Finding Sporadic Rules Using Apriori-Inverse

Yun Sing Koh and Nathan Rountree

Department of Computer Science, University of Otago, New Zealand
{ykoh, rountree}@cs.otago.ac.nz

Abstract. We define sporadic rules as those with low support but high confidence: for example, a rare association of two symptoms indicating a rare disease. To find such rules using the well-known Apriori algorithm, minimum support has to be set very low, producing a large number of trivial frequent itemsets. We propose "Apriori-Inverse", a method of discovering sporadic rules by ignoring all candidate itemsets above a maximum support threshold. We define two classes of sporadic rule: perfectly sporadic rules (those that consist only of items falling below maximum support) and imperfectly sporadic rules (those that may contain items over the maximum support threshold). We show that Apriori-Inverse finds all perfectly sporadic rules much more quickly than Apriori. We also propose extensions to Apriori-Inverse to allow us to find some (but not necessarily all) imperfectly sporadic rules.

1 Introduction

Association rule mining has become one of the most popular data exploration techniques, allowing users to generate unexpected rules from "market basket" data. Proposed by Agrawal et al. [1, 2], association rule mining discovers all rules in the data that satisfy a user-specified minimum support (minsup) and minimum confidence (minconf). Minsup represents the minimum amount of evidence (that is, number of transactions) we require to consider a rule valid, and minconf specifies how strong the implication of a rule must be to be considered valuable.

The following is a formal statement of association rule mining for transactional databases. Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items and D be a set of transactions, where each transaction T is a set of items such that $T \subseteq I$. An association rule is an implication of the form $X \rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \emptyset$. X is referred to as the *antecedent* of the rule, and Y as the *consequent*. The rule $X \rightarrow Y$ holds in the transaction set D with *confidence* $c\%$ if $c\%$ of transactions in D that contain X also contain Y. The rule $X \rightarrow Y$ has *support* of $s\%$ in the transaction set D, if $s\%$ of transactions in D contain $X \cup Y$ [2]. One measure of the predictive strength of a rule $X \rightarrow Y$ is its *lift* value, calculated as confidence($X \rightarrow Y$) / support(Y). Lift indicates the degree to which Y is more likely to be present when X is present; if lift is less than 1.0, Y is less likely to be present with X than Y's baseline frequency in D. The task of generating association rules is that of generating all rules that meet minimum

© Springer-Verlag Berlin Heidelberg 2005

support and minimum confidence, and perhaps meet further requirements such as having lift greater than 1.0.

The Apriori algorithm and its variations are used widely as association rule mining methods. However, several authors have pointed out that the Apriori algorithm, by definition, hinders us from finding rules with low support and high confidence [3, 4, 5]. Apriori generates *frequent* itemsets (i.e. those that will produce rules with support higher than minsup) by joining the frequent itemsets of the previous pass and pruning those subsets that have a support lower than minsup [2]. Hence, to generate rules that have low support, minsup must be set very low, drastically increasing the running time of the algorithm. This is known as the *rare item problem*. It means that, using the Apriori algorithm, we are unlikely to generate rules that may indicate events of potentially dramatic consequence. For example, we might miss out rules that indicate the symptoms of a rare but fatal disease due to the frequency of incidences not reaching the minsup threshold. Some previous solutions to this problem are reviewed in Section 2.

The aim of our research is to develop a technique to mine low support but high confidence rules effectively. We call such rules "sporadic" because they represent rare cases that are scattered sporadically through the database but with high confidence of occurring together. In order to find sporadic rules with Apriori, we have to set a very low minsup threshold, drastically increasing the algorithm's running time. In this paper, we adopt an Apriori-Inverse approach: we propose an algorithm to capture rules using a *maximum* support threshold. First, we define the notion of a *perfectly* sporadic rule, where the itemset forming the rule consists only of items that are all below the maximum support threshold. To enable us to find *imperfectly* sporadic rules, we allow maximum support to be increased slightly to include itemsets with items above maximum support. Finally, we demonstrate that Apriori-Inverse lets us find sporadic rules more quickly than using the Apriori algorithm.

2 Related Work

The most well-known method for generating association rules is the Apriori algorithm [2]. It consists of two phases: the first finds itemsets that satisfy a user-specified minimum support threshold, and the second generates association rules that satisfy a user-specified minimum confidence threshold from these "frequent" itemsets. The algorithm generates all rules that satisfy the two thresholds and avoids generating itemsets that do not meet minimum support, even though there may be rules with low support that have high confidence. Thus, unless minimum support is set very low, sporadic rules will never be generated. There are several proposals for solving this problem. We shall discuss the MSApriori (Multiple Supports Apriori), RSAA (Relative Support Apriori Algorithm) and Min-Hashing approaches.

Liu et al. [4] note that some individual items can have such low support that they cannot contribute to rules generated by Apriori, even though they may participate in rules that have very high confidence. They overcome this problem with a technique whereby each item in the database can have a minimum item support (MIS) given by the user. By providing a different MIS for different items, a higher minimum support is tolerated for rules that involve frequent items and lower minimum support for rules that involve less frequent items. The MIS for each data item i is generated by first specifying LS (the lowest allowable minimum support), and a value $\beta, 0 \leq \beta \leq 1.0$. MIS($i$) is then set according to the following formula:

$$\begin{aligned} M(i) &= \beta \times f \quad (O \leq \beta \leq 1) \\ MIS &= M(i), \text{if}(M(i) > LS) \\ &= LS, \text{otherwise} \end{aligned}$$

The advantage of the MSApriori algorithm is that it has the capability of finding some rare-itemset rules. However, the actual criterion of discovery is determined by the user's value of β rather than the frequency of each data item. Thus Yun et al. [5] proposed the RSAA algorithm to generate rules in which significant rare itemsets take part, without any "magic numbers" specified by the user. This technique uses *relative support*: for any dataset, and with the support of item i represented as $sup(i)$, relative support ($RSup$) is defined as:

$$\begin{aligned} RSup\{i_1, i_2, \ldots, i_k\} = max(\ & sup(i_1, i_2, \ldots, i_k)/sup(i_1), \\ & sup(i_1, i_2, \ldots, i_k)/sup(i_2), \\ & \ldots, \\ & sup(i_1, i_2, \ldots, i_k)/sup(i_k)) \end{aligned}$$

Thus, this algorithm increases the support threshold for items that have low frequency and decreases the support threshold for items that have high frequency. Like Apriori and MSApriori, RSAA is exhaustive in its generation of rules, so it spends time looking for rules which are not sporadic (i.e. rules with high support and high confidence). If the minimum-allowable relative support value is set close to zero, RSAA takes a similar amount of time to that taken by Apriori to generate low-support rules in amongst the high-support rules.

Variations on Min-Hashing techniques were introduced by Cohen [3] to mine significant rules without any constraint on support. Transactions are stored as a 0/1 matrix with as many columns as there are unique items. Rather than searching for pairs of columns that would have high support or high confidence, Cohen et al. search for columns that have high *similarity*, where similarity is defined as the fraction of rows that have a 1 in both columns when they have a 1 in either column. Although this is easy to do by brute-force when the matrix fits into main memory, it is time-consuming when the matrix is disk-resident. Their solution is to compute a hashing signature for each column of the matrix in such a way that the probability that two columns have the same signature is proportional to their similarity. After signatures are calculated, candidate pairs are generated, and then finally checked against the original matrix to ensure that they do indeed have strong similarity.

It should be noted that, like MSApriori and RSAA above, the hashing solution will produce many rules that have high support and high confidence, since only a *minimum* acceptable similarity is specified. It is not clear that the method will extend to rules that contain more than two or three items, since ${}^{m}C_{r}$ checks for similarity must be done where m is the number of unique items in the set of transactions, and r is the number of items that might appear in any one rule. Removing the support requirement entirely is an elegant solution, but it comes at a high cost of space: for n transactions containing an average of r items over m possible items, the matrix will require $n \times m$ bits, whereas the primary data structure for Apriori-based algorithms will require $n \times \log_2 m \times r$ bits. For a typical application of $n = 10^9$, $m = 10^6$ and $r = 10^2$, this is 10^{15} bits versus approximately 2×10^{12} bits.

For our application, we are interested in generating *only* sporadic rules, without having to wade through a lot of rules that have high support (and are therefore not sporadic), without having to generate any data structure that would not normally be generated in an algorithm like Apriori, and without generating a large number of *trivial* rules (e.g. those rules of the form A → B where the support of B is very high and the support of A rather low). In the next section, we propose a framework for finding certain types of sporadic rules.

3 Proposal of Apriori-Inverse

In the previous section, the techniques discussed generate all rules that have high confidence and support. Using them to find sporadic rules would require setting a low minsup. As a result, the number of rules generated can be enormous, with only a small number being significant sporadic rules. In addition, not all rules generated with these constraints are interesting. Some of the rules may correspond to prior knowledge or expectation, refer to uninteresting attributes, or present redundant information [6].

3.1 Types of Sporadic Rule

We refer to all rules that fall *below* a user-defined *maximum* support level (maxsup) but *above* a user-defined minimum confidence level (minconf) as *sporadic rules.* We further split sporadic rules into those that are *perfectly sporadic* (have no subsets above maxsup) and those that are *imperfectly sporadic.* We then demonstrate an algorithm, which we call Apriori-Inverse, that finds all perfectly sporadic rules.

Definition:

$A \rightarrow B$ is *perfectly sporadic* for maxsup s and minconf c iff

$$\begin{aligned} &\text{confidence}(A \rightarrow B) \geq c, \quad \text{and} \\ &\forall x : x \in (A \cup B), \quad \text{support}(x) < s \end{aligned}$$

That is, support must be *under* maxsup and confidence *at least* minconf, and no member of the set of $A \cup B$ may have support above maxsup. Perfectly sporadic rules thus consist of antecedents and consequents that occur rarely (that is, less often than maxsup) but, when they do occur, tend to occur together (with at least minconf confidence).

While this is a useful definition of a particularly interesting type of rule, it certainly does not cover all cases of rules that have support lower than maxsup. For instance, suppose we had an itemset $A \cup B$ with support(A) $= 12\%$, support(B) $= 16\%$, and support($A \cup B$) $= 12\%$, with maxsup $= 12\%$ and minconf $= 75\%$. Both $A \to B$ (confidence $= 100\%$) and $B \to A$ (confidence $= 75\%$) are sporadic in that they have low support and high confidence, but neither are *perfectly* sporadic, due to B's support being too high. Thus, we define *imperfectly* sporadic rules as the following:

Definition:

$A \to B$ is *imperfectly sporadic* for maxsup s and minconf c iff

$$\begin{array}{l} \text{confidence}(A \to B) \geq c, \quad \text{and} \\ \text{support}(A \cup B) < s, \quad \text{and} \\ \exists x : x \in (A \cup B), \quad \text{support}(x) \geq s \end{array}$$

That is, a rule is imperfectly sporadic if it meets the requirements of maxsup and minconf but has a subset of its constituent itemsets that has support above maxsup. Clearly, some imperfectly sporadic rules could be completely trivial or uninteresting: for instance, when the antecedent is rare but the consequent has support of 100%. What we should like is a technique that finds all perfectly sporadic rules and some of the imperfectly sporadic rules that are *nearly* perfect.

3.2 The Apriori-Inverse Algorithm

In this section, we introduce the Apriori-Inverse algorithm. Like Apriori, this algorithm is based on a level-wise search. On the first pass through the database, an inverted index is built using the unique items as keys and the transaction IDs as data. At this point, the support of each unique item (the 1-itemsets) in the database is available as the length of each data chain. To generate k-itemsets under maxsup, the $(k-1)$-itemsets are extended in precisely the same manner as Apriori to generate candidate k-itemsets. That is, a $(k-1)$-itemset i_1 is turned into a k-itemset by finding another $(k-1)$-itemset i_2 that has a matching prefix of size $(k-2)$, and attaching the last item of i_2 to i_1. For example, the 3-itemsets $\{1,3,4\}$ and $\{1,3,6\}$ can be extended to form the 4-itemset $\{1,3,4,6\}$, but $\{1,3,4\}$ and $\{1,2,5\}$ will not produce a 4-itemset due to their prefixes not matching right up until the last item.

These candidates are then checked against the inverted index to ensure they at least meet a *minimum absolute support* requirement (say, at least 5 instances) and are pruned if they do not (the length of the *intersection* of a data chain in the inverted index provides support for a k-itemset with k larger than 1). The process continues until no candidate itemsets can be generated, and then association rules are formed in the usual way.

It should be clear that Apriori-Inverse finds all perfectly sporadic rules, since we have simply inverted the downward-closure principle of the Apriori algorithm;

rather than all subsets of rules being over minsup, all subsets are under maxsup. Since making a candidate itemset longer cannot increase its support, all extensions are viable *except* those that fall under our minimum absolute support requirement. Those exceptions are pruned out, and are not used to extend itemsets in the next round.

```
Algorithm Apriori-Inverse
Input: Transaction Database $D$, maxsup value
Output: Sporadic Itemsets

(1) Generate inverted index $I$ of (item, [TID-list]) from $D$.
(2) Generate sporadic itemsets of size 1:
     $S_1 = \emptyset$
     for each item $i \in I$ do begin
         if count($I, i$)/|$D$| $<$ maximum support and
            count($I, i$) $>$ minimum absolute support
         then $S_1 = S_1 \cup i$
     end
(3) Find $S_k$, the set of sporadic $k$-itemsets where $k \geq 2$:
     for ($k = 2$; $S_{k-1} \neq \emptyset$; $k++$) do begin
         $S_k = \emptyset$
         for each $i \in \{$itemsets that are extns of $S_{k-1}\}$ do begin
            if all subsets of $i$ of size $k-1 \in S_{k-1}$
            and count($I, i$) $>$ minimum absolute support
            then $S_k$ = $S_k \cup i$
         end
     end
     return $\bigcup_k S_k$
```

Apriori-Inverse does not find any imperfectly sporadic rules, because it never considers itemsets that have support above maxsup; therefore, no subset of any itemset that it generates can have support above maxsup. However, it can be extended easily to find imperfectly sporadic rules that are nearly perfect: for instance, by setting maxsup$_i$ to maxsup/minconf where maxsup$_i$ is maximum support for imperfectly sporadic rules and maxsup is maximum support for reported sporadic rules.

3.3 The "Less Rare Itemset" Problem

It is, of course, true that rare itemsets may be formed by the combination of less rare itemsets. For instance, itemset A may have support 11%, itemset B support 11%, but itemset $A \cup B$ only 9%, making $A \cup B$ sporadic for a maxsup of 10% and $A \rightarrow B$ a valid imperfectly sporadic rule for minconf of 80%. However, Apriori-Inverse will not generate this rule if maxsup is set to 10%, for $A \rightarrow B$ is an imperfectly sporadic rule rather than a perfectly sporadic one.

It would be nice to be able to generate imperfectly sporadic rules as well. We note, however, that not all imperfectly sporadic rules are necessarily interesting:

in fact, many are not. One definition of rules that are trivial is proposed by Webb and Zhang [7], and a similar definition for those that are redundant given by Liu, Shu, and Ma [8]. Consider an association rule $A \rightarrow C$ with support 10% and confidence 90%. It is possible that we may also generate $A \cup B \rightarrow C$, with support 9% and confidence 91%: however, adding B to the left hand side has not bought us very much, and the shorter rule would be preferred. Another situation in which trivial rules may be produced is where a very common item is added to the consequent; it is possible that $A \rightarrow C$ has high confidence because the support of C is close to 100% (although in this case, it would be noticeable due to having a lift value close to 1.0). Therefore, we do not necessarily wish to generate *all* imperfectly sporadic rules.

We propose three different modifications of Apriori-Inverse, all of which produce rules that are not-too-far from being perfect. We refer to the modifications as "Fixed Threshold", "Adaptive Threshold", and "Hill Climbing". In general, we adjust the maxsup threshold to enable us to find at least some imperfectly sporadic rules: specifically, those that contain subsets that have support just a little higher than maxsup.

Fixed Threshold: In this modification, we propose adjusting the maximum support threshold before running Apriori-Inverse to enable us to find more rare itemsets. The maximum support threshold is adjusted by taking the proportion of the maximum support threshold and the minconf threshold. For example, given a minsup threshold of 0.20 and a minconf of 0.80, the new minsup threshold would be set to 0.2/0.8 = 0.25. However, during the generation of rules, we only consider itemsets that satisfy the original maximum support threshold. Rules that have supports which are higher than the *original* maxsup are not generated.

Adaptive Threshold: In this modification, we propose changing the maximum support by a small increment η (typically 0.001) at each value of k during the generation of sporadic k-itemsets. The threshold is increased until the number of itemsets in the current generation does not change when compared to the previous generation. In general, we search for a plateau where the number of itemsets found does not change.

Hill Climbing: Hill Climbing is an extension of Adaptive Threshold; it adjusts the maximum support threshold by adding an increment that is the product of a rate-variable η (like the learning constant for a gradient descent algorithm; but typically 0.01) and the gradient of the graph of the number of itemsets generated so far. Like the previous method we modify the threshold until the number of itemsets reaches a plateau. Using this method in a large dataset the plateau is likely to be found sooner, since the increment used becomes greater when the gradient is steep and smaller when the gradient becomes less steep.

4 Results and Discussion

In this section, we compare the performance of the standard Apriori algorithm program with the proposed Apriori-Inverse. We also discuss the results of three the different variation of Apriori-Inverse. Testing of the algorithms was carried out on six different datasets from the UCI Machine Learning Repository [9].

Table 1 displays results from implementations Apriori-Inverse and the Apriori algorithms. Each row of the table represents an attempt to find perfectly sporadic rules—with maxsup 0.25, minconf 0.75, and lift greater than 1.0—from the database named in the left-most column. For Apriori-Inverse, this just involves setting maxsup and minconf values. For the Apriori algorithm, this involves setting minsup to zero (conceptually; in reality, the algorithm has been adjusted to use a minimum absolute support of 5), generating all rules, then pruning out those that fall above maxsup. In each case, this final pruning step is not counted in the total time taken. In the first three cases, Apriori was able to generate all frequent itemsets with maxsup greater than 0.0, but for the final three it was not clear that it would finish in reasonable time. To give an indication of the amount of work Apriori is doing to find low-support rules, we lowered its minsup threshold until it began to take longer than 10,000 seconds to process each data set.

Table 1. Comparison of results of Apriori-Inverse and Apriori

Dataset	Apriori-Inverse (maxsup=0.25,minconf=0.75)				Apriori (minconf=0.75)					
	Rules	Passes	Average Sporadic Itemsets	Time (sec)	Min Sup	Rules with Min Sup < 0.25	Rules	Passes	Average Frequent Itemsets	Time (sec)
TeachingEval.	11	3	12	0.01	0	281	294	4	68	0.32
Bridges	9	3	8	0.01	0	24086	24436	9	405	6.44
Zoo	79	4	11	0.03	0	40776255	42535557	17	34504	8380.64
Flag	2456	7	128	1.32	0.11	16427058	16944174	14	57765	11560.77
Mushroom	1142015	13	3279	225.20	0.15	28709481	31894347	16	21654	11489.32
Soybean-Large	37859	10	307	6.51	0.43	0	101264259	17	46310	11550.22

Using Apriori, we were able to find all rules below a support of 0.25 for the Teaching Assistant Evaluation dataset, Bridges dataset, and Zoo dataset. However, using Apriori on the Flag dataset and Mushroom dataset, we could only push the minimum support down to 0.11 and 0.15 respectively, before hitting the time constraint of 10 thousand seconds. Compare this to Apriori-Inverse, finding all perfectly sporadic rules in just a few minutes for the Mushroom database. For the Soybean-Large dataset, no rules below a support of 43% could be produced in under 10 thousand seconds.

We conclude that, while Apriori is fine for discovering sporadic rules in small databases such as the first three in Table 1, a method such as Apriori-Inverse is required if sporadic rules under a certain maximum support are to be found in larger or higher-dimensional datasets. We also note that Apriori is finding a much larger number of rules under maxsup than Apriori-Inverse; this is, of course, due to Apriori finding all of the imperfectly sporadic rules as well as the perfectly sporadic rules. To take the Teaching Evaluation Dataset as an example, Apriori finds

$$\{\text{course}=11\} \rightarrow \{\text{instructor}=7\}$$
$$\{\text{course}=11,\ \text{nativeenglish}=2\} \rightarrow \{\text{instructor}=7\}$$
$$\{\text{course}=11\} \rightarrow \{\text{instructor}=7,\text{nativeenglish}=2\}$$

whereas, from this particular grouping, Apriori-Inverse only finds

$$\{\text{course}=11\} \rightarrow \{\text{instructor}=7\}$$

However, since the second and third rules found by Apriori have the same support, lift, and confidence values as the first, they both count as trivial according to the definitions given in [8] and [7]. Apriori-Inverse has ignored them (indeed, has never spent any time trying to generate them) because they are imperfect.

Table 2 shows a comparison of the methods used to allow Apriori-Inverse to find some imperfectly sporadic rules. The Fixed Threshold method finds the largest number of sporadic rules, because it is "overshooting" the maxsup thresholds determined by the two adaptive techniques, and therefore letting more itemsets into the candidate group each time. As a result, it requires fewer passes of the inverted index, but each pass takes a bit longer, resulting in longer running times. However, the times for the Fixed Threshold version seem so reasonable that we are not inclined to say that the adaptive techniques give any significant advantage. Determining a principled way to generate imperfectly sporadic rules—and determining a good place to *stop* generating then—remains an open research question. Nevertheless, we note that the time taken to generate all of the imperfectly sporadic rules by all three methods remains very much smaller than the time taken to find them by techniques that require a *minimum* support constraint.

Table 2. Comparison of results of extensions to Apriori-Inverse

Dataset	Fixed Threshold				Adaptive Threshold ($\eta = 0.001$)				Hill Climbing ($\eta = 0.01$)			
	Rules	Passes	Avg Spdc Sets	Time (sec)	Rules	Passes	Avg Spdc Sets	Time (sec)	Rules	Passes	Avg Spdc Sets	Time (sec)
TeachingEval.	46	4	22	0.01	11	6	12	0.03	11	6	12	0.04
Bridges	104	5	14	0.03	30	11	8	0.04	30	11	8	0.04
Zoo	203	5	15	0.03	203	19	13	0.14	203	19	13	0.12
Flag	12979	9	268	4.86	5722	31	165	9.49	13021	42	228	19.81
Mushroom	1368821	13	7156	791.82	1142015	26	3279	445.95	1142015	26	3279	474.88
Soybean-Large	1341135	11	801	31.47	95375	52	425	63.09	56286	30	352	27.38

5 Conclusion and Future Work

Existing association mining algorithms produce all rules with support *greater* than a given threshold. But, to discover rare itemsets and sporadic rules, we should be more concerned with *infrequent* items. This paper proposed a more efficient algorithm, Apriori-Inverse, which enables us to find perfectly sporadic rules without generating all the unnecessarily frequent items. We also defined the notion of imperfectly sporadic rules, and proposed three methods of finding them using Apriori-Inverse: Fixed Threshold, Adaptive Threshold, and Hill Climbing.

With respect to finding imperfectly sporadic rules, our proposed extensions to Apriori-Inverse are—at best—heuristic. More importantly, there are some types of imperfectly sporadic rule that our methods will not find at all. Our future work will involve ways of discovering rules such as $A \cup B \rightarrow C$ where neither A nor B is rare, but their association is, and C appears with $A \cup B$ with high confidence. This is the case of a rare association of common events (A and B) giving rise to a rare event (C). It is a particularly interesting form of imperfectly sporadic rule, especially in the fields of medicine (rare diseases) and of process control (disaster identification and avoidance).

References

1. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In Buneman, P., Jajodia, S., eds.: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data. (1993) 207–216
2. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In Bocca, J.B., Jarke, M., Zaniolo, C., eds.: Proceedings of the 20th International Conference on Very Large Data Bases, VLDB'94. (1994) 487–499
3. Cohen, E., Datar, M., Fujiwara, S., Gionis, A., Indyk, P., Motwani, R., Ullman, J.D., Yang, C.: Finding interesting association rules without support pruning. IEEE Transactions on Knowledge and Data Engineering **13** (2001) 64–78
4. Liu, B., Hsu, W., Ma, Y.: Mining association rules with multiple minimum supports. In: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Disconvery and Data Mining. (1999) 337–341
5. Yun, H., Ha, D., Hwang, B., Ryu, K.H.: Mining association rules on significant rare data using relative support. The Journal of Systems and Software **67** (2003) 181–191
6. Toivonen, H., Klemettinen, M., Ronkainen, P., Hatonen, K., Mannila, H.: Pruning and grouping of discovered association rules. In: ECML-95 Workshop on Statistics, Machine Learning, and Knowledge Discovery in Databases. (1995) 47–52
7. Webb, G.I., Zhang, S.: Removing trivial associations in association rule discovery. In: Abstracts Published in the Proceedings of the First International NAISO Congress on Autonomous Intelligent Systems (ICAIS 2002). (2002)
8. Liu, B., Hsu, W., Ma, Y.: Pruning and summarizing the discovered associations. In: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. (1999) 125–134
9. Blake, C., Merz, C.: UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html`, University of California, Irvine, Department of Information and Computer Sciences (1998)