Bounding and Estimating Association Rule Support from Clusters on Binary Data

Carlos Ordonez, Kai Zhao, Zhibo Chen
University of Houston
Department of Computer Science
Houston, TX 77204, USA

***Abstract*—The theoretical relationship between association rules and machine learning techniques needs to be studied in more depth. This article studies the use of clustering as a model for association rule mining. The clustering model is exploited to bound and estimate association rule support and confidence. We first study the efficient computation of the clustering model with K-means; we show the sufficient statistics for clustering on binary data sets is the linear sum of points. We then prove itemset support can be bounded and estimated from the model. Finally, we show support bounds fulfill the set downward closure property. Experiments study model accuracy and algorithm speed, paying particular attention to error behavior in support estimation. Given a sufficiently large number of clusters, the model becomes fairly accurate to approximate support. However, as the minimum support threshold decreases accuracy also decreases. The model is fairly accurate to discover a large fraction of frequent itemsets at different support levels. The model is compared against a traditional association rule algorithm to mine frequent itemsets, exhibiting better performance at low support levels. Time complexity to compute the binary cluster model is linear on data set size, whereas the dimensionality of transaction data sets has marginal impact on time.**

I. Introduction

Association rules [2] are a very popular data mining technique [15], [17], which have produced a significant body of research. There is research on efficient algorithms to discover association rules [3], [32], [16], [38], using item taxonomies [35], creating itemset summaries [8], incorporating constraints [40], creating summaries [10], [20] and applying them in different domains [9], [11], [29], among other topics. Unfortunately, association rules lack a basic model to understand their relationship to properties from the data set. In this work, we defend the idea of using clustering [7], [26], [12] as a model to discover association rules.

Association rules is a well studied topic, but most research has proposed efficient algorithms [24], [16], [32] or variants like quantitative [36], constrained [25], [40], taxonomy-based [35], among others. Studying their relationship to classical machine learning techniques has received less attention. Using a model to discover them, finding ways to summarize them, proposing metrics to approximate them or explaining their validity using alternative techniques are aspects that have not received enough attention. Our point of view is that clusters represent a more fundamental pattern than association rules, from which we can generate them. The mathematical relationship between clustering and association rules is important and interesting because clustering, on one hand, looks for global patterns producing disjoint subsets of points, whereas association rules uncover many local patterns referring to overlapping subsets of points. That is, both techniques have different goals.

This is a summary of our contributions. We first introduce a one-pass K-means clustering algorithm based on simplified sufficient statistics for binary data to build a model that summarizes a set of transactions as clusters. We prove clusters can be used to bound and approximate association rule metrics. Based on the clustering model three association metrics are introduced: lower, upper and average support. These new metrics provide bounds and an estimation for association support. Also, our metrics exhibit the subset downward closure property, which allows efficient bottom-up search algorithm that can work in main memory. We derive three metrics for association rule confidence reusing support bounds: lower, upper and average confidence. We perform an extensive experimental with real and synthetic data sets. We study error behavior varying the number of clusters and accuracy to discover a large set of frequent itemsets at different support thresholds.

This is an outline of the article. Basic definitions on clustering and association rules are presented in Section II. Section III explains how to exploit a clustering model to bound an estimate itemset support. Section IV discusses experiments studying error on support estimation, speed and scalability. Closely related work is presented in Section V. Section VI concludes the article.

II. Definitions

We give a unified set of definitions for association rules [2], [3], [32] and clustering [7], [28]. Items act as dimension subscripts for matrix manipulation.

A. Association Rules

Let $D = \{T_1, \ldots, T_n\}$ be a set of n transactions on d items, where $|T_i| << d$ (crucial to develop an efficient distance computation [26]) and $\mathcal{I} = \{1, 2, \ldots, d\}$ (each integer

Copyright 2008 IEEE. Published in Foundations on Data Mining (FDM, ICDM 2008 Workshop), 2008. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works, must be obtained from the IEEE.
http://doi.ieeecomputersociety.org/10.1109/ICDMW.2008.47

identifies one item). Each subset X of items s.t. $X \subseteq \mathcal{I}$ is called an *itemset*. An itemset X containing p items is called a p-itemset: $X = \{\mathrm{i}_1, \mathrm{i}_2, \ldots, \mathrm{i}_p\}$, where $\mathrm{i}_l \in \mathcal{I}$. The probability of appearance for one itemset $X \subseteq \mathcal{I}$, is called support and it is defined as $s(X) = |\{T_i | X \subseteq T_i\}|/n$. That is, the fraction of transactions in D that contain X. In an association rule $X \Rightarrow Y$, $X, Y \subset \mathcal{I}$, and $X \cap Y = \emptyset$. (X is called the antecedent and Y is called the consequent). $X \Rightarrow Y$ has a measure of reliability called confidence, given by $c(X \Rightarrow Y) = s(X \cup Y)/s(X)$. The standard association problem is to find all itemsets that have support above a minimum support threshold τ. The standard problem of mining association rules is to find all rules $X \Rightarrow Y$ s.t. $s(X \Rightarrow Y) \geq \tau$ and $c(X \Rightarrow Y) \geq \psi$.

B. Clustering

In the context of clustering D is analyzed as a binary data set with d dimensions. K-means partitions the data set into k disjoint groups based on a Euclidean distance similarity metric. The clustering model consists of matrices W, C, R, which contain the weights, the centroids (means) and the radiuses (variances), respectively for each cluster and a partition of D into k subsets, where k is user-specified. Matrices C and R are $d \times k$ and W is a $k \times 1$ matrix. To refer to to the jth cluster we use the j subscript (i.e. C_j, R_j). Therefore, C_j is the jth cluster centroid, R_j is the jth (diagonal) variance matrix and W_j is the weight of cluster j. Subscripts are used as follows. Subscript i is used to scan transactions: $i \in \{1, 2, \ldots, n\}$. Notice i alone is a subscript to index transactions, but i_j is item j. Subscript j is used to refer to one cluster: $j \in \{1, 2, \ldots, k\}$. Finally, subscript h is used for dimension: $h \in \{1, 2, \ldots, d\}$. Let $D_1, D_2, \ldots, D_k$ be the k disjoint subsets from D induced by clusters s.t. $D_j \cap D_g = \emptyset$ for $j \neq g$. The $diag[]$ is a transformation operator to transform a vector into a diagonal matrix or vice versa. The T superscript indicates vector and matrix transposition.

Example *2.1:* In Figure 1 we present a small input database D, where $d = 5, n = 8$. Figure 2 presents a clustering model where $k = 2$. Finally, Figure 3 presents an intuitive model summary. Items are grouped according to their cluster centroid value. The last column shows a basic itemset represented by each cluster, which can be used to derive itemset subsets. Thus the clustering summary can derive all frequent itemsets that can be obtained from D. Parameter λ is used to filter out long itemsets obtained from each cluster; it will be explained in more detail in Section III.

III. Bounds on Support Based on Clusters

Clustering is used as a statistical model for association rules. In this section we introduce an efficient algorithm to cluster binary data sets and a family of association rules metrics based on a clustering model.

A. Clustering Binary Data Sets

Since we are trying to use clusters as a model to discover associations the first task is clustering transactions. K-means [7], [31], [23], [17] is used to cluster transactions. From a scalability point of view, It is feasible to derive incremental K-means clustering versions [26] that can get a good solution in one pass or a few passes over a large data set.

T_i	Items
1	4 5
2	3 4
3	1 2
4	2
5	1 2
6	3 4 5
7	1
8	3 4

T_i	1 2 3 4 5
1	0 0 0 1 1
2	0 0 1 1 0
3	1 1 0 0 0
4	0 1 0 0 0
5	1 1 0 0 0
6	0 0 1 1 1
7	1 0 0 0 0
8	0 0 1 1 0

Fig. 1. Input transactions $d = 5, n = 8$.

$$W = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} C = \begin{bmatrix} 0.00 & 0.75 \\ 0.00 & 0.75 \\ 0.75 & 0.00 \\ 1.00 & 0.00 \\ 0.50 & 0.00 \end{bmatrix}$$

Fig. 2. Clustering model; $d = 5, k = 2$.

The K-means algorithm used in this work is an improvement of the incremental K-means algorithm for binary streams [26]. In this section we present the main features that make it suitable to cluster binary data sets and introduce useful theoretical results. K-means is significantly accelerated with sparse distance computation, sparse matrix addition and simplified sufficient statistics.

The K-means algorithm [26], like other scalable clustering algorithms [7], [44], exploits sufficient statistics, which are summaries of $D_1, D_2, \ldots, D_k$ represented by matrices L, Q, N that contain k sums of points, k sums of squared points and number of points per cluster, respectively. L is a $d \times k$ matrix, such that each column L_j is computed as $L_j = \sum_{x_i \in D_j} x_i$. Each column of Q is a diagonal matrix Q_j: $Q_j = \sum_{x_i \in D_j} x_i x_i^T$. Finally, matrix N is $k \times 1$ which counts the number of points per cluster (i.e. $N_j = |D_j|$). The update formulas for W, C, R based on sufficient statistics are:

$$W_j = \frac{N_j}{n} \tag{1}$$

$$C_j = \frac{1}{N_j} L_j \tag{2}$$

$$R_j = \frac{1}{N_j} Q_j - \frac{1}{N_j^2} L_j L_j^T \tag{3}$$

		C_j range for items			Itemsets
j	W_j	0.75-1	0.5-0.75	0.25-0.5	at $\lambda = 0.5$
1	0.5	3 4	5		3 4 5
2	0.5	1 2			1 2

Fig. 3. Clusters summary and itemsets.

The sufficient statistics can be simplified for clustering binary data. In fact, they are simpler than the sufficient statistics required for clustering numeric data.

Lemma *3.1:* Let $D_1, D_2, \ldots, D_k$ be k subsets representing a partition of D. Then the sufficient statistics required for computing C, R, W are only N and L because $L = Q$.

Proof: Based on the fact that x_i is a binary vector: $\sum x_i = \sum x_i x_i^T$. Therefore, $L = Q$. ■

This result states that we only need to update the model based on Equation (1) and Equation (2). That is, matrix R_j can be obtained from C_j:

$$R_j = diag[C_j] - C_j C_j^T \tag{4}$$

More importantly, storage space for clustering results is reduced to one half. Therefore, matrix R is not needed to estimate bounds for support and confidence. We can now derive a simple criterion to evaluate cluster quality.

Theorem *3.1:* R_{lj} has a minimum if and only if $C_{lj} = 0$ or $C_{lj} = 1$. Also, R_{lj} has a maximum if and only if $C_{lj} = 1/2$.

Proof: The result follows from $R_j = diag[C_j] - C_j C_j^T$ and the first derivative conditions for entry R_{lj} to have a maximum. ■

Theorem 3.1 states that it is best C_j entries are close to either 0 or 1, with the ideal case when $C_{lj} = 0$ or $C_{lj} = 1$.

B. One Pass K-means for Binary Data

When D is a sparse matrix and d is high, Euclidean distance is expensive to compute. In a typical binary (transaction) data set only a few dimensions have non-zero values. Therefore, we precompute a distance from every C_j to the null vector $\bar{0}$. To that purpose, we define the following k-dimensional vector: $\Delta_j = \delta(\bar{0}, C_j)$. Let $\delta_{ij} = \delta(x_i, C_j)$. Then

$$\delta_{ij} = \Delta_j + \sum_{l=1,(x_i)_l \neq 0}^{d} ((x_i)_l - C_{lj})^2 - C_{lj}^2. \tag{5}$$

This optimized distance computation decreases time complexity, but it does not affect quality of results. A similar idea is applied to update L, Q matrices in a sparse fashion after a transaction cluster membership is determined. Sparse matrix addition updates only the closest cluster dimensions where the point dimension value is 1.

Based on the improvements described in the previous section we present a one pass K-means algorithm. This is a high-level description. The input is the set of transactions $D = \{T_1, T_2, \ldots, T_n\}$. and k, the desired number of clusters as defined in Section II. The output is $\Theta = \{W, C\}$, describing the model and a partitioning of D into k subsets. A constant α is used to seed C based on d and k. The global statistics μ and Σ can be quickly computed from a sample. Standard deviations are computed as $\sigma_{ll} = \sqrt{\Sigma_{ll}}$. The cluster membership step is executed for every transaction (n times). Since this is the most frequently executed step the optimization for sparse matrix operations is essential to achieve good performance. These sparse operations include sparse distance computation for δ and sparse matrix addition to update L. δ_{ij} is efficiently computed using Δ_j. The M step is periodically executed every n/γ transactions (γ times). W, C are updated using Eq. (1) and Eq. (2). Dimensions (items) are ranked within each cluster by their value in C_j to make output easier to understand.

```
Input: $D = \{T_1, T_2, \ldots, T_n\}$ and $k$.
Output $\Theta = \{W, C\}$
$\alpha \leftarrow 1/(dk), \gamma \leftarrow log(n)$
FOR $j = 1$ TO $k$ DO                    /* Initialize */
   $C_j \leftarrow \mu \pm \alpha r$ diag$[\sigma]$, $W_j \leftarrow 1/k$
   $\Delta_j = \delta(\bar{0}, C_j) = C_j^T C_j$
   $L_j \leftarrow C_j,\ N_j \leftarrow 1$
ENDFOR

FOR $i = 1$ TO $n$ DO
  FOR $j = 1$ TO $k$ DO          /* distances */
    $\delta_j \leftarrow \delta(T_i, C_j)$      /* sparse operation */
  ENDFOR
  Let $J$ be s.t. $\delta_J \leq \delta_j, j = 1 \ldots k$
  $L_J \leftarrow M_J + x_i$ /* sparse operation */
  $N_J \leftarrow N_J + 1$
  IF( $i$ mod $(n/\gamma) = 0$ ) THEN    /* update model */
     FOR $j = 1$ TO $k$ DO
       $C_j \leftarrow M_j / N_j$
       $W_j \leftarrow N_j / \sum_{J=1}^{k} N_J$
       $\Delta_j \leftarrow C_j^T C_j$
     ENDFOR
  ENDIF
ENDFOR
```

Fig. 4. One pass K-means for binary data.

C. Bounds on Support

The following theoretical results are important to use the model to obtain bounds on association support. The support of an itemset in one cluster ($s(X, D_j)$) must be less or equal than the minimum support of any item, as shown in [32].

Lemma *3.2:* Let D_j be one cluster from D. Let $X = \{\mathrm{i}_1, \mathrm{i}_2, \ldots, \mathrm{i}_p\}$ be some p-itemset. Let C_j be the mean of transactions that belong to cluster j. Then $s(X, D_j) \leq \min((C_j)_{\mathrm{i}_1}, (C_j)_{\mathrm{i}_2}, \ldots, (C_j)_{\mathrm{i}_p})$.

Proof: (sketch) Induction on p. ■

The following theorem links clustering results with associations. Let the upper bound of $s(X)$ be defined as follows:

$$u(X) = \sum_{j=1}^{k} W_j \min((C_j)_{\mathrm{i}_1}, (C_j)_{\mathrm{i}_2}, \ldots, (C_j)_{\mathrm{i}_p}). \tag{6}$$

Theorem *3.2:* Let D be a set of n transactions. Let C, W be the matrices containing the means and weights respectively on some clustering of D into k subsets of transactions $D_1, D_2, \ldots, D_k$. Let X be any p-itemset given by $X = \{\mathrm{i}_1, \mathrm{i}_2, \ldots, \mathrm{i}_p\}$. Then $s(X) \leq u(X)$.

Proof: (sketch) By Lemma 3.2 we can sum k inequalities corresponding to each subset D_j, $s(X, D_j) \leq u(X, D_j)$ to get $s(X) \leq u(X)$ on D. ■

Theorem 3.2 asserts C and W can be used to mine all potential frequent itemsets, thereby eliminating the need to scan D. Nevertheless, the theorem does not guarantee that only true frequent itemsets are above the minimum support threshold. We may find some false-positive itemsets whose exact support is actually lower than that indicated by the model. The following result, based on Theorem 3.2, is applied to prune the lattice search space if associations are being mined from the clustering model using $u()$. Now we introduce a lower bound for the support of an association.

$$l(X) = \sum_{j=1}^{k} W_j \max(0, 1 + \sum_{l=1}^{p} ((C_j)_{\mathrm{i}_l} - 1)), \tag{7}$$

where max() is used to get a non-negative lower bound per cluster.

Theorem *3.3:* Let $X, C, W, \{D_1, \ldots, D_k\}$ be as stated in Theorem 3.2. Let X be a p-itemset $X = \{\mathrm{i}_1, \mathrm{i}_2, \ldots, \mathrm{i}_p\}$. Then $l(X) \leq s(X)$.

Proof: (sketch) The sum in Equation 7 computes a lower bound on the fraction of transactions that do not contain X for cluster j. ■

Since $u()$ and $l()$ are guaranteed bounds on the exact support of any association their average is a good estimator:

$$a(X) = \frac{l(X) + u(X)}{2}. \tag{8}$$

D. Downward Closure Property

The following result proves our metrics preserve the downward closure property (e.g. a subset of a frequent itemset is frequent).

Theorem *3.4:* Let X be a p-itemset and let $Y \subset X$, $Y = \{\mathrm{h}_1, \ldots, \mathrm{h}_q\}$, $q < p$. Then: (1) $l(X) \leq l(Y)$, (2) $u(X) \leq u(Y)$, (3) $a(X) \leq a(Y)$.

Proof: The proof follows by considering the extra items in X, which may decrease the bounds on Y. ■

Theorem 3.4 guarantees a *complete* set of associations with respect to any of the metrics using the standard bottom-up search algorithm. This result generalizes bottom-up search algorithms to work with a clustering model instead of the data set.

As explained above, upper support ($u()$) is safe in the sense that it produces a superset of all frequent itemsets, but it may generate infrequent (false positive) itemsets. In general, a subset of C_j entries close to zero indicate itemsets that are unlikely to be frequent. To solve such limitation, we propose to use a λ parameter which is used to filter out itemsets unlikely to be above τ (i.e. not frequent): clusters whose upper support for one specific itemset fall below τ do not contribute to the support estimation used for pruning. Therefore, we get an adjusted upper support for pruning below τ. The spectrum for λ is [0,1]. At one extreme $\lambda = 0$ does not eliminate any itemset: all frequent itemsets are discovered, but there are additional itemsets whose support is actually below the τ threshold. At the other extreme, $\lambda = 1$ allows only dimensions with zero variance in each cluster to participate in itemset generation; in this case all itemsets that are generated are frequent (there are zero false positives), but there may be many itemsets whose support was actually above the τ threshold (there are false negatives).

TABLE I
DATA SETS.

Data set	type	d	n
Chess	real	75	3196
Mushroom	real	127	8124
Votes	real	17	435
T10I4D100k	synthetic	100	100k
T10I4D1M	synthetic	100	1000k

IV. Experimental Evaluation

We first discuss our experimental setup and data sets. Experiments study two major aspects. First, we study relative error on support estimation and model accuracy to discover frequent itemsets varying the number of clusters and the minimum support threshold. Second, we evaluate speed and scalability.

A. Setup

Experiments were performed on a workstation with a 2.4 GHz CPU, 2 GB of memory and 160 GB on disk. The algorithms were programmed in the C++ language. The data sets were stored as text files. The clustering model matrices were stored on a binary file after their computation. The model was loaded into main memory before mining frequent itemsets. Each experiment was repeated five times and average measurements are reported. In general, elapsed times are reported in seconds.

B. Data Sets

Experiments use real and synthetic data sets. The real data sets were obtained from the UCI Machine Learning Repository [4] and are widely used in the research literature on association rule mining. Synthetic (transaction) data sets were created running the well-known IBM transaction data generator [3]. All data sets were transformed and converted to transaction format (itemset representation), as required by our proposal. Table I provides a summary of data sets.

We now explain how synthetic data sets were generated with the IBM transaction data generator [3]. This generator has seven parameters (indicating our notation in parenthesis): number of transactions D (n), number of items (d), number of patterns P (frequent itemsets), average transaction length T, average pattern length I, average correlation $corr$ and average rule confidence $conf$. Test files are named after the parameters with which they were created. The standard way [3], [15] is to use T, I and D to label files since those are the three most common parameters to change and the ones which play a more important role in performance (e.g. T10I4D100k). The transaction files we used had defaults D=1M (n=1000k), $d = 100$, T=10, I=4, P=100, $corr = 0.75$ and $conf = 0.75$.

C. Clustering Model Accuracy

Deafult Parameters Setting

Each data set has a different optimal setting for k. Therefore, there is no default value for k. On the other hand, we set $\lambda = 0.2$; Recall λ is used to filter out itemsets unlikely to be frequent.

Relative error on support was used to measure accuracy. For an itemset X it was computed as $e_s(X) = |a(X) - s(X)|/|s(X)|$, where average support $a(X)$ is obtained from the clustering model and $s(X)$ is the exact support value, as computed from D.

We focus on analyzing error for support estimation. The problem of choosing the best k is called model selection [23], [17]. A low k produces a simple, but probably inaccurate model, a high k will produce a more complex, but accurate model. We made a few runs to determine a good k for the real data sets, considering that when k is too high some clusters tend to have almost zero points. Therefore, we set k high enough so that relative error reached less than 10% and we did not get zero-weight clusters. In general, there was a slight variation in the cluster model quality from run to run. We selected the run that produced best clustering results out of five runs in order to estimate support bounds. This is a reasonable approach because in practice a data mining user picks the best model.

Error varying the number of clusters

Figure 5 analyzes error behavior varying k. The left graph shows error behavior for real data sets at $\tau = 0.2$, whereas the right graph analyzes error for the synthetic data set for two τ values. We discuss results for real data sets. As can be seen error rapidly decreases as k grows. The trend indicates asymptotic behavior. Despite the difference in data set sizes and dimensionality in all cases error goes below 5% when k reaches 80. On the other hand, error is high when k is low. We now turn our attention to the synthetic data set, where we built models with much higher k. In this case error shows a slower rate of decrease. The trend also seems asymptotic. We can also see there is a clear gap between $\tau = 0.05$ and $\tau = 0.10$; error roughly grows 100% when τ decreases by 50%. Interestingly enough, results are much better for real data sets than for the synthetic data set.

Error decreasing minimum support

Figure 6 shows error growth as τ decreases. The left graph analyzes error for real data sets. We can see error growth shows different behavior for each data set. Error growth is slow for the Votes data set. Error grows almost linearly for the Chess data set. Error grows fast for the Mushroom data set. The trend indicates we need to build more accurate models with higher k for Mushroom, probably not for Chess and not necessary for the Votes data set. The right graph analyzes error growth for the synthetic data set. The first model at $k = 800$ is twice as accurate as $k = 400$. We can see error grows linearly for high τ values, but it start growing faster at $\tau = 0.1$. The trend indicates the growth is not linear, but it does not seem exponential.

D. Comparing Speed and Scalability

We first compare our proposal versus the standard algorithm to mine association rules. We then study time complexity and scalability.

TABLE II
COMPARING MODEL AND A-PRIORI.

τ	from model	clustering+model	A-priori
0.20	1	1672	24
0.15	1	1672	43
0.10	1	1672	156
0.05	3	1674	645
0.02	11	1683	3347
0.01	36	1708	14806

Comparing clustering and A-priori

Table II compares the efficiency of the model with the standard A-priori algorithm. We must stress our proposal does not intend to substitute fast association rule algorithms [32], [16], but we include these comparisons to provide a relative performance benchmark. We used the synthetic data with $n = 1M$. The clustering model used had the number of clusters set to $k = 100$. These times include the time to compute exact support on a final pass. The first column varies τ (the minimum support threshold). The second column shows the time to discover frequent itemsets from the model, excluding the time to compute the model. The third column adds the time to compute the clustering model and the time to mine frequent itemsets. Finally, the fourth column shows the time for the traditional algorithm. As can be seen, the clustering model represents an efficient mechanism to produce all frequent itemsets, assuming the model is already tuned and stored. That is, we assume the model is computed a few times or even once. The second column shows clustering the data set takes most of the time. In this case, the standard algorithm is faster at high support levels, but the model becomes faster at low support levels. Notice the third column represents a pessimistic case in which the model is recomputed every time. Finally, we can see the A-priori algorithm suffers scalability problems due to the exponential growth of patterns. The basic reason the model is faster is because it uncovers long itemsets and it is efficiently manipulated in main memory.

Time Complexity and Speed

We now evaluate scalability and speed with large, high dimensional, data sets to only compute the models, as shown in Figure 7. The plotted times include the time to store models on disk, but exclude the time to mine frequent itemsets. We experimentally prove: (1) Time complexity to compute models is linear on data set size. (2) Sparse vector and matrix computations yield efficient algorithms, whose accuracy was studied before. (3) Dimensionality has minimal impact on speed, assuming average transaction size T is small. Transactions are clustered with Incremental K-means [26], introduced on Section III. Large transaction files were created with the IBM synthetic data generator [3] having defaults n = 1M, T=10, I=4.

Figure 7 shows time complexity to compute the clustering model. The first plot, on the left, shows time growth to build the clustering with a data set with one million records (T10I4D1M). As can be seen times grow linearly as n increases, highlighting the algorithms efficiency. On the other hand, notice d has marginal impact on time when

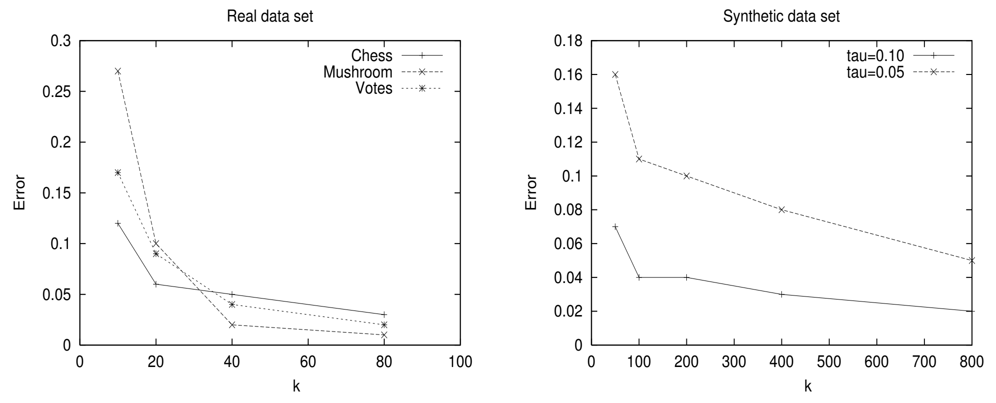

Fig. 5. Support estimation error increasing k.

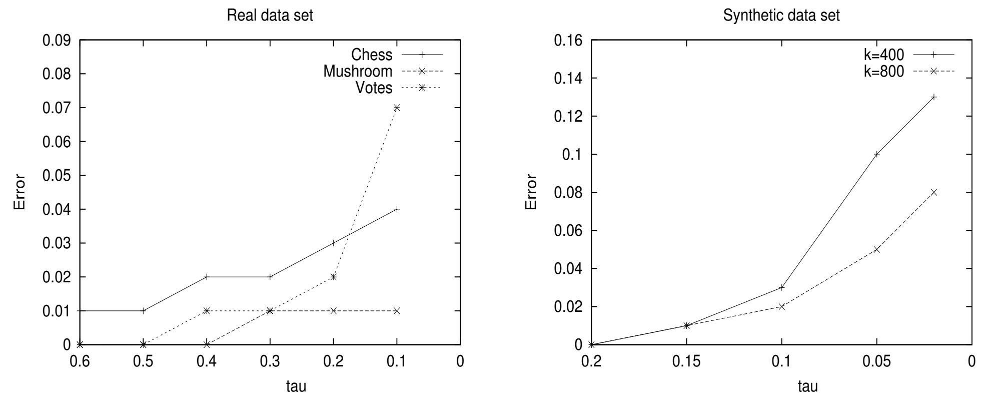

Fig. 6. Support estimation error decreasing τ.

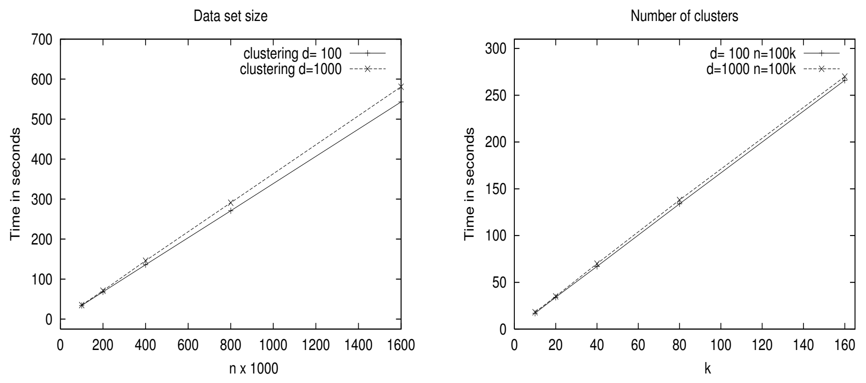

Fig. 7. Time complexity for clustering large data sets with K-means.

it is increased 10-fold on both models due to optimized sparse matrix computations. The second plot, on the right, in Figure 7 shows time complexity to compute clustering models increasing k on T10I4D100k. Remember k is the main parameter to control support estimation accuracy. In a similar manner to the previous experiments, times are plotted for two high dimensionalities, $d = 100$ and $d = 1,000$. As can be seen time complexity is linear on k, whereas time is practically independent from d. Therefore, our methods are competitive both on accuracy and time performance.

E. Summary

The clustering model provides several advantages. It is a descriptive model of the data set. It enables support estimation and it can be processed in main memory. It requires the user to specify the number of clusters as main input parameter, but it does not require support thresholds. More importantly, clusters can help discovering long itemsets appearing at very low support levels.

We now discuss accuracy. In general, the number of clusters is the most important model characteristic to improve accuracy. A higher number of clusters generally produces tighter bounds and therefore more accurate support estimations. The clustering model quality has a direct relationship to support estimation error. We introduced a parameter to improve accuracy when mining frequent itemsets from the model; this parameter eliminate "spurious" itemsets unlikely to be frequent. The clustering model is reasonably accurate on a wide spectrum of support values, but accuracy decreases as support decreases.

We conclude with a summary on time complexity and efficiency. When the clustering model is available, it is a significantly faster mechanism than the A-priori algorithm to search for frequent itemsets. Decreasing support impacts performance due to the rapid combinatorial growth on the number of itemsets. In general, the clustering model is much smaller than a large data set: $O(dk) < O(dn)$. A clustering model can be computed in linear time with respect to data set size. In typical transaction data sets dimensionality has marginal impact on time.

V. Related Work

There is a lot of research work on scalable clustering [1], [30], [28] and efficient association mining [24], [16], [40], but little has been done finding relationships between association rules and other data mining techniques. Sufficient statistics are essential to accelerate clustering [7], [30], [28]. Clustering binary data is related to clustering categorical data and binary streams [26]. The k-modes algorithm is proposed in [19]; this algorithm is a variant of K-means, but using only frequency counting on 1/1 matches. ROCK is an algorithm that groups points according to their common neighbors (links) in a hierarchical manner [14]. CACTUS is a graph-based algorithm that clusters frequent categorical values using point summaries. These approaches are different from ours since they are not distance-based. Also, ROCK is a hierarchical algorithm. One interesting aspect discussed in [14] is the error propagation when using a distance-based algorithm to cluster binary data in a hierarchical manner. Nevertheless, K-means is not hierarchical. Using improved computations for text clustering, given the sparse nature of matrices, has been used before [6]. There is criticism on using distance similarity metrics for binary data [12], but in our case we have proven K-means can provide reasonable results by filtering out most itemsets which are probably infrequent.

Research on association rules is extensive [15]. Most approaches concentrate on speeding up the association generation phase [16], [1]. Some of them use data structures that can help frequency counting for itemsets like the hash-tree, the FP-tree [16] or heaps [18]. Others resort to statistical techniques like sampling [38], statistical pruning [24]. In [34] global association support is bounded and approximated for data streams with the support of recent and old itemsets; this approach relies on discrete algorithms for efficient frequency computation instead of using machine learning models like our proposal. Our intent is not to beat those more efficient algorithms, but to show association rules can be mined from a clustering model instead of the transaction data set. In [5] it is shown that according to several proposed interest metrics the most interesting rules tend to be close to a support/confidence border. Reference [43] proves several instances of mining maximal frequent itemsets (a constrained frequent itemset search) are NP-hard and they are at least #P-hard, meaning they will remain intractable even if P=NP. This work gives evidence it is not a good idea to mine all frequent itemsets above a support threshold since the output size is combinatorial. In [13] the authors derive a bound on the number of candidate itemsets given the current set of frequent itemsets, when using a level-wise algorithm. Covers and bases [37], [21] are an alternative to summarize association rules using a combinatorial approach instead of a model. Clusters have some resemblance to bases in the sense that each cluster can be used to derive all subsets from a maximal itemset. The model represents an approximate cover for all potential associations.

We now discuss closely related work on establishing relationships between association rules and other data mining techniques. Preliminary results on using clusters to get lower and upper bounds for support is given in [27]. In general, there is a tradeoff between rules with high support and rules with high confidence [33]; this work proposes an algorithm that mines the best rules under a Bayesian model. There has been work on clustering transactions from itemsets [41]. However, this approach goes in the opposite direction: it first mines associations and from them tries to get clusters. Clustering association rules, rather than transactions, once they are mined, is analyzed in [22]. The output is a summary of association rules. The approach is different from ours since this proposal works with the original data set whereas ours produces a model of the data set. In [42] the idea of mining frequent itemsets with error tolerance is introduced. This approach is related to ours since the error is somewhat similar to the bounds we propose. Their algorithm can be used as a means to cluster transactions or perform estimation of query selectivity. In [39] the authors explore the idea of building approximate models for associations to see how they change over time.

VI. Conclusions

This article proposed to use clusters on binary data sets to bound and estimate association rule support and confidence. The sufficient statistics for clustering binary data are simpler than those required for numeric data sets and consist only of the sum of binary points (transactions). Each cluster represents a long itemset from which shorter itemsets can be easily derived. The clustering model on high dimensional binary data sets is computed with efficient operations on sparse matrices, skipping zeroes. We first presented lower and upper bounds on support, whose average estimates actual support. Model-based support metrics obey the well-known downward closure property. Experiments measured accuracy focusing on relative error in support estimations and efficiency with real and synthetic data sets. A clustering model is accurate to estimate support when using a sufficiently high number of clusters. When the number of clusters increases accuracy increases. On the other hand, as the minimum support threshold decreases accuracy also decreases, but at a different rate depending on the data set. The error on support estimation slowly increases as itemset length increases. The model is fairly accurate to discover a large set of frequent itemsets at multiple support levels. Clustering is faster than A-priori to mine frequent itemsets, without considering the time to compute the model. Adding the time to compute the model, clustering is slower than A-priori, at high support levels, but faster at low support levels. The clustering model can be built in linear time on data size. Sparse matrix operations enable fast computation with high dimensional transaction data sets.

There exist important research issues. We want to analytically understand the relationship between the clustering model and the error on support estimation. We need to determine an optimal number of clusters given a maximum error level. Correlation analysis and PCA represent a next step after the clustering model, but the challenges are updating much larger matrices and dealing with numerical issues. We plan to incorporate constraints based on domain knowledge into the search process. Our algorithm can be optimized to discover and periodically refresh a set of association rules on streaming data sets.

References

[1] C. Aggarwal and P. Yu. Finding generalized projected clusters in high dimensional spaces. In *ACM SIGMOD Conference*, pages 70–81, 2000.

[2] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Conference*, pages 207–216, 1993.

[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB Conference*, pages 487–499, 1994.

[4] A. Asuncion and D.J. Newman. *UCI Machine Learning Repository*. University of California, Irvine. School of Inf. and Comp. Sci., 2007.

[5] R. Bayardo and R. Agrawal. Mining the most interesting rules. In *ACM KDD Conference*, pages 145–154, 1999.

[6] R. Bekkerman, R. El-Yaniv, Y. Winter, and N. Tishby. On feature distributional clustering for text categorization. In *ACM SIGIR*, pages 146–153, 2001.

[7] P. Bradley, U. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In *Proc. ACM KDD Conference*, pages 9–15, 1998.

[8] A. Bykowsky and C. Rigotti. A condensed representation to find frequent patterns. In *ACM PODS Conference*, 2001.

[9] C. Creighton and S. Hanash. Mining gene expression databases for association rules. *Bioinformatics*, 19(1):79–86, 2003.

[10] L. Cristofor and D.A. Simovici. Generating an informative cover for association rules. In *ICDM*, pages 597–600, 2002.

[11] W. Ding, C.F. Eick, J. Wang, and X. Yuan. A framework for regional association rule mining in spatial datasets. In *Proc. IEEE ICDM Conference*, pages 851–856, 2006.

[12] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. J. Wiley and Sons, New York, 1973.

[13] F. Geerts, B. Goethals, and J.V. den Bussche. A tight upper bound on the number of candidate patterns. In *ICDM Conference*, pages 155–162, 2001.

[14] S. Guha, R. Rastogi, and K. Shim. ROCK: A robust clustering algorithm for categorical attributes. In *ICDE Conference*, pages 512–521, 1999.

[15] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, San Francisco, 1st edition, 2001.

[16] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. ACM SIGMOD Conference*, pages 1–12, 2000.

[17] T. Hastie, R. Tibshirani, and J.H. Friedman. *The Elements of Statistical Learning*. Springer, New York, 1st edition, 2001.

[18] J. Huang, S. Chen, and H. Kuo. An efficient incremental mining algorithm-QSD. *Intelligent Data Analysis*, 11(3):265–278, 2007.

[19] Z. Huang. Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(3):283–304, 1998.

[20] M. Kryszkiewicz. Mining with cover and extension operators. In *PKDD*, pages 476–482, 2000.

[21] M. Kryszkiewicz. Reducing borders of k-disjunction free representations of frequent patterns. In *ACM SAC Conference*, pages 559–563, 2004.

[22] B. Lent, A. Swami, and J. Widom. Clustering association rules. In *Proc. IEEE ICDE Conference*, pages 220–231, 1997.

[23] T.M. Mitchell. *Machine Learning*. Mac-Graw Hill, New York, 1997.

[24] S. Morishita and J. Sese. Traversing itemsets lattices with statistical pruning. In *ACM PODS Conference*, 2000.

[25] R.T. Ng, L.V.S. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained association rules. In *SIGMOD Conference*, pages 13–24, 1998.

[26] C. Ordonez. Clustering binary data streams with K-means. In *Proc. ACM SIGMOD Data Mining and Knowledge Discovery Workshop*, pages 10–17, 2003.

[27] C. Ordonez. A model for association rules based on clustering. In *Proc. ACM SAC Conference*, pages 549–550, 2005.

[28] C. Ordonez. Integrating K-means clustering with a relational DBMS using SQL. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 18(2):188–201, 2006.

[29] C. Ordonez, N. Ezquerra, and C.A. Santana. Constraining and summarizing association rules in medical data. *Knowledge and Information Systems (KAIS)*, 9(3):259–283, 2006.

[30] C. Ordonez and E. Omiecinski. Efficient disk-based K-means clustering for relational databases. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 16(8):909–921, 2004.

[31] S. Roweis and Z. Ghahramani. A unifying review of linear Gaussian models. *Neural Computation*, 11:305–345, 1999.

[32] A. Savasere, E. Omiecinski, and S. Navathe. An efficient algorithm for mining association rules. In *VLDB Conference*, pages 432–444, September 1995.

[33] T. Scheffer. Finding association rules that trade support optimally against confidence. *Intelligent Data Analysis*, 9(4):381–395, 2005.

[34] C. Silvestri and S. Orlando. Approximate mining of frequent patterns on streams. *Intelligent Data Analysis*, 11(1):49–73, 2007.

[35] R. Srikant and R. Agrawal. Mining generalized association rules. In *VLDB Conference*, pages 407–419, 1995.

[36] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *Proc. ACM SIGMOD Conference*, pages 1–12, 1996.

[37] R. Taouil, N. Pasquier, Y. Bastide, and L. Lakhal. Mining bases for association rules using closed sets. In *IEEE ICDE Conference*, page 307, 2000.

[38] H. Toivonen. Sampling large databases for association rules. In *Proc. VLDB Conference*, pages 134–145, 1996.

[39] A. Veloso, B. Gusmao, W. Meira, M.Carvalo, Parthasarathi, and M.J. Zaki. Efficiently mining approximate models of associations in evolving databases. In *PKDD Conference*, 2002.

[40] K. Wang, Y. He, and J. Han. Pushing support constraints into association rules mining. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 15(3):642–658, 2003.

[41] K. Wang, C. Xu, and B. Liu. Clustering transactions using large items. In *ACM CIKM Conference*, pages 483–490, 1999.

[42] C. Yang, U.M. Fayyad, and P. Bradley. Efficient discovery of error-tolerant of frequent itemsets in high dimensions. In *ACM KDD Conference*, pages 194–203, 2001.

[43] G. Yang. The complexity of mining maximal frequent itemsets and maximal frequent patterns. In *ACM KDD Conference*, pages 344–353, 2004.

[44] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient data clustering method for very large databases. In *Proc. ACM SIGMOD Conference*, pages 103–114, 1996.