Structure of association rule classifiers: a review.



DOI: 10.1109/ISKE.2010.5680784
Handle: http://hdl.handle.net/1942/11704

# Structure of Association Rule Classifiers: a Review

Koen Vanhoof Benoît Depaire
Transportation Research Institute (IMOB),
University Hasselt
3590 Diepenbeek, Belgium
koen.vanhoof@uhasselt.be benoit.depaire@uhasselt.be

*Abstract*— **This paper provides a short review of various association rule classifiers (ARC) that have been developed over the past decade and the common structure behind most ARCs. Furthermore, different pruning and classification schemes used in various ARCs are reviewed and two ARCs are discussed which break with the standard structure behind ARC.**

***Keywords: Classification, Association Rules, Pruning***

## I. INTRODUCTION

In 1998, Liu et al. [1] introduced CBA, the first association rule classifier (ARC). This new type of classification algorithm distinguishes itself from other classifiers by learning association rules from the training data and using these rules to classify new cases. As association rules typically represent local knowledge, all ARCs try to combine local knowledge to solve new cases. In the wake of Liu et al. [1], several other authors came up with different ARC implementations, such as CMAR [2], ARC-AC and ARC-BC [3], CPAR [4], CorClass [5] and ACRI [6], 2SARC1 and 2SARC2 [7] and ARUBAS [8].

At first sight, large differences seem to exist between these different ARCs in terms of generating a set of association rules and using this knowledge to classify new instances. Despite the differences, a common structure exists among the various ARC implementations which will be unraveled in this article.

The next section will introduce some concepts of frequent item sets and association rules, which are the basis of ARCs. Next, the common structure of many ARCs is introduced and different implementations are discussed. Furthermore, two ARCs which are exceptions to the common structure are discussed and finally some conclusions are formulated.

## II. FREQUENT ITEM SETS AND ASSOCIATION RULES

Research regarding frequent item sets and association rules goes back to an article of Agrawal et al. [9]. However frequent item set analysis typically deals with transactional data where each record is a transaction representing a set of items. Any two transactions can have a different size, i.e. contain a different number of items, and might have no items in common. Market basket analysis is a typical example of data analysis on transactional data. To measure how often a set of items occur together within a set of transactions, the item set's support is calculated as the number of transaction that contain the item set. If the support of the item set is above a predefined threshold, it is called a frequent item set.

In contrast to frequent item set analysis, classification analysis, which is what ARC's are used for, is mostly applied on rectangular data instead of transactional data. While a record in a transactional data set is represented as a set of items, e.g. $X_1 = \{A, B, D, E\}$, the same record in a rectangular data set is represented as a set of attribute-value pairs, e.g. $X_1 = \{\langle A, 1\rangle, \langle B, 1\rangle, \langle C, -\rangle, \langle D, 1\rangle, \langle E, 1\rangle\}$.

It is always possible to transform transactional data into rectangular data, by creating a binary attribute-value pair for each item. Note that a transformed rectangular record contains an attribute-value pair for all possible items, not only for those items present in the original transactional record. This is a first and important difference between transactional and rectangular data, i.e. transactional records differ in size as they only contain a subset of all possible items, while rectangular data records contain all possible attribute-value pairs and have equal size.

Another important difference between the two data set structures is that transactional data has a binary data nature, i.e. either an item is present or it is not, which explains why ARC algorithms typically require nominal data to work with. On the other hand, rectangular data can also represent non-binary data such as nominal, categorical and continuous data. Transforming rectangular data to transactional data can be done in two steps. First all attributes must be transformed into nominal attributes by discretizing continuous variables and ignoring the order of categorical variables. Next, the nominal attributes are transformed into binary dummy variables.

As this article focuses on ARC algorithms, the remainder of the text assumes data to be nominal and in a rectangular format. Therefore, some definitions related to frequent item sets need to be reformulated in terms of transactional data structures.

**Definition 1: Attribute-Value Pair $x^i$**

An attribute-value pair $x^i = < x_i , j >$ represents a nominal variable $x_i$ with value $j$ from possible values $j \in \{\emptyset, 1,2, \dots, J^i\}$. Value $\emptyset$ denotes a missing value. Furthermore, $x^i = x^{i'} \Leftrightarrow x_i = x_i' \neq \emptyset$.

**Definition 2: Record $X$**

A record $X = \{x^1, \dots, x^i, \dots, x^J\}$ represents a set of $J$ attribute value pairs.

**Definition 3: Item set *I***

An item set $I = \{x^1, \dots, x^i, \dots, x^J\}$ represents a set of $J$ attribute value pairs

From these definitions, it follows that there is no structural difference between a record and an item set. However, the term record is mainly used for the original data, while item sets are mainly subsets of the original data. Furthermore, while the value $\emptyset$ in a record represents a missing value, this value is interpreted as 'not present' rather than 'missing' when present in an item set.

**Definition 4: Record *X* contains Itemset *I* ($X \supseteq I$)**

Record $X = \{x^1, \dots, x^i, \dots, x^J\}$ contains item set $I = \{x^{1'}, \dots, x^{i'}, \dots, x^{J'}\}$ iff $x^i = x^{i'}\ \forall x^{i'} \in \{x^{i'} \in I \mid x_i' \neq \emptyset\}$

**Definition 5: Support of Itemset *I***

$Sup(I) = |\{X | X \supseteq I\}|$

**Definition 6: Frequent Itemset *I***

$I_z$ is a frequent item set iff $\mathrm{Sup}(I) > \alpha$, where $\alpha$ is a predefined threshold

**Definition 7: $I \cap I'$**

$I \cap I' = I''$ such that $\forall x^{i''} \in I''$ it holds that

$$x_i'' = \begin{cases} x_i'' = x_i & if\ x_i = x_i' \\ x_i'' = \emptyset & if\ x_i \neq x_i' \end{cases}$$

## III. Association Rules

Association rules are nothing less than rules built from two exclusively disjunctive item sets. One item set acts as the rule's body, which is also called the antecedent, while the second item set is the rule's head, which is also called the consequent.

**Definition 8: Association Rule *R***

An association rule $R: I^b \Rightarrow I^h$ consists of a body item set $I^b$ and a head item set $I^h$ such that $x_i = \emptyset, \forall x^i \in I^b \cap I^h$

Traditionally, two measures are generated to evaluate the quality of an association rule, i.e. the support and the confidence. The support of an association rule measures to what extent the data supports the pattern described by the rule. A low support can have different explanations. Typically, this implies that the rule only exists in a small local part of the data space. If the support is extremely low, it could also imply that the rule doesn't really exist and is rather some kind of measurement error.

**Definition 9: Support of Association Rule $R$: $I^b \Rightarrow I^h$**

$Sup(R) = Sup(I^b \cup I^h)$

The union of two disjunctive item sets is defined as follows:

**Definition 10: Union of two disjunct itemsets $I \cup I'$**

$\forall x^i \in I'' = I \cap I' : x_i = \emptyset \Rightarrow I \cup I' = I''$ such that $\forall x^{i''} \in I''$ it holds that

$$x_i'' = \begin{cases} x_i'' = x_i & if\ x_i' = \emptyset \\ x_i'' = x_i' & if\ x_i = \emptyset \end{cases}$$

While the support of an association rule measures how well the rule is supported by the data, the confidence of the rule reveals how accurate the rule's body predicts the rule's head. The confidence of a rule measures how many percent of the records that contain the body, also contain the head and is defined as follows:

**Definition 11: Confidence of Association Rule $R$: $I^b \Rightarrow I^h$**

$Conf(R) = Sup(R)/Sup(I^b)$

For classification purposes, not all association rules are interesting. Only those rules where the rule's head contains the class attribute and no other items can be used to classify new instances. Such rules are called classification association rules or CARs.

**Definition 12: Classification Association Rule $R^C$**

A classification association rule $R^C: I^b \Rightarrow I^h$ is an association rule such that $\forall x_i \in I^h \setminus \{x_y\} : x_i = \emptyset$ and $x_y \neq \emptyset$ where $x_y$ represents the class attribute of the classification problem.

Once the CARs are learned from the data, they are used to classify new data instances. In order to do so, a distinction must be made between CARs that match a record and CARs that cover a record. A CAR covers a record if the record contains the body of the CAR, while a CAR matches a record if the record contains both the body and the head of the CAR

**Definition 13: $R^C$ covers record $X$**

CAR $R^C: I^b \Rightarrow I^h$ covers record $X$ iff $X \supseteq I^b$

**Definition 14: $R^C$ matches record $X$**

CAR $R^C: I^b \Rightarrow I^h$ covers record $X$ iff $X \supseteq I^b$ and $X \supseteq I^h$

Furthermore, the size of a rule is defined as the number of non-empty attribute-value pairs in the rule's body

**Definition 15: The size of $R^C$**

The size of $R^C$, denoted as $|R^C|$, is the number of attribute-value pairs $x^i$ for which $x_i \neq \emptyset$

Finally a CAR $R_1^C$ can be a generalization of another CAR $R_2^C$, which is defined as follows:

**Definition 16: $R_1^C$ generalizes $R_2^C$**

$R_1^C: I^{b1} \Rightarrow I^{h1}$ generalizes $R_2^C: I^{b2} \Rightarrow I^{h2}$ iff $I^{h1} = I^{h2}$, $I^{b2} \supseteq I^{b1}$ and $|R_1^C| < |R_2^C|$

## IV. Association Rule Classifiers

### A. General Structure

Over the years, various authors have introduced their own association rule classifier. Although each implementation differs, a common structure of three consecutive steps can be identified across most ARCs.

**Step 1: Learn Classification Association Rules.** Obviously, all ARCs must start by generating a set of CARs from a given training set. Various algorithms have been used, such as Apriori which is used in CBA, ARC-AC, ARC-BC and ACRI.

Other CAR generating algorithms are FPGrowth, which is used by CMAR, a CAR generating algorithm based on FOIL used by CPAR and a CAR generating algorithm based on the Framework of Morishita and Sese used by CorClass. Algorithms such as Apriori and FPGrowth are exact algorithms which provide the exhaustive set of association rules meeting specific support and confidence criteria. These CAR generating algorithms produce the same set set of CARs generated, but differ in terms of computational complexity. One exception is the FOIL-based CAR generating algorithm used in the CMAR implementation, which is a heuristic rather than an exact solution and only gives an approximation of the exhaustive set of CARs meeting specific criteria.

**Step 2: Prune the set of Classification Association Rules.** Once the CARs are generated from the training set, most ARCs apply some pruning strategy to reduce the set of CARs as this can become enormous. Various strategies to prune the set of CARs exist, which will be discussed in the next subsection. However, some ARCs, such as CorClass and ACRI do not apply a separate pruning step, although setting a minimum confidence and support when generating the CARs can be interpreted as a pre-pruning strategy.

**Step 3: Classifying new instances.** Once the (pruned) set of CARs is finalized, they can be used to classify new instances. Various strategies have been developed, which will be discussed in section IV.B.

### B. Pruning the Association Rules

As mentioned before, even if there is no separate post-pruning step in the ARC algorithm, all ARC algorithm apply some sort of pre-pruning by setting a support and/or confidence threshold. Only CARs passing these thresholds will be kept in the set of association rules. These pruning techniques are isolated pruning techniques as they evaluate each CAR individually, in isolation from the other CARs.

Another isolated pruning technique, which is applied by CBA, is **Pessimistic Error Pruning** (PEP). PEP uses the pessimistic error rate based pruning method in C4.5 [10]. It calculates the pessimistic error rate of the entire rule and compares it with the pessimistic error rate of the rule obtained by deleting one item from the rule's body. If the pessimistic error rate of the complete rule is higher than the pessimistic error rate of the trimmed rule, than the complete rule is pruned. The results in [1] show that pessimistic error pruning has a strong impact on the number of CARs generated.

Another isolated pruning technique is **Correlation Pruning** (CorP), which is applied in CMAR. This pruning technique calculates the correlation between the rule's body and the rule's head. If the correlation is not statistically significant, the rule is pruned.

There are also non-isolated pruning techniques which take multiple rules into account in order to decide whether or not to prune a specific rule. A well known non-isolated pruning technique is the **Data Coverage Pruning** technique (DCP), which is applied in CBA, ARC-AC, ARC-BC and CMAR. DCP consists of two steps. First, the rule set is ordered according to confidence, support and rule size. Rules with the highest confidence go first. In case of a tie, rules with the highest support take precedence. In case of tie in terms of confidence and support, the smaller the rule, i.e. the more general a rule is, the higher the ranking. Once the rule set is ordered, the rules are taken one by one from the ordered rule set and are added to the final rule set until every record in the data set is matched at least $\alpha$ times. For CBA, ARC-AC and ARC-BC this parameter $\alpha$ is fixed to 1, while in CMAR $\alpha$ is a parameter which needs to be set by the user.

**Confidence Pruning** (ConfP) is a second non-isolated pruning technique which is used by CMAR, ARC-AC, ARC-BC. ConfP prunes all rules which are generalized by another rule with a higher confidence level.

### C. Classifying new instances

Once the CARs are generated and pruned, the ARC needs to use all these pieces of local knowledge to classify new instances. Various approaches have been developed, which can be classified as order-based classification or non-order-based classification. With order-based classification, the rules in the final rule set need to be ordered according to a specific criterion and this ordering has an influence on the predicted class, while non-order-based classifiers do not need such ordering.

A first order-based classification scheme is the **Single Rule Classification**. This approach orders the rules and then uses the first rule which covers the new instance to make a prediction. The predicted class is the selected rule's head. This classification scheme is used by CBA, CorClass and ACRI. CBA and CorClass order the rules according to confidence, support and rule size in the same way the data coverage pruning technique does. ACRI on the other hand, provides the

user four different ordering criteria to select from, i.e. a cosine measure, the support, the confidence or the coverage. The latter is defined as the number of items the body of a rule has in common with the new instance divided by the size of the rule.

A second order-based classification scheme is the **Multiple Rule Classification**. This approach, which is used by CPAR, retains only those rules which cover the new instance. Next, the rules are grouped per class value and ordered according to a specific criterion. Finally, a combined measure is calculated for the best $Z$ rules, where $Z$ is a parameter which needs to be set by the user. With CPAR, the rank of each rule is determined by the expected accuracy of the rule, which is calculated by the Laplace accuracy. This is defined as follows where $k$ denotes the number of classes in the classification problem:

**Definition 17: Laplace accuracy**

The Laplace accuracy of a CAR $R^C: I^b \Rightarrow I^h$ is $Laplace(R^C) = [Sup(R^C) + 1]/[Sup(I^b) + k]$

The combined measure in CPAR is nothing more than the average expected accuracy.

Finally, some ARCs use a **non-order-based classification** scheme, such as CMAR, ARC-AC, ARC-BC and CorClass. This classification scheme selects the rules which cover the new instance, groups them per class value and calculates a combined measure per class value. This approach is almost identical to the multiple rule classification scheme, except for the ordering step. Since this non-order based classification uses all the rules, such ordering is not required. The CMAR algorithm uses a weighted $\chi^2$ measure as combined measure, while ARC-AC and ARC-BC calculate the sum of the rules' confidence levels. CorClass uses a weighted sum as combined measure and defines three different weights. The first option is to give all rules a weight equal to one, which results in a majority voting classification scheme. The second and the third option uses some kind of ranking of the rules, which turns it into an order-based classification scheme.

## V. EXCEPTIONS

While most ARCs follow the design of learning the CARs, pruning the rule set and classifying the new instances, there are some ARCs which are an exception to this rule. These exceptions are classifiers which follow a totally different scheme. Three examples of such exceptions are 2SARC1, 2SARC2 [8] and ARUBAS [9].

These three classifiers firstly generate CARs from a training set, which makes them ARCs. However, instead of using these CARs directly to classify new instances, as the other ARCs do, the CARs are used to transform the attribute space into a new feature space. Subsequently, traditional data mining techniques are applied in this new feature space to classify the new instances.

In case of 2SARC1, each instance is described by class features, while as for 2SARC2, each instance is described by rule features. Once the instances are transformed into this new feature space, a neural network is trained and used to classify new instances.

ARUBAS on the other hand, uses the learned association rules to transform all instances into pattern space. Each CAR is considered to reveal a frequent pattern in the data which is strongly connected to a specific class. In pattern space, each CAR represents a binary attribute and instances receive value 1 if the CAR covers the instance. Next, the similarity between a new instance and all instances of a specific class is calculated in pattern space. The authors believe that pattern space can be a stronger representation space as each attribute in pattern space is actually a frequent pattern in the original attribute space with strong predictive power. The instance will be assigned to the class with the highest similarity.

## VI. CONCLUSIONS

Association rule classifiers are a type of classifier which have been around for more than a decade. Recently, they received quite some attention from researchers who focus on building global models from local patterns [11], [12]. Over time, different ARC algorithms have been developed, resulting in a wide variety of association rule pruning techniques and CAR classification schemes. Despite the variety, it is clear that most ARC algorithms have a common three step structure, i.e. first learn the association rules, next prune the set of CARs and finally classify new instances with the CAR set. The benefit of this clear common structure is that different elements from different ARC algorithms can easily be combined. Evaluating the effect of the various pruning techniques and association rule classifiers and the interaction among them could be an interesting direction for future research.

At the same time, some authors came up with an innovative but different approach to use association rule for classification. Instead of using the association rules directly, the CARs are used to transform the original attribute space into a more powerful representation. However, this approach has not been studied as extensively as the more common three step approach and most likely provides some interesting new paths for future research.

## REFERENCES


[1] Liu, B., Hsu, W. and Y. Ma, "Integrating classification and association rule mining," ACM Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD '98),pp. 80-86 , 1998.

[2] Li, W., Han, J. and J. Pei, "CMAR: Accurate and efficient classification based on multiple class-association rules," Proc. of the Int. Conf. on Data Mining (ICDM'01), pp. 369-376, 2001.

[3] Antonie, M.-L. and O.R. Zaïane, "Text document categorization by term association," Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02), pp. 19-26, 2002.

[4] Yin, X. and J. Han, "CPAR: classification based on predictive association rules," Proc. of the SIAM Int. Conf. on Data Mining, pp. 369-376, 2003.

[5] Zimmermann, A. and L. De Raedt, "CorClass: correlated association rule mining for classification," Discovery Science, Springer Berlin / Heidelberg, pp. 60-72, 2004.

[6] Rak, R., Stach, W., Zaïane, O.R. and M.-L. Antonie, "Considering re-occurring features in associative classifiers," Advances in Knowledge Discovery and Data Mining, Springer Berlin / Heidelberg, pp. 240-248, 2005.

[7] Antonie, M.-L., Zaïane, O.R. and R.C. Holte, "Learning to Use a Learned Model: A Two-Stage Approach to Classification," Proc. of the sixth Int. Conf. on Data Mining (ICDM 2006), pp. 33-42, 2006.

[8] Depaire, B., Vanhoof, K. and G. Wets, "ARUBAS: an association rule based similarity framework for associative classifiers," Data Mining Workshops, Int. Conf. on Data mining (ICDMW 2008), pp. 692-699, 2008.

[9] Agrawal, R., Imieliński, T. and A. Swami, "Mining association rules between sets of items in large databases," Proc. of the 1993 ACM SIGMOD Int. Conf. on Management of data, pp. 207-216, 1993.

[10] J.R. Quinlan, "C4.5: program for machine learning," Morgan Kaufman. 1992.