Package 'classoptimr'

November 18, 2017

Type Package

Title Optimal classification schemes for prediction models with continuous response variable

Version 0.1.0

Maintainer Andreas Hill <andreas.hill@usys.ethz.ch>

Description Core of this package is a heuristic optimization procedure (Simulated Annealing) that allows for identifying optimal classification schemes for prediction models with continuous response variables. The implemented methods were primarily developed to quantify the classification accuracy of prediction maps based on statistical models that provide predictions on a continuous scale. In many cases, these continuous predictions are afterwards discretized into classes for better visualization purposes without considering the resulting accuracies of the classification scheme. In a more general context, the optimization method can also be used to detect non-constant prediction performance of statistical models.

License GPL (>= 2)

Imports Hmisc, XML, RColorBrewer, zoo

LazyData TRUE

RoxygenNote 6.0.1

R topics documented:

classAccuracy *classAccuracy*

Description

Calculates common classification accuracies measures known from remote sensing applications based on a Confusion Matrix that compares Reference classes to their respective class predictions. The computed accuracy measures are:

- *Overall Classification Accuracy (OAA)*
- *Producer's Accuracy (PA)*
- *User's Accuracy (UA)*
- *Cohen's Kappa Coefficient (k)*
- *Quantity Disagreement*
- *Allocation Disagreement*

Usage

```
classAccuracy(..., conf.level = 0.95)

## S3 method for class 'hsmclass'
classAccuracy(object, conf.level = 0.95, ...)

## Default S3 method:
classAccuracy(refdata, predictions, equal.int = NA,
  def.int = NA, conf.level = 0.95, ...)
```

Arguments

`...`	further arguments passed to or used by methods.
`conf.level`	the confidence level used to compute the confidence intervals of the overall accuracy.
`object`	object of class `'hsmclass'` created by function `HSMclass`.
`refdata`	`vector` containing the values of the continuous response variable used in the prediction model.
`predictions`	`vector` containing the predictions for the response values of the prediction model. **Note:**(`refdata` and `predictions` have to correspond to each other).
`equal.int`	an equidistant class interval to be evaluated as classifciation scheme. Defaults to `NA`. Can only be set if `def.int` is `NA`.
`def.int`	a `vector` defining an arbitrary set of class breaks to be evaluated as classifciation scheme. Defaults to `NA`. Can only be set if `def.int` is `NA`.

Value

`classAccuracy` returns an object of class `"classaccur"`

An object of class `"classaccur"` returns a `list` of the following components:

`rsquared_val` the coefficient of determination of the prediction model, calulated based on `refdata` and `predictions`

.

`predictions` `vector` containing the predictions given to `classAccuracy`

`classwidth` `vector` containing the class width

`def.classbreaks` `vector` containing arbitrary class break values

`equal.classbreaks` `vector` containing equidistant class break values

`overall.accuracy` the overall accuracy of the classification scheme

`conf.oaa` `vector` containing `[1]` the lower confidence limit of the OAA, `[2]` the upper confidence limit of the OAA, and `[3]` the confidence level

`prodaccuracy` a `matrix` containing the producer's accuracy for each class

`usersaccuracy` a `matrix` containing the user's accuracy for each class

`no.ref.classes` a `vector` containing the number of reference data for each class

`cohenskappa` Cohen's Kappa Coefficient

`map.accuracy` the map accuracy

`Quantity.Disagreement` the quantity disagreement

`Allocation.Disagreement` the allocation disagreement

References

Congalton, R.G.; Green, K. *Assessing the Accuracy of Remotely Sensed Data: Principles and Practices*; Lewis Publications: Boca Raton, FL, USA, 1999; p. 137.

Richards, J.A. *Remote Sensing Digital Image Analysis: An Introduction*, 5th ed.; Springer: Berlin, Germany, 2013.

Hill, A., Breschan, J., & Mandallaz, D. (2014). Accuracy assessment of timber volume maps using forest inventory data and LiDAR canopy height models. *Forests*, **5(9)**, 2253-2275.

Examples

```
#----
# 1.) Example: classification accuracy for equidistant class width of 100:
#----

acc.equal<- classAccuracy(refdata.gr, predictions.gr, equal.int = 100)
summary(acc.equal)


#----
# 2.) Example: classification accuracy for arbitrary class breaks:
#----

acc.def<- classAccuracy(refdata.gr, predictions.gr,
```

```
                        def.int = c(0, 150, 200, 430, 610, 880))
summary(acc.def)


#----
# 3.) Example: classification accuracy for optimal class breaks:
#----

# run HSMclass:
## Not run:
hsm<- HSMclass(refdata.gr, predictions.gr, nclasses = 6,
               iterations = 1000, coolfactor=0.99, InitTemp = 80,
               weight.norefs = 2, weight.classwidth = 2)

# calculate accuracy:
acc.opti<- classAccuracy(hsm)
summary(acc.opti)

## End(Not run)
```

`classoptimr` *classoptimr: A package for identifying optimal classification schemes for models with continuous response and predictions*

Description

Core of this package is a heuristic optimization procedure (Simulated Annealing) that allows for identifying optimal classification schemes for models that use continuous response variables and produce predictions on a continuous scale. The implemented methods were primarily developed to quantify the classification accuracy of prediction maps based on statistical models that provide predictions on a continuous scale (see *references*). In many cases, these continuous predictions are afterwards discretized into classes for better visualization purposes without considering the resulting accuracies of the created classification scheme. In a more general modelling context, the optimization method can also be used to detect non-constant prediction performance of statistical models.

Functions

The package provides three main functions to apply:

- `HSMclass` Function to identify an optimal classification scheme for a predefined number of classes GIVEN a set of response *reference* and corresponding *predicted* values.
- `classAccuracy` Function to evaluate a classification scheme by calculating various classification accuracy measures.
- `create_qml` Function to create a qml-file of a classification scheme for visualization in the open source Geographical Information System *QGIS*.

References

Hill, A., Breschan, J., & Mandallaz, D. (2014). Accuracy assessment of timber volume maps using forest inventory data and LiDAR canopy height models. *Forests*, **5(9)**, 2253-2275.

create_qml *create_qml*

Description

Creates a qml-file of a given classification scheme produced by `classAccuracy` for visualization of the respective prediction map in QGIS.

Usage

```
create_qml(x, ...)

## S3 method for class 'classaccur'
create_qml(x, out.file, pallete.name = "YlOrRd", ...)
```

Arguments

`x`	object of class `'classaccur'` containing an evaluated classification scheme.
`...`	additional arguments, so far ignored.
`out.file`	destination-folder were the qml-file should be stored
`pallete.name`	code of color-palette to be used for visualization (see `display.brewer.all`). For an overview type `display.brewer.all()` into prompt. Defaults to "YlOrRd".

Examples

```
## -- create qml-file based on 'classaccur'-object:

# 1) get optimal classification scheme:

hsm<- HSMclass(refdata.gr, predictions.gr, nclasses = 6,
               iterations = 1000, coolfactor=0.99, InitTemp = 80,
               weight.norefs = 2, weight.classwidth = 2)

# 2) evaluate classification scheme:
acc.opti<- classAccuracy(refdata.gr, predictions.gr, def.int = hsm$best.classbreaks)

# 3) write qml-file for classification scheme:
## Not run:
create_qml(acc.opti, out.file = "opti_class.qml", pallete.name = "YlOrRd")

## End(Not run)
```

HSMclass *HSMclass*

Description

Identification of optimal classification schemes by Heuritic Search Method (HSM) by Simulated Annealing

Usage

```
HSMclass(refdata, predictions, nclasses, moveinterval = 10, iterations,
  coolfactor, InitTemp, weight.norefs, weight.classwidth,
  bestever.iteration = 10, progressbar = TRUE, trace = FALSE)
```

Arguments

refdata	vector containing the values of the continuous response variable used in the prediction model
predictions	predictions for the response values of the prediction model. **Note:**(refdata and predictions have to correspond to each other)
nclasses	number of classes for which an optimal classification scheme should be computed
moveinterval	controls if classes should be whole numbers (default is 10).
iterations	number of iterations used in heuristic
coolfactor	cooling factor of Simulated Annealing algorithm
InitTemp	intial Temperature of Simulated Annealing algorithm
weight.norefs	weight for maximizing number of reference data pefor each class
weight.classwidth	weight for minimizing the classwidth for each class
bestever.iteration	number of times the heuristic is repeatedly applied. If > 1, the best solution over all runs will be chosen as the optimal solution. Defaults to 10.
progressbar	Shows the progress of the heuristic. Defaults to TRUE.
trace	logical. If TRUE, prints current solution- and penalty-term values to the console.

Value

HSMclass returns an object of class "hsmclass".

An object of class "hsmclass" returns a list of the following components:

best.classbreaks	codevector containing the class break values of the optimal classification scheme
best.classwidth	codevector containing the class width of each class
no.refs.best	codevector containing the number of reference data in each class
BestSolution	value of best solution found by heuristic
Solutions	codevector containing all solution values of heuristic

`bestever.iterationmode`
: codevector containing the class break values of the optimal classification scheme

`Temperature`
: codevector containing the temperature values of the Simulated Annealing algorithm

`deltaF`
: codevector containing the differences between new solution and best solution at the respective iteration of the heuristic

`p`
: codevector containing the p-values (...). Improvement of Sol.Best always yields p = 1

`moved.per.iteration`
: codevector containing the number of classbreaks moved for respective iteration

`comp.time`
: information about the computation time

`call`
: the function call passed to function `HSMclass`

`settings`
: a `list` containing the function's inputs:

- `refdata`:
- `predictions`:
- `nclasses`:
- `iterations`:
- `moveinterval`:
- `coolfactor`:
- `InitTemp`:
- `weight.norefs`:
- `weight.classwidth`:

References

Hill, A., Breschan, J., & Mandallaz, D. (2014). Accuracy assessment of timber volume maps using forest inventory data and LiDAR canopy height models. *Forests*, **5(9)**, 2253-2275.

Examples

```
# ------------------------------------------------#
## PERFORM OPTIMIZATION OF CLASSIFICATION SCHEME:

#----
# 1.) Example: optimization for 6 classes:
#----

hsm.1<- HSMclass(refdata = refdata.gr, predictions = predictions.gr,
                 nclasses = 6,iterations = 1000, coolfactor=0.99,
                 InitTemp = 80, weight.norefs = 2, weight.classwidth = 2)
summary(hsm.1)


#----
# 2.) Example: optimization for 6 classes, run heuristic 100 times
#----           and pick best solution over all runs:
 ## Not run:
 hsm.2<- HSMclass(refdata = refdata.gr, predictions = predictions.gr,
                  nclasses = 6,
                  iterations = 1000, coolfactor=0.99, InitTemp = 80,
                  weight.norefs = 2, weight.classwidth = 2,
```

```
                  bestever.iteration = 100)
summary(hsm.2)

## End(Not run)


# ------------------------------------------------#
##  PERFORM ENTIRE ANALYSIS:

# define a set of equidistant intervals to evaluate:
equal.intervals<- seq(100,300,20)

# define corresponding number of classes:
n.classes<- ceiling(max(refdata.gr, predictions.gr)/equal.intervals)

# Chain of analysis:
# --> 1. Identify optimal classification scheme for all given number of classes
# --> 2. Calculate classification accuracy for equidistant class intervals
# --> 3. Calculate classification accuracy for corresponding optimal no. of classes

## Not run:
acc.equal<- list()
acc.opti<-  list()

lapply(seq_along(n.classes), function(x){

  hsm<- HSMclass(refdata.gr, predictions.gr, nclasses = n.classes[x],
                 iterations = 1000, coolfactor = 0.99, InitTemp = 80,
                 weight.norefs = 2, weight.classwidth = 2)

 acc.equal[[x]]<- classAccuracy(refdata.gr, predictions.gr, equal.int = equal.intervals[x])
 acc.opti[[x]]<- classAccuracy(hsm)

})

## End(Not run)
```

---

optcrit *optcrit*

Description

Displays the values of the 3 optimization criteria of HSMclass

Usage

```
optcrit(object, ...)

## S3 method for class 'summary.classaccur'
optcrit(object, ...)
```

Arguments

object	object of class 'summary.classaccur' containing an evaluated classification scheme.
...	additional arguments, so far ignored.

---

plot.hsmclass *Plotting objects of class* 'hsmclass'

Description

Function plot.hsmclass provides four plots that describe the behaviour of the Simulated Annealing Heuristic applied by HSMclass. The plots can be used to optimize the heuristics' behaviour by altering the optimization parameters coolfactor and InitTemp in HSMclass.

Usage

```
## S3 method for class 'hsmclass'
plot(x, ...)
```

Arguments

x	object of class 'hsmclass' containing results of a optimized classification scheme
...	additional arguments, so far ignored.

Examples

```
hsm<- HSMclass(refdata.gr, predictions.gr, nclasses = 6,
                iterations = 1000, coolfactor=0.99, InitTemp = 80,
                weight.norefs = 2, weight.classwidth = 2)
plot(hsm)
```

---

predictions.gr *A named vector containing timber volume predictions based on a multiple linear regression model for each terrestrial observed timber volume stored in* refdata.gr.

Description

A named vector containing timber volume predictions based on a multiple linear regression model for each terrestrial observed timber volume stored in refdata.gr.

Usage

```
predictions.gr
```

Format

An object of class numeric of length 67.

References

Hill, A., Breschan, J., & Mandallaz, D. (2014). Accuracy assessment of timber volume maps using forest inventory data and LiDAR canopy height models. *Forests*, 5(9), 2253-2275.

`refdata.gr` *A named vector containing observations of the terrestrial timber volume at 67 systematically arranged sample plots. Data is provided from a terrestrial forest inventory in the canton of Grisons, Switzerland in the year 2007*

Description

A named vector containing observations of the terrestrial timber volume at 67 systematically arranged sample plots. Data is provided from a terrestrial forest inventory in the canton of Grisons, Switzerland in the year 2007

Usage

```
refdata.gr
```

Format

An object of class `numeric` of length 67.

Source

The terrestrial data are kindly provided by the forest service of the canton grisons.

`summary.classaccur` *Summarizing Evaluation of Classification Accuracy*

Description

`summary` methods for class `'classaccur'`

Usage

```
## S3 method for class 'classaccur'
summary(object, ...)
```

Arguments

`object`	object of class `'classaccur'`
`...`	additional arguments, so far ignored.

Value

summary.classaccur returns an object of class "summary.classaccur".

An object of class "summary.classaccur" returns a list of the following components:

accmat	a data.frame summarizing the user's- and producer's accuracy
rsquared_val	the coefficient of determination of the prediction model, calulated based on refdata and predictions
predictions	vector containing the predictions given to classAccuracy
overall.accuracy	the overall accuracy of the classification scheme
conf.oaa	vector containing [1] the lower confidence limit of the OAA, [2] the upper confidence limit of the OAA, and [3] the confidence level
cohenskappa	Cohen's Kappa Coefficient
Quantity.Disagreement	the quantity disagreement
Allocation.Disagreement	the allocation disagreement

Examples

```
## -- Summarize classification accuracy:

acc.equal<- classAccuracy(refdata.gr, predictions.gr, equal.int = 100)
summ.<- summary(acc.equal)

# print summary-object:
summ.

# extract accuracy-data.frame:
summ.$accmat
```

---

summary.hsmclass *Summarizing Optimization of Classification Scheme*

Description

summary methods for class 'hsmclass'

Usage

```
## S3 method for class 'hsmclass'
summary(object, ...)
```

Arguments

object	object of class 'hsmclass'
...	additional arguments, so far ignored.

Value

`summary.hsmclass` returns an object of class `"summary.hsmclass"`.

An object of class `"summary.hsmclass"` returns a `list` of the following components:

`call`	the function call passed to function `HSMclass`
`iterations`	number of iterations used in heuristic
`bestever.iterationmode`	number of times the heuristic should be repeated. `NA` indicates that option was not used.
`classmat`	a `data.frame` summarizing the identified optimal classification scheme

See Also

`HSMclass`

Index