Summarizing Frequent Patterns Using Profiles

Gao Cong[1], Bin Cui[2], Yingxin Li[3], and Zonghong Zhang[4]

[1] The University of Edinburgh, UK
gao.cong@ed.ac.uk
[2] Department of Computer Science and Technology, Peking University, Beijing, China
cuibin@pku.edu.cn
[3] Department of Anesthesiology, School of Medicine, University of Virginia, USA
[4] Service and Applications, Institute for Infocomm Research, A-Star, Singapore
zhzhang@i2r.a-star.edu.sg

Abstract. Frequent pattern mining is an important data mining problem with wide applications. The huge number of discovered frequent patterns pose great challenge for users to explore and understand them. It is desirable to accurately summarizing the set of frequent patterns into a small number of patterns or profiles so that users can easily explore them. In this paper, we employ a probability model to represent a set of frequent patterns and give two methods of estimating the support of a pattern from the model. Based on the model, we develop an approach to grouping a set of frequent patterns into k profiles and the support of frequent pattern can be estimated fairly accurately from a relative small number of profiles. Empirical studies show that our method can achieve compact and accurate summarization in real-life data and the support of frequent patterns can be restored much more accurately than the previous method.

1 Introduction

Mining frequent patterns or itemsets is a fundamental and essential problem in many data mining applications, such as association rule mining, classification, and clustering (e.g. [3, 7, 17]). There are a host of frequent pattern mining algorithms (e.g. [3, 9]) that discover the complete set of patterns that occur in at least ξ (*minimum support*) fraction of a dataset. The complete set of frequent patterns is often huge in number, which makes the interpretability of frequent patterns very difficult. The concepts of closed frequent patterns and maximal frequent patterns usually can help in reducing the output size. However, they can only partly alleviate the problem. The size of closed frequent patterns (or maximal frequent patterns) often remains to be very large and thus it is still difficult for users to examine and understand them.

Recently, several proposals were made to discover k patterns or profiles. This allows users to specify the value of k and thus only discover a small number of patterns or approximation. The concept of top-k patterns is proposed by Han et al [10]. Although this provides users the option to discover only the k most frequent patterns, this is not a generalization of all frequent patterns satisfying a support threshold. k *covering sets* was proposed by Afrati et al. [1] to approximate a collection of frequent patterns, i.e. each frequent pattern is covered by at least one of the k sets. The proposal is interesting in generalizing the collection of patterns into k sets. However, the support information is

M.L. Lee, K.L. Tan, and V. Wuwongse (Eds.): DASFAA 2006, LNCS 3882, pp. 171–186, 2006.

ignored in the approximation and it is unknown how to recover the support of a pattern from the k sets. Support is a very important property of a pattern and plays a key role in distinguishing patterns.

Yan et al [19] proposed an approach to summarizing patterns into k profiles by considering both pattern information and support information; each cluster (profile) is represented with three elements: the *master pattern*, i.e. the union of the patterns in the cluster, the number of transactions supporting the clusters, the probability of items of the master pattern in the set of transactions supporting the pattern. The supports of frequent patterns can be estimated from the k clusters. It is assumed in [19] that the items in the master pattern are independent in each profile. The independence model is simple and easy to learn, but it is fairly inaccurate since items are usually not independent. However, it is too expensive to consider n-dimensional probability distribution, where n is the number of items.

In this paper, we adopt an alternative probability model to represent a profile composed of a set of frequent patterns. Instead of assuming the independence among items in [19], we consider the pairwise probabilities that are still easy to compute. From the pairwise probabilities, we build simple Bayesian Network to estimate the n- dimensional probability distribution, and thus can estimate the supports of the patterns. Alternatively, we can also compute a rough support estimation for the patterns directly from the pairwise probability. With the model, we can measure the similarity between two profiles (and patterns) using Killback-Leibler divergence and a complementary distance score, and thus arrange all patterns into a set of (hierarchial) groups. In the hierarchical tree, users can explore the frequent patterns in a top-down manner as suggested in [19]. Our methods can successfully summarize patterns into tens of profiles while the support of patterns can be recovered accurately. We conduct extensive experiments on several real datasets. Our method can summarize thousands of patterns accurately using only tens of profiles on all real datasets we tested. Compared with method in [19], our methods make great improvement in summarization quality measured by restoration error.

The rest of this paper is organized as follows: Section 2 will give the problem statement. In Section 3, we present the probability model ro represent profiles and methods of estimating the supports of frequent patterns from the profiles. We introduce algorithms for grouping frequent patterns into profiles in Section 4. The experimental results are reported in Section 5. Section 6 discusses the related work. Finally we conclude this paper in Section 7.

2 Problem Statement

Let $I = \{i_1, i_2, ..., i_n\}$ be a set of **items** which represent attribute values in a transaction database DB. A pattern (or itemset) X is a non-empty subset of *I*. Given a DB, the **support** of a pattern X, denoted as $sup(X)$, is the fraction of tuples in the DB which contains X.

Definition 1. ***Frequent Pattern:*** *Given a minimum support threshold ξ ($0 \leq \xi \leq 1$) and a database DB, a pattern X is* **frequent** *if $sup(X) \geq \xi$.*

Definition 2. ***Closed Frequent Pattern:*** *A frequent pattern X is closed if there does not exits a pattern X' such that $X \subset X'$ and X is contained by the same set of tuples as X'.*

Closed frequent patterns are a lossless compression of frequent patterns and the complete set of frequent patterns and their supports can be derived from the set of closed frequent patterns. The size of closed frequent patterns is usually (much) smaller than the size of frequent patterns. Hence, in this paper we summarize closed frequent patterns as [19] while the proposed method equally applies to summarize frequent patterns.

Table 1. An example of database transactions

Transaction	Number of transactions
acd	100
bcd	100
abcd	800

Table 1 shows a sample dataset, where the first column represents the transactions and the second the number of transactions. For example, 100 transactions have only items a, c, and d; and 100 transactions have only items b, c, and d. There are totally 1000 transactions in this example. If we set the minimum support at 50%, clearly pattern $< abcd >$ is frequent. Additionally, we know that all its subsets are frequent as well, i.e. $< a, b, c, d, ab, ac, ad, bc, bd, cd, abc, abd, acd, bcd, abcd >$. Among these 15 frequent patterns, we can find 4 closed patterns according to the $definition$ 2, which are $< cd, acd, bcd, abcd >$. As one can see, the number of closed frequent patterns is much less than that of frequent patterns.

We observe that in this example the supports of all the 4 closed patterns are very close to each other and they are similar in terms of their items. We could summarize the 4 patterns into one fermentative pattern $< abcd >$.

Problem statement. Given a set of frequent closed patterns $CP = X_1, X_2, ..., X_m$ that are mined from database $DB = t_1, t_2, ...t_n$, pattern summarization is to group the m closed frequent patterns into k pattern profiles, each of which is represented with a probability model.

3 Model of Profiles

Suppose that patterns X_1, ... X_s are grouped together to form a profile. The profile can be characterized with two important properties: one is *master pattern* $\chi = X_1 \cup X_2 \cup ... \cup X_s$ generated by the union of the m patterns in the group; the other is the set of transactions $DB_u = DB_{X_1} \cup DB_{X_2} \cup ... \cup DB_{X_s}$. Consider these information, we propose a probability model to represent the profile.

Definition 3. ***Profile Model:*** *Let $X_1, X_2, ..., X_s$ be a set of patterns and $DB' = \cup_i DB_{X_i}$, $i = 1, ..., s$. A summarization profile over $X_1, X_2, ..., X_s$ is a triple $\Phi =< \chi, \rho, \theta >$. $\chi = X_1 \cup X_2 \cup ... \cup X_s$ is the master pattern of $X_1, X_2, ..., X_m$; $\rho = |DB'|/|DB|$ is defined as the support of the profile; suppose that $\chi = \{x_1, x_2, ..., x_t\}$, θ is composed of two parts:*

1. $p(x_i = 1) = |DB_{x_i}|/|DB'|$, *where* $x_i \in \chi$ *and* DB_{x_i} *is the set of tuples in* DB' *that contain item* x_i.
2. $p(x_i = 1, x_j = 1) = |DB_{x_i \cup x_j}|/|DB'|$, *where* $x_i, x_j \in \chi \wedge i \neq j$ *and* $DB_{x_i \cup x_j}$ *is the set of tuples in* DB' *that contain items* x_i *and* x_j.

Given $p(x_i = 1)$, $p(x_j = 1)$ and $p(x_i = 1, x_j = 1)$, one can obtain the probability distribution of $p(x_i, x_j)$ in the set of tuples DB' using the inclusion-exclusion principle: $p(x_i = 1, x_j = 0) = p(x_i = 1) - p(x_i = 1, x_j = 1)$; $p(x_i = 0, x_j = 1) = p(x_j = 1) - p(x_i = 1, x_j = 1)$; and $p(x_i = 0, x_j = 0) = 1 - p(x_i = 1) - p(x_j = 1) + p(x_i = 1, x_j = 1)$.

Table 2. An example of profile

	a	b	c	d	ab	ac	ad	bc	bd	cd
probability	0.9	0.9	1	1	0.8	0.9	0.9	0.9	0.9	1

Using the dataset in Table 1, we can build a pattern profile for $< abcd >$. Table 2 shows the profiles by deriving the distribution vectors for the sample datasets. For example, $p(a) = \frac{100+800}{1000} = 0.9$. In addition, without accessing the original dataset, we can infer that $< abd >$ is less frequent than the $< acd >$. Pattern profile actually provides more information than the master pattern itself; it encodes the distribution of sub-patterns.

The k-set model in [1] represents the collection of patterns only with a master pattern $\chi = X_1 \cup X_2 \cup ... \cup X_s$, and thus the support information is lost. In [1], the profile is represented not only by the master pattern but also by the probability distribution of items in χ in the set of transactions $DB' = DB_{X_1} \cup DB_{X_2} \cup ... \cup DB_{X_s}$. The key difference of our model from that in [19] is that we include the pair-wise probability distribution of items in DB', while it is assumed that items are independent boolean random variables in [19]. As to be shown, the pair-wise probability not only allows us to build more accurate probability model to characterize the patterns in a profile, but also provides better measures to group patterns into profiles.

3.1 Estimate Support Using Profiles

In this subsection, we will present two methods of estimating the support of a pattern given a profile. The first is based on a simple Bayesian model derived from the pairwise probability and then applies on chain rule to compute the joint probability of items in a pattern; the second is simplified version of the first method.

Before presenting our methods, we first give some background on estimating the probability. The support of a pattern in dataset DB can be regarded as the summary of the probability that the pattern occurs in each transaction.

$$p(X|DB) = \sum_{t \in DB} p(X|t) * p(t)$$

where $p(t) = 1/|DB|$ and $p(X|t) = 1$ if $X \in t$, 0 otherwise.

We regard the probability of observing a pattern as the probability that the pattern is generated by its profile times the probability of observing the profile from a transaction.

$$p(X|t) \approx p(X|\Phi, t) * p(\Phi|t) \approx p(X|\Phi) * p(\Phi|t),$$

where we assume the conditional independence $p(X|\Phi, t) = p(X|\Phi)$. According to the model, one can estimate the support for a given pattern X from the profile that it belongs to. Given a profile Φ over a set of patterns $X_1, X_2, ..., X_s$, the estimated support for X based on the profile Φ is

$$\widehat{s}(X) = \Phi.\rho * p(X|\Phi) \tag{1}$$

where $\Phi.\rho = |DB_{X_1} \cup ... \cup DB_{X_s}|/|DB|$.

The problem here is how to estimate the $p(X|\Phi)$. Our first approach to estimating probability $p(X|\Phi)$ is to build a simple Bayesian network, a Chow-Liu tree model, for each profile using the pairwise probability. We first compute the mutual information between each pair of items in a profile, and then compute the minimum spanning tree from the full graph whose nodes are the items and edges are the mutual information. After obtaining the minimum spanning tree, i.e. a polytree Bayesian network, we can use the Pearls' belief propagation algorithm [15] to compute $p(X|\Phi)$.

The Chow-Liu tree approximates an nth-order distribution by a product of n -1 second order component distributions. The Chow-Liu Tree structure is proved to be the optimal one in the sense of Maximum Likelihood criterion [6]. Before introducing the algorithm for building Chow-Liu tree model [6], we first give the formula to compute mutual information between two variables:

$$I(X, Y) = \sum_x \sum_y p(x, y) log \frac{p(x, y)}{p(x)p(y)} \tag{2}$$

Intuitively, $I(X, Y)$ measures the amount of information that random variable X contains about Y (and vice versa). The higher of the value, the more correlated are the two variables; $I(X, Y) = 0$ if X and Y are independent.

We need to perturb the probabilities $p(x, y)$, $p(x)$ and $p(y)$ to avoid zero probabilities. For example, we compute $p'(x)$ as follows:

$$p'(x) = \lambda u + (1 - \lambda)p(x) \tag{3}$$

where λ is a constant, $0 < \lambda < 1$, and u is the prior of x and can be the background distribution of item x.

Algorithm 1 outlines how to learn a Chow-Liu tree structure for a profile Φ. In the beginning, it computes the mutual information using Equation 2 between any pairs of items in the profile Φ. In lines 4-10, it repeats until a Chow-Liu tree is discovered. The algorithm in lines 4-10 aims to find a spanning tree with the maximal mutual information, which can be implemented with Kruskal's algorithm of Prim's algorithm [8] for finding minimum spanning tree.

Complexity analysis. In order to compute the mutual information of each pair, we need to scan the database once to compute the probability. It takes $O(f^2 n)$, where n is the number of tuples supporting the profile i.e. $\Phi.\rho \times |DB|$ and f is the size the largest

Input: Transaction database DB
Profile $\Phi = < \chi, \rho, \theta >$
Output: a Chow-Liu tree model.

1. let MI be the set of mutual information $I(x_i, x_j)$ between any two items x_i, x_j in $\Phi.\chi$;
2. initiate a tree $T(V, E)$, where each item in $\Phi.\chi$ becomes one node in V and $E = \emptyset$;
3. initiate $k = 0$;
4. **while** $k < |\Phi.\chi| - 1$
5. select pair (x_i, x_j) such that $x_i, x_j = argmax_{x_i,x_j} I(x_i, x_j)$;
6. **if** there is no cycle formed in T **then**
7. add edge (x_i, x_j) into T;
8. $E = E \cup (x_i, x_j)$
9. $k = k + 1$;
10. $MI = MI \backslash \{(x_i, x_j)\}$;
11. **return** T;

Fig. 1. Algorithm CLtree

tuple. It takes $O(d^2)$ to compute the mutual information, where d is the size of $\Phi.\chi$. If one adopts Kruskal's algorithm to compute the minimum spanning tree (lines 4-10), it takes $O(d^2 log d)$; it takes $O(d^2 + d log d)$ if one uses Prim's algorithm [8]. Hence, Algorithm 1 can finish in $O(f^2 n + d^2 + d^2 log d)$ using Kruskal's algorithm.

After the Chow-Liu tree model is learned, we can compute $p(X|\Phi)$ based on the chain rule and some specified order of the items in χ:

$$P(X|\Phi) = p(x_1, \Phi) \prod p(x_i|x_{i-1}, ..., x_1, \Phi) \tag{4}$$

$p(x_1, \Phi)$ is already in the profile Φ and the $p(x_i|x_{i-1}, ..., x_1, \Phi)$, $i = 2, ..., d$ (d is the size of X) can be computed using the belief propagation algorithm [15]. The belief propagation algorithm is a message-passing scheme that updates the probability distributions for each node in a Bayesian network in response to observations of one or more variables, i.e. to compute the probability of x_i after the values of $x_{i-1}, ..., x_1$ are set as evidence. Hence, $\prod p(x_i|x_{i-1}, ..., x_1, \Phi)$ can be computed by taking the product of belief measures. On a polytree (Chow-Liu tree is a polytree), the belief propagation algorithm converges in time proportional to the number of edges in the tree, i.e. $|\Phi.\chi| - 1$. Note that there is no need to propagate the impact of each instantiation to the entire polytree; the propagation are transmitted only to those variables in $P(X|\Phi)$. Interested readers can refer to [15] for the algorithm details.

Our second method further approximates the nth-order distribution by replacing the higher order conditional probabilities with second order ones:

$$P(X|\Phi) = p(x_1, \Phi) \prod p(x_i|x_{i-1}, \Phi) \tag{5}$$

The above formula approximates $p(x_i|x_{i-1}, ..., x_1, \Phi)$ with $p(x_i|x_{i-1}, \Phi)$, and thus learning Bayesian network is not required. The second method will be more efficient than the first one. As to be shown in experiment, this simplified model can also improve greatly the independent model in [19].

The probability model of profiles provides a method of representing a set of patterns in a compact way and methods of recovering their supports. The remaining problem is how to group the set of patterns into k profiles.

4 Grouping Patterns

In this section, we first introduce the similarity measures between the profiles and then describe a clustering algorithm using the similarity measures.

The key of clustering patterns into profiles with high quality is a good distance measure for the patterns and profiles as well. At the beginning of the clustering, each pattern can be regarded as a profile with triple elements as described in the previous section. Hence, one ideal distance measure between two profiles Φ_p and Φ_q should consider (1) the overlapping of master patterns $\Phi_p.\chi$ and $\Phi_q.\chi$; (2) the similarity of the probability distribution of items in the two profiles, which can reflect the correlation between the transactions that support the two profiles; (3) the support of the two profiles.

These three factors are correlated one another. Kullback-Leibler divergence (KL-divergence) is widely used to compute the divergence between two probability distributions and is also adopted in [19]. We choose KL-divergence to measure the distance between two profiles since it considers the three factors, especially the first two, of the profiles: if two patterns differ greatly in the items (of their master patterns), or the two probability distributions differ greatly, the KL distance will be large; if the profiles are similar in master patterns as well as probability, it is highly possible that their supports are similar (Note that it is not sufficient).

The KL-divergence of two variable can be computed as follows:

$$\mathrm{KL}(x||y) = \sum_{x,y} p(x) log \frac{p(x)}{p(y)} \tag{6}$$

When the $p(x)$ and $q(x)$ have zero probability, KL$(x||y)$= ∞. This can be avoided by smoothing the $p(x)$ and $q(x)$ as Equation 3.

The smaller $KL(x||y)$ value is, the more similar of the distributions of variables x and y. In our profile model, there are distributions for pairwise variables and single variables and they contain overlapping information. We consider three combinations to compute the KL-divergence of the two profiles.

1. $\mathrm{KL}(\Phi_p||\Phi_q) = \sum_{x_i,x_j \in C} \mathrm{KL}(p(x_i,x_j)||q(x_i,x_j))$, where $C = \Phi_p.\chi \cup \Phi_q.\chi$;

   The formula is simple and computes the KL-divergence using joint distribution for every pair of items in the union of the master patterns of the two profiles.

2. $\mathrm{KL}(\Phi_p||\Phi_q) = \sum_{x_i,x_j \in C} \mathrm{KL}(p(x_i,x_j)||q(x_i,x_j)) + \sum_{x_i \in D} \mathrm{KL}(p(x_i)||q(x_i))$, where $C = \Phi_p.\chi \cap \Phi_q.\chi$ if $|\Phi_p.\chi \cap \Phi_q.\chi| > 1$; $C = \emptyset$ otherwise, and $D = \Phi_p.\chi \cup \Phi_q.\chi - C$;

   The formula computes the KL-divergence using the joint distribution for common pairs of items in the master patterns of the two profiles, and computes the KL-divergence using the distribution of single item for items that appear only in one master pattern of the two profiles.

3. $\mathtt{KL}(\Phi_p||\Phi_q) = \sum_{x_i,x_j \in \Phi_p.\chi \vee x_i,x_j \in \Phi_p.\chi} \mathtt{KL}(p(x_i,x_j)||q(x_i,x_j)) +$
$\alpha \times \mathtt{KL}(p(\Phi_p.\chi)||q(\Phi_p.\chi)) + \beta \times \mathtt{KL}(p(\Phi_q.\chi)||q(\Phi_q.\chi))$,
where $\alpha = 1$ if $|\Phi_p.\chi| = 1$; 0 otherwise, and $\beta = 1$ if $|\Phi_q.\chi| = 1$; 0 otherwise.

   The formula can be regarded as a combination of the above two methods: it employs the pair wise distribution if a pair appears in at least one of the master patterns; it employs the distribution for single item if the master pattern is a single item.

We introduce one complementary measure when two patterns have the same KL-divergence score. This often happens especially when the profile is composed of one pattern in the beginning of clustering. For example, consider three patterns $X_1 = \{abcd, 100\}$, $X_2 = \{abef, 500\}$ and $X_3 = \{ab, 600\}$, where the pattern and its support are separated by comma. It is easy to verify that $\mathrm{KL}(X_1||X_2)$ and $\mathrm{KL}(X_1||X_3)$ are the same whichever the three combination we use. In this example, suppose that we want to cluster the three patterns into two groups. Intuitively we should group X_2 and X_3 together but not X_1 and X_2 although their KL-divergences are the same. This is because X_2 and X_3 have similar number of support and thus their transactions that support them likely have large overlapping. This example implies that KL-divergence score alone may not be sufficient sometimes.

We can accurately compute the overlapping of two patterns (or profiles) Φ_p and Φ_q by

$$D(\Phi_p, \Phi_q) = (DB_{\Phi_p} \cap DB_{\Phi_q})/(DB_{\Phi_p} \cup DB_{\Phi_q})$$

This measure is proposed in [18] to compute the similarity of patterns. But it takes $O(|DB|)$ to compute one pair of patterns and thus is relatively expensive. Instead, we use a simplified score and find it achieves reasonably good results in all our experiments.

$$D'(\Phi_p, \Phi_q) = |\Phi_p.\rho - \Phi_q.\rho|/max(\Phi_p.\rho, \Phi_q.\rho) \tag{7}$$

where $\Phi_p.\rho$, $\Phi_q.\rho$ are the support of Φ_p and Φ_q respectively.

In what follows, we will introduce how to cluster the patterns based on the two measures introduced above. We adopt hierarchical agglomerative clustering to group profiles. Hierarchical clustering is shown to obtain stable results in [19] in clustering frequent patterns. But other clustering methods, such as K-means, can be adopted to cluster profiles.

Hierarchical clustering can not only produce k profiles, but also generate a dendrogram which allows users to explore the k profiles in a top-down manner. Algorithm 2 outlines the hierarchical clustering method for profiles. In line 4, the algorithm computes the pair-wise KL-divergence (Equation 6) and computes the complementary distance measure (Equation 7) in line 5. Note that we only compute the KL-divergence score between Φ_i and Φ_j using $\mathrm{KL}(\Phi_i||\Phi_j)$ ($i < j$) although the KL score is not symmetric. We found that the final clustering results are similar even if we compute $\mathrm{KL}(\Phi_j||\Phi_i)$. In lines 6-12, the algorithm repeats until the number of clusters becomes k. At each iteration, the algorithm picks two clusters that have the smallest KL-divergence score; if several pairs of clusters have the same smallest KL-divergence score, it picks

Input: Transaction database DB
Pattern set X= $\{X_1, X_2, ...X_m\}$
Number of profiles K
Output: a set of pattern profiles $\Phi_1, ..\Phi_k$.

1. initialize $k = m$ clusters, each of which contains one pattern;
2. **for** each Φ_i
3. **for** each Φ_j, $j < i$
/* compute the pairwise KL divergence between Φ_i and Φ_j;*/
4. $DIST1_{ij}$ = $KL(\Phi_i||\Phi_j)$
/* compute the complementary distance between Φ_i and Φ_j;*/
5. $DIST2_{ij}$ = $D'(\Phi_i, \Phi_j)$
6. **while** $k < K$
7. select Φ_p and Φ_q such that $DIST1_{pq}$ is the smallest among all $DIST1_{ij}$ and $DIST2_{pq}$ is the smallest among all $DIST2_{ij}$ whose $DIST1_{ij} = DIST1_{pq}$;
8. merge clusters Φ_p and Φ_q to a new cluster Φ_r;
9. $\Phi_r.\chi$ = $\Phi_p.\chi \cup \Phi_q.\chi$
10. DB_{Φ_r} = $DB_{\Phi_p} \cup DB_{\Phi_q}$
11. update profile of Φ_r
12. compute similarity scores between Φ_r and other profiles
13. scan dataset to update the K profiles
14. **return** Φ_i $(i = 1, ...K)$

Fig. 2. Algorithm HCluster

the pair of clusters with the minimum complementary distance. In line 9, the algorithm updates the master pattern of the new cluster by combining the master patterns from the two clusters generating the new cluster. In line 10, the algorithm computes the combined transactions supported by the new profile.

In line 11, the algorithm needs to update the probability of the new profile. One method is to rebuild the accurate profile after two profiles are combined. However, this is expensive since computing profile needs to scan the original dataset. Instead, we approximate the probability p(x,y) of the profiles (we approximate p(x) similarly):

$$p(x, y|\Phi_r) = \frac{\Phi_p.\rho}{\Phi_p.\rho + \Phi_q.\rho} p(x, y|\Phi_p) + \frac{\Phi_q.\rho}{\Phi_p.\rho + \Phi_q.\rho} p(x, y|\Phi_q) \tag{8}$$

Complexity analysis. The initial KL-divergence computation takes $O(m^2d^2)$, where m is the number of patterns and d is the size of the maximum master pattern of all profiles. The computation of initial complementary distance takes $O(m^2)$. For each cluster Φ_p, we maintain a distance list between Φ_p and other clusters and sort them in non-descending order. When a new cluster is generated, we create and sort a distance list for it in time $O(mlogm)$. Thus the hierarchical clustering itself can be done in $O(m^2logm)$. Since we adopt the approximation as Equation 8 to compute the profile of the merged cluster, and thus we do not need to scan the dataset. It can be updated in $O(d^2)$ time. Finally, we scan the dataset to update the K profiles, which takes d^2nK.

Quality evaluation. From a high quality profile, one should be able to estimate the support of a frequent pattern as close as possible. In this paper, we adopt the same quality measure as that used in [19], namely restoration error:

$$J = \frac{1}{|T|} \sum_{X \in T} \frac{|\widehat{s}(X) - s(X)|}{s(X)} \tag{9}$$

where T is the set of frequent patterns to be evaluated. Restoration error is the average relative error between the estimate support of a pattern and its real support. It is desirable that the restoration error is small, and thus profiles can provide an accurate estimation.

Given a pattern, it may be covered by the master patterns of several profiles. Hence, we need to estimate these supports for it. However, we only select the maximum one by following the method in [19].

$$\widehat{s}(X) = max_{\Phi_i} \widehat{s}(X|\Phi_i)(i \in [1, k]) \tag{10}$$

We realize that there are other options to make the selection. For example, given a pattern and several profiles whose master patterns cover the pattern, we can pick the profile whose master pattern is the most similar to the given pattern to estimate support for the given pattern.

5 Empirical Study

In this section, we report the performance evaluation of our summarization method. The algorithms were implemented with C++. All the experiments were conducted on 2.4GhZ, 512M memory Intel PC running Linux.

We used three real-life datasets:

- **Mushroom.** The Mushroom dataset consists of 8124 hypothetical mushroom samples with 119 distinct features; each sample has 23 features. This is a dense dataset and is available from the UCI machine leaning repository [1].
- **BMS-Webview1.** The BMS-Webview [22] is a web click-stream dataset. The dataset consists of 59602 web sessions (transactions) with 497 distinct product pages (items).
- **BMS-POS.** The BMS-POS [22] contains seven years worth of point-of-sale data from a large electronic retailer. Each item represents a product category and each transaction is a customer's purchase transaction consisting of all the product categories purchased at one time. The dataset consists of 515597 transactions with 1657 distinct items.

We first evaluate the three combinations of computing KL-divergence given in Section 4, then compare our methods with the existing method, and finally evaluate the effect of probability model on clustering results and restoration error. We employed restoration error as the evaluation metric as [19] does.

[1] http://www.ics.uci.edu/ mlearn/MLRepository.html

5.1 Comparison with Existing Method

Before we compare with the exiting summarizing method, we first evaluate the three combinations in Section 4 of computing the KL-divergence score. We apply the three combinations to compute the KL divergence in the clustering algorithm while using the Equation 5 to estimate the supports of frequent patterns. Figure 3 shows the results on the dataset BMS-Webview1. The combination 2 and combination 3 consistently outperform combination 1 while the difference between combination 2 and 3 is trivial. We have obtained qualitatively similar results on the other two datasets. One possible reason for the worse performance of combination 1 is that it considers some pairs of items that do not appear in the same profile, i.e. such pairs do not represent any profile, and thus it is not meaningful to compute the divergence of such pairs.

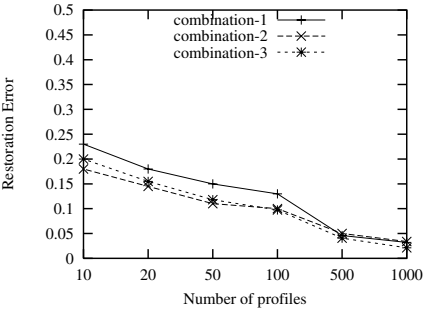

Fig. 3. The effect of different combinations to compute KL score on BMS-Webview1

We compare our algorithms against the latest work in [19] in terms of restoration error. The algorithm *summary* denotes the summarization method in [19]. According to the results of comparing three combinations given in Section 4, we use third combination to compute the KL divergence score for our methods. Furthermore, the algorithm *summary+* denotes the summarization method that uses the Equation 5 to estimate the support of a pattern; and the algorithm *summary++* employs the Equation 4 to estimate the support of a pattern.

We generated a set of 688 frequent closed patterns from Mushroom dataset by setting minimum support at 25% (The same set of frequent closed patterns was used in [19]). The maximum length of frequent patterns is 8. Figure 4 shows the average restoration error over the closed frequent patterns. Compared with the *summary* [19], both *summary+* and *summary++* can reduce the restoration error by at least 50%. The 688 patterns can be successfully summarized into 10 profiles with reasonable good quality: the average restoration error is less than 0.1.

We generated a set of 4195 frequent closed patterns from BMS-Webview by setting minimum support at 0.1% (the setting is the same as in [19]) and the maximum length of pattern is 6. Figure 5 shows the average restoration error over the set of closed frequent patterns. Both *summary+* and *summary++* outperform the *summary* by several times in terms of restoration errors. The *summary++* can reduce the restoration error of *summary+* by 50%. The 4195 patterns can be successfully summarized into 50 profiles

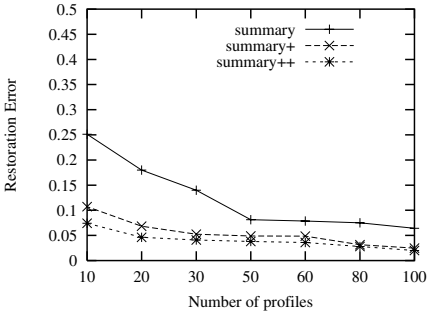

Fig. 4. Restoration Error for Mushroom

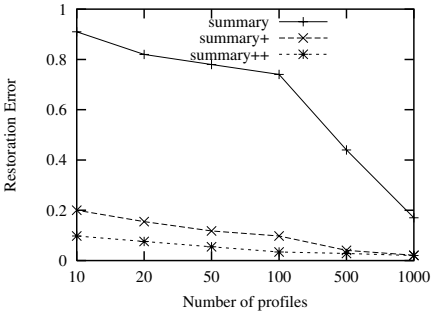

Fig. 5. Restoration Error for BMS-Webview1

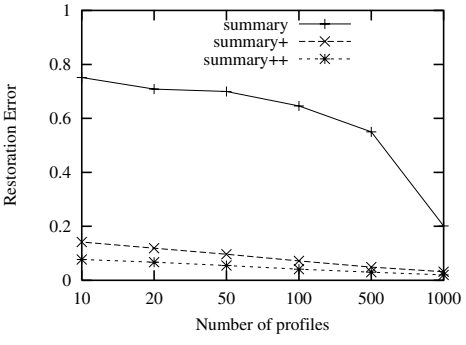

Fig. 6. Restoration Error for BMS-POS

with reasonable good quality: the average restoration error is 0.11 for *summary+* and 0.05 for *summary++* while it is 0.82 for *summary*. The restoration error is only 0.20 for *summary+* and 0.09 for *summary++* compared with 0.92 for algorithm *summary* when we cluster patterns into 10 profiles.

We generated a set of 6646 frequent closed patterns from BMS-POS by setting minimum support at 0.4% and the maximum length of frequent patterns is 6. Figure 6 shows the average restoration error over the set of closed frequent patterns. Both *summary+* and *summary++* outperform the *summary* by several times in terms of restoration errors. The *summary++* can reduce the restoration error of *summary+* by 50%. The 6646 patterns can be successfully summarized into 50 profiles with reasonable good quality: the average restoration error is less than 0.1 (0.096 for *summary+* and 0.053 for *summary++*). The restoration error is only 0.142 for *summary+* and 0.077 for *summary++* compared with 0.752 for the *summary* when we summarize patterns into 10 profiles.

5.2 The Effect of Probability Model

In this subsection, we try to distinguish the effect of profile model on the quality of support restoration (Section 3) and the quality of the clustering results (Section 4) although the support restoration is closely related to clustering quality. We group the set

of frequent patterns based on the model of algorithm *summary*, then build our probability model in each profile by accessing the dataset and use the Equation 5 to restore the support information. This is a hybrid of *summary* and *summary+*. We have found that it will yield qualitatively similar results if we make a hybrid from *summary* and *summary++*.

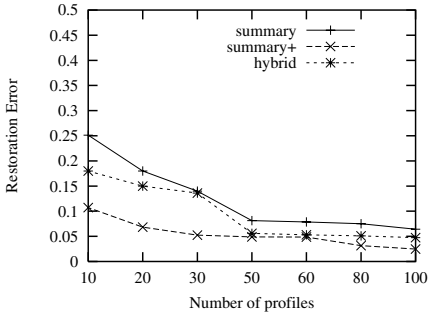

Fig. 7. Mushroom

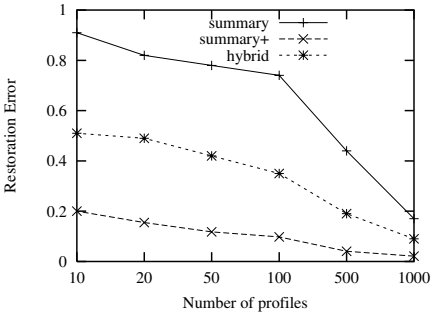

Fig. 8. BMS-Webview1

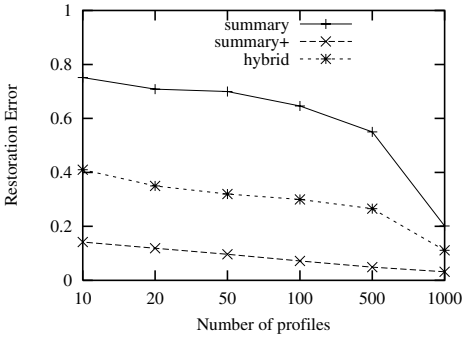

Fig. 9. BMS-POS

Figures 7-9 show the comparisons of the hybrid method with the *summary* and *summary+* on three datasets. We use the same setting for three datasets as that in the last subsection. The line of hybrid method lies between the those of *summary* and *summary+*. This means that the support estimation methods based on the probability model in Section 3 alone cannot achieve the improvement of *summary+* over *summary*. This implies that the clustering based on the probability model in Section 3 results in better clusters of frequent patterns than [19]. In other words, the probability model of this paper characterizes the frequent patterns better than the model in [19] does, and thus the calculated distances between clusters based on such model are more effective.

As a summary, both *summary+* and *summary++* greatly outperform *summary*. *Summary++* usually can improve *summary+* by 50% in terms of restoration accuracy while *summary+* is simple and is fast in terms of computation.

6 Related Work

Frequent pattern mining has received much attention in the past decade. Many frequent pattern mining algorithms have been proposed (e.g. [3, 9]). The number of frequent patterns can be very large and many of these frequent patterns may be redundant. To reduce the frequent patterns to a compact size, mining frequent closed patterns (e.g. [13]) has been proposed, which is a lossless compression of frequent patterns. Lossless here means that all frequent patterns together with their supports can be recovered from the closed patterns. Another lossless compression patterns of frequent patterns has been proposed to mine non-derivable frequent patterns [5].

There have been some other proposals to mine a subset of all frequent patterns. These methods are lossy in the sense that not all information about frequent patterns can be recovered. Maximal frequent pattern (e.g. [2]) is one of the most typical concepts in this category. In maximal frequent pattern, all frequent sub-patterns are removed, and thus the number of patterns is greatly reduced. However, the support information of all the sub-patterns are lost and maximal patterns can be large in number. Other lossy compression proposals include error-tolerant patterns [20], top-k patterns [10], condensed frequent pattern base [16], compressed frequent pattern sets [18].

The closest work to our study is the k approximation frequent sets [1] and k summarizing profiles [19]. The k approximation frequent sets use k frequent itemsets to cover a collection of frequent itemsets while trying to minimize the negative positive patterns; the set of frequent patterns can be deducted from the k approximate frequent sets. However, the support information of patterns is lost. The k summarizing profiles use a simple independence probability model to represent a set of patterns and cluster the profiles. One salient feature of k summarizing profiles is that support information can be restored relatively accurately. In this paper, we improve the probability model to represent model and propose new methods to derive support using our proposed probability model. Our profile model is also related to the probabilistic models developed in [14] for query approximation, where frequent patterns and their supports are used to estimate query selectivity, and independence model and Chow-Liu tree model are used for query approximation.

There are also many proposals about mining interesting rules with various interestingness measures [12]: post-processing to remove uninteresting rules [11], mining interesting rules [4], mining non-redundant association rules[21], and mining top-k covering rule groups [7]. There studies are very different from the pattern summarization.

7 Conclusion and Discussions

In this paper, we have revisited the pattern summarization problem. We proposed a probability model to represent a set of frequent patterns and two methods of estimating the support of a pattern from the model. With the model, we can arrange all patterns into a set of (hierarchial) clusters and thus users can explore the patterns in a top-down manner. Our methods can successfully summarize patterns into tens of profiles while the supports of patterns can be recovered reasonably accurately. Empirical studies show that our method can achieve accurate summarization in real-life data and the supports of frequent patterns can be restored more accurately than the previous method.

In future, we plan to investigate the application of other distance measure, such as Jensen-Shannon divergence, to compute the similarity of profiles and other clustering methods, such as the co-clustering based on information theory. We also plan to apply for clustering algorithms and co-clustering algorithm to the transaction database directly to obtain some clusters of items (one item could be in multiple clusters) and build a probability model for each cluster. It would be interesting to investigate the quality of such clusters to estimate the supports of frequent patterns.

References

1. F. Afrati, A. Gionis, and H. Mannila. Approximating a collection of frequent sets. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 12–19, 2004.
2. R. Agarwal, C. Aggarwal, and V. V. V. Prasad. Depth first generation of long patterns. In *Proc. of KDD*, 2000.
3. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 1994 Int. Conf. Very Large Data Bases (VLDB'94)*, pages 487–499, Sept. 1994.
4. R. J. Bayardo and R. Agrawal. Mining the most intersting rules. In *Proc. of ACM SIGKDD*, 1999.
5. T. Calders and B. Goethals. Mining all non-derivable frequent itemsets. In *Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*, 2002.
6. C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467, 1968.
7. G. Cong, K.-L. Tan, A. K. H. Tung, and X. Xu. Mining top-k covering rule groups for gene expression data. In *Proceedings of the ACM SIGMOD international conference on Management of data*, 2005.
8. T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. The MIT Press, 1990.
9. J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. 2000 ACM-SIGMOD Int. Conf. Management of Data (SIGMOD'00)*, 2000.
10. J. Han, J. Wang, Y. Lu, and P. Tzvetkov. Mining top.k frequent closed patterns without minimum support. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM)*, 2002.
11. B. Liu, W. Hsu, and Y. Ma. Pruning and summarizing the discovered associations. In *ACM KDD*, 1999.
12. E. R. Omiecinski. Alternative interest measures for mining associations in databases. *IEEE Transactions on Knowledge and Data Engineering*, 15(1):57–69, 2003.
13. N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal. Discovering frequent closed itemsets for association rules. In *Proc. 7th Int. Conf. Database Theory (ICDT'99)*, Jan. 1999.
14. D. Pavlov, H. Mannila, and P. Smyth. Beyond independence: Probabilistic models for query approximation on binary transaction data. *IEEE Transactions on Knowledge and Data Engineering*, 15(6):1409–1421, 2003.
15. J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.
16. J. Pei, G. Dong, W. Zou, and J. Han. Mining condensed frequent-pattern bases. *Knowl. Inf. Syst.*, 6(5):570–594, 2004.
17. J. Wang and G. Karypis. SUMMARY: Efficiently summarizing transactions for clustering. In *Proceedings of the 2004 IEEE International Conference on Data Mining (ICDM)*, 2004.
18. D. Xin, J. Han, X. Yan, and H. Cheng. Mining compressed frequent-pattern sets. In *VLDB*, 2005.

19. X. Yan, H. Cheng, J. Han, and D. Xin. Summarizing itemset patterns: a profile-based approach. In *Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, 2005.
20. C. Yang, U. Fayyad, and P. S. Bradley. Efficient discovery of error-tolerant frequent itemsets in high dimensions. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001.
21. M. Zaki. Generating non-redundant association rules. In *Proc. 2000 Int. Conf. Knowledge Discovery and Data Mining (KDD'00)*, 2000.
22. Z. Zheng, R. Kohavi, and L. Mason. Real world performance of association rule algorithms. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001.