Models for Association Rules based on Clustering and Correlation

Carlos Ordonez
University of Houston
Houston, TX 77204, USA *

Abstract

Association rules require models to understand their relationship to statistical properties of the data set. In this work, we study mathematical relationships between association rules and two fundamental techniques: clustering and correlation. Each cluster represents an important itemset. We show the sufficient statistics for clustering and correlation on binary data sets are the linear sum of points and the quadratic sum of points, respectively. We prove itemset support can be bounded and approximated from both models. Support bounds and support estimation obey the set downward closure property for fast bottom-up search for frequent itemsets. Both models can be efficiently computed with sparse matrix computations. Experiments with real and synthetic data sets evaluate model accuracy and speed. The clustering model is accurate to estimate support, given a sufficiently large number of clusters and it is more accurate than correlation, except for sets of two items. Accuracy increases as the number of clusters grows, but decreases as the minimum support threshold decreases. Once built, the clustering model represents a faster alternative than the traditional A-priori algorithm and the correlation model to mine associations. The correlation model is faster to compute than clustering, but it is less accurate. Time complexity to compute both models is linear on data set size, whereas dimensionality marginally impacts time when analyzing large transaction data sets.

Keywords: association rules; clustering; correlation; approximation

1 Introduction

Association rules describe frequent patterns found in large binary data sets [26, 16, 8]. Research on association rules has become extensive since their introduction in the article [2]. There has been significant research on developing fast algorithms [3, 33, 16] to discover association rules, generalizing association rules to use taxonomies [36], constraining [40] and applying them in different domains [9]. However, little has been done on understanding relationships between them and common statistical and machine learning techniques. Clustering [11, 27] finds groups of similar points according to some similarity metric, generally distance. On the other hand, correlation analysis [11, 17] is a basic method to understand relationships between two numeric variables and it is generally the first step towards building multivariate statistical models. In this work we show clustering and correlation analysis can be a statistical complement to association rule mining.

Even though association rules are a well researched topic, most work has focused on developing fast algorithms or proposing variations of association rules (constrained, quantitative, predictive, taxonomy-based and so on [15]). Studying their relationships to other techniques has received less attention. In this work we study relationships between association rules and two well-known techniques: clustering and correlation. We study how to bound and approximate association rule metrics based on a binary data clustering model. The relationship between clustering and association rules is important because clustering, on one hand, looks for global patterns producing in general a few disjoint subsets of points, whereas association rules uncover many local patterns referring to overlapping subsets of points. In general, it is hard to compare different association rules because it is not known if they refer to the same data points. We also study how

 The official version of this article was published in Intelligent Data Analysis (IDA Journal). 13(2):337-358, 2009. DOI: 10.3233/IDA-2009-0369

to bound and approximate association rule metrics based on the correlation matrix. The correlation matrix goes beyond co-occurrence of items showing the presence of an item may imply the absence of another item, or explaining relationships between two items in a more precise manner. Correlation can help understanding if the presence of one item is related to the presence or absence of another item. A pair of highly correlated items may suggest potential association rules. Correlations can explain why certain rules have low confidence and can help validating rules with high confidence but low support. In short, our proposal can be used to explore and understand a binary data set from a statistical perspective to discover association rules.

The article is organized as follows. Section 2 presents unified definitions on clustering and association rules. Section 3 explains the relationship of clusters to itemsets and discusses how to exploit binary data clusters to bound and estimate association metrics. Similarly, Section 4 explains how to bound association metrics from a correlation matrix. Section 5 presents experiments with real and synthetic data sets, evaluating approximation accuracy, speed and scalability. Section 6 discusses related work. Section 7 contains the conclusions of the article.

2 Definitions

This section provides definitions used throughout the article. Definitions on clustering [7, 30, 29] and multivariate statistics [17] are adapted to association rules [2, 3, 33]. The intuition behind definitions is that items are used as dimension subscripts.

2.1 Clustering and Correlation

Let n be the number of points in the data set X and d its dimensionality. Let $X = \{x_1, x_2, \ldots, x_n\}$, where each dimension X_l is a binary dimension (or item). That is, x_i is a binary vector. The data set X is formally a $d \times n$ binary matrix, where $X_{li} = 1$ if item l is present in point i, and 0 otherwise.

In our work we use the K-means algorithm as the basis to build clusters, where k is a user-specified number of clusters. The K-means clustering algorithm partitions the data set into k disjoint groups such that points in the same group are similar to each other and points across groups are different from each other according to Euclidean distance. The output is a model represented by matrices W, C, R, containing the weights, the centroids (means) and the radiuses (variances) respectively for each cluster and a partition of X into k subsets. Matrices C and R are $d \times k$ and W is a $k \times 1$ matrix. To refer to a column of C or R we use the j subscript (i.e. C_j, R_j). Therefore, C_j refers to the jth cluster centroid, R_j refers to the jth diagonal covariance matrix and W_j is the fraction that cluster j represents from X. Matrix R_j can be informally understood as a vector containing squared radiuses per dimension. We use the following convention for subscripts. For transactions we use i; $i \in \{1, 2, \ldots, n\}$. Note that i alone is a subscript to index transactions, whereas i_j refers to item j defined below. For cluster number we use j; $j \in \{1, 2, \ldots, k\}$ and to refer to one dimension we use l: $l \in \{1, 2, \ldots, d\}$. Let $Y_1, Y_2, \ldots, Y_k$ be the k disjoint subsets of X induced by clusters s.t. $Y_j \cap Y_g = \emptyset$ for $j \neq g$. The $diag[]$ notation will be used as a generic operator to obtain a diagonal matrix from a vector, or to convert the diagonal of a matrix into a vector. For matrix/vector transposition we use the T superscript; for instance C_j^T is the jth centroid as a row. Subscripts and matrices are summarized in Figure 1.

For correlation analysis the output is a $d \times d$ matrix ρ containing entries between -1 and 1. The number ρ_{ij} is also called the Pearson coefficient of items i and j. Matrix ρ will be computed from two summary matrices L and Q we will introduce later.

Size	value
k	# of clusters
d	dimensionality
n	data set size

Matrix	size	contents
C	$d \times k$	centroids
R	$d \times k$	variances
W	$k \times 1$	weights

Figure 1: Matrices names and subscripts.

T_i	Items
1	1 3
2	2 3 4
3	1 2
4	2
5	2 4

x_i	1 2 3 4
1	1 0 1 0
2	0 1 1 1
3	1 1 0 0
4	0 1 0 0
5	0 1 0 1

Figure 2: Input transactions $d = 4, n = 5$.

2.2 Association Rules

The following paragraphs provide definitions for association rules. We must introduce some notation for transactions given the sparse nature of transactions, understood as binary vectors. Let $T_i = \{l | X_{li} = 1, l \in \{1, 2, \ldots, d\}, i \in \{1, 2, \ldots, n\}\}$. That is, T_i is the set of non-zero coordinates of x_i; T_i can be understood as a transaction or an itemset to be defined below. Since transactions represent sparse vectors we have $|T_i| << d$.

Let $\mathcal{I}$ be a set of d integers identifying d items, $\mathcal{I} = \{1, 2, \ldots, d\}$. Each dimension of $\mathcal{S}$ corresponds to one item out of d items and vice-versa. The presence or absence of an item in a transaction is indicated by the presence or absence of its identifier. A set of items is called an *itemset*. An itemset A containing p items is called a p-itemset and we will refer to it as $A = \{\mathrm{i}_1, \mathrm{i}_2, \ldots, \mathrm{i}_p\}$, where $\mathrm{i}_l \in \mathcal{I}$. We will use the vectorial notation $(C_j)_{\mathrm{i}_l}$ with the following interpretation: "access the mean of dimension i_l in cluster j". In other words, the item itself is used as a dimension index to access the means from a certain cluster; an equivalent notation based on matrices is $C_{\mathrm{i}_l j}$, but it is harder to understand. The $(C_j)_{\mathrm{i}_l}$ notation will be used throughout the article whenever items refer to clusters. The intuition is that only the items given in an association are relevant to access the dimensions of some cluster.

Going back to common notation for association rules, we add the following definitions. An itemset has a measure of statistical significance called support. For an itemset $A \subseteq \mathcal{I}$, $s(X) = |\{T_i | A \subseteq T_i\}|/n$ for $i = 1 \ldots n$ (i.e. support is the fraction of transactions in X containing A). An *association rule* is an implication of the form $A \Rightarrow B$, where $A, B \subset \mathcal{I}$, and $A \cap B = \emptyset$, where A and B are itemsets. The A itemset is called the antecedent and B is the consequent of the rule. The rule $A \Rightarrow B$ has a measure of strength called confidence defined as $c(A \Rightarrow B) = s(A \cup B)/s(A)$. The problem of mining frequent itemsets is defined as finding all itemsets that have support above a minimum support threshold τ. The problem of mining association rules is to generate all rules that have support and confidence greater or equal than τ and ψ thresholds, respectively. Searching for all frequent itemsets is considered the most challenging problem [3, 15].

Example 2.1 *Figure 2 shows a small input database X, where $\mathcal{I} = \{1, 2, 3, 4\}$, $d = 4, n = 5$. Figure 1 provides a summary of the $k = 2$ clusters. Cluster 1 contains transactions 2 and 5 and cluster 2 contains transactions 1,3 and 4. In Figure 3 we show the output from a clustering run having as input X and $k = 2$. Figure 1 provides a nice summary of clusters that is easier to understand than matrices C and R, which have floating point numbers. Items appear grouped in ranges according to their C_j value relative to the cluster where they appear. Those C_j entries are the supports of each item with respect to each cluster. Items*

$$W = \begin{bmatrix} 0.4 \\ 0.6 \end{bmatrix} C = \begin{bmatrix} 0.00 & 0.66 \\ 1.00 & 0.66 \\ 0.50 & 0.33 \\ 1.00 & 0.00 \end{bmatrix} R = \begin{bmatrix} 0.00 & 0.22 \\ 0.00 & 0.22 \\ 0.25 & 0.22 \\ 0.00 & 0.00 \end{bmatrix}$$

Figure 3: Clustering model; $d = 4, k = 2$.

Table 1: Clustering model summary and itemsets.

		C_j range for items					Itemsets
j	W_j	0.8-1	0.6-0.8	0.4-0.6	0.2-0.4	0-0.2	at $\lambda = 0.4$
1	0.4	2 4		3		1	2 3 4
2	0.6		1 2		3	4	1 2

in C_j ranges closer to 1 (left side) are more likely to appear together. That is, they suggest frequent itemsets (associations). The cluster weight W_j gives an idea about association support on the entire data set. The cluster summary is not telling us the exact support values, but it is evident they can be approximated from W and C.

It is noteworthy clusters help us suggesting potential associations regardless of any pre-specified minimum support threshold. Certain item combinations may appear only in one cluster. Other item combinations may appear in several clusters. Given the similarity of cluster item combinations and associations, can we estimate association rule metrics based on a model instead of the data set X? This is studied in Section 3 for clustering and Section 4 for correlation.

3 Clustering

We consider clustering and correlation as the fundamental techniques to build models for association rules. This section explains how to efficiently cluster binary data sets and how binary clusters can be exploited to bound and approximate association metrics.

3.1 Fast Clustering of Binary Data Sets

In our work we use K-means [7, 32, 25, 17] as the base clustering algorithm. There are several important reasons behind choosing K-means. First and foremost, K-means produces good solutions under a variety of challenging conditions [30], like constant values, zero variance and skewed probabilistic distributions. On the other hand, K-means is one of the most widely used clustering algorithms in machine learning [25] and statistics [17]. Insights on K-means can be applied with similar clustering algorithms inspired by or based on K-means. K-means is deeply related to the Expectation-Maximization (EM) [10, 17, 32] algorithm for mixtures of Gaussians [32, 31]. Therefore, K-means opens up the possibility of exploiting maximum likelihood estimation methods with binary data sets. K-means has a simple interpretation of clusters as multidimensional spheres, which enables visualization of clusters on two or three dimensions. Even though K-means is a "flat" algorithm in the sense that it finds the best solution given a number of clusters, it is feasible to build hierarchical algorithms using K-means as a building block. Since the number of clusters is set by the user this enables the exploration of alternative solutions. From a scalability point of view, it is possible to derive fast incremental versions [27, 30] which can get a good solution in a few passes over the data set.

In this work a fast K-means algorithm is used to cluster high dimensional binary data sets. This algorithm is an improvement of the incremental K-means algorithm for binary streams [27]. 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.

3.2 Sufficient Statistics

As mentioned in Section 2, K-means receives k, the number of clusters, as input. Our K-means variant [27], like other scalable clustering algorithms [7, 46], uses sufficient statistics, which are summaries of $Y_1, Y_2, \ldots, Y_k$ represented by matrices L, Q, N that contain k sums of points, k sums of squared points and number of points per cluster respectively. In the following discussion L is a $d \times k$ matrix, such that each column L_j is computed as $L_j = \sum_{x_i \in Y_j} x_i$. Each column of Q is a diagonal matrix Q_j computed as $Q_j = \sum_{x_i \in Y_j} x_i x_i^T$. Matrix N is $k \times 1$ defined as $N_j = |Y_j|$. The update formulas for W, C, R based on sufficient statistics are:

$$W_j = \frac{N_j}{\sum_{g=1}^{k} N_g} \tag{1}$$

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

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

However, sufficient statistics can be simplified for clustering transactions. The following result states that the sufficient statistics for the problem of clustering binary vectors are simpler than the sufficient statistics required for clustering numeric vectors.

Lemma 3.1 *Let $Y_1, Y_2, \ldots, Y_k$ be k subsets representing a partition of X. 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, $L = Q$. □

The practical application of this result is that the equations used to update C, R, W are only the equations for W and C. Matrix R_j can be directly computed from C_j.

Lemma 3.2 $R_j = diag[C_j] - C_j C_j^T$.

Lemma 3.2 leads to the following simple criteria that can be used to assess cluster quality.

Theorem 3.1 *(1) R_{lj} has a minimum if and only if $C_{lj} = 0$ or $C_{lj} = 1$. (2) R_{lj} has a maximum if and only if $C_{lj} = 0.5$.*

Proof. By Lemma 3.2 $R_j = diag[C_j] - C_j C_j^T$. Then each entry $R_{lj} = C_{lj} - C_{lj}^2$. Let $x = C_{lj}$ and $q(x) = R_{lj} = C_{lj} - C_{lj}^2$. To prove (1) we proceed as follows. Since $q(x)$ is a sum of squared distances it can never be negative. Then $q(x) = 0$ is its minimum value. Then $x - x^2 = 0$. This equality can only hold when $x = 1$ or $x = 0$.

To prove the only if part of (2) we need to resort to some simple calculus computations. Then $\frac{\mathrm{d}q(x)}{\mathrm{d}x_l} = 1 - 2x = 0$ to find its maximum or its minimum. The solution to this equation is $x^* = 0.5$. Differentiating again $\frac{\mathrm{d}^2 q(x*)}{\mathrm{d}x_l} = -2$. This tells us that the maximum of $q(x)$ happens when $x = 0.5$. These steps can be reversed to prove the if part. This result holds for all d dimensions. □

The practical application of Theorem 3.1 is that the closer C_j entries are to either 0 or 1 the better, with the ideal case when $C_{lj} = 0$ or $C_{lj} = 1$. On the other hand, the closer C_j entries are to 0.5 the worse. When $C_{lj} = 0.5$ R_{lj} reaches its maximum value: $R_{lj} = 0.25$.

Example 3.1 *Going back to the example from Table 1 and Figure 2, observe cluster 1 suggests* $\{2, 4\}$ *is very likely to be a valid association since it is in the "top"* C_j *range. Cluster 1 also suggests* $\{2, 3, 4\}$ *as a potential association, but since 3 is in the middle 0.4-0.6 range this 3-itemset is much less unlikely to be valid. According to Theorem 3.1 cluster 1 is optimally clustered on dimensions 1,2,4, cluster 2 only on dimension 4 and cluster 1 on dimension 3 makes* R_{31} *reach the maximum. Cluster 1 is accurate to estimate metrics for associations involving items 1,2,4.*

The "itemsets" column on Table 1 shows the itemsets represented by each cluster at a minimum centroid threshold $\lambda = 0.4$*. The* λ *parameter is used to eliminate itemsets that are likely to be infrequent and will be explained in detail later. These two itemsets represent two maximal itemsets from which almost all frequent itemsets at minimum support* $\tau = 0.2$ *can be derived: only itemset* $\{1, 3\}$ *cannot be derived from these two itemsets. If we lowered* $\lambda = 0.2$ *then we would get all itemsets in D, but we would also include itemsets that are not frequent. This is a tradeoff we will study in the experimental section.*

3.3 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 ($s(A, Y_j)$) must be less or equal than the minimum support of any item, as shown in [33].

Lemma 3.3 *Let* Y_j *be one cluster from* X*. Let* $A = \{i_1, i_2, \ldots, i_p\}$ *be some* p*-itemset. Let* C_j *be the mean of transactions that belong to cluster* j*. Then* $s(A, Y_j) \leq min((C_j)_{i_1}, (C_j)_{i_2}, \ldots, (C_j)_{i_p})$.

Proof. The proof is by induction on $p = |A|$. First, it is verified for $p = 1$. The quantity $s(A)$ is by definition the fraction of transactions that contain A. Since A contains only one item, say i, then i appears in $N_j \times s(A)$ transactions. Therefore, $(C_j)_i = s(A)$. Hence the proposition holds for $p = 1$. Now assume the proposition holds for $i = p - 1$. Let $A = \{\mathrm{i}_1, \mathrm{i}_2, \ldots, \mathrm{i}_p\}$ be a p-itemset and let $B \subset A$ s.t. $B = \{\mathrm{i}_1, \ldots, \mathrm{i}_{p-1}\}$. By induction hypothesis $s(B) \leq min((C_j)_{\mathrm{i}_1}, (C_j)_{\mathrm{i}_2}, \ldots, (C_j)_{\mathrm{i}_{p-1}})$. Now the itemset $B \cup \{\mathrm{i}_p\}$ appears only in transactions where both B and $\{\mathrm{i}_p\}$ appear. Therefore, $s(A) \leq s(B) \leq min((C_j)_{\mathrm{i}_1}, \ldots, (C_j)_{\mathrm{i}_{p-1}})$ and $s(A) \leq s(\{\mathrm{i}_p\}) = (C_j)_{\mathrm{i}_p}$, and the lemma follows. □

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

$$u_s(A) = \sum_{j=1}^{k} W_j \mathrm{min}((C_j)_{\mathrm{i}_1}, (C_j)_{\mathrm{i}_2}, \ldots, (C_j)_{\mathrm{i}_p}). \tag{4}$$

Theorem 3.2 *Let* X *be a set of* n *transactions. Let* C, W *be the matrices containing the means and weights respectively on some clustering of* X *into* k *subsets of transactions* $Y_1, Y_2, \ldots, Y_k$*. Let* A *be any* p*-itemset given by* $A = \{i_1, i_2, \ldots, i_p\}$*. Then* $s(A) \leq u_s(A)$.

Proof. Let $s(A, Y_j)$ be the support of A referred to Y_j. Let $count(A, Y_j)$ be the number of times A appears in Y_j. By Lemma 3.3 $s(A, Y_j) \leq \mathrm{min}((C_j)_{\mathrm{i}_1}, (C_j)_{\mathrm{i}_2}, \ldots, (C_j)_{\mathrm{i}_p})$, for $j = 1 \ldots k$. Both sides of each inequality can be multiplied by $W_j = |Y_j|/n \geq 0$. Then $s(A, X_j)|Y_j|/n \leq W_j \mathrm{min}((C_j)_{\mathrm{i}_1}, (C_j)_{\mathrm{i}_2}, \ldots, (C_j)_{\mathrm{i}_p})$. Now the k inequalities for each j can be added side by side:
$(1/n) \sum_{j=1}^{k} s(A, Y_j)|Y_j| \leq \sum_{j=1}^{k} W_j \mathrm{min}((C_j)_{\mathrm{i}_1}, (C_j)_{\mathrm{i}_2}, \ldots, (C_j)_{\mathrm{i}_p})$. Now simplifying, $s(A, Y_j)|Y_j| = count(A, Y_j)$. But $Y_i \cap Y_j = \emptyset$ for $i \neq j$ and $\bigcup_{j=1}^{k} Y_j = X$. Therefore, $\sum_{j=1}^{k} count(A, Y_j) = count(A, X)$.

Substituting the formula on the left hand side of the last inequality with this latter expression proves the theorem. □

Theorem 3.2 guarantees that C and W are enough to know all potential frequent itemsets thereby avoiding making several passes over X. However, this theorem does not guarantee that only true frequent itemsets are above the minimum support threshold. There may be itemsets whose support is lower than that indicated by the model. The following result is based on Theorem 3.2 and is used to prune the lattice search space if associations are being mined based on the clustering model using $u_s()$. Now we introduce a lower bound for the support of an association.

$$l_s(A) = \sum_{j=1}^{k} W_j \text{max}(0, 1 + \sum_{l=1}^{p}((C_j)_{\mathrm{i}_l} - 1)), \tag{5}$$

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

Theorem 3.3 *Let $X, C, W, \{Y_1, \ldots, Y_k\}$ be as stated in Theorem 3.2. Let A be a p-itemset $A = \{i_1, i_2, \ldots, i_p\}$. Then $l_s(A) \leq s(A)$.*

Proof. The proof is by induction on itemset size I. For $I = 1$ assume $A = \{\mathrm{i}\}$. Then $l_s(A) = \sum_{j=1}^{k} W_j (C_j)_{\mathrm{i}}$ $= s(\{\mathrm{i}\})$ (because $(C_j)_{\mathrm{i}}$ provides exact support for $\{\mathrm{i}\}$ in Y_j) and the equality holds. Without loss of generality assume $B \subset A$, $|B| = p-1$ and $B = \{\mathrm{i}_1, \ldots, \mathrm{i}_{p-1}\}$. By the induction hypothesis suppose the inequality holds for $I = p - 1$ for B: $l_s(B) \leq s(B)$. Let f_a be the fraction of transactions where i_p does not appear: $-f_a = \sum_{j=1}^{k} W_j[(C_j)_{\mathrm{i}_p} - 1]$ (used to get $l_s(A)$). Let f_b be the fraction of transactions where i_p does not appear and B appears. It follows that $f_b \leq f_a$. Then $l_s(A) = l_s(B) - f_a \leq s(B) - f_a \leq s(B) - f_b = s(A)$ and the result follows. □

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

$$a_s(A) = \frac{l_s(A) + u_s(A)}{2}. \tag{6}$$

3.4 Downward Closure for Support and Itemset Search Algorithm

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 A be a p-itemset and let $B \subset A$, $B = \{h_1, \ldots, h_q\}$, $q < p$. Then:*
(1) $l_s(A) \leq l_s(B)$,
(2) $u_s(A) \leq u_s(B)$,
(3) $a_s(A) \leq a_s(B)$.

Proof.
(1) The sum on the left hand side ($l_s(A)$) contains all terms on the right hand side ($l_s(B)$) and each extra term $((C_j)_{\mathrm{i}_l} - 1) \leq 0$.
(2) Let $C = A - B = \{\mathrm{h}_{q+1}, \ldots, \mathrm{h}_p\}$. Then $A = B \cup C$. Then using the fact that $min()$ is distributive: $u_s(A) = u_s(B \cup C) = \sum_{j=1}^{k} W_j \text{min}((C_j)_{\mathrm{h}_1}, \ldots, (C_j)_{\mathrm{h}_q}, (C_j)_{\mathrm{h}_{q+1}}, \ldots, (C_j)_{\mathrm{h}_p}) \leq \sum_{j=1}^{k} W_j \text{min}((C_j)_{\mathrm{h}_1}, \ldots, (C_j)_{\mathrm{h}_q}) =$ $u_s(B)$.
(3): $a_s(A) \leq a_s(B)$ follows from (1) and (2). □

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

As explained before upper support ($u_s()$) is safe in the sense that it produces a superset of all frequent itemsets, but it may introduce some infrequent (false positive) itemsets. In general, a subset of C_j entries close to zero indicate itemsets that are unlikely to be frequent. We introduce a λ parameter for the algorithm that helps filtering out such infrequent itemsets: 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. 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). In the experimental section we study the whole spectrum for λ settings and justify a default value, which tends to be close to τ.

3.5 Bounds on Confidence

Let A and B be two itemsets s.t. $A \cap B = \emptyset$. Let the lower confidence of a rule be defined as

$$l_c(A \Rightarrow B) = \frac{l_s(A \cup B)}{u_s(A)} \tag{7}$$

Let the upper confidence of a rule be defined as

$$u_c(A \Rightarrow B) = \min(\frac{u_s(A \cup B)}{l_s(A)}, 1) \tag{8}$$

Note the symmetry between Equations 7 and 8. These equations lead to the following results that estimate tight bounds for association rule confidence based on a clustering model.

Theorem 3.5 *Let X be a set of transactions. Let C, W be a clustering model of X with k clusters. Let A and B be two itemsets s.t. $A \cap B = \emptyset$. Then $l_c(A \Rightarrow B) \leq c(A \Rightarrow B) \leq u_c(A \Rightarrow B)$.*

Proof. The proof follows from the definition of association rule confidence, Equations 7 and 8, Theorem 3.2 and Theorem 3.3. □

3.6 Varying the Number of Clusters k

The number of clusters k controls model accuracy to estimate support bounds. Consider the relative error of $a_s()$ support for a set of associations $\mathcal{A} = \{A_1, A_2, \ldots, A_N\}$ based on the clustering model parameters W, C. Let relative error be defined as $r(\mathcal{A}, W, C) = \frac{1}{N} \sum_{I=1}^{N} |a_s(A_I) - s(A_I)|/|s(A_I)|$. At one extreme $k = 1$ reduces the entire data set to one cluster. Such model is equivalent to computing only the global mean. In general, $r(\mathcal{A}, W, C)$ will slowly grow as p (itemset length) increases. At the other extreme, $k = n$, produces a complex model, where each input point becomes one cluster (without loss of generality points in the data set are assumed to be unique). In such case the model can provide exact support estimation for any p-itemsets since each cluster produces equal bounds. Therefore, $r(\mathcal{A}, W, C) = 0$. However, a model where $k = n$ is impractical and produces no summarization of the data set. Therefore, the ideal value for k will be somewhere between 1 and n, but closer to $k = 1$ to produce more concise models. If k is too high it will produce empty clusters, but K-means will generally find good clusters [7]. As the minimum support threshold τ is lowered many more longer p-itemsets will be found, since the number of associations grows in a combinatorial manner. However, in general, there will be a point at which further increasing k will

produce marginal or no improvement since model accuracy is asymptotic. There exist several clustering validity indices to determine an optimal k for numeric data [24], but we do not use them since a sufficiently high k will produce a small support estimation error. Studying validity indices is an aspect for future work.

4 Correlation

In this section we analyze transactions with correlation analysis, which is an alternative method available from statistics to study a data set from a bi-variate perspective. Even though correlation is not strictly a model by itself we refer to both clustering and correlation as models. In fact, dimensionality reduction models like principal component analysis (PCA) [17] or maximum likelihood factor analysis [32, 17] are models derived from the correlation matrix.

4.1 Sufficient Statistics

We refer to the d dimensions as $X_1, \ldots, X_d$ (not to be confused with the n points x_i, where x is in lower case). The following definitions basically generalize the definitions for clustering considering all item pairs. Let L be the *linear* sum of points, in the sense that each point is taken at power 1: $L = \sum_i x_i$. L is a $d \times 1$ matrix. In matrix (column-vector) form L is as follows:

$$L = \begin{bmatrix} \sum X_1 \\ \sum X_2 \\ \vdots \\ \sum X_d \end{bmatrix}$$

Let Q be the *quadratic* sum of points, in the sense that each point is squared with a cross-product. Q is a $d \times d$ matrix.

$$Q = XX^T = \sum_{i=1}^{n} x_i x_i^T. \tag{9}$$

Matrix Q has sums of squares in the diagonal and sums of cross-products off the diagonal:

$$Q = \begin{bmatrix} \sum X_1^2 & \sum X_1X_2 & \ldots & \sum X_1X_d \\ \sum X_2X_1 & \sum X_2^2 & \ldots & \sum X_2X_d \\ \vdots & & & \vdots \\ \sum X_dX_1 & \sum X_dX_2 & \ldots & \sum X_d^2 \end{bmatrix}$$

The correlation matrix ρ can be derived from matrices L and Q substituting each corresponding sum by the appropriate entries from L and Q: $\rho_{ab} = (nQ_{ab} - L_aL_b)/(\sqrt{nQ_{aa} - L_a^2}\sqrt{nQ_{bb} - L_b^2})$.

Each correlation value ρ_{ab} is also called the Pearson correlation coeficient [42]. The correlation matrix has a straightforward interpretation. If ρ_{ab} is close to 1 we say X_a and X_b have a strong positive correlation, meaning that if item a appears then item b also appears in the transaction. If ρ_{ab} is close to -1 we say X_a and X_b have a strong negative correlation, meaning that if item a appears then item b does not appear in the same transaction. Finally, if ρ_{ab} is close to 0 items a and b are considered independent.

The following result links sufficient statistics and correlation, for items in a similar way to clustering binary data.

Lemma 4.1 *Let X represent a binary data set on d dimensions. Then the sufficient statistics to compute the correlation ρ_{ab} between any two dimensions a, b is only Q.*

$$Q = \begin{bmatrix} 2 & & & \\ 1 & 4 & & \\ 1 & 1 & 2 & \\ 0 & 2 & 1 & 2 \end{bmatrix} \quad \rho = \begin{bmatrix} 1 & & & \\ -0.61 & 1 & & \\ 0.16 & -0.61 & 1 & \\ -0.67 & -0.40 & 0.17 & 1 \end{bmatrix}$$

Figure 4: Sufficient statistics for correlation.

Proof. The proof is similar to Lemma 3.1 based on the fact that x_i is a binary vector. Then $L_a = Q_{aa}$ or equivalently $L = diag[Q]$. $\square$

In practical terms, this means that whatever information we can infer from ρ we can infer it from Q. Then we can get the simpler equation:

$$\rho_{ab} = \frac{nQ_{ab} - Q_{aa}Q_{bb}}{\sqrt{nQ_{aa} - Q_{aa}^2}\sqrt{nQ_{bb} - Q_{bb}^2}} \tag{10}$$

This represents a duality: We proved that for clustering since all needed information is in L then Q can be ignored. But for correlation analysis, since L does not provide additional information about X compared to Q then L becomes redundant. Since Q is symmetrical we can compute only the lower or upper triangular submatrix; we will consider the lower triangular matrix, where $b \leq a$ for an entry Q_{ab}. Clearly, Q can give us exact support for 1-itemsets with its diagonal and for 2-itemsets with entries off the diagonal.

We have shown we can go from Q to ρ. But if we only have ρ we cannot obtain Q because ρ has 1s on the diagonal. However, if we have both L and ρ then we can easily go back to Q using the equation: $Q_{ab} = \frac{1}{n}(\rho_{ab}\sqrt{(nL_a - L_a^2)(nL_b - L_b^2)} + L_aL_b)$.

This equation means Q provides more information about X than ρ. Therefore, we concentrate on efficiently computing Q and using Q to derive bounds. That is, Q represents our fundamental correlation model.

Since Q may be a large matrix for high d Q can be summarized by binning correlations into appropriate ranges or using PCA or factor analysis techniques.

Example 4.1 *In Figure 4 we show the sufficient statistics Q and correlation matrix ρ for our data set from Figure 2. We can see from Q that there are several 2-itemsets with frequency 1 ($s(A) = 0.2$). However, we can see their corresponding correlation is different. Conversely, a similar correlation figure does not imply the corresponding support is similar. Thus the correlation matrix provides interesting information in addition to frequent itemsets.*

Sparse Computation of Q

For high d we can update Q in a sparse manner, similarly to efficiently updating N, L for clustering. That is, we only increment 1 for every item pair in a transaction. For small transactions the time to incrementally update Q will be slightly higher than the time to read the transaction because disk I/O is slower than memory access and Q can be maintained in memory. Therefore, time complexity is $O(t^2n)$ to compute Q, where t is the average transaction size (number of items).

4.2 Bounds on Support

Bounds on support are derived from Q, and not from ρ. Let A be a p-itemset. We define an upper bound on support as

$$u_s(A) = \frac{1}{n}\min(Q_{ab}) \tag{11}$$

for every pair $\{\mathrm{i}_a, \mathrm{i}_b\} \subseteq A$. For 1-itemsets and 2-itemsets $u_s(A) = s(A)$. This can be done in time $O(p^2)$ for one association A. We define a lower bound on support as

$$l_s(A) = \max(0, -1 + \frac{1}{n}[\min(Q_{ab}) + \max(Q_{cc})])$$

for every pair, $\{\mathrm{i}_a, \mathrm{i}_b\} \subseteq A$ s.t. $a \neq b$ and for $\mathrm{i}_c \in A$. These definitions lead to the following results:

Lemma 4.2 $l_s(A) \leq s(A) \leq u_s(A)$

Proof. $s(A) \leq u_s(A)$ follows from the downward closure of an itemset. The lower bound works as follows. The first term (zero) is used to avoid negative bounds. We can rewrite the second parameter of max() as $\min(Q_{ab})/n - \min(n - Q_{cc})/n = u_s(A) - \min(n - Q_{cc})/n$. Therefore, from $u_s(A)$ we are subtracting the minimum fraction of transactions that do *not* contain A. □

An estimation of support can be defined in the same manner as clustering, with a bounds average as: $a_s(A) = (l_s(A) + u_s(A))/2$. Alternatively, $u_s(A)$ can be used as an estimation for support since it is a tight bound. The bounds for rule confidence can be derived in an analogous manner to clustering. It is straightforward to prove downward closure: if $B \subset A$ for two itemsets then $l_s(B) \geq l_s(A)$ and $u_s(B) \geq u_s(A)$.

4.3 Time and Space Complexity

The clustering model matrices W, C take space $O(dk)$ and the correlation matrices Q take $O(d^2)$. Therefore, the clustering model matrices take linear space in d and the correlation matrix takes quadratic space in d. Notice both models do not depend on n and therefore they can substitute the data set X. If $k < d$ the the clustering model takes less space than correlation. Clustering is computed in time $O(tkn + dk) = O(tkn)$ and correlation in time $O(t^2 n)$ for large n and average transaction size t (average number of items per T_i). Since in general $k > t$ and K-means makes several passes over X then correlation is faster than clustering.

5 Experimental Evaluation

This section includes experimental evaluation. The experiments are divided into two major sections: (1) measuring model accuracy to estimate support from statistical models; (2) evaluating time performance and scalability.

All experiments were performed on a computer running at 3 GHz with 2 GB of memory. This machine had 160 GB on disk. Our algorithms were programmed in the C++ language. The data sets were stored as text files and the models were stored on binary files after their computation. Models were loaded into main memory for processing. Each experiment was repeated five times and average performance figures are reported. In general, processing times are reported in seconds.

5.1 Data Sets

We conducted experimental evaluation with 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 data sets were created with the well-known IBM transaction data generator [3]. All data sets were converted to transaction format (itemset representation) for efficient processing. Data sets are summarized in Table 2.

As mentioned above, synthetic data sets with high dimensionality were created 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

Table 2: Summary of data sets.

Data set	d	n	Contents
Chess	75	3196	Chess moves to win/lose game
Mushroom	127	8124	Edible/poisonous mushrooms
Votes	17	435	Voting data set
T10I4D1M	100	1000k	Synthetic transaction data set

Table 3: Real data sets: Accuracy varying k and τ (relative support error) at $\lambda = 0.2$.

Data set	tau	$k = 10$	$k = 20$	$k = 40$	$k = 80$
Chess	0.8	0.02	0.02	0.01	0.00
	0.2	0.12	0.06	0.05	0.03
Mushroom	0.7	0.01	0.01	0.00	0.00
	0.2	0.27	0.10	0.02	0.01
Votes	0.5	0.01	0.00	0.00	0.00
	0.1	0.36	0.23	0.10	0.05

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 $conf = 0.75$. That is, we created a large synthetic data set with one million binary points.

5.2 Accuracy

Parameters Setting and Experiments Overview

The clustering model has two parameters: k, the number of clusters and λ, the cluster centroid threshold (to filter out itemsets unlikely to be frequent). Each data set has a different optimal setting for k. Therefore, there is no default value for k. On the other hand, $\lambda = 0.2$; we justify this setting showing its spectrum from 0 to 1. The correlation model has no parameters.

The fundamental metric to evaluate accuracy was relative error, which for an itemset A it was computed as $|a_s(A) - s(A)|/|s(A)|$, where average support $a_s(A)$ is obtained from the clustering or the correlation model.

Since rule confidence is an equation on support we concentrate on analyzing error on support estimation. The problem of choosing the best k is called model selection [25, 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 is 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 reasonable since in practice several trial and error runs are made before selecting the best model.

Table 4: Synthetic data set: Accuracy varying k and τ (relative support error) at $\lambda = 0.2$.

τ	k =50	k =100	k =200	k =400	k =800
0.20	0.01	0.00	0.00	0.00	0.00
0.15	0.04	0.01	0.01	0.01	0.01
0.10	0.07	0.04	0.04	0.03	0.02
0.05	0.16	0.11	0.09	0.10	0.05
0.02	0.42	0.28	0.20	0.13	0.08

Varying the number of clusters (k)

Table 3 analyzes relative error as τ and k vary for real data sets with $\lambda = 0.2$. We picked two τ thresholds, a high value that produced a small number of frequent itemsets and a low one which produced a large number of frequent itemsets. As k increases relative error always decreases or remains constant. When $k = 80$ the relative error becomes zero for the high τ threshold and it is always $\leq 5\%$ for the lower τ threshold. In short, the clustering model is fairly accurate at $k = 80$. Interestingly enough, $k = 80$ produces excellent results from the three data sets, despite the fact that they have different d and statistical characteristics. In general, relative error decreases in a linear fashion with respect to k. Later we will analyze how relative error decreases as τ gradually decreases. From these results we conclude k, the number of clusters, is a model parameter that can control support estimation accuracy.

Table 4 presents an analysis of relative error for the synthetic transaction data set. In this case we vary both k and τ. Increasing k always decreases relative error monotonically, whereas decreasing τ always increases relative error, highlighting the combinatorial growth on the number of itemsets. In this case relative error quickly reaches zero for $\tau = 0.20$; models of increasing number of clusters preserve this remarkable accuracy. When $\tau = 0.15$ relative error goes down to 0.01 and remains the same through $k = 800$, this also represents a significantly high accuracy. For lower τ thresholds there is a steady decrease of relative error as k increases. The decrease is more significant for the lowest τ threshold. Therefore, these results confirm k is the main parameter to control accuracy of support estimation.

Varying the minimum support threshold τ

Table 5 provides a detailed analysis of relative error as τ decreases keeping k fixed. We set $k = 80$ for the real data sets and $k = 400$ for the synthetic transaction data set. The table includes p, the length of the longest frequent itemset and the total number of frequent itemsets discovered. For the Chess data set it was not possible to generate frequent itemsets longer than 5 because their number reached over 1 million. Therefore, we decided to set the length threshold at $p = 5$, which still produced an extremely large number of frequent itemsets. For all data sets, the number of itemsets grows rapidly as τ decreases. On the other hand, relative error grows relatively slow for real data sets ands it grows fast for the synthetic data set. The growth of relative error is more related to the growth on the number of itemsets than their length.

Varying the centroid threshold (λ)

Table 6 shows accuracy behavior as λ varies. Recall that the main purpose of λ is eliminating spurious (false positive) itemsets when generating itemsets from the model. First of all, relative error always decreases as λ increases. For two data sets the effect of λ is outstanding. For the synthetic data set the decrease is remarkable bringing down relative error from more than 90% at $\lambda = 0$ to 8% at $\lambda = 0.2$. For the Votes data set the decrease is also significant going from 14% to 8% at the lowest λ levels. For the Chess and Mushroom data sets relative error decreases much slower, but it was very low to start with. Second, the percentage of itemsets that are indeed frequent after verifying their support shows a steady increase as λ increases. When λ

Table 5: Real data sets: Accuracy and number of itemsets varying τ (relative support error) with $k = 80, \lambda = 0.2$.

Data set	τ	error	p	no. of itemsets
Chess	0.6	0.01	5	34672
	0.4	0.02	5	173001
	0.2	0.03	5	420878
Mushroom	0.6	0.00	5	51
	0.4	0.00	7	565
	0.2	0.01	15	45425
Votes	0.4	0.00	3	26
	0.2	0.02	7	302
	0.1	0.07	8	1014
T10I4D1M	0.20	0.00	2	21
	0.10	0.03	6	184
	0.05	0.09	8	842
	0.02	0.13	9	2858
	0.01	0.28	10	10566

Table 6: Accuracy varying λ.

Data set	λ	error	no. of itemsets	frequent	discovered
Chess	0.0	0.03	173309	58%	100%
k=80	0.2	0.03	172790	63%	99%
τ=0.4	0.4	0.03	169558	76%	98%
	0.6	0.02	123687	97%	71%
	0.8	0.02	41817	100%	24%
Mushroom	0.0	0.02	45439	86%	100%
k=80	0.2	0.02	45423	88%	99%
τ=0.2	0.4	0.00	43454	97%	96%
	0.6	0.00	41803	100%	92%
	0.8	0.00	41421	100%	91%
Votes	0.0	0.14	1028	50%	100%
k=80	0.2	0.08	1013	76%	99%
τ=0.1	0.4	0.05	891	90%	87%
	0.6	0.04	596	98%	58%
	0.8	0.02	338	100%	33%
T10I4D1M	0.0	0.90	1004	5%	100%
τ=0.05	0.2	0.08	835	96%	87%
k=400	0.4	0.08	769	98%	77%
	0.6	0.07	696	98%	69%
	0.8	0.06	559	100%	56%

Table 7: Comparing accuracy: clustering versus correlation.

Data set	Model	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$	$p=6$	avg
Chess	clustering	0.00	0.01	0.02	0.02	0.03	0.04	0.03
τ=0.4	correlation	0.00	0.00	0.02	0.11	0.25	0.38	0.33
Mushroom	clustering	0.00	0.00	0.01	0.03	0.03	0.03	0.03
τ=0.2	correlation	0.00	0.00	0.15	0.24	0.42	0.54	0.44
Votes	clustering	0.00	0.02	0.04	0.06	0.11	0.21	0.06
τ=0.1	correlation	0.00	0.00	0.31	0.39	0.69	0.88	0.65
T10I4D1M	clustering	0.00	0.05	0.09	0.11	0.13	0.15	0.09
τ=0.05	correlation	0.00	0.00	0.30	0.69	0.88	0.96	0.76

reaches 1 100% of itemsets are frequent, highlighting good accuracy for items that cluster well. Third, since λ prunes out some frequent itemsets we also analyze what % of itemsets are retained as λ increases. The bottom value $\lambda = 0$ represents an unrestricted mining from the model where all itemsets are discovered (at the price of some false positives). The maximum λ value represents a restricted mining process looking only at items where the model behaves almost perfectly. We can see $\lambda = 0.8$ eliminates a significant fraction of itemsets for all data sets, except Mushroom. On the other hand, $\lambda = 0$ maintains 100% of frequent itemsets, but there are additional itemsets whose support is actually below the τ threshold. Clearly, there is a tradeoff between the % of frequent itemsets and the % of discovered itemsets. The best λ thresholds are 0.2 and 0.4. $\lambda = 0.2$ is a threshold that maintains about 90% of discovered itemsets and at the same time eliminates a significant fraction of spurious itemsets. Therefore, $\lambda = 0.2$ is a good default value.

Comparing Clustering and Correlation Accuracy

Table 7 compares clustering and correlation for itemsets of increasing length on all data sets. The table also includes the relative error average. These accuracy measurements were done at low τ values to test the models under stringent conditions. Both models are perfectly accurate for 1-itemsets, as expected. Clustering provides excellent results for 2-itemsets, slightly behind correlation, which is 100% accurate. For itemsets of length 3 to 6 clustering is significantly more accurate than correlation, with a relative error rate about five times better. For itemsets of length 6 correlation approaches 100% relative error for the Votes and the synthetic data set, and it is around 50% for the two other data sets. Since the number of itemsets grows in an exponential manner as p increases the relative error average favors clustering. These results show clustering is a better model than correlation to find frequent itemsets. However, correlation represents a fast alternative for itemsets up to length 3, where a higher error (around 30% in our case) may be acceptable to the user.

5.3 Time Performance

Comparing clustering, correlation and A-priori

Figure 5 compares processing time for the models with A-priori on T10I4D1M, which has one million records by decreasing τ (minimum support threshold). Notice these plots use a logarithmic scale for time: differences among methods are one order of magnitude or more. For the clustering model we use $k = 100$ (a medium accuracy model) and $k = 400$ (a highly accurate model). The left graph in Figure 5 compares both models with A-priori to discover frequent itemsets, excluding the time to compute the respective models; such time measurements do not include the time to build models based on the assumption that a model is built once (or a few times), but association rules are mined frequently. In this case we only show the clustering model at $k = 400$, which is four times slower than $k = 100$. These times to compute itemsets

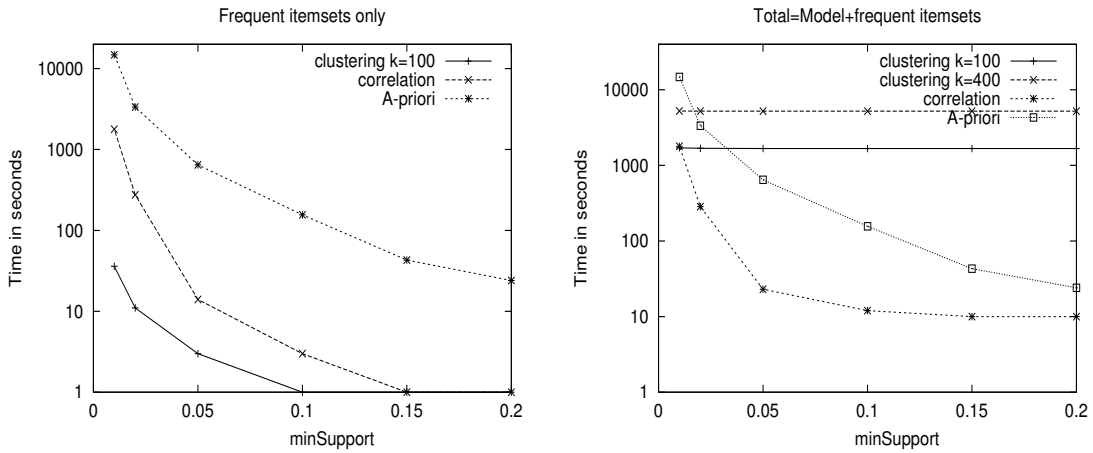

Figure 5: Time comparison with large transaction data sets decreasing τ.

includes the time to compute exact support on a final pass over the data set. As can be seen, the clustering model is the fastest, followed by correlation, leaving A-priori far behind in the third place. The basic reason for the speedup is that the clustering model represents a summarized (compressed), yet accurate, version of the large data set, preserving its statistical properties.

The right graph in Figure 5 includes the overall time to find frequent itemsets, including the time to build the models. That is, we evaluate time performance under pessimistic conditions. Since the time to build the clustering model dominates the overall time for the clustering model its plots are practically constant lines. On the other hand, the trend for the correlation model is practically the same as the trend to find only frequent itemsets because the time contribution to build the correlation model is marginal. A-priori is two orders of magnitude faster than the clustering model at high support levels, but it becomes slower at low support levels. The trend indicates the clustering model will be the winner as the minimum support threshold decreases. The combinatorial growth of itemsets impacts all methods, but the impact is more significant for A-priori. We investigated why correlation was so slow at low support levels and it turned that it generated an extremely large number of itemsets, most of which ended being below the τ threshold. That is, loose support bounds impacted time performance. From these experiments we conclude that a clustering model represents an efficient mechanism to discover frequent itemsets, even at low support levels, which are particularly challenging. There is a high price to build an accurate clustering model, but in general that is done only once, or a few times. On the other hand, the correlation model becomes inaccurate at low support levels and such inaccuracy actually makes the process slower since most long itemsets are infrequent.

Time Complexity and Speed to Compute Models

In this section we evaluate scalability and speed with large, high dimensional, data sets to only compute the models, as shown in Figure 6. 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 [27], introduced on Section 3. The sum of cross products matrix Q for correlation is efficiently derived with with the sparse matrix computations introduced in Section 4. Large transaction files were created with the IBM synthetic data generator [3] having defaults $n = 1$M, T=10, I=4.

The left plot in Figure 6 shows times to build the clustering and correlation models with a data set with one million records (T10I4D1M). As can be seen times grow linearly as n increases, highlighting

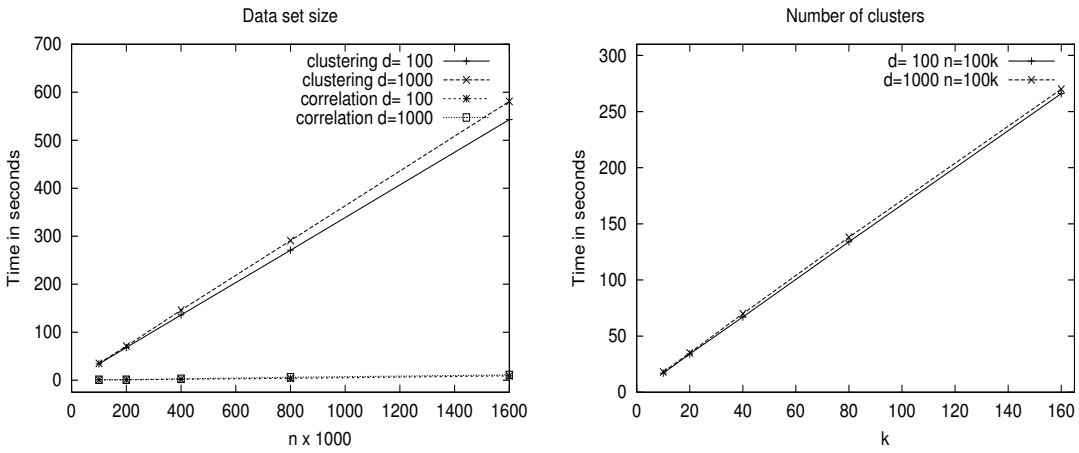

Figure 6: Time to compute statistical models for large data sets.

the algorithms efficiency. Notice correlation is significantly faster than clustering: its elapsed times are two orders of magnitude smaller. On the other hand, we can observe d has minimal impact on performance when it is increased 10-fold on both models; sparse matrix computations take advantage of the small transaction size, which remains constant on both data sets.

The right plot in Figure 6 shows time to get clustering models increasing k on T10I4D100k. Recall k is the main parameter to control model complexity and support estimation accuracy. In a similar manner to our previous analysis, times are plotted for two high dimensionalities, $d = 100$ and $d = 1,000$. We can see time complexity is linear on k and d has a marginal impact on processing time.

5.4 Summary

We draw the following general conclusions from accuracy experiments. Both clustering and correlation are built without support thresholds, which helps uncovering patterns at low support levels. Clustering requires tuning two parameters, the number of clusters and the centroid threshold, being the number of clusters the most important (the higher the better). On the other hand, correlation has no parameters. Clustering support bounds tightness depends on clustering quality, whereas bounds derived from the correlation auxiliary matrix do not depend on how correlated dimensions are. Both clustering and correlation analysis are accurate for short itemsets and show a decrease in accuracy as the minimum support threshold is lowered. The clustering model provides highly accurate support estimations for longer itemsets, given a high k. In general, clustering is more accurate than correlation to bound and estimate support, except for 2-itemsets, where it is slightly less accurate. Correlation analysis gives exact support estimation for 2-itemsets and a reasonable support approximation for 3-itemsets. Picking the right number of clusters will depend on how well the data set clusters and how accurate bounds and estimations are desired. The clustering model provides three main advantages: the model provides a summary of statistical characteristics of the data set, the model allows getting approximate values of support, and the model can be maintained in main memory since it is generally much smaller than the data set.

This is a summary from a time and space efficiency perspective. If already available, the clustering model is faster than A-priori and correlation to search frequent itemsets. Decreasing support impacts performance on all alternatives due to the combinatorial growth on the number of itemsets. In general, the clustering model is much smaller than a large data set: $O(dk) < O(dn)$. The clustering model takes less space than the correlation matrix when $k < d$ since it is $O(dk)$; clusters generally provide a more concise summary about the data set than correlation. On the other hand, the correlation matrix takes quadratic space

($O(d^2)$) and provides a less concise and less accurate summary of the data set. Both clustering and correlation can be computed in linear time with respect to data set size. The correlation matrix can be computed much faster than the clustering model.

6 Related Work

There has been a lot of work on both scalable clustering [1, 30, 29] and efficient association mining [26, 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, 29]. Clustering binary data is related to clustering categorical data and binary streams [27]. The k-modes algorithm is proposed in [20]; 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 [13]. 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 [13] 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 [11], but our point is that clusters can be used to discover associations.

There exist many efficient approaches to mine associations. 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 [19]. Reference [19] proposes an efficient algorithm based on a heap than can efficiently mine frequent itemsets with new records or decreasing support; compared to our approach, both clustering and correlation can be efficiently and incrementally updated through their sufficient statistics and since they are statistical models they are independent from any minimum support level. Others resort to statistical techniques like sampling [38], statistical pruning [26]. In [35] 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 statistical models like our proposal. 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. Another important issue is filtering out spurious rules [14]; the authors propose improved confidence and lift metrics which are more robust to noise in the data set. Reference [45] 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 [12] the authors derive a tight bound on the number of candidate itemsets given the current set of frequent itemsets, when using a level-wise algorithm. Our work provides a less tight bound, but from a statistical perspective. 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.

Related work on relationships between association rules and other techniques follows. Preliminary results on using clusters to get lower and upper bounds for support is given in [28]. On the other hand, a closely related algorithm that derives association rules from clusters is presented in [23]. The authors show clusters can be used to compute frequent itemsets faster than A-priori and sampling-based methods; their support approximation is related to our proposed upper support bound; they also use a threshold to eliminate unlikely itemsets from each cluster. There are several important differences though. First, they consider clusters as a mechanism to compress the data set rather than a statistical model; the authors use a hierarchical clustering algorithm based on the Jaccard coefficient, whereas we use the popular K-means algorithm based on distance; their algorithm requires a similarity threshold for the Jaccard coefficient; our

approach can produce a good clustering model in just two passes over the data set; such approach may miss some frequent itemsets above a support threshold; their threshold to eliminate unlikely itemsets is related to itemset similarity whereas our is related to the cluster mean; in addition we also propose lower bounds for support and our estimation is based on the bounds average. From a statistical perspective, we show how to derive sufficient statistics for clustering and correlation. In general, there is a tradeoff between rules with high support and rules with high confidence [34]; 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 [44] 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. Mining ratio rules based on an improved PCA technique is proposed in [18]; our correlation bounds can be generalized to a PCA model. Reference [43] analyzes relationships between item correlations and association rules and derives a bound to prune pairs of items; the authors focus only on positive correlations above a user-specified threshold and defend the idea of not computing the entire correlation matrix. In contrast, we have shown the entire correlation matrix can be efficiently computed because each transaction is a sparse vector and we studied how association rule metrics are linked to correlation: correlated and uncorrelated items can indeed appear together.

7 Conclusions

This article studied mathematical relationships between association rules and two fundamental models: clustering and correlation. Models help understanding statistical properties of a binary data set for association rule mining and can be used to approximate support and confidence. Sufficient statistics for clustering consist of the linear sum of points, whereas sufficient statistics for correlation are the sum of squared points. Each cluster represents an itemset and the sufficient statistics for correlation measure support for all pairs of items. The clustering model and the correlation matrix can be efficiently updated with sparse matrix operations to process large, high dimensional, data sets. We introduced two families of association rule metrics based on clustering and correlation, respectively. Each family of metrics provides bounds and estimations for support and confidence, the standard metrics for association rules. Confidence bounds and estimated confidence are based only on support bounds. Support metrics exhibit the set downward closure property for efficient bottom-up search for frequent itemsets. The clustering model uses the number of clusters as the main input parameter, whereas the correlation model does not have any parameters. We introduce a secondary parameter, used during frequent itemset search from clusters, to filter out itemsets unlikely to be frequent, by including only item combinations above a centroid threshold that gives more weight to item co-occurrence. We conducted experiments with real and synthetic data sets to evaluate accuracy and time performance, focusing on frequent itemset search given a minimum support threshold. Accuracy was evaluated with the average of relative error on support estimation. In general, clustering models with a sufficiently high number of clusters are accurate. Relative error decreases with a higher number of clusters, showing asymptotic behavior, but grows slowly as itemset length increases. That is, accuracy increases as the number of clusters increases, but it decreases as the minimum support threshold decreases. The minimum centroid threshold is set to a low value greater than zero that preserves most frequent itemsets, but excludes a significant fraction of infrequent itemsets. Relative error on support estimation grows as itemset length increase with both models, but it grows at a higher rate for correlation. We compared the models with the A-priori algorithm as mechanisms to find frequent itemsets. Excluding the time to compute models, when decreasing minimum support clustering is the fastest mechanism, followed by A-priori, and correlation is left on the

third place; the three techniques are impacted by the combinatorial growth of the number of itemsets, but A-priori is more heavily impacted. Considering the overall time to build the model and to find frequent itemsets the clustering model is the slowest alternative at high support levels, but the fastest at very low support levels. The reason correlation had bad time performance is because its bounds are loose for long itemsets, making the algorithm generate an extremely large number of itemsets which turn out to be infrequent. From a time complexity perspective, both models can be computed in linear time with respect to data set size, but correlation is significantly faster than clustering. For sparse binary data sets, such as transaction data sets, high dimensionality has a marginal impact on speed due to the efficient sparse matrix operations. In short, clustering represents a more accurate, but also more expensive, alternative than correlation to understand a data set from a statistical perspective. Correlation on the other hand, is very fast to compute, is completely accurate for associations with two items and can get reasonable support bounds for short itemsets. Therefore, clustering and correlation models complement each other to explore a data set for association rules.

There are several directions for future work. It is feasible to build a hybrid model combining clustering and correlation. For instance, the correlation matrix can be computed for a subset of items in one tight cluster or conversely, a clustering model can be computed for a set of points having a subset of uncorrelated items. The clustering model and the correlation matrix can be incrementally updated on-line through sufficient statistics, allowing maintenance of a set of interesting association rules. Even further, both models enable the statistical analysis of streaming data. Incorporating constraints into the models to mine predictive association rules is another important direction, with wide applicability. The correlation matrix leads to principal component analysis and factor analysis, which are fundamental techniques to analyze high dimensional data sets. Therefore, we can create sophisticated descriptive models combining mixtures of probabilistic distributions and factor analysis under a maximum likelihood estimation framework, like the Expectation-Maximization algorithm. We want to derive analytical bounds on relative error for both models under different probabilistic distributions.

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] A.P. Dempster, N.M. Laird, and D. Rubin. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of The Royal Statistical Society*, 39(1):1–38, 1977.

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

[12] 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.

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

[14] M. Hahsler and K. Hornik. New probabilistic interest measures for association rules. *Intelligent Data Analysis*, 11(5):437–455, 2007.

[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] C. Hu, B. Zhang, Y. Wang, S. Yan, Z. Chen, Q. Wang, and Q. Yang. Learning quantifiable associations via principal sparse non-negative matrix factorization. *Intelligent Data Analysis*, 9(6):603–620, 2005.

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

[20] 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.

[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] F. Liu, Z. Lu, and S. Lu. Mining association rules using clustering. *Intelligent Data Analysis*, 5(4):309–326, 2001.

[24] U. Maulik and S. Bandyopadhyay. Performance evaluation of some clustering algorithms and validity indices. *IEEE Trans. Pattern Anal. Mach. Intell. (TPAMI)*, 24(12):1650–1654, 2002.

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

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

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

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

[29] 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.

[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] C. Ordonez and E. Omiecinski. Accelerating EM clustering to find high-quality solutions. *Knowledge and Information Systems (KAIS)*, 7(2):135–157, 2005.

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

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

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

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

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

[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] N.A. Weiss. *Elementary Statistics*. Addison-Wesley, 2005.

[43] H. Xiong, S. Shekhar, P.N. Tan, and V. Kumar. TAPER: A two-step approach for all-strong-pairs correlation query in large databases. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 18(4):493–508, 2006.

[44] 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.

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

[46] 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.