Direct Interesting Rule Generation

Jiuyong Li
Department of Mathematics and Computing
The University of Southern Queensland
Australia, 4350
jiuyong@usq.edu.au

Yanchun Zhang
School of Computer Science and Mathematics
Victoria University
Australia, 8001
yzhang@csm.vu.edu.au

Abstract

An association rule generation algorithm usually generates too many rules including a lot of uninteresting ones. Many interestingness criteria are proposed to prune those uninteresting rules. However, they work in post-pruning process and hence do not improve the rule generation efficiency. In this paper, we discuss properties of informative rule set and conclude that the informative rule set includes all interesting rules measured by many commonly used interestingness criteria, and that rules excluded by the informative rule set are forwardly prunable, i.e. they can be removed in the rule generation process instead of post pruning. Based on these properties, we propose a Direct Interesting rule Generation algorithm, DIG, to directly generate interesting rules defined by any of 12 interestingness criteria discussed in this paper. We further show experimentally that DIG is faster and uses less memory than Apriori.

1 Introduction

Data mining is a growing research area due to the increasing popularity of dealing with a large number of data in various fields. We can obtain benefits when understanding data in a meaningful form rather than a collection of tedious alphanumeric symbols.

Association rule discovery has been a central issue in data mining, because of the simplicity of the problem statement and the efficiency of pruning by support. However, an association rule generation algorithm may generate too many rules and selecting interesting rules is a big task.

Many interesting criteria are proposed to select user-interesting rules. However, they are usually used in the post pruning process hence do not improve the efficiency of rule generation.

In this paper, we prove that the informative rule set includes all interesting rules according to 12 interestingness criteria. We further propose a direct algorithm to generate interesting rules, and show experimentally that the proposed algorithm is faster and uses less memory than Apriori. DIG improves interesting rule discovery efficiency without more memory consumption and works on both transactional and relational data sets.

2 Informative rule set and its properties

An association rule set [1] is defined by the minimum support and the minimum confidence constraints. A lot of rules satisfying these constraints are not interesting. One major reason is that many rules are redundant. For example, $a \Rightarrow z$, $ab \Rightarrow z$ and $abc \Rightarrow z$ carry the similar message. Why do we need those complex ones having long antecedents? A complex rule covers a subset of instances covered by its simple form rules, so it has to provide something more than the simple form rules to be interesting. This extra provided by the complex rules is to be measured by an interestingness criterion. In the context of classic association rules, we expect that they have higher confidence than their simple form rules. Based on these observations, we define the informative rule set as follows.

Given a set of items $I = \{i_1, i_2, \ldots, i_m\}$, and a collection of transactions $D = \{T_1, T_2, \ldots, T_n\}$, where $T_i \subset I|_{i \leq n}$. D is called a transactional data set. An itemset is a subset of I, and is called a l-itemset if it contains l items. The support for itemset S is its occurrence probability in D, denoted by $P(S)$. $S \Rightarrow q$ is called a rule if $P(Sq) > \sigma$ [1] and $P(q|S) > P(q)$, where σ is called the minimum support and $P(q|S) = \frac{P(Sq)}{P(S)}$ is called the confidence of the rule. Given two rules $S \rightarrow i_q$ and $V \rightarrow i_q$ where $S \subset V$, we say the latter is more specific than the former or the former is more general than the latter.

[1]For simplicity, in the rest of this paper we use upper case letters, e.g. $S, V, Z,$ to stand for itemsets, and lower case letters, e.g. $a, b, c, p, q,$ to stand for items. We abbreviate $S \cup q$ as Sq.

Definition 1 We call a rule set the informative rule set if it satisfies the following two conditions: 1) it contains all rules that satisfy the minimum support; and 2) it excludes all more specific rules with no greater confidence than one of its more general form rule.

In the above definition, we did not specify the minimum confidence constraint. We suppose that a user may impose another minimum interestingness threshold since the discussion in this paper is to generate general interesting rules.

The informative rule set was defined in our previous work, and we concluded that it is the smallest subset of an association rule set having the same predictive power as the association rule set based on a confidence priority predictive model [12, 14]. In this paper, we reveal its general practical implication.

Theorem 1 *Rules excluded by the informative rule set are uninteresting measured by many interestingness criteria, namely odd ratio, lift (interest or strength), gain, added-value, Klosgen, conviction, p-s, Laplace, estimate accuracy, cosine, certainty factor and Jaccard.*

Proof In this proof, we use $AX \Rightarrow c$ to stand for a more specific form rule of $A \Rightarrow c$. $AX \Rightarrow c$ is excluded by the informative rule set because of $P(c|AX) \leq P(c|A)$. We will prove that by all criteria listed above, rule $AX \Rightarrow c$ is ranked lower than rule $A \Rightarrow c$ and hence is uninteresting. When two rules have the same value by an interesting metric, the shorter rule is ranked higher. We also know that $P(A) \geq P(AX)$.

Odds ratio is a classic statistical metric to measure the association between events. odd-ratio $(A \Rightarrow c)$ = $\frac{P(Ac)P(\neg A \neg c)}{P((\neg A)c)P(A\neg c)}$ (the definition of $\neg c$ can be referred to the paragraph before Lemma 1 in this section). We rewrite the odd ratio by support and confidence as follow: odd-ratio $(A \Rightarrow c)$ = $\frac{P(c|A)}{1-P(c|A)} / \frac{P(c|\neg A)}{1-P(c|\neg A)}$.

Since $P(c|A) \geq P(c|AX)$, $\frac{P(c|A)}{1-P(c|A)} \geq \frac{P(c|AX)}{1-P(c|AX)}$. Now we need to prove that $\frac{P(c|\neg A)}{1-P(c|\neg A)} \leq \frac{P(c|\neg(AX))}{1-P(c|\neg(AX))}$ to get odd-ratio$(A \Rightarrow c) \geq$ odd-ratio$(AX \Rightarrow c)$. To reach this goal, we need to prove that $P(c|\neg A) \leq P(c|\neg(AX))$ given $P(c|A) \geq P(c|AX)$.

$P(c|\neg A) = \frac{P((\neg A)c)}{P(\neg A)} = \frac{P(c)-P(Ac)}{1-P(A)} = \frac{P(c)/P(A)-P(c|A)}{1/P(A)-1} \leq \frac{P(c)/P(A)-P(c|AX)}{1/P(A)-1}$. Consider function $f(x) = \frac{\alpha x-\beta}{x-1}$ (β is a constant) monotonically increases with x when $\beta > \alpha$. We know that $\frac{1}{P(A)} \leq \frac{1}{P(AX)}$ and $P(c|AX) > P(c)$, so $\frac{P(c)/P(A)-P(c|AX)}{1/P(A)-1} \leq \frac{P(c)/P(AX)-P(c|AX)}{1/P(AX)-1} = P(c|\neg(AX))$. As a result, $P(c|\neg A) \leq P(c|\neg(AX))$.

Hence, odd-ratio$(A \Rightarrow c) \geq$ odd-ratio$(AX \Rightarrow c)$, and rule $A \Rightarrow c$ is ranked higher than rule $AX \Rightarrow c$ by odds ratio.

Lift [22], also known as interest [6] or strength [9], is a widely used metric to rank the interestingness of association rules. It has been used in IBM Intelligent Miner. lift$(A \Rightarrow c)$ = $\frac{P(Ac)}{P(A)P(c)} = \frac{P(c|A)}{P(c)}$. Consider $P(c|AX) \leq P(c|A)$. We have lift$(A \Rightarrow c)$ = $\frac{P(c|A)}{P(c)} \geq \frac{P(c|AX)}{P(c)}$ = lift$(AX \Rightarrow c)$. Hence, rule $A \Rightarrow c$ is ranked higher than rule $AX \Rightarrow c$ by the lift.

Gain [10] is an alternative for confidence. gain $(A \Rightarrow c)$ = $P(Ac) - \theta \times P(A)$ where θ is a fractional constant between 0 and 1, and only rules obtaining positive gain are interesting. We rewrite gain as the following: gain$(A \Rightarrow c)$ = $P(A)(P(c|A) - \theta)$. Consider $P(A) \geq P(AX)$ and $P(c|A) \geq P(c|AX) \geq \theta$ (otherwise both rules are uninteresting or answer is clear). We have gain$(A \Rightarrow c) \geq$ gain$(AX \Rightarrow c)$. Hence, rule $A \Rightarrow c$ is ranked higher than rule $AX \Rightarrow c$ by gain.

Two other similar metrics of gain are added-value and Klosgen. added-value $(A \Rightarrow c) = P(c|A) - P(c)$. Klosgen $(A \Rightarrow c)$ = $\sqrt{P(Ac)}(P(c|A) - P(c))$. We go through the similar proof procedure and draw the same conclusion. Rule $A \Rightarrow c$ is ranked higher than rule $AX \Rightarrow c$ by added value and Klosgen metrics.

Conviction [6] is used to measure deviations from the independence by considering outside negation. conviction $(A \Rightarrow c)$ = $\frac{P(A)P(\neg c)}{P(A\neg c)}$. We simplify the conviction as $(A \Rightarrow c)$ = $\frac{1-P(c)}{1-P(c|A)}$. Consider $P(c|A) \geq P(c|AX)$. We have conviction$(A \Rightarrow c) \geq$ conviction$(AX \Rightarrow c)$. Hence, rule $A \Rightarrow c$ is ranked higher than rule $AX \Rightarrow c$ by the conviction.

P-s metric [18] is a classic interesting criterion for rules and is proposed by Piatesky-Shaprio. p-s$(A \Rightarrow c) = P(Ac) - P(A)P(c)$. We rewrite it as the following: p-s$(A \Rightarrow c) = P(A)(P(c|A) - P(c))$. Consider $P(c|A) \geq P(c|AX)$ and $P(A) \geq P(AX)$. We have p-s$(A \Rightarrow c) \geq$ p-s$(AX \Rightarrow c)$. Hence, rule $A \Rightarrow c$ is ranked higher than rule $AX \Rightarrow c$ by the p-s metric.

Laplace [7, 21] accuracy is a metric for classification rules. Strictly speaking, it is not an interestingness metric for association rules. However, association rules have been used to solve classification problems. So we consider it still. Laplace$(A \Rightarrow c) = \frac{P(Ac)|D|+1}{P(A)|D|+k}$, where $|D|$ is the number of transactions in D and k is the number of classes. We rewrite it as follows: Laplace$(A \Rightarrow c)$ = $\frac{P(c|A)|D|+1/P(A)}{|D|+k/P(A)}$. Consider $P(c|A) \geq P(c|AX)$. We have Laplace$(A \Rightarrow c) \geq \frac{P(c|AX)|D|+1/P(A)}{|D|+k/P(A)}$. Function $f(x) = \frac{\alpha|D|+x}{|D|+kx}$ (α is a constant) monotonically decreases with x when $\alpha \times k > 1$. Usually, for classification rules the minimum confidence is set to be greater than 0.5 and $k \geq 2$. Hence $\alpha \times k > 1$ is satisfied. Consider $\frac{1}{P(A)} \leq$

$\frac{1}{P(AX)}$. We have Laplace($A \Rightarrow c$) $\geq \frac{P(c|AX)|D|+1/P(A)}{|D|+k/P(A)} \geq \frac{P(c|AX)|D|+1/P(AX)}{|D|+k/P(AX)}$ = Laplace($AX \Rightarrow c$). Hence, rule $A \Rightarrow c$ is ranked higher than rule $AX \Rightarrow c$ by laplace accuracy.

Estimate true accuracy [16] is used in [13] to suppress rules with a slight confidence improvement but a big support loss over their simple form rules. Accuracy ($A \Rightarrow c$) = $P(c|A) - z_N\sqrt{\frac{P(c|A)(1-P(c|A))}{P(Ac)|D|}}$, where $|D|$ is the number of transactions in D and z_N is a constant related with a statistical confidence interval. In classification, the minimum confidence of a rule is usually set to be greater than 0.5. In this case, accuracy($A \Rightarrow c$) $\geq$ accuracy($AX \Rightarrow c$) given $P(c|A) \geq P(c|AX)$ and $P(A) \geq P(AX)$. Hence, rule $A \Rightarrow c$ is ranked higher than rule $AX \Rightarrow c$ by the estimate accuracy.

We consider some other metrics discussed in [20].

Cosine($A \Rightarrow c$) = $\frac{P(Ac)}{\sqrt{P(A)P(c)}}$. We rewrite it as cosine($A \Rightarrow c$) = $\frac{\sqrt{P(A)}P(c|A)}{\sqrt{P(c)}}$. Consider that $P(AX) \leq P(A)$ and $P(c|AX) \leq P(c|A)$. We have cosine($A \Rightarrow c$) $\geq$ cosine($AX \Rightarrow c$). Hence, rule $A \Rightarrow c$ is ranked higher than rule $AX \Rightarrow c$ by cosine.

Certainty-factor ($A \Rightarrow c$) = $\frac{P(c|A)-P(c)}{1-P(c)}$. Consider $P(c|A) \geq P(c|AX)$. We have certainty-factor($A \Rightarrow c$) $\geq$ certainty-factor($AX \Rightarrow c$). Hence, rule $A \Rightarrow c$ is ranked higher than rule $AX \Rightarrow c$ by certainty factor.

Jaccard($A \Rightarrow c$) = $\frac{P(Ac)}{P(A)+P(c)-P(Ac)}$. We rewrite it as Jaccard($A \Rightarrow c$) $= \frac{P(c|A)}{1+P(c)/P(A)-P(c|A)}$. Consider $P(A) \geq P(AX)$. Jaccard($A \Rightarrow c$) $\geq \frac{P(c|A)}{1+P(c)/P(AX)-P(c|A)}$. Function $f(x) = \frac{x}{\alpha - x}$ (α is a constant.) monotonically increases with x when $\alpha > 0$. Consider $P(c|A) \geq P(c|AX)$ and $1 + P(c)/P(AX) > 0$. We have Jaccard($A \Rightarrow c$) $\geq \frac{P(c|A)}{1+P(c)/P(AX)-P(c|A)} \geq \frac{P(c|AX)}{1+P(c)/P(AX)-P(c|AX)}$ = Jaccard($AX \Rightarrow c$). Hence, rule $A \Rightarrow c$ is ranked higher than rule $AX \Rightarrow c$ by Jaccard.

The theorem is proved. □

Now we will present two properties of informative rule set to facilitate the design of an efficient rule generation algorithm. We do not provide proofs here, and please refer to [14] for details.

We introduce a special item $\neg q$, which appears in all transactions where q does not occur. Itemset $\neg(ab)$ means both a and b do not occur in a transaction, and hence $\neg(ab) = \neg a \neg b$. One obvious usefulness of this special item is to separate the support of an itemset into two parts, e.g. $P(S) = P(S\neg q) + P(Sq)$ and $P(S\neg q) = P(S) - P(Sq)$, or $P(q) = P(\neg Sq) + P(Sq)$ and $P(\neg Sq) = P(q) - P(Sq)$.

In the following lemma, we use $Sp \Rightarrow q$ to denote a more specific form rule of $S \Rightarrow q$. All more specific form rule of $Sp \Rightarrow q$ is $SZp \Rightarrow q$ for $Z \neq \emptyset$, $p, q \notin Z$ and $S \neq Z$.

Lemma 1 *If $P(S\neg q) = P(Sp\neg q)$, then for any item q rule $Sp \Rightarrow q$ and all its more specific form rules do not occur in the informative rule set.*

The meaning of this lemma is very clear. If a more specific rule r does not reduce negative instances from one of its more general form rule, then all more specific rules of r will not reduce negative instance from at least one of its more general form rules. Hence they are all excluded from the informative rule set. For example, if $P(a\neg q) = P(ab\neg q)$, then $P(q|ab) \leq P(q|a)$, further we also have $P(q|abX) \leq P(q|aX)$. $abX \Rightarrow q$ will not occur in the informative rule set because $aX \Rightarrow q$ is more general.

We have a look why this property is similar to the one that support has. Given itemset S is infrequent. SZ is infrequent because of upwards closure property of infrequent itemsets. This enables us to totally ignore all super itemsets of S in rule generation process. Given SZq is frequent. We usually have to test all rules from $S \Rightarrow q$ to $SZ \Rightarrow q$ where Z can be any itemset. However, if we observe that $P(S\neg q) = P(V\neg q)$ where V is a $(|S|-1)$-itemset and $V \subset S$, then to generate interesting rules we need not to test all rules from $S \Rightarrow q$ to $SZ \Rightarrow q$ according to the lemma.

Now we have a look at a useful corollary.

Corollary 1 *If $P(S) = P(Sp)$, then rule $Sp \Rightarrow q$ for any q and all its more specific form rules do not occur in the informative rule set.*

Both lemma and corollary are very useful for efficient algorithm design.

3 Direct interesting rule generation

An association rule generation algorithm prunes infrequent itemsets forwardly. An itemset is frequent if its support is greater than the minimum support. An itemset is potentially frequent only if all its subsets are frequent, and this property is used to limit the number of itemsets to be searched. This is called *upwards closure* property of infrequent itemset and is useful for forward pruning.

In the process of the association rule generation, confidence plays no role for the efficiency. It only controls the number of rules. It is well-known that an association rule generation algorithm produces too many uninteresting rules. According to the previous analysis, we know that those rules excluded by the informative rule set are uninteresting. In addition, we have upwards closure properties to forwardly prune those uninteresting rules. These enable us to design efficient algorithm to directly generate interesting rules.

3.1 Candidate representation

To facilitate the implementation of forward pruning by the lemma and the corollary, we define a rule candidate as a pair of (itemset, target-set), denoted by (S, C). The target-set C is a set of items that are possible consequences of rules. Initially, we set $S = C$. We call (S_1, C_1) as a sub candidate of (S_2, C_2) if $S_1 \subset S_2$. Equivalently, (S_2, C_2) is a super candidate of (S_1, C_1).

Target-set C is a sub or equal set of S and a candidate represents a number of potential rules. For example, candidate (abc, abc) indicates three potential rules $ab \Rightarrow c$, $ac \Rightarrow b$ and $bc \Rightarrow a$, and candidate (abc, ab) indicates two potential rules $ac \Rightarrow b$ and $bc \Rightarrow a$. Please note that we generate interesting single target rules as defined in the previous section.

The removal of items from target-set C is determined by the lemma, and we will show this in Subsection 3.3.

Usually, candidate $(S, \emptyset)$ is a legal candidate. We build a super candidate from its sub candidates. Though there is no potential rule for candidate $(abc, \emptyset)$, but there may be rule $abc \Rightarrow d$. Constructing candidate $(abcd, d)$ needs candidate $(abc, \emptyset)$.

Candidate $(S, \emptyset)$ is useless when no non-empty target-set super candidate can be built from it. Formally, given a set of items $I = \{i_1, i_2, \ldots, i_m\}$, itemset S and V where $V \neq S$. The target-set of candidate $(S, \emptyset)$ is permanently empty if there is no possibility to form rule $SV \Rightarrow p$, for all p and V where $p \in I \wedge p \notin SV$, to be in the informative rule set. This means that itemset S and all its super itemsets could not be the antecedent of an interesting rule.

The existence of a candidate depends on two conditions: (1) itemset S is frequent, and (2) target-set C is not permanently empty.

The determination of frequent S is straightforward, and hence we discuss how to determine of a permanently empty target-set.

The first criterion follows the corollary.

Criterion 1 *If $P(Sp) = P(S)$, then the target-set of candidate $(SZp, \emptyset)$ is permanently empty.*

We know that rule $Sp \Rightarrow q$ and all its more general form rules do not occur in the informative rule set according to the corollary. We note that q can be any item, so we need not to test rules from supersets of Sp.

This criterion is associated with a 100% confidence rule. If $P(Sp) = P(S)$, then $P(p|S) = 1$. When we know S always implies p, it is redundant to have $SZp \Rightarrow q$ since $SZ \Rightarrow q$ is more general.

For example, candidate $(abc, \emptyset)$ means rule $ab \Rightarrow c$, $ac \Rightarrow b$, $bc \Rightarrow a$ and their more specific rules are not in the informative rule set. For more details please refer to Subsection 3.3. However, for any $q \notin \{abc\}$ rule $abc \Rightarrow q$ and its more specific form rules may still be in the informative rule set and this is the reason for us to keep this candidate. If we know $sup(abc) = sup(ab)$, then according to the corollary rule $abc \Rightarrow q$ and all its more specific form rules are not in the informative rule set. Hence $(abc, \emptyset)$ is permanently empty.

We will present another criterion after presenting the candidate generation function.

3.2 Candidate generator

For easy understanding and comparison, we present Candidate-generator for the informative rule set discovery in the similar way of Apriori-gen. We call a candidate l-candidate if its itemset is a l-itemset. A l-candidate set includes all l-candidates. In the following discussions, we assume all items in an itemset are in the alphabetic order.

```
Function  Candidate-generator
      // Combining
1)  for each pair of candidates $(S_{l-1}p, C_p)$ and $(S_{l-1}q, C_q)$
            in $l$-candidate set
2)        insert candidate $(S_{l+1}, C)$
              where $S_{l+1} = S_{l-1}pq$ and $C = (C_pq) \cap (C_qp)$
              in the $(l+1)$-candidate set
      // Pruning
3)        for all $S_l \subset S_{l+1}$
4)        let $r = S_{l+1} \backslash S_l$
5)              if candidate $(S_l, C_l)$ does not exist
6)              then remove candidate $(S_{l+1}, C)$ and return
7)              else $C = C \cap (C_l r)$
8)        If the target-set of $(S_{l+1}, C)$ is permanently empty
9)        then remove the candidate
```

We first explain lines 1 and 2. Suppose that we have two candidates (abc, a) and (abd, ad). When we extend itemset $\{abc\}$ to itemset $\{abcd\}$, we should add item d to the target-set of candidate (abc, a) because it is new to itemset (abc). For the same reason, we add item c to the target-set of candidate (abd, ad). We then intersect two extended target-sets, $\{ad\}$ and $\{acd\}$, and put the intersection as the target-set of the new candidate. The new candidate is $(abcd, ad)$. The intersection of target-sets here and in line 7 is to ensure that removed items from the target-set of a candidate never appear in the target-set of its super candidates. The correctness is guaranteed by the lemma since any item removal in the target-set is determined by the lemma as shown in Subsection 3.3.

Then we explain line 3 to 9. Suppose that we have new candidate $(abcd, ad)$. It is the combination of (abc, a) and (abd, ad). We need to check if candidates identified by itemsets $\{acd\}$ and $\{bcd\}$ exist. Suppose that they do exist and are (acd, cd) and (bcd, b). Because item a

could not be the consequence of a rule including itemset $\{acd\}$, and item d could not be the consequence of a rule including itemset $\{bcd\}$, the new candidate looks like $(abcd, \emptyset)$. This is achieved by line 7, e.g. when $S_l = \{acd\}$, $C = \{ad\} \cap (\{cd\} \cup \{b\}) = \{d\}$, and when $S_1 = \{bcd\}$, $C = \{d\} \cap (\{b\} \cup \{a\}) = \emptyset$.

Last, we need to determine if a target-set is permanently empty. We presented Criterion 1 before, and here present another criterion.

Criterion 2 *Consider candidate $(Sp, \emptyset)$. If for every candidate (Sq, C_q) there is $q \notin C_q$, then the target-set of $(Sp, \emptyset)$ is permanently empty.*

We first examine the next level candidate. Based on line 2 in Candidate-generator, the new candidate must be $(Spq, \emptyset)$. We note "for all" condition in the criterion. We can obtain $(SZp, \emptyset)$ in a recursive way. Hence the target-set of $(Sp, \emptyset)$ is permanently empty.

Here is an example to show how Criterion 2 works. Suppose that there are only three candidates $(abc, \emptyset)$, (abd, a) and (abe, a) beginning with $\{ab\}$ in 3-candidate set. Since item d is not in the target-set of (abd, a) and item e is not in the target-set of (abe, a), $(abc, \emptyset)$ has a permanently empty target-set according to Criterion 2 and hence is removed. Consequently, all its super candidates will not be generated.

3.3 More pruning

We have a pruning process in candidate generation, and will have another pruning process after counting the support of candidates. This is a key issue to make use of the confidence for pruning and to have an efficient algorithm. In the following algorithm, σ is the minimum support, and $\{S \backslash c\}$ means set S less $\{c\}$.

```
Function   Prune($l+1$)
// $l+1$ is the new level where candidates are counted.
1)  for each candidate $(S, C)$ in $(l+1)$-candidate set
2)        if $P(S) \leq \sigma$ then remove candidate $(S, C)$
3)        else for each $c \in C$
                // test the satisfaction of the lemma
4)              if there is a sub candidate $(S', C')$
                    in $l$-candidate set such that
                    $c \in C'$ and $P(\{S \backslash c\} \neg c) = P(\{S' \backslash c\} \neg c)$
5)              then remove $c$ from $C$
6)        if $C$ is permanently empty then remove $(S, C)$
```

We prune a rule candidate from two aspects, the infrequency of the itemset and the permanently empty set of the target-set. In line 2 a candidate with infrequent itemset is removed. From line 3 to 6, we limit possible targets in the target-set of a candidate (a possible target is equivalent to a potential rule) by the lemma. In line 7 we consider removing a candidate when its target-set is empty.

For example, consider a candidate (abc, abc) in 3-candidate set. It includes three potential rules $ab \Rightarrow c$, $ac \Rightarrow b$ and $bc \Rightarrow a$. How can we omit some items from the target-set? The removal of an item in the target-set means that a potential rule and all its more specific rules are permanently removed. We do this according to the lemma. If we observe $P(ab\neg c) = P(a\neg c)$, then by the lemma rule $ab \Rightarrow c$ and all its more specific form rules are not in the informative rule set. As a result we remove item c from the target-set. The candidate should look like (abc, ab), and includes two potential rules, $bc \Rightarrow a$ and $ac \Rightarrow b$. The Candidate-generator ensures that item c never appears in target-sets of all super candidates of (abc, ab).

The determination of a permanently empty target-set has been introduced in the previous subsection, and hence we do not repeat them here. In our implementation, for convenience of applying Criterion 1, we call a candidate as *r*estricted candidate if the support of its itemset equals to that of one of its sub itemsets. This restricted status is inheritable because $P(Sp) = P(S) \Longrightarrow P(SZp) = P(SZ)$. A super candidate inherits this status from any sub candidates. According to criterion 1, the target-set of a restricted candidate is permanently empty if it is empty. In addition, a restricted candidate also inherits the target-set from its sub candidate. According to the corollary we could not form any rules to be in the informative rule set if their righthand side items are not in the target-set of the candidate where the restricted status is from. Therefore, there is no need to extend the target-set of a restricted candidate.

3.4 DIG algorithm

Now we are able to present our algorithm for the informative rule set generation.

```
Algorithm   DIG: Direct Interesting rule Generation
Input: data set $D$, the minimum support $\sigma$ and the minimum
interestingness threshold $\theta$.
Output: an interesting rule set $R$

1)  Set $R = \emptyset$
2)  Count support of 1-itemsets and 2-itemsets by arrays
3)  build 1 and 2-candidate sets
4)  Prune 1 and 2- candidate sets
5)  Select interesting rules to $R$
6)  Generate 3-candidate set
7)  While new candidate set is not empty
8)        Count support of itemsets for new candidates
9)        Prune candidates in the new candidate set
10)       Select interesting rules to $R$
11)       Generate next level candidate set
12)       Return rule set $R$
```

The minimum interestingness threshold θ can be set by any of 12 interestingness criteria discussed in Section 2. Two main functions are discussed in the previous subsections.

In the above algorithm, we use rule $\emptyset \Rightarrow c$ to prune 1-candidates. For example, if 80% customers buy bread when they shop in a supermarket, then rule, egg $\Rightarrow$ bread, with 75% confidence states nothing new. Hence the rule is uninteresting. Rule $\emptyset \Rightarrow$ bread can prune those uninteresting rules. One may argue that that rule egg $\Rightarrow$ bread is interesting if it has very low confidence, say 20%. In this case, we may formulate the rule as egg $\Rightarrow \neg$ bread, and call the rule as a negative rule. Interestingness of negative rules is beyond the discussions of this paper.

The correctness of this algorithm is guaranteed by the completeness of enumerating all itemsets by the first two lines of Candidate-generator and accurate forward pruning uninteresting rules by Lemma 1 and Corollary 1. Its time complexity is determined by the number of candidates, and so is its memory usage.

Please note line 9 in function Candidate-generator and line 6 in function Prune, not only are all super candidates of a candidate with the infrequent itemset removed, but also all super candidates of a candidate with a permanently empty target-set. Accordingly, DIG does not generate all frequent itemsets.

DIG is more efficiency than an association rule generation algorithm and its memory usage is smaller because it uses confidence to prune uninteresting rules in addition to the support pruning.

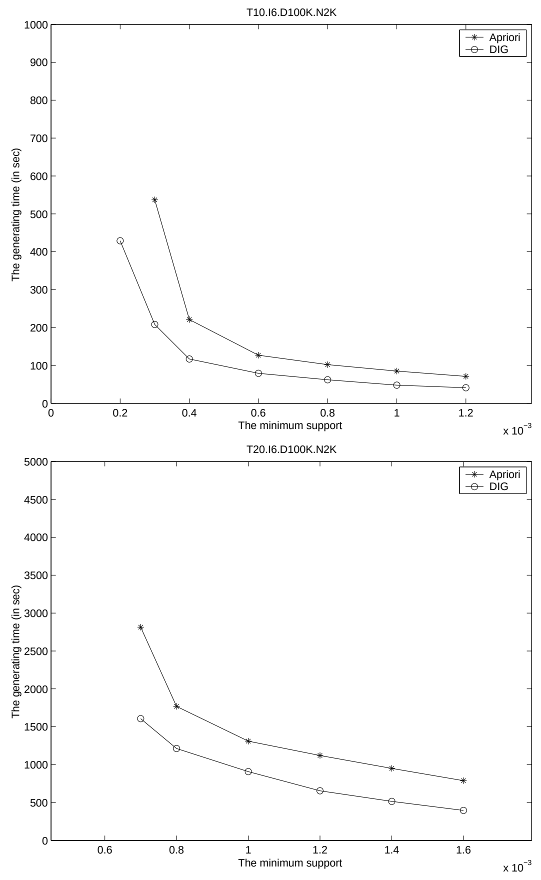

Figure 1. The comparison of time efficiency

4 Experimental results

In this section, we show the efficiency of DIG in the comparison with a well-known association rule generation algorithm, Apriori [1]. Though some association rules algorithms [11, 19, 24] are faster than Apriori, they usually use more memory to trade for faster speed. We note that in most applications, an association rule generation algorithm fails because it runs out of the computer memory. So we stick with Apriori, and will compare memory usage with Apriori.

Two test transactional data sets, T10.I6.D100K.N2K and T20.I6.D100K.N2K, are generated by the synthetic data generator from QUEST of IBM Almaden research center. Two data sets contain 1000 items and 100,000 transactions each. Our experiments were conducted on a Dell computer with 2G memory and a 2.4GHZ Intel processor running Red Hat Linux 7.3. Apriori is implemented with the storage structure of prefix tree and so is DIG.

We first examine the time efficiency of DIG. It was compared against Apriori. The rule generation time of DIG and Apriori are listed in Figure 1. We can see that DIG is faster than Apriori and the efficiency improvement is more significant when the support is low.

To demonstrate DIG improve efficiency without any additional memory consumption, we list the number of total candidates for different rule sets in Figure 2. We can see that DIG uses less memory than Apriori does. We also note that the time efficiency improvement is consistent with the candidate reduction. DIG searches less candidates for interesting rues. This is a major reason for efficiency improvement.

When we apply DIG to relational data sets, the efficiency improvement is much more significant than to transactional data sets. This is because a relational data set is denser than a transactional data set and the lemma and corollary are more efficient in forward pruning. Importantly, DIG enables us to generate interesting rules in relational data with lower support than Apriori does.

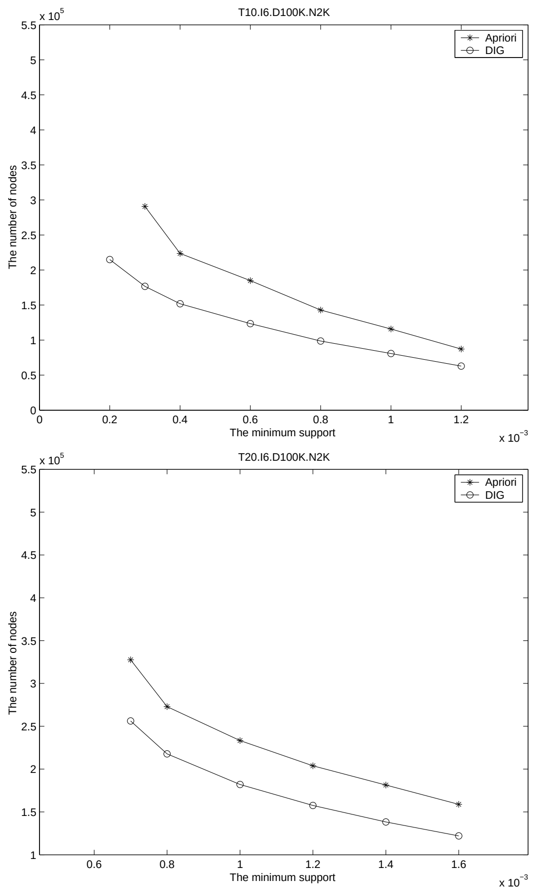

Figure 2. The comparison of memory efficiency

5 Related work

Association mining [1] has been studied for many years. Most research work has been on how to mine frequent itemsets efficiently since it is the base for association rule forming. Apriori [2] is a widely accepted approach. There are many other, like [17, 11, 19, 24]. Most of them use more memory to trade for faster speed. A comparison research was conducted in [25].

Many association rule generation algorithms undergo two stages. They first generate all frequent itemsets and then form rules among them. At the second stage, a large amount of rules are generated, and many of them are uninteresting. Much research focuses on how to select interesting rules, and many interestingness criteria are proposed, such as lift [22] (interest [6] or strength [9]), gain [10], conviction [6], p-s [18], and Laplace [7, 21]. A recent study on interestingness criteria is reported in [20]. Actually, all interestingness criteria only decrease the number of rules but do not increase the efficiency of rule mining.

Direct interesting rule generation algorithms, which do not generate all frequent itemsets first, can improve efficiency of association rule mining. A direct algorithm usually utilizes additional forward pruning besides support pruning. Closed itemset property, Chi-square and confidence (or laplace accuracy) have been used for such pruning.

Non-redundant association rule set generation [23], presented by M. Zaki, utilizes closed itemset properties for forward pruning. This pruning alone is not very efficient for rule generation on transactional data. Its effect is similar to the corollary in this paper, and we characterized the relationship between the non-redundant association rule set and the informative rule set in [14].

Upward closure property of chi-square was used to generate correlated association rule set [5] by S. Brin et al. This rule set is very meaningful and reveals correlation betweens items. However, both complexity and inaccuracy of chi-square test increase when the number of cells in a contingency table grows. In [8], a simplified chi-squared criterion is used, but generated rules are restricted to those with 1-itemset antecedents.

Confidence based pruning is used by R. Bayardo et al for generating constraint rule set [4] and optimality rule sets [3]. However, practical implication of these rule sets needs further clarification. Further, the presented rule generation algorithms are unsuitable for applications without constraint targets or with a large number of targets since they focus only on one fixed consequence at one time.

OPUS algorithms proposed by G. Webb to systematically search for rule sets also utilize a variant confidence, Laplace accuracy, for pruning[21]. These "optimal rule sets" are quite different from association rule based optimal rule sets since rules in an OPUS based rule set are generated in a AQ-like covering algorithm [15]. An OPUS algorithm scans a data set at least as many times as the number of rules. A modified OPUS algorithm [22] to generate association rules scans a data set as many times as the number of different antecedents in the generated rule set. As a result, it may not be efficient when a data set cannot be retained in the main memory.

This work is very similar to the above two studies in terms of confidence based pruning. However, the well-defined informative rule set has clear practical implication. The presented algorithm generates all rules with respect to all possible consequences once, and scans a data set only as many times as the length of the longest rule in the generated rule set. The algorithm presented in this paper works on both transactional and relational data sets, whereas algorithms of above two studies handle only relational data sets.

6 Conclusions

In this paper, we discussed properties of the informative rule set. We proved that all rules excluded by the informative rule set are uninteresting by 12 interestingness criteria and that excluded rules are forwardly prunable. These properties enabled us to design an efficient algorithm, DIG, to directly generate interesting rules. We showed experimentally that DIG is faster and uses less memory than Apriori. DIG avoids time consuming post pruning that an algorithm to generate interesting rules through an association rule set has to undergo.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining associations between sets of items in massive databases. In *Proc. of the ACM SIGMOD Int'l Conference on Management of Data*, pages 207–216, 1993.

[2] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the Twentieth International Conference on Very Large Databases*, pages 487–499, Santiago, Chile, 1994.

[3] R. Bayardo and R. Agrawal. Mining the most interesting rules. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 145–154, N.Y., 1999. ACM Press.

[4] R. Bayardo, R. Agrawal, and D. Gunopulos. Constraint-based rule mining in large, dense database. In *Proc. of the 15th Int'l Conf. on Data Engineering*, pages 188–197, 1999.

[5] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 26(2):265, 1997.

[6] S. Brin, R. Motwani, J. D. Ullman, and S. Tsur. Dynamic itemset counting and implication rules for market basket data. In *Proceedings, ACM SIGMOD International Conference on Management of Data: SIGMOD 1997: May 13–15, 1997, Tucson, Arizona, USA*, volume 26(2), pages 255–264, NY, USA, 1997. ACM Press.

[7] P. Clark and R. Boswell. Rule induction with CN2: Some recent improvements. In *Machine Learning - EWSL-91*, pages 151–163, 1991.

[8] E. Cohen, M. Datar, S. Fujiwara, A. Gionis, P. Indyk, R. Motwani, J. Ullman, and C. Yang. Finding interesting associations without support pruning. In *16th International Conference on Data Engineering (ICDE' 00)*, pages 489–500, Washington - Brussels - Tokyo, 2000. IEEE.

[9] V. Dhar and A. Tuzhilin. Abstract-driven pattern discovery in databases. *IEEE Transactions on Knowledge and Data Engineering*, 5(6), 1993.

[10] T. Fukuda, Y. Morimoto, S. Morishita, and T. Tokuyama. Data mining using two-dimensional optimized association rules: scheme, algorithms, and visualization. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4–6, 1996*, pages 13–23, New York, 1996. ACM Press.

[11] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. 2000 ACM-SIGMOD Int. Conf. on Management of Data (SIGMOD'00)*, pages 1–12, May, 2000.

[12] J. Li, H. Shen, and R. Topor. Mining the smallest association rule set for prediction. In *Proceedings of 2001 IEEE International Conference on Data Mining (ICDM 2001)*, pages 361–368. IEEE Computer Society Press, 2001.

[13] J. Li, H. Shen, and R. Topor. Mining the optimal class association rule set. *Knowledge-Based System*, 15(7):399–405, 2002.

[14] J. Li, H. Shen, and R. Topor. Mining informative rule set for prediction. *Journal of intelligent information systems*, in press.

[15] R. Michalski, I. Mozetic, J. Hong, and N. Lavrac. The AQ15 inductive learning system: an overview and experiments. In *Proceedings of IMAL 1986*, Orsay, 1986. Université de Paris-Sud.

[16] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[17] J. S. Park, M. Chen, and P. S. Yu. An effective hash based algorithm for mining association rules. In *ACM SIGMOD Intl. Conf. Management of Data*, 1995.

[18] G. Piatetsky-Shapiro. Discovery, analysis and presentation of strong rules. In G. Piatetsky-Shapiro, editor, *Knowledge Discovery in Databases*, pages 229–248. AAAI Press / The MIT Press, Menlo Park, California, 1991.

[19] P. Shenoy, J. R. Haritsa, S. Sudarshan, G. Bhalotia, M. Bawa, and D. Shah. Turbo-charging vertical mining of large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD-99)*, ACM SIGMOD Record 29(2), pages 22–33, Dallas, Texas, 1999. ACM Press.

[20] P. Tan, V. Kumar, and J. Srivastava. Selecting the right interestingness measure for association patterns. In *Proceedings of the eighth ACMKDD international conference on knowledge discovery and data mining*, pages 32 – 41, Edmonton, Canada, 2002. ACM press.

[21] G. I. Webb. OPUS: An efficient admissible algorithm for unordered search. In *Journal of Artificial Intelligence Research*, volume 3, pages 431–465, 1995.

[22] G. I. Webb. Efficient search for association rules. In *Proceedinmgs of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-00)*, pages 99–107, N. Y., 2000. ACM Press.

[23] M. J. Zaki. Generating non-redundant association rules. In *6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 34–43, August 2000.

[24] M. J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. New algorithms for fast discovery of association rules. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, page 283. AAAI Press, 1997.

[25] Z. Zheng, R. Kohavi, and L. Mason. Real world performance of association rule algorithms. In *Proceedings of the seventh International Conference on Knowledge Discovery and Data Mining (SIGKDD-01)*, 2001.