

Support Envelopes: A Technique for Exploring the Structure of Association Patterns

Michael Steinbach

Dept. of Comp. Sci. & Eng.

University of Minnesota

steinbac@cs.umn.edu

Pang-Ning Tan

Dept. of Comp. Sci. & Eng.

Michigan State University

ptan@cse.msu.edu

Vipin Kumar

Dept. of Comp. Sci. & Eng.

University of Minnesota

kumar@cs.umn.edu

August 31, 2004

Abstract

This paper introduces support envelopes—a new tool for analyzing association patterns—and illustrates some of their properties, applications, and possible extensions. Specifically, the *support envelope* for a transaction data set and a specified pair of positive integers (m, n) is the set of items and transactions that need to be searched to find any association pattern involving m or more transactions and n or more items. For any transaction data set with M transactions and N items, there is a unique lattice of at most $M * N$ support envelopes that captures the structure of the association patterns in that data set. Because support envelopes are not encumbered by a support threshold, this *support* lattice provides a complete view of the association structure of the data set, including association patterns that have low support. Furthermore, the boundary of the support lattice—the *support boundary*—has at most $\min(M, N)$ envelopes and is especially interesting since it bounds the maximum sizes of potential association patterns—not only for frequent, closed, and maximal itemsets, but also for patterns, such as error-tolerant itemsets, that are more general. The association structure can be represented graphically as a two-dimensional scatter plot of the (m, n) values associated with the support envelopes of the data set, a feature that is useful in the exploratory analysis of association patterns. Finally, the algorithm to compute support envelopes is simple and computationally efficient, and it is straightforward to parallelize the process of finding all the support envelopes.

Keywords: support envelope, association analysis, formal concept analysis, error-tolerant itemsets

1 Introduction

This paper introduces *support envelopes*, which provide a compact and computationally efficient approach for capturing the high-level structure of support based association patterns such as frequent, closed, and error-tolerant itemsets [9, 10, 14, 21, 19]. More specifically, the *support envelope* for a transaction data set and a pair of positive integers (m, n) is the set of items and transactions that need to be searched to find all patterns involving a minimum of m transactions and n items. Thus, the number of distinct envelopes is at most $M * N$, where M is the number of transactions and N is the number of items. For sparse datasets, the number of support envelopes is typically much less than $M * N$.

Like other association patterns, e.g., frequent and closed itemsets, support envelopes form a lattice. (We will call this lattice the *support lattice*.) In particular, for any binary matrix with M rows (transactions) and N columns (items), there is a unique lattice that captures the structure of the association patterns for that matrix. Standard association analysis techniques to find frequent, closed, or maximal itemsets break down when the support threshold becomes too low, and thus, cannot provide information about patterns with low levels of support. However, because support envelopes are not encumbered by a support threshold, the support lattice provides a complete view of the association structure of the data set, including association patterns that have low support.

The boundary (positive border [12]) of the support lattice—the *support boundary*—is especially interesting since it bounds the maximum sizes of potential association patterns. For example, if an envelope on the support boundary is characterized by the pair of integers (m, n) , then there are no non-empty envelopes or association patterns with larger values of m and n . To illustrate, if the support boundary of a data set contains the support envelope characterized by $(10, 5)$, then the data set cannot contain a frequent itemset with 5 or more items that has more than 10 supporting transactions. Furthermore, the support boundary is not just a bound on frequent, closed, and maximal itemsets, but also on patterns, such as error-tolerant itemsets (ETIs) [19], that are more general. Another attractive property of the support boundary is that it contains at most $\min(M, N)$ support envelopes.

In addition to providing us with a theoretical foundation for understanding—at least in part—the high-level structure of association patterns, support envelopes allow the association structure of a binary data to be represented graphically. In particular, the (m, n) values associated with the envelopes in the data set can be displayed in a two-dimensional scatter plot, which provides a global view of the structure of association patterns in a data set. Additional information can be added to this scatter plot by coloring each point with the density—fraction of ones—of its corresponding support envelope. Support envelopes are often fairly sparse, i.e., have low density, but in some cases, support envelopes can be relatively dense, and hence, are interesting not just because of the information that they provide about other association patterns, but also as patterns in their own right. Thus, for some data sets, e.g., the kosarak data set discussed in Section 4, we can use the scatter plot to identify dense envelopes that can be investigated to find association patterns of considerable interest, e.g., low support itemsets with many items.

Support envelopes are also interesting from an algorithmic point of view. We present a simple iterative algorithm that, given only (m, n) and a binary data matrix, finds the support envelope for all the support-based patterns characterized by (m, n) in time proportional to (a) nnz , the number of non-zero entries in the data matrix, and (b) the number of iterations, which is typically small, i.e., less than 10 for data sets in this paper. This algorithm can compute any individual support envelope without using the preceding elements of the support lattice. (As discussed later, using the lattice does, however, improve efficiency somewhat, i.e., the number of iterations is typically reduced to 2 or 3.) The computational complexity of computing the entire support lattice is $O(M * N * nnz)$, while the computational complexity of finding only the support boundary is $O(nnz * \min(N \log(M), M \log(N)))$. Thus, unlike many association mining tasks, the computational complexity of finding the support lattice or support boundary is not inherently exponential. Also, because support envelopes can be computed independently of the lattice, it should be straightforward to parallelize finding all envelopes or finding the envelopes on the support boundary.

Finally, we consider two simple extensions of support envelopes. First, support envelopes involving only the items of one specific transaction provide a view of patterns with respect to a particular transactions, i.e., the set of such ‘restricted’ support envelopes will involve only those association patterns that are guaranteed to contain the specified transaction. Second, we consider support envelopes that involve the additional constraint that a certain fraction of the occurrences of the item must be in the support envelope. Such an approach eliminates items that are frequent across many patterns.

Overview Section 2 introduces support envelopes and an algorithm for finding them via an informal example, while Section 3 provides a formal analysis of support envelopes their properties, and also discusses algorithms for finding an individual support envelope, the envelopes on the support boundary, or the set of all envelopes for a data set. Examples of the application of support envelopes to explore real transaction data sets are provided in Section 4, and extensions of the support envelope are considered in Section 5. Section 6 discusses the relationship of support envelopes to other work, and we conclude, in Section 7, with a summary and indications for future work.

2 Support Envelopes and Support Boundaries: An Informal Introduction

This section contains an informal introduction to support envelopes, lattices, and boundaries. Consider the following task: Construct an algorithm to identify all the items and transactions that are involved in frequent itemsets with at least n items, where by frequent we mean that the itemset must appear in m or more transactions. Instead of trying to find the interesting frequent patterns, we are asking for something simpler—for the items and transactions involved in such patterns.

We will use the data set provided in Table 1. Transactions have the numeric labels 1 through 12, while items are labelled with the letters ‘A’ through ‘E.’ This table also shows

the sums of each row and column, which represent, respectively, the number of items in each transaction and the number of transactions containing each item. For example, transaction 1 contains 4 items, while item A occurs in 6 transactions.

2.1 The Support Envelope

We start with a simple idea: Any itemset that has a support requirement of m must contain items that occur in at least m transactions. Similarly, any itemset that contains n items must be supported by transactions that have at least n items. Thus, for example, suppose we are interested in all itemsets that contain at least $n = 3$ items and have at least $m = 3$ supporting transactions. From Table 1, we see that we need consider only transactions (rows) 1, 3, 7, 8, 10, and 11, since all other transactions have fewer than 3 items. We are not saying that these transactions are definitely involved in frequent itemsets with 3 or more items, only that they might be.

Turning our attention to items, we now evaluate which items are involved in itemsets of size 3 or more with a support of 3 or more. We might first look at the number of transactions in which each item appears—see Table 1—but such an approach does not take into account our previous observation that some transactions should be eliminated. In particular, it is only the support of items within transactions 1, 3, 7, 8, 10, and 11 that is important, as these are the only transactions capable of supporting an itemset with 3 or more items. After computing support within only those transactions, we get the support for the five items to be, in order, 4, 2, 6, 6, 5—see Table 2. Thus, we can eliminate item B.

Of course, once we have eliminated item B, we have invalidated our previous counts of how many items occur in each transaction. After taking the elimination of item B into consideration, the counts for transactions 1, 3, 7, 8, 10, and 11 are, respectively, 4, 3, 4, 4, 4, and 2—see Table 3. Thus, transaction 11 can be eliminated—see Table 4. Further iterations yield no change and thus, we obtain the result that all itemsets with 3 or more items and 3 or more supporting transactions occur among some subset of items A, C, D, and E and some subset of transactions 1, 3, 7, 8, and 10. This set of items and transactions is a support envelope.

To informally verify the correctness of this result, we computed the frequent itemsets with a support threshold of 3. These itemsets are shown in Table 5 and are consistent with the results of our example, i.e., they show that the only itemsets with support of 3 or more are those involving items A, C, D, and E. It is straightforward to verify—see tables 1 and 4—that the only transactions that contain at least three of these items are 1, 3, 7, 8, and 10.

2.2 The Support Lattice and Boundary

For the data set in Table 1, all the support envelopes can readily be computed by using the algorithm that we developed above for each value of (m, n) , where $1 \leq m \leq 12$ and $1 \leq n \leq 6$. (This algorithm, which we call the Support Envelope Algorithm (SEA), is

Table 1: Original transaction data set.

	A	B	C	D	E	row sum
1	1	0	1	1	1	4
2	0	1	0	1	0	2
3	0	1	1	1	1	4
4	0	0	1	0	1	2
5	0	1	0	1	0	2
6	0	1	0	1	0	2
7	1	0	1	1	1	4
8	1	0	1	1	1	4
9	1	0	0	1	0	2
10	1	0	1	1	1	4
11	0	1	1	1	0	3
12	1	0	0	0	1	2
col sum	6	5	7	10	7	

Table 2: Data after eliminating rows with less than 3 items. Step 1.

	A	B	C	D	E	row sum
1	1	0	1	1	1	4
3	0	1	1	1	1	4
7	1	0	1	1	1	4
8	1	0	1	1	1	4
10	1	0	1	1	1	4
11	0	1	1	1	0	3
col sum	4	2	6	6	5	

Table 3: Data after eliminating columns with less than 3 supporting transactions. Step 2.

	A	C	D	E	row sum
1	1	1	1	1	4
3	0	1	1	1	3
7	1	1	1	1	4
8	1	1	1	1	4
10	1	1	1	1	4
11	0	1	1	0	2
col sum	4	6	6	5	

Table 4: Data after eliminating rows with less than 3 items. Step 3.

	A	C	D	E	row sum
1	1	1	1	1	4
3	0	1	1	1	3
7	1	1	1	1	4
8	1	1	1	1	4
10	1	1	1	1	4
col sum	4	5	5	5	

Table 5: Frequent itemsets with support 3 or more.

1 item	2 items	3 items	4 items
A B C	AC AD AE BD	ACD ACE	ACDE
D E	CD CE DE	ADE CDE	

formally described in Section 3.3.) Not all values of m and n will yield non-empty support envelopes, and some (m, n) pairs may yield the same support envelope, i.e., the same set of transactions and items. Support envelopes can be organized as a lattice [5] by defining one support envelope to be ‘less than’ (a subset of) a second support envelope if the items and transactions of the first are a subset of the items and transactions of the second.

In Figure 1, we see a tree-like display of the support lattice. Nodes represent individual support envelopes and the lines reflect the fact that the lower support envelope is a subset of the higher support envelope. Inside each box representing a support envelope are the values of m and n that characterize the envelope, as well as the sets of transactions and items involved in the envelope. Thus, for the envelope represented by the topmost node in Figure 1, each of the items in the support envelope occurs in at least 5 transactions and each transaction in the support envelope contains at least 2 items. Since these conditions are satisfied by all attributes and objects, this *root* (or *top*) envelope contains all the transactions and items in the data set. While not illustrated by Figure 1, support envelopes can have multiple parents and/or children.

Since each pair of integers (m, n) is associated with a support envelope (possibly the empty support envelope), we can represent the entire set of support envelopes as a rectangular plot of an M by N grid of points (M is the number of transactions and N is the number of items), where each point is shaded according to its associated support envelope, and where each support envelope is assigned a different shade of gray. Figure 2 shows the set of support envelopes using such a representation. Once again, we have used the same shade of gray for all points i.e., (m, n) pairs, that represent the same support envelope. Thus, the block in the upper left corner of Figure 2 represents the fact that all support envelopes with $m \leq 5$ and $n \leq 2$, are identical, i.e., contain the same transactions and items. To identify the support envelope associated with each block in the figure, we find the largest possible values of m and n of the block, i.e., we use the values of m and n from the the lower right corner of the block. Thus, for example, the big dark block in the upper left corner is the $(5, 2)$ support envelope. Also, as we will prove in the next section, a support envelope that is to the right and/or below a particular support envelope is a subset of any higher (more leftmost) support envelope, i.e., the transactions and items of the lower (more rightmost) support envelope are subsets of the transactions and items of the higher (more leftmost) support envelope.

For larger data sets, it is more convenient to view the set of support envelopes as a scatter plot of the (m, n) values. Such a plot is shown in Figure 3. To provide more information, we have colored each of the points representing a support envelope by its density, i.e., by the fraction of ones in the block defined by the transactions and items in the envelope. (This information can be gathered at almost no cost when finding the support envelopes.) Density

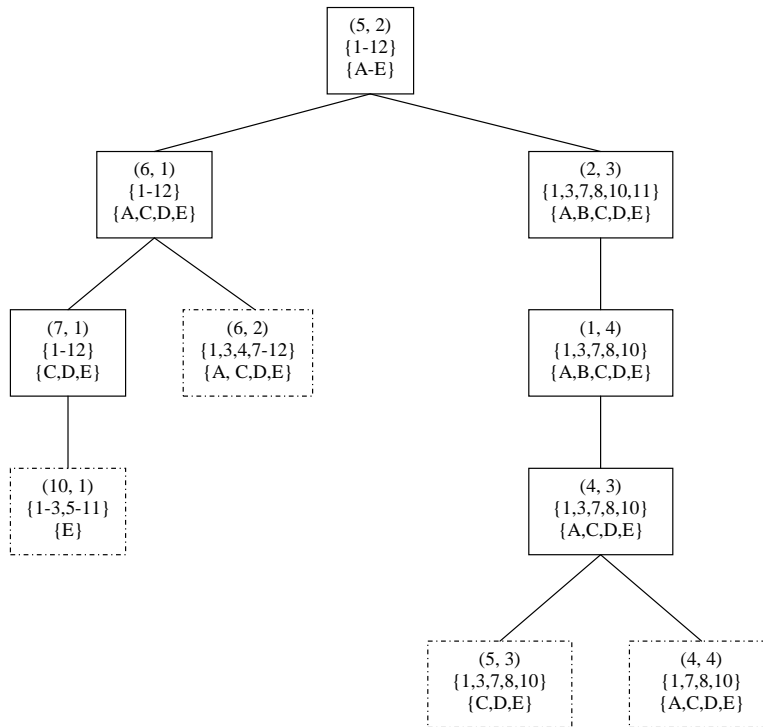


Figure 1: Support lattice for sample data.

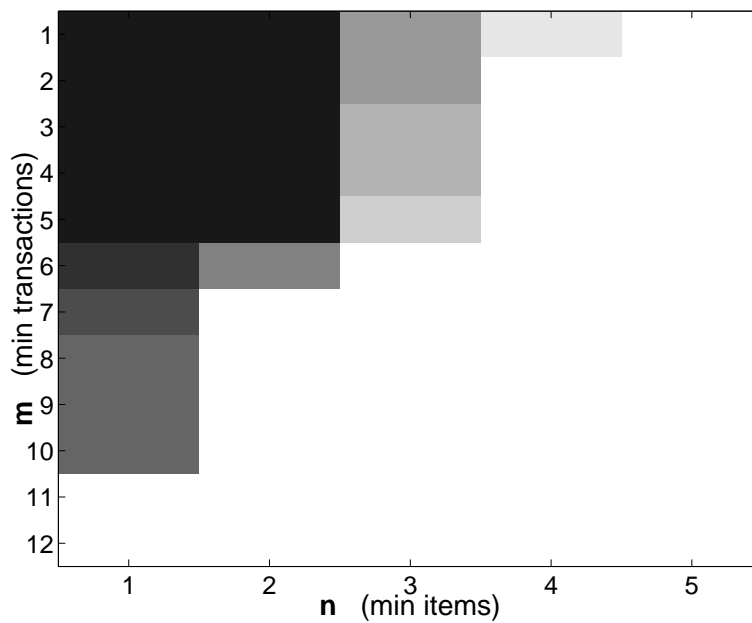


Figure 2: Support lattice for sample data - rectangular format.

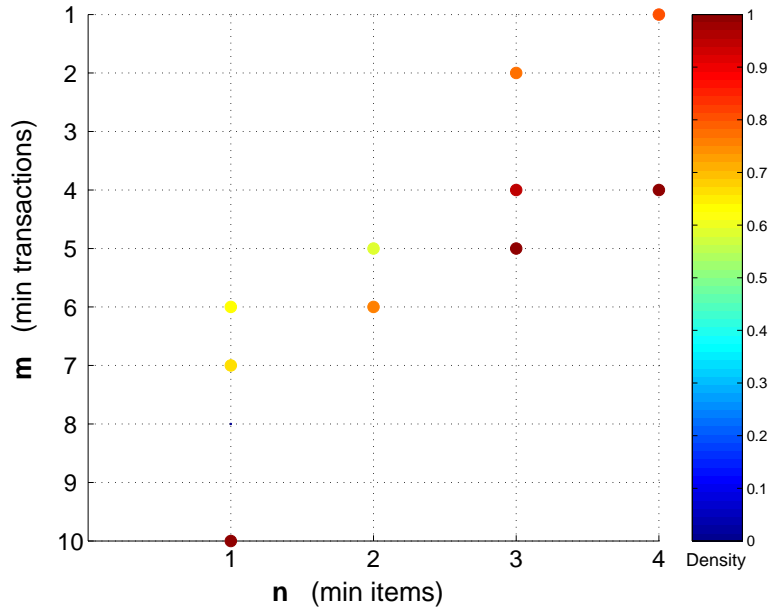


Figure 3: Support lattice for sample data - scatter plot. (Must be viewed in color.)

information is important when a support envelope is relatively dense, e.g., more than 50% dense. In such cases, the envelope is an interesting pattern in own right, not just because of the information it provides about other association patterns. We show examples of this for real transaction data sets later in this paper.

A scatter plot does not explicitly show all the information of the lattice, i.e., which envelopes are the immediate descendants or parents of other envelopes. However, most of this information is captured by the position of the points that represent envelopes. Specifically, an envelope (point) that is below and/or to the right of a second envelope (point) is contained by that envelope. In our graphical exploration of support envelopes later in the paper, we will focus on the scatter plot of all the envelopes of a data set.

3 Support Envelopes: A Formal Introduction

After a quick review of notation and some other preliminaries, we define support envelopes, analyze the algorithm for finding support envelopes, and prove some properties of support envelopes.

3.1 Preliminaries

In this section we take care of a few preliminaries.

3.1.1 Notation

An overview of the notation used in this paper is provided in Table 6.

Table 6: Summary of Notation

Notation	Description
\mathcal{D}	Data matrix of M rows and N columns
$\mathcal{T} = \{t_1, t_2, \dots, t_M\}$	Set of objects (transactions, rows) of \mathcal{D}
$\mathcal{I} = \{i_1, i_2, \dots, i_N\}$	Set of attributes (items, variables, columns) of \mathcal{D}
X, Y	A set of attributes (items)
R, S	A set of objects (transactions)
m	Minimum number of transactions for an envelope
n	Minimum number of items for an envelope
mm	Total number of transactions in an envelope
nn	Total number of items in an envelope

3.1.2 Patterns Characterized by (m, n)

Support envelopes capture the structure of all association patterns that can be characterized with respect to a minimum level of support, i.e., m , and a minimum number of items, i.e., n . We make this more formal in the following definition.

Definition 1. *Pattern Characterized by (m, n)*

A pattern characterized by a pair of positive integers (m, n) is a pattern that involves a set of attributes X and a set of transactions R such that (a) every attribute of X occurs in at least m transactions of R and every transaction of R contains at least n items of X , and (b) m and n are maximal.

Many common types of association patterns fall into this category, e.g., frequent itemsets and variations such as closed and maximal itemsets [9]. Error-tolerant itemsets (ETIs) [19], or more properly a slight variation of ETIs, also can also considered to be a pattern characterized a pair of interger by (m, n) , as we indicate in the following discussion

3.1.3 Symmetric Error-Tolerant Itemsets

An ETI is a frequent itemset that allows a specified fraction ϵ of the items to be missing from any one transaction, e.g., if $\epsilon = 0.2$, then 1/5 of the items in a transaction can be missing. While ETIs are an important approach for extending association patterns, there is a problem with ETIs as currently defined. For example, consider an ETI with $\epsilon = 0.2$, which consists of a set of 10 transactions and 5 items. While each transaction must contain at least 4 out of 5 items, items are not necessarily treated ‘equally,’ e.g., each transaction could be missing the same item. The originators of ETIs noted this issue, which can be problematic in the

degenerate case just mentioned, and avoided this degenerate case in an unspecified manner in their implementation. However, it seems desirable, even beyond the current discussion, to modify the notion of an ETI to be more in line with that of frequent itemsets, where all items play an equal role in the pattern. To this end, we define a symmetric ETI. [19].

Definition 2. *Symmetric Error Tolerant Itemset*

A symmetric ETI is a frequent itemset that allows a specified fraction ϵ of the items to be missing from any one transaction and a specified fraction β of transactions to be missing a particular item.

Any symmetric ETI can be characterized by a pair of integers (m, n) . For example, assume that we have a symmetric ETI has 12 transactions and 4 items, with $\epsilon = 1/4$ and $\beta = 1/3$. Then, $m = 8$, i.e., each item must appear in 8 transactions in the ETI, and $n = 3$, i.e., each transaction must contain at least 3 of the items in the ETI. Thus, symmetric ETI's are contained in support envelopes. The importance of this is that the support boundary is not just a bound on frequent, closed, and maximal itemsets, but also on patterns, such as symmetric ETIs, that are more general.

On a theoretical note, a support envelope can be considered as a type of symmetric ETI. Suppose that we have a support envelope characterized by (m, n) . If mm is the total number of transactions in the envelope and nn is the total number of items in the envelope, then the support envelope is a symmetric ETI with $\epsilon = n/nn$ and $\beta = m/mm$. However, support envelopes have properties not shared by all symmetric ETIs.

3.2 Definition of a Support Envelope

We now formally define a support envelope.

Definition 3. *Support Envelope*

A support envelope $E_{m,n}$ characterized by a pair of positive integers (m, n) is a set of items $X \subseteq \mathcal{I}$ and a set of transactions $R \subseteq \mathcal{T}$ which contains all patterns characterized by any pair of positive integers (m', n') such that $m' \geq m$ and $n' \geq n$.

Expressed another way, the transactions and items of the support envelope are the union, respectively, of all the transactions and items of all patterns characterized by any (m', n') , such that $m' \geq m$ and $n' \geq n$. Also, by construction, there cannot be more than one support envelope per (m, n) pair. However, as discussed previously and below, the support envelopes for different pairs of integers (m, n) may be identical, and in that case, we will refer to the support envelope using the largest values of m and n .

3.3 The Support Envelope Algorithm (SEA)

The algorithm for finding support envelopes—SEA—is shown below. We analyze the complexity, convergence, and correctness of SEA, as well as discussing how SEA can be used

Algorithm 1 Support Envelope Algorithm (SEA).

- 1: {Input: Data matrix \mathcal{D} and a pair of positive integers m and n }
 - 2: **repeat**
 - 3: Eliminate all rows whose sum is less than n .
 (Eliminate all transactions with fewer than n items.)
 - 4: Eliminate all columns whose sum is less than m .
 (Eliminate all items in fewer than m transactions.)
 - 5: **until** \mathcal{D} does not change
 - 6: **return** the set of rows (transactions) and columns (items) remaining in \mathcal{D}
-

Table 7: Evaluation Data Sets

Data Set	# Transactions	# Items	Density
chess	3196	75	0.49
mushroom	8124	119	0.19
la1	3204	31472	0.0048
kosarak	990,002	41,270	0.00019

Table 8: Statistics From Computing Support Envelopes

Data Set	# Envelopes	Time (sec)	Avg Iter
chess	353 (37)	3.4 (0.6)	2.1
mushroom	535 (22)	4.7 (0.8)	2.1
la1	26,678 (168)	570.2 (29.6)	3.1
kosarak	110,606 (318)	46,807 (390.8)	3.0

to used as the basis of algorithms for finding envelopes on the support boundary or all the envelopes of a data set. Initial implementations of these algorithms are available from [17].

Some data sets that will be used to illustrate various aspects of the following algorithms are listed in Table 7. The first three data sets can be found at the Frequent Itemset Mining Implementations Repository [7], while the LA1 data set is from the LA data of TREC-5 [1]. Table 8 shows various statistics related to the computation of support envelopes for the given data sets. (The results in this table were gathered by experiments performed on a 3.2 GHz Intel Xeon[®] Linux system.) The first number in the *# Envelopes* column is the number of envelopes for a data set, while the number in parentheses is the number of envelopes on the support boundary. The first number in the *Time* column is the time (in seconds) required to compute all the support envelopes for a data set, while the number in parentheses is the time (in seconds) to compute the support boundary.

Complexity: The time complexity of SEA is proportional to (a) the number of iterations, k , and (b) the amount of time to test each row and column to see if it should be eliminated. For a sparse matrix representation, the time to sum a row or column and compare this sum to m or n , is proportional to the number of non-zero entries in the row or column.

In turn, the total time required to check which rows and columns should be eliminated is proportional to, nnz , the number of non-zero entries in the matrix. Thus, the overall time complexity of SEA is $O(M * N * nnz)$, where k is the number of iterations required.

If a parent of a support envelope is known, then finding a support envelope is somewhat more efficient because we can use the transactions and items of the parent envelope to provide a smaller starting matrix. This effect is more important for larger data sets. For example, with the kosarak data set shown in Table 7, the average number of iterations is reduced from 8 to 3 by using a parent envelope. For the LA1 data set, which is significantly smaller, the average number of iterations is reduced from 4 to 3. For mushroom and chess, the reduction is from 3 to 2.

The space required for SEA is $O(nnz)$ plus the amount of space required to store the support envelopes. While storage of the row and column indices can potentially be large, there are a couple strategies that can be used to significantly reduce storage requirements. First, we can store only the indices in which an envelope differs from a parent envelope. (Preliminary investigations indicate that this approach seems promising.) A second approach to reducing storage is to not store the row and column indices at all since the entire envelope can be quickly recomputed if needed.

Convergence: We show that SEA converges to a solution in a fixed number of steps.

Theorem 3.1. *SEA converges to a solution in a fixed number of steps whenever the input is a binary matrix \mathcal{D} and positive integers m and n .*

Proof. Either SEA eliminates a row or column at each step, eventually producing an empty matrix, or else SEA terminates after a step in which \mathcal{D} is unchanged. Thus, SEA will converge to a solution in at most $\min(M, N)$ steps. \square

As mentioned, typically, convergence occurs in a much smaller number of steps, i.e., less than 10 for the data sets in Table 7.

Correctness: Here we show that SEA finds support envelopes.

Theorem 3.2. *Given the data matrix \mathcal{D} and a pair of positive integers (m, n) , the set of items $Y \subseteq \mathcal{I}$ and transactions $S \subseteq \mathcal{T}$ returned by SEA are the support envelope of \mathcal{D} characterized by (m, n) , i.e., $SEA(\mathcal{D}, m, n) = E_{m,n}$.*

Proof. Consider a pattern that is characterized by $m' \geq m$ and $n' \geq n$ and that involves the set of items $X \subseteq \mathcal{I}$ and the set of transactions $R \subseteq \mathcal{T}$. By Definition 1, every transaction in R must contain at least n items of X and every item in Y must occur in at least m transactions of R . Consequently, none of the transactions of R can be eliminated in Step 3 of SEA since they are ‘supported’ by the items of Y . Also, none of the items of Y can be eliminated in Step 4 of SEA since they are ‘supported’ by the transactions of R . Thus, $X \subseteq Y$ and $R \subseteq S$, i.e., the items and transactions of this pattern are subsets of the set of items and attributes found by SEA. From this, we conclude that the items and attributes returned by SEA contain those of $E_{m,n}$. We still need to show that $E_{m,n}$ contains the items and transactions returned by SEA. However, the set of transactions and items returned by SEA is certainly a pattern characterized by (m, n) and must belong to $E_{m,n}$. \square

An Algorithm for Finding the Boundary A straightforward approach to finding the support boundary is to perform a binary search over possible values of n for each value of m , or to perform a binary search over possible values of m for each value of n . Such an approach results in a conservatively estimated time complexity of $O(nnz * \min(N \log(M), M \log(N)))$. (Actually, we have observed that it is somewhat better to choose do a binary search over values of n if there are fewer rows than columns, and vice-versa.) We omit a detailed description of this algorithm. An implementation of this algorithm can be found at [17] and statistics are presented in Table 8.

Algorithms for Finding All Support Envelopes A simple approach to finding all the support envelopes is, for each value of m , to find envelopes for each value of n up to the limit imposed by the support boundary. (Again, we have observed that it is somewhat better to perform our algorithm by varying n for a fixed m if there are fewer rows than columns, and vice-versa.) An implementation of this algorithm can be found at [17], and statistics from the run of this algorithm are presented in Table 8.

Scalability A basic strategy for parallelizing the previous algorithm is to have each processor compute all the envelopes for a fixed value of m (or n) and all values of n (or m). A similar approach could be used to parallelize the algorithm for finding the support boundary. Issues that would need to be considered include load balancing and the generation of the duplicate support envelopes by different processors.

Another important scalability issue that needs investigation is how to best adapt the SEA algorithm for cases where the data is large, i.e., the data does not fit in main memory and it is desirable to access the data in a sequential manner and to make as few passes over the data as possible. SEA can be implemented to access data in a sequential manner if two copies of the data are kept, one with the data ordered by rows and the other with the data ordered by columns.

3.4 Properties of Support Envelopes

We have already encountered a number of the properties of support envelopes in the previous discussion. For completeness and clarity, we summarize the properties of support envelopes in this section. Actual proofs of these properties can be found in Appendix A.

For for each pair of integers (m, n) , where $1 \leq m \leq M$ and $1 \leq n \leq N$, there is an associated support envelope—perhaps the empty support envelope—which is the support envelope that SEA will return when given (m, n) and the data set \mathcal{D} . However, as we saw in the example, the same support envelope may be associated with more than one pair of integers since the set of items and transactions that meets the (m, n) constraints may actually meet stronger constraints. (We characterize a support envelope by the strongest support and item constraints (i.e., by the maximum (m, n) values) that it satisfies.) A consequence of these two facts is that the number of support envelopes in a data set is at most $M * N$.

One support envelope may contain another support envelope, i.e., the items and transactions of one envelope may be subsets of the items and transactions of another envelope. Said another way, a support envelope defines a block of the data set (matrix) \mathcal{D} , and this

block may contain the blocks that correspond to other envelopes. Indeed, for a given support envelope characterized by (m, n) , any envelope with a larger value of m , or a larger value of n , or both, must be a subset of the original envelope. This is not surprising, since increasing m and n will necessarily eliminate items and transactions.

Like frequent itemsets, support envelopes form a lattice. For support envelopes the order relationship is the subset relationship described above. Support envelopes which contain no other (non-empty) envelopes are said to be on the support boundary.

An interesting property that was not mentioned in the example, is that the density of envelopes, i.e., the fraction of 1's in the block represented by the envelope, increases as we move down the lattice. Thus, if one envelope contains another, it must be less dense (or at least no more dense) than the envelope that it contains. This means that the support boundary must contain the densest envelopes in the support lattice. This does not mean, however, that every envelope on the support boundary is denser than any envelope elsewhere in the support lattice.

There can be at most $\min(M, N)$ envelopes on the support boundary. otherwise, two envelopes on the boundary would have the same value of m or n and one would contain the other. For a similar reason, the values of m and n are inversely related as we move along the support boundary, i.e., if we order the support envelopes to have increasing values of m , then successive envelopes will have decreasing values of n .

4 Exploring Transaction Data Sets with Support Envelopes

In this section we demonstrate how support envelopes can be used to explore the transaction data sets of Table 7.

4.1 Dense Data Sets: Mushroom and Chess

We begin by showing the support envelopes for the mushroom and chess data sets in figures 4 and 5, respectively. As before, we have colored the points to show the density of the support envelopes, i.e., the fraction of 1's in block of items and attributes defined by the envelope. Both of these data sets are relatively dense compared to many transaction data sets and thus, the overall densities of the envelopes are relatively high. For both data sets, the density of the envelopes are highest for envelopes with many transactions, but few items. This reflects the fact that both data sets have a few items that occur in most transactions, and these items appear in a support envelope with a single pattern since no other collection of items has such a high support. In some cases, these 'overly frequent' items should be discarded since they yield little information.

Also, for both data sets, there are no envelopes near the origin. This reflects the fact that every item occurs in some minimum number of transactions and that every transaction contains some minimum number of items. For example, for mushroom, every item occurs

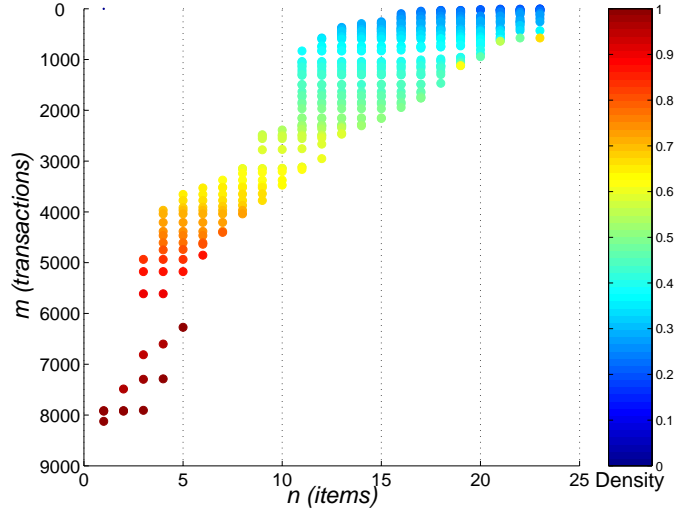


Figure 4: Support envelopes for mushroom data set. Must be viewed in color.

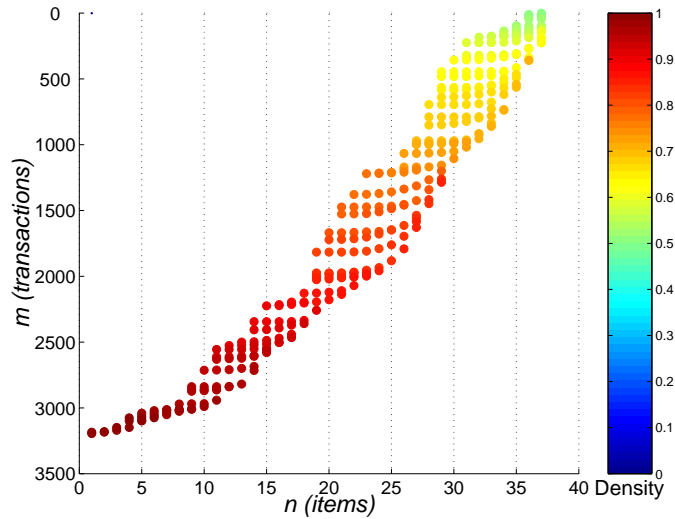


Figure 5: Support envelopes for chess data set. Must be viewed in color.

in at least 4 transactions and every transaction contains exactly 23 items. For chess, every transaction contains 37 items and every item occurs in at least 1 transaction. However, all association patterns characterized by (m, n) values where there are no points in the figures, can be found in the transactions and items of the envelopes on the inner boundary of the support lattice. Indeed, recall that the root envelope corresponding to $(1, 1)$ is guaranteed to encompass all association patterns in the data set. For mushroom, the root envelope is characterized by the pair $(4, 23)$, while for chess, the root envelope is characterized by $(1, 37)$.

Despite some similarities, the two sets of support envelopes are quite distinct, e.g., the boundary of the mushroom data set curves inward (towards the origin), while that of the

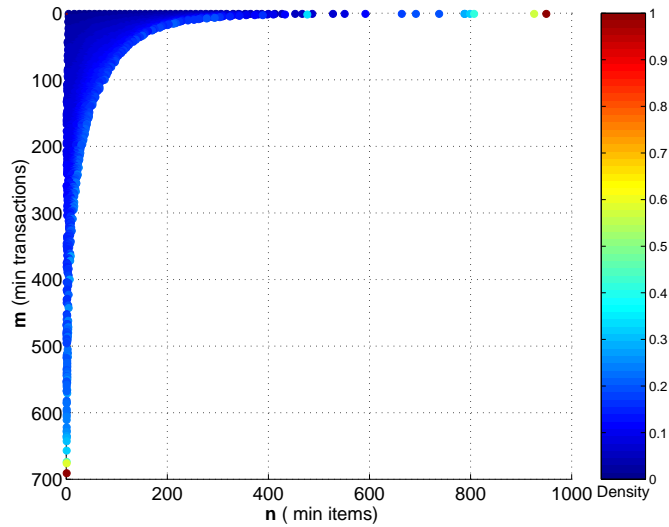


Figure 6: Support envelopes for LA1 data set. Must be viewed in color.

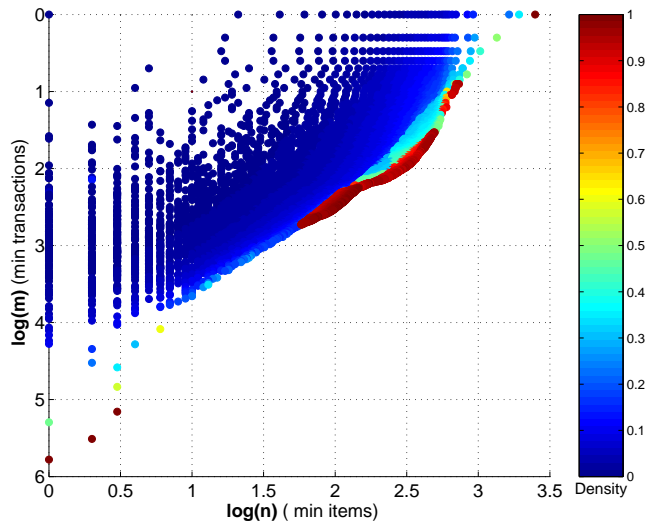


Figure 7: Support envelopes for kosarak data set. Note: log scale. Must be viewed in color.

chess data set curves outward. Our observation, which is based on a limited number of data sets, is that for sparser data, the boundary is concave, while for denser data sets, the boundary is convex. Of course, the boundary can also be more complicated, i.e., the boundary is both concave and convex in different sections.

As another example of how we might use support envelopes to analyze the structure of support patterns in a data set, we remark that there is an unusually dense support envelope with 576 transactions and 23 items in the mushroom data set. It has a density of 0.65,

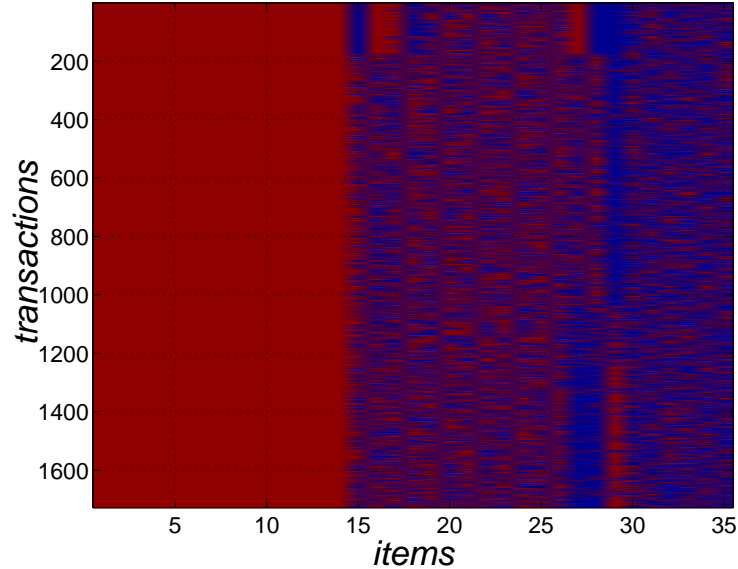


Figure 8: Plot of mushroom support envelope (576,23).

while most of the envelopes around it have a density of roughly 0.3–0.4. Since this support envelope is somewhat anomalous, we show its plot in Figure 8. The items and transactions of the envelope have been sorted to put the denser region of the support envelope toward the upper left hand corner. Each red pixel corresponds to a 1 in the envelope, while each blue pixel corresponds to a 0. The figure shows that there is a group of 14 items that occur together in every one of the 1728 transactions in the envelope. (Remember that the total number of transactions (mm) and items (nn) in an envelope is different from the m (support) and n (item) constraints.)

Upon further analysis, we discovered that one of the columns was the column 48, ‘gill-color:buff.’ There are exactly 1728 instances of item 48, every one of which occurs with 13 other items (one of which is ‘poisonous’). This support envelope is somewhat more dense than the others around it because the co-occurrence of 13 items is larger than is typical for this data set.

4.2 Sparse Data Sets: LA1 and kosarak

For contrast, we show a graph of the LA1 support envelopes in Figure 6. LA1 is much more sparse and has more items, although fewer transactions, than either mushroom or chess. The support envelopes are relatively sparse compared to those of both mushroom and chess. However, the density does increase from 0.0048 for envelopes near the origin (upper left) to around 0.2 for envelopes at the boundary. The two red dots in the upper right and lower left represent, respectively, the longest document and the most frequent item (word). The support boundary is extremely concave in LA1.

The set of kosarak envelopes is shown in Figure 7. We have used \log_{10} scales for both

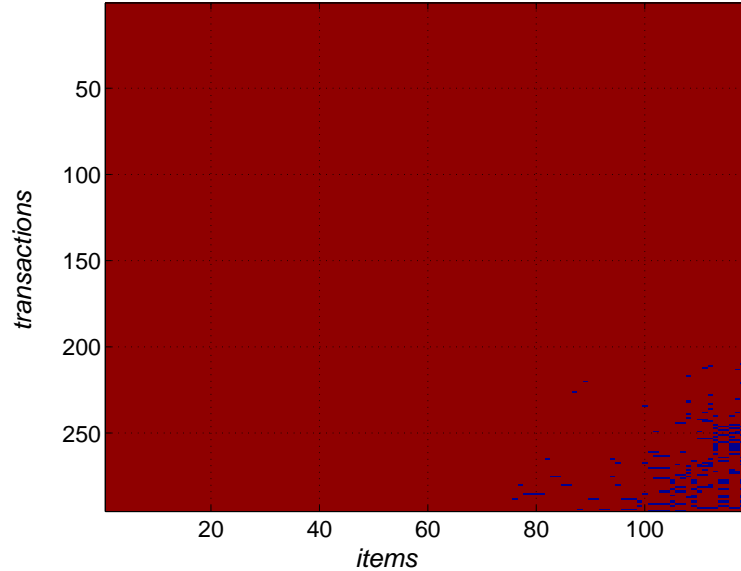


Figure 9: Plot of kosarak support envelope (276,113).

the m and n axes because the m and n values have a much wider range of values. However, if this were not the case, the support boundary of kosarak would be even more concave than that of LA1.

Interestingly, some of envelopes on or near the kosarak support boundary, with values of m between 100 and 1000 (between values of 2 and 3 on the log10 scale) have relatively high density. We investigated the boundary support envelope (276, 113). Note that it would be difficult to investigate patterns with such low support using traditional approaches. Indeed, the results of the FIMI workshop [7], show that current algorithms to find frequent, closed, or maximal itemsets start to experience a sharp growth in run time at a support threshold of about 0.1% (990 transactions.) The support value for envelope (276, 113) is even lower than this, i.e., less than 0.03%. Figure 9 shows a plot of the support envelope corresponding to (276, 113), which has a density 0.993. This envelope contains a frequent itemset of approximately 80 items and 250 transactions, and we tried to find this itemset using the algorithm, `fpmax`, which is available from the FIMI website [7]. The program ran for more than a week on a 2.8 GHz Intel Xeon[®] and produced a file with almost 5 million maximal itemsets. (We are not sure whether the program terminated normally since memory usage had grown to 1.2 GB shortly before it terminated.) In contrast, the support boundary for kosarak was computed in approximately 15 minutes on the same machine. (Computing the boundary on a newer machine took only about 7 minutes.)

5 Extensions of the Basic Approach

In this section, we discuss two additional extensions of the basic approach: computing support envelopes with respect to specific transactions or items and adding additional constraints to support envelopes.

5.1 Transaction Specific Support Envelopes

When support envelopes are not very dense, they are of interest only for the patterns that can be extracted from them and for what they tell us about the overall structure of association patterns. However, if a support envelope is relatively dense, then the envelope is interesting as a pattern in its own right. Thus, it may be useful to seek special situations where support envelopes are more likely to be dense. To that end, we consider support envelopes associated with a particular transaction (or item).

To illustrate this idea we show an example using the LA1 data set. We constructed a data set specific to the first document (transaction) in LA1 by eliminating all words (items) that do not appear in this document. (This document comes from the ‘Financial’ class.) We then computed the support boundary of this document specific data set. Since we are working in the document domain, we display the results by a table that shows the parameters of the support envelopes and the words (items) that are part of the support envelopes. (LA1 was processed using standard information retrieval techniques, e.g., the words are stemmed and stop words are eliminated.) The first support envelope consists of the selected document and all its words, while successive envelopes consist of smaller subsets of these words. For successive envelopes along the boundary, the number of documents increases, while the number of words (usually) decreases. In other words, the pattern represented by the support envelope is becoming more general as we move along the support boundary from lower to higher values of m . The most interesting support envelopes are those in the middle, since they are not as specific as a particular document, nor so general (frequent) as to be uninteresting. The envelope with the highest value of m contains the word, ‘home,’ which is the most frequent word contained by the first document.

These support envelopes have a clear theme of a bank bailout of the home savings and loan industry. However, the patterns represented by these support envelopes are only moderately strong, i.e., the support envelopes that contain more than a few documents and items tend to have densities in the range of 0.5 to 0.7. Nonetheless, recall that the regular support envelopes for LA1 only have a density that is, at best, around, 0.2.

5.2 Constrained Support Envelopes

A special case of extending support envelopes that is easy to implement and interpret involves adding a constraint to the support envelope process. For example, very frequent items show up in many association patterns, but provide little useful information. One way to address this issue is to require that the fraction of an item involved in an association pattern meet

Table 9: Support boundary envelopes LA1 restricted to words (items) in the first document (transaction).

m	n	mm	nn	words
1	26	1	26	bailout bank billion board clos cost countri deal expect feder goal govern hemorrhag home hoyle industri karl loan overall quote rest sav spokesman stop texa throughout
2	16	2	16	bailout bank billion board clos cost deal expect feder govern home industri loan rest sav texa
3	13	3	13	bailout bank billion board cost deal feder govern home industri loan sav texa
5	11	10	15	bailout bank billion board cost countri deal expect feder govern home industri loan rest sav
7	10	30	21	bailout bank billion board clos cost countri deal expect feder goal govern home industri loan rest sav spokesman stop texa throughout
13	9	38	17	bank billion board clos cost countri deal expect feder govern home industri loan sav spokesman stop throughout
21	8	46	14	bank billion board clos cost countri deal expect feder govern home industri loan sav
33	7	80	14	bank billion board clos cost countri deal expect feder govern home industri loan sav
54	6	150	14	bank billion board clos cost countri deal expect feder govern home industri loan sav
80	5	197	11	bank billion board cost countri deal expect feder govern home industri
121	4	348	11	bank billion board cost countri deal expect feder govern home ndustri
193	3	552	9	board cost countri deal expect feder govern home industri
298	2	532	4	countri expect govern home
676	1	676	1	home

some minimum threshold. (This is formally defined by the notion of a hyperclique pattern [18].) It is straightforward to add such a constraint to the computation of support envelopes. In other words, given a specified fraction f between 0 and 1, we eliminate an item if does not occur in at least m transactions and at least f of its total support occurs among the transactions of the support envelope. For example, if $f = 0.5$, then at least half of the supporting transactions of an items must occur in the support envelope. While a threshold could also be applied to the rows as well, we do not pursue that approach here.

As an example, we find the support boundary specific to the first document in the LA1 data set using an threshold of 0.5. The results are shown in Table 10. Note that there are far fewer envelopes, but that they still seem to capture much of the meaning of the previous envelopes. In particular, the first envelope on the boundary no longer consists of all the

Table 10: Support boundary envelopes LA1 restricted to words in the first document and with the additional constraint that 50% of the occurrences of a word must be in the envelope.

m	n	mm	nn	words
140	3	894	16	bank billion board clos cost countri deal expect feder goal govern home industri rest spokesman stop
287	2	669	5	countri expect feder govern home
676	1	676	1	home

terms from the first document of LA1 that was used to generate the reduced data set.

6 Related Work

To our knowledge, the notion of a support envelope is new, although the concept of an ‘envelope’ is common in mathematics, where an envelope is a mathematical entity—usually a curve—bounds a collection of other mathematical entities. The related notion of a cover has been used before in association analysis, but for association rules [4, 13, 15].

The idea of support envelopes was inspired partly by various concepts in lattice theory [5] and formal concept analysis [6], especially those ideas that have found their way into association analysis [8, 6, 12]. In particular, a key motivating concept for us is the notion of efficiently representing frequent itemsets via a lattice of closed itemsets [2, 14, 20]. More generally, closed itemsets are a specific example of condensed representations [3, 11]. However, support envelopes are not a condensed representation since it is necessary to keep the original matrix if the goal is to use support envelopes to actually find itemsets.

The idea of error tolerant itemsets (ETIs) [19] also played an important role in our thinking. In particular, ETIs emphasize the notion that is useful to consider a version of frequent itemsets that relaxes the requirement that all items be contained in all transactions. Support envelopes also embrace this idea, and as we showed in Section 3.1.3, are a special type of ETI.

Finally, there has been some recent work on computing tight lower bounds for distributions of frequent and maximal frequent itemsets [16]. Although this is a different line of inquiry than we have pursued in this paper, such work is also quite relevant to understanding the structure of support based patterns in transaction data.

7 Conclusions and Future Work

In this paper, we introduced support envelopes, a new tool that is useful for exploring the high-level structure of association patterns in a transaction data set. Important aspects of support envelopes are that they are not encumbered by a support threshold and that they provide information not only about frequent, closed, and maximal itemsets, but also about more general patterns, such as symmetric error-tolerant itemsets. Support envelopes

provide both a theoretical basis for understanding this structure and a graphical technique for visualizing the structure. Furthermore, there are simple and efficient algorithms to compute a single support envelope, the support envelopes on the support boundary, or all the support envelopes of a data set.

There is considerable potential for future work. Current implementations for finding the support boundary support or all support envelopes and could be improved and parallelized to provide additional performance. Also, the theoretical properties of support envelopes should be further explored. In particular, the extensions of support envelopes deserve further investigations since they may be useful both for finding actual patterns, as well as for providing additional information about the overall structure of association patterns. In particular, we hope to investigate whether we can extend the notion of support envelopes to continuous data. Finally, further work is necessary to more fully understand what information can be extracted from scatter plots of support envelopes.

8 Acknowledgments

This work was partially supported by NASA grant # NCC 2 1231, by DOE/LLNL grant W-7045-ENG-48, by NSF grant IIS-0308264, and by the Army High Performance Computing Research Center under the auspices of the Department of the Army, Army Research Laboratory cooperative agreement number DAAD19-01-2-0014. The content of this work does not necessarily reflect the position or policy of the government and no official endorsement should be inferred. Access to computing facilities was provided by AHPCRC and Minnesota Supercomputing Institute.

References

- [1] Text retrieval conference.
- [2] Jean-Francois Boulicaut and Artur Bykowski. Frequent closures as a concise representation for binary data mining. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 62–73, 2000.
- [3] J.F. Boulicaut, A. Bykowski, and C. Rigotti. Free-sets: a condensed representation of boolean data for the approximation of frequency queries. *Data Mining and Knowledge Discovery Journal (DMKD)*, 7(1):5–22, 2003.
- [4] Laurentiu Cristofor and Dan A. Simovici. Generating an informative cover for association rules. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan*, pages 597–600. IEEE Computer Society, 2002.
- [5] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 2nd edition, May 2002.

- [6] B. Ganter and R. Wille. *Formal Concept Analysis – Mathematical Foundations*. Springer, May 1999.
- [7] Bart Goethals and Mohammed J. Zaki. Frequent itemset mining implementations repository. This site contains a wide-variety of algorithms for mining frequent, closed, and maximal itemsets, <http://fimi.cs.helsinki.fi/>.
- [8] Dimitrios Gunopulos, Heikki Mannila, Roni Khardon, and Hannu Toivonen. Data mining, hypergraph transversals, and machine learning (extended abstract). In *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 209–216. ACM Press, 1997.
- [9] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, August 2000.
- [10] Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. Algorithms for association rule mining – a general survey and comparison. *SIGKDD Explorations*, 2(1):58–64, July 2000.
- [11] Heikki Mannila and Hannu Toivonen. Multiple uses of frequent sets and condensed representations (extended abstract). In *Knowledge Discovery and Data Mining*, pages 189–194, 1996.
- [12] Heikki Mannila and Hannu Toivonen. Levelwise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.
- [13] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Closed set based discovery of small covers for association rules. In *Proc. 15emes Journees Bases de Donnees Avancees, BDA*, pages 361–381, 1999.
- [14] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1):25–46, 1999.
- [15] V. Pudi and J. Haritsa. Generalized closed itemsets: A technique for improving the conciseness of rule covers. In *Proc. of 19th IEEE Intl. Conf. on Data Engineering, Bangalore, India, March 2003*, pages 714–716. IEEE Computer Society, 2003.
- [16] Ganesh Ramesh, William A. Maniatty, and Mohammed J. Zaki. Feasible itemset distributions in data mining: theory and application. In *Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 284–295. ACM Press, 2003.
- [17] Michael Steinbach. Preliminary implementation of support envelope algorithms. <http://www.cs.umn.edu/steinbac/se/se.html>.

- [18] H. Xiong, P. Tan, and V. Kumar. Mining strong affinity association patterns in data sets with skewed support distribution. In *Proc. of the 3rd IEEE International Conf. on Data Mining*, pages 387–394, 2003.
- [19] Cheng Yang, Usama M. Fayyad, and Paul S. Bradley. Efficient discovery of error-tolerant frequent itemsets in high dimensions. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 194–203. ACM Press, 2001.
- [20] Mohammed J. Zaki and Ching-Jui Hsiao. Charm: An efficient algorithm for closed itemset mining. In *Second SIAM International Conference on Data Mining*, 2002.
- [21] Mohammed Javeed Zaki and Mitsunori Ogihara. Theoretical foundations of association rules. *3rd SIGMOD'98 Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD)*, Seattle, WA, June 1998.

A Proofs of Support Envelope Properties

The following theorems establish some properties of support envelopes:

Subset Properties We can define the subset relationship \subseteq between two envelopes, E and E' as follows: $E \subseteq E'$ if and only if the transactions of $E \subseteq$ the transactions of E' & the items of $E \subseteq$ the items of E' . A notion of subset without equality can be similarly defined. The following theorems establish that envelopes with larger values of m and/or n are subsets of those with smaller values of m and/or n .

Theorem A.1. *For a fixed m , as n increases, the transactions and items of successive support envelopes are subsets of the transactions and items of the transactions and items of the preceding support envelopes, i.e., $E_{m,n+1} \subseteq E_{m,n}$.*

Proof. The ‘block’ consisting of the transactions and items of the support envelope, $E_{m,n+1}$, will belong to $E_{m,n}$ since every transaction of $E_{m,n+1}$ will meet the m threshold from the support within the items of $E_{m,n+1}$ and every item of $E_{m,n+1}$ will meet the n threshold from the support within the transactions of $E_{m,n+1}$. □

Theorem A.2. *For a fixed n , as m increases, the transactions and items of successive support envelopes are subsets of the transactions and items of the transactions and items of the preceding support envelopes, $E_{m+1,n} \subseteq E_{m,n}$.*

Proof. Similar to the previous proof. □

Theorem A.3. *If we have two support envelopes, $E_{m,n}$ and $E_{m+1,n+1}$, then $E_{m+1,n+1} \subseteq E_{m,n}$.*

Proof. This follows from the previous two results and the transitivity of the subset operator. □

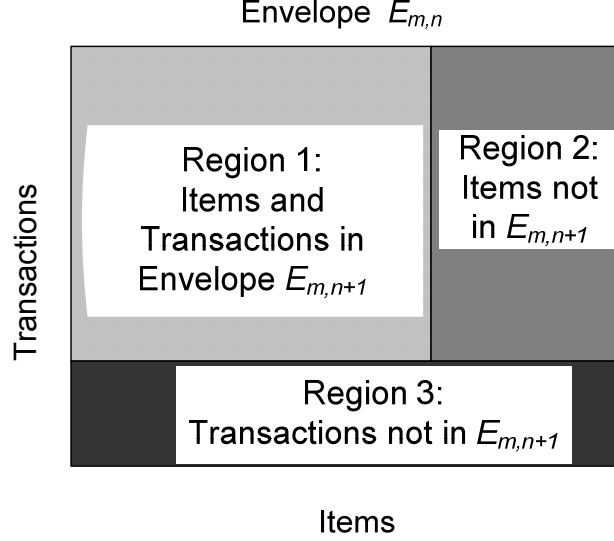


Figure 10: How an envelope, $E_{m,n}$ is characterized by which portions are part of envelope $E_{m,n+1}$.

Density Properties We define the density of a support envelope as the fraction of the block of D that consists of 1's. Not surprisingly, the density of envelopes increases with increasing values of m and n . Thus, the top or root envelope, which consists of the entire data matrix, represent the least dense envelope, while the envelopes on the support boundary are the most dense. However, that does not mean that they will be dense in absolute terms.

Theorem A.4. *For a fixed m , as n increases, the density of successive support envelopes increases, i.e., $\text{density}(E_{m,n+1}) \geq \text{density}(E_{m,n})$.*

Proof. Figure 10 shows which portions of an envelope $E_{m,n}$ corresponds to a successive envelope $E_{m,n+1}$, assuming that transactions and items are appropriately reordered. The transactions in Region 3, i.e, that were in $E_{m,n}$, but which are not in $E_{m,n+1}$, must have had exactly m items within the block defined by $E_{m,n}$. (The value of m is unchanged between the two envelopes, and thus, the only way that one of these transactions would have been eliminated from $E_{m,n+1}$ is if it only has m items.) However, each transaction in Region 1, i.e., the set of transactions that belong to $E_{m,n+1}$, contains at least $m + 1$ items, and is thus, denser than any of the transactions in Region 3. The items in Region 2, i.e., those items that are in $E_{m,n}$, but are not in $E_{m,n+1}$, must contain less than m items in the transactions of Region 1. Thus, the items in Region 1 are less dense than than the items in Region 2. Hence, $E_{m,n+1}$, which consists of Region 1 is more dense than regions 2 and 3, and $\text{density}(E_{m,n+1}) \geq \text{density}(E_{m,n})$. \square

Theorem A.5. *For a fixed n , as m increases, the density of successive support envelopes increases, i.e., $\text{density}(E_{m+1,n}) \geq \text{density}(E_{m,n})$.*

Proof. Similar to the previous proof. \square

Theorem A.6. *If we have two support envelopes, $E_{m,n}$ and $E_{m+1,n+1}$, then $\text{density}(E_{m+1,n+1}) \geq \text{density}(E_{m,n})$.*

Proof. This follows from the previous two results and the transitivity of density. \square

Properties Related to the Support Boundary The support boundary consists of those envelopes which do not have any children, i.e., do not contain any other envelopes. As the following theorems show, for envelopes on the support boundary, as m increases, n decreases, and vice versa. Intuitively, a stronger support constraint reduces the number of items possible.

Theorem A.7. *If, for any $m < M$, we find the maximum possible n , i.e., n_max_m , for which a non-empty concept can be found, then $n_max_{m+1} \leq n_max_m$.*

Proof. For m and any $n > n_max_m$, the associated support envelope, $E_{m,n}$, is empty. By a previous result, $E_{m+1,n} \subseteq E_{m,n}$. Thus, if $n_max_{m+1} > n_max_m$, then $E_{m+1,q}$ is also empty. \square

Theorem A.8. *If, for any $n < N$, we find the maximum possible m , i.e., m_max_n , for which a non-empty support envelope can be found, then $m_max_{n+1} \leq m_max_n$.*

Proof. Similar to the previous proof. \square

Theorem A.9. *m and n are inversely related on the boundary, i.e., as m increases, n decreases, and vice-versa.*

Proof. Suppose not. Then there exist two support envelopes on the boundary characterized by $(m1, n1)$ and $(m2, n2)$, respectively, such that $m2 > m1$ and $n2 > n1$. However, from our previous results, we know that (a) $E_{m1,n2}$ is empty and (b) $E_{m2,n2} \subseteq E_{m1,n2}$. Thus, $E_{m2,n2}$ is empty, which is a contradiction as it cannot be a boundary concept. Thus, if $m2 > m1$, then $n1 < n2$. \square

If mm and nn are, respectively, the number of transactions and items in a support envelope, then mm and nn do not have a strictly anti-monotonic relationship for boundary concepts. However, for the data sets that we have examined, mm and nn do have an anti-monotone relationship, although with a few mild exceptions.

Theorem A.10. *There can be at most $\min(M, N)$ envelopes on the pattern boundary.*

Proof. We prove this for the situation where there are fewer objects than attributes, i.e., $N < M$. Suppose that there are more than M unique boundary envelopes. Then two of these envelopes must have the same value of n , since n can take on values only from 1 to $N < M$. Let these two envelopes be $E_{m1,n}$ and $E_{m2,n}$. If $m1 = m2$, then $E_{m1,n} = E_{m2,n}$, and we have violated the assumption that we are only considering unique boundary envelopes. Without loss of generality, assume that $m2 > m1$. Then, $E_{m1,n} \subseteq E_{m2,n}$ and this contradicts the assumption that $E_{m1,n}$ is a boundary envelope. The proof of the case of fewer objects than attributes is similar. \square

Lattice Properties As Theorem A.11 indicates, support envelopes form a lattice [5]. Furthermore, we say that an envelope E' is a child of another envelope E if $E' \subset E$ and there is no other envelope E'' such that $E'' \subset E$ and $E' \subset E''$. (In lattice theory terminology, E' is covered by E .) In this case, we also say that E is a parent of e_2 . In the data sets that we have used, we typically see that an envelope has zero, one or two children. However, envelopes can have more children. For example, the mushroom data set described in Table 7, has 14 envelopes out of 535 with more than 2 children, including 2 with 9 children. Likewise, an envelope can have more than two parents. For example, in the mushroom data set, one envelope has 12 parents.

Theorem A.11. *The set of all support envelopes $E_{m,n}$ is a lattice with the envelope, $E_{1,1}$, which contains all attributes and objects, as the top (maximum) element and the empty support envelope, which has no transactions or items as the bottom element of the lattice.*

Proof. (Sketch) A lattice [5] is a set together with an order relation and a meet and join operator. The \subseteq relation that we have already defined is an order relation for support envelopes, since it inherits the reflexive, anti-symmetric, and transitive properties of the original subset relation. In the ordinary lattice of sets the join and meet operators are union and intersection, respectively. If we define these operations on support envelopes by defining them in a pairwise manner, e.g., the intersection of two support envelopes is just the intersection of their respective sets of transactions and items. It is then straightforward to show that the meet and join operations of support envelopes are these pairwise intersection and join operations, respectively. The top element of lattice is the support envelope corresponding to the $(1, 1)$, i.e., the entire data set \mathcal{D} . The bottom element of the data set is the empty envelope. \square

Sets of Equivalent (m, n) Pairs A single support envelope may correspond to a number of (m, n) pairs, which, to use algebraic terminology, constitute an equivalence class. From Figure 2, we might assume that such equivalence classes of (m, n) pairs are either a single point or a rectangle. While this is often true, it is not true in general. More specifically, as the following theorems indicate, the shape of the equivalent (m, n) pairs can be a staircase shape as indicated in Figure 11.

Theorem A.12. *If we have two support envelopes, E_{m,n_1} and E_{m,n_2} , with $n_1 < n_2$ and $E_{m,n_1} = E_{m,n_2}$, then $E_{m,n_1} = E_{m,n_2} = E_{m,n}$, $\forall n, n_1 \leq n \leq n_2$.*

Proof. $E_{m,n} \subseteq E_{m,n_1}$ and $E_{m,n_2} \subseteq E_{m,n}$ by Theorem A.2. Since $E_{m,n_1} \subseteq E_{m,n_2}$, the transitivity of the subset operator implies that $E_{m,n_1} = E_{m,n}$. \square

For a horizontal ‘line’ of (m, n) values, if both endpoints are associated with a particular support envelope, then all the points in between must also be associated with the same envelope.

Theorem A.13. *If we have two support envelopes, $E_{m_1,n}$ and $E_{m_2,n}$, with $m_1 < m_2$ and $E_{m_1,n} = E_{m_2,n}$, then $E_{m_1,n} = E_{m_2,n} = E_{m,n}$, $\forall m, m_1 \leq m \leq m_2$.*

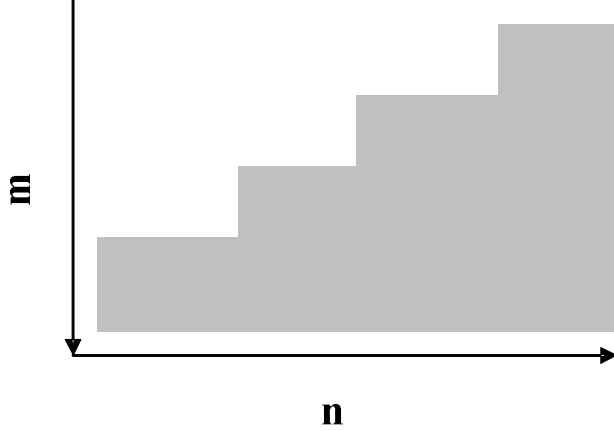


Figure 11: Most general shape of (m, n) pairs corresponding to a single support envelope.

Proof. Similar to A.12. □

For a vertical ‘line’ of (m, n) values, if both endpoints are associated with a particular support envelope, then all the points in between must also be associated with the same envelope.

Theorem A.14. *If we have two support envelopes, E_{m_1, n_1} and E_{m_2, n_2} , with $m_1 < m_2$, $n_1 < n_2$, and $E_{m_1, n_1} = E_{m_2, n_2}$, then $E_{m_1, n_1} = E_{m_2, n_2} = E_{m, n}$, $\forall m$, $m_1 \leq m \leq m_2$, $\forall n$, $n_1 \leq n \leq n_2$.*

Proof. $E_{m_1, n_1} \subseteq E_{m_1, n_2}$ and $E_{m_2, n_2} \subseteq E_{m_1, n_2}$ by theorems A.1 and A.2. Since $E_{m_1, n_1} \subseteq E_{m_2, n_2}$, the transitivity of the subset operator implies that $E_{m_1, n_1} = E_{m_1, n_2}$. In a similar manner we can show that $E_{m_1, n_1} = E_{m_2, n_2}$. We then have support envelopes at four points of a square, and the application of theorems A.1 and A.2 completes the proof. □

For a diagonal ‘line’ of (m, n) values, if both endpoints are associated with a particular support envelope, then not only are all the points on the line associated with the same envelope, but all points in the block defined by the two points are also be associated with the same envelope.

Theorem A.15. *If we have two support envelopes, E_{m_2, n_1} and E_{m_1, n_2} , with $m_1 < m_2$, $n_1 < n_2$, and $E_{m_2, n_1} = E_{m_1, n_2}$, then $E_{m_2, n_1} = E_{m_1, n_2} = E_{m_2, n}$, $\forall n$, $n_1 \leq n \leq n_2$, and $E_{m_2, n_1} = E_{m_1, n_2} = E_{m_1, n_2}$, $\forall m$, $m_1 \leq m \leq m_2$.*

Proof. Because E_{m_2, n_1} and E_{m_1, n_2} are equal, each transaction in both envelopes has n_2 items and each item occurs in m_2 transactions. Thus, this support envelope is the support envelope E_{m_2, n_2} . Application of theorems A.1 and A.2 completes the proof. □

Two anti-diagonal (m, n) points are a more complicated situation. Basically, if we draw a horizontal line from the leftmost/bottommost point and a vertical line from the rightmost/topmost point, the point of intersection between those two lines and all points in

between must be associated with the same envelope as the two points. This fact is what causes the potential staircase shape of a support envelope.