Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Technology 6 (2012) 485 – 492

Procedia Technology

2nd International Conference on Communication, Computing & Security [ICCCS-2012]

A prototype-based modified DBSCAN for gene clustering

Damodar Reddy Edla[a,]*, Prasanta K. Jana[a], *IEEE Senior Member*

[a]Department of Computer Science & *Engineering,* Indian School of Mines, Dhanbad, Jharkhand-826 004, India

Abstract

In this paper, we propose, a novel DBSCAN method to cluster the gene expression data. The main problem of DBSCAN is its quadratic computational complexity. We resolve this drawback by using the prototypes produced from a squared error clustering method such as *K*-means. Then, the DBSCAN technique is applied efficiently using these prototypes. In our algorithm, during the iterations of DBSCAN, if a point from an uncovered prototype is assigned to a cluster, then all the other points of such prototype belongs to the same cluster. We have carried out excessive experiments on various two dimensional artificial and multi-dimensional biological data. The proposed technique is compared with few existing techniques. It is observed that proposed algorithm outperforms the existing methods.

Keywords: Clustering; gene expression data; DBSCAN; squared error method; prototype.

1. Introduction

The recent advances in biotechnology (Meagher, 2002) and bioinformatics (Searls, 2000) allow researchers to measure a large number of expression levels of genes collected over time under different conditions. It has been a problem for a long time to handle such large number of data patterns together. The experiments of an indispensable tool called Microarray (Ban-Dor et al., 1999) have made it possible to measure gene expression levels for thousand of genes simultaneously. The analysis of large data produced by these experiments facilitates potential insight into gene function and regulatory mechanisms (Ban-Dor et al., 1999). But it is not an easy task to analyze each and every entity of such enormous data individually. Clustering has been proved to work efficiently in this direction. Classification through clustering is the most common problem that often arises in various fields (Cai et al., 2007; Liu et al., 2012; Garibaldi et al., 2006). Clustering is a widely used knowledge discovery technique of data mining to expose the inherent structure of the given data which is useful

* Corresponding author. +91-8986666998; fax: 91-326-2296563.
E-mail address: dr.reddy.cse@gmail.com.

doi:10.1016/j.protcy.2012.10.058

for further analysis. In other words, Clustering aggregates similar input patterns into distinct, mutually exclusive subsets referred to as clusters (Xiao et al., 2003). In case of genome clustering, we group the genes with similar changes in their expression into a separate cluster. Here, we assume that there is a mapping (unknown) that assigns a label to each of the genes and the objective of clustering is to tend to this mapping, i.e. to assign every gene to a group. In the last few decades, a significant amount of research has been carried out on clustering and a number of clustering algorithms (Li and Tian, 2007; Nakamura and Kehtarnavaz, 1998; Frossyniotis et al., 2004) have been developed. However, a special attention is paid for genome clustering because of the distinct characteristics of gene expression data.

Du et al., 2008 has proposed an algorithm called *PK*-means to cluster the GE data by combining both the *K*-means and particle-pair optimizer (PPO) which is a variation of the particle swarm optimization (PSO). Barigov and Mardaneh, 2006 developed an incremental algorithm which is a new version of global *K*-means clustering for gene clustering. A genetic clustering algorithm named GAGR has been proposed to cluster the genome data using *K*-means (Chang et al., 2009). A semi-supervised clustering technique named GO Fuzzy has been proposed by Tari et al., 2009. Compared to the other clustering models, the density-based methods such as DBSCAN (Ester et al., 1996) have been extensively used for clustering large data such as microarray. Jiang et al., 2003 proposed a density-based hierarchical clustering called DHC for time series GE data. Bohm et al., 2004 proposed a new density-based clustering method for GE type of data by extending the well-founded notion of density connected clusters. However, most of these methods do not meet all the requirements because of the diverse characteristics of the complex data such as microarray. Moreover, most of these methods have quadratic computational complexity.

Considering all these issues, we propose, here, a prototype based DBSCAN method which has low computational complexity. The proposed method is experimented on several artificial and biological data sets (UCIMLR, 2012; Du et al., 2008). The results are compared with few existing techniques namely, I-DBSCAN (Viswanath and Pinkesh, 2006), DBCAMM (Ren et al., 2012), VDBSCAN (Liu et al., 2007) and KFWDBSCAN (Huang and Bian, 2009).

The rest of this paper is arranged as follows. In Section 2, we illustrate all the necessary terminologies. Then we describe the proposed algorithm in Section 3. Then Section 4 presents the detailed experimental analysis. Then Section 5 concludes the paper followed by some useful references.

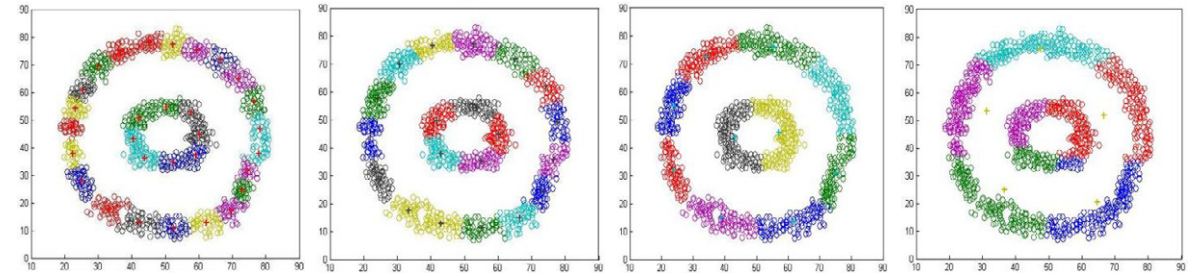
2. Terminologies

2.1. DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is one of the famous density based clustering technique (Ester et al., 1996) to cluster the data of arbitrary shapes. A cluster in this model is described as a linked region that exceeds a given density threshold (Bicici and Yuret, 2007). The functioning of DBSCAN is directed by two well known definitions, namely, *density-reachability* and *density-connectability*, which depend on two predefined parameter values of the DBSCAN clustering known as the size of the neighbourhood denoted by ε and the minimum points in a cluster N_{min}. In DBSCAN, we begin with a random point x and finds all the points which are density-reachable from x with respect to ε and N_{min}. If x is a core point, then the formation of a cluster is completed with respect to ε and N_{min}. It is obvious to note that no points are density-reachable from x when x is a border point in case of which DBSCAN begins with an unclassified point to repeat the same process. The two parameters ε and N_{min} direct the notion of DBSCAN and decide the quality of clusters. These two parameters are used in a global way in the whole DBSCAN, i.e. the values of these parameters are stable for all the clusters. DBSCAN visits each point of the database, possibly multiple times. Usually all the DBSCAN methods have a time complexity of $O(N^2)$.

2.2. Prototype

We define a prototype p of the given set S as a representative point for a group of points of S. In the proposed method the prototypes are chosen from the K-means algorithm. This method generates a partition of the given data that aims to minimize the squared error. Since the K-means is used to produce the prototypes, its definition with respect to K-means is as follows. Let $S = \{X_1, X_2, \ldots, X_N\}$ is the given set of N data points where each point $X_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,d}) \in R^d$ is of dimension d and K be the required number of clusters $C_1, C_2, \ldots, C_K$. Now the K-means is applied with K' ($>K$) number of desired clusters $C_1, C_2, \ldots, C_{K'}$. Then we consider the centroids $c_1, c_2, \ldots, c_{K'}$ of these clusters as the prototypes to represent the points of their corresponding clusters. In proposed method, we initially form K' number of prototypes where K' is assumed to be sufficiently larger than the number of clusters K which is unknown. Several prototype based methods are found in the literature. A multi prototype clustering algorithm is proposed by Liu et al., 2009. This method uses the prototypes produced from squared error clustering method. A major drawback of this method is that it requires the number of clusters a priori. A minimum spanning tree based prototype clustering algorithm has proposed by Luo et al., 2010. This method exploits the prototypes produced by the MST using the split and merge scheme. However, this method involves four parameters which are difficult to estimate correctly in advance. The below Figs. 1(a-d) shows 30, 20, 10 and 5 prototypes formed in the cluster-inside-cluster data.



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

Fig. 1. Result of squared error clustering on cluster-inside-cluster data with number of prototypes (a) 30; (b) 20; (c) 10; (d) 5

3. Proposed modified DBSCAN method

Given a set S of N data points, we first choose a number K' which is larger than the required number of clusters. We then apply any squared error clustering method (such as K-means) on the given data to form K' number of subclusters, say, $\{P_1, P_2, \ldots, P_{K'}\}$. The corresponding centroids $\{p_1, p_2, \ldots, p_{K'}\}$ are chosen as the prototypes. Now, we apply the DBSCAN algorithm on the given data by avoiding the unnecessary distance computations with the help of prototypes produced by squared error clustering. The key feature of this method is as follows. If a point x of a prototype p is within the ε-neighbourhood of a point y of another prototype q, then all the points represented by both of these prototypes p and q belong to the same cluster. The main scheme of the proposed method is as follows. Initially, all the K' prototypes are marked as unclassified. We start randomly with a subcluster P_m represented by the prototype p_m. Add the subcluster P_m to cluster C_j (j equals to 1 initially) and mark its prototype p_m as classified. Then, the Euclidean distances of all the points $x_i \in P_m$ from p_m are calculated. We now start with the point $x_i \in P_m$ whose distance from p_m is maximum and compute its ε-neighborhood $N_\varepsilon(S, x_i)$. At this moment, distances from the point x_i to all the prototypes (except p_m) are calculated to apply the DBSCAN on the points whose prototype is closer to x_i. This computation is needed to deal only with the points of closer prototypes and to skip the points of far prototypes. Assume, p_n is the closer prototype of x_i. Then we find whether there is any point $y \in P_n$ such that y is directly density-reachable (see definition 4) from x_i with respect to ε and N_{min}. If so, then add the whole subcluster P_n to C_j and mark p_n as

classified. Now, we consider an unclassified prototype p_r which is closer to the point x_i to repeat the same process. On the other side, when no point of P_n is directly density-reachable from x_i, then we can ignore x_i since we have considered the prototypes with respect to x_i in sorted order. After ignoring x_i, we consider the next farthest point of P_m from its prototype p_m and repeat the same process for t number of farthest points of P_m from p_m. With this, the initially considered prototype P_m is exhausted. If any new subclusters are amalgamated to the cluster C_j during this iteration with respect to the subcluster P_m, then the same process is repeated for each of this newly merged subcluster to proceed further. The stage of complete formation of a cluster can be identified as follows. If at least one of the K' prototypes is unclassified and no points of any unclassified prototype are directly density-reachable from the points of any classified prototypes. In this case we increment the value of j by 1 and the points of next coming cluster are stored in C_j. The whole process is terminated when all the K' prototypes are classified. Unlike other DBSCAN methods here the value of N_{min} is fixed at 1. Because, if a point y of another prototype P_n is directly density-reachable from x_i, all the other points of P_n and P_m belong to the same cluster. The algorithm is presented as follows.

Algorithm MDBSCAN(*S*, *t*)

***Input*:** A set S of n data points and the number of clusters k; ***Output*:** The set C_i of clusters
***Functions and variables used*:**
t: Maximum number of points of any prototype to apply the DBSCAN.
ε : A neighborhood constant.
N_{min} : Limit of the minimum number of points for the DBSCAN.
$SE(S, K')$: A function to produce the K' ($>K$) subclusters $\{P_1, P_2, \ldots, P_{K'}\}$ using squared error clustering.
P: A set to store the K' subclusters $\{P_1, P_2, \ldots, P_{K'}\}$.
p: A set to store the K'prototypes $\{p_1, p_2, \ldots, p_{K'}\}$ which are the centroids of $\{P_1, P_2, \ldots, P_{K'}\}$.
$Sort(x_i)$: A function to sort the points x_i.
$d(x,y)$: Euclidean distance between the points x and y.
ε-$nbhd(x)$: A function to find the ε-neighborhood of a point x.
Directly density-reachable$(x, \varepsilon, N_{min})$.
C_k: Set of k clusters. /* initially empty */
i, j, k, l, *count*: Temporary variables /* k, l are 1 and *count* is 0 initially */, $Temp_l$: Temporary set.

Step 1: Call squared error clustering function $SE(S, K')$ to find a set of K' subclusters $\{P_1, P_2, \ldots, P_{K'}\}$.
Step 2: Find the K' prototypes $\{p_1, p_2, \ldots, p_{K'}\}$ as the centroids of $\{P_1, P_2, \ldots, P_{K'}\}$
Step 3: Mark all the prototypes $p_1, p_2, \ldots, p_{K'}$ as unclassified.
Step 4: Randomly start with a subcluster P_m with an unclassified prototype p_m.
Step 5: Add P_m to C_k, i.e. $C_k \leftarrow C_k \cup P_m$. Then mark p_m as classified.
Step 6: Calculate $d(x_i, p_m)\ \forall\ x_i \in P_m$ and sort these distances in descending order using *Sort*();
Step 7: Start with a point $x_i \in P_m$ whose distance from p_m is maximum, $count \leftarrow count + 1$;
Step 8: Calculate $d(x_i, p_j)\ \forall\ p_j \in p\text{-}\{p_m\}$ and sort these distances in ascending order using
Step 9: Find the closer unclassified prototype (say p_n) to x_i.
Step 10: Find the ε-neighbourhood of x_i by calling the function ε-$nbhd(x_i)$.
Step 11: If there exists at least one point $y \in P_n$ such that y is directly density-reachable from x_i with respect to ε and N_{min}, then
{
Add the subcluster P_n to C_k and $Temp_l$.
/* i.e. $Temp_l \leftarrow Temp_l \cup P_n$; $C_k \leftarrow C_k \cup P_n$ */
Mark p_n as classified.

count ← *count* + 1;
Repeat from Step 9 with next closer prototype to x_i.
}
Else
{
If count < *t*, then

Repeat from Step 7 with new x_i ($\in P_m$) value taken as the $(count+1)^{th}$ farthest point from p_m.

Else
{
If ($Temp_l \neq \phi$) then
{
$l \leftarrow l + 1$;
Repeat Steps 6 to 11 for every subcluster of $Temp_l$.
}
Else
{
If all the K' prototypes are classified then go to Step 12;
Else { $k \leftarrow k + 1$;
Go to Step 4; }
}
}
}
Step 12: Output $C_q \; \forall \; q = 1, 2,\ldots,k$ and Exit ( );

3.1 Time complexity

The initial phase of squared error clustering algorithm runs in $O(K'.N.T)$ time, where N is the number of points, K' is the number of subclusters and T is the number of iterations. As the K' value is fixed here, it requires $O(N.T)$ time. Assume m is the maximum number of points for any K' subclusters. The distances from the points of all K' subclusters are computed with respect to only their prototypes (centroids) which require $O(m.K')$. All the points of K' subclusters are sorted with respect to their distances from the corresponding prototypes in $O(K'.m.\log m)$ time. Then we have computed the distances from t number of points of all the subclusters to the K' prototypes. This is needed $O(K'.t^2)$ computation time. The ε-neighbourhood is calculated for t number of points of all the K'subclusters with respect to only their neighbouring subclusters (say q). The computation time for this task is $O(K'.t.q.m)$ time. Here, all the values of K', m, t and q are very small compared to the given value of N. Therefore, The overall time complexity of the proposed method is maximum $\{O(N.T), O(K'.t.q.m)\}$. Here the values of K', t, q and m are significantly small compared to N.

4 Experimental analysis

We tested the proposed algorithm on various artificial and biological data sets using MATLAB. The experiments were performed on an Intel Core 2 Duo Processor machine with T9400 chipset, 2.53 GHz CPU and 4 GB RAM running on the platform Microsoft Windows Vista. In order to compare with our algorithm, we have implemented few existing techniques, namely, I-DBSCAN (Viswanath and Pinkesh, 2006), DBCAMM (Ren et al., 2012), VDBSCAN (Liu et al., 2007) and KFWDBSCAN (Huang and Bian, 2009). The results are as follows.

4.1 Results of artificial data

We have used the 6 artificial data namely, *Two-spiral*, *Cluster-inside-cluster*, 4-L, *Half-kernel*, *Crescent-full moon* and *Outlier* for the experimentation. Initially, the *K*-means is applied on all these data sets with the number of prototypes 25. The prototypes of all the above data are shown by Figs. 2(a), 2(c), 2(e), 2(g), 2(i) and 2(k) respectively. The proposed method is experimented on all the above illustrated data sets. It has successfully produced desired clusters as shown in the figures 2(b), 2(d), 2(f), 2(h), 2(j) and 2(l) respectively.

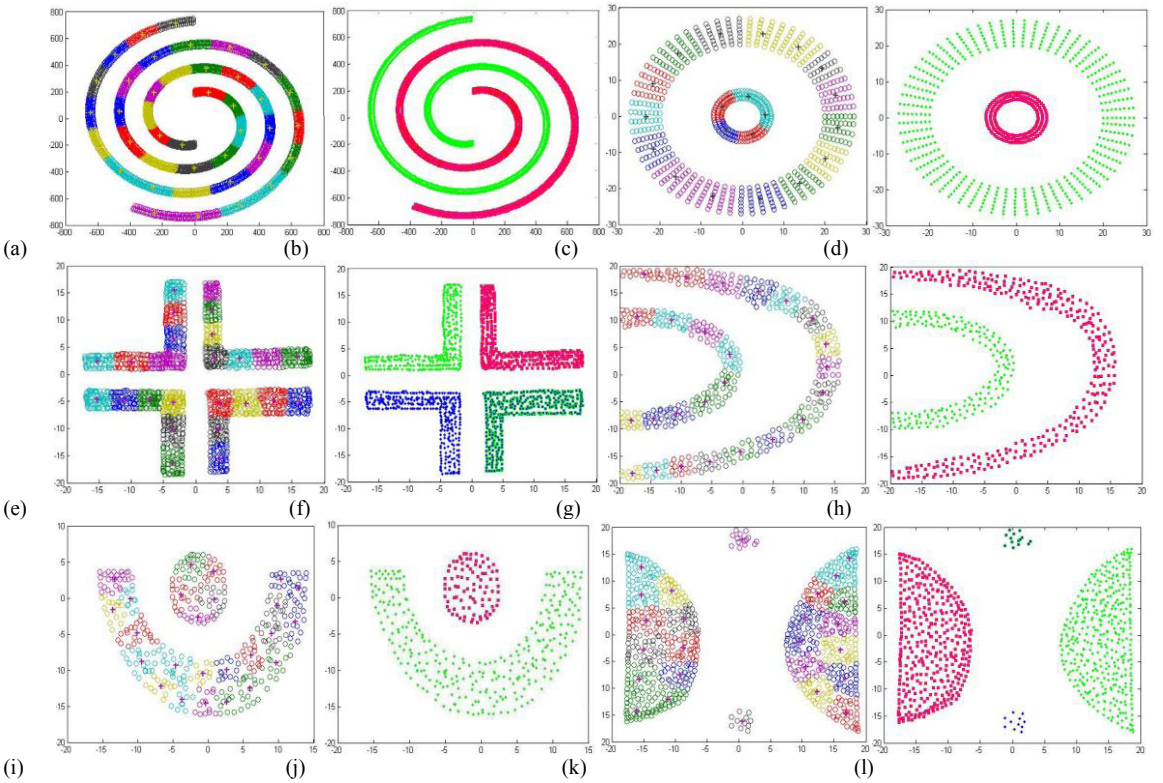


Fig. 2. (a),(c),(e),(g),(i),(k) prototypes of 2-spiral, cluster-inside-cluster, 4-L, half-kernel, crescent-full moon and outlier data; (b),(d),(f),(h),(j),(l) result of proposed method on 2-spiral, cluster-inside-cluster, 4-L, half-kernel, crescent-full moon and outlier data

4.2 Results of biological data using error ratio

Here, we compare the results of 8 biological data sets (UCIMLR, 2012), namely, iris, wine, statlog heart, breast tissue, pima-Inida-diabetes, cloud, blood transfusion and yeast using the error rate proposed by Khan and Ahmed, 2004 as follows.

$$ER = \frac{Number\ of\ misclassified\ objects}{Total\ number\ of\ objects} \times 100\% \quad (1)$$

It can be seen that less value of *ER* indicates more quality. The comparison results of the proposed method with I-DBSCAN, DBCAMM, VDBSCAN and KFWDBSCAN using the error rate (*ER*) are shown in Table 1. It is easy to observe that the proposed method produces better results over the existing.

Table 1. Comparison results of eight biological data sets using the error rate (*ER*)

(A: I-DBSCAN, B: DBCAMM, C: VDBSCAN, D: KFWDBSCAN, E: proposed method)

Data	Size	Misclassified Patterns					*ER*				
		A	B	C	D	E	A	B	C	D	E
Iris	150	12	14	21	19	**5**	8.0	9.3	14.0	12.6	**3.3**
Wine	178	10	12	18	22	**11**	5.6	6.7	10.1	12.3	**6.1**
St. Heart	270	44	27	31	29	**13**	16.2	10.0	11.4	10.7	**4.8**
B. Tissue	106	23	5	12	9	**2**	21.6	4.7	11.3	8.4	**1.8**
P.I. Diabetes	768	33	41	56	71	**21**	4.2	5.3	7.2	9.2	**2.7**
Cloud	1024	47	35	76	58	**17**	4.5	3.4	7.4	5.6	**1.6**
B. Transfusion	748	56	21	34	73	**23**	7.4	2.8	4.5	9.7	**3.0**
Yeast	1484	24	19	26	55	**8**	1.6	1.2	1.7	3.7	**0.5**

4.3 Results of gene expression data using runtime

In this section, we consider four gene expression datasets for the experimentation, namely Yeast cell-cycle, Sporulation, Lymphoma/Leukemia and Diauxic (Du et al., 2008). Then the proposed and the existing methods are applied on these four gene expression data sets. It is obvious to note from the comparison results shown in Figs. 3 (a-d) that the proposed method has less computational time compared to the existing algorithms.

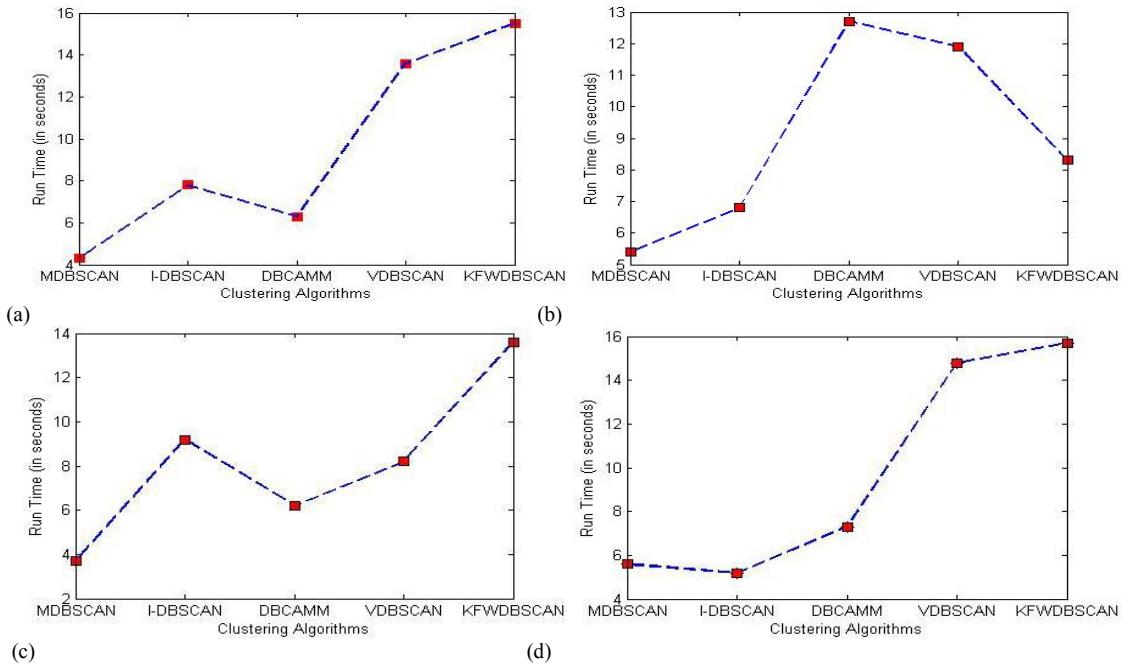


Fig. 3. Run time comparison of the proposed MDBSCAN method with I-DBSCAN, DBCAMM, VDBSCAN and KFDBSCAN on the gene expression data sets, (a) Yeast cell-cycle; (b) Sporulation; (c) Lymphoma/Leukemia; (d) Diauxic

5. Conclusion and future work

We have proposed a method to speed up the DBSCAN algorithm using the prototypes produced by the squared error clustering method. The proposed method is applied on various artificial and biological data. The results are compared with various existing techniques in terms of error rate and run time. All the comparison

results have shown the effectiveness of the proposed modified DBSCAN method over the existing methods. Our algorithm is insensitive to the selection of initial prototypes and it is able to produce the clusters of arbitrary shapes. However, if the number of clusters is large, it may not be easy to start with appropriate number of prototypes. If the ε is chosen unusually, it affects the final clusters. Therefore, in future, we work in this direction of choosing the appropriate number of prototypes and automating the ε-value.

Acknowledgements

We sincerely thank the Council of Scientific & Industrial Research (CSIR), New Delhi, India, for supporting this work under the grant (No. 25(0177)/ 09/EMR-II).

References

Meagher, RB., 2002. Post-genomics Networking of Biotechnology for Interpreting Gene Function. Current Opinion in Plant Biology 5(2), p. 135.

Searls, DB., 2000. Using Bioinformatics in Gene and Drug Discovery. Drug Discovery Today 5(4), p. 135.

Ban-Dor, A., Shamir, R., Yakhini, Z., 1999. Clustering Gene Expression Patterns. Computational Biology 6, p. 281.

Cai, W., Chen, S., Zhang, D., 2007. Fast and Robust Fuzzy *C*-means Clustering Algorithms incorporating Local Information for Image Segmentation. Pattern Recognition 40, p. 825.

Liu, Z., Zheng, Q., Xue, L., Guan, X., 2012. A Distributed Energy-Efficient Clustering Algorithm with Improved Coverage in Wireless Sensor Networks. Future Generation Computer Systems 28(5), p. 780.

Garibaldi, U., Costantini, D., Donadio, S., Viarengo, P., 2006. Herding and Clustering in Economics: The Yule-Zipf-Simon Model. Computational Economics 27, p. 115.

Xiao, X., Dow, ER., Eberhart, R., Miled, ZB., Oppelt, RJ., 2003. "Gene clustering using self-organizing maps and particle swarm optimization," 17th International Symposium on Parallel and Distributed Processing. Washington, DC, USA.

Li, X., Tian, Z., Optimum Cut-based Clustering. Signal Processing 87, p. 2491.

Nakamura, E., Kehtarnavaz, N., 1998. Determining Number of Clusters and Prototype Locations via Multi-Scale Clustering. Pattern Recognition Letters 19, p. 1265.

Frossyniotis, D., Likas, A., Stafylopatis, A., 2004. A Clustering Method based on Boosting. Pattern Recognition Letters 25, p. 641.

Bagirov, AM., Mardaneh, K., 2006. "Modified global *K*-means algorithm for clustering in gene expression data sets," Workshop on Intelligent Systems for Bioinformatics (WISB '06). Darlinghurst, Australia.

Chang, D., Hang, XZ., Zheng, C., 2009. A Genetic Algorithm with Gene Rearrangement for *K*-means Clustering. Pattern Recognition 42, p. 1210.

Tari, L., Baral, C., Kim, S., 2009. Fuzzy *C*-means Clustering with Prior Biological Knowledge. Biomedical Informatics 42, p. 74.

Ester, M., Kriegel, HP., Sander, J., Xu, X., 1996. "A density-based algorithm for discovering clusters in large spatial databases with noise," 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96). Portland, Oregon.

Jiang, D., Pei, J., Zhang, A., 2003. "DHC: a density-based hierarchical clustering method for time series gene expression data," 3rd IEEE Symposium on Bioinformatics and Bioengineering. Buffalo, New York.

Bohm, C., Kailing, K., Kriegel, H., Kroger, P., 2004. "Density connected clustering with local subspace preferences," 4th IEEE International Conference on Data Mining (ICDM '04). Washington, DC, USA.

UCI Machine Learning Repository, http://archive.ics.uci.edu/ml/datasets.html.

Du, Z., Wang, Y., Ji, Z., 2008. *PK*-means: A New Algorithm for Gene Clustering. Computational Biology and Chemistry 32, p. 243.

Viswanath, P., Pinkesh, R., 2006. "I-DBSCAN: a fast hybrid density based clustering method," 18th International Conference on Pattern Recognition (ICPR-2006). Hong Kong, China.

Ren, Y., Liu, X., Liu, W., 2012. DBCAMM: A Novel Density based Clustering Algorithm via using the Mahalanobis Metric. Applied Soft Computing 12, p. 1542.

Liu, P., Zhou, D., Wu, N., 2007. "VDBSCAN: varied density based spatial clustering of applications with noise," International Conference on Service Systems and Service Management. Shanghai, China.

Huang, M., Bian, F., 2009. "An improved density-based spatial clustering algorithm based on key factors of object's distribution," International Joint Conference on Artificial Intelligence (JCAI '09). Washington, DC, USA.

Bicici, E., Yuret, D., 2007. "Locally scaled density based clustering," 8th International conference on Adaptive and Natural Computing Algorithms (ICANNGA '07). Warsaw, Poland.

Liu, M., Jiang, X., Kot, AC., 2009. A Multi-Prototype Clustering Algorithm. Pattern Recognition 42, p. 689.

Luo, T., Zhong, C., Li, H., Sun, X., 2010. "A multi-prototype clustering algorithm based on minimum spanning tree," 7th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2010). Yantai, China.

Khan, SS., Ahmad, A., 2004. Cluster Centre Initialization Algorithm for *K*-means Clustering. Pattern Recognition Letters 25, p. 1293.