Package 'switchBox'

April 23, 2016

Version 1.4.0

Date 2014-03-20

Title Utilities to train and validate classifiers based on pair switching using the K-Top-Scoring-Pair (KTSP) algorithm.

Author Bahman Afsari <bahman@jhu.edu>, Luigi Marchionni <marchion@jhu.edu>

Maintainer Bahman Afsari <bahman@jhu.edu>, Luigi Marchionni <marchion@jhu.edu>

Depends R (>= 2.13.1)

Description The package offer different classifiers based on comparisons of pair of features (TSP), using various decision rules (e.g., majority wins principle).

biocViews Software, StatisticalMethod, Classification

License GPL-2

NeedsCompilation yes

R topics documented:

switchBox-package 2
KTSP.Classifiy 3
KTSP.Train 4
matTesting 6
matTraining 7
SWAP.CalculateSignedScore 8
SWAP.Filter.Wilcoxon 10
SWAP.KTSP.Classifiy 11
SWAP.KTSP.Statistics 13
SWAP.KTSP.Train 16
testingGroup 19
trainingGroup 20

Index **22**

switchBox-package *A package to train and apply K-Top-Scoring-Pair (KTSP) classifiers.*

Description

The switchBox package allows to train and apply a K-Top-Scoring-Pair (KTSP) classifier with learning mechanism proposed in Afsari et al (AOAS, 2014) and as used by Marchionni et al (BMC Genomics, 2013). KTSP is an extension of the TSP classifier described by Geman and colleagues (Bioinformatics, 2005). The TSP algorithm is a simple binary classifier based on the reversal ordering across phenotypes of two measurements (e.g. gene expression reversals from normal to cancer.

switchBox package features

The switchBox package contains several utilities enabling to:

A) Filter the features to be used to develop the classifier (*i.e.*, differentially expressed genes);

B) Compute the scores for all available feature pairs to identify the top performing TSP;

C) Compute the scores for selected feature pairs to identify the top performing TSP;

D) Identify the number of K TSP to be used in the final classifier using the analysis of variance;

E) Compute individual TSP votes for one class or the other and combine the votes based on user defined methods;

F) Classify new samples based on the top KTSP based on various methods.

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>

References

Afsari et al., "Rank Discriminants for Predicting Phenotypes from RNA Expression.", *Annals of Applied Statistics*, 2014, to appear.

Marchionni et al., "A simple and reproducible breast cancer prognostic test.", *BMC Genomics*, 2013, **14**(1):336-342 http://www.ncbi.nlm.nih.gov/pubmed/23682826

Tan et al., "Simple decision rules for classifying human cancers from gene expression profiles.", *Bioinformatics* (2005) **21**(20), 3896-3904. http://www.ncbi.nlm.nih.gov/pubmed/16105897

Xu et al., "Robust prostate cancer marker genes emerge from direct integration of inter-study microarray data" *Bioinformatics* (2005) **21**(20), 3905-3911. http://www.ncbi.nlm.nih.gov/pubmed/16131522

Geman et al. "Classifying gene expression profiles from pairwise mRNA comparisons" *Statistical applications in genetics and molecular biology* (2004) **3**.1 : 1071. http://www.ncbi.nlm.nih.gov/pubmed/16646797

KTSP.Classifiy *Function to classify samples using a KTSP classifier.*

Description

KTSP.Classify classifies new test samples using KTSP coming out of the function KTSP.Train. This function was used in Marchionni et al, 2013, BMC Genomics, and it is maintained only for backward compatibility. It has been replaced by SWAP.KTSP.Classify.

Usage

```
KTSP.Classify(data, classifier, combineFunc)
```

Arguments

data	the test data: a matrix in which the rows represent the genes and the columns the samples.
classifier	The output of KTSP.Train, a KTSP classifier.
combineFunc	A user defined function to combine the predictions of the individual K TSPs. If missing the consensus classification among the majority of the TSPs will be used.

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>

References

See switchBox for the references.

See Also

KTSP.Train, SWAP.KTSP.Classify,

Examples

```
##################################################
### Load gene expression data for the training set
data(trainingData)

### Turn into a numeric vector with values equal to 0 and 1
trainingGroupNum <- as.numeric(trainingGroup) - 1

### Show group variable for the TRAINING set
table(trainingGroupNum)
```

```
################################################
### Train a classifier using default filtering function based on the Wilcoxon test
classifier <- KTSP.Train(matTraining, trainingGroupNum, n=8)

### Show the classifier
classifier


################################################
### Testing on new data

### Load the example data for the TEST set
data(testingData)

### Turn into a numeric vector with values equal to 0 and 1
testingGroupNum <- as.numeric(testingGroup) - 1

### Show group variable for the TEST set
table(testingGroupNum)

### Apply the classifier to one sample of the TEST set using
### sum of votes grearter than 2
testPrediction <- KTSP.Classify(matTesting, classifier,
     combineFunc = function(x) sum(x) < 2.5)

### Show prediction
table(testPrediction, testingGroupNum)
```

KTSP.Train *Funtion for training the K-TSP classifier.*

Description

KTSP.Train trains a K-TSP classifier for the specific phenotype of interest. The classifiers resulting from using this function can be passed to KTSP.Classify for samples classification. This function was used in Marchionni et al, 2013, BMC Genomics, and it is maintained only for backward compatibility. It has been replaced by SWAP.KTSP.Train.

Usage

```
KTSP.Train(data, situation, n)
```

Arguments

data	the matrix of the values (usually gene expression) to be used to train the classifier. The columns represents samples and the rows represents the genes.
situation	an integer vector containing the training labels. Its elements should be one or zero.

n	The number of disjoint TSP used for classification. If before n pairs, the score drops to zero, the TSP with zero score are ignored.

Value

The KTSP classifier, a list containing the following elements:

TSPs	a matrix containing TSPs indexes.
score	a vector containing TSPs scores.
geneNames	a matrix containing TSPs feature names.

It should be passed to KTSP.Classify for classification of test samples.

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>

References

See switchBox for the references.

See Also

KTSP.Classify, SWAP.KTSP.Train,

Examples

```
##################################################
### Load gene expression data for the training set
data(trainingData)

### Turn into a numeric vector with values equal to 0 and 1
trainingGroupNum <- as.numeric(trainingGroup) - 1

### Show group variable for the TRAINING set
table(trainingGroupNum)


##################################################
### Train a classifier using default filtering function based on the Wilcoxon test
classifier <- KTSP.Train(matTraining, trainingGroupNum, n=8)

### Show the classifier
classifier
```

matTesting *Gene expression matrix for test set data*

Description

A numerical matrix containing gene expression matrix for 70 genes and 307 breast cancer patients (test set data) from the Buyse et al cohort (see the mammaPrintData package).

Usage

```
data(testingData)
```

Format

The matTesting matrix contains normalized expression values for the 70 gene signature (rows) across 307 samples (columns). Group information (emph"bad" versus "good" prognosis) is shown in colnames(matTesting).

Details

This dataset corresponds to the breast cancer patients' cohort published by Buyse and colleagues in JNCI (2006). The gene expression matrix was obtained from the mammaPrintData package as described by Marchionni and colleagues in BMC Genomics (2013).

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>

References

See switchBox for the references.

See Also

matTraining

Examples

```
### Load gene expression data for the test set
data(testingData)

### Show the class of the ``matTesting'' object
class(matTesting)

### Show the dimentions of the ``matTesting'' matrix
dim(matTesting)

### Show the first  10 sample names of the ``matTest'' matrix
```

```
head(colnames(matTesting), n=10)
testingGroup[1:10]
```

---

matTraining *Gene expression matrix for training set data*

Description

A numerical matrix containing gene expression matrix for 70 genes and 78 breast cancer patients (training set data) from the Glas et al cohort (see the mammaPrintData package).

Usage

```
data(trainingData)
```

Format

The matTraining matrix contains normalized expression values for the 70 gene signature (rows) across 78 samples (columns). Group information (emph"bad" versus "good" prognosis) is shown in colnames(matTraining).

Details

This dataset corresponds to the breast cancer patients' cohort published by Glas and colleagues in BMC Genomics (2006). The gene expression matrix was obtained from the mammaPrintData package as described by Marchionni and colleagues in BMC Genomics (2013).

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>

References

See switchBox for the references.

See Also

matTesting

Examples

```
### Load gene expression data for the training set
data(trainingData)

### Show the class of the ``matTraining'' object
class(matTraining)
```

```
### Show the dimentions of the ``matTraining'' matrix
dim(matTraining)

### Show the first  10 sample names of the ``matTraining'' matrix
head(colnames(matTraining), n=10)
```

SWAP.CalculateSignedScore

Function to calculate the pair-wise scores.

Description

SWAP.CalculateSignedScore calculates the pair-wise scores between features pairs. The user may pass a filtering function to reduce the number of starting features, or provide a restricted set of pairs to limit the reported scores to this list.

Usage

```
SWAP.CalculateSignedScore(inputMat, phenoGroup,
  FilterFunc = SWAP.Filter.Wilcoxon, RestrictedPairs, ...)
```

Arguments

inputMat	is a numerical matrix containing the measurements (*e.g.*, gene expression data) to be used to build the K-TSP classifier. The columns represent samples and the rows represent the features (*e.g.*, genes). The number of columns must agree with the length of phenoGroup. Note that rownames(inputMat) will be construed as feature names (*e.g.*, gene symbols) in all subsequent analyses.
phenoGroup	is a factor containing the training phenotypes with two levels.
FilterFunc	is a filtering function to reduce the starting number of features to be used to identify the Top Scoring Pairs (TSPs). The default filter is based on the Wilcoxon rank-sum test and alternative filtering functions can be passed too (see SWAP.Filter.Wilcoxon for details). Note the filtering function must return feature names, i.e. a subset of rownames(inputMat).
RestrictedPairs	is a character matrix with two columns containing the feature pairs to be considered for score calculations. Each row should contain a pair of feature names matching the rownames(inputMat). If RestrictedPairs is missing all available feature pairs will be considered.
...	Additional argument passed to the filtering function FilterFunc.

Value

The output is a list containing the following items:

`labels`	the levels (phenotypes) in `phenoGroup`.
`P`	a matrix or a vector containing the probability of comparisons in samples with phenotype equal to `label[1]`. In case `RestrictedPairs` is not given, `P` is a matrix and `P[i,j] = P(InputMat1[i,] < InputMat2[j,] | phenoGroup == $label[1])`. In case `RestrictedPairs` is given, `P[k] = P(InputMat1[RestrictedPairs[k,1],] < InputMat2[RestrictedPairs[k,2],] | phenoGroup == $label[1])`.
`Q`	a matrix or a vector containing the probability of comparisons in samples with phenotype `label[2]`.
`score`	a matrix or a vector containing the pair-wise scores. Basically, `score = P - Q + C`. The `C` term is the tie breaker and proportion to the secondary score to avoid the ties.

Note that the `P`, `Q`, and `score` list elements are matrices when scores are computed for all possible feature pairs, while they are vectors when scores are computed for restricted pairs defined by `RestrictedPairs`.

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>

References

See switchBox for the references.

See Also

See `SWAP.KTSP.Train`, `SWAP.Filter.Wilcoxon`, and `SWAP.KTSP.Statistics`.

Examples

```
### Load gene expression data for the training set
data(trainingData)

### Show group variable for the TRAINING set
table(trainingGroup)

### Compute the scores using all features (a matrix will be returned)
scores <- SWAP.CalculateSignedScore(matTraining,  trainingGroup, FilterFunc=NULL, )

### Show scores
class(scores)
dim(scores$score)

### Get the scores for a couple of features
diag(scores$score[ 1:3 , 5:7 ])
```

```
### Compute the scores using the default filtering function for 20 features
scores <- SWAP.CalculateSignedScore(matTraining, trainingGroup, featureNo=20)

### Show scores
dim(scores$score)

### Creating some random pairs
set.seed(123)
somePairs <- matrix(sample(rownames(matTraining), 25, replace=FALSE), ncol=2)

### Compute the scores for restricted pairs (a vector will be returned)
scores <- SWAP.CalculateSignedScore(matTraining, trainingGroup,
    FilterFunc = NULL, RestrictedPairs = somePairs )

### Show scores
class(scores$score)
length(scores$score)
```

`SWAP.Filter.Wilcoxon` *Statistical feature filtering based on Wilcoxon test on the ranks of expressions.*

Description

`SWAP.Filter.Wilcoxon` filters the features to top differential expressed to be used for KTSP classifier implementation.

Usage

```
SWAP.Filter.Wilcoxon(phenoGroup, inputMat, featureNo = 100, UpDown = TRUE)
```

Arguments

`phenoGroup`	a factor with levels containing training labels for the phenotype of interest.
`inputMat`	a numerical matrix containing feature measurements to be used to implement the classifier (*e.g.*, the set of gene expression values). The columns of this matrix correspond to samples and must correspond to `phenoGroup`. The rows represent the features and `rownames(inputMat)` will be used as feature names.
`featureNo`	an integer specifying the number of different features to be returned.
`UpDown`	logical value specifying whether an equal proportion of features displaying opposite change across the two phenotypes should be returned (*e.g.*an equal number of up- and down-regulated genes).

Value

The names of the features that survived the statistical filtering, i.e. differential expressed features.

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>

References

See switchBox for the references.

See Also

SWAP.KTSP.Classify, SWAP.Filter.Wilcoxon, SWAP.CalculateSignedScore

Examples

```
### Load gene expression data for the training set
data(trainingData)

### Return equal numbers of up- and down- regulated features (default)
SWAP.Filter.Wilcoxon(trainingGroup, matTraining, featureNo=10)

### Return the top 10 differentially expressed features irrispective to
### the direction of change.
### By setting the argument 'UpDown' equal to FALSE the number of
### up- and down- regulated features can be different
SWAP.Filter.Wilcoxon(trainingGroup, matTraining, featureNo=10, UpDown=FALSE)
```

SWAP.KTSP.Classifiy *Function to classify samples using a KTSP classifier.*

Description

SWAP.KTSP.Classify classifies new test samples using KTSP coming out of the function SWAP.KTSP.Train.

Usage

```
SWAP.KTSP.Classify(inputMat, classifier, DecisionFunc)
```

Arguments

inputMat	is a numerical matrix containing the measurements (*e.g.*, gene expression data) to be used with a K-TSP classifier to classify the samples in a specific class or the other. In this numerical matrix the columns represent the samples and the rows represent the features (*e.g.*, genes) used by the classification rule. Note that rownames(inputMat) will be used to select the features (*e.g.*, gene symbols) contained in the K-TSP classifier.
classifier	the classifier obtained by invoking SWAP.KTSP.Train.

DecisionFunc is the function used to generate the final classification prediction by combining the comparisons of the TSPs in the classifier. By default each sample is classified according to the class voted by the majority of the TSPs ("majority wins" principle). Different decision rules can be also specified using alternative functions passed DecisionFunc, as described below (see "details").

Details

The SWAP.KTSP.Classify classifies new test samples based on a specific decision rule. By default, each sample is classified based on the the majority voting rule of the comparisons of TSPs in the classifier. Alternative rules can be defined by the user and passed to SWAP.KTSP.Classify using the argument DecisionFunc. A decision function takes as its input a logical vector x corresponding to the individual decision of each TSP (TRUE if the first feature in the pair is larger then the second, FALSE in the opposite case). The output of the DecisionFunction is a single logical value summarizing all votes of the individual TSPs (see examples below).

Value

This function returns the predicted class for each sample in the form of a factor.

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>

References

See switchBox for the references.

See Also

SWAP.KTSP.Train, SWAP.Filter.Wilcoxon, SWAP.CalculateSignedScore

Examples

```
##################################################
### Load gene expression data for the training set
data(trainingData)


### Show group variable for the TRAINING set
table(trainingGroup)


##################################################
### Train a classifier using default filtering function based on the Wilcoxon test
classifier <- SWAP.KTSP.Train(matTraining, trainingGroup, krange=c(3, 5, 8:15))

### Show the classifier
classifier
```

```
### Apply the classifier to the TRAINING set using default decision rule
trainingPrediction <- SWAP.KTSP.Classify(matTraining, classifier)

### Resubstitution performance in the TRAINING set
### Define a "positive" test result if needed
table(trainingPrediction, trainingGroup)


### Use an alternative DecideFunction to classify each patient
### Here for instance at least two TSPs must agree
trainingPrediction <- SWAP.KTSP.Classify(matTraining, classifier,
 DecisionFunc = function(x) sum(x) > 5.5 )

### Contingency table for the TRAINING set
table(trainingPrediction, trainingGroup)


##################################################
### Testing on new data

### Load the example data for the TEST set
data(testingData)

### Show group variable for the TEST set
table(testingGroup)

### Apply the classifier to one sample of the TEST set using default decision rule
testPrediction <- SWAP.KTSP.Classify(matTesting[ , 1, drop=FALSE], classifier)

### Show prediction
testPrediction

### Apply the classifier to the complete the TEST set
### using decision rule defined above (agreement of two TSPs)
testPrediction <- SWAP.KTSP.Classify(matTesting,
  classifier, DecisionFunc = function(x) sum(x) > 5.5)

### Show prediction
head(testPrediction, n=10)

### Contingency table for the TEST set
table(testPrediction, testingGroup)
```

SWAP.KTSP.Statistics *Function computing TSP votes (comparisons) and combine their votes. The default is the kTSP statistics, sum of the votes.*

Description

SWAP.KTSP.Statistics computes the votes in favor of one of the classes or the other for each TSP. This function also computes the final, combined, consensus of all TSP votes based on a specific decision rules. The default is the kTSP statistics, sum of the votes.

Usage

```
SWAP.KTSP.Statistics(inputMat, classifier, CombineFunc)
```

Arguments

inputMat	is a numerical matrix containing the measurements (*e.g.*, gene expression data) to be used to compute the individual TSP votes and their consensus. like the matrix used for training classifier (in SWAP.KTSP.Train function), inputMatrix rows represent the features and the columns represent the samples.
classifier	the classifier obtained by invoking SWAP.KTSP.Train.
CombineFunc	is the function used to combine the votes (i.e., comparisons) of individual TSPs contained in the classifier. By default, the consensus is the count of the votes taking into account the order of the features in each TSP. Using this argument alternative aggregating functions can be also passed to SWAP.KTSP.Statistics as described below (see "details").

Details

For each TSP in the KTSP classifier, SWAP.KTSP.Statistics computes the vote in favor of one of classes or the other. This function also aggregates the individual TSP votes and computes a final consensus of all TSP votes based on specific combination rules. By default, this combination is achieved by counting the comparisons (votes) of TSPs as follows: If the first feature is larger than the second one, the TSP vote is positive, else the TSP vote is negative. Different combination rules can also be specified by defining an alternative combination function and by passing it to SWAP.KTSP.Statistics using the CombineFunc argument. A combination function takes as its input a logical vector x corresponding to the sample TSP comparisons (TRUE if the first feature in the pair is larger then the second, FALSE in the opposite case). The output of the CombineFunction is a single value summarizing the votes of all individual TSPs (see examples below). Note that CombineFunction function must operate on a logical vector as input and the outcome must be real value number.

Value

A list containing the following two components:

statistics	a named vector containing the aggregated summary statistics computed by CombineFunc. The names correspond to samples and are derived from colnames(inputMat).
comparisons	a logical matrix containing the individual TSP votes (TRUE if the first pair feature is larger then the second one, FALSE otherwise). The columns of this matrix correspond to TSP comparisons and are named accordingly using feature names derived from rownames(inputMat). The columns of this matrix correspond to the samples and are named accordingly using colnames(inputMat).

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>

References

See switchBox for the references.

See Also

SWAP.KTSP.Classify, SWAP.Filter.Wilcoxon, SWAP.CalculateSignedScore

Examples

```
####################################################
### Load gene expression data for the training set
data(trainingData)


### Show group variable for the TRAINING set
table(trainingGroup)


####################################################
### Train a classifier using default filtering function based on the Wilcoxon test
classifier <- SWAP.KTSP.Train(matTraining, trainingGroup,
      FilterFunc = NULL, krange=8)

### Show the TSP in the classifier
classifier$TSPs


####################################################
### Compute the TSP votes and combine them using various methods

### Here we will use the count of the signed TSP votes
ktspStatDefault <- SWAP.KTSP.Statistics(inputMat = matTraining,
    classifier = classifier)

### Here we will use the sum of the TSP votes
ktspStatSum <- SWAP.KTSP.Statistics(inputMat = matTraining,
    classifier = classifier, CombineFunc=sum)

### Here, for instance, we will apply a hard treshold equal to 2
ktspStatThreshold <- SWAP.KTSP.Statistics(inputMat = matTraining,
    classifier = classifier,  CombineFunc = function(x) sum(x) > 2 )

### Show components
names(ktspStatDefault)

### Show some of the votes
head(ktspStatDefault$comparisons[ , 1:2])
```

```
### Show default statistics
head(ktspStatDefault$statistics)

### Show statistics obtained using the sum
head(ktspStatSum$statistics)

### Show statistics obtained using the hard threshold
head(ktspStatThreshold)

### Make a heatmap showing the individual TSPs votes
colorForRows <- as.character(1+as.numeric(trainingGroup))
heatmap(1*ktspStatDefault$comparisons, scale="none",
    margins = c(10, 5), cexCol=0.5, cexRow=0.5,
    labRow=trainingGroup, RowSideColors=colorForRows)
```

SWAP.KTSP.Train *Function for training the K-TSP classifier.*

Description

SWAP.KTSP.Train trains a binary K-TSP classifier. The classifiers resulting from using this function can be passed to SWAP.KTSP.Classify for samples classification.

Usage

```
SWAP.KTSP.Train(inputMat, phenoGroup, krange = c(3, 5, 7:10),
  FilterFunc = SWAP.Filter.Wilcoxon, RestrictedPairs, ...)
```

Arguments

inputMat	is a numerical matrix containing the measurements (*e.g.*, gene expression data) to be used to build the K-TSP classifier. The columns represent samples and the rows represent the features (*e.g.*, genes). The number of columns must agree with the length of phenoGroup. Note that rownames(inputMat) will be used as the feature names (*e.g.*, gene symbols) in all subsequent analyses.
phenoGroup	is a factor with two levels containing the phenotype information used to train the K-TSP classifier. In order to identify the best TSP to be included in the classifier, the features contained in inputMat will be compared between the two groups defined by this factor. Levels from phenoGroup will be also used to reorder the features in each TSP such as the first feature is larger than the second one in the group corresponding to first level, and *vice-versa*.
krange	an integer (or a vector of integers) defining the candidate number of Top Scoring Pairs (TSPs) from which the algorithm chooses to build the final classifier. The algorithm uses the mechanism in Afsari et al (AOAS, 2014) to select the number of pairs and pair of features.

`FilterFunc` is a filtering function to reduce the starting number of features to be used to identify the Top Scoring Pairs (TSP). The default filter is differential expression test based on the Wilcoxon rank-sum test and alternative filtering functions can be passed too (see `SWAP.Filter.Wilcoxon` for details). The output of the function must be subset of `rownames(inputMat)`

`RestrictedPairs` is a character matrix with two columns containing the feature pairs to be considered for score calculations. Each row should contain a pair of feature names matching the rownames of `inputMat`. If RestrictedPairs is missing all available feature pairs will be considered.

`...` Additional argument passed to the filtering function `FilterFunc`.

Value

The KTSP classifier, in the form of a list, which contains the following components:

`name` The classifier name.

`TSPs` A k by 2 matrix, containing the feature names for each TSP. These names correspond to the `rownames(inputData)`. In this matrix each row corresponds to a specific TSP. For each TSP (*i.e.* row in the `TSPs` matrix) the order of the features is such that the first one is on average smaller than the second one in the phenotypic group defined by the first levels of the `phenoGroup` factor and *vice-versa*. The algorithm uses the mechanism in Afsari et al (2014) to select the number of pairs and pair of features.

`$score` scores TSP for the top k TSPs.

`$label` The class labels. These labels correspond to the `phenoGroup` factor lelves and will be used lable any new sample classified by the `SWAP.KTSP.Classify` function.

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>

References

See switchBox for the references.

See Also

`SWAP.KTSP.Classify`, `SWAP.Filter.Wilcoxon`, `SWAP.CalculateSignedScore`

Examples

```
################################################
### Load gene expression data for the training set
data(trainingData)
```

```
### Show group variable for the TRAINING set
table(trainingGroup)


#################################################
### Train a classifier using default filtering function based on the Wilcoxon test
classifier <- SWAP.KTSP.Train(matTraining, trainingGroup, krange=c(3, 5, 8:15))

### Show the classifier
classifier


#################################################
### Train another classifier from the top 4 best features
### according to the deafault  filtering function
classifier <- SWAP.KTSP.Train(matTraining, trainingGroup,
      FilterFunc=SWAP.Filter.Wilcoxon, featureNo=4)

### Show the classifier
classifier


#################################################
### To use all features "FilterFunc" must be set to NULL
classifier <- SWAP.KTSP.Train(matTraining, trainingGroup, FilterFunc=NULL)

### Show the classifier
classifier


#################################################
### Train a classifier using and alternative filtering function.
### For instance we can use the a  "t.test"  to selec the features
### with an absolute t-statistics larger than a specified quantile
topRttest <- function(situation, data, quant = 0.75) {
out <- apply(data, 1, function(x, ...) t.test(x ~ situation)$statistic )
names(out[ abs(out) > quantile(abs(out), quant) ])
}

### Show the top features selected
topRttest(trainingGroup, matTraining, quant=0.95)

### Train a classifier using the alternative filtering function
### and also define the maximum number of TSP using "krange"
classifier <- SWAP.KTSP.Train(matTraining, trainingGroup,
      FilterFunc = topRttest, quant = 0.75, krange=c(15:30) )

### Show the classifier
classifier


#################################################
### Training with restricted pairs
```

```
### Define a set of specific pairs to be used for classifier development
### For this example we will a random set of features
### In a real example these pairs should  be provided by the user.
set.seed(123)
somePairs <- matrix(sample(rownames(matTraining), 6^2, replace=FALSE), ncol=2)
head(somePairs, n=3)
dim(somePairs)

### Train a classifier using the restricted feature pairs and the default filtering
classifier <- SWAP.KTSP.Train(matTraining, trainingGroup,
      RestrictedPairs = somePairs, krange=3:16)

### Show the classifier
classifier
```

---

testingGroup *Testing set phenotypes*

Description

A factor with two levels describing the phenotypes for the testing data (Buyse et al cohort, (see the mammaPrintData package).

Usage

```
data(testingData)
```

Format

The matTesting factor contains phenotypic information for the 307 samples of the testing dataset.

Details

This phenotype factor corresponds to the breast cancer patients' cohort published by Buyse and colleagues in JNCI (2006). The gene expression matrix was obtained from the mammaPrintData package as described by Marchionni and colleagues in BMC Genomics (2013).

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>

References

See switchBox for the references.

See Also

trainingGroup

Examples

```
### Load gene expression data for the test set
data(testingData)

### Show the class of the ``testingGroup'' object
class(testingGroup)

### Show group variable
table(testingGroup)
```

`trainingGroup` *Training set phenotypes*

Description

A factor with two levels describing the phenotypes for the training data (Glas et al cohort, see the `mammaPrintData` package).

Usage

```
data(trainingData)
```

Format

The `trainingGroup` factor contains phenotypic information for the 78 samples of the training dataset.

Details

This phenotype factor corresponds to the breast cancer patients' cohort published by Glas and colleagues in BMC Genomics (2006). The information was obtained from the `mammaPrintData` package as described by Marchionni and colleagues in BMC Genomics (2013).

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>

References

See switchBox for the references.

See Also

`testingGroup`

Examples

```
### Load gene expression data for the training set
data(trainingData)

### Show the class of the ``trainingGroup'' object
class(trainingGroup)

### Show group variable
table(trainingGroup)
```

Index

∗Topic **KTSP, classification, prediction**
KTSP.Classifiy, 3
KTSP.Train, 4
SWAP.KTSP.Classifiy, 11
SWAP.KTSP.Statistics, 13
SWAP.KTSP.Train, 16
∗Topic **Pairwise score**
SWAP.CalculateSignedScore, 8
∗Topic **datasets**
matTesting, 6
matTraining, 7
testingGroup, 19
trainingGroup, 20
∗Topic **package**
switchBox-package, 2

KTSP.Classifiy, 3
KTSP.Classify, *4, 5*
KTSP.Classify *(*KTSP.Classifiy*)*, 3
KTSP.Train, *3*, 4

mammaPrintData, *6, 7*, *19, 20*
matTesting, 6, *7*
matTraining, *6*, 7

SWAP.CalculateSignedScore, 8, *11, 12*, *15*, *17*
SWAP.Filter.Wilcoxon, *9*, 10, *11, 12*, *15*, *17*
SWAP.KTSP.Classifiy, 11
SWAP.KTSP.Classify, *3*, *11*, *15–17*
SWAP.KTSP.Classify *(*SWAP.KTSP.Classifiy*)*, 11
SWAP.KTSP.Statistics, *9*, 13
SWAP.KTSP.Train, *4, 5*, *9*, *11, 12*, *14*, 16
switchBox, *3*, *5–7*, *9*, *11, 12*, *15*, *17*, *19, 20*
switchBox *(*switchBox-package*)*, 2
switchBox-package, 2

testingGroup, 19, *20*
trainingGroup, *19*, 20