Grid-Clustering: An Efficient Hierarchical Clustering Method for Very Large Data Sets

Erich Schikuta
Institute of Applied Computer Science and Information Systems
University of Vienna
schiki@ifs.univie.ac.at

Abstract

Clustering is a common technique for the analysis of large images. In this paper a new approach to hierarchical clustering of very large data sets is presented. The GRIDCLUS algorithm uses a multidimensional grid data structure to organize the value space surrounding the pattern values, rather than to organize the patterns themselves. The patterns are grouped into blocks and clustered with respect to the blocks by a topological neighbor search algorithm. The runtime behavior of the algorithm outperforms all conventional hierarchical methods. A comparison of execution times to those of other commonly used clustering algorithms, and a heuristic runtime analysis are presented.

1. Introduction

Clustering methods are extremely important for explorative data analysis, which is an important approach for the analysis of images. Previously presented algorithms can be divided into hierarchical algorithms, e.g. single-linkage, complete-linkage, etc. and partitional algorithms, e.g. K-MEANS, ISODATA, etc. (see [3]). All of these methods suffer from specific drawbacks, when handling large numbers of patterns. The hierarchical methods provide structural information, as dendrograms, but are suitable only for a small number of patterns. With growing numbers the computational expense increases, due to the calculation of a dissimilarity matrix, where each pattern is compared to all others. The partitional methods are to some less resource consuming. However they lack methodical freedom because of the prerequisite of a "good guess" of structural information, e.g. the numbers and the positions of the initial cluster centers. If the choice of the initial clustering is not appropriate, the partitional methods also become very calculation intensive in computing new cluster centers.

The hierarchical Grid-Clustering algorithm proposed in this paper clusters the patterns according to the structure of the embedding space. It combines and refines the ideas presented by Warnekar et al. [7] and Broder [2]. We introduce a density index to compare and evaluate the possible pattern cells to find cluster centers and combine neighbor cells. Further, we implement a new grid structure to organize the value space surrounding the patterns, which overcomes the problems of the kd-B-tree. This algorithm achieves an appealing performance gain in comparison to all conventional algorithms.

2. Grid-Clustering

2.1. Basic outline

Conventional cluster algorithms calculate a distance based on a dissimilarity metric (e.g. Euclidean distance, etc.) between patterns or cluster centers. The patterns are clustered accordingly to the resulting dissimilarity index.

The Grid-Clustering algorithm presented here uses the idea of Warnekar (1979) to organize the value space containing the patterns. For that reason we use a variation of the multidimensional data structure of the Grid File, which we call *Grid Structure*. The patterns are treated as points in a d-dimensional value space and are randomly inserted into the Grid Structure. These points are stored according to their pattern values, preserving the topological distribution. The Grid Structure partitions the value space and administrates the points by a set of surrounding rectangular shaped blocks.

A *block* is a d-dimensional hyper rectangle (i.e. a rectangular shaped cube) containing up to a maximum of bs patterns (bs = block size). Let $X = \{x_1, x_2, \dots x_n\}$ be the set of n patterns. x_i is the i-th pattern consisting of a tuple of describing features $(a_{i1}, a_{i2}, \dots a_{id})$, where d is the number of dimensions. The following properties are satisfied (ϕ is the empty set)

$$\text{for all } x_i, x_i \in B_j$$
$$B_j \cap B_k = \phi, \text{ if } j \neq k \quad \text{and}$$
$$B_j \neq \phi \quad \text{and}$$
$$\cup B_j = X$$

The algorithm clusters the blocks B_i (and hence the patterns X) into a nested sequence of nonempty and disjoint clusterings, where $(C_{u1}, C_{u2}, \dots C_{uW_u})$ is the u-th clustering and W_i is the number of clusters of the i-th clustering. Initially (0-th clustering) each block is a cluster, i.e. $C_{0j} = B_j$, j = 1, ... b and $W_0 = b$. The blocks can be seen as the initialization of cluster centers in a preclustering phase. The cardinality of these centers is dependent on the block size and is defined by $1 < p_B < bs$, where p_B is the number of patterns contained in block B.

The proposed Grid-Clustering algorithm uses the block information of the Grid Structure and clusters the patterns according to their surrounding blocks.

The following Figure 1 shows the value space partition after the sequential insertion of 1000 2-dimensional patterns with 3 major clusters into a Grid Structure. It is easy to recognize how the rectangular blocks adapt to the distribution of the patterns.

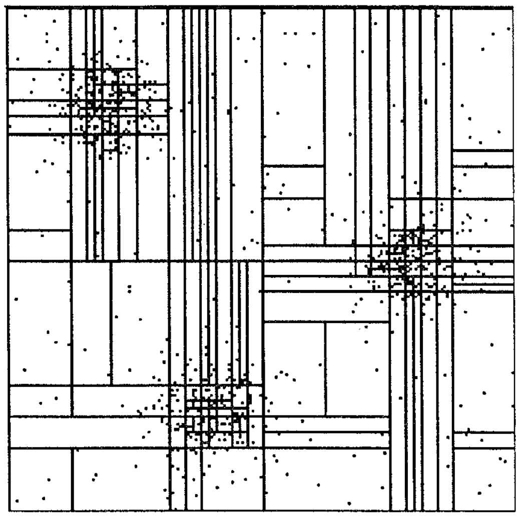


Figure 1: block structure for 1000 patters, 3 clusters

The algorithm calculates the *density index* of each block via the numbers of patterns and the spatial volume of the block. The spatial volume V_B of a block B is the Cartesian product of the extents e of block B in each dimension, i.e.

$$V_B = \prod_i e_{Bi}, \quad i = 1, \ldots d$$

The density index D_B of block B is defined as the ratio of the actual number of patterns p_B contained in block B to the spatial volume V_B of B, i.e.

$$D_B = \frac{p_B}{V_B}$$

The blocks are sorted accordingly to their density indices. The result is an ordered sequence $<B_{p(i)}>$, where p(i) denotes a permutation of the index i defining the sorted order of the blocks B.

Blocks which have the same density index are called *ties*. For example the blocks in Figure 2 are ties.

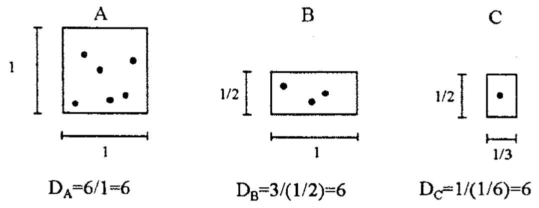


Figure 2: tie blocks

Blocks with the highest density index (obviously with highest pattern correlation) become clustering centers. The remaining blocks are then clustered iteratively in order of their density index, thereby building new cluster centers or merging with existing clusters.

Only blocks adjacent to a cluster ("neighbor") can be merged. A neighbor search is conducted, starting at the cluster center and inspecting adjacent blocks. If a neighbor block is found, the search proceeds recursively with this block. This is similar to the traversal of a graph to find the spanning tree [1], where the blocks represent the nodes. An edge between two nodes exists if these blocks are adjacent. The traversal can be done by a depth-first-search (DFS) algorithm which inspects only blocks with density indices smaller than or equal to the value of the iteratively processed block.

2.2. The Grid structure

The patterns are inserted into an adapted Grid File, the Grid Structure, which partitions the pattern space into blocks. The Grid File is a multidimensional data structure, which adapts gracefully to the distribution of patterns X in the value space of the domains of X. The Grid Structure is a main memory data structure. It lacks the external disk storage support and the data manipulation facilities of the original Grid File.

The Grid Structure consists of d scales (one for each dimension), the grid directory (a d-dimensional array) and b data blocks (Figure 3).

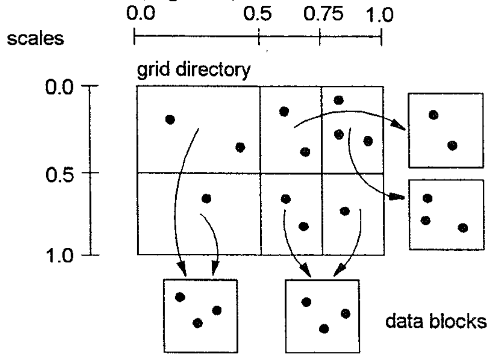


Figure 3: 2-dimensional Grid Structure

Each scale is a 1-dimensional array. A value of this array represents a (d-1)-dimensional hyperrectangle, which partitions the value space. The grid directory is a d-dimensional dynamic array and represents the grid partition produced by the d scales. The data blocks contain the stored patterns. Each element of the grid directory refers to a data block. It is possible that two or more directory elements reference the same data block. The value space defined by the union of the directory elements referencing the data block i is called block region V_{Bi}. Each block region is always shaped like a d-dimensional rectangular box.

For an exact description, we refer to [4].

3. Algorithm

3.1. The Grid-Clustering algorithm

As described in section 2 the proposed Grid-Clustering algorithm consists of 5 main parts.

- Insertion of patterns into the Grid Structure
- Calculation of density indices
- Sorting of the blocks resp. density indices
- Identification of cluster centers
- Traversal of neighbor blocks

3.2. Algorithm GRIDCLUS

GRIDCLUS consists of a main module, which iteratively processes all blocks, and a recursive procedure NEIGHBOR-SEARCH, which assigns the blocks to existing clusters.

The number of the current run is stored in u. After completion of a run, W[u] stores the number of clusters of run u. C[u, v] is a set-valued variable containing the clustered blocks of run u and cluster v.

In accordance with the definition of a block, each block automatically represents a unique cluster This situation is not handled directly by the algorithm, it can be viewed as run 0 with W[u] = b and C containing b clusters with one block.

```
Algorithm GRIDCLUS
Initialization ( u := 0, W[] := {}, C[][] := {} )
Create the Grid Structure
Calculate the block density indices $D_{Bi}$
Generate a sorted block sequence S = <$B_1$.. $B_2$, ... $B_b$>
Mark all blocks 'not active' and 'not clustered'
while a 'not active' block exists do
    u := u + 1
    Mark first $B_i$ .. $B_j$ with equal density index 'active'
    for each 'not clustered' block $B_{k'}$ := $B_i$ .. $B_j$ do
        create a new cluster set C[u]
        W[u] := W[u] + 1
        C[u, W[u]] $\leftarrow$ $B_{k'}$
        mark block $B_{k'}$ 'clustered'
        NEIGHBOR-SEARCH($B_{k'}$, C[u, W[u]])
    endfor
    for each 'not active' block $B_l$ do
        W[u] := W[u] + 1
        C[u, W[u]] $\leftarrow$ $B_l$
    endfor
    Mark all blocks 'not clustered'
endwhile
```

```
Procedure NEIGHBOR-SEARCH(B, C)
for each 'active' and 'not clustered' neighbor $B_n$ of B do
    C $\leftarrow$ $B_n$
    mark block Bn clustered
    NEIGHBOR-SEARCH($B_n$, C)
endfor
```

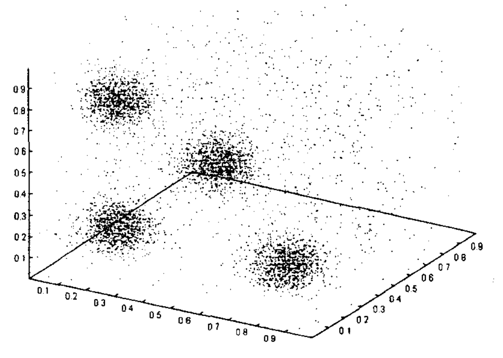
4. Experimental results

Explorative data analysis of real-life data must often deal with data sets which are very large (number of patterns > 1000) and/or high dimensional (number of attributes > 3). Conventional hierarchical algorithms often fail to reach a solution due to large execution time and/or memory space exhaustion. Partitional methods require a good initial "guess" of the structural data information.

GRIDCLUS provides an efficient solution to these problems.

The following example is hard or impossible to solve with conventional methods. We try to cluster a test data set of 5000 3-dimensional patterns, where 80 % are clustered into 4 groups. The remaining 20 % represent noise and are evenly distributed over the 3-dimensional value. The cluster centers are arranged such that the clusters are hidden in projected 2-dimensional views of the data set. This data set is too large to cluster with hierarchical methods. For a partitional approach, structural information, like the number of clusters and initial cluster centers, must be provided. In practice, this information would be supplied by the analyst by visual inspection. In current available systems (e.g. SPSS or SAS), only 2-dimensional projected views are supported. Analyzing data sets, which have multiple descriptive attributes, or a "hidden" cluster distribution (clusters, which can not be distinguished in projected views because of overlapping positions) is difficult and prone to error.

Figure 4 contains a 3-dimensional spatial picture of the pattern distribution and its respective 2-dimensional projections. In this example only 3 clusters are visible in the projections because of overlapping pattern concentrations. Hence the analytical view can produce an erroneous guess about the actual distribution of data, which can in turn lead to incorrect initial values for a partitional clustering method. In cases such as this, GRIDCLUS shows its advantages. For this data set, GRIDCLUS produces the dendrogram of Figure 5 in which the 4 clusters are clearly visible.



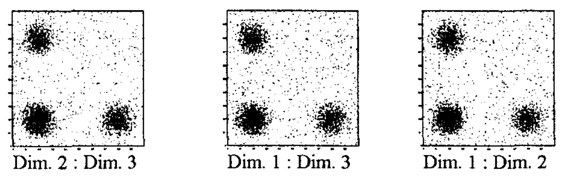


Figure 4: 5000 3-dimensional patterns, 4 clusters

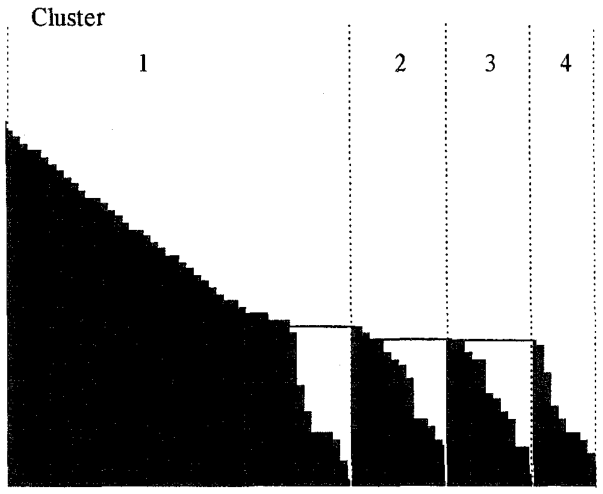


Figure 5: Dendrogram created by GRIDCLUS

Since Figure 5 is an original screen dump, therefore subject to low screen resolution the vertical dendrogram lines (5000) appear as a compact area.

The algorithm requires less than 3 minutes on a conventional personal computer with no initial information, to produce this result. The structural information of the data set is clearly and automatically recognized.

5. Analysis

We compare GRIDCLUS to several well known clustering algorithms and, present a heuristic algorithm analysis [6]. This is due to the fact that the Grid File has proved itself to be extremely difficult to analyze formally, results exist for only a few known data distributions [5].
The data sets of the test runs were analyzed using conventional methods of a commercial statistical package, the SPSS system. Since the personal computer (PC) version of the SPSS package was able to accomplish the task only for small data sets, we processed the larger sets with the SPSS installed on a mainframe. The PC system was equipped with a mathematical coprocessor, and the mainframe was an ES9000 running VM. The pure processor time for the completion of the applied algorithm was measured and the results are given in seconds.

The investigated algorithms were (see Figure 6):

- single linkage, SPSS-PC ("HM, PC")
- quick cluster, SPSS-PC ("PM, PC")
- single linkage, SPSS-mainframe ("HM, MF")
- single linkage, SPSS-mainframe, extrapolated ("HM, MF, ex.")
- GRIDCLUS, PC ("GC, PC")

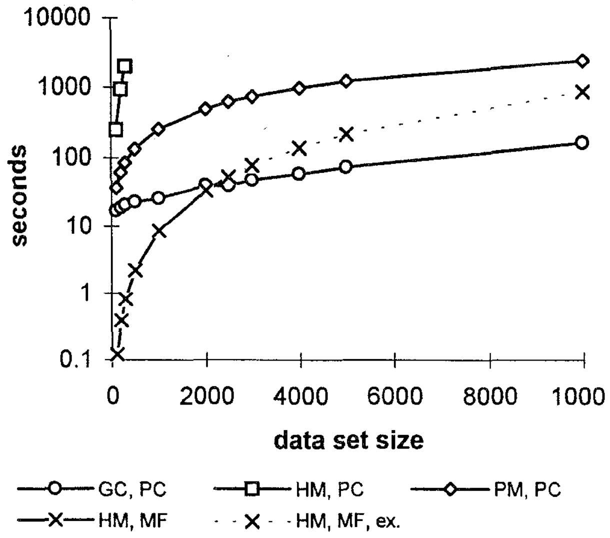


Figure 6: Comparison of the execution times

Different hierarchical methods provided by the SPSS system were checked. Execution times are given only for the single linkage algorithm, because the other hierarchical algorithms had a similar runtime behavior. As partial clustering method we used the quick cluster algorithm. The correct number of clusters was supplied as an input. Due to virtual memory exhaustion on the mainframe, the execution times of pattern numbers greater than 2000 were extrapolated. Note that the scale of the y-axis is logarithmic.

GRIDCLUS outperformed all other analysis methods by far. It performed better on a PC than the hierarchical methods of the SPSS system on the mainframe. Importantly it delivered results where the other algorithms failed because of runtime or memory exhaustion.

To explain the runtime behavior of GRIDCLUS, we must examine the different tasks of the algorithm in more detail. The following Figure 7 shows correlation of the runtime and memory characteristics of GRIDCLUS. The scale of the y-axis is logarithmic. The scale ticks represent different units (time and size) and are therefore not directly comparable. The chart shows the influence of the different algorithm characteristics on each other and helps to explain the runtime behavior. The execution time of the algorithm is divided into the time for Grid Structure creation ("GS creation") and for neighbor search clustering ("Clustering"). The corresponding numbers of Grid Structure blocks ("blocks"), directory entries ("dir. entries") and clustering iterations ("iterations", while-statement of GRIDCLUS, defines the levels of the dendrogram) for the GRIDCLUS test run depicted in Figure 7 are shown.

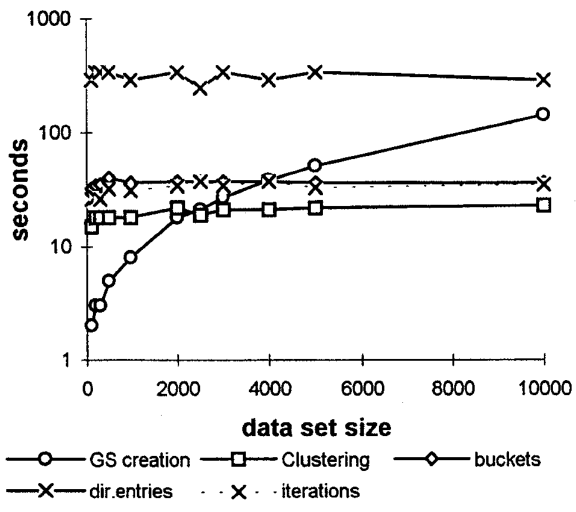


Figure 7: GRIDCLUS runtime and memory requirements

It shows that the Grid Structure creation of GRIDCLUS increases with larger data set size. This is obvious considering that the ratio of the blocksize to the data set size is constant. The number of blocks is dependent only on the blocksize ratio, and not on the data set size. This always results in the creation of the approximately same number of blocks and of directory entries for a given blocksize ratio. Therefore, the recursive clustering algorithm always handles the same number of blocks.

We can conclude that Grid Structure creation characterizes the runtime behavior of GRIDCLUS.

6. Conclusion

A new hierarchical clustering method, Grid-Clustering, has been proposed. Unlike conventional methods this method organizes the space surrounding the patterns and not the patterns themselves. To accomplish this task it uses a multidimensional grid data structure. The resulting block partitioning of the value space is clustered via a topological neighbor search. The Grid-Clustering method outperformed all conventional hierarchical and partitional methods. Additionally, it clusters data sets which were previously not tractable due to their size or their dimensionality, and it does so without any supplementary input information.

7. Acknowledgments

The author wishes to thank Heinz Hromada for the implementation of the algorithm and the interactive cluster analysis system. Additionally, the author thanks the Center for Research on Parallel Computation and Rice University for the computing facilities used to perform the performance tests. This work was partially supported by grant J0742-PHY of the Austrian FWF.

8. References

[1] Aho A.V., Hopcroft J.E. and Ullman J.D., The design *and analysis of computer algorithms*, Addison-Wesley, Reading, MA, 1974

[2] Broder A.J., Strategies for efficient incremental nearest neighbor search, *Pattern Recognition*, 23, 1/2, 171-178, 1990

[3] Dubes R., Jain A.K., Clustering methodologies in exploratory data analysis, *Advances in Computers* 19, Academia Press, 113-228, 1980

[4] Nievergelt J., Hinterberger H., Sevcik K.C., The grid file: an adaptable, symmetric multikey file structure, *ACM Transactions on Database Systems* 9, 38-71, 1984

[5] Regnier M., Analysis of grid file algorithms, *BIT* 25, 335-357, 1985

[6] Schikuta E., Grid-Clustering: An Efficient Clustering Method for Very Large Data Sets, Tech. Rep. No. TR-96201, Inst. Applied Computer Science and Information Systems, University of Vienna, Austria, 1996

[7] Warnekar C.S., Krishna G., A heuristic clustering algorithm using union of overlapping pattern-cells, *Pattern Recognition*, 11, 85-93, 1979