International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:8, No:1, 2014

Review and Comparison of Associative Classification Data Mining Approaches

Suzan Wedyan

***Abstract*—**Associative classification (AC) is a data mining approach that combines association rule and classification to build classification models (classifiers). AC has attracted a significant attention from several researchers mainly because it derives accurate classifiers that contain simple yet effective rules. In the last decade, a number of associative classification algorithms have been proposed such as Classification based Association (CBA), Classification based on Multiple Association Rules (CMAR), Class based Associative Classification (CACA), and Classification based on Predicted Association Rule (CPAR). This paper surveys major AC algorithms and compares the steps and methods performed in each algorithm including: rule learning, rule sorting, rule pruning, classifier building, and class prediction.

***Keywords*—**Associative Classification, Classification, Data Mining, Learning, Rule Ranking, Rule Pruning, Prediction.

I. Introduction

DATA mining is the process of discovering and extracting hidden patterns from different data types in order to guide decision makers in making decisions. The discovery process can be an automatic or semi-automatic [1]. It involves the utilization of discovery algorithms and data analysis to produce particular patterns in the data under acceptable computational efficiency constraints. According to Fayyad et al. [2], data mining is a step of knowledge discovery in databases (KDD) steps that aims to extract knowledge from existing data. Other KDD main steps are data selection, data pre-processing, transformation, data mining, and evaluation. Data mining requires performing different tasks including classification, clustering, association rule discovery, pattern recognition, regression, etc. Often, the input and the goal determine the appropriate task.

There are two type of learning in data mining. These are: supervised and unsupervised. In supervised learning, the goal, also known as the class is predetermined and known in the training data set and the search for the output is guided by that goal. For example, in credit card scoring application, the goal is to whether the financial institution should grant a credit card or not to the client. Therefore, the search for the knowledge (rules) from the training data set is restricted to this goal where the rule consequent (right hand side) should be either "yes" or "no". On the other hand, when the case of learning involves training data set with no class attribute and the search for knowledge is not restricted to a specific attribute, this case is considered unsupervised learning. Association rule discovery task [3] is a common example of unsupervised learning where the aim is to discover the correlations among items in the shopping carts [1]. Association rule discovery and classification are closely related data mining tasks with the exception that association rule finds relationships among attribute values in a database whereas classification's goal is allocating class labels to unseen data known (test data set) as correctly as possible. Liu et al. [4] proposed a hybrid approach of association rules and classification mining, called associative classification (AC). In AC mining, the training phase is about searching for the hidden knowledge primarily using association rule algorithms and then a classification model (classifier) is constructed after sorting the knowledge in regards to certain criteria and pruning useless and redundant knowledge. Several research studies [4]-[9] showed that AC mining has advantages over other traditional classification approaches such as decision tree [10] covering and rule induction [11]. AC is often capable of building efficient and accurate classification systems, since it utilizes association rule discovery methods in the training phase [4] which finds all possible relationships among the attribute values in the training data set. This in turn leads to extract all hidden rules that can be missed by other classification algorithms. Moreover, the rules produced in AC are easy to understand and manually updated by end-user, unlike neural network and probabilistic approaches, which produce classification models that are hard to understand.

During the last decade, several AC algorithms have been proposed including: CBA [4], CMAR [12], ARC-AC [13], CPAR [5], CAAR [14], Negative-Rules [15], Live and Let Live (L^3) [16], MMAC [17], MCAR [18], CACA [19], ACCF [7], ACN [20], ACCR [9], and ADA [21]. This paper discusses the main concept of the associative classification, and also gives an example to demonstrate its main steps. Further, the different common associative classification algorithms are presented with their different methodologies in rule learning, ranking, pruning and prediction procedures.

The rest of this paper is organized as follows: Associative classification mining and the different data representation models are discussed in Section II. Section III surveys the different learning methods employed in AC. Different rule ranking methods in AC are surveyed in Section IV, Section V Presents the different strategies employed during building the classifier and the rules pruning procedures. Section VI reviews the different prediction methods in AC. Finally, conclusions and future work are discussed in Section VII.

S. Wedyan is with the Department of Computer Science, Amman ArabUniversity, Amman, Jordan (e-mail: susanwedyan@gmail.com).

II. Associative Classification Mining

Associative Classification (AC) is a common classification learning approach in data mining that adopts association rule discovery methods and classification to build the classification models. According to Thabtah et al. [18], AC is considered a special case of the association rule where the target attribute is considered in the rule's right hand side. For example, in a rule such as: *C* must be the target attribute, the input, the training data set *D* has a number of distinct attributes *A*, *A2*…, *An* and *C* is the target (class). Attributes could be nominal (has a finite set of possible values) or continuous (real or integer). Continuous attributes should be discretized using a discretization method in order to transform them into categorical attributes.

Associative classification depends mainly on two important thresholds called minimum support (*MinSupp*) and minimum confidence (*MinConf*). Minimum support represents the frequency of the attribute value and its associated class in the training data set from the size of that data set. Whereas minimum confidence represents the frequency of the attribute value and its related class in the training data set from the frequency of that attributes value in that training data. The frequent *ruleitem* is that the attribute value with its associated class that passes *MinSupp*, frequent 1- *ruleitem*, it is the frequent *ruleitem* that belongs to a single attribute, and frequent 2- *ruleitem* belongs to two attributes and so on.

In general, an AC algorithm operates in three main phases. In the first phase, AC searches for hidden correlations between the attribute values and the class attribute values in the training data set. Once all frequent *ruleitems* are found, the rules "Class Association Rule" (CARs) are generated from them in "if-then" format. In the second phase, ranking and pruning procedures start process, at this stage, CARs are ranked according to a certain number of parameters such as confidence and support values to ensure that rules with high confidence are given higher priority to be selected as part of the classifier. However, since the number of rules generated run into several thousands, and many of them are both redundant and not discriminative among the classes, rule pruning are needed to discard the contradicting and duplicating rules from the complete set of CARs. The output of the second phase is the set of CARs which represents the final classifier model. Lastly, the classification model is utilized to predict the class values on new unseen data set (test data).

A. AC Related Definitions and Example

Assume we have a training dataset D of |D| tuples and this dataset consists of a group of attributes, where each attribute has more than *a* value and *C* is the set of classes that can forecast the class of test cases. An AC algorithm can be formalized thorough the following definitions [22]:

Definition 1: An *AttributeValue* can be described as an attribute name A_i and its value a_i, denoted (A_i, a_i).

Definition 2: The j_{th}*row* or a *training case* in *D* can be described as a list of attribute values $(A_{j1}, a_{j1})\ldots (A_{jk}, a_{jk})$, plus a class denoted by c_j.

Definition 3: An *AttributeValueSet* can be described as a set of disjoint attribute values contained in a training case, denoted $< (A_{i1}, a_{i1})\ldots (A_{ik}, a_{ik})>$.

Definition 4: A *ruleitem r* is of the form <*antecedent*, *c*>, where *antecedent* is an *AttributeValueSet* and *c*ε*C*is a class.

Definition 5: The actual occurrence (*actoccr*) of a *ruleitem*r in*D* is the number of cases in *D* that match *r'santecedent*.

Definition 6: The support count (*suppcount*) of *ruleitem r* = <*antecedent*, *c*> is the number of cases in *D* that matches *r'santecedent*, and belongs to a class *c*.

Definition 7: A *ruleitem*r passes the *minsupp* threshold if, *suppcount*(*r*)/ |*D*| ≥ *minsupp*. Such a *ruleitem* is said to be a *frequent ruleitem*.

Definition 8: A *ruleitem*r passes the minimum confidence (*MinConf*) threshold if *SuppCount*(*r*) / *actoccr*(*r*) ≥ *MinConf*.

Definition 9: A rule is represented as: *Antecedent* $\rightarrow c$, where antecedent is an *Attribute Value Set* and the consequent is a *class*.

As mentioned before, the AC algorithm is obtained by performing three main steps, step one*ruleitems* discovery, step two, the classifier building from the discovered rules, finally once the obtained classifier is built, it predicts the class labels of all cases in test data set in order to evaluate the classifier through evaluation measures such as accuracy and error rate.

The following example shows the process of AC algorithm technique in the discovery rules and building the classifier. The training data set is given in Table I, where Att1 and Att2 represent the attributes and (Class) represents the class label. The *MinSupp* and *MinConf* thresholds are 20% and 60%, respectively. AC algorithm such as CBA [4]first discovers frequent *rulesitems* by performing a large number of passes on the training data set. The number of passes depends on the number of attributes.

TABLE I
Training Data set

Row Number	Att1	Att2	Class
1	a	d	C2
2	a	d	C1
3	a	e	C1
4	b	e	C1
5	b	e	C1
6	b	e	C2
7	b	f	C2
8	b	f	C2
9	a	d	C1
10	c	e	C2

Frequent *ruleitems* are represented in (Table II). The frequent *ruleitem* that passes a predefined threshold *MinConf* is treated as a candidate rule R. Once all frequent *ruleitems* are found, the rules (CARs) are extracted from them to build the classifier. A rule is inserted to the classifier if it covers certain number of cases in the training data set. After discovered all the rules to be a part of the final classifier, the classifier is evaluated against an independent data set (test data) to obtain its effectiveness. Within Table II, the rows in bold will be

discarded since they didn't pass the *MinConf*, the rest of the rows contain the rules that will form the classifier.

TABLE II
FREQUENT ITEMS DERIVED BY CBA FROM TABLE I

Frequent attribute value	Support	Confidence
<a>c1	30%	75%
<b>, c1	**20%**	**40%**
<b>, c2	30%	60%
<d>, c1	20%	66%
<e>, c2	**20%**	**40%**
<e>, c1	30%	60%
<f>, c2	20%	100%
<a, d>,c1	20%	66%
<b, e>, c1	20%	66%
<b, f>, c2	20%	100%

B. Data Representation in Associative Classification

1. Horizontal Data Representations

AC mining has adopted horizontal data representation format from association rule discovery. The first AC algorithm (CBA) [4] has employed the horizontal layout where the database consists of a group of transactions, where each transaction has an identifier followed by a list of items contained in that transaction. In other words, the training data set in the horizontal data format contains the number of rows in which each number's row has associated with a list of attribute values. Table I which was introduced earlier presents the horizontal data format.

2. Vertical Data Representations

Zaki et al. [23] introduced the concept of vertical data representation in association rule discovery. Vertical approach transforms the original training data set into a table that contains transactions identifiers (TIDs) which guide the locations of each attribute value in the training data set, and then it employs simple intersections among these TIDs to discover frequent values and produce the CARs. This saves training time and reduces I/O overhead [23], [24]. In AC mining algorithms such as MMAC [17], MCAR [18], have extended the vertical layout of the association rules discovery algorithm proposed by Zaki and Gouda [24]. The discovery of frequent *ruleitems* in the vertical data format is accomplished by simple intersections of disjoint attribute values locations. Vertical format employs the sets of IDs of any frequent items of size N-1 to discover the possible frequent items of size N during the rule discovery step. The result of an intersection among the TIDs of two items gives a new TID list, which has the locations where both items occur together in the input data. This new TID list can be used to calculate the support and confidence of the new item resulted from the intersection. For example, the following frequent 2-ruleitem form Table III: <(Att1,b),(Att2,f)>Their frequencies are represented by the following TIDs lists (4,5,6,7,8) and (7,8) respectively. The result of the intersection (7, 8) denotes the row numbers in the training data in which both *ruleitems* have appeared. Then by locating the row numbers of the class "C2" we simply find out that this candidate 2-ruleitem cardinality, i.e. 2, denotes the support count. If the support of this new item has sufficient support (greater than the *MinSupp*), then it considered as a potential rule.

3. Rough Set Theory Data Representation

Han et al. [25] proposed an AC mining algorithm that combines rough set theory [26], [27], association rule (Apriori) [28], and a covering heuristic named Michalski [29].The hybrid algorithm employs Rough set theory which is a rule discovery method that used to discard redundant attributes from training data sets, and to choose reducts of attribute-value pairs that can represent the complete training data set in a decision table context, a rough set algorithm named ROSETTA [26] is used to select the reducts of attribute-value pairs. A reduct of the decision table is the subset of the attribute values and the class attribute that represent the whole table.

III. LEARNING APPROACHES IN ASSOCIATIVE CLASSIFICATION

The first step in AC is to discover the set of CARs which can decompose it into two sub-steps, first the frequent *ruleitems* discovery, second the rule generation. Once all frequent *ruleitems* are extracted, associative classification algorithms produce the complete set of CARs which in turn is used to form the classifier. Several learning approaches have been used effectively in AC algorithms. This section surveyed these approaches in details.

A. Apriori Level-Wise Search

Agrawal and Srikant [28] proposed an algorithm called Apriori that discovers the frequent *itemsets* through multi scans over the training data set. They represent frequent *itemsets* as k-frequent items, where k is the current number of the scan (i.e., 1-frequent *itemsets* in the first scan, 2- frequent *itemsets* in the second scan). For each rule item, a support is computed through scan over the data. The rule item is added to the frequent items if its support passes a predefined support threshold *MinSupp*.

In Apriori, the discovery of frequent *itemsets* is implemented in a level-wise fashion, where in each iteration, a complete database scan is compulsory to generate the new candidate *itemsets* from frequent *itemsets* already found in the previous iteration. That is, frequent k-frequent items are found by means (k-1)-frequent items which are used to find (k+1)-frequent items. If an *itemset* is infrequent at any scan, it will be omitted because any *itemset* that will contain it will remain infrequent. The Apriori algorithm uses this property to prune candidate *itemsets* that have infrequent subsets before counting their support at any scan, and therefore, saving the time of computing new *itemset*.

TABLE III
VERTICAL DATA REPRESENTATION OF TABLE I

(Att1, a)	(Att1, b)	(Att1, c)	(Att2, d)	(Att2, e)	(Att2, f)	(Class, C1)	(Class, C2)
1	4	10	1	3	7	2	1
2	5		2	4	8	3	6
3	6		9	5		4	7
9	7			6		5	8
	8			10		9	10

Liu et al. [4] proposed an AC algorithm called CBA, which uses the Apriori algorithm in order to generate frequent *ruleitems*. CBA algorithm scans the whole training data to find frequent *ruleitems*at each level. A rule item that passes the minimum support threshold is added to the frequent *ruleitems*, otherwise it should be removed. In order to find all candidate *ruleitems* in a specific level, possible combinations of all frequent *ruleitems* of previous level are performed. After that, a scan over the training data is accomplished to separate the frequent *ruleitems* based on their supports. However, this approach requires repetitive scanning over the database, which increases the required processing time.

A major concern of the Apriori algorithm is the high computational time needed to find frequent *ruleitems* from all possible candidates at each level. Finding frequent *ruleitems* from all possible candidate *ruleitems* at each level is considered as a bottleneck of Apriori algorithm. In particular, if the data set is highly correlated and contains many attributes and, the *MinSupp* threshold is low, the potential number of candidate *ruleitems* at each level will be huge. While the support of each candidate *ruleitem* must be computed at each level in order to decide whether it's frequent or not, the algorithm may consume considerable CPU time and storage. Li et al. [12] conducted an experimental study which shows that the increased number of rules may cause serious problems such as over-fitting the training data set and misleading classification, which happens when multiple rules in the classifier cover a single test object with different class labels. However, any AC algorithm can achieve a good performance when the training data set (candidate *ruleitems*) is small.

Several AC algorithms use CBA learning style during the learning step and added some enhancements to overcome some of CBA's deficiencies that have been inherited from Apriori. For instance CBA (2) [30], was disseminated to overcome the problem of not generating CARs for minority class labels in the training data set (The class balancing issue). Further, CMAR [12] algorithm was developed to improve the searching for frequent *ruleitems* and to introduce a compact data structure to achieve this goal.

An AC with negative rules (ACN) was proposed by Kundu et al. [20]. ACN extends the Apriori algorithm to mine a relatively large set of negative association rules and then uses both positive and negative rules to build a classifier. The experimental results in [20] using UCI datasets [31] showed that ACN has better average classification accuracy compared to three classification methods in (CBA, CMAR, and C4.5)

A new CBA-like algorithm called (CARGBA) proposed by Kundu et al. [32], this algorithm is divided into two main sections; one is CARGBA Rule Generator which responsible for rules generation and the other is CARGBA classifier builder that constructs the classifier. The CARGBA rule generation is performed using two steps: (a) generates all rules using Apriori technique. And (b) generates all rules using Apriori technique but in a reverse manner. The experimental results on 6 data sets from UCI database [31] showed that CARGBA algorithm has better average classification accuracy in comparison with C4.5, CBA and CMAR algorithms.

Niu et al. [9] proposed ACCR, which extends the Apriori algorithm to generate classification rules, if the user sets the support threshold too high, many good quality rules will be ignored. On the other hand, if the support value is set too low, the problem of many redundant rules will be generated which consequently consumes more processing time and storage. The authors of [9] developed a metric measure of rules called "compactness" that stores *ruleitems* with low support but high confidence, which ensures that high quality rules are not deleted, The experimental results illustrated that the ACCR algorithm has better accuracy in comparison with CBA and CMAR algorithms against the UCI data sets [31].

Huang et al. [33] proposed a technique called Associative Classification Based on All-Confidence ACAC to handle rule generation. They added the all confidence notion to the Apriori algorithm. All confidence notion measures the degree of mutual association in an *itemset*. The ACAC work steps are similar to the steps used in the Apriori algorithm. The *ruleitems* whose all confidence and support passes predefined all confidence and minimum support threshold, respectively, are added to the k-candidate *ruleitems*. These candidate *ruleitems* will be used to extract the classifier rules. The candidate ruleitem that passes the *MinSupp* threshold is added to the classifier rules. In order to achieve the k+1-candidate *ruleitems*, all possible combination of k-candidate *ruleitems* are preceded that will be examined using all confidence and minimum support. The Whole process is repeated until k-candidate *ruleitems* becomes empty. Huang et al. [33] applied ACAC algorithm with and without all confidence technique on Mushroom dataset [31] using Ten-fold cross validation. The results showed that the runtime of the ACAC algorithm is about 16% of that without all-confidence and the number of rules generated by ACAC is about 36% of that without all-confidence while the accuracy of both is high.

A closed Frequent Itemset approach, which is a closely related approach to Apriori learning style that reduces the number of candidate *itemsets* and improves the searching for

frequent *itemsets* proposed by Li et al. [7] in an algorithm called ACCF. An *itemset* is said to be closed if none of its immediate supersets has similar support value as that of the *itemset*. ACCF employs the association rule discovery algorithm Charm [34] to reduce the size of the resulting classifiers in AC. Experimental results against eighteen different data sets from the UCI data repository [31]showed that ACCF produced slightly better classifier with respect to accuracy as well as smaller sized classifiers than CBA.

B. Vertical Mining Approach

Vertical mining concept utilizes simple intersection among item IDs to find the rules In AC mining. As mentioned earlier in Section II, datasets can be represented using one of the following techniques; vertical representation and horizontal representation. In horizontal representation, which is used by the Apriori algorithm [28], data is stored in a table, where each record represents a transaction that contains a set of attributes and each record has an identifier. As explained earlier, frequent items are discovered through multi-scans over data set. The support counts of the items are computed in each scan to check if the items are frequent or not, and therefore, increasing the computational time. To reduce the number of passes over the input database in Apriori, the Eclat algorithm has been presented in [23]. Eclat has introduced the concept of vertical database representation. In vertical representation, the dataset is stored in a table where each record represents an attribute. The values of each attribute are the transactions IDs that contains the intended item. In 2003, Zaki and Gouda [24] proposed a variation of the Eclat algorithm, called dEclat. The dEclat algorithm uses a newer vertical layout which only stores the differences in the transactions identifiers (TIDs) of a candidate *itemset* from its generating frequent *itemsets*. This considerably reduces the size of the memory. A vertical mining concept utilizes simple intersection among item IDs to find the rules. In a vertical mining, each item in the training data is converted into *ColumnId* and *RowId* representation, this representation holds information about items frequencies in the training data which requires only a single database scan, the data set is scanned only once to produce frequent 1-itemset. Then, the frequent k-*itemsets* are discovered by the intersection process between k-1-itemsets. Using this process, there is no need to scan the dataset in each iteration to obtain the supports of new candidate *itemsets*. Therefore, it reduces the need computational time [23].

In AC mining, algorithms such as MMAC [17], MCAR [18] and CACA [19] are modified transactions identifiers (TIDs) of a candidate *rulitems*, the Tid-list intersection learning approach used in association rule to handle classification. MCAR consists of two main phases: Rules generation and a classifier builder. In the first phase, the training data set is scanned once to discover frequent one-*ruleitems*, where items support with related class and class frequency are stored in an array vertically, and then MCAR combines *ruleitems* generated to produce candidate *ruleitems* involving more attributes. The intersection process between tow k-*itemsets* produce a new k+1-itemset, which holds the TIDs where both k-*itemsets* occur together in the training data set. Any rule item with support and confidence larger than *MinSupp* and *MinConf* respectively, is created as a candidate rule. In the second phase, rules created are used to build a classifier by considering their effectiveness on the training data set.

In [19], a new class based AC approach called CACA was proposed. CACA first scans the training data set and stores the data in vertical format, the attribute values are arranged in a descending order based on their counted frequency. CACA discard any attribute that fails to satisfy the *MinSupp* and for the attribute values that pass the *MinSupp*. CACA intersect them based on the class strategic to cut down the searching space of frequent pattern. The attribute in a class group that passes the *MinConf* threshold is inserted in the Ordered Rule Tree (OR-Tree) as a path from the root node and, its support, confidence and class are stored at the last node in the path.

C. FOIL Decision Tree Approach

Quinlan and Cameron-Jones [35] proposed the First Order Inductive Learner (FOIL) this learning strategy produces rules for each class cases in the training data. The learning rules in FOIL algorithm is a greedy fashion measured by FOIL-gain. The training data is dived into two subsets algorithm to generate the rules for each available class *C*, one subset contains positive cases that associated with class *C* and the other subset contains negative cases that associated with others class labels. FOIL starts with an empty rule, and then it computes the FOIL-gain for each attribute value belonging to *C* to select the attribute value with the largest FOIL-gain and adds it in the rule antecedent. Foil-gain measure is the key to success in FOIL learning strategy, which is about assessing the information gained for a particular rule after adding an attribute value to that rule.

A greedy AC algorithm called CPAR was proposed in [5]. CPAR adopts FOIL algorithm in generating the rules from data sets. It seeks for the best rule condition that brings the most gain among the available ones in the data set. Once the condition is identified, the weights of the positive examples associated with it will be deteriorated by a multiplying factor, and the process will be repeated until all positive examples in the training dataset are covered. The searching process for the best rule condition is the most time consuming process of CPAR since the gain for every possible item needs to be calculated in order to determine the best item gain. In the rules generation process, CPAR derives not only the best condition but also all similar ones since there are often more than one attribute items with similar gain.

D. Frequent Pattern Growth Approach

Han et al. [36] proposed the FP-growth algorithm to make the Aproiri algorithm more efficient (in terms of CPU time and memory usage). The FP-growth algorithm constructs a highly dense FP-tree that represents the training dataset, where each path in the tree represents an object in the training dataset. FP-growth discovers the frequent *ruleitems* without candidate *ruleitems* generation by performing two steps. The

process of FP-growth is performed through two steps. In the first step, a compact data structure, which they called FP-tree, is built using two passes over the training data set. In the second step, extracts frequent *ruleitems* directly from the FP-tree. During first step, the algorithm scans the training data set two times; the first one to compute the support for each single item of data set in order to omit infrequent ones and, the other reads each transaction to build the FP-tree. Once the FP-tree is built, the frequent *ruleitems* are extracted from the tree using pattern growth method by means patterns of length one in the FP-tree. Each frequent pattern may contains other frequent patterns occurring with it in the FP-tree, these frequent patterns must be generated and stored in a conditional FP-tree using the pattern links. The mining process is performed by concatenating the pattern with those produced from the conditional FP-tree. However, when the dataset is large, the FP-growth memory requirements becomes extremely large, which is primary drawback of the FP-growth technique. FP-growth has been successfully implemented by some association classification algorithms to minimize the frequent *ruleitems* step such as Malware detection AC [8], L^3 [16], L^3G [37], and CMAR [12]. The first AC algorithm that employed FP- Growth learning approach is CMAR, The process of rule generation in CMAR algorithm is described as follow: "the rules are stored in a descending order according to the frequency of attribute values appearing in the antecedent, in a prefix tree data structure known as a CR-tree". The constructed CR-tree represents generated rules. Each path stating from the root node to the last node, which contains the support, confidence and class of the intended rule, represents a generated rule. Different data sets from UCI data collection [31] are used in their experiments. The results showed that the CMAR algorithm is more accurate than decision tree C4.5 and CBA algorithms. In addition, the results revealed that 50–60% space of the main memory can be saved. In 2002, L^3 [16] has employed CMAR learning strategy in rule generation, though this algorithm adds on CMAR the concept of lazy pruning. Malware detection AC adopted the CMAR learning strategy in order to improve the performance involving the searching for correlations between the security features and the class attribute.

E. Repetitive Learning and Multiple Class Labels Approach

Traditionally, there are two types of classification problems; these are single label, and multi-label. In single label classification, each instance in the input data is associated with only one class. In cases where the input data set contains just two class labels, then the problem is called binary classification. If more than two classes are available, the problem is named multi-class. The majority of current AC algorithms extract single label classifiers in which the consequent of the rules in the classifier contains only one class However, some domain applications like medical diagnoses need existence of multiple class labels for each data instance.

Consider for example medical diagnosis where symptoms such as nausea, headache and sore throat could relate to three types of illness "tonsillitis", "migraine" or "flu", which are stored in a database. Some instances in the database are associated with two or three types of illness, and others are associated with a single type, thus this kind of data set can be considered multi-label. Assume that the frequencies of the symptoms (nausea, headache, sore throat) together in the database are 38, 36 and 26 with "tonsillitis", "migraine" or "flu" classes, respectively. Now, a traditional single class algorithm discovers only the rule associated with the largest frequency class ("tonsillitis"), and discards the other existing classes. Though, it is advantageous to find the other rules since they are valuable information having a large representation in the database. Meaning the two ignored rules by the single label classifier may take a role in the prediction step and may be very useful to the end user.

The MMAC algorithm [17] is the first multi-label algorithm in AC, Which is a repetitive learning algorithm that derives "If-Then" form. MMAC generates the multiple labels rules in a separate phase called recursive learning which requires multiple training data sets scans. For instance, MMAC consists of three stages, rules generation, recursive learning and classification. In the first phase, it scans the training data to extracts the first single labels classifier in the first iteration. Then all training cases associated with the derived rules are discarded, and the remaining unclassified cases in the original training data set comprise a new data set. In the second stage, MMAC finds all rules from T' to builds another single label classifier and removes all cases in T' which are associated with the generated rules, and so forth, until no further frequent rule items can be found. Finally, the rule sets that derived during each iteration are merged to form a global multi-label classifier that is then tested against test data. The results obtained from different data sets have indicated that the MMAC approach is an accurate and effective classification technique.

Veloso et al. [6] proposed a multiple label classification algorithm called Correlated Lazy Associative Classifier (CLAC) that adopts lazy classification approach [16], [37] in which it delays the reasoning process until a test case is given. The proposed method allows the presence of multiple classes in the consequent of the rules produced. Unlike binary classification which does not consider the correlation among classes, the proposed method takes into account classes relationships and training data overlapping with these classes. The learning strategy used by CLAC assigns a weight consisting of the confidence and support value of the rule(s) having the class and belonging to the test case, and then the class labels applicable to the test case get sorted by their weights. The method then assign the test case the largest class weight, and considers the test case a new feature and iteratively assigns new class labels to the test case until no more labels can be found. Furthermore, this learning method deals with the small disjoints (rules that cover limited number of training data), which removing them may reduce classification. The main drawback of CLAC lazy approach is that it does not allow the consequent of the rules to contain multiple labels, which is the main goal of multi-label classification in data mining.

F. Weighted Class Association Rule Mining

Association rule mining techniques assume that all items of the transactional database have the same importance (weight). However, there are situations where some items have more importance and are more significant than others. Sun and Bai [38] proposed an approach called weighted association rule mining that takes into account the weights of the items. The approach generates the weighted association rules by using two thresholds: weighted support and weighted confidence, instead of support and confidence. The rules that pass the predefined weighted support and confidence thresholds are added to the weighted association rule.

Ibrahim and Chandran [39] have adopted the weighted rules approach to construct the associative classifier. They used Information Gain notion before rule generation in order to reduce the number of produced rules. The information gain must be calculated for each attribute, the attribute with maximum value is considered as the best splitting attribute which will be used to generate the rules. The weighted class association rules are generated based on the two quality measurement factors of the rule. These are: weighted support and weighted confidence. The *ruleitems* that pass the predefined weight support and weighted confidence are added to the frequent weighted *ruleitems* set, while the others are added to infrequent.

IV. Rule Ranking Procedures

Most of the AC algorithms order the generated (CARs) using a group of parameters such as confidence, support to distinguish the rules in which it gives high confidence and support rules higher ranks. This is crucial since usually rules with higher ranks are tested first during the predicting of test cases, and the resulting classifier accuracy depends heavily on rules used during the prediction phase. There are several different criteria have developed in AC during sorting rules. For instance, CBA [4] considers the rule's confidence and supports the main criteria for rule favoring. This section shed the light on different rule ranking procedures developed in AC mining algorithms.

A. Confidence, Support, and Rule Cardinality Procedure

In the CBA algorithm, the rules are sorted according to three attributes: confidence, support and antecedent length. Firstly, the rules are sorted based on the confidence. If there are two rules have the same confidence they are sorted based on their supports. If their supports are also identical, they are sorted based on that generated earlier which means the CBA selects the rule based on lower antecedent length. Finally, if the antecedent lengths are identical, the algorithm sorts them randomly. Many AC algorithms employ this sorted procedure such as CBA (2) [30], CARGBA [32], ACCF [7].

Given two rules, r_i and r_j, $r_i$precedes r_j if

1. The confidence of $r_i$is larger than that of r_j
2. The confidences of r_i and r_j are the identical, but the support of $r_i$is larger than that of r_j
3. The confidence and support of r_i and r_j are the identical, but the $r_i$contains less number of attributes in its antecedent than that of r_j

Fig. 1 CBA rule ranking method

B. Confidence, Support, Rule Cardinality and Class Distribution Procedure

According to Thabtah et al. [18] the probability of having rules or more with similar confidence and support is high; therefore, the rule ranking process must not be limited to these parameters only. The authors proposed a new ranking rule technique to limit rule random selection. In addition to the parameters that are used in CBA algorithm, the authors proposed a new parameter called class distribution. Class distribution represents the occurring times number of a class in the training dataset. The rules are ranked based on the confidence, support, antecedent length and class distribution (in the order given). If the rules have equivalent values for all the four parameters, they are sorted randomly.

Given two rules, r_a and r_b, r_a precedes r_b if

1. The confidence of r_a is greater than that of r_b.
2. The confidence values of r_a and r_b are the same, but the support of r_a is greater than that of r_b.
3. Confidence and support values of r_a and r_b are the same, but r_a has fewer conditions in its left hand side than of r_b.
4. Confidence, support and cardinality of r_a andr_b are the same, but r_a is associated with a more representative class than that ofr_b.
5. All above criteria are identical for r_a and r_b, but r_a was generated from items that have higher order in the training data than that of r_b.

Fig. 2 MCAR rule ranking method

Rule ranking has been defined differently by associative algorithms. CBA and its successors considered confidence and support the main criteria for rule preference, and MCAR adds upon CBA the class distribution of the rules if two or more rules have identical confidence and support.

C. Lazy Ranking Procedures

Baralis and Torino [16] proposed a new sorted Procedure in their L^3 lazy AC algorithms. L^3 chose the antecedent lengths based on more attributes value hold, which is the opposite of the CBA rule ranking procedure. For example, In L^3ranking procedure if R1<senior, high>, and R2<senior> have the same confidence and supports R1 got higher ranked than R2 because it holds more attributes. These kinds of rules are named specific rules, it noticed that the specific rules in this procedure preferred more than general rules.

Given two rules, r_i and r_j,r_i precedes r_j if

1. The confidence of r_i is larger than that of r_j
2. The confidences of r_i and r_j are the identical, but the support of r_i is larger than that of r_j
3. The confidence and support values of r_i and r_j are the identical, but the r_i contains more number of attributes in its antecedent than that of r_j

Fig. 3L^{3}rule ranking method

D.Information Gain

The Information gain is mathematical measure mainly employed in decision trees such as C4.5 [10] by using the information gain in decision tree to measures how well a given attribute. It divides the training data cases into classes. The attribute with the highest information is chosen. In order to define information gain, first, it should measure the amount of information in an attribute using Entropy.

Given a set of training data cases *D* of *R* outcomes,

$$\text{Entropy}(D) = \sum -P_k \log_2 P_k \quad (1)$$

where P_k is the probability that *D* belongs to class *k*. The information gain of a set of data cases on attribute *A* is defined as:

$$\text{Gain}(D,A) = \text{Entropy}(D) - \sum ((|D_a| / |D|) * \text{Entropy}(D_a)) \quad (2)$$

where the sum is over each value of all possible values of attribute A, Da = subset of D for which attribute A has value a, |Da| = number of data cases in Da, |D| = number of data cases in D.

V. Building the Classifier and Rules Pruning

After all rules are found in the training phase and get ranked, AC algorithm evaluates the rules to come out with the most significant ones for building the final classifier for prediction. In AC, a classifier consists of a set of rules that are built from the training dataset. A major concern about AC algorithms is that they relatively produce a high number of rules that build the classifier [12]. Therefore, it is slowing the classification process, and some of these rules may be useless for classifier and redundant. Moreover, these rules can be discarded to in order to produce increase the effectiveness and the accuracy of the classifier. Many researchers had proposed solution to eliminate such rules. In this section, major techniques for building an AC classifier and techniques for rule pruning are surveyed.

A. Database Coverage Method

Database coverage is a pruning method that is used to reduce the size of the classifier. The technique was proposed by Liu et al. [4] in the CBA algorithm. Database coverage technique checks if each rule covers at least an object of the training dataset. If so, the rule is added to the classifier and its corresponding training object is deleted from the training dataset. This process is terminated once all training objects are deleted or all ordered rules have been examined. In case all the rules are evaluated and the training data is not empty, the remaining uncovered training cases are used to generate the default class rule which represents the largest frequency class in the remaining unclassified cases. It should be noted that before the database coverage terminates, the first rule which has the least number of errors is identified as the cut-off rule. All the rules after this rule are not included in the final classifier, since they only produce errors. This method has been utilized by many AC algorithms including CBA (2) [30], CMAR [12], CAAR [14], ACN [20], and ACCF [7].

```
Input: The set of sorted rules R and the training dataset D
Output: The classifier C

For each rule r_i in R do
    Mark all applicable cases in D that match r_i's body
    If r_i correctly classifies an case in D
        Insert r_i into the classifier
        Discard all cases in D covered by r_i
end if
If r_i cover no cases in D
        Delete r_i
end if
end
If D is not empty
    Generate a default rule for the largest frequency class in D
    Mark the least error rule in R as a cutoff rule.
end if
```

Fig. 4 Database Coverage rule evaluation method

B. Full and Partial Match Rule Evaluation Methods

This section discuses different rule pruning methods that primarily consider full or partial matching between the selected rule and the training case, in particular, database coverage method in [4], considers a rule significant if its body fully matches the training case and the rule has a common class with the training case class. Thabtah et al. [40] and Abu Mansour et al. [41] have evaluated different methods based on exact rule matching and partial rule matching of the rule body and the training case. Moreover, they evaluated the correctness of the rule's class with that of the training data when covering the training case.

1. Full Match Pruning Method (FMP)

The full match pruning method starts with the highest ranked rule, if the rule body matches any training case without the demand of the class correctness between the rule and the training case, then all training cases covered by the rule are marked for deletion and the rule gets inserted into the classifier. In cases where a rule cannot cover a training case then the rule is discarded. Full match pruning method terminates when the training data set gets empty or there are no rules left to be evaluated. In [46] Phishing Associative Classification algorithm (PAC) has adopted this pruning method in order to reduce over-fitting of the resulting classifier as well as its size.

```
Input: Training data set T and Set of Ranked Rules R
Output: Classifier (C1)

R' = sort(R);
For each rule r_i in R' Do
Find all applicable training cases in T that fully match r_i'scondition
Insert the rule at the end of C1
Remove all training cases in T covered byr_i.
else
   Discard r_iand remove it from R
end
Next r
```

Fig. 5 FMP rule evaluation method

2. High Precedence Method (HP)

In High Precedence (HP), after the complete set of rules are found and then ranked, a rule gets inserted into the classifier if its body partly matches the training case and also without class correctness between the rule class and that of the training case. This method iterates over the ranked rules starting with the highest ranked one, all training cases covered by the selected rule are discarded and the rule is inserted into the classifier. Any rule that does not cover a training case is removed. The loop terminates when either the training data set is empty or all rules are tested.

3. High Classify Pruning Method (HCP)

Once the rules are extracted and ranked, the High Classify Pruning Method goes over the complete set of rules and applies each rule against the training data set. If the rule partially covers a training case and has a common class to that of the training case it will be inserted in the classifier and all training cases covered by the rule are removed. The method repeats the same process for each rule remaining until the training data set becomes empty, and it considers the rules within the classifier in the prediction step.

```
Input: Given a set of generated rules R, and training data set T
Output: classifier (Cl)

R' = sort(R);
For each rule r_iin R' Do
    Find all applicable training cases in T that match r_i's condition
If r_i correctly classifies a training case in T
   Insert the rule at the end of Cl
Remove all training cases in T covered by r_i
end if
If r_i cannot correctly cover any training case in T
Remove r_i from R
end if
end for
```

Fig. 6 HCP rule evaluation method

C. Mathematical Based Pruning

1. Pessimistic Error Estimation

Decision tree algorithms like C4.5 [10] and See5 [42] are used pessimistic error estimation. Generally, there are two pruning strategies in decision trees; pre-pruning and post pruning. pre-pruning has been used in data mining within decision trees [10] by replacing a sub-tree with a leaf node. The decision of the replacement a sub-tree with a leaf node or to keep the sub-tree unchanged accurse by calculating the estimated error using the pessimistic error estimation measure on training data set. In AC mining, CBA is the first algorithm has employed pessimistic error pruning. In which for a rule R, CBA removes one of the attribute value in its antecedent to make a new rule R', then it compares the estimated error of R' with that of R. If the expected error of R' is smaller than that of R, then the original rule R gets replaced with the new rule R'.

2. X2 Testing

The chi-square test ($\chi 2$) is a mathematics evaluation testing that use $\chi 2$ for a group of objects to decide their independence or correlation, it evaluate the relation between two objects to decide if they are correlated or not [1]. In AC mining CMAR is the first AC algorithm that employed a weighted version of $\chi 2$, CMAR exceeds the correlation between the antecedent and the consequent of the rule to decide if the rule will be deleted or will be kept.

D. Long Rules Pruning

Redundant rules are produced as a result of the process of building rules in AC algorithms. In any AC algorithm, the process of introducing frequent *k-ruleitem* is accomplished by joining frequent *k-1-ruleitem*. Once the frequent *rulitems* are produced, the CARs are extracted from them. Therefore, some of these rules may belong to the same class. Long rule p runing discards specific rules that have confidence values larger than their subset general rules. This rule pruning method eliminates rules redundancy since many of the discovered rules have common attribute values in their antecedents. As a result the classifier may contain redundant rules and this becomes obvious particularly when the classifier size is large.

Li et al. [12] proposed redundant rule pruning technique for CMAR algorithm. Once all CARs are generated and ordered, they are examined to specify the redundant ones that will be pruned. Long rule pruning has been used in some algorithms (e.g. [15] [43]).

E. Rule Pruning Based On Indiscernibility Relationship

In database coverage, the rule that corresponds with at least a training object is added to the classifier and its corresponding object is removed from the training dataset. While the objects covered by the rules are all deleted, selection of other rules derived from the deleted objects may be affected. In order to overcome this problem, Jiang et al. [44] proposed method called rule pruning based on indiscernibility relationship. They defined relationship as indiscernibility when there are two rules that correspond to the same training data object. (i.e., the two rules have indiscernible role when classifying the object). Suppose we have two rules Ri : Xi →Ci and Rj : Xj →Cj that corresponds with the same training data object O. Rule Ri and Rj are γ-indiscernible for object O if Xi; Xj∈O and (conf(Ri)_ conf(Rj) <= γ where γ∈ [0,0.5] is an indiscernible index. This means that Ri and Rj have indiscernible role when classifying object

O. Therefore, they should be retained together when object is deleted by one of them.

F. Lazy Method

In lazy associative algorithms, only the rules that lead to misclassification on the training data set are pruned, on the other hand, all the rules that cover the training data are stored in a compact set to be used later during the prediction, unlike data base coverage based method, which prune any rule that does not cover a training case. In lazy pruning, after a complete set of rules are discovered and ranked, every rule selected that covers a training data case and have both the same class, the rule is inserted into a primary rule set and all its covered training cases will be deleted. In case the selected rule does not cover any training data case it will be removed from the classifier. The process is repeated until the all rules are tested or the all training data cases are discarded. The output of the class of the pruning is contain two sets one called the primary rules that covers training cases, and the other is called the secondary set that contains the rules were never used in the training data cases

The results of applying this pruning method in [37] on twenty six different data sets from [31] revealed that algorithms such as L^3andL^3Gthat employ lazy pruning method outperforms classification accuracy on average 1.63% against algorithms that used data base coverage such as CBA [4], CMAR [12].

G.Conflict Rules Pruning

In some datasets in which they considered dense datasets or multi-label i.e. multiple class labels associated with a training case, sometimes the probability of introducing two rules with the same antecedent and different class labels becomes high. These two rules are called conflicting rules [43]. The conflicting rules are defined as follows: given two rules R1: X→C1 and R2: X→C2, R1 and R2 are conflicting rules because they hold the same antecedent (X) and belong to deferent class labels C1 and C2. Having conflicting rules may cause the classifier to misclassify the test case that have the antecedent of the conflicting rules. Therefore, conflicting rules must be discarded.

VI. Class Prediction Methods

The key objective of building a classifier of any data mining algorithm is to allocate the appropriate class label class to test cases, which is called class prediction or forecasting. Prediction is the final and most important step in classification. It helps in determining the accuracy of the classifiers produced and, it uses the classifier's set to make the decision of assigning the class label to the test instance. In this section, some of these prediction methods are discussed.

A. Single Accurate Rule Prediction

When using the single rule prediction and given a classifier with a set of rules R and a test case t, the only rule that has the higher confidence between the rules R and fully matches the test case body is taken, if there are no rules cover the test case, the default class is taken which is the most class frequent in the training data. This prediction method called maximum likelihood several algorithms employed this method such as CBA [4], MCAR [12], CACA [19], ACN [20], ACCR [9] and others.

Input: Classifier (*R*), test data set (*Ts*), array *Tr*
Output: Prediction error rate *Pe*

Given a test data (*Ts*), the classification process works as follow:

For each test case *ts* Do
For each rule *r* in the set of ranked rules *R* Do
Find all applicable rules that match *ts* body and store them in *Tr*
If *Tr* is not empty Do
If there exists a rule *r* that fully matches *t*s condition
assign *r*'s class to *ts*
end if
else assign the default class to *ts*
end if
empty *Tr*
end
end
compute the total number of errors of *Ts*;

Fig. 7 CBA prediction method

There is a definite advantage of using just one rule in predicting test cases since only the highest applicable rule in the classifier has been used which is simple and efficient approach.

B. Group of Rules Prediction

One rule prediction method works fine especially when there is just a single rule applicable to the test case. However, in circumstances when more than one rule with close confidence values is applicable to the test case, this method decision is questionable since the selection of a single rule to make the class assignment is inappropriate. Thus, using all rules contributes to the prediction decision in these circumstances more appropriately. In this subsection, the different multiple rules prediction methods are discussed.

In [45] two prediction methods have been proposed based on group based prediction, the first method called Dominant Class, and the second method called Highest Group Confidence. Dominant Class works by selecting the rules from the classifier that match the conditions of the test case. Then it divides these rules into different groups based on class label to assign the test case with the class of the group that has largest count of rules. In the Highest Group Confidence the rules in the classifier also divide into groups based on class label. However, Highest Group Confidence assigns the class to the test case based on the group that has the largest average confidence.

```
Input: Classifier (R), test data set (Ts), array Tr
Output: error rate Pe

Given a test data (Ts), the classification process works as follow

For each test cases Do
    Assign=false
For each rule r in the set of ranked rules R Do
 Find all applicable rules that match ts body and store them in Tr
  If Tr is not empty Do
If there exists a rule r that matches any ts condition
      countper class +=1
end if
else assign the default class to ts and assign=true
end if
If assign is false then assign the dominant class count to ts
empty Tr
end
end
compute the total number of errors of Ts;
```

Fig. 8 Dominant class prediction method

C. Laplace Accuracy Method

CPAR algorithm [5] uses the Laplace accuracy method to assign the class to the test case, after all rules are generated and ranked, CPAR marks all the rules in the classifier that cover the test case and divide them into two groups based on the classes label, then it calculate the expected accuracy for the two groups to assign the test case with the class that achieved the highest average accuracy. The expected accuracy for each a rule (R) is obtained as follows:

$$Laplace(R) = \frac{(p_c(R)+1)}{(p_{tot}(R)+p)} \quad (3)$$

where:

P is the number of classes in the training data set

$p_{tot}(R)$ is the number of cases matching r antecedent

$p_c(R)$ is the number of training cases covered by R that belong to class c.

Experimental results [5] using twenty six dataset from UCI repository showed that CPAR performs slightly higher with reference to classification accuracy than CBA and decision trees C4.5.

D. Weighted Chi-Square Method

In the CMAR algorithm, Li et al. [12] used Weighted Chi-Square (Max $\chi 2$) method. CMAR starts by collecting the rules R_k that matches a test case t. If class labels of the collected rules R_k belong to the same class label, this class label is assigned to the test case t. Otherwise, when the collected rules R_k belong to the different classes, they are divided into multi groups based on their class labels. Each group has multi rules with the same class. CMAR compute the strength of each group, the class label of the group that its strength is the highest is assigned to the test case.

The Max $\chi 2$ of R_k is defined as:

$$\max \chi^2 = (\min\{Support(Cond), Support(c)\} - \frac{Support(Cond)Support(c)}{|T|})^2 |T| u \quad (4)$$

where,

$$u = \frac{1}{Support(Cond)Support(c)} + \frac{1}{Support(Cond)(|T| - Support(c))} + \frac{1}{(|T| - Support(Cond))Support(c)} + \frac{1}{(|T| - Support(Cond))(|T| - (Support(c))}$$

Zaïane and Antonie [13] have developed a prediction method that closely related to the CMAR prediction method, where the class of the subset of rules in R_k with the dominant class gets assigned to the test case t. Experimental results [12] on twenty six datasets in UCI repository showed that CMAR has better average classification accuracy in comparison with CBA and C4.5.

VII. Conclusions

Associative Classification (AC) is a promising data mining approach that integrates association rules mining and classification. AC mining builds more accurate classifiers than traditional classification technique such as decision trees and rule induction. In AC mining there are three main steps which are rule generation, classifier building and prediction appropriate class labels for test cases. This paper sheds the light and critically compared the different methods steps in AC mining algorithms. Furthermore, it discussed the different data representation models in AC mining algorithms such as horizontal and vertical. The intending work in near future is comprehensive experimental studies that compare the different AC algorithms methods with reference to classification accuracy, training time, and number of rules.

References

[1] Witten, I., and Frank, E. (2000) Data mining: practical machine learning tools and techniques with Java implementations. San Francisco: Morgan Kaufmann.

[2] Fayyad, U., and Irani, K. (1993) Multi—interval discretization of continues-valued attributes for classification learning. Proceedings of IJCAI, pp. 1022-1027. 1993.

[3] Agrawal, R., Imielinski, T., Swami, A (1993) Mining Association Rules between Sets of Items in Large Databases. In: Proc. of the ACM SIGMOD Conference, pp. 207–216, 1993.

[4] Liu, B., Hsu, W., and Ma, Y. (1998) Integrating classification and association rule mining. Proceedings of the Knowledge Discovery and Data Mining Conference- KDD, pp. 80-86. New York, NY.

[5] Yin, X., and Han, J. (2003) CPAR Classification based on predictive association rule. Proceedings of the –the SIAM International Conference on Data Mining -SDM, pp. 369-376, 2003.

[6] Veloso A., Meira W., Gonçalves M., Zaki. M (2007) Multi-label Lazy Associative Classification. Proceedings of the Principles of Data Mining and Knowledge Discovery - PKDD, pp. 605-612, 2007

[7] Li X., Qin D, and Yu C. (2008) ACCF: Associative Classification Based on Closed Frequent Itemsets. Proceedings of the Fifth International Conference on Fuzzy Systems and Knowledge Discovery -. FSKD. pp. 380-384, 2008.

[8] Ye Y., Jiang Q., and Zhuang W. (2008) Associative classification and post-processing techniques used for malware detection. Proceedings of the 2nd International Conference on Anti-counterfeiting, Security and Identification, 2008 –ASID, pp. 276-279, 2008.

[9] Niu Q., Xia S. and Zhang L. (2009). Association Classification Based on Compactness of Rules. Proceedings of the Second International Workshop on Knowledge Discovery and Data Mining - WKDD, pp.245-247, 2009.
[10] Quinlan, J. (1993) C4.5: Programs for machine learning. San Mateo, CA: Morgan Kaufmann.
[11] Jensen, D., and Cohen, P. (2000) Multiple comparisons in induction algorithms. Machine Learning 38(3), (pp. 309 – 338), 2000
[12] Li, W., Han, J., and Pei, J. (2001) CMAR: Accurate and efficient classification based on multiple-class association rule. Proceedings of the IEEE International Conference on Data Mining –ICDM, pp. 369-376, 2001.
[13] Antonie, M., Zaïane, O. (2002) Text document categorization by term association. Proceedings of the IEEE International Conference on Data Mining, (pp. 19-26). Maebashi City, Japan
[14] .Xu, X., Han, G., and Min H. (2004) A novel algorithm for associative classification of images blocks. Proceedings of the fourth IEEE International Conference on Computer and Information Technology, (pp. 46-51), 2004.
[15] Antonie, M., Zaïane, O. (2004). An associative classifier based on positive and negative rules. Proceedings of the 9th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (pp. 64 - 69). Paris, France.
[16] Baralis, E., and Torino, P. (2002) A lazy approach to pruning classification rules. Proceedings of the 2002 IEEE ICDM'02, (pp. 35).
[17] Thabtah, F., Cowling, P., and Peng, Y. (2004) MMAC: A new multi-class, multi-label associative classification approach. Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM '04), (pp. 217-224). Brighton, UK.
[18] Thabtah, F., Cowling, P., and Peng, Y. (2005) MCAR: Multi-class classification based on association rule approach. Proceeding of the 3rd IEEE International Conference on Computer Systems and Applications (pp. 1-7).Cairo, Egypt.
[19] Tang Z. and Liao Q. (2007). A New Class Based Associative Classification Algorithm. IMECS 2007: 685-689, 2007.
[20] Kundu G., Islam M., Munir S. and Bari M. (2008). ACN: An Associative Classifier with Negative Rules, 11th IEEE International Conference on Computational Science and Engineering. pp. 369-375,.
[21] Wang X., Yue K., Niu W., and Shi Z. (2011). An approach for adaptive associative classification. Expert Systems with Applications: An International Journal, Volume 38 Issue 9, pp. 11873-11883, 2011.
[22] Thabtah F (2006): Rule Preference Effect in Associative Classification Mining. Journal of Information and Knowledge Management, volume 5(1):13-20, 2006. WorldScinet, 2006.
[23] Zaki, M., Parthasarathy, S., Ogihara, M., and Li, W. (1997) New algorithms for fast discovery of association rules. Proceedings of the 3rd KDD Conference (pp. 283-286), 1997.
[24] Zaki, M., and Gouda, K. (2003) Fast vertical mining using diffsets. Proceedings of the ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 326 – 335, 2003
[25] Han J., Lin T. Y., Li J., Cercone N. (2007). Constructing Associative Classifiers from Decision Tables. Proceedings of the International conference: Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing –RSFDGrC , pp. 305-313, 2007
[26] Øhrn, A.(2001) ROSETTA Technical Reference Manual. Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim, Norway
[27] Pawlak, Z. (1991): Rough Sets: Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Dordrecht
[28] Agrawal, R., and Srikant, R. (1994) Fast algorithms for mining association rule. Proceedings of the 20th International Conference on Very Large Data Bases- VLDP,pp. 487-499, 1994.
[29] Michalski, R. (1980) Pattern recognition as rule-guided induction inference. IEEE Trans. on Pattern Analysis and Machine Intelligence 2, 349–361, 1980.
[30] Liu, B., Ma, Y., and Wong, C-K. (2001) Classification using association rules: weakness and enhancements. In Vipin Kumar, et al, (eds) Data mining for scientific applications, 2001.
[31] Merz, C., and Murphy, P. (1996) UCI repository of machine learning databases. Irvine, CA, University of California, Department of Information and Computer Science.
[32] Kundu G., Munir S., Md. Islam M., and Murase K. (2007) A Novel Algorithm for Associative Classification. Proceedings of the International Conference on Neural Information Processing- ICONIP, pp. 453-459, 2007.
[33] Huang, Z., Zhou, Z., He, T., & Wang, X. (2011, November). "ACAC: Associative Classification based on All-Confidence". IEEE International Conference on Granular Computing (GrC), pp. 289-293, 2011.
[34] Zaki M., Hsiao CJ (2002) CHARM: an efficient algorithm for closed itemset mining. Proceedings of the 2002SIAMinternational conference on data mining (SDM'02) , pp 457–473, 2002
[35] Quinlan, J., and Cameron-Jones, R. (1993) FOIL: A midterm report. Proceedings of the European Conference on Machine Learning, (pp. 3-20), Vienna, Austria.
[36] Han, J., Pei, J., and Yin, Y. (2000) Mining frequent patterns without candidate generation. Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, pp. 1-12, 2000.
[37] Baralis, E., Chiusano, S., and Graza, P. (2004) on support thresholds in associative classification. Proceedings of the 2004 ACM Symposium on Applied Computing, (pp. 553-558). Nicosia, Cyprus, 2004.
[38] Sun, K. and Bai, F. (2008, April). "Mining Weighted Association Rules without Pre-assigned Weights". IEEE Transactions on Knowledge and Data Engineering, Vol. 20, Issue: 4, pp. 489 – 495, 2008.
[39] Ibrahim, S., and Chandran, K.R., (2011, November). "Compact Weighted Class Association Rule Mining using Information Gain". International Journal of Data Mining & Knowledge Management Process (IJDKP) Vol.1, No.6, 2011.
[40] Thabtah F., Mahmood Q., McCluskey L., Abdel-jaber H (2010). A new Classification based on Association Algorithm. Journal of Information and Knowledge Management, Vol 9, No. 1, pp. 55-64. World Scientific.
[41] Abumansour H., Hadi W., McCluskey L., Thabtah F. (2010). Associative Text Categorisation Rules Pruning Method. Proceedings of the Linguistic and Cognitive Approaches to Dialog Agents Symposium (LaCATODA-10), RafalRzepka (Ed.), at the AISB, Pp. 39-44. UK.
[42] Quinlan, J. (1998) Data mining tools See5 and C5.0. Technical Report, RuleQuest Research.
[43] Antonie M., Zaïane O. R. and Coman A. (2003). Associative Classifiers for Medical Images, Lecture Notes in Artificial Intelligence 2797, Mining Multimedia and Complex Data, (pp. 68-83), Springer-Verlag
[44] Jiang Y, Liu Y, Liu X, Yang S (2010) Integrating classification capability and reliability in associative classification: A β-stronger model. Expert Systems with Applications, 37(5):3953-3961, 2010.
[45] Thabtah F., Hadi W., Abdelhamid N., Issa A.(2011) Prediction Phase in Associative Classification Mining. Journal of Knowledge Engineering and Software Engineering. World Science, 2011.
[46] Wedyan S., and Wedyan F. (2013) An Associative Classification Data Mining Approach for Detecting Phishing Websites, Journal of Emerging Trends in Computing and Information Sciences, 12(5):xx-xx, 2013.