Evaluating Subspace Clustering Algorithms *

Lance Parsons
lparsons@asu.edu

Ehtesham Haque
Ehtesham.Haque@asu.edu

Huan Liu
hliu@asu.edu

Department of Computer Science Engineering
Arizona State University, Tempe, AZ 85281

Abstract

Clustering techniques often define the similarity between instances using distance measures over the various dimensions of the data [12, 14]. Subspace clustering is an extension of traditional clustering that seeks to find clusters in different subspaces within a dataset. Traditional clustering algorithms consider all of the dimensions of an input dataset in an attempt to learn as much as possible about each instance described. In high dimensional data, however, many of the dimensions are often irrelevant. These irrelevant dimensions confuse clustering algorithms by hiding clusters in noisy data. In very high dimensions it is common for all of the instances in a dataset to be nearly equidistant from each other, completely masking the clusters. Subspace clustering algorithms localize the search for relevant dimensions allowing them to find clusters that exist in multiple, possibly overlapping subspaces. This paper presents a survey of the various subspace clustering algorithms. We then compare the two main approaches to subspace clustering using empirical scalability and accuracy tests.

1 Introduction and 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 of the data [12, 14]. Subspace clustering is an extension of traditional clustering that seeks to find clusters in different subspaces within a dataset. 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 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 used somewhat successfully to improve cluster quality. These algorithms find a subset of dimensions on which to perform clustering by removing irrelevant and redundant dimensions. The problem with feature selection arises when the clusters in the dataset exist in multiple, possibly overlapping subspaces. Subspace clustering algorithms attempt to find such clusters.

Different approaches to clustering often define clusters in different ways. One type of clustering creates 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. Another approach to clustering creates overlapping clusters, allowing an instance to belong to more than one group. Usually these methods also allow an instance to be an outlier, belonging to no particular cluster. No one type of clustering is better than the others, but some are more appropriate to certain problems. Domain specific knowledge about the data is often very helpful in determining which type of cluster formation will be the most useful.

There are a number of excellent surveys of clustering techniques available. The classic book by Jain and Dubes [13] offers an aging, but comprehensive look at clustering. Zait and Messatfa offer a comparative study of clustering methods [25]. Jain *et al.* published another survey in 1999 [12]. More recent data mining texts include a chapter on clustering [9, 11, 14, 22]. Kolatch presents an updated hierarchy of clustering algorithms in [15]. One of the more recent and comprehensive surveys was published as a technical report by Berhkin and includes a small section on subspace clustering [4]. Gan presented a small survey of subspace clustering methods at the Southern Ontario Statistical Graduate Students Seminar Days [8]. However, there is very little available that deals with the subject of subspace clustering in a comparative and comprehensive manner.

2 Subspace Clustering 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.

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

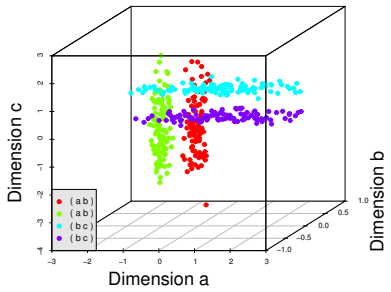


Figure 1: Sample dataset with four clusters, each in two dimensions with the third dimension being noise. Points from two clusters can be very close together, confusing many traditional clustering algorithms.

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 1. When k-means is used to cluster this sample data, it does a poor job of finding the clusters 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. This suggests that we reduce the dimensionality of the dataset by removing the irrelevant dimensions.

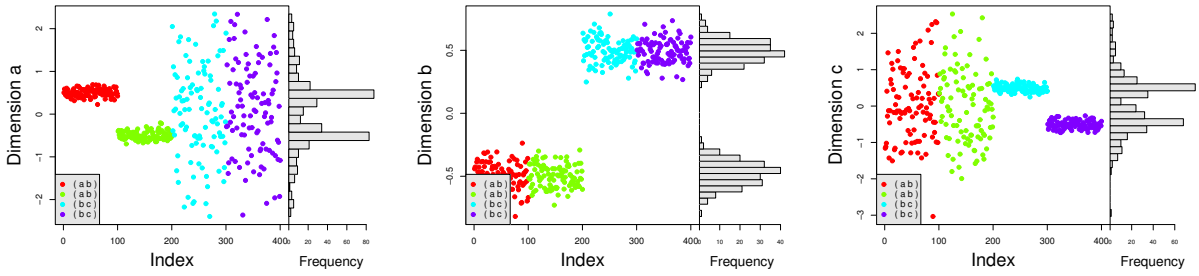
Feature transformation techniques such as Principle Component Analysis do not help in this instance. Since relative distances are preserved and the effects of the irrelevant dimension remain. Instead, we might try using a feature selection algorithm to remove one or two dimensions. Figure 2 shows the data projected in a single dimension (organized by index on the x-axis for ease of interpretation). If we only remove one dimension, we produce the graphs in Figure 3. We can see that none of these projections of the data are sufficient to fully separate the four clusters.

However, it is worth noting that the first two clusters are easily separated from each other and from the rest of the data when projected into dimensions a and b (Figure 3(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 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 3(b)). Thus, the key to finding each of the clusters in this dataset is to look in the appropriate subspaces. 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. There are two main approaches to subspace clustering based the search method employed by the algorithm. The next two sections discuss these approaches and present a summary of the various subspace clustering algorithms.

3 Bottom-Up Subspace Search Methods

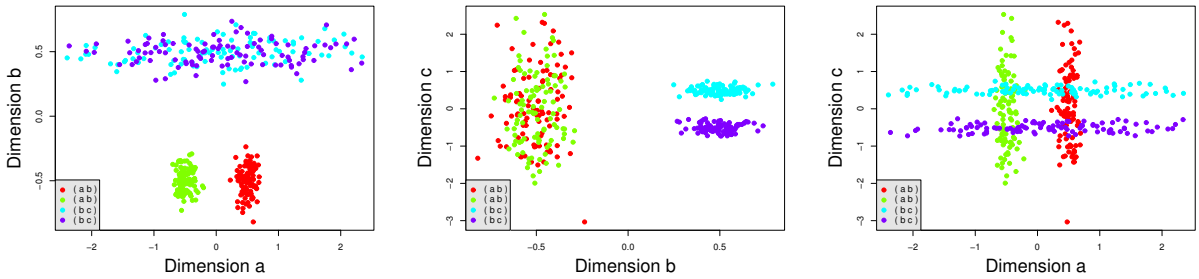
The bottom-up search method take advantage of the downward closure property of density to reduce the search space, using an APRIORI style approach. Algorithms first create a histogram for each dimension and selecting those bins with densities above a given threshold. The downward closure property of 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.

3.1 CLIQUE The CLIQUE algorithm [3] was one of the first subspace clustering algorithms. The algorithm combines density and grid based clustering and uses an APRIORI style search technique to find dense subspaces. Once the dense subspaces are found they are sorted by *coverage*, defined as the fraction of the dataset the dense units in the subspace represent. 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



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

Figure 2: 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 3: 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.

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.

The region growing, density based approach to generating clusters allows CLIQUE to find clusters of arbitrary shape, in any number of dimensions. Clusters may be found in the same, overlapping, or disjoint subspaces. The DNF expressions used to represent clusters are often very interpretable and can describe overlapping clusters, 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.

3.2 ENCLUS The ENCLUS [6] subspace clustering method is 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 cell density 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. The search 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.

3.3 MAFIA The MAFIA [10, 17, 18] algorithm extends CLIQUE by using 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 resulting 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.

3.4 Cell-based Clustering CBF [5] 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.

3.5 CLTree The CLuster Tree algorithm [16] follows the bottom-up strategy, evaluating each dimension separately and then using only those dimensions with areas of high density in further steps. It uses a modified decision tree algorithm to adaptively partition each dimension into bins, separating areas of high density from areas of low density. The decision tree splits correspond to the boundaries of bins. Hypothetical, uniformly distributed noise is "added" to the dataset and the algorithm attempts to find splits to create nodes of pure noise or pure data. 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.

3.6 Density-based Optimal projective Clustering DOC [20] is a Monte Carlo algorithm that blends the grid based approach used by the bottom-up approaches and the iterative improvement method from the top-down approaches. 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. This set can is 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, given by the user. p and X are both obtained through random sampling and the algorithm is repeated with the best result being reported.

4 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 assigend 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.

4.1 PROCLUS PROjected CLUStering [1] was the first top-down subspace clustering algorithm. Similar to CLARANS [19], 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 selects a set of potential medoids that are far apart using a greedy algorithm. 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. Once the subspaces have been selected for each medoid, average Manhattan segmental distance is used to assign points to medoids, forming clusters. The refinement phase computes a new list of relevant dimensions for each medoid based on the clusters formed and reassigns points to medoids, removing outliers.

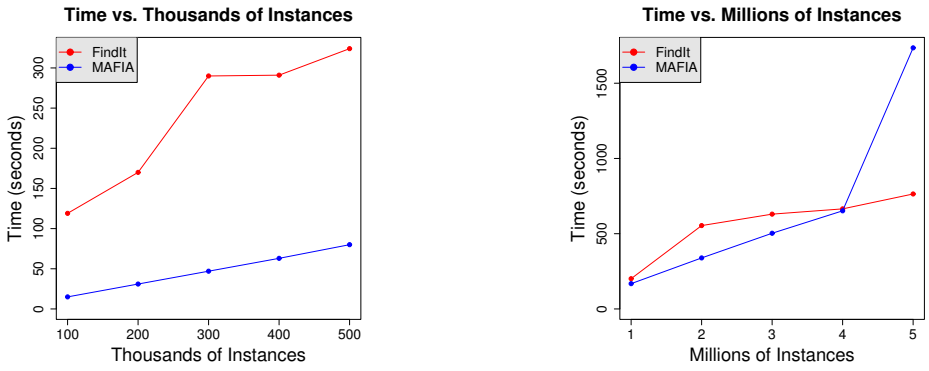
The distance based approach of PROCLUS is biased toward clusters that are hyper-spherical in shape. Also, while clusters may be found in different subspaces, the subspaces must be of similar sizes since the the user must input the average number of dimensions for the clusters. Clusters are represented as sets of instances with associated medoids and subspaces and form non-overlapping partitions of the dataset with possible outliers. PROCLUS is actually somewhat faster than CLIQUE due to the sampling of large datasets. However, using a small number of representative points can cause PROCLUS to miss some clusters entirely.

4.2 ORCLUS Arbitrarily ORiented projected CLUSter generation [2] is an extended version of the algorithm PROCLUS [1] that looks for non-axes 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.

4.3 FINDIT Fast and INtelligent subspace clustering algorithm using DImension VoTing (FINDIT) [23] uses a unique distance measure called the *Dimension Oriented Distance* (DOD). DOD 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 [23]. 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 employs sampling techniques like the other top-down algorithms. Sampling helps to improve performance, especially with very large datasets. The results of empirical experiments using FINDIT and MAFIA are shown in §5.

4.4 δ-Clusters The δ-Clusters algorithm [24] uses a distance measure that attempts to capture the *coherence* exhibited by subset of instances on subset of attributes. Coherent instances may not be close, but instead both follow a similar trend, offset from each other. One coherent instance can be derived from another by shifting by an offset. *PearsonR* correlation [21] 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 entry (attribute or instance) brings to the cluster. The iterative process terminates when individual improvement levels off in each cluster.

4.5 COSA Clustering On Subsets of Attributes (COSA) [7] assigns weights to each dimension for each instance, not each cluster. Starting with equally weighted dimensions, the algorithm examines the k nearest neighbors 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 used to update the distances used in the *KNN* calculation. The process is 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. Af-



(a) Thousands

(b) Millions

Figure 4: Running time *vs.* instances

ter clustering, the weights for each dimension of cluster members are compared and an overall importance value for each dimension is calculated for each cluster.

5 Empirical Evaluation

In this section we measure the performance of representative top-down and bottom-up algorithms. We chose MAFIA [10], an advanced version of the bottom-up subspace clustering method, and FINDIT [23], 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 that allow us to control 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 4. 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 [1].

In Figure 4(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 sampling, which eventually gives FINDIT a performance edge in 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 §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 di-

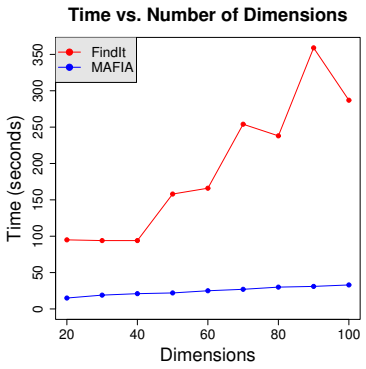


Figure 5: Running time *vs.* number of dimensions

mensions of the dataset. Figure 5 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 5. 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.

5.3 Subspace Detection and Cluster Accuracy

In addition to comparing scalability, we also compared how accurately each algorithm was able to determine the clusters and corresponding subspaces in the dataset. The results are presented in the form of 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 the premature pruning of 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 in 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 3 and 4 show the results from MAFIA and FINDIT 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 an entire cluster and was unable to find all 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. Sampling can add to this problem, as some clusters may only be weakly represented.

6 Conclusions

High dimensional data is becoming 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 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

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 $D = 20$.

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 perform the clustering and the subspace selection simultaneously, but in small subspaces, adding one dimension at a time. This allows these algorithms to scale much more easily with both the number of instances in the dataset as well as 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 also 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.

References

[1] Charu C. Aggarwal, Joel L. Wolf, Phillip S. Yu, Cecilia Procopiuc, and Jong Soo 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.

[2] Charu C. Aggarwal and Phillip 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.

[3] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar 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.

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

[5] Jae-Woo Chang and Du-Seok 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.

[6] Chun-Hung Cheng, Ada Waichee Fu, and Yi Zhang. Entropy-based subspace clustering for mining numerical data. In *Proceedings of the fifth ACM SIGKDD*

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 3: 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 $D = 100$.

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

international conference on Knowledge discovery and data mining, pages 84–93. ACM Press, 1999.

[7] Jerome H. Friedman and Jacqueline J. Meulman. Clustering objects on subsets of attributes. 2002.

[8] Guojun Gan. Subspace clustering for high dimensional categorical data. May 2003. Talk Given at SOSGSSD.

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

[10] Sanjay Goil, Harsha Nagesh, and Alok 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.

[11] 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.

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

[13] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[14] Micheline Kamber Jiawei Han. *Data Mining : Concepts and Techniques*, chapter 8, pages 335–393. Morgan Kaufmann Publishers, 2001.

[15] Erica Kolatch. Clustering algorithms for spatial databases: A survey, 2001.

[16] 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.

[17] Harsha S. Nagesh. High performance subspace clustering for massive data sets. Master's thesis, Northwestern Univeristy, 2145 Sheridan Road, Evanston IL 60208, June 1999.

[18] Harsha S. Nagesh, Sanjay Goil, and Alok Choudhary. A scalable parallel subspace clustering algorithm for massive data sets. June 2000.

[19] 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.

[20] Cecilia M. Procopiuc, Michael Jones, Pankaj 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.

[21] 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.

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

[23] 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.

[24] Jiong Yang, Wei Wang, Haixun 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.

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