Associative Classification in R: arc, arulesCBA, and rCBA

by Michael Hahsler, Ian Johnson, Tomáš Kliegr and Jaroslav Kuchař

Abstract Several methods for creating classifiers based on rules discovered via association rule mining have been proposed in the literature. These classifiers are called associative classifiers and the best-known algorithm is Classification Based on Associations (CBA). Interestingly, only very few implementations are available and, until recently, no implementation was available for R. Now, three packages provide CBA. This paper introduces associative classification, the CBA algorithm, and how it can be used in R. A comparison of the three packages is provided to give the potential user an idea about the advantages of each of the implementations. We also show how the packages are related to the existing infrastructure for association rule mining already available in R.

Introduction

Association rule learning (Agrawal et al., 1993) was initially designed for data exploration to discover interesting patterns in very large and sparse datasets. Several years after its inception, association rule learning was also adapted to create rule-based classification models. The first algorithm called CBA (Classification Based on Associations) was introduced by Liu et al. (1998). While there were multiple follow-up algorithms providing some improvements in classification performance (e.g., CPAR (Yin and Han, 2003) and FARC-HD-OVO (Elkano et al., 2015)), these performance gains are offset by deterioration of comprehensibility of the produced set of rules. For some practical applications, CBA still provides a very good balance between accuracy, speed, and model comprehensibility. Unlike many more recent approaches, CBA classifiers are easy to interpret and apply: the ruleset is relatively small, rules are crisp (i.e., not fuzzy rules), and rules are sorted according to strength. CBA uses a simple first-match strategy for classification, where the first matching rule determines the predicted class.

With the exception of fuzzy approaches such as FARC-HD, associative classification approaches require a dataset in the form of transactions, i.e., all attributes need to be binary indicators and thus numeric attributes in the input data need to be discretized. This puts additional demands on the user and may deteriorate model fit on datasets with numerical attributes. Another disadvantage relating to CBA and most other associative classification approaches is that these algorithms require the user to specify a minimum support and a minimum confidence threshold for association rule mining. The performance (accuracy and speed) is typically very sensitive to proper selection of these threshold values. Setting these thresholds too high can result in the classifier underfitting the dataset or even an empty rule list. Too low values can lead to combinatorial explosion with an excessive number of rules generated, leading to speed and memory issues. Another limitation that applies specifically to CBA is that even when the user specifies reasonable thresholds, CBA typically produces more rules than other related approaches (Alcala-Fdez et al., 2011). These limitations may be the reason why CBA implementations have not been available in many computational environments for machine learning and statistics. However, in the last two years, three packages with CBA implementations appeared on CRAN (listed by date of the first release): **rCBA** (Kuchar, 2018), **arc** (Kliegr, 2018) and **arulesCBA** (Johnson and Hahsler, 2019). Each of these packages offers some enhancements over the original CBA algorithm to address some shortcomings of association rule-based classification.

The goal of this paper is to introduce prospective users to the concepts used in associative classification and the CBA algorithm in particular. We provide detailed information on the three available R packages and the enhancements they provide, followed by hands-on examples.

The paper is organized as follows. We first introduce association rule mining followed by a discussion of the CBA algorithm. We present existing CBA implementations and focus on the features and the use of the three new R implementations. We conclude with a short comparison of features and a run-time comparison on a typical dataset.

Background: Association rule mining

Associative classifiers like CBA are based on association rules. Mining association rules was first introduced by Agrawal et al. (1993) and, following the notation used by Agrawal et al. (1993), Hahsler et al. (2005) and Tan et al. (2006), can formally be defined as:

Let $\mathbf{D} = \{t_1, t_2, \ldots, t_m\}$ be a set of transactions called the *database*, and let $I = \{i_1, i_2, \ldots, i_n\}$ be

transaction ID	**items**
1	milk, bread
2	bread, butter
3	beer
4	milk, bread, butter
5	bread, butter

Figure 1: An example supermarket database with five transactions.

the set of all *items* considered in the database. Each transaction in **D** has a unique transaction ID and contains a subset of the items in I. To illustrate the concepts, we use a small example from the supermarket domain introduced by Hahsler et al. (2005). The set of items is $I = \{\text{milk}, \text{bread}, \text{butter}, \text{beer}\}$ and a small database containing five transactions with these items is shown in Figure 1. An example rule for the supermarket could be $\{\text{milk}, \text{bread}\} \Rightarrow \{\text{butter}\}$ meaning that if milk and bread is bought, customers also may buy butter.

A *rule* is defined as an expression $X \Rightarrow Y$ where $X, Y \subseteq I$ and $X \cap Y = \emptyset$. The sets of items (for short *itemsets*) X and Y are called *antecedent* (left-hand-side or LHS) and *consequent* (right-hand-side or RHS) of the rule. Often rules are restricted to only a single item in the consequent. *Association rules* are rules which surpass user-specified minimum support and minimum confidence thresholds. The *support*, $\text{supp}(X)$, of an itemset X is a measure of importance defined as the proportion of transactions in the dataset which contain the itemset. The *confidence* of a rule is defined as $\text{conf}(X \Rightarrow Y) = \text{supp}(X \cup Y)/\text{supp}(X)$, measuring how likely it is to see Y in a transaction containing X.

An association rule $X \Rightarrow Y$ needs to satisfy

$$\text{supp}(X \cup Y) \geq \sigma \quad \text{and} \quad \text{conf}(X \Rightarrow Y) \geq \delta,$$

where σ and δ are the minimum support and minimum confidence thresholds, respectively. For example, the rule $\{\text{milk}, \text{bread}\} \Rightarrow \{\text{butter}\}$ has a support of $1/5 = 0.2$ and a confidence of $0.2/0.4 = 0.5$ in the database in Figure 1, which means that for 50% of the transactions containing milk and bread, the rule is correct. Confidence can be interpreted as an estimate of the probability $P(Y|X)$, the probability of finding the RHS of the rule in transactions under the condition that these transactions also contain the LHS (see, e.g., Hipp et al., 2000).

Another popular measure for the importance of a rule is *lift* (Brin et al., 1997). The lift of a rule is defined as $\text{lift}(X \Rightarrow Y) = \text{supp}(X \cup Y)/(\text{supp}(X)\,\text{supp}(Y))$, and can be interpreted as the deviation of the support of the whole rule from the support expected under independence given the supports of the LHS and the RHS. Lift values greater than one indicate positive associations between the rule's LHS and RHS.

Because associative classification is based on association rules, transaction data is required as the input. Here each object (or instance) needs to be converted into a transaction containing only binary items. Discrete variables can be converted into items using a set of 0-1 dummy variables, one for each possible value. Continuous variables need to be first discretized and then converted. Typically, discretization for associative classifiers is performed using a class-based (also called supervised) discretization strategy, which identifies ranges for several intervals using information from the class variable (Yin and Han, 2003). The most popular method for class-based discretization is Minimum Description Length Principle (MDLP) discretization (Fayyad and Irani, 1993), which uses a greedy procedure to find cut points based on the entropy of the induced partition of the data with respect to the class variable. MDLP was also used in the initial paper on CBA (Liu et al., 1998). The procedure uses a stopping rule to find the appropriate number of cut points. One of the advantages of MDLP is that there are no external parameters to be set; the number of bins is determined automatically.

The CBA algorithm

Liu et al. (1998) proposed the first approach to associative classification called CBA. In CBA, a special type of association rules called *Class Association Rules (CARs)* are used for classification. A CAR is an association rule that conforms to the additional constraint that the consequent (RHS) of the rule is a single item that is associated with a class label for the classification problem. CBA proposed the following steps to perform associative classification (Vanhoof and Depaire, 2010):

1. Mine a set of class association rules (CARs),
2. prune and sort the rules,
3. classify new objects using the RHS of the first matching rule.

Within the original paper, the first step is handled by a modification of the popular APRIORI algorithm (Agrawal and Srikant, 1994) for mining CARs. The modification includes an optional pruning step based on the rule's pessimistic error rate with the goal to reduce the size of the set of considered CARs. According to results reported in Liu et al. (1998), the absence of pessimistic pruning does not affect classifier accuracy and a regular implementation of APRIORI can be used. The output of association rule learning algorithms is determined by two parameters, the minimum confidence and the support thresholds. In light of classification, confidence gives the proportion of objects correctly classified by the rule in the training set. Therefore it can be seen as an optimistic estimate of the accuracy of the rule.

The main obstacles for straightforward use of the discovered CARs as a classifier are the excessive number of rules discovered even on small datasets, the fact that contradicting rules are generated, and the absence of a default rule. To address these issues, CBA employs rule sorting and a *data coverage pruning* procedure to reduce the number of rules. Two variants were proposed in the original paper (Liu et al., 1998): the direct M1 version, and the M2 version which reduces data access. Accessing data fewer times is especially useful if the data is too large to be stored in main memory. The amount of available main memory has increased substantially since the original paper was published, and all the discussed implementations store all data in main memory, making the improvements of M2 less relevant. For pruning, the rules are first ranked in the order of their strength:

1. Rule A is ranked higher if confidence of rule A is greater than that of rule B.
2. For rules tied for 1, rule A is ranked higher if support of rule A is greater than that of rule B.
3. For rules tied for 1 and 2, rule A is ranked higher if rule A is produced before rule B in the mining process. Since APRIORI applies breath-first search, rule A is ranked higher if rule A has fewer conditions (i.e., a smaller antecedent set) than rule B.

Rules are processed in ranking order. After each rule is processed, the matching (covered) transactions are removed. If a rule does not correctly cover at least one instance, it is deleted (pruned). In CBA, data coverage pruning is combined with *default rule pruning*. A default rule is a rule added to the end of the rule set with the majority class in the uncovered transactions in the RHS and an empty LHS. This rule ensures that a query instance is always classified even if it is not matched by any other rule in the classifier. The algorithm prunes all rules below the current rule, if a default rule inserted at that place reduces the total number of errors on the training set.

Other algorithms. Since CBA was introduced, several competing associative classification approaches have been proposed to optimize accuracy, training time, and ruleset size. Two popular extensions of CBA are CMAR (Li et al., 2001) and CPAR (Yin and Han, 2003). A multiclass-focused approach called Multiclass Associative Classification (MAC) (Abdelhamid et al., 2012) has been proposed for expanding CBA with the goal of more accurately addressing classification problems with many different class labels.

Separate from CBA is a small class of associative classifiers called rule-induction classifiers which generate excessively large rulesets and then use greedy pruning strategies to reduce the size while maintaining classification accuracy. Common examples of this technique are RIPPER (Cohen, 1995) and SLIPPER (Cohen and Singer, 1999).

Recently, instead of relying on heuristics, several optimization approaches have been proposed for selecting the rules used by the classifier. Scalable Bayesian Rule Lists Model (Yang et al., 2017) tries to identify a small subset of mined CARs by optimizing the posterior of a Bayesian hierarchical model over rule lists. The method is implemented in the R package **sbrl** (Yang et al., 2019). Azmi et al. (2019) propose to view the items in CARs as binary predictor variables and use logistic regression with L1 regularization to find a small subset of association rules for classification. The approach is available in **arulesCBA** as function `RCAR()`.

While several alternative approaches have been introduced, CBA still acts as a strong contender in associative classification and is typically used as the benchmark against which new methods are assessed (Alcala-Fdez et al., 2011). A comparison between CBA and selected successors is performed in Kliegr (2017).

Implementations

There are only a few implementations of CBA available. Table 1 shows them ordered by the first release date and summarizes the used licenses and programming languages.

In the following, we discuss the three currently available implementations of CBA in R. We will first present each package individually and then compare the packages by providing code for the same

Table 1: Review of existing CBA implementations

Software	1st release	License	Language	Notes
DM-II	2001	commercial	unknown	Original implementation by Liu et al. (1998). See http://www.comp.nus.edu.sg/~dm2/
LUCS-KDD	2004	not stated	Java	Endorsed by Bing Liu, the main CBA author. See http://cgi.csc.liv.ac.uk/~frans/KDD/Software/CBA/cba.html
KEEL	2010	GPL-3	Java	At the writing of this paper not available via **RKEEL**. See http://sci2s.ugr.es/keel/
rCBA	2015	Apache 2.0	R	See https://CRAN.R-project.org/package=rCBA
arc	2016	AGPL-3	R with Java	See https://CRAN.R-project.org/package=arc
arulesCBA	2016	GPL-3	R with C	From the authors of the **arules** R package. See https://CRAN.R-project.org/package=arulesCBA

classification problem implemented with each of the packages. We will use as the example dataset the well-known iris dataset (Fisher, 1936) and split it into 80% for training and 20% for testing.

```
data("iris")
iris <- iris[sample(seq(nrow(iris))), ]
iris_train <- iris[1:(nrow(iris)*.8), ]
iris_test <- iris[-(1:(nrow(iris)*.8)), ]
```

The data contains 150 flowers described by four quantitative variables representing different measurements and a categorical variable indicating one of three different species.

```
head(iris_train)
```

```
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
18           5.1         3.5          1.4         0.3     setosa
93           5.8         2.6          4.0         1.2 versicolor
91           5.5         2.6          4.4         1.2 versicolor
92           6.1         3.0          4.6         1.4 versicolor
126          7.2         3.2          6.0         1.8  virginica
149          6.2         3.4          5.4         2.3  virginica
```

The classification problem we use for the examples is to predict species using the four measurements.

All three packages integrate with the infrastructure for association rule mining in R implemented in package **arules** (Hahsler et al., 2005) and the ecosystem of related packages (Hahsler et al., 2011). While the presented packages can perform discretization, the conversion of a dataset with continuous variable to a set of transactions with binary items, and mining class association rules (CARs) internally and transparent to the user, we will give here a short example of how the packages **arules** and **arulesCBA** can be used to perform these tasks. First, we discretize the data using supervised discretization based on minimum description length principle (MDLP) offered by packages like **discretization** (Kim, 2012). Here we use the `discretizeDF.supervised` function provided in **arulesCBA**.

```
library("arules")
library("arulesCBA")

iris_train_disc <- discretizeDF.supervised(Species ~ ., data = iris_train,
  method = "mdlp")
head(iris_train_disc)
```

```
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
18   [-Inf,5.55) [3.35, Inf]   [-Inf,2.6) [-Inf,0.75)     setosa
93   [5.55, Inf] [-Inf,2.95)   [2.6,4.75) [0.75,1.75) versicolor
91   [-Inf,5.55) [-Inf,2.95)   [2.6,4.75) [0.75,1.75) versicolor
92   [5.55, Inf] [2.95,3.35)   [2.6,4.75) [0.75,1.75) versicolor
126  [5.55, Inf] [2.95,3.35)  [5.05, Inf] [1.75, Inf]  virginica
149  [5.55, Inf] [3.35, Inf]  [5.05, Inf] [1.75, Inf]  virginica
```

Now we can convert the discretized data into transactions which automatically converts factors into binary items with labels composed of variable name and factor labels.

```
trans_train <- as(iris_train_disc, "transactions")
inspect(head(trans_train, n = 3))
```

```
    items                      transactionID
[1] {Sepal.Length=[-Inf,5.55),
     Sepal.Width=[3.35, Inf],
     Petal.Length=[-Inf,2.6),
     Petal.Width=[-Inf,0.75),
     Species=setosa}                    18
[2] {Sepal.Length=[5.55, Inf],
     Sepal.Width=[-Inf,2.95),
     Petal.Length=[2.6,4.75),
     Petal.Width=[0.75,1.75),
     Species=versicolor}                93
[3] {Sepal.Length=[-Inf,5.55),
     Sepal.Width=[-Inf,2.95),
     Petal.Length=[2.6,4.75),
     Petal.Width=[0.75,1.75),
     Species=versicolor}                91
```

Note that the class variable is translated into several items, all starting with `Species=`. From these transactions, CARs can be mined by restricting the items which can appear in the right-hand-side of the rules. This can be done with the APRIORI implementation available in **arules** by specifying appearance restrictions.

```
rules <- apriori(trans_train, parameter = list(support = 0.01, confidence = 0.8),
  appearance = list(rhs = grep("Species=",  itemLabels(trans_train), value = TRUE),
    default = "lhs"))
```

arulesCBA contains a convenience function called `mineCARs` to make setting the appropriate appearance easier using the standard formula interface.

```
rules <- mineCARs(Species ~ ., data = trans_train, support = 0.01, confidence = 0.8)
rules
```

```
set of 78 rules
```

```
inspect(head(rules, n = 3))
```

```
    lhs                           rhs                   support confidence lift count
[1] {Sepal.Width=[3.35, Inf]}  => {Species=setosa}     0.19    0.85       2.6  23
[2] {Petal.Length=[5.05, Inf]} => {Species=virginica}  0.27    1.00       3.0  32
[3] {Petal.Length=[2.6,4.75)}  => {Species=versicolor} 0.29    0.97       2.9  35
```

Test data can be discretized consistently with the training data using `discretizeDF`, which applies the discretization used in the second argument to the data in the first argument. Followed by a conversion to transactions.

```
iris_test_disc <- discretizeDF(iris_test, iris_train_disc)
trans_test <- as(iris_test_disc, "transactions")
```

While these steps are performed in most cases by the discussed packages internally, it is still helpful to understand the process. One of the advantages of associative classifiers is that the rule base can be inspected and, therefore, it is important to understand the transformations used to create items. Next, we will discuss the packages in alphabetical order.

Package arc

The R package **arc** (Kliegr, 2018) provides a pure R implementation of the rule pruning step of CBA. The association rule learning step is handled by the implementation of APRIORI in package **arules**. **arc** implements the M1 version of the CBA pruning step (Liu et al., 1998) and offers in addition automatic discretization and threshold tuning. A CBA model can be learned for the `iris` dataset as follows.

```
library("arc")
classifier <- arc::cba(iris_train, "Species")
```

The function `cba()` will create an instance of the `CBARuleModel` class for the iris dataset using *Species* as the class variable. Note that discretization is performed and that the support and confidence thresholds are automatically found.

The resulting S4 object holds a list of rules, a list of cut points (if discretization was automatically performed), the name of the class attribute, and a list of attribute types. The slot `rules` of the `CBARuleModel` object contains the rule base, which can be inspected by:

```
inspect(classifier@rules)
```

```
    lhs                          rhs                  support confidence lift count
[1] {Petal.Length=[-Inf;2.45],
     Petal.Width=[-Inf;0.75]}  => {Species=setosa}       0.333       1.00  3.0    40
[2] {Sepal.Length=(5.75; Inf],
     Petal.Length=(4.95; Inf],
     Petal.Width=(1.75; Inf]}  => {Species=virginica}    0.258       1.00  3.2    31
[3] {Sepal.Length=(5.75; Inf],
     Sepal.Width=[-Inf;3.15],
     Petal.Width=(1.75; Inf]}  => {Species=virginica}    0.200       1.00  3.2    24
[4] {Sepal.Length=(5.75; Inf],
     Petal.Length=(2.45;4.95],
     Petal.Width=(0.75;1.75]}  => {Species=versicolor}   0.200       1.00  2.8    24
[5] {Sepal.Length=(5.45;5.75],
     Sepal.Width=[-Inf;3.15],
     Petal.Length=(2.45;4.95],
     Petal.Width=(0.75;1.75]}  => {Species=versicolor}   0.092       1.00  2.8    11
[6] {Sepal.Length=(5.75; Inf],
     Sepal.Width=(3.15; Inf],
     Petal.Length=(2.45;4.95]} => {Species=versicolor}   0.042       1.00  2.8     5
[7] {Petal.Length=(2.45;4.95],
     Petal.Width=(0.75;1.75]}  => {Species=versicolor}   0.333       0.98  2.7    40
[8] {}                         => {Species=virginica}    0.308       0.31  1.0     0
```

Predictions for new data can be obtained using `predict()`. The new data is discretized automatically to match the rules.

```
predict(classifier, head(iris_test))
```

```
[1] virginica  setosa     versicolor virginica  setosa     versicolor
Levels: setosa versicolor virginica
```

Next, we discuss the new features of automatic discretization and threshold tuning.

Automatic discretization. Since association rule classification is a supervised task, the discretization can take advantage of using the class label. In the **arc** package, automatic discretization with MDLP is enabled by default. All numeric explanatory attributes with three or more distinct values are by default subject to discretization. The package relies on the **discretization** package (Kim, 2012). The **arc** package provides several convenience functions that allow to perform discretization of all attributes at once, addressing some of the shortcomings of the `mdlp` function from the **discretization** package, such as the inability to handle missing values, or skip non-numeric attributes. Only attributes containing at least a preset number of distinct values are discretized. The package is also capable of discretizing the target attribute if necessary. For this purpose, unsupervised discretization (clustering) is used.

Automatic threshold tuning. Association rule learning is notorious for how difficult it is to set the minimum support and minimum confidence thresholds. The necessity to set these thresholds applies also to CBA. The **arc** package contains an optional procedure for automatic setting of these thresholds detailed in (Kliegr and Kuchar, 2019) . The package contains a wrapper for the `apriori` function from the **arules** package that iterative changes mining parameters (maximum antecedent length, minimum support threshold and minimum confidence threshold) until a desired number of rules is obtained, all options are exhausted or a preset time limit is reached. The desired number of rules can be specified by the `target_rule_count` parameter.

The **arc** package also supports manual specification of thresholds:

```
classifier <- arc::cba(iris_train, "Species",
  rulelearning_options = list(minsupp = 0.05, minconf = 0.9,
```

```
    minlen = 1, maxlen = 5, maxtime = 1000, target_rule_count = 50000,
    trim=TRUE, find_conf_supp_thresholds = FALSE))

classifier@rules

set of 3 rules
```

Unlike other implementations of CBA, which also implement the M2 version of CBA described by Liu et al. (1998), the **arc** package relies solely on the M1 version. However, the implementation does not follow the originally proposed way relying on iterative processing of rules in the sort order. Instead, the pruning steps in M1 are implemented using more efficient multiplication of sparse matrices exposed by the **arules** package, which relies on the optimized C code from the **Matrix** package (Bates and Maechler, 2017).

Package arulesCBA

The **arulesCBA** package (Johnson and Hahsler, 2019) is an extension of the **arules** package and strives to integrate seamlessly with its association rule mining infrastructure. The packages allows the user to set a time limit for rule mining, exposed by the **arules** package. The core operations of **arulesCBA** are implemented in a mixture of R and C to speed up processing. **arulesCBA** implements both versions of the pruning step, M1 and the optimized M2 version. The code for the pruning algorithm is heavily optimized by using rule-indexed sparse matrix representation, sparse matrix operations via package **Matrix** (Bates and Maechler, 2017) and prefix trees.

The arulesCBA interface. In **arulesCBA**, classifiers are created using the `CBA()` function. An advantage of this package for R users is that it it consistently uses the well-known formula interface for building classifier models and for supervised discretization. Users can provide a number of options to the function to tune discretization, rule mining, and model building. The following is the list of available parameters to the CBA function.

- `formula`: A symbolic description of the model to be fitted using a standard `formula` object of the from:

  $$\text{class} \sim \text{explanatory variables}$$

  The class is the variable name (part of the item label before `=`). Explanatory variables are separated using $+$ and the special dot symbol . for all variables is also allowed.
- `data`: A `data.frame` containing the training data. If necessary, discretization is automatically applied. Alternatively, also a transaction set can be supplied.
- `support,confidence`: Minimum support and confidence thresholds for mining CARs with APRIORI.
- `parameter,control`: Parameter and control lists passed on to the `apriori()` function from the **arules** package.
- `disc.method`: Discretization method for factorizing numeric input (default: "mdlp"). One of ('mdlp', 'caim', 'chi2', 'caac', 'ameva', 'chimerge', 'extendedchi2', 'modchi2').

A classifier for the iris dataset can be learned as follows.

```
library("arulesCBA")
classifier <- arulesCBA::CBA(Species ~ ., data = iris_train,
   supp = 0.05, confidence = 0.9)

classifier

CBA Classifier Object
Class: Species (labels: setosa, versicolor, virginica )
Default Class: Species=setosa
Number of rules: 2
Classification method: first
Description: CBA algorithm by Liu, et al. 1998 with support=0.05 and
     confidence=0.9
```

`CBA()` returns an object of class `CBA` which contains all needed information for classification. A print method shows the settings used for the classifier. Prediction follows the usual approach in R.

```
predict(classifier, head(iris_test))

[1] virginica  setosa     versicolor virginica  setosa     setosa
Levels: setosa versicolor virginica
```

The rule base is stored as a rules object from **arules** and can be extracted for inspection using the `rules()` function.

```
inspect(rules(classifier))

    lhs                           rhs                  support confidence lift count
[1] {Petal.Width=(1.75, Inf]}  => {Species=virginica}     0.29       1.00  3.1    35
[2] {Sepal.Length=(5.55, Inf],
     Petal.Width=(0.8,1.75]}   => {Species=versicolor}    0.26       0.91  2.7    31
```

Note that only two rules are shown, while **arc** above produced three rules. The reason is that **arulesCBA** stores the default class `Species=setosa` separate from the rule base while **arc** includes it.

Advanced use of arulesCBA. **arulesCBA** is implemented with flexibility and future extensions in mind. For example, to have optimal control over the discretization process, the user can discretize the data manually before learning the classifier. The discretization functions in **arules** and **arulesCBA** retain enough information so that `predict()` can later automatically discretize the new data.

Another extension implemented in `CBA_ruleset()` allows the user to create an associative classifier by providing a custom rule base in the form of a *rules* object. For example, we can easily create a classifier from a set of CARs using, for example, majority voting instead of CBA's first-match strategy for classification.

```
rules <- mineCARs(Species ~ ., trans_train,
  parameter = list(support = 0.01, confidence = 0.8))

classifier <- arulesCBA::CBA_ruleset(Species ~ ., rules, method = "majority")
classifier

CBA Classifier Object
Class: Species (labels: setosa, versicolor, virginica )
Default Class: Species=versicolor
Number of rules: 78
Classification method: majority
Description: Custom rule set
```

This allows the user to experiment with different pruning methods and classification strategies.

Package rCBA

The **rCBA** package (Kuchar, 2018) was the first available implementation of the CBA algorithm on CRAN. The main algorithms are implemented in Java and it is the only R implementation that supports the use of multiple CPU cores during pruning. The package provides wrapper functions for pruning, prediction, and the FPGrowth association rule mining algorithm (Han et al., 2004). **rCBA** includes both, the M1 and the M2 version of the CBA algorithm. It also includes data coverage pruning and automatic threshold tuning.

Model building with automatic tuning of parameters and APRIORI is done as follows.

```
library("rCBA")
classifier <- rCBA::build(iris_train)

inspect(classifier$model)

1       {Petal.Width=0.2} => {Species=setosa}   0.183       1.00  2.9
2   {Petal.Width=1.3} => {Species=versicolor}   0.108       1.00  3.0
3      {Petal.Length=1.4} => {Species=setosa}   0.100       1.00  2.9
4      {Petal.Length=1.5} => {Species=setosa}   0.092       1.00  2.9
5    {Petal.Width=1.8} => {Species=virginica}   0.083       1.00  3.1
6    {Petal.Width=2.3} => {Species=virginica}   0.058       1.00  3.1
7       {Petal.Width=0.4} => {Species=setosa}   0.058       1.00  2.9
```

```
8   {Petal.Width=1.4} => {Species=versicolor}   0.058       1.00  3.0
9       {Sepal.Width=3.5} => {Species=setosa}   0.050       1.00  2.9
10      {Petal.Width=0.3} => {Species=setosa}   0.050       1.00  2.9
11   {Petal.Width=2.1} => {Species=virginica}   0.050       1.00  3.1
12   {Petal.Length=4} => {Species=versicolor}   0.042       1.00  3.0
13 {Petal.Length=4.7} => {Species=versicolor}   0.033       1.00  3.0
14  {Sepal.Length=7.7} => {Species=virginica}   0.033       1.00  3.1
15  {Petal.Width=1.2} => {Species=versicolor}   0.033       1.00  3.0
16      {Petal.Width=0.1} => {Species=setosa}   0.033       1.00  2.9
17   {Petal.Width=1.9} => {Species=virginica}   0.033       1.00  3.1
18    {Petal.Width=1} => {Species=versicolor}   0.033       1.00  3.0
19     {Sepal.Length=5.1} => {Species=setosa}   0.058       0.88  2.6
20 {Petal.Length=4.5} => {Species=versicolor}   0.050       0.86  2.6
21  {Petal.Length=5.1} => {Species=virginica}   0.042       0.83  2.6
22  {Petal.Width=1.5} => {Species=versicolor}   0.058       0.78  2.3
23 {Sepal.Length=5.5} => {Species=versicolor}   0.033       0.67  2.0
24                 {} => {Species=virginica}   0.325       0.33  1.0
```

```
rCBA::classification(head(iris_test), classifier$model)
```

```
[1] versicolor versicolor versicolor versicolor setosa     versicolor
Levels: setosa versicolor
```

Pruning methods. **rCBA** implements both version of the proposed pruning algorithms (Liu et al., 1998) the direct M1 version, and the optimized M2 version. It also offers an option to only use data coverage pruning, called *data coverage for business rule (dcbr)* (Kliegr et al., 2014).

Selection of algorithms for rule learning. The CBA algorithm can generally rely on any rule learning algorithm (Liu et al., 1998). By default it uses the APRIORI implementation in **arules**, but it can also use **rCBA**'s own implementation of the FP-Growth algorithm Han et al. (2004) for the association learning step.

```
rulebase <- rCBA::fpgrowth(iris_train, support = 0.05, confidence = 0.9,
  consequent = "Species")
rulebase <- rCBA::pruning(iris_train, rulebase, method = "m2cba")

rCBA::classification(head(iris_test), rulebase)
```

```
[1] versicolor versicolor versicolor versicolor setosa     versicolor
Levels: setosa versicolor
```

Automatic threshold tuning. Since pure random or grid search do not use any background knowledge of the algorithm, these approaches are unsuitable for optimizing the parameters of association rule learning. The implementation for the parameter optimization in **rCBA** is based on the simulated annealing (SA) algorithm which addresses this problems. The objective criterion, which is optimized against, is the accuracy of the model. A detailed description of the approach can be found in Kliegr and Kuchar (2019).

Comparison of R implementations

In order to help the user to decide which package addresses best the particular use case, Table 2 presents a comparison of the features and limitations of the packages. Since all three packages implement the same algorithm, we did not compare classification accuracy between implementations, but performed a small run-time comparison instead.

We compare the different implementations on some standard classification problems. The results are shown in Table 3. The datasets are available in the packages **mlbench**, **datasets**, **arules**, and the Lymphography dataset (Lymph) (Mickalski et al., 1986) was obtained from the UCI repository[1]. The number of transactions ranges from 101 to 48842 and the number of items (after discretization) from 15 to 147. We used for the comparison a minimum confidence threshold of 0.5, a maximal rule length of 10 and set the minimum support so a reasonable number of classification association rules (CARs)

[1] https://archive.ics.uci.edu/ml/datasets.html

Table 2: Comparison of features in CBA implementations in R

Feature	arc	arulesCBA	rCBA
CBA pruning	M1	M1/M2	M1/M2
Language	R	R + C	R + Java
Built-in discretization	MDLP	MDLP and others	No
Automatic threshold tuning	Unsupervised	No	Supervised
Recommended problem size	Small number of rules and instances	Many rules, many instances	Medium number of rules and instances

Table 3: Run time comparison for different datasets (* indicates that the algorithm ran out of memory).

Dataset	Transactions	Items	Support	Confidence	CARs	Rule base	Accuracy
Zoo	101	26	0.01	0.50	3607	8	0.97
Lymph	147	63	0.10	0.50	16087	40	0.90
Iris	150	15	0.01	0.50	119	8	0.96
Ionosphere	351	147	0.40	0.50	16321	10	0.89
BreastCancer	699	91	0.01	0.50	5541	64	0.99
Pima	768	19	0.01	0.50	3536	76	0.80
Vehicle	846	77	0.08	0.50	13987	143	0.60
Adult	48842	115	0.10	0.50	932	6	0.77

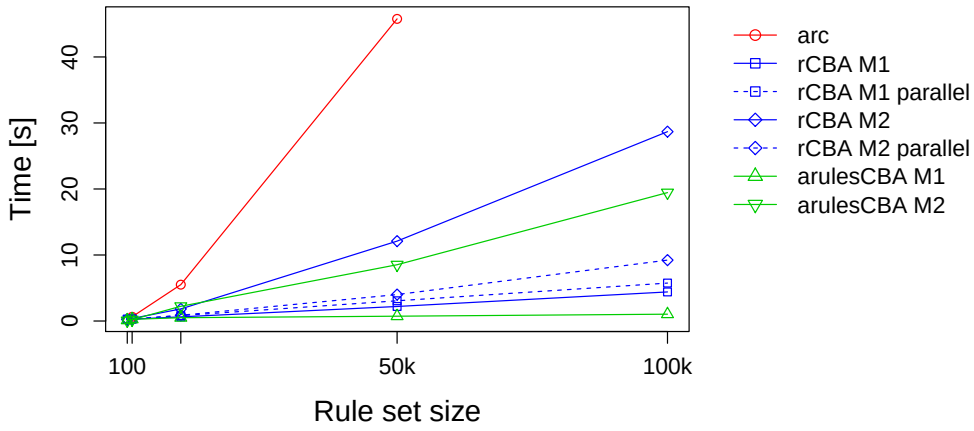
	arc	rCBA				arulesCBA	
Dataset		M1	M1 parallel	M2	M2 parallel	M1	M2
Zoo	447.49	344.39	366.09	344.99	345.57	250.40	130.19
Lymph	3153.19	903.75	1070.12	1375.30	966.58	452.91	1539.03
Iris	117.98	282.57	332.43	273.37	283.54	147.33	102.74
Ionosphere	20350.29	13888.05	14180.38	14622.77	15185.99	4786.10	6766.72
BreastCancer	1895.51	488.06	620.62	1239.72	646.96	607.53	775.93
Pima	2508.53	837.69	854.77	847.39	837.06	707.97	982.45
Vehicle	50186.80	2741.48	2961.82	2935.02	3051.59	3966.96	12662.39
Adult	N/A*	10549.87	3395.19	9725.68	3091.15	1192.03	11737.05
Average	11237*	3754.48	2972.68	3920.53	3051.05	1513.90	4337.06

was produced. CBA pruned the CARs to between 6 and 143 rules and achieves an accuracy (in sample testing) of typically around 90%. Only difficult datasets like Pima, Vehicle and Adult have worse results.

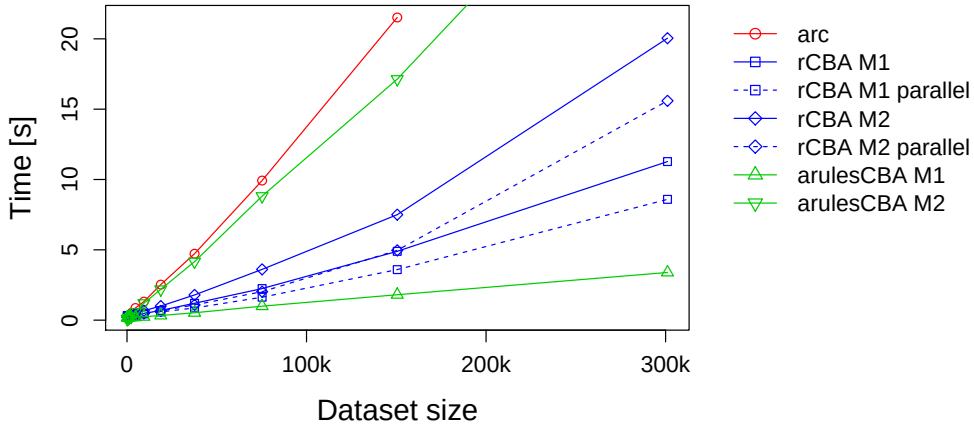
To compare run time, we conducted experiments on a standard laptop with an Intel Core i5-8250U CPU @ 1.60GHz with 4 cores and 8GB of RAM running R version 3.6.1 on Ubuntu 19.10. The package versions used for the comparison are: **arc**: 1.2, **rCBA**: 0.4.3, **arulesCBA**: 1.1.5. We disabled automatic threshold tuning. To remove the effect of random system load, we executed each algorithm ten times on each dataset and report the average execution time. The results are summarized in Table 3. **arc** produces the longest run times due to its pure R implementation. For Adult, the largest dataset **arc** ran out of memory. **rCBA** executes faster than **arc**. Both M2 and parallel execution using multi-core support in Java only improve the run time for the largest dataset. However, there the improvement is quite significant, reducing the run time to a third. **arulesCBA**'s M1 implementation is on average the fastest while the M2 implementation's performance deteriorates on larger datasets.

Since many datasets of interest are typically larger then the standard datasets, we perform additional experiments to assess run time sensitivity for the number of input rules and the dataset size. For the experiments, we use the Lymph dataset. For assessing sensitivity to ruleset size, we oversample the dataset to 500 transactions and mine CARs with a minimum support of 0.05, a minimum confidence of 0.5 and a maximal rule length of 10. This results in more than 100000 rules. We then evaluate run time for building classifiers from the first 100, 1000, 10000, and 100000 mined rules. The results are shown in Figure 2(a). We see that M2 is generally slower then the corresponding M1 implementations. This might be due to the fact that the tested implementation hold all data in main memory, while M2 was designed for situations where the data does not reside in main memory. However, parallel execution helps **rCBA**'s M2 implementation. **arulesCBA**'s M1 implementation is the fastest.

To assess the sensitivity to dataset size, we fix the ruleset size to 500 and increased the dataset size by oversampling every round by a factor of 2. In Figure 2(b), we see a similar result to the sensitivity to the number of rules. Parallel execution in **rCBA** helps both algorithms and **arulesCBA**'s



(a) Sensitivity to ruleset size.



(b) Sensitivity to dataset size.

Figure 2: Comparison of the run time of different implementations on an oversampled Lymphography dataset

M1 implementation is the fastest. All packages are integrated with the **arules** infrastructure, where **arulesCBA** has the most consistent integration. **arc** and **rCBA** offer automatic threshold tuning, which will help users with applying associative classification for practical applications.

Conclusion

In this paper, we reviewed associative classifiers based on the CBA algorithm. While the algorithm is cited in many papers about classifiers based on association rule mining, there are only very few implementations available. This paper discussed three recent implementations in R packages. Due to the differences in implementation language (R, C, and Java) and additional implemented features, each of the packages has its strengths. We hope that this review and the provided examples help users to experiment with associative classifiers and that the packages will be used by the research community to develop new methods.

Acknowledgments

Tomas Kliegr was supported by long term institutional support of research activities by Faculty of Informatics and Statistics, University of Economics, Prague.

Ian Johnson was supported by the Goldwater Foundation and the President's Scholars program at Southern Methodist University, Dallas, TX, USA.

Bibliography

N. Abdelhamid, A. Ayesh, F. Thabtah, S. Ahmadi, and W. Hadi. Mac: A multiclass associative classification algorithm. *Journal of Information & Knowledge Management*, 11(02):1250011, 2012. [p3]

R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc. ISBN 1-55860-153-8. [p3]

R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216. ACM Press, 1993. URL https://doi.org/10.1145/170035.170072. [p1]

J. Alcala-Fdez, R. Alcala, and F. Herrera. A fuzzy association rule-based classification model for high-dimensional problems with genetic rule selection and lateral tuning. *IEEE Transactions on Fuzzy Systems*, 19(5):857–872, 2011. [p1, 3]

M. Azmi, G. C. Runger, and A. Berrado. Interpretable regularized class association rules algorithm for classification in a categorical data space. *Information Sciences*, 483:313–331, 2019. ISSN 0020-0255. URL https://doi.org/0.1016/j.ins.2019.01.047. [p3]

D. Bates and M. Maechler. *Matrix: Sparse and Dense Matrix Classes and Methods*, 2017. URL https://CRAN.R-project.org/package=Matrix. R package version 1.2-8. [p7]

S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data*, pages 255–264, Tucson, Arizona, USA, May 1997. [p2]

W. W. Cohen. Fast effective rule induction. In *Machine Learning Proceedings 1995*, pages 115–123. Elsevier, 1995. [p3]

W. W. Cohen and Y. Singer. A simple, fast, and effective rule learner. *AAAI/IAAI*, 99:335–342, 1999. [p3]

M. Elkano, M. Galar, J. A. Sanz, A. Fernández, E. Barrenechea, F. Herrera, and H. Bustince. Enhancing multiclass classification in farc-hd fuzzy classifier: On the synergy between n-dimensional overlap functions and decomposition strategies. *IEEE Transactions on Fuzzy Systems*, 23(5):1562–1580, Oct 2015. ISSN 1063-6706. URL https://doi.org/10.1109/TFUZZ.2014.2370677. [p1]

U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. *Artificial intelligence, 13*, page 1022–1027, 1993. [p2]

R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(7): 179–188, 1936. [p4]

M. Hahsler, B. Grün, and K. Hornik. arules - a computational environment for mining association rules and frequent item sets. *Journal of Statistical Software*, 14(15):1–25, 9 2005. ISSN 1548-7660. URL http://www.jstatsoft.org/v14/i15. [p1, 2, 4]

M. Hahsler, S. Chelluboina, K. Hornik, and C. Buchta. The arules R-package ecosystem: analyzing interesting patterns from large transaction data sets. *Journal of Machine Learning Research*, 12(Jun): 2021–2025, 2011. [p4]

J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8(1):53–87, Jan. 2004. ISSN 1384-5810. [p8, 9]

J. Hipp, U. Güntzer, and G. Nakhaeizadeh. Algorithms for association rule mining – A general survey and comparison. *SIGKDD Explorations*, 2(2):1–58, 2000. [p2]

I. Johnson and M. Hahsler. *arulesCBA: Classification Based on Association Rules*, 2019. URL https://CRAN.R-project.org/package=arulesCBA. R package version 1.1.5. [p1, 7]

H. Kim. *discretization: Data preprocessing, discretization for classification.*, 2012. URL https://CRAN.R-project.org/package=discretization. R package version 1.0-1. [p4, 6]

T. Kliegr. QCBA: Postoptimization of quantitative attributes in classifiers based on association rules. *arXiv preprint arXiv:1711.10166*, 2017. [p3]

T. Kliegr. *arc: Association Rule Classification*, 2018. URL https://CRAN.R-project.org/package=arc. R package version 1.2. [p1, 5]

T. Kliegr and J. Kuchar. Tuning hyperparameters of classification based on associations (cba). In *Proceedings of the 19th Conference Information Technologies - Applications and Theory ITAT'19*. CEUR-WS.org, 2019. [p6, 9]

T. Kliegr, J. Kuchař, D. Sottara, and S. Vojíř. Learning business rules with association rule classifiers. In A. Bikakis, P. Fodor, and D. Roman, editors, *Rules on the Web. From Theory to Applications: 8th International Symposium, RuleML 2014, Co-located with the 21st European Conference on Artificial Intelligence, ECAI 2014, Prague, Czech Republic, August 18-20, 2014. Proceedings*, pages 236–250, Cham, 2014. Springer International Publishing. ISBN 978-3-319-09870-8. [p9]

J. Kuchar. *rCBA: CBA Classifier for R*, 2018. URL https://CRAN.R-project.org/package=rCBA. R package version 0.4.3. [p1, 8]

W. Li, J. Han, and J. Pei. CMAR: Accurate and efficient classification based on multiple class-association rules. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, ICDM '01, pages 369–376, Washington, DC, USA, 2001. IEEE Computer Society. ISBN 0-7695-1119-8. [p3]

B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, KDD'98, pages 80–86. AAAI Press, 1998. [p1, 2, 3, 4, 5, 7, 9]

R. S. Mickalski, I. Mozetic, H. J., and H. Lavrack. The multi purpose incremental learning system AQ15 and its testing application to three medical domains. In *Proceedings of the 5th National Conference on Artificial Intelligence*, 1986. [p9]

P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006. ISBN 0321321367. [p1]

K. Vanhoof and B. Depaire. Structure of association rule classifiers: a review. In *2010 International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, pages 9–12, November 2010. [p2]

H. Yang, C. Rudin, and M. Seltzer. Scalable bayesian rule lists. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3921–3930. JMLR. org, 2017. [p3]

H. Yang, M. Chen, C. Rudin, and M. Seltzer. *sbrl: Scalable Bayesian Rule Lists Model*, 2019. URL https://CRAN.R-project.org/package=sbrl. R package version 1.2. [p3]

X. Yin and J. Han. CPAR: Classification based on predictive association rules. In *Proceedings of the SIAM International Conference on Data Mining*, pages 369–376, San Franciso, 2003. SIAM Press. [p1, 2, 3]

Michael Hahsler
Office of Information Technology and Department of Engineering Management, Information, and Systems
Bobby B. Lyle School of Engineering
Southern Methodist University
P. O. Box 750123, Dallas, TX 75275, USA
mhahsler@lyle.smu.edu

Ian Johnson
Google,
Boulder, CO, USA
ianjjohnson@icloud.com

Tomáš Kliegr
Department of Information and Knowledge Engineering
Faculty of Informatics and Statistics
University of Economics, Prague
Winston Churchill Sq. 4, Prague, Czech Republic
ORCiD https://orcid.org/0000-0002-7261-0380
first.last@vse.cz

Jaroslav Kuchař
Web Intelligence Research Group

Faculty of Information Technology
Czech Technical University in Prague
Thákurova 9, 160 00, Prague, Czech Republic
first.last@fit.cvut.cz