Description

Calls the function svm from the package e1071 that provides an interface to the award-winning LIBSVM routines. For S4 method information, see svmCMA-methods

Usage

svmCMA(X, y, f, learnind, probability, ...)

Arguments

| Х | Gene expression data. Can be one of the following: |
|-------------|--|
| | • A matrix. Rows correspond to observations, columns to variables. |
| | • A data.frame, when f is <i>not</i> missing (s. below). |
| | • An object of class ExpressionSet. |
| У | Class labels. Can be one of the following: |
| | • A numeric vector. |
| | • A factor. |
| | • A character if X is an ExpressionSet that specifies the phenotype variable. |
| | • missing, if X is a data.frame and a proper formula f is provided. |
| | WARNING: The class labels will be re-coded to range from 0 to $K-1$, where K is the total number of different classes in the learning set. |
| f | A two-sided formula, if X is a data.frame. The left part correspond to class labels, the right to variables. |
| learnind | An index vector specifying the observations that belong to the learning set. May be missing; in that case, the learning set consists of all observations and predictions are made on the learning set. |
| probability | logical indicating whether the model should allow for probability predictions. |
| | Further arguments to be passed to svm from the package e1071 |

Value

An object of class cloutput.

Note

Contrary to the default settings in e1071:::svm, the used kernel is a linear kernel which has turned to be out a better default setting in the small sample, large number of predictors - situation, because additional nonlinearity is mostly not necessary there. It additionally avoids the tuning of a further kernel parameter gamma, s. help of the package e1071 for details.

Nevertheless, hyperparameter tuning concerning the parameter cost must usually be performed to obtain reasonale results, s. tune.

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

Christoph Bernau <bernau@ibe.med.uni-muenchen.de>

References

Boser, B., Guyon, I., Vapnik, V. (1992)
A training algorithm for optimal margin classifiers.
Proceedings of the fifth annual workshop on Computational learning theory, pages 144-152, ACM Press.
Change Chill, Change and Line Chill, Learn LIDSVM, a litera for Super AM, and the Marking Market and Super AM.

Chang, Chih-Chung and Lin, Chih-Jen : LIBSVM: a library for Support Vector Machines http: //www.csie.ntu.edu.tw/~cjlin/libsvm

Schoelkopf, B., Smola, A.J. (2002) Learning with kernels. *MIT Press, Cambridge, MA*.

toplist

See Also

```
compBoostCMA, dldaCMA, ElasticNetCMA, fdaCMA, flexdaCMA, gbmCMA, knnCMA,
ldaCMA,LassoCMA,nnetCMA,pknnCMA,plrCMA,pls_ldaCMA,pls_lrCMA,pls_rfCMA,
pnnCMA,qdaCMA,rfCMA,scdaCMA,shrinkldaCMA
```

Examples

```
### load Golub AML/ALL data
data(golub)
### extract class labels
golubY <- golub[,1]</pre>
### extract gene expression
golubX <- as.matrix(golub[,-1])</pre>
### select learningset
ratio <- 2/3
set.seed(111)
learnind <- sample(length(golubY), size=floor(ratio*length(golubY)))</pre>
### run _untuned_linear SVM
svmresult <- svmCMA(X=golubX, y=golubY, learnind=learnind,probability=TRUE)</pre>
### show results
show(svmresult)
ftable(svmresult)
plot(svmresult)
```

toplist

Display 'top' variables

Description

This is a convenient method to get quick access to the most important variables, based on the result of call to GeneSelection.

Usage

toplist(object, k = 10, iter = 1, show = TRUE, ...)

Arguments

| object | An object of genesel. |
|--------------|--|
| k | Number of top genes for which information should be displayed. Defaults to 10. |
| iter | teration number (learningset) for which tuning results should be displayed. |
| show | Should the results be printed ? Default is TRUE. |
| | Currently unused argument. |
| iter show | teration number (learningset) for which tuning results should be displayed Should the results be printed ? Default is TRUE. |

Value

The type of output depends on the gene selection scheme. For the multiclass case, if gene selection has been run with the "pairwise" or "one-vs-all" scheme, then the output will be a list of data.frames, each containing the gene indices plus variable importance for the top k genes. The list elements are named according to the binary scenarios (e.g., 1 vs. 3). Otherwise, a single data.frame is returned.

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

References

Slawski, M. Daumer, M. Boulesteix, A.-L. (2008) CMA - A comprehensive Bioconductor package for supervised classification with high dimensional data. *BMC Bioinformatics 9: 439*

See Also

genesel, GeneSelection, plot, genesel-method

tune-methods Hyperparameter tuning for classifiers

Description

Performs hyperparameter tuning for the following signatures:

Methods

X = "matrix", y = "numeric", f = "missing" signature 1 X = "matrix", y = "factor", f = "missing" signature 2 X = "data.frame", y = "missing", f = "formula" signature 3 X = "ExpressionSet", y = "character", f = "missing" signature 4 For further argument and output information, consult tune.

tune

Hyperparameter tuning for classifiers

Description

Most classifiers implemented in this package depend on one or even several hyperparameters (s. details) that should be optimized to obtain good (and comparable !) results. As tuning scheme, we propose three fold Cross-Validation on each learningset (for fixed selected variables). Note that learningsets usually do not contain the complete dataset, so tuning involves a second level of splitting the dataset. Increasing the number of folds leads to larger datasets (and possibly to higher accuracy), but also to higher computing times.

For S4 method information, s. link {tune-methods}

Usage

```
tune(X, y, f, learningsets, genesel, genesellist = list(), nbgene, classifier, f
```

Arguments

| • | |
|--------------|--|
| Х | Gene expression data. Can be one of the following: |
| | • A matrix. Rows correspond to observations, columns to variables. |
| | • A data.frame, when f is not missing (s. below). |
| | • An object of class ExpressionSet. |
| У | Class labels. Can be one of the following: |
| | • A numeric vector. |
| | • A factor. |
| | • A character if X is an ExpressionSet that specifies the phenotype variable. |
| | • missing, if X is a data.frame and a proper formula f is provided. |
| f | A two-sided formula, if X is a data.frame. The left part correspond to class labels, the right to variables. |
| learningsets | An object of class learningsets. May be missing, then the complete datasets is used as learning set. |
| genesel | Optional (but usually recommended) object of class genesel containing vari- able importance information for the argument learningsets |
| genesellist | In the case that the argument genesel is missing, this is an argument list passed to GeneSelection. If both genesel and genesellist are missing, no variable selection is performed. |
| nbgene | Number of best genes to be kept for classification, based on either genesel or the call to GeneSelection using genesellist. In the case that both are missing, this argument is not necessary. note : |
| | • If the gene selection method has been one of "lasso", "elasticnet", "boosting", nbgene will be reset to min(s, nbgene) where s is the number of nonzero coefficients. |
| | • if the gene selection scheme has been "one-vs-all", "pairwise" for the multiclass case, there exist several rankings. The top nbgene will be kept of <i>each</i> of them, so the number of effective used genes will sometimes be much larger. |
| classifier | Name of function ending with CMA indicating the classifier to be used. |
| fold | The number of cross-validation folds used within each learningset. Default is 3. Increasing fold will lead to higher computing times. |
| strat | Should stratified cross-validation according to the class proportions in the com- plete dataset be used ? Default is FALSE. |
| grids | A named list. The names correspond to the arguments to be tuned, e.g. k (the number of nearest neighbours) for knnCMA, or cost for svmCMA. Each element is a numeric vector defining the grid of candidate values. Of course, several hyperparameters can be tuned simultaneously (though requiring much time). By default, grids is an empty list. In that case, a pre-defined list will be used, s. details. |
| trace | Should progress be traced ? Default is TRUE. |
| | Further arguments to be passed to classifier, of course not one of the arguments to be tuned (!). |

Details

The following default settings are used, if the arguments grids is an empty list:

```
gbmCMA n.trees = c(50, 100, 200, 500, 1000)
compBoostCMA mstop = c(50, 100, 200, 500, 1000)
LassoCMA norm.fraction = seg(from=0.1, to=0.9, length=9)
ElasticNetCMA norm.fraction = seg(from=0.1, to=0.9, length=5), lambda2
   = 2^{(5:1)}
plrCMA lambda = 2^{-4:4}
pls_ldaCMA comp = 1:10
pls_lrCMA comp = 1:10
pls_rfCMA comp = 1:10
rfCMA mtry = ceiling(c(0.1, 0.25, 0.5, 1, 2) \times sqrt(ncol(X))), nodesize
   = c(1, 2, 3)
knnCMA k=1:10
pknnCMA k = 1:10
scdaCMA delta = c(0.1, 0.25, 0.5, 1, 2, 5)
pnnCMA sigma = c(2^{-2:2}),
nnetCMA size = 1:5, decay = c(0, 2^{(-(4:1))})
svmCMA,kernel = "linear" cost = c(0.1, 1, 5, 10, 50, 100, 500)
svmCMA, kernel = "radial" cost = c(0.1, 1, 5, 10, 50, 100, 500), gamma
   = 2^{\{-2:2\}}
svmCMA,kernel = "polynomial" cost = c(0.1, 1, 5, 10, 50, 100, 500),
   degree = 2:4
```

Value

An object of class tuningresult

Note

The computation time can be enormously high. Note that for each different learningset, the classifier must be trained fold times number of possible different hyperparameter combinations times. E.g. if the number of the learningsets is fifty, fold = 3 and two hyperparameters (each with 5 candidate values) are tuned, 50x3x25=3750 training iterations are necessary !

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

Christoph Bernau
dernau@ibe.med.uni-muenchen.de>

References

Slawski, M. Daumer, M. Boulesteix, A.-L. (2008) CMA - A comprehensive Bioconductor package for supervised classification with high dimensional data. *BMC Bioinformatics 9: 439*

tuningresult-class

See Also

tuningresult, GeneSelection, classification

Examples

```
## Not run:
### simple example for a one-dimensional grid, using compBoostCMA.
### dataset
data(golub)
golubY <- golub[,1]</pre>
golubX <- as.matrix(golub[,-1])</pre>
### learningsets
set.seed(111)
lset <- GenerateLearningsets(y=golubY, method = "CV", fold=5, strat =TRUE)</pre>
### tuning after gene selection with the t.test
tuneres <- tune(X = golubX, y = golubY, learningsets = lset,</pre>
              genesellist = list(method = "t.test"),
              classifier=compBoostCMA, nbgene = 100,
              grids = list(mstop = c(50, 100, 250, 500, 1000)))
### inspect results
show(tuneres)
best(tuneres)
plot(tuneres, iter = 3)
## End(Not run)
```

tuningresult-class "tuningresult"

Description

Object returned by the function tune

Slots

- hypergrid: A data.frame representing the grid of values that were tried and evaluated. The number of columns equals the number of tuned hyperparameters and the number rows equals the number of all possible combinations of the discrete grids.
- tuneres: A list whose lengths equals the number of different learningsets for which tuning has been performed and whose elements are numeric vectors with length equal to the number of rows of hypergrid (s.above), containing the misclassification rate belonging to the respective hyperparameter/hyperparameter combination. In order to to get an overview about the best hyperparameter/hyperparameter combination, use the convenience method best
- method: Name of the classifier that has been tuned.
- fold: Number of cross-validation fold used for tuning, s. argument of the same name in tune

Methods

show Use show (tuninresult-object) for brief information.

best Use best (tuningresult-object) to see which hyperparameter/hyperparameter combination has performed best in terms of the misclassification rate, s. best, tuningresultmethod plot Use plot(tuningresult-object, iter, which) to display the performance of hyperparameter/hyperparameter combinations graphically, either as one-dimensional or as two-dimensional (contour) plot, s. plot, tuningresult-method

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

See Also

tune

varseloutput-class "varseloutput"

Description

An object returned by the functions described in filter, usually not created directly by the user.

Slots

varsel: numeric vector of variable importance measures, e.g. absolute of genewise statistics.

Methods

No methods are currently defined.

Author(s)

Martin Slawski <ms@cs.uni-sb.de>

Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

See Also

filter, clvarseloutput

Index

*Topic **datasets** golub, 35 khan, 37 *Topic **multivariate** Barplot, 53 best,1 boxplot, 2 classification, 3 classification-methods, 2cloutput-class, 5 clvarseloutput-class, 6 CMA-package, 7 compare,8 compare-methods, 8compBoostCMA, 11 compBoostCMA-methods, 10 dldaCMA, 13 dldaCMA-methods, 13 ElasticNetCMA, 16 ElasticNetCMA-methods, 15 evaloutput-class, 18 evaluation, 19 evaluation-methods, 18fdaCMA, 21 fdaCMA-methods, 21 filter, 23 flexdaCMA, 24 flexdaCMA-methods, 24 ftable, 26 gbmCMA, 27 gbmCMA-methods, 27 GenerateLearningsets, 29 genesel-class, 31 GeneSelection, 32GeneSelection-methods, 32internals, 35 join, 36 join-methods, 36 knnCMA, 38 knnCMA-methods, 37 LassoCMA, 40 LassoCMA-methods, 39 ldaCMA, 42

ldaCMA-methods, 42 learningsets-class,44 nnetCMA, 45 nnetCMA-methods, 45 obsinfo,47 pknnCMA, 48 pknnCMA-methods, 48 Planarplot, 50 Planarplot-methods, 50 plot, 52 plot tuningresult, 54 plrCMA, 55 plrCMA-methods, 55 pls_ldaCMA, 57 pls_ldaCMA-methods, 57 pls_lrCMA, 59 pls_lrCMA-methods, 59 pls_rfCMA, 62 pls_rfCMA-methods, 61 pnnCMA, 64 pnnCMA-methods, 63qdaCMA, 66 gdaCMA-methods, 65 rfCMA.68 rfCMA-methods, 67 roc, 69 scdaCMA, 71 scdaCMA-methods, 70 shrinkldaCMA, 73 shrinkldaCMA-methods, 72 summary, 74 svmCMA, 75 svmCMA-methods, 75 toplist,77 tune,78 tune-methods, 78 tuningresult-class, 81 varseloutput-class, 82 Barplot, 53

best, 1, 54, 81
best, tuningresult-method, 81
best, tuningresult-method (best), 1
bklr (internals), 35

INDEX

bkreg(internals), 35 39, 41, 43, 46, 49, 51, 56, 58, 61, 63, 65, 67, 69, 72, 74, 77 boxplot, 2 boxplot, evaloutput-method, 18 dldaCMA, data.frame, missing, formula-method boxplot, evaloutput-method (dldaCMA-methods), 13 (boxplot), 2 dldaCMA, ExpressionSet, character, missing-method (dldaCMA-methods), 13 care.dev(internals), 35 dldaCMA, matrix, factor, missing-method care.exp(internals), 35 (dldaCMA-methods), 13 characterplot (internals), 35 dldaCMA, matrix, numeric, missing-method classification, 2, 3, 7, 9, 10, 20, 30, 34, (dldaCMA-methods), 13 36, 44, 52, 81 dldaCMA-methods, 13, 13 classification, data.frame, missing, formula-method (classification-methods), 2 ElasticNetCMA, 4, 6, 7, 12, 14, 15, 16, 22, classification, ExpressionSet, character, missi25, 28eff, 39, 41, 43, 46, 49, 51, 56, 58, 61, 63, 65, 67, 69, 72, 74, 77, 80 (classification-methods), 2 classification, matrix, factor, missing-medanta cNetCMA, data.frame, missing, formula-metho (classification-methods), 2 (ElasticNetCMA-methods), 15 classification, matrix, numeric, missing Endenshind NetCMA, ExpressionSet, character, missing-(classification-methods), 2 (ElasticNetCMA-methods), 15 classification-methods, 3ElasticNetCMA, matrix, factor, missing-method classification-methods, 2 (ElasticNetCMA-methods), 15 cloutput, 4, 6, 9, 14, 19, 22, 25, 26, 28, 36, ElasticNetCMA, matrix, numeric, missing-method 38, 43, 46, 49, 52, 56, 58, 60, 62, 64, (ElasticNetCMA-methods), 15 66, 69-71, 73, 76 ElasticNetCMA-methods, 15 evaloutput, 2, 20, 74 cloutput (cloutput-class), 5 cloutput-class, 5 evaloutput (evaloutput-class), 18 clvarseloutput, 4, 9, 12, 17, 19, 36, 69, evaloutput-class, 18 evaluation, 2, 4, 8-10, 18, 19, 26, 36, 47, 82 48, 70, 75 clvarseloutput (clvarseloutput-class), 6 evaluation, list-method clvarseloutput-class, 6(evaluation-methods), 18 CMA (CMA-package), 7 evaluation-methods, 19 CMA-package, 7 evaluation-methods, 18 compare, 8, 8, 20, 36, 75 fdaCMA, 4, 6, 7, 12, 14, 17, 21, 21, 22, 25, 28, compare, list-method 39, 41, 43, 46, 49, 51, 56, 58, 61, 63, (compare-methods), 8 65, 67, 69, 72, 74, 77 compare-methods,8fdaCMA, data.frame, missing, formula-method compare-methods, 8(fdaCMA-methods), 21 compBoostCMA, 4, 6, 7, 11, 11, 14, 17, 22, fdaCMA, ExpressionSet, character, missing-method 25, 28, 33, 39, 41, 43, 46, 49, 51, 56, 58, 61, 63, 65, 67, 69, 72, 74, 77, 80 (fdaCMA-methods), 21 compBoostCMA, data.frame, missing, formufdanetrix, factor, missing-method (fdaCMA-methods), 21 (compBoostCMA-methods), 10 compBoostCMA, ExpressionSet, character, mbscMag.maturized numeric, missing-method (fdaCMA-methods), 21 (compBoostCMA-methods), 10 compBoostCMA, matrix, factor, missing-methods, 21 fdaCMA-methods, 21 (compBoostCMA-methods), 10 compBoostCMA, matrix, numeric, missing-mftildter, 23, 34, 82 flexdaCMA, 4, 6, 7, 12, 14, 17, 22, 24, 24, (compBoostCMA-methods), 10 28, 39, 41, 43, 46, 49, 51, 56, 58, 61, compBoostCMA-methods, 11 63, 65, 67, 69, 72, 74, 77 compBoostCMA-methods, 10 flexdaCMA, data.frame, missing, formula-method dldaCMA, 4, 6, 7, 12, 13, 13, 17, 22, 25, 28, (flexdaCMA-methods), 24

INDEX

```
flexdaCMA, ExpressionSet, character, missinan, method
        (flexdaCMA-methods), 24
                                             knnCMA, 4, 6, 7, 12, 14, 17, 22, 25, 28, 37, 38,
                                                      41, 43, 46, 49, 51, 56, 58, 61, 63, 65,
flexdaCMA, matrix, factor, missing-method
                                                      67, 69, 72, 74, 77, 79, 80
        (flexdaCMA-methods), 24
flexdaCMA, matrix, numeric, missing-methdodnCMA, data.frame, missing, formula-method
        (flexdaCMA-methods), 24
                                                      (knnCMA-methods), 37
flexdaCMA-methods, 24
                                             knnCMA, ExpressionSet, character, missing-method
flexdaCMA-methods, 24
                                                      (knnCMA-methods), 37
ftable, 26
                                             knnCMA, matrix, factor, missing-method
ftable, cloutput-method, 5, 7
                                                      (knnCMA-methods), 37
ftable, cloutput-method (ftable),
                                             knnCMA, matrix, numeric, missing-method
        26
                                                      (knnCMA-methods), 37
ftest (filter), 23
                                             knnCMA-methods, 37, 38
                                             kruskaltest (filter), 23
gbmCMA, 4, 6, 7, 12, 14, 17, 22, 25, 27, 27, 39,
        41, 43, 46, 49, 51, 56, 58, 61, 63, 65,
                                             LassoCMA, 4, 6, 7, 12, 14, 17, 22, 25, 28, 33,
        67, 69, 72, 74, 77, 80
                                                      39, 40, 40, 43, 46, 49, 51, 56, 58, 61,
gbmCMA, data.frame, missing, formula-method
                                                      63, 65, 67, 69, 72, 74, 77, 80
        (gbmCMA-methods), 27
                                             LassoCMA, data.frame, missing, formula-method
gbmCMA, ExpressionSet, character, missing-method (LassoCMA-methods), 39
       (gbmCMA-methods), 27
                                             LassoCMA, ExpressionSet, character, missing-method
gbmCMA, matrix, factor, missing-method
                                                      (LassoCMA-methods), 39
        (gbmCMA-methods), 27
                                             LassoCMA, matrix, factor, missing-method
gbmCMA, matrix, numeric, missing-method
                                                      (LassoCMA-methods), 39
        (gbmCMA-methods), 27
                                             LassoCMA, matrix, numeric, missing-method
gbmCMA-methods, 27
                                                      (LassoCMA-methods), 39
gbmCMA-methods, 27
                                             LassoCMA-methods, 39
GenerateLearningsets, 7, 29, 34, 44
                                             ldaCMA, 4, 6, 7, 12, 14, 17, 22, 25, 28, 39, 41,
genesel, 3, 33, 53, 77-79
                                                      42, 42, 46, 49, 51, 56, 58, 61, 63, 65,
genesel (genesel-class), 31
                                                      67, 69, 72, 74, 77
genesel-class, 31
                                             ldaCMA, data.frame, missing, formula-method
GeneSelection, 3, 4, 7, 10, 11, 16, 23, 30,
                                                      (ldaCMA-methods), 42
        31, 32, 32, 39, 40, 44, 50, 53, 77–79,
                                              ldaCMA, ExpressionSet, character, missing-method
        81
                                                      (ldaCMA-methods), 42
GeneSelection, data.frame, missing, formula-method
IdaCMA, matrix, factor, missing-method
        (GeneSelection-methods), 32
                                                      (ldaCMA-methods), 42
GeneSelection, ExpressionSet, character, missing-method
IdaCMA, matrix, numeric, missing-method
        (GeneSelection-methods), 32
                                                      (ldaCMA-methods), 42
GeneSelection, matrix, factor, missing-method
IdaCMA-methods, 42, 42
        (GeneSelection-methods), 32
GeneSelection, matrix, numeric, missing-method
(GeneSelection-methods) 32 learningsets
        (GeneSelection-methods), 32
                                                     (learningsets-class), 44
GeneSelection-methods, 32
                                             learningsets-class, 44
golub, 33, 35
                                             limmatest (filter), 23
golubcrit (filter), 23
                                             mklr(internals), 35
internals, 35
                                             mkreg (internals), 35
                                             my.care.exp(internals), 35
join, 4, 36, 36, 52
join, list-method (join-methods),
                                             nnetCMA, 4, 6, 7, 12, 14, 17, 22, 25, 28, 39,
        36
                                                      41, 43, 45, 45, 46, 49, 51, 56, 58, 61,
join-methods, 36
                                                      63-65, 67, 69, 72, 74, 77, 80
join-methods, 36
```

INDEX

nnetCMA, data.frame, missing, formula-metharCMA, 4, 6, 8, 12, 14, 17, 22, 25, 28, 39, 41, (nnetCMA-methods), 45 43, 46, 49, 51, 55, 55, 58, 61, 63, 65, nnetCMA, ExpressionSet, character, missing-meth62, 69, 72, 74, 77, 80 plrCMA, data.frame, missing, formula-method (nnetCMA-methods), 45 nnetCMA, matrix, factor, missing-method (plrCMA-methods), 55 (nnetCMA-methods), 45 plrCMA, ExpressionSet, character, missing-method nnetCMA, matrix, numeric, missing-method (plrCMA-methods), 55 (nnetCMA-methods), 45 plrCMA, matrix, factor, missing-method nnetCMA-methods, 45, 45 (plrCMA-methods), 55 plrCMA, matrix, numeric, missing-method obsinfo, 18, 47, 75 (plrCMA-methods), 55 obsinfo, evaloutput-method plrCMA-methods, 55, 55 (evaloutput-class), 18 pls_ldaCMA, 4, 6, 8, 12, 14, 17, 22, 25, 28, 39, 41, 43, 46, 49, 51, 56, 57, 57, 58, pknnCMA, 4, 6, 7, 12, 14, 17, 22, 25, 28, 38, 61, 63, 65, 67, 69, 72, 74, 77, 80 39, 41, 43, 46, 48, 48, 51, 56, 58, 61, pls_ldaCMA, data.frame, missing, formula-method 63, 65, 67, 69, 72, 74, 77, 80 (pls_ldaCMA-methods), 57 pknnCMA, data.frame, missing, formula-method pls_ldaCMA, ExpressionSet, character, missing-method (pknnCMA-methods), 48 pknnCMA, ExpressionSet, character, missing-method pls_ldaCMA, matrix, factor, missing-method (pls_ldaCMA-methods), 57 (pknnCMA-methods), 48 (pls_ldaCMA-methods), 57 pknnCMA, matrix, factor, missing-method pls_ldaCMA, matrix, numeric, missing-method (pknnCMA-methods), 48 (pls_ldaCMA-methods), 57 pknnCMA, matrix, numeric, missing-method pls_ldaCMA-methods, 57 (pknnCMA-methods), 48 pls_ldaCMA-methods, 57 pknnCMA-methods, 48, 48 pls_lrCMA, 4, 6, 8, 12, 14, 17, 22, 25, 28, Planarplot, **50**, 50 Planarplot, data.frame, missing, formula-method 39, 41, 43, 46, 49, 51, 56, 58, 59, 59, 63, 65, 67, 69, 72, 74, 77, 80 (Planarplot-methods), 50 Planarplot, ExpressionSet, character, missing-method (pls_lrCMA-methods), 59 (Planarplot-methods), 50 Planarplot, matrix, factor, missing-methed pls_lrCMA, ExpressionSet, character, missing-meth (pls_lrCMA-methods), 59 (Planarplot-methods), 50 Planarplot, matrix, numeric, missing-method_lrCMA, matrix, factor, missing-method (pls_lrCMA-methods), 59 (Planarplot-methods), 50 pls_lrCMA, matrix, numeric, missing-method Planarplot-methods, 50 $(pls_lrCMA-methods), 59$ Planarplot-methods, 50 pls_lrCMA-methods, 59 plot, 52 pls_lrCMA-methods, 59 plot tuningresult, 54 pls_rfCMA, 4, 6, 8, 12, 14, 17, 22, 25, 28, plot,cloutput,missing-method 39, 41, 43, 46, 49, 51, 56, 58, 61, 62, (plot), 52 65, 67, 69, 72, 74, 77, 80 plot, cloutput-method, 5, 7 pls_rfCMA, data.frame, missing, formula-method plot, cloutput-method (plot), 52 (pls_rfCMA-methods), 61 plot,genesel,missing-method pls_rfCMA, ExpressionSet, character, missing-meth (Barplot), 53 (pls_rfCMA-methods), 61 plot, genesel-method, 31, 78 pls_rfCMA, matrix, factor, missing-method plot, genesel-method (Barplot), 53 (pls_rfCMA-methods), 61 plot, tuningresult, missing-method (plot tuningresult), 54 pls_rfCMA, matrix, numeric, missing-method plot, tuningresult-method, 82 (pls_rfCMA-methods), 61 pls_rfCMA-methods, 62 plot, tuningresult-method (plot tuningresult), 54 pls_rfCMA-methods, 61 pnnCMA, 4, 6, 8, 12, 14, 17, 22, 25, 28, 39, 41, plotprob(internals), 35

43, 46, 49, 51, 56, 58, 61, 63, 64, 67, scdaCMA, matrix, factor, missing-method 69, 72, 74, 77, 80 (scdaCMA-methods), 70 pnnCMA, data.frame, missing, formula-methsondaCMA, matrix, numeric, missing-method (pnnCMA-methods), 63 (scdaCMA-methods), 70 pnnCMA, ExpressionSet, character, missingsonbatCMAd methods, 70, 71 (pnnCMA-methods), 63 show, cloutput-method pnnCMA, matrix, factor, missing-method (cloutput-class), 5 (pnnCMA-methods), 63 show, evaloutput-method pnnCMA, matrix, numeric, missing-method (evaloutput-class), 18 (pnnCMA-methods), 63 show, genesel-method pnnCMA-methods, 63, 64 (genesel-class), 31 show, learningsets-method gdaCMA, 4, 6, 8, 12, 14, 17, 22, 25, 28, 39, 41, (learningsets-class), 44 43, 46, 49, 51, 56, 58, 61, 63, 65, 66, show, tuningresult-method 69, 72, 74, 77 (tuningresult-class), 81 qdaCMA, data.frame, missing, formula-method inkldaCMA, 4, 6, 8, 12, 14, 17, 22, 25, (qdaCMA-methods), 65 28, 39, 41, 43, 46, 49, 51, 56, 58, 61, qdaCMA, ExpressionSet, character, missing-method3, 65, 67, 69, 72, 73, 77 (qdaCMA-methods), 65 shrinkldaCMA, data.frame, missing, formula-method qdaCMA, matrix, factor, missing-method (shrinkldaCMA-methods), 72 (qdaCMA-methods), 65 shrinkldaCMA, ExpressionSet, character, missingqdaCMA, matrix, numeric, missing-method (shrinkldaCMA-methods), 72 (qdaCMA-methods), 65 shrinkldaCMA, matrix, factor, missing-method qdaCMA-methods, 65, 66 (shrinkldaCMA-methods), 72 shrinkldaCMA,matrix,numeric,missing-method rfCMA, 4, 6-8, 12, 14, 17, 22, 25, 28, 39, 41, (shrinkldaCMA-methods), 72 43, 46, 49, 51, 56, 58, 61-63, 65, 67, shrinkldaCMA-methods, 72, 72, 73 68, 68, 72, 74, 77, 80 summary, 74 rfCMA, data.frame, missing, formula-method summary, evaloutput-method, 18 (rfCMA-methods), 67 rfCMA, ExpressionSet, character, missing-method (summary),74 (rfCMA-methods), 67 svmCMA, 4, 6, 8, 12, 14, 17, 22, 25, 28, 39, 41, rfCMA, matrix, factor, missing-method 43, 46, 49, 51, 56, 58, 61, 63, 65, 67, (rfCMA-methods), 67 *69*, *72*, *74*, *75*, *75*, *79*, *80* rfCMA, matrix, numeric, missing-method svmCMA, data.frame, missing, formula-method (rfCMA-methods), 67 (svmCMA-methods), 75 rfCMA-methods, 67, 68 svmCMA,ExpressionSet,character,missing-method rfe(filter), 23 (svmCMA-methods), 75 roc, 69 svmCMA, matrix, factor, missing-method roc, cloutput-method, 5, 7 (svmCMA-methods), 75 roc, cloutput-method (roc), 69 svmCMA, matrix, numeric, missing-method ROCinternal (internals), 35 (svmCMA-methods), 75 roundvector (internals), 35 svmCMA-methods, 75, 75 rowswaps (internals), 35 toplist, *31*, *53*, 77 safeexp(internals), 35 scdaCMA, 4, 6, 8, 12, 14, 17, 22, 25, 28, 39, toplist, genesel-method (toplist), 41, 43, 46, 49, 51, 56, 58, 61, 63, 65, 77 67, 69, 70, 71, 74, 77, 80 ttest (filter), 23 scdaCMA, data.frame, missing, formula-metthode, 4, 7, 12, 28, 30, 34, 44, 54, 56, 58, 60, (scdaCMA-methods), 70 64, 76, 78, 78, 81, 82

(scdaCMA-methods), 70 64, 76, 78, 78, 81, 82 scdaCMA, ExpressionSet, character, missing, formula-method (scdaCMA-methods), 70 (tune-methods), 78

```
tune,ExpressionSet,character,missing-method
       (tune-methods), 78
tune,matrix,factor,missing-method
       (tune-methods), 78
tune,matrix,numeric,missing-method
      (tune-methods), 78
{\tt tune-methods, 78}
tuningresult, 1, 4, 54, 80, 81
tuningresult
      (tuningresult-class),81
tuningresult-class, 81
varseloutput, 6, 23
varseloutput
       (varseloutput-class), 82
varseloutput-class, 82
welchtest (filter), 23
wilcoxtest (filter), 23
```