Package 'switchBox'

March 7, 2025

Version 1.42.0

Date 2016-10-03

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>, Wikum Dinalankara <wdinala1@jhmi.edu>

Maintainer Bahman Afsari <bahman@jhu.edu>, Luigi Marchionni <marchion@jhu.edu>, Wikum Dinalankara <wdinala1@jhmi.edu>

Depends R (>= 2.13.1), pROC, gplots

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

git_url https://git.bioconductor.org/packages/switchBox

git_branch RELEASE_3_20

git_last_commit 174dd0f

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2025-03-06

Contents

`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.Calculate.BasicTSPScores

Function to calculate basic TSP scores.

Description

SWAP.Calculate.BasicTSPScores calculates basic TSP scores.

Usage

```
SWAP.Calculate.BasicTSPScores(phenoGroup, inputMat1,
  inputMat2 = NULL, classes = NULL, RestrictedPairs = NULL,
    handleTies = FALSE, verbose = FALSE, score_opts=list())
```

Arguments

phenoGroup	is a factor containing the training phenotypes with two levels.
inputMat1	is a numerical matrix containing the measurements (*e.g.*, gene expression data) for choosing the first item of a top scoring pair.
inputMat2	is a numerical matrix containing the measurements for choosing the second item of a top scoring pair. If NULL, inputMat1 will be used for this.
classes	is a character vector of length 2 providing the phenotype class labels (case followed by control). If NULL, the levels of phenoGroup will be taken as the labels.
RestrictedPairs	is a character matrix with two columns containing the feature pairs to be considered for score calculations.
handleTies	is a logical value indicating whether tie handling should be enabled or not. FALSE by default.
verbose	is a logical value indicating whether status messages will be printed or not throughout the function. FALSE by default.
score_opts	is a list of additional variables that will be passed on to the scoring function.

Value

The output is a list containing the following items:

labels	the levels (phenotypes) in phenoGroup.
score	is a vector containing the pair-wise scores.
tieVote	is a vector indicating the class the pair would vote for in the case of a tie.

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>, Wikum Dinalankara <wdinala1@jhmi.edu>

References

See switchBox for the references.

See Also

See SWAP.Calculate.SignedTSPScores

Examples

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

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

### Compute the scores
scores = SWAP.Calculate.BasicTSPScores(trainingGroup, matTraining[1:3, ])

# View the scores
scores$score
```

SWAP.Calculate.SignedTSPScores

Function to calculate signed TSP scores.

Description

SWAP.Calculate.SignedTSPScores calculates signed TSP scores. The input provided to this function should be already sanitized; to filter features and calculate pairwise scores, use SWAP.CalculateScores instead.

Usage

```
SWAP.Calculate.SignedTSPScores(phenoGroup, inputMat1,
  inputMat2 = NULL, classes = NULL, RestrictedPairs = NULL,
    handleTies = FALSE, verbose = FALSE, score_opts=list())
```

Arguments

phenoGroup	is a factor containing the training phenotypes with two levels.
inputMat1	is a numerical matrix containing the measurements (*e.g.*, gene expression data) for choosing the first item of a top scoring pair.
inputMat2	is a numerical matrix containing the measurements for choosing the second item of a top scoring pair. If NULL, inputMat1 will be used for this.
classes	is a character vector of length 2 providing the phenotype class labels (case followed by control). If NULL, the levels of phenoGroup will be taken as the labels.
RestrictedPairs	is a character matrix with two columns containing the feature pairs to be considered for score calculations.
handleTies	is a logical value indicating whether tie handling should be enabled or not. FALSE by default.
verbose	is a logical value indicating whether status messages will be printed or not throughout the function. FALSE by default.
score_opts	is a list of additional variables that will be passed on to the scoring function.

Value

The output is a list containing the following items:

labels	the levels (phenotypes) in phenoGroup.
score	is a vector containing the pair-wise scores.
tieVote	is a vector indicating the class the pair would vote for in the case of a tie.

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>, Wikum Dinalankara <wdinala1@jhmi.edu>

References

See switchBox for the references.

See Also

See SWAP.Calculate.BasicTSPScores

Examples

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

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

### Compute the scores
scores = SWAP.Calculate.SignedTSPScores(trainingGroup, matTraining[1:3, ])

# View the scores
scores$score
```

SWAP.CalculateScores *Function to calculate the pair-wise scores with any given score function.*

Description

SWAP.CalculateScores 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. The user can also pass a score-calculating function by either passing one of the scoring functions available in the package(i.e. SWAP.Calculate.SignedTSPScores and SWAP.Calculate.BasicTSPScores) or a custom function.

Usage

```
SWAP.CalculateScores(inputMat, phenoGroup, classes = NULL, FilterFunc = SWAP.Filter.Wilcoxon,
  RestrictedPairs = NULL, handleTies = FALSE, verbose = FALSE,
  score_fn = signedTSPScores, score_opts = list(), ...)
```

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.
classes	is a character vector of length 2 providing the phenotype class labels (case followed by control). If NULL, the levels of phenoGroup will be taken as the labels.
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.
handleTies	is a logical value indicating whether tie handling should be enabled or not. FALSE by default.
verbose	is a logical value indicating whether status messages will be printed or not throughout the function. FALSE by default.
score_fn	is a function for calculating TSP scores. By default, the signed TSP scores as calculated by SWAP.Calculate.SignedTSPScores will be used. The user can also provide SWAP.Calculate.BasicTSPScores to obtain basic TSP scores. The output of any custom function should correspond to the same strucure as the output from these two functions.
score_opts	is a list of additional variables that will be passed on to the scoring function as the score_opts argument.
...	Additional argument passed to the filtering function FilterFunc.

Value

The output is a list containing the following items:

labels	the levels (phenotypes) in phenoGroup.
score	is a vector containing the pair-wise scores.

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>, Wikum Dinalankara <wdinala1@jhmi.edu>

References

See switchBox for the references.

See Also

See SWAP.KTSP.Train, SWAP.Filter.Wilcoxon, SWAP.Calculate.BasicTSPScores, SWAP.Calculate.SignedTSPSc 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.CalculateScores(matTraining,  trainingGroup, FilterFunc=NULL)
```

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, handleTies = FALSE, verbose = FALSE, ...)
```

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.

handleTies	is a logical value indicating whether tie handling should be enabled or not. FALSE by default.
verbose	is a logical value indicating whether status messages will be printed or not throughout the function. FALSE by default.
...	Additional argument passed to the filtering function FilterFunc.

Value

The output is a list containing the following items:

labels	the levels (phenotypes) in phenoGroup.
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>, Wikum Dinalankara <wdinala1@jhmi.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.GetKTSP.PredictionStats

Function for computing various performance measures related to prediction.

Description

Given a list of predicted labels and true labels, provides accuracy, sensitivity, specificity, balanced accuracy (i.e. (sensitivity+specificity)/2), and AUC if decision values are given.

Usage

```
SWAP.GetKTSP.PredictionStats(predictions, truth, classes=NULL,
  decision_values=NULL)
```

Arguments

predictions	is a vector or factor of predicted classes.
truth	is a vector or factor of the true class labels.
classes	is a character vector of length 2 providing the phenotype class labels (case followed by control). If NULL, the levels of phenoGroup will be taken as the labels.
decision_values	is a vector providing the decision values (such as sum of votes from a k-TSP classifier). Will be used to compute AUC if provided.

Value

A vector providing accuracy, sensitivity, specificity, and balanced accuracy , and if decision_values is prodvided, area under the ROC curve (AUC).

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>, Wikum Dinalankara <wdinala1@jhmi.edu>

References

See switchBox for the references.

See Also

SWAP.KTSP.Classify

Examples

```
### Load gene expression data
data(trainingData)
data(testingData)

### train 1-TSP
classifier = SWAP.Train.1TSP(matTraining, trainingGroup)
predictions = SWAP.KTSP.Classify(matTesting, classifier)

### get performance results
SWAP.GetKTSP.PredictionStats(predictions, testingGroup)
```

SWAP.GetKTSP.Result *Function for prediction followed by computing various performance measures related to prediction.*

Description

Given a kTSP classifier and data matrix and class labels, calculates the predictions and vote sums and then applies SWAP.GetKTSP.PredictionStats.

Usage

```
SWAP.GetKTSP.Result(classifier, inputMat, Groups,
  classes=NULL, predictions=FALSE, decision_values=FALSE)
```

Arguments

classifier	a k-TSP classifier computed using SWAP.KTSP.Train or SWAP.Train.1TSP.
inputMat	is a matrix of data with rows being the features (such as gene names, if the matrix if gene expression data) and columns being the samples.
Groups	is a factor or a vector providing the phenotype class each sample belongs to. It should correspond to the order of samples given by the columns of inputMat.
classes	is a vetor of length 2 providing the two phenotype or class labels of Groups.
predictions	is a logical indicating whether to return the predictions or not.
decision_values	is a logical indicating whether to return the decision values or not.

Value

A list with items:

stats	A vector providing accuracy, sensitivity, specificity, balanced accuracy, and AUC.
roc	An ROC curve object produced by the pROC package.

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>, Wikum Dinalankara <wdinala1@jhmi.edu>

References

See switchBox for the references.

See Also

SWAP.GetKTSP.PredictionStats

Examples

```
### Load gene expression data
data(trainingData)
data(testingData)

require(pROC)

### train 1-TSP
classifier = SWAP.Train.1TSP(matTraining, trainingGroup)

### get performance results
SWAP.GetKTSP.Result(classifier, matTesting, testingGroup)$stats
```

SWAP.GetKTSP.TrainTestResults

Trains a kTSP on given training data and provides performance on testing data.

Description

Trains a kTSP on given training data and provides getkTSPResult output for both training and testing data.

Usage

```
SWAP.GetKTSP.TrainTestResults(trainMat, trainGroup, testMat,
  testGroup, classes=NULL, predictions=FALSE,
  decision_values=FALSE, ...)
```

Arguments

trainMat	is a matrix of data for training with rows being the features (such as gene names, if the matrix if gene expression data) and columns being the samples.
trainGroup	is a factor or a vector providing the phenotype class each training sample belongs to. It should correspond to the order of samples given by the columns of trainMat.
testMat	is a matrix of data for testing with rows being the features (such as gene names, if the matrix if gene expression data) and columns being the samples.
testGroup	is a factor or a vector providing the phenotype class each testing sample belongs to. It should correspond to the order of samples given by the columns of testMat.
classes	is a vetor of length 2 providing the two phenotype or class labels.
predictions	is a logical indicating whether to return the predictions or not.
decision_values	is a logical indicating whether to return the decision values or not.
...	any further arguments to be passed on for k-TSP training.

Value

A list with items:

classifier	The trained k-TSP classifier.
train	Training performance.
train	Testing performance.

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>, Wikum Dinalankara <wdinala1@jhmi.edu>

References

See switchBox for the references.

See Also

SWAP.GetKTSP.Result

Examples

```
### Load gene expression data
data(trainingData)
data(testingData)

require(pROC)

### perform training and testing
result = SWAP.GetKTSP.TrainTestResults(matTraining, trainingGroup,
  matTesting, testingGroup, featureNo=100)

### view results
result$train
```

```
result$test
```

SWAP.Kby.Measurement *K selection for a kTSP classifier.*

Description

SWAP.Kby.Measurement can be supplied to a kTSP classifier training function to select an optimal k by adding top-scoring pairs to maximize a given measurement such as accuracy or sensitivitiy over the training data.

Usage

```
SWAP.Kby.Measurement(inputMat, phenoGroup,
  scoreTable, classes, krange,
  k_opts=list(disjoint=TRUE, measurement="auc")
```

Arguments

inputMat	is a numerical matrix containing the measurements (*e.g.*, gene expression data) to be used to build the K-TSP classifier.
phenoGroup	is a factor with two levels containing the phenotype information used to train the K-TSP classifier.
scoreTable	a data frame output of SWAP.MakeTSPTable containing TSPs and the accuracy of individuals pairs over the training data.
classes	is a character vector of length 2 providing the phenotype class labels (case followed by control). If NULL, the levels of phenoGroup will be taken as the labels.
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.
k_opts	is a list of additional variables: disjoint is a logical indicating whether the selected pairs should be disjoint (i.e. features not repeated), and measurement is the given measurement to be maximized: it can be accuracy, sensitivity, specificity or auc.

Value

A vector of indices of length k indicating which pairs from scoreTable should be selected.

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>, Wikum Dinalankara <wdinala1@jhmi.edu>

References

See switchBox for the references.

See Also

SWAP.Kby.Ttest, SWAP.MakeTSPTable

SWAP.Kby.Ttest *K selection for a kTSP classifier.*

Description

SWAP.Kby.Ttest can be supplied to a kTSP classifier training function to select an optimal k via performing t-tests.

Usage

```
SWAP.Kby.Ttest(inputMat, phenoGroup,
  scoreTable, classes, krange,
  k_opts=list())
```

Arguments

inputMat	is a numerical matrix containing the measurements (*e.g.*, gene expression data) to be used to build the K-TSP classifier.
phenoGroup	is a factor with two levels containing the phenotype information used to train the K-TSP classifier.
scoreTable	a data frame output of SWAP.MakeTSPTable containing TSPs and the accuracy of individuals pairs over the training data.
classes	is a character vector of length 2 providing the phenotype class labels (case followed by control). If NULL, the levels of phenoGroup will be taken as the labels.
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.
k_opts	is not used and is left for conforming to the arguments of k_selection_fn.

Value

A vector of indices of length k indicating which pairs from scoreTable should be selected.

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>, Wikum Dinalankara <wdinala1@jhmi.edu>

References

See switchBox for the references.

See Also

SWAP.Kby.Measurement, SWAP.MakeTSPTable

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>, Wikum Dinalankara <wdinala1@jhmi.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.CV *Performs k-fold cross validation.*

Description

Partitions the data into k folds and applies SWAP.GetKTSP.TrainTestResults for each fold. Then it combines prediction votes by dividing the vote sums by the number of TSPs in each fold to produce an overall cross-validation result.

Usage

```
SWAP.KTSP.CV(inputMat, Groups, classes = NULL, k = 4,
  folds = NULL, randomize = TRUE, ...)
```

Arguments

inputMat	is a matrix of data with rows being the features (such as gene names, if the matrix if gene expression data) and columns being the samples.
Groups	is a factor or a vector providing the phenotype class each sample belongs to. It should correspond to the order of samples given by the columns of inputMat.
classes	is a vetor of length 2 providing the two phenotype or class labels.
k	an integer giving the number of folds to use.
folds	a list containing the samples to be used in each fold; if NULL, the data will be split into k folds maintaning the proportions between the classes.
randomize	is a logical indicating whether to randomize the sample order before diving into k folds.
...	any further arguments to be passed on for k-TSP training.

Value

A list with items:

folds	A list containing the sample indices used in each fold.
cv	A list containing the classifier, training performance and testing performance for each fold.
stats	Overall cross-validation performance.
roc	ROC curve object for overall cross-validation performance.

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>, Wikum Dinalankara <wdinala1@jhmi.edu>

References

See switchBox for the references.

See Also

SWAP.KTSP.LOO

Examples

```
### Load gene expression data
data(trainingData)
data(testingData)

require(pROC)

### perform leave one out cross-validation
result = SWAP.KTSP.CV(matTraining, trainingGroup, featureNo=100)

### print results
result$stats
```

SWAP.KTSP.LOO *Performs leave one out cross validation.*

Description

Performs leave one out cross validation; then it combines prediction votes by dividing the vote sums by the number of TSPs in each fold to produce an overall cross-validation result.

Usage

```
SWAP.KTSP.LOO(inputMat, Groups, classes = NULL, ...)
```

Arguments

inputMat	is a matrix of data with rows being the features (such as gene names, if the matrix if gene expression data) and columns being the samples.
Groups	is a factor or a vector providing the phenotype class each sample belongs to. It should correspond to the order of samples given by the columns of inputMat.
classes	is a vetor of length 2 providing the two phenotype or class labels.
...	any further arguments to be passed on for k-TSP training.

Value

A list with items:

loo	A list containing the classifier, training performance and testing performance for each fold.
decision_values	Decision values obtained for each left-out sample.

predictions	Predicted classes for each left-out sample.
stats	Overall peformance results.
roc	ROC curve object for overall performance.

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>, Wikum Dinalankara <wdinala1@jhmi.edu>

References

See switchBox for the references.

See Also

SWAP.KTSP.CV

Examples

```
### Load gene expression data
data(trainingData)
data(testingData)

require(pROC)

### perform leave one out cross-validation
result = SWAP.KTSP.LOO(matTraining, trainingGroup, featureNo=100)

### print results
result$stats
```

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>, Wikum Dinalankara <wdinala1@jhmi.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 *Deprecated 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. Note that this function is deprecated and we recommend SWAP.Train.KTSP for training k-TSP classifiers.

Usage

```
SWAP.KTSP.Train(inputMat, phenoGroup, krange = 2:10,
  FilterFunc = SWAP.Filter.Wilcoxon, RestrictedPairs,
  handleTies = FALSE, verbose = FALSE, ...)
```

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. Default is the range from 2 to 10.
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.
handleTies	is a logical value indicating whether tie handling should be enabled or not. FALSE by default.
verbose	is a logical value indicating whether status messages will be printed or not throughout the function. FALSE by default.
...	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>, Wikum Dinalankara <wdinala1@jhmi.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
```

SWAP.MakeTSPTable *Make a table of TSPs in order of TSP score.*

Description

Given the output from SWAP.CalculateScores and a number maxk, makes a table of the top maxk pairs. The output of this function can be provided to a k-selection function such as SWAP.Kby.Ttest or SWAP.Kby.Measurement to test out different k-selection methods.

Usage

```
SWAP.MakeTSPTable(Scores, maxk, disjoint = TRUE)
```

Arguments

Scores	is the output of a scoring function such as SWAP.CalculateScores containing a vector of TSP scores.
maxk	is an integer: the number of pairs to select.
disjoint	a logical indicating whether only disjoint pairs should be selected or not.

Value

A data frame of maxk pairs, their score and tieVote.

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>, Wikum Dinalankara <wdinala1@jhmi.edu>

References

See switchBox for the references.

See Also

SWAP.Kby.Ttest, SWAP.Kby.Measurement

Examples

```
### load gene expression data
data(trainingData)

### calculate scores
scores = SWAP.CalculateScores(matTraining, trainingGroup, featureNo=5)

### make top 5 pair table
SWAP.MakeTSPTable(scores, 5, FALSE)
```

SWAP.PlotKTSP.GenePairBoxplot

Plots a feature pair as boxplots.

Description

Plots two genes or features as a pair of boxplots; optionally, individual samples can be plotted on top of the boxplots as points; for this points can be colored by either gene, or class, or whether first gene < second gene is TRUE or FALSE for each sample.

Usage

```
SWAP.PlotKTSP.GenePairBoxplot(genes, inputMat, Groups=NULL,
  classes=NULL, points=FALSE, point_coloring="byGene",
  colors=c(), point_colors=c(), ...)
```

Arguments

genes	is a vector of length two providing the pair (from the rownames of inputMat) of features to be plotted.
inputMat	is a matrix of data with rows being the features (such as gene names, if the matrix if gene expression data) and columns being the samples.
Groups	is a factor or a vector providing the phenotype class each sample belongs to. It should correspond to the order of samples given by the columns of inputMat.
classes	is a vetor of length 2 providing the two phenotype or class labels of Groups.
points	is a logical value indicating whether to overlay the boxplot with points for individual samples or not.
point_coloring	can be either 'byGene' or 'byClass' indicating whether to color the points by gene/feature or by phenotype. A third option is 'byDirection' indicating to color the points by whether the first gene is less than the second gene.
colors	is a character vector indicating the color to be used for each boxplot.
point_colors	is a character vector indicating the color to be used for the points.
...	any further arguments are supplied to the boxplot function.

Value

Produces a pair of boxplots indicating the distribution of the measured values for the pair of features/genes.

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>, Wikum Dinalankara <wdinala1@jhmi.edu>

References

See switchBox for the references.

See Also

SWAP.PlotKTSP.GenePairClassesBoxplot

Examples

```
### Load gene expression data
data(trainingData)

### train 1-TSP
classifier = SWAP.Train.1TSP(matTraining, trainingGroup)

### plot top pair
SWAP.PlotKTSP.GenePairBoxplot(classifier$TSPs, matTraining,
  points=TRUE, point_coloring="byGene")
```

SWAP.PlotKTSP.GenePairClassesBoxplot

Plots a feature pair as seperated by class as boxplots.

Description

Plots two genes or features, each as a pair of boxplots seperated to two classes or phenotypes.

Usage

```
SWAP.PlotKTSP.GenePairClassesBoxplot(genes, inputMat, Groups,
  classes=NULL, points=FALSE, ordering="byGene",
  colors=c(), point_colors=c(), point_directions=FALSE, ...)
```

Arguments

genes	is a vector of length two providing the pair (from the rownames of inputMat) of features to be plotted.
inputMat	is a matrix of data with rows being the features (such as gene names, if the matrix if gene expression data) and columns being the samples.
Groups	is a factor or a vector providing the phenotype class each sample belongs to. It should correspond to the order of samples given by the columns of inputMat.
classes	is a vetor of length 2 providing the two phenotype or class labels of Groups.
points	is a logical value indicating whether to overlay the boxplot with points for individual samples or not.
ordering	can be either 'byGene' or 'byClass' respectively indicating whether to plot two adjacent boxplots for each class/phenotype or two adjacent boxplots for each gene/features.
colors	is a character vector indicating the color to be used for each class or gene boxplots.
point_colors	is a character vector indicating the color to be used for the points.
point_directions	is a logical indicating whether to color the points by whether the first gene is less than the second gene.
...	any further arguments are supplied to the boxplot function.

Value

Produces a pair of boxplots indicating the distribution of the measured values for the pair of features/genes.

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>, Wikum Dinalankara <wdinala1@jhmi.edu>

References

See switchBox for the references.

See Also

SWAP.PlotKTSP.GenePairBoxplot

Examples

```
### Load gene expression data
data(trainingData)

### train 1-TSP
classifier = SWAP.Train.1TSP(matTraining, trainingGroup)

### plot top pair
SWAP.PlotKTSP.GenePairClassesBoxplot(classifier$TSPs, matTraining,
  trainingGroup, levels(trainingGroup),
  points=TRUE, ordering="byGene")
```

SWAP.PlotKTSP.GenePairScatter

Make a scatter plot of two features.

Description

Makes a scatter plot of a pair of features/genes.

Usage

```
SWAP.PlotKTSP.GenePairScatter(inputMat, Groups,
  classes, genes, colors=c(), legends=c(), ...)
```

Arguments

inputMat	is a matrix of data with rows being the features (such as gene names, if the matrix if gene expression data) and columns being the samples.
Groups	is a factor or a vector providing the phenotype class each sample belongs to. It should correspond to the order of samples given by the columns of inputMat.
classes	is a vetor of length 2 providing the two phenotype or class labels of Groups.

genes	is a vector of length one or more providing the names (from the rownames of inputMat) of the features to be plotted.
colors	is a character vector indicating the color to be used for each phenotype.
legends	is a character vector providing any additional information to be appended to the phenotype label in the legend.

Value

Produces a scatter plot containing points for each sample colored by the phenotype, with two axes being the measurements for the given two features.

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>, Wikum Dinalankara <wdinala1@jhmi.edu>

References

See switchBox for the references.

See Also

SWAP.PlotKTSP.Genes

Examples

```
### Load gene expression data
data(trainingData)

### train 1-TSP
classifier = SWAP.Train.1TSP(matTraining, trainingGroup)

### plot top pair
SWAP.PlotKTSP.GenePairScatter(matTraining, trainingGroup, levels(trainingGroup), classifier$TSPs)
```

SWAP.PlotKTSP.Genes *Plot features seperated by phenotype*

Description

Makes line plots of one or more features seperated by phenotype.

Usage

```
SWAP.PlotKTSP.Genes(inputMat, Groups, classes, genes,
  colors=c(), legends=c(), ...)
```

Arguments

inputMat	is a matrix of data with rows being the features (such as gene names, if the matrix if gene expression data) and columns being the samples.
Groups	is a factor or a vector providing the phenotype class each sample belongs to. It should correspond to the order of samples given by the columns of inputMat.
classes	is a vetor of length 2 providing the two phenotype or class labels of Groups.
genes	is a vector of length one or more providing the names (from the rownames of inputMat) of the features to be plotted.

Value

Produces a plot containing a line for each feature plotted, the x-axis being the ordering of samples and the y-axis being the measured value (such as gene expression).

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>, Wikum Dinalankara <wdinala1@jhmi.edu>

References

See switchBox for the references.

See Also

SWAP.PlotKTSP.GenePairScatter

Examples

```
### Load gene expression data
data(trainingData)

### train 1-TSP
classifier = SWAP.Train.1TSP(matTraining, trainingGroup)

### plot top pair
SWAP.PlotKTSP.Genes(matTraining, trainingGroup, levels(trainingGroup), classifier$TSPs)
```

SWAP.PlotKTSP.TrainTestROC

Plots an ROC curve for training and testing results.

Description

Given the output from SWAP.GetKTSP.TrainTestResults(), plots the training and testing ROC curves.

Usage

```
SWAP.PlotKTSP.TrainTestROC(result, colors=c(), legends=c(), ...)
```

Arguments

result	is either the output from SWAP.GetkTSPTrainTestResults, or if manually prepared, a list with items trainroc and testroc items, where each is an ROC object produced by the pROC library.
colors	is a character vector indicating the color to be used for each curve.
legends	is a character vector providing any additional information to be appended to each curve label in the legend.

Value

Produces a plot with two ROC curves corresponding to training results and testing/validation results.

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>, Wikum Dinalankara <wdinala1@jhmi.edu>

References

See switchBox for the references.

See Also

SWAP.GetkTSPTrainTestResults

Examples

```
### Load gene expression data
data(trainingData)
data(testingData)

require(pROC)

### perform training and testing
result = SWAP.GetKTSP.TrainTestResults(matTraining, trainingGroup,
  matTesting, testingGroup, featureNo=100)

### plot ROC curves
SWAP.PlotKTSP.TrainTestROC(result)
```

SWAP.PlotKTSP.Votes *Plots a heatmap of k-TSP votes.*

Description

Given a k-TSP classifer and a matrix of data, plots a heatmap of the votes of the pairs computed on the given data.

Usage

```
SWAP.PlotKTSP.Votes(classifier, inputMat,
  Groups=NULL, CombineFunc, ...)
```

Arguments

classifier	is a k-TSP classifier produced by SWAP.KTSP.Train.
inputMat	is a matrix of data with rows being the features (such as gene names, if the matrix if gene expression data) and columns being the samples.
Groups	is a factor or a vector providing the phenotype class each sample belongs to. It should correspond to the order of samples given by the columns of inputMat. These phenotype labels will be added to the x-axis of the heatmap.
CombineFunc	is a function corresponding to the CombineFunc argument of the SWAP.KTSP.Classify function.

Value

Produces a heatmap where the color indicates a vote of 1 or 0 for a given sample by a top scoring pair.

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>, Wikum Dinalankara <wdinala1@jhmi.edu>

References

See switchBox for the references.

See Also

SWAP.KTSP.Train

SWAP.Train.1TSP *Function for training the 1-TSP classifier.*

Description

SWAP.Train.1TSP trains a binary TSP classifier with a single top scoring pair. The classifiers resulting from using this function can be passed to SWAP.KTSP.Classify for samples classification.

Usage

```
SWAP.Train.1TSP(inputMat, phenoGroup, classes = NULL,
  FilterFunc = SWAP.Filter.Wilcoxon, RestrictedPairs = NULL,
  handleTies = FALSE, disjoint = TRUE,
  score_fn = signedTSPScores, score_opts = NULL,
  verbose = FALSE, ...)
```

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*.
classes	is a character vector of length 2 providing the phenotype class labels (case followed by control). If NULL, the levels of phenoGroup will be taken as the labels.
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.
handleTies	is a logical value indicating whether tie handling should be enabled or not. FALSE by default.
disjoint	is a logical value indicating whether only disjoint pairs should be considered in the final set of selected pairs; i.e. all features occur only once among the set of TSPs.
score_fn	is a function for calculating TSP scores. By default, the signed TSP scores as calculated by SWAP.Calculate.SignedTSPScores will be used. The user can also provide SWAP.Calculate.BasicTSPScores to obtain basic TSP scores. The output of any custom function should correspond to the same strucure as the output from these two functions.
score_opts	is a list of additional variables that will be passed on to the scoring function as the score_opts argument.
verbose	is a logical value indicating whether status messages will be printed or not throughout the function. FALSE by default.
...	Additional argument passed to the filtering function FilterFunc.

Value

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

name	The classifier name.
TSPs	A 1 by 2 matrix, containing the feature names for the selected TSP. These names correspond to the rownames(inputData).
score	scores TSP for the top TSP.

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.
tieVote	indicates which class the pair would vote for in case of a tie.

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>, Wikum Dinalankara <wdinala1@jhmi.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.Train.1TSP(matTraining, trainingGroup)

### Show the classifier
classifier
```

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

Description

SWAP.Train.KTSP 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.Train.KTSP(inputMat, phenoGroup, classes = NULL, krange = 2:10,
  FilterFunc = SWAP.Filter.Wilcoxon, RestrictedPairs = NULL,
  handleTies = FALSE, disjoint = TRUE,
  k_selection_fn = KbyTtest, k_opts = list(), score_fn = signedTSPScores,
  score_opts = NULL, verbose = FALSE, ...)
```

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*.
classes	is a character vector of length 2 providing the phenotype class labels (case followed by control). If NULL, the levels of phenoGroup will be taken as the labels.
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. Default is the range from 2 to 10.
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.
handleTies	is a logical value indicating whether tie handling should be enabled or not. FALSE by default.
disjoint	is a logical value indicating whether only disjoint pairs should be considered in the final set of selected pairs; i.e. all features occur only once among the set of TSPs.
k_selection_fn	is a function for selecting the optimal k once the TSP scores have been calculated for all the candidate pairs. This can be either SWAP.Kby.Measurement or SWAP.Kby.Ttest(default), or a user defined function.
k_opts	a list of additional arguments to be passed on to a custom k selection function.
score_fn	is a function for calculating TSP scores. By default, the signed TSP scores as calculated by SWAP.Calculate.SignedTSPScores will be used. The user can also provide SWAP.Calculate.BasicTSPScores to obtain basic TSP scores. The output of any custom function should correspond to the same strucure as the output from these two functions.
score_opts	is a list of additional variables that will be passed on to the scoring function as the score_opts argument.
verbose	is a logical value indicating whether status messages will be printed or not throughout the function. FALSE by default.
...	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.
`tieVote`	indicates which class the pair would vote for in case of a tie.

Author(s)

Bahman Afsari <bahman.afsari@gmail.com>, Luigi Marchionni <marchion@jhu.edu>, Wikum Dinalankara <wdinala1@jhmi.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.Train.KTSP(matTraining, trainingGroup)

### Show the classifier
classifier


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

```
### 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