Summarization Techniques for Visualization of Large Multidimensional Datasets

Technical Report TR-2005-35

Sarat M. Kocherlakota, Christopher G. Healey

Knowledge Discovery Lab
Department of Computer Science, North Carolina State University
Raleigh, NC 27695-8207

Email: smkocher@unity.ncsu.edu

Abstract

One of the main issues confronting visualization, is how to effectively display large, high dimensional datasets within a limited display area, without overwhelming the user. In this report, we discuss a data summarization approach to tackle this problem. Summarization is the process by which data is reduced in a meaningful and intelligent fashion, to its important and relevant features. We survey several different techniques from within computer science, which can be used to extract various characteristics from raw data. Using summarization techniques intelligently within visualization systems, could potentially reduce the size and dimensionality of large, high dimensional data, highlight relevant and important features, and enhance comprehension.

1 Introduction

The area of visualization is primarily focused on representing raw data in the form of images, thereby providing users with the ability to visually analyze and explore large, complex datasets [16, 35]. Visualization techniques assist users in managing and displaying data in an intelligent and intuitive fashion. Visualization can be a great asset in the discovery of relationships and dependencies that may exist within the data.

Faced with the problem of handling larger and larger datasets, recent studies in visualization have emphasized increasing the information content of the displayed images. Datasets comprising millions (if not billions) of data elements, with each data element possibly having hundreds of attributes, have become commonplace across a variety of domains and applications. Issues related to dataset size and dimensionality were outlined, by the original NSF panel on visualization [28]. Current visualization techniques that deal with large multidimensional datasets have been effective in visualizing data containing up to at most a few million data values. Moreover, as we approach these limits, the images produced by current visualization techniques become cluttered, and visually difficult, even impossible to interpret [13, 17]. This diminishes the user's ability to visually analyze the data and recognize trends, clusters, outliers or gaps within the data [13]. The problem of effectively representing these large, multidimensional datasets without overwhelming the user's ability to comprehend the resulting images continues to pose a serious challenge to the area of visualization [16].

To face this challenge, recent studies have proposed a tighter coupling between visual and non-visual methods for data analysis and exploration [25, 33, 34]. In particular, incorporating data summarization techniques within visualization systems may enhance our ability to manage and visualize very large datasets of high dimensionality. Data summarization techniques can reduce the size and complexity of large multidimensional datasets to more manageable proportions. They can also highlight the relevant aspects of the data more clearly, leading to more coherent visualizations, and also facilitating more accurate and efficient visual analysis.

Summarization is performed using various techniques. These techniques are designed for the automated and unsupervised analysis and exploration of raw data, followed by the generation of effective summaries based on the analysis [2, 3, 22, 24]. Unfortunately, many of these techniques are more suited to univariate (or low dimensional) data, and may not be effective in dealing with datasets of high dimensionality [1]. Also, many data mining and machine learning techniques are complicated. Both their intermediate as well as their final results may not be easily understood by non-expert users. In this scenario, directly visualizing the results produced by these automated methods would provide neither any insight into the data, nor any understanding of the summarization process itself. For data summarization to be effective during visualization, a system should support the user in understanding as well as participating in the process.

In this report, we will focus on methods that could be enhanced and applied to summarize data within the context of visualization. The techniques we discuss include association rule mining [3, 4], outlier detection [1, 23, 34], clustering [22, 24], data classification [18, 27], data aggregation [8, 13], and the principal component analysis (PCA) technique, among others.

The remainder of the paper is organized as follows. In Section 2, we discuss techniques for association rule mining. Section 3 is devoted to outlier detection algorithms. In Section 4, we describe clustering and clustering techniques. In Section 5, we investigate classification techniques. Section 6 focuses on data aggregation. In Section 7, we discuss dimensionality reduction, and, in Section 8, we end with summarization techniques designed for spatial data. Finally, some conclusions are drawn in Section 9.

2 Association rule mining

Association rule mining techniques are primarily focused on the discovery of patterns and dependencies in datasets. An *association rule* is an expression of the form $X \Rightarrow Y$, where X and Y are sets of items. In a multidimensional dataset, an item could denote a particular attribute value, and X and Y could denote combinations of individual attribute values. For a multidimensional dataset D representing n data attributes $A = \{A_1, ..., A_n\}$, where each data element $e_i \in D$ is a combination of n individual data values, one for each attribute, the expression $X \Rightarrow Y$ signifies that, if e_i contains X then e_i probably also contains Y.

Association rule mining makes use of two measures: *support* and *confidence*. The support of a set of items X, denoted by $supp(X)$, is the fraction of data elements in D that contain X. The support of an association rule $X \Rightarrow Y$, denoted by $supp(X \Rightarrow Y)$, is the fraction of the data elements in D that contain the conjunction $X \cup Y$. The confidence of such a rule, denoted by *conf*$(X \Rightarrow Y)$, is the fraction of the data elements in D containing X, that also contain Y [3]. Mathematically, $supp(X \Rightarrow Y) = supp(X \cup Y)$, and *conf*$(X \Rightarrow Y)$ $= supp(X \cup Y)/supp(X)$ [19].

Based on these measures, the task of association rule mining is to find all association rules within a dataset D that satisfy certain user-defined thresholds for minimum support and minimum confidence, denoted by *min-sup* and *min-conf* respectively [4]. The approach to finding such rules generally involves two major steps. In the first step, all sets of attribute values that satisfy *min-sup* within D are found. Next, these sets are used to generate rules of the form $X \Rightarrow Y$ that satisfy *min-conf*[3].

Hipp, Guntzer and Nahaeizadeh present a general survey for association rule mining algorithms in [19]. They note that the most popular among association rule mining techniques are Apriori [4] and Apriori-based algorithms.

2.1 Apriori

Apriori, first introduced by Agrawal et al., is an algorithm that is used to find sets of items within a dataset that satisfy a user-defined minimum support threshold [4]. We begin with a discussion of the terminologies relevant to the algorithm [4]. A set of k items is referred to as a k-itemset. A k-itemset X that has $supp(X) \geq$ *min-sup* is referred to as a *large* k-itemset. The set of all large k-itemsets is denoted by L_k. A *candidate* k-itemset or a k-candidate itemset is used to denote a potentially large k-itemset, and the set of all such k-candidate itemsets is

denoted by C_k.

Apriori works in the following manner. It first counts the number of occurrences of each individual item in D to find all large 1-itemsets L_1. Each subsequent pass k involves two phases. In the first phase L_{k-1}, the set of $(k-1)$-itemsets found large in the previous pass, is used to generate a set of k-candidate itemsets C_k. In the second phase, D is scanned to compute the support of each of the k-candidate itemsets in C_k. Those that are found large (i.e. those that have support greater than or equal to *min-sup*) are used to form L_k, the set of large k-itemsets.

The k-candidate itemsets in C_k are generated from L_{k-1} through two steps. In the first step, referred to as the *join* step, k-candidate itemsets are generated by merging all $(k-1)$-itemsets $X, Y \in L_{k-1}$, which share their first $(k-2)$ items, i.e. $X.item_1 = Y.item_1$, $X.item_2 = Y.item_2$, ..., $X.item_{k-2} = Y.item_{k-2}$, $X.item_{k-1} \neq Y.item_{k-1}$ [4]. In the second step, referred to as the *prune* step, all itemsets $c \in C_k$ for which some $(k-1)$ subset of c is not in L_{k-1} are deleted.

To illustrate the algorithm let us consider an example [4]. Let L_1, the set of all large 1-itemsets in a dataset D generated during the first phase of the algorithm, contain the individual items { {1}, {2}, {3}, {5} }. To construct C_2 from L_1, all 2-itemset combinations of items from L_1 are generated, producing C_2 = { {1 2}, {1 3}, {1 5}, {2 3}, {2 5}, {3 5}, }. In the second phase of the algorithm, suppose the itemsets {1 5} and {3 5}, are removed from C_2 as they were found to have low support in D. The remaining itemsets define L_2 = { {1 2}, {1 3}, {2 3}, {2 5} }. In the first phase of the pass $k = 3$ L_2 is used to generated C_3. During the *join* step, {1 2} is merged with {1 3} to generate the itemset {1 2 3}. Likewise, {2 3} is merged with {2 5}to generate {2 3 5}. C_3 now contains { {1 2 3}, {2 3 5} }. During the *prune* step, there are three $k-1 = 2$-itemsets for candidate { 2 3 5}: {2 3}, {2 5} and {3 5}. Although {2 3} and {2 5} are contained in L_{k-1}, {3 5} is not. So the candidate itemset {2 3 5} is removed from C_3. The candidate itemset { {1 2 3} } is accepted since {1 2}, {2 3} and {1 3} are all in L_{k-1}. C_3 now contains only { {1 2 3} }. The remaining itemset in C_3, if found large in the second phase of the current pass, becomes L_3.

Once combinations of items that satisfy *min-sup* in D are identified, they are used to generate association rules. For each large itemset Z, all rules having the form $X \Rightarrow (Z - X)$ are generated, where X is any subset of Z, and $supp(Z)/supp(X) \geq$ *min-conf* . Initially, for every large itemset Z, all rules having only one item on the right hand side of the expression $X \Rightarrow (Z - X)$ are generated and tested. Rules that satisfy the minimum confidence requirement are further expanded to build additional rules.

Another algorithm introduced in [4] called AprioriTID extends the Apriori-based approach. In this algorithm the entire dataset D is not used for counting support at the end of each pass. Instead, AprioriTID uses only those items that were found large from the previous pass in an effort to improve the efficiency.

Research on association rule mining was initially targeted at transaction databases. Subsequent work has revealed that association rules can also be effective over a wider range of applications and datasets. Visualizing the patterns, rules and dependencies discovered using association rule mining could simultaneously highlight the important characteristics of the data,

as well as help reduce the amount of information to be displayed.

2.2 Drawbacks

Association rule mining algorithms do have certain drawbacks, such as:

1. *Algorithmic complexity*: Increase in size and dimensionality of the data reduces the efficiency of the algorithm. This means that the algorithm may be unsuited to very large multidimensional datasets [19].

2. *Determining usefulness of generated rules*: Association rule mining of large multidimensional datasets could result in the generation of thousands of association rules. Evaluating the relevance and utility of these rules can be a complex task by itself.

3. *Counting infrequent items*: Association rule mining algorithms scan the entire dataset to discover itemsets that satisfy minimum (user-specified) support and confidence thresholds. This process also includes counting support for itemsets that have low support within the dataset. This can also limit the efficiency of association rule mining techniques [4].

4. *Handling spatial datasets*: Techniques for association rule mining were designed to discover relationships in transaction data, in which attributes assume binary values i.e. either 0 or 1. In general however, data attributes assume values that cover a broad, continuous range of real numbers [36]. Counting the support of each individual attribute value can pose serious problems to association rule mining algorithms.

In the next section we deal with the problem of outlier detection in large multidimensional datasets.

3 Outlier Detection

Outliers can be defined as "data elements which are very different from the rest of the data based on some measure" [1]. Outliers can also be described as data elements that deviate from other observations by so much that they arouse the suspicion of being produced by a mechanism different than that which generated the other observations [15]. The detection and visualization of outliers within large multidimensional datasets could provide insight into abnormalities in the underlying data.

There are many different approaches to outlier detection. The distance-based approach by Knorr and Ng discovers outliers using a full-dimensional distance metric between any two data elements in a high dimensional information space [23]. A data element e_i is considered an outlier with respect to parameters k and λ, if at least a fraction k of the elements in a dataset D lie at a distance greater than λ from e_i [23, 1]. Unfortunately, this technique is sensitive to the parameter λ: a large value of λ could result in few outliers, while a small value of λ could

result in a large number of outliers. Also, the technique may not be effective for datasets that exhibit a complex structure, for instance, clusters of varying densities [6].

Breunig et al. suggest identifying outliers based on the densities of local neighborhoods. The basic idea is simple: if some element e_i is farther away from its surrounding elements, when compared to the neighbors e_j that lie around it, then e_i is marked as a local outlier. In this technique, outliers are found by comparing densities of different sets of data elements. Elements which are "outlying" relative to their local neighborhoods are called *local outliers* [6]. For each element e_i, its k nearest neighbors are identified, denoted by $N_k(e_i)$. If an element $e_j \in N_k(e_i)$ is within a user-defined distance λ from e_i, then e_j is said to lie in the local neighborhood of e_i, and the reachability distance of e_i with respect to e_j, denoted by $d(e_i, e_j)$, is set to λ. If e_j lies at a distance greater than λ, then $d(e_i, e_j)$ is set to the actual distance between e_i and e_j. The local reachability density of e_i, denoted by $\delta_{local}(e_i)$, is then calculated as the inverse of the average of $d(e_i, e_j)$ for all $e_j \in N_k(e_i)$, that is:

$$\delta_{local}(e_i) = \frac{1}{k} \sum_{\forall e_j \in N_k(e_i)} \frac{1}{d(e_i, e_j)} \tag{1}$$

Finally, the local outlier factor of e_i with respect to $N_k(e_i)$, denoted by $LOF(e_i)$, is computed as the average of the ratios of the local reachability density $\delta_{local}(e_j)$ of every $e_j \in N_k(e_i)$ to $\delta_{local}(e_i)$, that is:

$$LOF(e_i) = \frac{1}{k} \sum_{\forall e_j \in N_k(e_i)} \frac{\delta_{local}(e_j)}{\delta_{local}(e_i)} \tag{2}$$

The outlier factor denotes the degree to which e_i is considered to be a local outlier. If the outlier factor exceeds a user-chosen threshold, e_i is marked as a local outlier.

Both the distance-based approach [23] and the density-based approach [6] are more suited to low dimensional datasets. The sparse nature of high dimensional data makes it difficult to determine the locality of a data element [1]. Also, to determine local densities of elements in high dimensional data, a meaningful concept of distance is necessary. Calculating full-dimensional distances between data elements in high dimensional space can also be computationally expensive.

3.1 Evolutionary Algorithm Technique

One approach to finding outliers in high dimensional data is by examining data under different low dimensional projections of the data attributes. This approach is based on the observation that the density of data varies when examined under different attribute subsets, and that the attributes may contribute differently to the behavior of each data element [1]. In most cases, a data element is dependent only on a small subset of relevant data attributes. Meaningful distances in high-dimensional information space can be determined more effectively by using fewer, more relevant data attributes [18].

The approach adopted by Aggarwal and Yu is based on the principle that outliers are patterns that have a very low presence within the data. Such patterns could be found efficiently,

by examining low dimensional projections of the data which are abnormally sparse, i.e. that have extremely low density [1]. The density measure of an attribute subset is referred to as its *sparsity coefficient*. Examining every lower-dimensional attribute subset to determine its sparsity can be done using a brute force approach. Unfortunately, as the dimensionality of the data increases, the search space of the attribute subsets also grows exponentially, making brute force exploration inefficient.

To avoid this, Aggarwal and Yu suggest using an evolutionary algorithm technique. According to this theory, the limited amount of resources available in nature to all species creates competition among the species, and only the fittest of them survive. Fitter individuals mate with each other more often, producing still better individuals. In a similar fashion, the evolutionary algorithm technique combines promising sparse low dimensional attribute projections, and also allows sparser subsets to combine with each other more frequently, to produce highly sparse lower dimensional attribute projections.

Let m be the total number of elements in D. For the purpose of calculating the sparsity coefficient, each of the n attributes of D is divided into ϕ ranges, each of which contain an equal number of elements f_m, where $f = \frac{1}{\phi}$. Next, consider selecting a subset of d attributes, $d < n$, then choosing one subrange for each of these d attributes. If the data in D is uniformly distributed across each attribute, we may compute the expected fraction of elements f^d that will have attribute values that fall within the selected ranges of the d-attribute subset we have constructed.

The total number of elements from D expected to lie within the d-attribute subset is therefore $m \cdot f^d$, and the standard deviation of this number is $\sqrt{m \cdot f^d \cdot (1 - f^d)}$. In practice however, attribute values are rarely uniformly distributed and independent. This means that the actual number of elements k in D that lie within the attribute subset is not $m \cdot f^d$. This difference is used to calculate the sparsity coefficient [1]:

$$S(D) = \frac{k - m \cdot f^d}{\sqrt{m \cdot f^d \cdot (1 - f^d)}} \tag{3}$$

Each d-attribute subset is encoded as a string recording the combination of attributes present in the subset. Each position in the encoding represents a particular data attribute, and each value at the attribute's position represents the specific subrange selected for that attribute. These encodings are also known as *solutions*.

The evolutionary algorithm starts with a user-selected d and ϕ, which are used to select an initial set of p solutions S. During the *selection* step, the sparsity coefficient of each solution in S is calculated. The more negative the sparsity coefficient, the higher a solution's rank. Next, a *crossover* step selects pairs of solutions from S, with higher ranked solutions having a higher probability of being chosen. l-attribute subsets from each solution pair ($l < d$) are identified. Those l-attribute subsets that are sparser are recombined, to produce a new pair of d-attribute projections. The new solutions replace the old pair that generated them. Finally, a *mutation* step picks random positions in the solution strings in S and replaces the value with a number between 1 and ϕ, with a certain probability. These three steps of selection, crossover and mutation are repeated until the sparsity coefficients of the solutions stabilize.

Each of the final solutions in S represents a pattern of data elements with an abnormally sparse projection of the high dimensional dataset. Data elements that match these attribute patterns are designated as outliers.

Although this technique can find high dimensional outliers efficiently, it is dependent on the parameters d and ϕ. Also, in order to make use of evolutionary search methods, a good understanding of the problem is important. Designing specific algorithms for selection, recombination and mutation which work effectively for a given problem is often a non-trivial task [1].

4 Clustering

Given a dataset D of m data elements, a clustering algorithm divides D into groups or clusters where the data elements in the same cluster are similar to one another relative to data elements in different clusters [12]. Clustering can also be described as a method for classifying data in an exclusive manner, where each data element belongs to exactly one subset or cluster [20]. Clustering algorithms are useful in detecting underlying structure within the data.

In this section we describe two methods for clustering, namely, k-means and self organizing maps.

4.1 k-means Clustering

One of the most widely used clustering methods is the k-means clustering algorithm (KMC). k-means can be formally described as follows: "Given a set of m data elements in D comprising of n attributes, and an integer k, determine a set of k elements in D called centers, so as to minimize the distance from every element to its nearest center [21]." Each element is attached to its nearest center, thereby subdividing elements in D into k clusters. This approach of decomposing a dataset into disjoint clusters is also known as "partitional clustering" [22].

KMC first initializes a set of k cluster centers $G \in D, i = 1, ..., k$. Cluster centers can be assigned, for instance, in a random fashion. Once the centers are initialized, the clustering algorithm assigns each of the remaining, unselected data elements to the center that it is most similar to, i.e. the center that is closest in value. If $c(e_i)$ denotes the index of the center closet to a data element e_i, then the goal of k-means clustering is to minimize the mean-squared distance between each e_i and its nearest cluster center $g_{c(e_i)}$. This distance or *distortion error* [21] is given by[22]:

$$E_k = \sum_{i \in D} \|e_i - g_{c(e_i)}\|^2$$

When all the data elements have been grouped, the positions of each cluster center is recomputed based on the distances between the data elements within each cluster. The e_i which is closest to all the elements within the cluster is assigned as the new cluster center. Once all cluster centers have been recomputed in this fashion, the remaining e_i are reassigned to the new centers.

This process is repeated several times. At the end of each iteration, the recomputed cluster centers start to resemble the actual cluster centers more closely. The algorithm terminates when convergence is achieved, i.e. the distortion error does not improve significantly.

KMC has some drawbacks [22]. The initial assignment of the cluster centers can affect the efficiency of convergence to the true cluster centers. Choosing an appropriate value for k is also significant to the performance of the algorithm. Ideally, k should be as close as possible to the actual number of clusters present within the dataset. Recomputing centers during each iteration of the algorithm affects the efficiency of the technique, especially for large datasets of high dimensionality. Also, before clustering algorithms can be applied to large multidimensional data, analyzing whether the data exhibits a tendency to cluster is important.

4.2 Self Organizing Maps

Self organizing maps (SOM), first introduced by Kohonen, are used to organize unstructured data much like the k-means clustering approach [24]. SOM-based algorithms can generate clusters from raw data. They can also produce lower dimensional projections of high dimensional data.

The SOM algorithm works on the principle of competitive learning, an adaptive process by which neurons in a neural network become more and more sensitive to different input categories. A self organizing map generally consists of a two dimensional network of neurons arranged in a grid. Initially, each cell is assigned a reference vector (or reference value) in a random fashion. Every data element e_i from the input dataset is assigned to the neuron with reference vector that best represents e_i. Locating the closest reference vector in the SOM approach, is very similar to searching for the closest center in the k-means clustering approach. The closest reference vector is called the *winning vector*, since the the neurons on the grid compete to learn (adapt to) the input data element. The winning vector is then updated to represent e_i more closely. The reference vectors around the winning vector are also adjusted, with the amount a reference vector learns from the new data element dependent on how close its vector is to that of the new element. This process is repeated several times over the entire dataset until the reference vectors represent the data elements of D as accurately as possible.

Once the neurons arrange themselves so that further iterations do not produce any significant changes to their positions, the algorithm is terminated. At this stage, the reference vectors in the grid tend to topologically arrange themselves such that adjacent cells on the grid represent similar data elements in the information space [25]. This property of a SOM wherein similar data elements are grouped to nearby reference vectors on the grid, is also referred to as *topology preservation* [9]. A SOM can be used to identify similarities and differences within an information space, and can hence serve as a clustering tool [25]. It also has the capability to generalize, and "learns" the data in an unsupervised fashion [9]. As with clustering techniques in general, it does not require any prior knowledge about the data.

However, SOM techniques are often computationally intensive. Efficiency of this technique also depends on the initial arrangement of reference vectors. Since the reference vectors are assigned in a random fashion initially, SOM based techniques produce different results each

time the technique is applied. Generating multiple maps may be necessary to choose the most effective map. Finally, SOM based techniques are susceptible to missing data values within the input data, which is a problem that both clustering and projection based algorithms also suffer from [22].

5 Data Classification

Classification techniques (also referred to as classifiers) generally require a training dataset T for which the attribute values and data classes of each element are known in advance. Based on this information, classification techniques predict the class of unclassified data elements $e_i \in D$ [5]. The performance measure of the classifier, referred to as its *classification accuracy*, is the percentage of elements for which the classifier makes correct class predictions [5].

5.1 Rule Set Based Classification

Rule set based classification algorithms apply a set of classification rules generated from a training dataset to predict the class of an unclassified data element. Classification rules are of the form $X \Rightarrow c$, where X (the antecedent) contains a combination of attribute value pairs, and c (the consequent) denotes a corresponding class.

Traditionally, machine learning algorithms such as ID3 and C4.5 are used for class prediction [31]. In these methods, T is partitioned into smaller and smaller disjoint subsets using distinct values of the data attributes. During the partitioning process, the attribute with the highest probability of distributing T evenly is selected from among the set of all attributes. T is then subdivided into disjoint subsets based on ranges of values from the selected attribute. Each of these subsets is then recursively subdivided in a similar fashion. The partitioning is represented using a decision tree, where each node represents an attribute or an attribute subrange. The process ends when all attributes have been used and the leaves of the tree contain only individual classes. The class assigned to each leaf is the one with the highest probability among the elements belonging to the attribute subrange represented by the parent of the leaf. In cases, where each of the classes is equally probable, a class is chosen at random and assigned to the leaf. The decision tree constructed from T is then used to classify a new dataset. Each path from the root of the tree to a leaf represents a classification rule. New unclassified data elements $e_i \in D$ are then matched against these classification rules.

Rule-based classification has certain drawbacks, however. The way the data is subdivided may not reflect its actual distribution [30]. Classifiers select a single attribute at a time to partition the subset of data, and do not examine attribute combinations for possible relevance. This can lead to a sub-optimal partitioning, especially in high dimensional data space, where not all attributes are of equal importance. Also, constructing decision trees for large multidimensional datasets can be computationally intensive. Decision tree classifiers will fail to classify data elements with missing attribute values, especially if the missing values involve the attribute that forms the root of the decision tree.

Another approach to generating classification rules is association rule mining [3]. Association rules that derive predefined classes are known as *class association rules*. Association rule mining algorithms such as Apriori [4] and Apriori-based techniques can be used to generate a complete set of class association rules from training data. Since association rules are generated at each stage of the mining process, they can contain any number of attribute value pairs. Because of this, a complete class association rule set containing all class association rules is more adept at classifying data containing missing attribute values than traditional classifiers.

Unfortunately, association rule mining algorithms generate a large number of rules when applied to large multidimensional data. Trying to match each unclassified $e_i \in D$ against each rule from the complete rule set can be inefficient. In many cases, rules are redundant, and not all rules that are used to derive the same class are equally powerful [26, 27]. For two rules r_1 and r_2, r_1 is stronger than r_2 if r_1 is as accurate as r_2 and has fewer attribute-value pairs in its antecedent.

To address this problem, Li et al. suggest pruning weak rules at each stage of the association rule mining algorithm [27]. At each step k, only strong rules are considered for the rule generation at step $k+1$. This results in an optimal set of rules with the same predictive power as a complete class association rule set and with far fewer rules, making the classification process both efficient and robust at handling missing data.

5.2 Nearest Neighbor Classification

Nearest neighbor techniques for data classification involve finding the nearest neighbor $e_{NN} \in T$ of an unclassified data element $e_i \in D$, and assigning the class of e_{NN} to e_i [5]. In the k nearest neighbor classification technique, the k nearest neighbors of e_i are used to determine its class.

To determine the nearest neighbors of e_i, the distances between e_i and all $e_j \in T$ are usually determined using a L_1 (Manhattan distance) or L_2 (Euclidean distance) metric, where all dimensions or attributes contribute equally. As the dimensionality of the data increases however, the data tends to become sparse causing most elements e_j to appear to be relatively equidistant from e_i. This makes it difficult to find a meaningful e_{NN}.

One way to address this problem is to take advantage of the fact that all attributes in a high dimensional dataset, are not equally relevant in describing the behavior of e_i [1, 18]. Meaningful nearest neighbors can be determined by first finding a relevant lower dimensional projection P_{best}, then finding the nearest neighbor from among the data elements that are identified by P_{best}. The relevance of a particular projection is determined by the density of the cluster of data elements around e_i.

The *projected nearest neighbor technique* proposed by Hinneburg et al. adopts this approach. The relevance of each projection P_k depends on the distribution of distances of all e_j identified by P_k, from e_i. The distance $d_k(e_i, e_j)$ is measured using a L_p metric. A user-defined minimum threshold d_{min} is used to identify a meaningful neighborhood around e_i. The higher the number of points that lie in the neighborhood, the more relevant the projection.

To find P_{best}, a brute force approach that examines every possible projection is not efficient.

Hinneburg et al. use a combination of a genetic search method (similar to the evolutionary approach in [1]), and a greedy search algorithm. The genetic search algorithm is initially used to find the best set of relevant three to five dimensional projections. These projections are then refined using a greedy search algorithm. The greedy search algorithm, according to [18], is efficient at finding the best 1-dimensional projections. Here, it is used to find the best 1-dimensional projection from among the set of attributes not present in a projection P_k generated by the genetic search algorithm. P_k is then extended by this new, best 1-dimensional result. The combined search strategy provides an efficient solution to the problem of finding the relevant combination of attributes for the purpose of nearest neighbor search.

6 Aggregation

Data aggregation is the process by which a collection of several data elements or objects is reduced to a single element or object that is representative of the collection. This single object is called an *aggregate*. For instance, an aggregate f_{aggr} of a group of 1-dimensional data elements $e_1, ..., e_m$, can be computed as the average of their corresponding values $a_{1,1}, ..., a_{1,m}$. Besides averages, aggregates could also represent totals, variances, or highest or lowest values [14, 29].

Aggregation reduces large volumes of data into smaller and more manageable sizes. In visualization systems, they are used to summarize datasets and simplify the visualization [14]. In the Aggregate Manipulator (AM) system devised by Goldstein and Roth [14], users can create and control the range (scope) of values for each attribute, as well as select the aggregate type (average, sum, variance, highest or lowest value). Based on the user selections, the data elements are aggregated and the aggregates are visualized as points. The drawback of this system however, is that only a single aggregate type at a time can be chosen for visualization.

In the Aggregation Eye (AE) system devised by Mockus [29], a rectangular widget is used to select ranges of values for two attributes represented as a 2-dimensional grid. Data elements whose attribute values fall within the selected range are then aggregated by selecting an aggregate type similar to those used in AM. These aggregates are represented using rectangular glyphs. The size and color of the glyphs are used to display additional information about the elements' the attribute values. For instance, higher values can be represented with larger glyphs and lower values with smaller glyphs, or colors varying from red to blue can be used to denote values ranging from high to low. However, while the visual features of the glyphs (size and color) allow additional aggregate types to be represented, only ranges of two attributes at a time can be applied on the selection grid.

A similar aggregation based technique is also employed in the Snap-Together Visualization (STV) system developed by Fredrikson, et al. [13]. Here, sliders are used to control the range of values for each attribute, and aggregates representing the corresponding data elements are visualized using rectangular glyphs whose size and color are used to encode additional attribute information or aggregate types. This system also allows users to summarize data from several attributes and their ranges. The aggregates are displayed using map-based visualizations and

bar charts. Such multiple views allow users a better perspective of the data.

All of these techniques depend on the availability of domain knowledge to be effective. In AE for instance, information about what constitutes high and low for each attribute is required to decompose the attribute into meaningful sub-ranges. Choosing an appropriate aggregate type is also important. Nominal data, for instance, cannot be meaningfully represented by totals or averages. Also, aggregation is only performed on the ranges chosen interactively by the users.

In [8], Chuah introduces two aggregation based visualization techniques that support automated aggregation. In the first technique, called *SolarPlot*, a circular histogram with attribute values plotted along its circumference is used to visualize a 1-dimensional dataset. The length of the circumference determines the number of data elements that can be effectively displayed. When the number of elements exceeds the number of pixels along the circumference, multiple attribute values must share a single pixel. These values are then summarized using aggregation methods and the resulting aggregate is displayed at that pixel location. Users can interactively manipulate the circumference of the histogram to visualize the data at multiple levels of detail. As the circumference grows, the level of aggregation decreases dynamically to allow more individual detail to be displayed. Sections of the circle can be marked by users as areas of interest. These areas can then be magnified to show more local detail. While this technique simplifies large datasets, it does not support the ability to visualize multiple dimensions simultaneously.

The second technique discussed in [8] is the Aggregate Tree Map visualization. Aggregate Tree Maps are useful for visualizing hierarchical data. Here, each node in a data hierarchy is represented within a triangular region. The triangle is subdivided horizontally into regions corresponding to levels in the data hierarchy. The root node is represented near the apex of the triangle and the leaf nodes are represented along its base. Each horizontal region is further divided vertically in such a way that the space allotted to each node in the horizontal area is directly proportional to its number of children. Sub-regions are shaded with different colors to represent different ranges of the number of children. This technique is unsuited, however, to datasets for which data hierarchy is not known apriori.

7 Reducing Dimensionality

Dimensionality reduction maps a n-dimensional dataset D, to p new dimensions, where $p < n$. The reduction is designed to retain the important aspects of D, which are of interest to the user. One popular method for dimensionality reduction is Principal Component Analysis (PCA).

7.1 Principal Component Analysis

In PCA, attributes $A_1, ..., A_n$ are combined linearly to construct a series of orthogonal principal component axes $Z_1, ..., Z_n$ spanning the n-dimensions in D [35]. The direction of each axis Z_i is chosen in a way that maximizes the variance σ^2 captured from D with the axes ordered such that $\sigma^2(Z_1) \geq ... \geq \sigma^2(Z_n)$.

In practice, a limited number of axes p, $p \ll n$, can often capture most of the variation in D. A user-specified cutoff parameter τ is used to choose p such that $\sigma^2(Z_1) + ... + \sigma^2(Z_p) \geq \tau$. To calculate the principal component axes, the variance between all pairs of attributes A_i and A_j are determined. This is represented by a $n \times n$ covariance matrix C, where:

$$C_{i,j} = \frac{\sum_{k=1}^{m} D_{k,i} D_{k,j}}{n-1} - \frac{\sum_{k=1}^{m} D_{k,i} \sum_{k=1}^{m} D_{k,j}}{n} \quad (4)$$

Here, $D_{k,i}$ and $D_{k,j}$ represent the corresponding values of attributes A_i and A_j for the k^{th} element in D, respectively. Next, eigenvalues λ_i and eigenvectors Z_i are computed using C, where Z_i represent the principal component axes and λ_i represent the variance captured by Z_i. τ is used to select the first p principal component axes that capture the required variance.

Visualizing the reduced data in principal component space, can be challenging, however, since users have no way to intuitively convert data from principal component space back to the original attribute space. Because of this, data must be transformed from principal component space back to n-dimensional space prior to visualization.

In [35], Walter and Healey use PCA to reduce the dimensionality of a large multidimensional dataset. The reduced data is then simplified using geometric simplification techniques. The simplified data produces images containing far fewer elements compared to to the original dataset, while simultaneously retaining areas of high variance and important details.

8 Spatial Data Summarization

Datasets that include an explicit coordinate system within their attributes are often referred to as spatial datasets. Meteorological datasets with latitude, longitude and elevation attributes, and medical scans with slice locations and inter-slice distance attributes, are instances of spatial data. Distances and neighborhoods can be determined based on the spatial locations of the elements. Traditional data mining algorithms do not consider these properties when dealing with spatial data. These algorithms treat all elements as independent of each other [34], and incorporate spatial attributes, and even temporal attributes, simply as additional dimensions within the mining process [32].

In the summarization approaches by Ester et al. [11, 10] and Shekhar et al. [34], data elements are not treated independent of each other. Instead, elements that lie within a user-defined distance threshold are termed spatial neighbors. These neighborhoods are then used to detect clusters, trends [11] and outliers [34]. Ester et al. also consider relative differences in non-spatial attribute values when defining spatial neighborhoods [11, 10]. In their approach, two elements e_i and e_j are considered spatial neighbors if e_i and e_j lie within a distance threshold of each other, and if e_i's non-spatial attribute values do not differ from those of e_j, by more than a user-defined similarity threshold.

For spatial cluster detection, a new cluster C is created for an $e_i \in D$ that does not belong to any cluster. Then, for all elements $e_j \in D$ not part of any cluster, e_j is added to C if e_j is a spatial neighbor of e_i. The process is repeated until all elements in D belong to a

cluster. The clusters are then visualized to display both the spatial and the non-spatial attribute characteristics of the dataset. Spatial trends (increasing or decreasing) are also discovered using the concept of spatial neighbors. A spatial trend tracks the change in one or more non-spatial attribute values when moving away from a certain specific location [11]. To detect decreasing trends starting from a location e_i, e_i's neighbors are examined to find an e_j whose attribute values are lower than that of e_i. This e_j is added to the current trend. Then, e_j is examined in a similar fashion. Increasing trends, are also detected in the same fashion.

Ester et al. consider only one spatial neighbor at a time in characterizing spatial data. In the outlier detection technique by Shekhar et al. [34], a neighborhood of k nearest neighbors is evaluated. Here, outliers are elements whose non-spatial attribute values deviate significantly from those of its k nearest spatial neighbors. The "outlierness" of any element e_i is determined by the equation:

$$\left| \frac{S(e_i) - \mu_S}{\sigma_S} \right| > \theta \tag{5}$$

Here, $S(e_i)$ is a value that denotes the difference between e_i's attribute values, and the average of the attribute values of its k nearest neighbors. μ_S and σ_S are individual values that denote the average and the standard deviation of $S(e_i)$, for all $e_i \in D$, respectively. θ denotes a user-specified confidence value: if the outlierness of e_i is greater than θ, it is marked as a spatial outlier. This technique can be extended to include temporal attributes, to discover temporal and spatial-temporal outliers.

Both, the spatial data characterization technique of Ester et al. [10], and the outlier detection technique of Shekhar et al. [34] described above, are based on the assumption that spatial data elements usually exhibit a strong correlation with elements that lie in their neighborhood. This property is also referred to as *spatial autocorrelation* [7]. However, spatial autocorrelation does not account for datasets that exhibit a high spatial frequency. Also, these techniques are suited only to spatial datasets. Finally, computing distances between spatial data elements, especially when dealing with very large datasets, can adversely affect the efficiency of the algorithms.

9 Conclusions

In this report we examined how summarizing large multidimensional datasets can be advantageous to visualization systems. Specifically, we studied how techniques from pattern detection, classification and clustering, dimensionality reduction, and aggregation can be used to compress large multidimensional datasets into smaller datasets which still retain the important characteristics of the original data. Summarizing large datasets before visualizing the data, could lead to more effective visualizations, which in turn would support more efficient and accurate visual analysis.

There are several important issues that need to be addressed to use summarization effectively during visualization. First, we must investigate how specific summarization techniques

can be applied effectively to large datasets, especially temporal and spatial-temporal datasets. Second, we need to understand which information is most relevant to a broad spectrum of users and domains. Third, we must determine how various visualization techniques can be applied to accurately display summarized data.

Addressing these issues could provide a better understanding of the benefits of summarization to visualization systems. It may also offer insights into the task of effectively managing large multidimensional datasets.

References

[1] AGGARWAL, C. C., AND YU, P. S. Outlier detection for high dimensional data. In *SIGMOD Conference* (2001).

[2] AGRAWAL, R., GEHRKE, J., GUNOPULOS, D., AND RAGHAVAN, P. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. ACM Conf. on Management of Data* (1998), pp. 94–105.

[3] AGRAWAL, R., IMIELINSKI, T., AND SWAMI, A. N. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data* (Washington, D.C., 26–28 1993), P. Buneman and S. Jajodia, Eds., pp. 207–216.

[4] AGRAWAL, R., AND SRIKANT, R. Fast algorithms for mining association rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB* (1994), J. B. Bocca, M. Jarke, and C. Zaniolo, Eds., Morgan Kaufmann, pp. 487–499.

[5] ANKERST, M., KASTENMÜLLER, G., KRIEGEL, H.-P., AND SEIDL, T. 3D shape histograms for similarity search and classification in spatial databases. In *Advances in Spatial Databases, 6th International Symposium, SSD'99* (Hong Kong, China, 1999), R. Güting, D. Papadias, and F. Lochovsky, Eds., vol. 1651, Springer, pp. 207–228.

[6] BREUNIG, M. M., KRIEGEL, H.-P., NG, R. T., AND SANDER, J. LOF: identifying density-based local outliers. In *ACM SIGMOD Conference Proceedings 2000* (2000), pp. 93–104.

[7] CHAWLA, S., SHEKHAR, S., WU, W., AND OZESMI, U. Modeling spatial dependencies for mining geospatial data. In *Proc. of the 1st SIAM International Conference on Data Mining* (Chicago, IL, 2001).

[8] CHUAH, M. Dynamic aggregation with circular visual designs. In *Proceedings IEEE Symposium on Information Visualization 1998* (1998), pp. 35–43.

[9] DEBOECK, G., AND KOHONEN, T., Eds. *Visual Explorations in Finance with Self Organizing Maps*. Springer - Verlag, Menlo Park, California, 1998.

[10] ESTER, M., KRIEGEL, H.-P., AND SANDER, J. Spatial data mining: A database approach. In *Fifth Symposium on Large Spatial Databases (SSD'97)* (Berlin, Germany, 1997), M. Scholl and A. Voisard, Eds., vol. 1262, Springer, pp. 48–66.

[11] ESTER, M., KRIEGEL, H.-P., AND SANDER, J. Knowledge discovery in spatial databases. In *23rd German Conference on Artificial Intelligence, KI'99* (Bonn, Germany, 1999), vol. 1701, Springer, pp. 61–74.

[12] FLEXER, A. On the use of self-organizing maps for clustering and visualization. In *Intelligent Data Analysis 5* (2001), IOS Press, pp. 373–384.

[13] FREDRIKSON, A., NORTH, C., PLAISANT, C., AND SHNEIDERMAN, B. Temporal, geographical, and categorial aggregations viewed through coordinated displays: A case study with highway incident data. In *Workshop on New Paradigms in Information Visualization and Manipulation* (1999), pp. 26–34.

[14] GOLDSTEIN, J., AND ROTH, S. F. Using aggregation and dynamic queries for exploring large data sets. In *Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems* (New York, April 1994), ACM, pp. 23–29.

[15] HAWKINS, D. *Identification of Outliers*. Chapman and Hall, 1980.

[16] HEALEY, C. G., AMANT, R. S., AND CHANG, J. Assisted visualization of e-commerce auction agents. *Proceedings Graphics Interface 2001* (2001).

[17] HEALEY, C. G., AND ENNS, J. T. Building perceptual textures to visualize multidimensional datasets. *Proceeding Visualization '98* (1998), 111–118.

[18] HINNEBURG, A., AGGARWAL, C. C., AND KEIM, D. A. What is the nearest neighbor in high dimensional spaces? In *The VLDB Journal* (2000), pp. 506–515.

[19] HIPP, J., GÜNTZER, U., AND NAKHAEIZADEH, G. Algorithms for association rule mining – a general survey and comparison. *SIGKDD Explorations 2*, 1 (July 2000), 58–64.

[20] JAIN, A., AND DUBES, R. *Algorithms for Clustering Data*. Prentice - Hall, Englewood Cliffs, NJ, 1988.

[21] KANUNGO, T., MOUNT, D., NETANYAHU, N., PAITKO, C., SILVERMAN, R., AND WU, A. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 24*, 7 (July 2002), 881–892.

[22] KASKI, S. Data exploration using self-organizing maps. *Acta Polytechnica Scandinavica, Mathematics, Computing and Management in Engineering Series No. 82* (March 1997).

[23] KNORR, E., AND NG, R. Distance-based outliers in large data sets. In *VLDB Conference Proceedings* (1998).

[24] KOHONEN, T. *Self Organizing Maps*. Springer Verlag, Berlin, 1995.

[25] KREUSELER, M., AND SCHUMANN, H. A flexible approach for visual data mining. *IEEE Transactions on Visualization and Computer Graphics 8*, 1 (2002), 39–51.

[26] LI, J. *Optimal and Robust Rule Set Generation*. PhD thesis, School of Computing and Information Technology, Griffith University, Australia, 2002.

[27] LI, J., TOPOR, R., AND SHEN, H. Construct robust rule sets for classification. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (2002), pp. 564 – 569.

[28] MCCORMICK, B., DEFANTI, T., AND BROWN, M. Visualization in scientific computing - a synopsis. *IEEE Computer Graphics and Applications 7 7* (1987), 61–70.

[29] MOCKUS, A. Navigating aggregation spaces. In *Proc. IEEE Conference on Information Visualization '98* (Los Alamitos, CA, 1998), IEEE.

[30] PERNER, P., AND APTE, C. Empirical evaluation of feature subset selection based on real-world data set. In *Principles of Data Mining and Knowledge Discovery* (2000), D. A. Zighed, J. Komorowski, and J. Zytkow, Eds., Springer Verlag, pp. 575–580.

[31] QUINLAN, J. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[32] RODDICK, J., AND LEES, B. Paradigms for spatial and spatio-temporal data mining. In *Geographic Data Mining and Knowledge Discovery* (2001), H. Miller and J. Han, Eds., Taylor and Francis, pp. 33–50.

[33] RODRIGUES, J. F., TRAINA, A. J. M., AND JR, C. T. Enhancing data visualization techniques. In *Proc. Third IEEE Intl. Workshop on Visual Data Mining - VDM@ICDM'03, Melbourne - CA, USA* (2003), pp. 97–112.

[34] SHEKHAR, S., LU, C.-T., AND ZHANG, P. Detecting graph-based spatial outliers: algorithms and applications (a summary of results). In *the Seventh SIGKDD Int'l Conference on Knowledge Discovery and Data Mining (KDD 2001)* (2001), pp. 371–376.

[35] WALTER, J. D., AND HEALEY, C. G. Attribute preserving dataset simplification. In *Proceedings IEEE Symposium on Information Visualization 2001* (2001), pp. 113–120.

[36] WANG, W., YANG, J., AND MUNTZ, R. R. TAR: Temporal association rules on evolving numerical attributes. In *Seventeenth International Conference on Data Engineering, ICDE 2001* (Heidelberg, Germany, 2001), IEEE Computer Society, pp. 283–292.