Package ‘CMA’

November 20, 2016

Type  Package
Title  Synthesis of microarray-based classification
Version  1.32.0
Date  2009-09-14
Author  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>
Depends  R (>= 2.10), methods, stats, Biobase
Suggests  MASS, class, nnet, glmnet, e1071, randomForest, plsgenomics, gbm, mgcv, corpcor, limma, st, mvtnorm
Description  This package provides a comprehensive collection of various microarray-based classification algorithms both from Machine Learning and Statistics. Variable Selection, Hyperparameter tuning, Evaluation and Comparison can be performed combined or stepwise in a user-friendly environment.

Collate  classes.r GenerateLearningsets.r GeneSelection.r tune.r classification.r evaluation.r join.r compare.r Planarplot.r compBoostCMA.r dldaCMA.r ElasticNetCMA.r ldaCMA.r flexdaCMA.r gbmCMA.r knnCMA.r LassoCMA.r ldaCMA.r nnetCMA.r pknnCMA.r plrCMA.r pls_lldaCMA.r pls_lrcMA.r pls_rfCMA.r pnmCMA.r qdaCMA.r rfCMA.r scldaCMA.r weighted_mcr.r wmc.r shrinkldaCMA.r svmCMA.r filter.r internals.r

License  GPL (>= 2)
biocViews  Classification, DecisionTree
NeedsCompilation  no

R topics documented:

  CMA-package ................................................................. 3
  Barplot ........................................................................... 4
  best ............................................................................... 5
  boxplot ........................................................................... 6
  classification ................................................................. 7
  classification-methods ..................................................... 9
  cloutput-class ............................................................... 9
  clvarseloutput-class ....................................................... 10
R topics documented:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>compare</td>
<td>11</td>
</tr>
<tr>
<td>compare-methods</td>
<td>13</td>
</tr>
<tr>
<td>compBoostCMA</td>
<td>14</td>
</tr>
<tr>
<td>compBoostCMA-methods</td>
<td>16</td>
</tr>
<tr>
<td>dldaCMA</td>
<td>16</td>
</tr>
<tr>
<td>dldaCMA-methods</td>
<td>18</td>
</tr>
<tr>
<td>ElasticNetCMA</td>
<td>19</td>
</tr>
<tr>
<td>ElasticNetCMA-methods</td>
<td>21</td>
</tr>
<tr>
<td>evaloutput-class</td>
<td>21</td>
</tr>
<tr>
<td>evaluation</td>
<td>22</td>
</tr>
<tr>
<td>evaluation-methods</td>
<td>24</td>
</tr>
<tr>
<td>fdaCMA</td>
<td>24</td>
</tr>
<tr>
<td>fdaCMA-methods</td>
<td>26</td>
</tr>
<tr>
<td>filter</td>
<td>27</td>
</tr>
<tr>
<td>flexdaCMA</td>
<td>27</td>
</tr>
<tr>
<td>flexdaCMA-methods</td>
<td>29</td>
</tr>
<tr>
<td>ftable</td>
<td>30</td>
</tr>
<tr>
<td>gbmCMA</td>
<td>30</td>
</tr>
<tr>
<td>gbmCMA-methods</td>
<td>32</td>
</tr>
<tr>
<td>GenerateLearningsets</td>
<td>33</td>
</tr>
<tr>
<td>genesel-class</td>
<td>34</td>
</tr>
<tr>
<td>GeneSelection</td>
<td>35</td>
</tr>
<tr>
<td>GeneSelection-methods</td>
<td>38</td>
</tr>
<tr>
<td>golub</td>
<td>38</td>
</tr>
<tr>
<td>internals</td>
<td>39</td>
</tr>
<tr>
<td>join</td>
<td>39</td>
</tr>
<tr>
<td>join-methods</td>
<td>40</td>
</tr>
<tr>
<td>khan</td>
<td>40</td>
</tr>
<tr>
<td>knnCMA</td>
<td>41</td>
</tr>
<tr>
<td>knnCMA-methods</td>
<td>43</td>
</tr>
<tr>
<td>LassoCMA</td>
<td>43</td>
</tr>
<tr>
<td>LassoCMA-methods</td>
<td>43</td>
</tr>
<tr>
<td>ldaCMA</td>
<td>45</td>
</tr>
<tr>
<td>ldaCMA-methods</td>
<td>45</td>
</tr>
<tr>
<td>learningsets-class</td>
<td>47</td>
</tr>
<tr>
<td>nnetCMA</td>
<td>48</td>
</tr>
<tr>
<td>nnetCMA-methods</td>
<td>50</td>
</tr>
<tr>
<td>obsinfo</td>
<td>51</td>
</tr>
<tr>
<td>pknnCMA</td>
<td>52</td>
</tr>
<tr>
<td>pknnCMA-methods</td>
<td>53</td>
</tr>
<tr>
<td>Planarplot</td>
<td>54</td>
</tr>
<tr>
<td>Planarplot-methods</td>
<td>55</td>
</tr>
<tr>
<td>plot</td>
<td>56</td>
</tr>
<tr>
<td>plot tuningresult</td>
<td>57</td>
</tr>
<tr>
<td>plrCMA</td>
<td>58</td>
</tr>
<tr>
<td>plrCMA-methods</td>
<td>59</td>
</tr>
<tr>
<td>pls_ldaCMA</td>
<td>60</td>
</tr>
<tr>
<td>pls_ldaCMA-methods</td>
<td>61</td>
</tr>
<tr>
<td>pls_lrCMA</td>
<td>62</td>
</tr>
<tr>
<td>pls_lrCMA-methods</td>
<td>63</td>
</tr>
<tr>
<td>pls_rfCMA</td>
<td>64</td>
</tr>
<tr>
<td>pls_rfCMA-methods</td>
<td>65</td>
</tr>
</tbody>
</table>
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 \texttt{GenerateLearningsets}
2. (Optionally): Perform variable selection using \texttt{GeneSelection}
3. (Optionally): Perform hyperparameter tuning using \texttt{tune}
4. Perform \texttt{classification} using 1.-3.
5. Repeat 2.-4. based on 1. for several methods: \texttt{compBoostCMA}, \texttt{dldaCMA}, \texttt{ElasticNetCMA}, \texttt{fdaCMA}, \texttt{flexdaCMA}, \texttt{gbmCMA}, \texttt{knnCMA}, \texttt{ldaCMA}, \texttt{LassoCMA}, \texttt{nnetCMA}, \texttt{pklrCMA}, \texttt{pls_ldaCMA}, \texttt{pls_lrCMA}, \texttt{pls_rfcma}, \texttt{pnnCMA}, \texttt{qdaCMA}, \texttt{rfCMA}, \texttt{scdaCMA}, \texttt{shrinkldacma}, \texttt{svmCMA}
6. Evaluate the results from 5. using \texttt{evaluation} and make a comparison by calling \texttt{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


---

**Barplot**

**Barplot of variable importance**

**Description**

This method can be seen as a visual pendant to \texttt{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 \texttt{GeneSelection}, s. \texttt{genesel}.

**Arguments**

- \texttt{x} An object of class \texttt{genesel}
- \texttt{top} Number of top genes whose variable importance should be displayed. Defaults to 10.
- \texttt{iter} Iteration number (\texttt{learningset}) for which variable importance should be displayed.
- ... Further graphical options passed to \texttt{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


See Also

genesel, GeneSelection, toplist

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.
**Description**

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

**Arguments**

- **x**: An object of class `evaloutput`.
- **...**: Further graphical parameters passed to the classical `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**


**See Also**

evaluation
classification

General method for classification with various methods

Description

Most general function in the package, providing an interface to perform variable selection, hyper-parameter 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, classifier, tuneres, trace = TRUE, models=FALSE,...)

Arguments

X  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.

y  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 variable 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 each 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.
---|---
tuneres | Analogous to the argument genesel - object of class `tuningresult` containing 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 (default), no hyperparameter tuning will be performed at all. To use pre-defined hyperparameter grids, the argument is `tuninglist = list(grids = list())`.
trace | Should progress be traced? Default is `TRUE`.
models | a logical value indicating whether the model object shall be returned
---|---

### 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 conveniently 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 <berna@ibe.med.uni-muenchen.de>

### References


### See Also

`GeneSelection`, `tune`, `evaluation`, `compBoostCMA`, `dldaCMA`, `ElasticNetCMA`, `fdaCMA`, `flexdaCMA`, `gbmCMA`, `knnCMA`, `ldaCMA`, `LassoCMA`, `nnetCMA`, `pknCMA`, `plrCMA`, `pls ldaCMA`, `pls lrCMA`, `pls rFCMA`, `pnnCMA`, `qdaCMA`, `rfCMA`, `scdaCMA`, `shrinkldaCMA`, `svmCMA`

### Examples

```r
### 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)
### 1. GeneSelection
selttest <- GeneSelection(golubX, golubY, learningsets = lset, method = "t.test")
### 2. tuning
tunek <- tune(golubX, golubY, learningsets = lset, genesel = selttest, nbgene = 20, classifier = knnCMA)
### 3. classification
knn1 <- classification(golubX, golubY, learningsets = lset, genesel = selttest, tuneres = tunek, nbgene = 20, classifier = knnCMA)
### steps 1.-3. combined into one step:
knn2 <- classification(golubX, golubY, learningsets = lset, genesellist = list(method = "t.test"), classifier = knnCMA, tuninglist = list(grids = list(k = c(1:8))), nbgene = 20)
### show and analyze results:
knnjoin <- join(knn2)
show(knn2)
eval <- evaluation(knn2, measure = "misclassification")
show(eval)
summary(eval)
boxplot(eval)

# End(Not run)

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.

cloutput-class "cloutput"

Description

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

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 classifier 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).
model: List containing the constructed classifiers.

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, scdCMA, shrinkldaCMA, svmCMA

clvarseloutput-class "clvarseloutput"

Description
Object returned by all classifiers that can perform 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 cloutput and varsel, s. below.
Slots

learnind: Vector of indices that indicates which observations were 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 classifier 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 "vareloutput", directly.

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

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**

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

**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 default, all measures are computed, using `evaluation` with scheme = "iterationwise". Note that "sensitivity", "specificity", "auc" cannot be computed for the multiclass case.
  - "misclassification" The misclassification rate.
  - "sensitivity" The sensitivity or 1-false negative rate. Can only be computed for binary classification.
  - "specificity" The specificity or 1-false positive rate. Can only be computed for binary classification.
  - "average probability" The average probability assigned to the correct class. Requirement is that the used classifier provides probability estimations. The optimum performance is 1.
  - "brier score" The Brier Score is generally defined as \[\sum_{\text{all observation } i} \sum_{\text{all classes } k} \left( \mathbb{I}(y_i=k) - P(k) \right)^2\], with \(\mathbb{I}()\) denoting the indicator function and \(P(k)\) the estimated probability for class \(k\). The optimum performance is 0.
  - "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`.

- `aggfun`: Function that determines how performance among different iterations are aggregated. 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 <bernaue@ibe.med.uni-muenchen.de>

References


See Also

classification, evaluation

Examples

```r
## Not run:
### compare the performance of several discriminant analysis methods
### for the Khan dataset:
data(khan)
khanX <- as.matrix(khan[,,-1])
khanY <- khan[,1]
set.seed(27611)
fiveCV10iter <- GenerateLearningsets(y=khanY, method = "CV", fold = 5, niter = 2, strat = TRUE)
### candidate methods: DLDA, LDA, QDA, pls_LDA, sclda
class_dlda <- classification(X = khanX, y=khanY, learningsets = fiveCV10iter, classifier = dldaCMA)
### peform GeneSelection for LDA, FDA, QDA (using F-Tests):
genesel_da <- GeneSelection(X=khanX, y=khanY, learningsets = fiveCV10iter, method = "f.test")
###
class_lda <- classification(X = khanX, y=khanY, learningsets = fiveCV10iter, classifier = ldaCMA, genesel= genesel_da)
class_qda <- classification(X = khanX, y=khanY, learningsets = fiveCV10iter, classifier = qdaCMA, genesel = genesel_da)

### 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", "average probability"))
print(comparison)
## End(Not run)
```

Description

Compare different classifiers for the following signatures:
Methods

clresultlist = "list" signature 1
For further argument and output information, consult compare

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 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 = 100, nu = 0.1, models=FALSE, ...)

Arguments

X  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.

y  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.

models  a logical value indicating whether the model object shall be returned

...  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**


**See Also**

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

**Examples**

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

Diagonal Discriminant Analysis

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

Usage
dldaCMA(X, y, f, learnind, models=FALSE, ...)
**Arguments**

**\( X \)**  
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 (see below).  
- An object of class `ExpressionSet`.  

**\( y \)**  
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.

**\( \text{learnind} \)**  
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.

**models**  
a logical value indicating whether the model object shall be returned  
...  
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**

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

**See Also**

`compBoostCMA`, `ElasticNetCMA`, `FdaCMA`, `flexdaCMA`, `gbmCMA`, `knnCMA`, `ldaCMA`, `LassoCMA`, `nnetCMA`, `pknnCMA`, `plrCMA`, `pls_ladCMA`, `pls_lrcMA`, `pls_rfCMA`, `pnnCMA`, `qdaCMA`, `rfCMA`, `scdaCMA`, `shrinkldaCMA`, `svmCMA`
Examples

```r
### 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 DLDA
dldaresult <- dldaCMA(X=golubX, y=golubY, learnind=learnind)
### show results
show(dldaresult)
ftable(dldaresult)
plot(dldaresult)
### multiclass example:
### 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(ratio*length(khanY)))
### run LDA
ldaresult <- dldaCMA(X=khanX, y=khanY, learnind=learnind)
### show results
show(ldaresult)
ftable(ldaresult)
plot(ldaresult)
```

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`. 
ElasticNetCMA

Classification 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.

The method can be used for variable selection alone, s. GeneSelection.

For S4 method information, see ElasticNetCMA-methods.

Usage

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

Arguments

X  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. note: by default, the predictors are scaled to have unit variance and zero mean. Can be changed by passing standardize = FALSE via the ... argument.

y  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.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.

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

models  a logical value indicating whether the model object shall be returned

...  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. `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


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

```r
### load Golub AML/ALL data
data(golub)
### extract class labels
golubY <- golub[,1]
### extract gene expression
golubX <- as.matrix(golub[,-1])
### select learning set
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=0.5)
show(result)
ftable(result)
plot(result)
```
ElasticNetCMA-methods

Classification 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

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 consistently 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 Evaluation of classifiers

Description

The performance of classifiers can be evaluated by six different measures and two different schemes that are described more precisely below.
For S4 method information, s. evaluation-methods.

Usage

evaluation(clresult, cltrain = NULL, cost = NULL, y = NULL, measure = c("misclassification", "sensitivity", "specificity", "average probability", "brier score", "auc", "0.632", "0.632+")

scheme = c("iterationwise", "observationwise", "classwise"))

Arguments

cresult A list of objects of class cloutput or clvarseloutput
cltrain An object of class cloutput in which the whole 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 predicted class is j-1, provided the conventional coding 0,...,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.
y A vector containing the true class labels. Only needed if scheme = "classwise".
measure Performance measure to be used:
"misclassification" The misclassification rate.
"sensitivity" The sensitivity or 1-false negative rate. Can only be computed for binary classification.
"specificity" The specificity or 1-false positive rate. Can only be computed for binary classification.
"average probability" The average probability assigned to the correct class. Requirement is that the used classifier provides probability estimations. The optimum performance is 1.
"brier score" The Brier Score is generally defined as \( \sum_{i} \sum_{k} (I(y_i=k)-P(k))^2 \),
with \( I() \) denoting the indicator function and \( P(k) \) the estimated probability for class k. The optimum performance is 0.
"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.

"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 iterations 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 <bernaure@ibe.med.uni-muenchen.de>

References


See Also
evaloutput, classification, compare

Examples

### simple linear discriminant analysis example using bootstrap datasets:
### datasets:
data(golub)
golubY <- golub[,1]
### extract gene expression from first 10 genes
golubX <- as.matrix(golub[,2:11])
### generate 25 bootstrap datasets
set.seed(333)
boots <- GenerateLearningsets(y = golubY, method = "bootstrap", ntrain = 30, niter = 10, strat = TRUE)
### run classification()

```r
ldalist <- classification(X=golubX, y=golubY, learningsets = bootds, classifier=ldaCMA)
```

### Evaluation:

```r
eval_iter <- evaluation(ldalist, scheme = "iter")
eval_obs <- evaluation(ldalist, scheme = "obs")
show(eval_iter)
show(eval_obs)
summary(eval_iter)
summary(eval_obs)
```

### auc with boxplot

```r
eval_auc <- evaluation(ldalist, scheme = "iter", measure = "auc")
boxplot(eval_auc)
```

### which observations have often been misclassified ?

```r
obsinfo(eval_obs, threshold = 0.75)
```

---

### 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`.

---

### 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**

```r
fdaCMA(X, y, f, learning, comp = 1, plot = FALSE, models=FALSE)
```

**Arguments**

- **X**  
  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`.

- **y**  
  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.

\( \text{learnind} \) 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.

\( \text{comp} \) 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.

\( \text{plot} \) Should the projections onto the space spanned by the optimal projection directions be plotted? Default is FALSE.

\( \text{models} \) a logical value indicating whether the model object shall be returned.

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

**Examples**

```r
### 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
```
```r
ratio <- 2/3
set.seed(111)
learnind <- sample(length(golubY), size=floor(ratio*length(golubY)))
### run FDA
fdaresult <- fdaCMA(X=golubX, y=golubY, learnind=learnind, comp = 1, plot = TRUE)
### show results
show(fdaresult)
ftable(fdaresult)
plot(fdaresult)
### multiclass example:
### load Khan data
data(khan)
### extract class labels
khanY <- khan[,1]
### extract gene expression from first 10 genes
khanX <- as.matrix(khan[,2:11])
### select learning set
set.seed(111)
learnind <- sample(length(khanY), size=floor(ratio*length(khanY)))
### run FDA
fdaresult <- fdaCMA(X=khanX, y=khanY, learnind=learnind, comp = 2, plot = TRUE)
### show results
show(fdaresult)
ftable(fdaresult)
plot(fdaresult)
```

---

### 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`. 

---


**Filter functions for Gene Selection**

**Description**

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

**Usage**

```r
ttest(X, y, learnind, ...)
welchtest(X, y, learnind, ...)
test(X, y, learnind, ...)
kruskaltest(X, y, learnind, ...)
limmatest(X, y, learnind, ...)
golubcrit(X, y, learnind, ...)
rfe(X, y, learnind, ...)
shrinkcat(X, y, learnind, ...)
```

**Arguments**

- `X`: A numeric matrix of gene expression values.
- `y`: 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**


---

**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 $n_{\text{learn}} \times K$ indicator matrix of the class labels on $X$. This gives rise to ‘non-linear discriminant 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, models=FALSE, ...)`

Arguments

X  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.

y  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 directions be plotted? Default is FALSE.

models  a logical value indicating whether the model object shall be returned

...  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

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

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 \( n \times K \) indicator matrix of the class labels on \( X \). This gives rise to ‘non-linear discriminant 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

\[
\begin{align*}
X &= \text{"matrix"}, \ y = \text{"numeric"}, \ f = \text{"missing"} \quad \text{signature 1} \\
X &= \text{"matrix"}, \ y = \text{"factor"}, \ f = \text{"missing"} \quad \text{signature 2} \\
X &= \text{"data.frame"}, \ y = \text{"missing"}, \ f = \text{"formula"} \quad \text{signature 3} \\
X &= \text{"ExpressionSet"}, \ y = \text{"character"}, \ f = \text{"missing"} \quad \text{signature 4}
\end{align*}
\]

For further argument and output information, consult \texttt{flexdaCMA}. 

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

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, models=FALSE,...)
Arguments

X  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.

y  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.

models  a logical value indicating whether the model object shall be returned

Value

An object 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

The state of boosting.

See Also

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

Examples

```r
### load Golub AML/ALL data
data(golub)
### extract class labels
golubY <- golub[,1]
### extract gene expression
golubX <- as.matrix(golub[,-1])
### select learning set
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)
```

### 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`.
GenerateLearningsets  Repeated Divisions into learn- and test 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

GenerateLearningsets(n, y, method = c("LOOCV", "CV", "MCCV", "bootstrap"),
fold = NULL, niter = NULL, ntrain = NULL, strat = FALSE)

Arguments

- **n**: The total number of observations in the available data set. May be missing if y is provided instead.
- **y**: A vector of class labels, either numeric or a factor. Must 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 observations.
  - "bootstrap" Learning sets are generated by drawing n times with replacement 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".
- **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 fold*niter 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
genesel-class

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

Is cross-validation valid for small-sample microarray classification?
Bioinformatics, 20(3), 374-380

Prediction error estimation: a comparison of resampling methods.
Bioinformatics, 21(15), 3301-3307

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")
show(loo)
# five-fold-CV
CV5 <- GenerateLearningsets(n=40, method="CV", fold=5)
show(CV5)
# MCCV
mccv <- GenerateLearningsets(n=40, method = "MCCV", niter=3, ntrain=30)
show(mccv)
# Bootstrap
boot <- GenerateLearningsets(n=40, method="bootstrap", niter=3)
# 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)

genesel-class  "genesel"

Description

Object returned from a call to GeneSelection
**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", "one-vs-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

---

**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. GeneSelection-methods.

**Usage**

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

\(X\) 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.

\(y\) 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.

\(\text{learningsets}\) An object of class learningsets. May be missing, then the complete datasets is used as learning set.

\(\text{method}\) A character specifying the method to be used:
- \(\text{t.test}\) two-sample \(t\) test (equal variances for both classes assumed).
- \(\text{welch.test}\) Welch modification of the \(t\) test (unequal variances for both classes).
- \(\text{wilcox.test}\) Wilcoxon rank sum test.
- \(\text{f.test}\) 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 \(\text{method} = \text{t.test}\) in the two-class case.
- \(\text{kruskal.test}\) Multi-class generalization of the Wilcoxon rank sum test and the nonparametric pendant to the F test, respectively.
- \(\text{limma}\) ’Moderated t’ statistic for the two-class case and ’moderated F’ statistic for the multiclass case, described in Smyth (2003). Requires the package \text{limma}.
- \(\text{rfe}\) One-step Recursive Feature Elimination, based on the Support Vector Machine. The method is decribed in Guyon et al. (2002). Requires the package \text{e1071}. Take care that appropriate hyperparameters are passed by the ... argument.
- \(\text{rf}\) Random Forest Variable Importance Measure. Requires the package \text{randomForest}
- \(\text{lasso}\) L1 penalized logistic regression leads to sparsity with respect to the variables used. Calls the function \text{LassoCMA}, which requires the package \text{glmpath}. \textbf{warning}: Take care that appropriate hyperparameters are passed by the ... argument.
- \(\text{elasticnet}\) Penalized logistic regression with both L1 and L2 penalty, claimed by Zhou and Hastie (2004) to select 'variable groups'. Calls the function \text{ElasticNetCMA}, which requires the package \text{glmpath}. \textbf{warning}: Take care that appropriate hyperparameters are passed by the ... argument.
- \(\text{boosting}\) Componentwise boosting (Buehlmann and Yu, 2003) has been shown to mimic the LASSO (Efron et al., 2004; Buehlmann and Yu, 2006). Calls the function \text{compBoostCMA} Take care that appropriate hyperparameters are passed by the ... argument.
- \(\text{golub}\) The (theoretically unfounded) variable selection criterion used by Golub et al. (1999), s. \text{golub}.
- \(\text{shrinkcat}\) The correlation-adjusted t-score from Zuber and Strimmer (2009)
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
- 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 <bernau@ibe.med.uni-muenchen.de>

References


See Also

`filter.GeneSelection`, `tune.classification`

Examples

```r
# 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(111)
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)
toplist(selttest, k = 10, iter = 1)
plot(selttest, iter = 1)
```

GeneSelection-methods  General method for variable selection with various methods

Description

Performs gene selection for the following signatures:

Methods

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

For further argument and output information, consult `GeneSelection`.

---

golub  

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

Description

s. below

Usage

```r
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


Examples

data(golub)

---

**Description**

Not intended to be called directly by the user.

**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 successful join is that the used dataset and classifier 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`

```
join-methods

Combine list elements returned by the method `classification`
```

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`.

```
khan

Small blue round cell tumor dataset of Khan et al. (2001)
```

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

Examples

data(khan)

knnCMA  _Nearest Neighbours_

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, models=FALSE, ...)

Arguments

- **X**  
  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_.

- **y**  
  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.

- **models**  
  a logical value indicating whether the model object shall be returned

- **...**  
  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, see pknncma.

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, ldaCMA, LassoCMA, nnetCMA, pknncma, plrCMA, pls_ladCMA, 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[,-1])
### select learning set
ratio <- 2/3
set.seed(111)
learnind <- sample(length(golubY), size=floor(ratio*length(golubY)))
### 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]
### extract gene expression
khanX <- as.matrix(khan[,,-1])
### select learning set
set.seed(111)
learnind <- sample(length(khanY), size=floor(ratio*length(khanY)))
### run knn
result <- knnCMA(X=khanX, y=khanY, learnind=learnind, k = 5)
### show results
show(result)
ftable(result)
**knnCMA-methods**

**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`.

**LassoCMA**

**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 developed 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**

```r
LassoCMA(X, y, f, learning, norm.fraction = 0.1, models=FALSE,...)
```

**Arguments**
- **X** 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. **note**: by default, the predictors are scaled to have unit variance and zero mean. Can be changed by passing `standardize = FALSE` via the ... argument.
- **y** 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.
LassoCMA

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.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.

models

a logical value indicating whether the model object shall be returned

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


See Also

compBoostCMA, dldaCMA, ElasticNetCMA, fdaCMA, flexdaCMA, gbmCMA, knnCMA, ldaCMA, nnetCMA, pknnCMA, plrCMA, pls_lldaCMA, 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 learning set
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)

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 developed an algorithm to compute the entire solution path of the Lasso for an arbitrary generalized linear model, implemented in the package \texttt{glmnet}. The method can be used for variable selection alone, s. \texttt{GeneSelection}

Methods

\begin{itemize}
  \item \texttt{X = "matrix", y = "numeric", f = "missing"} signature 1
  \item \texttt{X = "matrix", y = "factor", f = "missing"} signature 2
  \item \texttt{X = "data.frame", y = "missing", f = "formula"} signature 3
  \item \texttt{X = "ExpressionSet", y = "character", f = "missing"} signature 4
\end{itemize}

For references, further argument and output information, consult \texttt{LassoCMA}.

ldaCMA  
Linear Discriminant Analysis

Description

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

For S4 method information, see \texttt{ldaCMA-methods}.

Usage

\texttt{ldaCMA(X, y, f, learnind, models=FALSE, ...)}

Arguments

\begin{itemize}
  \item \texttt{X}  
    Gene expression data. Can be one of the following:
    \begin{itemize}
      \item A \texttt{matrix}. Rows correspond to observations, columns to variables.
      \item A \texttt{data.frame}, when \texttt{f} is not missing (s. below).
      \item An object of class \texttt{ExpressionSet}.
    \end{itemize}
  \item \texttt{y}  
    Class labels. Can be one of the following:
\end{itemize}
• 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.

\( \text{learnind} \)  
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.

\( \text{models} \)  
a logical value indicating whether the model object shall be returned

...  
Further arguments to be passed to \( \text{lda} \) from the package \texttt{MASS}

**Value**

An object of class \texttt{cloutput}.

**Note**

Excessive variable selection has usually to performed before \( \text{ldaCMA} \) 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**

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

**See Also**

\texttt{compBoostCMA, dldaCMA, ElasticNetCMA, fdaCMA, flexdaCMA, gbmCMA, knnCMA, LassoCMA, nnetCMA, pknnCMA, plrCMA, pls\_ldaCMA, pls\_lrCMA, pls\_rfCMA, pnnCMA, qdaCMA, rfCMA, scdaCMA, shrinkldaCMA, svmCMA}

**Examples**

```r
### 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)
```
### ldaCMA-methods

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`.

#### learningsets-class

"learningsets"

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

learnmatrix: A matrix of dimension niter \times 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 <bernau@ibe.med.uni-muenchen.de>

See Also

GenerateLearningsets, GeneSelection, tune, classification

---

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, models=FALSE,...)

Arguments

- **X**  
  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.

- **y**  
  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.

`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.

`models`  
a logical value indicating whether the model object shall be returned

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 `<berbau@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, glmCMA, knnCMA, ldaCMA, LassoCMA, nnetCMA, pknnCMA, plrCMA, pls_ldaCMA, pls_lrCMA, pls_rfCMA, pnnCMA, qdaCMA, rfCMA, scdaCMA, shrinkldaCMA, svmCMA`
Examples

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

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`. 
Classifiability of observations

Description

Some observations are harder to classify than others. It is frequently of interest to know which observations are consistently misclassified; these are candidates 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 below `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 consistently misclassified observations and the corresponding performance measure.
- **notclassified**: The indices of those observations not classified at all, s. details.

Author(s)

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

References


See Also

evaluation
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, models=FALSE, ...)

Arguments

X  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.

y  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.

models  a logical value indicating whether the model object shall be returned

...  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.
pknnCMA-methods

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`, `fdacom`, `flexdaCMA`, `gbmCMA`, `knnCMA`, `ldaCMA`, `LassoCMA`, `nnetCMA`, `plrcma`, `pls_ldaCMA`, `pls_lrCMA`, `pls_rfCMA`, `pnnCMA`, `qdaCMA`, `rfCMA`, `scdaCMA`, `shrinkldaCMA`, `svmCMA`

Examples

```r
### 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])
### select learning set
ratio <- 2/3
set.seed(111)
learnind <- sample(length(golubY), size=floor(ratio*length(golubY)))
### run probabilistic k-nearest neighbours
result <- pknnCMA(X=golubX, y=golubY, learnind=learnind, k = 3)
### show results
show(result)
ftable(result)
plot(result)
```

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`. 
Planarplot

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.

For S4 method information, s. Planarplot-methods.

Usage

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

Arguments

X  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.

y  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 exactly 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.
Planarplot-methods

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)
set.seed(111)
learnind <- sample(golubn, size=floor(2/3*golubn))
Planarplot(X=golubX, y=golubY, learnind=learnind, predind=c(2,4),
          classifier=ldaCMA)
```

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.
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

- **x**: 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


See Also

`cloutput`
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

- **x**: An object of class `tuningresult`.
- **iter**: Iteration number (learning set) for which tuning results should be displayed.
- **which**: Character vector (maximum length is two) naming the arguments for which tuning results should be display. Default is NULL; if the number of tuned hyperparameter 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


See Also

tune, tuningresult
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, models=FALSE,...)
```

Arguments

- **X**: 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`.

- **y**: 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.

- **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.

- **models**: a logical value indicating whether the model object shall be returned

- **...**: 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

Biostatistics 5:427-443.

See Also

compBoostCMA, d1ldaCMA, ElasticNetCMA, fdaCMA, flexdaCMA, gbmcma, knnCMA, ldaCMA, LassoCMA, nnetCMA, pknnCMA, pls_1dCMA, pls_1rCMA, pls_rfCMA, pnnCMA, qdaCMA, rfCMA, scdaCMA, shrinkldaCMA, svmCMA

Examples

```r
### 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])
### select learningset
ratio <- 2/3
set.seed(111)
learnind <- sample(length(golubY), size=floor(ratio*length(golubY)))
### run penalized logistic regression (no tuning)
plrresult <- plrCMA(X=golubX, y=golubY, learnind=learnind)
### show results
show(plrresult)
ftable(plrresult)
plot(plrresult)
### multiclass example:
### load Khan data
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)
```

### 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])
### select learningset
ratio <- 2/3
set.seed(111)
learnind <- sample(length(golubY), size=floor(ratio*length(golubY)))
### run penalized logistic regression (no tuning)
plrresult <- plrCMA(X=golubX, y=golubY, learnind=learnind)
### show results
show(plrresult)
ftable(plrresult)
plot(plrresult)
### multiclass example:
### load Khan data
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)

### 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])
### select learningset
ratio <- 2/3
set.seed(111)
learnind <- sample(length(golubY), size=floor(ratio*length(golubY)))
### run penalized logistic regression (no tuning)
plrresult <- plrCMA(X=golubX, y=golubY, learnind=learnind)
### show results
show(plrresult)
ftable(plrresult)
plot(plrresult)
### multiclass example:
### load Khan data
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)

---

### 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])
### select learningset
ratio <- 2/3
set.seed(111)
learnind <- sample(length(golubY), size=floor(ratio*length(golubY)))
### run penalized logistic regression (no tuning)
plrresult <- plrCMA(X=golubX, y=golubY, learnind=learnind)
### show results
show(plrresult)
ftable(plrresult)
plot(plrresult)
### multiclass example:
### load Khan data
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)

---

**Description**

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

\[
\begin{align*}
X &= \text{"matrix"}, \ y = \text{"numeric"}, \ f = \text{"missing"} \quad \text{signature 1} \\
X &= \text{"matrix"}, \ y = \text{"factor"}, \ f = \text{"missing"} \quad \text{signature 2} \\
X &= \text{"data.frame"}, \ y = \text{"missing"}, \ f = \text{"formula"} \quad \text{signature 3} \\
X &= \text{"ExpressionSet"}, \ y = \text{"character"}, \ f = \text{"missing"} \quad \text{signature 4}
\end{align*}
\]

For further argument and output information, consult \texttt{plrCMA}.

pl\_ldaCMA

\textit{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 \texttt{plsgenomics}.

For S4 method information, see \texttt{pls\_ldaCMA}\-methods.

Usage

\texttt{pls\_ldaCMA}(X, y, f, learnind, comp = 2, plot = FALSE, models = FALSE)

Arguments

- \texttt{X} Gene expression data. Can be one of the following:
  - A matrix. Rows correspond to observations, columns to variables.
  - A data.frame, when \texttt{f} is not \texttt{missing} (s. below).
  - An object of class ExpressionSet.

- \texttt{y} Class labels. Can be one of the following:
  - A numeric vector.
  - A factor.
  - A character if \texttt{X} is an ExpressionSet that specifies the phenotype variable.
  - \texttt{missing}, if \texttt{X} is a data.frame and a proper formula \texttt{f} is provided.

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

- \texttt{learnind} An index vector specifying the observations that belong to the learning set. May be \texttt{missing}; in that case, the learning set consists of all observations and predictions are made on the learning set.

- \texttt{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 \texttt{tune}.

- \texttt{plot} If \texttt{comp} \(\leq 2\), should the classification space of the Partial Least Squares components be plotted? Default is \texttt{FALSE}.

- \texttt{models} a logical value indicating whether the model object shall be returned.
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

Tumor classification by partial least squares using microarray gene expression data.
*Bioinformatics* 18, 39-50

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

Examples

```r
### 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 classifier, without tuning
plsresult <- pls_ldaCMA(X=khanX, y=khanY, learnind=learnind, comp = 4)
### show results
show(plsresult)
ftable(plsresult)
plot(plsresult)
```

 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 = "numeric", f = "formula" signature 3
X = "ExpressionSet", y = "character", f = "missing" signature 4

For further argument and output information, consult pls_ldaCMA.

Description

This method constructs a classifier that extracts Partial Least Squares components that form 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, models=FALSE)

Arguments

X 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.

y 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.

lambda Parameter controlling the amount of L2 penalization for logistic regression, usually taken to be a small value in order to stabilize estimation in the case of separable data.

plot If comp <= 2, should the classification space of the Partial Least Squares components be plotted? Default is FALSE.

models a logical value indicating whether the model object shall be returned.
**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**

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`, `pkncMA`, `plrcMA`, `pls_ldacMA`, `pls_rfCMA`, `pnnCMA`, `qdacMA`, `rfCMA`, `scdacMA`, `shrinkldacMA`, `svmCMA`

**Examples**

```r
### 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 PLS, combined with logistic regression
result <- pls_lrCMA(X=golubX, y=golubY, learnind=learnind)
### show results
show(result)
ftable(result)
plot(result)
```

---

**Description**

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

\[ X = \text{"matrix"}, \ y = \text{"numeric"}, \ f = \text{"missing"} \]  signature 1
\[ X = \text{"matrix"}, \ y = \text{"factor"}, \ f = \text{"missing"} \]  signature 2
\[ X = \text{"data.frame"}, \ y = \text{"missing"}, \ f = \text{"formula"} \]  signature 3
\[ X = \text{"ExpressionSet"}, \ y = \text{"character"}, \ f = \text{"missing"} \]  signature 4

For further argument and output information, consult \texttt{pls_lrCMA}.

\begin{itemize}
  \item \texttt{pls_rfCMA} \hspace{2cm} \textit{Partial Least Squares followed by random forests}
\end{itemize}

Description

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

For S4 method information, see \texttt{pls_rfCMA-methods}.

Usage

\texttt{pls_rfCMA(X, y, f, learnind, comp = 2 * nlevels(as.factor(y)), seed = 111, models=FALSE, \ldots)}

Arguments

\begin{itemize}
  \item \texttt{X} 
    Gene expression data. Can be one of the following:
    \begin{itemize}
      \item A \texttt{matrix}. Rows correspond to observations, columns to variables.
      \item A \texttt{data.frame}, when \texttt{f} is not missing (s. below).
      \item An object of class \texttt{ExpressionSet}.
    \end{itemize}
  \item \texttt{y} 
    Class labels. Can be one of the following:
    \begin{itemize}
      \item A \texttt{numeric} vector.
      \item A \texttt{factor}.
      \item A \texttt{character} if \texttt{X} is an \texttt{ExpressionSet} that specifies the phenotype variable.
      \item \texttt{missing}, if \texttt{X} is a \texttt{data.frame} and a proper formula \texttt{f} is provided.
    \end{itemize}
  \item \texttt{f} 
    A two-sided formula, if \texttt{X} is a \texttt{data.frame}. The left part correspond to class labels, the right to variables.
  \item \texttt{learnind} 
    An index vector specifying the observations that belong to the learning set. May be \texttt{missing}; in that case, the learning set consists of all observations and predictions are made on the learning set.
  \item \texttt{comp} 
    Number of Partial Least Squares components to extract. Default ist two times the number of different classes.
  \item \texttt{seed} 
    Fix Random number generator seed to \texttt{seed}. This is useful to guarantee reproducibility of the results, due to the random component in the random Forest.
  \item \texttt{models} 
    a logical value indicating whether the model object shall be returned
  \item \texttt{\ldots} 
    Further arguments to be passed to \texttt{randomForests} from the package of the same name.
\end{itemize}
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

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

See Also

`compBoostCMA`, `dlldaCMA`, `ElasticNetCMA`, `fdacMA`, `flexdaCMA`, `gbmCMA`, `knnCMA`, `ldaCMA`, `lassoCMA`, `nnetCMA`, `pknnCMA`, `plrCMA`, `pls_ldacMA`, `pls_lrCMA`, `pnnCMA`, `qdaCMA`, `rfCMA`, `scdacMA`, `shrinkldaCMA`, `svmCMA`

Examples

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

---

pnnCMA

Probabilistic Neural Networks

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, models = FALSE)

Arguments

X 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.
  Each variable (gene) will be scaled for unit variance and zero mean.

y 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 not be missing.

sigma Standard deviation of the Gaussian Kernel used.
This hyperparameter should be tuned, s. tune. The default is 1, but this generally 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.

models a logical value indicating whether the model object shall be returned
Value
An object of class \texttt{cloutput}.

Note
There is actually no strong relation of this method to Feed-Forward Neural Networks, s. \texttt{nnetCMA}.

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

References

See Also
\texttt{compBoostCMA}, \texttt{dlldaCMA}, \texttt{ElasticNetCMA}, \texttt{fdaCMA}, \texttt{flexdaCMA}, \texttt{gbmCMA}, \texttt{knnCMA}, \texttt{ldaCMA}, \texttt{LassoCMA}, \texttt{nnetCMA}, \texttt{pknnCMA}, \texttt{plRCMA}, \texttt{pls\_ldaCMA}, \texttt{pls\_lrCMA}, \texttt{pls\_rfCMA}, \texttt{qdaCMA}, \texttt{rfCMA}, \texttt{scdaCMA}, \texttt{shrinkldaCMA}, \texttt{svmCMA}

Examples
```r
### 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)
learnind <- sample(length(golubY), size=floor(ratio*length(golubY)))
### run PNN
pnnresult <- pnnCMA(X=golubX, y=golubY, learnind=learnind, sigma = 3)
### show results
show(pnnresult)
ftable(pnnresult)
plot(pnnresult)
```

\texttt{pnnCMA-methods} \hspace{1cm} \textit{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.

Description

This method constructs the given classifier using the specified training data, gene selection and tuning results. Subsequently, class labels are predicted for new observations.

Usage

prediction(X.tr,y.tr,X.new,f,classifier,genesel,models=F,nbgene,tuneres,...)

Arguments

X.tr
Training 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.

X.new
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.

y.tr
Class labels of training observation. 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 for classifier construction 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.

genesel
Optional (but usually recommended) object of class genesel containing variable importance information for the argument learningsets. In this case the object contains a single variable selection. Appropriate genesel-objects can be obtained using the function genesel without learningset and setting X=X.tr and y=y.tr (i.e. corresponding to the training data of this function).
prediction

**nbgene**

Number of best genes to be kept for classification, based on either genesel or the call to **GeneSelection** using genesel list. 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 each 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.

**tuneres**

Analogous to the argument genesel - object of class `tuningresult` containing information about the best hyperparameter choice for the argument learning sets. Appropriate tuning-objects can be obtained using the function tune without learning sets and setting parameters `X=X.tr, y=y.tr` and `genesel=genesel` (i.e. using the same training data and gene selection as in this function)

**models**

a logical value indicating whether the model object shall be returned

**Details**

This function builds the specified classifier and predicts the class labels of new observations. Hence, its usage differs from those of most other prediction functions in R.

**Value**

A object of class `predoutput-class`; Predicted classes can be seen by `show(predoutput)`

**Author(s)**

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

**References**


**See Also**

GeneSelection, tune, evaluation, compBoostCMA, dldaCMA, ElasticNetCMA, fdaCMA, flexdaCMA, gbmCMA, knnCMA, ldaCMA, LassoCMA, nnetCMA, pknnCMA, plrCMA, pls_ladCMA, pls_lrcma, pls_rfcma, pnnCMA, qdaCMA, rfcma, scdCMA, shrinkldaCMA, svmCMAclassification

**Examples**

```r
### a simple k-nearest neighbour example
### datasets
## Not run: plot(x)
data(golub)
golubY <- golub[,1]
golubX <- as.matrix(golub[,-1])
```
### Splitting data into training and test set

```r
X.tr<-golubX[1:30]
X.new<-golubX[31:39]
y.tr<-golubY[1:30]
```

### 1. GeneSelection

```r
selttest <- GeneSelection(X=X.tr, y=y.tr, method = "t.test")
```

### 2. tuning

```r
tunek <- tune(X.tr, y.tr, genesel = selttest, nbgene = 20, classifier = knnCMA)
```

### 3. classification

```r
pred <- prediction(X.tr=X.tr,y.tr=y.tr,X.new=X.new, genesel = selttest, tuneres = tunek, nbgene = 20, classifier = knnCMA)
```

### show and analyze results:

```r
show(pred)
```

```r
## End(Not run)
```

---

**prediction-methods**  
*General method for predicting label of new observations*

**Description**

Perform prediction signatures:

**Methods**

- `X.tr = "matrix", X.new="matrix", y.tr='any', f = "missing"` signature 1
- `X.tr = "data.frame", X.new="data.frame", y.tr = "missing", f = "formula"` signature 2
- `X.tr = "ExpressionSet", X.new = "ExpressionSet", y.tr = "character", f = "missing"` signature 3

For further argument and output information, consult `classification`.

---

**predoutput-class**  
*"predoutput"

**Description**

Object returned by the function `prediction`

**Slots**

- `Xnew`: Gene Expression matrix of new observations
- `yhat`: Predicted class labels for the new data.
- `model`: List containing the constructed classifier.

**Methods**

- `show` Returns predicted class labels for the new data.
qdaCMA

Author(s)
Christoph Bernau <bernaue@ibe.med.uni-muenchen.de>
Anne-Laure Boulesteix <boulesteix@ibe.med.uni-muenchen.de>

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

qdaCMA

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.
For S4 method information, see qdaCMA-methods.

Usage
qdaCMA(X, y, f, learnind, models=FALSE, ...)

Arguments
X  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.

y  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.

models  a logical value indicating whether the model object shall be returned

...  Further arguments to be passed to qda from the package MASS
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>

References


See Also

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

Examples

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

```
qdaCMA-methods

```r
qdaresult <- qdaCMA(X=khanX, y=khanY, learnind=learnind)
### show results
show(qdaresult)
ftable(qdaresult)
plot(qdaresult)
```

### qdaCMA-methods

#### Quadratic Discriminant Analysis

**Description**

Performs a quadratic discriminant analysis under the assumption of a multivariate normal distribution in each class 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`.

---

**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**

```r
rfCMA(X, y, f, learnind, varimp = TRUE, seed = 111, models=FALSE,type=1, scale=FALSE,importance=TRUE, ...)```

**Arguments**

- `X` 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`.
- `y` 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 \texttt{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 \texttt{data.frame}. The left part correspond to class labels, the right to variables.

\( \text{learnind} \) 
An index vector specifying the observations that belong to the learning set. May be \texttt{missing}; in that case, the learning set consists of all observations and predictions are made on the learning set.

\( \text{varimp} \) 
Should additional information for variable selection be provided? Defaults to \texttt{TRUE}.

\( \text{seed} \) 
Fix Random number generator seed to \( \text{seed} \). This is useful to guarantee reproducibility of the results.

\( \text{models} \) 
a logical value indicating whether the model object shall be returned

\( \text{type} \) 
Parameter passed to function \texttt{importance}. Either 1 or 2, specifying the type of importance measure (1=mean decrease in accuracy, 2=mean decrease in node impurity).

\( \text{scale} \) 
Parameter passed to function \texttt{importance}. For permutation based measures, should the measures be divided by their standard errors?

\( \text{importance} \) 
Parameter passed to function \texttt{randomForest}. Should importance of predictors be assessed by permutation?

\( \ldots \) 
Further arguments to be passed to \texttt{randomForest} from the package of the same name.

**Value**
If \( \text{varimp} \), then an object of class \texttt{clvarseloutput} is returned, otherwise an object of class \texttt{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**
\texttt{compBoostCMA}, \texttt{dldaCMA}, \texttt{ElasticNetCMA}, \texttt{fdaCMA}, \texttt{flexdaCMA}, \texttt{gbmCMA}, \texttt{knncMA}, \texttt{ldaCMA}, \texttt{LassoCMA}, \texttt{nnetCMA}, \texttt{pknnCMA}, \texttt{plRCMA}, \texttt{pls ldaCMA}, \texttt{pls lrCMA}, \texttt{pls rFCMA}, \texttt{pnnCMA}, \texttt{qdaCMA}, \texttt{scdaCMA}, \texttt{shrinkldaCMA}, \texttt{svmCMA}
Examples

```r
### 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)
```

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

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 classifiers. 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 specify 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


See Also

evaluation

---

cldaCMA | 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*.

For S4 method information, see scdaCMA-methods.

Usage

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

Arguments

- **X** 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.

- **y** 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.

models  a logical value indicating whether the model object shall be returned

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>


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 learning set
set.seed(111)
learnind <- sample(length(khanY), size=floor(2/3*length(khanY)))
### run Shrunken Centroids classifier, without tuning
scdaresult <- scdaCMA(X=khanX, y=khanY, learnind=learnind)
### show results
show(scdaresult)
ftable(scdaresult)
plot(scdaresult)
scdaCMA-methods  Shrunken Centroids Discriminant Analysis

Description

The nearest shrunken centroid classification algorithm is detailedly 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.

shrinkldaCMA  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

Usage

shrinkldaCMA(X, y, f, learnind, models=FALSE, ...)

Arguments

X  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.

y  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.
**shrinkldaCMA**

**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.

models
A logical value indicating whether the model object shall be returned

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.

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

References
A shrinkage approach to large-scale covariance estimation and implications for functional genomics.
*Statistical Applications in Genetics and Molecular Biology, 4*:32.

See Also
`compBoostCMA`, `dldaCMA`, `ElasticNetCMA`, `fdaCMA`, `flexdaCMA`, `gbmCMA`, `knnCMA`, `LdaCMA`, `LassoCMA`, `nnetCMA`, `pknnCMA`, `plrCMA`, `pls_LdaCMA`, `pls_lrCMA`, `pls_rfCMA`, `pnnCMA`, `qdaCMA`, `rfCMA`, `scdaCMA`, `svmCMA`.

Examples
```r
### 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 shrinkage-LDA
result <- shrinkldaCMA(X=golubX, y=golubY, learnind=learnind)
```
### show results
show(result)
ftable(result)
plot(result)

---

**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.

---

**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-point-summary, 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.

**Value**

No return.

**Note**

That the results normally differ for different evaluation schemes ("iterationwise" or "observation-wise").
svmCMA

Author(s)

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

See Also
evaluation, compare, obsinfo.

---

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, models=FALSE, seed=341,...)

Arguments

X 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.
y 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.
seed Fix random number generator for reproducibility.
models a logical value indicating whether the model object shall be returned
...

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 reasonable 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


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

```r
### 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 untuned linear SVM
svmresult <- svmCMA(X=golubX, y=golubY, learnind=learnind, probability=TRUE)
### show results
show(svmresult)
ftable(svmresult)
plot(svmresult)
```
Support Vector Machine

**Description**

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

**Methods**

- \( X = \text{"matrix"}, y = \text{"numeric"}, f = \text{"missing"} \) signature 1
- \( X = \text{"matrix"}, y = \text{"factor"}, f = \text{"missing"} \) signature 2
- \( X = \text{"data.frame"}, y = \text{"missing"}, f = \text{"formula"} \) signature 3
- \( X = \text{"ExpressionSet"}, y = \text{"character"}, f = \text{"missing"} \) signature 4

For further argument and output information, consult `svmCMA`.

**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**

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

**Arguments**

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

**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


See Also

genesel, GeneSelection, plot, genesel-method

---

**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, fold = 3, strat = FALSE, grids = list(), trace = TRUE, ...)

**Arguments**

- **X** 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.

- **y** 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 variable 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.
tune

\textbf{nbgene} \quad \text{Number of best genes to be kept for classification, based on either genesel or the call to \texttt{GeneSelection} using genesel\texttt{list}. In the case that both are missing, this argument is not necessary. \textbf{note:}}

- If the gene selection method has been one of "lasso", "elasticnet", "boosting", \texttt{nbgene} will be reset to \texttt{min(s, nbgene)} where \texttt{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 \texttt{nbgene} will be kept of each of them, so the number of effective used genes will sometimes be much larger.

\textbf{classifier} \quad \text{Name of function ending with \texttt{CMA} indicating the classifier to be used.}

\textbf{fold} \quad \text{The number of cross-validation folds used within each \texttt{learningset}. Default is 3. Increasing \texttt{fold} will lead to higher computing times.}

\textbf{strat} \quad \text{Should stratified cross-validation according to the class proportions in the complete dataset be used? Default is \texttt{FALSE}.}

\textbf{grids} \quad \text{A named list. The names correspond to the arguments to be tuned, e.g. \texttt{k} (the number of nearest neighbours) for \texttt{knnCMA}, or cost for \texttt{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.}

\textbf{trace} \quad \text{Should progress be traced? Default is \texttt{TRUE}.}

\textbf{...} \quad \text{Further arguments to be passed to classifier, of course not one of the arguments to be tuned (!).}

\textbf{Details}

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

\begin{itemize}
  \item \texttt{gbmCMA n.trees = c(50, 100, 200, 500, 1000)}
  \item \texttt{compBoostCMA mstop = c(50, 100, 200, 500, 1000)}
  \item \texttt{LassoCMA norm.fraction = seq(from=0.1, to=0.9, length=9)}
  \item \texttt{ElasticNetCMA norm.fraction = seq(from=0.1, to=0.9, length=5), alpha = 2^{-(5:1)}}
  \item \texttt{plrCMA lambda = 2^{-4:4}}
  \item \texttt{plsldaCMA comp = 1:10}
  \item \texttt{plslrcMA comp = 1:10}
  \item \texttt{plsrfCMA comp = 1:10}
  \item \texttt{rfCMA mtry = ceiling(c(0.1, 0.25, 0.5, 1, 2)*sqrt(ncol(X))), nodesize = c(1,2,3)}
  \item \texttt{knnCMA k=1:10}
  \item \texttt{pknnCMA k = 1:10}
  \item \texttt{scdaCMA delta = c(0.1, 0.25, 0.5, 1, 2, 5)}
  \item \texttt{pnnCMA sigma = c(2^{(-2:2)})}
  \item \texttt{nnetCMA size = 1:5, decay = c(0, 2^{(-4:1)})}
  \item \texttt{svmCMA, kernel = "linear" cost = c(0.1, 1, 5, 10, 50, 100, 500)}
  \item \texttt{svmCMA, kernel = "radial" cost = c(0.1, 1, 5, 10, 50, 100, 500), gamma = 2^{(-2:2)}}
  \item \texttt{svmCMA, kernel = "polynomial" cost = c(0.1, 1, 5, 10, 50, 100, 500), degree = 2:4}
\end{itemize}
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 \( \text{fold} \times \text{times number of possible different hyperparameter combinations} \times \text{times. E.g. if the number of the learningsets is fifty,} \text{fold = 3} \) and two hyperparameters (each with 5 candidate values) are tuned, \( 50 \times 3 \times 25 = 3750 \) training iterations are necessary!

Author(s)

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

References


See Also

tuningresult, GeneSelection, classification

Examples

```r
## Not run:
### simple example for a one-dimensional grid, using compBoostCMA.
### dataset
data(golub)
golubY <- golub[,1]
golubX <- as.matrix(golub[,-1])
### learningsets
set.seed(111)
lset <- GenerateLearningsets(y=golubY, method = "CV", fold=5, strat =TRUE)
### tuning after gene selection with the t.test
tuneres <- tune(X = golubX, y = golubY, learningsets = lset,
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)
```
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.

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 hyperparmater/hyperparameter combination, use the convenience method best

method: Name of the classifier that has been tuned.

tfold: Number of cross-validation fold used for tuning, s. argument of the same name in tune

Methods

show Use show(tuningresult-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,tuningresult-method

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

weighted.mcr  Tuning / Selection bias correction

Description

Performs subsampling for several classifiers or a single classifier with different tuning parameter values or numbers of selected genes. Eventually, a specific procedure for correcting for the tuning or selection bias, which is caused by optimal selection of classifiers or tuning parameters, is applied.

Usage

weighted.mcr(classifiers, parameters, nbgenes, sel.method, X, y, portion, niter=100, shrinkage=F)
Arguments

classifiers A character vector of the several CMA classifiers that shall be used. If the same classifier shall be used with different tuning parameters it must appear several times in this vector.

parameters A character containing the tuning parameter values corresponding to the classification methods in classifiers. Must have the same length as classifiers.

nbgenes A numeric vector indicating how many variables shall be selected by sel.method for the corresponding classifier. Must have the same length as classifiers.

sel.method The CMA-method (represented as a string) that shall be applied for variable selection. If this parameter is set to 'none' no variable selection is performed.

X The matrix of gene expression data. Can be one of the following. Rows correspond to observations, columns to variables.

y Class labels. Can be one of the following:
  * A numeric vector.
  * A factor.

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.

portion A numeric value which indicates the portion of observations that will be used for training the classifiers.

niter The number of subsampling iterations.

shrinkage A logical value indicating whether shrinkage (WMCS) shall be applied.

Details

The algorithm tries to avoid the additional computational costs of a nested cross validation by estimating the corrected misclassification rate of the best classifier by a weighted mean of all classifiers included in the subsampling approach.

Value

An object of class wmcresult which provides the corrected and uncorrected misclassification rate of the best classifier as well as weights and misclassification rates for all classifiers used in the subsampling approach.

Author(s)

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

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

References


See Also

wmc, classification, GeneSelection, tune, evaluation,
Examples

#inputs
classifiers<-rep('knnCMA',7)
nbgenes<-rep(50,7)
parameters<-c('k=1','k=3','k=5','k=7','k=9','k=11','k=13')
portion<-0.8
niter<-100
data(golub)
X<-as.matrix(golub[,,-1])
y<-golub[,1]
sel.method<-'t.test'
#function call
wmcr<-weighted.mcr(classifiers=classifiers,parameters=parameters,nbgenes=nbgenes,sel.method=sel.method,X=X,y=y,portion=portion,niter=niter)

weighted.mcr-methods

General method for tuning / selection bias correction

Description

Perform tuning / selection bias correction in subsampling for the following signatures:

Methods

classifiers="character",parameters="character",nbgenes="numeric",sel.method="character",X = "matrix", y = signature 1
classifiers="character",parameters="character",nbgenes="numeric",sel.method="character",X = "matrix", y = signature 2
classifiers="character",parameters="character",nbgenes="missing",sel.method="character",X = "matrix", y = signature 3

For further argument and output information, consult weighted.mcr.

wmc

Tuning / Selection bias correction based on matrix of subsampling fold errors

Description

Perform tuning / selection bias correction for a matrix of subsampling fold errors.

Usage

wmc(mcr.m,n.tr,n.ts,shrinkage=F)

Arguments

mcr.m A matrix of resampling fold errors. Columns correspond the the fold errors of a single classifier.
n.tr Number of observations in the resampling training sets.
n.ts Number of observations in the resampling test sets.
shrinkage A logical value indicating whether shrinkage (WMCS) shall be applied.
Details

The algorithm tries to avoid the additional computational costs of a nested cross validation by estimating the corrected misclassification rate of the best classifier by a weighted mean of all classifiers included in the subsampling approach.

Value

A list containing the corrected misclassification rate, the index of the best method and a logical value indicating whether shrinkage has been applied.

Author(s)

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

References


See Also

weighted.mcr, classification, GeneSelection, tune, evaluation,

Description

Perform tuning / selection bias correction for a matrix of subsampling fold errors for the following signature:

Methods

mcr.m="matrix",n.tr="numeric",n.ts="numeric" signature 1

For further argument and output information, consult wmc.
`wmcr.result-class"` 

**Description**

Object returned by function `weighted.mcr`.

**Slots**

- `corrected.mcr`: The corrected misclassification rate for the best method.
- `best.method`: The method which performed best in the subsampling approach.
- `mcrs`: Misclassification rates of all classifiers used in the subsampling approach.
- `weights`: The weights used for the different classifiers in the correction method.
- `cov`: Estimated covariance matrix for the misclassification rates of the different classifiers.
- `uncorrected.mcr`: The uncorrected misclassification rate of the best method.
- `ranges`: Minimum and maximal mean misclassification rates as well as the theoretical bound for nested cross validation (averaging over foldwise minima or maxima respectively).
- `mcr.m`: Matrix of resampling fold errors, columns correspond to the fold errors of a single classifier.
- `shrinkage`: A logical value indicating whether shrinkage (WMCS) has been applied.

**Methods**

- `show`: Use `show(wmcr.result-object)` for brief information

**Author(s)**

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

**See Also**

`weighted.mcr`
Index

*Topic datasets
  golub, 38
  khan, 40
*Topic multivariate
  Barplot, 4
  best, 5
  boxplot, 6
  classification, 7
  classification-methods, 9
  cloutput-class, 9
  clvareoutput-class, 10
  CMA-package, 3
  compare, 11
  compare-methods, 13
  compBoostCMA, 14
  compBoostCMA-methods, 16
  dldaCMA, 16
  dldaCMA-methods, 18
  ElasticNetCMA, 19
  ElasticNetCMA-methods, 21
  evaloutput-class, 21
  evaluation, 22
  evaluation-methods, 24
  fdaCMA, 24
  fdaCMA-methods, 26
  filter, 27
  flexdaCMA, 27
  flexdaCMA-methods, 29
  ftable, 30
  gbmCMA, 30
  gbmCMA-methods, 32
  GenerateLearningsets, 33
  genesel-class, 34
  GeneSelection, 35
  GeneSelection-methods, 38
  internals, 39
  join, 39
  join-methods, 40
  knnCMA, 41
  knnCMA-methods, 43
  LassoCMA, 43
  LassoCMA-methods, 45
  ldaCMA, 45
  ldaCMA-methods, 47
  learningsets-class, 47
  nnetCMA, 48
  nnetCMA-methods, 50
  obsinfo, 51
  pknnCMA, 52
  pknnCMA-methods, 53
  Planarplot, 54
  Planarplot-methods, 55
  plot, 56
  plot tuningresult, 57
  plrCMA, 58
  plrCMA-methods, 59
  pls_ldaCMA, 60
  pls_ldaCMA-methods, 61
  pls_lrCMA, 62
  pls_lrCMA-methods, 63
  pls_rfCMA, 64
  pls_rfCMA-methods, 65
  pnnCMA, 66
  pnnCMA-methods, 67
  prediction, 68
  prediction-methods, 70
  predoutput-class, 70
  qdaCMA, 71
  qdaCMA-methods, 73
  rfCMA, 73
  rfCMA-methods, 75
  roc, 75
  scdaCMA, 76
  scdaCMA-methods, 78
  shrinkldaCMA, 78
  shrinkldaCMA-methods, 80
  summary, 80
  svmCMA, 81
  svmCMA-methods, 83
  toplist, 83
  tune, 84
  tune-methods, 87
  tuningresult-class, 87
  varseloutput-class, 88

*Topic tuning bias, selection bias, corrected misclassification
nnetCMA.data.frame,missing,formula-method (nnetCMA-methods), 50
nnetCMA.ExpressionSet,character,missing-method (nnetCMA-methods), 50
nnetCMA,matrix,missing-method (nnetCMA-methods), 50
nnetCMA,matrix,numeric,missing-method (nnetCMA-methods), 50
nnetCMA-methods, 48, 50

obsinfo, 21, 51, 81
obsinfo,evaloutput-method (evaloutput-class), 21

pknnCMA, 4, 8, 10, 15, 17, 20, 25, 29, 32, 42, 44, 46, 49, 52, 53, 55, 59, 61, 63, 65, 67, 69, 71, 72, 74, 77, 79, 82, 85
pknnCMA.data.frame,missing.formula-method (pknnCMA-methods), 53
pknnCMA,ExpressionSet,character,missing-method (pknnCMA-methods), 53
pknnCMA,matrix,missing-method (pknnCMA-methods), 53
pknnCMA,matrix,numeric,missing-method (pknnCMA-methods), 53
pknnCMA-methods, 52, 53
Planarplot, 54, 55
Planarplot.data.frame,missing.formula-method (Planarplot-methods), 55
Planarplot,ExpressionSet,character,missing-method (Planarplot-methods), 55
Planarplot,matrix,missing-method (Planarplot-methods), 55
Planarplot,matrix,numeric,missing-method (Planarplot-methods), 55
Planarplot-methods, 55
plot, 56
plot,tuningresult, 57
plot,cloutput,missing-method(plot), 56
plot,cloutput-method(plot), 56
plot,genesel,missing-method(Barpplot), 4
plot,genesel-method(Barpplot), 4
plot,tuningresult,missing-method(plot
tuningresult), 57
plot,tuningresult-method(plot
tuningresult), 57
plotprob(internals), 39
plrCMA, 4, 8, 10, 15, 17, 20, 25, 29, 32, 42, 44, 46, 49, 53, 55, 59, 60, 61, 63, 65, 67, 69, 71, 72, 74, 77, 79, 82, 85
plrCMA.data.frame,missing.formula-method (plrCMA-methods), 59
plrCMA,ExpressionSet,character,missing-method (plrCMA-methods), 59
plrCMA,matrix, factor,missing-method (plrCMA-methods), 59
plrCMA,matrix,numeric,missing-method (plrCMA-methods), 59
plrCMA-methods, 58, 59
pls_ldaCMA.data.frame,missing.formula-method (pls_ldaCMA-methods), 61
pls_ldaCMA,ExpressionSet,character,missing-method (pls_ldaCMA-methods), 61
pls_ldaCMA,matrix, factor,missing-method (pls_ldaCMA-methods), 61
pls_ldaCMA,matrix,numeric,missing-method (pls_ldaCMA-methods), 61
pls_ldaCMA-methods, 61
pls_lrCMA, 4, 8, 10, 15, 17, 20, 25, 29, 32, 42, 44, 46, 49, 53, 55, 59, 61, 62, 64, 65, 67, 69, 71, 72, 74, 77, 79, 82, 85
pls_lrCMA.data.frame,missing.formula-method (pls_lrCMA-methods), 63
pls_lrCMA,ExpressionSet,character,missing-method (pls_lrCMA-methods), 63
pls_lrCMA,matrix, factor,missing-method (pls_lrCMA-methods), 63
pls_lrCMA,matrix,numeric,missing-method (pls_lrCMA-methods), 63
pls_lrCMA-methods, 63
pls_rfCMA, 4, 8, 10, 15, 17, 20, 25, 29, 32, 42, 44, 46, 49, 53, 55, 59, 61, 63, 64, 66, 67, 69, 71, 72, 74, 77, 79, 82, 85
pls_rfCMA.data.frame,missing.formula-method (pls_rfCMA-methods), 65
pls_rfCMA,ExpressionSet,character,missing-method (pls_rfCMA-methods), 65
pls_rfCMA,matrix, factor,missing-method (pls_rfCMA-methods), 65
pls_rfCMA,matrix,numeric,missing-method (pls_rfCMA-methods), 65
pls_rfCMA-methods, 65
pnnCMA, 4, 8, 10, 15, 17, 20, 25, 29, 32, 42, 44, 46, 49, 53, 55, 59, 61, 63, 65, 68, 69, 71, 72, 74, 77, 79, 82, 85
pnnCMA.data.frame,missing.formula-method (pnnCMA-methods), 67
pnnCMA,ExpressionSet,character,missing-method (pnnCMA-methods), 67
pnnCMA,matrix, factor,missing-method (pnnCMA-methods), 67
INDEX

pnnCMA, matrix, numeric, missing-method (pnnCMA-methods, 67
pnnCMA-methods, 66, 67
prediction, 68
prediction, data.frame, missing, data.frame, formula-method (prediction-methods, 70
prediction, ExpressionSet, character, ExpressionSet, missing-method (prediction-methods, 70
prediction, matrix, ANY, matrix, missing-method (prediction-methods, 70
prediction-methods, 70
predoutput (predoutput-class, 70
predoutput-class, 70
qdaCMA, 4, 8, 10, 15, 17, 20, 25, 29, 32, 42, 44, 46, 49, 53, 55, 59, 61, 63, 65, 67, 69, 71, 71, 73, 74, 77, 79, 82
qdaCMA, data.frame, missing, formula-method (qdaCMA-methods, 73
qdaCMA, ExpressionSet, character, missing-methods (qdaCMA-methods, 73
qdaCMA, matrix, factor, missing-method (qdaCMA-methods, 73
qdaCMA, matrix, numeric, missing-method (qdaCMA-methods, 73
qdaCMA-methods, 71, 73
rfCMA, 4, 8, 10, 11, 15, 17, 20, 25, 29, 32, 42, 44, 46, 49, 53, 55, 59, 61, 63–65, 67, 69, 71, 72, 73, 75, 77, 79, 82, 85
rfCMA, data.frame, missing, formula-method (rfCMA-methods, 75
rfCMA, ExpressionSet, character, missing-methods (rfCMA-methods, 75
rfCMA, matrix, factor, missing-method (rfCMA-methods, 75
rfCMA, matrix, numeric, missing-method (rfCMA-methods, 75
rfCMA-methods, 73, 75
rfe (filter), 27
roc, 75
roc, cloutput-method (roc), 75
ROCinternal (internals), 39
ROCloutput (internals), 39
rowswaps (internals), 39
safeexp (internals), 39
scdaCMA, 4, 8, 10, 15, 17, 20, 25, 29, 32, 42, 44, 46, 49, 53, 55, 59, 61, 63, 65, 67, 69, 71, 72, 74, 76, 78, 79, 82, 85
scdaCMA, data.frame, missing, formula-method (scdaCMA-methods, 78
scdaCMA, ExpressionSet, character, missing-method (scdaCMA-methods, 78
scdaCMA, matrix, factor, missing-method (scdaCMA-methods, 78
scdaCMA, matrix, numeric, missing-method (scdaCMA-methods, 78
scdaCMA-methods, 78
show, cloutput-method (cloutput-class, 9
show, evaloutput-method (evaloutput-class, 21
show, genesel-method (genesel-class, 34
show, learningsets-method (learningsets-class, 47
show, predoutput-method (predoutput-class, 70
show, tuningresult-method (tuningresult-class, 87
show, wncr.result-method (wncr.result-class, 92
shrinkldaCMA (filter), 27
shrinkldaCMA, 4, 8, 10, 15, 17, 20, 25, 29, 32, 42, 44, 46, 49, 53, 55, 59, 61, 63, 65, 67, 69, 71, 72, 74, 77, 78, 80, 82
shrinkldaCMA, data.frame, missing, formula-method (shrinkldaCMA-methods, 80
shrinkldaCMA, ExpressionSet, character, missing-method (shrinkldaCMA-methods, 80
shrinkldaCMA, matrix, factor, missing-method (shrinkldaCMA-methods, 80
shrinkldaCMA, matrix, numeric, missing-method (shrinkldaCMA-methods, 80
shrinkldaCMA-methods, 78, 80, 80
summary, 80
summary, evaloutput-method (summary, 80
svmCMA, 4, 8, 10, 15, 17, 20, 25, 29, 32, 42, 44, 46, 49, 53, 55, 59, 61, 63, 65, 67, 71, 72, 74, 77, 79, 81, 83, 85
svmCMA, data.frame, missing, formula-method (svmCMA-methods, 83
svmCMA, ExpressionSet, character, missing-method (svmCMA-methods, 83
svmCMA, matrix, factor, missing-method (svmCMA-methods, 83
svmCMA, matrix, numeric, missing-method (svmCMA-methods, 83
svmCMA-methods, 81, 83
toplist, 4, 5, 35, 83
toplist, genesel-method (toplist), 83
ttest (filter), 27
tune, 4, 8, 14, 31, 34, 38, 47, 48, 57, 58, 60, 62, 66, 69, 82, 84, 87–89, 91
INDEX

tune, data.frame, missing, formula-method
(tune-methods), 87
tune, ExpressionSet, character, missing-method
(tune-methods), 87
tune, matrix, factor, missing-method
(tune-methods), 87
tune, matrix, numeric, missing-method
(tune-methods), 87
tune-methods, 87
tuningresult, 5, 8, 57, 69, 86
tuningresult (tuningresult-class), 87
tuningresult-class, 87

varseloutput, 11, 27
varseloutput (varseloutput-class), 88
varseloutput-class, 88

weighted.mcr, 88, 90, 91
weighted.mcr, character, character, missing, character, matrix, factor-method
(weighted.mcr-methods), 90
weighted.mcr, character, character, numeric, character, matrix, factor-method
(weighted.mcr-methods), 90
weighted.mcr, character, character, numeric, character, matrix, numeric-method
(weighted.mcr-methods), 90
weighted.mcr-methods, 90
welchtest (filter), 27
wilcoxtest (filter), 27
wmc, 89, 90, 91
wmc, matrix, numeric, numeric-method
(wmc-methods), 91
wmc-methods, 91
wmcr.result (wmcr.result-class), 92
wmcr.result-class, 92