Adaptive Grids for Clustering Massive Data Sets*

***Harsha Nagesh*[†]*, Sanjay Goil*[‡]*, and Alok Choudhary*[§]

Clustering is a key data mining problem. Density and grid based technique is a popular way to mine clusters in a large multi-dimensional space wherein clusters are regarded as dense regions than their surroundings. The attribute values and ranges of these attributes characterize the clusters. Fine grid sizes lead to a huge amount of computation while coarse grid sizes result in loss in quality of clusters found. Also, varied grid sizes result in discovering clusters with different cluster descriptions. The technique of Adaptive grids enables to use grids based on the data distribution and does not require the user to specify any parameters like the grid size or the density thresholds. Further, clusters could be embedded in a subspace of a high dimensional space. We propose a modified bottom-up subspace clustering algorithm to discover clusters in all possible subspaces. Our method scales linearly with the data dimensionality and the size of the data set. Experimental results on a wide variety of synthetic and real data sets demonstrate the effectiveness of Adaptive grids and the effect of the modified subspace clustering algorithm. Our algorithm explores at-least an order of magnitude more number of subspaces than the original algorithm and the use of adaptive grids yields on an average of two orders of magnitude speedup as compared to the method with user specified grid size and threshold.

*This work was supported in part by NSF Young Investigator Award CCR-9357840, NSF CCR-9509143 and Department of Energy ASCI/ASAP program

[†]Mr.Harsha Nagesh, Bell-labs Research, Murray Hill, NJ, harsha@research.bell-labs.com

[‡]Dr. Sanjay Goil, Sun Microsystems Inc., sanjay.goil@eng.sun.com

[§]Prof. Alok Choudhary, Department of ECE, Northwestern University, choudhar@ece.nwu.edu

1 Introduction

The ability to identify *interesting* patterns in large scale consumer data empowers business establishments to leverage on the information obtained. Clustering process is a data mining technique which finds such patterns, previously unknown in large scale data, embedded in a large multi-dimensional space. Clustering techniques find application in several fields. Clustering web documents based on web logs has been studied in [1], customer segmentation based on similarity of buying interests is explored as collaborative filtering in [2], [3] explores the detection of clusters in geographic information systems. Clustering algorithms need to address several issues. Scalability of these algorithms with the data base size is as important as their scalability with the dimensionality of the data sets. Effective representation of the detected clusters is as important as cluster detection. Simplicity in reporting clusters among those found improves the usability of the algorithm. Further, most clustering algorithms require key input parameters from the user which are hard to determine and also greatly control the process of cluster detection. Clusters could also be embedded in a subspace of the total data space. Detection of clusters in all possible subspaces results in an exponential algorithm as the number of subspaces is exponential in the data dimension. Most of the earlier works in statistics and data mining [4, 5] operate and find clusters in the whole data space. Many clustering algorithms [3, 6, 5] require user input of several parameters like the number of clusters, average dimensionality of the cluster, etc. which are not only difficult to determine but are also not practical for real-world data sets.

We use a grid and density based approach for cluster detection in subspaces. Density based approaches regard clusters as high density regions than their surroundings. In a grid and density approach a multi-dimensional space is divided into a large number of hyper-rectangular regions and regions which have more points than a specified threshold are identified as dense. Finally dense hyper-rectangular regions that are adjacent to each other are merged to find the embedded clusters. The quality of results and the computation requirements heavily depend on the number of bins in each dimension. Hence, determination of bin sizes automatically based on the data distribution greatly helps in finding correct clusters of high quality and reduces the computation substantially.

1.1 Contributions

In this paper we present MAFIA [1], a scalable subspace clustering algorithm using adaptive computation of the finite intervals (bins) in each dimension, which are merged to explore clusters in higher dimensions. Adaptive grid sizes improves the clustering quality by concentrating on the portions of the data space which have more points and thus are more likely to be part of a cluster region enabling minimal length DNF expressions, important for interpreting results by the end-user. MAFIA does not require any key user inputs and takes in only the strength of the clusters that needs to be discovered in the given data set. Further, we present a modified bottom-up algorithm for cluster detection in all possible subspaces previously

[1] Merging of Adaptive Finite Intervals

unexplored by other subspace clustering algorithms. We describe recent work on clustering techniques in databases in Section 2. Density and grid based clustering is presented in Section 3, where we describe subspace clustering as introduced by [7] and also describe our approach of using adaptive grids. Section 4 presents the modified subspace clustering algorithm with a theoretical analysis. Finally, Section 5 presents the performance evaluation on a wide variety of synthetic and real data sets with large number of dimensions, highlighting both scalability of the algorithms and the quality of the clustering. Section 6 concludes the paper.

2 Related Work

Clustering algorithms have long been studied in statistics [8], machine learning [9], pattern recognition, image processing [10] and databases. k-means, k-mediods, CLARANS [11], BIRCH [5], CURE [12] are some of the earlier works. However, none of these algorithms detect clusters in subspaces. PROCLUS [6], a subspace clustering algorithm finds representative cluster centers in an appropriate set of cluster dimensions. It needs the number of clusters, k, and the average cluster dimensionality, l, as input parameters, both of which are not possible to be known a-priori for real data sets. Density and grid based approaches regard clusters as regions of data space in which objects are dense and are separated by regions of low object density [13]. The grid size determines the computations and the quality of the clustering. CLIQUE, a density and grid based approach for high dimensional data sets [7], detects clusters in the highest dimensional subspaces taking the size of the grid and a global density threshold for clusters as inputs. The computation complexity and the quality of clustering is heavily dependent on these parameters. ENCLUS [14], an entropy based subspace clustering algorithm requires a prohibitive amount of time to just discover interesting subspaces in which clusters are embedded and requires entropy thresholds as inputs, which is not intuitive for the user.

3 Density and Grid based Clustering

Density based approaches regard clusters as high density regions than their surroundings. A common way of finding high-density regions in the data space is based on the grid cell densities [13]. A histogram is constructed by partitioning the data space into a number of non-overlapping regions and then mapping the data points to each cell in the grid. Equal length intervals are used in [7] to partition each dimension, which results in uniform volume cells. The number of points inside the cell with respect to the volume of the cell can be used to determine the density of the cell. Clusters are unions of connected high density cells. Two k-dimensional cells are connected if they have a common face in the k-dimensional space or if they are connected by a common cell. Creating a histogram that counts the points contained in each unit is infeasible in high dimensional data as the number of hyper-rectangles is exponential in the dimensionality of the data set. Subspace clustering further complicates the problem as it results in an explosion of such units. One needs to create histograms in hyper-rectangles formed in all possible subspaces. A bottom-up approach of finding dense units and merging them to find dense clusters in higher dimensional subspaces has been proposed in CLIQUE [7]. Each dimension

is divided into a user specified number of intervals, ϵ. The algorithm starts by determining 1-dimensional dense units by making a pass over the data. In [7] candidate dense cells in any k dimensions are obtained by merging the dense cells in $(k-1)$ dimensions which share the *first* $(k-2)$ dimensions. A pass over data is made to find which of the candidate dense cells are actually dense. The algorithm terminates when no more candidate dense cells are generated. In [7] candidate dense units are pruned based on a minimum description length technique to find the dense units only in *interesting* subspaces. However, as noted in [7] this could result in missing some dense units in the pruned subspaces. In order to maintain the high quality of clustering we do not use this pruning technique.

3.1 Adaptive Grids

We propose an adaptive interval size in which bins are determined based on the data distribution in a particular dimension. The size of the bin and hence number of bins in each dimension in turn determine the computation and quality of clustering. Finer grids leads to an explosion in the number of candidate dense units (CDUs), while coarser grids leads to fewer bins, and regions with noise data might also get propagated as dense cells. Also, a user defined uniform grid size may fail to detect many clusters or may yield very poor quality results. A single pass over the data is done in order to construct a histogram in every dimension. Algorithm 3.1 describes the steps of the adaptive grid technique. The domain of each dimension is divided into fine intervals, each of size x. The size of each bin, x, is selected such that each dimension has a minimum of 1000 fine bins. If the range of the dimension is from m to n then we set the number of bins in that dimension to be $\max(1000, (n-m))$ and correspondingly find the value of x. We assume that all attributes have been normalized to the same base quantity while finding m, n and hence x. The maximum of the histogram value within a window is taken to reflect the window value. Adjacent windows whose values differ by less than a threshold percentage, β, are merged together to form larger windows ensuring that we divide the dimensions into variable sized bins which capture the data distribution. However, in dimensions where data is uniformly distributed this results in a single bin and indicates much less likelihood of finding a cluster. However, we split the domain into a small fixed number of partitions, set a high threshold as this dimension is less likely to be part of a cluster and collect statistics for these bins to examine further. This technique greatly reduces the computation time by limiting the degree of bin contribution from non-cluster dimensions . Variable sized bins are assigned a variable threshold. A variable sized bin is likely to be part of a cluster if it has a significantly (which we call *cluster dominance factor* α [15]) greater number of points than it would have had, had the data been uniformly distributed in that dimension. Thus, for a bin of size a in a dimension of size D_i we set its threshold to be $\frac{\alpha a N}{D_i}$, where N is the total number of data points.

Effect of Adaptive Grids on Computation

Figure 1 shows the histogram of data in two dimensions. Figure 1(a) shows the user defined uniform sized grid used in the approach of [7] resulting in a large number of CDUs (rectangles in the figure). With the increase in data set size, each pass

Algorithm 3.1 Adaptive Grid Computation

D_i - Domain of A_i
N - Total number of data points in the data set
a - Size of a generic bin
for each dimension $A_i, i \in (1, \ldots d)$
  Divide D_i into windows of some small size x
  Compute the histogram for each unit of A_i, and set the value of
  the window to the maximum in the window
  From left to right merge two adjacent units if they are within a threshold β
  /* Single bin implies an equi-distributed dimension */
  if(number of bins == 1)
    Divide the dimension A_i into a fixed number of equal partitions.
  Compute the threshold of each bin of size a as $\frac{\alpha a N}{D_i}$
end

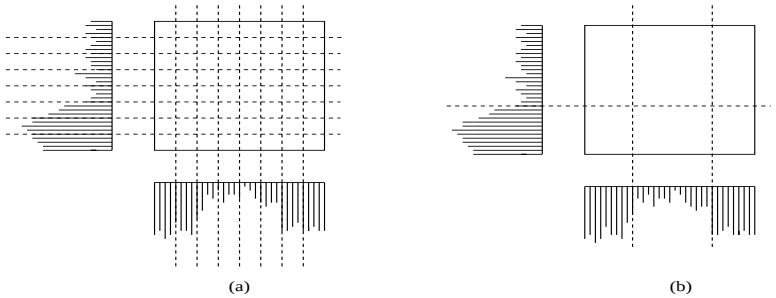


Figure 1. *(a) Uniform grid size (b) Adaptive grid size*

over the data in the bottom-up algorithm results in evaluating a large number of candidate dense units. However, as seen in Figure 1(b) adaptive grids makes use of the data distribution and computes the minimum number of bins as required in each dimension resulting in very few CDUs.

Effect of Adaptive Grids on Quality of Clustering

Figure 2 shows a 'plus' shaped cluster ($abcdefghijkl$) as discovered by [7] and our algorithm using adaptive grids. The cluster reported by CLIQUE, *pqrs*, is shown in Figure 2(a). Parts of the cluster discovered by [7] are not in the original defined cluster $abcdefghijkl$ and also parts of the original cluster are thrown away as outliers. MAFIA uses adaptive grid boundaries and so the cluster definitions are minimal DNF expressions representing the clusters accurately. MAFIA develops grid boundaries very close to the boundaries of the cluster and reports $abcdefghijkl$ shown in Figure 2(b) as the cluster with the DNF expression $(l, y) \wedge (m, z) \wedge (n, y) \wedge (m, x) \wedge (m, y)$.

4 MAFIA Implementation

MAFIA consists mainly of the following steps. The algorithm starts by making a pass over the data in chunks of B records to enables scaling to out-of-core data sets. A histogram of the data is built in each dimension as elaborated in Algorithm 3.1. The adaptive finite intervals in every dimension is determined and the bin sizes

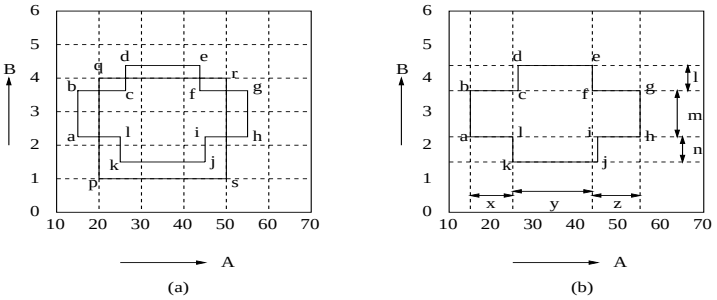


Figure 2. *(a) Cluster discovered by CLIQUE (b) Cluster discovered by MAFIA*

and thresholds for every bin formed is found. Each bin thus found is considered to be a CDU. CDUs in dimension k are built by combining dense units of dimension $k-1$ such that they share *any* of the $k-2$ dimensions. The section 4.1 elaborates the steps involved in this process. Further we elaborate our modified bottom-up algorithm and provide a theoretical analysis. The algorithm spends most of its time in making a pass over the data and finding out the dense units among the CDUs formed in every dimension. Repeated passes over the data need to be done to build dense units of higher dimensions. After finding the histogram count of the CDUs in a particular dimension, dense units are identified and dense unit data structures are built for the next higher dimension. A CDU is dense if its histogram count is greater than the thresholds of all the bins that form the CDU. The algorithm terminates when no more dense units exists. Clusters are finally reported at the end of the program.

Algorithm 4.1 MAFIA Algorithm

N - Number of records; d - Dimensionality of data
A_i - i^{th} attribute $i \in d$; B - Number of records that fit in memory buffer
  Read data in chunks of B records and
  build a histogram in each dimension $A_i, i \in (1, \ldots, d)$
  Determine adaptive intervals using the histogram in each
  dimension $A_i, i \in d$ and also fix the threshold level
  Set candidate dense units to the bins found in each dimension
  Set current dimensionality, k to 1
  while (no more dense units are found)
    if ($k > 1$) { Find-candidate-dense-units();}
    Read data in chunks of B records and for every record
    populate the candidate dense units
    Identify-dense-units();
    Register non dense units with the print data structures
    Build-dense-unit-data-structures();
  report-clusters();
  end

4.1 Building Candidate Dense Units

The approach in [7] fails to explore all possible candidate dense cells. For example, consider two 3-D dense units $\{a_1, c_7, b_8\}$ and $\{c_7, b_8, d_9\}$, where (a, b, c, d) are the bins in the dimensions indicated by their subscripts. Figure 3 shows the process of combination of these two 3-D dense units to form a potential 4-D CDU. We can

easily see that the two dense units results in a 4-D CDU $\{a_1, c_7, b_8, d_9\}$ which is not formed by the approach in [7]. Thus our approach of building k-dimensional CDUs sharing *any* of the $(k-2)$ dimensions *provably* explores all the CDUs. We shall now present results to illustrate the large number of subspaces that are explored correctly by our algorithm.

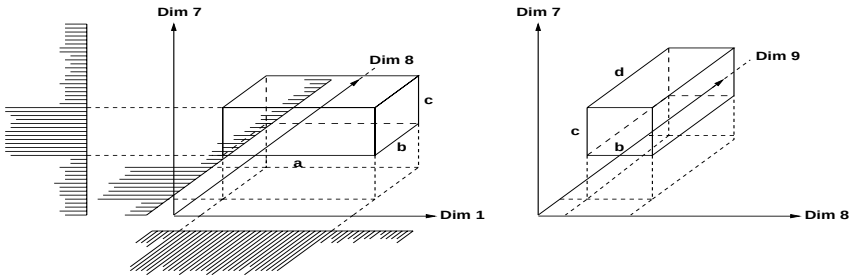


Figure 3. MAFIA: *Building A CDU in dimension 4 ($\{a_1, c_7, b_8, d_9\}$)*

Let Ndu be the number of dense units in dimension $k-1$ which combine to form the CDUs in dimension k. Let d be the dimension of the data used in the clustering process. We shall evaluate the total number of CDUs that could be generated by MAFIA and the approach in [7] and finally compare the total number of subspaces that MAFIA explores compared to [7].

MAFIA: Candidate Dense Unit Generation Process

Candidate dense units in k dimensions, are obtained by merging any two dense cells, represented by an ordered set of $(k-1)$ dimensions, such that they share **any** *of the $(k-2)$ dimensions.* Each dense unit (du) needs to be compared with every other dense unit to identify the CDUs, resulting in an $O(Ndu^2)$ algorithm. Let us assume that the given data in a d-dimensional space has clusters embedded in all possible subspaces resulting in dense units of dimension $k-1$ in all ${}^dC_{k-1}$ subspaces for a worst case analysis. Two dense units can be combined together if they share any of their $k-2$ dimensions and further, if the bins in the matched dimensions also match. The $k-1$ dimensions of each dense unit can be looked as a sequence of $k-1$ distinct integers, all of which are less than d. Thus the generation of a k dimensional CDU is equivalent to extending a $k-1$ sequence to a k sequence of integers. The number of sequences of length $k-1$ is given by ${}^dC_{k-1}$. Each $k-1$ subsequence can be extended into a k sequence in $\{d-(k-1)\}$ ways. Hence a particular set of $\{d-(k-1)\}$ sequences each of length k (obtained by extending a given $k-1$ sequence to a k sequence) contains a common $k-1$ length subsequence. Any two dense units belonging to this set can be combined to form a CDU. Thus for a given $k-1$ subsequence, we can form ${}^{\{d-(k-1)\}}C_2$ CDUs. If N_{MAFIA} represents the total number of possible CDUs that can be generated, we have

$$N_{\text{MAFIA}} = {}^dC_{k-1} \times {}^{\{d-(k-1)\}}C_2 \tag{1}$$

as there are ${}^dC_{k-1}$ subsequences each of length $k-1$.

CLIQUE: Candidate Dense Unit Generation Process

Candidate dense units in k dimensions, are obtained by merging any two dense cells, represented by an ordered set of $(k-1)$ *dimensions, such that they share* **first** *of the* $(k-2)$ *dimensions.* Let N_{CLIQUE}, be the total number of CDUs generated for a data set containing clusters with complete subspace coverage in dimension k. N_{CLIQUE} can be found by using an exhaustive enumeration technique. As before, consider the problem of extending a $k-1$ sequence (dense unit) to a k sequence. Let a $k-1$ sequence be formed with number i in the i^{th} position, $\{1, 2, \ldots, (k-1)\}$. We know from the analysis of MAFIA that we can form ${}^{d-(k-1)}C_2$ CDUs with this $(k-1)$ sequence. Now consider the $k-1$ subsequence $\{1, 2, \ldots, k\}$, where the $(k-1)^{th}$ position contains the integer k. It can be easily seen that we have $(d-k)$ sequences each of length k which have the given subsequence in common. Thus the set of $(d-k)$ sequences can be combined among themselves to give a total number of ${}^{d-k}C_2$ CDUs. In an analogous manner when we change the integer present in the $(k-1)^{th}$ position from $k-1$ to d the total number of CDUs formed is

$$ {}^{d-(k-1)}C_2 + {}^{d-k}C_2 + \ldots + {}^{2}C_2 \tag{2} $$

We shall now change the integer present in the $(k-2)^{nd}$ position and perform a similar analysis. If the integer $k-1$ occupies the $(k-2)^{nd}$ position and we change the the integer present in the $k-1$ position from k to d we can form a total of

$$ {}^{d-k}C_2 + {}^{d-(k+1)}C_2 + \ldots + {}^{2}C_2 \tag{3} $$

CDUs as explained in equation 2. Similarly, if k occupies the $(k-2)^{nd}$ position we can form an additional

$$ {}^{d-(k+1)}C_2 + {}^{d-(k+2)}C_2 + \ldots + {}^{2}C_2 \tag{4} $$

CDUs. By varying the integer in the $(k-2)^{nd}$ position we can form a total of

$$ \{{}^{d-k}C_2 + {}^{d-(k+1)}C_2 + \ldots + {}^{2}C_2\} + \{{}^{d-(k+1)}C_2 + \ldots + {}^{2}C_2\} + \ldots + {}^{2}C_2 \tag{5} $$

CDUs. The above equation is the summation of results in equation 3, 4 and the like by varying the integer in the $(k-2)^{nd}$ position from $k-1$ to $d-2$. In a similar manner we change the integer present in each position, i, of a $k-1$ sequence and by an enumerative technique find that N_{CLIQUE} is given by

$$
\begin{aligned}
&= [{}^{d-(k-1)}C_2 + {}^{d-k}C_2 + \ldots + {}^{2}C_2] + \\
&\quad [\{{}^{d-k}C_2 + {}^{d-(k+1)}C_2 + \ldots + {}^{2}C_2\} + \{{}^{d-(k+1)}C_2 + \ldots + {}^{2}C_2\} + \ldots + \{{}^{2}C_2\}] + \\
&\quad \vdots
\end{aligned}
$$

which is the summation of the results obtained in the above equations 2-5.

Observing that $\sum_{i=2}^{n} {}^{i}C_2 = \frac{n(n^2-1)}{6}$, we have

$$ N_{\text{CLIQUE}} = \sum_{n_{k-3}=2}^{n-(k-2)} \cdots \sum_{n_2=2}^{n_3-3} \sum_{n_1=2}^{n_2-2} \sum_{i=2}^{n_1-1} \frac{i(i^2-1)}{6} \tag{6} $$

where, $n_1, n_2, \ldots, n_{k-3}$ are all equal to $d-(k-1)$ in the final substitution.

Given a value for k and d we can evaluate the value N_{CLIQUE} using a tool like *Mathematica*. When the dimension of the data set, d, is 10 and k is 5 we have N_{CLIQUE} to be 210 and N_{MAFIA} to be 3150. Thus for 10-D data sets with complete subspace coverage in all the 5 dimensional subspaces MAFIA explores 15 times more number of subspaces than [7] and thus covers all possible subspaces to discover embedded clusters. Although in real life datasets complete subspace coverage is rare, clustering algorithms need to explore them all for correctness and completeness.

Figure 4 illustrates the process of building CDUs in dimension 3 for a data set in 10 dimensions. Each CDU and, similarly a dense unit, in the d^{th}-dimension is completely specified by the d dimensions of the unit and their corresponding d bin indices. The vertical dashed lines in the figure separate the CDUs and the dense units from other CDUs and dense units. Let the number of dense units be denoted by Ndu and the number of CDUs by $Ncdu$. It is easy to see that each dense unit needs to be examined with every other dense unit to form CDUs and thus results in an $O(Ndu^2)$ algorithm. Dense units which could not be combined with any other dense unit will be registered as a potential cluster region. It can be seen from Figure 4 that the process of CDU generation may lead to identical CDUs being formed. The repeat CDUs are eliminated by comparing each CDU with every other CDU, resulting in an $O(Ncdu^2)$ complexity filtering algorithm.

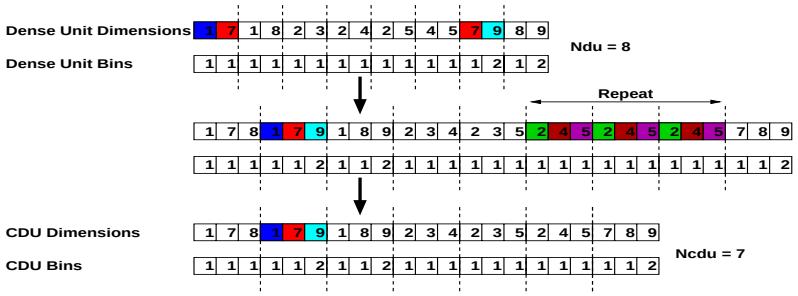


Figure 4. *Building Candidate Dense Units. (Current Dimension, k = 3)*

4.2 Un-supervised Clustering Algorithm

Clusters which are a proper subset of a higher dimension cluster are eliminated and only unique clusters of the highest dimensionality are presented to the end user increasing the usability of the algorithm. MAFIA requires a single user input (*cluster dominance factor* α) which specifies the strength of the clusters to be discovered. α indicates the magnitude of deviation of the histogram values from that of uniform distribution. A value of α greater than 1.5 has been accepted to be sufficient deviation to be considered significant in the field of statistics and data mining [15]. Discovering clusters with higher values of α yields clusters in the data set which are more dominant than the others in terms of the number of data points contained in the cluster. Hence, choosing a suitable value of α is straightforward. Our algorithm can report clusters for any given range of *alpha* in a single run. The parameter

β used in merging neighboring windows is used in the process of finding adaptive grids. A low value of β results in merging adjacent bins having nearly identical histogram values. As histogram values of adjacent bins are rarely low values of β result in large number of bins resulting in a greater computation time but high quality clusters. High values of β merge all bins in a given dimension and yielding poor quality clusters. Our algorithm is not very sensitive to the value of β. Clusters of high quality are discovered efficiently by MAFIA when too low or too high values of β are avoided. A value of β in the range of 25% to 75% has worked well in our experiments.

4.3 Analysis

Let k represent the highest dimensionality of any dense unit in the data set. The running time of the algorithm is exponential in k as all its projections, $O(2^k)$ subset dimensions, are also dense. However, with the use of adaptive grids the number of CDUs is greatly reduced and thus enables MAFIA to scale gracefully with the data dimensionality and size. Let N be the total number of records, B the number of records that fit in memory buffer and let γ be the I/O access time for a block of B records from the disk. The complexity of the algorithm is then $O(c^k)$, where c is a constant. The total I/O time is $O(\frac{N}{B}k\gamma)$ as data is read in chunks of B records in each of k passes of the algorithm. The total time complexity of the algorithm is then $O(c^k + \frac{N}{B}k\gamma)$.

5 Performance Evaluation

We present performance results of MAFIA in terms of scalability of the algorithm with database size, data dimensionality and cluster dimensionality. Further, we compare CLIQUE with MAFIA, in terms of quality of the clustering and computation time. We also provide a sensitivity analysis of MAFIA with respect to the parameters α and β. Finally we shall present results on real world data sets. The results reported are extracted on a machine with 256 MB of main memory and a 400 MHz Pentium II processor running a Linux.

5.1 Data Sets

We created a data generator to produce large synthetic data sets. The extent of a cluster in each of its dimensions can be controlled and so also the max and min of each dimension enabling generation of arbitrary shaped clusters. All dimensions are scaled to lie in the range $[0, \ldots, 100]$. Data points are generated such that each unit cube, part of the user defined cluster, in this scaled space contains *at least* one point. The data so generated is scaled back appropriately into the user specified attribute ranges. This method, as against randomly populating the user defined cluster region as used in [7], ensures cluster generation exactly as defined by the user and helps to validate the results. For the remaining attributes we select a value at random from a uniform distribution over the entire range of the attribute. An additional 10% noise records is added to the data set. Values for all the attributes in these noise records are independently drawn at random over the entire range of the attribute. Also, user specified cluster definition is permuted to ensure no dependency on the order of input records.

5.2 Experimental Results

Effect of using Adaptive Grids

Table 1 shows the speedup obtained over CLIQUE on 3 different data sets given. We set the threshold percentages for the bins in various dimensions depending on the bin size based on Algorithm 1. However, while running CLIQUE we set the threshold τ to the same uniform high value as given in Table 1 in all dimensions so that it could discard a larger number of candidate dense units in every pass over the data. It is easy to see that higher the value of τ, larger will be the number of candidate dense units discarded as non-dense units and thus CLIQUE would discover clusters faster. Each dimension is divided into 10 bins for the results reported for CLIQUE. While running [7] we do not prune the subspaces as it could potentially reduce the quality of clusters obtained. Table 1 shows that MAFIA performs 66.82 to 133.8 times better than CLIQUE for these data sets. The results presented for CLIQUE are by using the modified candidate dense unit generation process along with the usage of an user specified fixed grid size and threshold. The modified algorithm results in exploring a much larger number of subspaces. Higher speedups have been observed for data sets with much larger dimensionality and database size. This is due to the very nature of our algorithm which decides a minimum set of bins in a dimension based on its *interestingness* as observed from the data histogram in every dimension. This results in a set of candidate dense units much lower than the one obtained by equal number of bins in all dimensions. The latter results in an explosion of candidate dense units in a very high dimension space.

Table 1. *Speedups obtained over CLIQUE using Adaptive Grids*

	Data Dim	Data Size	Cluster Dim	Number of Clusters	Threshold %	Speedup
Data Set A	20	8.85 Million	5	5	2	133.8
Data Set B	30	0.65 Million	8	1	7	66.82
Data Set C	10	0.25 Million	5	3	2	82.13

Subspace overlap of clusters heavily affects the computation time. Candidate dense units explodes with increase in the number of distinct cluster dimensions as we consider more combinations of distinct dimensions. This effect would be even more pronounced in CLIQUE as it treats all dimensions equally without exploiting the data distribution in each dimension. A direct comparison to the results reported in [7] is not possible since they do not mention the subspace overlap of the clusters.

Effect of Adaptive Grids on Computational Complexity

We generated a data set containing a *single* 7 dimensional cluster in a 10 dimensional data space. The data set contained 5.4 million records. In this experiment we ran MAFIA and compared it with our modified implementation of [7]. This modification of the algorithm drastically increases the search space for finding the embedded clusters. Table 2 shows the number of CDUs (Ncdu) and the number of dense units (Ndu) generated in this experiment. Results presented for [7] are with 10 uniform sized bins in each dimension and a threshold percentage of 1% for all dimensions. MAFIA discovered correctly the single cluster embedded. However,

CLIQUE discovered 75 unique clusters each of dimension 6 and 546 unique clusters each of dimension 7. Most of these clusters contained at least one cluster dimension which was not part of the original defined cluster. Since [7] treats all dimensions of the data set in the same way by forming uniform sized grids, its computation time grows drastically. However, MAFIA exploits the data distribution in each dimension by forming adaptive grids and thus greatly reduces the computation time by forming as few bins as required in each dimension. This results in very few candidate dense units being generated as seen in Table 2. In this experiment, MAFIA took just 691 seconds while the modified implementation of CLIQUE took 79162 seconds resulting in a factor of 114.56 speedup.

Table 2. *Candidate Dense Units generated by MAFIA and CLIQUE*

Dimension		2	3	4	5	6	7	8
MAFIA	Ncdu	21	35	35	21	7	1	0
	Ndu	21	35	35	21	7	1	0
CLIQUE	Ncdu	2313	5739	19215	38484	42836	24804	5820
	Ndu	535	1572	3337	3870	2312	546	0

Quality of Results

We compare the quality of the results obtained by MAFIA with those of CLIQUE, shown in Table 3. The results are for a relatively small data set with $400,000$ records in 10 dimensions with 2 clusters each in a different 4 dimensional subspace. In the first case we set the number of bins to be 10 in every dimension and also set a threshold of 1% uniformly in all dimensions (as implemented in CLIQUE). In the second case we set arbitrary number of bins in each dimension (with a minimum of 5 bins to a maximum of 20 bins per dimension). The threshold in each dimension is set to 1%. In the first case CLIQUE reported the correct dimensions of the 2 clusters, however, it detected the 2 clusters only partially and large parts of the clusters were thrown away as outliers. In the second run, with a variable number of bins in each dimension, it completely failed to detect one of the clusters and, as before, the single cluster was partially detected. This is due to the inherent nature of CLIQUE which uses fixed discretization of the dimensions and hence results in a loss in the quality of the cluster obtained. When we ran MAFIA on the same data set, both the clusters and the cluster boundaries in each dimension were accurately reported. On a data set with 1.6 million records in a 10-D space containing 2 clusters, one in a 4 dimensional subspace and the other in a 6 dimensional subspace we performed a similar experiment. CLIQUE partially reported the two clusters with a mismatch of 1 dimension in both the reported clusters. Also, the boundaries reported by CLIQUE were a crude approximation of the embedded clusters. However, MAFIA reported both the clusters with their exact cluster dimensions and cluster boundaries as was defined for the data set.

Table 3. *Comparison of the quality of results obtained by MAFIA and CLIQUE*

	Cluster Dimensions	Clusters Discovered
CLIQUE (fixed 10 bins)	{1,7,8,9},{2,3,4,5}	{1,7,8,9},{2,3,4,5}
CLIQUE (variable bins)	{1,7,8,9},{2,3,4,5}	{2,3,4,5}
MAFIA	{1,7,8,9},{2,3,4,5}	{1,7,8,9},{2,3,4,5}

Sensitivity Analysis for α and β

In this experiment we show that MAFIA is not sensitive to the variation of β and also demonstrate ease of use of the cluster dominance factor α. The 20 dimensional data set used in this experiment contains 5 clusters in 5 different 5-dimensional subspaces and has 8.85 million records. The parameter α is a measure of the amount of deviation from uniform distribution. Higher the value of α more interesting and dominant is the cluster discovered. We fix the value of β at 35% for this run. As we gradually increase the value of α from 1.5 to 4 in steps of 0.5, we observed that the number of clusters discovered decreases as α increases. This is an interesting feature in that it enables us to discover clusters in the data set which have a required amount of deviation from that of uniform distribution of data. Thus selecting a suitable value or a range for α is very easy.

β controls the number of bins formed in each dimension. Lower values of β result in merging adjacent bins which have nearly identical histogram values and higher values of β result in merging all the bins in a given dimension. Also, very low value of β could yield high quality clusters at the cost of a great computation time. We fix the value of α to be 1.5 for this run. For a value of 1% β we observed 6 clusters and when β was increased from 10% to 75% in steps of 5% we always discovered 5 clusters. For an extreme value of 150% β we discovered 3 clusters. Thus values of β between 25% and 75% work extremely well MAFIA. This behavior with respect to both *alpha* and *beta* has been observed consistently in all the data sets used in this paper.

Scalability with Database Size

Figure 5 shows the results for scalability with the database size for a 20 dimensional data with the number of records ranging from 1.45 million to 11.8 million. There were 5 clusters embedded in 5 different 5-dimensional subspaces. The thresholds for different bins were determined automatically by Algorithm 1. The time spent in cluster detection almost shows a direct linear relationship with the database size. The linear behavior is because the number of passes over the database depends only on the dimensionality of the embedded cluster. An increase in the size of the database just means that more data has to be scanned on every pass over the database while finding the dense units resulting in a linear increase in time.

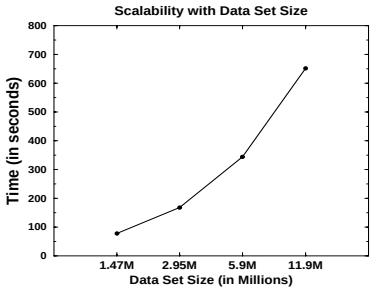
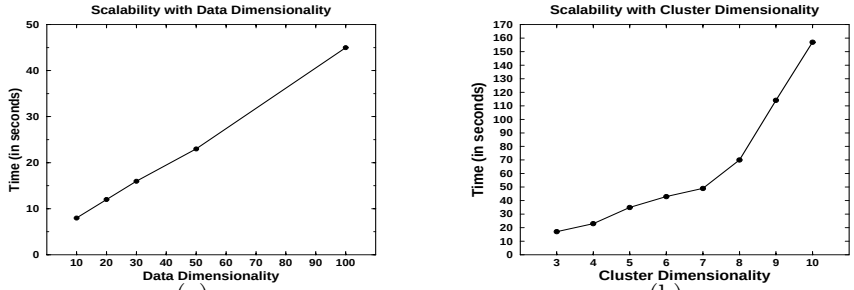


Figure 5. *With increasing database size:* 20*D data, five 5-D clusters*

Scalability with Dimensionality of Data and Cluster Dimensionality

In Figure 6(a) we see that MAFIA scales very well with the increase in data dimension. The results shown are on a data set of 250,000 records with 3 clusters each in a five dimensional subspace, with a total of 9 distinct dimensions. The linear behavior is due to the fact that our algorithm makes use of data distribution in every dimension and only depends on the number of distinct cluster dimensions. Additional non cluster dimensions only result in the algorithm making a pass over more data in every pass over the data set. CLIQUE not only depends on distinct cluster dimensions but also on the data dimensionality. Hence it exhibits a quadratic behavior with respect to the data dimensionality as reported in [7].

Figure 6(b) shows the scalability observed with increasing cluster dimensionality in MAFIA. The results reported are for a 30-dimensional data set with 650,000 records containing 1 cluster. Results show that the time increase with cluster dimensionality reflects the time complexity of the algorithm, which is exponential in the number of distinct cluster dimensions. The time taken to discover clusters increases greatly with the increase in the cluster dimensionality. This is due to the fact that higher cluster dimensionality results in a large subspace coverage and a large number of candidate dense units are formed in the subspace covered by the cluster. Thus each pass on the data needs to populate a large number of candidate dense units and increase in cluster dimensionality also increases the number of passes over the data set.



(a) (b)

Figure 6. *With increasing (a) Data Dimensionality (b) Cluster Dimensionality*

Real Data Sets

We applied MAFIA on three real world data sets to discover embedded clusters in different subspaces. The data sets used are of varying dimensionality and size.

1. **One Day Ahead Prediction of DAX** The data set is a one day ahead prediction of the German Stock index (DAX, Deutscher Aktien Index) based on twelve input time series which includes different stock indices, bond indices and inflation indicators. Detailed explanation of the DAX data set and description of the inputs can be found in [16]. The results reported are for the data set in 22 dimensions with 2757 records and cluster dominance factor 2. Clusters reported in Table 4 are all unique and took just 8 seconds to discover.

Table 4. *Clusters Discovered in the DAX Data Set*

Cluster Dimension	3	4	5	6
Number of Clusters Discovered	161	134	104	24

2. **Ionosphere Data** We applied MAFIA to the radar data that was collected by a system in Goose Bay, Labrador [17]. The data set is of 34 dimensions with 351 records we set the cluster dominance factor as 2. We discovered 158 unique clusters in 3-D subspaces and 32 unique clusters in 4-D subspaces. However, when we increased the cluster dominance factor to 3 we discovered one single cluster in a 3-D subspace. PROCLUS [6] has reported two clusters one each in 31 and 33 dimensions for this data set. However, we believe that this could be in part due to an incorrect value of l, the average cluster dimensionality, chosen by the user. Further, [6] also requires the user to specify k, the number of clusters in the data set which cannot be known a-priori for real data sets.

3. **Each-Movie Recommendation Data** We applied MAFIA to a massive data set which contained movie ratings collected by DEC Systems Research Center [18] over a period of 18 months. During this period 72916 users entered a total of $2,811,983$ (≈ 2.8 Million) numeric ratings for 1628 different movies (films and videos). Each rating is characterized by four numbers containing information about the user-id, movie-id, a score $(0-1)$ and a weight $(0-1)$. In this 4-D data set we discovered 7 clusters all of dimension 2 in 28 seconds. Clusters discovered by MAFIA revealed interesting information about which set of movies were rated most by which set of users. The experiment reveals the scalability with respect to the data size of MAFIA on real data. Our implementation of CLIQUE took 151 seconds and found clusters in dimensions 2, 3 and 4. For this experiment using CLIQUE, we had 10 uniform sized bins in all dimensions with a uniform threshold of 1% in each dimension. However, when we ran CLIQUE with an arbitrary number of bins in each dimension we discovered a different set of clusters. Thus evaluating the results and finding the correct set of clusters would not be easy with CLIQUE.

6 Conclusions

In this paper we presented MAFIA, an efficient scalable algorithm for subspace clustering using a density and grid based approach with adaptive finite intervals. This performs two orders of magnitude better than CLIQUE and also improves the quality of clustering greatly. MAFIA requires a very intuitive single user input, which makes it a far more un-supervised data mining algorithm than any other existing subspace clustering algorithm. The scalable algorithm developed can handle massive data sets and makes it a viable algorithm for discovering clusters in high dimensional data for large scale data mining applications. Experimental evaluations on a variety of synthetic and real data sets, with varying dimensionality and database sizes, show the gains in performance and quality of clusters. The use of adaptive grids in MAFIA leads to large improvements over CLIQUE both in terms of computation time and also quality of clustering.

Bibliography

[1] D. Boley, M. Gini, R. Gross, E.H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Partitioning-based clustering for web document categorization. *Decision Support Systems Journal*, 1999.

[2] L.H. Ungar and D.P. Foster. Clustering methods for collaborative filtering. *AAAI Workshop on Recommendation Systems*, 1998.

[3] R.T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. *Proc. 20th Int. Conf. on Very large Data Bases, Santiago, Chile*, pages 144–155, 1994.

[4] M. Ester, H-P. Kriegel, J. Sander, and X. Xu. A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proc. of the 2nd International Conference on Knowledge Discovery in Databases and Data Mining*, 1996.

[5] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient data clustering method for very large databases. In *Proc. ACM SIGMOD International Conference on Management of Data*, 1996.

[6] C. Aggarwal, C. Procopiuc, J.L. Wolf, P.S. Yu, and J.S. Park. A Framework for Finding Projected Clusters in High Dimensional Spaces. In *Proc. ACM SIGMOD International Conference on Management of Data*, 1999.

[7] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. ACM SIGMOD International Conference on Management of Data*, 1998.

[8] P. Arabie and L.J Arabie. An Overview of Combinatorial Data Analysis. *Clustering and Classification, World Scientific Publication*, pages 5–63, 1996.

[9] R.S Michalski and R.E. Stepp. Learning from Observation: Conceptual Clustering. *Machine Learning: An Artificial Intelligence Approach*, I:331–363, 1983.

[10] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

[11] R.T. Ng and J. Han. Efficient and Effective Clustering Methods for Spatial Data Mining. In *Proc. 20th International Conference on Very Large Databases*, 1994.

[12] S. Guha, R. Rastogi, and K Shim. CURE: An Efficient Clustering Algorithm for Large Databases. In *Proc. ACM SIGMOD International Conference on Management of Data*, 1998.

[13] A.K. Jain and R.C Dubes. *Algorithms for Clustering Data*. Prentice-Hall Inc., 1988.

[14] C. Cheng, A.W. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999.

[15] Peter J. Bickel and Kjell A. Doksum. *Mathematical Statistics: Basic Ideas and Selected Topics*. Prentice Hall, 1976.

[16] A. S. Weigend and H. G. Zimmermann. Exploiting local relations as soft constraints to improve forecasting. *Journal of Computational Intelligence in Finance*, 6:14–23, 1998.

[17] C.L. Blake and C.J. Merz. UCI repository of machine learning databases. In *University of California, Irvine, Dept. of Information and Computer Sciences*, 1998. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[18] Paul McJones. Digital Equipment Corporation. In *Systems Research Center*, 1997. `http://www.research.digital.com/SRC/`.