A New Clustering Algorithm Based on Distance and Density*

Xiaopeng Yu[1], Deyi Zhou[2], Yan Zhou[2]

[1]Computer School,Wuhan University, P.R.China, 430072
[2] College of Economics and Trade, Huazhong Agricultural University, P.R.China, 430070
myhbwh@163.com

Abstract- **Clustering is an important data exploration task. Several algorithms for clustering large data sets have been proposed in the literature using different methodologies, which can detect arbitrary shaped clusters where clusters are defined as dense regions separated by low-density regions. Traditional DBSCAN is an important density-based clustering algorithm. But it is difficult to set its two density thresholds $(\varepsilon, MinPts)$ properly. And large volume of main memory must be available in order to guarantee DBSCAN to run smoothly. In this paper, a new DBSCAN based on *k*-nearest neighbors (KNN) is proposed, which merges KNN and DBSCAN to enhance DBSCAN. Firstly, the window-width of each data point is determined and the whole data set is partitioned into some fuzzy cluster (*FC*) by the KNN based on KDE. Next, the local parameters $(\varepsilon, MinPts)$ of each *FC* are unsupervisedly determined according to the entropy theory. Finally, each local ε is mapped to the global ε, and each *FC* is separately clustered. The experimental results show that our clustering method achieves better performance on the quality of the resulting clustering and the results are not sensitive to the parameter *k*.**

Keywords: **Clustering; *k*-nearest neighbors; DBSCAN; entropy theory**

I. INTRODUCTION

Clustering is the task of grouping similar objects together with respect to a distance or, equivalently, a similarity measure. Its objective is to group a large amount of data samples into several clusters, maximizing the similarity between data samples in the same cluster while minimizing that between data samples in different clusters. It is important for many exploratory and discovery tasks including machine learning, pattern recognition, and data mining. Numerous clustering algorithms are proposed in the literature [1]. These algorithms can be partitioned into five categories: partitioning, hierarchical, density-based and grid-based methods, etc [2]. Variants of K-clustering, such as K-means and ISODATA [1], [3] are the partitioning clustering methods that are most widely used because they are computationally attractive. However, they are not very stable and very sensitive to outliers and they often do not work well when the clusters are of different size, shape, and density. In contrast, agglomerative hierarchical clustering (AHC) is more stable but it is not feasible for a large data set. While group average and complete link are not as vulnerable to noise, they have trouble with varying densities and cannot handle clusters of different shapes and sizes [4].

Density-based clustering methods include Denclust (DENsity-based CLUstEring) [5], [15] and DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [6], etc. Density-based clustering has advantages over K-clustering and AHC in discovering clusters of arbitrary shapes and sizes. The DENCLUE is based on Kernel Density Estimation (KDE) and is sensitive to user's parameters (σ, ξ).

STING and CLIQUE are the grid-based clustering. STING (Statistical Information Grid) [7] is a method that relies on a hierarchical division of the data space into rectangular cells. The shape of its result is isothetic. The boundaries of all clusters are horizontal or vertical. CLIQUE (Clustering In QUEst)[8] merges the density-based and grid-based methods. But the results of these two methods are very imprecise [9].

Ester M [6] developed a clustering algorithm DBSCAN based on a density-based notion of clusters. It is designed to discover clusters of arbitrary shape. The key idea in DBSCAN is that for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points. DBSCAN can effectively handle the noise points (outliers). DBSCAN does not need the number of clusters in advance where it automatically detects the clusters with their natural number. As in most other approaches, it faces difficulty in setting density threshold properly, and the quality of the resulting clustering depends on an appropriate choice of the parameters $(\varepsilon, MinPts)$. Some algorithms have been offered in order to enhance DBSCAN in [17], [18]. These algorithms use dataset partitioning as a preprocessing stage to reduce the search space of the clustering technique, and detect clusters in an efficient way. But they need users to perceive the parameter *k* (which is the number of clusters in the data set and influences the final resulting clustering) in advance, which requires users should have the field knowledge. Moreover, reference [17] still needs user to set ε, and reference [18] even needs to specify the two parameters $(\varepsilon, MinPts)$.

In this paper, a new DBSCAN based on KNN is proposed and it is called KNNDBSCAN. Firstly it uses KNN to partition the data set into some *FCs* and unsupervisedly determines the appropriate parameters. Then DSBCAN is

*Supported by the Fund of the National Natural Science (Project No. 70373016) and the State Key Laboratory of Software Engineering in Wuhan University

0-7803-8971-9/05/$20.00 ©2005 IEEE

separately applied to each *FC* to improve the performance and reduce the number of scans. It is obvious that applying the DBSCAN separately on each partition is parallel by nature where a parallel algorithm can be designed for applying DBSCAN to each partition in a parallel manner. The resulting clustering is not influenced by the parameters k and has better qualities and efficiency than the result of the traditional DBSCAN.

The rest of this paper is organized as follows. In the next section we discuss DBSCAN and give its analysis. In Section 3 we propose the KNN based on KDE. Section 4 gives KNNDBSCAN algorithm and its analysis. The paper concludes in Section 5.

II. RELATED WORK

DBSCAN [6] finds dense clusters automatically for a given density threshold. By definition density threshold is specified by two parameters: neighborhood radius (ε) and threshold number of points in c-neighborhood ($MinPts$). According to the definitions, a point p is directly density-reachable from a point q if p is in the $\varepsilon-$ neighborhood of q. A point p is density-reachable from q if there is a chain of points p_i, where $i=1...n$ and p_{i+1} is directly density-reachable from p_i; q is p_1 and p is p_{n+1}. A point p is density-connected to another point q if there is a point o such that both p and q are density-reachable from o. DBSCAN starts by bringing in a point to a temporary storage *(tempstore,* e.g. list) and finding its $\varepsilon-$ neighborhood. If the $\varepsilon-$ neighborhood of a data point contains less than $MinPts$ points then it is marked as noise and another point is brought into *tempstore.* Otherwise, all $\varepsilon-$ neighborhood points are brought into *tempstore.* The whole process repeats until all points are considered. In short, DBSCAN groups the density-connected points together as a dense cluster and removes the points that are not density-connected as noise. The algorithm is given as follows:

Algorithm - Density-based Clustering

Input: Data $\{x_i\}$, $\varepsilon, MinPts$

Output: Dense Clusters

1. *ClusterId* = 1
2. For i = 1 to *N*
3. if x_i is UNCLASSIFIED
4. if x_i is not NOISE
5. ExpandCluster(x_i)

   /* finds all density-connected points of x_i */
6. *ClusterId* = *ClusterId* + 1

DBSCAN does not perform any sort of preclustering and operates directly on the entire database. As a result, for large-scale databases, DBSCAN needs large volume of main memory support and could incur substantial I/O costs. Furthermore, during the clustering process, the number of a cluster's seed objects for expansion increases monotonously and its size is unpredictable. So large volume of main memory must be available in order to guarantee DBSCAN to run smoothly.

If a naive approach is adopted to find the $\varepsilon-$ neighborhood, then for each point x_i in function Expand-Cluster, $O(N)$ is required. Hence the overall complexity is $O(N^2)$. In order to reduce this time complexity DBSCAN uses R*-tree to find $\varepsilon-$ neighborhood of each point. For each point a region query is generated and the leaf nodes of R*-tree that satisfy the query are included in the comparison to find the $\varepsilon-$ neighborhood. The height of an R*-tree is log *N* for a data of *N* points in the worst case. The authors argue that the average run time complexity of DBSCAN is $O(n \log n)$. This complexity is based on the assumption that the number of comparisons inside the minimum bounding rectangles (MBRs) in the leaf nodes of the R*-tree is very small compared to *N*. This may hold true for low dimensional data but it degrades rapidly with increasing dimensionality. This is true for other index structures such as SR-tree, X-tree. This phenomenon is known as *curse of dimensionality.* The reason for this degradation is for high-dimensional data each region query requires to access most of the MBRs. Recent results [19] show that a simple sequential scan of the whole data outperforms any of the space partitioning methods on uniformly distributed data when the dimensionality exceeds around 10.

It is difficult to know the density threshold before hand. A very high value of density threshold will produce many small and very dense clusters while a very small value will generate a few clusters or just one cluster consisting of all data points. DBSCAN aids the user to set density threshold manually [6]. User is shown a graph and asked to take a decision regarding density threshold value.

Research by M. Ankerst shows that the global density threshold does not exist in a complex data set, in which the local densities are very different [11]. So, it is not feasible to use a global neighborhood radius to cluster the whole data set.

Therefore, it is necessary to propose an algorithm to determine the density thresholds unsupervisedly and save main memory. In this paper, KNNDBSCAN is proposed, which merges KNN and DBSCAN. KNN aims to offer DBSCAN the unsupervised density threshold. Firstly, through the way of KNN based on KDE, the data set is partitioned into several *FC* whose densities are obviously different, which cannot make these *FCs* influence each other. Then the local ε of each *FC* is determined. Next, each local $(\varepsilon, MinPts)$ is mapped to the global ε and each *FC* is separately clustered in parallel, which keeps the partition instead of the whole dataset, reduces the number of dataset

scans and save main memory.

III. KNN KDE-BASED CLASSIFICATION

A. KNN KDE-based Rule

KDE is based on the idea that the influence of each data point can be modeled formally using a mathematical function, called the kernel. The kernel function can be seen as a function, which describes the influence of a data point within its neighborhood. The kernel function is applied to each data point. An estimate of the overall density of the data space can be calculated as the sum of the influences of all data points. Clusters can then be determined mathematically by identifying density attractors. Density attractors are local maximum of the overall density function. Determining the density attractors can be done efficiently by a hill-climbing procedure. If the overall density function is continuous and differentiable at every point, the hill-climbing procedure may be guided by the gradient of the overall density function. The mathematical form of the overall density function also allows clusters of arbitrary shape to be described in a very compact mathematical form, namely by a simple equation of the overall density function.

Definition 1 (Window-width). *WW* is the radius of certain volume that is influenced by a data sample. Different points may have different window-width.

Definition 2 (Standard Cluster). If the number of the points in a cluster is more than or equal to the threshold λ, the cluster can be called the standard cluster. A cluster, whose samples number is less than λ, is nonstandard and does not need the next clustering and should be removed. The threshold λ is close to user and can be easily determined. If user knows little about the applied fields, λ may be neglected.

Definition 3 (Related Neighbor). In the k nearest neighbors of a data sample x, the part that possibly belong to the same cluster as x are called its related neighbors. Its number h is between *zero* and k. The other part is the unrelated neighbors.

Given a data set $D=\{x_1,...x_n\}$ and the set of clusters presented in the dataset $\{c_i \mid i=[1\cdots m], m\le n\}$. One data set is shown in Fig.1.

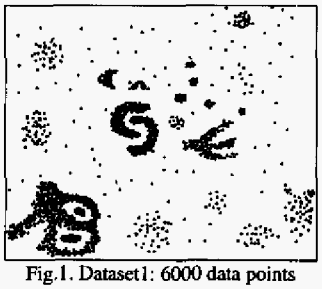

Fig.1. Dataset1: 6000 data points

The KNN query of the point x is the set of its nearest neighbors $K_{nn}(x)=\{x_j \mid j=[1\cdots k]\}$. The aim of KDE-based KNN clustering is to partition $K_{nn}(x)$ into two parts according to the distance from the neighbor to x. One part that is called the related neighbors is possible in the same cluster as x, while the other is not.

The Euclidean distance set is $D_{nn}(x)=\{d_j=d(x,x_j) \mid j=[1\cdot\cdot k]\}$. It is obvious that $d_{j=[1..k]}$ is increasing. The kernel function is a Gaussian function here, and it is defined as

$$f_{Gauss}^{D_{nn}}(x_i)=\sum_{j=1}^{k} e^{\frac{(d_i-d_j)^2}{2}}$$

Lemma 1. $f(x)$ must be a rough sequence, which means $f(x)$ must shape one / some waves.

Proof $\because f(x_1)=\sum_{j=1}^{k} e^{\frac{(d_1-d_j)^2}{2}}, f(x_2)=\sum_{j=1}^{k} e^{\frac{(d_2-d_j)^2}{2}}$

$$\therefore f(x_1)-f(x_2)=\sum_{j=3}^{k} e^{-\frac{(d_1-d_j)^2}{2}}-\sum_{j=3}^{k} e^{-\frac{(d_2-d_j)^2}{2}}$$

$$\because d_1 \le d_2 \Rightarrow (d_1-d_j)^2 \ge (d_2-d_j)^2$$

$$\Rightarrow e^{\frac{(d_1-d_j)^2}{2}} \le e^{\frac{(d_2-d_j)^2}{2}}, j=[3\cdots k]$$

$$\Rightarrow f(x_1)-f(x_2)\le 0 \Rightarrow f(x_1)\le f(x_2)$$

$$\therefore \exists \alpha \in [1,k], f(x_\alpha)\ge f(x_1)$$

$$\because d_{k-1}\le d_k \Rightarrow (d_k-d_j)^2 \ge (d_{k-1}-d_j)^2$$

$$\Rightarrow f(x_k) \le f(x_{k-1}),\ j=[1\cdots k-2]$$

$$\therefore \exists \beta \in [1,k], f(x_\beta)\ge f(x_k) \ \square$$

Therefore, $f(x)$ must shape one / some waves. Then, a hill-climbing procedure can separate the first wave $\{(x_j, f(x_j)) \mid 1<j\le h, h\le k\}$. $x_{j=[1...h]}$ are regarded as the related neighbors(*RN*) in the same cluster as x. And the *WW* at the point x is equal to d_h, namely the distance from x to the hth neighbor x_h. If the point x is in a high-density cluster, its *WW* will relatively be small; otherwise, its *WW* will be large.

The algorithm is shown as follows. First, we determine the *WW* and the related neighbors of each point x. Next, starting from the x having the minimum *WW*, the point and its related neighbors are recursively traversed to generate those *FCs*. Therefore, *FCi* begins with the smallest *WW* and will be gradually separated according to the length of *WW*. And the high-density *FC* can be separated earlier.

Algorithm – KDE-based KNN:
Input: *D*, *k*, *i* =1

Output: *FC, WW, RN*
1. Determine (*WW, RN*)
   //Determine *WW* and *RN* of all *x*
2. Sort (*WW*)
3. Do While *D* is not NULL
4. *X0*=Min (*WW, D*) //Return the minimum *WW* in *D*
5. *FCi*=Clustering_Begin(*x0*)
   //Add *x0* and its related neighbors to *FCi* recursively
6. Remove (*D, FCi*) //Remove *FCi* from *D*
7. *i*=*i*+1
8. Loop
9. Trim (*FC*)
   //Remove those nonstandard *FCi* whose point
   //numbers is less than λ

All the rest *FCs* are the fuzzy clusters of the above data set (shown in Fig.1) and are shown in Fig. 2. We can find that different density clusters are basically separated. The following four resulting clustering are basically the same even though the parameters *k* are different, which means that the parameter *k* does not seriously influence the result of KNN based on KDE.

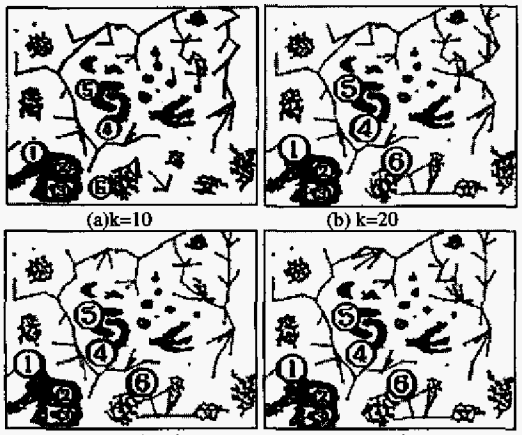

Fig. 2. The resulting clustering of KNN

B. Discussion of the Parameter k

When *k* is larger, each point will have more neighbors. When *k* is smaller, the number of these neighbors is much smaller and these neighbors are the small set and cannot be regarded as the object of simulation.

In fact, if the parameter *k* is not a very small value, the probability that *k* influences *FC* will be small. Suppose the parameter *k* is k_1 and the corresponding *WW* of a stochastic point x is WW_1. The related neighbors are the set $\{x_i \mid d(x,x_i) <= WW_1\}$. When k changes from k_1 to k_2, because k_1 and k_2 are large, all the neighbors between x_{k1} and x_{k2} are far from the related neighbors. Because the limitation of $e^{\frac{(d_l-d_k)^2}{2}}$ $(k \to \infty)$ tends to be zero, the latter density estimation $\sum_{j=1}^{k1} e^{\frac{(d_l-d_k)^2}{2}}$ of the related neighbors is approximately equal to the previous $\sum_{j=1}^{k2} e^{\frac{(d_l-d_k)^2}{2}}$. Therefore, the *WW* will basically keep stable, which will probably make all *FC* stable.

The *WW* estimation chart of a stochastic sample is shown as Fig.3. When *k* is equal to 10, 20, 50 or 100, the first wave is basically stable, which means that the *WW* is basically stable. In this research, the probability of a *WW* keeping stable is more than or equal to 98% and the parameter *k* is a value between 10 and100, *k*<<N.

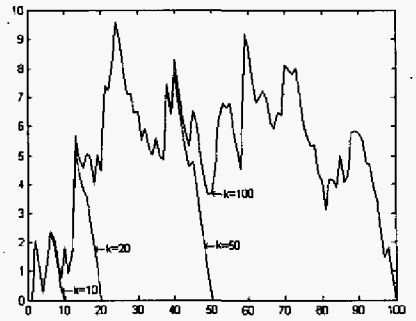

Fig.3. WW estimation chart of a stochastic sample

However, there are exceptional cases, in which certain *WW* will be changed when *k* is changed. When *WW* is smaller than the previous one, these related neighbors should be a subset of those previous related neighbors, which will not influence *FC*. This phenomenon is discovered in $FC_{4,5}$ (shown in Fig.2). When *WW* is larger than the previous one, the sample will have more related neighbors. If the sample is inside a *FC*, this will not influence *FC* (e.g. $FC_{1,2,3}$). Only when the sample is on the boundary of a *FC*, it is possible that the *FC* will be changed (e.g. FC_6). This small possibility phenomenon cannot seriously influence the large data set and will be farther processed in the next step.

IV. KNNDBSCAN ALGORITHM

A. Entropy optimization

We say the data has orderly configurations if it has distinct clusters, other it has disorderly of chaotic configuration. Entropy is a measure of the amount of uncertainty in the outcome of a random experiment, or equivalently, a measure of the information obtained when the outcome is observed. This concept has been defined in various ways [13] and generalized in different applied fields, such as communication theory [14] etc.

From entropy theory, we know that entropy (or probability) is less for orderly configurations, and more for disorderly configurations. We try to visualize the complete data set from individual data points then an orderly configuration mean that for most individual data points there

are some data points close to it (i.e., they probably belong to the same cluster), and others away from it. In a similar reasoning, a disorderly configuration means the most of the data points are scattered randomly. So, we can evaluate entropy at the cluster center to weigh the effectiveness of the kernel density estimation.

B. Estimation of the Parameter ε and $MinPts$

The parameter ε determines the neighborhood radius of a point. In fact, the *WW* of each data point is its real influence and has been determined in section 3.1. If ε is too small then the $\varepsilon-$neighborhood may be very less or even 0 which may result in too many small size clusters. If ε is too large the $\varepsilon-$neighborhood will be very large and performance of density-based clustering may reduce as clusters merge together.

Given a $FC_i = \{x_j \mid j \le n_i, n_i \le n\}$. n_i is the number of the data points in FC_i. All *WW* are uniquely normalized to the set $\{w_t \mid t \le n_i\}$. Let $temp_j$ be the number of neighbor of x_j in neighborhood radius w_t. The ε estimation entropy is defined as:

$$\varepsilon\_En = -\sum_{j=1}^{ni} \frac{(RN_j - temp_j)^2}{S} \log \frac{(RN_j - temp_j)^2}{S},$$

$S = \sum_{j=1}^{ni} (RN_j - temp_j)^2$. RN_j is the number of related neighbors of the point x_j. w_t which makes ε_En the minimum is namely the optimized parameter ε of FC_i. Next for each cluster, $MinPts$ is determined using a simple arithmetic: $MinPts = \frac{\varepsilon Volume_k}{totalVolume_k} * n_k$, where $k = 1 \ldots n'$, n' is the number of *FC*, and n_k is number of points in FC_k. $TotalVolume_k$ and $\varepsilon Volume_k$ have been researched in [18].

According to the above idea, the parameter ε of each FC can be determined, and different *FC* has the different parameter $MinPts$. The set $\{MinPts_i \mid i \le n'\}$ can describe all the parameter $MinPts$. Let the parameter ε and $MinPts$ of the *FC* that has the minimum $MinPts$ be the global parameters ε_s and $MinPts_s$. And let the coefficient k_i be equal to $\varepsilon_s / \varepsilon_i$. Therefore, when ε_i is mapped to ε_s, the neighborhood radius estimation of the points in each *FCi* is namely equal to $\varepsilon_i * k_i$ and the distance between any two points in *FCi* is multiplied by k_i in order to search their neighbors.

C. Proposed Algorithm

According to the above description and analysis, the KNNDBSCAN algorithm can be described as follows. The rest clusters (*C*) are the resulting clustering and shown in Fig.4. The areas made up of lined-points are the resulting clustering and the noises are marked with the sign ×.

Algorithm – KNNDBSCAN
Input: *D*, *k*, *i* =1
Output: *Cluster*
1. Determine (*WW*, *RN*)
   //Determine *WW* and *RN* of all *x*
2. Generaete_FC (*D*, *WW*)
   // Generate fuzzy clusters, the numbers is n'
3. For i=1 to n'
4. Determine_Local_ε (*FCi*)
   // Detetmine local ε according to the entropy theory
5. Next
6. Determine_Density_Threshold (ε, $MinPts$)
7. Parallel_DBSCAN (*FC*)
   // local ε is mapped to ε_s, cluster each *FCi* in parallel
8. Trim (*Cluster*)
   //Remove *Clusteri* whose point numbers is less than λ

The traditional DBSCAN algorithm requires two parameters $(\varepsilon, MinPts)$. But the new algorithm only requires one parameter k which can unsupervisedly determine the density thresholds $(\varepsilon, MinPts)$. And its result is not sensitive to the parameter *k*.

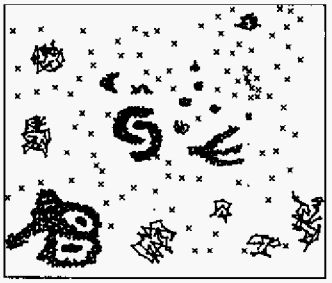

Fig. 4. Results of KNNDBSCAN on Dataset1
(ε =0.6 and $MinPts$ =5)

The time complexity of KNNDBSCAN depends mainly on three parts: (1) the calculation of KNN; (2) the calculation of local ε; (3) DBSCAN. To retrieve the k-neighborhood of an object *x*, a region query with the center *x* and the radius is used. If a tree-based spatial index can be used, the run-time is reduced to $O(n \log n)$ since region queries are supported efficiently by spatial access methods such as the R*-tree or the X-tree [11] .The second part of time complexity is obviously less than $O(n \log n)$ which is the time complexity of KNN, because the step only searches the neighbors in the neighborhood radius and the radius has been determined. The third is not more than the time complexity

$O(n \log n)$ of traditional DBSCAN. Let the numbers of all points in each FC be set $\{n_j \mid 1 \le j \le n', n_j \le n, \sum_{j=1}^{n'} n_j = n\}$ and $n_{\max}$ is the maximum value in the set. $\sum_{j=1}^{n'} (n_j \log n_j)$ is less than or equal to $\sum_{j=1}^{n'} (n_j \log n_{\max})$,namely $n \log n_{\max}$,which is less than or equal to $n \log n$. Moreover, this step can be run in parallel as explained before. As the size of the data in a FC is usually much smaller than that of whole data, so the neighborhood search is more efficient which leads to increase in speed. Therefore, the time complexity of KNNDBSCAN is $O(n \log n)$, which is appropriately the same as that of traditional DBSCAN.

V. CONCLUSION

DBSCAN is an important clustering algorithm. It has many advantages. The quality of the resulting clustering depends on an appropriate choice of the parameters as most other approaches. The main problem is that it is difficult to determine the global parameters ε and $MinPts$. In this paper, a new DBSCAN called KNNDBSCAN based on k-nearest neighbors is proposed, which merges KNN and DBSCAN. The whole data set is firstly partitioned into some fuzzy clusters (FC) that have more orderly configurations by the KNN based on KDE algorithm. Then, the local parameters $(\varepsilon, MinPts)$ of each FC are unsupervisedly determined according to the entropy theory. In the end, each local ε is mapped to the global parameter ε_s which makes the global parameters suitable for the whole data set, and each FC is individually clustered in parallel which increase in speed and save main memory. The quality of the resulting clustering is not sensitive to the parameter k. An experimental prototype system has been developed, implemented, and shown that the proposed algorithm is robust, and has better quality and efficiency.

REFERENCES

[1] A.K. Jain, M.N. Mutry, P.J. Flynn," Data Clustering- A Review," ACM Computing Surveys, 1999, pp264-323

[2] JiaWei Han, Micheline Kamber. Data Mining Concepts and Techniques. China Machine Press, 2004,pp242-245

[3] G.H. Ball, D.J. Hall,"A novel method of data analysis and pattern classification," Springfield, Stanford, 1965, pp80-85

[4] L. Eltoz, U Steinbach,and V. Kumar,"A new shared nearest neighbor clusteing algoithm and its applications," AHPCRC, Tech. Rep. 134, 2002, pp200-206

[5] Alexander Hinneburg, Daniel A. Keim, "A General Approach to Clustering in Large Databases with Noise," Knowledge and Information Systems, vol 5, 2003, pp387-415

[6] M Ester, H.-P. Knegel, I. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," In proceeding of Knowledge Discovery and Data Mining, 1996,pp 226-231

[7] Wang W., Yang J., and Muntz R., "STING: A Statistical Information Grid Approach to Spatial Data Mining", Proc.23rd Int. Conf. on Very Large Data Bases, Morgan Kanfnmon, 1997,pp186-195

[8] Rakesh Agrawal, Johannes Gchrke, and Dimilrios,"Automatic Subspacc Clustoring of High Dimensional Data Mining Applications," Proc. ACM SIGMOD Int. Conf. on Management of Data, 1998, pp94-105

[9] Yasser El-Sonbaty, M. A. Ismail, and Mohamed Farouk,"An Efficient Density Based Clustering Algorithm for Large Databases," ICTAI 2004. 16th IEEE International Conference, pp673 – 677

[10] Gan Wen-yan, LI De-yi,"Hierarchical Clustering based on Kernel Density Estimation. Journal of System Siumlation," vol 2 2004,pp302-305, 309

[11] M. Ankerst, M. M. Breunig, and H-P Kriegel,"OPTICS: Ordering Points to Identify the Clustering Structure," In Proceeding of ACM SIGMOD, 1999,pp49-60

[12] Linder berg T,"Scale-space for discrete signals {J}," IEEE Trans. Pattern analysis and Machine Intelligence, 2000, pp1396-1410

[13] Hahn-Ming Lee,Chih-Ming Chen,Jyh-Ming Chen, and Yu-Lu Jou,"An efficient fuzzy classifier with feature selection based on fuzzy entropy," IEEE Trans.Syst. , Man, Cybern.B. , 2001, pp426-432

[14] Ching J.Y., Wong A.K.C.,and Chan K.C.C.,"Class-dependent discretization for inductive learning from continuous and mexed-mode data," IEEE Trans. Pattern Anal.Maching Intell., 1995,pp641-651

[15] Hinneburg A, Keim D A.,"An efficient approach to clustering in large multimedia databases with noise," In Proceeding of the 4th International Conference on Knowledge Discovery and Data mining [C], 1998,pp58-65.

[16] G.Kollios, D.Gunopulos,and N.Koudas,"Efficient biased sampling for approximate clustering and outlier detection in large data sets," IEEE Trans. Knowledge and Data Engineering, vol.15, pp1170-1187, 2003

[17] Manoranjan Dash,Huan Liu,"'1+1>2':Merging Distance and Density Based Clustering", Proceedings. Seventh International Conference, 2001,pp32-39

[18] Yasser El-Sonbaty, M.A.Ismail, and Mohamed Farouk,"An Efficient Density Based Clustering Algorithm for Large Databases", ICTAI 2004. 16th IEEE International Conference, pp673 – 677

[19] R.Weber,H.J Schek, and S.Blott,"A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," In Proceedings of the Intl. Conf. On very large Databases, 1998, pp194-205

[20] Peter Bajcsy , Narendra Ahuja,"Location- and density-based hierarchical clustering using similarity analysis," IEEE Trans. Pattern Analysis and Machine Intelligence, 1998,pp1011-1015