A Max-Min Approach for Hiding Frequent Itemsets

George V. Moustakides Vassilios S. Verykios

Dept. of Computer and Communication Engineering
University of Thessaly, Volos, GREECE
{moustaki,verykios}@inf.uth.gr

Abstract

In this paper we are proposing a new algorithmic approach for sanitizing raw data from sensitive knowledge in the context of mining of association rules. The new approach (a) relies on the maxmin criterion which is a method in decision theory for maximizing the minimum gain and (b) builds upon the border theory of frequent itemsets.

1 Introduction

Oftentimes, privacy policies and regulations enforce us to sanitize a data set from possible confidential information and/or knowledge. The task we are called for solving in this paper, known as *association rule hiding*, is to remove certain patterns from a database that we consider sensitive, by changing the data in such a way that everything but the sensitive knowledge remains intact. The solution proposed here builds upon the idea that we can minimize the impact of the changes in the data, by considering only to minimize the impact on the positive border of the frequent patterns. As long as we succeed in maximizing the minimum gain, all the non-sensitive frequent patterns which are not in the positive border remain above the support threshold which means that they are preserved.

The rest of this paper is organized as follows. Section 2 presents related work. In Section 3 we state the problem by providing all the necessary background information, and in Section 4 we expose our proposed maxmin technique for the association rule hiding problem. In Section 5 we present two algorithms and in Section 6 we discuss how the proposed algorithms behave in comparison with other similar algorithms proposed elsewhere. Finally, in Section 7 we conclude our study.

2 Related Work

The problem of hiding sensitive knowledge either in the form of frequent itemsets or in the form of sensitive rules appeared for the first time in an early paper by Atallah et. al. in [2]. The authors in this first work had investigated the problem of hiding sensitive frequent patterns by using a greedy approach. Dasseni et. al. in [3] generalized the problem by considering the hiding of both sensitive frequent itemsets and sensitive rules. The algorithms initially proposed in [3] were later on improved and evaluated for their performance in [9]. Saygin et. al. in [7] consider the problem of hiding frequent patterns and rules by using unknowns. An in depth experimentation and evaluation of distortion and blocking techniques has been performed by Pontikakis et. al. in [6]. Zaiane et. al. in [5] present a new formalization of the association rule hiding problem which try to remove/hide the inference channels created by rules that exist in a released rule base.

3 Problem Formulation

We consider the knowledge hiding problem in the context of frequent itemset mining of the association rule discovery framework. In a database D, a large or frequent itemset is a set of items A chosen from the universe of possible items $I = \{i_1, i_2, i_3, \dots, i_n\}$ that has greater support – denoted as $supp$ – by a minimum user specified support threshold. The support of an itemset indicates the number of database transactions that an itemset appears in. In the association rule mining framework, by specifying a minimum support threshold, we are in a position to discover the large itemsets in an increasing order of cardinality.

The formalism of the border presented in [4] is important in our problem formulation. Let P be a set of itemsets, and $\preceq$ a partial order on P. Further, let F be closed downwards under the relation $\preceq$. The *border* $Bd(F)$ of F consists of those itemsets ϕ such that all more general itemsets than ϕ are in F and no itemsets more specific than ϕ is in F. Those

Sixth IEEE International Conference on Data Mining - Workshops (ICDMW'06)
0-7695-2702-7/06 $20.00 © 2006 IEEE

itemsets ϕ in $Bd(F)$ that are in F are called the *positive border* $Bd^+(F)$, and those itemsets ϕ in $Bd(F)$ that are not in F are the *negative border* $Bd^-(F)$. Let us also consider a subset S of the set of frequent itemsets F which includes the sensitive rules in F, that needs to become safe in a statistical sense. Such a requirement will force S to move to $P \setminus F$ and then it will necessitate a revision of both the positive and negative border. The same idea was initially proposed by Sun and Yu in [8].

4 A Max-Min Approach for Itemset Hiding

The Max-Min approach that we propose, relies on the fact that we succeed in the hiding of the sensitive itemsets while at the same time we minimize the impact of the hiding process to the non-sensitive information. We achieve this by considering only the effects of the hiding process to the itemsets on the positive border, after the border has been revised by taking into consideration the sensitive large itemsets.

4.1 Border Revision Theory

labelborder The algorithm that revises the positive border, starts from the minimum level of the frequent itemset lattice corresponding to the shorter sensitive itemsets. It then proceeds upwards to the itemset lattice by removing from the lattice, all the sensitive itemsets along with their super itemsets, until it reaches the upper limit of the lattice, where it stops. After that, it moves downward the new frequent lattice and it checks whether a frequent itemset has a cover. An itemset cover is a super itemset of this itemset that is also frequent. If a frequent itemset in the revised frequent lattice does not have a cover, then it belongs to the new positive border. The algorithm stops, when it reaches the minimum level of the revised frequent itemset lattice.

In a similar manner, the algorithm that revises the negative border, starts from the bottom of the revised frequent itemset lattice, which has been produced after the revision of the positive border. After the formation of the revised positive and negative border, the algorithms proposed in this paper can take over the hiding of the sensitive itemsets by actually modifying the database. Both of the algorithms accept as input, the revised positive border, as well as the intersection of the revised negative border with the list of sensitive itemsets.

4.2 Hiding of a Sensitive Itemset

Let us assume that a sensitive itemset needs to be hidden. For every item that belongs to a sensitive itemset we list the set of positive border itemsets which depend on it. We call an item that belongs to a sensitive itemset a *tentative victim item*, and a set of maximally large itemset that belong to the revised positive border and is affected by a tentative victim item, as a *tentative victim itemset*. We also call the set of tentative victim itemsets that depend on the same tentative victim item vi as the vi-list. In every vi-list, we select the itemset(s) with the minimum support which we call *minimum support itemsets*. A minimum support itemset is the most sensitive one among the itemsets in each vi-list since it is the closest to the borderline between the positive and the negative border.

From among all the minimum border itemsets, we select the itemset(s) with the highest (maximum) support. We call such an itemset the *max-min* itemset, since this is the only itemset among the different minimum support itemsets which is the maximum distance away from the border. The max-min itemset determines the tentative victim item through which the hiding of the sensitive itemset will take place. We call such an item a *victim* item. The proposed algorithms modify the victim item indicated by the max-min itemset in such a way that the value of the support of the max-min itemset, if possible, not to be modified.

In the following discussion we present a number of theorems which are cornerstone in the functionality of the proposed Max-Min algorithms. The first theorem is concerned with two tentative victim itemsets that achieve the minimum support value for their corresponding vi-lists. The theorem indicates that if the support values for two minimum support itemsets are different, then by modifying the tentative victim item that corresponds to the larger minimum support itemset, the support of the smaller minimum support itemset remains unaffected. This is true independently of whether the larger minimum support itemset is affected or not by the modification. We state the theorem more formally next.

Theorem 1 *If two minimum support itemsets S_A and S_B contain the tentative victim items A and B respectively and have their supports such as $supp(S_A) > supp(S_B)$, then the minimum support itemset S_B does not contain A.*

PROOF Let us assume that S_A and S_B are two minimum support itemsets corresponding to the tentative victim items A and B respectively. We also assume that $supp(S_A) > supp(S_B)$ and that S_B contains the tentative victim item A. Since S_B contains the tentative victim item A, it should also appear in the list of tentative victim itemsets of tentative victim item A along with S_A. Because $supp(S_A) > supp(S_B)$, it means that the S_B is the minimum support itemset for the tentative victim item A. But we claimed at the beginning that the minimum support itemset for A is S_A. For this reason, we have proved that S_B cannot contain sensitive item A.$\square$

The direct consequence of Theorem 1 is that if we modify the victim item A, and this has as a side effect that the max-min decreases by one, no other minimum support value will be affected by this change.

The second theorem and its accompanying lemma apply to the case when two sets of minimum support itemsets corresponding to two different tentative victim items happen to have the same support. Below we state formally a theorem that ensures that if the victim item corresponding to one set of minimum support itemsets can change without affecting the support of its minimum support itemsets, then the other set of minimum support itemsets will not be affected either.

Theorem 2 *Let $L_A = \{A_1, A_2, \ldots, A_K\}$ and $L_B = \{B_1, B_2, \ldots, B_M\}$, the vi-lists for two tentative victim items A and B correspondingly which are contained in sensitive itemset S. Let also $L_{S_A} = \{A_{i_1}, A_{i_2}, \ldots, A_{i_k}\}$ and $L_{S_B} = \{B_{j_1}, B_{j_2}, \ldots, B_{j_m}\}$, the sets of minimum support itemsets that correspond to the tentative victim items, such that $s_A = supp(A_{i_1}) = supp(A_{i_2}) = \ldots = supp(A_{i_k}) < supp(A_i)$where $i \notin \{i_1, i_2, \ldots, i_k\}$ and $s_B = supp(B_{j_1}) = supp(B_{j_2}) = \ldots = supp(B_{j_m}) < supp(B_j)$ where $j \notin \{j_1, j_2, \ldots, j_m\}$. If $s_A = s_B$ and the support of S is decreased through A without the border itemsets in L_{S_A} to be affected, then no itemset in L_{S_B} will be affected either.*

PROOF If an itemset in L_{S_B} is affected by the decrease of the sensitive itemset through A, then this means that the affected itemset contains A. If it contains A, it should also belong to L_{S_A}. But the theorem claims that no itemset from L_{S_A} is affected from the change in A. Because of that, no itemset in L_{S_B} will be affected either. □

The lemma below applies to the cases in complement to those covered by Theorem 2 that the change in a victim item causes a minimum support itemset from its corresponding vi-list to loose support. If it so happens that the minimum support itemset which looses support is the only one affected in the vi-list by the change in the victim item, and at the same time it is not included in the other set of minimum support itemsets, then no minimum support itemset from the second set will be affected. The statement and its proof follows more formally below.

Lemma 1 *Let $s_A = s_B$ and the support of S is decreased through A. Let also $A_{i_1}, A_{i_2}, \ldots, A_{i_r}$ be a set of tentative victim itemsets in L_{S_A} which are both different from all the tentative victim itemsets in L_{S_B} and are also affected by the decrease in the support of S through A, while the rest of the border itemsets in L_{S_A} are not affected. If this holds, then no tentative victim itemset in L_{S_B} is affected from the decrease of S through A.*

PROOF Since $A_{i_1}, A_{i_2}, \ldots, A_{i_r} \in L_{S_A}$ are both the only border itemsets in L_{S_A} which are affected by the change in S through A, and they are different from all the border itemsets in L_{S_B}, no border itemset in L_{S_B} will be affected. If a border itemset $B_{i_s} \in L_{S_B}$ is being affected by a change in A, then this border itemset should contain A and also belong to L_{S_A}. Therefore we would have an additional itemset besides $A_{i_1}, A_{i_2}, \ldots, A_{i_r} \in L_{S_A}$ affected by the change, which leads to a contradiction.□

4.3 Hiding of Sets of Sensitive Itemsets

The algorithms perform a sorting of the sensitive itemsets based on their support in an increasing order of support, and start the hiding process from the sensitive itemset with the minimum support. After the algorithms finish the hiding of the minimum support itemset, they continue with the next itemset in order until they are done with the hiding of every sensitive itemset. By adopting a heuristic like that we enforce the requirement of the minimum impact followed by all the proposed Max-Min algorithms, since the sensitive itemset with the minimum support is the one that it is closer to the border, and it is hidden in the smaller number of iterations. We can also easily prove that the support of the minimum support itemset remains constantly smaller than the support of all the other sensitive itemsets that remain to be hidden. This has as a result that the complete hiding of a certain sensitive itemset can be considered in isolation from the hiding of all the other sensitive itemsets. The following theorem makes the above idea concrete.

Theorem 3 *Let $S_1, S_2, \ldots, S_n$ be the sensitive itemsets which are sorted in increasing order of their supports. The support of the minimum support sensitive itemset S_1 will be constantly smaller than the supports of all the other sensitive itemsets $S_2, S_3, \ldots, S_n$ during the hiding of this itemset.*

PROOF In every iteration of a Max-Min algorithm, the sensitive itemset in the foreground (the itemset selected each time by the algorithm for hiding) looses one point from its support. For any other sensitive itemset, the algorithm may or may not reduce the support of this itemset by at most one. For this reason, the difference in the support of the itemset selected for hiding with all the rest, after the application of the Max-Min algorithm, will be at least as it was at the beginning of the hiding process. □

Because of the fact that different sensitive itemsets (except the one on which the algorithm is applied each time) may loose different amounts of support, after the hiding of each sensitive itemset, the remaining (not hidden yet) itemsets need to be sorted again in the increasing order of their supports. A Max-Min algorithm needs to decide also which sensitive itemset to select next for hiding in case there is a tie in the supports of different itemsets. In this case, the algorithm considers the longer sensitive itemset first because this itemset offers to the algorithm the higher degree of freedom with respect to the selection of the victim items.

5 Max-Min Algorithms

In the following discussion we present two Max-Min algorithms that hide sensitive itemsets in increasing sophistication.

5.1 Algorithm MaxMin 1

The first algorithm MaxMin 1 hides a sensitive itemset by selecting the victim item in such a way that the side effects to the minimum support itemsets in the positive border are minimized. To achieve this, it hides the victim item from a transaction which supports the corresponding sensitive itemset, so as the support of the minimum support itemsets except possibly the support of the max-min itemsets remains unchanged. MaxMin 1 algorithm assures that if the value of the max-min is unique, then by modifying the max-min itemset, no other minimum border itemset is modified. This property holds because of Theorem 1. In case the max-min value is attained by more than one tentative victim itemsets where each one corresponds to different tentative victim items, MaxMin 1 randomly selects the victim item to use for hiding. It is also indifferent to the algorithm the number of different border itemsets that attain the max-min value, because all of them indicate the same victim item. The pseudocode of the proposed MaxMin 1 algorithm is shown in Figure 1.

```
Initialize list of sensitive itemsets
While sensitive itemsets not hidden {
  Sort sensitive itemsets in increasing support
  Select the minimum support sensitive itemset
  Build the vi-list representation
  While selected sensitive itemset not hidden {
    Find max-min itemset by randomly resolving ties
    Choose 1st transaction containing sensitive itemset
    Remove the victim item
    If sensitive itemset not hidden Then
      Revise vi-list representation
    Else exit inner loop  }
  Remove selected sensitive itemset from the list
  If list of sensitive itemsets is empty Then
     Exit inner loop   }
```

Figure 1. Pseudocode for MaxMin 1.

5.2 Algorithm MaxMin 2

Algorithm MaxMin 2 checks whether the reduction in the support of the sensitive itemset can be achieved without modifying the support of any max-min itemset. By using Theorem 1 we know in advance that no other minimum support itemset will be affected in this case. For achieving to reduce the support of the sensitive itemset without affecting the support of the max-min itemsets, we need to ensure that each of the max-min itemsets is not a subset of the sensitive itemset, and at the same time that there are transactions that provide support to the sensitive itemset without supporting the max-min itemset. In order to be able to get this information we need to maintain for every sensitive itemset S and every max-min itemset V in the vi-list, the list of transactions that support them. If we denote the lists of these transactions as L_S and L_V respectively, then by computing the difference between these two lists (sets) $L_S - L_V$ we will know whether we can reduce the support of the sensitive itemset without affecting the support of the max-min itemset, if the result is a non-empty list.

In those cases where there are more than one max-min itemsets that correspond to different tentative victim items we go through all the vi-lists containing max-min itemsets and check whether we can reduce the support of the sensitive itemset without affecting any of the max-min itemsets in each vi-list. If we can do this, then by making use of Theorem 2 we can ensure that no other max-min itemset in any other vi-list will be affected. In order to do this, we need to find transactions that support the sensitive itemset but do not support any of the max-min itemsets in the vi-list. For this, we compute again the lists of transactions that support both the sensitive itemset and the set of max-min itemsets in every vi-list and we compute the difference of the lists that correspond to the max-min itemsets from the list of the sensitive itemset. If the result is not empty, then we can reduce the support of the sensitive itemset without affecting any other itemset. In case where the previous scenario is not feasible we consider pairs of vi-lists in every

```
Initialize list of sensitive itemsets
While sensitive itemsets not hidden {
  Sort sensitive itemsets in increasing order of support
  Select the minimum support sensitive itemset
  Build vi-list representation
  While selected sensitive itemset not hidden {
    Determine the max-min itemsets
    If max-min attained by a vi-list {
      Compute Lsensitive - Lmax-min
      If the list is not empty Then
        Remove victim item from a transaction in the list
      Else Remove victim itemset from a transaction that
           affects minimum number of max-min itemsets
    } Else {
      While not done with $vi1$-lists {
        Compute Lsensitive-Lmax-min
        If the list is not empty Then
          Remove victim itemset from a transaction
        Else While not done with vi2-list and vi1 != vi2 {
            Compute Lvi1-list'-Lvi1-list''-Lvi2-list
            If list is not empty Then
              Remove victim item from a transaction
            else Remove victim item that affects minimum
                 number of max-min itemsets } } }
  If sensitive itemset not hidden then
    Revise vi-list representation
  else exit inner loop }
  Remove selected sensitive itemset from the list
  If list of sensitive items is empty then
  Exit inner loop }
```

Figure 2. Pseudocode for MaxMin 2.

iteration of the algorithm. We call the first list in the pair as $vi1$-list and the second $vi2$-list. Then what we try to do is to find a set of minimum support itemsets in a $vi1$-list that will loose support by a change in the corresponding victim item, which are different from all the minimum support itemsets of another $vi2$-list that attain the max-min value, while we can maintain the support of the rest of the minimum support itemsets in $vi1$-list, then by Lemma 1 we can ensure that the max-min value in $vi2$-list will remain the same. Finally, there is no other option than to reduce the support of the sensitive itemset while we reduce the support of the max-min itemset as well. The sketch of the MaxMin 2 algorithm is given in Figure 2.

6 Experiments and Evaluation

We have compared our algorithms with the border based approach which proposed by Sun and Yu in [8]. For evaluation purposes we have created a table that indicates the performance of our algorithms versus the performance of the border based approach for hiding different sets of itemsets from the itemset lattice of the example presented in [8]. Table 1 contains four columns. The first presents the

Table 1. Comparison of Border based, MaxMin 1 and MaxMin 2 algorithms.

Sensitive Itemsets	Border Based	MaxMin 1	MaxMin 2
ab	$2/0$	$2/0$	$2/0$
ad	$4/1$	$4/1$	$4/0$
cd	$4/2$	$4/3$	$4/3$
abd	$1/0$	$1/0$	$1/0$
cde	$1/1$	$1/2$	$1/1$
ab, acd	$4/1$	$3/0$	$3/0$
ac, abd	$4/1$	$3/1$	$3/0$
ad, bcd	$5/1$	$5/1$	$5/0$
bc, cde	$2/1$	$2/1$	$2/1$
ce, abd	$2/0$	$2/1$	$2/0$
ac, abd, cde	$4/1$	$4/2$	$3/1$
ab, de, acd	$5/2$	$4/1$	$3/0$
ac, ad, bcd	$5/0$	$6/1$	$5/0$
abd, acd, cde	$4/2$	$3/2$	$3/2$
abd, acd, bcd	$4/0$	$3/0$	$3/0$
ab, bc, cd, de	$9/2$	$8/2$	$7/0$

set of sensitive itemsets and the rest three indicate the performance of the algorithms compared. The performance of each algorithm is measured by using the notation m/n where m indicates the number of changes in raw data and n indicates the number of side effects. The information listed in Table 1 indicates that for the reference example, as the number of sensitive itemsets grows, MaxMin 2 algorithm outperforms both the Border based and the MaxMin 1 algorithm. The results presented above are consistent with similar comparisons that we have performed with larger data sets generated by the IBM Synthetic Data Generator [1].

7 Conclusions

We have presented two new algorithms which rely on the maxmin criterion for the hiding of sensitive itemsets in an association rule hiding framework. Both algorithms apply the idea of the maxmin criterion in order to minimize the impact of the hiding process to the revised positive border which is produced by removing the sensitive itemsets and their super itemsets from the lattice of frequent itemsets. By restricting the impact on the border, we can be very efficient in the selection of items which must be selected for hiding, while at the same time it can be ensured that non-border itemsets are protected from hiding. An initial evaluation indicated that the proposed algorithms being at the same time less computationally demanding, outperform other similar approaches.

References

[1] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *Proceedings of the Twentieth VLDB*, 1994.

[2] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, and V. S. Verykios. Disclosure limitation of sensitive rules. In *Proceedings of the 1999 IEEE Knowledge and Data Engineering Exchange Workshop*, pages 45–52, 1999.

[3] E. Dasseni, V. S. Verykios, A. K. Elmagarmid, and E. Bertino. Hiding association rules by using confidence and support. In *Proceedings of the 4th International Workshop on Information Hiding*, pages 369–383, 2001.

[4] H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.

[5] S. R. M. Oliveira, O. R. Zaïane, and Y. Saygin. Secure association rule sharing. In *Proceedings of the 8th Pacific-Asia Conference (PAKDD 2004)*, pages 74–85, 2004.

[6] E. D. Pontikakis, V. S. Verykios, and Y. Theodoridis. On the comparison of association rule hiding heuristics. In *Hellenic Database Management Symposium*. 2004.

[7] Y. Saygin, V. S. Verykios, and C. Clifton. Using unknowns to prevent discovery of association rules. *ACM SIGMOD Record*, 30(4):45–54, 2001.

[8] X. Sun and P. S. Yu. A border-based approach for hiding sensitive frequent itemsets. In *Proceedings of the Fifth IEEE International Conference on Data Mining (ICDM 2005)*, pages 426–433, 2005.

[9] V. S. Verykios, A. K. Emagarmid, E. Bertino, Y. Saygin, and E. Dasseni. Association rule hiding. *IEEE Transactions on Knowledge and Data Engineering*, 16(4):434–447, 2004.