Package 'arulesCBA'

April 24, 2018

Version 1.1.3-1

Date 2018-04-23

Title Classification Based on Association Rules

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 (>= 1.6-0), discretization (>= 1.0-1)

Imports methods, testthat

License GPL-3

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

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

NeedsCompilation yes

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

Repository CRAN

Date/Publication 2018-04-23 22:20:44 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 = FALSE,
     class.weights = NULL, disc.method = "mdlp")
```

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 factorizing numeric input (default: `"mdlp"`). See `discretizeDF.supervised` for more supervised discretization methods.

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

```
data("iris")

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

predict(classifier, head(iris))
```

CBA *Classification Based on Association Rules Algorithm (CBA)*

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 = FALSE,
disc.method = "mdlp")
```

Arguments

formula	A symbolic description of the model to be fitted. Has to be of form class ~ . or class ~ predictor1 + predictor2.
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 TRUE, indicates that LHS support should be used for rule mining. lhs.support rule mining is considerably slower than normal mining.
disc.method	Discretization method for factorizing numeric input (default: "mdlp"). See discretizeDF.supervised for more supervised discretization methods.

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 is possible that no rules will be included in the classifier, and only a default class.

Value

Returns an object of class CBA.object representing the trained classifier.

Author(s)

Ian Johnson

References

Liu, B. Hsu, W. and Ma, Y (1998). Integrating Classification and Association 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

CBA.object, discretizeDF.supervised, apriori, rules, transactions.

Examples

```
data("iris")

# learn a classifier using automatic default discretization
classifier <- CBA(Species ~ ., data = iris, supp = 0.05, conf = 0.9)
classifier

# make predictions for the first few instances of iris
predict(classifier, head(iris))
```

```
# inspect the rule base
inspect(rules(classifier))

# learn classifier from transactions
trans <- as(discretizeDF.supervised(Species ~ ., iris), "transactions")
classifier <- CBA(Species ~ Sepal, data = trans, supp = 0.05, conf = 0.9)
classifier
predict(classifier, head(trans))
```

CBA.object *Objects for Classifiers Based on Association Rules*

Description

Objects for classifiers based on association rules have class "CBA". A creator function CBA_ruleset() and several methods are provided.

Usage

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

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

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 class association rules mined with mineCars or apriori (from **arules**).
method	Classification method "first" found rule or "majority".
weights	Rule weights for method majority. Either a quality measure available in rules or a numeric vector of the same length are rules can be specified. 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.
x, object	An object of class CBA.
newdata	A data.frame or transactions containing rows of new entries to be classified.
...	Additional arguments currently not used.

Details

`CBA_ruleset` creates a new object of class `CBA` using the provides rules as the rule base. For method `"first"`, the user needs to make sure that the rules are predictive and sorted from most to least predictive.

Value

`CBA_ruleset()` 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.

`predict` retuns predicted labels for `newdata`.

`rules` returns the rule base.

Author(s)

Michael Hahsler

See Also

`CBA`, `mineCARs`, `apriori`, `rules`, `transactions`.

Examples

```
data("iris")
iris.disc <- discretizeDF.supervised(Species ~., iris)

# create transactions
trans <- as(iris.disc, "transactions")
truth <- iris.disc$Species

# create rule base with CARs
cars <- mineCARs(Species ~ ., trans, parameter = list(support = .01, confidence = .8))

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

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

# look at the rule base
rules(cl)
```

```
# make predictions
prediction <- predict(cl, trans)
table(prediction, truth)

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

prediction <- predict(cl, trans)
table(prediction, truth)
```

`discretizeDF.supervised` *Supervised Methods to Convert Continuous Variables into Categorical Variables*

Description

This function implements several supervised methods to convert continuous variables into a categorical variables (factor) suitable for association rule mining and building associative classifiers. A whole data.frame is discretized (i.e., all numeric columns are discretized).

Usage

```
discretizeDF.supervised(formula, data, method = "mdlp", dig.lab = 3, ...)
```

Arguments

`formula`	a formula object to specify the class variable for supervised discretization and the predictors to be discretized in the form `class ~ .` or `class ~ predictor1 + predictor2`.
`data`	a data.frame containing continuous variables to be discretized
`method`	discretization method. Available are: `"mdlp"`, `"caim"`, `"cacc"`, `"ameva"`, `"chi2"`, `"chimerge"`, `"extendedchi2"`, and `"modchi2"`.
`dig.lab`	integer; number of digits used to create labels.
`...`	Additional parameters are passed on to the implementation of the chosen discretization method.

Details

`discretizeDF.supervised` only implements supervised discretization. See `discretizeDF` in package **arules** for unsupervised discretization.

Value

`discretizeDF` returns a discretized data.frame. Discretized columns have an attribute `"discretized:breaks"` indicating the used breaks or and `"discretized:method"` giving the used method.

Author(s)

Michael Hahsler

See Also

Unsupervised discretization from **arules**: discretize, discretizeDF.

Details about the available supervised discretization methods from **discretization**: mdlp, caim, cacc, ameva, chi2, chiM, extendChi2, modChi2.

Examples

```
data("iris")
summary(iris)

# supervised discretization using Species
iris.disc <- discretizeDF.supervised(Species ~ ., iris)
summary(iris.disc)

attributes(iris.disc$Sepal.Length)

# discretize the first few instances of iris using the same breaks as iris.disc
discretizeDF(head(iris), methods = iris.disc)

# only discretize predictors Sepal.Length and Petal.Length
iris.disc2 <- discretizeDF.supervised(Species ~ Sepal.Length + Petal.Length, iris)
head(iris.disc2)
```

---

mineCARs *Mine Class Association Rules*

Description

Class Association Rules (CARs) are association rules that have only items with class values in the RHS (Liu, et al., 1998).

Usage

```
mineCARs(formula, data, parameter = NULL, control = NULL, ...)
```

Arguments

formula	A symbolic description of the model to be fitted.
data	An object of class transactions containing the training data.
parameter, control	Optional parameter and control lists for the apriori algorithm.
...	Additional parameters are currently ignored.

Value

Returns an object of class rules.

Author(s)

Michael Hahsler

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

apriori, rules, transactions.

Examples

```
data("iris")

iris.disc <- discretizeDF.supervised(Species ~ ., iris)
iris.trans <- as(iris.disc, "transactions")

# mine CARs with items for "Species" in the RHS
cars <- mineCARs(Species ~ ., iris.trans, parameter = list(support = 0.3))
inspect(cars)

# restrict the predictors to items starting with "Sepal"
cars <- mineCARs(Species ~ Sepal, iris.trans, parameter = list(support = 0.1))
inspect(cars)
```

---

wCBA *Classification Based on Association Rules*

Description

Build a classifier using a naive rule-weighting algorithm. The algorithm is currently in development, and is not yet formally documented.

Usage

```
wCBA(formula, data, support = 0.2, confidence = 0.8,
    verbose = FALSE, parameter = NULL, control = NULL,
    sort.parameter = NULL, lhs.support = FALSE, class.weights = NULL,
    disc.method = "mdlp")
```

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.
class.weights	Weights that should be assigned to the rows of each class (ordered by appearance in levels(classColumn))
disc.method	Discretization method for factorizing numeric input (default: `"mdlp"`). See `discretizeDF.supervised` for more supervised discretization methods.

Details

Mines association rules on input data and creates a weighted-vote classifier where a rules weight is the product of its support and confidence. Default class is set to the most common class in the training data.

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

```
data("iris")

classifier <- wCBA(Species ~ ., data = iris, supp = 0.05, conf = 0.9)

predict(classifier, head(iris))
```

Index