ARUBAS: An Association Rule Based Similarity Framework for Associative Classifiers

Benoît Depaire
Data Analysis and Modeling,
Hasselt University,
3590 Diepenbeek, Belgium
benoit.depaire@uhasselt.be

Koen Vanhoof
Data Analysis and Modeling,
Hasselt University,
3590 Diepenbeek, Belgium
koen.vanhoof@uhasselt.be

Geert Wets
Data Analysis and Modeling,
Hasselt University,
3590 Diepenbeek, Belgium
geert.wets@uhasselt.be

Abstract

This article introduces ARUBAS, a new framework to build associative classifiers. In contrast with many existing associative classifiers, it uses class association rules to transform the feature space and uses instance-based reasoning to classify new instances. The framework allows the researcher to use any association rule mining algorithm to produce the class association rules. Every aspect of the framework is extensively introduced and discussed and five different fitness measures used for classification purposes are defined. The empirical results determine which fitness measure is the best and compares the framework with other classifiers. These results show that the ARUBAS framework is able to produce associative classifiers which are competitive with other classification techniques. More specifically, with ARUBAS-Scheffer-ϕ_5 we have introduced a parameter-free algorithm which is competitive with classification techniques such as C4.5, RIPPER and CBA.

1. Introduction

Within the data mining community, research on classification techniques has a long and fruitful history. But classification techniques based on association rules, which are also called associative classifiers (AC), are relatively new. The first associative classifier CBA was introduced by Liu et al. in 1998 [8]. During the last decade, various other associative classifiers were introduced, such as e.g. CMAR [7], ARC-AC and ARC-BC [2], CPAR [14], CorClass [15], ACRI [10] and a two stage approach by Antonie et al. [3].

Almost every AC contains two major data mining steps, an association rule (AR) mining stage and a classification stage which uses the mined rules from the first stage directly. During the first stage, a set of class association rules (CARs) is learned from the training instances. A class association rule is a very specific type of association rule where the consequent is a class value. In the literature on AC, various AR mining techniques have been used. Most approaches such as CBA [8], ARC-AC and ARC-BC [2] and ACRI [10] use an adapted version of the Apriori algorithm, while CMAR [7] uses the FPTRee algorithm, CPAR [14] uses a greedy FOIL algorithm to learn the rules and CorClass [15] is based on a technique developed by Morishita and Sese [9]. Most of these AC also prune the set of CARs during or after the AR mining stage. For example, Apriori based techniques set a minimum support to reduce the set of CARs, while other techniques such as CBA and CMAR remove rules which do not cover at least one training instance not considered by a higher ranked rule.

For classifying new instances with the set of mined CARs, three different approaches can be discerned, i.e. using a single rule, using a subset of rules or using all rules. An example of an AC which uses a single rule is CBA, which classifies an instance by using the single best rule covering the instance. CPAR is an example of an AC which uses a subset of rules. It first gathers all rules covering the new instance and selects the best n rules per class. Next, it calculates the average Laplace accuracy per class and predicts the class with the highest average accuracy. CMAR, ARC-AC and ARC-BC use a similar approach as CPAR but use all rules covering a class to calculate an average score per class. They also use different score statistics than CPAR, i.e. the weighted χ^2 measure [7] or the sum of confidences [2]. Finally, it's worthwhile mentioning that many current AC are dependent on some sort of rule ordering mechanism. Some need this ordering for selecting the single or n best CARs (e.g. CBA, CPAR) while others need it for pruning the original set of CARs (e.g. CMAR). The most common ranking mechanism is the database coverage which is based on the support, confidence and cardinality of the rules, but other techniques such as the cosine measure [10] also exist. For a more detailed discussion and review of these and other associative classifiers, the reader is referred to [12].

Recently, a new associative classification method which differs from the two stage approach of the previously discussed techniques was introduced by Antonie et al. [3]. The first stage of learning the association rules from the training data remains, but instead of using the CARs to classify the new instances, the rules are used to transform the feature space. Next, a neural network in this new feature space is used as a classifier.

In this article, we want to present a new framework for associative classifiers which shows some resemblance with [3]. Similar to their approach, we also use the learned CARs to transform the feature space. However, in our approach the classification is done through case-based reasoning. Our approach is also rather a framework than a specific classifier, since any association rule algorithm can be used for learning the CARs. Therefore, the main focus of this article is the introduction and discussion of this novel approach. First, in the next section, we shall describe the separate steps of the framework. Then, in section 3, we shall run some experiments to show the empirical potential of our framework. The fourth section discusses the current limitations of the framework and gives some useful directions for future research while the fifth section concludes the paper.

2. ARUBAS framework

2.1. Association rule mining

First, we shall discuss some concepts regarding association rule mining which will act as a brief introduction to the reader unfamiliar with this particular field of data mining. In this article, we shall mainly follow the notation used by Thabtah [12].

A data set with training instances is denoted as T and has m distinct attributes $A_1, A_2, \cdots, A_m$ and C is a list of classes. The number of records or instances in T is denoted $|T|$. Attributes can be categorical or continuous in which case they must be discretized first. Thus, an instance X_t can be described as a combination of attribute names A_i and values a_{ij}, plus a class value denoted by c_j. The combination of an attribute name A_i and a value a_i, denoted $\langle(A_i, a_i)\rangle$, is called an item or a literal. An itemset is a set of k items, denoted $\langle(A_{i1}, a_{i1}), \cdots, (A_{ik}, a_{ik})\rangle$. A CAR R consists of an itemset and a class value and is represented in the form $R : \langle(A_{i1}, a_{i1}), \cdots, (A_{ik}, a_{ik})\rangle \Rightarrow c$.

The purpose of association rule mining is to discover interesting rules or associations within the given data set. The interestingness of an association rule can be expressed through several statistics among which support and confidence are the most common. The support $sup(R)$ of an association rule is the probability that the rule occurs within the data. A rule $R : \langle(A_{i1}, a_{i1}), \cdots, (A_{ik}, a_{ik})\rangle \Rightarrow c$ has support s if $s\%$ of the instances in T contain $\langle(A_{i1}, a_{i1}), \cdots, (A_{ik}, a_{ik})\rangle$ and have class c. In other words it expresses how much evidence there is that this rule is real and not some sampling artifact.

The confidence $conf(R)$ of a rule $R : \langle(A_{i1}, a_{i1}), \cdots, (A_{ik}, a_{ik})\rangle \Rightarrow c$ expresses which percentage of the instances containing $\langle(A_{i1}, a_{i1}), \cdots, (A_{ik}, a_{ik})\rangle$ have class c. In other words, the confidence of the rule is the conditional probability that the consequent is true under the condition of the antecedent.

Various association rule mining algorithms have been developed, such as e.g. the original Apriori algorithm [1], which has been extensively studied and modified [5], or the FP-growth method [6]. For AC, most of these algorithms need to be slightly modified so only class values are allowed in the rule's consequent.

2.2. Pattern space

The main idea behind the ARUBAS framework, is that we transform the original feature space into a more powerful feature space. The original feature space is called the attribute space, where each record $X_t = (a_1, \cdots, a_m, c)$ is coded as a set of attribute values and a class value. In attribute space, each dimension consists of a single attribute. In the new feature space, which we will call pattern space, each dimension will consist of a combination of attributes, also called a pattern, which is denoted as $P_p = \langle(A_{i1}, a_{i1}), \cdots, (A_{ik}, a_{ik})\rangle$. However, we are only interested in combinations of attributes (or patterns) which are strongly associated with a single class value, since only these will make the feature space more powerful.

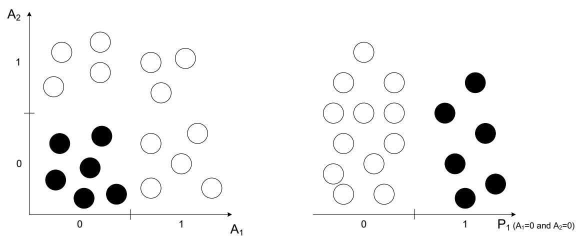
Finding patterns which are strongly related with a single class value is the goal of every class association rule mining technique. Therefore, the first step in the ARUBAS framework is to use any CAR mining technique to find a set of CARs, which is used to transform the feature space. The antecedent of each CAR, which represents an itemset, will become a pattern P_p and hence a dimension in the new feature space. The value of an instance X_t for a pattern P_p is either 1 if the instance contains the pattern or 0 if it doesn't. Therefore, if p denotes the number of CARs found during the CAR mining technique, the transformation can be expressed as follows:

$$\Theta : X_t = (A_1, \cdots, A_m, C) \mapsto X_t = (P_1, \cdots, P_p, C) \tag{1}$$

Let us illustrate how this transformation can result into a more powerful feature space. Assume that the original feature space has two attributes A_1, A_2 and a class C. The attributes are continuous and thus discretization is needed.

Assume discretization resulted in a single cut-point for both attributes, making them both binary. The discretized attributes are denoted as $A_1^* = \{0, 1\}$ and $A_2^* = \{0, 1\}$. The class $C = \{0, 1\}$ is already binary.

Figure 1. Feature space transformation.



The left side of Figure 1 shows the original feature space. It shows that the instances with class value 1 (black) are in the lower left corner of the domain. CAR mining algorithms would probably discover the following rule: $R : \langle (A_1, 0), (A_2, 0)\rangle \Rightarrow 1$, which has a support of 33% and a confidence of 100%. Consequently, the antecedent of this rule will become the pattern $P_1 = \langle (A_1, 0), (A_2, 0)\rangle$. Assuming that this is the only CAR found, the new feature space will only have two dimensions, the pattern and the class. The right side of Figure 1 shows the new feature space. As we can see, the instances can now be separated by a single plane, while the original space would need two separating planes, making the classification problem easier. Although this example is rather academic and a simplified version of reality, it distinctively illustrates the potential benefits of transforming the instances from attribute space to pattern space.

The main reason why pattern space is more powerful is because it expresses the data in terms of attribute combinations (patterns) which are strongly related with a specific class value. However, pattern space can also become more complex than the original attribute space, because a few attributes with limited values can already produce many patterns. Therefore, it's important to limit the number of patterns to the most important ones in order to prevent this combinatorial explosion. This can be achieved e.g. by setting appropriate thresholds for support and confidence.

Finally, it should be noted that not every location in pattern space is feasible. For example, given $P_1 = \langle (A_1, 0)\rangle$ and $P_2 = \langle (A_1, 1)\rangle$, the following instance $X_t = (P_1 = 1, P_2 = 1, \cdots)$ cannot exist since no instance can contain both patterns at the same time.

2.3. Instance similarity

Once the CARs are mined and the transformation is performed, we rely on instance-based reasoning to classify new instances. This implies that new instances must be compared with training instances for which we already know the class value. This comparison will be based on some notions of similarity. To measure the similarity between a new instance X_N and a known training instance X_T, we focus on the patterns contained by both instances and how many patterns both instances have in common. However, we only focus on those patterns coming from the CARs which predicted the class value of the training instance X_T. This makes sense since patterns are attribute combinations which are associated to a specific class value and if that class value differs from the class value of X_T, verifying if this pattern is shared with X_N doesn't add evidence that X_N should have the same class value as X_T. Therefore, to distinguish them from regular patterns, we shall denote patterns which were derived from rules predicting the class value of the training instance X_T as P^T.

The similarity between a training instance X_T and a new instance X_N is measured as follows:

$$s(X_T, X_N) = \frac{\sum_{i=1}^{p} P_{Ti}^T P_{Ni}^T}{max\left[\sum_{i=1}^{p} P_{Ti}^T, \sum_{i=1}^{p} P_{Ni}^T\right]}. \qquad (2)$$

This measure expresses which percentage of patterns the instance containing the fewest patterns shares with the instance containing the most patterns. For example, if $X_T = (1, 1, 1, 1, c_1)$ and $X_N = (1, 0, 0, 0, ?)$, then X_N shares 1 out of 4 patterns contained by X_T and therefore $s(X_T, X_N) = 0.25$. Furthermore, this measure has the following interesting properties:

$$\forall i \in 0, \cdots, p : P_{Ti}^T = P_{Ni}^T \Leftrightarrow s(X_T, X_N) = 1, \qquad (3)$$

$$\forall i \in 0, \cdots, p : P_{Ti}^T \neq P_{Ni}^T \Leftrightarrow s(X_T, X_N) = 0. \qquad (4)$$

However, this measure doesn't take into account if a new instance and a test instance contain many patterns or only a few. The data from table 1 illustrate how this can lead to misleading similarity values. Although training instance X_{T1} and new instance X_{N1} are identical, they only share one pattern. Training instance X_{T2} and new instance X_{N2} are not identical and will have a lower similarity than $s(X_{T1}, X_{N1})$. However, because they share more patterns, one could argue there is more evidence that both will share the same class value. Therefore, a weighting factor is introduced. This weighting factor calculates which percentage of all patterns related to the class value of X_T are contained by either X_T or X_N. The weighting factor can be expressed as follows, given that p^T denotes the number of patterns derived from rules predicting the class value of instance X_T:

$$\alpha_{TN} = \frac{\sum_{i=1}^{p} 1 - (1 - P_{Ti}^T)(1 - P_{Ni}^T)}{p^T} \qquad (5)$$

Table 1. Example Data

X_{T1}	(1,0,0,0,c_1)
X_{N1}	(1,0,0,0,c_1)
X_{T2}	(1,0,1,1,c_2)
X_{N2}	(1,1,1,1,c_2)

Based on the similarity between a new instance and a training instance and their weighting factor, one can calculate the similarity between the new instance and a specific class. This is done by taking the weighted average between the new instance X_N and each training instance X_t with class value c. Let $|T_c|$ denote the number of training instances which have class value c, then the similarity between a new instance X_N and a class c can be expressed as

$$S_c(X_N) = \frac{\sum_{t=1}^{|T_c|} \alpha_{tN} s(X_t, X_N)}{|T_c|} \tag{6}$$

Finally, we also introduce the concept of class cohesion. This is the average weighted similarity between each pair of training instances having class value c. It is an indication how similar these training cases are themselves. It is measured as follows:

$$Coh_c = \frac{2}{|T_c|(|T_c|-1)} \sum_{t_1=1}^{|T_c|-1} \sum_{t_2=1}^{|T_c|} \alpha_{t_1 t_2} s(X_{t_1}, X_{t_2}) \tag{7}$$

2.4. Fitness measures

Now that we have introduced some concepts of similarity and class cohesion, we can define several fitness measures which will be used to classify new instances. In fact, each fitness measure expresses how well a new instance fits a specific class. In total we have defined 5 different fitness measures. The first two measures assess how the cohesion of a class changes when the new instance is added to the class. Intuitively, one would expect that the cohesion will increase if the instance is added to the correct class. The cohesion of a class after the new instance is added is denoted as Coh'_c. The first fitness measure calculates the absolute change in class cohesion:

$$\phi_1(c) = Coh'_c - Coh_c. \tag{8}$$

The second fitness measure calculates the relative change in class cohesion:

$$\phi_2(c) = \frac{Coh'_c - Coh_c}{Coh_c}. \tag{9}$$

The third fitness measure is based on three assumptions. Firstly, the measure should increase if the class size increases, since this class contains more evidence (= training instances with known class value). Secondly, the measure should increase if the class has a higher cohesion, because this indicates a stronger and more reliable structure within the class. Thirdly, the fitness measure should increase if the similarity between the new instance and the class increases. This measure is expressed as:

$$\phi_3(c) = log(|T_c|) S_c(X_N) Coh(c). \tag{10}$$

The fourth fitness measure is similar to the previous, but drops the first assumption. It can be expressed as:

$$\phi_4(c) = S_c(X_N) Coh(c). \tag{11}$$

The fifth fitness measure is similar to the third measure, but drops the first and second assumption. It can be expressed as:

$$\phi_5(c) = S_c(X_N). \tag{12}$$

2.5. Framework summary

The main idea behind the association rule based similarity framework is that classification is based on similarity between a new instance and an entire class. However, this similarity is not measured in the original attribute space, but in the pattern space, which is constructed by means of CARs. Algorithm 1 illustrates how all the pieces fit together.

Algorithm 1 The ARUBAS framework

```
1: Learn CARs from the training data by means of a association rule mining algorithm
2: Use the antecedents of the CARs as patterns to transform the training data to pattern space
3: for each new instance $X_N$ do
4:     for each class $C$ do
5:         Calculate the fitness measure
6:     end for
7:     Assign $X_N$ to the class with the highest fitness value
8:     In case of a tie, predict the majority class
9:     In case of multiple majority classes, select a majority class at random
10: end for
```

3. Empirical results

In this section, we would like to perform two experiments. First, we would like to compare the five fitness measures in terms of classifier accuracy. Secondly, we would like to assess the accuracy of our framework against other well-known classifiers. Since this article focusses on the introduction and conceptual discussion of this framework, the goal of the second experiment is to verify if the framework is competitive with existing techniques.

For the experiments in this paper, we used Scheffer's association rule mining algorithm [11]. Scheffer's algorithm is based on the idea that during association rule mining, a larger support has to be traded against a higher confidence. Furthermore, when a rule has a large support, one can be much more certain that the observed confidence is close to the true accuracy of the rule. In a Bayesian framework, Scheffer translates confidence and support into an expected accuracy on future data. One of the benefits of his algorithm is that the researcher does not need to set support or confidence tresholds. One only needs to determine the number of rules. The algorithm dynamically sets the expected accuracy threshold during the mining process and also prunes redundant rules.

Because the ARUBAS framework doesn't have any parameters of its own, combining it with Scheffer's AR mining technique provides an AC with only one parameter, i.e. the number of CARs to learn. We will call this the ARUBAS-Scheffer AC.

Own experiments have shown that for the ARUBAS-Scheffer AC, the evolution of the accuracy when changing the number of CARs on a separate test set is parallel to the evolution of the accuracy on the original training set. In other words, the evolution of the accuracy on the training set for different number of CARs gives a reliable indication of how many CARs need to be mined to find the maximum accuracy on the separate test set. The pseudo-code of the ARUBAS-Scheffer AC is provided by Algorithm 2. As becomes clear from this code, the ARUBAS-Scheffer AC is a parameter-free classifier.

Both experiments will compare several classifiers and try to find significant differences between them in terms of accuracy. In this article, we follow the methodology to compare multiple classifiers suggested by [4]. First, we will calculate unbiased estimates of the classifiers' accuracies by means of a ten-fold cross validation for 23 UCI datasets[1]. Next, we use the Friedman test to detect statistically significant differences between the classifiers in terms of average accuracy. The Friedman test is a non-parametric equivalent of the repeated-measures ANOVA test, but is based on the ranking of the algorithms on each data set instead of the true accuracy estimates. In his paper, Demšar discusses several reasons why the ANOVA test is unappropriate for comparing multiple classifiers. If according to the Friedman test, differences between the classifiers exist, we use the Nemenyi test to compare each classifier with all other classifiers. The Nemenyi test controls for family-wise error in multiple hypothesis testing and is similar to the Tukey test for ANOVA. The results of the Nemenyi test is shown by means of critical difference diagrams.

[1]These are: balance scale, car, cmc, credit, australian, crx, echocardiogram, ecoli, glass, haberman, hayes-roth, heart, iris, lenses, mammographic masses, monk1, monk2, monk3, pima-indians diabetes, post-operative, tae, tic-tac-toe, wine, yeast.

Algorithm 2 The ARUBAS-Scheffer AC

```
1: numCars = optNumCars =1000
2: maxAccuracy = 0
3: repeat
4:     Learn numCars CARs from the training data by means of Scheffer's association rule mining algorithm
5:     Execute the ARUBAS framework and evaluate the current accuracy on the training data
6:     if current accuracy $>$ maxAccuracy then
7:         maxAccuracy = current accuracy
8:         optNumCars = numCars
9:     end if
10:    numCars = numCars - 10
11: until numCars = 0
12: Learn optNumCars Cars from the training data by means of Scheffer's association rule mining algorithm
13: Execute the ARUBAS framework and evaluate the accuracy on the test data
```

Table 2. Average rank on 23 UCI data sets for ARUBAS-Scheffer ACs with different fitness measures.

	ϕ_1	ϕ_2	ϕ_3	ϕ_4	ϕ_5
Average Rank	3.70	3.11	3.13	3.04	2.02

3.1. Experiment 1: fitness measure comparison

The first experiment compares the accuracies on 23 UCI data sets of 5 different ARUBAS-Scheffer associative classifiers, each one using a different fitness measure. Table 2, shows the average rank of each classifier over the 23 UCI data sets. These results show that the ARUBAS-Scheffer AC with the fifth fitness measure, was the second best classifier on average. The other classifiers were third or fourth on average. The Friedman test provide a χ_F^2 value of 13.54 with 4 degrees of freedom which has a p-value smaller than 0.01. This test indicates that there are statistically significant differences in accuracy among these five classifiers.

Figure 2 shows the results of the Nemenyi test. Groups of classifiers that are not significantly different at $p = 0.1$ are connected. According to the Nemenyi test, the ARUBAS-Scheffer AC which uses the fifth fitness measure, i.e. $\phi_5(C) = S_C(X_N)$, performs significantly better than the other classifiers. Therefore, we shall continue with this particular fitness measure for the next experiment and we will denote this classifier as the ARUBAS-Scheffer-ϕ_5 AC. These results also illustrate that similarity should be measured only in terms of weighted average similarity between a new instance and all known test instances. Remarkably, the class cohesion or class size should not be taken into account.

3.2. Experiment 2: ARUBAS classifier versus other classifiers

In our second experiment we want to verify how competitive our ARUBAS framework is against other existing classifiers. We compared the ARUBAS-Scheffer-ϕ_5 classifier with the C4.5, Ripper, 1R and CBA algorithms. For the C4.5 and Ripper algorithm, we built two models, i.e. a pruned and an unpruned version. The ARUBAS-Scheffer-ϕ_5, C4.5, Ripper and 1R models were built with the WEKA software [13] and the CBA model was built with the DM-II CBA software (version 2.1) [8]. All models were built with default values for the various parameters. Furthermore, all models used ten-fold CV for an unbiased accuracy estimate. All models except CBA used the same way to split the data during the 10-CV stage. For CBA this was not possible because the model was built with different software. However, because 10-CV provides an unbiased estimate of the accuracy, a sound comparison of the models remains possible.

Table 3 shows the average rank of each classifier over the 23 UCI data sets. The Friedman test has a χ^2_F value of 24.89 with 6 degrees of freedom and a p-value smaller than 0.01. This indicates that there are statistically significant differences in accuracy among these 7 classifiers. More information is given in Figure 3, which shows the results of the Nemenyi test. All groups of classifiers that are not significantly different at $p = 0.1$ are connected. From these results we see that our ARUBAS-Scheffer-ϕ_5 classifier is competitive with C4.5, RIPPER and CBA and outperforms 1R, which represents the most simple and naive classifier. However, note that in contrast with the other classifiers, ARUBAS-Scheffer-ϕ_5 is a parameter free classifier. In terms of average ranking it shares a second position with the pruned version of RIPPER. Only the pruned version of C4.5 has a higher ranking. However, it should be noted that these differences in average ranking are not statistically significant at 0.1.

4. Remarks and directions for future research

The results of the empirical analysis have shown that the ARUBAS framework can provide associative classifiers which are competitive with other existing classifiers. However, the current framework still contains some limitations which need attention in the future. Firstly, the similarity between a test instance and a training instance gives the same weight to each pattern (i.e. dimension). However, if one considers that a pattern is derived from an association rule with a specific support and confidence, the similarity measure might benefit from some kind of pattern weight. Therefore, future research should focus on how the quality measures of association rules can be translated into some kind of pattern weight.

Secondly, the ARUBAS framework itself doesn't have a pruning step. The only pruning is done implicitly by the AR mining algorithm in the first step. In this article, we used Scheffer's AR mining algorithm which automatically prunes redundant rules and selects the n best rules based on predicted accuracy. However, the framework might benefit from a separate pruning stage which is independent from the AR mining algorithm used. Not only should future research investigate the pruning of CARs, it should also analyze the potential of pruning training instances. Since our framework is based on some kind of fitness between new instances and training instances, removing noisy training instances will most likely improve the accuracy.

Furthermore, future research is certainly needed to evaluate the impact of the AR mining algorithm on the accuracy of the framework. It could be interesting to study if there are AR mining algorithms which provide significantly better sets of CARs for our framework or whether the framework is insensitive to the selected algorithm.

Finally, from a methodological point of view, the second analysis, i.e. the comparison between ARUBAS-Scheffer-ϕ_5 and the other classifiers, should have been performed on a different set of data sets or should have been performed simultaneously with the first analysis. However, due to practical reasons we separated both analysis. This decision has no impact on each analysis separately and their respective conclusions. However, the exact level of significance, albeit from the most pessimistic point of view, that the conclusions of both analysis are true is not 10% but 19%.

5. Conclusions

In this article we introduced a new framework for building associative classifiers. The main idea is to use association rules to transform the feature space and to perform instance-based reasoning to classify new instances.

We have defined 5 different fitness measures between a new instance and the training instances with a specific class. The empirical results of the first experiment showed that one of those fitness measure, i.e. $\phi_5(c) = S_c(X_N)$, significantly leads to better results. We therefore suggest to use this fitness measure. Remarkably, the class cohesion or

Figure 2. Comparison of 5 ARUBAS-Scheffer ACs (Nemenyi test).

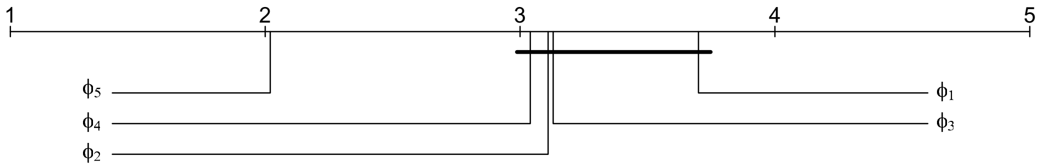
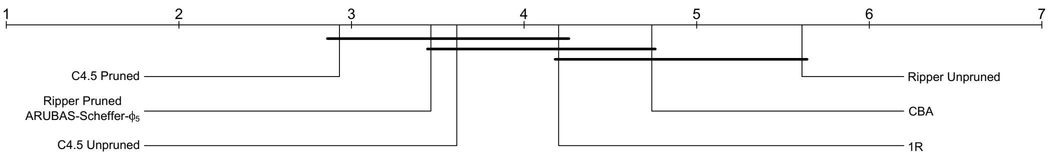


Table 3. Average rank on 23 UCI data sets for ARUBAS-Scheffer-ϕ_5 and 6 other classifiers.

	C4.5 pr.	Ripper pr.	ARUBAS	C4.5 unpr.	Ripper unpr.	CBA	1R
Avg Rank	2.93	3.46	3.46	3.61	4.20	4.74	5.61

Figure 3. Comparison ARUBAS-Scheffer-ϕ_5 AC and 6 other classifiers (Nemenyi test).



class size should not be taken into account.

The second empirical experiment evaluated our ARUBAS framework against other classifiers and showed that it is capable of producing associative classifiers which are competitive with other classifiers. More specifically, with ARUBAS-Scheffer-ϕ_5 we introduced a parameter-free algorithm which is competitive with classification techniques such as C4.5, RIPPER and CBA. These results confirm our belief that this new approach for building associative classifiers possesses enough potential for future research. We hope that this article and the reported results will provide a good starting point for researchers who wish to study this framework.

References

[1] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *SIGMOD '93: Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, pages 207–216, New York, NY, USA, 1993. ACM.

[2] M.-L. Antonie and O. R. Zaïane. Text document categorization by term association. In *ICDM '02: Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM'02)*, page 19, Washington, DC, USA, 2002. IEEE Computer Society.

[3] M.-L. Antonie, O. R. Zaïane, and R. C. Holte. Learning to use a learned model: A two-stage approach to classification. In *ICDM '06: Proceedings of the Sixth International Conference on Data Mining*, pages 33–42, Washington, DC, USA, 2006. IEEE Computer Society.

[4] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30, 2006.

[5] B. Goethals and M. Zaki, editors. *FIMI'03: Workshop on Frequent Itemset Mining Implementations*, volume 90 of *CEUR Workshop Proceedings series*, 2003.

[6] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29(2):1–12, 2000.

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

[8] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *ACM Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD '98)*, pages 80–86, New York City, NY, August 1998. (The CBA system can be downloaded from http://www.comp.nus.edu.sg/ dm2).

[9] S. Morishita and J. Sese. Transversing itemset lattices with statistical metric pruning. In *PODS '00: Proceedings of the nineteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 226–236, New York, NY, USA, 2000. ACM.

[10] R. Rak, W. Stach, O. R. Zaïane, and M.-L. Antonie. Considering re-occurring features in associative classifiers. In *Advances in Knowledge Discovery and Data Mining*, volume 3518/2005 of *Lecture Notes in Computer Science*, pages 240–248. Springer Berlin / Heidelberg, 2005.

[11] T. Scheffer. Finding association rules that trade support optimally against confidence. In *PKDD '01: Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 424–435, London, UK, 2001. Springer-Verlag.

[12] F. Thabtah. A review of associative classification mining. *Knowl. Eng. Rev.*, 22(1):37–65, 2007.

[13] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.

[14] 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 Francisco, CA: SIAM Press, 2003.

[15] A. Zimmermann and L. De Raedt. Corclass : Correlated association rule mining for classification. In *Discovery Science*, volume 3245/2004 of *Lecture Notes in Computer Science*, pages 60–72. Springer Berlin / Heidelberg, 2004.