Research Survey

Using clustering techniques to find Association Rules for Continuous Attributes

Authors:

[redacted]

Student ID:

[redacted]

Contents

List of Figures

1 Problem Domain

The field of data mining has received particular attention in the last two decades, from a range of diverse sectors spanning commerce, marketing, defence, education and scientific research. Data mining presents opportunities for substantial improvements in profitability, quality of service and process refinement.

For example, businesses hope to better understand their customers to enable more targeted marketing campaigns, and to refine their products and services. They also need better-informed predictions of market patterns to enable more profitable management of resources and assets. Education providers aim to better understand their students' learning styles and challenges, allowing them to refine their curriculum and modes of delivery. Scientists collect large volumes of sensory data from sources such as biological experiments, environmental observations or chemical processes, and seek to form sound hypotheses and conclusions from this data. Engineers in infrastructure sectors seek to design intelligent systems which respond to changes in demand, load, constraints, and distributions thereof, based on historical and current monitoring data.

The promise of extracting new levels of insight from large volumes of collected data is highly enticing, yet at times highly elusive. The greatest challenges in data mining revolve around: the sheer volume and orders of magnitude; the diversity and complexity of the data collected; and the extraction of subtle information hidden in this data.

1.1 Data Volume

Data mining operations are typically performed on massive volumes of data. A supermarket, for example, may record 15,000 transactions per week[1], and hence a data mining operation which covers the last 5 years must process over 4 million complex transactions for each store, with each transaction potentially consisting of multiple products. For a national chain, this figure further increases by several orders of magnitude. This scale imposes strict constraints on the computational complexity of algorithms used to process this data. Not only may a system be expected to process non-trivial volumes of data from day 1, but it may also be required to scale by several orders of magnitude within the first few years – or even months – of its deployment, in line with the organisation's growth. Due to both the size and volatility of magnitudes involved, algorithms with complexities such as $O(n^5)$ or $O(n^n)$ quickly become problematic. However, efficient algorithms are difficult to develop given that the problems being solved are commonly non-trivial (see below). Algorithms and processing techniques are continuing to emerge and evolve, as is discussed in this research survey.

1.2 Data Complexity

Data mining is often required to accommodate diverse data types and be able to process them effectively. This is because data mining is commonly used to better understand the real world, which is often far from simple or discrete. While data attributes may in some cases be boolean, richer attributes are commonly found in the form of categorical or continuous data[2]. Some examples include an entity's class (car model, city, colour), person's age, monetary value, date/time, item quantity or spacial measurement. Since these classes of attribute types carry more information, they also complicate the comparison process which is required when identifying relationships and patterns within the data. This leads to more sophisticated algorithms and compounds the computation costs posed by massive data volumes. Simplifying this rich data would reduce the challenge, however categorical data cannot be meaningfully simplified (unless a taxonomical hierarchy exists), and continuous data may simplified by partitioning however this presents a tradeoff: cost vs. quality. Simplifying the data reduces the computational complexity and cost, while also discarding information in the data and hence risking a serious degradation in the quality of insights extracted. Clustering and other forms of machine learning have been examined as more intelligent and less "lossy" methods of data simplification, as discussed in this paper, though solutions appear non-trivial.

1.3 Information Subtlety

Data mining is typically employed to expose data characteristics which are not directly apparent. Simpler techniques such as statistical averaging are frequently found to be inadequate, since the insights presented are of limited usefulness, and the data being analysed is often diverse and may reflect many contributing factors. For example, while a city planner may benefit from knowing the city's average property value, it may be far more useful to identify correlations between property values and geographical regions, population density or zoning. Data mining is hence required to go deeper by exposing more subtle correlations among data, gleaming more profound insights. This is both its greatest strength and greatest challenge.

Effective identification of such subtle correlations can be realised and enhanced by using machine learning techniques, such as classification, clustering and association rule learning. Each of these techniques are useful for different types of problems, and each exposes different insights. Care must be taken, however, when applying algorithms from these techniques since each algorithm may impose a model that risks oversimplifying the problem domain, and hence the information exposed may be of degraded quality. The data simplification problem previously mentioned is a key example.

2 Problem Definition

The goal of this project is to develop a method for extracting association rules from continuous data, using clustering. Since association rules do not naturally lend themselves to continuous attributes, we believe that clustering is a feasible method to bridge this gap, enabling new kinds of analysis.

Continuous data has previously presented a challenge when attempting to extract meaningful association rules from a dataset. This is due to the *if* $\Longrightarrow$ *then* implication nature of association rules, a logical relation which lends itself to boolean observations[3]. Conversely, continuous data is far from boolean in nature.

A most basic approach of approximating boolean relations in continuous data is equality testing. In this case, every possible value for the continuous attribute may be considered as a discrete transaction attribute. However, continuous data may have a notably high resolution, in which case the discovery of transaction attributes with "equal" values will be too rare to be useful, and will be excluded by many algorithms due to lack of statistical support[2]. Additionally, the number of possible association rule attributes which may be enumerated from a single continuous attribute's values may be computationally prohibitive.

An improvement over this approach would be inequality testing: partition the values into intervals, or equivalently, reduce the resolution of the continuous attribute. The result would be that similar or approximate values would be considered "equal", which makes intuitive sense: people of age 20.1 years and 20.2 years could have similar shopping habits from which meaningful associations may be formed, compared to people of age 60 years. However, this raises the question: how close is close enough, or what should be the size of these intervals? Intervals too large risk developing associations which include diverse transactions where they are less meaningful, reducing the confidence of the associations since it is possible that many transactions may not "fit well"[2]. They also omit meaningful information such as if an association is due to many transactions of a precise value, in contrast to a more broadly distributed range of values.

Applying clustering to continuous attributes may produce more meaningful partitioning, reflecting patterns in non-uniformly distributed values. However, it is probable that meaningful clusters may not be directly observable in a given continuous attribute alone, but rather only when in the context of a correlation between other attributes. Hence, the clustering of association rules should instead be considered.

This presents several challenges. Developing association rules from continuous data will produce large volumes of association rules, which imposes a significant cost on a transactional database. Hence, mining methods should ideally minimise the number of passes and database reads overall, while deriving and clustering associations.

As our work is to combine several areas of data mining in general, in terms of building the knowledge needed to create a method for mining association rules on continuous attributes we need to look into the related work in association rule mining in general, the clustering of association rules, and other methods of mining association rules on continuous data attributes.

3 General Association Rule Mining

In general when we are discussing Association Rule Mining we are talking about mining association rules on categorical or binary data. An association rule can be thought of as an implication between two sets of attributes of the form $X \implies Y$ which have two measures associated with it: confidence, and support[4]. The confidence of an association rule c in a set of transactions is the percentage of transactions containing X also contain Y ($Pr(X|Y)$[5]). The support of an association rule s in a set of transactions it the percentage of transactions that contain $X \cup Y$ ($Pr(X \cap Y)$ [5])

There have been several key algorithms which have been created to find these associated rules. The major algorithms we will discuss include:

- Apriori [4]
- Partition [6]
- FP-Growth [7]

3.1 Apriori Algorithm

The Apriori algorithm was designed to solve the finding of any association rule which satisfies both a minimum support and minimum confidence. If the rule satisfies both of the required values then it is deemed a significant rule. This algorithm has been used as the foundation for several other key algorithms including the Partition algorithm previously mentioned. The paper introduces two groups of itemsets in regards to the support of a rule. If a particular itemset has the minimum support, then it is called a *large itemset*, otherwise the itemset is known as a *small itemset*.

The algorithm is completed in two phases:

1. Itemset generation and classification. (Deals with the support of the rule)
2. Association rule generation from the large itemsets (Deals with the confidence of the rule)

In order to generate the itemsets, the algorithm generate candidate itemsets and checks whether each of the itemset has the minimum support by scanning through the data set. The number of scans of the database is dependent on the number of iterations needed to find all itemsets with the minimum support. This can cause the runtime of the algorithm to be high as the overhead of a database read can be quite high.

3.2 Partition Algorithm

The partition algorithm was designed to improve the Apriori algorithm by decreasing the number of reads from the database significantly, with a maximum of two database reads independent of both the minimum support and the number of partitions. This is in contrast to the Apriori algorithm which reads the database a variable number of times based on the minimum support, and other characteristics of the data.

In order to restrict the number of database reads the Partition Algorithm uses a single scan to get a superset of all possible large itemsets, it then uses partitioning to allow the calculation of all of the supports of a itemset to be computed in a non-exponential time. The Partition Algorithm has two phases:

1. Partition the database logically and generate large itemsets for each partition separately.

2. For each large itemset that is found in phase one, find the support of the itemset in the global scope. (entire database)

While one of the performance weakness of the Apriori Algorithm is in the I/O operations, the Partition Algorithm eliminates this issue. As with most algorithms there is a performance issue with the Partition Algorithm. As the Partition Algorithm is a divide and conquer algorithm, the divide step gives rise to the efficiency of the algorithm as a whole. For maximum performance there are two factors in choosing a partitioning scheme for the data:

1. Minimise the number of partitions.

2. Minimise the number of elements within the partitions.

These two factors show that the Partition Algorithm requires some trade-off to be complete for optimal performance. The number of partitions needs to be minimised as the collected itemsets generated in the first phase will have too many false positives within it, meaning that the performance of the second phase is impacted poorly. The size of the partitions also needs to be minimised for two reasons. Firstly if the partition size is too large, then the search space for the itemsets will still be large which is the reasoning behind partitioning the data set to begin with. Secondly, if the size of the partition does not fit easily into the data cache, or main memory when this happens a severe degradation in the performance of the algorithm will occur due to "*thrashing*"

3.3 FP-Growth Algorithm

The paper that introduced this algorithm wanted to create the FP-Growth algorithm as an alternative to the Apriori algorithm, or similar algorithms. The motivation behind this algorithm is that they found a different bottleneck within the Apriori Algorithm than the Partition Algorithm did, this was the candidate generation phase of the algorithm that is present in both Apriori, and Partition. In order to get around this phase the FP-Growth algorithm introduces the concept of frequent pattern mining.

Pattern A subset of items in the superset I of all items within the transaction database D.

Frequent Pattern A pattern which has a support larger than a minimum threshold support, where the support of the pattern is the number of transactions containing the pattern within D.

To perform frequent pattern mining, the idea of a FP-tree is introduced, this is a compact data structure where each node of an item prefix subtree consists of item name, count, and node-link. Item name and count are self-explanatory in this case, but the node link is the link to the next node in the entire FP-tree with the same name (in order to cross to another prefix subtree as required). The FP-tree has a single root node, which has a set of the item prefix subtree as its children. The other component of a FP-tree is a frequent item header table which contains entries of the node name, and a pointer to the first node (header node) within the FP-tree.

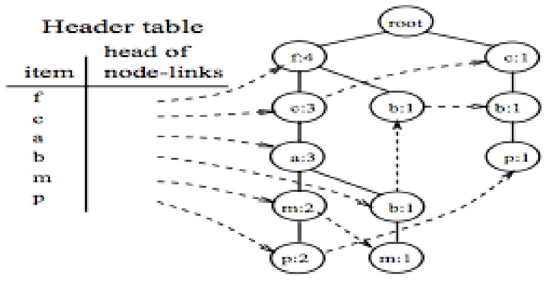

Figure 1: Example FP-Tree[7]

In order to build the FP-tree we first need to get the frequent items and their supports from the transaction database. We then sort these items according to their supports in descending order. Then for each transaction in the database, we assign the first element in the sorted list to be a child of the root if there is no such element already there and then construct the rest of the tree recursively updating the count where there is a duplicate node. Like the Partition Algorithm, this tree construction takes two reads of the database the first is to generate the frequent items and their supports, the second to perform the construction of the FP tree.

Now that we can get an FP tree, the algorithm to generate the frequent patterns is to recursively follow the paths of the tree, and build up the set of the frequent patterns while updating the support for each pattern. In order for these patterns to be frequent this needs to be done so that the support of the pattern is greater than the minimum support threshold. An association rule can be built from any frequent pattern as any subset of the items within the pattern can be associated with the other items in the pattern. e.g. for the frequent pattern f, g, h, k, o an equivalent association rule could be: $[f, g, h] \implies [k, o]$

3.4 Extensions on Association Rule Mining

While association rule mining is used to generate associations between different categories of items it suffers from several weaknesses. One of the weaknesses is the inability to check for negative association i.e. if one set of items is absent then another set is likely to be present. This weakness leads to the introduction of a third measure on an association rule, known as its correlation/interest[8]. This third measure can be calculated in numerous ways, these include:

- Collective Strength [8]
- Chi-Squared Independence Test[9]
- Interest [10]
- Conviction [10]

3.4.1 Collective Strength

The collective strength is a way to consider the correlation of a itemset as a number in the range $[0, \infty]$ this measure is used to eliminate those cases where the interest of a rule is defined to be close to one (barely interesting) when they are perfectly correlated. It does this by defining to categories of transactions for the rule:

1. violating transactions
2. non-violating transactions

A transaction is a violating transaction if some of the items within an itemset are present in the transaction but others are not. The violation $v(I)$ is the fraction of violating transactions over the entire transaction space. The collective strength of an itemset I can be defined by:

$$C(I) = \frac{1 - v(I)}{1 - E[v(I)]} * \frac{v(I)}{E[v(I)]} \tag{1}$$

Where $E[v(I)]$ is the expected violation rate assuming independence, and $1 - v(I)$ is the rate of non-violating transactions.

3.4.2 Chi-Squared Test

The chi-squared independence test is a well known statistical measure of the independence of a set of items. The chi-squared test measures the normalised deviation from the expectation of independence. This is done by the following formulae:

$$\chi^2 = \sum_{r \in R} \frac{(O(r) - E[r])^2}{E[r]} \tag{2}$$

Where O(r) is the observed number of instances falling into cell r, R is the set of all possible items allowing for all combinations to be explored. A major flaw in the Chi-Squared Independence test is that it is computationally expensive and thus not appropriate for large datasets.[8]

In the collective strength and chi-squared tests an expected value is required assuming independence. This expected value can be generated in a couple of ways. The first is to assume that every value has equal probability i.e. uniformly distributed. The other is to assume that the distribution of values is not uniform but does follow some distribution curve e.g. normal distribution, t-distribution, etc. If it is unknown what distribution to use we can use some known test data to check the quality of the analysis under the different types of distribution.

3.4.3 Interest and Conviction

The interest of an association rule of the form $A \implies B$ is calculated using:

$$I = \frac{Pr(A \wedge B)}{Pr(A)Pr(B)} \tag{3}$$

Where the further the interest is from one the higher the dependence between the two itemsets (the stronger the association rule)

The conviction of an itemset was designed to address the issue with interest being symmetric, meaning that the interest of $A \implies B$ is equal to the interest of $B \implies A$ this has been covered by using the conviction formulae:

$$C = \frac{Pr(A)Pr(\overline{B})}{Pr(A \wedge B)} \tag{4}$$

4 Mining Association Rules on Continuous Data

In general association rules, each side of the implication is made up of boolean, or categorical items. This means that for every rule an item is within an itemset on an equality. We now need to consider the issue of mining association rules on continuous attributes. The main issue in mining quantitative rules is the mapping of quantitative attributes to their categorical counterparts[2]. The simplest way of mapping between quantitative and categorical association rule mining is to generate equality items for every possible value of the quantitative attribute of interest, this will work for small ranges of possible values but any large range will become computationally expensive, and the rules generated will not have the required statistical support to be classified as an interesting rule[2]. This means that we need to generate a new definition for an itemset of an association rule with a quantitative attribute. There are two competing definitions for these:

1. Interval based association rules

2. Statistical property based association rules

For the definition of an interval based association rule, the notation is as follows: $(A \in [x, y]) \implies B$ where $x < y$. This says that if the value of the quantitative attribute A is in the range [x,y] then B is true. B can also be a quantitative attribute so that a range of values on one quantitative attribute can imply a range of values on another quantitative attribute.[11]

For the definition of the statistical property based association rule we define the following notation: population subset $\implies$ mean value for the subset, where the rule is interesting if the mean of the population subset is significantly different from the mean value of the whole population. This definition is designed to cover the issue with the interval based mining where the interval has high support but some values in the interval has little to no statistical significance.[12]

4.1 Interval Based Association Rules

There are several algorithms that have been used to generate the intervals for interval based mining. These algorithms include:

- Convex Hull Bucketing Algorithm
- Kadanes Algorithm
- Optimal Rectangle Finding

4.1.1 Convex Hull Bucketing Algorithm

In order to explain the workings of the bucketing algorithm[11] we need to define some key terms in the algorithm.

Buckets of a domain A sequence of disjoint ranges. i.e. $[x_i, y_i]$ where $x_i \leq y_i < x_{i+1}$

Interval The combination of sequential buckets of a particular domain/attribute. e.g. the interval $[x_s, y_t]$ is the combination of buckets $B_s, B_{s+1}, \ldots, B_t$.

Convex Hull The smallest polygon where a line between any two points is entirely within the polygon i.e. no point is outside of the polygon.

The objective of a bucketing algorithm is to find some interval (or sequence of buckets) that has a support above a defined minimum support, an interval with this property is known as *ample*. The support of an interval based rule is the number of tuples within the interval[13]. An optimal interval is an ample interval whose rule has a maximum confidence. In order to generate this interval a geometric algorithm approach can be used. If we define bucket $B_i = [u_i, v_i]$, we can define a sequence of points

$$Q_k = (\sum_{i=1}^{k} u_i, \sum_{i=1}^{k} v_i) \tag{5}$$

with $Q_0 = (0, 0)$. The confidence of interval $[m+1, n]$ is defined to be the slope between Q_m and Q_n, and the support of the interval $[m+1, n]$ is defined to be $x_m - x_n$ where x_m is the x coordinate of Q_m, and x_n is the x coordinate of Q_n. In order to find the optimal range we can use a convex hull algorithm. A convex hull of a set of planar points (2D points) can be discovered using an O(nlogn) algorithm[14] If we define an upper hull of the set of points between Q_m and the last point to be U_m, and the value $r(m) = min\{i|m + 1 \leq i$ is an ample pair $\}$ the optimal interval has the property such that the tangent of the line between Q_m and $U_{r(m)}$ is maximised with the point with the maximum x-coordinate for where the tangent touches the upper hull is the terminating point Q_n. With this information we can generate the optimal interval to be $[x_m, y_n]$ where x_m is the x-coordinate of Q_m and y_n is the y coordinate of Q_n

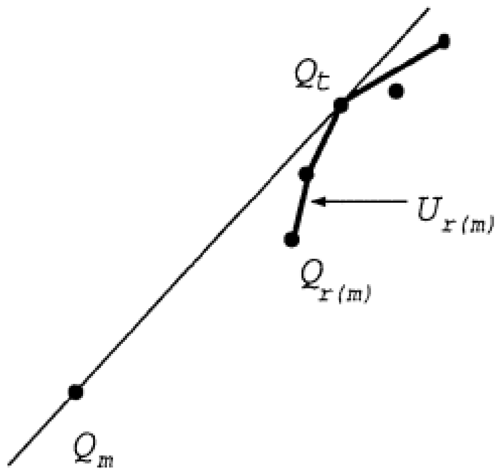

Figure 2: Upper hull tangents for convex hull algorithm[11]

4.1.2 Kadanes Algorithm

This algorithm uses buckets like the previous algorithm but runs in O(N) time to find the maximum gain interval, where N is the number of buckets. The gain of an interval I is calculated by:

$$gain_\theta(I) = \sum_{i=s}^{t} v_i - \theta * \sum_{i+s}^{t} u_i \tag{6}$$

In order to generate the maximum gain interval we generate an array (X) which contains the gain of every bucket. To find the maximum gain interval we use X[m,n] to be the gain of the interval [m,n] (the sum of the gains in X to be between the points m and n). To solve in linear time we need to use a dynamic programming approach. We define the following functions[15]:

$$Max(i) = max_{s\leq i}(X[s,i]) \tag{7}$$

$$Max(\leq i) = max_{s\leq t\leq i}(X[s,t]) \tag{8}$$

$$Max(i+1) = max\{0, Max(i)\} + X(i+1) \tag{9}$$

$$Max(\leq i+1) = max\{Max(i+1), Max(\leq i)\} \tag{10}$$

The final maximum gain range is found by $Max(\leq N)$ where N is the number of buckets. This range gives a good approximation of a high quality interval to use for quantitative association rule mining. The algorithm using convex hulls is more accurate in finding the optimal range but has a higher computational cost.

4.1.3 Optimal Rectangle Finding

In both the bucketing algorithms (Kadanes and Convex Hull) we only consider a singular dimension for the quantitative association rule mining i.e. we only looked into a single quantitative attribute. This means that we can not have a high quality association rule of the form:

$$(A \in [s,t]) \wedge (B \in [x,y]) \implies C \tag{11}$$

where A and B are quantitative attributes, and C can be either a quantitative or categorical attribute. In order to obtain a good pair of intervals we need to consider both at the same time, as if we just consider A and B separately when we combine the two intervals into one rule the quality of the rule (decided by support, confidence, and interest) might be less than it could be.

We call the highest quality interval pair, the optimal rectangle. In order to find the optimal rectangle we define a two dimensional plane which consists of a matrix of buckets of size N_A X N_B. For every transaction in the database denoted by t, $t[A]$ is the value of A in transaction t, and $t[B]$ is the value of B in transaction t. Then for every pair of buckets we compute the number of tuples falling inside of that pair (u_{ij}), and the number of tuples in the pair of buckets which hold to the conditional being considered (v_{ij}). We then define a region P to be a range of values in both A and B (a subset of G) which we can define the support and hit of P to be:

$$\mathit{support}(P) = \sum_{G(i,j) \in P} u_{ij} \tag{12}$$

$$\mathit{hit}(P) = \sum_{G(i,j) \in P} v_{ij} \tag{13}$$

We then define the gain of a region P to be similar to the gain of a one dimensional interval:

$$\mathit{gain}(P) = \mathit{hit}(P) - \theta * \mathit{support}(P) \tag{14}$$

As with the finding of the optimal gain interval, our goal now is to find the region P which maximises the gain.

In order to find the region P in a reasonable time we choose a pair of row indexes for the region, and compute the total gain of each column with the rows constrained by our the current considered rows. We then find the interval of consecutive columns which have the maximum gain using Kadanes' algorithm. Once we have done this for all pairs of rows we have found the region P with the highest gain for our association rule.

While rectangular regions produce easy to understand rules they do tend to overfit the data which can lead to incorrect associations being made for the general case outside of the data it is being run on. There are two methods of defining a two-dimensional region which try to solve this problem. These methods are:

- X-Monotone Regions
- Rectilinear Regions

An x-monotone region is defined to be a region where its intersection with every column is undivided[15]. An optimal x-monotone region can be found by finding the maximum gain region for each column of the plane and joining them with a union. While this does reduce the overfitting of the data in comparison with the rectangle regions the overfitting issue is still present and serious[13]. Another issue is that the bounds of the region is difficult to understand as it is "*notchy*".

A rectilinear region is a region which can be defined as a series of four sided regular polygons joined as a union. This implies that there are four types of basic rectilinear regions:

1. Widening from left-to-right.
2. Narrowing from left-to-right.
3. Upwards slant from left-to-right.
4. Downwards slant from left-to-right.

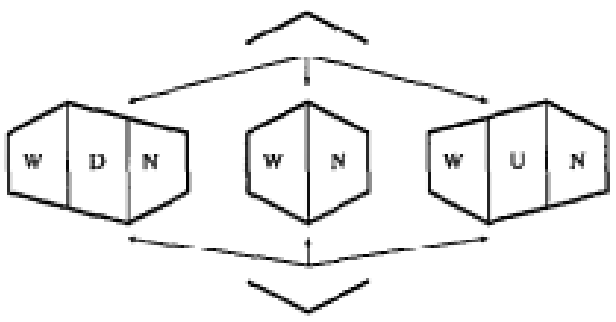

Figure 3: Possible combinations for three rectilinear regions[13]

Each of these types of basic regions can be joined as long as the horizontal lines only change from increasing to decreasing monotonically once. In order to compute the optimal rectilinear regions we need to find the best region of each type in terms of the gain of the region. We do this by generating a recurrence relation for each of the types of region. (See [13])

This algorithm addresses the issue of overfitting that is present in both rectangular and x-monotone optimal regions well. The regions defined by the rectilinear regions have a higher confidence than the rectangular regions, and have less variance in confidence between the testing and training data in comparison to x-monotone which shows that the overfitting issue has been improved significantly. Another strength of the rectilinear algorithm is that the confidence variance is stable independent of the number of buckets being used, which is not the case in regards to the x-monotone optimal regions algorithm.

While the performance of the rectilinear algorithm is worse than both the rectangle, and x-monotone algorithms dependent on the number of buckets being used, this independence in variance allows for the choice can be made for a smaller number of buckets if the confidence of the rules can be sacrificed for the performance of the algorithm and visa-versa.

4.2 Statistical Property Mining

As mentioned previously statistical property mining uses a different generic form of an association rule than the interval based association rules, this is of the form:

$$\text{population-subset} \implies \text{mean of values in subset} \tag{15}$$

This rule type is only interesting if the mean of the subset is significantly different from the overall population mean. There are two main types of these rules

Basic Rule An association rule that is not contained within any other rule.

Sub-Rule An association rule that is contained within another rule, but is significantly different from the top-level rule. (it is of interest)

The rules that we want to find (desired rules) have the property that they are either a basic rule, or they are a sub-rule of another desired rule. This second property allows for a recursive definition of a desired rule. The paper for these rules[12], have introduced two algorithms for the Quantitative $\implies$ Quantitative (the window algorithm), and the Categorical $\implies$ Quantitative.

4.2.1 Window Algorithm

The window algorithm takes in the transaction database, two quantitative attributes (x and y), and a minimum difference (or threshold value). The basic idea of the algorithm is to output rules of the form: $x \in [a, b] \implies$ Average of y is: . To achieve this we have the transaction database sorted by attribute x, this allows us to find contiguous sets of transactions for significant differences in mean. The window algorithm finds all rules where the region being considered are either "*above-average*" or "*below-average*", the term above/below average is a record where the attribute y is above/below the current mean +/- the minimum difference accepted. The rules generated by the algorithm are desirable iff. the variance is statistically significant (from Z-test). In order to find all the sub-rules of the currently considered rule, we recursively call the window function on the region of the database that was found to be the rule.

4.2.2 Mining Categorical $\implies$ Quantitative Rules

In order to generate rules of the form Categorical $\implies$ Quantitative, we use a frequent set based algorithm. There are three stages of the algorithm:

1. Find all frequent sets for categorical attributes of the database.
2. Calculate the mean and variance for the quantitative attributes on each frequent set.
3. Find all basic rules and sub-rules.

In order to calculate the rules we generate a frequency set lattice where each descendent of every node is a superset of the node, conversely a parent of a node is a subset of that node. From this lattice we can use Breadth-First Search to find all basic rules and sub-rules. This lattice structure implies that we find the sub-rules first, and then increase the generality of the rules until they are no longer of significance or interest.

5 Clustering Association Rules

Clustering is the process of taking a set of data and discovering unknown relations between different data points. Each subset of the data, known as a cluster, is defined by how close the data within a cluster is between each other, and how far apart they are from other clusters[16]. There are two types of clustering soft clustering, and hard clustering. Soft clustering means that a value can be in more than one cluster, whereas hard clustering implies that a data point is in at most one cluster. Clustering is closely related to classification mining which takes a set of data, and a set of classes and associates data points with a particular class based on the distance with the data points already within the class. Clustering is an unsupervised (does not need training data) version of classification mining. There are several types of clusters, the main ones that we will discuss are:

- Partition Clusters
- Hierarchical Clusters

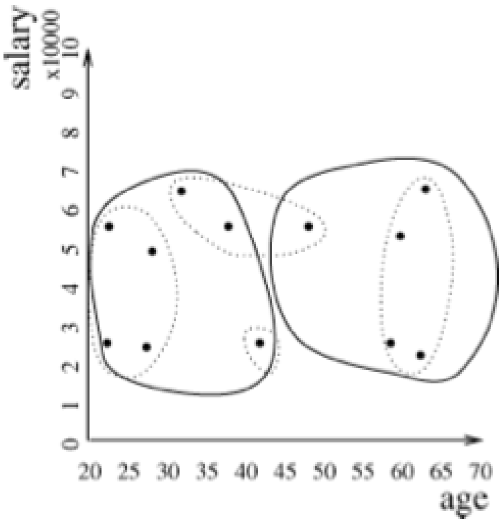

Figure 4: Example of Partition Clusters[17]

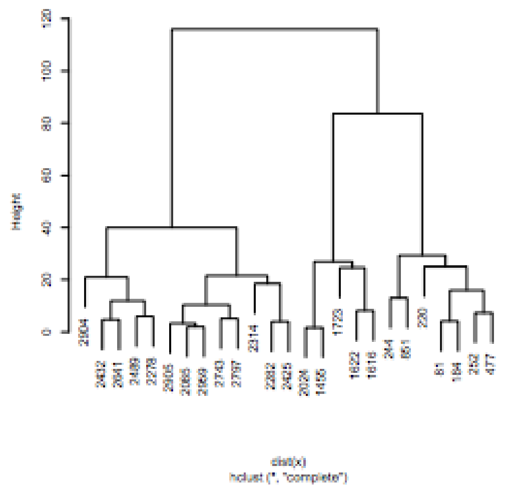

Figure 5: Example of Hierarchical Clusters[18]

Partition clusters are the basic form of cluster described above where there is no relationship between clusters except for their distance. In contrast the hierachical clusters have a tree structure where each level of the tree shows a differerent granularity of data, where child clusters are more specific than their parents (the distance metric has less possible variance within the cluster)[19]. In terms of clustering association rules there are three options for when to do the clustering process:

1. Before performing the association rule mining.
2. Perform clustering as an association rule mining algorithm.
3. After performing the association rule mining.

These options have different reasoning and objectives behind them. In terms of performing clustering before association rule mining, the main objective for this is to increase the performance of the overall application as clustering allows for association rule mining to be done in parallel on the different clusters[20]. In terms of performing clustering as an association rule mining algorithm, the idea between clusters and association rules is that each cluster contains related items which can be used to generate association rules from within that cluster[16][21]. In terms of performing clustering after association rule mining, the main objective for this is to group related rules together to obtain more information about them, and to prune uninteresting rules from the rule set[3][22][23].

5.1 Before Rule Mining

As mentioned previously the main objective with clustering prior to running association rule mining is to reduce the runtime of the application. In this we are going to be discussing two methods of association rule mining which use clustering to reduce the search space.

- Apriori Based[24]
- Fuzzy Clustering[20]

5.1.1 Apriori Based

The Apriori Algorithm[4] is one of the most commonly used association rule mining algorithms in general association rule mining. The idea behind using this algorithm on clusters is to be able to use the join and prune phases to ensure that the output is a good quality rule. In the paper[24] they use this algorithm for the candidate generation of Quantitative Frequent Itemsets (QFRs) to be able to define Class Association Rules (CAR). This paper does not add any new ideas but is discussed to show how association rule mining can be used as a step for a more complete application.

5.1.2 Fuzzy Clustering

The idea behind the algorithm is to reduce the memory footprint, and runtime of performing formal context associations (context aware association rules) by performing a clustering algorithm prior to the association rule mining. A fuzzy clustering algorithm, ensures that for every item in the database the suitability of each defined cluster is calculated and is used to show how closely related an item is to others in each cluster. The suitability is defined as a number in between 0 & 1, with numbers close to 1 meaning it is strongly associated with that cluster, and conversely numbers close to 0 are weakly associated with that cluster. The process defined by the paper[20] is to follow the following steps:

1. Cluster the data using fuzzy clustering methods
2. Perform association rule mining between clusters ($C_1 \implies C_2$ for clusters C_1, C_2)
3. Generate association rules from these cluster rules on the attributes (to get the classic form of association rules)

5.2 Rule Mining with Clusters

In general the usage of clustering algorithms to perform association rule mining is using the concept that the items within the clusters are closely related, and can be extended to generate a set of association rules when they meet the minimum support and confidence thresholds. There are several methods involved with the generation of association rules with clustering:

- Rule estimation
- Clustering model
- Conceptual Association Rule Mining.

In terms of rule estimation the clustering is very similar to pre-association rule mining clustering as it is a step before. The difference is that it has a way to generalise the association rules generated to the entire population as an estimation bound. These bounds are generated for both the support and the confidence measures to allow for any pruning to be done when necessary[16].

The clustering model of association rule mining is to use the clustering algorithm to generate the required frequent itemsets, which can then be used to build the association rules with the support and confidence measures being taken into consideration[21].

Conceptual association rule mining, is an extension of association rule mining which mines higher level concepts of the data than standard association rule mining does. The method to generate these conceptual rules is to use a process for generating these rules which involves:

1. Soft Clustering
2. Concept Hierarchies Generation
3. Rule Generation
4. Rule Unification

The clustering and hierarchical generation is then used to easily build the rules required based on the concepts generated. Soft clustering is required here as a transaction may contain records of different concepts and if hard clustering is used then these concepts might be missed. The concept hierachies generation builds multiple heirachies based on the clusters generated in the previous sections, a concept hierarchy is that every concept can be associated with other concepts in a parent-child relationship. Rules are then generated from the parent-child relationships in each generated hierarchy. These relationships need to meet the requirement that the parent is not all services. The rules are of the form: child $\implies$ parent. Rule unification takes each of the rulesets generated by the previous step and merges them into a single set.[25]

5.3 After Rule-Mining

The objective of performing clustering on the rules generated by association rule mining algorithms discussed earlier is to reduce the number of rules which allow for an increase in both the interest level of the rules, and the understandability of the rules by humans. We will be discussing three key methods of clustering association rules in order to solve these issues with large rulesets.

- BitOp Based Association Rule Clustering System
- Interest Rule Pruning
- Distance Based Clustering

The BitOp based association rule clustering system[3] is a system designed to cluster association rules with two attributes on the left hand side, like the output of the rectangle finding association rule algorithm described in section 4.1.3, but with a necessary condition of having the RHS categorical. It does this by generating a matrix for each value of the categorical attribute, with the dimensions being the attributes from the quantitative attributes, and plot all the association rules for that categorical value. We wish to find the minimum possible number of clusters in the resultant grid. The BitOp algorithm uses bitwise operations on these matrices in order to find the clusters by considering each row in turn and using bitwise AND to find possible locations of clusters.

The interest rule pruning algorithm[22] performs both the association rule mining and clustering on a database separately and then combines them to determine the interest of a rule, pruning those who do not meet a defined threshold. We determine the interest of a rule as the minimum distance between the clusters associated with the items of each rule. We define the distance between clusters to be the distance between the centroids of the clusters.

Distance based clustering uses a new metric of measuring the distance between two rules which is normalised so that certain characteristics (e.g. support) are not placed as a dependency of the rules distances.[23] The new metric uses probabilistic measures on the transactions within the database. As most clustering mechanisms use a distance metric to measure the closeness of clusters, we can substitute this new metric into a clustering algorithm with little change to the rest of the algorithm. The defined metric between rule i and rule j is:

$$dist(i,j) = 1 - \frac{|m(BS_i, BS_j)|}{|m(BS_i)| + |m(BS_j)| - |m(BS_i, BS_j)|} \tag{16}$$

Where BS_x is the set of items contained within rule x, m(X) is the set of all transactions containing the itemset X. We define a distance interesting if it is neither 1 or 0. A distance of 1 means there are no common items, a distance of 0 means the rules have identical set of items.

6 Conclusion

The purpose of this research survey is to introduce the topic of using clustering techniques to mine association rules on continous data attributes of a database. In this survey we introduced the topics of association rule mining, mining quantitative associations, and the clustering of association rules. We introduced several key algorithms and concepts required to perform our research while keeping in mind the work that has already been done and the strengths and weaknesses of each method.

7 References

[1] Food Marketing Institute. Supermarket facts. `http://www.fmi.org/research-resources/supermarket-facts`, 2013.

[2] Ramakrishnan Srikant and Rakesh Agrawal. Mining quantitative association rules in large relational tables. In *ACM SIGMOD Record*, volume 25, pages 1–12. ACM, 1996.

[3] Brian Lent, Arun Swami, and Jennifer Widom. Clustering association rules. In *Data Engineering, 1997. Proceedings. 13th International Conference on*, pages 220–231. IEEE, 1997.

[4] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules, 1994.

[5] Marie Plasse, Ndeye Niang, Gilbert Saporta, Alexandre Villeminot, and Laurent Leblond. Combined use of association rules mining and clustering methods to find relevant links between binary rare attributes in a large data set. *Computational Statistics & Data Analysis*, 52(1):596–613, 2007.

[6] Ashok Savasere, Edward Robert Omiecinski, and Shamkant B Navathe. An efficient algorithm for mining association rules in large databases. Technical report, Georgia Institute of Technology, 1995.

[7] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *ACM SIGMOD Record*, volume 29, pages 1–12. ACM, 2000.

[8] Charu C. Aggarwal and Philip S. Yu. A new framework for itemset generation. In *In: PODS 98, Symposium on Principles of Database Systems*, pages 18–24, 1998.

[9] Sergey Brin, Rajeev Motwani, and Craig Silverstein. Beyond market baskets: Generalizing association rules to correlations, 1997.

[10] Khalil M Ahmed, Nagwa M El-Makky, and Yousry Taha. A note on beyond market baskets: Generalizing association rules to correlations. *ACM SIGKDD Explorations Newsletter*, 1:46–48, 2000.

[11] Takeshi Fukuda, Yasuhido Morimoto, Shinichi Morishita, and Takeshi Tokuyama. Mining optimized association rules for numeric attributes. In *Proceedings of the fifteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 182–191. ACM, 1996.

[12] Yonatan Aumann and Yehuda Lindell. A statistical theory for quantitative association rules. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 261–270. ACM, 1999.

[13] Kunikazu Yoda, Takeshi Fukuda, Yasuhiko Morimoto, Shinichi Morishita, and Takeshi Tokuyama. Computing optimized rectilinear regions for association rules. In *KDD*, volume 97, pages 96–103, 1997.

[14] Gang Mei, J.C. Tipper, and Nengxiong Xu. An algorithm for finding convex hulls of planar point sets. In *Computer Science and Network Technology (ICCSNT), 2012 2nd International Conference on*, pages 888–891, Dec 2012.

[15] Takeshi Fukuda, Yasukiko Morimoto, Shinichi Morishita, and Takeshi Tokuyama. Data mining using two-dimensional optimized association rules: Scheme, algorithms, and visualization. *SIGMOD Rec.*, 25(2):13–23, June 1996.

[16] Carlos Ordonez. A model for association rules based on clustering. In *SAC*, pages 545–546, 2005.

[17] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. Automatic subspace clustering of high dimensional data, 2005.

[18] David M Blei. Hierarchical clustering. *Lecture Slides, February*, 2008.

[19] Pavel Berkhin. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer, 2006.

[20] Amel Grissa-Touzi, Aicha Thabet, and Minyar Sassi. Efficient reduction of the number of associations rules using fuzzy clustering on the data. In *ICSI (2)*, pages 191–199, 2011.

[21] Carlos Ordonez. Models for association rules based on clustering and correlation. *Intell. Data Anal.*, 13(2):337–358, 2009.

[22] Yanchang Zhao, Chengqi Zhang, and Shichao Zhang. Discovering interesting association rules by clustering. In *Australian Conference on Artificial Intelligence*, pages 1055–1061, 2004.

[23] Gunjan Gupta, Alexander Strehl, and Joydeep Ghosh. Distance based clustering of association rules. In *In Intelligent Engineering Systems Through Artificial Neural Networks (Proceedings of ANNIE 1999*, pages 759–764. ASME Press, 1999.

[24] Takashi Washio, Koutarou Nakanishi, and Hiroshi Motoda. A classification method based on subspace clustering and association rules. *New Generation Comput.*, 25(3):235–245, 2007.

[25] Thanh Tho Quan, Linh N. Ngo, and Siu Cheung Hui. An effective clustering-based approach for conceptual association rules mining. In *RIVF*, pages 1–7, 2009.