A SHORT SURVEY ON CAR ALGORITHMS

Ilia Mitov, Stefan Karastanev, Krassimir Markov

Abstract: *In this paper we discuss in detail CAR algorithms. We present the main types of algorithms for association rule mining, pruning techniques, quality rule measures and rule ordering strategies. We also describe a number of specific CAR algorithms.*

Keywords: *CAR algorithms, Survey*

1. Introduction

Association rule mining quickly became a popular instrument to model relationships between class labels and features from a training set [Bayardo, 1998]. It appeared initially within the field of market basket analysis for discovering interesting rules from large data collections [Agrawal et al, 1993]. Since then, many associative classifiers were proposed, mainly differing in the strategies used to select rules for classification and in the heuristics used for pruning rules. "Class association rules" (CAR) algorithms have its important place in the family of classification algorithms.

Zaïane and Antonie suggested that the five major advantages of associative classifiers are the following [Zaïane and Antonie, 2005]:

- the training is very efficient regardless of the size of the training set;
- training sets with high dimensionality can be handled with ease and no assumptions are made on dependence or independence of attributes;
- the classification is very fast;
- classification based on association methods present higher accuracy than traditional classification methods [Liu et al, 1998] [Li et al, 2001] [Thabtah et al, 2005] [Yin and Han, 2003];
- the classification model is a set of rules easily interpreted by human beings and can be edited [Sarwar et al, 2001].

Within the data mining community, research on classification techniques has a long and fruitful history. However, classification techniques based on association rules, are relatively new. The first associative classifier CBA was introduced by [Liu et al, 1998]. During the last decade, various other associative

classifiers were introduced, such as CMAR [Li et al, 2001], ARC-AC and ARC-BC [Zaiane and Antonie, 2002], CPAR [Yin and Han, 2003], CorClass [Zimmermann and De Raedt, 2004], ACRI [Rak et al, 2005], TFPC [Coenen and Leng, 2005], HARMONY [Wang and Karypis, 2005], MCAR [Thabtah et al, 2005], CACA [Tang and Liao, 2007], ARUBAS [Depaire et al, 2008], etc.

CAR-algorithms are based on a relatively simple idea. Given a training set with transactions where each transaction contains all features of an object in addition to the class label of the object, the association rules are constructed, which have as consequent a class label. Such association rules are named "class association rules" (CARs).

Generally the structure of CAR-algorithms consists of three major data mining steps:

1. Association rule mining.
2. Pruning (optional).
3. Recognition.

The mining of association rules is a typical data mining task that works in an unsupervised manner. A major advantage of association rules is that they are theoretically capable of revealing all interesting relationships in a database. But for practical applications the number of mined rules is usually too large to be exploited entirely. This is why the pruning phase is stringent in order to build accurate and compact classifiers. The smaller the number of rules a classifier needs to approximate the target concept satisfactorily, the more human-interpretable is the result.

2. Association Rule Mining

Association rule mining was first introduced in [Agrawal et al, 1993]. It aims to extract interesting correlations, frequent patterns, associations, or casual structures among sets of instances in the transaction databases or other data repositories.

Association rule mining itself has a wide range of application domains such as market basket analysis, medical diagnosis/research, Website navigation analysis, homeland security and so on. In parallel, it participates as a step in the training process of CAR classifiers.

The datasets can be represented in two forms:

- transactional datasets;
- rectangular datasets.

In transactional datasets each record (transaction) can contain different number of items and order of the items can be arbitrary.

In rectangular datasets each record has the same number of attributes and position of the attribute value is fixed and corresponds to the attribute.

These differences are not particularly difficult to address since there is an easy way of converting transactional to binary rectangular dataset by ordering all possible items and pointing the presence of a concrete item with 1 (true) and respectively the absence with 0 (false).

The rectangular dataset also become transactional representation using attribute-value pairs in description of each record.

The description of the problem of association rule mining is firstly presented in [Agrawal et al, 1993]. The description of the problem provided below follows the one given in [Goethals, 2002].

Let $\mathbf{D}$ be a set of items.

A set $X = \{i_1,...,i_k\} \subseteq \mathbf{D}$ is called an itemset or a k-itemset if it contains k items.

A transaction over $\mathbf{D}$ is a couple $T = (tid, I)$ where tid is the transaction identifier and I is an itemset. A transaction $T = (tid, I)$ is said to support an itemset $X \subseteq \mathbf{D}$ if $X \subseteq I$.

A transaction database D over $\mathbf{D}$ is a set of transactions over $\mathbf{D}$.

The cover of an itemset X in D consists of the set of transaction identifiers of transactions in D that support X: $cover(X, D) := \{tid \mid (tid, I) \in D, X \subseteq I\}$.

The support of an itemset X in D is the number of transactions in the cover of X in D: $support(X, D) := |cover(X, D)|$. Note that $|D| = support(\{\}, D)$.

An itemset is called frequent if its support is no less than a given absolute minimal support threshold $MinSup$, with $0 \le MinSup \le |D|$.

Let D be a transaction database over a set of items $\mathbf{D}$, and $MinSup$ a minimal support threshold. The collection of frequent itemsets in D with respect to $MinSup$ is denoted by $F(D, MinSup) := \{X \subseteq \mathbf{D} \mid support(X, D) \ge MinSup\}$.

An association rule is an expression of the form $X \Rightarrow Y$, where X and Y are itemsets, and $X \cap Y = \{\}$. Such a rule expresses the association that if a transaction contains all items in X, then that transaction also contains all items in Y. X is called the body or antecedent, and Y is called the head or consequent of the rule.

The support of an association rule $X \Rightarrow Y$ in D, is the support of $X \cup Y$ in D. An association rule is called frequent if its support exceeds a given minimal support threshold $MinSup$.

The confidence or accuracy of an association rule $X \Rightarrow Y$ in D is the conditional probability of having Y contained in a transaction, given that X is contained in that transaction:

$$confidence(X \Rightarrow Y, D) := P(Y \mid X) = \frac{support(X \cup Y, D)}{support(X, D)}.$$

The rule is called confident if $P(Y \mid X)$ exceeds a given minimal confidence threshold $MinConf$, with $0 \le MinConf \le 1$.

Especially in the case of classification association rules the head consists of only one attribute-value pair. In the case of rectangular data one of the columns contains class labels that divide the dataset into separate extensional parts.

Generally, an association rules mining algorithm consists of the following steps:

1. The set of candidate k-item-sets is generated by 1-extensions of the large (k-1)-item-sets generated in the previous iteration.
2. Supports for the candidate k-item-sets are generated by a pass over the database.
3. Item-sets that do not have the minimum support are discarded and the remaining item-sets are called large (frequent) k-item-sets.
4. This process is repeated until no more large item-sets are found to generate association rules from those large item-sets with the constraints of minimal confidence.

In many cases, the algorithms generate an extremely large number of association rules, often in thousands or even millions; in addition to this the association rules are sometimes very large. It is nearly impossible for the end-users to comprehend or validate such large number of complex association rules, thereby limiting the usefulness of the data mining results. Several researchers suggested strategies aimed at reducing the number of association rules:

- extracting of rules based on user-defined templates or instance constraints [Baralis and Psaila, 1997] [Ashrafi et al, 2004];
- developing interestingness measures to select only interesting rules [Hilderman and Hamilton, 2002]. For instance [Jaroszewicz and Simovici, 2002] proposed a solution to the problem using the Maximum Entropy approach;
- proposing inference rules or inference systems to prune redundant rules and thus present smaller, and usually more understandable sets of association rules to the user [Cristofor and Simovici, 2002];

– creating new frameworks for mining association rule to find association rules with different formats or properties [Brin et al, 1997].

Depending of the specificity of the observed problem many additional question arise. For instance [Liu et al, 1999] present an approach to the rare instance problem. The dilemma that arises in the rare instance problem is that searching for rules that involve infrequent (i.e., rare) instances requires a low support but using a low support will typically generate many rules that are of no interest. Using a high support typically reduces the number of rules mined but will eliminate the rules with rare instances. The authors attack this problem by allowing users to specify different minimum supports for the various instances in their mining algorithm.

The computational cost of association rules mining can be reduced by sampling the database, by adding extra constraints on the structure of patterns, or through parallelization.

Techniques for association rule discovery have gradually been adapted to parallel systems in order to take advantage of the higher speed and greater storage capacity that they offer. The transition to a distributed memory system requires the partitioning of the database among the processors, a procedure that is generally carried out indiscriminately. [Parthasarathy et al, 2001] wrote an excellent survey on parallel association rule mining with shared memory architecture covering most trends, challenges, and approaches adopted for parallel data mining.

3. Creating Association Rules

During the first stage, several techniques for creating association rules are used, which mainly are based on:

- Apriori algorithm [Agrawal and Srikant, 1994] (CBA, ARC-AC, ARC-BC, ACRI, ARUBAS);
- FP-tree algorithm [Han and Pei, 2000] (CMAR);
- FOIL algorithm [Quinlan and Cameron-Jones, 1993] (CPAR);
- Morishita & Sese Framework [Morishita and Sese, 2000] (CorClass).

Generating association rules can be made from all training transactions together (such it is in ARC-AC, CMAR, CBA) or can be made for transactions grouped by class label (as it is in ARC-BC), which offers small classes a chance to have representative classification rules.

We provide a brief overview of some distinctive algorithms created during the recent years, which are used or can be implemented at the step of creating the pattern set of CAR algorithms.

3.1. AIS

The AIS algorithm [Agrawal et al, 1993] was the first algorithm proposed for mining association rule in the early 90s, when a task for emulating the biological immune system in the real world scenarios became actual. AIS algorithm uses candidate generation to detect the frequent item-sets. The candidates are generated on the fly and are compared with previously found frequent item-sets. In this algorithm only one instance of consequent association rules are generated, which means that the consequent of those rules only contain one instance, for example we only generate rules like $X \cap Y \Rightarrow Z$ but not those rules as $X \Rightarrow Y \cap Z$. The main drawbacks of the AIS algorithm are too many passes over the whole database and too many candidate item-sets that finally turned out to be small are generated, which requires considerable memory and involves significant effort that turned out to be useless.

3.2. Apriori

The Apriori [Agrawal and Srikant, 1994] is the most popular algorithm for producing association rules. It created new opportunities to mine the data. Since its inception, many scholars have improved and optimized the Apriori algorithm and have presented new Apriori-like algorithms. Apriori uses pruning techniques to avoid measuring certain item-sets, while guaranteeing completeness. These are the item-sets that the algorithm can prove will not turn out to be large.

However, there are two bottlenecks of the Apriori algorithm. One is the complex candidate generation process that uses most of the time and memory because of the multiple scans of the database. Based on the Apriori algorithm, many new algorithms were designed with some modifications or improvements.

The Apriori algorithm for finding frequent item-sets makes multiple passes over the data. In the k-th pass it finds all item-sets having k instances called the k-item-sets. Each pass consists of two phases. Let F_k represent the set of frequent k-item-sets, and C_k the set of candidate k-item-sets (potentially frequent item-sets). The candidate generation phase where the set of all frequent (k-1)-item-sets, F_{k-1}, found in the $(k-1)$-th pass is applied first and it is used to generate the candidate item-sets C_k. The candidate generation procedure ensures that C_k is a superset of the set of all frequent k-item-sets. A specialized hash-tree data structure is used to store C_k. Then, data is scanned in the support counting phase. For each transaction, the candidates in C_k contained in the transaction are determined using the hash-tree data structure and their support count is incremented. At the end of the pass, C_k is examined to determine which of the candidates are frequent, yielding F_k. The algorithm terminates when F_k or C_{k+1} becomes empty.

Several optimizations of Apriori algorithm are available, such as:

- PASCAL [Bastide et al, 2000], which introduces the notions of key patterns and use inference of other frequent patterns from the key patterns without access to the database;
- Category-based Apriori algorithm [Do et al, 2003], which reduces the computational complexity of the mining process by bypassing most of the subsets of the final item-sets;
- Apriori-T [Coenen et al, 2004], which makes use of a "reverse" set enumeration tree where each level of the tree is defined in terms of an array (i.e. the T-tree data structure is a form of Trie);
- FDM [Cheung et al, 1996], which is a parallelization of Apriori for shared machines, each with its own partition of the database. At every level and on each machine, the database scan is performed independently on the local partition. Then a distributed pruning technique is employed.

3.3. FP-Tree

FP-Tree [Han and Pei, 2000] is another milestone in the development of association rule mining, which breaks the main bottlenecks of Apriori [Kotsiantis and Kanellopoulos, 2006]. The frequent item-sets are generated with only two passes over the database and without any candidate generation process. FP-tree is an extended prefix-tree structure storing crucial, quantitative information about frequent patterns. Only frequent length-1 instances will have nodes in the tree, and the tree nodes are arranged in such a way that more frequently occurring nodes will have better chances of sharing nodes than less frequently occurring ones. FP-Tree scales much better than Apriori because as the support threshold goes down, the number as well as the length of frequent item-sets increase dramatically. The frequent patterns generation process includes two sub processes: constructing the FT-Tree, and generating frequent patterns from the FP-Tree. The mining result is the same with Apriori series algorithms.

To sum up, the efficiency of FP-Tree algorithm accounts for three reasons:

- The FP-Tree is a compressed representation of the original database because only those frequent instances are used to construct the tree, other irrelevant information are pruned.
- This algorithm only scans the database twice.
- FP-Tree uses a divide and conquers method that considerably reduced the size of the subsequent conditional FP-Tree.

Every algorithm has his limitations, for FP-Tree it is difficult to be used in an interactive mining system. Another limitation is that FP-Tree is that it is not suitable for incremental mining.

3.4. TreeProjection

The innovation brought by TreeProjection [Agarwal et al, 2000] is the use of a lexicographic tree, which requires substantially less memory than a hash tree. The number of nodes in its lexicographic tree is exactly that of the frequent item-sets. The support of the frequent item-sets is counted by projecting the transactions onto the nodes of this tree. This improves the performance of counting the number of transactions that have frequent item-sets.

The lexicographic tree is traversed in a top-down fashion. The efficiency of TreeProjection can be explained by two main factors:

- the transaction projection limits the support counting in a relatively small space.
- the lexicographical tree facilitates the management and counting of candidates and provides the flexibility of picking efficient strategy during the tree generation and transaction projection phrases.

3.5. Matrix Algorithm

The Matrix Algorithm [Yuan and Huang, 2005] generates a matrix, which entries 1 or 0 by passing over the database only once, and then the frequent candidate sets are obtained from the resulting matrix. Finally, association rules are mined from the frequent candidate sets. Experimental results confirm that the proposed algorithm is more effective than Apriori Algorithm.

3.6. Sampling Algorithms

For obtaining associations, several algorithms use sampling. Some examples are provided below:

- Toivonen's sampling algorithm [Toivonen, 1996]. This approach is a combination of two phases. During phase 1 a sample of the database is obtained and all associations in the sample are found. These results are then validated against the entire database. To maximize the effectiveness of the overall approach, the author makes use of lowered minimum support on the sample. Since the approach is probabilistic (i.e. dependent on the sample containing all the relevant associations) not all the rules may be found in this first pass. Those associations that were deemed not frequent in the sample but were actually frequent in the entire dataset are used to construct the complete set of associations in phase 2;

- Progressive sampling [Parthasarathy, 2002] is yet another approach; it relies on a novel measure of model accuracy (self-similarity of associations across progressive samples), the identification of a representative class of frequent item-sets that mimic (extremely accurately) the self-similarity values across the entire set of associations, and an efficient sampling methodology that hides the overhead of obtaining progressive samples by overlapping it with useful computation;
- Sampling Error Estimation algorithm [Chuang et al, 2005] aims to identify an appropriate sample size for mining association rules. It has two advantages. First, it is highly efficient because an appropriate sample size can be determined without the need of executing association rules. Second, the identified sample size is very accurate, meaning that association rules can be highly efficiently executed on a sample of this size to obtain a sufficiently accurate result;
- Sampling large datasets with replacement [Li and Gopalan, 2004] is used when data comes as a stream flowing at a faster rate than can be processed. Li and Gopalan derive the sufficient sample size based on central limit theorem for sampling large datasets with replacement.

3.7. Partition

Partition [Savasere et al, 1995] is fundamentally different from other algorithms because it reads the database at most two times to generate all significant association rules. In the first scan of the database, it generates a set of all potentially large item-sets by scanning the database once and dividing it in a number of non-overlapping partitions. This set is a superset of all frequent item-sets so it may contain item-sets that are not frequent. During the second scan, counters for each of these item-sets are set up and their actual support is measured.

3.8. FOIL

FOIL (First Order Inductive Learner) is an inductive learning algorithm for generating classification association rules (CARs) developed by Quinlan and Cameron-Jones in 1993 [Quinlan and Cameron-Jones, 1993] and further developed by Yin and Han to produce the PRM (Predictive Rule Mining) CAR generation algorithm [Yin and Han, 2003]. PRM was then further developed, by Yin and Han, to produce CPAR (Classification based on Predictive Association Rules).

FOIL is a sequential covering algorithm that learns first-order logic rules. It learns new rules one at a time, removing the positive examples covered by the latest rule before attempting to learn the next rule.

The hypothesis space search performed by FOIL is best understood by viewing it hierarchically. Each iteration through FOIL'S outer loop adds a new rule to its disjunctive hypothesis. The effect of each new rule is to generalize the current disjunctive hypothesis (i.e., to increase the number of instances it classifies as positive), by adding a new disjunct. Viewed at this level, the search is a specific-to-general search through the space of hypotheses, beginning with the most specific empty disjunction and terminating when the hypothesis is sufficiently general to cover all positive training examples. The inner loop of FOIL performs a finer-grained search to determine the exact definition of each new rule. This inner loop searches a second hypothesis space, consisting of conjunctions of literals, to find a conjunction that will form the preconditions for the new rule. FOIL employs a specific performance FOIL Gain that differs from the entropy measure. This difference follows from the need to distinguish between different bindings of the rule variables and from the fact that FOIL seeks only rules that cover positive examples [Mitchell, 1997].

3.9. Morishita & Sese Framework

This framework [Morishita and Sese, 2000] efficiently computes significant association rules according to common statistical measures such as a chi-squared value or correlation coefficient. Because of anti-monotonicity of these statistical metrics, Apriori algorithm is not suitable for associative rule generation. Morishita and Sese present a method of estimating a tight upper bound on the statistical metric associated with any superset of an item-set, as well as the novel use of the resulting information of upper bounds to prune unproductive supersets while traversing item-set lattices.

4. Rule Quality Measures

The process of generating association rules usually creates an extremely big number of patterns. This bottleneck imposes the necessity of measuring the significance, respectively redundancy of the generated rules and ordering using different criteria.

Here we will mention some examples of used ranking of association rules.

For a rule P and a class-labeled data set $D = \{R_i \mid i = 1,...,n\}$ several kinds of rule quality measures and combinations of them are used:

- The time of generation of the rule. This is a weak restriction used when all constrains before order two rules in equal places;
- $ncovers(P)$: the number of instances covered by P (i.e. $R_i : body(P) \subseteq body(R_i)$);

- $pos(P)$: the number of instances correctly classified by P (i.e. $R_i : body(P) \subseteq body(R_i)$ and $head(P) = head(R_i)$);
- $neg(P)$: the number of negative instances covered by P (i.e. $R_i : body(P) \subseteq body(R_i)$ and $head(P) \neq head(R_i)$);
- $|D|$: the number of instances in D;
- Coverage: $coverage(P) = \frac{ncovers(P)}{|D|}$;
- Accuracy: $accuracy(P) = \frac{pos(P)}{ncovers}$;
- Cardinality: $card(P) = |body(P)|$;
- Pessimistic error rate: $PER(P) = \frac{neg(P)+1}{neg(P)+pos(P)+2}$
- p_i is the probability of class c_i in D;
- Expected information: $Info(D) = -\sum_{i=1}^{m} p_i * \log_2(p_i)$;
- Information gain: $InfoGain(D) = Info(D) - \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$;
- FOIL gain (it favors rules that have high accuracy and cover many positive instances): $FOILGain(P,P') = pos(P') \times \left(\log_2 \frac{pos(P')}{pos(P')+neg(P')} - \log_2 \frac{pos(P)}{pos(P)+neg(P)} \right)$;

Further measures can be defined but those listed above are the most basic ones.

5. Pruning

In order to reduce the produced association rules, pruning in parallel with (pre-pruning) or after (post-pruning) creating association rules is performed. Different heuristics for pruning during rule generation are used, mainly based on minimum support, minimum confidence and different kinds of error pruning [Kuncheva, 2004]. In post-pruning phase, criteria such as data coverage (ACRI) or correlation between consequent and antecedent (CMAR) are also used.

During the pruning phase or in classification stage, different ranking criteria for ordering the rules are used. The most common ranking mechanisms are based on the support, confidence and cardinality of the rules, but other techniques such as the cosine measure and coverage measure (ACRI) also exist; we can mention amongst them:

- Pruning by confidence: retain more general rules with higher accuracy: $|R_1| < |R_2|$ and $conf(R_1) < conf(R_2)$, than R_1 is pruned (used in ARC-AC, ARC-BC);
- Pruning by precedence: special kind of ordering using "precedence" (CBA and MCAR);
- Correlation pruning: statistical measuring of the rule significance using weighted χ^2 (CMAR).

6. Recognition

In the recognition stage, three different approaches can be discerned [Depaire et al, 2008]:

1. using a single rule.
2. using a subset of rules.
3. using all rules.

An example which uses a single rule is CBA. It classifies an instance by using the single best rule covering the instance.

CPAR 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.

Additionally to support, coverage and confidence, ACRI uses also the cosine measure.

CMAR uses all rules covering a class to calculate an average score per class.

CMAR selects the rule with the highest χ^2 measure from the candidate set.

ARC-AC and ARC-BC use the sum of confidences as score statistics.

A different approach is proposed in TFPC, which suggests to consider the size of the antecedent and to favor long rules before making an allowance for confidence and support.

When a subset or all rules is being used, several order-based combined measures can be applied:

- Select all matching rules;
- Group rules per class value;
- Order rules per class value according to criterion;
- Calculate combined measure for best Z rules;

— Laplace Accuracy (CPAR): if k is the number of class values then $LA = \frac{support(R)+1}{support(body(R))+k}$

7. Some Representatives of CAR Algorithms

In this subsection we present briefly several representatives of the CAR Algorithm.

7.1. CBA

In CBA [Liu et al, 1998], Apriori is applied to create the association rules.

For measuring the significance of the rules a special "precedence" definition is given:

$P^1 \succ P^2$ (rule P^1 precedes rule P^2) if:

1. $confidence(P^1) > confidence(P^2)$;
2. $confidence(P^1) = confidence(P^2)$ but $support(P^1) > support(P^2)$;
3. $confidence(P^1) = confidence(P^2)$, $support(P^1) = support(P^2)$, but P^1 is generated earlier than P^2.

Pruning is based on the pessimistic error rate based pruning method in C4.5.

Condition 1. Each training case is covered by the rule with the highest precedence among the rules that can cover the case.

Condition 2. Every rule correctly classifies at least one remaining training case when it is chosen.

The key point is instead of making one pass over the remaining data for each rule, the algorithm to find the best rule to cover each case.

During the recognition CBA just searches in the pruned and ordered list for the first rule that covers the instance to be classified. The prediction is the class label of that classification rule. If no rule covers the instance, CBA uses the default class calculated during pruning. If the decision list is empty, the majority class of the training instance will be assigned to each test instance as default.

7.2. CMAR

CMAR [Li et al, 2001] employs a novel data structure, CR-tree, to compactly store and efficiently retrieve a large number of rules for classification. CR-tree is a prefix tree structure to explore the sharing among rules, which achieves substantial compactness.

In the phase of rule generation, CMAR computes the complete set of rules. CMAR prunes some rule and only selects a subset of high quality rules for classification. CMAR adopts a variant of FP-growth

method, which is much faster than Apriori-like methods, especially in the situations where large data sets, low support threshold, and long patterns exist. The specificity of CMAR is also that it finds frequent pattern and generates rules in one step.

For every pattern, CMAR maintains the distribution of various class labels among data objects matching the pattern. This is done without any overhead in the procedure of counting (conditional) databases. On the other hand, CMAR uses class label distribution to prune. Once a rule is generated, it is stored in a CR-tree.

The number of rules generated by class-association rule mining can be huge. To make the classification effective and also efficient, we need to prune rules to delete redundant and noisy information. According to the facility of rules on classification, a global order of rules is composed.

CMAR employs the following methods for rule pruning:

1. Using general and high-confidence rule to prune more specific and lower confidence ones.
2. Selecting only positively correlated rules.
3. Pruning rules based on database cover.

In the phase of classification, for a given data object, CMAR selects a small set of high confidence matching the object, highly related rules and analyzes the correlation among those rules.

7.3. ARC-AC and ARC-BC

In 2002, Zaïane and Antonie offered new associative-based classifiers for text categorization – ARC-AC and ARC-BC [Zaïane and Antonie, 2002]. For building association rules they used Apriori-like algorithm. They have considered two different approaches for extracting term-category association rules and for combining those rules to generate a text classifier.

In the first approach ARC-BC (Association Rule-based Categorizer by Category), each category is considered as a separate collection and the association rule mining applied to it. Once the frequent item-sets are discovered, the rules are simply generated by making each frequent item-set the antecedent of the rule and the current category the consequent.

The ARC-AC (Association Rule-based Categorizer for All Categories) considers all categories at whole. In this case one antecedent can be found with different consequents. During the recognition they introduce "dominant factor", which is the proportion of rules of the most dominant category in the applicable rules to the query.

7.4. CPAR

A greedy associative classification algorithm called CPAR was proposed in [Yin and Han, 2003]. CPAR adopts FOIL [Quinlan and Cameron-Jones, 1993] strategy 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 data set are covered.

The searching process for the best rule condition is time consuming process for CPAR since the gain for every possible item needs to be calculated in order to determine the best item gain. Thus, CPAR uses an efficient data structure, i.e. PN Array, to store all the necessary information for calculation of the items gain. In the rules generation process, CPAR derives not only the best condition but all close similar ones since there are often more than one attribute items with similar gain.

7.5. CorClass

CorClass [Zimmermann and De Raedt, 2004] directly finds the best correlated associations rules for classification by employing a branch-and-bound algorithm, using so called Morishita & Sese Framework [Morishita and Sese, 2000]. It follows the strategy in which calculating the upper bounds on the values attainable by specializations of the rule currently considered. The upper bound finally allows dynamic rising of the pruning threshold, differing from the fixed minimal support used in existing techniques. This will result in earlier termination of the mining process. Since the quality criterion for rules is used directly for pruning, no post-processing of the discovered rule set is necessary.

The algorithm uses two strategies for classifying a new object

1. Decision List: Rank all the rules (rules are ranked by quality according to some criterion) and use the first rule satisfied by an example for classification.
2. Weighted Combination: The general way to do this is to collect all such rules, assign each one a specific weight and for each class predicted by at least one rule sum up the weights of corresponding rules. The class value having the highest value is returned.

7.6. ACRI

The task of ACRI (Associative Classifier with Reoccurring Items) [Rak et al, 2005] is to combine the associative classification with the problem of recurrent items.

A delicate issue with associative classifiers is the use of a subtle parameter: support. Support is a difficult threshold to set, inherited from association rule mining. It is known in the association rule mining

field that the support threshold is not obvious to tune in practice. The accuracy of the classifier can be very sensitive to this parameter.

The algorithm for mining associations in ACRI is based on earlier work of the authors Apriori-based MaxOccur [Zaiane et al, 2000]. The building of the classification model follows their previous ARC-BC approach. The rational is based on the efficiency of this method in the case of non-evenly distributed class labels. MaxOccur run on transactions from each known class separately makes the core of the rule generator module. It mines the set of rules with reoccurring items from the training set.

These rules associate a condition set with a class label such that the condition set may contain items preceded by a repetition counter. The classification process might be considered as plain matching of the rules in the model to the features of an object to classify. Different classification rules may match, thus the classifier module applies diverse strategies to select the appropriate rules to use.

In addition, simple matching is sometimes not possible because there is no rule that has the antecedent contained in the feature set extracted from the object to classify. With other associative classifiers, a default rule is applied, either the rule with the highest confidence in the model or simply assigning the label of the dominant class. The ACRI approach has a different strategy allowing partial matching or closest matching by modeling antecedents of rules and new objects in a vector space.

7.7. TFPC

TFPC (Total From Partial Classification) [Coenen and Leng, 2005] is a classification association rule mining algorithm founded on the TFP (Total From Partial) association rule mining algorithm; which, in turn, is an extension of the Apriori-T (Apriori Total).

TFP (Total From Partial) algorithm builds a set enumeration tree structure, the P-tree, that contains an incomplete summation of support-counts for relevant sets. Using the P-tree, the algorithm uses an Apriori-like procedure to build a second set enumeration tree, the T-tree, that finally contains all the frequent sets (i.e. those that meet the required threshold of support), with their support-counts. The T-tree is built level by level, the first level comprising all the single items (attribute-values) under consideration. In the first pass, the support of these items is counted, and any that fail to meet the required support threshold are removed from the tree. Candidate-pairs are then generated from remaining items, and appended as child nodes. The process continues, as with Apriori, until no more candidate sets can be generated.

The class-competition is solved by using support and confidence measures.

7.8. HARMONY

HARMONY [Wang and Karypis, 2005] directly mines for each training instance one of the highest confidence classification rules that it supports and satisfies a user specified minimum support constraint, and builds the classification model from the union of these rules over the entire set of instances. Thus HARMONY employs an instance-centric rule generation framework and mines the covering rules with the highest confidence for each instance, which can achieve better accuracy. Moreover, since each training instance usually supports many of the discovered rules, the overall classifier can better generalize to new instances and thus achieve better classification performance.

To achieve high computational efficiency, HARMONY mines the classification rules for all the classes simultaneously and directly mines the final set of classification rules by pushing deeply some effective pruning methods into the projection-based frequent item-set mining framework. All these pruning methods preserve the completeness of the resulting rule-set in the sense that they only remove from consideration rules that are guaranteed not to be of high quality.

7.9. MCAR

MCAR (Multi-class Classification based on Association Rule) [Thabtah et al, 2005] uses an efficient technique for discovering frequent items and employs a rule ranking method which ensures detailed rules with high confidence.

During the rules generation MCAR scans the training data set to discover frequent single items, and then recursively combines the items generated to produce items involving more attributes. After that the rules are used to generate a classifier by considering their effectiveness on the training data set, using expanded definition of "precedence".

$P^1 \succ P^2$ (rule P^1 precedes rule P^2) if:

$confidence(P^1) > confidence(P^2)$;

$confidence(P^1) = confidence(P^2)$ but $support(P^1) > support(P^2)$;

$confidence(P^1) = confidence(P^2)$, $support(P^1) = support(P^2)$,

but $ActAcc(P^1) = ActAcc(P^2)$;

All conditions before are the same, but $card(P^1) < card(P^2)$;

Last condition: P^1 is generated earlier than P^2.

7.10. CACA

The following innovations are integrated in CACA [Tang and Liao, 2007]:

- use the class-based strategy to cut down the searching space of frequent pattern;
- design a structure call Ordered Rule Tree (OR-Tree) to store the rules and their information which may also prepare for the synchronization of the two steps;
- redefine the compact set so that the compact classifier is unique and not sensitive to the rule reduction;
- synchronize the rule generation and building classifier phases.

Class-based strategy: Given a training data set D with k classes, the principle idea of class based rule mining is to divide the single attribute value set C_{all} for all classes into k smaller ones for every class, that is, to limit the searching in k low dimensional spaces other than a high dimensional one.

OR-Tree: To facilitate they design a structure, called Ordered-Rule-Tree (OR-Tree), under the inspiration of CR-Tree used in CMAR to store and rank rules. It is composed with a tree structure and an ordered list. When a rule $P^i = (c^i \mid a_1^i, ..., a_n^i)$ satisfying the support and confidence thresholds is generated, attribute values $a_1^i, ..., a_n^i$ are stored as nodes in this tree according to their frequency in D in descending order. The last node points to an information node storing the rule's information such as class label, support and confidence. Each rule can and only can have one information node. The ordered list is designed to organize all rules in the tree. Each node in the chain points to a certain rule. Nodes pointing to the rules with higher priority are closer to the head node, while those pointing to the rules with lower priority are farther from the head node.

The ranking rule criteria are as follows:

$P^1 \succ P^2$ (P^1 precedes P^2) if:

$confidence(P^1) > confidence(P^2)$;

$confidence(P^1) = confidence(P^2)$ but $support(P^1) > support(P^2)$;

$confidence(P^1) = confidence(P^2)$ and $support(P^1) = support(P^2)$ but $card(P^1) < card(P^2)$ (P^1 is more general than P^2);

Equal previous conditions, but P^1 is generated earlier then P^2.

To ensure compact classifier to be unique and not sensitive to the rule reduction, the redundant rules are defined as follows:

Definition of redundant rule: Given P^1, P^2 and P^3, that belong to rule set R, P^2 is redundant if:

- $P^1 = (c^1 \mid a_1^1,...,a_{k1}^1)$, $P^2 = (c^2 \mid a_1^2,...,a_{k2}^2)$:

  $c^1 \neq c^2$, $(a_1^1,...,a_{k1}^1) \subseteq (a_1^2,...,a_{k2}^2)$, $P^1 \succ P^2$;

- $P^1 = (c^1 \mid a_1^1,...,a_{k1}^1)$, $P^2 = (c^2 \mid a_1^2,...,a_{k2}^2)$:

  $c^1 = c^2$, $(a_1^1,...,a_{k1}^1) \subset (a_1^2,...,a_{k2}^2)$, $P^1 \succ P^2$;

- $P^1 = (c^1 \mid a_1^1,...,a_{k1}^1)$, $P^2 = (c^2 \mid a_1^2,...,a_{k2}^2)$, $P^3 = (c^3 \mid a_1^3,...,a_{k3}^3)$:

  $c^1 = c^2 \neq c^3$, $(a_1^1,...,a_{k1}^1) \subset (a_1^2,...,a_{k2}^2)$, $(a_1^1,...,a_{k1}^1) \subset (a_1^3,...,a_{k3}^3)$, $P^1 \succ P^2 \succ P^3$.

Definition of compact rule set: For rule set R, if $\mathrm{R}' \subset \mathrm{R}$, any redundant rule $P \notin \mathrm{R}'$ and R' is unique, then R' is the compact set of R.

CACA technically combined the rule generation and the building classifier phases together. Once a new rule is generated, the algorithm visits the OR-Tree partially to recognize its redundancy, stores it in the OR-Tree and ranks it in the rule set. Not only can the synchronization simplify the procedure of associative classification but also apply the pruning skill to shrink the rule mining space and raise the efficiency.

7.11. ARUBAS

In contrast with many existing associative classifiers, ARUBAS [Depaire et al, 2008] 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. Five different fitness measures are used for classification purposes.

The main idea behind the ARUBAS framework, is transformation of the original feature space into a more powerful feature space. The original feature space is called the attribute space, where each record $R^i = (c^i \mid a_1^i,...,a_n^i)$ 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 $P_p = \langle (A_{i1}, a_{i1}),...,(A_{ik}, a_{ik}) \rangle$. For achieving more power for the feature space, only combinations of attributes (or patterns) which are strongly associated with a single class value is given.

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 item-

set, will become a pattern P_p and hence a dimension in the new feature space. The value of an instance R^i for a pattern dimension P_p is 1 (if the instance contains the pattern) or 0 (if it doesn't).

The instance similarity is used for classifying new instances. To measure the similarity between a new instance R^i and a known training instance R^t ARUBAS focuses on the patterns contained by both instances and how many patterns both instances have in common, but on those patterns coming from the CARs which predicted the class value of the training instance R^t.

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. This similarity is not measured in the original attribute space, but in the pattern space, which is constructed by means of CARs.

8. Conclusion

This chapter provided an overview of the area of CAR-classifiers. CAR algorithms have its important place in the family of classification algorithms with several advantages, such as: efficiency of the training regardless of the training set; easy handling with high dimensionality; very fast classification; high accuracy; human comprehensible classification model.

We observed all typical steps in the whole classification process of CAR algorithms: generating the rules, pruning, and recognition.

In the phase of generating the rules several techniques are observed: the pioneer AIS, most used Apriori, alternative FP-Tree, TreeProjection, Matrix Algorithm, Sampling Algorithms, Partition, FOIL and Morishita & Sese Framework.

The pruning is important step in the learning process of CAR algorithms, applied as preprocessing step, in parallel of association rule mining or after it. Here we made a brief observation of several rule quality measures and rule ordering schemes, used in CAR algorithms.

In the recognition phase we also observed different types of choosing final decision – using simple rule or set of rules with different types of ordering schemas.

Finally, using the proposed framework, typical for CAR algorithms, we analyze the some representatives of CAR algorithms: CBA, CMAR, ARC-AC and ARC-BC, CPAR, CorClass, ACRI, TFPC, HARMONY, MCAR, CACA, ARUBAS, showing wide variety of proposed techniques.

Bibliography

[Agarwal et al, 2000] Agarwal, R. Aggarwal, C., Prasad V.: A tree projection algorithm for generation of frequent item-sets. In J. of Parallel and Distributed Computing, 61/3, 2001, pp.350-371.

[Agrawal and Srikant, 1994] Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In Proc. 20th Int. Conf. Very Large Data Bases, 1994, pp.487-499.

[Agrawal et al, 1993] Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. Proc. of the ACM SIGMOD Int. Conf. on Management of Data, Washington, DC, 1993, pp. 207-216.

[Ashrafi et al, 2004] Ashrafi, M., Taniar, D. Smith, K.: A new approach of eliminating redundant association rules, LNCS, Vol.3180, 2004, pp.465-474.

[Baralis and Psaila, 1997] Baralis, E., Psaila, G.: Designing templates for mining association rules. Journal of Intelligent Information Systems, 9/1, 1997, pp.7-32.

[Bastide et al, 2000] Bastide, Y., Taouil, R., Pasquier, N., Stumme, G., Lakhal, L.: Mining frequent patterns with counting inference. ACM SIGKDD Explorations Newsletter, 2000

[Bay, 2000] Bay, S.: Multivariate discretization of continuous variables for set mining. In Proc. of the 6th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, 2000, pp.315-319.

[Brin et al, 1997] Brin, S., Motwani, R., Ullman, J., Tsur, S.: Dynamic itemset counting and implication rules for market basket data. In Proc. ACM SIGMOD Int. Conf. on Management of Data, 1997, pp.255-264.

[Cheung et al, 1996] Cheung, D., Han, J., Ng, V., Fu, A., Fu, Y.: A fast distributed algorithm for mining association rules. In Proc. of 1996 Int. Conf. on Parallel and Distributed Information Systems, Miami Beach, Florida, 1996, pp.31-44.

[Chuang et al, 2005] Chuang, K., Chen, M., Yang, W.: Progressive sampling for association rules based on sampling error estimation, LNCS, Vol.3518, 2005, pp.505-515.

[Coenen and Leng, 2005] Coenen, F., Leng, P.: Obtaining Best Parameter Values for Accurate Classification. Proc. ICDM'2005, IEEE, pp.597-600.

[Coenen et al, 2004] Coenen, F., Goulbourne, G., Leng, P.: Tree structures for mining association rules. Data Mining and Knowledge Discovery, Kluwer Academic Publishers, 8, 2004, pp.25-51.

[Cristofor and Simovici, 2002] Cristofor, L., Simovici, D.: Generating an informative cover for association rules. In Proc. of the IEEE Int. Conf. on Data Mining, 2002.

[Depaire et al, 2008] Depaire, B., Vanhoof, K., Wets, G.: ARUBAS: an association rule based similarity framework for associative classifiers. IEEE Int. Conf. on Data Mining Workshops, 2008, pp.692-699.

[Do et al, 2003] Do, T.D., Hui, S.-C., Fong, A.: Mining frequent item-sets with category-based constraints, LNCS, Vol. 2843, 2003, pp.76-86.

[Goethals, 2002] Goethals, B.: Efficient Frequent Pattern Mining. PhD Thesis, Transnationale Univeriteit Limburg, 2002.

[Han and Pei, 2000] Han, J., Pei, J.: Mining frequent patterns by pattern-growth: methodology and implications. ACM SIGKDD Explorations Newsletter 2/2, 2000, pp.14-20.

[Hilderman and Hamilton, 2002] Hilderman, R., Hamilton, H.: Knowledge Discovery and Interest Measures. Kluwer Academic, Boston, 2002.

[Jaroszewicz and Simovici, 2002] Jaroszewicz, S., Simovici, D.: Pruning redundant association rules using maximum entropy principle, LNCS, Vol. 2336, 2002, pp.135-142.

[Kotsiantis and Kanellopoulos, 2006] Kotsiantis, S., Kanellopoulos, D.: Association rules mining: a recent overview. GESTS International Transactions on Computer Science and Engineering, 32/1, 2006, pp. 71-82

[Kuncheva, 2004] Kuncheva, L.: Combining Pattern Classifiers: Methods and Algorithms. Willey, 2004.

[Li and Gopalan, 2004] Li, Y., Gopalan, R.: Effective sampling for mining association rules. LNCS, Vol. 3339, 2004, pp.391-401.

[Li et al, 2001] Li, W., Han, J., Pei, J.: CMAR: Accurate and efficient classification based on multiple class-association rules. In: Proc. of the IEEE Int. Conf. on Data Mining ICDM, 2001, pp.369-376.

[Liu et al, 1998] Liu, B., Hsu, W., Ma, Y.: Integrating classification and association rule mining. In Knowledge Discovery and Data Mining, 1998, pp.80-86.

[Liu et al, 1999] Liu, B. Hsu, W., Ma, Y.: Mining association rules with multiple minimum supports. Proc. Knowledge Discovery and Data Mining Conf., 1999, pp.337-341.

[Mitchell, 1997] Mitchell, T.: Machine Learning, McGraw-Hill, 1997.

[Morishita and Sese, 2000] Morishita, S., Sese, J.: Transversing itemset lattices with statistical metric pruning. In Proc. of the 19th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, 2000, pp.226-236.

[Parthasarathy et al, 2001] Parthasarathy, S., Zaki, M., Ogihara, M., Li, W.: Parallel data mining for association rules on shared memory systems. Journal Knowledge and Information Systems, 3/1, 2001, pp. 1-29.

[Parthasarathy, 2002] Parthasarathy, S.: Efficient progressive sampling for association rules. Proc. of Int. Conf. on Data Mining, 2002, pp.354-361.

[Quinlan and Cameron-Jones, 1993] Quinlan, J., Cameron-Jones, R.: FOIL: A midterm report. In Proc. of European Conf. On Machine Learning, Vienna, Austria, 1993, pp.3-20.

[Rak et al, 2005] Rak, R., Stach, W., Zaiane, O., Antonie M.-L.: Considering re-occurring features in associative classifiers. In Advances in Knowledge Discovery and Data Mining, LNCS, Vol. 3518, 2005, pp.240-248.

[Sarwar et al, 2001] Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based collaborative filtering recommendation algorithms. In: World Wide Web, 2001, pp.285-295.

[Savasere et al, 1995] Savasere, A., Omiecinski, E., Navathe, S.: An efficient algorithm for mining association rules in large databases. The 21st VLDB Conference, 1995, pp.

[Tang and Liao, 2007] Tang, Z., Liao, Q.: A new class based associative classification algorithm. IAENG International Journal of Applied Mathematics, 2007, 36/2, pp.15-19.

[Thabtah et al, 2005] Thabtah, F., Cowling, P., Peng, Y.: MCAR: multi-class classification based on association rule. In Proc. of the ACS/IEEE 2005 Int. Conf. on Computer Systems and Applications, Washington, DC, 2005, p.33.

[Toivonen, 1996] Toivonen, H.: Sampling large databases for association rules. In The VLDB Journal, 1996, pp.134-145.

[Wang and Karypis, 2005] Wang, J., Karypis, G.: HARMONY: Eficiently Mining the Best Rules for Classification. In Proc. of SDM, 2005, pp.205-216.

[Yin and Han, 2003] Yin, X., Han, J.: CPAR: Classification based on predictive association rules. In SIAM Int. Conf. on Data Mining (SDM'03), 2003, pp.331-335.

[Yuan and Huang, 2005] Yuan, Y., Huang, T.: A matrix algorithm for mining association rules. LNCS, Vol. 3644, 2005, pp.370-379.

[Zaiane and Antonie, 2002] Zaiane, O., Antonie, M.-L.: Classifying text documents by associating terms with text categories. J. Australian Computer Science Communications, 24/2, 2002, pp.215-222.

[Zaiane and Antonie, 2005] Zaiane, O., Antonie, M.-L.: On pruning and tuning rules for associative classifiers. In Proc. of Int. Conf. on Knowledge-Based Intelligence Information & Engineering Systems, LNCS, Vol. 3683, 2005, pp.966-973.

[Zaiane et al, 2000] Zaiane, O., Han, J., Zhu, H.: Mining recurrent items in multimedia with progressive resolution refinement. In Int. Conf. on Data Engineering, 2000, pp.461-470.

[Zimmermann and De Raedt, 2004] Zimmermann, A., De Raedt, L.: CorClass: Correlated association rule mining for classification. In Discovery Science, LNCS, Vol. 3245, 2004, pp.60-72.

Authors' Information

Ilia Mitov – *Institute of Mathematics and Informatics, BAS, Bulgaria; e-mail: mitov@foibg.com*

Major Fields of Scientific Research: *Multi-dimensional information systems; Data Processing and Mining*

Stefan Karastanev – *Institute of Mechanics, BAS, Bulgaria;*

e-mail: stefan@imbm.bas.bg

Major Fields of Scientific Research: *Software Engineering, Data Processing and Mining, Data structures in information systems.*

Krassimir Markov – *Institute of Information Theories and Applications, Bulgaria e-mail: markov@foibg.com*

Major Fields of Scientific Research: *General theoretical information research, Multi-dimensional information systems; Data Processing and Mining*