Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 Keller Hall
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 11-007

Automatic Detection Of Vaccine Adverse Reactions By Incorporating Historical Medical Conditions

Zhonghua Jiang and George Karypis

March 21, 2011

Automatic Detection Of Vaccine Adverse Reactions By Incorporating Historical Medical Conditions

Zhonghua Jiang
Computer Science & Engineering
University of Minnesota, Twin Cities
zjiang@cs.umn.edu

George Karypis
Computer Science & Engineering
University of Minnesota, Twin Cities
karypis@cs.umn.edu

Abstract

Identifying medical conditions that are correlated with vaccine adverse reactions can not only provide better understanding of how adverse reactions are triggered but also have the potential of detecting new adverse reactions that are otherwise hidden. We formulate this problem as mining frequent patterns with constraints. The major constraint we use is called the minimum *dual-lift* constraint, where *dual-lift* is a novel measure we propose to evaluate correlations in a pattern. We also introduce the notation of minimum *improvement* constraint to remove redundancy in generated pattern set. We come up with a novel approach to upper bound the *dual-lift* measure which helps to prune the search space. Experimental results show that our algorithm works significantly better than the baseline on dense datasets. Our algorithm is also tested on the real world VAERS database. Some interesting vaccine adverse reactions identified are presented.

1. INTRODUCTION

The Vaccine Adverse Event Reporting System (VAERS) is a passive post-marketing surveillance system to monitor vaccine safety and is co-managed by the Food and Drug Administration (FDA) and the Centers for Disease Control and Prevention (CDC) [1]. Adverse reactions that occur after the administration of vaccines are reported. Each report also contains historical medical conditions under which the vaccine adverse reactions are developed, such as, medications the vaccine recipient was taking, pre-existing physician diagnosed allergies and birth defects, and any illness at the time of vaccination, etc.

Detecting vaccine adverse reactions in VAERS is a well known challenging task due to the inherent limitations of spontaneous reporting system. For example, the number of doses of administered vaccine (the "denominator data") is not available, which is necessary for calculating reporting rates; the reported incidences are not verified which partially contributes to the wide range of data quality; the unvaccinated control groups are missing. Under reporting is a well known issue for spontaneous reporting systems. The reporting rates are different from vaccine to vaccine and from adverse reaction to adverse reaction. The seriousness of an adverse reaction is also affecting reporting rates [2].

Traditional usage of VAERS is to identify evidences that suggest an adverse reaction might be caused by vaccination and warrant further investigation [3]-[4]. We extend the state of art by bringing historical medical conditions into the picture, that is, to identify evidences that suggest certain vaccines might cause adverse reactions for patients with specific medical conditions. The solution to the extended problem is a set of patterns, each of which is a triplet $\langle$*medical conditions*, *vaccines*, *adverse reactions*$\rangle$ satisfying additional constraints.

The main contribution of this work is two folded. Firstly, we introduce a novel measure called *dual-lift* to capture correlations between vaccines, adverse reactions and medical conditions. The main constraint a pattern needs to satisfy is called minimum *dual-lift* constraint which says its *dual-lift* is larger or equal to some user specified threshold. Secondly, we come up with an efficient algorithm *DLiftMiner* (which stands for *Dual-Lift Miner*) for mining constraint patterns in VAERS database. The ability to push constraints into pattern mining may lead to substantial performance improvement. Different strategies are needed for different types of constraints. Well studied constraints include monotone and anti-monotone constraints, succinct constraint, convertible constraint, boundable constraint, and loose monotone constraint [5]-[6]. Our *DLiftMiner* algorithm shows that the minimum *dual-lift* constraint is boundable, so the branch and bound algorithm can be applied successfully.

The remaining of this paper is organized as follows. In Section 2, related works to adverse reaction detection and constraint pattern mining are briefly discussed. In Section 3, we introduce notations and definitions that are used through out the rest of the paper. In Section 4, we discuss *dual-lift* measure in detail. In Section 5, the constraint pattern mining problem is formulated. The *DLiftMiner* algorithm is explained in Section 6. Especially, we focus on a novel approach to upper bound the *dual-lift* measure. Experimental results with both synthetic and real world datasets are included in Section 7. And finally Section 8 concludes.

2. RELATED WORK

Various techniques have been developed to help uncover potential adverse reactions without the incorporation of historical medical conditions. The proportional reporting ratio (PRR) was first described by Finney [3]. For the contingency Table illustrated in Table 1, PRR is $PRR_{ij} = [a/(a+b)]/[c/(c+d)]$. Evans *et al* [7] extend it to screened proportional reporting ratio (SPPR): $a \geq 3, PRR \geq 2$, and Yates-corrected chi-square≥ 4. Empirical Bayes geometric mean (EBGM) [8] calculates the expected value of the base 2 logarithm of the ratio between the estimated reporting ratio and that under the assumption of no causal relationship, given the observed count of reports. EB05 [9] uses the lower-bound of the 90% confidence interval of EBGM. Reporting odds ratio (ROR) [10]-[11] is defined as product of the exposure odds among the cases with respect to the exposure odds among the non-cases and is calculated by $ROR = (ad)/(bc)$. Bate *et al* uses information component (IC) as the measure of associations and a Bayesian confidence propagation neural network (BCPNN) [12] is developed. Similar to [12], the multi-item gamma Poisson shrinker (MGPS) [13]-[4] also utilizes Bayesian inference to adjust for data variance.

Table 1: Contingency Table for Vaccine Reaction Pair.

	Reaction j	Other Reactions
Vaccine i	a	b
Other Vaccines	c	d

In recent years, there are several works conducted to mine association rules directly and effectively from input database without the costly feature selection step. Many of these papers involve implementing certain pruning strategies. Harmony [14] is an instance centric algorithm which mines directly for each instance a covering rule with the highest confidence. A method to discover top-k covering rules for the input gene expression data is in [15]. In [16], the branch-and-bound algorithm is applied to discriminative measure *information gain* (IG). Cheng *et al* [16] introduce a feature centered approach which eliminates training instances on a progressively shrinking FP-tree. In TAPER algorithm [17], an upper bound of Pearson's correlation is shown to have the conditional monotone property. LPMiner [18] utilizes smallest valid extension (SVE) property to prune the search space. Brin *et al* [19] show an upward closure property of correlation measure: if a set of items S is correlated, so is every super set of S. In [20], the pruning strategy is to upper bound the *support*, *confidence* and a new measure called *improvement*. In [21], an upper bound of chi-square is derived based on convexity of chi-square function.

3. DEFINITIONS AND NOTATION

A *vaccine-adverse reaction* database DB is a set of records of the form $\langle tid, M, V, R\rangle$, where *tid* is a unique record id. Each record can be interpreted as that an individual with historical medical conditions M developed adverse reactions R after vaccination of V. We will refer to the sets of items in M, V, and R as the m-items, v-items and r-items, respectively.

A record $\langle tid, M, V, R\rangle$ is said to contain itemset I if $I \subseteq M \cup V \cup R$. The *support* of I in DB, denoted by $supp(I|DB)$, is the total number of records in DB that contain I. The *transaction count* of I in DB, denoted by $tcnt(I|DB)$, is the number of records for which $I = M \cup V \cup R$. An association rule is of the form $I \rightarrow r$, where I is the prefix itemset and r is the suffix itemset. The *confidence* of $I \rightarrow r$ is defined as

$$conf(I \rightarrow r|DB) = \frac{supp(I \cup r|DB)}{supp(I|DB)}. \quad (1)$$

The *lift* of $I \rightarrow r$ is defined as

$$lift(I \rightarrow r|DB) = \frac{conf(I \rightarrow r|DB)}{conf(\emptyset \rightarrow r|DB)}, \quad (2)$$

where $\emptyset$ is the null set.

A *conditional database* $DB_{|A}$ is a set of records from DB which contain itemset A. A *projected database* DB_B is formed by projecting all records from DB onto itemset B, that is, removing all items which are not in B from each record in DB. A *projected conditional database* $DB_{B|A}$ is defined as projected database from $DB_{|A}$ on itemset B. Let set $DB^{(d)}$ contain all itemsets whose transaction counts in DB are non-zero, and similarly $DB_B^{(d)}$ ($DB_{B|A}^{(d)}$) contain all itemsets whose transaction counts in DB_B ($DB_{B|A}$) are non-zero.

4. DUAL-LIFT MEASURE

Our objective is to identify all patterns of the form $\langle M, V, R\rangle$ such that there are evidences to suggest adverse reactions R are likely to be developed if individuals with historical medical conditions M are vaccinated with V. The first step is to develop a measure to capture correlations among M, V, and R. And patterns whose values of the measure are larger than or equal to user specified threshold are selected into the solution set.

One may consider to use $lift(M \cup V \rightarrow R|DB)$ as the measure. However, this measure has the weakness that M and V are not distinguished from each other, so that the correlation may be introducted by one of them, say V, but not the other one, say M. Another possibility is

$$\mathcal{L} = \frac{P(M \cup V \cup R|DB)}{P(M|DB)P(V|DB)P(R|DB)}, \quad (3)$$

where $P(I|DB)$ (I is M, V, R or $M \cup V \cup R$) is the probability of I estimated in DB (for example, $P(I|DB)$ can be estimated by $conf(\emptyset \rightarrow I|DB)$). The problem with this measure is that $\mathcal{L}$ can be high if any two of the three sets are correlated. In fact, it is easy to see that $\mathcal{L} = lift(M \cup V \rightarrow R|DB)lift(M \rightarrow V|DB)$. So $\mathcal{L}$ can be high if only M and V are highly correlated.

Instead, let us consider quantity $lift(V \rightarrow R|DB_{|M})$. Intuitively, it detects vaccine adverse reaction pairs $\langle V, R\rangle$ in a subset of records containing medical conditions M. It makes sense because if individuals with medical conditions M are susceptible to $\langle V, R\rangle$, we should be able to detect $\langle V, R\rangle$ in $DB_{|M}$. Another advantage of this measure is that we can uncover potentially important vaccine adverse reaction pairs $\langle V, R\rangle$ ($lift(V \rightarrow R|DB_{|M})$ is high), which are difficult to detect by looking at the whole database ($lift(V \rightarrow R|DB)$ is low). However, there is a limitation of this measure if used

alone. That is, $\langle V, R\rangle$ may not be correlated with M at all. To overcome this limitation, we propose to combine this quantity with $lift(M \rightarrow R|DB_{|V})$. Intuitively, $lift(M \rightarrow R|DB_{|V})$ is high meaning that $conf(M \cup V \rightarrow R|DB)$ is much larger than $conf(V \rightarrow R|DB)$, so individuals with medical conditions M have much higher chance of developing R when vaccinated with V. Based on this analysis, we define:

DEFINITION 4.1 (*dual-lift*). *The dual-lift of a pattern $\langle M, V, R\rangle$, denoted by*

$$dual\_lift(\langle M, V, R\rangle|DB), \tag{4}$$

is defined as

$$\min(lift(V \rightarrow R|DB_{|M}), lift(M \rightarrow R|DB_{|V})). \tag{5}$$

5. PROBLEM FORMULATION

In this section, we formulate the problem of detecting vaccine adverse reactions with historical medical conditions in the framework of constraint pattern mining. In our formulation, there are three components that define the interesting patterns: (1) minimum *dual-lift* constraint, which utilizes the *dual-lift* measure developed in Section 4, (2) minimum *improvement* constraint, which aims to remove *redundent* patterns, and (3) minimum *support* constraint, which defines the set of *frequent* patterns. We define them formally in the following:

DEFINITION 5.1 (MINIMUM *dual-lift* CONSTRAINT). *The pattern $\langle M, V, R\rangle$ satisfies minimum dual-lift constraint in database DB if its dual-lift is larger or equal to the user specified lift threshold l_0.*

DEFINITION 5.2 (*improvement*). *The improvement of pattern $\langle M, V, R\rangle$ is defined as the minimum of*

$$dual\_lift(\langle M, V, R\rangle|DB) - dual\_lift(\langle M', V', R'\rangle|DB). \tag{6}$$

for any of its proper sub-pattern $\langle M', V', R'\rangle$, that is, $M' \subseteq M \wedge V' \subseteq V \wedge R' \subseteq R$ and $M' \cup V' \cup R' \neq M \cup V \cup R$, satisfying minimum dual-lift constraint.

DEFINITION 5.3 (MINIMUM *improvement* CONSTRAINT). *A pattern $\langle M, V, R\rangle$ satisfies minimum improvement constraint if its improvement is larger or equal to user specified improvement threshold m_0. A pattern which does not satisfy minimum improvement constraint is said to be redundant.*

DEFINITION 5.4 (MINIMUM *support* CONSTRAINT). *The pattern $\langle M, V, R\rangle$ satisfies minimum support constraint if*

$$supp(M \cup V \cup R|DB) \geq s_0, \tag{7}$$

where s_0 is the user specified minimum support. A pattern satisfying minimum support constraint is called a frequent pattern.

In this paper, we only consider the case when R contains a single reaction. With the aid of above definitions, our problem can be stated as:

PROBLEM 5.5. *Given database DB, the goal is to search for all possible non-redundant frequent patterns $\langle M, V, r\rangle$ satisfying minimum dual-lift constraint, where M is a set of m-items, V is a set of v-items, and r is a set containing a single r-item.*

6. DLIFTMINER ALGORITHM

6.1 Branch and Bound Algorithm

Algorithm 1 DLiftMiner

Input: An input vaccine adverse reaction database DB^0

1: read records in DB^0; count frequencies of m-items and v-items.
2: read records again and construct database DB as a *FP-Tree*, where m-items and v-items are internal nodes and r-items are leaves. m-items and v-items are sorted based on their frequencies from large to small along the tree from top to bottom.
3: remove infrequent items in database DB.
4: Let M_1, V_1, R_1 be sets of m-items, v-items, r-items in DB.
5: $DB^m \leftarrow DB_{M_1 \cup R_1}$; $DB^v \leftarrow DB_{V_1 \cup R_1}$
6: **for** r in R_1 **do**
7: $\quad l[r] \leftarrow l_0$
8: **end for**
9: $M_0 \leftarrow \emptyset$; $V_0 \leftarrow \emptyset$
10: branch_and_bound(DB, DB^m, DB^v, l, M_0, V_0)

Our *DLiftMiner* algorithm is described in algorithm 1, where the most important part *branch_and_bound* is described in algorithm 2. The *DLiftMiner* algorithm sets proper initial values for parameters, then passes the control to algorithm *branch_and_bound*. Input parameters of *branch_and_bound* can be interpreted as the following. l contains the current *dual-lift* threshold for each r-item. M_0, and V_0 are sets of m-items and v-items that have been searched. DB can be understood as $DB^0_{|M_0 \cup V_0}$ with infrequent items removed (lines 14, 31, 32); DB^m can be understood as the projected database from $DB^0_{|M_0}$ onto m-items and r-items in DB (lines 17, 20, 33); DB^v can be understood as the projected database from $DB^0_{|V_0}$ onto v-items and r-items in DB (lines 17, 20, 34).

The *branch_and_bound* algorithm starts with calculating *dual-lift* for current pattern $\langle M_0, V_0, r\rangle$ for each r-item r of DB (lines 2-6). L^m_r (L^v_r) is the quantity $lift(V_0 \rightarrow |DB^0_{|M_0})$ ($lift(M_0 \rightarrow |DB^0_{|V_0})$). To see why lines 2-6 are correct, consider:

$$\begin{aligned} & lift(V_0 \rightarrow r|DB^0_{|M_0}) \\ = & \frac{conf(V_0 \rightarrow r|DB^0_{|M_0})}{conf(\emptyset \rightarrow r|DB^0_{|M_0})} \\ = & \frac{conf(\emptyset \rightarrow r|DB^0_{|M_0 \cup V_0})}{conf(\emptyset \rightarrow r|DB^0_{|M_0})} \\ = & \frac{conf(\emptyset \rightarrow r|DB)}{conf(\emptyset \rightarrow r|DB^m)} \\ = & \frac{supp(r|DB)/|DB|}{supp(r|DB^m)/|DB^m|}, \end{aligned} \tag{8}$$

Algorithm 2 branch_and_bound(DB, DB^m, DB^v, l, M_0, V_0)

```
1: for r in all r-items of DB do
2:    S_r ← supp(r|DB)/|DB|
3:    S_r^m ← supp(r|DB^m)/|DB^m|
4:    S_r^v ← supp(r|DB^v)/|DB^v|
5:    L_r^m ← S_r/S_r^m, L_r^v ← S_r/S_r^v
6:    dual-lift_r = min(L_r^m, L_r^v)
7:    if dual-lift_r ≥ l[r] then
8:       print ⟨M_0, V_0, r⟩.
9:       l[r] ← dual-lift_r + m_0.
10:   end if
11: end for
12: while DB has more m-items or v-items do
13:   i ← next m-item or v-item of DB
14:   DB' ← DB_|i
15:   if i is m-item then
16:      M_0' ← M_0 ∪ i, V_0' ← V_0
17:      DB'^m ← DB_|i^m, DB'^v ← DB^v
18:   else
19:      V_0' ← V_0 ∪ i, M_0' ← M_0
20:      DB'^m ← DB^m, DB'^v ← DB_|i^v
21:   end if
22:   let R be the set of frequent r-items in DB'
23:   for r in R do
24:      bnd_r^m ← b^u_{r|DB'}/b^l_{r|DB'^m}, bnd_r^v ← b^u_{r|DB'}/b^l_{r|DB'^v}
25:      dual-lift-bnd_r = min(bnd_r^m, bnd_r^v)
26:      if dual-lift-bnd_r < l[r] then
27:         remove r from R
28:      end if
29:   end for
30:   if |R| > 0 then
31:      construct DB'; i and r-items not in R are not
         copied
32:      remove infrequent items in DB'
33:      construct DB'^m; remove items that are not in DB'
34:      construct DB'^v; remove items that are not in DB'
35:      branch_and_bound(DB', DB'^m, DB'^v, l, M_0', V_0')
36:   end if
37:   remove i from DB
38:   if i is m-item then
39:      remove i from DB^m
40:   else
41:      remove i from DB^v
42:   end if
43: end while
```

and similarly,

$$lift(M_0 \to r|DB^0_{|V_0}) = \frac{supp(r|DB)/|DB|}{supp(r|DB^v)/|DB^v|}. \quad (9)$$

If the calculated *dual-lift* is larger or equal to its threshold for the corresponding r-item, $l[r]$ is updated (line 9). Then it determines whether the next item i should be projected on by finding the set R which contains frequent r-items in DB' (or $DB_{|i}$) whose *dual-lift* upper bound is larger than or equal to current threshold. *dual-lift* upper bounds are calculated in lines 24, 25 as $dual\text{-}lift\text{-}bnd_r$, where bnd_r^m (bnd_r^v) is upper bound of $lift(V_0' \cup V \to r|DB^0_{|M_0' \cup M})$ ($lift(M_0' \cup M \to r|DB^0_{|V_0' \cup V})$) for any frequent patterns $\langle M, V, r\rangle$ in DB'. Line 24 will be explained in more detail later. If R is not empty, new databases DB', DB'^m, DB'^v are constructed with necessary items removed (lines 30-36), and *branch_and_bound* is called with updated parameters (line 35). Note that there is no need to actually construct DB', DB'^m and DB'^v in lines 14-20; we can operate on original databases DB, DB^m and DB^v to evaluate *lift* upper bounds in line 24. Finally, item i is removed and the next round of loop starts (lines 38-42).

6.2 Support Based Pruning

Support based pruning can be derived from minimum *support* constraint in definition 5.4. Our algorithm applies support based pruning in line 3 of *DLiftMiner* and lines 32-34 of *branch_and_bound*, where infrequent items are removed, and line 22 of *branch_and_bound*, where only frequent r-items are considered. Any r-item is infrequent if its support is less than s_0. In our formulation, every interesting pattern is associated with an r-item. For any m-item or v-item i we define its *maximum support per class* as $\max_r supp(i \cup r|DB)$. Any m-item or v-item whose *maximum support per class* is less than s_0 is considered infrequent and thus removed.

6.3 Improvement Based Pruning

For any r-item r, patterns to be explored by the branch and bound algorithm are super-patterns of $\langle M_0, V_0, r\rangle$. From the definition of minimum *improvement* constraint, the *dual-lift* thresholds for patterns to be explored should be no less than m_0 plus the maximum *dual-lift* values for any proper sub-patterns of $\langle M_0, V_0, r\rangle$ that branch and bound has discovered. These *dual-lift* thresholds are saved in l. When a new pattern is discovered, its *dual-lift* has to be no less than current *dual-lift* threshold. That means, this new pattern must have largest *dual-lift* value among its discovered sub-patterns. So updating corresponding *dual-lift* threshold is straightforward: simply add m_0 to *dual-lift* value of the new pattern (line 9).

However, it should be made clear that our algorithm does not take into account all redundancy defined in definition 5.3. This is due to that not all sub-patterns of $\langle M_0, V_0, r\rangle$ were processed before pattern $\langle M_0, V_0, r\rangle$. For example, assume $M_0 = \{m_1, m_2\}$ and $V_0 = \{v_1\}$ and assume these three items were added to $M_0 \cup V_0$ in the order of m_1, v_1, m_2, then sub-patterns processed before $\langle\{m_1, m_2\}, v_1, r\rangle$ are $\langle\emptyset, \emptyset, r\rangle$, $\langle m_1, \emptyset, r\rangle$, and $\langle m_1, v_1, r\rangle$. Other sub-patterns (for example, $\langle m_2, v_1, r\rangle$) are processed after $\langle\{m_1, m_2\}, v_1, r\rangle$. Solving this issue is out of scope of this paper and is left as future research.

6.4 Lift Based Pruning

From line 24 in branch and bound algorithm, we need to find upper bounds for quantities $lift(V_0' \cup V \rightarrow r|DB^0_{|M_0' \cup M})$ and $lift(M_0' \cup M \rightarrow r|DB^0_{|V_0' \cup V})$ for any frequent patterns $\langle M, V, r\rangle$ in DB'.

This problem can be tackled by observing that *lift* is the ratio of two *confidences*. In fact,

$$\begin{aligned} & lift(V_0' \cup V \rightarrow r|DB^0_{|M_0' \cup M}) \\ = & \frac{conf(V_0' \cup V \rightarrow r|DB^0_{|M_0' \cup M})}{conf(\emptyset \rightarrow r|DB^0_{|M_0' \cup M})} \\ = & \frac{conf(M \cup V \rightarrow r|DB^0_{|M_0' \cup V_0'})}{conf(M \rightarrow r|DB^0_{|M_0'})}, \end{aligned} \tag{10}$$

and similarly,

$$lift(M_0' \cup M \rightarrow r|DB^0_{|V_0' \cup V}) = \frac{conf(M \cup V \rightarrow r|DB^0_{|M_0' \cup V_0'})}{conf(V \rightarrow r|DB^0_{|V_0'})}. \tag{11}$$

Because DB' is $DB^0_{|M_0' \cup V_0'}$ with some items removed, DB'^m (DB'^v) is $DB^0_{|M_0'}$ ($DB^0_{|V_0'}$) with some items removed, and $\langle M, V, r\rangle$ is from DB', it is not hard to see that $DB^0_{|M_0' \cup V_0'}$, $DB^0_{|M_0'}$, and $DB^0_{|V_0'}$ can be replaced by DB', DB'^m and DB'^v in equations 10 and 11. So we have

$$lift(V_0' \cup V \rightarrow r|DB^0_{|M_0' \cup M}) = \frac{conf(M \cup V \rightarrow r|DB')}{conf(M \rightarrow r|DB'^m)}, \tag{12}$$

and

$$lift(M_0' \cup M \rightarrow r|DB^0_{|V_0' \cup V}) = \frac{conf(M \cup V \rightarrow r|DB')}{conf(V \rightarrow r|DB'^v)}. \tag{13}$$

Equations 12 and 13 suggest that, to find upper bound of *lift*, we can find the upper bound of the numerator and lower bound of denominator. So, line 24 should be correct if we introduce

- $bnd^u_{r|DB'}$ be upper bound of $conf(M \cup V \rightarrow r|DB')$ for any frequent patterns $\langle M, V, r\rangle$ in DB',
- $bnd^l_{r|DB'^m}$ be lower bound of $conf(M \rightarrow r|DB'^m)$ for any frequent patterns $\langle M, \emptyset, r\rangle$ in DB'^m,
- $bnd^l_{r|DB'^v}$ be lower bound of $conf(V \rightarrow r|DB'^v)$ for any frequent patterns $\langle \emptyset, V, r\rangle$ in DB'^v.

6.5 Bound the Confidence

The above analysis translates the problem of bounding the *lift* into the problem of bounding the *confidence*, which can be defined as:

PROBLEM 6.1 (BOUNDING THE CONFIDENCE). *For the database DB, itemsets I_0, and r, the problem of bounding the* confidence *is to find upper and lower bound for the quantity $conf(I \rightarrow r|DB)$ for any $I \subseteq I_0$ such that $supp(I \cup r|DB) \geq s_0$.*

Note that, I_0's for $bnd^u_{r|DB'}$, $bnd^l_{r|DB'^m}$, and $bnd^l_{r|DB'^v}$ discussed above are implicitly defined as all non-r-items of corresponding databases.

There is a simple solution to this problem. The upper bound of $conf(I \rightarrow r|DB)$ can be chosen as 1. The lower bound can be derived as

$$conf(I \rightarrow r|DB) = \frac{supp(I \cup r|DB)}{supp(I|DB)} \geq s_0/|DB|. \tag{14}$$

However, our goal is to come up with an approach which works significant better than this naive solution.

We introduce the following two lemmas (see appendix for proofs):

LEMMA 6.2. *Given database DB, itemsets I_0, and r, for any $I \subseteq I_0$, define vector $\mathbf{V}_I$ whose x-component $V_{I,x}$ is $supp(I|DB_{I_0})$ and y-component $V_{I,y}$ is $supp(I|DB_{I_0|r})$. We have $conf(I \rightarrow r|DB)$ is equal to slope of $\mathbf{V}_I$ and $supp(I \cup r|DB)$ is equal to y-component of $\mathbf{V}_I$.*

LEMMA 6.3. *Following the settings in lemma 6.2, for any itemset A, we define vector $\mathbf{v}_A$ whose x-component $v_{A,x}$ is $tcnt(A|DB_{I_0})$ and y-component $v_{A,y}$ is $tcnt(A|DB_{I_0|r})$. Let set $\mathcal{A}_I = \{A|A \in DB^{(d)}_{I_0} \wedge I \subseteq A\}$, then*

$$\mathbf{V}_I = \sum_{A \in \mathcal{A}_I} \mathbf{v}_A. \tag{15}$$

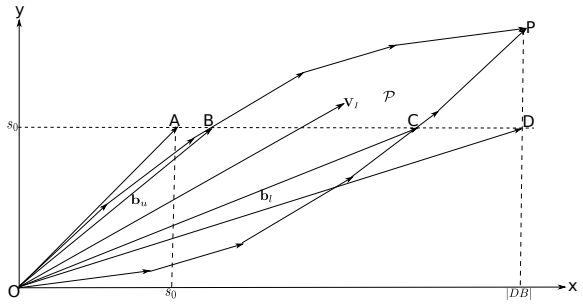

Figure 1: Bound the Confidence

Based on lemmas 6.2 and 6.3, Figure 1 illustrates how we can find an efficient solution to problem 6.1. Let us define $\mathcal{F}$ be the set of vectors $\mathbf{v}_A$ for all A in $DB^{(d)}_{I_0}$. Sorting vectors in $\mathcal{F}$ by slope from large to small and connecting them head to tail gives piece-wise linear curve $\overline{OBP}$. Similarly, sorting vectors in set $\mathcal{F}$ by slope from small to large and connecting them head to tail gives piece-wise linear curve $\overline{OCP}$. If we define another set $\mathcal{P}$ which contains sum of all possible subsets of vectors in $\mathcal{F}$, it can be proven that any vectors in set $\mathcal{P}$ are inside the region enclosed by $\overline{OBP}$ and $\overline{OCP}$. Lemma 6.3 suggests that $\mathbf{V}_I$ is also equal to sum of a subset of vectors from $\mathcal{F}$. So we should have $\mathbf{V}_I \in \mathcal{P}$ and $\mathbf{V}_I$ is inside region $\overline{OBPCO}$. Line $y = s_0$ cuts this region into two parts (we assume that the y-coordinate of point P is larger or equal to s_0). All vectors in $\mathcal{P}$ whose y-component are larger or equal to s_0 are inside the top region $\overline{BCPB}$. From lemma 6.2, any association rule $I \rightarrow r$ ($I \subseteq I_0$) whose $supp(I \cup r|DB)$ is

larger or equal to s_0 is associated with a vector $\mathbf{V}_I$ in region $\overline{BCPB}$ whose slope is exactly $conf(I \to r|DB)$. And it can be easily seen that slopes of all vectors inside region $\overline{BCPB}$ are upper bounded by slope of $\overline{OB}$ and lower bounded by slope of $\overline{OC}$. This leads to the conclusion that slopes of $\overline{OB}$ and $\overline{OC}$ are upper and lower bounds of $conf(I \to r|DB)$.

This result is summarized into the following theorem:

THEOREM 6.4. *Following the settings in lemma 6.3, define* $\mathcal{F} = \{\mathbf{v}_A | A \in DB^{(d)}_{I_0}\}$ *and sort vectors in* $\mathcal{F}$ *based on their slopes from small to large, and label them as* $\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n$, *where* n *is the total number of vectors. Define,*

- $b^l_{I_0 \to r|DB}$ *as slope of the following vector*

$$\mathbf{b}_l = \mathbf{f}_1 + \mathbf{f}_2 + \cdots + \mathbf{f}_{i_l - 1} + \alpha * \mathbf{f}_{i_l}, \qquad (16)$$

  where $0 < \alpha \leq 1$, $1 \leq i_l \leq n$ *are chosen such that y-component of* $\mathbf{b}_l$ *is equal to* s_0.
- $b^u_{I_0 \to r|DB}$ *as slope of the following vector*

$$\mathbf{b}_u = \mathbf{f}_n + \mathbf{f}_{n-1} + \cdots + \mathbf{f}_{i_u + 1} + \beta * \mathbf{f}_{i_u}, \qquad (17)$$

  where $0 < \beta \leq 1$, $1 \leq i_u \leq n$ *are chosen such that y-component of* $\mathbf{b}_u$ *is equal to* s_0.

Then we have, $b^l_{I_0 \to r|DB} \leq conf(I \to r|DB) \leq b^u_{I_0 \to r|DB}$, *for any* $I \subseteq I_0$ *such that* $supp(I \cup r|DB) \geq s_0$.

We can combine the bounding approach of theorem 6.4 with the naive approach (slopes of $\overline{OA}$ and $\overline{OD}$ in Figure 1) proposed earlier. This leads to different versions of *DLiftMiner* algorithms that are summarized in Table 2.

Table 2: Different versions of *DLiftMiner* algorithms.

DLiftMiner	applies theorem 6.4 for both upper and lower bounds of confidences
DLiftMiner-U	identical to *DLiftMiner* except confidence upper bound is replaced by 1
DLiftMiner-N	identical to *DLiftMiner* except confidence lower bound is replaced by the naive approach that is, $b^l_{r\|DB} = s_0/\|DB\|$
DLiftMiner-UN	identical to *DLiftMiner* except confidence upper bound is replaced by 1, and confidence lower bound is replaced by the naive approach
DLiftMiner-L	identical to *DLiftMiner* except confidence lower bound is replaced by zero that is, no *lift* pruning is applied

6.6 Implementations

Table 3: An example database

tid	m-items	v-items	r-items
1	m_1, m_2	v_1, v_2	r_1, r_3
2	m_1, m_3	v_1, v_3	r_1, r_2
3	m_1, m_4	v_2, v_3	r_3, r_4
4	m_2, m_3	v_2, v_3	r_1, r_3
5	m_2, m_3	v_1, v_4	r_2, r_4
6	m_2, m_4	v_3, v_4	r_3

Our implementation of *DLiftMiner* algorithm utilizes a modified *FPTree* data structure. And the bounding approach proposed in theorem 6.4 can be easily applied. This is mainly because each itemset A in $DB^{(d)}_{I_0}$ is mapped to a path along the tree excluding leaf nodes. We illustrate this by an example below.

Consider a mini-database in table 3 as an example. After this database is parsed, three $FPTree$ like data structures DB, DB^m, DB^v are initialized (see figure 2). Note that the same m-items, v-items and r-items are maintained in a double linked list, which is not shown in the figure. There are some modifications made to the standard $FPTree$. First, m-items and v-items are stored as internal nodes of the tree, while r-items are leaves of the tree. m-items and v-items are sorted by total frequency from large to small. Second, between m-items (or v-items) and r-items, we insert a sequence of separators, which are linked together just like any other items. Separators facilitate the calculation of bounds as will be explained later. Each separator node and leaf node is associated with a support count, which is the support of itemset from root to itself along the tree. Because each record can contain multiple r-items, the support count on a separator is not necessarily equal to sum of support counts of its r-item children. Fourth, for each m-item and v-item, the *maximum support per class* is maintained instead of the total support. For each r-item the total support is stored as usual. We have selected minimum *support* s_0 to be 2, the only infrequent item is v_4, which has been removed from DB and DB^v.

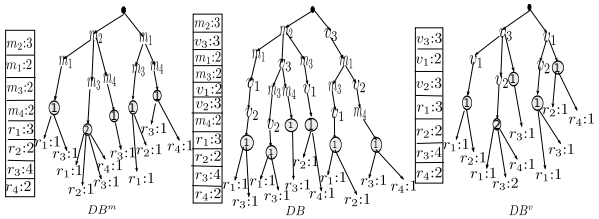

Figure 2: Databases after initialization

Let us select $l_0 = 1.5$. *branch_and_bound* starts by evaluating two *dual-lift* measure for each r-item in lines 2-6. In our example ,due to M_0 and V_0 are empty, the *dual-lift* values have to be equal to 1 for all r-items. In a more general scenario, the size of databases $|DB|$, $|DB^m|$, and $|DB^v|$ can be calculated as sum of support counts stored in all separators. The *support* values of r-items can be calculated as sum of support counts stored in corresponding r-item nodes. However, these values are pre-computed and readily available. If the calculated *dual-lift* is larger or equal to *dual-lift* threshold, the pattern is saved and *dual-lift* thresholds need to be updated (line 9).

In our implementation, the next item i in line 13 of the *branch_and_bound* is the m-item or v-item at the bottom of *FPTree DB*. For this example, it is m_4. *branch_and_bound* estimates *dual-lift* upper bound based on databases DB', DB'^m, and DB'^v. When i is m-item, DB'^v is just DB^v, DB'^m is $DB^m_{|i}$ and DB' is $DB_{|i}$ (line 17) where the latter two are shown in figure 3. Note that at this step, DB' and DB'^m do not need to be constructed explicitly (figure 3 is only for illustration's purpose). We can simply fetch separator nodes and leaf nodes under m_4 in DB and DB^m for our computation. There are two r-items in DB', namely, r_3 and

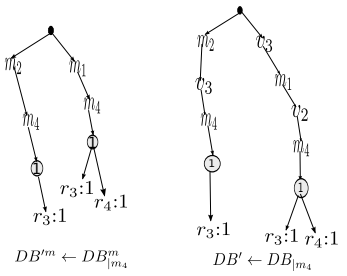

Figure 3: Databases conditional on next item $i = m_4$

r_4. Since r_4 is infrequent ($supp(r_4|DB') = 1$), we only need to compute *lift* upper bounds for r_3.

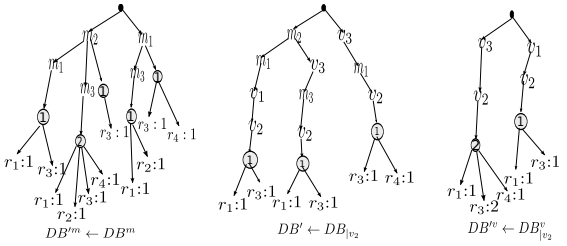

Figure 4: Databases in the second round

Next, we explain in detail how $b^u_{r_3|DB'}$ can be easily calculated. Following notations in theorem 6.4 , I_0 contains all m-items and v-items in DB'. There are two itemsets in $DB^{(d)}_{I_0|i}$, each of them is represented as a path from root to separator nodes on tree DB'. Let $A_1 = \{m_2, v_3, m_4\}$ and $A_2 = \{v_3, m_1, v_2, m_4\}$. $tcnt(A_1|{DB'}_{I_0}) = 1$ which is stored in the separator node following A_1; $tcnt(A_2|{DB'}_{I_0}) = 1$ which is stored in the separator node following A_2. And $tcnt(A_1|{DB'}_{I_0|r_3}) = 1$ which is stored in node r_3 following A_1; $tcnt(A_2|{DB'}_{I_0|r_3}) = 1$ which is stored in node r_3 following A_2. According to theorem 6.4, $\mathcal{F} = \{(1,1),(1,1)\}$. Then, we need to sort vectors in $\mathcal{F}$ by slope from large to small and add them sequentially until the y-component of added vector is equal to s_0. This computation is straightforward and finally the slope of summed vector is $b^u_{r_3|DB'} = 1$. Similarly, the set $\mathcal{F}$ for calculating $b^l_{r|DB'^v}$ is given by $\{(1,0),(2,2),(1,1),(1,1),(1,0)\}$. We need to sort them by slope from small to large and finally $b^l_{r|DB'^v} = 0.5$. Similarly, $b^l_{r|DB'^m} = 1$. So, $bnd^m_{r_3} = 1$ and $bnd^v_{r_3} = 2$. This tells us that the *dual-lift* for r_3 is 1, and the *dual-lift* threshold is violated. Set R is empty, and lines 30-36 are skipped. We remove m_4 from DB and DB^m and continue to next round of the loop.

The next item i to be considered is v_2. Three databases used for estimating *lift* upper bounds are shown in figure 4. We omit the details and simply present the results: $bnd^m_{r_1} = 2.5$, $bnd^v_{r_1} = 1.5$, $bnd^m_{r_3} = 2$, $bnd^v_{r_3} = 1$. r_4 is infrequent in DB' and thus not considered. So finally the R set contains one r-item r_1. Lines between 26 and 32 are executed. First, DB' is construct from $DB_{|v_2}$ where v_2 and r_3, r_4 are not copied. Then, infrequent items v_1 and m_3 are removed from DB'. Finally, projecting DB^m and $DB^v_{|v_2}$ onto DB' gives DB'^m and DB'^v. Note that some branches in DB'^m and DB'^v are

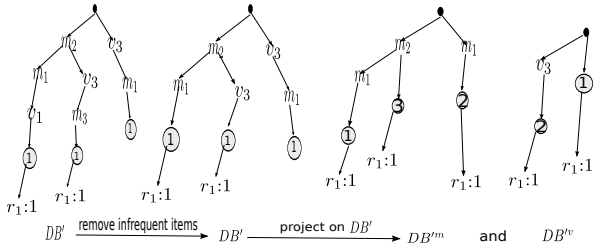

Figure 5: Projected databases in the second round

merged. DB', DB'^m and DB'^v are input to recursively call *branch_and_bound*.

7. EXPERIMENTAL RESULTS

7.1 Performance Study with Synthetic Data

Table 4: Parameters for synthetic dataset.

ntrans	number of transactions	$10k$
nitems	number of items	500
npats	number of maximal potentially large itemsets	100
patlen	average size of the maximal potentially large itemsets	30
tlen	average size of transactions	30

To evaluate the performance and scalability of our algorithm, we use the IBM Quest market-basket synthetic data generator [22] to generate a series of datasets. The generator program takes the parameters described in Table 4. We split the generated items into m-items, v-items and r-items from small id's to large id's so that m-items cover 57%, v-items cover 33% and r-items cover the rest 10%. We keep only transactions containing all three types of items in the dataset. The parameters for the first synthetic dataset are also in Table 4 and the actual size of dataset is 7401.

In our experiment, we set $s_0 = 100$, $m_0 = 0$ and l_0 takes value 2, 3, 4 and 5. Another parameter controlling the running is called *pattern length* L, which is defined to be the sum of sizes of current sets of m-items and v-items, that is, $L = |M_0| + |V_0|$. We specify a *maximum pattern length* $L_0 = 8$, and *DLiftMiner* stops when $L > L_0$. The experimental results are summarized in Table 5 and Figure 6.

Table 5: Running time for the first synthetic dataset.

dual-lift \model	*DLiftMiner-L*	*DLiftMiner-UN*	*DLiftMiner-N*	*DLiftMiner-U*	*DLiftMiner*
2	87.50	30.93	31.28	2.67	2.63
3	87.50	18.22	18.08	1.15	1.10
4	87.50	9.97	10.10	0.88	0.83
5	87.50	8.73	8.50	0.63	0.63

*All times are in minutes

Table 5 shows the running times for different models with different *dual-lift* parameters. From this Table, one can see the dramatic improvements made by our pruning approach. Our best algorithm *DLiftMiner* takes only 3.0%, 1.3%, 0.99%, and 0.72% of the running time compared to the baseline approach *DLiftMiner-L* when *dual-lift* takes values 2, 3, 4 and 5, respectively. Even compared with *DLiftMiner-UN*, where only naive pruning is applied, *DLiftMiner* takes only about 7% of its running time.

Table 5 also shows how the different pruning methods impact the performance of *DLiftMiner*. It can be seen that

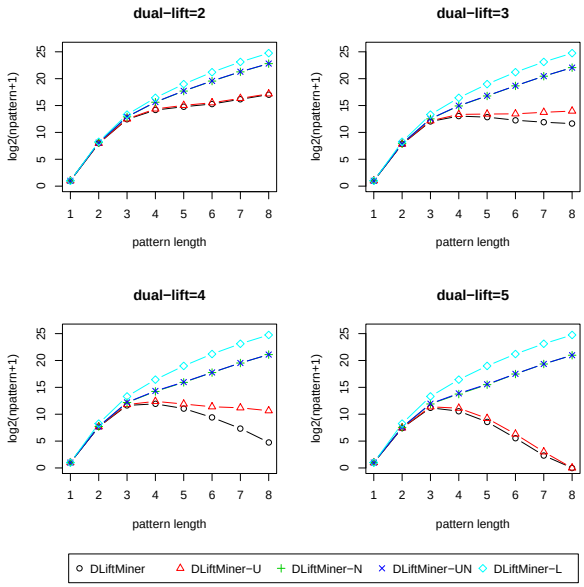

Figure 6: pruning effects with synthetic dataset

there are dramatic performance improvements when the confidence lower bound is improved (see *DLiftMiner-L* versus *DLiftMiner-UN* and *DLiftMiner-N* versus *DLiftMiner-U*). However the upper bound of *confidence* does not seem to play much role in reducing the running time, so that *DLiftMiner* and *DLiftMiner-U* (*DLiftMiner-N* and *DLiftMiner-UN*) have almost the same results.

Figure 6 plots the number of patterns processed (*npattern*) versus different *pattern length* for different versions of the algorithms. The vertical axis has been scaled to log scale ($log_2(npattern + 1)$). It can be clearly seen that including *confidence* upper bound does not help much in pruning more patterns away.

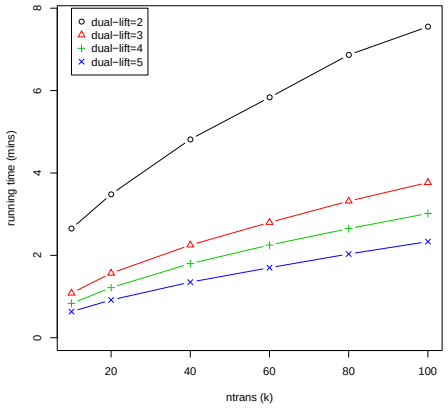

Figure 7: scalability of *DLiftMiner*

The scalability study of *DLiftMiner* algorithm is illustrated in Figure 7. We generate another 5 datasets by setting *ntrans* to be $20k$, $40k$, $60k$, $80k$ and $100k$ (the other parameters are the same as in Table 4). Due to that only transactions with all three types of items are kept, the actual sizes of these datasets are 14824, 29536, 44301, 59036 and 73729. For these datasets, we set corresponding minimum support s_0 to be 200, 400, 600, 800 and 1000. Figure 7 shows the running time of *DLiftMiner* algorithm versus different *ntrans* for four different *dual-lift* values. It can be seen that our *DLiftMiner* algorithm scales linearly with the size of input dataset.

Table 6: Number of patterns generated by *DLiftMiner*.

*dual-lift**improvement*	*NONE*	0	0.5	1.0
2	79355	40517	20558	19171
3	6772	4329	3271	3049

In the above experiments, we set the *improvement* measure m_0 to be 0. To evaluate how different values of m_0 impact the *DLiftMiner* algorithm, we calculated the numbers of patterns generated by *DLiftMiner* for different choices of *improvements* when *dual-lift* takes values 2 and 3. These results are summarized in Table 6, in which, *NONE* means no redundancy is taken into account. One can see that a large fraction of redundant patterns are removed during the mining process when $m_0 = 0$, but setting m_0 higher does not reduce as many patterns. However, we are not able to observe significant extra pruning by incorporating the *improvement* measure. This may partially due to the fact that our *DLiftMiner* algorithm does not take into account all redundancy defined in definition 5.3.

7.2 VAERS dataset

To evaluate the ability of the *dual-lift* measure to identify interesting patterns, we applied our *DLiftMiner* algorithm to VAERS [1] dataset. At the point of this paper, VAERS contains 21 years (from 1990 to 2010) of vaccination reports plus an additional non-domestic dataset. And we downloaded all of them.

The fields that are used as historical medical conditions or m-items include: *OTHER_MEDS*, *CUR_ILL*, *HISTORY*, and *PRIOR_VAX*. These are observables that are available before vaccination is given. *OTHER_MEDS* contains narrative about prescription or non-prescription drugs the vaccine recipient was taking at the time of vaccination. *CUR_ILL* contains narrative about any illness at the time of vaccination. *HISTORY* contains narrative about any pre-existing physician-diagnosed allergies, birth defects, medical conditions that existed at the time of vaccination. *PRIOR_VAX* provides prior vaccination event information. The set of vaccines or v-items is given by the *VAX_TYPE* field. The adverse reactions, or r-items, are called symptoms in this dataset. The Medical Dictionary for Regulatory Activities (MedDRA) coding system is used for symptoms. We remove those symptoms which do not appear very informative, for example, "Unevaluable event", "Accidental overdose", "Computerised tomogram normal", "Drug exposure during pregnancy" and so on. Also we remove all "injection site" symptoms and all symptoms containing word "negative". Further-

Table 7: Subset of vaccine-symptom associations with medical conditions in S_3.

Symptom	vaccine code(s)	medical condition(s)	*dual-lift*
Autism	MMR	history-"fever"	4.7
C-reactive protein increased	PNC	history-"Pregnancy"	9.3
Drug toxicity	HEP	history-"Pregnancy"	7.8
Febrile convulsion	DTPHIB	history-"seizure"	3.3
Influenza like illness	LYME	history-"Anxiety"	3.4
Injected limb mobility decreased	PPV	cur_ill-"allergy"	3.6
Irritability	HIBV	other_meds-"prevacid"	5.2
Otitis media	(MMR,OPV)	cur_ill-"Ear infection"	3.0
	(DTP,MMR)		3.4
	(HIBV,MMR,OPV)		3.5
	(DTP,HIBV,MMR)		3.6
	(DTP,MMR,OPV)		3.9
Rash vesicular	VARZOS	cur_ill-"Hypothyroidism"	3.3
Swelling	(IPV,MMR)	prior_sym-"Swelling"	3.8
Throat tightness	FLU	(history-"Asthma",history-"allergy")	3.7
Urine human chorionic gonadotropin positive	HPV4	history-"Pregnancy"	8.8
Varicella	VARCEL	cur_ill-"Asthma"	4.2

more we include some outcomes as r-items as well: *DIED*, *DISABLE* (disability), *ER_VISIT*, *HOSPITAL* (hospitalized), *L_THREAT* (life threatening), *RECOVD* (recovered) and *X_STAY* (prolonged hospitalization).

One challenge we face is that fields we extract for historical medical conditions or m-items are free text and thus not coded. We split the free text into a bag of words (unigrams). To capture some important terms with more than one word, we also split the text by different delimiters: space, comma, semicolon, period and combinations of them. Among the set of all terms generated, only those with frequency greater than or equal to 100 are kept. Furthermore we remove those terms that are obviously non-medical related. When we observe several terms apparently have the same meaning, we map them to a normalized term. For example, "allergy codeine", "allergy to codeine" and "allergic to codeine" are mapped to "codeine allergy". Following this simple procedure the total number of terms we get is 1187. For each record and field, the identified terms that are contained in the textual description are included as m-items.

For our experiment, we set $s_0 = 15$, $l_0 = 3$ and $m_0 = 0$. And we generated the following sets of patterns:

- S_1 contains all *non-redundant frequent* patterns $\langle V, r\rangle$ such that $lift(V \rightarrow r|DB) \geq l_0$. Note that since only one *lift* measure is involved, the concept of *redundancy* here should be re-defined accordingly. S_1 is the solution to the traditional adverse reaction detection problem, where historical medical conditions are not taken into account. In our experiment, we found 3617 patterns in S_1.
- S_2 contains all *non-redundant frequent* patterns $\langle M, V, r\rangle$ satisfying minimum *dual-lift* constraint. S_2 is the solution to problem 5.5. Experiments show that our idea of combining two *lift* measures with *redundancy* is quite selective. Only 169 patterns are found in S_2.
- S_3 is the set of patterns in S_2 whose vaccine reaction pairs $\langle V, r\rangle$ also appear in S_1. S_3 contains the identified medical conditions under which adverse reactions are more likely to be developed for a subset of patterns in S_1. We got the size of S_3 is 91.
- S_4 contains the rest of patterns in S_2. S_4 is the set of new vaccine adverse reactions that cannot be discovered without incorporation of historical medical conditions. In our experiment, S_4 has 78 patterns.

Due to limited space, we include only some of the patterns and the associated *dual-lift* values from S_3 and S_4 into Tables 7 and 8. For each vaccine set V, we present the pattern with the highest *dual-lift*. Note that in Tables 7, 8 (and Tables 10, 11) parenthesis implies vaccine or medical condition interaction, "history" is extracted from *HISTORY* field, "cur_ill" is extracted from *CUR_ILL* field, "other_meds" is extracted from *OTHER_MEDS* field, and "prior_sym" is symptom developed for prior vaccination and is extracted from *PRIOR_VAX* field. Vaccine adverse reactions listed in these two Tables can be interpreted easily. However, whether there are truly any causality involved in these patterns has to be determined by domain experts.

The Vaccine Injury Table is a list of vaccine-adverse reaction associations that the Institute of Medicine has determined are causal [23]-[24] (see Table 9). The Vaccine Injury Table we are using is from [24] with coding system for symptoms changing from Coding Symbols for a Thesaurus of Adverse Reaction Terms (COSTART) to MedDRA. Only four of eight symptoms are listed in Table 9 because no patterns were detected for another four (Arthritis, Encephalitis, Brachial plexopathy and Poliomyelitis).

Table 10 contains patterns in set S_1 whose symptoms are in Table 9 and vaccine sets have at least one of the reported vaccines for corresponding symptoms. Our results in Table 10 suggests that vaccine-vaccine interactions play important

Table 8: Subset of vaccine-symptom associations with medical conditions in S_4.

Symptom	vaccine code(s)	medical condition(s)	*dual-lift*
hospital-"Y"	HPV4	other_meds-"estradiol"	3.6
	PNC	history-"diabetes"	3.2
Drug ineffective	RAB	other_meds-"Hepatitis"	7.5
Erythema	VARCEL	(cur_ill-"allergy",history-"allergy")	3.6
Herpes zoster	(MMR,VARCEL)	history-"Otitis media"	4.6
Hypotension	TD	other_meds-"purified protein derivative"	3.3
Joint range of motion decreased	FLU	history-"deficit"	3.8
Lymphadenopathy	MMR	other_meds-"premarin"	5.5
Nervous system disorder	HEP	history-"Pregnancy"	6.3
Otitis media	(DTP,HIBV,OPV)	prior_sym-"fever"	3.2
Rash	(HIBV,OPV)	history-"penicillin"	3.2
Swelling	IPV	prior_sym-"Swelling"	3.6
	(DTAP,MMR)		3.2
Urticaria	(DTAP,IPV,MMR)	history-"premature"	3.4
Vasodilatation	DTP	(history-"Asthma",history-"allergy")	4.5
	OPV	(history-"allergy",history-"ceclor")	4.2
Wheezing	DTAP	(history-"allergy",other_meds-"albuterol")	3.0
White blood cell count increased	PPV	history-"Anxiety"	3.8

Table 9: Vaccine-Symptom associations from vaccine injury Table.

Symptom	Reported vaccine code(s)
Anaphylactic reaction	DT,DTAP,DTAPH,DTP
	HEP,IPV,MMR,TD
Encephalopathy	DTAP,DTAPH
	DTP,MMR
Intussusception	RV
Thrombocytopenic purpura	M[a] ,MM[a] ,MMR,MR[a]

[a] Vaccines excluded from analysis due to low or zero frequencies

Table 10: Vaccine-Symptom associations from vaccine injury Table found in S_1.

Symptom	Vaccine code(s)	Lift
Encephalopathy	(DTAP, HIBV)	3.2
	(HIBV, MMR)	3.1
Intussusception	RV	85.2
	(IPV, RV)	95.5
	(HIBV, RV)	86.4
	(DTAP, RV)	88.2
	(HIBV, IPV, RV)	105.0
	(DTAP, IPV, RV)	101.8
	(DTAP, HIBV, RV)	95.5
	(DTAP, HIBV, IPV, RV)	110.1
Thrombocytopenic purpura	(HIBV , MMR)	4.5

role when developing these symptoms. Rotavirus vaccine (RV) was licensed on August 31 1998 for routine use in infants in a three-dose series given at two, four and six months of age. In mid-May 1999, nine reports were submitted. On October 14 1999, the vaccine manufacturer voluntarily withdrew its product from the market [25]. Our result in Table 10 shows RV-intussusception can be detected, and it also shows RV can interact with other vaccines and develop intussusception together. Table 11 contains patterns in set S_2 whose symptoms and at least one of the vaccines appear in Vaccine Injury Table. Only two patterns are detected, both of which also appear in set S_4. This suggests that by incorporating historical medical conditions, we are likely to identify new patterns other than existing well known ones.

8. CONCLUSIONS

In summary, we formulate the problem of detecting vaccine adverse reactions by incorporating historical medical conditions as a constraint frequent pattern mining problem. We propose to use a novel measure called *dual-lift* to evaluate the significance of patterns. A novel bounding approach is developed for *confidence* and *lift*. Experimental results show that the pruning method is effective on dense datasets. We also presented some interesting vaccine adverse reactions discovered from VAERS database.

9. REFERENCES

[1] C. RT, R. SC, M. JR, and *et al*, "The vaccine adverse event reporting system (vaers)," *Vaccine*, vol. 12, pp. 542–550, 1994.

[2] R. S and C. R, "The reporting sensitivities of two passive surveillance systems for vaccine adverse events," *Am J Public Health*, vol. 85, pp. 1706–1709, 1995.

[3] F. DJ, "Systemic signalling of adverse reactions to drugs," *Methods Inf Med*, vol. 13, pp. 1–10, 1974.

[4] S. A, M. SG, and O. R, "Use of screening algorithms and computer systems to efficiently signal higher-than expected combinations of drugs and events in the us fda's spontaneous reports database," *Drug Saf*, vol. 25 (6), pp. 381–92, 2002.

[5] J. Pei and J. Han, "Can we push more constraints into frequent pattern mining?," *In Proceedings of ACM SIGKDD'00*, 2000.

Table 11: Vaccine-Symptom associations from vaccine injury Table found in S_2 and S_4.

Symptom	Vaccine code(s)	Medical Conditions	*dual-lift*
Anaphylactic reaction	MMR	cur_ill-"allergy"	3.9
	(MMR, VARCEL)	history-"allergy"	3.0

[6] F. Bonchi and C. Lucchese, "Pushing tougher constraints in frequent pattern mining," *In Proc. of PAKDD*, 2005.

[7] E. SJ, W. PC, and D. S, "Use of proportional reporting ratios (prrs) for signal from spontaneous adverse drug reaction reports," *Pharmacoepidemiol Drug Safe*, vol. 10, pp. 483–486, 2001.

[8] D. W, "Bayesian data mining in large frequency tables, with an application to the fda spontaneous reporting system," *Am Statistician*, vol. 53, pp. 177–190, 1999.

[9] D. W. and P. D, "Empirical bayes screening for multi-item associations," *Proceedings of the Seventh ACM SIGKDD international Conference on Knowledge Discovery and Data Mining*, pp. 67–76, 2001.

[10] V. P. E, D. W, and van Groothest K, "Application of quantitative signal detection in the dutch spontaneous reporting system for adverse drug reactions," *Drug Saf*, vol. 26 (5), pp. 293–301, 2003.

[11] V. der Heijden P, van Puijenbroek E, and van Buuren S *et al*, "On the assessment of adverse drug reactions from spontaneous reporting systems: the influence of under-reporting on odds ratios," *Stat Med*, vol. 21, pp. 2027–44, 2002.

[12] B. A, L. M, and E. I. *et al*, "A bayesian neural network method for adverse drug reaction signal generation," *Eur J Clin Pharmacol*, vol. 54, pp. 315–21, 1998.

[13] D. W, "Bayesian data mining in large frequency tables, with an application to the fda spontaneous reporting system," *Am Stat*, vol. 53 (3), pp. 170–90, 1999.

[14] J. Wang and G. Karypis, "On mining instance-centric classification rules," *IEEE Trans. Knowl. Data Eng.*, vol. 18(11), pp. 1497–1511, 2006.

[15] G. Cong, K.-L. Tan, A. K. H. Tung, and X. Xu, "Mining top-k covering rule groups for gene expression data," *In Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA. ACM*, pp. 670–681, 2005.

[16] H. Cheng, X. Yan, J. Han, and P. S. Yu, "Direct discriminative pattern mining for effective classification," *In Proceedings of the 24th International Conference on Data Engineering, Cancun, Mexico. IEEE*, pp. 169–178, 2008.

[17] H. Xiong, S. Shekhar, P.-N. Tan, and V. Kumar., "Exploiting a support-based upper bound of pearson's correlation coefficient for efficiently identifying strongly correlated pairs," *KDD'04, August, Seattle, Washington, USA. ACM*, pp. 22–25, 2004.

[18] M. Seno and G. Karypis, "Lpminer: An algorithm for finding frequent itemsets using length-decreasing support constraint," *ICDM'01, Nov*, 2001.

[19] S. Brin, R. Motwani, and C. Silverstein, "Beyond market basket: Generalizing association rules to correlations," *In Proc. of SIGMOD'97*, pp. 265–276, 1997.

[20] R. J. B. Jr., R. Agrawal, and D. Gunopulos, "Constraint-based rule mining in large, dense databases," *In Proc. of the 15th Int'l Conf. on Data Engineering*, pp. 188–197, 1999.

[21] S. Morishita and J. Sese, "Traversing itemset lattices with statistical metric pruning," *POD, Dallas, T X USA. ACM*, 2000.

[22] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," *Proceedings of the 20th VLDB Conference, Santiago, Chile*, 1994.

[23] S. KR, H. CJ, and J. R. (eds), *Adverse Events Associated with Childhood Vaccines: Evidence Bearing on Casuality*. Vaccine Safety Committee, Institute of Medicine: National Academy Press: Washington, DC, 1994.

[24] D. Banks, E. J. Woo, and *et al*, "Comparing data mining methods on the vaers database," *Pharmacoepidemiology and Drug Safety*, vol. 14, pp. 601–609, 2005.

[25] M. T. Niu, D. E. Erwin, and M. M. Braun, "Data mining in the us vaccine adverse event reporting system (vaers): early detection of intussusception and other events after rotavirus vaccination," *Vaccine*, vol. 19, pp. 4627–4634, 2001.

APPENDIX

A. PROOF OF LEMMA 6.2

We only need to prove the following two equations:

$$supp(I \cup r|DB) = supp(I|DB_{I_0|r}), \tag{18}$$

and,

$$conf(I \to r|DB) = \frac{supp(I|DB_{I_0|r})}{supp(I|DB_{I_0})}. \tag{19}$$

For any record $A \in DB$ that contains $I \cup r$, apparently $A \in B_{|r}$. Let $A' \in DB_{I_0|r}$ be the record after projecting A onto I_0. Because $I \subseteq I_0$, A' should also contain I. For the opposite direction, let $B \in DB_{I_0|r}$ containing I. Its corresponding record $B' \in DB_{|r}$ also contains I. So $B' \in DB$ contains $I \cup r$. The conclusion is that there is a one to one correspondence between records containing $I \cup r$ in DB and records containing I in $DB_{I_0|r}$. So $supp(I \cup r|DB) = supp(I|DB_{I_0|r})$.

Follow similar proof in previous part, we have $supp(I|DB) =$

$supp(I|DB_{I_0})$. From the definition of *confidence*, we get

$$conf(I \to r|DB) = \frac{supp(I \cup r|DB)}{supp(I|DB)} = \frac{supp(I|DB_{I_0|r})}{supp(I|DB_{I_0})}. \tag{20}$$

B. PROOF OF LEMMA 6.3

It can be seen that lemma 6.3 is equivalent to the following:

$$supp(I|DB_{I_0}) = \sum_{A \in \mathcal{A}_I} tcnt(A|DB_{I_0}), \tag{21}$$

$$supp(I|DB_{I_0|r}) = \sum_{A \in \mathcal{A}_I} tcnt(A|DB_{I_0|r}), \tag{22}$$

What equation 21 says is nothing more than that the number of records containing I in DB_{I_0} is equal to sum of transaction counts of each itemset containing I in $DB^{(d)}_{I_0}$. The proof is easy and omitted.

If in equation 22, $\mathcal{A}_I$ is replaced by $\mathcal{A'}_I = \{A | A \in DB^{(d)}_{I_0|r} \wedge I \subseteq A\}$, the proof would be identical to equation 21. Extra attention is needed for those itemsets that $\mathcal{A}_I$ and $\mathcal{A'}_I$ do not share. Because $DB_{I_0|r} \subseteq DB_{I_0}$, we have $DB^{(d)}_{I_0|r} \subseteq DB^{(d)}_{I_0}$, and consequently, $\mathcal{A'}_I \subseteq \mathcal{A}_I$. Let B be an arbitrary itemset in $\mathcal{A}_I$ but not in $\mathcal{A'}_I$, then $B \notin DB^{(d)}_{I_0|r}$. So we must have $tcnt(B|DB_{I_0|r}) = 0$. We conclude that all itemsets in $\mathcal{A}_I$ but not in $\mathcal{A'}_I$ do not contribute to the sum in equation 22, so equation 22 is valid.

C. PROOF OF THEOREM 6.4

The proof of the second part of theorem 6.4 is very similar to the first part and are omitted.

Let us introduce notation $||\mathbf{f}||$ to denote slope of vector $\mathbf{f}$ for any vector whose x-component is positive. The following lemma is applied repeatedly.

LEMMA C.1. *For $n \geq 2$, let $\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_n$ be a set of 2-D vectors satisfying $f_{i,x} > 0$ for $i = 1, 2, \ldots, n$. Assume $||\mathbf{f}_1|| \leq ||\mathbf{f}_2|| \leq \cdots \leq ||\mathbf{f}_n||$. We have*

1) $||\sum_{i=1}^{n} \mathbf{f}_i|| \leq ||\mathbf{f}_n||$,

2) $||\sum_{i=1}^{n} \mathbf{f}_i|| \geq ||\mathbf{f}_1||$.

PROOF. 1) If $n = 2$, we need to prove $(f_{1,y} + f_{2,y})/(f_{1,x} + f_{2,x}) \leq f_{2,y}/f_{2,x}$, which is equivalent to $f_{1,y} * f_{2,x} \leq f_{2,y} * f_{1,x}$, or $f_{1,y}/f_{1,x} \leq f_{2,y}/f_{2,x}$. The last inequality is just $||\mathbf{f}_1|| \leq ||\mathbf{f}_2||$, which is assumed to be correct. If $n > 2$, let us assume the conclusion is correct for $n - 1 \geq 2$. Let $\mathbf{F}_{n-1} = \sum_{i=1}^{n-1} \mathbf{f}_i$. So, $||\mathbf{F}_{n-1}|| \leq ||\mathbf{f}_{n-1}|| \leq ||\mathbf{f}_n||$. So we can have $||\mathbf{F}_{n-1} + \mathbf{f}_n|| \leq ||\mathbf{f}_n||$ due to the proof for the case $n = 2$. We have just shown the conclusion is also correct for n.

2) similar to 1) and thus omitted. □

Let $\mathbf{q}$ be an arbitrary vector in $\mathcal{P}$ whose y-component is larger or equal to s_0, and,

$$\mathbf{q} = \mathbf{f}_{j_1} + \cdots + \mathbf{f}_{j_a} + \gamma * \mathbf{f}_{i_l} + \mathbf{f}_{k_1} + \cdots + \mathbf{f}_{k_b}, \tag{23}$$

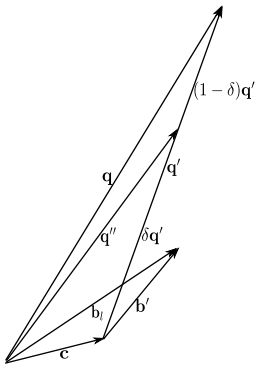

Figure 8: proof of theorem 6.4

where $1 \leq j_1 < \cdots < j_a < i_l < k_1 < \cdots < k_b \leq n$, and $\gamma = 0$ or 1. Our goal is to prove $||\mathbf{q}|| \geq ||\mathbf{b}_l||$.

Let $\mathbf{c} = \mathbf{f}_{j_1} + \cdots + \mathbf{f}_{j_a} + \gamma\alpha * \mathbf{f}_{i_l}$ and $\mathbf{q}' = \gamma(1-\alpha) * \mathbf{f}_{i_l} + \mathbf{f}_{k_1} + \cdots + \mathbf{f}_{k_b}$, so $\mathbf{q} = \mathbf{c} + \mathbf{q}'$. Let us introduce $\mathbf{b}'$ so that $\mathbf{b}_l = \mathbf{c} + \mathbf{b}'$; this is possible because $1 \leq j_1 < \cdots < j_a \leq i_l - 1$. Based on lemma C.1, we can get: if $q'_x > 0$, then $||\mathbf{q}'|| \geq ||\mathbf{f}_{i_l}||$; if $c_x > 0$, then $||\mathbf{c}|| \leq ||\mathbf{f}_{i_l}||$; if $b'_x > 0$, then $||\mathbf{b}'|| \leq ||\mathbf{f}_{i_l}||$.

Let us rule out some special cases. If $q'_x = 0$, then $c_x = s_0$ and $b'_x = 0$. So $\mathbf{b}_l = \mathbf{c} = \mathbf{q}$ and $||\mathbf{q}|| \geq ||\mathbf{b}_l||$ is certainly correct. If $c_x = 0$, then $q'_x > 0$ and $b'_x > 0$, so $||\mathbf{q}'|| \geq ||\mathbf{f}_{i_l}|| \geq ||\mathbf{b}'||$, thus $||\mathbf{q}|| \geq ||\mathbf{b}_l||$. If $q'_x > 0$, $c_x > 0$, but $b'_x = 0$, then $||\mathbf{q}|| = ||\mathbf{c} + \mathbf{q}'|| \geq ||\mathbf{c}|| = ||\mathbf{b}_l||$.

The only case left is $q'_x > 0$, $c_x > 0$ and $b'_x > 0$. We already know $||\mathbf{q}'|| \geq ||\mathbf{b}'||$, $||\mathbf{q}'|| \geq ||\mathbf{c}||$ and $q'_x \geq b'_x$. Since $q'_x \geq b'_x$, there exists $0 < \delta \leq 1$, such that $\delta * q'_x = b'_x$. Define $\mathbf{q}'' = \mathbf{c} + \delta * \mathbf{q}'$. Then $q''_x = b_{l,x}$. Because $||\mathbf{q}'|| \geq ||\mathbf{b}'||$, we have $\delta * q'_y \geq b'_y$, or, $q''_y \geq b_{l,y}$, which leads to $||\mathbf{q}''|| \geq ||\mathbf{b}_l||$. From lemma C.1, due to $||\mathbf{q}'|| \geq ||\mathbf{c}||$, we get $||\mathbf{q}''|| \leq ||\delta * \mathbf{q}'|| = ||\mathbf{q}'||$. So, finally, $||\mathbf{q}|| = ||\mathbf{q}'' + (1-\delta)\mathbf{q}'|| \geq ||\mathbf{q}''|| \geq ||\mathbf{b}_l||$.