Measures on Boolean Polynomials and Their Applications in Data Mining

Szymon Jaroszewicz [a] Dan A. Simovici [a,*] Ivo Rosenberg [b]

[a] *University of Massachusetts at Boston, Department of Computer Science, Boston, Massachusetts 02125, USA*

[b] *Université de Montréal, C.P. 6128, succ. A, Montréal, P.Q. H3C 3J7, Canada*

Abstract

We characterize measures on free Boolean algebras and we examine the relationships that exist between measures and binary tables in relational databases. It is shown that these measures are completely defined by their values on positive conjunctions, and a formula that yields this value is obtained using the method of indicators. An extension of the notion of support that is well suited for tables with missing values is presented. Finally, we obtain Bonferroni-type inequalities that allow for approximative evaluations of these measures for several types of queries. An approximation algorithm and an analysis of the results produced is also included.

Key words: free Boolean algebra, measure, Bonferroni-type inequality, inclusion-exclusion, missing values, frequent itemset

1 Introduction

The focus of this paper is a study of measures on free Boolean algebras with a finite number of generators (abbreviated as MFBAs). As we shall see, these measures play an important role in query optimization in relational databases, and also, in the study of frequent sets in data mining. We obtain general Bonferroni-type inequalities for sizes of arbitrary Boolean queries.

* Corresponding Author

Email addresses: sj@cs.umb.edu (Szymon Jaroszewicz), dsim@cs.umb.edu (Dan A. Simovici), rosenb@dms.umontreal.ca (Ivo Rosenberg).

URLs: www.cs.umb.edu/∼sj (Szymon Jaroszewicz), www.cs.umb.edu/∼dsim (Dan A. Simovici), www.dms.umontreal.ca/∼rosenb (Ivo Rosenberg).

The origin of our investigation resides in a series of seminal papers by H. Mannila et al. ([1–3]) in which the idea of using supports of attribute sets discovered with a data mining algorithm to obtain the size of a database query was introduced.

We start from the premise that Boolean algebra is the natural framework for estimating the size of queries applied to relational databases. Indeed, the set of conditions that specify a relational algebra selection (or a **where** clause in a **select** SQL phrase) is build from atomic conditions of the form $A\omega a$ or $A\omega B$, where A, B are attributes, a is a value of the domain of the attribute A, and ω is one of the relational operators $<, \leq, >, \geq, =, \neq$. Then, more complex conditions are built using the connective symbols **and**, **or**, **not**. As we shall see, this is precisely the free Boolean algebra over the set of atomic conditions. Thus, the size of the answer to a particular selection p can be regarded as a measure over the free Boolean algebra generated by atomic conditions and the value of this measure on p can be used in query processing in relational databases (see [4], for example).

After introducing basic definitions and notations concerning Boolean algebras and relational databases in the next section, we give a representation result for measures on free Boolean algebras that show that any such functions can be induced by a table.

Section 3 presents an inclusion-exclusion principle for MFBAs using the method of indicator variables. The same section presents Bonferroni-type inequalities [5] that allow us to generate bounds on the value of measures of polynomials. These results are significant for estimations of the size of queries in relational databases, or for support estimations for item sets in data mining. Such issues are explored in Section 4. Also, we extend the notion of support for queries to tables with missing values. Our approach has the advantage of generating values that are probabilistically consistent, unlike the techniques used in [6,7].

We conclude the paper by presenting an algorithm that computes bounds on support of an itemset based on a collection of itemsets with known supports.

2 Measures on The Set of Polynomials and Tables

Let $\mathcal{B} = (B, \mathbf{0}, \mathbf{1}, ^-, \vee, \wedge)$ be a Boolean algebra, where $\mathbf{0}, \mathbf{1} \in B$ are two distinguished elements of $\mathcal{B}$, $^-$ is a unary operation, and $\vee, \wedge$ are two binary associative, commutative, and idempotent operations that satisfy the usual axioms of Boolean algebras (see, for example [8]). Here $\mathbf{0}$ and $\mathbf{1}$ are the least and the largest element of the algebra, respectively.

We define $x^b = x$ if $b = 1$ and $x^b = \bar{x}$ if $b = 0$, for $x \in B$ and $b \in \{0, 1\}$.

It is a well-known fact that a Boolean algebra $\mathcal{B} = (B, \mathbf{0}, \mathbf{1}, \bar{}, \vee, \wedge)$ defines a Boolean ring structure, $\mathcal{B} = (B, \mathbf{0}, \mathbf{1}, \wedge, \oplus)$, where $\wedge$ plays the role of the multiplication, and $\oplus$ the role of addition, where

$$x \oplus y = (x \wedge \bar{y}) \vee (\bar{x} \wedge y)$$

for $x, y \in B$. This ring is unitary, commutative, and has characteristic 2 (since $x \oplus x = \mathbf{0}$ for every x). Also, $\mathbf{1} \oplus x = \bar{x}$.

Let $A = \{a_1, \ldots, a_n\}$ be a set of n variables. The set $\mathsf{pol}(A)$ of *Boolean polynomials of the n variables in A* is defined inductively by:

(1) $\mathbf{0}$, $\mathbf{1}$, and each a_i belong to $\mathsf{pol}(A)$ for $1 \leq i \leq n$;
(2) if p, q belong to $\mathsf{pol}(A)$, then $\bar{p}$, $(p \vee q)$, and $(p \wedge q)$ belong to $\mathsf{pol}(A)$.

If $p, q \in \mathsf{pol}(A)$, then we denote by $(p \oplus q)$ the polynomial $((p \wedge \bar{q}) \vee (\bar{p} \wedge q))$. A Boolean polynomial $(\cdots((p_1 \omega p_2)\omega p_3)\omega \cdots \omega p_n)$ is denoted by $(p_1 \omega p_2 \omega \cdots \omega p_n)$, where $\omega \in \{\vee, \wedge, \oplus\}$.

Let $\mathcal{B} = (B, \mathbf{0}, \mathbf{1}, \bar{}, \vee, \wedge)$ be a Boolean algebra and let $A = \{a_1, \ldots, a_n\}$ be a set of n variables. The n-ary function $f_p : B^n \longrightarrow B$ generated by a polynomial $p \in \mathsf{pol}(A)$ is defined in the usual way. We write $p = q$ for $p, q \in \mathsf{pol}(A)$ if $f_p = f_q$.

Let $\vec{b} = (b_1, \ldots, b_n)$ be a sequence of elements of the set $\{0, 1\}$. An A-minterm is a Boolean polynomial

$$p_{\vec{b}} = a_1^{b_1} \wedge \cdots \wedge a_n^{b_n},$$

The set of A-minterms is denoted by $\mathsf{mint}(A)$. Any Boolean polynomial in $\mathsf{pol}(A)$ can be uniquely written as a disjunction of some subset of A-minterms (up to the order of the disjuncts). This observation implies that the Boolean algebra $\mathsf{pol}(A)$ is isomorphic to the Boolean algebra of collections of subsets of the set A; thus, $\mathsf{pol}(A)$ has 2^{2^n} elements.

For a set of polynomials $M = \{p_1, \ldots, p_n\}$ and $J = \{j_1, \ldots, j_m\} \subseteq \{1, \ldots, n\}$ we denote by p_J the conjunction $p_{j_1} \wedge \cdots \wedge p_{j_m}$. For the special case, when $J = \emptyset$ we write $p_J = \mathbf{1}$.

A *measure* on a Boolean algebra $\mathcal{B} = (B, \mathbf{0}, \mathbf{1}, \bar{}, \vee, \wedge)$ is a non-negative, real-valued function $\mu : B \longrightarrow \mathbb{R}$ such that $\mu(x \vee y) = \mu(x) + \mu(y)$ for every $x, y \in B$ such that $x \wedge y = \mathbf{0}$.

Let $A = \{a_1, \ldots, a_n\}$ be a set of variables. In this context, we find it convenient to use the relational database terminology and we refer to the the members of A as *attributes*. We attach a set $\mathsf{Dom}(a_i)$ to each attribute a_i such that

$|\mathsf{Dom}(a_i)| \geq 2$. The set $\mathsf{Dom}(a_i)$ is the *domain of* a_i.

A table is a triple $\tau = (T, A, \rho)$, where T is the name of the table, $A = \{a_1, \ldots, a_n\}$ is the heading of the table and $\rho = \{t_1, \ldots, t_m\}$ is a finite set of functions of the form $t_i : A \longrightarrow \bigcup_{a \in A} \mathsf{Dom}(a)$ such that $t_i(a) \in \mathsf{Dom}(a)$ for every $a \in A$. Following the relational database terminology we shall refer to these functions as A-tuples, or simply as tuples. If $\mathsf{Dom}(a_i) = \{0, 1\}$ for $1 \leq i \leq n$, then τ is a binary table.

Let $\tau = (T, A, \rho)$ be a binary table. A *query on the table* τ is a Boolean polynomial in $\mathsf{pol}(A)$. This definition of queries is a formalization of the usual notion of queries in databases.

Example 2.1 To retrieve in SQL all tuples t of τ such that at least two of $t(a_1), t(a_2)$ and $t(a_3)$ equal 1 we write:

select * **from** T
where $(a_1 = 1$ **and** $a_2 = 1)$
or $(a_2 = 1$ **and** $a_3 = 1)$
or $(a_1 = 1$ **and** $a_3 = 1)$

The condition specified in the **where** clause of this **select** phrase corresponds to the polynomial $(a_1 \wedge a_2) \vee (a_2 \wedge a_3) \vee (a_1 \wedge a_3)$. □

A query p defines a table (T_p, A, ρ_p), where ρ_p is is defined inductively as follows:

(1) $\rho_{\mathbf{0}} = \emptyset$ and $\rho_{\mathbf{1}} = \rho$;
(2) if $p = a_i$, then $\rho_p = \{t \in \rho | t(a_i) = \mathbf{1}\}$;
(3) if $p = \bar{q}$, then $\rho_p = \rho - \rho_q$;
(4) if $p = (q_1 \vee q_2)$, then $\rho_p = \rho_{p_1} \cup \rho_{p_2}$ and,
(5) if $p = (q_1 \wedge q_2)$, then $\rho_p = \rho_{p_1} \cap \rho_{p_2}$.

It is easy to see that for a conjunction

$$p = a_{i_1}^{b_1} \wedge \cdots \wedge a_{i_m}^{b_m},$$

where $b_i \in \{0, 1\}$ for $1 \leq i \leq m$, the set ρ_p consists of those tuples t such that $t(a_{i_\ell}) = b_\ell$ for $1 \leq \ell \leq m$.

Theorem 2.2 *A function $\mu : \mathsf{pol}(A) \longrightarrow \mathbb{N}$ is a measure if and only if there exists a binary table $\tau = (T, A, \rho)$ such that $\mu(p) = |\rho_p|$ for all $p \in \mathsf{pol}(A)$.*

Proof. Suppose that $\tau = (T, A, \rho)$ is a table. Define the mapping $\mu_\tau : \mathsf{pol}(A) \longrightarrow \mathbb{R}$ by $\mu(p) = |\rho_p|$ for every $p \in \mathsf{pol}(A)$. Let p, q be two polynomials such that $(p \wedge q) = \mathbf{0}$. Then, $\mu_\tau(p \vee q) = |\rho_{p \vee q}| = |\rho_p \cup \rho_q|$. Since

$p \wedge q = \mathbf{0}$ we have $\rho_p \cap \rho_q = \emptyset$, so $\mu_\tau(p \vee q) = \mu_\tau(p) + \mu_\tau(q)$. Thus, μ_τ is a measure on $\mathsf{pol}(A)$.

Conversely, let μ be a measure on $\mathsf{pol}(A)$, where $A = \{a_1, \ldots, a_n\}$. If $\vec{b} = (b_1, \ldots, b_n) \in \{0,1\}^n$, $p_{\vec{b}} = a_1^{b_1} \wedge \cdots \wedge a_n^{b_n}$ is a minterm and $\mu(p_{\vec{b}}) = k$ consider a set $\sigma_{p_{\vec{b}}}$ of k tuples $t^1_{\vec{b}}, \ldots, t^k_{\vec{b}}$, where $t^j_{\vec{b}}(a_i) = b_i$ for every i, j, $1 \leq j \leq k$, and $1 \leq i \leq n$. Define the table $\tau_\mu = (T, A, \rho_\mu)$, where $\rho = \bigcup\{\sigma_{p_{\vec{b}}} | p_{\vec{b}} \in \mathsf{mint}(A)\}$.

We claim that $\mu(p) = |\rho_p|$ for every polynomial $p \in \mathsf{pol}(A)$. Suppose that p can be expressed as a disjunction of minterms $p = p_{\vec{b}_1} \vee \cdots \vee p_{\vec{b}_k}$, where $\vec{b}_1, \ldots, \vec{b}_k \in \{\mathbf{0}, \mathbf{1}\}^n$. Then, $\mu(p) = \sum_{j=1}^k \mu(p_{\vec{b}_j})$, because $p_{\vec{b}_l} \wedge p_{\vec{b}_h} = \mathbf{0}$ when $l \neq h$. On the other hand, $|\rho_p| = |\bigcup_{j=1}^k \rho_{p_{\vec{b}_j}}| = \sum_{j=1}^k |\rho_{p_{\vec{b}_j}}|$, so $\mu(p) = |\rho_p|$. ∎

We shall refer to μ_τ as the *measure induced by the table* τ on $\mathsf{pol}(A)$.

Measures induced by tables are generated by pseudo-Boolean functions which range over the set $\mathbb{N}$ (see [9]). Namely, let $A = \{a_1, \ldots, a_n\}$ be a set of n attributes. Define the pseudo-Boolean function $f : \{0,1\}^n \longrightarrow \mathbb{N}$ by $f(b_1, \ldots, b_n) = \mu_\tau(p_{b_1,\ldots,b_n})$. Then, it is easy to verify that for every polynomial $p \in \mathsf{pol}(A)$ we have

$$\mu_\tau(p) = \sum\{f(\vec{b}) \mid p_{\vec{b}} \in \mathsf{mint}(A) \text{ and } p_{\vec{b}} \leq p\}. \tag{1}$$

Conversely, if $f : \{0,1\}^n \longrightarrow \mathbb{N}$ is an integer-valued, non-negative pseudo-Boolean function, then the function μ defined as in Equality (1) is clearly a measure on $\mathsf{pol}(A)$.

In the next section we regard the set of minterms $\mathsf{mint}(A)$ as a sample space and each polynomial $p \in \mathsf{pol}(A)$ as an event on this sample space. The event p occurs in $p_{\vec{b}}$ if $p_{\vec{b}} \leq p$. Thus, if μ is a measure on $\mathsf{pol}(A)$, then the mapping $P_\mu : \mathsf{pol}(A) \longrightarrow \mathbb{R}$ given by $P_\mu(p) = \frac{\mu(p)}{\mu(\mathbf{1})}$ is a probability on $\mathsf{pol}(A)$.

3 An Inclusion-Exclusion Principle for MFBAs

Let $A = \{a_1, \ldots, a_n\}$ be a set of n variables. If $I = \{i_1, \ldots, i_m\}$ is a subset of $\{1, \ldots, n\}$, then we denote the conjunction $a_{i_1} \wedge \cdots \wedge a_{i_m}$ by a_I.

It is known that every polynomial $p \in \mathsf{pol}(A)$ can be uniquely written as

$$p = \sum_I^{\oplus} c_I \wedge a_I,$$

where the summation $\sum^{\oplus}$ involves the "exclusive or" operation $\oplus$ and is extended to all subsets I of $\{1, \ldots, n\}$. The coefficients c_I belong to the set $\{\mathbf{0}, \mathbf{1}\}$.

Thus, for a measure μ on $\mathsf{pol}(A)$ it is interesting to evaluate $\mu(p_1 \oplus p_2 \oplus \cdots \oplus p_m)$, where $p_1, \ldots, p_m$ are polynomials in $\mathsf{pol}(A)$.

The *indicator random variable* of a polynomial p (see [5]) is the variable I_p defined by:

$$I_p(p_{\vec{b}}) = \begin{cases} 1 & \text{if } p_{\vec{b}} \leq p \\ 0 & \text{otherwise.} \end{cases}$$

for $p_{\vec{b}} \in \mathsf{mint}(A)$. Note that the expected value $E[I_p]$ of I_p equals $P_\mu(p)$.

If $M = \{p_1, \ldots, p_n\}$ is a set of polynomials and $J = \{j_1, \ldots, j_m\} \subseteq \{1, \ldots, n\}$, then $p_{M,J}$ is the polynomial $p_{j_1} \wedge \ldots \wedge p_{j_m}$; it is easy to see that $I_{p_{M,J}} = I_{p_{j_1}} \cdots I_{p_{j_m}}$.

For a set of polynomials M define $S^\mu_{M,k}$ as:

$$S^\mu_{M,k} = \sum\{P_\mu(p_{M,K}) \mid |K| = k\}.$$

The number of k-subsets K of M such that $p_{M,K}$ holds is given by the random variable $\sum\{I_{p_{M,K}} \mid |K| = k\}$. By the previous observation

$$S^\mu_{M,k} = \sum\{E(I_{p_{M,K}}) \mid |K| = k\} = E\left[\sum\{I_{p_{M,K}} \mid |K| = k\}\right].$$

Let ν_M be the random variable on $\mathsf{mint}(A)$ such that $\nu_M(p_{\vec{b}}) = |\{p_i \in M \mid p_{\vec{b}} \leq p_i\}|$. Note that ν_M gives the number of events in M that hold and, therefore, the random variable $\binom{\nu_M}{k}$ gives the number of k-subsets Q of M such that $p_{M,Q}$ holds, which means that $\binom{\nu_M}{k} = \sum\{I_{p_{M,K}} \mid |K| = k\}$, and

$$S^\mu_{M,k} = E\left[\binom{\nu_M}{k}\right]. \tag{2}$$

The equality (2) is the basis of the method of indicators, that is a method of proving probabilistic identities by taking expectations of their non-probabilistic counterparts, see [5] for details.

Theorem 3.1 *Let $\mu : \mathsf{pol}(A) \longrightarrow \mathbb{R}$ be a measure on the free Boolean algebra $\mathsf{pol}(A)$, where $A = \{a_1, \ldots, a_n\}$. If $M = \{p_1, \ldots, p_m\}$ is a set of m polynomials of $\mathsf{pol}(A)$, then*

$$\mu(p_1 \oplus \cdots \oplus p_m) = \mu(\mathbf{1}) \cdot \sum_{k=1}^{m} (-2)^{k-1} \cdot S^\mu_{M,k}. \tag{3}$$

Proof. Let $a \in \mathbb{N}$, note that $(-1)^a = \sum_{k=0}^{a} (-2)^k \binom{a}{k}$, which yields, after

elementary transformations:

$$\sum_{k=1}^{a}(-2)^{k-1}\binom{a}{k} = (-1)^a - 1 = \begin{cases} 0 & \text{if } a \text{ is even} \\ 1 & \text{if } a \text{ is odd.} \end{cases}$$

This implies

$$\sum_{k=1}^{\nu_M}(-2)^{k-1}\binom{\nu_M}{k} = \sum_{k=1}^{|M|}(-2)^{k-1}\binom{\nu_M}{k} = \begin{cases} 0 & \text{if } \nu_M \text{ is even} \\ 1 & \text{if } \nu_M \text{ is odd.} \end{cases}$$

By taking expectations of both sides, and using equality (2) we get

$$E\left[\sum_{k=1}^{|M|}(-2)^{k-1}\binom{\nu_M}{k}\right] = \sum_{k=1}^{|M|}(-2)^{k-1}S^{\mu}_{M,k} = P_\mu(\nu_M \text{ is odd}) = P_\mu(p_1 \oplus \ldots \oplus p_m).$$

which yields the desired equality. ∎

Corollary 3.2 *Let $\mu, \mu' : \mathsf{pol}(A) \longrightarrow \mathbb{R}$ be two measures on the free Boolean algebra $\mathsf{pol}(A)$, where $A = \{a_1, \ldots, a_n\}$. If $\mu(p) = \mu'(p)$ for every conjunction p of the form $p = a_{i_1} \wedge \cdots \wedge a_{i_m}$, then $\mu = \mu'$.*

Proof. The result follows immediately from Theorem 3.1. ∎

Example 3.3 Consider the "majority polynomial" $p_{maj} = (a_1 \wedge a_2) \vee (a_2 \wedge a_3) \vee (a_1 \wedge a_3)$. For $f_{p_{maj}}$ we have $f_{p_{maj}}(x_1, x_2, x_3) = 1$ if and only if at least two of its arguments are equal to 1. Note that

$$p_{maj} = (a_1 \wedge a_2) \oplus (a_2 \wedge a_3) \oplus (a_1 \wedge a_3).$$

Theorem 3.1 allows us to write

$$\begin{aligned} \mu(p_{maj}) = {} & \mu(a_1 \wedge a_2) + \mu(a_2 \wedge a_3) + \mu(a_1 \wedge a_3) \\ & -2\mu((a_1 \wedge a_2) \wedge (a_2 \wedge a_3)) - 2\mu((a_1 \wedge a_2) \wedge (a_1 \wedge a_3)) \\ & -2\mu((a_2 \wedge a_3) \wedge (a_1 \wedge a_3)) + 4\mu((a_1 \wedge a_2) \wedge (a_2 \wedge a_3) \wedge (a_1 \wedge a_2)) \\ = {} & \mu(a_1 \wedge a_2) + \mu(a_2 \wedge a_3) + \mu(a_1 \wedge a_3) - 2\mu(a_1 \wedge a_2 \wedge a_3). \end{aligned}$$

□

Corollary 3.2 shows that the values of a measure on $\mathsf{pol}(A)$ are completely determined by its values on positive conjunctions of the form a_I for $I \subseteq \{1, \ldots, n\}$. Note that the contribution of every tuple of a table $\tau = (T, A, \rho)$ of the form $(b_1, \ldots, b_n)$ to the value of $\mu_\tau(I)$ equals 1 for every set I such that $I \subseteq \{i \in \{1, \ldots n\} \mid b_i = 1\}$.

Next, we obtain Bonferroni-type inequalities [5] that give bounds on the value of $\mu(p_1 \oplus \ldots \oplus p_m)$. To this end we need the following technical result:

Define W_b^a for $a, b \in \mathbb{N}$ and $b \leq a$ as $W_b^a = \sum_{k=b}^{a}(-2)^{k-1}\binom{a}{k}$. Alternatively, W_b^a can be written as

$$W_b^a = (-2)^{b-1}\sum_{k=b}^{a}(-2)^{k-b}\binom{a}{k} = (-2)^{b-1}\sum_{\ell=0}^{a-b}(-2)^{\ell}\binom{a}{b+\ell}.$$

Lemma 3.4 *The signs of the members of the sequence* $(W_b^a, W_{b+1}^a, \ldots, W_a^a)$ *are alternating.*

Proof. Define

$$U_b^a = \sum_{\ell=0}^{a-b}(-2)^{\ell}\binom{a}{b+\ell}$$

for $a, b \in \mathbb{N}$ and $b \leq a$. Since $W_b^a = (-2)^{b-1}U_b^a$ it suffices to prove that the numbers U_b^a are have all the same sign.

Note that $U_b^b = 1$ for $b \in \mathbb{N}$. We can write:

$$\begin{aligned}
U_b^a =& \sum_{\ell=0}^{a-b}(-2)^{\ell}\binom{a}{b+\ell} \\
=& \binom{a-1}{b} + \binom{a-1}{b-1} - 2\binom{a-1}{b+1} - 2\binom{a-1}{b} \\
&+2^2\binom{a-1}{b+2} + 2^2\binom{a-1}{b+1} - 2^3\binom{a-1}{b+3} - 2^3\binom{a-1}{b+2} + \cdots \\
&\vdots \\
&+(-2)^{a-b-1}\binom{a-1}{a-1} + (-2)^{a-b-1}\binom{a-1}{a-2} + (-2)^{a-b}\binom{a-1}{a-1} \\
=& \binom{a-1}{b-1} - U_b^{a-1}.
\end{aligned}$$

Thus, we obtain

$$U_b^a = \binom{a-1}{b-1} - U_b^{a-1}. \tag{4}$$

We claim that $0 \leq U_b^a \leq \binom{a}{b-1}$ for $0 \leq b \leq a$. This can be shown by induction on $a \geq b$. The basis step $a = b$ is immediate. Suppose that the double inequality holds for $a-1$, that is, $0 \leq U_b^{a-1} \leq \binom{a-1}{b-1}$. Then, it is clear that $U_b^a \geq 0$. To show that $U_b^a \leq \binom{a}{b-1}$ we need to verify that $\binom{a-1}{b-1} - U_b^{a-1} \leq \binom{a}{b-1}$. Since $\binom{a}{b-1} = \binom{a-1}{b-1} + \binom{a-1}{b-2}$ for $b \geq 2$ the last inequality follows. ∎

Theorem 3.5 *For any* $r, s \in \mathbb{N}$ *we have:*

$$\mu(\mathbf{1}) \cdot \sum_{k=1}^{2r}(-2)^{k-1}S_k^{\mu} \leq \mu(p_1 \oplus \ldots \oplus p_m) \leq \mu(\mathbf{1}) \cdot \sum_{k=1}^{2s+1}(-2)^{k-1}S_k^{\mu}.$$

Proof. By equality (2) and Lemma 3.4 we get that for any $r, s \in \mathbb{N}$

$$\sum_{k=1}^{2r}(-2)^{k-1}\binom{a}{k} \leq \sum_{k=1}^{a}(-2)^{k-1}\binom{a}{k} \leq \sum_{k=1}^{2s+1}(-2)^{k-1}\binom{a}{k},$$

implying

$$\sum_{k=1}^{2r}(-2)^{k-1}\binom{\nu_M}{k} \leq \sum_{k=1}^{|M|}(-2)^{k-1}\binom{\nu_M}{k} \leq \sum_{k=1}^{2s+1}(-2)^{k-1}\binom{\nu_M}{k}.$$

By applying expectations and using equality (2) we get the desired result. ∎

Example 3.6 Consider a table τ given below

a_1	a_2	a_3
0	0	0
0	1	0
1	0	0
0	0	0
0	1	0
1	0	1
0	1	1
1	1	0
1	1	0
0	1	1
1	0	1
1	1	0
1	1	1

and the majority polynomial p_{maj} from Example 3.3. We have $\mu(a_1 \wedge a_2) = 4$, $\mu(a_1 \wedge a_3) = 3$, $\mu(a_2 \wedge a_3) = 3$, giving $\mu(p_{maj}) \leq 10$. Also $\mu((a_1 \wedge a_2) \wedge (a_1 \wedge a_3)) = \mu((a_1 \wedge a_2) \wedge (a_2 \wedge a_3)) = \mu((a_1 \wedge a_3) \wedge (a_2 \wedge a_3)) = 1$ giving $\mu(p_{maj}) \geq 4$. The true value of $\mu(p_{maj})$ is 8. □

4 Applications in Data Mining and Database Query Optimization

In this section we examine the accuracy of the computation of the size of a query using the inclusion-exclusion principle. Then, we extend the notion of support for queries that apply to tables with missing values.

In database query optimization and in data mining, it is often necessary to estimate the number of rows in a database table satisfying a given query. Unfortunately, in most cases, the exact number of rows satisfying a query cannot be computed exactly and has to be estimated (usually using the assumption

of statistical independence between attributes). Following datamining terminology we will occasionally refer to sets of attributes as itemsets.

Let $\tau = (T, A, \rho)$ be a binary table and let $K = \{a_{k_1}, \ldots, a_{k_m}\}$ be a set of attributes, $K \subseteq A$. The support of the set K relative to the table τ is the value of the probability $P_{\mu_\tau}(a_{k_1} \wedge \cdots \wedge a_{k_m})$:

$$\mathtt{supp}_\tau(K) = \frac{|\{t \in \rho \mid t(a) = \mathbf{1} \text{ for all } a \in K\}|}{|\rho|}.$$

In other words, the support of an attribute set K in the table τ is defined by the value of the measure induced by the table on the Boolean polynomial that describes the attribute set. By extension, we can regard the number $\frac{\mu_\tau(q)}{\mu_\tau(\mathbf{1})}$ as the support of the query q and we denote this number by $\mathtt{supp}_\tau(q)$. Indeed, if $q \in \mathsf{pol}(A)$ is a query involving a table $\tau = (T, A, \rho)$ such that q can be written as

$$q = c \oplus \overset{\oplus}{\sum_{I \in \mathcal{I}}} a_I,$$

where $c \in \{\mathbf{0}, \mathbf{1}\}$ and $\mathcal{I}$ is a collection of subsets of $\{1, \ldots, n\}$, then $\mathtt{supp}_\tau(q)$ can be obtained from Theorem 3.1 using the numbers $\mathtt{supp}_\tau(a_I)$. Methods that obtain approximative estimations of query sizes been proposed [2], including the use of Maximum Entropy Principle. An open problem raised was estimating the quality of such an approximation.

The computation of the size of the query using Theorem 3.1 can be often simplified if there is a known maximal number of 1 components in the tuples of the table. For example, in a store that sells 1000 items (corresponding to 1000 attributes in a table that contains the records of purchases) it is often the case that we can use an empirical limit of, say, 8 items per tuple. In this case, conjunctions that contain more than 8 conjuncts can be discarded and the estimation is considerably simplified. Even, if such an upper bound cannot be imposed apriori, it is often the case that we can discard large conjunctions (which have low support). However, there are some risks when approximations of this nature are performed due to the the large values of coefficients that multiply the supports for large conjunctions.

Indeed, consider the tables $\tau^n_{odd} = (T_o, A, \rho_{odd})$, $\tau^n_{even} = (T_e, A, \rho_{even})$, where

$$\rho_{odd} = \{t \in \mathsf{Dom}(A) \mid n_1(t) \text{ is odd}\} \text{ and } \rho_{even} = \{t \in \mathsf{Dom}(A) \mid n_1(t) \text{ is even}\},$$

where $n_1(t)$ denotes the number of attributes equal to 1 in tuple t and $|A| = n$.

Note that for proper subset K of A, we have $\mathtt{supp}_{\tau_{odd}}(K) = \mathtt{supp}_{\tau_{even}}(K)$,

while

$$\mathtt{supp}_{\tau^n_{odd}}(A) = \begin{cases} 1 & \text{if } n \text{ is odd} \\ 0 & \text{otherwise,} \end{cases} \text{ and } \mathtt{supp}_{\tau^n_{even}}(A) = \begin{cases} 1 & \text{if } n \text{ is even} \\ 0 & \text{otherwise.} \end{cases}$$

Thus, from the point of view of the supports of any proper subset of the attribute set the tables τ^n_{odd} and τ^n_{even} are indiscernible. However, the support of certain queries can be vastly different on these tables. For example, consider the polynomial $p = a_1 \oplus a_2 \oplus \ldots \oplus a_n$. We have $\mathtt{supp}_{\tau^n_{odd}}(p) = 1$ and $\mathtt{supp}_{\tau^n_{even}}(p) = 0$. So, ignoring the term that corresponds to the support for a single attribute set (note that this is also the attribute set with the smallest possible support) has a huge impact on $\mu_\tau(p)$. Note that the result is consistent with Theorem (3.1) which gives the set of attributes A a coefficient 2^{n-1}. We stress however that the negative result above does not rule out practical applicability of approximating the values of μ_τ since the parity function query used above is by no means a typical database query.

Frequently, real world datasets contain missing values; this makes important to adequately address this issue. Here we present a generalization of the notion of support which takes missing values into account. The idea is related to the *hot deck imputation* of missing values where each missing value is replaced by a value randomly drawn from some distribution (see [10]).

Suppose that $\tau = (T, A, \rho)$ is a table such that $A = \{a_1, \ldots, a_n\}$ and $\mathsf{Dom}(a_i) = \{0, u, 1\}$ for $1 \leq i \leq n$. The symbol u represents *null values*, that is, values that are missing or undefined. With every attribute $a_i \in A$ we associate a real number $\alpha_i \in [0, 1]$. Intuitively, this number corresponds to the probability of $a_i = 1$, and can be obtained using the non-missing values for the attribute or based on background knowledge.

Let α be a non-negative number, and let $b, c \in \{0, 1\}$. Define

$$\alpha^{\langle b,c \rangle} = \begin{cases} \alpha & \text{if } b = 1 \text{ and } c = 0 \\ 1 - \alpha & \text{if } b = 0 \text{ and } c = 0 \\ 1 & \text{if } c = 1, \end{cases} \text{ and } b^{(c)} = \begin{cases} b & \text{if } c = 1 \\ u & \text{if } c = 0, \end{cases}$$

For a table $\tau = (T, A, \rho)$ let $\mu^u_\tau : \mathsf{pol}(A) \longrightarrow \mathbb{R}$ be defined as follows. For a minterm $a_1^{b_1} \wedge \cdots \wedge a_n^{b_n}$ let

$$\mu^u_\tau(a_1^{b_1} \wedge \cdots \wedge a_n^{b_n}) = \sum_{(c_1,\ldots,c_n) \in \{0,1\}^n} \prod_{i=1}^{n} \alpha_i^{\langle b_i, c_i \rangle} \cdot \frac{|\{t \in \rho \mid t(a_i) = b_i^{(c_i)} \text{ for } 1 \leq i \leq n\}|}{|\rho|}$$

For an arbitrary boolean polynomial p define

$$\mu^u_\tau(p) = \sum_{p_{\vec{d}} \in \mathsf{mint}_p} \mu^u_\tau(p_{\vec{d}})$$

where mint_p is the set of minterm implicants of p.

Theorem 4.1 *For every table $\tau = (T, H, \rho)$, μ^u_τ is a measure on $\mathsf{pol}(A)$.*

Proof. Since μ^u_τ is clearly non-negative, it remains to be shown that $\mu^u_\tau(p_1 \vee p_2) = \mu^u_\tau(p_1) + \mu^u_\tau(p_2)$ for every $p_1, p_2 \in \mathsf{pol}(A)$ such that $p_1 \wedge p_2 = \mathbf{0}$. Note that if $p_1 \wedge p_2 = \mathbf{0}$ then $\mathsf{mint}_{p_1} \cap \mathsf{mint}_{p_2} = \emptyset$, and

$$\mu^u_\tau(p_1 \vee p_2) = \sum_{p_{\vec{d}} \in \mathsf{mint}_{p_1}} \mu^u_\tau(p_{\vec{d}}) + \sum_{p_{\vec{d}} \in \mathsf{mint}_{p_2}} \mu^u_\tau(p_{\vec{d}}) = \mu^u_\tau(p_1) + \mu^u_\tau(p_2).$$

∎

Example 4.2 Let $n = 2$, we have:

$$\begin{aligned}
&\mu^u_\tau(a_1 \oplus a_2)\\
&= \mu^u_\tau(\bar{a_1} \wedge a_2) + \mu^u_\tau(a_1 \wedge \bar{a_2})\\
&= \mathtt{supp}_\tau(a_1 = 0 \wedge a_2 = 1) + (1-\alpha_1)\mathtt{supp}_\tau(a_1 = u \wedge a_2 = 1)\\
&\quad + \alpha_2 \mathtt{supp}_\tau(a_1 = 0 \wedge a_2 = u) + (1-\alpha_1)\alpha_2 \mathtt{supp}_\tau(a_1 = a_2 = u)\\
&\quad + \mathtt{supp}_\tau(a_1 = 1 \wedge a_2 = 0) + \alpha_1 \mathtt{supp}_\tau(a_1 = u \wedge a_2 = 0)\\
&\quad + (1-\alpha_2)\mathtt{supp}_\tau(a_1 = 1 \wedge a_2 = u) + \alpha_1(1-\alpha_2)\mathtt{supp}_\tau(a_1 = a_2 = u).
\end{aligned}$$

□

The benefit of using arbitrary measures instead of probabilities or supports in previous sections is that results on inclusion-exclusion principle automatically apply to μ^u_τ. Also, the fact that μ^u_τ is a measure makes the proof of the following theorem straightforward.

Theorem 4.3 *For every table $\tau = (T, A, \rho)$ such that $A = \{a_1, \ldots, a_n\}$ and $\mathsf{Dom}(a_i) = \{0, u, 1\}$ for $1 \le i \le n$, and every collection of sets of attributes $\mathcal{A} = \{a_{I_1}, \ldots, a_{I_m} \mid I_j \subseteq \{1, \ldots, n\}\}$ there is a probability distribution P over A such that for every $a_{I_r} \in \mathcal{A}$, $P\{\bigwedge_{j \in I_r}(a_j = 1)\} = \mu^u_\tau(\bigwedge_{j \in I_r}(a_j = 1))/|\rho|$.*

Proof. We prove the theorem by showing that $\mu^u_\tau/|\rho|$ is a probability distribution. Since μ^u_τ is a measure, it suffices to show that $\mu^u_\tau(\mathbf{1}) = |\rho|$. For any $a_i \in A$ we have:

$$\begin{aligned}\mu^u_\tau(\mathbf{1}) &= \mu^u_\tau(a_i \vee \bar{a_i}) = \mu^u_\tau(a_i) + \mu^u_\tau(\bar{a_i}) \\ &= \mathtt{supp}_\tau(a_i = 1) + \alpha_i \mathtt{supp}_\tau(a_i = u) \\ &\quad + \mathtt{supp}_\tau(a_i = 0) + (1 - \alpha_i)\mathtt{supp}_\tau(a_i = u) \\ &= \mathtt{supp}_\tau(a_i = 1) + \mathtt{supp}_\tau(a_i = 0) + \mathtt{supp}_\tau(a_i = u) = |\rho|.\end{aligned}$$

∎

The importance of the above theorem is that if we use some datamining algorithm (e.g. Apriori) to find μ^u_τ for a collection of sets of attributes, then their values of μ^u_τ are probabilistically consistent. Other approaches to mining frequent itemsets in the presence of missing values can be found in [6,7]. However, both these approaches can produce probabilistically inconsistent results. Specifically, the technique used in [6] is to count the support of an itemset only on the portion of the table where it is valid. For example, consider the table $\tau = (T, a_1a_2, \rho)$, given by

T

a_1	a_2
1	1
1	u
0	u
0	u

Using the method from [6] the support of attribute a_2 is counted only in the first row, giving $\mathtt{supp}_\tau(a_2) = 100\%$. Similarly $\mathtt{supp}_\tau(a_1) = 50\%$, and $\mathtt{supp}_\tau(a_1a_2) = 100\%$, but this means $\mathtt{supp}_\tau(a_1a_2) > \mathtt{supp}_\tau(a_1)$, which is impossible. In the method proposed in [7] the probability for each attribute is estimated from the part of the data where the attribute is defined. When computing how much support does a row with a missing value contribute for an itemset, this probabilities are summed for each attribute (see [7] for details). In the table above this will give $\mathtt{supp}_\tau(a_1) = 50\%$, $\mathtt{supp}_\tau(a_2) = 100\%$, and $\mathtt{supp}_\tau(a_1a_2) = [(0.5{\cdot}1{+}0.5{\cdot}1){+}(0.5{\cdot}1{+}0.5{\cdot}1){+}2(0.5{\cdot}0{+}0.5{\cdot}1)]/4 = 75\%$, and $\mathtt{supp}_\tau(a_1a_2) > \mathtt{supp}_\tau(a_1)$. Using our μ^u_τ measure with $\alpha_2 = 1$ gives consistent values of $\mathtt{supp}_\tau(a_1) = 50\%$, $\mathtt{supp}_\tau(a_2) = 100\%$, and $\mathtt{supp}_\tau(a_1a_2) = 50\%$.

5 Support Approximations Using Bonferroni-type Inequalities

The question of estimating supports of general Boolean expressions based on supports of frequent itemsets discovered by a datamining algorithm was initiated in [1]. The accuracy of this estimation (using the inclusion-exclusion

principle) is influenced by the supports of the frequent items set; when, for various reasons, some of these supports are missing this accuracy may be compromised. The problem has been addressed in [11,12] but the results presented there can be applied only for the case when we know supports of all itemsets up to a given size. This is usually not the case with datamining algorithms which compute supports of only some of the itemsets of a given size. A similar problem has been addressed in the area of statistical data protection, where it is important to assure that inferences about individual cases cannot be made from marginal totals (see [13,14] for an overview). Those methods concentrate on obtaining the most accurate bounds possible (in order to rule out information disclosure), computational efficiency being a secondary concern. Algorithms usually involve repeated iterations over full contingency tables [14], branch and bound search [13] or numerous applications of linear programming.

We use recursively Bonferonni inequalities to estimate supports of missing itemsets. In their original form the inequalities require that we know supports of all itemsets up to a given size. We address the problem by using the inequalities recursively to estimate supports of missing itemsets. The advantage of Bonferroni inequalities is that we can choose an arbitrary limit on the size of the marginals involved, thus allowing for trading off accuracy for speed. Our experiments revealed that it is possible to obtain good bounds even if only marginals of small size are used.

Example 5.1 Consider a binary table τ whose heading is $A = abc$ and assume that the distribution of the values of the tuples in this table is given by:

a	0	0	0	0	1	1	1	1
b	0	0	1	1	0	0	1	1
c	0	1	0	1	0	1	0	1
Frequency	0	0	0.1	0.25	0.1	0.25	0.05	0.25

A run of the Apriori algorithm ([15]) on a dataset conforming to that distribution, with the minimum support of 0.35 will yield the following itemsets:

Itemset	a	b	c	ac	bc
Support	0.65	0.65	0.75	0.5	0.5

To estimate the unknown support of the itemset abc we can use Bonferroni inequalities of the form:

$$\begin{aligned}
\mathtt{supp}_\tau(abc) &\geq 1 - \mathtt{supp}_\tau(\bar{a}) - \mathtt{supp}_\tau(\bar{b}) - \mathtt{supp}_\tau(\bar{c}), && (5)\\
\mathtt{supp}_\tau(abc) &\leq 1 - \mathtt{supp}_\tau(\bar{a}) - \mathtt{supp}_\tau(\bar{b}) - \mathtt{supp}_\tau(\bar{c}) && (6)\\
&\quad + \mathtt{supp}_\tau(\bar{a}\bar{b}) + \mathtt{supp}_\tau(\bar{a}\bar{c}) + \mathtt{supp}_\tau(\bar{b}\bar{c}).
\end{aligned}$$

Note that since the support of ab is below the minimum support its value is not returned by the Apriori algorithm and this creates a problem for this estimation. All the itemset supports, except for $\mathtt{supp}_\tau(\bar{a}\bar{b})$, in the previous expression can be determined from known itemset supports using inclusion-exclusion principle. For example, we have

$$\mathtt{supp}_\tau(\bar{a}\bar{c}) = 1 - \mathtt{supp}_\tau(a) - \mathtt{supp}_\tau(c) + \mathtt{supp}_\tau(ac) = 0.1.$$

Since all needed probabilities are known exactly, the lower bound (5) is easy to compute giving

$$\mathtt{supp}_\tau(abc) \geq 1 - 0.35 - 0.35 - 0.25 = 0.05.$$

To compute the upper bound we proceed as follows.

Since $\mathtt{supp}_\tau(\bar{a}\bar{b})$ is not known, we apply Bonferroni inequalities recursively to get an upper bound for it. We have

$$\mathtt{supp}_\tau(\bar{a}\bar{b}) = 1 - \mathtt{supp}_\tau(a) - \mathtt{supp}_\tau(b) + \mathtt{supp}_\tau(ab),$$

and, since ab is not frequent, we know that its support is less than the 0.35 minimum support, giving

$$\mathtt{supp}_\tau(\bar{a}\bar{b}) < 1 - \mathtt{supp}_\tau(a) - \mathtt{supp}_\tau(b) + \mathtt{minsupp} = 0.05.$$

Substituting into (7) we get

$$\begin{aligned}
\mathtt{supp}_\tau(abc) &< 1 - \mathtt{supp}_\tau(\bar{a}) - \mathtt{supp}_\tau(\bar{b}) - \mathtt{supp}_\tau(\bar{c})\\
&\quad + 0.05 + \mathtt{supp}_\tau(\bar{a}\bar{c}) + \mathtt{supp}_\tau(\bar{b}\bar{c})\\
&= 1 - 0.35 - 0.35 - 0.25 + 0.05 + 0.1 + 0.1 = 0.3.
\end{aligned}$$

Note that both bounds are not trivial since the lower bound is greater than 0, and the upper bound is less than the minimum support. □

5.1 A Recursive Procedure for Computing Bonferroni Bounds from Frequent Itemsets

Since the Apriori algorithm only discovers supports of itemsets (as opposed to other types of queries), we need to express all inequalities in terms of supports of itemsets.

Theorem 5.2 *Let $q_1, \ldots, q_m$ be m queries in $\mathsf{pol}(A)$. The following inequalities hold for any $t \in \mathbb{N}$:*

$$\sum_{k=0}^{2t+1}(-1)^k \sum_{r<i_1<\ldots<i_k\leq m} \mathtt{supp}_\tau(q_1 \wedge \ldots \wedge q_r \wedge q_{i_1} \wedge \ldots \wedge q_{i_k}) \leq \mathtt{supp}_\tau(q_1 \wedge \ldots \wedge q_r \wedge \bar{q}_{r+1} \wedge \ldots \wedge \bar{q}_m) \leq \sum_{k=0}^{2t}(-1)^k \sum_{r<i_1<\ldots<i_k\leq m} \mathtt{supp}_\tau(q_1 \wedge \ldots \wedge q_r \wedge q_{i_1} \wedge \ldots \wedge q_{i_k}).$$

Proof. By Rényi's Theorem [16] it suffices to prove the claim for $q_i \in \{\mathbf{1}, \mathbf{0}\}$ for all $1 \leq i \leq m$. When $q_i = \mathbf{0}$ for some $1 \leq i \leq r$, then both sides of the inequalities reduce to 0 and the result is immediate. For the case $q_i = \mathbf{1}$ for all $1 \leq i \leq r$ we have $\mathtt{supp}_\tau(q_1 \wedge \ldots \wedge q_r \bar{q}_{r+1} \wedge \ldots \wedge \bar{q}_m) = \mathtt{supp}_\tau(\bar{q}_{r+1} \ldots \bar{q}_m)$, and for all k and for all $r < i_1 < \ldots < i_k \leq m$, $\mathtt{supp}_\tau(q_1 \wedge \ldots \wedge q_r \wedge q_{i_1} \wedge \ldots \wedge q_{i_k}) = \mathtt{supp}_\tau(q_{i_1} \wedge \ldots \wedge q_{i_k})$. The result now follows from Bonferroni inequalities. □

Corollary 5.3 *Let $a_1 \wedge a_2 \wedge \cdots \wedge a_r \wedge \bar{a}_{r+1} \wedge \bar{a}_{r+2} \wedge \cdots \wedge \bar{a}_m$ be a minterm. The following inequalities hold for any natural number t:*

$$\sum_{k=0}^{2t+1}(-1)^k \sum_{r<i_1<\ldots<i_k\leq m} \mathtt{supp}_\tau(a_1 \wedge \cdots \wedge a_r \wedge a_{i_1} \wedge \cdots \wedge a_{i_k}) \leq \mathtt{supp}_\tau(a_1 \wedge \cdots \wedge a_r \wedge \bar{a}_{r+1} \wedge \cdots \wedge \bar{a}_m) \leq \sum_{k=0}^{2t}(-1)^k \sum_{r<i_1<\ldots<i_k\leq m} \mathtt{supp}_\tau(a_1 \wedge \cdots \wedge a_r \wedge a_{i_1} \wedge \cdots \wedge a_{i_k})$$

Proof. This statement follows immediately from Theorem 5.2. □

Below we present results which form the basis of our algorithm for approximative computations of supports of itemsets. The binomial symbol $\binom{n}{k}$ will allow negative values of n, in which case its value is defined by the usual formula

$$\binom{n}{k} = \frac{n(n-1)\cdots(n-k+1)}{k!}.$$

Lemma 5.4 *For $m, k, h, s \in \mathbb{N}$ such that $m > s$ we have:*

$$\sum_{k=0}^{s}(-1)^{s-k}\binom{m-k-1}{s-k}\binom{h}{k} = \binom{h-m+s}{s}.$$

Proof. In [17], p. 169, it is shown that for every $a, b, c, d \in \mathbb{N}$ we have:

$$\sum_{k=0}^{a}(-1)^k\binom{a-k}{b}\binom{c}{k-d} = (-1)^{a+b}\binom{c-b-1}{a-b-d}. \tag{7}$$

By using the complimentary combinations in Equality 7 we can write:

$$\sum_{k=0}^{s}(-1)^{s-k}\binom{m-k-1}{s-k}\binom{h}{k}=\sum_{k=0}^{s}(-1)^{s+k}\binom{m-k-1}{m-s-1}\binom{h}{k}=$$
$$(-1)^s\cdot\sum_{k=0}^{s}(-1)^k\binom{m-k-1}{m-s-1}\binom{h}{k}=(-1)^s\cdot(-1)^{2m-2-s}\binom{h-m+s}{s}$$
$$=\binom{h-m+s}{s}.$$

Note that the application of the formula

$$\binom{m-k-1}{s-k}=\binom{m-k-1}{m-s-1}$$

in the above chain of equalities is justified because $m-k-1\leq m-s-1\geq 0$. □

Note that if $h=m$, the previous lemma implies

$$\sum_{k=0}^{s}(-1)^{s-k}\binom{m-k-1}{s-k}\binom{m}{k}=1.$$

Our method of obtaining bounds is based on the following theorem

Theorem 5.5 *Let $\tau=(T,A,\rho)$ be a table and let $a_1,\ldots,a_m$ be attributes in A. Then, if $m>2t$ we have:*

$$\mathtt{supp}_\tau(a_1\wedge a_2\wedge\ldots\wedge a_m)\leq\sum_{k=0}^{2t}(-1)^k\binom{m-k-1}{2t-k}S_k$$

and if $m>2t+1$:

$$\mathtt{supp}_\tau(a_1\wedge a_2\wedge\ldots\wedge a_m)\geq\sum_{k=0}^{2t+1}(-1)^{k+1}\binom{m-k-1}{2t+1-k}S_k$$

where

$$S_k=\sum_{1\leq i_1<\ldots<i_k\leq m}\mathtt{supp}_\tau(a_{i_1}\wedge\ldots\wedge a_{i_k}),$$

and $S_0=1$.

Proof. We use the method of indicators previously discussed.

Let ν_m be a random variable equal to the number of events $A_1, \ldots, A_m$ that actually occur. By Lemma 5.4 we have:

$$\sum_{k=0}^{s}(-1)^{s-k}\binom{m-k-1}{s-k}\binom{\nu_m}{k} = \binom{\nu_m - m + s}{s}$$
$$= \begin{cases} 1 & \text{if } \nu_m = m \\ 0 & \text{if } \nu_m < m \text{ and } \nu_m \geq m - s \\ \binom{\nu_m - m + s}{s} & \text{if } \nu_m < m - s. \end{cases}$$

By taking expectations of the above equation we get

$$\sum_{k=0}^{s}(-1)^{s-k}\binom{m-k-1}{s-k}S_k = \mathtt{supp}_\tau(\nu_m = m)$$
$$+ \sum\left\{\binom{\nu_m(\omega) - m + s}{s}\mathtt{supp}_\tau(\omega) : \omega \in \Omega, \nu_m(\omega) < m - s\right\},$$

where Ω denotes the space of elementary events. Note that when $\nu_m < m - s$ the sign of $\binom{\nu_m - m + s}{s}$ is identical to that of $(-1)^s$. Replacing s by $2t$ or $2t+1$ yields the result. □

6 The Estimation Algorithm

The main problem in using Bonferroni-type inequalities on collections of frequent itemsets is that some of the probabilities in the S_k sums are not known. We solved this problem by estimating the missing probabilities using Theorem 5.5. In Figure we give an algorithm for computing bounds on support of an itemset based on a collection of itemsets with known supports.

Of course upper and lower bounds for itemsets are cached during computations to avoid repeated evaluations for the same itemset. The parameter r controls the maximum size of marginals (itemsets) used in the estimation.

The use of minsupp in step 5 of function U requires some comment. Including the value of minsupp in the minimum is possible only if we can determine that the estimated itemset I is not frequent. This can be done for example, if $\mathcal{F}$ contains all frequent itemsets, or when $\mathcal{F}$ contains all frequent itemsets up to a given size k, and $|I| \leq k$. If we don't know whether I is frequent or not, we have to drop minsupp from the minimum.

Algorithm 1
Input: Itemset I, natural number r, collection $\mathcal{F}$ of itemsets, and their supports
Output: Bounds $L(I), U(I)$ on the support of I
The algorithm is implemented by functions L and U given below

Function $L(I, \mathcal{F}, r)$.
(1) If $I \in \mathcal{F}$
(2) return $\mathtt{supp}_\tau(I)$
(3) else
(4) return $\max_{-1 \leq 2t+1 \leq r} \sum_{k=0}^{2t+1} S^L\left((-1)^{k+1}\binom{m-k-1}{2t+1-k}, I, \mathcal{F}, k\right)$

Function $U(I, \mathcal{F}, r)$.
(1) If $I \in \mathcal{F}$
(2) return $\mathtt{supp}_\tau(I)$
(3) else
(4) $U \longrightarrow \min_{0 \leq 2t \leq r} \sum_{k=0}^{2t} S^U\left((-1)^{k}\binom{m-k-1}{2t-k}, I, \mathcal{F}, k\right)$
(5) $U \longrightarrow \min\{U, \mathsf{minsupp}, \min_{J \subset I} U(J)\}$
(6) return U

The functions S^L and S^U are defined below

Function S^L(real coefficient c, itemset $I = a_1 a_2 \ldots a_m$, $\mathcal{F}$, integer k)
(1) If $k = 0$ return c
(2) If $c \geq 0$
(3) return $c \cdot \sum_{i_1 < \ldots < i_k \leq m} L(a_{i_1} a_{i_2} \ldots a_{i_k}, \mathcal{F}, k-1)$
(4) else
(5) return $c \cdot \sum_{i_1 < \ldots < i_k \leq m} U(a_{i_1} a_{i_2} \ldots a_{i_k}, \mathcal{F}, k-1)$

Function S^U(real coefficient c, itemset $I = a_1 a_2 \ldots a_m$, $\mathcal{F}$, integer k)
(1) If $k = 0$ return c
(2) If $c \geq 0$
(3) return $c \cdot \sum_{i_1 < \ldots < i_k \leq m} U(a_{i_1} a_{i_2} \ldots a_{i_k}, \mathcal{F}, k-1)$
(4) else
(5) return $c \cdot \sum_{i_1 < \ldots < i_k \leq m} L(a_{i_1} a_{i_2} \ldots a_{i_k}, \mathcal{F}, k-1)$

Fig. 1. Algorithm for computing bounds on support of an itemset based on supports of a collection of its subsets.

6.1 Experimental results

In this section we present experimental evaluation of the bounds. Our algorithm works best on dense datasets, which are more difficult to mine for frequent itemsets than sparse ones. However, the algorithm was tested on both dense and sparse data (artificial market basket data was used). The rest of the paper is focused on experiments performed on dense databases.

As dense databases we used the `mushroom` database from the UCI Machine Learning Archive [18], and a census data of elderly people from the University of Massachusetts at Boston Gerontology Center available at

`http://www.cs.umb.edu/~sj/datasets/census.arff.gz`.

Since both datasets involve multivalued attributes, we replaced each attribute (including binary ones) with a number of Boolean attributes, one for each possible value of the original attribute.

Before we present a detailed experimental study of the quality of bounds, we present the results of applying the bounds to a practical task. Suppose that we did not have enough time or computational resources to run the Apriori (or similar) algorithm completely, and we decided to stop the algorithm after finding frequent itemsets of size less than or equal to 2. We then use lower bounds to find frequent itemsets of size greater than 2. The experimental results for `mushroom` and `census` databases are shown in Figures 2 and 3 respectively.

The figures show, for various values of minimum support, the true number of frequent itemsets of sizes 3 and 4, the number of itemsets that we discovered to be frequent by using our bounds, and the ratio of the two numbers.

For large values of minimum support we are more likely to classify an itemset correctly than for smaller ones. The data shows that for itemsets with largest support the chances of actually being determined to be frequent without consulting the data can be as high as 80%.

We now present an experimental analysis of the bounds obtained. In what follows, by *trivial bounds* for the support of an itemset I we mean 0 for the lower bound, and for the upper bound: the minimum of the upper bounds of the supports of all proper subsets of I and of the minimum support. As in the example above here too we mine frequent itemsets with at most two items, and compute bounds for larger ones.

Table 1 (a) contains the results for the `census` dataset with minimum support of 1.8%.

Itemset size	Min. support	18%	25%	30%	37%	43%	49%	55%	61%	73%
3	Frequent	1761	893	498	308	152	70	45	23	13
	Est. Freq.	345	244	179	127	86	54	34	19	10
	ratio (%)	19.59%	27.32%	35.94%	41.23%	56.58%	77.14%	75.56%	82.61%	76.92%
4	Frequent	4379	1769	795	368	147	48	29	16	6
	Est. Freq.	298	202	131	85	53	31	18	10	2
	ratio (%)	6.81%	11.42%	16.48%	23.10%	36.05%	64.58%	62.07%	62.50%	33.33%

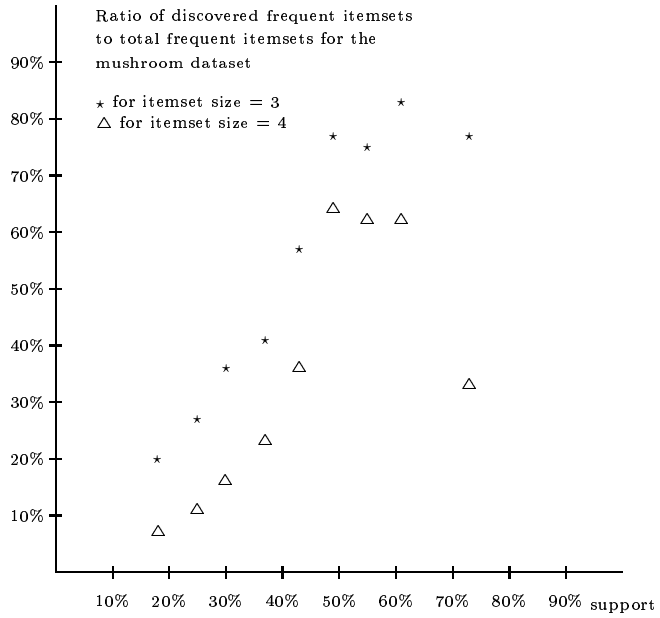

Fig. 2. Discovered vs. total frequent itemsets for the `mushroom` dataset

Itemset size	Min. support	1%	2%	3%	5%	10%	15%	30%	50%
3	Frequent	1701	1377	1145	879	503	312	112	40
	Est. Freq.	154	149	146	137	108	90	47	21
	ratio (%)	9.05%	10.82%	12.75%	15.59%	21.47%	28.85%	41.96%	52.50%
4	Frequent	5050	3560	2728	1901	852	485	105	20
	Est. Freq.	103	98	94	85	64	48	18	3
	ratio (%)	2.04%	2.75%	3.45%	4.47%	7.51%	9.90%	17.14%	15.00%

Fig. 3. Ratios of discovered to total frequent itemsets for the `census` data

The parameter r in Algorithm 1 was chosen for each itemset I to be $|I| - 1$ for maximum accuracy. This causes an increase in estimation time for larger itemsets. Later in the section we present results showing that limiting the value of r can give very fast estimates with a very small impact on the quality of the bounds. All experiments were run on a 100MHz Pentium machine with 64MB of memory.

The bounds obtained are fairly accurate. The width of the interval between the lower and upper bounds varied from 0.048 to 0.019 for itemsets of size 3. Note that the estimates become more and more accurate for larger itemsets. The reason is that the bulk of large itemsets will have subsets whose support is very small, thus giving better average trivial bounds. Nontrivial upper bounds

itemset size	3	4	5	6
average interval width	0.0482797	0.0313103	0.0228579	0.0196316
average upper bound	0.0568679	0.0319395	0.0228771	0.0196316
average lower bound	0.00858817	0.000629199	1.925e-05	0
itemsets with nontrivial bounds	7.04%	0.59%	0.04%	0.00%
itemsets with nontrivial lower	4.06%	0.39%	0.02%	–
average lower improvement	0.211321	0.161151	0.0962518	–
itemsets with nontrivial upper	6.43%	0.47%	0.03%	–
average upper improvement	0.0225656	0.00983444	0.00262454	–
time [ms/itemset]	0.2	0.3	1	7

(a) 1.8% minimum support, all itemsets

itemset size	3	4	5	6
average interval width	0.102848	0.105024	0.106997	0.110767
average upper bound	0.127438	0.109572	0.107491	0.110767
average lower bound	0.0245896	0.00454846	0.00049354	0
itemsets with nontrivial bounds	20.17%	4.25%	0.58%	0.02%
itemsets with nontrivial lower	11.64%	2.82%	0.46%	–
average lower improvement	0.211321	0.161151	0.106164	–
itemsets with nontrivial upper	18.41%	3.43%	0.40%	0.02%
average upper improvement	0.0225656	0.00983444	0.00333985	0.00338427

(b) 1.8% minimum support, frequent itemsets only

itemset size	3	4	5	6
average interval width	0.171608	0.205194	0.222602	0.231362
average upper bound	0.235004	0.223174	0.225491	0.231362
average lower bound	0.0633963	0.0179804	0.00288882	0
itemsets with nontrivial bounds	48.55%	16.79%	3.40%	0.14%
itemsets with nontrivial lower	30.00%	11.16%	2.72%	–
average lower improvement	0.211321	0.161151	0.106164	–
itemsets with nontrivial upper	44.00%	13.56%	2.33%	0.14%
average upper improvement	0.0238776	0.00983444	0.00333985	0.00338427

(c) 9% minimum support, frequent itemsets only

Table 1
Results for the `census` dataset

occur slightly more frequently than nontrivial lower bounds; however, lower bounds give on average much better improvement over the trivial bounds (this is due to the fact that our trivial upper bounds are quite sophisticated, while the trivial lower bound is just assumed to be 0).

The percentage of itemsets having nontrivial bounds is quite small. However those itemsets who have high support (and thus are the most interesting) are more likely to get interesting nontrivial bounds. This can be seen in Tables 1(b) and 1(c), where up to 48% of itemsets have nontrivial bounds proving the usefulness of Theorem 5.5. Note that in this case the interval width increases with the size of the itemsets. This is due to the fact that for high supports we don't have large number of itemsets with low supports that would create trivial upper bounds.

The conclusions were analogous for the `mushroom` database.

Table 2 shows how the choice of the argument r in Algorithm 1 influences the computation speed and the quality of the bounds. The results when r is set to the highest possible value (size of the estimated itemset minus one) is given in Table 1(a).

Census Data with 1.8% Minimum Support

$r = 2$				
itemset size	3	4	5	6
average interval width	0.0482797	0.0315442	0.022993	0.0196671
average upper bound	0.0568679	0.0321734	0.0230122	0.0196671
average lower bound	0.00858817	0.000629199	1.925e-05	0
itemsets with nontrivial bounds	7%	1%	0.10%	0%
time [ms/itemset]	0.18	0.24	0.34	0.46
$r = 3$				
itemset size	3	4	5	6
average interval width	0.0482797	0.0313103	0.0228666	0.0196328
average upper bound	0.0568679	0.0319395	0.0228859	0.0196328
average lower bound	0.00858817	0.000629199	1.925e-05	0
itemsets with nontrivial bounds	7%	0.50%	0%	0%
time [ms/itemset]	0.18	0.3	0.53	0.92

Table 2
Influence of the order of inequalities on the bounds

Census Data with 1.8% Minimum Support

itemset size	3	4	5	6	7
avg interval width	0.040498	0.081989	0.0668155	0.0392651	0.0180174
average upper bound	0.171319	0.120666	0.0685168	0.0392925	0.0180174
average lower bound	0.130821	0.0386768	0.00170127	2.73405e-05	0
time [ms/itemset]	0.24	0.46	0.96	2.54	5.12

Table 3
Estimates for itemsets with negations

The results show that limiting the value of r to 2 or 3 gives a large speedup at a negligible decrease in accuracy. This is the approach we recommend. Also note that the proportion of itemsets with nontrivial bounds is higher for lower values of r. The same experiments repeated for frequent itemsets only yielded analogous results, so we omitted the data here.

Our last experimental result concerns estimating support of conjunctions allowing negated items using Corollary 5.3. Table 3 shows the results for the `census` ataset, with supports of all frequent 1- and 2-itemsets known (1.8% minimum support). In each of the itemsets exactly two of the items were negated. Again the inequalities gave fairly tight bounds.

7 Conclusions and Open Problems

We studied properties of measures defined on free Boolean algebras arising naturally in the evaluation of sizes of queries applied to binary tables in relational databases. A method of obtaining bounds for support of database queries based on supports of frequent itemsets discovered by a datamining algorithm was presented by generalizing the Bonferroni inequalities. Specialized bounds for estimating support of itemsets, itemsets with negated items, as well as bounds for arbitrary queries have been obtained. An experimental evaluation of the bounds shows that the bounds are capable of providing useful approximations.

Various other specialized Bonferroni inequalities for other types of queries could be considered. General inequalities in Theorem 3.1 can be used for this but the bounds they give are not always tight. It has also been shown in [19] that for certain queries it is not possible to obtain tight bounds at all. Nevertheless, we believe that it is possible to obtain useful bounds for a large family of practically useful queries.

Another important direction of future research is the investigation of various other types of inequalities (e.g., sharp Bonferroni inequalities) to improve the tightness of currently available bounds. An interesting challenge would be applying other (besides the method of indicators) methods of prooving inequalities presented in [5] like the method of polynomials or the geometric method.

References

[1] H. Mannila, H. Toivonen, Multiple uses of frequent sets and condensed representations, in: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD'96), Portland, Oregon, 1996, pp. 189–194.

[2] H. Mannila, Combining discrete algorithms and probabilistic approaches in data mining, in: L. DeRaedt, A. Siebes (Eds.), Principles of Data Mining and Knowledge Discovery, Vol. 2168 of Lecture Notes in Artificial Intelligence, Springer-Verlag, Berlin, 2001, p. 493.

[3] D. Pavlov, H. Mannila, P. Smyth, Beyond independence: Probabilistic models for query approximation on binary transaction data, ICS TR-01-09, University of California, Irvine (2001).

[4] C. T. Yu, W. Meng, Principles of Database Query Processing for Advanced Applications, Morgan Kaufmann, San Francisco, 1998.

[5] J. Galambos, I. Simonelli, Bonferroni-type Inequalities with Applications, Springer, 1996.

[6] A. Ragel, B. Crémilleux, Treatment of missing values for association rules, in: X. Wu, R. Kotagiri, K. B. Korb (Eds.), Proceedings of the 2nd Pacific-Asia Conference on Research and Development in Knowledge Discovery and Data Mining (PAKDD-98), Vol. 1394 of LNAI, Springer, Berlin, 1998, pp. 258–270.

[7] J. R. Nayak, D. J. Cook, Approximate association rule mining, in: Proceedings of the Florida Artificial Intelligence Research Symposium, 2001.

[8] S. Rudeanu, Boolean Functions and Equations, North-Holland, Amsterdam, 1974.

[9] P. L. Hammer, S. Rudeanu, Pseudo-Boolean Methods for Bivalent Programming, Vol. 23 of Lecture Notes in Mathematics, Springer-Verlag, Cambridge, 1966.

[10] J. T. Lesser, W. E. Kalsbeek, Nonsampling errors in surveys, Wiley, New York, 1992.

[11] N. Linial, N. Nisan, Approximate inclusion-exclusion, in: ACM Symposium on Theory of Computing, 1990, pp. 260–270.

[12] J. Kahn, N. Linial, A. Samorodnitsky, Inclusion-exclusion: Exact and approximate, Combinatorica 16 (1996) 465–477.

[13] A. Dobra, Computing sharp integer bounds for entries in contingency tables given a set of fixed marginals, Tech. rep., Department of Statistics, Carnegie Mellon University (2001).

[14] L. Buzzigoli, A. Giusti, An algorithm to calculate the lower and upper bounds of the elements of an array given its marginals, in: Statistical Data Protection (SDP'98), Eurostat, Luxembourg, 1999, pp. 131–147.

[15] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A. I. Verkamo, Fast discovery of association rules, in: U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (Eds.), Advances in Knowledge Discovery and Data Mining, AAAI Press, Menlo Park, 1996, pp. 307–328.

[16] A. Rényi, Quelques remarques ser les probabilites des evenements dependants, Journal de Mathematique 37 (1958) 393–398.

[17] R. L. Graham, D. E. Knuth, O. Patashnik, Concrete Mathematics, Addison-Wesley, Reading, Massachusetts, 1989.

[18] C. L. Blake, C. J. Merz, UCI Repository of machine learning databases, University of California, Irvine, Dept. of Information and Computer Sciences, 1998.

[19] S. Jaroszewicz, D. Simovici, I. Rosenberg, An inclusion-exclusion result for boolean polynomials and its applications in data mining, in: Proceedings of the Discrete Mathematics in Data Mining Workshop, SIAM Datamining Conference, Washington, D.C., 2002.