Fast Adaptive Clustering for Very Large Datasets

KUN-CHE LU, DON-LIN YANG, and JUNGPIN WU*
Department of Information Engineering and Computer Science
*Department of Statistics
Feng Chia University
No. 100, Wenhwa Rd., Taichung
TAIWAN 40724

Abstract: - In this paper, we propose an efficient and effective clustering method that requires to scan a dataset only once. The original dataset is transformed first and merged into a hyper image of controllable size. Unlike traditional methods, the dissimilarity measurement between objects is calculated once for all objects by using various image processing methodologies, such as morphological operations. Image connect component extraction is thereby used to extract clusters from the hyper image. The proposed method is easy to use for clustering data in way of fuzzy and hierarchical fashion readily under a single dataset scan. It is also efficient for incremental and dynamic clustering without additional scan of the original dataset. Experimental results show that the proposed method is robust and stable under various parameter settings such that it is more effective and useful than traditional clustering methods, especially for very large datasets.

Key-Words: - Clustering, Dynamic clustering, Fuzzy clustering, Large dataset, Morphology, Hyper image.

1 Introduction

In most database application systems, the record dimensionality is usually stationary; however, the number of records may grow outrageously. As the Internet becomes popular, the transaction size can grow exponentially. Therefore, many techniques are toward devising incremental or dynamic-update data processing algorithms to deal with the vast and increasing data size. Apparently, the demand of more scalable algorithms to handle large dataset than to handle large dimension is rather urgent.

Clustering is an unsupervised learning technique used in various applications of different domains. The goal of clustering is to partition data points into groups (clusters), furthermore maximizing the within group similarity and between group dissimilarity. Clustering technologies can be roughly categorized into partition based [6], hierarchical based [2], density based [4], and grid based [8] approaches. Partition based clustering algorithms like the well known K-means can converge fast, but it may be affected by the initial solution a lot. It is also limited to extract clusters with particular shapes, and the result number of clusters has to be predetermined. The iterative distance calculation is very computational expensive. Moreover, its result quality can be affected by outliers quite a lot.

Hierarchical based clustering algorithms can be further categorized as agglomerative (bottom-up) or divisive (top-down). In hierarchical clustering, users do not need to specify the cluster number in advance. However generating the complete dendogram is time consuming, especially for large data sets. And extracting particular shaped clusters is even more computational expensive.

Density based clustering algorithms can extract clusters with particular shape as long as their affiliates are density reachable. However, how to define a good distance measure is a hard problem. Although it can easily filter out outliers, using a single global distance measure may not well characterize local data distributions. Moreover, the distance calculation is time consuming and complex.

Grid based clustering algorithms first partition the data space into grids, then use a single exemplar to represent all the points within a grid to speed up the clustering process. It is difficult to extract shaped cluster either, and how to determine a good grid size affects the result quality and processing time.

Fuzzy clustering [1] usually produces a better result than crisp clustering, but it also requires much more time and space.

In this paper, we propose a new approach to solve the problems mentioned above. It possesses the advantages of those discussed approaches without their disadvantages. Shaped clusters can be extracted by using various image processing techniques in our new method. It can cluster data points into predefined number of groups or find an optimal partition by using a given energy function efficiently. It only needs to scan the original dataset once regardless the

number of data records. And the required memory space is measurable and manageable. It forgoes the use of traditional time consuming iterative distance comparison methods, and applies image processing methods to determine neighboring data points once for all. In addition, when the original dataset is changed, it only needs to scan the update dataset once to extract the renewed clusters. Our approach is easy to fuzzify, and more efficient and space saving than traditional fuzzy clustering algorithms. Experimental results show that the proposed method is much more useful and robust than other existing clustering methods.

2 Our Proposed Model

The general processing flow of our model is very similar to the image processing steps, except the processed image is obtained from merged dataset. The details of each step are described in the following sections.

2.1 Data Merge

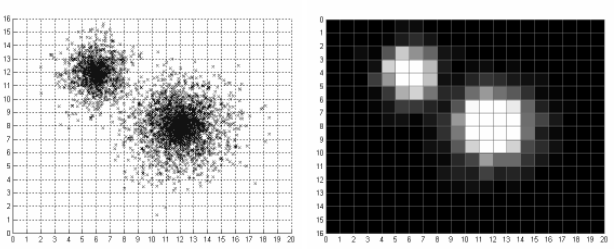
The original database $X = \{x_1,x_2,\ldots,x_n\}$ is first transformed and merged into a hyper image $I = \{p_1,p_2,\ldots,p_m\}$. The mapping relation Φ projecting a data point $x_n = (a_1,a_2,\ldots,a_k)$ to a hyper image pixel $p_m = (d_1,d_2,\ldots,d_k)$ is defined as:

$$\Phi \equiv x_n = (a_1,a_2,\ldots,a_k) \rightarrow p_m = (d_1,d_2,\ldots,d_k),$$
$$\text{satisfying } d_i = \left\lfloor \frac{a_i - X_{i.\min}}{q_i} \right\rfloor \text{ and}$$
$$q_i = \frac{(X_{i.\max} - X_{i.\min})}{bin_i} \text{ for } i = 1,\ldots,k. \quad (1)$$

The variable *bin* is the quantization level. This mapping relation is many to one because of the floor operation. We can also use ceiling operation instead. A weighted value $w(d_1,d_2,\ldots,d_k)$ is used to record how many data points are mapped into a hyper image pixel. I.e.

$$w(d_1,d_2,\ldots,d_k) = p(d_1,d_2,\ldots,d_k) = \#^{-}[x(a_1,a_2,\ldots,a_k) \rightarrow p(d_1,d_2,\ldots,d_k)], \text{ for } x \in X. \quad (2)$$

For visualization purpose, we can take this weighted value as the gray level of a given hyper image pixel. The pixel becomes brighter when this value is larger. Fig. 1 shows an illustrative example of this data merge process by using a 2D dataset. In this example, the *bin* number in X-axis is 20 and 16 in Y-axis. After obtaining such a hyper image, we can



(a) (b)

Fig. 1. Illustration of the data merge process: (a) example 2D dataset, (b) example dataset after merging to a hyper image.

apply various image processing methods on the image and extract clusters from it efficiently.

The hyper image requires $(\prod_{i=1}^{k} q_i)\cdot \log w$ bits of memory space. Since the needed space to finish a clustering process is calculable, a user could derive a suitable *bin* value based on the available memory. One advantage of the proposed approach is that the required hyper image size can be approximately independent to the data size. For example, if we use 32 bits for the weighted value, then one single pixel can accommodate $2^{32} =$ 4,294,967,296 data records. And one hyper image can approximately cope with $(bin^k \times 2^{32})/2$ data records, which can obviously be used to handle any huge databases in current applications. Moreover, as in image processing domains, we can use lossy representation to further reduce the hyper image size.

2.2 Image Pre-processing

After we have projected and merged the dataset into a hyper image, various image processing methods can be further exploited to enhance, refine, or tune the within clustered image objects. Followings are some of these methods:

1. Global or localized **thresholding** can filter outliers or noisy pixels efficiently and effectively. Statistically, under normal condition, one can use expectation value as threshold to automatically select natural populations.
2. **Smoothing, blur,** and/or **sharping** can alter the boundary and/or shape of clusters.
3. Global or localized image **enhancement** can enhance clustered objects or interested small clusters.
4. Morphological **dilation** can be used to fill holes of cluster or to merge clusters. It generally expands an image.
5. Morphological **erosion** can be used to split a cluster or eliminate noisy clusters. It generally shrinks an image.

Listed above are basic spatial image processing techniques, where some of them are generally supported by hardware nowadays. In addition, various image transform methods can also be used. For example, Wavelet transform can be used to reduce hyper image size, merge neighborhood objects, and reduce outliers efficiently [7]. Its effect is similar to applying zoom-out to an image by using a factor of two on each dimension. In contrast to wavecluster [7], our approach is much more generalized and useful such that it can produce more exquisite result with efficiency.

2.3 Image Segmentation

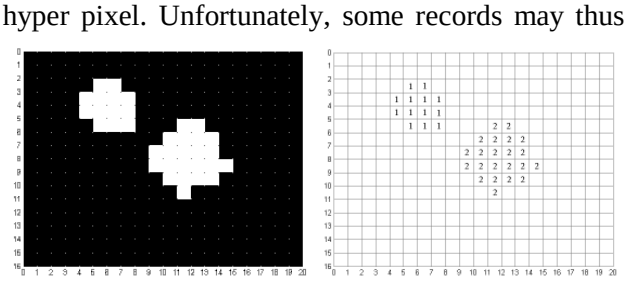
There are many existing image segmentation algorithms for extracting objects from image data. Using these techniques, we can easily find clusters from a hyper image. Here we consider only image connect component extraction technology since it is a general purpose and fundamental approach for image segmentation.

Basic setups regarding connect component extraction are the *connectivity* and the *similarity*. Usually it is applied after a thresholding. Since the connect component extraction deals with a binary image, the similarity measurement is thus reduced to a simpler and faster bit comparison problem. Incorporating with suitable image preprocessing methods with thresholding, one can apply connect component extraction process easier and more efficient without compromising its quality.

Fig. 2(a) shows a thresholded image of the dataset in Fig. 1 by using two times of expectation value as the threshold value. Fig. 2(b) shows its 8-connect component extraction result in a label form. The cluster of a record belonging to is then determined by the cluster label of the hyper image pixel being projected to.

2.4 Image Post-processing

As depicted above, the cluster of a record belonging to is obtained by the cluster label of its corresponding hyper pixel. Unfortunately, some records may thus map to background pixels and be clustered as noise data even though it is near an image cluster. If this phenomenon is undesirable, we can fuzzify the result to make compensation accordingly.



(a) (b)

Fig. 2. (a) Thresholding result of the dataset in Fig. 2, (b) connect component extraction result of the image in (a).

The most efficient and easy method is the image blur operation. An illustrative example was shown in the experiment section 3.3. Since the blurred region between clusters may be superimposed, the blur operation is applied separately on each cluster channel of the hyper image. Each blurred image can be viewed as the membership value of a result cluster. By deriving the maximal membership value across different clusters, a hyper image pixel close to an object will thus be assigned to it.

We know the membership value is discrete, say 0 to 255, and the fuzzifying process is applied only once on the final result. Therefore, this kind of fuzzifying is more efficient and space saving than the traditional fuzzy clustering. It can also be done through hardware easily.

In addition, in some applications we may require all the records be clustered without discarding any record, as in K-means based approaches. One applicable way to achieve this is using the k-nearest neighbor (k-NN) method to determine each background pixel's nearest objects. Various cluster distance measures can be used for the distance measurement, such as single link, complete link [5].

2.5 Energy Evaluation

An energy function can be used to determine the optimal cluster number, cluster shape, cluster size, and the like. Various index for cluster validity can be found in many literatures. By repeating the steps from Sections 2.2 to 2.4, users can use distinct image processing setups on the hyper image and search for an optimal clustering based on a given energy function or building a dendogram as in the hierarchical clustering methods. For instance, users can gradually apply a dilation or zoom-out operation on a hyper image to merge neighborhood clusters, and building a dendogram to extract clusters with a user-specified cluster number, or determine an optimal one automatically.

An energy function can also be used to determine optimal setup and image processing arrangements for a hyper image. Besides, the determined image processing operations can be combined or superimposed in one pass in the further clustering processes.

2.6 Dynamic Clustering

The generated hyper image can be stored for consequent use. When the original dataset is updated, we can easily make changes of the corresponding

hyper image pixel via adding or deleting its intensity weight value accordingly. No scan of the original database is required. In addition, various image compression techniques can be used to store the hyper image.

For the purpose of online clustering, the hyper image can be maintained in the memory. By applying the predetermined image processing setups on the up-to-date hyper image, one can extract clusters on the fly.

3 Experiments

In our experiments, we verify the performance and accuracy of the proposed method by using synthetic datasets and phantom MRI image datasets, respectively. We use MATLAB for programming. The performance of MATLAB may not be comparable with C or C++, but it provides an extensive set of functionality on array manipulation and image processing aspects. Our programs were ran on a PC with 2.53G GHz Pentium(R) 4 CPU and 2GB RAM.

3.1 Synthetic Dataset Experiments

The synthetic datasets were generated by using a normal distribution based on the criteria of Table 1. The standard deviation of the data points in each cluster is 1.

Table 1. Parameters of the synthetic datasets

Parameter code	Description
Transaction number: $\underline{T}$	Number of transactions in the dataset.
Dimensionality: $\underline{D}$	Dimensionality of the dataset.
Cluster number: $\underline{C}$	The underlying cluster number in the dataset.
Manners to allot record number to a cluster: $\underline{N}$	1: Normal distribution; 0: Uniform distribution, each underlying cluster contains (T/C) records.
Range: $\underline{R}$	Range of entire generated cluster centers.

First, we performed the scalability experiments under various data sizes from 10^4 to 10^7, with the dimensionality from 2 to 3. The *bin* value in these experiments was set to 100 for all dimensions. The image preprocessing steps include one thresholding, one erosion, and one dilation. We use 8-connect for the 2D dataset to determine the connected components and 26-connect for the 3D dataset. Since the dataset is scanned only once, we divide the total clustering time into two parts for analysis. One is the dataset scanning time, and the other is the hyper image processing time after the entire dataset has been scanned. The results are shown in Fig. 3.

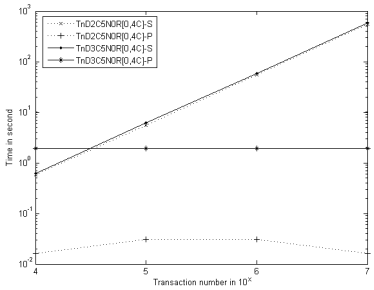


Fig. 3. Data scan and process time of various T and D.

We can see that the data processing time is independent to the parameter settings, and the data scan time scales linearly to the data size. The underlying cluster number in the above experiments was fixed at 5. Then we tested the model by using various C and N with D = 2 and 3. In these experiments, the *bin* value was set to 200 for all dimensions. The resultant data processing time of our method was independent to underlying cluster number and cluster size. The fact is due to that the required calculation to process an image is independent to the image content.

Furthermore the scalability of the proposed method under various *bin* values from 50 to 300 with data dimensionalities from 2 to 3 is shown in Fig. 4.

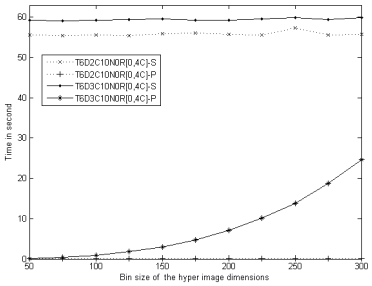
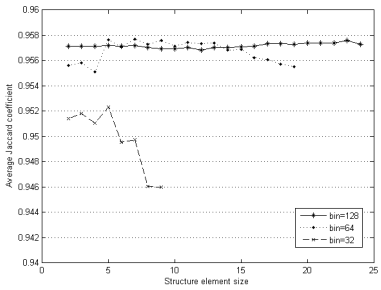


Fig. 4. Data scan and processing time of various *bin* and D.

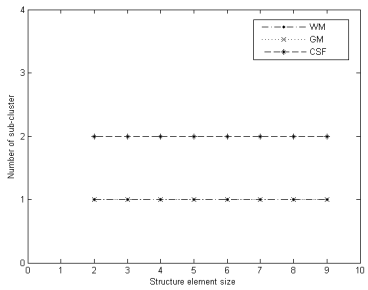
3.2 MRI Image Experiments

We use phantom MRI image of human brain from [3] with provided ground truth to evaluate the clustering quality of the proposed method. This dataset contains T1-, T2-, and PD weighted MRI volumes. A three dimensional vector formed by T1, T2, and PD slices of the middle volumes was used as the test dataset. The slices are of size 179x216. In other words, there are 38664 test records, each comprised of three feature dimensions.

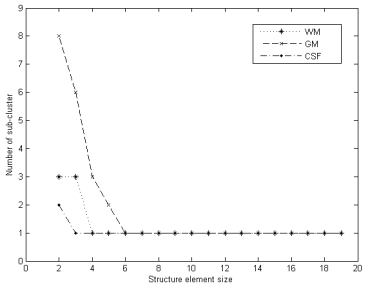
In the medical image, although human brain contains several materials, the most concerned parts are usually intracranial gray matter (GM), white matter (WM), and cerebrospinal fluid (CSF). Thus we evaluated the clustering quality against these three materials.



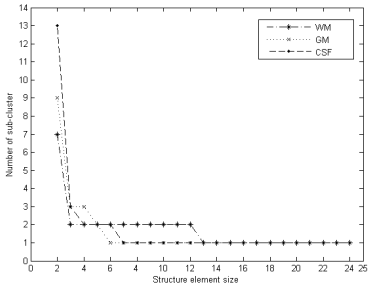
(a) result quality



(b) *bin* value: 32

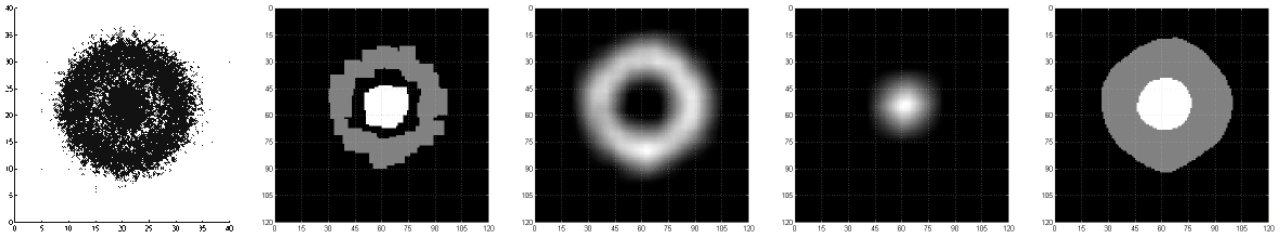


(c) *bin* value: 64



(d) *bin* value: 128

Fig. 5. (a) Average Jaccard coefficients of the three ground truth intracranial materials w.r.t. corresponding labeled result clusters under variant dilation degrees and *bin* values, (b)-(d) the correspondent cluster numbers of each result material under *bin* value=32, 64, and 128, respectively.



(a) (b) (c) (d) (e)

Fig. 6. Clustering two sphere shaped clusters, (a) original dataset, (b) result of connect component extraction, (c) result of fuzzy compensation using blur on cluster 1, (d) blur result on cluster 2, (c) the final clustering result.

In our approach, partial volume pixels may be removed as outliers. However in the clustering problem of this kind, each pixel must be assigned to a cluster as in the case of K-means algorithm. Thus we applied the k-NN method to assign each outlier to a nearest cluster in the hyper image space.

The result cluster number does not set a priori. Each result cluster was labeled as white matter, gray matter, or CSF by which one the maximal value corresponding to the ground truth material was derived. The clusters with the same label were merged into one. The total cluster number in each material and the Jaccard coefficient between the ground truth material and the merged cluster were used for the clustering result evaluation. In practice, the Jaccard coefficient is the higher the better, and the sub-cluster number in each result material is the lower the better. This is because when the cluster labeling process is done by human, it can save a lot of labor work.

In these experiments, we adopted one erosion and one dilation after a thresholding as the image preprocessing setups. The thresholding value was fixed at 3. In other words, using 2 bits for the hyper image pixel weight value is sufficient. We iteratively enlarge the image dilation to merge clusters. The used structure element for the image erosion was fixed by a cube of size 2. And the one for the image dilation was iteratively enlarged from this size. Fig. 5(a) shows the average Jaccard coefficients among these three materials under various dilation degrees and *bin* values. Figures 5(b)-(d) show the corresponding result cluster numbers of each material under different *bin* values.

The Jaccard coefficient of *bin* = 32 in Fig. 5(a) may seem not to be compatible with others; however, their difference is actually very tiny. And the result

sub-cluster number in each result material can facilitate the cluster labeling process.

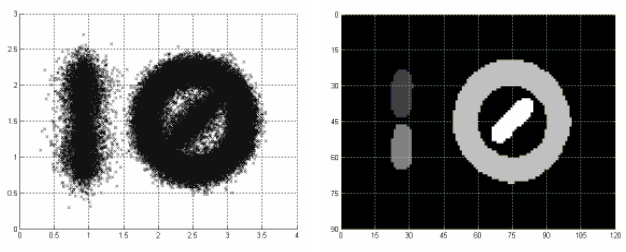
3.3 Special Dataset Experiments

In the following experiments, we generated some special datasets to further demonstrate the clustering ability of the proposed method under various circumstances. Fig. 6(a) shows intuitively two clustered data points with sphere shape. Fig. 6(b) shows a hyper image of *bin* size 120 in each data dimension after using connect component extraction of the proposed method. Figures 6(c)-(d) show a fuzzy compensation of the clustering result in Fig. 6 (b) by using image blur technique. A Gaussian blur was applied to each cluster channel. And the final clustering result was derived as shown in Fig. 6(e). The two spherical clusters were described very well.

It is hard to separate these two clusters by using other existing methods. Distance based clustering algorithms may successfully separate these two by using a very small distance measure. However, it will fail when there exists a density reachable path between these two clusters. Fig. 7 is a more apparent example for showing the limitation of other existing algorithms. The data clusters in the right and left parts of Fig. 7(a) are connected and overlapped. According to our knowledge, none of the existing clustering algorithms can intuitively cluster the dataset into the four groups as shown in Fig. 7(b). The connected clusters can be broken at the isthmuses using image morphological operations efficiently and effectively. Inevitable, cluster overlapping will exist in practical dataset. Thus our model is much more useful than other methods.

4 Conclusion

Our proposed method shows excellent scalability in the dataset size. The image processing time is even independent to the dataset cardinality. Our approach is simple, intuitive, and easy to use. We also show that it can be used to perform on-line, dynamic, hierarchical, and fuzzy clustering efficiently and effectively. Under various clustering scenarios and parameter settings, our method requires to scan the original dataset only once. And the total processing time and space are predictable, calculable, and manageable.



(a) (b)

Fig. 7. Clustering four intuitive clusters: (a) original dataset, (b) clustering result of our proposed method.

Experimental results show that the proposed method is robust and stable under different parameter settings and data distributions. A moderate bin size is adequate to derive good clustering result. It can even cluster data points very well in a small hyper image space. The clustering result is highly controllable while various well-developed image processing methods can be used to extract clusters according to the way we desire. Therefore our approach is more useful and efficient than the other existing methods. The feasibility of selecting other image processing techniques to tailor the clustering process for better results and more applications will be further studied in the future.

References:

[1] Höppner F., Klawonn F., Kruse R., and Runkler T., *Fuzzy cluster analysis*, Wiley Press, New York, 1999.

[2] Guha S., Rastogi R., Shim K., CURE: An efficient clustering algorithm for large databases, *Proc. of the ACM SIGMOD Int'l Conf. on Management of Data, Seattle, ACM Press*, 1998, pp. 73-84.

[3] http://www.bic.mni.mcgill.ca/brainweb/

[4] Ester M., Kriegel H., Sander J., and Xu X., A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, *Proceedings of 2nd International Conference on KDD*, 1996.

[5] Margaret H. Dunahm, *Data Mining Introductory and Advanced Topics*, Prentice Hall, New Jersey, 2003.

[6] Ng R. T. and Han J., Efficient and Effective Clustering Methods for Spatial Data Mining, *Proceedings of the 24th VLDB Conference*, 1994, pp. 144-155.

[7] Sheikholeslami, S. Chatterjee, and Zhang A., WaveCluster: A Multi- Resolution Clustering Approach for Very Large Spatial Databases, *Proceedings of the 24th VLDB Conference*, 1998, New York City.

[8] Wei Wang, Jiong Yang, and Richard Muntz, STING: A statistical information grid approach to spatial data mining, *Proceedings of the 23rd VLDB Conference*, 1997, pp. 186-195, Athens, Greece.