

Using Constraints to Generate and Explore Higher Order Discriminative Patterns

Michael Steinbach, Haoyu Yu, Gang Fang, and Vipin Kumar

University of Minnesota,
Minneapolis, MN, USA
{steinbach, gangfang, kumar}@cs.umn.edu
haoyu@msi.umn.edu
<http://www.cs.umn.edu/~kumar>

Abstract. Discriminative pattern mining looks for association patterns that occur more frequently in one class than another and has important applications in many areas including finding biomarkers in biomedical data. However, finding such patterns is challenging because higher order combinations of variables may show high discrimination even when single variables or lower-order combinations show little or no discrimination. Thus, generating such patterns is important for evaluating discriminative pattern mining algorithms and better understanding the nature of discriminative patterns. To that end, we describe how such patterns can be defined using mathematical constraints which are then solved with widely available software that generates solutions for the resulting optimization problem. We present a basic formulation of the problem obtained from a straightforward translation of the desired pattern characteristics into mathematical constraints, and then show how the pattern generation problem can be reformulated in terms of the selection of rows from a truth table. This formulation is more efficient and provides deeper insight into the process of creating higher order patterns. It also makes it easy to define patterns other than just those based on the conjunctive logic used by traditional association and discriminant pattern analysis.

Keywords: association analysis, frequent patterns, discriminative pattern mining, linear programming, synthetic data generation

1 Introduction

Given a transaction data set where the transactions have class labels, discriminative pattern mining looks for association patterns that occur more frequently in one of the classes than the others. Although such patterns may not cover all the transactions in a data set, they can provide useful classification rules if the discrimination provided by the patterns is accurate enough for the application under consideration. One potentially useful area for discriminative patterns is medical studies that involve a set of subjects that are divided into cases (those with a specific disease) and controls (those without the disease). In such situations, any discriminative patterns that are discovered can help identify important

* This work was supported by NSF grant IIS-0916439. Computing resources were provided by the Minnesota Supercomputing Institute.

Table 1. Values of three binary variables and their combinations. The first 10 rows belong to one class (cases), while the last 10 belong to another (controls). The last row is the number of ones in cases minus the number of ones in controls.

	x_1	x_2	x_3	x_1x_2	x_1x_3	x_2x_3	$x_1x_2x_3$
Cases	1	1	1	1	1	1	1
	1	0	0	0	0	0	0
	1	1	1	1	1	1	1
	0	1	0	0	0	0	0
	0	1	0	0	0	0	0
	0	0	1	0	0	0	0
	0	0	1	0	0	0	0
	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
	1	0	0	0	0	0	0
Controls	1	0	1	0	1	0	0
	0	1	1	0	0	1	0
	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
	0	0	0	0	0	0	0
	1	1	0	1	0	0	0
	1	0	1	0	1	0	0
	0	1	1	0	0	1	0
	1	1	0	1	0	0	0
	0	0	0	0	0	0	0
ones in cases - ones in controls	0	0	0	0	0	0	2

factors in the disease and/or its development. The discovery of such *biomarkers* is often of great benefit.

However, finding such patterns is challenging because higher order combinations of individual variables may show high discrimination even when single variables or lower-order combinations only show little or no discrimination. To illustrate, Table 1 shows the values of three binary variables (x_1, x_2, x_3) for 10 cases and controls, along with their combinations: x_1x_2 , x_1x_3 , x_2x_3 , and $x_1x_2x_3$, which are the logical *and* of the indicated variables. The last row in the table shows the difference in the occurrence of ones between cases and controls. All single variables, and the pairs they form, show no discrimination between cases and controls, but the combination of all three variables does. If ones indicate the presence of a particular genetic characteristic, then this may indicate, for example, that the presence of all three genetic factors is necessary for the development of a disease.

It is easy to construct examples of such patterns involving only two variables, e.g., let both variables have 5 ones in both cases and controls, but let the ones in the two variables overlap completely in cases and not at all in controls. However, it becomes more challenging for patterns of size 3 and higher. There may even be some question about the existence of such patterns. Examples of higher order patterns that have better discrimination than any lower order pattern composed of the same variables can indeed be found for sets of variables beyond 3, although as later discussion will show, the difference between the discriminative power of a combination of binary variables and its subpatterns decreases as the number of variables increases.¹ To address such fundamental questions concerning the

¹ As measured in terms of the *DiffSup* measure used in this paper, but perhaps not in terms of other measures, such as statistical significance.

nature of discriminative patterns and provide patterns for testing the performance of current discriminative pattern finding algorithms, this paper proposes a constraint-based approach to specifying and generating such higher order patterns. In the rest of the paper, *higher order pattern* will refer *only* to those higher order patterns having more discriminative power than any of their subsets.

Although there are synthetic data generators for ordinary association patterns, such as the IBM Quest Market-Basket Synthetic Data Generator described in [1], we know of no such generator for the higher order discriminative patterns we have been discussing. However, during the course of our research into algorithms on discriminative pattern mining, we encountered the need for such a capability. As a result, we developed the constraint based approach described in this paper. More specifically, we realized that such patterns can be defined using mathematical constraints which are then solved with widely available software that uses linear programming techniques (or extensions of it) to find solutions to resulting optimization problems. Despite the challenges in creating the proper models and limitations on the size of patterns it is feasible to produce, the desired example patterns can usually be generated (often in a relatively short time, i.e., usually seconds or minutes) for patterns up to size 9 or 10.²

The insights that can be gained into the nature of higher order patterns are, however, perhaps even more important than generating higher order discriminative patterns for testing discriminative pattern finding algorithms. For instance, through experimental runs and theoretical analysis, we explore the maximum discriminative power that can be expected of a higher order pattern for different numbers of variables. In addition, this work has resulted in a formulation of the pattern generation problem in terms of the selection of rows from a truth table. This formulation is more efficient and provides deeper insight into the process of creating higher order patterns than the formulation obtained from a straightforward translation of the desired pattern characteristics into mathematical constraints. It also makes it easy to define patterns other than just those based on the conjunctive logic used by traditional association and discriminant pattern analysis. For example, it is possible to define a pattern that is present if j out of the k variables have a value of 1. It is also possible to impose additional constraints on the patterns to further tailor them to specific needs.

Overview: In Section 2 we begin with a more formal definition of discriminative patterns and of the *DiffSup* measure that is used to evaluate the discriminative power of such patterns. Section 3 presents the basic approach to pattern specification and generation, while Section 4 describes a more powerful approach that is more efficient and more general. Experimental results are presented in Section 5, and more formally analyzed in Section 6. Section 7 summarizes the paper and the areas for future work.

2 Discriminative Patterns

This section provides a formal description of discriminative patterns and the *DiffSup* measure that is often used to measure the discriminative power of a

² By the time pattern sizes of 9 or 10 are reached, computational difficulties arise in some cases.

binary variable or a set of binary variables. This is followed by a brief overview of previous work in discriminative pattern mining. As mentioned, we are not aware of previous work in generating synthetic discriminative patterns. (Again, we are only considering those patterns whose discriminative power is greater than that of any subpattern.)

2.1 Definitions

Let D be a binary transaction data set consisting of m transactions, each of which is a subset of n possible items. D can be represented as a binary data matrix, where each item is a binary variable (column of the matrix), each transaction is a row, and the i_j^{th} entry is 1 if transaction i has item j . For instance, the transactions (rows) could be subjects in a medical study, while the binary variables (columns) represent the presence or absence of various genetic features in a subject. In many cases the transactions are divided into classes, e.g., cases (those subjects with a condition) and controls (those without). Without loss of generality, we only consider discriminative patterns for the binary class problem. An extension to multiple classes is described in [2].

Assume there are $m = m_A + m_B$ transactions, where m_A is the number of cases and m_B is the number of controls. Instead of viewing an itemset, X , as a set of items, we will find it more convenient to use an equivalent representation in which the itemset is represented as a logical conjunction of Boolean variables, i.e., as $X = x_{i_1}x_{i_2}\dots x_{i_k}$, where x_{i_j} is the Boolean variable associated with the j^{th} item. Let $support\ count_A(X)$ and $support\ count_B(X)$ be the number of transactions for which X is true in classes A and B , respectively. Then, the relative supports of X in classes A and B are defined as $RelSup_A(X) = \frac{support\ count_A(X)}{m_A}$ and $RelSup_B(X) = \frac{support\ count_B(X)}{m_B}$, respectively.

$DiffSup$, which was originally defined in [2] is the absolute difference of the relative supports of X in classes A and B .

$$DiffSup(X) = |RelSup_A(X) - RelSup_B(X)| \quad (1)$$

An itemset, X , is r -discriminative if $DiffSup(X) \geq r$. The goal of discriminative pattern mining is to discover all patterns in a data set with $DiffSup$ greater than or equal to r . However, higher order patterns are not useful or interesting unless their discriminating power is greater than the discriminating power of their subpatterns.

Other measures of ‘goodness’ for discriminative patterns are sometimes used. For instance, instead of taking the difference of relative support, an alternative measure of discriminative power, the *Growth Rate*, can be defined as the ratio of the two supports [4]. Other variations include information gain [3], the χ^2 -test [2], the Gini index [13], the odds ratio [13], and various other measures [12]. These discriminative measures are generally not anti-monotonic as shown by [4, 2, 3], a fact that poses significant challenges both for pattern finding and generation. It is possible that our approach could be used to generate higher order patterns for some of these other measures, but we have not yet explored that possibility.

2.2 Previous Work in Discriminative Pattern Mining

Discriminative patterns have been investigated by a number of researchers, sometimes under other names. Dong and Li [4] define *emerging patterns (EP)* as itemsets with a large growth rate (support ratio) between two classes. Emerging patterns have been extended to several special cases such as jumping emerging patterns [9] and minimal emerging patterns [11, 10]. In [2], contrast sets (CSETs), were proposed as another possible formulation of discriminative patterns and an algorithm to mine them, CSET, was presented. In [8], contrast set mining was shown to be a special case of rule learning, where the contrast set is the antecedent of a rule whose consequent is a group. As mentioned above, various statistical discriminative measures have also been studied for discriminative pattern mining. There has also been recent work on the efficient discovery of low support discriminative patterns from dense and high-dimensional data sets [6]. Another approach [5] builds a decision tree with frequent patterns at each decision node that partition the data set into successively purer subsets.

3 Defining Higher Order Patterns with Constraints

The starting situation is a set of k binary variables which take on some assignment of zeros and ones (truth values) for a set of m_A cases and m_B controls in a set of $m = m_A + m_B$ subjects. The goal is to find one or more assignments of values to the variables that maximizes the *DiffSup* of the combination of all variables, while ensuring that the *DiffSup* of lower order combinations is less than a specified limit. A more formal definition is provided below.

Problem Statement: *Find an assignment of truth values to m instances (m_A in cases, m_B in controls) of the variables, x_1, x_2, \dots, x_k , that satisfies the following objective and constraints:*

$$\text{maximize } \text{DiffSup}(x_1 x_2 \dots x_k)$$

$$\text{subject to } \text{DiffSup}(x_{i_1} x_{i_2} \dots x_{i_j}) \leq \text{limit}, \text{ where } 1 \leq j < k$$

□

This statement of the problem combines constraint satisfaction with optimization. Generating solutions for just the constraint portion of the problem would typically yield size k patterns whose *DiffSup* is less than that of their subpatterns and thus not interesting.

This type of constrained optimization problem, although simply stated, needs to be translated into a practical optimization model. For this purpose, we chose AMPL (A Modeling Language for Programming) [7]. “AMPL is a computer language for describing production, distribution, blending, scheduling and many other kinds of problems known generally as large-scale optimization or mathematical programming” and has a long history of use in the optimization community. Although we used a commercial implementation of AMPL, along with the IBM ILOG CPLEX Optimizer as a solver, there are non-commercial versions of AMPL (or AMPL subsets) available. For instance, GLPK (GNU Linear Programming Kit), may also provide a suitable platform for solving the problem posed above. However, we have not tried our examples on GLPK.

A key problem in the translation, and indeed, in the solution of this problem is the fact that the satisfaction of the constraints and maximization of the objective depend on the combinations of all variables. This poses at least two major challenges. First, the number of the combinations is $2^k - 1$, where k is the number of individual variables. Thus, the number of constraints grows exponentially with k . However, for pattern sizes of 10 or less (1023 combinations or less) we found this manageable.

The second challenge arises because the values of the combinations are, of course, functions of the variable values being sought. This functional relationship needs to be specified either implicitly or explicitly. When an implicit approach is used, the AMPL model file requires (1) defining a variable for each combination, 7 in all, and (2) defining constraints for each of the combinations. This approach generates a higher order pattern with the desired properties, but the model file rapidly increases in size. Although this file could be generated automatically, we also discovered that the solver was having difficulty producing a solution once we got to five variables. Thus, although this representation provides a relatively direct translation of the problem statement given above, it is not as compact or efficient as the approach described in the next section.

4 A More General Approach

Given the limitations of the previous solution, we sought a new formulation that would provide a more flexible and efficient approach to specifying and generating higher order patterns, as well as providing deeper insight into the nature of higher order patterns and the ability to specify patterns other than those involving logical *and*.

The approach is the following. For k variables, define the truth table that gives the possible values of the variables and all the combinations of variables up to size k . For now, the combinations are assumed to be logical *and*. For example, a truth table for 3 variables and its combinations is shown in Table 2. The same constraint problem is solved, but the constraint solution is transformed into one of selecting, m_A rows for cases and m_B rows for controls, from the 2^k rows of the truth table, so that the constraints given in the problem statement above are satisfied. In this approach, the relationships between the combinations of variables and individual variables are defined explicitly and do not need to be specified as constraints. As a result, the solver has far fewer constraints to handle and runs (in our experience) far more quickly.

The model file is also much simpler and does not need to be changed as the number of variables changes. Figure 1 shows the new AMPL model file. This file is much smaller than the model file for the original approach (which is not shown in order to save space) and is not specific to any number of variables. This approach is also more flexible than the original one in several ways. First, variable combinations can be defined to be something other than logical *and* without changing the model file. For instance, we could define a combination of size k to be 1 if at least $k - 1$ of the variables are 1. This is not to say that any truth table that can be created will be meaningful or even have a feasible solution. Second, we could choose to only use some of the rows of the truth table.

Table 2. Truth table for three binary variables and their combinations.

row	x_1	x_2	x_3	x_1x_2	x_1x_3	x_2x_3	$x_1x_2x_3$
1	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0
3	0	1	0	0	0	0	0
4	0	1	1	0	0	1	0
5	1	0	0	0	0	0	0
6	1	0	1	0	1	0	0
7	1	1	0	1	0	0	0
8	1	1	1	1	1	1	1

Again, this may not yield a feasible solution. However, for example, it is possible to omit the 0 row (i.e., the assignment of zeros to all variables) and still obtain solutions, sometimes with a similar value of the objective function.³

Admittedly, this approach also has a number of limitations. The table grows in size exponentially as the number of variables increases. The number of constraints that must be solved by the solver, although fewer than in the previous approach where combinations were defined via constraints, is still exponential in k as well, specifically $2^k - 1$. (There is no escaping that unless the problem statement is changed.) In practice, we found that for a larger number of variables (9 or 10), larger numbers of cases and controls (200 or more), and smaller limits on *DiffSup* (0.3 or less) the solver either ran out of memory (we were using 32 bit AMPL and CPLEX) or took so long to run that we stopped the job. Nonetheless, most of the other cases ran in very little time, i.e., just a few seconds or minutes.

For both approaches, it is possible to add additional constraints. For instance, we used this algorithm plus a few additional constraints to generate the specialized discriminative patterns found by the algorithm described in [6]. This algorithm finds only a subset of discriminative patterns, but can find patterns missed by other discriminative mining algorithms. By specifying and generating these patterns, we gained a better understanding of the types of patterns this algorithm discovers or misses.

5 Experimental Results

This section presents experimental results that show how the optimal *DiffSup* value that is found varies with the *DiffSup* limit for the subpatterns, the number of variables, and the size of the data set, i.e., number of cases and controls. To show this for both balanced and unbalanced data sets, we used six cases. For both unbalanced and balanced data sets, m_A took the values 10, 25, 50, 100, 200, and 250. For unbalanced data sets m_B was double, while for balanced data sets it was, of course, the same. We summarize the results and present a few plots that support our summary.

First, the optimal *DiffSup* value does not vary much with the size of the data set for a fixed limit and number of variables. Intuitively, once the data set is large enough to achieve a certain pattern, this pattern can be repeated multiple times

³ However, the row of all zeros does play an important role in creating a data set with the optimal value and often shows up in our solutions.

```

param limit >= 0.0 ; param mA > 0 ; param mB > 0 ;

param nv0 >= 2 ; param nv = (2^nv0) ; param nvml = nv - 1 ;

set CLASSA = 1 .. mA by 1 ;
set CLASSB = (mA+1) .. (mA+mB) by 1 ;

set SINGLES = 1 .. nv0 by 1 ;

set SUBJECT = 1 .. (mA+mB) by 1 ;
set COMBINATION = 1 .. (nv-1) by 1 ;
set COMBINATIONm0 = 1 .. (nv-2) by 1 ;

set VALUES = 1 .. nv by 1 ;
set LINKS = { i in SUBJECT, j in VALUES } ;

param L { i in VALUES, j in COMBINATION } ;
var Select { ( i , j ) in LINKS } binary ;

maximize diffX0: (1/mA) * sum { i in CLASSA } ( sum { j in VALUES } Select[ i , j ] * L[ j , nvml ] ) -
(1/mB) * sum { i in CLASSB } ( sum { j in VALUES } Select[ i , j ] * L[ j , nvml ] ) ;

subject to dij { k in COMBINATIONm0 } :
-limit <= (1/mA) * ( sum { i in CLASSA } ( sum { j in VALUES } Select[ i , j ] * L[ j , k ] ) ) -
(1/mB) * ( sum { i in CLASSB } ( sum { j in VALUES } Select[ i , j ] * L[ j , k ] ) ) <= limit ;

subject to oneperSUB { i in SUBJECT } : sum{ (i,j) in LINKS } Select[ i , j ] = 1 ;

```

Fig. 1. AMPL model file for the truth table approach. The truth table implicitly specifies the relationship between a combination of variables and the individual variables for larger data sets. This is true for both balanced and unbalanced data sets. This is illustrated in Figure 2 for a balanced data set with 5 variables which shows the optimal *DiffSup* values across different *DiffSup* subpattern constraint levels and numbers of cases. Because of this lack of variation, we illustrate the rest of the points we want to make by using data sets with $m_A = 50$.

A minor point is that the optimal *DiffSup* value sometimes decreases slightly as m_A increases from 10 to 25, e.g., for limits 0.3 and 0.5. As we will see in the next section, reducing the *DiffSup* values of subpatterns less than size k while maximizing the *DiffSup* of the size k pattern requires adding a certain number of records in cases and controls to ‘balance out’ the occurrence of the subpatterns of size less than k . Intuitively, this is more difficult when the data set size is small. Our conjecture is that for 5 variables data sets of size 10 and 25 are both too small to perform this balancing process as well as in larger data sets, but that the balancing works more efficiently for a data set of size 10 than for a data set of size 25. However, a formal analysis is needed to confirm this.

The second major observation is that for a given limit and data set size, the optimal *DiffSup* value decreases as the number of variables increases. This is not surprising, since intuitively, there are far more constraints to be satisfied as the number of variables increases. (Recall the number of constraints is $2^k - 1$.) This is shown for both the balanced and the unbalanced cases by figures 3 and 4. These figures also show that the optimum attained in an unbalanced data set is often less than that of the corresponding balanced set, at least for smaller values of the limit. Also note that for both balanced and unbalanced, the *DiffSup* optimum does not change much once 9 or 10 variables are reached. Note that

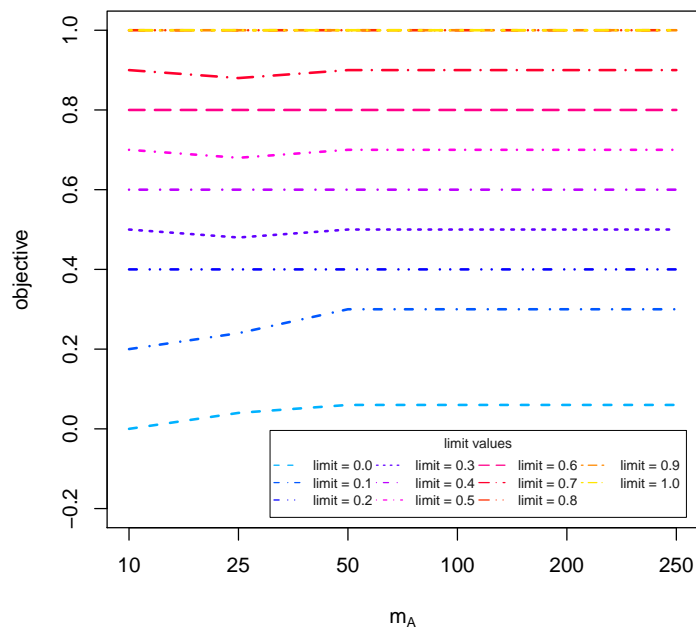


Fig. 2. Optimal *DiffSup* values for 5 variables across different limits and numbers of cases. $m_A = m_B$.

the results for 10 variables are shown as a dashed line, while the results for 9 variables are shown just as orange crosses, without any accompanying line.

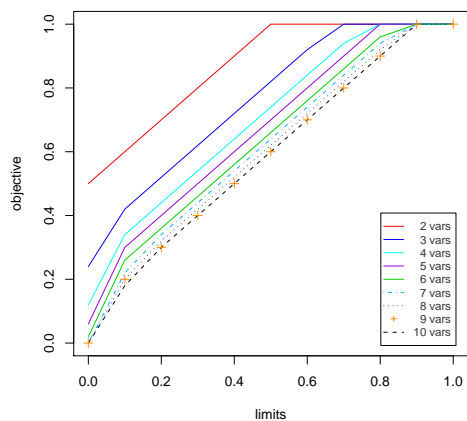


Fig. 3. Plot of optimal *DiffSup* for different limits and number of variables. $m_A = m_B = 50$.

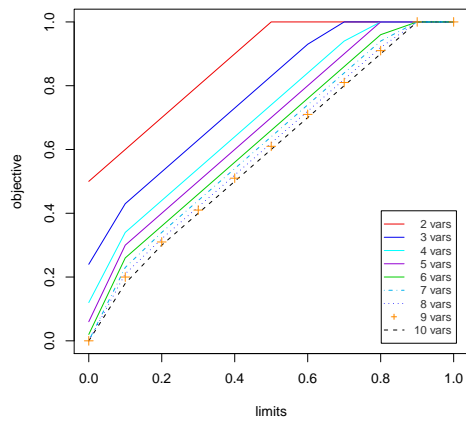


Fig. 4. Plot of optimal *DiffSup* for different limits and number of variables. $m_A = 50, m_B = 100$.

6 Formal Analysis

It is possible, at least in some cases, to perform a more formal analysis of this approach to determine the optimal *DiffSup* as the number of variables increases. We briefly sketch this approach for *limit* = 0 and balanced data sets. The approach is to fill out the m rows of the data file one by one. We begin by putting a row consisting of all ones in the cases. Every combination, including that of k variables, has the value 1. To achieve the goal of maximizing the *DiffSup* for the size k combination, while maintaining every other combination with a 0 *DiffSup*, it is necessary to add rows to cases and controls that ‘cancel out’ all other combinations.

We begin by canceling out the $\binom{k}{k-1} = k$ subpatterns of size $k - 1$. This can be done by adding, for each such subpattern, the row from the truth table that contains a 1 for that pattern. For example, if $k = 3$, then there are 3 pairs of variables. Rows 4, 6, and 7 of Table 2 are the rows in the truth table that will cancel the net occurrence for these pairs when placed in controls. This accomplishes the goal that the *DiffSup* of the pairs is 0. More generally, canceling out the k subpatterns of size $k - 1$ requires that k rows be added to controls.

However, doing this increases the occurrence of the size $k - 2$ subpatterns in controls by 2. Each $k - 1$ subpattern contains $\binom{k-1}{k-2} = k - 1$ subpatterns of size $k - 2$. Thus, when $k = 3$, pairs contain two individual subpatterns of size 1 (which is obvious). Since there are $\binom{k}{k-2} = k(k - 1)/2$ patterns of size $k - 2$, the total occurrence in controls of a $k - 2$ size pattern from the k rows just added is $k(k - 1)/(k(k - 1)/2) = 2$. More generally, adding these k rows increases the occurrence in controls of size $k - i$ patterns, $1 \leq i \leq (k - 1)$, by i . (Proof omitted.) For instance, when $k = 3$, pairs have an occurrence in controls of 1 and individual variables have an occurrence of 2.

Thus, some rows need to be added to cases to cancel out the excess count that was added to controls. This can be done by adding $\binom{k}{k-2}$ rows from the truth table, where these rows all have a value of 1 for a particular $k - 2$ size pattern, but do not have a 1 for any higher level pairs. While this reduces the *DiffSup* of all the $k - 2$ size patterns to 0, size $k - 3$ patterns now have an excess count of 1 in cases. This process continues, adding $\binom{k}{k-i}$ patterns of size $k - i$ alternately to cases and control until k rows corresponding to individual variables are added to either cases or controls.

Thus, adding one row of all ones to cases requires adding many rows to both cases and controls to achieve 0 *DiffSup* in the subpatterns. It is possible to compute how many rows are added to cases and controls and then to compute the value of *DiffSup* for the size k pattern. $m_A = \sum \binom{k}{i}$, $i = k, k - 2, \dots, 2$ or 1 depending on whether k is even or odd, respectively, and $m_B = \sum \binom{k}{i}$, $i = k - 1, k - 3, \dots, 1$ or 2 depending on whether k is even or odd, respectively. These values will differ by 1 and thus, to get the *DiffSup* of the patterns to actually cancel will require adding the row consisting of all zeros from the truth table to either cases or controls to make the data set balanced. This represents the best solution (proof omitted) for *limit* = 0 and thus constitutes an upper bound.

Table 3 shows the number of rows this process generates in cases and controls for each variable size (omitting the one zero row that must be added to balance the data set). This table also shows the maximum *DiffSup* attainable. We can make some observations (which can be more formally proven), e.g., that the $m_A = m_B = 2^{k-1}$ and $DiffSup = 1/2^{k-1}$. This agrees with Figure 3 and even appears to hold for the unbalanced data results shown in Figure 4. This result could be used as the basis for a simple algorithm that creates optimal size k patterns for $limit = 0$ and balanced data sets that are multiples of size 2^k .

Table 3. Table showing the m_A and m_B values required for 0 *DiffSup* subpatterns and maximal *DiffSup* for the size k pattern.

num vars	2	3	4	5	6	7	8	9	10
m_A	1	4	7	16	31	64	127	256	511
m_B	2	3	8	15	32	63	128	255	512
<i>DiffSup</i>	0.5	0.25	0.125	0.0625	0.0313	0.0156	0.0078	0.0039	0.0020

Although this analysis is interesting and yields some useful insights, much more analysis is possible and could well yield equally interesting results.

7 Conclusion and Future Work

We have presented a constraint based approach for generating higher order combinations of individual variables which may show high discrimination even when the single variables or lower order combinations little or no discrimination. The basic approach quickly lead to a new approach based on building a data set by choosing rows from a truth table. This formulation provides more insight into the process of creating higher order patterns than does the formulation obtained from a straightforward translation of the desired pattern characteristics into mathematical constraints, as was demonstrated by the derivation of an upper bound for the *DiffSup* of a size k pattern for a balanced data set and $limit = 0$. It also allows for the easy definition of patterns other than just those based on the traditional conjunctive logic used by traditional association and discriminant pattern analysis. Experimental results and formal analysis were presented to give insight into the behavior of higher order patterns.

There are many directions worthy of further investigation. First, there are many aspects of the results presented here that remain to be addressed. One is establishing bounds on the *DiffSup* of size k patterns for different types of data sets, different numbers of variables, and different limits. We plan to investigate alternative approaches for analyzing the problem that may provide such answers more readily than the two approaches presented in this paper. Another task is to investigate truth tables that reflect types of logic other than logical *and* or that require solutions that don't involve a row of all zeros. Finding larger sets of solutions, even if slightly suboptimal, might also be useful since the goal is to generate different types of discriminative patterns. Along similar lines, we could subject different subpatterns to different constraints. We could also

explore other types of solution methods, either other constraint based methods or non-constraint based methods. It is even conceivable that a thorough analysis could produce a non-optimization based algorithm for generating higher order discriminative patterns. It would be especially interesting to investigate if the insights that arise from our work can lead to new or improved discriminative pattern mining algorithms. Yet another significant challenge is to investigate other measures of discrimination .

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB '94: Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.
2. S. Bay and M. Pazzani. Detecting group differences: Mining contrast sets. *Data Mining and Knowledge Discovery*, 5(3):213–246, 2001.
3. H. Cheng, X. Yan, J. Han, and C.-W. Hsu. Discriminative frequent pattern analysis for effective classification. In *Proceedings of International Conference on Data Engineering*, pages 716–725, 2007.
4. G. Dong and J. Li. Efficient mining of emerging patterns: Discovering trends and differences. In *Proceedings of the 2001 ACM SIGKDD international conference on knowledge discovery in databases*, pages 43–52, 1999.
5. W. Fan, K. Zhang, H. Cheng, J. Gao, X. Yan, J. Han, P. Yu, and O. Verscheure. Direct mining of discriminative and essential frequent patterns via model-based search tree. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 230–238, New York, NY, USA, 2008. ACM.
6. G. Fang, G. Pandey, W. Wang, M. Gupta, M. Steinbach, and V. Kumar. Mining low-support discriminative patterns from dense and high-dimensional data. *IEEE Transactions on Knowledge and Data Engineering*, 2010 (In press).
7. R. Fourer, D. M. Gay, and B. W. Kernighan. A modeling language for mathematical programming. *Manage. Sci.*, 36(5):519–554, 1990.
8. D. A. N. Geoffrey I. Webb, Shane M. Butler. On detecting differences between groups. *Proceeding of the ACM SIGKDD international conference on knowledge discovery in databases*, pages 256–265, 2003.
9. J. Li, G. Dong, and K. Ramamohanarao. Making use of the most expressive jumping emerging patterns for classification. *Knowledge and Information systems*, 3(2):131–145, 2001.
10. J. Li, G. Liu, and L. Wong. Mining statistically important equivalence classes and delta-discriminative emerging patterns. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 430–439. ACM New York, NY, USA, 2007.
11. E. Loekito and J. Bailey. Fast mining of high dimensional expressive contrast patterns using zero-suppressed binary decision diagrams. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 307–316. ACM New York, NY, USA, 2006.
12. S. Morishita and J. Sese. Transversing itemset lattices with statistical metric pruning. In *Proceedings of the nineteenth ACM Symposium on Principles of database systems*, pages 226–236. ACM New York, NY, USA, 2000.
13. P.-N. Tan, M. Steinbach, and V. Kumar. Introduction to data mining. *Addison-Wesley*, 2005.