Effective and Efficient Itemset Pattern Summarization: Regression-based Approaches

Ruoming Jin, Muad Abu-Ata, Yang Xiang, Ning Ruan
Department of Computer Science, Kent State University
Kent, OH, 44242, USA
{jin,mabuata,yxiang,nruan}@cs.kent.edu

ABSTRACT

In this paper, we propose a set of novel regression-based approaches to effectively and efficiently summarize frequent itemset patterns. Specifically, we show that the problem of minimizing the restoration error for a set of itemsets based on a probabilistic model corresponds to a *non-linear regression problem*. We show that under certain conditions, we can transform the non-linear regression problem to a *linear regression problem*. We propose two new methods, k*-regression* and *tree-regression*, to partition the entire collection of frequent itemsets in order to minimize the restoration error. The K-regression approach, employing a K-means type clustering method, guarantees that the total restoration error achieves a local minimum. The tree-regression approach employs a decision-tree type of top-down partition process. In addition, we discuss alternatives to estimate the frequency for the collection of itemsets being covered by the k representative itemsets. The experimental evaluation on both real and synthetic datasets demonstrates that our approaches significantly improve the summarization performance in terms of both accuracy (restoration error), and computational cost.

Categories and Subject Descriptors: H.2.8 [Database Management]: Database Applications - Data Mining
General Terms: Algorithms, Performance
Keywords: frequency restoration, pattern summarization, regression

1. INTRODUCTION

Since its introduction in [3], frequent pattern mining has received a great deal of attention and quickly evolved into a major research subject in data mining. The tools offered by frequent pattern mining research span a variety of data types, including itemsets, sequences, trees, and graphs [25, 4, 31, 5]. Researchers from many scientific disciplines and business domains have demonstrated the benefits from frequent pattern analysis—insight into their data and knowledge of hidden mechanisms [10]. At the same time, frequent pattern mining serves as a basic tool for many other data mining tasks, including association rule mining, classification, clustering, and change detection [14, 32, 13, 15]. Recently, standard frequent mining tools, like Apriori, have been incorporated into several commercial database systems [18, 30, 23].

The growing popularity of frequent pattern mining, however, does not exempt it from criticism. One of the major issues facing frequent pattern mining is that it can (and often does) produce an unwieldy number of patterns. So-called complete frequent pattern mining algorithms try to identify all the patterns which occur more frequently than a minimal support threshold (θ) in the desired datasets. A typical complete pattern mining tool can easily discover tens of thousands, if not millions, of frequent patterns. Clearly, it is impossible for scientists or any domain experts to manually go over such a large collection of patterns. In some sense, the frequent patterns themselves are becoming the "data" which needs to be mined.

Indeed, reducing the number of frequent patterns has been a major theme in frequent pattern mining research. Much of the research has been on itemsets; itemset data can be generalized to many other pattern types. One general approach has been to mine only patterns that satisfy certain constraints; well-known examples include mining maximal frequent patterns [21], closed frequent patterns [19] and non-derivable itemsets [8]. The last two methods are generally referred to as lossless compression since we can fully recover the exact frequency of any frequent itemsets. The first one is lossy compression since we cannot recover the exact frequencies. Recently, Xin et al. [27] generalize closed frequent itemsets to discover a group of frequent itemsets which δ-cover the entire collection of frequent itemsets. If one itemset is a subset of another itemset and its frequency is very close to the frequency of the latter superset, i.e., within a small fraction (δ), then the first one is referred to as being δ-covered by the latter one. However, the patterns being produced by all these methods are still too numerous to be very useful. Even the δ-cover method easily generates thousands of itemset patterns. At the same time, methods like top-k frequent patterns [11], top-k redundancy-aware patterns [26], and error-tolerant patterns [29] try to rank the importance of individual patterns, or revise the frequency concept to reduce the number of frequent patterns. However, these methods generally do not provide a good representation of the collection of frequent patterns.

This leads to the central topic of this paper: what are good criteria to concisely represent a large collection of frequent itemsets, and how can one find the optimal representations efficiently? Recently, several approaches have been proposed to tackle this issue [2, 28, 24]. Two key criteria being employed for evaluating the concise representation of itemsets are the *coverage criterion* and *frequency criterion*. Generally speaking, the coverage criterion assumes the concise representation is composed of a small number of itemsets with the entire collection of frequent itemsets represented or covered by those itemsets. Typically, if one itemset is a subset of another, then we refer to the first one as being *covered* (or represented) by the latter one. The frequency criterion refers to the capability of the frequency of any frequent itemset to be inferred (or estimated) from the concise representation. Following these criteria, we summarize the previous efforts.

1.1 Prior Work on Frequent Pattern Summarization

The Spanning Set Approach [2]: In this work, the authors de-

fine a formal coverage criterion and propose to use K itemsets as a concise representation of a collection of frequent itemsets. Basically, the K itemsets are chosen to maximally cover the collection of frequent itemsets. They further show that finding such K itemsets corresponds to the classical set-cover problem and thus is NP-hard. The well-known greedy algorithm is applied to find the concise representation.

The major problem with this approach is that the frequency (or the support measure) is not considered. Indeed, how to effectively integrate a frequency criterion with the coverage criterion is an open problem pointed out by the authors in [2].

The Profile-Based Approach [28]: Yang *et al.*'s work [28] is the first attempt to answer the call from [2]. Their concise representation is derived from the K clusters formed by all frequent itemsets. All frequent itemsets in a cluster are covered and estimated by a so-called *pattern profile*, which contains three components. The first component is an itemset which is the union of all the frequent itemsets in the cluster. For instance, if a cluster contains three frequent itemsets, $\{a, c\}$, $\{a, d, e\}$ and $\{c, f, g\}$, then their first component is the itemset $\{a, c, d, e, f, g\}$. Let D be the set of transactions in which these itemsets occur. Then, the second component is the total number of transactions in D, and the third component (also referred to as the *distribution vector*) records all the relative frequencies for each individual item in D. Given this, a frequent itemset in the cluster is estimated based on the independence assumption: for instance, the frequency of $\{a, c\}$ is estimated as $p(\{a, c, d, e, f, g\}) \times p(a) \times p(c)$, where $p(\{a, c, d, e, f, g\})$ is the relative size of D in the entire database, and $p(a), p(c)$ are the relative frequencies for items a and c in D. To form the K clusters, they propose applying the Kullback-Leibler divergence of the distribution vectors to pairs of pattern profiles. Initially, the profile for an individual frequent itemset is derived by treating it as a single-member cluster. Given this, they apply hierarchical agglomerative and k-means clustering to partition the collection of itemsets into K clusters.

The Markov Random Field Approach [24]: Wang *et al.* target better estimation of the frequent itemsets. They propose the construction of a global Markov random field (MRF) model to estimate the frequencies of frequent itemsets. This model utilizes the dependence between items, specifically, the conditional independence, to reduce their estimation error. Specifically, this approach processes all the frequent itemsets in a level-wise fashion. In each level, it identifies those frequent itemsets which cannot be estimated well by the current model, i.e., the estimation error is higher than a user-defined tolerance threshold (δ), and add those itemsets into the model. Then, it will re-train the model after each level if any new itemsets are added. In the empirical study, their model shows better estimation accuracy than earlier approaches.

Though the last two approaches make significant progress towards restoring frequencies for frequent itemsets, they fall short of being effective and efficient for pattern summarization from several perspectives. First, with regard to effectiveness, the key question is how we can minimize the estimation error (restoration error) given the collection of itemsets. Both approaches, however, are very heuristic in nature and so do not provide any theoretical justification of why their methods can achieve such goal. As for efficiency, both approaches are computationally expensive. For instance, Yang *et al.*'s method must repetitively access the original database, and the heuristics they introduce to avoid such access do not maintain good estimation results. Wang *et al.*'s approach repetitively invokes the expensive probabilistic learning procedures, such as the Junction tree inference and MCMC procedure [24]. Finally, neither method considers coverage rate; it is not clear how their methods can integrate with the spanning set approach to provide frequency estimation in addition to the coverage criterion. In other words, the open problem raised by [2] has yet to be answered.

1.2 Problem Definition

Based on our discussion, we see that the effective and efficient restoration of the frequency of summarized frequent itemsets remains a significant open problem. Before we formally define this problem, we present some basic notation.

Let I be the set of all items $I = \{o_1, o_1, \cdots, o_m\}$. An itemset or a pattern P is a subset of $I (P \subseteq I)$. A transaction database is a collection of itemsets, $D = \{d_1, d_2, \cdots, d_n\}$, such that $d_i \subseteq I$, for all $i = 1, 2, \cdots, n$. The collection of transactions that contains P is represented as $D_P = \{d_i | P \subseteq d_i \text{ and } d_i \in D\}$. The frequency of an itemset P is denoted as $f(P) = |D_P|/|D|$. Let F_α be the collection of frequent itemsets with minimum support α: $F_\alpha = \{P : |D_P| \geq \alpha\}$.

DEFINITION 1. **(Restoration Function and Restoration Error)** *A restoration (estimation) method for a set of itemsets S is a function mapping S to a value between 0 to 1: $\hat{f} : S \rightarrow [0, 1]$. The restoration quality can be measured by a p-norm for the relative error (or alternatively the absolute error):*

$$E_p = \sum_{P \in S} |\frac{\hat{f}(P) - f(P)}{f(P)}|^p = \sum_{P \in S} |1 - \frac{\hat{f}(P)}{f(P)}|^p$$

For computational purpose, the 2-norm is chosen in this study. Clearly, the best restoration quality a restoration method can achieve is $E_p = 0$. However, using this measure, this would mean either we record the frequency of each itemset or the restoration method scans the original database. Such methods provide neither a succinct representation nor better interpretation of the collection of itemsets. We would like a restoration function to be more concise.

To build a concise restoration function for S, certain probabilistic models with a list of parameters $\Theta = (\theta_1, \cdots, \theta_m)$ are commonly employed. Generally speaking, the fewer the number of parameters, the more concise the model is. For instance, in [28], a probabilistic model employs the independence assumption for all the items in a set of S. The number of parameters for this model is bounded by the number of items in S. For a more complex model, such as the MRF model [24], the number of parameters can change depending on the number of constraints being incorporated into the model. Given this, we can see that finding a concise restoration function needs to consider both the feature selection and model fitting issues [12].

In this study, we will investigate how to identify the best parameters for the restoration function utilizing probabilistic models based on the independence assumption and its extension (the factor-graph model). Note that our work is distinguished from the existing work [28, 24] since we formalize the restoration problem as an *optimization problem* and seek methods to directly optimize the restoration error. By comparison, the existing methods do not recognize the optimization nature of this problem and simply develop some heuristics for the restoration purpose. Given this, we introduce our first research question as follows.

Research Problem 1: *Given a set of itemsets S and assuming a single probabilistic model either based on the independence assumption or conditional independence assumption, what is the best restoration method, i.e., the optimal parameters, which can minimize the restoration error?*

Research problem 1 will answer the question of how well a single probabilistic model can maximally restore the frequency

for a collection of itemsets. However, using a single probabilistic model to restore the frequency of the entire collection of frequent itemsets, F_α, can lead to very high restoration error. The general strategy to handle this problem is to partition the entire collection into several parts and to restore the frequencies for each part independently. Research problem 2 studies how to find a good partition of F_α so that total restoration error can be minimized. Research problem 3 studies the situation where the individual parts are given. This essentially corresponds to the open problem raised in [2].

Research Problem 2: In this problem, we study the best partition of F_α so that the total restoration error can be minimized. *Assuming we can partition the entire collection of frequent itemsets into K disjoint subsets:*

$$F_\alpha = F_\alpha^1 \cup F_\alpha^2 \cup \cdots \cup F_\alpha^K$$

where $F_\alpha^i \cap F_\alpha^j = \emptyset$ for $i \neq j$, and we can build a restoration function for each partition individually, how can we minimize the total restoration error (the sum of the restoration errors from each restoration function)?

$$\min_{F_\alpha^1,\cdots,F_\alpha^K} \sum_{i=1}^{K} \sum_{P_i \in F_\alpha^i} (1 - \frac{\hat{f}_i[\mathbf{\Theta}_i](P_i)}{f(P_i)})^2.$$

Here, $\hat{f}_i$ is the restoration function for partition F_α^i, and $\mathbf{\Theta}_i$ is the optimal parameter set for $\hat{f}_i$ to minimize the restoration error in partition F_α^i, i.e., to minimize $\sum_{P_i \in F_\alpha^i} (1 - \frac{\hat{f}_i[\Theta_i](P_i)}{f(P_i)})^2$.

Research Problem 3: In [2], to approximate F_α, K representative itemsets $A_1, A_2, \cdots, A_K$ are chosen:

$$F_\alpha \approx \bigcup_{i=1}^{k} 2^{A_i} = 2^{A_1} \cup 2^{A_2} \cup \cdots \cup 2^{A_k}$$

where, 2^{A_i} is the power set of A_i, i.e., containing all the subsets of A_i. *Given the K representative itemsets, how we can derive a good restoration function to estimate the frequencies of the itemsets being covered by the K itemsets?*

1.3 Our contributions:

In this paper, we propose a set of novel regression-based approaches to effectively and efficiently summarize frequent itemset patterns. Specifically, our contributions are as follows.

1. We show that to minimize the restoration error for a set of itemsets, seeking the optimal restoration function based on a probabilistic model corresponds to a *non-linear regression problem*. We show that under certain conditions, we can transform the non-linear regression problem to the *linear regression problem*, which can be solved efficiently.

2. We propose two new methods, the *K-regression approach* and the *tree-regression partition approach* to partition the entire collection of frequent itemsets F_α to minimize the restoration error. The K-regression approach employs a K-means type clustering method. The K-partition achieved by K-regression is guaranteed to achieve a local minimum of the total restoration error. The tree-regression partition approach employs a decision-tree type of top-down partitioning process. It tries to produce K disjoint subsets of itemsets, where the total restoration error can be minimized.

3. We discuss how to apply the *K-regression approach* to estimate the frequency for the collection of itemsets being covered by the K representative itemsets. Our method provides a positive answer for the open problem raised in [2].

4. We have evaluated our approaches on both real and synthetic datasets. Our approaches show significance performance improvement in terms of both summarization accuracy (restoration error), and summarization computational cost.

2. REGRESSION FOR OPTIMAL RESTORATION FUNCTIONS

2.1 Regression for Independence Probabilistic Model

The restoration function based on the independence probabilistic model for a given set of itemsets S is as follows. Under the independence assumption, all the items in S are totally independent. In addition, we also assume such independence is held only for a fraction of the entire database D, where the fraction is denoted as $p(S)$. The relative frequency for an item a_i in S is denoted as $p(a_i)$. Then for an itemset $I_s \in S$, we can apply the independence assumption to estimate the frequency of I_s as follows:

$$\hat{f}(I_s) = p(S) * \prod_{a_j \in I_s} p(a_j)$$

As an example, to estimate the frequency of $\{a, c, d\} \in S$, we have:

$$\hat{f}(\{a, c, d\}) = p(S) * p(a) * p(c) * p(d)$$

Given this, our optimal restoration function corresponding to the parameter set $\mathbf{\Theta}$=($p(S)$, $p(a_1)$, $\cdots$, $p(a_n)$), which minimizes the total restoration error, is given as follows:

$$\min_{\mathbf{\Theta}} \sum_{I_s \in S} (1 - \frac{\hat{f}[\mathbf{\Theta}](I_s)}{f(I_s)})^2 = \min_{\mathbf{\Theta}} \sum_{I_s \in S} (1 - \frac{p(S) * \prod_{a_i \in I_s} p(a_i)}{f(I_s)})^2$$

This corresponds to a nonlinear least square optimization problem. We can further formulate it as a regression problem. For each itemset $I_s \in S$, we define the *dependent variable*, y, to be the frequency of I_s, $f(I_s)$, and the *independent variables*, $\mathbf{1}_{\{a_1 \in I_s\}}, \cdots, \mathbf{1}_{\{a_n \in I_s\}}$, to be the indicators of items in I_s, i.e., $\mathbf{1}_{\{a_i \in I_s\}} = 1$ if $a_i \in I_s$, and $\mathbf{1}_{\{a_i \in I_s\}} = 0$, otherwise. Given this, the regression function can be written:

$$y = f(I_s) \approx \hat{f}[\mathbf{\Theta}](x_1, \cdots, x_n) = p(S) * \prod_{i=1}^{n} p(a_i)^{\mathbf{1}_{\{a_i \in I_s\}}} \quad (*)$$

Several well-known methods, including *Gauss-Newton method*, the *method of steepest descent*, and the *Marquardt algorithm*, can be deployed to identify the optimal parameters Θ to minimize the total restoration error [22]. However, these algorithms are generally very computationally expensive. Therefore, in this study, we consider another commonly used alternative approach, to transform the nonlinear regression into linear regression. Applying a logarithmic transformation on both side of $(*)$, we have

$$\log f(I_s) \approx \log p(S) + \sum_{i=1}^{n} x_i \log p(a_i)$$

Assuming S has a total of l itemsets, $I_1, \cdots, I_l$, then we can represent our linear regression problem as follows:

$$\begin{bmatrix} \log f(I_1) \\ \log f(I_2) \\ \vdots \\ \log f(I_l) \end{bmatrix} \approx \begin{bmatrix} 1 & \mathbf{1}_{\{a_1 \in I_1\}} & \cdots & \mathbf{1}_{\{a_n \in I_1\}} \\ 1 & \mathbf{1}_{\{a_1 \in I_2\}} & \cdots & \mathbf{1}_{\{a_n \in I_2\}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \mathbf{1}_{\{a_1 \in I_l\}} & \cdots & \mathbf{1}_{\{a_n \in I_l\}} \end{bmatrix} \begin{bmatrix} \log p(S) \\ \log p(a_1) \\ \log p(a_2) \\ \vdots \\ \log p(a_n) \end{bmatrix}$$

We denote the left most vector as $\mathbf{Y}$, the matrix as $\mathbf{X}$, and the right most vector as β. Thus, the system of linear equations is denoted as

$$\mathbf{Y} \approx \mathbf{X} \cdot \beta$$

The parameters β with the least square error,

$$\min_{\beta} ||\mathbf{X} \cdot \beta - Y||^2 = \min_{\beta} \sum_{i=1}^{l} (\log \frac{\hat{f}(I_i)}{f(I_i)})^2,$$

can be derived by the standard linear regression technique [12]:

$$\beta = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$$

Clearly, linear regression provides an elegant solution for finding the optimal parameters for the restoration function. However, we need to consider several important issues regarding the linear and nonlinear regression.

The Bounded Parameter Problem: An important problem facing both nonlinear and linear regression is that each parameter in $\mathbf{\Theta}$ is bounded:

$$0 \leq p(S) \leq 1, 0 \leq p(a_i) \leq 1, \forall a_i \in A$$

However, the general solutions for regression will not take this into consideration. A simple analysis can show that the identified parameters will always be non-negative. Thus, we only have to deal with the problem that the parameters can be potentially larger than 1. Here, we discuss two alternative strategies: The first strategy is based on the assumption that the main goal of a restoration function is to minimize the restoration error. Given this, the bounded parameter problem is not significant. But, we still would like the estimator $\hat{f}$ to be bounded, i.e., $\hat{f} \leq 1$. The simple solution is to introduce the *wrapped restoration function*, $\hat{f}'$:

$$\hat{f}' = \hat{f}(\hat{f} \leq 1); \hat{f}' = 1(\hat{f} > 1)$$

We note that the total restoration error introduced by the wrapped restoration function ($\hat{f}'$) is always *smaller* than the original restoration function $\hat{f}$.

The other strategy is to find the optimal parameters which satisfy the bounded parameter constraint. This problem is referred to as the bounded-variable (or bound constraint) least square problem [9]. Several methods have been developed to numerically search for the optimal solution.

The Optimality of Linear Regression: Logarithmic transformation and linear regression provide an elegant solution for the optimal restoration function. However, we note that the criterion optimized in the linear regression is different from the original total restoration error. Thus, a key question is how the optimal parameters identified in linear regression can minimize the restoration error. To answer this question, we need to analyze the relationship between these two criteria, which we denote as C_N (for nonlinear regression) and C_L (for linear regression):

$$C_N = \sum_{i=1}^{l} (1 - \frac{\hat{f}(I_i)}{f(I_i)})^2 \ \ and \ \ C_L = \sum_{i=1}^{l} (\log \frac{\hat{f}(I_i)}{f(I_i)})^2$$

First, we can see that they both tend to minimize the difference between $\hat{f}(I_i)$ and $f(I_i)$. Clearly,

$$(1 - \frac{\hat{f}(I_i)}{f(I_i)})^2 \to 0 \Leftrightarrow \frac{\hat{f}(I_i)}{f(I_i)} \to 1 \Leftrightarrow (\log \frac{\hat{f}(I_i)}{f(I_i)})^2 \to 0$$

This only suggests their convergence tendency. Taylor expansion provides a more detailed analysis:

$$\log(1+x) = x - x^2/2 + x^3/3 - \cdots, \ for \ |x| < 1.$$

Thus, we have

$$\begin{aligned} C_L &= \sum_{i=1}^{l} (\log \frac{\hat{f}(I_i)}{f(I_i)})^2 = \sum_{i=1}^{l} (\log (1 + \frac{\hat{f}(I_i) - f(I_i)}{f(I_i)}))^2 \\ &= \sum_{i=1}^{l} (\frac{\hat{f}(I_i) - f(I_i)}{f(I_i)} - (\frac{\hat{f}(I_i) - f(I_i)}{f(I_i)})^2/2 + \cdots \\ &= \sum_{i=1}^{l} (\frac{\hat{f}(I_i) - f(I_i)}{f(I_i)})^2 + O((\frac{\hat{f}(I_i) - f(I_i)}{f(I_i)})^3) \end{aligned}$$

where $O(z)$ represents the term having the same order of magnitude as z. Note that here we assume $|\frac{\hat{f}(I_i) - f(I_i)}{f(I_i)}| < 1$, i.e., $\hat{f}(I_i)$ does not overestimate $f(I_i)$ by a factor of two: $\hat{f}(I_i) < 2f(I_i)$ Under this condition, C_L can be reasonably close to C_N. In Section 5, we empirically evaluate the restoration accuracy by the linear regression method and our results will show our method indeed achieves very small restoration errors.

Validity of Independence Assumption: An interesting question is the validity of the restoration function model. Since we apply the independence assumption to model the set of itemsets S, is it a probabilistically valid model? Statistical tests [6], such as Person's Chi-Square test and Wilks' G^2 statistics, can be used to test the independence hypothesis. For the restoration purpose, we generally do not explicitly perform these tests. The better the independence model fits the data, the more accurate the estimation will be. Therefore, we will try to find an optimal partition for a set of itemsets, such that each partition can fit the model nicely, i.e. all its itemsets can be estimated accurately. This essentially corresponds to finding several independence models to represent S. Finding the optimal partition to minimize the total restoration error is the central topic of Section 3.

2.2 Generalized Restoration Function based on Itemsets

Here, we go beyond the independence assumption and try to capture the correlation between items in the restoration function. Basically, we generalize the independence model to utilize not only items, but also itemsets in the restoration function. Formally, the restoration function $\hat{f}$ for a given set of itemsets S is written as ($I_s \in S$):

$$\hat{f}(I_s) = \mu_0 \prod_{j:T_j \in T} \mu_j^{\mathbf{1}_{\{T_j \subseteq I_s\}}}$$

where, T is the set of itemsets being incorporated in the restoration function. Note that this model can be viewed as an *factor graph* model [17], where the restoration function factors into a product of several *local functions*, each having an itemset as argument, i.e. $\mu_j^{\mathbf{1}_{\{T_j \subseteq I_s\}}}$. However, this model is different from the MRF (Markov Random Field) model used by Pavlov *et al.* [20] and Wang *et al.* [24], though the format bears some similarity. This model does not form a probabilistic measure over all the itemsets in A, which records all the items in S.

Besides, a subtle but important difference is that in MRF, the estimation for an itemset frequency corresponds to a marginal distribution, which requires the expensive sum-over-product computation. Our model only needs the product computation. For restoration, this computation cost for constructing and performing estimation is much cheaper than the MRF model's cost. As we show later, it also achieves better estimation accuracy than the MRF model.

Given a factorization format for the restoration function, the key issue is how to choose a set of itemsets, T, for minimizing the restoration error. Essentially, this is a feature selection problem for the nonlinear regression. However, no standard solution is available for such task. To deal with the difficulty, we transform the nonlinear regression into linear regression by logarithmic transformation and then we utilize a *shrinkage method* or *subset selection method*.

Specifically, let $T = \{T_1, \cdots, T_n\}$ be our candidate itemsets (candidate features) which can be incorporated into the restoration function. For computational purpose, we can choose T to be all frequent itemsets with size less than k. Applying a logarithmic transformation for the factorization model, we have

$$\log f(I_s) \approx \log \hat{f}(I_s) = \log \mu_0 + \sum_{j=1}^{n} \mathbf{1}_{\{T_j \subseteq I_s\}} \log \mu_j.$$

Assuming S has a total l itemsets, $I_1, \cdots, I_l$, then we can represent our linear regression problem as follows:

$$\begin{bmatrix} \log f(I_1) \\ \log f(I_2) \\ \vdots \\ \log f(I_l) \end{bmatrix} \approx \begin{bmatrix} 1 & \mathbf{1}_{\{a_1 \in I_1\}} & \cdots & \mathbf{1}_{\{a_n \in I_1\}} \\ 1 & \mathbf{1}_{\{a_1 \in I_2\}} & \cdots & \mathbf{1}_{\{a_n \in I_2\}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \mathbf{1}_{\{a_1 \in I_l\}} & \cdots & \mathbf{1}_{\{a_n \in I_l\}} \end{bmatrix} \begin{bmatrix} \log \mu_0 \\ \log \mu_1 \\ \log \mu_2 \\ \vdots \\ \log \mu_n \end{bmatrix}$$

The standard *stepwise selection* procedures can be employed to choose a good set of features to minimize the sum of squared error $\sum_{i=1}^{l} (\log \frac{\hat{f}(I_i)}{f(I_i)})^2$. Alternatively, we can apply a shrinkage method, such as *Ridge regression* and *Lasso regression*, to find a good set of features [12].

3. OPTIMAL PARTITION FOR RESTORATION

As we mentioned before, using a single probabilistic model to restore the frequency for all the frequent itemsets F_α is likely to result in high restoration error. Instead, we can try to partition F_α into several clusters, and then construct the optimal restoration function for each cluster. In this section, we introduce two heuristic algorithms which try to find the optimal partition of F_α so that the total restoration error can be minimized. Subsection 3.1 introduces the *k-Regression* approach which employs a k-means type clustering method. The k-partition achieved by k-regression is guaranteed to achieve a local minimum of the total restoration error. Subsection 3.2 discusses the *tree-regression* partition approach which employs a decision-tree type of top-down partitioning process.

3.1 K-Regression Approach

Algorithm 1 employs a k-means type clustering procedure. Initially, the entire collection of frequent itemsets F_α is randomly partitioned into k groups (Line 1). Then, we apply the regression approach to find the optimal parameters which can minimize the restoration error for each group (Line 2). The regrouping step (Lines 4-7) will apply all the k computed restoration functions to estimate the frequency for each itemset (Line 5) and assign it to the group whose restoration function results in the smallest error (Line 6). After the regrouping, we will again apply the regression approach to compute the new restoration function for each newly-formed group (Line 8). We will compute the total restoration error by adding the restoration error of the k groups. This process is repeated (Lines 4-9) until the total restoration error converges, generally by testing if the improvement for the total restoration error is less than a certain threshold.

Algorithm 1 K-Regression(F_α, K)

1: randomly cluster F_α into K disjoint groups: F_α^1, $F_\alpha^2, \cdots, F_\alpha^K$;
2: apply regression on each group, F_α^i to find the restoration function $\hat{f}[\mathbf{\Theta}_i]$ with optimal parameters $\mathbf{\Theta}_i$;
3: **repeat**
4: **for all** itemset $I_s \in F_\alpha$ **do**
5: compute the estimation error for each restoration function, $\epsilon_i = |f(I_s) - \hat{f}[\mathbf{\Theta}_i](I_s)|, \forall i, 1 \leq i \leq K$;
6: reassign I_s to the group which minimizes ϵ_i;
7: **end for**
8: apply regression on each new group, F_α^i to find the restoration function $\hat{f}[\mathbf{\Theta}_i]$ with optimal parameters $\mathbf{\Theta}_i$;
9: compute the new total restoration error;
10: **until** the total restoration error converges to a local minimum or the improvement is very small
11: output the K groups, $F_\alpha^1, \cdots, F_\alpha^K$, and their restoration functions, $\hat{f}[\mathbf{\Theta}_1], \cdots, \hat{f}[\mathbf{\Theta}_K]$

Note that the k-regression is generic in the sense that it may be applied with any of the approaches discussed in Section 2—the independence model or its generalization and their corresponding regression (nonlinear and linear). In the following, we investigate several important issues for *k-Regression*.

Convergence Property for K-Regression: The major difference which distinguishes the k-regression algorithm from the k-means algorithm is that k-regression repetitively computes the optimal parameters for each newly formed group using regression and use the estimation error to determine the reassignment for each itemset. Will this algorithm converge? We provide a positive answer for this question. Let E_i be the restoration error the regression approach tries to optimize for each local cluster. For instance, if we apply the nonlinear regression, we directly optimize $E_i = \sum_{P_i \in F_\alpha^i} (1 - \frac{\hat{f}_i[\Theta_i](P_i)}{f(P_i)})^2$. If we apply the linear regression, we optimize $E_i = \sum_{P_i \in F_\alpha^i} (\log \frac{\hat{f}_i[\Theta_i](P_i)}{f(P_i)})^2$. If we apply the ridge regression for the generalized restoration function, we optimize $E_i = \sum_{P_i \in F_\alpha^i} (\log \frac{\hat{f}_i[\Theta_i](P_i)}{f(P_i)})^2 + \lambda \sum_{j=1}^{n} (\log \mu_j)^2$. Given this, we define the *total restoration error* $E = \sum_{i=1}^{K} E_i$.

THEOREM 1. *Each iteration (Line 4-9) in the k-regression algorithm will decrease the total restoration error monotonically, i.e., the new total restoration error is always less than or equal to the total restoration error. In other words, the k-Regression algorithm will tend to converge the total restoration error to a local minimum.*

Proof: This result can be established based on the following observation. Let E be the total error based on the k groups and their corresponding restoration functions. Consider the start of a new iteration from Line 4. In the regrouping step, the estimation error ϵ_i for each itemset is monotonically decreasing since $\epsilon_i = \arg\min_i |\hat{f}(I_s) - f(I_s)|$. If we use the current restoration function and the new grouping to compute the intermediate to-

tal restoration error E', we can easily have $E \geq E'$. Then, the regression will compute a new restoration function based on the new group, which will minimize the restoration error for each group. Thus, for the new total restoration error E'', we have

$$E \geq E' \geq E''$$

□

Shrinking Property for the Covering Itemsets: Another interesting property of k-regression relates to the *covering itemsets*. For a group of itemsets F_α^i, the covering itemset, denoted as A_i, is the set of all the items appearing in F_α^i. To use the covering itemsets to represent the collection itemsets, we would like a low false positive ratio: $|2^{A_i} \cap F_\alpha^i|/|2^{A_i}|$. Though the k-regression algorithm does not directly minimize this criteria, a related quantity, the size of the covering itemset, $|A_i|$ will indeed monotonically decrease.

THEOREM 2. *At each iteration in k-regression, the size of the covering set of the newly-formed i group, F_α^i, is less than or equal to the size of the covering set for the previous i group.*

Proof: This is based on the observation that if we reassign an itemset I_s to a new group i according to the minimal estimation error, then I_s must be covered by A_i, the covering set of the original i group ($I_s \subseteq A_i$). This is because if I_s is not covered by I_s, the restoration function will estimate the frequency I_s to be 0. Clearly, any other groups which can cover I_s will have a better estimation accuracy. Given this, after regrouping, each new member in group i is already covered by the original A_i. Thus, the new covering set, which is the union of all the new members in group i, will also be a subset of the original one. □

This property shows that all the covering sets tend to shrink or at least maintain the same size after each iteration. In Section 4, we will utilize this property to derive the restoration function for the given k representative itemsets of F_α.

Computational Complexity: The computational complexity depends on the restoration function and its corresponding regression method. Here, we consider the restoration function based on the independence probabilistic model and then apply linear regression. Given this, the computational complexity for the k-regression algorithm is $O(L|F_\alpha|N^2)$, where L is the number of iterations, N is the total number of items in F_α, and $O(|F_\alpha|N^2)$ is the complexity of a typical regression solver [9].

3.2 Tree-Regression Approach

This approach tries to provide a high level description of the entire collection of frequent itemsets F_α through a decision tree structure. For instance, Figure 1 shows a decision tree partition for a set of itemsets. The non-leaf node records a condition for an item x, such that all the itemsets of the left partition contain x, and all of the itemsets of the right partition do not. The leaf node is the final partition of the set of itemsets. Given this, the problem is how to partition F_α into k disjoint parts, such the total restoration error can be minimized (under the condition that each part will have its own restoration function). Here, we introduce a greedy algorithm for this purpose.

Algorithm 2 builds the decision tree partition in a top-down and greedy fashion. At the root node, we will try to partition the entire set of frequent itemsets F_α into two disjoint sets. The criteria for choosing the best split condition, i.e., including item x or not, is based on the total restoration error. Suppose the original set of itemsets is split into two sets, and then we construct the optimal restoration function for each of them independently. The sum of the two restoration errors from the two sets is denoted as the total restoration error. Once we identify the item x which

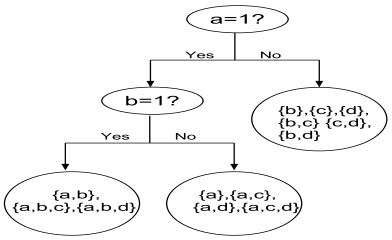


Figure 1: A decision-tree partition for a set of itemsets

results in the best partition with minimal restoration error, we will put the two new sets into a priority queue. The queue will determine which set of itemsets will be partitioned next. The reason for introducing such a queue is that we will only partition F_α into k parts. We will always choose the set in the queue with the highest average restoration error (the restoration error for the set over the number of itemsets in the set) to be partitioned first. This procedure (repetitively choosing set of itemsets from the queue and split it into two parts) will continue until we have our k partition.

Algorithm 2 TreeRegression(F_α, K)

```
1: Put F_α as the only element into priority queue Q;
2: repeat
3:    let S the first partition in Q;
4:    for all item x ∈ A do {A is the covering itemset of S}
5:       split S into two parts using item x;
6:       apply regression on each part to find the restoration
         function with optimal parameters;
7:       compute the restoration errors for each part and their
         sum as the total restoration error for this partitioning;
8:    end for
9:    select the partitioning whose total error is the smallest;
10:   score both S_1 and S_2, the two parts of S from the best
      partitioning, with their average restoration error (total er-
      ror over the number of itemsets in the parts);
11:   put S_1 and S_2 into the priority Q (the higher the average
      error, the earlier in the queue);
12: until Q has K parts
13: output the K parts in Q and the corresponding partitioning
    conditions
```

As in the k-regression algorithm, the computational complexity of the tree-regression algorithm depends on the format of the restoration function and the corresponding regression method. Assuming the restoration function is based on the independence probabilistic model and then parameter-fitted using linear regression, the computational complexity for the tree-regression algorithm is $O(K(|F_\alpha|N^2)N) = O(K|F_\alpha|N^3)$, where $O(|F_\alpha|N^2)$ is the complexity of the regression solver, and the additional N is the cost for testing each split condition. In general, the tree-regression has higher computational cost than k-regression. However, it provides a simpler description for the k-partition of the entire collection of frequent itemsets.

4. RESTORATION FUNCTION FOR K REPRESENTATIVE ITEMSETS

In [2], to approximate F_α, k representative itemsets $A_1, A_2, \cdots, A_K$ are chosen: $F_\alpha \approx \bigcup_{i=1}^{k} 2^{A_i} = 2^{A_1} \cup 2^{A_2} \cup \cdots \cup 2^{A_k}$ where, 2^{A_i} is the power set of A_i, i.e., containing all the subsets of A_i.

(How to extract these k representative itemsets is omitted. The detail is in [2]). In this section, we consider how to derive a good restoration function to estimate the frequencies of the itemsets being covered by the k itemsets.

Here, the main issue is that an itemset can be covered by more than one set of k representative itemsets. If we look at each powerset of the representative itemsets as the grouping of itemsets, these groups are not disjoint, but overlapped with each other. Further, we assume each powerset has its own restoration function. The problem is that when we try to restore the frequency of the itemsets, assuming we have only the k representative itemsets, we will know which restoration function we should use for recovery.

Our address this problem with a two-step procedure. In the first step, we will try to construct a restoration function for each representative itemset. Then, in the second step, we propose a weighted average method to build a global restoration function. For the first step, we consider two alternatives: 1) we simply apply all the itemsets which are subsets of A_i, i.e., 2^{A_i}, for regression to find the optimal restoration function; 2) we modify the k-regression algorithm so that each itemset is grouped to only the one itemset which covers it. Basically, in Line 1 of Algorithm 1, for each itemset being covered by the k representative itemsets, we will randomly assign it to one of the representative itemsets which covers it. Then, based on Theorem 2 (the shrinking property), the k-regression will assign each itemset to one representation itemset and minimize the total restoration error.

For step two, the global restoration function combines the k local restoration functions produced in step 1 to estimate the frequency of an itemset. The frequency of a given itemset I_s is estimated to be the average of all the local functions, whose corresponding representative itemsets cover I_s:

$$\hat{f}(I_s) = \frac{\sum_{I_s \subseteq A_i} \hat{f}_i(I_s)}{\sum_{I_s \subseteq A_i} 1}.$$

5. EXPERIMENTAL RESULTS

In this section, we study the performance of our proposed approaches, *k-regression* and *tree-regression*, on both real and synthetic datasets. We will directly compare our approaches against Yan *et al.*'s pattern profile method (the k-means approach) [28], which has become the benchmark for other research [24]. We implement both our approaches using C++ and R [1], which is one of the most popular open-source programming environment for statistical computing. The implementation of pattern profile method is provided by the author of [24]. We use Chris Bolget's Apriori implementation to collect the frequent itemsets [7].

5.1 Experimental Setup

All tests were run on an AMD Opteron 2.0GHz machine with 2GB of main memory, running Linux (Fedora Core 4), with a 2.6.17 x86 64 kernel. We use four real datasets and two synthetic datasets. The real datasets are chess, mushroom, BMS-POS and BMS-WebView-1 [16]. The synthetic datasets are T10I4D100K and T40I10D100K. All dataset information are publicly available at the FIMI repository (http://fimi.cs.helsinki.fi/). The main characters of the datasets are summarized in table 1.

We evaluate the summarization performance in terms of the restoration error and the summarization time. Even though our approach seeks to optimize L_2 error (i.e. 2-norm. See Definition 1), we report the average L_1 (1-norm) error as below, to compare our results with previous efforts [28].

datasets	$\lvert\mathcal{I}\rvert$	$\lvert\mathcal{T}\rvert$	$\lvert DB\rvert$	density
chess	75	3,196	118,252	high
mushroom	119	8,124	186,852	medium
BMS-POS	1,658	515,597	3,866,978	low
BMS-WebView-1	497	59,602	149,005	low
T10I4D100K	1,000	100,000	$\approx$ 1,000,000	low
T40I10D100K	1,000	100,000	$\approx$ 4,000,000	medium-low

Table 1: dataset characters; $|\mathcal{I}|$ is the number of distinct items; $|\mathcal{T}|$ is the number of transactions; $|DB|$ is the size of the transaction database in terms of the total items

$$E = \frac{1}{|S|} \sum_{P \in S} |\frac{\hat{f}(P) - f(P)}{f(P)}| = \frac{1}{|S|} \sum_{P \in S} |1 - \frac{\hat{f}(P)}{f(P)}|$$

In addition, we note that in our approaches, we restore the frequency for all frequent itemsets. Thus, we use $S = F_\alpha$ for a given support. The pattern profile only restores the frequency for all closed frequent itemsets. Thus, the corresponding S would be the set of closed itemsets.

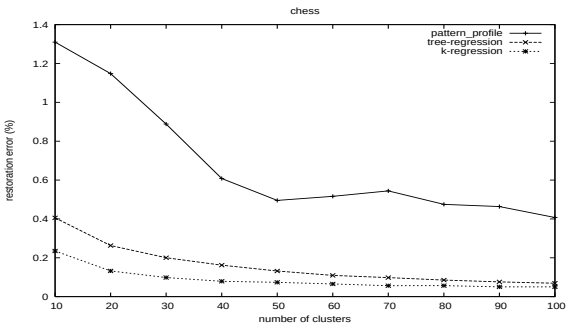
5.2 Results

Restoration Performance Evaluation on Real Datasets: Figure 2 shows the average restoration error for the four real datasets, chess, mushroom, BMS-POS, BMS-WebView-1, with support level 75%, 25%, 0.8%, and 0.2%, respectively. Their corresponding number of frequent itemsets are 7635, 3781, 1695, and 798, respectively. For each dataset, we compare the average restoration errors of the three methods: the pattern profile method [28], the k-regression method, and the tree-regression method. Here, we use simple linear regression for the independence probabilistic model discussed in Subsection 2.1. For each dataset, we vary the number of clusters from 10 to 100.

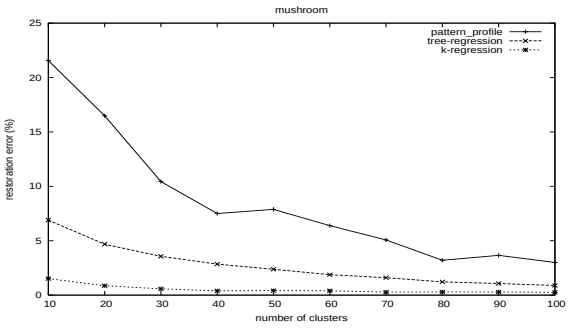
From Figure 2, we can see that the two new methods, tree-regression and k-regression, achieve consistently and significantly smaller restoration errors than the pattern profile method. Table 2(e) shows the average restoration errors for figures 2(b) 2(a) 2(c) 2(d). On average, k-regression achieves lower error than the pattern profile method by a factor of approximately 8, 17, 7, and 20 times for chess, mushroom, BMS-POS and BMS-WebView-1, respectively. On average, tree-regression achieves lower restoration error than the pattern profile method by a factor of 4 3, 5, and 3 times for chess, mushroom, BMS-POS and BMS-WebView-1 datasets respectively. In addition, the running time of k-regression is generally much faster than that of tree-regression, especially, when the number of distinct items is relatively large. This is understandable since in tree-regression, for each partition we will choose the best splitting item from all the distinct items. Such iteration can be rather costly. Figure 3(a) compares the running time of the three methods using BMS-POS dataset. Due to space limitations, we omit other figures on the running time using real dataset.

In summary, we can see that k-regression performs restoration more efficiently and accurately.

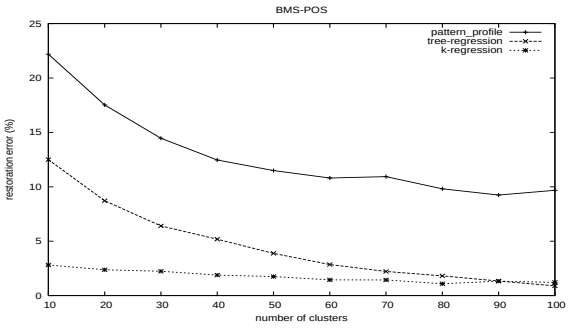
Scalability Study using Synthetic Datasets: In this group of experiments, we compare the performance of the pattern profile methods and k-regression with respect to different support level on the two synthetic datasets $T10I4D100K$ and $T40I10D100K$. Figure 3 shows the experimental results. Here, we choose the number of clusters to be 20, 50 and 80 for both methods. For instance, k-regression_20 means running the k-regression method with 20 clusters. Figure 3(b) and 3(c) shows the restoration error



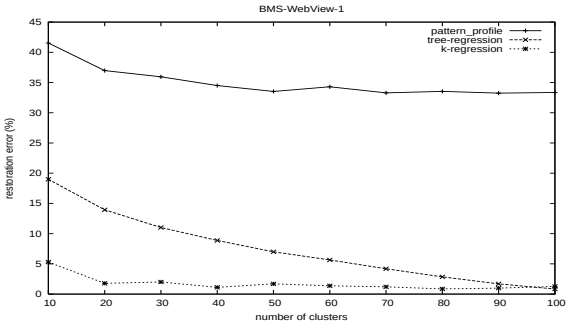
(a) chess restoration error



(b) mushroom restoration error



(c) BMS-POS restoration error



(d) BMS-WebView-1 restoration error

(e) average restoration error summarization			
datasets	pattern profile	regression tree	K-regression
mushroom	8.5%	2.7%	0.53%
Chess	0.69%	0.16%	0.09%
BMS-WebView-1	35.0%	7.5%	1.8%
BMS-POS	12.9%	4.6%	1.8%

Figure 2: Average Restoration Errors for Real Datasets

		restoration error		running time (sec)	
dataset	method	$k = 10$	$k = 30$	$k = 10$	$k = 30$
mushroom 40%	indep. model	0.45%	0.27%	14.2	30.6
	forward stepwise	19.0%	2.7%	133.8	58.7
chess 89%	indep. model	0.049%	0.039%	14.8	17.2
	forward stepwise	1.9%	1.0%	166.9	128.9

Table 2: Generalized restoration function using forward stepwise feature selection

against the support level. Figure 3(d) and 3(e) shows the running time against the support level.

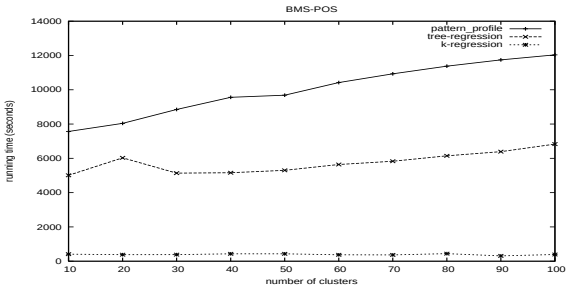
In both datasets, k-regression performs much more accurate estimation than the pattern profile method. On average, k-regression has only 0.07% and 0.8% restoration error, while the pattern profile method has 10% and 6.8% restoration error, on the $T10I4D100K$ and $T40I10D100K$ dataset, respectively. In terms of the running time, the k-regression method runs an average of 20 and 8 times faster than the pattern profile method on the $T10I4D100K$ and $T40I10D100K$ dataset, respectively. In addition, the restoration error of the pattern profile method increases rapidly as the support level decreases, while the k-regression method maintains almost the same level of accuracy.

Independence Model vs. Generalized Restoration Function: In this experiment, we study if the generalized restoration function can perform better than the independence model. Our experiment shows that the independence model with simple linear regression actually works better than the generalized restoration function with forward stepwise feature selection. For the latter method, we utilize all the frequent 2-itemsets in addition to the 1-items in the model and then try to select the best features in the final restoration function. Table 2 shows the k-regression results of the experiment on mushroom and chess datasets. The results show that latter method is both much more expensive and not as accurate as the simple linear regression. We identify two causes for the performance difference. First, the greedy subset selection method does not always capture the best features for the model. Second, the independence model already can fit the data well, so the extra feature offered by the 2-itemsets does not help to boost the estimation accuracy. In certain cases, the generalized restoration function does in fact improve the accuracy of the restoration function. However, considering the percentage improvement and the cost of the computation, we conclude that the independence model together with simple linear regression is a better choice for the restoration function.

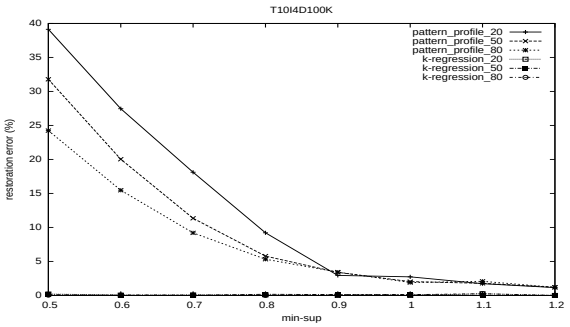
Restoration Function for the K representative itemsets: In this experiment, we compare the two alternatives in Section 4 to restore the frequency for the itemsets which are subsets of the given k-representative itemsets, produced by the greedy algorithm introduced in [2]. The two methods are the k-regression method and the method which distributes an itemset to all the representative itemsets which covers it (referred to as the *distribute* method. We found that in most cases, the k-regression method works as well as or better than the *distribute* methods. In particular, in certain cases, such the mushroom datasets of Figure 4, the k-regression method performs significantly better than the other one. Here, the mushroom is at 25% support level and chess is at 75% support level. The reason we believe that the *distribute* method works not as good as the k-regression is that it does not utilize re-partitioning like the k-regression does. Thus, the independence model is likely not to work well.

6. CONCLUSIONS

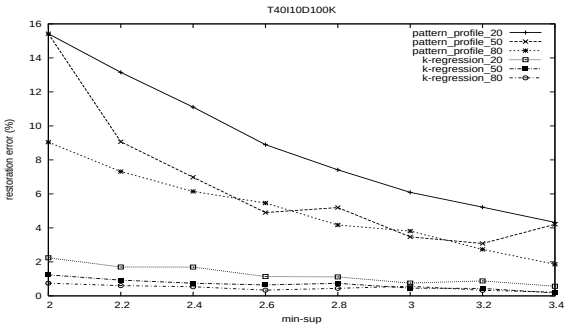
In this work, we have introduced a set of regression-based ap-



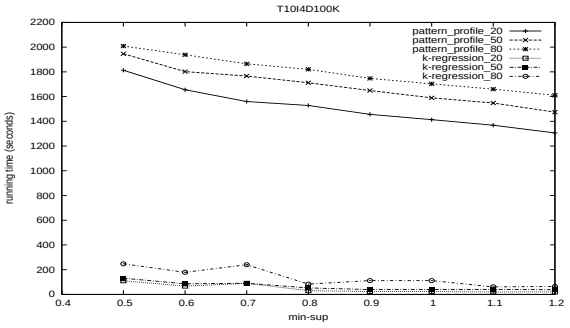
(a) BMS-POS running time



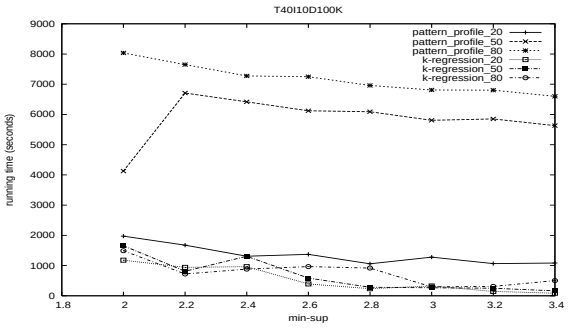
(b) T10I4D100K restoration error



(c) T40I10D100K restoration error



(d) T10I4D100K running time



(e) T40I10D100K running time

Figure 3: Running Time and Synthetic Datasets

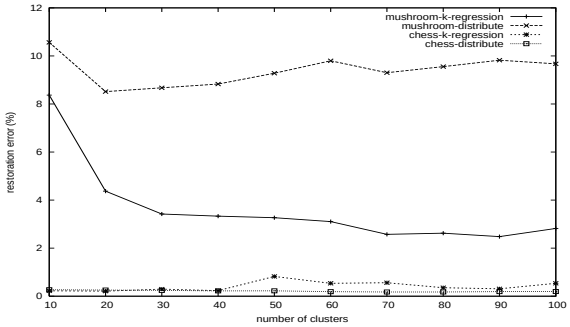


Figure 4: Restoration Function for K Representative Itemsets

proaches to summarize frequent itemsets. We have shown how the restoration problem can be formulated as a non-linear least square optimization problem and how linear regression can be applied to solve it efficiently. The two methods we proposed successfully marry the well-know k-means and decision tree algorithms with the linear regression problem. In addition, the k-regression methods can be naturally applied to the open problem of how to estimate the frequency for the collection of itemsets being covered by k representative itemsets. The experimental results on both real and synthetic datasets have shown that our method can achieve orders of magnitude improvement in accuracy over the pattern profile approach, with much smaller running time. We believe our approaches offer an interesting way to handle estimation problems for other types of data as well. In the future, we plan to investigate how our methods can be applied to other patterns, include subtrees and subgraphs.

7. REFERENCES

[1] The r project for statistical computing. http://www.r-project.org/.

[2] Foto Afrati, Aristides Gionis, and Heikki Mannila. Approximating a collection of frequent sets. In *KDD*, 2004.

[3] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD Conference*, pages 207–216, May 1993.

[4] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, 1994.

[5] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*, pages 3–14, 1995.

[6] Alan Agresti. *Categorical Data Analysis*. Wiley, 2002.

[7] Christan Borgelt. Apriori implementation. http://fuzzy.cs.Uni-Magdeburg.de/ borgelt/Software.

[8] Toon Calders and Bart Goethals. Non-derivable itemset mining. *Data Min. Knowl. Discov.*, 14(1):171–206, 2007.

[9] Gene H. Golub and Charles F. Van Loan. *matrix computations, 3rd*. The John Hopkins University Press, 1996.

[10] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2000.

[11] Jiawei Han, Jianyong Wang, Ying Lu, and Petre Tzvetkov. Mining top-k frequent closed patterns without minimum support. In *ICDM*, 2002.

[12] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer-Verlag, 2001.

[13] Jun Huan, Wei Wang, Deepak Bandyopadhyay, Jack Snoeyink, Jan Prins, and Alexander Tropsha. Mining protein family-specific residue packing patterns from protein structure graphs. In *Eighth International Conference on Research in Computational Molecular Biology (RECOMB)*, pages 308–315, 2004.

[14] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *Principles of Knowledge Discovery and Data Mining (PKDD2000)*, pages 13–23, 2000.

[15] Ruoming Jin and Gagan Agrawal. A systematic approach for optimizing complex mining tasks on multiple datasets. In *Proceedings of the ICDE Conference*, 2006.

[16] Ron Kohavi, Carla Brodley, Brian Frasca, Llew Mason, and Zijian Zheng. KDD-Cup 2000 organizers' report: Peeling the onion. *SIGKDD Explorations*, 2(2):86–98, 2000. http://www.ecn.purdue.edu/KDDCUP.

[17] F. R. Kschischang, B. J. Frey, and H. A. Loeliger. Factor graphs and the sum-product algorithm. *Information Theory, IEEE Transactions on*, 47(2):498–519, 2001.

[18] Wei Li and Ari Mozes. Computing frequent itemsets inside oracle 10g. In *VLDB*, pages 1253–1256, 2004.

[19] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. In *ICDT '99: Proceeding of the 7th International Conference on Database Theory*, 1999.

[20] Dmitry Pavlov, Heikki Mannila, and Padhraic Smyth. Beyond independence: Probabilistic models for query approximation on binary transaction data. *IEEE Trans. Knowl. Data Eng.*, 15(6):1409–1421, 2003.

[21] Jr. Roberto J. Bayardo. Efficiently mining long patterns from databases. In *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, 1998.

[22] G. A. F. Seber and C. J. Wild. *Nonlinear Regression*. John Weiley & Sons, Inc., 1989.

[23] Craig Utley. Microsoft sql server 9.0 technical articles: Introduction to sql server 2005 data mining. http://technet.microsoft.com/en-us/library/ms345131.aspx.

[24] Chao Wang and Srinivasan Parthasarathy. Summarizing itemset patterns using probabilistic models. In *KDD*, 2006.

[25] Takashi Washio and Hiroshi Motoda. State of the art of graph-based data mining. *SIGKDD Explor. Newsl.*, 5(1):59–68, 2003.

[26] Dong Xin, Hong Cheng, Xifeng Yan, and Jiawei Han. Extracting redundancy-aware top-k patterns. In *KDD*, 2006.

[27] Dong Xin, Jiawei Han, Xifeng Yan, and Hong Cheng. Mining compressed frequent-pattern sets. In *VLDB*, 2005.

[28] Xifeng Yan, Hong Cheng, Jiawei Han, and Dong Xin. Summarizing itemset patterns: a profile-based approach. In *KDD*, 2005.

[29] M. T. Yang, R. Kasturi, and A. Sivasubramaniam. An Automatic Scheduler for Real-Time Vision Applications. In *Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS)*, 2001.

[30] Takeshi Yoshizawa, Iko Pramudiono, and Masaru Kitsuregawa. SQL based association rule mining using commercial RDBMS (IBM db2 UBD EEE). In *Data Warehousing and Knowledge Discovery*, pages 301–306, 2000.

[31] Mohammed J. Zaki. Efficiently mining frequent trees in a forest. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80, 2002.

[32] Mohammed J. Zaki and Charu C. Aggarwal. Xrules: an effective structural classifier for xml data. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 316–325, 2003.