Mining Approximate Frequent Itemsets In the Presence of Noise: Algorithm and Analysis

[1]Jinze Liu, [1]Susan Paulsen, [2] Xing Sun, [1]Wei Wang, [1,2]Andrew Nobel, [1]Jan Prins
[1]Department of Computer Science
[2]Department of Statistics and Operations Research
University of North Carolina, Chapel Hill, NC 27599
[1]{liuj, paulsen, weiwang, prins}@cs.unc.edu
[2]{xingsun, nobel}@email.unc.edu

Abstract

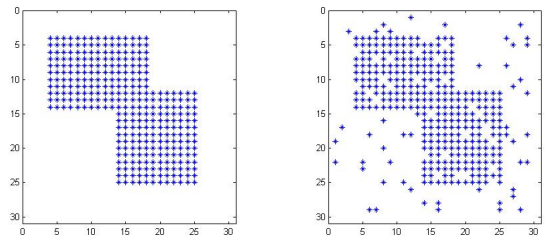
Frequent itemset mining is a popular and important first step in the analysis of data arising in a broad range of applications. The traditional "exact" model for frequent itemsets requires that every item occur in each supporting transaction. However, real data is typically subject to noise and measurement error. To date, the effect of noise on exact frequent pattern mining algorithms have been addressed primarily through simulation studies, and there has been limited attention to the development of noise tolerant algorithms.

In this paper we propose a noise tolerant itemset model, which we call approximate frequent itemsets (AFI). Like frequent itemsets, the AFI model requires that an itemset has a minimum number of supporting transactions. However, the AFI model tolerates a controlled fraction of errors in each item and each supporting transaction. Motivating this model are theoretical results (and a supporting simulation study presented here) which state that, in the presence of even low levels of noise, large frequent itemsets are broken into fragments of logarithmic size; thus the itemsets cannot be recovered by a routine application of frequent itemset mining. By contrast, we provide theoretical results showing that the AFI criterion is well suited to recovery of block structures subject to noise.

We developed and implemented an algorithm to mine AFIs that generalizes the level-wise enumeration of frequent itemsets by allowing noise. We propose the noise-tolerant support threshold, a relaxed version of support, which varies with the length of the itemset and the noise threshold. We exhibit an Apriori property that permits the pruning of an itemset if any of its sub-itemset is not sufficiently supported. Several experiments presented demonstrate that the AFI algorithm enables better recoverability of frequent patterns under noisy conditions than existing frequent itemset mining approaches. Noise-tolerant support pruning also renders an order of magnitude performance gain over existing methods.

1 Introduction

Relational databases are ubiquitous, cataloging everything from market-basket data [1] to gene-expression data [8, 7]. One common representation for relational databases is a binary matrix. Rows in the matrix correspond to objects, while columns represent various attributes of the objects. The binary value of each matrix entry then indicates the presence (1) or absence (0) of an attribute for a given object. For example, in a market-basket database, rows represent transactions, columns represent product items, and a binary entry indicates whether an item is contained in a given transaction [1, 2]. Frequent itemset mining [1] is a key technique for the analysis of such data.



Embedded Pattern(X) Observed Pattern(Y)

Figure 1: Patterns with and without noise.

In the binary representation, a *frequent itemset* corresponds to a sub-matrix of 1s containing a sufficiently large set of rows (transactions). Although frequent itemset mining was originally developed to discover association rules, its broader application provides the basis for subspace clustering and for building classifiers. In these applications the ultimate goal is to discover interesting associations between object and attribute sets, rather than associations among attributes alone. One important experimental application of frequent itemset mining is the exploration of gene expression data, where the joint discovery of both the set of conditions that significantly effect gene regulation and the set of co-regulated genes is of great interest.

In real data applications a "1" can be accidentally recorded as "0" and vice versa. In a *transaction database*, the noise can arise from both accidents of

the market and the vagaries of human behavior. Items expected to be purchased together by a customer might not appear together in a particular transaction either because one item is out of stock or because it has been overstocked by the customer. *Microarray data* is likewise subject to measurement noise, stemming from the underlying experimental technology and the stochastic nature of the studied biological behavior. In addition, uncertainty involved in choosing the proper thresholds when imputing discrete observations from the continuous gene expression values can introduce error. Figure 1 illustrates how pattern in the data – although perceptible – is obscured by noise. While frequent itemsets and the algorithms that generate them have been well studied, the difficulties that arise from noise have not been adequately addressed.

In general, the noise present in real applications undermines the ultimate goal of traditional frequent itemset algorithms: recovering itemsets that appear without error in a sufficient fraction of transactions. In fact, as we discuss below, when noise is present, classical frequent itemset algorithms discover multiple small fragments of the true itemset, but miss the true itemset itself. The problem is worse for the most interesting, longer itemsets as they are more vulnerable to noise.

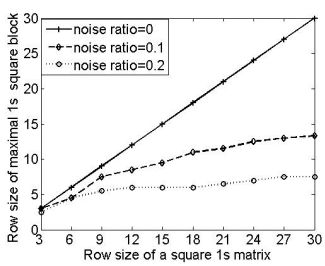


Figure 2: When noise is present, the observed size of the largest square sub-matrix of 1's increases far more slowly than the size of the initial square matrix of 1's. (Note: noise ratio refers to the value of p).

1.1 Fragmentation of Patterns by Noise

In order to analyze the potential effects of noise on frequent pattern mining, Sun and Nobel [17] considered a simple statistical model for the observed binary data matrix $\mathbf{Y}$. Formally,

$$\mathbf{Y} = \mathbf{X} \oplus \mathbf{Z}, \tag{1.1}$$

where $\mathbf{Y}$, $\mathbf{X}$ and $\mathbf{Z}$ are $m \times n$ binary matrices and $\oplus$ is the entry-wise exclusive-or operation (modulo 2 sum). The matrix $\mathbf{X}$ contains the unobserved "true" data values of interest, in the absence of noise, and $\mathbf{Z}$ is a binary noise matrix whose entries $z_{i,j}$ are independent Bernoulli random variables with $P(z_{i,j} = 1) = p = 1 - P(z_{i,j} = 0)$ for some $p \in (0, 1/2)$. In this case we will write $\mathbf{Z} \sim \text{Bern}(p)$. An example is shown in Figure 1. The statistical model (1.1) is equivalent to the standard communication model, widely studied in information theory, in which the values of $\mathbf{X}$ are observed after being passed through a binary symmetric channel. It is the binary version of the standard additive noise model in statistics inference.

Suppose for the moment that $m = n$, and let $M(\mathbf{Y})$ be the largest k such that $\mathbf{Y}$ contains a $k \times k$ submatrix of 1s, or equivalently, the largest k such that $\mathbf{Y}$ contains k transactions having k common items. The following proposition is proposed in [17]. It extends the earlier result on the clique number of random graphs by Bollobás *et.al* [4, 5] to binary random matrices.

Proposition 1. *With probability 1, $M(\mathbf{Y}) \leq 2\log_a n - 2\log_a \log_a n$ when n is sufficiently large, regardless of the structure of $\mathbf{X}$. Here $a = (1-p)^{-1}$.*

Proposition 1 shows that, even for small noise levels $p > 0$, large blocks of 1s or other structures in the true matrix $\mathbf{X}$ leave behind only fragments of logarithmic size in $\mathbf{Y}$. Thus no exact frequent itemset mining algorithm will be able to recover such underlying structure directly from $\mathbf{Y}$.

To demonstrate this effect, we added noise to a square matrix of 1s. Each entry of the initial matrix was changed to 0 with some probability p, independently from entry to entry. We applied standard frequent itemset mining to the corrupted matrix, and applied this process to matrices of different sizes. Figure 2 plots the size of the largest recovered square sub-matrix of 1s against the size of the original matrix, for different values of p (corresponding to different levels of data corruption). In the presence of noise, only a fraction of the initial block of 1s was recovered, and this fraction diminished with an increase in the size of the original matrix. Furthermore, the number of unique itemsets reported increased exponentially with both corruption level and original block size (see the spurious itemsets curve shown in Figure 10 in the experiment section).

The failure of classical frequent itemset mining to detect simple patterns in the presence of random errors compromises the ability of these algorithms to detect associations, cluster items, or build classifiers when such errors are present. Noise is ubiquitous in real data: it presents new challenges for algorithm development, and its consequences should not ignored. In this paper, we focus on noise-tolerant frequent itemset mining of the binary matrix representation of databases.

1.2 Approximate Frequent Itemset Models The formal setting of our problem is as follows. The available data take the form of an $n \times m$ binary matrix D. Each row of D corresponds to a transaction t and each column of D corresponds to an item i. The t, i-th element of D, denoted $D(t, i)$, is 1 if transaction t contains item i, and 0 otherwise. Let $T_0 = \{t_1, t_2, \ldots, t_n\}$ and $I_0 = \{i_1, i_2, \ldots, i_m\}$ be the set of transactions and items associated with D, respectively. An itemset is called *frequent*, if the fraction of transactions supporting it exceeds a given threshold, $minsup \in (0, 1]$.

One natural algorithmic approach for handling errors is to relax the requirement that a sub-matrix determined by the frequent itemset consists entirely of 1s, and allow it instead to contain a large fraction of 1s (and a small fraction of 0s), *e.g.*, the "presence" signal [6, 18]. This requirement is evidently a necessary condition, but it is not sufficient to define a sub-matrix of interest. To see why this is the case, consider the matrix shown in Figure 3.

The matrix in Figure 3 contains 3 sub-matrices $\{A, B, C\}$. The fraction of 1s in each sub-matrix is the same, namely 75%, however, the 1s are distributed quite differently in each. In sub-matrix A, each row and column contains 75% 1s, but in sub-matrix B and C the 1s are concentrated in the dense sub-matrix $B \cap C$. Both column g and row 7 are in a sense free riders on $B \cap C$. Clearly, neither sub-matrix B nor C should be used for association rule mining or classification purposes: in sub-matrix B row 7 does not support any item in the itemset, and in sub-matrix C item g is not supported by any transaction. It is possible to generate many more sub-matrices like B and C by combining any of the remaining columns and rows with $B \cap C$ to form a sub-matrices with densities of at least 75%.

Besides requiring a large fraction of 1s in a sub-matrix of interest, we advocate imposing two other conditions. First, for a given itemset, a *supporting transaction* should contain most of the items. Second, to be included in an itemset, an *associated item* has to appear in most of the supporting transactions. In the binary matrix representation, this means that the fraction of 0s in each row and each column of the sub-matrix representing the *approximate itemset* has to fall below a user-defined threshold. The threshold may differ for rows versus columns, and is denoted by ϵ_r and ϵ_c, respectively. If the approximate itemset has sufficiently many rows, it is judged to be an *approximate frequent itemset* (AFI).

Definition 1.1. *Let D be as above, and let $\epsilon_r, \epsilon_c \in [0, 1]$. An itemset $I \subseteq I_0$ is an approximate frequent itemset AFI(ϵ_r, ϵ_c), if there exists a set of transactions $T \subseteq T_0$ with $|T| \geq |T_0| \cdot minsup$ such that the following two conditions hold:*

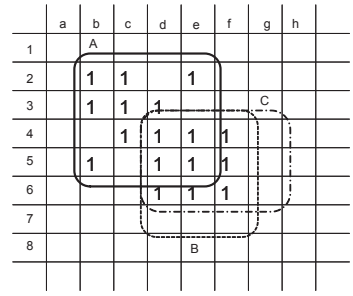


Figure 3: A binary matrix with three weak AFI(0.25) They can be more specifically classified as, A: AFI(0.25, 0.25); B: AFI(*, 0.25); C: AFI(0.25, *).

$$1.\ \forall i \in T, \frac{1}{|I|} \sum_{j \in I} D(i, j) \geq (1 - \epsilon_r);$$

$$2.\ \forall j \in I, \frac{1}{|T|} \sum_{i \in T} D(i, j) \geq (1 - \epsilon_c);$$

Let AFI(ϵ_r, ϵ_c) denote the collection of all AFI sub-matrices of D. Classical or exact frequent itemsets (EFI) are a special case of AFI, where both noise thresholds ϵ_r and ϵ_c are set to zero. In cases where the noise in either the rows or the columns is not restricted, AFI($\epsilon_r, *$) or AFI($*, \epsilon_c$) is used to denote the corresponding families. The noise threshold replaced by "$*$" means that no constraint is employed for the corresponding parameter, or the noise threshold is 1, i.e., $\epsilon = 1$. We also define the sub-matrices that satisfy the global noise constraint as *weak* AFIs in Definition 1.2.

Definition 1.2. *Let D be as above, and let $\epsilon \in [0, 1]$. An itemset $I \subseteq I_0$ is a weak AFI(ϵ) if there exists a set of transactions $T \subseteq T_0$ with $|T| \geq |T_0| \cdot minsup$ such that the following condition holds:*

$$\frac{1}{|I||T|} \sum_{i \in T} \sum_{j \in I} D(i, j) \geq 1 - \epsilon \tag{1.2}$$

According to our definition, the three sub-matrices in Figure 3 are weak AFIs. However, only sub-matrix A constitutes a valid AFI$(0.25, 0.25)$. B and C do not satisfy the constraints of AFI$(0.25, 0.25)$, but they are valid AFI$(*, 0.25)$ and AFI$(0.25, *)$ respectively.

Note that an AFI(ϵ_r, ϵ_c) also qualifies as both an AFI($\epsilon_r, *$) and an AFI($*, \epsilon_c$). The relationships among the various criteria are summarized in the Venn diagram of Figure 4. The differences in the sizes of the families and the maximum lengths of itemsets contained in each leads to substantial differences in the computational

costs for the algorithms that search for them. This will be further elaborated upon in the experimental sections.

In this work we proceed from the premise that, while the exact frequent itemset criterion is too restrictive, simple application of the weak AFI, AFI($\epsilon_r, *$) and AFI($*, \epsilon_c$) criteria allows poor approximations to frequent itemsets.

Yang *et.al.* [18] have developed models equivalent to the weak AFI and AFI($\epsilon_r, *$), but use the terms weak ETI and strong ETI, respectively, instead. For ease in comparing the competing criteria, we adopt their terminology for the remainder of this paper.

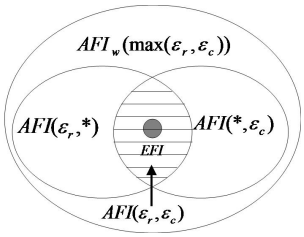


Figure 4: Relationships of various AFI criteria.

1.3 Challenges and Contributions Accommodating the refined noise criteria creates substantial algorithmic challenges not posed by exact frequent itemset mining. First and foremost, the AFI criterion distinguishes itself from traditional exact frequent itemsets as it violates the anti-monotone (Apriori) property. An exact itemset cannot be frequent if any of its sub-itemsets fails to be frequent. However, a sub-itemset of an AFI need not be an AFI. For example, given $minsup = 4$ and $\epsilon_r = \epsilon_c = 25\%$, sub-matrix A in Figure 3 is a valid AFI, but none of its sub-itemsets have sufficient support to be an AFI. The *minsup* can no longer be employed as a pruning threshold for itemset. No accurate pruning threshold has ever been found in any of existing work on noise-tolerant itemset mining. As a result, algorithms, such as ETI mining, have to rely on heuristics to prune the search space. These heuristics do not guarantee the completeness of the search. Another algorithm to discover dense itemsets [13] enforces the constraint that all sub-itemsets of a dense itemset must be frequent. Since this algorithm requires minsup support for all sub-itemsets, it can fail to identify larger itemsets that have sufficient support.

Noise-tolerance also affects the way in which supporting transactions are maintained in the algorithm. With exact frequent itemset mining, a transaction supporting an itemset also supports its sub-itemsets. This property is fundamental to any depth-first approach. This property, however, does not hold for AFI: one cannot derive the support set of an AFI from the common support sets of its sub-patterns, as is done in exact frequent itemset mining. (Examination of sub-matrix A in Figure 3 makes this clear.) To solve this problem, the algorithms proposed in[18, 13] require repeated scans of the entire database to identify the support for each itemset. The exponential number of potential itemsets makes this very expensive.

In this paper, we investigate the noise-tolerant property of approximate frequent itemsets that provides both the algorithmic basis for itemset generation, and the potential for pruning based on an AFI's support. The property is a generalization of Apriori under noisy conditions and includes the Apriori property as a special case when noise is absent. By incorporating noise-tolerant attributes, we designed an efficient and effective approach for mining the complete set of approximate frequent itemsets.

1.4 Outline The rest of the paper is organized as follows. Section 2 outlines related work in the area of noise-tolerant itemset mining. Section 3 contains a theoretical analysis showing how the AFI criterion can be used to recover block structures in the presence of noise, a problem for which standard frequent pattern mining fails. Section 4 presents the algorithm and two pruning strategies. Assessment of the AFI algorithm on synthetic and real data sets and an examination of its scalability are presented in Section 5. Section 6 concludes the paper.

2 Background and Related Work

In the standard frequent itemset problem [1, 2], the goal is to enumerate all the frequent itemsets in D; there is no allowance for noise. This corresponds to our AFI definition when $\epsilon_r = \epsilon_c = 0$.

Noise-tolerant itemsets were first discussed by Yang *et al.* [18], who proposed two error tolerant models, termed weak error-tolerant itemsets (ETI)(equivalent to weak AFI) and strong ETI (equivalent to AFI($\epsilon, *$)). As noted in the discussion of Figure 3, the ETI models do not preclude columns of zeros. Although this problem is identified by Yang *et al.* [18], it is not resolved in their paper. In addition, without an efficient pruning technique the authors had to employ a variety of heuristics and sampling techniques instead. In [13] the authors seek weak ETIs by constraining the subsets of ETIs to also be weak ETIs. This constraint may not only miss valid itemsets of interest, but also generates irrelevant itemsets, such as cluster (B) in Figure 3.

Other lines of work to find itemsets tolerating noise are [15, 12]. These approaches admit only a fixed number of 0s in the itemsets. In contrast to our AFI model, the fraction of noise can not vary with the size

of a submatrix defining an itemset, and therefore, is not guaranteed to be bounded relative to the size of the result. The support envelope technique [15] identifies regions of the data matrix where each transaction contains at least a given number of items and each item appears in at least a given number of transactions. The support envelope is a tool for exploring and visualizing high-level itemset structures in a data matrix. The paper defines a symmetric error tolerant itemset model (Symmetric ETI). It is similar to AFI but restricts the fraction of errors allowed to be the same for columns and rows. Also, no additional properties or algorithms for the symmetric ETI are developed by the authors.

3 Recovery of Block Structures in Noise

In this section we present some theoretical support for the AFI model in the context of a simple recovery problem for matrices with noise. Proposition 1 of Section 1.1 shows that exact frequent itemset mining cannot directly recover blocks of 1s and other structures in the presence of noise. The weak ETI (weak AFI), ETI (AFI(ϵ_r, *)) and AFI model address this problem by allowing zeros in their target sub-matrices. One means of validating and comparing these criteria is to see if they are able to recover simple structures in cases where exact frequent pattern mining fails. To this end, we show how the AFI model can be applied to the simple problem of recovering a sub-matrix of 1s set against a background of zeros when noise is present. (A complete analysis can be found in [17]). For simplicity, we only consider square matrices and sub-matrices. However, analogous results hold for rectangular matrices and sub-matrices.

Let $\mathbf{X}$ be an $n \times n$ binary matrix that consists of an $l \times l$ sub-matrix C^* of 1s, with all other entries equal to 0. (Note that the rows and columns of C^* need not be contiguous.) Suppose that we observe $\mathbf{Y} = \mathbf{X} \oplus \mathbf{Z}$, where $\mathbf{Z} \sim \text{Bern}(p)$, with $0 < p < 1/2$, and wish to accurately recover C^*. Let p_0 be any number such that $p < p_0 < 1/2$, and let $\tau = 1 - p_0$ be an associated error threshold. If C is a sub-matrix of $\mathbf{X}$, let $C \in \text{AFI}_\tau(\mathbf{X})$ denote the fact that every row and column of C has at least $100\,\tau\%$ 1s.

In order to recover C^*, we identify the largest square AFI in the observed matrix $\mathbf{Y}$ having an error threshold τ. More precisely, let $\mathcal{C}$ be the family of all square sub-matrices C of $\mathbf{X}$ such that $C \in \text{AFI}_\tau(\mathbf{X})$, and define

$$\hat{C} \;=\; \text{argmax}_{C \in \mathcal{C}} |C|$$

to be any maximal sized sub-matrix in $\mathcal{C}$. Note that $\hat{C}$ depends only on the observed matrix $\mathbf{Y}$. Let

$$\Lambda \;=\; |\hat{C} \cap C^*| / |\hat{C} \cup C^*|$$

measure the overlap between the estimated index set $\hat{C}$ and the true index set C^*. Then $0 \le \Lambda \le 1$, and values of Λ close to one indicate better overlap. A sketch of proof of the following theorem can be found in the appendix.

THEOREM 1. *Let $\hat{C}$ be the estimate of C^* based on the family $AFI_\tau(\mathbf{X})$ as described above. Let $\delta = p - p_0 > 0$. When n is sufficiently large, for any $0 < \alpha < 1$ and l satisfying $l > 16\alpha^{-1}(\log_b n + 2)$,*

$$P\left(\Lambda \le \frac{1-\alpha}{1+\alpha}\right) \;\le\; \Delta_1(l) + \Delta_2(\alpha, l). \tag{3.3}$$

Here $\Delta_1(l) = 2e^{-3\,\delta^2\,l/8p}$, $\Delta_2(\alpha, l) = 2n^{-\frac{1}{4}\alpha l + 4\log_b n}$, and the log base $b = \exp\{3(1-2\,p_0)^2/8p\}$.

The following is an example illustrating Theorem 1. Let X be a $n \times n$ binary matrix with $n = 800$ and let C^* be a $l \times l$ submatrix of X with $l = 400$. Suppose the noise level $p = 0.1$ and suppose the user specified noise level $p_0 = 0.15$. When $\alpha = \frac{1}{4}$, since $l > 16\alpha^{-1}(\log_b n + 2) = 360.1$, it follows Theorem 1 that $P\left(\Lambda \le \frac{3}{5}\right) \le 2(e^{-3.75} + 800^{-10.448}) = 0.047$, i.e. the probability that the overlap of the recovered AFI and C* will be less than 0.6 is small (less than 5%).

The conditions of Theorem 1 require that the noise level $p < 1/2$ and that the user-specified parameter p_0 satisfy $p < p_0 < 1/2$. Thus, in advance, one only needs to know an upper bound on the noise level p. A similar recovery result can be established for the weak ETI model. However, the proof is considerably more complicated, and more importantly, the recovery method requires exact knowledge of the noise level p. It appears that the same restriction is necessary for recovery with the ETI model as well. In the context of the simple recovery problem, the two-way restriction of the AFI model has direct advantages over the weak-ETI model.

Here we illustrate the essential ideas behind the proof of Theorem 1. Note that the entries of $\mathbf{Y}$ in C^* are i.i.d. Bernoulli$(1-p)$ random variables. Consequently, the sum of each row and each column of C^* has a Binomial$(l, 1-p)$ distribution. Using this fact and the condition that $1 - p_0 < 1 - p$, it can be shown that the probability that any row or column of C^* has average density less than $1 - p_0$ is very small. This implies that $C^* \in \text{AFI}_\tau(X)$ with high probability. Since $\hat{C}$ is the maximal sized sub-matrix in $\text{AFI}_\tau(X)$, it follows that $|\hat{C}|$ is greater than or equal to $|C^*|$ with high probability. Now, we want to show that $\hat{C}$ can not be too large either, and that it can only contain a small proportion of entries outside C^*. When $\hat{C}$ is much larger than C^*, it must contain a large number of rows (or columns) whose

entries are from outside C^*. The definition of $\hat{C}$ via the AFI criterion implies that each such row (column) has density greater than τ. Moreover, the rows (columns) will necessarily contain a large rectangular region with entries from outside C^*, and this region should also have density greater than τ. But as the entries of $\mathbf{Y}$ outside C^* are i.i.d. Bernoulli(p), the probability of finding a rectangular region as above is very small.

Theorem 1 can readily be applied to the asymptotic recovery of structure in a sequential framework. Suppose that $\{\mathbf{X}_n : n \geq 1\}$ is a sequence of square binary matrices, where $\mathbf{X}_n$ is $n \times n$ and consists of an $l_n \times l_n$ sub-matrix C_n^* of 1s with all other entries equal to 0. For each n we observe $\mathbf{Y}_n = \mathbf{X}_n \oplus \mathbf{Z}_n$, where $Z_n \sim$Bern(p), and wish to recover C_n^*. Let Λ_n measure the overlap between C_n^* and the estimate $\hat{C}_n$ produced by the AFI recovery method above. The following corollary of Theorem 1 shows that, under suitable conditions on l_n, $\hat{C}_n$ provides asymptotically consistent estimates of C_n^*. The proof can be found in the appendix.

Corollary 1. *If $l_n > 16\psi(n)(\log_b n + 2)$ where $\psi(n) \to \infty$ as $n \to \infty$, then with probability one*

$$\Lambda_n \leq \frac{1 - \psi(n)^{-1}}{1 + \psi(n)^{-1}}$$

when n is sufficiently large.

4 AFI Mining Algorithm

Mining approximate frequent itemsets poses a number of new algorithmic challenges beyond those faced when mining exact itemsets. The foremost difficulty is that noise-tolerant itemset mining cannot employ the anti-monotone property that has led to the success of frequent itemset mining. The development of an efficient algorithm for finding AFIs calls for new itemset generation strategies to limit the search space. We present a noise-tolerant Apriori property in Section 4.1.1. In addition, the AFI criteria allow the number of errors to increase with the size of the itemset. It is therefore critical to take account of the additional errors in an itemset as its dimensionality increases while collecting the supporting transactions. Solving this problem is the key to AFI mining, and is addressed in subsection 4.1.2. The AFI mining algorithm adapts the methods of level-wise breadth-first frequent itemset mining to this new setting, and takes advantage of our new techniques to generate noise-tolerant approximate frequent itemsets.

4.1 Mining AFIs The algorithm's enumeration of the AFI differs from the existing work of weak ETI algorithm[18] in the following aspects: First, even though the Apriori property doesn't hold for any type of AFI(except those that allow no noise), we have developed a noise-tolerant Apriori property (Theorem 4.1) and apply it to prune and generate candidate itemsets. Secondly, by taking different approaches in extending the itemsets, we are able to collect the support of an noise-tolerant itemset based on the support set in the sub-itemsets.

4.1.1 Noise-Tolerant Support Pruning The anti-monotone property of exact frequent itemsets is the key to minimizing exponential searches in frequent itemset mining. In particular, the anti-monotone property ensures that a $(k+1)$ exact itemset can be pruned if any one of its k sub-itemsets is not frequent. However, this property is no longer true for any variation of AFI. Instead, in this paper, we derive a noise-tolerant support to serve as the Apriori pruning threshold. The noise-tolerant support is determined by the size of the itemset and the noise thresholds. This support threshold leads to substantial performance gain for our algorithm.

Theorem 4.1. *Given a support threshold minsup, if a length $(k+1)$-itemset I' is an AFI(ϵ_r, ϵ_c), then for any of its k item subset $I \subseteq I'$, the number of transactions containing no more than ϵ_r fraction of noise in I is at least*

$$n \cdot minsup \cdot \left(-\frac{k\epsilon_c}{\lfloor k\epsilon_r \rfloor + 1}\right) \tag{4.4}$$

Proof: By assumption, there exists a set of transactions T' such that $|T'| \geq n \cdot minsup$ and $T' \times I' \in$ AFI(ϵ_r, ϵ_c) Let I be a k item subset of I' with support set T. Thus each $t \in T$ contains at most $k\epsilon_r$ zeros on I.

Let $num_0(C)$ be a function that returns the number of 0s in any submatrix C of D. Since the transactions in $T' \setminus T$ do not support I, each such transaction contains more than $k\epsilon_r$ zeros on I. It follows that

$$\begin{aligned} num_0((T' \setminus T) \times I) &\geq |T' \setminus T| \cdot (\lfloor k\epsilon_r \rfloor + 1) \\ &\geq (|T'| - |T|) \cdot (\lfloor k\epsilon_r \rfloor + 1) \end{aligned}$$

As $T' \times I'$ is an AFI, each item in I contains at most $\epsilon_c|T'|$ zeros on T'. Therefore,

$$num_0(T' \times I) \leq k \cdot |T'| \cdot \epsilon_c. \tag{4.5}$$

Combining the last two inequalities gives

$$\begin{aligned} (|T'| - |T|) &\cdot (\lfloor k\epsilon_r \rfloor + 1) \\ &\leq num_0((T' \setminus T) \times I) \\ &\leq num_0(T' \times I) \\ &\leq k \cdot |T'| \cdot \epsilon_c \end{aligned} \tag{4.6}$$

where the second inequality follows from the fact that $T' \setminus T \subseteq T'$. Expressing the last inequality in terms of $|T|$ yields

$$|T| \geq |T'| \left(1 - \frac{k\epsilon_c}{\lfloor k\epsilon_r \rfloor + 1}\right)$$

$$\geq n \cdot minsup \cdot \left(1 - \frac{k\epsilon_c}{\lfloor k\epsilon_r \rfloor + 1}\right) \tag{4.7}$$

Based on the bound of Theorem 4.1 we make the following definition.

DEFINITION 4.1. *Given ϵ_c, ϵ_r and minsup, the* **noise-tolerant pruning support** *for a length-k itemset is,*

$$minsup^k = minsup \cdot \left(1 - \frac{k\epsilon_c}{\lfloor k\epsilon_r \rfloor + 1}\right)_+ \tag{4.8}$$

Here $(a)_+ = \max\{a, 0\}$.

The noise-tolerant support threshold is used as the basis of a pruning strategy for AFI mining. The strategy removes supersets of a given AFI$(\epsilon_r, *)$ I from further consideration when the number of transactions which contain less than ϵ_r fraction of errors in I is less than $n \cdot minsup^k$. In the special case that $\epsilon_r = \epsilon_c = 0$, $minsup^k = minsup$, which is consistent with the anti-monotone property of exact frequent itemsets [1]. The support threshold decreases as ϵ_c increases and as ϵ_r decreases. In the former case, a less stringent column constraint is applied to a block with fixed row constraints, and conversely in the case of decreasing ϵ_r. In particular, the support threshold is equal to 0 when $k \cdot \epsilon_c > \lceil k \cdot \epsilon_r \rceil$. Therefore, no pruning can be applied at all.

4.1.2 0/1 Extensions Starting with singleton itemsets, the AFI algorithm generates $(k+1)$-itemsets from k-itemsets in a sequential fashion. The number of 0s allowed in the itemset grows with the length of the itemset in a discrete manner. If $\lfloor (k+1)\epsilon_r \rfloor > \lfloor k\epsilon_r \rfloor$, then transactions supporting the $(k+1)$-itemset are permitted one more zero than transactions supporting k-itemsets. When $\lfloor (k+1)\epsilon_r \rfloor = \lfloor k\epsilon_r \rfloor$, no additional zeros are allowed. For example, if $\epsilon_r = 0.25$, additional zeros are permitted in transactions when extending itemsets of length $3, 7, 11$ and so on. Whether the maximal number of zeros will increase in a $(k+1)$ itemset makes a difference in deriving its set of supporting transactions. Intuitively, if an additional zero is allowed at level $(k+1)$, any transaction supporting a k itemset should also support its $(k+1)$ superset. On the other hand, when the maximum number of zeros allowed in an itemset stay the same at level $(k+1)$, a transaction that does not support k itemset will not have enough 1s to support its $(k+1)$ superset. These two properties are formally addressed in Lemma 4.1 and Lemma 4.2 as 1-Extension and 0-Extension respectively.

LEMMA 4.1. *(1-Extension) If $\lfloor k \cdot \epsilon_r \rfloor = \lfloor (k+1) \cdot \epsilon_r \rfloor$ then any transaction that does not support a k-itemset will not support its $(k+1)$ item superset.*

The Lemma is based on the fact that if no additional noise is allowed when generating a $(k+1)$ itemset, a transaction that does not support a k-itemset will not support its $(k+1)$ superset since the number of 1s it contains is always smaller than or equal or $\frac{\lfloor k*\epsilon \rfloor - 1 + 1}{k+1} < \epsilon$. Thus if $\lfloor k \cdot \epsilon_r \rfloor = \lfloor (k+1) \cdot \epsilon_r \rfloor$ then the transaction set of a $(k+1)$ itemset I is the *intersection* of the transaction sets of its length k subsets. This is called a *1-extension.*

LEMMA 4.2. *(0-Extension) If $\lfloor k \cdot \epsilon_r \rfloor + 1 = \lfloor (k+1) \cdot \epsilon_r \rfloor$ then any transaction supporting a k-itemset also supports its (k+1) supersets.*

The procedure of 0-extension illustrates how noise can be incorporated into a frequent itemset. If additional noise is allowed for a $(k+1)$ itemset relative to a k itemset, it is intuitive that a transaction that supports a k-itemset will also support its $(k+1)$-item supersets, no matter whether the $(k+1)^{th}$ entry is 1 or 0. To utilize this property, if $\lfloor k \cdot \epsilon_r \rfloor + 1 = \lfloor (k+1) \cdot \epsilon_r \rfloor$, the transaction set of a $(k+1)$ itemset I is the *union* of the transaction sets of its length k subsets. This is called a *0-extension.*

0-extension and 1-extension suggest two basic steps to be taken for efficient maintenance of the supporting transactions. They allow the algorithm to obtain the support transactions of an itemset from its item subsets while avoiding the repeated scan of databases that plagues the algorithms proposed by [18, 13]. In the next section, we illustrate through an example the use of the two techniques together with noise-tolerant support-based pruning method.

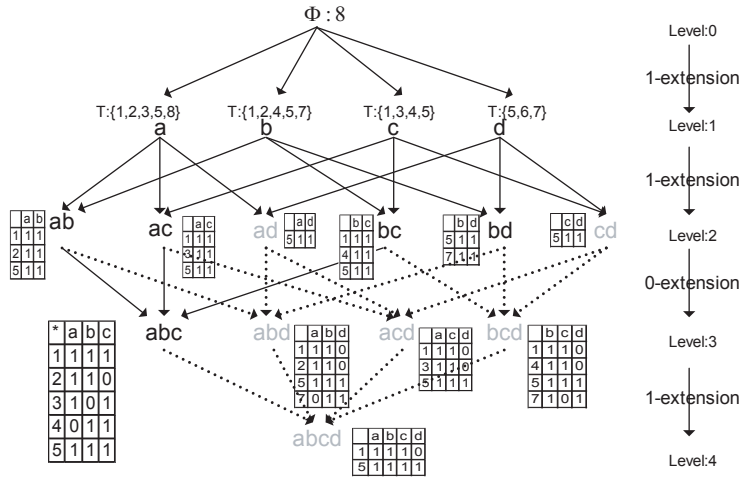
4.2 An Example In this section we present a simple example in which the data matrix D of Figure 5 is used to illustrate the AFI algorithm. Let $\epsilon_r = \epsilon_c = 1/3$ and let $minsup = 0.5$. The number of transactions in the database, n, equals 8. We wish to find the complete set of AFIs in D. In this case, the algorithm proceeds as follows.

Step 1: $k = 1, minsup^1 = 0.5$. The database is scanned once and the support of each singleton item is recorded.

Step 2: $k = 2, minsup^2 = 0.5 * 1/3$. As $\lfloor k \cdot \epsilon_r \rfloor = \lfloor (k-1) \cdot \epsilon_r \rfloor$, no additional 0's are allowed and a 1-extension is performed. In particular, the transaction

	a	b	c	d
1	1	1	1	0
2	1	1	0	0
3	1	0	1	0
4	0	1	1	0
5	1	1	1	1
6	0	0	0	1
7	0	1	0	1
8	1	0	0	0

(A)



(B)

Figure 5: (A) Sample database; (B) Level wise mining of AFI in database (A). See Section 4.2 for more details. Only black colored itemsets will be generated by AFI, while every itemset including the grey-colored itemsets will have to be generated to mine ETIs.

set of the itemset ab is obtained by intersecting the transaction set of a, equal to $\{1,2,3,5,8\}$, with that of b, equal to $\{1,2,4,5,7\}$; the result is $\{1,2,5\}$. Since the number of transactions supporting ad and cd is equal to 1. Therefore, their supports are below the support threshold $minsup^2$, any AFI that contains them can be pruned. These itemsets are colored gray in Figure 5.

Step 3: $k = 3, minsup^3 = 0.5 * 2/3$. In this case, $\lfloor k \cdot \epsilon_r \rfloor = \lfloor (k-1) \cdot \epsilon_r) \rfloor + 1$. Thus one additional 0 is allowed in 3-itemsets, and a 0-extension (union of transaction sets) is performed. For example, a transaction supports itemset abc if it supports any of $\{ab, ac, bc\}$; the transaction set of abc is the union of the transaction sets for $\{ab, ac, bc\}$, which is $\{1,2,5\} \cup \{1,3,5\} \cup \{1,4,5\} = \{1,2,3,4,5\}$.

Step 4: $k = 4, minsup^4 = 0.5 * 1/3$. Because of support constraint $minsup$, i.e, 0.5, $\{a,b,c,d\}$ cannot be a valid AFI. No further extension of the current itemset is possible since all of the search space is covered.

Step 5: The candidate AFIs are $\{b,d\}$ and $\{a,b,c\}$. The first does not satisfy the $minsup$ size constraint. The second is readily shown to be a valid AFI, and constitutes the output of the algorithm.

4.3 Global Pruning In order for an individual item i to appear in an AFI, its overall support must exceed $minsup \cdot n \cdot (1-\epsilon_c)$. During the level-wise generation of AFI$(\epsilon_r, *)$, the total number of transactions under consideration in a given level will decrease or remain the same, and the number of transactions supporting an individual item will have the same property. If the support of item i among the transactions at level k drops below $minsup \cdot n \cdot (1-\epsilon_c)$, then i can not appear in any AFI generated at levels $k' \geq k$. In particular, any itemset containing i can be eliminated from consideration. To illustrate, in the example presented in Figure 5 the number of transactions supporting an itemset should not be below 4. In addition, the number of supporting transactions for an individual item has to be above $\lceil 4 \cdot (1-\epsilon_c) \rceil = \lceil 4(1-1/3) \rceil = 3$. The set of transactions remaining at level 2 is $T = \{1,2,3,5,7\}$; the number of transactions in T supporting item d is 2, which is less

Algorithm 1
Input: D, ϵ_r, ϵ_c, $minsup$
Output: The sets of approximate frequent itemsets

```
for i = 1 : m
        T(i)=genSupport(D, i);
k = 1;
L_1=∪_{i>0}^{m}{i};
repeat
    k := k+1;
    L_k := GenCandidateItemset(L_{k-1}, minsup^{k-1})
    if (⌊k · ε_r⌋ = ⌊(k+1) * ε_r⌋)
            T(L_k) := 1-Extension(I, L_{k-1});
      else
            T(L_k) := 0-Extension(I, L_{k-1});
    AFI_p := AFI_p ∪ L_k;
until L_k is ∅
AFI :=filter(AFI_p, minsup, ε_c)
return AFI
```

Figure 6: AFI mining algorithm. L_k represents the set of itemsets at level k; $T(I)$ returns the support set of I.

than 3, so any itemset in level $k \geq 2$ containing d can be eliminated from consideration.

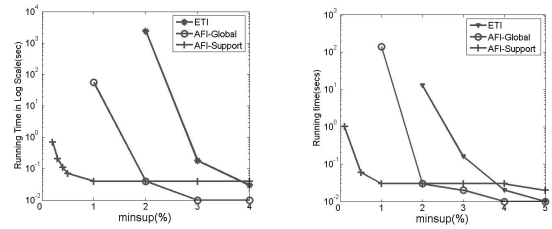
4.4 Identification of AFI The AFI algorithm so far generates a superset of approximate frequent itemsets. The postprocessing of this subset can be done separately from the level-wise generation since it will neither benefit nor prohibit the traversing of the search space. The verification of whether an AFI($\epsilon_r, *$) is an AFI can be easily done by simply checking the percentage of 0's in each candidate itemset. Finding a maximal AFI in an AFI($\epsilon_r, *$) is more difficult. In the technical report [9], we describe a heuristic algorithm for this problem that scales linearly with respect to $|T| + |I|$, where I is an itemset supported by a transaction set T. The algorithm works by removing transactions having a large number of zeros, beginning with those whose zeros are aligned with low density items. Due to space limitations, a complete description of this algorithm is omitted.

5 Experiments

We performed four experiments to assess the performance of AFI. The first explored the scalability of the AFI mining algorithm and the effectiveness of the pruning methods. The second experiment used synthetic data to compare the results of AFI mining to exact frequent itemset mining and ETI. Finally, we applied AFI to a zoology data set with known underlying patterns.

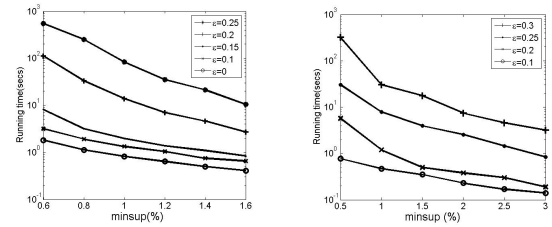
5.1 Scalability Two data sets were employed to measure scalability. The first, T10KI100, was generated by the IBM synthetic data set generator. It contains 10K transactions and 100 items, with an average of 10 items per transaction. The second data set was the chess data set, which is available from the UCI machine learning repository[20]. It contains 28K transactions and 65 items with at least one third nonzero elements per transaction. We built the exhaustive level-wise algorithm presented in [18] to discover the complete set of strong ETIs. The experiments were run on a 2GHz PC with 2G memory.

Figure 7 presents the run-time performance for both data sets, with $\epsilon_r = \epsilon_c = 20\%$. All algorithms performed well when *minsup* was 5% or higher; however, ETI was not able to compete when *minsup* dropped below 2%. In contrast to AFI, ETI lacks an effective pruning strategy; therefore, a much larger set of candidate itemsets may be generated in order to build the complete set of ETIs. In addition, because the noise criterion of ETI is less stringent, the maximum length of an ETI can be much larger than that of an AFI. This leads to an exponentially larger number of candidate itemsets. Both shortfalls explain why AFI can outperform ETI by such a large margin. AFI mining with downward pruning appears to be superior to global support pruning, especially when the minimum support is low. The AFI algorithm employing both pruning strategies was also tested, although not shown in the Figure; the performance was almost the same as AFI using only the support pruning property.



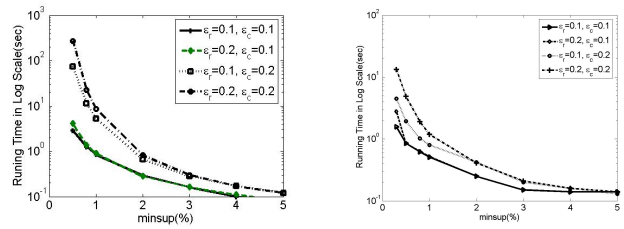
(A) T10KI100 (B)Chess

Figure 7: Comparison between AFI and ETI



(A) T10KI100 (B)Chess

Figure 8: The running time of AFI with noise-tolerant support pruning varying *minsup* and ϵ. $\epsilon = \epsilon_c = \epsilon_r$.



(A) T10KI100 (B)Chess

Figure 9: The running time of AFI with noise-tolerant support pruning as *minsup* varies. $\epsilon_c \neq \epsilon_r$.

We tested the scalability of our algorithm as the noise threshold and minimum support varied. The result is shown in Figure 8. To reduce the parameter space, the transaction-wise threshold ϵ_r was set equal to the item-wise noise threshold ϵ_c in this set of experiments. Figure 8 shows that running time increases with increases in noise tolerance, as expected. Here the al-

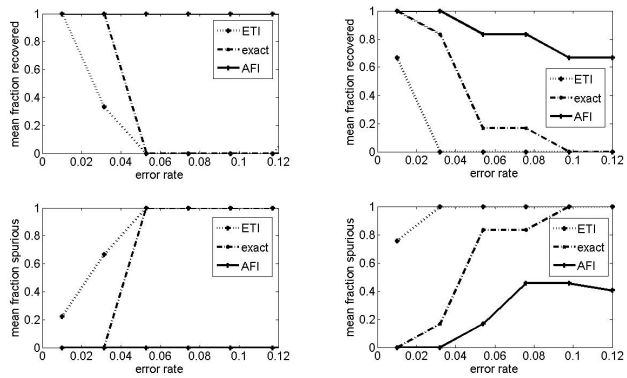
gorithm is essentially looking for approximate frequent itemsets with higher-dimensionality (detailed results are in [10]). Allowing more noise in an itemset results in larger approximate frequent itemsets; consequently, more candidate itemsets have to be explored, and computation increases exponentially with respect to the dimensionality of the itemsets. Nevertheless, even with a very high error rate of 30%, our algorithm proves competent in finding the complete sets of AFI in a reasonable time.

Figure 9 shows how different transaction-wise and item-wise noise thresholds can affect performance. Relatively speaking, reducing the item-wise error constraint leads to a greater reduction in running time than reducing the transaction-wise error constraint, as the former leads to higher levels of pruning according to Theorem 4.1.

5.2 Quality Testing with Synthetic Data In addition to run-time performance we also tested the quality of the results produced by AFI. To do so we created data with an embedded pattern and then overlaid random errors. By knowing the true patterns, we were able to assess the quality of the various results. To each synthetic data set created, an exact method (ETI with $\epsilon_c = 0$), ETI and AFI were each applied.

To evaluate the performance of an algorithm on a given data set, we employed two measures that jointly describe quality: "recoverability" and "spuriousness." Recoverability is the fraction of the embedded patterns recovered by an algorithm, while spuriousness is the fraction of the mined results that fail to correspond to any planted cluster. A truly useful data mining algorithm should achieve high recoverability with little spuriousness to dilute the results. A detailed description of the two measures is given in [9]. Multiple data sets were created and analyzed to explore the relationship between increasing noise levels and the quality of the result. Noise was introduced by bit-flipping each entry of the full matrix with a probability equal to p. The probability p was varied over different runs from 0.01 to 0.2. The number of pattern blocks embedded also varied, but the results were consistent across this parameter. Here we present results when 1 or 3 blocks were embedded in the data matrix (Figure 10(A) and (B), respectively).

In both cases, the exact method performed poorly as noise increased. Beyond $p = 0.05$ the original pattern could not be recovered, and all of the discovered patterns were spurious. In contrast, the error-tolerant algorithms, ETI and AFI, were much better at recovering the embedded matrices at the higher error rates. However, the ETI algorithm reported many more spurious results than AFI. Although it may discover the embedded patterns, ETI also reports many additional patterns that are not of interest, often including irrelevant columns. The AFI algorithm consistently demonstrated higher recoverability of the embedded pattern while maintaining a lower level of spuriousness.



(A) Single Cluster (B) Multiple Clusters

Figure 10: Algorithm quality versus noise level.

5.3 Zoo Data Set We also applied AFI to a database downloaded from the UCI Machine Learning Repository[20]. The Zoo Database contains 101 instances and 18 attributes (animal name, 15 boolean attributes, 2 numerics). The boolean attributes are *hair*, *feathers*, *eggs*, *milk*, *airborne*, *aquatic*, *predator*, *toothed*, *backbone*, *breathes*, *venomous*, *fins*, *tail*, *domestic* and *catsize*. The numeric attributes are *legs* and *type*, where the *type* attribute appears to be the class attribute. All the instances are classified into 7 classes (*mammals, birds, fish*, etc.).

One task could be to discover the common features of a set of animals in the same class. For example, mammals produce milk, are covered in hair, are toothed, and grow tails. However, not every mammal exhibits these common features: platypuses lack teeth and dolphins are hairless. If such exceptions are not tolerated, it is hard to find the complete set of features that characterizes a class.

For testing purposes, we adopted the 7 classes into which the instances were already categorized as the true underlying pattern. Then we examined how well the competing frequent itemset mining methods recovered these classes. We focused on the 4 classes with at least 5 instances and where each class had least 3 commonly shared features.

The exact method, ETI(ϵ_r), and AFI(ϵ_r, ϵ_c) were each applied to the dataset. When we required a perfect

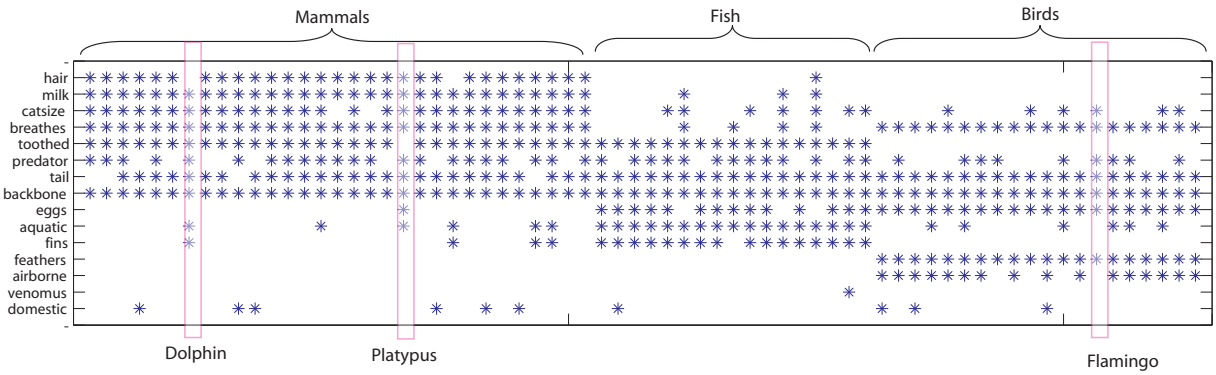


Figure 11: Three AFI blocks discovered in the zoo dataset. * indicates the presence of a feature.

match between the output of a method and the true pattern, only AFI was able to recover 3 out of the 4 classes. Here "perfect match" refers to a step in the evaluation of the output, not the criteria for adding a transaction to the support of an itemset. When the criteria for a match was relaxed to 85% overlap, then AFI recovered the fourth class: bugs. Figure 11 displays the sets of animals and their common features identified by AFI.

Neither the exact method nor ETI were able to recover a single class under the perfect match evaluation criterion. Exact frequent itemset mining generated subsets of the animals in each class and then found subsets of their common features. The instance *flamingo* presented a typical problem: in this data set *flamingo* lacks the *airborne* attribute – perhaps because the zoo clipped their wings. Thus flamingo cannot be included in the class *bird* with the common feature *airborne*.

Although such "errors" as clipped wings are accommodated by ETI, sometimes the type of tolerance featured by ETI identified irrelevant items. It identified *fin* and *domestic* as common features for *mammals*, which is not generally true. Because only the row-wise constraint was applied, the set of features discovered was not reliable.

6 Conclusion

In this paper we have outlined an algorithm for mining approximate frequent itemsets from noisy data. The AFI model places two criteria on the fraction of noise in both the rows and columns, and so ensures a relatively reasonable distribution of the error in any patterns found. Our work generalizes the classical level-wise frequent itemset mining based on the Apriori-property into a new algorithm where the Apriori-property does not hold and noise has to be incorporated. Our work generates not only more reasonable and useful itemsets than classical frequent itemset mining and existing noise-tolerant frequent itemset mining, but it is computationally more efficient as well. We are currently investigating depth-first methods for approximate frequent itemset mining.

References

[1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases In SIGMOD 1993.

[2] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A Verkamo. Fast discovery of association rules. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors, Advances in Knowledge Discove and Data Mining, chapter 12, pages 307328. AAAI Pre 1996.

[3] C. Becquet , S. Blachon, B. Jeudy, J.F. Boulicaut, O. Gandrillon. Strong-association-rule mining for large-scale gene-expression data analysis: a case study on humaMining gene expression databases for association rules. n SAGE data. Genome Biol. 2002

[4] B. Bollobás. Random Graphs second edition. Cambridge Studies in Advanced Mathematics, 2001.

[5] B. Bollobás, P. Erdős. Cliques in random graphs. Mathematical Proceedings of the Cambridge Philosophical Society, 80:419C427, 1976.

[6] D. Chakrabarti, S. Papadimitriou, D. Modha and C. Faloutsos, Fully Automatic Cross-Assocations. KDD 2004.

[7] C. Creighton, S. Hanash. Mining gene expression databases for association rules. Bioinformatics. 2003 Jan;19(1):79-86.

[8] C. Gao , Anthony K. H. Tung, X. Xin, P. Feng , J. Yang. "FARMER: Finding Interesting Rule Groups in Microarray Datasets".

[9] J. Liu, S. Paulsen, W.Wang, A. Nobel, J. Prins. Mining Approximate Frequent Itemset from Noisy Data. Technical Report(TR05-015) of Department of Compuater Science, UNC-Chapel Hill, 2005 Jun.

[10] J. Liu, S. Paulsen, W. Wang, A. Nobel, J. Prins. "Mining Approximate Frequent Itemset from Noisy Data", In ICDM 2005.
[11] J. Pei, A. K. Tung, and J. Han. Fault-tolerant frequent pattern mining: Problems and challenges. In Workshop on Research Issues in Data Mining and Knowledge Discovery, 2001.
[12] J. Pei, G. Dong, W. Zou, J. Han. Mining Condensed Frequent-Pattern Bases. Knowledge and Information Systems, Volume 6, Issue 5. 2002.
[13] J. K. Seppanen, H. Mannila. Dense Itemsets. In SIGKDD 2004.
[14] S. Bernstein,,The theory of Probabilities. Gastehizdat Publishing House. Moscow 1946
[15] M. Steinbach, P. Tan, V. Kumar. Support envelopes: a technique for exploring the structure of association patterns. In SIGKDD 2004.
[16] P. J. Unmack. Biogeography of Australian freshwater fishes. Journal of Biogeography. Vol. 28: pages 1053–1089. Blackwell Science Ltd. 2001.
[17] X. Sun, and A.B. Nobel, Significance and recovery of block structures in binary matrices with noise. 2005 Tech. Rep. Department of Statistics and Operation Research, UNC Chapel Hill.
[18] Cheng Yang, Usama Fayyad, Paul S. Bradley. Efficient discovery of error-tolerant frequent itemsets in high dimensions. In SIGKDD 2001.
[19] M. C. Yang,, Q. G. Ruan,, J. J Yang., S. Eckenrode, S. Wu, R. A. McIndoe, and J. X. She., A statistical method for .agging weak spots improves normalization and ratio estimates in microarrays, Physiol. Genomics, 7:4553, 2001.
[20] UCI machine learning repository. (http://www.ics.uci.edu/ mlearn/MLSummary.html).

7 Appendix

The detailed proofs of the following Lemma 1 and Lemma 2 can be found in [17]. We only state them here.

LEMMA 1. *Under the conditions of Theorem 1,*

$$P\left(|\hat{C}| \le l^2\right) \le \Delta_1(l). \tag{7.9}$$

LEMMA 2. *For any sufficiently large n, let $\mathcal{A} = \{C : C \in \mathcal{C}_n$ such that $|C| > \frac{l^2}{2}$ and $\frac{|C \cap C^{*c}|}{|C|} \ge \alpha\}$. Let $A = \{\mathcal{A} \ne \emptyset\}$. If $l \ge 16\alpha^{-1}(\log_b n + 2)$, then*

$$P(A) \le \Delta_2(\alpha, l) \tag{7.10}$$

Proof of Theorem 1: Let E be the event that $\{\Lambda \le \frac{1-\alpha}{1+\alpha}\}$. It is clear that E can be expressed as the union of two disjoint events E_1 and E_2, where

$$E_1 = \{|\hat{C}| \le |C^*|\} \cap E$$

and

$$E_2 = \{|\hat{C}| > |C^*|\} \cap E$$

On the other hand, by the definition of Λ, inequality $\Lambda \le \frac{1-\alpha}{1+\alpha}$ can be rewritten equivalently as

$$1 + \frac{|\hat{C} \cap C^{*c}|}{|\hat{C} \cap C^*|} + \frac{|\hat{C}^c \cap C^*|}{|\hat{C} \cap C^*|} \ge \frac{1+\alpha}{1-\alpha}.$$

Moreover, when $|\hat{C}| > |C^*|$, one can verified the trivial fact that $|\hat{C} \cap C^{*c}| > |\hat{C}^c \cap C^*|$, which also implies that

$$1 + \frac{|\hat{C} \cap C^{*c}|}{|\hat{C} \cap C^*|} + \frac{|\hat{C}^c \cap C^*|}{|\hat{C} \cap C|} \le 1 + 2\frac{|\hat{C} \cap C^{*c}|}{|\hat{C} \cap C^*|}.$$

Furthermore, one can verified that $E_2 \subset E_2'$, where

$$E_2' = \{|\hat{C}| > |C^*|\} \cap \left\{1 + 2\frac{|\hat{C} \cap C^*|}{|\hat{C} \cap C^*|} \ge \frac{1+\alpha}{1-\alpha}\right\}.$$

Therefore, it suffices to bound $P(E)$ by $P(E_1)$ and $P(E_2')$ separately.

Immediately, one can bound $P(E_1)$ by $\Delta_1(l)$ via Lemma 1. It remains to bound $P(E_2')$. Notice that inequality

$$1 + 2\frac{|\hat{C} \cap C^{*c}|}{|\hat{C} \cap C^*|} \ge \frac{1+\alpha}{1-\alpha} \text{ implies } \frac{|\hat{C} \cap C^{*c}|}{|\hat{C}|} \ge \alpha.$$

Therefore, by Lemma 2, one can bound $P(E_2')$ by

$$P(E_2') \le P\left(E_2' \,\middle|\, |\hat{C}| \ge l^2\right) \le \Delta_2(\alpha, l),$$

where the first inequality holds because the unconditional probability is always less or equal to the conditional probability. Consequently, we have

$$P\left(\Lambda \le \frac{1-\alpha}{1+\alpha}\right) \le \Delta_1(l) + \Delta_2(\alpha, l). \tag{7.11}$$

Proof of Corollary 1: Theorem 1 implies that if we can bound both $\Delta_1(l_n)$ and $\Delta_2(\psi(n)^{-1}, l_n)$ by $2n^{-2}$ for any sufficiently large n, then Borel -Cantelli Lemma can be applied to establish the almost sure convergency.

When n is sufficiently large, the condition $l_n > 16\psi(n)(\log_b n + 2)$ and $\psi(n) \to n$, implies $l_n > 2(\frac{3}{4}(p - p_0)^2 \log_b e)^{-1} \log_b n$. By plugging this lower bound of l_n into $\Delta_1(l_n)$, one can get $\Delta_1(l_n) < 2n^{-2}$. Meanwhile, by plugging the condition that $l_n > 16\psi(n)(\log_b n + 2)$ into $\Delta_2(\psi(n)^{-1}, l_n)$, one can get $\Delta_2(\psi^{-1}(n), l_n) < 2n^{-2}$.