Package 'arulesCBA'

April 3, 2017

Version 1.1.1

Date 2017-02-15

Title Classification Based on Association Rules

Author Ian Johnson [aut, cre, cph], Michael Hahsler [ctb]

Description Provides a function to build an association rule-based classifier for data frames, and to classify incoming data frames using such a classifier.

Maintainer Ian Johnson <ianjjohnson@icloud.com>

Depends R (>= 3.2.0), Matrix (>= 1.2-0), arules (>= 1.5-2)

Imports caret, methods, testthat

Suggests e1071, gmodels, discretization

License GPL-3

BugReports https://github.com/ianjjohnson/arulesCBA

NeedsCompilation yes

Repository CRAN

Date/Publication 2017-04-03 20:19:05 UTC

R topics documented:

bCBA *Classification Based on Association Rules*

Description

Build a classifier using a transaction boosting classification by association algorithm. The algorithm is currently in development, and is not yet formally documented.

Usage

```
bCBA(formula, data, support = 0.2,
    confidence = 0.8, gamma = 0.05, cost = 10.0,
    verbose=FALSE, parameter = NULL, control = NULL,
    sort.parameter=NULL, lhs.support=TRUE, class.weights=NULL,
disc.method="cluster", disc.categories=10)
```

Arguments

`formula`	A symbolic description of the model to be fitted. Has to be of form `class ~ .`. The class is the variable name (part of the item label before =).
`data`	A data.frame containing the training data.
`support, confidence`	Minimum support and confidence for creating association rules.
`gamma, cost`	Hyperparameters for the bCBA algorithm.
`verbose`	Optional logical flag to allow verbose execution, where additional intermediary execution information is printed at runtime.
`parameter, control`	Optional parameter and control lists for apriori.
`sort.parameter`	Ordered vector of arules interest measures (as characters) which are used to sort rules in preprocessing.
`lhs.support`	Logical variable, which, when set to default value of True, indicates that LHS support should be used for rule mining.
`class.weights`	Weights that should be assigned to the rows of each class (ordered by appearance in levels(classColumn))
`disc.method`	Discretization method for arules discretize function for factorizing numeric input (default: "cluster")
`disc.categories`	Number of discretization categories for arules discretize function for factorizing numeric input (default: 10)

Details

Formats the input data frame and calls a C implementation of a transaction-boosted classification algorithm which is currently being developed. This R package provides an interface to the current most stable release

Before the 'bCBA' algorithm in C is executed, association rules are generated with the Apriori algorithm from the arules package.

A default class is selected for the classifier. Note that for datasets which do not yield any strong association rules it's possible that no rules will be included in the classifier, and only a default class.

Value

Returns an object of class `CBA` representing the trained classifier with fields:

`rules`	the classifier rule base.
`default`	deault class label.
`levels`	levels of the class variable.

Author(s)

Ian Johnson

See Also

`predict.CBA`, `CBA`, `apriori`, `rules`, `transactions`.

Examples

```
library(arulesCBA)
library(discretization)
data(iris)
irisDisc <- mdlp(iris)$Disc.data
irisDisc <- as.data.frame(lapply(irisDisc, as.factor))
classifier <- bCBA(Species ~ ., data = irisDisc, supp = 0.05, conf=0.9, lhs.support=TRUE)
result <- predict(classifier, irisDisc)
```

`CBA` *Classification Based on Association Rules*

Description

Build a classifier based on association rules mined for an input dataset. The CBA algorithm used is a modified version of the algorithm described by Liu, et al. 1998.

Usage

```
CBA(formula, data, support = 0.2, confidence = 0.8,
    verbose=FALSE, parameter = NULL, control = NULL,
    sort.parameter = NULL, lhs.support = TRUE,
disc.method = "cluster", disc.categories = 10)
```

Arguments

formula	A symbolic description of the model to be fitted. Has to be of form `class ~ .`. The class is the variable name (part of the item label before =).
data	A data.frame containing the training data.
support, confidence	Minimum support and confidence for creating association rules.
verbose	Optional logical flag to allow verbose execution, where additional intermediary execution information is printed at runtime.
parameter, control	Optional parameter and control lists for apriori.
sort.parameter	Ordered vector of arules interest measures (as characters) which are used to sort rules in preprocessing.
lhs.support	Logical variable, which, when set to default value of True, indicates that LHS support should be used for rule mining.
disc.method	Discretization method for arules discretize function for factorizing numeric input (default: "cluster")
disc.categories	Number of discretization categories for arules discretize function for factorizing numeric input (default: 10)

Details

Formats the input data frame and calls a C implementation of the CBA algorithm from Liu, et al. 1998 split up into three stages to build a classifier based on a set of association rules.

Before the CBA algorithm in C is executed, association rules are generated with the Apriori algorithm from the arules package.

A default class is selected for the classifier. Note that for datasets which do not yield any strong association rules it's possible that no rules will be included in the classifier, and only a default class.

Value

Returns an object of class `CBA` representing the trained classifier with fields:

rules	the classifier rule base.
default	default class label.
levels	levels of the class variable.

Author(s)

Ian Johnson

References

Liu, B. Hsu, W. and Ma, Y (1998). Integrating Classification and Assocoiation Rule Mining. *KDD'98 Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining,* New York, 27-31 August. AAAI. pp. 80-86.

See Also

`predict.CBA`, `apriori`, `rules`, `transactions`.

Examples

```
library(arulesCBA)
library(discretization)

data(iris)
irisDisc <- mdlp(iris)$Disc.data
irisDisc <- as.data.frame(lapply(irisDisc, as.factor))

classifier <- CBA(Species ~ ., data = irisDisc, supp = 0.05, conf=0.9, lhs.support=TRUE)

result <- predict(classifier, irisDisc)
```

CBA_ruleset *Rules-based Classifier from Association Rules*

Description

Use a set of association rules for classification.

Usage

```
CBA_ruleset(formula, rules, method = "first", weights = NULL, default = NULL,
  description = "Custom rule set")
```

Arguments

`formula`	A symbolic description of the model to be fitted. Has to be of form `class ~ .`. The class is the variable name (part of the item label before =).
`rules`	A set of association rules (from **arules**).
`method`	Classification method `"first"` found rule or `"majority"`.
`weights`	Rule weights for method majority. If missing, then equal weights are used
`default`	Default class of the form `variable=level`. If not specified then the most frequent RHS in rules is used.,
`description`	Description field used when the classifier is printed.

Details

This method just takes a user provided set of association rules to produce a classifier. The user needs to make sure that the rules are predictive and sorted from most to least predictive.

Value

Returns an object of class `CBA` representing the trained classifier with fields:

`rules`	the classifier rule base.
`class`	class variable.
`levels`	levels of the class variable.
`default`	default class label.
`method`	classification method.
`weights`	rule weights.

Author(s)

Michael Hahsler

See Also

`predict.CBA`, `apriori`, `rules`, `transactions`.

Examples

```
library("caret")

data(iris)
irisDisc <- as.data.frame(lapply(iris[1:4],
  function(x) discretize(x, categories=9)))
irisDisc$Species <- iris$Species

# create transactions
trans <- as(irisDisc, "transactions")
truth <- irisDisc$Species

# create rule base
rules <- apriori(trans, parameter=list(support=.01, confidence = .8),
  appearance = list(rhs=grep("Species=", itemLabels(trans), value = TRUE), default = "lhs"))

rules <- rules[!is.redundant(rules)]
rules <- sort(rules, by = "conf")

# create classifier
cl <- CBA_ruleset(Species ~ ., rules)
cl

p <- predict(cl, trans)
confusionMatrix(p, truth)
```

```
# use weighted majority
cl <- CBA_ruleset(Species ~ ., rules, method = "majority", weights = "lift")
cl

p <- predict(cl, trans)
confusionMatrix(p, truth)
```

---

predict *Classification with CBA classifier*

Description

Uses a classifier based on association rules to classify a new set of data entries.

Usage

```
## S3 method for class 'CBA'
predict(object, newdata, ...)
```

Arguments

object	A CBA classifier object with a default class and a sorted list of association rules
newdata	A data.frame containing rows of new entries to be classified
...	Additional arguments not used.

Details

Runs a linear pass through newdata and uses the CBA classifier to assign it a class.

Value

Returns a vector of class labels, one for rows in newdata.

Author(s)

Ian Johnson

Examples

```
data(iris)
irisDisc <- as.data.frame(lapply(iris[1:4], discretize, categories=9))
irisDisc$Species <- iris$Species
irisDisc <- irisDisc[sample(1:nrow(irisDisc)),]


# train classifier on the first 100 examples
classifier <- CBA(Species ~ ., irisDisc[1:100,], supp = 0.05, conf=0.8)
classifier
```

```
# predict the class for the remaining 50 examples
results <- predict(classifier, irisDisc[101:150,])
table(results, irisDisc$Species[101:150])

# use caret to get more statistics
library("caret")
confusionMatrix(results, irisDisc$Species[101:150])
```

`rules` *Get the Association Rules from a CBA classifier*

Description

Returns the association rules used in the model from a CBA classifier object.

Usage

```
rules(classifier)
```

Arguments

`classifier`	An object of class `CBA`.
`...`	Additional arguments not used

Details

Returns an arules rule set (object of class `rules`) of all of the association rules used in the classifier.

Author(s)

Ian Johnson

Examples

```
data(iris)
irisDisc <- as.data.frame(lapply(iris[1:4],
  function(x) discretize(x, categories=9)))
irisDisc$Species <- iris$Species

classifier <- CBA(Species ~ ., irisDisc, supp = 0.05, conf=0.9)
rules <- rules(classifier)
inspect(rules)
```

Index