Package 'TSEAL'

July 2, 2024

Type Package

Title Time Series Analysis Library

Version 0.1.3

Description The library allows to perform a multivariate time series classification based on the use of Discrete Wavelet Transform for feature extraction, a step wise discriminant to select the most relevant features and finally, the use of a linear or quadratic discriminant for classification. Note that all these steps can be done separately which allows to implement new steps. Velasco, I., Sipols, A., de Blas, C. S., Pastor, L., & Bayona, S. (2023) <doi:10.1186/S12938-023-01079-X>. Percival, D. B., & Walden, A. T. (2000,ISBN:0521640687). Maharaj, E. A., & Alonso, A. M. (2014) <doi:10.1016/j.csda.2013.09.006>.

License Artistic-2.0

URL https://github.com/vg-lab/TSEAL

BugReports https://github.com/vg-lab/TSEAL/issues

Depends R (>= 4.3.0)

Imports bigmemory, caret, checkmate, magrittr, MASS, methods, parallel, parallelly, pryr, statcomp, stats, synchronicity, utils, waveslim, wdm

Suggests spelling, testthat (>= 3.0.0)

Config/testthat/edition 3

Config/testthat/parallel faslse

Encoding UTF-8

Language en-US

RoxygenNote 7.3.1

NeedsCompilation no

Author Iván Velasco [aut, cre, cph] (<https://orcid.org/0000-0002-2314-5670>)

Maintainer Iván Velasco <ivan.velasco@urjc.es>

Repository CRAN

Date/Publication 2024-07-02 12:10:07 UTC

Contents

`availableFeatures` *availableFeatures*

Description

Print the available features for the `MultiWaveAnalysis` and `StepDiscrim`

Usage

```
availableFeatures()
```

Value

A `data.frame` containing the name of the characteristics and their abbreviations for use in the code. For example, to use variances and correlations, the vector c("Var", "Cor") will be used.

See Also

- `MultiWaveAnalysis`
- `StepDiscrim`
- `StepDiscrimV`

Examples

```
availableFeatures()
```

`availableFilters` *availableFilters*

Description

Print the available filters for the wave analysis

Usage

```
availableFilters()
```

Value

A `data.frame` containing all supported filters

See Also

- `MultiWaveAnalysis`

Examples

```
availableFilters()
```

`chooseLevel` *Select the DWT level of decomposition based on wavelet filter, data series length and a user choice*

Description

Select the DWT level of decomposition based on wavelet filter, data series length and a user choice

Usage

```
chooseLevel(choice, filter, N)
```

Arguments

choice	Valid values: • "Conservative" : $J < log_2(N/(L-1)+1)$ • "Max" : $J \leq log_2(N)$ • "Supermax" : $J \leq log_2(1.5 * N)$
filter	Wavelet transform filter name. To see the available filters use the function availableFilters
N	Number of observations. Must be a positive integer

Value

Number of level of decomposition based in selection criteria

References

Percival, D. B. and A. T. Walden (2000) Wavelet Methods for Time Series Analysis. Cambridge: Cambridge University Press.

Examples

```
lev <- chooseLevel("conservative", "haar", 8)
```

---

classify *Classifies observations based on a pretrained model.*

Description

This function allows to classify observations based on a pretrained model that could have been obtained in several ways (such as using the train model function). T

Usage

```
classify(data, ...)
```

Arguments

data	The data to be classified. This data can be either the raw data , or a MultiWaveAnalysis object generated earlier.
...	Additional arguments

Value

A factor with predicted class of each observation

See Also

- trainModel
- classify.array
- classify.MultiWaveAnalysis

`classify.array` *Classifies observations based on a pretrained model.*

Description

This function allows to classify observations based on a pretrained model that could have been obtained in several ways (such as using the train model function).

Usage

```
## S3 method for class 'array'
classify(data, model, ...)
```

Arguments

`data`	Sample from the population (dim x length x cases)
`model`	pretrained discriminant model (lda or qda)
`...`	Additional arguments

Value

A factor with predicted class of each observation

See Also

- trainModel

Examples

```
load(system.file("extdata/ECGExample.rda",package = "TSEAL"))
# We simulate that the second series has been obtained after
Series1 <- ECGExample[, , 1:9]
Series2 <- ECGExample[, , 10, drop = FALSE]

# Training a discriminant model
MWA <- MultiWaveAnalysis(Series1, "haar", features = c("var"))
MWADiscrim <- StepDiscrim(MWA, c(rep(1, 5), rep(2, 4)), maxvars = 5,
                          features = c("var"))
model <- trainModel(MWADiscrim, c(rep(1, 5), rep(2, 4)), "linear")

# Using the discriminant trained on new data
prediction <- classify(Series2, model)
```

`classify.MultiWaveAnalysis` *Classifies observations based on a pretrained model.*

Description

This function allows to classify observations based on a pretrained model that could have been obtained in several ways (such as using the train model function).

Usage

```
## S3 method for class 'MultiWaveAnalysis'
classify(data, model, ...)
```

Arguments

`data`	Data to be classified by the model. Remember that it must be an object of type MultiWaveAnalysis. Note that it should have the same variables selected as those used to generate the model.
`model`	pretrained discriminant model (lda or qda)
`...`	Additional arguments

Value

A factor with predicted class of each observation

See Also

- `trainModel`

Examples

```
load(system.file("extdata/ECGExample.rda",package = "TSEAL"))
# We simulate that the second series has been obtained after
Series1 <- ECGExample[, , 1:9]
Series2 <- ECGExample[, , 10, drop = FALSE]

# Training a discriminant model
MWA <- MultiWaveAnalysis(Series1, "haar", features = c("var"))
MWADiscrim <- StepDiscrim(MWA, c(rep(1, 5), rep(2, 4)), maxvars = 5,
                          features = c("var"))
model <- trainModel(MWADiscrim, c(rep(1, 5), rep(2, 4)), "linear")

# Using the discriminant trained on new data
MWA2 <- MultiWaveAnalysis(Series2, "haar", features = c("var"))
MWA2Discrim <- SameDiscrim(MWA2, MWADiscrim)
prediction <- classify(MWA2Discrim, model)
```

`extractSubset` *Extract observations from a MultiWaveAnalysis*

Description

This function permits to extract certain observations from a MultiWaveAnalysis

Usage

```
extractSubset(MWA, indices)
```

Arguments

`MWA`	MultiWaveAnalysis from which the desired observations will be extracted
`indices`	Indices that will indicate which observations will be extracted

Value

A list with two elements:

- MWA: The MultiWaveAnalysis provided minus the extracted observations.
- MWAExtracted: A new MultiWaveAnalysis with the extracted observations

Examples

```
load(system.file("extdata/ECGExample.rda",package = "TSEAL"))
MWA <- MultiWaveAnalysis(ECGExample, "haar", features = "Var")
aux <- extractSubset(MWA, c(1, 2, 3))
MWATrain <- aux[[1]]
MWATest <- aux[[2]]
```

`generateStepDiscrim` *Generate StepDiscrim from raw data*

Description

This function allows to obtain in a single step the complete MultiWaveAnalysis and the selection of the most discriminating variables of the MultiWaveAnalysis.

Usage

```
generateStepDiscrim(
  series,
  labels,
  f,
  maxvars,
  VStep,
  lev = 0,
  features = c("Var", "Cor", "IQR", "PE", "DM"),
  nCores = 0
)
```

Arguments

series	Sample from the population (dim x length x cases)
labels	Labeled vector that classify the observations
f	Selected filter for the MODWT (to see the available filters use the function availableFilters
maxvars	Maximum number of variables included by the StepDiscrim algorithm (Note that if you defined this, can not define VStep). Must be a positive integer
VStep	Minimum value of V above which all other variables are considered irrelevant and therefore will not be included. (Note that if you defined this, can not defined maxvars).Must be a positive number. For more information see StepDiscrim documentation.
lev	Determines the number of decomposition levels for MODWT (by default the optimum is calculated). Must be a positive integer, where 0 corresponds to the default behavior.
features	A list of characteristics that will be used for the classification process. To see the available features see availableFeatures
nCores	Determines the number of processes that will be used in the function, by default it uses all but one of the system cores. Must be a positive integer, where 0 corresponds to the default behavior

Value

A MultiWaveAnalysis with the most discriminant variables based on the features indicated.

See Also

- MultiWaveAnalysis
- StepDiscrim
- StepDiscrimV

Examples

```
load(system.file("extdata/ECGExample.rda",package = "TSEAL"))
# The dataset has the first 5 elements of class 1
# and the last 5 of class 2.
labels <- c(rep(1, 5), rep(2, 5))
MWADiscrim <- generateStepDiscrim(ECGExample, labels, "haar",
  features = c("Var"), maxvars = 5
)
# or using the VStep option
MWADiscrim <- generateStepDiscrim(ECGExample, labels, "haar",
 features = c("Var", "Cor"), VStep = 0.7
)
```

KFCV *K-Fold Cross Validation (KFCV)*

Description

This function performs the K-Fold Cross Validation (KFCV) process with different types of input parameters.

Usage

```
KFCV(data, ...)
```

Arguments

`data`	Starting data to generate the validation. It can be either the raw data, or a previously generated MultiWaveAnalysis object.
`...`	Additional arguments

Value

Not return value, used as generic function

See Also

- `KFCV.array`
- `KFCV.MultiWaveAnalysis`

KFCV.array *Generates and validates a discriminant model generated directly from the data.*

Description

It generates and validates a discriminant model starting from the data. First , a MultiWaveAnalysis object is obtained according to the selected characteristics ,filter and levels. Then, the most important features are selected using a stepwise discriminant that allows to select a maximum number of variables (maxvars) or a minimum enhancement step (VStep). Finally, the model is trained using the subset of features and validated using K-Fold Cross Validation (KFCV).

Usage

```
## S3 method for class 'array'
KFCV(
  data,
  labels,
  f,
  method,
  maxvars,
  VStep,
  k = 5L,
  lev = 0L,
  features = c("Var", "Cor", "IQR", "PE", "DM"),
  returnClassification = FALSE,
  nCores = 0,
  ...
)
```

Arguments

data	Sample from the population (dim x length x cases)
labels	Labeled vector that classify the observations
f	Selected filter for the MODWT (to see the available filters use the function availableFilters
method	Selected method for the discriminant. Valid values "linear" "quadratic"
maxvars	Maximum number of variables included by the StepDiscrim algorithm (Note that if you defined this, can not define VStep). Must be a positive integer greater than 0.
VStep	Minimum value of V above which all other variables are considered irrelevant and therefore will not be included. (Note that if you defined this, can not defined maxvars). Must be a positive number and greater than 0. For more information see StepDiscrim documentation
k	The number of folds in KFCV. Must be a positive integer lower or equal than the number of observations

lev — Determines the number of decomposition levels for MODWT (by default the optimum is calculated using the "conservative" strategy). Must be a positive integer (including 0 to auto-select the level)

features — A list of characteristics that will be used for the classification process. To see the available features see availableFeatures

returnClassification — Allows to select if the raw result classification is returned.

nCores — Determines the number of processes that will be used in the function, by default it uses all but one of the system cores. Must be a positive integer, where 0 corresponds to the default behavior.

... — Additional arguments

Value

- if returnClassification is false return a object of class confusionMatrix
- if returnClassification is true, it returns a list containing an object of the confusionMatrix class and a vector with the classification result.

See Also

- LOOCV
- LOOCV.MultiWaveAnalysis
- availableFilters
- availableFeatures

Examples

```
load(system.file("extdata/ECGExample.rda",package = "TSEAL"))
labels <- c(rep(1, 5), rep(2, 5))
CM <- KFCV(ECGExample, labels, "haar", "linear",
  maxvars = 5,
  features = c("Var"), returnClassification = FALSE
)
# or with VStep
CMV <- KFCV(ECGExample, labels, "haar", "linear",
 k = 5,
 VStep = 5,
 features = c("Var"), returnClassification = FALSE
)
```

KFCV.MultiWaveAnalysis *KFCV*

Description

Performs k-fold cross-validation where groups are chosen randomly. In case the value k is not divisor of the number of observations the last group will have nobs mod k observations.

Usage

```
## S3 method for class 'MultiWaveAnalysis'
KFCV(data, labels, method, k = 5L, returnClassification = FALSE, ...)
```

Arguments

data	MultiWaveAnalysis (MWA) object obtained with MultiWaveAnalysis and preferably obtained a subset of its characteristics (StepDiscrim,StepDiscrimV)
labels	labeled vector that classify the observations.
method	Selected method for discrimination. Valid options "linear" "quadratic"
k	the number of folds in KFCV. Must be a positive integer and lower or equal than the number of observations
returnClassification	Allows to select if the raw result classification is returned.
...	Additional arguments

Value

- if returnClassification is false return a object of class confusionMatrix
- if returnClassification is true, it returns a list containing an object of the confusionMatrix class and a vector with the classification result.

Examples

```
load(system.file("extdata/ECGExample.rda",package = "TSEAL"))
MWA <- MultiWaveAnalysis(ECGExample, "haar", features = c("var"))
MWADiscrim <- StepDiscrim(MWA, c(rep(1, 5), rep(2, 5)), 5,
              features = c("var"))
CM <- KFCV(MWADiscrim, c(rep(1, 5), rep(2, 5)), "linear", 5,
  returnClassification = FALSE
)
```

LOOCV *Leave-One-Out Cross Validation*

Description

This function performs the Leave-One-Out Cross Validation (LOOCV) process with different types of input parameters.

Usage

```
LOOCV(data, ...)
```

Arguments

data	Starting data to generate the validation. It can be either the raw data, or a previously generated MultiWaveAnalysis object.
...	Additional arguments

Value

Not return value, used as generic function

See Also

- LOOCV.array
- LOOCV.MultiWaveAnalysis

LOOCV.array *Generates and validates a discriminant model generated directly from the data.*

Description

It generates and validates a discriminant model starting from the data. First, a MultiWaveAnalysis object is obtained according to the selected characteristics, filter and levels. Then, the most important features are selected using a stepwise discriminant that allows to select a maximum number of variables (maxvars) or a minimum enhancement step (VStep). Finally, the model is trained using the subset of features and validated using Leave-One-Out Cross Validation (LOOCV).

Usage

```
## S3 method for class 'array'
LOOCV(
  data,
  labels,
  f,
  method,
  maxvars,
  VStep,
  lev = 0,
  features = c("Var", "Cor", "IQR", "PE", "DM"),
  returnClassification = FALSE,
  nCores = 0,
  ...
)
```

Arguments

data	Sample from the population (dim x length x cases)
labels	Labeled vector that classify the observations
f	Selected filter for the MODWT (to see the available filters use the function availableFilters
method	Selected method for the discriminant. Valid values "linear" "quadratic"
maxvars	Maximum number of variables included by the StepDiscrim algorithm (Note that if you defined this, can not define VStep). Must be a positive integer greater than 0.
VStep	Minimum value of V above which all other variables are considered irrelevant and therefore will not be included. (Note that if you defined this, can not defined maxvars). Must be a positive number and greater than 0. For more information see StepDiscrim documentation
lev	Determines the number of decomposition levels for MODWT (by default the optimum is calculated using the "conservative" strategy). Must be a positive integer (including 0 to auto-select the level)
features	A list of characteristics that will be used for the classification process. To see the available features see availableFeatures
returnClassification	Allows to select if the raw result classification is returned.
nCores	Determines the number of processes that will be used in the function, by default it uses all but one of the system cores. Must be a positive integer, where 0 corresponds to the default behavior.
...	Additional arguments

Value

- if returnClassification is false return a object of class confusionMatrix
- if returnClassification is true, it returns a list containing an object of the confusionMatrix class and a vector with the classification result.

See Also

- LOOCV
- LOOCV.MultiWaveAnalysis
- availableFilters
- availableFeatures

Examples

```
load(system.file("extdata/ECGExample.rda",package = "TSEAL"))
labels <- c(rep(1, 5), rep(2, 5))
CM <- LOOCV(ECGExample, labels, "haar", "linear",
  maxvars = 5,
  features = c("Var"), returnClassification = FALSE
)
# or with VStep
CMV <- LOOCV(ECGExample, labels, "haar", "linear",
 VStep = 5,
 features = c("Var", "Cor"), returnClassification = FALSE
)
```

LOOCV.MultiWaveAnalysis

LOOCV

Description

Performs a leave-one-cross-validation (LOOCV) method on a MultiWaveAnalysis object. It is advisable to have selected a subset of all features (StepDiscrim,StepDiscrimV)

Usage

```
## S3 method for class 'MultiWaveAnalysis'
LOOCV(data, labels, method, returnClassification = FALSE, ...)
```

Arguments

data	MultiWaveAnalysis object obtained with MultiWaveAnalysis function and preferably obtained a subset of its characteristics (StepDiscrim, StepDiscrimV)
labels	Labeled vector that classify the observations.
method	Selected method for discrimination. Valid options "linear" "quadratic"
returnClassification	Allows to select if the raw result classification is returned.
...	Additional arguments

Value

- if returnClassification is false return a object of class confusionMatrix
- if returnClassification is true, it returns a list containing an object of the confusionMatrix class and a vector with the classification result.

See Also

- LOOCV
- LOOCV.array
- StepDiscrim
- StepDiscrimV

Examples

```
load(system.file("extdata/ECGExample.rda",package = "TSEAL"))
MWA <- MultiWaveAnalysis(ECGExample, "haar", features = c("var"))
MWADiscrim <- StepDiscrim(MWA, c(rep(1, 5), rep(2, 5)), 5,
                          features = c("var"))
CM <- LOOCV(MWADiscrim, c(rep(1, 5), rep(2, 5)), "linear")
```

MultiWaveAnalysis *Generate a MultiWave analysis*

Description

Generates a multivariate analysis by calculating a series of features from the result of applying MODWT to the input data.

Usage

```
MultiWaveAnalysis(
  series,
  f,
  lev = 0,
  features = c("Var", "Cor", "IQR", "PE", "DM"),
  nCores = 0
)
```

Arguments

series	Sample from the population (array of three dimensions [dim, length, cases]
f	Selected wavelet filter for the analysis. To see the available filters use the function availableFilters

lev	Wavelet decomposition level by default is selected using the "conservative" strategy. See chooseLevel function. Must be a positive integer (including 0 to auto-select the level)
features	It allows to select the characteristics to be calculated for the analysis. To see the available features use the function availableFeatures
nCores	Determines the number of processes that will be used in the function, by default it uses all but one of the system cores. Must be a positive integer, where 0 corresponds to the default behavior

Value

A multivariate analysis with the characteristics indicated in the parameter features. This is an object of class MultiWaveAnalysis with contains * Features: A list with the computed features * StepSelection: A selection with the most discriminant features StepDiscrim * Observations: Number of total observations * NLevels: Number of levels selected for the decomposition process * Filter: Filter used in the decomposition process

See Also

- availableFilters
- availableFeatures

Examples

```
load(system.file("extdata/ECGExample.rda",package = "TSEAL"))
MWA <- MultiWaveAnalysis(ECGExample,
  f = "haar", lev = 0,
  features = c("Var", "Cor"), nCores = 0
)
```

SameDiscrim *Allows to select the same variables for a given StepDiscrim*

Description

Allows to perform the same variable selection in a new MWA object starting from a MWA object with the variables already selected (it is advisable that the parameters of the MWA and of the selection are the same).

Usage

```
SameDiscrim(MWA, MWADiscrim)
```

Arguments

MWA	MultiWaveAnalysis object on which variables are to be selected.
MWADiscrim	MultiWaveAnalysis object on which certain variables have been previously selected, using StepDiscrim or StepDiscrimV

Value

An object of class MultiWaveAnalysis with the same variables selected as in the MWADiscrim object.

See Also

- StepDiscrim
- StepDiscrimV

Examples

```
load(system.file("extdata/ECGExample.rda",package = "TSEAL"))
# We simulate that the second series has been obtained after
Series1 <- ECGExample[, , 1:9]
Series2 <- ECGExample[, , 10, drop = FALSE]
MWA <- MultiWaveAnalysis(Series1, "haar", features = c("var"))
MWADiscrim <- StepDiscrim(MWA, c(rep(1, 5), rep(2, 4)), 5,
  features = c("var")
)

MWA2 <- MultiWaveAnalysis(Series2, "haar", features = c("var"))
MWA2Discrim <- SameDiscrim(MWA2, MWADiscrim)
# At this point MWA2Discrim has the same variables that MWADiscrim
# and can be used in a pretrained model with MWADiscrim
```

StepDiscrim *Select the most discriminating variables*

Description

Stepwise discriminant analysis to determine the best subset of variables. Introduces variables so as to maximize at each step the Lawley-Hotelling trace (=Rao's V). This measure is proportional to the mean Mahalanobis distance.

Usage

```
StepDiscrim(
  MWA,
  labels,
  maxvars,
  features = c("Var", "Cor", "IQR", "PE", "DM"),
  nCores = 0
)
```

Arguments

MWA	MultiWaveAnalysis object obtained with MultiWaveAnalysis function
labels	Labeled vector that classify the observations.
maxvars	The number of desired values. Must be a positive integer
features	A list of characteristics that will be used for the classification process. To see the available features see availableFeatures
nCores	Determines the number of processes that will be used in the function, by default it uses all but one of the system cores. Must be a positive integer, where 0 corresponds to the default behavior

Details

Based on StepDiscrim of R.E. Strauss

Value

A MultiWaveAnalysis object with the maxvars most discriminant variables. This object contains: * Features: A list with the initial computed features * StepSelection: The maxvars most discriminant variables * Observations: Number of total observations * NLevels: Number of levels selected for the decomposition process * filter: Filter used in the decomposition process

See Also

- MultiWaveAnalysis
- StepDiscrimV

Examples

```
load(system.file("extdata/ECGExample.rda",package = "TSEAL"))
MWA <- MultiWaveAnalysis(ECGExample, "haar", features = c("var"))
MWADiscrim <- StepDiscrim(
  MWA, c(rep(1, 5), rep(2, 5)), 5,
  c("Var")
)
```

`StepDiscrimV` *Select the most discriminating variables*

Description

Stepwise discriminant analysis to determine the best subset of variables. Introduces variables so as to maximize at each step the Lawley-Hotelling trace (=Rao's V). This measure is proportional to the mean Mahalanobis distance.The process ends when in one step the value of the Lawley-Hotelling trace is less than a given value.

Usage

```
StepDiscrimV(
  MWA,
  labels,
  VStep,
  features = c("Var", "Cor", "IQR", "PE", "DM"),
  nCores = 0
)
```

Arguments

`MWA`	MultiWaveAnalysis object obtained with MultiWaveAnalysis function
`labels`	Labeled vector that classify the observations.
`VStep`	Determine the minimum value of V to continue adding new variables. Ex if an determinate step the maximum V is 0.2 but VStep is 0.3 the algorithm end. Must be greater than 0.
`features`	A list of characteristics that will be used for the classification process. To see the available features see `availableFeatures`
`nCores`	Determines the number of processes that will be used in the function, by default it uses all but one of the system cores. Must be a positive integer, where 0 corresponds to the default behavior

Details

Based on StepDiscrim of R.E. Strauss

Value

A MultiWaveAnalysis object with the most discriminant variables. This Object contains: * Features: A list with the initial computed features * StepSelection: The most discriminant variables selected by this function * Observations: Number of total observations * NLevels: Number of levels selected for the decomposition process * filter: Filter used in the decomposition process

See Also

- MultiWaveAnalysis
- StepDiscrim

Examples

```
load(system.file("extdata/ECGExample.rda",package = "TSEAL"))
MWA <- MultiWaveAnalysis(ECGExample, "haar", features = c("var"))
MWADiscrim <- StepDiscrimV(
  MWA, c(rep(1, 5), rep(2, 5)), 0.1,
  c("Var")
)
```

---

testFilters *testFilters*

Description

This function performs a test with a series of filters defined by the user, for the maximum number of variables determined. This function can be used to compare the performance of different filters with a different number of variables to be considered and the differences between a linear and a quadratic discriminant.

Usage

```
testFilters(
  series,
  labels,
  maxvars,
  filters = c("haar", "d4", "d6", "d8", "la8"),
  features = c("Var", "Cor", "IQR", "PE", "DM"),
  lev = 0
)
```

Arguments

series	Samples from the population (dim x length x cases)
labels	Labeled vector that classify the observations.
maxvars	maximum number of variables included by the StepDiscrim algorithm. Must be grater than 0 and, in normal cases, lesser than 100
filters	Vector indicating the filters to be tested. To see the available filters use the function availableFilters
features	A list of characteristics that will be used for the classification process. To see the available features see availableFeatures
lev	Wavelet decomposition level, by default is selected using the "conservative" strategy. See chooseLevel function.

Value

A list that each element contains:

- CM: confusion matrix with a particular configuration using LOOCV
- Classification: a vector with the raw classification result. "1" if the observation belongs to the population 1 and "2" if belongs to the population 2.
- NVars: the total numbers of variables have been taken into account in the classification process
- Method: type of classifier used.
- Filter: filter used in the MultiWave analysis process
- Features: vector containing the features taken into account

See Also

- LOOCV
- MultiWaveAnalysis
- StepDiscrim
- availableFilters
- availableFeatures

Examples

```
load(system.file("extdata/ECGExample.rda",package = "TSEAL"))
# The dataset has the first 5 elements of class 1
# and the last 5 of class 2.
labels <- c(rep(1, 5), rep(2, 5))
result <- testFilters(ECGExample, labels, features=c("var","cor"),
          filters= c("haar","d4"), maxvars = 3)
```

---

testModel *Computes a classification from a pretrained discriminant*

Description

This function uses a pretrained linear discriminant to classify a set of test data. As output it returns a confusion matrix and optionally the raw classification result.

Usage

```
testModel(model, test, labels, returnClassification = FALSE, ...)
```

Arguments

model	Trained linear discriminant. see trainModel
test	MultiWaveAnalysis class object to be used as test set.
labels	Vector that determines the class to which each of the observations provided in the test set belongs.
returnClassification	Allows to select if the raw result classification is returned.
...	Additional arguments

Value

- if returnClassification is false return a object of class confusionMatrix
- if returnClassification is true, it returns a list containing an object of the confusionMatrix class and a vector with the classification result.

See Also

testModel

Examples

```
load(system.file("extdata/ECGExample.rda",package = "TSEAL"))
# The dataset has the first 5 elements of class 1
# and the last 5 of class 2.
labels <- c(rep(1, 5), rep(2, 5))
MWA <- generateStepDiscrim(ECGExample, labels, "haar", maxvars = 5, features = c("var"))
aux <- extractSubset(MWA, c(1, 2, 9, 10))
MWATest <- aux[[1]]
MWATrain <- aux[[2]]
ldaDiscriminant <- trainModel(MWATrain, labels[3:8], "linear")
CM <- testModel(ldaDiscriminant, MWATest, labels[c(1, 2, 9, 10)])
```

---

trainModel *Generate a Discriminant Model*

Description

This function allows training of a discriminant model using different inputs

Usage

```
trainModel(data, ...)
```

Arguments

data	Starting data to generate a discriminator (linear or quadratic). This starting data can be either the raw data, or a MultiWaveAnalysis object generated earlier.
...	Additional arguments

Value

A trained discriminant model

See Also

- trainModel.array
- trainModel.MultiWaveAnalysis

---

trainModel.array *Generates a discriminant model from training data.*

Description

It generates a discriminant model starting from the training data, which must be provided in 2 groups depending on their classification. The method first obtains the variances and correlations using MODWT, the f filter is applied with a number of levels lev. Then a subset of all the generated features will be obtained by means of a stepwise discriminant, which can be driven by a maximum number of features or by a minimum metric to be met. Finally, the selected discriminant model is trained with the subset obtained.

Usage

```
## S3 method for class 'array'
trainModel(
  data,
  labels,
  f,
  method,
  maxvars,
  VStep,
  lev = 0,
  features = c("Var", "Cor", "IQR", "PE", "DM"),
  nCores = 0,
  ...
)
```

Arguments

data	Sample from the population (dim x length x cases)
labels	Labeled vector that classify the observations
f	Selected filter for the MODWT (to see the available filters use the function availableFilters)
method	Selected method for the discriminant. Valid values "linear" "quadratic"
maxvars	Maximum number of variables included by the StepDiscrim algorithm (Note that if you defined this, can not define VStep). Must be a positive integer greater than 0.
VStep	Minimum value of V above which all other variables are considered irrelevant and therefore will not be included. (Note that if you defined this, can not defined maxvars).Must be a positive number greater than 0. For more information see StepDiscrim documentation
lev	Determines the number of decomposition levels for MODWT (by default the optimum is calculated). Must be a positive integer
features	A list of characteristics that will be used for the classification process. To see the available features see availableFeatures
nCores	Determines the number of processes that will be used in the function, by default it uses all but one of the system cores. Must be a positive integer, where 0 corresponds to the default behavior.
...	Additional arguments

Value

A discriminant model object (lda or qda)

See Also

- StepDiscrim
- StepDiscrimV
- trainModel

Examples

```
load(system.file("extdata/ECGExample.rda",package = "TSEAL"))
# The dataset has the first 5 elements of class 1 and the last 5 of class 2.
labels <- c(rep(1, 5), rep(2, 5))
model <- trainModel(ECGExample, labels, "d6", "linear",
  maxvars = 5, features = c("Var")
)
# or using VStep
modelV <- trainModel(ECGExample, labels, "d6", "linear",
    VStep = 14.5, features = c("Var")
)
```

`trainModel.MultiWaveAnalysis` *Generates a discriminant model from an already generated "MultiWaveAnalysis".*

Description

Generates a discriminant model from an already generated "MultiWaveAnalysis".

Usage

```
## S3 method for class 'MultiWaveAnalysis'
trainModel(data, labels, method, ...)
```

Arguments

`data`	A MultiWaveAnalysis object obtained with MultiWaveAnalysis function
`labels`	Labeled vector that classify the observations.
`method`	Selected method for discrimination. Valid options are "linear" and "quadratic"
`...`	Additional arguments

Value

A discriminant model based on selected method. It can be an object of the class lda or qda.

See Also

- `MultiWaveAnalysis`
- `StepDiscrim`
- `StepDiscrimV`

Examples

```
load(system.file("extdata/ECGExample.rda",package = "TSEAL"))
MWA <- MultiWaveAnalysis(ECGExample, "d6", features = c("Var"))
MWADiscrim <- StepDiscrim(MWA, c(rep(1, 5), rep(2, 5)), 5,
  features = c("Var")
)
model <- trainModel(MWADiscrim, c(rep(1, 5), rep(2, 5)), "linear")
```

Index