Subspace Clustering for High Dimensional Data: A Review *

Lance Parsons
Department of Computer Science Engineering
Arizona State University
Tempe, AZ 85281
lparsons@asu.edu

Ehtesham Haque
Department of Computer Science Engineering
Arizona State University
Tempe, AZ 85281
Ehtesham.Haque@asu.edu

Huan Liu
Department of Computer Science Engineering
Arizona State University
Tempe, AZ 85281
hliu@asu.edu

ABSTRACT

Subspace clustering is an extension of traditional clustering that seeks to find clusters in different subspaces within a dataset. Often in high dimensional data, many dimensions are irrelevant and can mask existing clusters in noisy data. Feature selection removes irrelevant and redundant dimensions by analyzing the entire dataset. Subspace clustering algorithms localize the search for relevant dimensions allowing them to find clusters that exist in multiple, possibly overlapping subspaces. There are two major branches of subspace clustering based on their search strategy. Top-down algorithms find an initial clustering in the full set of dimensions and evaluate the subspaces of each cluster, iteratively improving the results. Bottom-up approaches find dense regions in low dimensional spaces and combine them to form clusters. This paper presents a survey of the various subspace clustering algorithms along with a hierarchy organizing the algorithms by their defining characteristics. We then compare the two main approaches to subspace clustering using empirical scalability and accuracy tests and discuss some potential applications where subspace clustering could be particularly useful.

Keywords

clustering survey, subspace clustering, projected clustering, high dimensional data

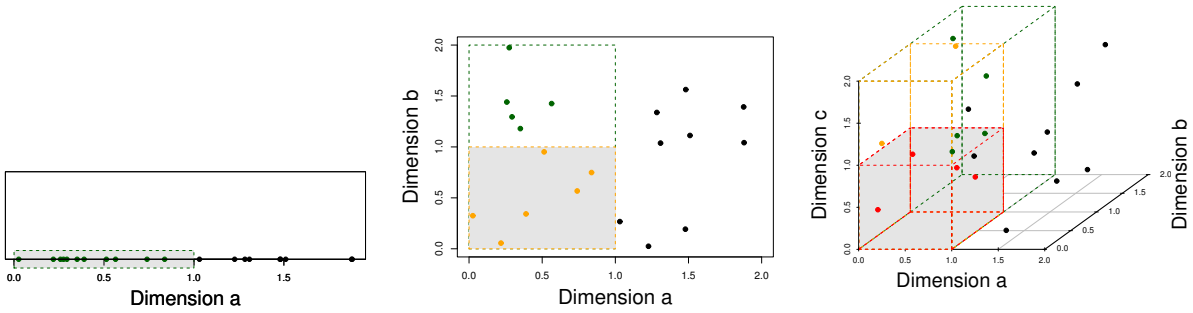
1. INTRODUCTION & BACKGROUND

Cluster analysis seeks to discover groups, or *clusters*, of similar objects. The objects are usually represented as a vector of measurements, or a point in multidimensional space. The similarity between objects is often determined using distance measures over the various dimensions in the dataset [44; 45]. Technology advances have made data collection easier and faster, resulting in larger, more complex datasets with many objects and dimensions. As the datasets become larger and more varied, adaptations to existing algorithms are required to maintain cluster quality and speed. Traditional clustering algorithms consider all of the dimensions of an input dataset in an attempt to learn as much as possible about each object described. In high dimensional data, however, many of the dimensions are often irrelevant. These irrelevant dimensions can confuse clustering algorithms by hiding clusters in noisy data. In very high dimensions it is common for all of the objects in a dataset to be nearly equidistant from each other, completely masking the clusters. Feature selection methods have been employed somewhat successfully to improve cluster quality. These algorithms find a subset of dimensions on which to perform clustering by removing irrelevant and redundant dimensions. Unlike feature selection methods which examine the dataset as a whole, subspace clustering algorithms localize their search and are able to uncover clusters that exist in multiple, possibly overlapping subspaces.

Another reason that many clustering algorithms struggle with high dimensional data is the *curse of dimensionality*. As the number of dimensions in a dataset increases, distance measures become increasingly meaningless. Additional dimensions spread out the points until, in very high dimensions, they are almost equidistant from each other. Figure 1 illustrates how additional dimensions spread out the points in a sample dataset. The dataset consists of 20 points randomly placed between 0 and 2 in each of three dimensions. Figure 1(a) shows the data projected onto one axis. The points are close together with about half of them in a one unit sized bin. Figure 1(b) shows the same data stretched into the second dimension. By adding another dimension we spread the points out along another axis, pulling them further apart. Now only about a quarter of the points fall into a unit sized bin. In Figure 1(c) we add a third dimension which spreads the data further apart. A one unit sized bin now holds only about one eighth of the points. If we continue to add dimensions, the points will continue to spread out until they are all almost equally far apart and distance is no longer very meaningful. The problem is exacerbated when objects are related in different ways in different subsets of dimensions. It is this type of relationship that subspace clustering algorithms seek to uncover. In order to find such clusters, the irrelevant features must be removed to allow the clustering algorithm to focus on only the relevant dimensions. Clusters found in lower dimensional space also tend to be more easily interpretable, allowing the user to better direct further study.

Methods are needed that can uncover clusters formed in various subspaces of massive, high dimensional datasets and

*Supported in part by grants from Prop 301 (No. ECR A601) and CEINT 2004.



(a) 11 Objects in One Unit Bin (b) 6 Objects in One Unit Bin (c) 4 Objects in One Unit Bin

Figure 1: The *curse of dimensionality*. Data in only one dimension is relatively tightly packed. Adding a dimension stretches the points across that dimension, pushing them further apart. Additional dimensions spreads the data even further making high dimensional data extremely sparse.

represent them in easily interpretable and meaningful ways [48; 66]. This scenario is becoming more common as we strive to examine data from various perspectives. One example of this occurs when clustering query results. A query for the term "Bush" could return documents on the president of the United States as well as information on landscaping. If the documents are clustered using the words as attributes, the two groups of documents would likely be related on different sets of attributes. Another example is found in bioinformatics with DNA microarray data. One population of cells in a microarray experiment may be similar to another because they both produce chlorophyll, and thus be clustered together based on the expression levels of a certain set of genes related to chlorophyll. However, another population might be similar because the cells are regulated by the circadian clock mechanisms of the organism. In this case, they would be clustered on a different set of genes. These two relationships represent clusters in two distinct subsets of genes. These datasets present new challenges and goals for unsupervised learning. Subspace clustering algorithms are one answer to those challenges. They excel in situations like those described above, where objects are related in multiple, different ways.

There are a number of excellent surveys of clustering techniques available. The classic book by Jain and Dubes [43] offers an aging, but comprehensive look at clustering. Zait and Messatfa offer a comparative study of clustering algorithms in [77]. Jain *et al.* published another survey in 1999 [44]. More recent data mining texts include a chapter on clustering [34; 39; 45; 73]. Kolatch presents an updated hierarchy of clustering algorithms in [50]. One of the more recent and comprehensive surveys was published by Berhkin and includes a small section on subspace clustering [11]. Gan presented a small survey of subspace clustering methods at the Southern Ontario Statistical Graduate Students Seminar Days [33]. However, there was little work that dealt with the subject of subspace clustering in a comprehensive and comparative manner.

In the next section we discuss feature transformation and feature selection techniques which also deal with high dimensional data. Each of these techniques work well with particular types of datasets. However, in Section 3 we show an example of a dataset for which neither technique is well suited, and introduce subspace clustering as a potential solution. Section 4 contains a summary of many subspace clustering algorithms and organizes them into a hierarchy according to their primary characteristics. In Section 5 we analyze the performance of a representative algorithm from each of the two major branches of subspace clustering. Section 6 discusses some potential real world applications for subspace clustering in Web text mining and bioinformatics. We summarize our conclusions in Section 7. Appendix A contains definitions for some common terms used throughout the paper.

2. DIMENSIONALITY REDUCTION

Techniques for clustering high dimensional data have included both feature transformation and feature selection techniques. Feature transformation techniques attempt to summarize a dataset in fewer dimensions by creating combinations of the original attributes. These techniques are very successful in uncovering latent structure in datasets. However, since they preserve the relative distances between objects, they are less effective when there are large numbers of irrelevant attributes that hide the clusters in sea of noise. Also, the new features are combinations of the originals and may be very difficult to interpret the new features in the context of the domain. Feature selection methods select only the most relevant of the dimensions from a dataset to reveal groups of objects that are similar on only a subset of their attributes. While quite successful on many datasets, feature selection algorithms have difficulty when clusters are found in different subspaces. It is this type of data that motivated the evolution to subspace clustering algorithms. These algorithms take the concepts of feature selection one step further by selecting relevant subspaces for each cluster separately.

2.1 Feature Transformation

Feature transformations are commonly used on high dimensional datasets. These methods include techniques such as principle component analysis and singular value decomposition. The transformations generally preserve the original, relative distances between objects. In this way, they summarize the dataset by creating linear combinations of the attributes, and hopefully, uncover latent structure. Feature

transformation is often a preprocessing step, allowing the clustering algorithm to use just a few of the newly created features. A few clustering methods have incorporated the use of such transformations to identify important features and iteratively improve their clustering [24; 41]. While often very useful, these techniques do not actually remove any of the original attributes from consideration. Thus, information from irrelevant dimensions is preserved, making these techniques ineffective at revealing clusters when there are large numbers of irrelevant attributes that mask the clusters. Another disadvantage of using combinations of attributes is that they are difficult to interpret, often making the clustering results less useful. Because of this, feature transformations are best suited to datasets where most of the dimensions are relevant to the clustering task, but many are highly correlated or redundant.

2.2 Feature Selection

Feature selection attempts to discover the attributes of a dataset that are most relevant to the data mining task at hand. It is a commonly used and powerful technique for reducing the dimensionality of a problem to more manageable levels. Feature selection involves searching through various feature subsets and evaluating each of these subsets using some criterion [13; 54; 63; 76]. The most popular search strategies are greedy sequential searches through the feature space, either forward or backward. The evaluation criteria follow one of two basic models, the *wrapper* model and the *filter* model [49]. The wrapper model techniques evaluate the dataset using the data mining algorithm that will ultimately be employed. Thus, they "wrap" the selection process around the data mining algorithm. Algorithms based on the filter model examine intrinsic properties of the data to evaluate the feature subset prior to data mining.

Much of the work in feature selection has been directed at supervised learning. The main difference between feature selection in supervised and unsupervised learning is the evaluation criterion. Supervised wrapper models use classification accuracy as a measure of goodness. The filter based approaches almost always rely on the class labels, most commonly assessing correlations between features and the class labels [63]. In the unsupervised clustering problem, there are no universally accepted measures of accuracy and no class labels. However, there are a number of methods that adapt feature selection to clustering.

Entropy measurements are the basis of the filter model approach presented in [19; 22]. The authors argue that entropy tends to be low for data that contains tight clusters, and thus is a good measure to determine feature subset relevance. The wrapper method proposed in [47] forms a new feature subset and evaluates the resulting set by applying a standard k-means algorithm. The EM clustering algorithm is used in the wrapper framework in [25]. Hybrid methods have also been developed that use a filter approach as a heuristic and refine the results with a clustering algorithm. One such method by Devaney and Ram uses category utility to evaluate the feature subsets. They use the COBWEB clustering algorithm to guide the search, but evaluate based on intrinsic data properties [23]. Another hybrid approach uses a greedy algorithm to rank features based on entropy values and then uses k-means to select the best subsets of features [20].

In addition to using different evaluation criteria, unsupervised feature selection methods have employed various search methods in attempts to scale to large, high dimensional datasets. With such datasets, random searching becomes a viable heuristic method and has been used with many of the aforementioned criteria [1; 12; 29]. Sampling is another method used to improve the scalability of algorithms. Mitra *et al.* partition the original feature set into clusters based on a similarity function and select representatives from each group to form the sample [57].

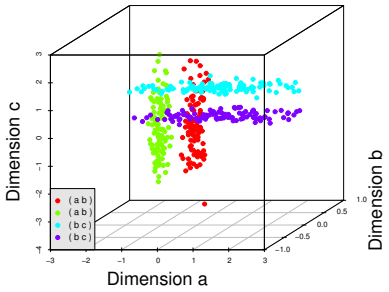


Figure 2: Sample dataset with four clusters, each in two dimensions with the third dimension being noise. Points from two clusters can be very close together, confusing many traditional clustering algorithms.

3. SUBSPACE CLUSTERING THEORY

Subspace clustering is an extension of feature selection that attempts to find clusters in different subspaces of the same dataset. Just as with feature selection, subspace clustering requires a search method and an evaluation criteria. In addition, subspace clustering must somehow limit the scope of the evaluation criteria so as to consider different subspaces for each different cluster.

3.1 Motivation

We can use a simple dataset to illustrate the need for subspace clustering. We created a sample dataset with four hundred instances in three dimensions. The dataset is divided into four clusters of 100 instances, each existing in only two of the three dimensions. The first two clusters exist in dimensions a and b. The data forms a normal distribution with means 0.5 and -0.5 in dimension a and 0.5 in dimension b, and standard deviations of 0.2. In dimension c, these clusters have $\mu = 0$ and $\sigma = 1$. The second two clusters are in dimensions b and c and were generated in the same manner. The data can be seen in Figure 2. When k-means is used to cluster this sample data, it does a poor job of finding the clusters. This is because each cluster is spread out over some irrelevant dimension. In higher dimensional datasets this problem becomes even worse and the clusters become impossible to find, suggesting that we consider fewer dimensions.

As we have discussed, using feature transformation techniques such as principle component analysis does not help in this instance, since relative distances are preserved and the effects of the irrelevant dimension remain. Instead we might try using a feature selection algorithm to remove one or two dimensions. Figure 3 shows the data projected in a single dimension (organized by index on the x-axis for ease

of interpretation). We can see that none of these projections of the data are sufficient to fully separate the four clusters. Alternatively, if we only remove one dimension, we produce the graphs in Figure 4.

However, it is worth noting that the first two clusters (red and green) are easily separated from each other and the rest of the data when viewed in dimensions a and b (Figure 4(a)). This is because those clusters were created in dimensions a and b and removing dimension c removes the noise from those two clusters. The other two clusters (blue and purple) completely overlap in this view since they were created in dimensions b and c and removing c made them indistinguishable from one another. It follows then, that those two clusters are most visible in dimensions b and c (Figure 4(b)). Thus, the key to finding each of the clusters in this dataset is to look in the appropriate subspaces. Next, we explore the various goals and challenges of subspace clustering.

3.2 Goals and Challenges

The main objective of clustering is to find high quality clusters within a reasonable time. However, different approaches to clustering often define clusters in different ways, making it impossible to define a universal measure of quality. Instead, clustering algorithms must make reasonable assumptions, often somewhat based on input parameters. Still, the results must be carefully analyzed to ensure they are meaningful. The algorithms must be efficient in order to scale with the increasing size of datasets. This often requires heuristic approaches, which again make some assumptions about the data and/or the clusters to be found.

While all clustering methods organize instances of a dataset into groups, some require clusters to be discrete partitions and others allow overlapping clusters. The shape of clusters can also vary from semi-regular shapes defined by a center and a distribution to much more irregular shapes defined by units in a hyper-grid. Subspace clustering algorithms must also define the subset of features that are relevant for each cluster. These subspaces are almost always allowed to be overlapping. Many clustering algorithms create discrete partitions of the dataset, putting each instance into one group. Some, like k-means, put every instance into one of the clusters. Others allow for outliers, which are defined as points that do not belong to any of the clusters. Still other clustering algorithms create overlapping clusters, allowing an instance to belong to more than one group or to be an outlier. Usually these approaches also allow an instance to be an outlier, belonging to no particular cluster. Clustering algorithms also differ in the shape of clusters they find. This is often determined by the way clusters are represented. Algorithms that represent clusters using a center and some distribution function(s) are biased toward hyper-elliptical clusters. However, other algorithms define clusters as combinations of units of a hyper-grid, creating potentially very irregularly shaped clusters with flat faces. No one definition of clusters is better than the others, but some are more appropriate for certain problems. Domain specific knowledge is often very helpful in determining which type of cluster formation will be the most useful.

The very nature of unsupervised learning means that the user knows little about the patterns they seek to uncover. Because of the problem of determining cluster quality, it is often difficult to determine the number of clusters in a particular dataset. The problem for subspace algorithms is compounded in that they must also determine the dimensionality of the subspaces. Many algorithms rely on an input parameter to assist them, however this is often not feasible to determine ahead of time and is often a question that one would like the algorithm to answer. To be most useful, algorithms should strive for stability across a range of parameter values.

Since there is no universal definition of clustering, there is no universal measure with which to compare clustering results. However, evaluating cluster quality is essential since any clustering algorithm will produce some clustering for every dataset, even if the results are meaningless. Cluster quality measures are just heuristics and do not guarantee meaningful clusters, but the clusters found should represent some meaningful pattern in the data in the context of the particular domain, often requiring the analysis of a human expert.

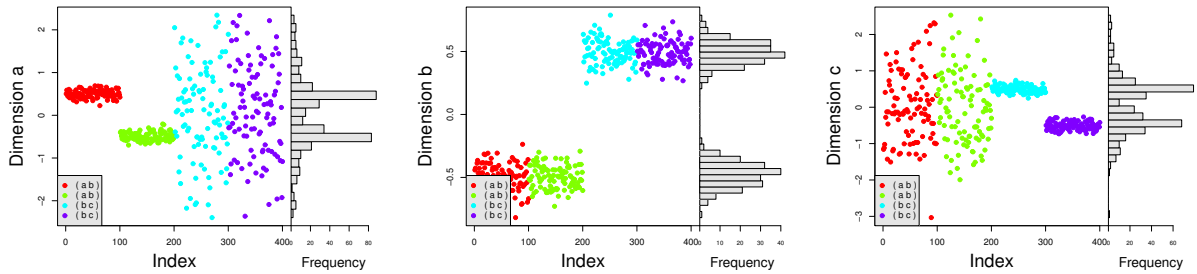
In order to verify clustering results, domain experts require the clusters to be represented in interpretable and meaningful ways. Again, subspace clusters must not only represent the cluster, but also the subspace in which it exists. Interpretation of clustering results is also important since clustering is often a first step that helps direct further study.

In addition to producing high quality, interpretable clusters, clustering algorithms must also scale well with respect to the number of instances and the number of dimensions in large datasets. Subspace clustering methods must also be scalable with respect to the dimensionality of the subspaces where the clusters are found. In high dimensional data, the number of possible subspaces is huge, requiring efficient search algorithms. The choice of search strategy creates different biases and greatly affects the assumptions made by the algorithm. In fact, there are two major types of subspace clustering algorithms, distinguished by the type of search employed. In the next section we review subspace clustering algorithms and present a hierarchy that organizes existing algorithms based on the strategy they implement.

4. SUBSPACE CLUSTERING METHODS

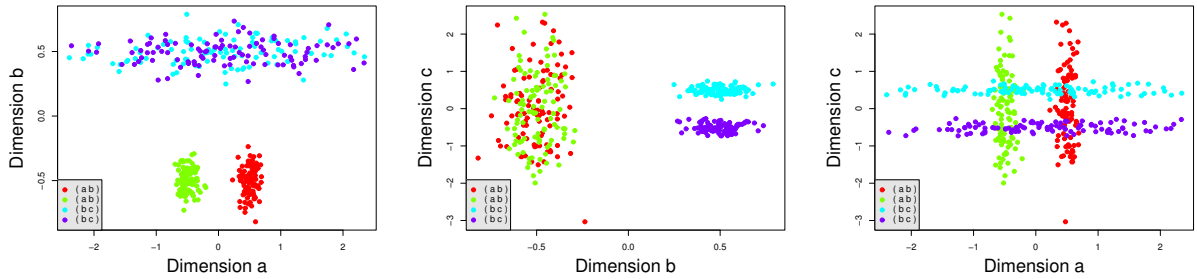
This section discusses the main subspace clustering strategies and summarizes the major subspace clustering algorithms. Figure 5 presents a hierarchy of subspace clustering algorithms organized by the search technique employed and the measure used to determine locality. The two main types of algorithms are distinguished by their approach to searching for subspaces. A naive approach might be to search through all possible subspaces and use cluster validation techniques to determine the subspaces with the best clusters [43]. This is not feasible because the subset generation problem is intractable [49; 14]. More sophisticated heuristic search methods are required, and the choice of search technique determines many of the other characteristics of an algorithm. Therefore, the first division in the hierarchy (Figure 5)) splits subspace clustering algorithms into two groups, the top-down search and bottom-up search methods.

Subspace clustering must evaluate features on only a subset of the data, representing a cluster. They must use some measure to define this context. We refer to this as a "measure of locality". We further divide the two categories of subspace clustering algorithms based on how they determine a measure of locality with which to evaluate subspaces.



(a) Dimension a (b) Dimension b (c) Dimension c

Figure 3: Sample data plotted in one dimension, with histogram. While some clustering can be seen, points from multiple clusters are grouped together in each of the three dimensions.



(a) Dims a & b (b) Dims b & c (c) Dims a & c

Figure 4: Sample data plotted in each set of two dimensions. In both (a) and (b) we can see that two clusters are properly separated, but the remaining two are mixed together. In (c) the four clusters are more visible, but still overlap each other are are impossible to completely separate.

4.1 Bottom-Up Subspace Search Methods

The bottom-up search method take advantage of the downward closure property of density to reduce the search space, using an APRIORI style approach. Algorithms first create a histogram for each dimension and selecting those bins with densities above a given threshold. The downward closure property of density means that if there are dense units in k dimensions, there are dense units in all $(k-1)$ dimensional projections. Candidate subspaces in two dimensions can then be formed using only those dimensions which contained dense units, dramatically reducing the search space. The algorithm proceeds until there are no more dense units found. Adjacent dense units are then combined to form clusters. This is not always easy, and one cluster may be mistakenly reported as two smaller clusters. The nature of the bottom-up approach leads to overlapping clusters, where one instance can be in zero or more clusters. Obtaining meaningful results is dependent on the proper tuning of the grid size and the density threshold parameters. These can be particularly difficult to set, especially since they are used across all of the dimensions in the dataset. A popular adaptation of this strategy provides data driven, adaptive grid generation to stabilize the results across a range of density thresholds.

The bottom-up methods determine locality by creating bins for each dimension and using those bins to form a multi-dimensional grid. There are two main approaches to accomplishing this. The first group consists of CLIQUE and ENCLUS which both use a static sized grid to divide each dimension into bins. The second group contains MAFIA, Cell Based Clustering (CBF), CLTree, and DOC. These methods distinguish themselves by using data driven strategies to determine the cut-points for the bins on each dimension. MAFIA and CBF use histograms to analyze the density of data in each dimension. CLTree uses a decision tree based strategy to find the best cutpoint. DOC uses a maximum width and minimum number of instances per cluster to guide a random search.

4.1.1 CLIQUE

CLIQUE [8] was one of the first algorithms proposed that attempted to find clusters within subspaces of the dataset. As described above, the algorithm combines density and grid based clustering and uses an APRIORI style technique to find clusterable subspaces. Once the dense subspaces are found they are sorted by *coverage*, where coverage is defined as the fraction of the dataset covered by the dense units in the subspace. The subspaces with the greatest coverage are kept and the rest are pruned. The algorithm then finds adjacent dense grid units in each of the selected subspaces using a depth first search. Clusters are formed by combining these units using using a greedy growth scheme. The algorithm starts with an arbitrary dense unit and greedily grows a maximal region in each dimension until the union of all the regions covers the entire cluster. Redundant regions are removed by a repeated procedure where smallest redundant regions are discarded until no further maximal region can be removed. The hyper-rectangular clusters are then defined by a Disjunctive Normal Form (DNF) expression.

CLIQUE is able to find many types and shapes of clusters

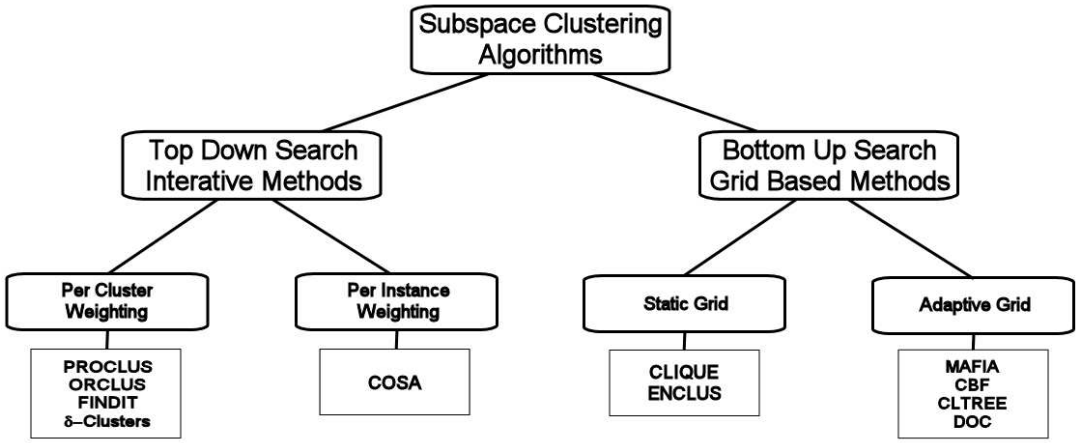


Figure 5: Hierarchy of Subspace Clustering Algorithms. At the top there are two main types of subspace clustering algorithms, distinguished by their search strategy. One group uses a top-down approach that iteratively updates weights for dimensions for each cluster. The other uses a bottom-up approach, building clusters up from dense units found in low dimensional space. Each of those groups is then further categorized based on how they define a context for the quality measure.

and presents them in easily interpretable ways. The region growing density based approach to generating clusters allows CLIQUE to find clusters of arbitrary shape. CLIQUE is also able to find any number of clusters in any number of dimensions and the number is not predetermined by a parameter. Clusters may be found in the same, overlapping, or disjoint subspaces. The DNF expressions used to represent clusters are often very interpretable. The clusters may also overlap each other meaning that instances can belong to more than one cluster. This is often advantageous in subspace clustering since the clusters often exist in different subspaces and thus represent different relationships.

Tuning the parameters for a specific dataset can be difficult. Both grid size and the density threshold are input parameters which greatly affect the quality of the clustering results. Even with proper tuning, the pruning stage can sometimes eliminate small but important clusters from consideration. Like other bottom-up algorithms, CLIQUE scales well with the number of instances and dimensions in the dataset. CLIQUE (and other similar algorithms), however, do not scale well with number of dimensions in the output clusters. This is not usually a major issue, since subspace clustering tends to be used to find low dimensional clusters in high dimensional data.

4.1.2 ENCLUS

ENCLUS [17] is a subspace clustering method based heavily on the CLIQUE algorithm. However, ENCLUS does not measure density or coverage directly, but instead measures entropy. The algorithm is based on the observation that a subspace with clusters typically has lower entropy than a subspace without clusters. Clusterability of a subspace is defined using three criteria: coverage, density, and correlation. Entropy can be used to measure all three of these criteria. Entropy decreases as the density of cells increases. Under certain conditions, entropy will also decrease as the coverage increases. *Interest* is a measure of correlation and is defined as the difference between the sum of entropy measurements for a set of dimensions and the entropy of the multi-dimension distribution. Larger values indicate higher correlation between dimensions and an interest value of zero indicates independent dimensions.

ENCLUS uses the same APRIORI style, bottom-up approach as CLIQUE to mine significant subspaces. Pruning is accomplished using the downward closure property of entropy (below a threshold ω) and the upward closure property of *interest* (i.e. correlation) to find minimally correlated subspaces. If a subspace is highly correlated (above threshold ϵ), all of it's superspaces must not be minimally correlated. Since non-minimally correlated subspaces might be of interest, ENCLUS searches for *interesting subspaces* by calculating interest *gain* and finding subspaces whose entropy exceeds ω and interest gain exceeds ϵ'. Once interesting subspaces are found, clusters can be identified using the same methodology as CLIQUE or any other existing clustering algorithm. Like CLIQUE, ENCLUS requires the user to set the grid interval size, Δ. Also, like CLIQUE the algorithm is highly sensitive to this parameter.

ENCLUS also shares much of the flexibility of CLIQUE. ENCLUS is capable of finding overlapping clusters of arbitrary shapes in subspaces of varying sizes. ENCLUS has a similar running time as CLIQUE and shares the poor scalability with respect to the subspace dimensions. One improvement the authors suggest is a pruning approach similar to the way CLIQUE prunes subspaces with low coverage, but based on entropy. This would help to reduce the number of irrelevant clusters in the output, but could also eliminate small, yet meaningful, clusters.

4.1.3 MAFIA

MAFIA [35] is another extension of CLIQUE that uses an adaptive grid based on the distribution of data to improve efficiency and cluster quality. MAFIA also introduces parallelism to improve scalability. MAFIA initially creates a histogram to determine the minimum number of bins for a dimension. The algorithm then combines adjacent cells of similar density to form larger cells. In this manner, the dimension is partitioned based on the data distribution and the re-

sulting boundaries of the cells capture the cluster perimeter more accurately than fixed sized grid cells. Once the bins have been defined, MAFIA proceeds much like CLIQUE, using an APRIORI style algorithm to generate the list of clusterable subspaces by building up from one dimension. MAFIA also attempts to allow for parallelization of the the clustering process.

In addition to requiring a density threshold parameter, MAFIA also requires the user to specify threshold for merging adjacent windows. Windows are essentially subset of bins in a histogram that is plotted for each dimension. Adjacent windows within the specified threshold are merged to form larger windows. The formation of the adaptive grid is dependent on these windows. Also, to assist the adaptive grid algorithm, MAFIA takes a default grid size as input for dimensions where the data is uniformly distributed. In such dimensions, the data is divided into a small, fixed number of partitions. Despite the adaptive nature of the grids, the algorithm is rather sensitive to these parameters.

Like related methods, MAFIA finds any number of clusters of arbitrary shape in subspaces of varying size. It also represents clusters as DNF expressions. Due to parallelization and other improvements, MAFIA performs many times faster than CLIQUE on similar datasets. It scales linearly with the number of instances and even better with the number of dimensions in the dataset (see section 5). However, like the other algorithms in this category, MAFIA's running time grows exponentially with the number of dimensions in the clusters.

4.1.4 Cell-based Clustering Method (CBF)

Cell-based Clustering (CBF) [16] attempts to address scalability issues associated with many bottom-up algorithms. One problem for other bottom-up algorithms is that the number of bins created increases dramatically as the number of dimensions increases. CBF uses a cell creation algorithm that creates optimal partitions by repeatedly examining minimum and maximum values on a given dimension which results in the generation of fewer bins (cells). CBF also addresses scalability with respect to the number of instances in the dataset. In particular, other approaches often perform poorly when the dataset is too large to fit in main memory. CBF stores the bins in an efficient filtering-based index structure which results in improved retrieval performance.

The CBF algorithm is sensitive to two parameters. *Section threshold* determines the bin frequency of a dimension. The retrieval time is reduced as the threshold value is increased because the number of records accessed is minimized. The other parameter is *cell threshold* which determines the minimum density of data points in a bin. Bins with density above this threshold are selected as potentially a member of a cluster. Like the other bottom-up methods, CBF is able to find clusters of various sizes and shapes. It also scales linearly with the number of instances in a dataset. CBF has been shown to have slightly lower precision than CLIQUE but is faster in retrieval and cluster construction [16].

4.1.5 CLTree

CLTree [53] uses a modified decision tree algorithm to select the best cutting planes for a given dataset. It uses a decision tree algorithm to partition each dimension into bins, separating areas of high density from areas of low density. Users can browse the tree and refine the final clustering results. CLTree follows the bottom-up strategy, evaluating each dimension separately and then using only those dimensions with areas of high density in further steps. The decision tree splits correspond to the boundaries of bins and are chosen using modified gain criteria that essentially measures the density of a region. Hypothetical, uniformly distributed noise is "added" to the dataset and the tree attempts to split the read data from the noise. The noise does not actually have to be added to the dataset, but instead the density can be estimated for any given bin under investigation.

While the number and size of clusters are not required ahead of time, there are two parameters. The first, min_y, is the minimum number of points that a region must contain to be considered interesting during the pruning stage. The second, min_rd, is the minimum relative density between two adjacent regions before the regions are merged to form a larger cluster. These parameters are used to prune the tree, and results are sensitive to their settings. CLTree finds clusters as hyper-rectangular regions in subspaces of different sizes. Though building the tree could be expensive ($O(n^2)$), the algorithm scales linearly with the number of instances and the number of dimensions of the subspace.

4.1.6 DOC

Density-based Optimal projective Clustering (DOC) is somewhat of a hybrid method that blends the grid based approach used by the bottom-up approaches and the iterative improvement method from the top-down approaches. DOC proposes a mathematical definition for the notion of an optimal projective cluster [64]. A projective cluster is defined as a pair (C, D) where C is a subset of the instances and D is a subset of the dimensions of the dataset. The goal is to find a pair where C exhibits a strong clustering tendency in D. To find these optimal pairs, the algorithm creates a small subset X, called the discriminating set, by random sampling. Ideally, this set can be used to differentiate between relevant and irrelevant dimensions for a cluster. For a given a cluster pair (C, D), instances p in C, and instances q in X the following should hold true: for each dimension i in D, $|q(i) - p(i)| <= w$, where w is the fixed side length of a subspace cluster or hyper-cube, given by the user. p and X are both obtained through random sampling and the algorithm is repeated with the best result being reported.

The results are heavily dependent on parameter settings. The value of w determines the maximum length of one side of a cluster. A heuristic is presented to choose an optimal value for w. DOC also requires a parameter α that specifies the minimum number of instances that can form a cluster. α along with w determine the minimum cluster density. Another parameter β is provided to specify the balance between number of points and the number of dimensions in a cluster. The clusters are hyper-rectangular in shape and represented by a pair (C, D), meaning the set of data points C is projected on the dimension set D. The running time grows linearly with the number of data points but it increases exponentially with the number of dimensions in the dataset.

4.2 Iterative Top-Down Subspace Search Methods

The top-down subspace clustering approach starts by finding an initial approximation of the clusters in the full feature space with equally weighted dimensions. Next each

dimension is assigned a weight for each cluster. The updated weights are then used in the next iteration to regenerate the clusters. This approach requires multiple iterations of expensive clustering algorithms in the full set of dimensions. Many of the implementations of this strategy use a sampling technique to improve performance. Top-down algorithms create clusters that are partitions of the dataset, meaning each instance is assigned to only one cluster. Many algorithms also allow for an additional group of outliers. Parameter tuning is necessary in order to get meaningful results. Often the most critical parameters for top-down algorithms is the number of clusters and the size of the subspaces, which are often very difficult to determine ahead of time. Also, since subspace size is a parameter, top-down algorithms tend to find clusters in the same or similarly sized subspaces. For techniques that use sampling, the size of the sample is another critical parameter and can play a large role in the quality of the final results.

Locality in top-down methods is determined by some approximation of clustering based on the weights for the dimensions obtained so far. PROCLUS, ORCLUS, FINDIT, and δ-Clusters determine the weights of instances for each cluster. COSA is unique in that it uses the k nearest neighbors for each instance in the dataset to determine the weights for each dimension for that particular instance.

4.2.1 PROCLUS

PROCLUS [5] was the first top-down subspace clustering algorithm. Similar to CLARANS [58], PROCLUS samples the data, then selects a set of k medoids and iteratively improves the clustering. The algorithm uses a three phase approach consisting of *initialization*, *iteration*, and *cluster refinement*. Initialization uses a greedy algorithm to select a set of potential medoids that are far apart from each other. The objective is to ensure that each cluster is represented by at least one instance in the selected set. The iteration phase selects a random set of k medoids from this reduced dataset, replaces bad medoids with randomly chosen new medoids, and determines if clustering has improved. Cluster quality is based on the average distance between instances and the nearest medoid. For each medoid, a set of dimensions is chosen whose average distances are small compared to statistical expectation. The total number of dimensions associated to medoids must be $k * l$, where l is an input parameter that selects the average dimensionality of cluster subspaces. Once the subspaces have been selected for each medoid, average Manhattan segmental distance is used to assign points to medoids, forming clusters. The medoid of the cluster with the least number of points is thrown out along with any medoids associated with fewer than $(N/k) * minDeviation$ points, where $minDeviation$ is an input parameter. The refinement phase computes new dimensions for each medoid based on the clusters formed and reassigns points to medoids, removing outliers.

Like many top-down methods, PROCLUS is biased toward clusters that are hyper-spherical in shape. Also, while clusters may be found in different subspaces, the subspaces must be of similar sizes since the the user must input the average number of dimensions for the clusters. Clusters are represented as sets of instances with associated medoids and subspaces and form non-overlapping partitions of the dataset with possible outliers. Due to the use of sampling, PROCLUS is somewhat faster than CLIQUE on large datasets. However, using a small number of representative points can cause PROCLUS to miss some clusters entirely. The cluster quality of PROCLUS is very sensitive to the accuracy of the input parameters, which may be difficult to determine.

4.2.2 ORCLUS

ORCLUS [6] is an extended version of the algorithm PROCLUS[5] that looks for non-axis parallel subspaces. This algorithm arose from the observation that many datasets contain inter-attribute correlations. The algorithm can be divided into three steps: assign clusters, subspace determination, and merge. During the assign phase, the algorithm iteratively assigns data points to the nearest cluster centers. The distance between two points is defined in a subspace E, where E is a set of orthonormal vectors in some d-dimensional space. Subspace determination redefines the subspace E associated with each cluster by calculating the covariance matrix for a cluster and selecting the orthonormal eigenvectors with the least spread (smallest eigenvalues). Clusters that are near each other and have similar directions of least spread are merged during the merge phase. The number of clusters and the size of the subspace dimensionality must be specified. The authors provide a general scheme for selecting a suitable value. A statistical measure called the *cluster sparsity coefficient*, is provided which can be inspected after clustering to evaluate the choice of subspace dimensionality. The algorithm is computationally intensive due mainly to the computation of covariance matrices. ORCLUS uses random sampling to improve speed and scalability and as a result may miss smaller clusters. The output clusters are defined by the cluster center and and an associated partition of the dataset, with possible outliers.

4.2.3 FINDIT

A Fast and Intelligent Subspace Clustering Algorithm using Dimension Voting, FINDIT [74] is similar in structure to PROCLUS and the other top-down methods, but uses a unique distance measure called the *Dimension Oriented Distance (DOD)*. The idea is compared to voting whereby the algorithm tallies the number of dimensions on which two instances are within a threshold distance, ϵ, of each other. The concept is based on the assumption that in higher dimensions it is more meaningful for two instances to be close in several dimensions rather than in a few [74]. The algorithm typically consists of three phases, namely *sampling phase*, *cluster forming phase*, and *data assignment phase*. The algorithms starts by selecting two small sets generated through random sampling of the data. The sets are used to determine initial representative medoids of the clusters. In the cluster forming phase the correlated dimensions are found using the *DOD* measure for each medoid. FINDIT then increments the value of ϵ and repeats this step until the cluster quality stabilizes. In the final phase all of the instances are assigned to medoids based on the subspaces found.

FINDIT requires two input parameters, the minimum number of instances in a cluster, and the minimum distance between two clusters. FINDIT is able to find clusters in subspaces of varying size. The *DOD* measure is dependent on the ϵ threshold which is determined by the algorithm in a iterative manner. The iterative phase that determines the best threshold adds significantly to the running time of the algorithm. Because of this, FINDIT employs sampling tech-

niques like the other top-down algorithms. Sampling helps to improve performance, especially with very large datasets. Section 5 shows the results of empirical experiments using the FINDIT and MAFIA algorithms.

4.2.4 δ-Clusters

The δ-Clusters algorithm uses a distance measure that attempts to capture the*coherence* exhibited by subset of instances on subset of attributes [75]. Coherent instances may not be close in many dimensions, but instead they both follow a similar trend, offset from each other. One coherent instance can be derived from another by shifting by the offset. *PearsonR* correlation [68] is used to measure coherence among instances. The algorithm starts with initial seeds and iteratively improves the overall quality of the clustering by randomly swapping attributes and data points to improve individual clusters. *Residue* measures the decrease in coherence that a particular attribute or instance brings to the cluster. The iterative process terminates when individual improvement levels off in each cluster.

The algorithm takes as parameters the number of clusters and the individual cluster size. The running time is particularly sensitive to the cluster size parameter. If the value chosen is very different from the optimal cluster size, the algorithm could take considerably longer to terminate. The main difference between δ-Clusters and other methods is the use of coherence as a similarity measure. Using this measure makes δ-clusters suitable for domains such as microarray data analysis. Often finding related genes means finding those that respond similarly to environmental conditions. The absolute response rates are often very different, but the type of response and the timing may be similar.

4.2.5 COSA

Clustering On Subsets of Attributes (COSA) [32] is an iterative algorithm that assigns weights to each dimension for each instance, not each cluster. Starting with equally weighted dimensions, the algorithm examines the k nearest neighbors (knn) of each instance. These neighborhoods are used to calculate the respective dimension weights for each instance. Higher weights are assigned to those dimensions that have a smaller dispersion within the knn group. These weights are used to calculate dimension weights for pairs of instances which are in turn used to update the distances used in the knn calculation. The process is then repeated using the new distances until the weights stabilize. The neighborhoods for each instance become increasingly enriched with instances belonging to its own cluster. The dimension weights are refined as the dimensions relevant to a cluster receive larger weights. The output is a distance matrix based on weighted inverse exponential distance and is suitable as input to any distance-based clustering method. After clustering, the weights of each dimension of cluster members are compared and an overall importance value for each dimension for each cluster is calculated.

The number of dimensions in clusters need not be specified directly, but instead is controlled by a parameter, λ, that controls the strength of incentive for clustering on more dimensions. Each cluster may exist in a different subspaces of different sizes, but they do tend to be of similar dimensionality. It also allows for the adjustment of the k used in the knn calculations. Friedman claims that the results are stable over a wide range of k values. The dimension weights are calculated for each instance and pair of instances, not for each cluster. After clustering the relevant dimensions must be calculated based on the dimension weights assigned to cluster members.

5. EMPIRICAL COMPARISON

In this section we measure the performance of representative top-down and bottom-up algorithms. We chose MAFIA [35], an advanced version of the bottom-up subspace clustering method, and FINDIT [74], an adaptation of the top-down strategy. In datasets with very high dimensionality, we expect the bottom-up approaches to perform well as they only have to search in the lower dimensionality of the hidden clusters. However, the sampling schemes of top-down approaches should scale well to large datasets. To measure the scalability of the algorithms, we measure the running time of both algorithms and vary the number of instance or the number of dimensions. We also examine how well each of the algorithms were able to determine the correct subspaces for each cluster. The implementation of each algorithm was provided by the respective authors.

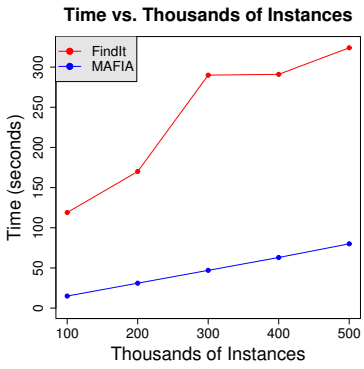
5.1 Data

To facilitate the comparison of the two algorithms, we chose to use synthetic datasets so that we have control over of the characteristics of the data. Synthetic data also allows us to easily measure the accuracy of the clustering by comparing the output of the algorithms to the known input clusters. The datasets were generated using specialized dataset generators that provided control over the number of instances, the dimensionality of the data, and the dimensions for each of the input clusters. They also output the data in the formats required by the algorithm implementations and provided the necessary meta-information to measure cluster accuracy. The data values lie in the range of 0 to 100. Clusters were generated by restricting the value of relevant dimensions for each instance in a cluster. Values for irrelevant dimensions were chosen randomly to form a uniform distribution over the entire data range.

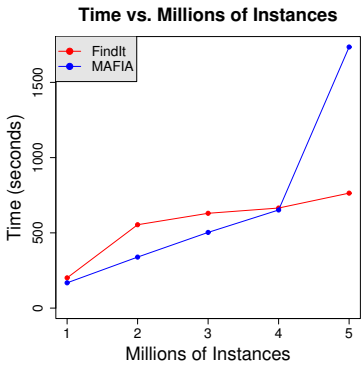
5.2 Scalability

We measured the scalability of the two algorithms in terms of the number of instances and the number of dimensions in the dataset. In the first set of tests, we fixed the number of dimensions at twenty. On average the datasets contained five hidden clusters, each in a different five dimensional subspace. The number of instances was increased first from from 100,000 to 500,000 and then from 1 million to 5 million. Both algorithms scaled linearly with the number of instances, as shown by Figure 6. MAFIA clearly out performs FINDIT through most of the cases. The superior performance of MAFIA can be attributed to the bottom-up approach which does not require as many passes through the dataset. Also, when the clusters are embedded in few dimensions, the bottom-up algorithms have the advantage that they only consider the small set of relevant dimensions during most of their search. If the clusters exist in high numbers of dimensions, then performance will degrade as the number of candidate subspaces grows exponentially with the number of dimensions in the subspace [5].

In Figure 6(b) we can see that FINDIT actually outperforms MAFIA when the number of instances approaches four million. This could be attributed to the use of random sam-



(a) Thousands



(b) Millions

Figure 6: Running time *vs.* number of instances for FINDIT (top-down) and MAFIA (bottom-up). MAFIA outperforms FINDIT in smaller datasets. However, due to the use of sampling techniques, FINDIT is able to maintain relatively high performance even with datasets containing five million records.

pling, which eventually gives FINDIT a performance edge with huge datasets. MAFIA on the other hand, must scan the entire dataset each time it makes a pass to find dense units on a given number of dimensions. We can see in Section 5.3 that sampling can cause FINDIT to miss clusters or dimensions entirely.

In the second set of tests we fixed the number of instances at 100,000 and increased the number of dimensions of the dataset. Figure 7 is a plot of the running times as the number of dimensions in the dataset is increased from 10 to 100. The datasets contained, on average, five clusters each in five dimensions. Here, the bottom-up approach is clearly superior as can be seen in Figure 7. The running time for MAFIA increased linearly with the number of dimensions in the dataset. The running time for the top-down method, FINDIT, increases rapidly as the the number of dimensions increase. Sampling does not help FINDIT in this case. As the number of dimensions increase, FINDIT must weight each dimension for each cluster in order to select the most relevant ones. The bottom-up algorithms like MAFIA, however, are not adversely affected by the additional, irrelevant dimensions. This is because those algorithms project the data into small subspaces first adding only the interesting dimensions to the search.

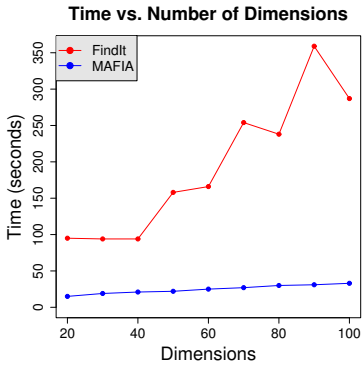


Figure 7: Running time *vs.* number of dimensions for FINDIT (top-down) and MAFIA (bottom-up). MAFIA consistently outperforms FINDIT and scales better as the number of dimensions increases.

5.3 Subspace Detection & Accuracy

In addition to comparing the algorithms scalability, we also compared how accurately each algorithm was able to determine the clusters and corresponding subspaces in the dataset. The results are presented as a *confusion matrix* that lists the relevant dimensions of the input clusters as well as those output by the algorithm. Table 1 and Table 2 show the best case input and output clusters for MAFIA and FINDIT on a dataset of 100,000 instances in 20 dimensions. The bottom-up algorithm, MAFIA, discovered all of the clusters but left out one significant dimension in four out of the five clusters. Missing one dimension in a cluster can be caused by prematurely pruning a dimension based on a coverage threshold, which can be difficult to determine. This could also occur because the density threshold may not be appropriate across all the dimensions in the dataset. The results were very similar in tests where the number of instances was increased up to 4 million. The only difference was that some clusters were reported as two separate clusters, instead of properly merged. This fracturing of clusters is an artifact of the grid-based approach used by many bottom-up algorithms that requires them to merge dense units to form the output clusters. The top-down approach used by FINDIT was better able to identify the significant dimensions for clusters it uncovered. As the number of instances was increased, FINDIT occasionally missed an entire cluster. As the dataset grew, the clusters were more difficult to find among the noise and the sampling employed by many top-down algorithms cause them to miss clusters.

Tables 4 and 3 show the results from MAFIA and FINDIT respectively when the number of dimensions in the dataset is increased to 100. MAFIA was able to detect all of the clusters. Again, it was missing one dimension for four of the five clusters. Also higher dimensionality caused the same problem that we noticed with higher numbers of instances, MAFIA mistakenly split one cluster into multiple separate clusters. FINDIT did not fare so well and we can see that FINDIT missed entire clusters and was unable to find all

Cluster	1	2	3	4	5
Input	(4, 6, 12, 14, 17)	(1, 8, 9, 15, 18)	(1, 7, 9, 18, 20)	(1, 12, 15, 18, 19)	(5, 14, 16, 18, 19)
Output	(4, 6, 14, 17)	(1, 8, 9, 15, 18)	(7, 9, 18, 20)	(12, 15, 18, 19)	(5, 14, 18, 19)

Table 1: MAFIA misses one dimension in 4 out 5 clusters with $N = 100,000$ and $D = 20$.

Cluster	1	2	3	4	5
Input	(11, 16)	(9, 14, 16)	(8, 9, 16, 17)	(0, 7, 8, 10, 14, 16)	(8, 16)
Output	(11, 16)	(9, 14, 16)	(8, 9, 16, 17)	(0, 7, 8, 10, 14, 16)	(8, 16)

Table 2: FINDIT uncovers all of the clusters in the appropriate dimensions with $N = 100,000$ and $D = 20$.

of the relevant dimensions for the clusters it did find. The top-down approach means that FINDIT must evaluate all of the dimensions, and as a greater percentage of them are irrelevant, the relevant ones are more difficult to uncover. The use of sampling can exacerbate this problem, as some clusters may be only weakly represented.

While subspace clustering seems to perform well on synthetic data, the real test for any data mining technique is to use it to solve a real-world problem. In the next section we discuss some areas where subspace clustering may prove to be very useful. Faster and more sophisticated computers have allowed many fields of study to collect massive stores of data. In particular we will look at applications of subspace clustering in information integration systems, web text mining, and bioinformatics.

6. APPLICATIONS

Limitations of current methods and the application of subspace clustering techniques to new domains drives the creation of new techniques and methods. Subspace clustering is especially effective in domains where one can expect to find relationships across a variety of perspectives. Some areas where we feel subspace clustering has great potential are information integration system, text-mining, and bioinformatics. Creating hierarchies of data sources is a difficult task for information integration systems and may be improved upon by specialized subspace clustering algorithms. Web search results can be organized into clusters based on topics to make finding relevant results easier. In Bioinformatics, DNA microarray technology allows biologists to collect an huge amount of data and the explosion of the World Wide Web threatens to drown us in a sea of poorly organized information. Much of the knowledge in these datasets can be extracted by finding and analyzing the patterns in the data. Clustering algorithms have been used with DNA microarray data in identification and characterization of disease subtypes and for molecular pathway elucidation. However, the high dimensionality of the microarray and text data makes the task of pattern discovery difficult for traditional clustering algorithms [66]. Subspace clustering techniques can be leveraged to uncover the complex relationships found in data from each of these areas.

6.1 Information Integration Systems

Information integration systems are motivated by the fact that our information needs of the future will not be satisfied by closed, centralized databases. Rather, increasingly sophisticated queries will require access to heterogeneous, autonomous information sources located in a distributed manner and accessible through the Internet. In this scenario, query optimization becomes a complex problem since the data is not centralized. The decentralization of data poses a difficult challenge for information integration systems, mainly in the determination of the best subset of sources to use for a given user query [30]. An exhaustive search on all the sources would be a naive and a costly solution. In the following, we discuss an application of subspace clustering in the context of query optimization for an information integration system developed here at ASU, Bibfinder [60].

The Bibfinder system maintains coverage statistics for each source based on a log of previous user queries. With such information, Bibfinder can rank the sources for a given query. In order to classify previously unseen queries, a hierarchy of query classes is generated to generalize the statistics. For Bibfinder, the process of generating and storing statistics is proving to be expensive. A combination of subspace clustering and classification methods offer a promising solution to this bottleneck. The subspace clustering algorithm can be applied to the query list with the queries being instances and the sources corresponding to dimensions of the dataset. The result is a rapid grouping of queries where a group represents queries coming from the same set of sources. Conceptually, each group can be considered a query class where the classes are generated in a one-step process using subspace clustering. Furthermore, the query classes can be used to train a classifier so that when a user query is given, the classifier can predict a set sources likely to be useful.

6.2 Web Text Mining

The explosion of the world wide web has prompted a surge of research attempting to deal with the heterogeneous and unstructured nature of the web. A fundamental problem with organizing web sources is that web pages are not machine readable, meaning their contents only convey semantic meaning to a human user. In addition, semantic heterogeneity is a major challenge. That is when a keyword in one domain holds a different meaning in another domain making information sharing and interoperability between heterogeneous systems difficult [72].

Recently, there has been strong interest in developing ontologies to deal with the above issues. The purpose of ontologies is to serve as a semantic, conceptual, hierarchical model representing a domain or a web page. Currently, ontologies are manually created, usually by a domain expert who identifies the key concepts in the domain and the interrelationships between them. There has been a substantial amount of research effort put toward the goal of automat-

Cluster	1	2	3	4	5
Input	(1, 5, 16, 20, 27, 58)	(1, 8, 46, 58)	(8, 17, 18, 37, 46, 58, 75)	(14, 17, 77)	(17, 26, 41, 77)
Output	(5, 16, 20, 27, 58, 81)	*None Found*	(8, 17, 18, 37, 46, 58, 75)	(17, 77)	(41)

Table 3: FINDIT misses many dimensions and and entire cluster at high dimensions with with $N = 100,000$ and $D = 100$.

Cluster	1	2	3	4	5
Input	(4, 6, 12, 14, 17)	(1, 8, 9, 15, 18)	(1, 7, 9, 18, 20)	(1, 12, 15, 18, 19)	(5, 14, 16, 18, 19)
Output	(4, 6, 14, 17)	(8, 9, 15, 18) (1, 8, 9, 18) (1, 8, 9, 15)	(7, 9, 18, 20)	(12, 15, 18, 19)	(5, 14, 18, 19)

Table 4: MAFIA misses one dimension in four out of five clusters. All of the dimensions are uncovered for cluster number two, but it is split into three smaller clusters. $N = 100,000$ and $D = 100$.

ing this process. In order to automate the process, the key concepts of a domain must be learned. Subspace clustering has two major strengths that will help it learn the concepts. First, the algorithm would find subspaces, or sets of keywords, that were represented important concepts on a page. The subspace clustering algorithm would also scale very well with the high dimensionality of the data. If the web pages are presented in the form of a document-term matrix where the instances correspond to the pages and the features corresponds to the keywords in the page, the result of subspace clustering will be the identification of set of keywords (subspaces) for a given group of pages. These keyword sets can be considered to be the main concepts connecting the corresponding groups. Ultimately, the clusters would represent a domain and their corresponding subspaces would indicate the key concepts of the domain. This information could be further utilized by training a classifier with domains and representative concepts. This classifier could then be used to classify or categorize previously unseen web pages. For example, suppose the subspace clustering algorithm identifies the domains Hospital, Music, and Sports, along with their corresponding concepts, from a list of web pages. This information can then be used to train a classifier that will be able to determine the category for newly encountered webpages.

6.3 DNA Microarray Analysis

DNA microarrays are an exciting new technology with the potential to increase our understanding of complex cellular mechanisms. Microarray datasets provide information on the expression levels of thousands of genes under hundreds of conditions. For example, we can interpret a lymphoma dataset as 100 cancer profiles with 4000 features where each feature is the expression level of a specific gene. This view allows us to uncover various cancer subtypes based upon relationships between gene expression profiles. Understanding the differences between cancer subtypes on a genetic level is crucial to understanding which types of treatments are most likely to be effective. Alternatively, we can view the data as 4000 gene profiles with 100 features corresponding to particular cancer specimens. Patterns in the data reveal information about genes whose products function together in pathways that perform complex functions in the organism. The study of these pathways and their relationships to one another can then be used to build a complete model of the cell and its functions, bridging the gap between genetic maps and living organisms [48].

Currently, microarray data must be preprocessed to reduce the number of attributes before meaningful clusters can be uncovered. In addition, individual gene products have many different roles under different circumstances. For example, a particular cancer may be subdivided along more than one set of characteristics. There may be subtypes based on the motility of the cell as well as the cells propensity to divide. Separating the specimens based on motility would require examining one set of genes, while subtypes based on cell division would be discovered when looking at a different set of genes [48; 66]. In order to find such complex relationships in massive microarray datasets, more powerful and flexible methods of analysis are required. Subspace clustering is a promising technique that extends the power of traditional feature selection by searching for unique subspaces for each cluster.

7. CONCLUSION

High dimensional data is increasingly common in many fields. As the number of dimensions increase, many clustering techniques begin to suffer from the *curse of dimensionality*, degrading the quality of the results. In high dimensions, data becomes very sparse and distance measures become increasingly meaningless. This problem has been studied extensively and there are various solutions, each appropriate for different types of high dimensional data and data mining procedures.

Subspace clustering attempts to integrate feature evaluation and clustering in order to find clusters in different subspaces. Top-down algorithms simulate this integration by using multiple iterations of evaluation, selection, and clustering. This process selects a group of instances first and then evaluates the attributes in the context of that cluster of instances. This relatively slow approach combined with the fact that many are forced to use sampling techniques makes top-down algorithms more suitable for datasets with large clusters in relatively large subspaces. The clusters uncovered by top-down methods are often hyper-spherical in nature due to the use of cluster centers to represent groups of similar instances. The clusters form non-overlapping partitions of the dataset. Some algorithms allow for an additional group of outliers that contains instances not related to any cluster or each other (other than the fact they are all outliers). Also, many require that the number of clusters and the size of the subspaces be input as parameters. The user must use their domain knowledge to help select and tune these settings.

Bottom-up algorithms integrate the clustering and subspace

selection by first selecting a subspace, then evaluating the instances in the that context. Adding one dimension at a time, they are able to work in relatively small subspaces. This allows these algorithms to scale much more easily with both the number of instances in the dataset and the number of attributes. However, performance drops quickly with the size of the subspaces in which the clusters are found. The main parameter required by these algorithms is the density threshold. This can be difficult to set, especially across all dimensions of the dataset. Fortunately, even if some dimensions are mistakenly ignored due to improper thresholds, the algorithms may still find the clusters in a smaller subspace. Adaptive grid approaches help to alleviate this problem by allowing the number of bins in a dimension to change based on the characteristics of the data in that dimension. Often, bottom-up algorithms are able to find clusters of various shapes and sizes since the clusters are formed from various cells in a grid. This means that the clusters can overlap each other with one instance having the potential to be in more than one cluster. It is also possible for an instances to be considered an outlier and not belong any cluster.

Clustering is a powerful data exploration tool capable of uncovering previously unknown patterns in data. Often, users have little knowledge of the data prior to clustering analysis and are seeking to find some interesting relationships to explore further. Unfortunately, all clustering algorithms require that the user set some parameters and make some assumptions about the clusters to be discovered. Subspace clustering algorithms allow users to break the assumption that all of the clusters in a dataset are found in the same set of dimensions.

There are many potential applications with high dimensional data where subspace clustering approaches could help to uncover patterns missed by current clustering approaches. Applications in bioinformatics and text mining are particularly relevant and present unique challenges to subspace clustering. As with any clustering techniques, finding meaningful and useful results depends on the selection of the appropriate technique and proper tuning of the algorithm via the input parameters. In order to do this, one must understand the dataset in a domain specific context in order to be able to best evaluate the results from various approaches. One must also understand the various strengths, weaknesses, and biases of the potential clustering algorithms.

8. REFERENCES

[1] D. Achlioptas. Database-friendly random projections. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 274–281. ACM Press, 2001.

[2] C. C. Aggarwal. Re-designing distance functions and distance-based applications for high dimensional data. *ACM SIGMOD Record*, 30(1):13–18, 2001.

[3] C. C. Aggarwal. Towards meaningful high-dimensional nearest neighbor search by human-computer interaction. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 593–604, 2002.

[4] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. On the surprising behavior of distance metrics in high dimensional space. In *Database Theory, Proceedings of 8th International Conference on*, pages 420–434, 2001.

[5] C. C. Aggarwal, J. L. Wolf, P. S. Yu, C. Procopiuc, and J. S. Park. Fast algorithms for projected clustering. In *Proceedings of the 1999 ACM SIGMOD international conference on Management of data*, pages 61–72. ACM Press, 1999.

[6] C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional spaces. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 70–81. ACM Press, 2000.

[7] C. C. Aggarwal and P. S. Yu. Outlier detection for high dimensional data. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 37–46. ACM Press, 2001.

[8] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 94–105. ACM Press, 1998.

[9] N. Alon, S. Dar, M. Parnas, and D. Ron. Testing of clustering. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 240–250, 2000.

[10] D. Barbará, Y. Li, and J. Couto. Coolcat: an entropy-based algorithm for categorical clustering. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 582–589. ACM Press, 2002.

[11] P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[12] E. Bingham and H. Mannila. Random projection in dimensionality reduction: applications to image and text data. In *Knowledge Discovery and Data Mining*, pages 245–250, 2001.

[13] A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *Artificial Intelligence*, 97:245–271, 1997.

[14] A. Blum and R. Rivest. Training a 3-node neural networks is NP-complete. *Neural Networks*, 5:117 – 127, 1992.

[15] Y. Cao and J. Wu. Projective art for clustering data sets in high dimensional spaces. *Neural Networks*, 15(1):105–120, 2002.

[16] J.-W. Chang and D.-S. Jin. A new cell-based clustering method for large, high-dimensional data in data mining applications. In *Proceedings of the 2002 ACM symposium on Applied computing*, pages 503–507. ACM Press, 2002.

[17] C.-H. Cheng, A. W. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 84–93. ACM Press, 1999.

[18] G. M. D. Corso. Estimating an eigenvector by the power method with a random start. *SIAM Journal on Matrix Analysis and Applications*, 18(4):913–937, October 1997.

[19] M. Dash, K. Choi, P. Scheuermann, and H. Liu. Feature selection for clustering - a filter solution. In *Data Mining, 2002. Proceedings. 2002 IEEE International Conference on*, pages 115–122, 2002.

[20] M. Dash and H. Liu. Feature selection for clustering. In *Proceedings of the Fourth Pacific Asia Conference on Knowledge Discovery and Data Mining, (PAKDD-2000). Kyoto, Japan*, pages 110–121. Springer-Verlag, 2000.

[21] M. Dash, H. Liu, and X. Xu. '$1 + 1 > 2$': merging distance and density based clustering. In *Database Systems for Advanced Applications, 2001. Proceedings. Seventh International Conference on*, pages 32–39, 2001.

[22] M. Dash, H. Liu, and J. Yao. Dimensionality reduction of unsupervised data. In *Tools with Artificial Intelligence, 1997. Proceedings., Ninth IEEE International Conference on*, pages 532–539, November 1997.

[23] M. Devaney and A. Ram. Efficient feature selection in conceptual clustering. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 92–97, 1997.

[24] C. Ding, X. He, H. Zha, and H. D. Simon. Adaptive dimension reduction for clustering high dimensional data. In *Data Mining, 2002. Proceedings., Second IEEE International Conference on*, pages 147–154, December 2002.

[25] J. G. Dy and C. E. Brodley. Feature subset selection and order identification for unsupervised learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 247–254, 2000.

[26] S. Epter, M. Krishnamoorthy, and M. Zaki. Clusterability detection and initial seed selection in large datasets. Technical Report 99-6, Rensselaer Polytechnic Institute, Computer Science Dept., Rensselaer Polytechnic Institute, Troy, NY 12180, 1999.

[27] L. Ertöz, M. Steinbach, and V. Kumar. Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data. In *Proceedings of the 2003 SIAM International Conference on Data Mining*, 2003.

[28] D. Fasulo. An analysis of recent work on clustering algorithms. Technical report, University of Washington, 1999.

[29] X. Z. Fern and C. E. Brodley. Random projection for high dimensional data clustering: A cluster ensemble approach. In *Machine Learning, Proceedings of the International Conference on*, 2003.

[30] D. Florescu, A. Y. Levy, and A. O. Mendelzon. Database techniques for the world-wide web: A survey. *SIGMOD Record*, 27(3):59–74, 1998.

[31] D. Fradkin and D. Madigan. Experiments with random projections for machine learning. In *Proceedings of the 2003 ACM KDD*. ACM Press, 2002.

[32] J. H. Friedman and J. J. Meulman. Clustering objects on subsets of attributes. `http://citeseer.nj.nec.com/friedman02clustering.html`, 2002.

[33] G. Gan. Subspace clustering for high dimensional categorical data. `http://www.math.yorku.ca/~gjgan/talk.pdf`, May 2003. Talk Given at SOS-GSSD (Southern Ontario Statistics Graduate Student Seminar Days).

[34] J. Ghosh. *Handbook of Data Mining*, chapter Scalable Clustering Methods for Data Mining. Lawrence Erlbaum Assoc, 2003.

[35] S. Goil, H. Nagesh, and A. Choudhary. Mafia: Efficient and scalable subspace clustering for very large data sets. Technical Report CPDC-TR-9906-010, Northwestern University, 2145 Sheridan Road, Evanston IL 60208, June 1999.

[36] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. Cluster validity methods: part i. *ACM SIGMOD Record*, 31(2):40–45, 2002.

[37] M. Halkidi, Y. Batistakis, and M. Vazirgiannis. Clustering validity checking methods: part ii. *ACM SIGMOD Record*, 31(3):19–27, 2002.

[38] G. Hamerly and C. Elkan. Learning the k in k-means. To be published, obtained by Dr. Liu at ACM-SIGKDD 2003 conference., 2003.

[39] J. Han, M. Kamber, and A. K. H. Tung. *Geographic Data Mining and Knowledge Discovery*, chapter Spatial clustering methods in data mining: A survey, pages 188–217. Taylor and Francis, 2001.

[40] A. Hinneburg, D. Keim, and M. Wawryniuk. Using projections to visually cluster high-dimensional data. *Computing in Science and Engineering, IEEE Journal of*, 5(2):14–25, March/April 2003.

[41] A. Hinneburg and D. A. Keim. Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. In *Very Large Data Bases, Proceedings of the 25th International Conference on*, pages 506–517, Edinburgh, 1999.

[42] I. Inza, P. Larraaga, and B. Sierra. Feature weighting for nearest neighbor by estimation of bayesian networks algorithms. Technical Report EHU-KZAA-IK-3/00, University of the Basque Country, PO Box 649. E-20080 San Sebastian. Basque Country. Spain., 2000.

[43] A. K. Jain and R. C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[44] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys (CSUR)*, 31(3):264–323, 1999.

[45] M. K. Jiawei Han. *Data Mining : Concepts and Techniques*, chapter 8, pages 335–393. Morgan Kaufmann Publishers, 2001.

[46] S. Kaski and T. Kohonen. Exploratory data analysis by the self-organizing map: Structures of welfare and poverty in the world. In A.-P. N. Refenes, Y. Abu-Mostafa, J. Moody, and A. Weigend, editors, *Neural Networks in Financial Engineering. Proceedings of the Third International Conference on Neural Networks in the Capital Markets, London, England, 11-13 October, 1995*, pages 498–507. World Scientific, Singapore, 1996.

[47] Y. Kim, W. Street, and F. Menczer. Feature selection for unsupervised learning via evolutionary search. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 365–369, 2000.

[48] I. S. Kohane, A. Kho, and A. J. Butte. *Microarrays for an Integrative Genomics*. MIT Press, 2002.

[49] R. Kohavi and G. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273–324, 1997.

[50] E. Kolatch. Clustering algorithms for spatial databases: A survey, 2001.

[51] T. Li, S. Zhu, and M. Ogihara. Algorithms for clustering high dimensional and distributed data. *Intelligent Data Analysis Journal*, 7(4):?–?, 2003.

[52] J. Lin and D. Gunopulos. Dimensionality reduction by random projection and latent semantic indexing. In *Proceedings of the Text Mining Workshop, at the 3rd SIAM International Conference on Data Mining*, May 2003.

[53] B. Liu, Y. Xia, and P. S. Yu. Clustering through decision tree construction. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 20–29. ACM Press, 2000.

[54] H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery & Data Mining*. Boston: Kluwer Academic Publishers, 1998.

[55] A. McCallum, K. Nigam, and L. H. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–178. ACM Press, 2000.

[56] B. L. Milenova and M. M. Campos. O-cluster: scalable clustering of large high dimensional data sets. In *Data Mining, Proceedings from the IEEE International Conference on*, pages 290–297, 2002.

[57] P. Mitra, C. A. Murthy, and S. K. Pal. Unsupervised feature selection using feature similarity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):301–312, 2002.

[58] R. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proceedings of the 20th VLDB Conference*, pages 144–155, 1994.

[59] E. Ng Ka Ka and A. W. chee Fu. Efficient algorithm for projected clustering. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 273–, 2002.

[60] Z. Nie and S. Kambhampati. Frequency-based coverage statistics mining for data integration. In *Proceedings of the International Conference on Data Engineering 2004*, 2003.

[61] A. Patrikainen. Projected clustering of high-dimensional binary data. Master's thesis, Helsinki University of Technology, October 2002.

[62] D. Pelleg and A. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 727–734, San Francisco, 2000. Morgan Kaufmann.

[63] J. M. Pena, J. A. Lozano, P. Larranaga, and I. Inza. Dimensionality reduction in unsupervised learning of conditional gaussian networks. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(6):590–603, June 2001.

[64] C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali. A monte carlo algorithm for fast projective clustering. In *Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 418–427. ACM Press, 2002.

[65] B. Raskutti and C. Leckie. An evaluation of criteria for measuring the quality of clusters. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99*, pages 905–910, Stockholm, Sweden, July 1999. Morgan Kaufmann.

[66] S. Raychaudhuri, P. D. Sutphin, J. T. Chang, and R. B. Altman. Basic microarray analysis: grouping and feature reduction. *Trends in Biotechnology*, 19(5):189–193, 2001.

[67] S. M. Rüger and S. E. Gauch. Feature reduction for document clustering and classification. Technical report, Computing Department, Imperial College, London, UK, 2000.

[68] U. Shardanand and P. Maes. Social information filtering: algorithms for automating word of mouth. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217. ACM Press/Addison-Wesley Publishing Co., 1995.

[69] L. Talavera. Dependency-based feature selection for clustering symbolic data, 2000.

[70] L. Talavera. Dynamic feature selection in incremental hierarchical clustering. In *Machine Learning: ECML 2000, 11th European Conference on Machine Learning, Barcelona, Catalonia, Spain, May 31 - June 2, 2000, Proceedings*, volume 1810, pages 392–403. Springer, Berlin, 2000.

[71] C. Tang and A. Zhang. An iterative strategy for pattern discovery in high-dimensional data sets. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 10–17. ACM Press, 2002.

[72] H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hübner. Ontology-based integration of information - a survey of existing approaches. In *In Stuckenschmidt, H., ed., IJCAI-01 Workshop*, pages 108–117, 2001.

[73] I. H. Witten and E. Frank. *Data Mining: Pratical Machine Leaning Tools and Techniuqes with Java Implementations*, chapter 6.6, pages 210–228. Morgan Kaufmann, 2000.

[74] K.-G. Woo and J.-H. Lee. *FINDIT: a Fast and Intelligent Subspace Clustering Algorithm using Dimension Voting*. PhD thesis, Korea Advanced Institute of Science and Technology, Taejon, Korea, 2002.

[75] J. Yang, W. Wang, H. Wang, and P. Yu. δ-clusters: capturing subspace correlation in a large data set. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 517–528, 2002.

[76] L. Yu and H. Liu. Feature selection for high-dimensional data: a fast correlation-based filter solution. In *Proceedings of the twentieth International Conference on Machine Learning*, pages 856–863, 2003.

[77] M. Zait and H. Messatfa. A comparative study of clustering methods. *Future Generation Computer Systems*, 13(2-3):149–159, November 1997.

APPENDIX

A. DEFINITIONS AND NOTATION

Dataset - A dataset consists of a matrix of data values. Rows represent individual *instances* and columns represent *dimensions*.

Instance - An *instance* refers to an individual row in a dataset. It typically refers to a vector of d measurements. Other common terms include *observation*, *datum*, *feature vector*, or *pattern*. N is defined as the number of instances in a given dataset.

Dimension - A *dimension* refers to an *attribute* or *feature* of the dataset. d is defined as the number of dimensions in a dataset.

Cluster - A group of instances in a dataset that are more similar to each other than to other instances. Often, similarity is measured using a distance metric over some or all of the dimensions in the dataset. k is defined as the number of clusters in a dataset.

Clustering - The process of discovering groups (clusters) of instances such that instances in one cluster are more similar to each other than to instances in another cluster. The measures used to determine similarity can vary a great deal, leading to a wide variety of definition and approaches.

Subspace - A *subspace* is a subset of the d dimensions of a given dataset.

Clusterability - A measure of how well a given set of instances can be clustered. Intuitively, this is a measure of how "tight" the results of clustering are. Having tight clustering means that intra cluster distances tend to be much larger than inter cluster distances.

Subspace Clustering - These clustering techniques seek to find clusters in a dataset by selecting the most relevant dimensions for each cluster separately. There are two main approaches. The top-down iterative approach and the bottom-up grid approach. Subspace clustering is sometimes referred to as *projected clustering*.

Measure of Locality - The measure used to determine a context of instances and/or features in which to apply a quality measure. Top-down subspace clustering methods use either an approximate clustering or the nearest neighbors of an instance. Bottom-up approaches use the cells of a grid created by defining divisions within each dimension.

Feature Transformation - Techniques that generate new features by creating combinations of the original features. Often, a small number of these new features are used to summarize dataset in fewer dimensions.

Feature Selection - The process of determining and selecting the dimensions (features) that are most relevant to the data mining task. This technique is also known as *attribute selection* or *relevance analysis*.