Growing Decision Trees On Support-Less Association Rules

Ke Wang
National University of Singapore
Simon Fraser University
wangk@cs.sfu.ca

Senqiang Zhou
Simon Fraser University
szhoua@cs.sfu.ca

Yu He
National University of Singapore
hey@comp.nus.edu.sg

Categories and Subject Descriptors
H.2.8 [**Information Systems**]: Database Management—*Database Applications*

General Terms
association rules, data mining, decision tree

1. INTRODUCTION
In *association mining*, first studied in database [1], the task is to find all association rules that are individually predictive in terms of high confidence. Association rules, first motivated by market-basket analysis, capture co-occurrence of events, such as coffee and sugar are often sold together. In *classification mining*, mainly studied in machine learning, statistic, pattern recognition, information retrieval (See [5, 14], for example), the task is to find a classifier, usually a set of rules, that are collectively qualified in terms of associating new cases with classes. The two problems have been largely investigated independently, despite their similarity of discovering association of some kind. Interesting questions arise: what issues must be addressed to turn association rules into a classifier? how does this approach compare to traditional classification methods? In this paper, we answer these questions and propose a general method for turning an arbitrary set of rules, in particular, association rules, into a classifier. In the past few years, many association rules and mining algorithms were proposed. However, the user often faces difficulties in making sense out of association rules. Indeed, no indication is given by association rules as to whether a specific but more confident rule or a general but less confident rule should be used to recommend products to new customers, and what hit rate a set of association rules will result in. Knowing such information is extremely important in a business decision making. Addressing such issues is the topic of this paper.

We take DT (decision tree) induction [5, 14], the state-of-the-arts, as a representative of traditional classification approaches. Table 1 gives a quick comparison between DT induction and association mining on a number of issues related to classification.

DT induction. DT induction performs a heuristic-based local search by appending promising attributes to rules in the order of goodness. Such one-attribute-at-a-time selection may diminish the typical structure that several attributes collectively determine the class. In addition, DT induction assumes an unnatural tree structure on the sharing of features among rules. The advantage of DT induction, however, is its systematic, accuracy-driven pruning of rules where the worth of rules is measured by their contribution to the overall accuracy of the classifier. Indeed, the exclusiveness of rules and the tree generalization hierarchy enable a systematic bottom-up pruning of DT without worrying about the interaction between rules. For more details about DT induction, please refer to [5, 14].

Association mining. The association mining searches globally for all rules according to the joint predictiveness of several attributes, i.e., confidence, and evaluates each rule individually without interaction of other rules. The result is the *full* set of rules (above the specified thresholds on support and confidence) that cover the training cases in all possible ways. The richness of rules gives this approach the potential of finding the true classification structure in the data. Unfortunately, this strength turns out to be a weakness when pruning overfitting rules because association rules are interrelated by covering common cases and by a non-tree generalization hierarchy. Recent association based classification [2, 4, 11, 13, 16] has used the minimum support to prune overfitting rules. This pruning suffers from the dilemma that rules of high support tend to have low confidence, but predictivity often depends on high confidence. Also, it is highly questionable to deal with overfitting by a user-specified minimum support. Often, the support requirement is unknown in advance, very small [6], and non-uniform across rules [15]. In these cases, an accuracy-driven pruning is more robust than a threshold-driven pruning.

Our approach. To use association rules for classification, the key is to prune overfitting rules on an accuracy-driven basis. We take two steps to achieve this goal. First, we abandon the ad-hoc minimum support and employ association rules satisfying only the minimum confidence, called

Features	DT	Association
rule generating	collectively	individually
search criterion on	attributes	rules
search strategy	local	global
rules	limited	rich
rule exclusiveness	yes	no
rule interaction	no	yes
generalization hierarchy	tree	lattice
pruning strategy	systematic accuracy-driven	ad-hoc threshold-based

Table 1: Comparison of DT approach and association approach

confident rules, to build a classifier. Rules with very little support are not statistically meaningful, and they will be pruned in the second step below. To find confident rules without examining all rules, we propose a confidence-based pruning that exploits a certain monotonicity of confidence (Section 2). Second, since confident rules tend to be specific, pruning overfitting rules is crucial for achieving a high accuracy. Our approach is to build a DT-like structure using association rules, and to leverage the accuracy-driven pruning of DT induction (Section 3). The resulting DT is called *ADT* (*association based decision tree*), and it combines the richness of association rules and the accuracy-driven pruning of DT induction.

2. MINING CONFIDENT RULES

The database is represented by a relational table T over m *non-class attributes* $A_1, \ldots, A_m$ and one *class attribute* C. A case has the form $< a_1, \ldots, a_m, c >$, where a_i are values of A_i and c is a class of C. A rule, or a k-rule, has the form $A_{i_1} = a_{i_1} \wedge \ldots \wedge A_{i_k} = a_{i_k} \rightarrow C = c$, with each attribute occurring at most once. By prefixing a value with its attribute, we can omit attributes and write a rule as $a_{i_1}, \ldots, a_{i_k} \rightarrow c$. We use x to denote one or more values. A case t and a rule $x \rightarrow c$ *match* if t contains all the values in x. A rule $x, a_i \rightarrow c$ is a *A_i-specialization* of $x \rightarrow c$ if a_i is a value of A_i. $|T|$ denotes the number of cases in T, and $num(x)$ denotes the number of cases in T that contain all the values in x. The *support* of rule $x \rightarrow c$, denoted $sup(x \rightarrow c)$, is $num(x,c)/|T|$. The *confidence* of rule $x \rightarrow c$, denoted $conf(x \rightarrow c)$, is $num(x,c)/num(x)$. Given a minimum confidence $minconf$, a rule is *confident* if $conf(x \rightarrow c) \geq minconf$.

DEFINITION 2.1 (MINING CONFIDENT RULES). The problem of *mining confident rules* is to find all confident rules for a given minimum confidence. □

Since mining confident rules does not require a minimum support, the classic support-based Apriori [1, 3] is not applicable. We must exploit a confidence-based pruning to prune unpromising rules as early as possible. This change turns out to be drastic. On the one hand, confidence no longer enjoys the downward closure as enjoyed by support-based Apriori: if being young and male together positively impacts buying the Internet service, $Age.young \rightarrow Buy.yes$ and $Gender.M \rightarrow Buy.yes$ could have lower confidence than $Age.young, Gender.M \rightarrow Buy.yes$. Consequently, even though shorter rules are not confident, longer rules may be. On the other hand, confidence does not enjoy the upward closure either because $Age.old, Gender.F \rightarrow Buy.yes$ could have lower confidence than $Age.old \rightarrow Buy.yes$ and $Gender.F \rightarrow Buy.yes$. Thus, a straightforward pruning based on the downward or upward closure does not work.

To motivate our confidence-based pruning, consider rules:

r1: $Age.young \rightarrow Buy.yes$
r2: $Age.young, Gender.M \rightarrow Buy.yes$
r3: $Age.young, Gender.F \rightarrow Buy.yes$.

Suppose that r1 has confidence of 0.60, that is, 60% of young people buy the Internet service. We can infer a lower bound on the confidence of at least one of r2 and r3 that specialize $r1$ using the exclusive conditions $Gender.M$ or $Gender.F$. The key observation is that, since the two conditions are mutually exclusive, if one condition impacts confidence negatively, the other condition must impact confidence positively. Consequently, at least one of r2 and r3 has as much confidence as r1. We can exploit this property to prune r1 if none of r2 and r3 is confident. Let us call this property the *existential upward closure*.

THEOREM 2.1 (EXISTENTIAL UPWARD CLOSURE). Consider any attribute A_i not in rule $x \rightarrow c$. (i) Some A_i-specialization of $x \rightarrow c$ has at least the confidence of $conf(x \rightarrow c)$. (ii) If $x \rightarrow c$ is confident, so is some A_i-specialization of $x \rightarrow c$.

The existential upward closure suggests the following level-wise search of confident rules. Assume that all confident k-rules are generated (starting with $k = m$, the number of non-class attributes). A candidate $(k-1)$-rule $x \rightarrow c$ is generated *only if* for *every* attribute A_i not in $x \rightarrow c$, some A_i-specialization of $x \rightarrow c$ is confident. By storing the confident k-rules in a relational table of k non-class columns, we can implement this rule generation using expressions in relational algebra. We omit the detail. To find whether generated candidate $(k-1)$-rules are actually confident, we scan the database (resident on disk for large databases) once to count their confidence. For efficient counting, candidates can be stored in a hash-tree as in [1]. For the rest of the paper, we assume that confident rules for a given $minconf$ are found. We focus on the problem of constructing a classifier using such rules.

3. FROM ASSOCIATION RULES TO A CLASSIFIER

Let $\mathcal{C}$ denote the set of confident rules plus the *default rule*, $\emptyset \rightarrow c$, where c is the majority class in T. To build a classifier from $\mathcal{C}$, the following questions must be answered. First, if a case matches more than one rule in $\mathcal{C}$, which of them should be used to cover the case? Second, are there any rules in $\mathcal{C}$ that are never used? If yes, how are they identified? To answer these questions, we define two binary relations for comparing rules. Let $size(r)$ denote the number of values in rule r, and let $lhs(r)$ denote the left-hand side of r. We

assume that the values in a rule are ordered lexicographically. Similarly, rules can be ordered lexicographically. For example, rule $A_1.2, A_2.1 \rightarrow 1$ precedes rule $A_2.1 \rightarrow 0$ lexicographically.

DEFINITION 3.1 (BINARY RELATIONS). Consider two rules r and r' in $\mathcal{C}$. We say that r is *ranked higher than* r', written as $r \prec_R r'$, if the following conditions hold: $conf(r) > conf(r')$; or if $conf(r) = conf(r')$, but $sup(r) > sup(r')$; or if $sup(r) = sup(r')$, but $size(r) < size(r')$; or if $size(r) = size(r')$, but r precedes r' in the lexicographical order of rules. We say that r is more *general than* r' (or r' is more *special than* r), written $r \prec_G r'$, if $lhs(r) \subseteq lhs(r')$. □

The following principle governs the preference of rules.

DEFINITION 3.2 (MCF PRINCIPLE). If there are choices, the rule of the highest rank has the priority. This is called the *most-confident-first principle (MCF principle)*. □

The rationale of the MCF principle is very simple: the most predictive rule has the priority. However, this tends to favor specific rules that often have high confidence. We will prune overfitting rules in the next section. To turn $\mathcal{C}$ into a classifier, we apply the MCF principle to resolve the conflict in covering a case: the *covering rule* of a given case is the rule that matches the case and has the highest possible rank. Note that each case has exactly one covering rule in $\mathcal{C}$. To classify a new case, the predicted class is the class in the covering rule of the case. In the following, we use the term *initial classifier* to refer to this classifier.

Under the MCF principle, a specific rule that does not have higher rank than all general rules is never used. This is because some general rule will match whatever cases the specific rule matches and has a higher rank. Clearly, such rules are redundant. From now on, we assume that redundant rules are removed from $\mathcal{C}$. This often cuts down substantially the number of rules because many specializations of rules do not improve the confidence. Here is the definition of redundant rules.

DEFINITION 3.3 (REDUNDANT RULES). A rule r in $\mathcal{C}$ is *redundant* if there is some rule r' in $\mathcal{C}$ that is more general and ranked higher than r, that is, $r' \prec_G r$ and $r' \prec_R r$. □

With only non-redundant rules, we can show that left-hand sides of rules are distinct, and thus, that the generalization relationship is a lattice over left-hand sides of rules. Suppose that two rules r and r' have the same left-hand side. By definition, $r \prec_G r'$ and $r' \prec_G r$. Since either $r \prec_R r'$ or $r' \prec_R r$, at least one of the two rules is redundant. Since we have removed all redundant rules, every rule in $\mathcal{C}$ must have a distinct left-hand side.

4. PRUNING THE CLASSIFIER

To boost the accuracy of the initial classifier, it is crucial to prune overfitting rules. Consider a rule r. If r is pruned, the MCF principle implies that the highest ranked general rule, say r', will cover the cases covered by r. In a sense, r' acts on behalf of r in case that r is pruned. A key to our pruning is to convert the lattice of association rules into a tree of such "acting relationship" between rules. This tree is called the ADT below.

DEFINITION 4.1 (ADT). Consider a non-default rule r in $\mathcal{C}$. The *parent* of r is the rule in $\mathcal{C}$ that is more general than r and has the highest possible rank. The *ADT (association based decision tree)* for $\mathcal{C}$ contains a node for each rule in $\mathcal{C}$ and an edge from a non-default rule to its parent. □

ADT is a tree with general rules at higher levels and specific rules at low levels, and the default rule at the root. To avoid pruning "good" general rules, i.e., over-pruning, we shall prune child rules before pruning parent rules. If child rules (and their subtrees) of a parent rule are pruned, their parent rule is made a leaf rule and covers all the cases previously covered by the rules in the subtrees at the child rules. Whether or not child rules are pruned, the pruning consideration is repeated at higher levels in a bottom-up manner until the root of ADT is considered. To decide whether to prune child rules, we compare the "estimated error" of ADT on new cases before the pruning with that after the pruning. To estimate the error, we adopt the *pessimistic* estimation for pruning DT [14]. The basic idea is to use the error of a rule observed in training cases to estimate the error on new cases. For the detail of pessimistic estimation, please refer to [14]. This method guarantees to produce a classifier of the minimum estimated error, with respect to the bottom-up pruning.

ADT differs from DT in several important ways. First, ADT is built from rules produced elsewhere, and the purpose of ADT is to prune overfitting rules. In principle, ADT is applicable to rules produced by any rule generator, not necessarily association rule mining. In contrast, DT induces rules by itself using the information gain or its variants. Second, ADT applies the MCF principle to select rules for classification, whereas a unique root-to-leaf path is followed in DT. Third, the most profound, ADT is a structure about the "acting relationship" between rules, whereas DT is a structure about rules (by storing values in rules along edges). Consequently, ADT does not impose the actual structure on rules. This decoupling of the rule structure from the relationship structure makes it possible to build ADT using externally generated rules such as association rules. A major benefit of this approach is the combination of the richness of association rules with the systematic pruning of DT induction.

5. EXPERIMENTS

We evaluate ADT using 21 datasets from UCI Repository [12]. We compare ADT with C4.5 [14], NB [7], TAN [9], CBA [11], and LB [13]. C4.5 is the classic state-of-the-arts of classification method. NB is a classifier based on Naive Bayes, which is reasonably accurate in most cases. TAN is an extension of NB and outperforms many Bayesian Network approaches. CBA uses association rules for classification. LB extends NB using long itemsets found by association mining. For all methods, the parameters are set to

Dataset	C4.5-con	C4.5-dis	TAN	NB	LB	CBA	ADT	Average
australia	0.85660	0.86520	0.84492	0.85942	0.86668*	0.84920	0.85507	0.85672
balance	0.56000	0.56000	0.64000	0.77760	0.77760	0.68320	0.79680*	0.68502
bridges	0.65720	0.67020		0.63236	0.63236	0.67100	0.68952*	0.56466
car	0.91660	0.91660	0.94260*	0.85568	0.88582	0.93820	0.92057$\surd$	0.91086
crx	0.85660*	0.85360		0.85072	0.85652	0.84220	0.85217$\surd$	0.73025
diabetes	0.73720	0.73700	0.73986	0.75030*	0.73594	0.72940	0.73856$\surd$	0.73832
flare	0.82340	0.82640	0.82629	0.80376	0.82348	0.80120	0.83004*	0.81922
glass	0.68080	0.69040	0.68095	0.69046	0.69046	0.70960	0.71428*	0.69385
iris	0.93320	0.93980*	0.92000	0.92666	0.92666	0.92000	0.92000	0.92661
monks-1	0.97840	0.97840	1.00000*	0.74596	1.00000*	1.00000*	1.00000*	0.95753
monks-2	0.61639	0.61639	0.62166	0.62666	0.62666	0.76340*	0.73000$\surd$	0.65731
monks-3	0.98920*	0.98920	0.98727	0.96364	0.96546	0.97120	0.98909$\surd$	0.97929
new-thyr	0.91620	0.93480	0.93953	0.94418	0.94418	0.94400	0.95348*	0.93948
nursery	0.96540	0.96540	0.93495	0.90340	0.94636	0.98120	0.98256*	0.95418
page-blo	0.96880*	0.96560	0.95374	0.93234	0.96142	0.95400	0.95210	0.95543
post-ope	0.68880	0.71100	0.68888	0.66668	0.66668	0.53340	0.71111*	0.66665
shuttle-s	0.99720	0.99580	0.99775*	0.99190	0.99690	0.99740	0.99706$\surd$	0.99628
tic-tac-toe	0.84080	0.84080	0.74345	0.70158	0.68900	0.99080*	0.97382$\surd$	0.82575
voting	0.96560*	0.96560		0.90517	0.93100	0.94040	0.94942$\surd$	0.80817
wine	0.89720	0.87420	0.98857	0.98858*	0.98858	0.91980	0.93142	0.94119
zoo	0.95000*	0.95000	0.94000	0.95000	0.95000	0.94000	0.94000	0.94571
Average	0.84740	0.84982	0.85502	0.83176	0.85056	0.86093	0.87748*	0.85328

Table 2: Accuracy

default values as suggested in the literature. For example, CBA is run with the minimum support of 0.5% plus the pruning option; for TAN, the smoothing factor is 5, etc. For our algorithm ADT, the minimum confidence, $minconf$, is 50%, and $minmeri$ is 10%.

We study the accuracy and the compactness of classifiers by the 5-fold cross validation. All measures are the average of the 5 folds. If a data set was pre-partitioned into the training set and testing set, we combine them before applying the 5-fold cross validation. The same fold partitioning is used across all algorithms. Since only C4.5 deals with continuous attributes, each fold is discretized (independently) using entropy discretization of the MLC++ system [10].

Accuracy. Table 2 shows the accuracy of the classifiers produced by the seven methods. "C4.5-con" stands for C4.5 applied without pre-discretization, and "C4.5-dis" stands for C4.5 applied with pre-discretization. Since TAN is not applicable to data sets with missing values, no result is obtained for such data sets. For each data set, the most accurate classifier is marked with *. The last row contains the average accuracy of each classifier, and the last column contains the average accuracy of each data set. From the table, ADT is superior in several ways. First, ADT scores the highest average accuracy, 2.8% and 3% higher than that of "C4.5-dis" and "C4.5-con", and 1.7% higher than that of the second best, CBA. Second, ADT scores the most number of *, i.e., the highest accuracy. Third, shown in the last two columns, for the 13 data sets for which ADT does not score *, ADT is still above the average for 8 of them, marked with $\surd$, and is close to the average for the other 5.

Size of classifiers. Table 3 shows the size of classifiers in terms of the number of rules. The size information is not available for LB, NB, and TAN. ADT produces the smallest classifier for many data sets. Interestingly, there seems to be a strong correlation between the datasets on which ADT produces the most accurate classifier and the datasets on which ADT produces the smallest classifier.

Additional experiments show that $minconf$ and $minmeri$ have a consistent and easily identifiable working range. We omit the detail here. In general, the phase of constructing ADT from confident rules is very fast, i.e., within seconds. The bottleneck is the phase of mining confident rules, especially for high dimensionality. A detailed performance study on mining confident rules will be reported elsewhere.

6. RELATED WORK

Classification and association were studied in isolation for many years until recently. [11] uses association rules to build a classifier and prunes rules using both the minimum support and the pessimistic estimation. In particular, [11] estimates the error of a rule based on the confidence and support of the rule, without regard to the interaction of other rules. In effect, this considers each case repeatedly for all covering rules. We deal with the rule interaction by the MCF principle so that rules have non-overlapping coverage of training cases, therefore, reflecting more truthfully the underlying prediction model. [8, 13] combines several association rules to classify a new case, thereby, partially addressing the "low support" of classification rules because a combined rule has a lower support. In [16], multi-level association rules are used to build hierarchical classifiers where both the class space and the feature space are organized into a taxonomy. All these works rely on a minimum support to prune specific rules. Finding association rules without a support require-

Dataset	C4.5-con	C4.5-dis	CBA	ADT	Average
australia	34.4	23.2*	129.2	43.8	57.6
balance-	35	35	117.2	22.2*	52.3
bridges	30.4*	30.4*	34.8	48.8	36
car	164.2	164.2	126.8*	194.4	162.4
crx	29.2	20.8*	127.2	49.2	56.6
diabetes	41	16*	38.8	16*	27.9
flare	2.6	5.2	28.8	1*	9.4
glass	42.2	27	25.8*	34.6	32.4
iris	7.8	4*	5.6	4.6	5.5
monks-1	44.4	44.4	22	15.4*	31.5
monks-2	21.8*	21.8*	117.8	111.8	68.3
monks-3	19	19	16.6	4.2*	14.7
new-thyr	11.4	12	11.4	11.2*	11.5
nursery	467.4	467.4	411.6	269.8*	404
page-blo	75.4*	128.8	202	146.2	138.1
post-ope	1.6	1*	23.4	1*	6.7
shuttle-s	27.8*	41	55.4	27.8	38
tic-tac-toe	119.8	119.8	26.8*	26.8*	73.3
voting	10.2*	10.2	31.8	25.2	19.3
wine	10.6	12.8	10.2*	15	12.1
zoo	17.8	17.8	8.6*	15.2	14.8
Average	57.8	58.1	74.8	51.6*	60.6

Table 3: Size of classifiers

ment is also studied in [6], but only rules between two values are considered. Such rules are too general for classification.

7. CONCLUSION

Association rules are rich, but lacking of a systematic method of pruning overfitting rules for classification. DT induction, in contrast, has an accuracy-driven pruning, but imposes restrictive structures on rules. To overcome the limitation of each, we proposed a method to build a DT from association rules, i.e., ADT. The advantage of ADT is the strength of both approaches. To give DT induction the full pruning power, we used all confident association rules without any support requirement. A confidence-based mining was proposed for finding all such rules. Experiments shown that the proposed ADT not only builds more accurate classifiers, but also does this by finding more truthful structures, as indicated by the smaller size of classifiers.

8. REFERENCES

[1] R. Agrawal, T. Imilienski, and A. Swami. Mining association rules between sets of items in large datasets. *SIGMOD*, 207–216, 1993

[2] K. Ali, S. Manganaris, and R. Srikant. Partial classification using association rules. *KDD*, 115–118, 1997

[3] R. Agrawal and R. Srikant. Fast algorithm for mining association rules. *VLDB*, 487-499, 1994

[4] R.J. Bayardo. Brute-force mining of high-confidence classification rules. *KDD*, 123-126, 1997

[5] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and regression trees.* Wadsworth, Belmont, CA, 1984

[6] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J.D. Ullman, and C . Yang. Finding interesting associations without support pruning. *ICDE*, 489–499, 2000

[7] R. Duda, and P. Hart. *Pattern classification and Scene analysis.* John Wiley & Sons, New York, 1973

[8] G. Dong, X. Zhang, L. Wong, and J. Li. CAEP: Classification by aggregating emerging patterns. *International Conference on Discovery Science*, 1999,

[9] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997

[10] R. Kohavi, G. John, R. Long, D. Manley, and K. Pfleger. MLC++: a machine learning library in C++. *Tools with Artificial Intelligence*, 740–743, 1994

[11] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. *KDD*, 80–86, 1998

[12] C.J. Merz, and P. Murphy. *UCI repository of machine learning databases*, 1996 (http://www.cs.uci.edu/m̃learn/MLRepository.html)

[13] D. Meretakis, and B. Wuthrich. Extending naive bayes classifiers using long itemsets. *SIGKDD*, 165–174, 1999

[14] J.R. Quinlan. *C4.5: programs for machine learning.* Morgan Kaufmann, 1993

[15] K. Wang, Y. He, J. Han. Pushing support constraints into frequent itemset mining. *VLDB*, 2000

[16] K. Wang, S.Q. Zhou, and S.C. Liew. Building hierarchical classifiers using class proximity. *VLDB*, 363–374, 1999