Estimating Distance Threshold for Subspace Clustering

Bhagyashri Kelkar[1], Sunil Rodd[2], Umakant Kulkarni[3]

[1]*Sanjay Ghodawat University, India*
[2]*Gogte Institute of Technology, India*
[3] *Sri Dharmasthala Manjunatheshwara College of Engineering and Technology, Dharwad, India*

Abstract: With tremendous growth of internet based applications and data capturing techniques datasets with large dimensionality have become very common. Due to curse of dimensionality such high-dimensional datasets cannot be analyzed by traditional clustering methods. To overcome the curse, subspace clustering algorithms are proposed which identify clusters existing in subspaces of such datasets. However it has been observed that, these algorithms are highly sensitive to distance threshold parameter and improper tuning may result into false objects included in clusters, merging of many clusters into one or division of clusters into many parts. The paper proposes a novel technique to estimate distance threshold for subspace clustering automatically from input numerical data. The empirical results obtained on synthetic as well as real datasets highlight that the proposed method is highly effective in estimating the distance threshold for subspace clustering.

Keywords: Subspace clustering, Similarity threshold, Parameter estimation, Distance measures, High dimensional data, Clustering

I. INTRODUCTION

Clustering also referred as segmentation is a data mining technique intended to classify entities into groups. It is unsupervised method meaning no training data is available to guide the clustering process. It has many applications in business intelligence, pattern recognition, web search etc. Today, datasets having tens/ hundreds of dimensions are very common due to advancements in data collection and storage technology. Such data is referred as high dimensional data [1]. A high dimensional data is inherently sparse. Hence clustering based on complete set of dimensions is not possible due to curse of dimensionality [2]. All objects are at equal distance from each other in a dataset having high dimensionality. Subspace clusters are the clusters existing in subsets of dimensions (termed as subspaces) of a high dimensional data. Subspace clustering methods are extensions to feature selection techniques which identify only relevant attributes based on some criterion. Subspace clustering algorithms first identify the possible subspaces i.e. subsets of attributes important for clustering. Then a clustering algorithm works on the identified subspaces to find clusters. Fig. 1 displays 4 subspace clusters present in a dataset having 16 attributes and 18 objects.

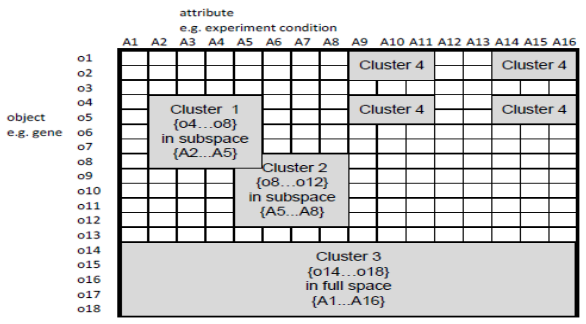

attribute
e.g. experiment condition
A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14 A15 A16
o1 o2 o3 o4 o5 o6 o7 o8 o9 o10 o11 o12 o13 o14 o15 o16 o17 o18
object
e.g. gene
Cluster 4
Cluster 4
Cluster 4
Cluster 4
Cluster 1 {o4…o8} in subspace {A2...A5}
Cluster 2 {o8…o12} in subspace {A5...A8}
Cluster 3 {o14…o18} in full space {A1...A16}

Fig. 1 Four clusters identified in subspaces of sixteen attributes and eighteen objects

Meaningful results can be obtained only if the density threshold and size of cells are properly tuned. Improper parameter values can result into many clusters being merged into one or single cluster partitioned into many clusters. In density based methods, a user specified value – the epsilon parameter specifies the radius of neighbourhood. All the objects which are within epsilon neighbourhood from an object are neighbours of that object. Another parameter minpts specifies the density threshold for identifying dense regions. A data item having dense neighbourhood is marked as core point. Dense regions are merged to form clusters. Thus the parameter – epsilon is very crucial in identifying dense regions. Advantage of density-based subspace clustering approach is it can identify arbitrary shaped clusters. This is one of the reasons for popularity of this approach for subspace clustering. The limitation of this approach is that it is not scalable due to excessive database scans. To speed up the subspace clustering process, several refinements to density-based techniques are proposed [3, 4, 5] which are highly scalable.

II. RELATED WORK

Subspace clustering is useful in business decision process, for product recommendations, social networking etc. Hence these algorithms have gained popularity in recent years. Search for subspaces can be done in bottom-up or top-down manner. Bottom up methods first identify lower dimensional subspace clusters and then merge them into higher dimensional clusters. Generally a pruning criterion is applied on lower dimensional subspace clusters to reduce computational complexity. CLIQUE is a grid based approach using bottom up strategy [6]. Each dimension is discretized into equal sized cells. The cells having strength of data points more than user specified unit selectivity threshold are marked dense. Adjacent dense units are joined to form higher dimensional subspace clusters. The algorithm is sensitive to positioning of cells. MAFIA is an improvement over CLIQUE by using adaptive grids [7]. ENCLUS [8] uses entropy criterion to identify candidate subspaces and to prune uninteresting subspaces. The authors in [10] use Chernoff-Hoeffding and notion of support to propose SCHISM that mines maximal subspaces. The dataset is converted into a vertical representation and maximal interesting subspaces are mined by using backtracking based depth first search approach. SUBCLU [9] is based on greedy bottom-up approach and finds subspaces which have axis-parallel orientation. As in DBSCAN, the process is controlled by two parameters - epsilon and minpts. Top-down approaches initially assume all points in one cluster and then remove irrelevant attributes iteratively to form low dimensional subspace clusters. FINDIT [12], PROCLUS [11], ORCLUS [13] are some of the top-down approaches.

Subspace clustering algorithms are also categorized as Hard or Soft clustering methods. Hard subspace clustering methods assume equal weightage to each dimension in the clustering process. Soft subspace clustering methods generalize hard approach to avoid loss of information due to noisy attributes. Soft subspace clustering algorithms build a non-binary relationship between attributes and clusters, by deciding importance of each attribute in forming a subspace cluster. Thus each dimension is part of each subspace cluster but has different significance factor. Some prominent examples are FSC [14], FWKM [16], and Attribute Weighting Algorithm (AWA) [17]. The authors in [18] propose CKS-EWFC-K and CKS-EWFC-F algorithms, which work by mapping feature space into the composite kernel space in order to cluster datasets having combinations of inner structures.

Irrespective of the approach followed by subspace clustering algorithms to identify dense regions, these methods are affected by common problems induced due to parameter setting [15]. As in traditional clustering, the process of subspace clustering is usually directed by input parameters such as: density threshold, cell granularity, duplication factor, minimum cluster size etc. values of which are accepted from the user. Many a times, the purpose and meaning of these parameters is non-obvious. The results are highly sensitive to the input parameters i.e. the quality of output and execution time is drastically affected by slight changes in the values. For example, in CLIQUE if the number of intervals is small, it results in increased cell width leading to inclusion of noise in the output and increased computational complexity. Conversely a larger interval causes lossy results. Optimal clustering results can be possible only after repeated trail runs every time with a new set of parameter values [21]. Table 1 shows a comparative chart of input parameters accepted by some prominent subspace clustering algorithms.

Clustering is generally a computation intensive task and on high dimensional data the complexity increases in three directions –in terms of count of embedded clusters, dimensions and objects. With improper parameter values, the task becomes practically infeasible in many cases. Subspace clustering directed by user specified parameter values dilutes the concept of unsupervised learning. Hence it is desirable to have parameter-free or if not possible at least parameter-light subspace clustering solutions for high dimensional datasets. Such algorithms can be designed by utilizing the knowledge hidden in the data itself and by avoiding human intervention.

III. PROPOSED METHOD

Similarity reflects the strength of relationship amongst two data items. Thus measurement of distances is mandatory in the clustering process. In case of numerical attributes, similarity/distance measurement is generally done by applying a distance measure such as Minkowski, Euclidean, Manhattan distance on the attribute values. Euclidean distance (straight line distance) is the most popular distance metric for numerical data. Let P= $(p_1, p_2,\ldots, p_d)$ and Q= $(q_1,q_2,\ldots,q_d)$ be two data points described by d numeric attributes. The Euclidean distance between A and B is defined as

$$\text{Distance}(P,Q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_d - q_d)^2}$$

Manhattan (city block) distance is another well known measure which measures distance between two objects in terms of blocks e.g. 4 blocks over and 3 blocks down. It is defined as

$$\text{Distance}(P, Q) = |p_1-q_1| + |p_2-q_2| + \ldots.. + |p_d-q_d|$$

A generalization of Manhattan and Euclidean distances is Minkowski distance defined as

$$\text{Distance}(P, Q) = \sqrt[h]{(p_1 - q_1)^h + (p_2 - q_2)^h + \ldots + (p_d - q_d)^h}$$

Conventional distance based clustering algorithms calculate similarity between objects over all dimensions. In case of high dimensional data, the similarity between objects is calculated over identified subspace i.e. on subset of attributes. In order to make the subspace clustering process less dependent on user expertise, it is essential to determine the distance threshold value automatically from the data to be clustered. As an attempt in this direction, a novel method is proposed in this paper. The method is based on greedy paradigm to estimate the distance threshold from the input data. It uses a bottom-up approach similar to CLIQUE to find subspace clusters. As in CLIQUE, the proposed algorithm first identifies dense regions in each dimension based on the distance threshold value it has computed. Then it builds higher dimensional clusters by merging these 1-dimensional clusters. It operates with default values of parameters which specify how coarse or fine the resulting clusters should be.

The steps followed by the algorithm are explained in Algorithm 1 below.

Table I Input parameters accepted by some prominent subspace clustering approaches

Clustering Approach	Input Parameters
CLIQUE	No. of Intervals and Unit Selectivity Threshold
ENCLUS	Entropy Threshold, Interest gain Threshold
PROCLUS	Average Dimensions count, Clusters Count
MAFIA	Cluster Dominance Factor
DOC	Size Of Grid, Density Threshold, Balance Factor Between Points and Dimensions
DENCOS	Equal Length Intervals, Unit Strength Factor, Maximum Subspace Cardinality
DUSC	Density Threshold
DENCLU	Density Threshold and Neighbourhood Radius
OPTIGRID	Density Threshold and Neighbourhood Radius
SUBCLU	Density Threshold and Neighbourhood Radius
FIRES	Density Threshold and Neighbourhood Radius
DiSH	Density Threshold and Neighbourhood Radius
PreDeCon	Density Threshold and Neighbourhood Radius, two Preference Parameters
INSCY	Neighbourhood Radius and Density Threshold, Redundancy Factor

Algorithm 1. //Proposed Algorithm to find distance threshold separately for each dimension

Input: Data set of containing n objects each having d dimensions (attributes)

Output: Distance threshold vector Dist[] containing thresholds for each of d dimensions

Pseudo code:

```
for i  = 1 to d

        Sort unique values in i^th dimension in non-decreasing order to form 1-dimensional vector V.
        Avg[1: n-4] = 0
        Dist [1:d]= ∞
        for j = 1 to ( n – 4 )
                sum_of_distance = 0
                for k = 1 to 4
                        sum_of_distance = sum_of_distance + (v[j+k] - v[j+k-1] )
                Avg[j]=sum_of_distance/4

        Least_avg_region=Find location of least element of vector Avg//Find the densest 5 elements
                                                                    // group of objects
        Dist[i]=maximum of distance of consecutive elements in the group pointed by
              Least_avg_region
        Round Dist[i] to nearest decimal value
        // Dist[i] is the distance threshold for dimension i
```

Step I: Estimation of distance threshold separately for each attribute - In this step, objects in each attribute are arranged in non-decreasing order. Then a closely packed 5 elements region in it is identified and in case of tie, it is resolved randomly. The window size is set to five elements because subspace clusters having less than five objects are assumed to be non-significant [19]. The maximum separation between two consecutive elements in the group is rounded to next decimal point and is recorded as the distance threshold for that dimension.

Step II: Formation of 1-dimensional subspace clusters in each attribute - In this step, dense regions of objects in each dimension are found based on distance threshold value identified in Step I. Initially each object in the dimension is placed in a separate cluster. If the distance between any one object from a cluster and any one object in another cluster is within the distance threshold then the two clusters are merged. The process of merging two clusters is repeated until no new clusters can be formed. This step works in a way similar to single linkage clustering. The 1-dimensional clusters which are having density of objects less than object_density_threshold are marked non-significant and pruned from further processing. The default value of object_density_threshold is set to 5.

Step III: Formation of higher dimensional subspace clusters - Higher dimensional clusters are formed by connecting 1-dimensional subspace clusters sharing the same objects. If an outlier object is by chance becomes part of a one dimensional cluster, it will be absent in clusters present in remaining dimensions and its support will be below attribute_support_threshold. Such objects get eliminated in this step. Value of k is different for every subspace cluster and if it is less than attribute_support_threshold, those clusters are pruned from further processing. The default value of attribute_support_threshold is set 5.

Step IV: Removal of redundant clusters - A subspace cluster is dispensable if it is subset of a larger subspace cluster. In step III many redundant clusters may be formed. In Step IV, redundant clusters are removed by applying an algorithm proposed in [20].

IV. EMPIRICAL EVALUATION

The performance of the proposed method on synthetic and real numeric datasets is discussed in this section. Execution time and quality of output is compared with other well known subspace clustering algorithms.

A. Experiment environment

Experiments were performed on a computer having Intel(R) Pentium® P6200 CPU @ 2.13 GHz, 2.00 GB RAM, R version 3.4.3, Windows 7 OS.

B. Evaluation measures

Subspace clustering algorithms require evaluation measures different from conventional clustering algorithms. As mentioned in [21], if the input data has n objects and d dimensions then a subspace cluster is a collection of subset of sub-objects i_{jk}, 1<= j <= n, 1 <= k <= d. Hence the proposed algorithm is evaluated in terms of object based measures - F-measure, accuracy and object and subspace based measures - Relative Non Intersecting Area (RNIA) and Clustering Error (CE). The scalability of the algorithm is tested in terms of execution time.

Precision: A high precision indicates that most of the sub-objects in the output cluster are true sub-objects and there are very few false sub-objects. Optimal value of precision is 1.0.
Precision = True Positives / (True Positives + False Positives)

Recall: A high recall indicates that the identified clusters cover a large fraction of the true sub-objects. Optimal value of recall is 1.0.
Recall = True Positives / (True Positives + False Negatives)

F1-measure: F1-measure represents the balance between recall and precision, i.e. extent of conformity of output to the true clusters and to what extent the algorithm is able to exclude false results. Optimal value of F1-measure is 1.0. If the true result contains m subspace clusters, the F1 value is calculated obtained as follows:

$$\mathrm{F1} = \sum_{i=1}^{m} \left(\frac{(2 * \mathrm{Precision(i)} * \mathrm{Recall(i)})}{(\mathrm{Precision(i)} + \mathrm{Recall(i)})} \right)$$

Accuracy: Accuracy is the measure of the extent to which the algorithm is able to mark true sub-objects as part of output clusters and separate true outliers, out of all available sub-objects. Optimal value of accuracy is 1.0.
Accuracy = (True Positives + True Negatives) / (count of all sub-objects)

RNIA: The subspace clustering quality is high if it covers all and only true sub-objects and does not include objects not supposed to be part of any subspace cluster. This aspect is measured by Relative nonintersecting area (RNIA) measure. Let U indicates count of objects in union of true and output clusters. Let I indicates count of objects common to true and output clusters. Then RNIA = (U - I)/U. Optimal value of RNIA is 0.0.

CE: RNIA measure does not reflect the case when a true cluster is partitioned into several small clusters in the output or several true clusters are merged to form an output cluster. CE measure reduces the clustering quality value in such cases. Here a mapping of true and output clusters is first generated. Let U indicates count of objects in union of true and output clusters. Let I’ indicates count of objects common to mapped and output clusters. Then CE = (U – I’)/U. Optimal value of CE is 0.0.

C. *Experimental Results*

1) *Results on synthetic data*

For evaluation of the proposed method, a synthetic numerical, real valued data generator was implemented in R. The generator accepts description of the dataset from the user such as count of objects and dimensions, maximum and minimum values in all attributes, the count of embedded subspace clusters, count of objects and dimensions in each embedded subspace cluster, noise percentage and the standard deviation of values in subspace clusters. Advantage of using synthetic data for the experiments is that, true class of each sub-object is known to which the output can be compared. Using the data generator, five synthetic datasets with n=1000 objects were generated each having 5 subspace clusters containing 10 objects and 10 attributes randomly embedded in the data. The dimensionality of the five datasets was 100, 200, 300, 400 and 500 respectively. An implementation of CLIQUE, FIRES [3], P3C [22], PROCLUS [11] and SUBCLU is available in package 'Subspace' of R [21]. The parameter setting used during the experiments for FIRES, PROCLUS and SUBCLU was as provided in the package. The setting was changed for CLIQUE and P3C as shown in Table 2 for getting some output, as the default values could not produce any result. The proposed algorithm was executed with default values of object_density_threshold and attribute_support_threshold set to 5. A comparison of execution time and quality of the output of the proposed method with abovementioned subspace clustering algorithms is presented in Fig. 2.

Result Analysis

As shown in Fig. 2(a), the clustering error of the proposed method is 0.06 for synthetic dataset with 400 dimensions and 100 objects and in rest of the cases it is 0 which is the optimum. Other algorithms show clustering error more than 0.98. This is because the experiments were done with default values of parameter specified in the package 'Subspace'. As highlighted in [21], getting values of optimal parameter setting requires repeated runs of these algorithms every time with a new set of values. Against to this, the proposed algorithm could get optimal results in single run, without accepting any input from the user. In case of RNIA measure as shown in Fig. 2(b), the same observation holds true and RNIA value of the clustering produced by the algorithm is near optimal. Fig. 2(c) highlights that, F1 value of the proposed method is more than 0.96 for d=400 and in other cases it is 1 which is again the optimal value. CLIQUE could produce F1-value upto 0.3 and for rest of the algorithms it is less than 0.11. As shown in Fig. 2(d), accuracy of the proposed method is optimal i.e. 1.0 in all cases except for d=400 where it is 0.94. CLIQUE shows maximum accuracy as 0.59 on d=500 and FIRES shows 0.78 on d=300 which is better than remaining algorithms. Table 3 displays the time taken in seconds for the processing by each of the algorithms. As indicated in the table, the proposed method has the best execution time in the group for all cases. The results show that, the algorithm produces near optimal results on synthetic datasets. The proposed method is highly scalable and shows negligible increase in runtime with increased dimensionality. The optimal results are outcomes of accurate distance threshold estimation.

2) *Results on real data*

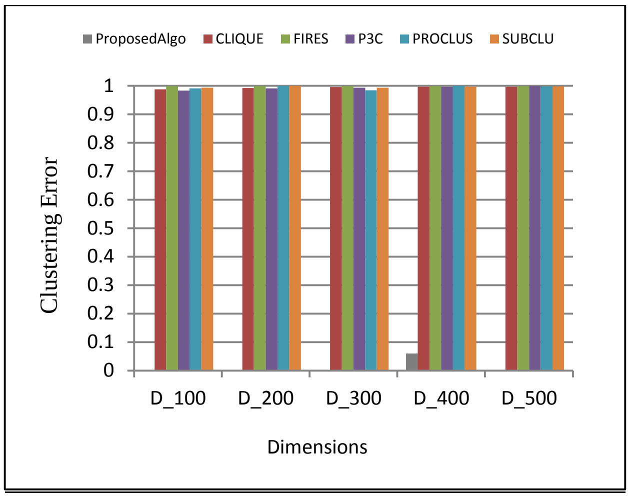
Real datasets available on UCI machine learning repository [23] were used for evaluating the performance of the proposed algorithm in comparison with other algorithms mentioned in Table 2. The parameter setting for all other algorithms is as specified in Table 2. Default values of object_density_threshold and attribute_support_threshold were set to 5. Table 4 gives the specification of the datasets used in the study.

Result Analysis

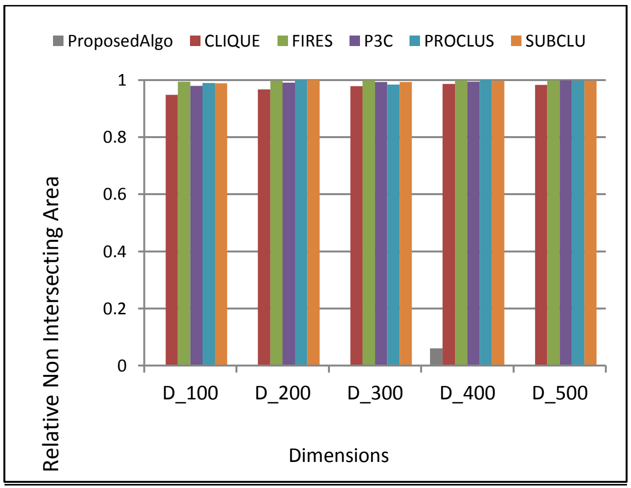
Fig. 3(a) shows the clustering error on various real datasets by each of the algorithms. The clustering error of the proposed method is the minimum compared with other algorithms. RNIA value of FIRES is found to be the least in the group followed by RNIA of the proposed algorithm. Accuracy of FIRES is found to be good, followed by the proposed method. F1 value of the proposed method is the highest in the group. Table 5 shows the comparison of processing time on real datasets. The table reflects that the proposed algorithm shows the least execution time in the group in most of cases.

Table II Parameter setting for subspace clustering algorithms

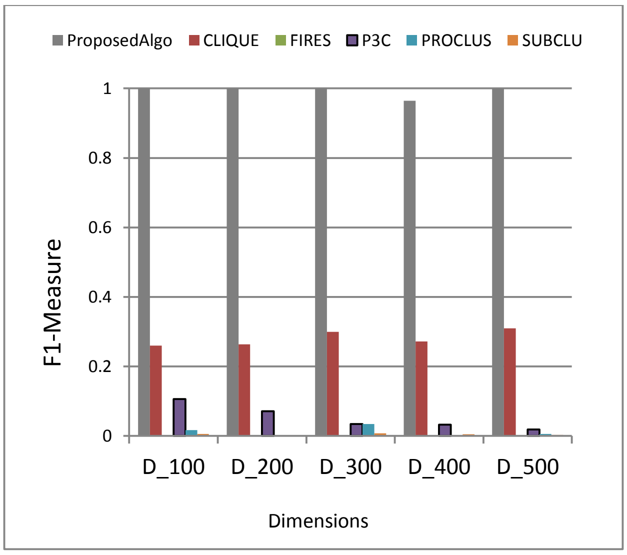
Algorithm	**Parameter Settings**
CLIQUE	xi = 50, tau = 0.03
FIRES	base_dbscan_minpts = 4,base_dbscan_epsilon = 1, k = 1, minimumpercent = 25, minclu = 1, mu = 1, post_dbscan_epsilon = 1, split = 0.66, post_dbscan_minpts = 1
P3C	ChiSquareAlpha = 0.50, PoissonThreshold = 2
PROCLUS	k=12, d=2.5
SUBCLU	epsilon = 1, minSupport = 5



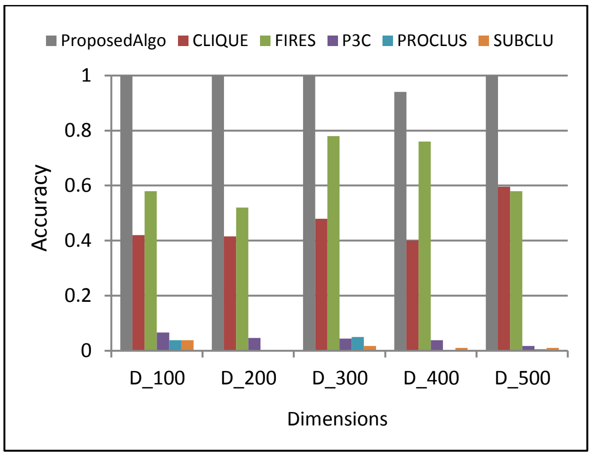
(a)



(b)



(c)



(d)

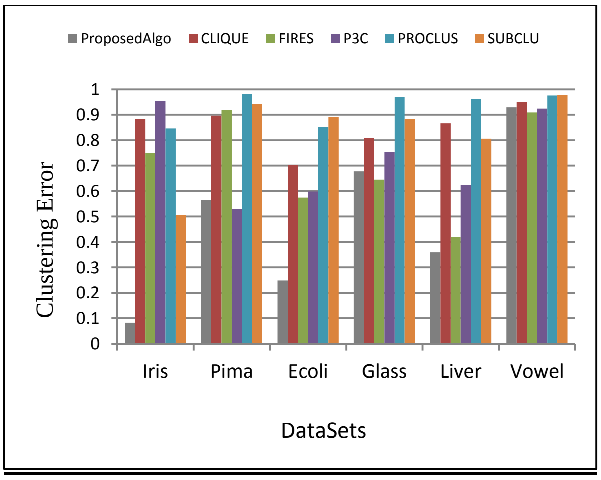
Fig. 2 Performance on synthetic data

Table III. Execution Time in seconds on synthetic datasets having1000objects

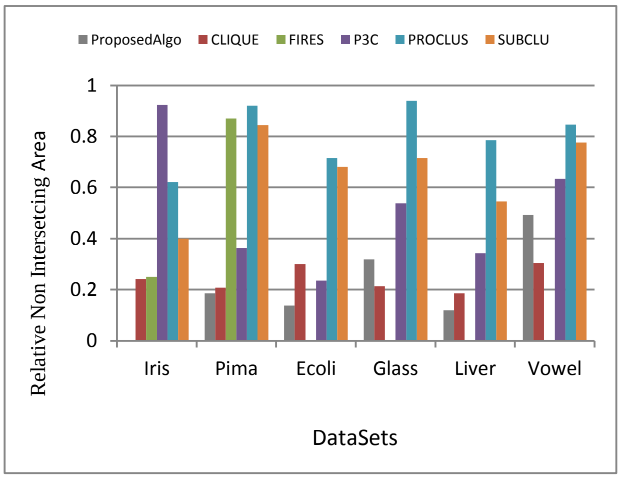
	Dim. 100	Dim. 200	Dim. 300	Dim. 400	Dim. 500
Proposed Algorithm	0.57	1.37	1.76	2.79	3.46
CLIQUE	0.58	1.65	7.49	14.07	25.15
FIRES	40.34	246.33	520.78	1757.3	5262.71
P3C	334.37	648.1	2502.03	2768.17	1888.22
PROCLUS	24.23	25.18	54.48	244.5	123.56
SUBCLU	4.74	37.05	62.8	156.86	74.18

Table IV Description of real datasets

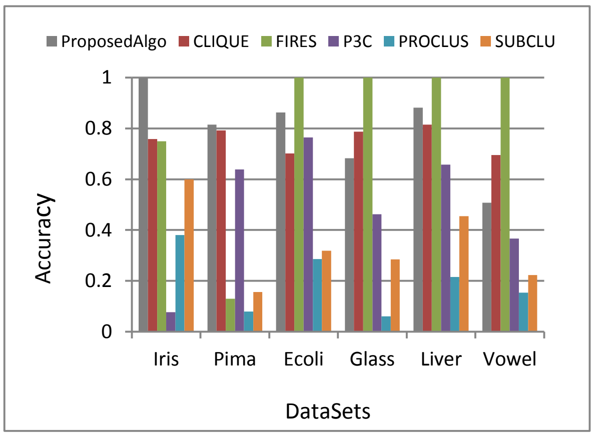
Dataset	Classes in the data	Attributes in the data	Instances in the data
Iris	3	4	140
Pima	2	8	768
Ecoli	8	7	336
Liver	2	6	345
Glass	6	9	214
Vowel	11	13	990



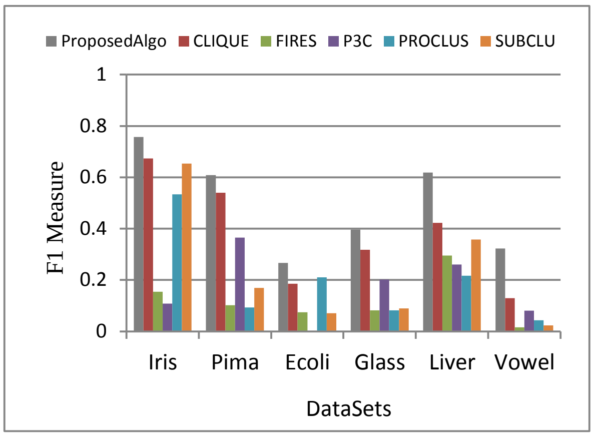
(a)



(b)



(c)



(d)

Fig. 3 Performance on real datasets

Table V Execution Time on Real Datasets in Seconds

	Data Iris	Data Pima	Data Ecoli	Data Glass	Data Liver	Data Vowel
Proposed Algorithm	0.01	0.11	0.03	0.11	0.03	0.61
CLIQUE	0.05	0.25	0.20	0.21	0.06	1.05
FIRES	0.05	0.20	0.08	0.05	0.06	1.17
P3C	0.03	1.06	0.14	0.18	0.07	43.89
PROCLUS	0.05	0.16	0.10	0.05	0.06	0.47
SUBCLU	0.01	0.11	0.03	0.04	0.05	0.25

V. CONCLUSION

The working of clustering algorithms is generally controlled by set of input parameters which act as threshold values to control unwanted results from the output. However, it has been observed that, improper tuning of parameters highly affect efficiency and quality of the clustering results. Parameter-based execution limits the ability of a clustering algorithm to reveal interesting and novel knowledge due to constraints put by the parameter values which are based on knowledge of the operator. A novel method to estimate an important input parameter- the distance threshold is proposed in this paper and it is subsequently used to find subspace clusters. Experimental evaluation on real and synthetic datasets show that the proposed method is able to estimate distance threshold accurately which results in high quality output in terms of accuracy and F1-measure. The method also proves its effectiveness in terms of subspace and object based evaluation measures - clustering error and RNIA. The algorithm shows linear execution time on real and synthetic datasets. Future direction in this field could be estimation of other important parameters such as density threshold.

VI. REFERENCES

[1] D. Lee and Junho, Shim, Impact parameter analysis of subspace clustering, *International Journal of Distributed Sensor Networks*, (2015).

[2] Bellman, R., *Adaptive Control Processes: A Guided Tour*, Princeton University Press, Princeton, New Jersey, (1961).

[3] H.-P. Kriegel, P. Kröger, M. Renz, and S. H. R. Wurst. A generic framework for efficient subspace clustering of high-dimensional data. In *Proceedings of the 5th IEEE International Conference on Data Mining*, pages 250–257, Houston, TX, 2005.

[4] I. Assent, R. Krieger, E.l Müller, and T. Seidl. DUSC: Dimensionality unbiased subspace clustering. In *Proceedings of the 7th IEEE International Conference on Data Mining*, pages 409–414, Omaha, NE, 2007.

[5] E. Müller, I. Assent, R. Krieger, S. Günnemann, and T. Seidl. DensEst: Density estimation for data mining in high dimensional spaces. In *Proceedings of SIAM International Conference on Data Mining*, pages 175–186, Sparks, NV, 2009.

[6] Agrawal, R et al., Automatic subspace clustering of high dimensional data for data mining applications, In: Proc. of the *ACM SIGMOD International conference on Management of data, Seattle, WA, USA,* (1998), pp. 94–105.

[7] Goil, S et al., *Mafia: Efficient and scalable subspace clustering for very large data sets*, Technical Report CPDC-TR-9906-010, Northwestern University, (1999).

[8] C.-H. Cheng, A. W. Fu, and Y. Zhang. 1999 Entropy-based subspace clustering for mining numerical data". In: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 84–93.

[9] Kr¨oger, P. et al., Density-Connected Subspace Clustering for High-Dimensional Data, In: *Proceedings of SIAM Int. Conf. on Data Mining (SDM'04)*, (2004), pp. 246–257.

[10] K. Sequeira and M. Zaki, 2004. SCHISM: A new approach for interesting subspace mining. In ICDM, pages 186–193.

[11] Aggarwal, C. C. et al., Fast algorithms for projected clustering, In Proceedings of the *ACM International Conference on Management of Data (SIGMOD), Philadelphia, PA,* (1999), pp. 61–72.

[12] Woo, K. G. and Lee, J.H., FINDIT: A Fast and Intelligent Subspace Clustering Algorithm using Dimension Voting, *Information and Software Technology*, Vol-46 Issue 4, (2002), pp. 255-271.

[13] Aggarwal, C. C. and Yu, P. S., Finding generalized projected clusters in high dimensional spaces, In: Proceedings of the *ACM SIGMOD International conference on Management of data*, *Dallas, TX, USA*, (2000) , pp. 70–81.

[14] G. Gan, J. Wu, A convergence theorem for the fuzzy subspace clustering (FSC) algorithm, Pattern Recognition, Vol. 41, 2008, 1939-1947

[15] D. Lee and Junho, Shim, Impact parameter analysis of subspace clustering, *International Journal of Distributed Sensor Networks*, (2015).

[16] L. Jing, M.K. Ng, J. Xu, J.Z. Huang, Subspace clustering of text documents with feature weighting k-means algorithm, Proc. 9th Pacific-Asia Conf. on Knowl. Disc. and Data Mining (PAKDD05), Vietnam, 2005, pp. 802–812

[17] Chan, E.Y., Ching, W.K., Ng, M.K., Huang, Joshua Z, An optimization algorithm for clustering using weighted dissimilarity measures, *Pattern recognition* 37 (5), (2004), pp. 943–952.

[18] Jun, Wang, Deng, Zhaohong, Choi, Kup-Sze, Jiang, Yizhang, Luo, Xiaoqing, Chung, Fu-Lai, Shitong, Wang, Distance metric learning for soft subspace clustering in composite kernel space. *Pattern Recognition* 52 (C), (2016), pp. 113–134.

[19] Zhang, H., Tang, Y. et al., A novel subspace clustering method based on data cohesion model, in *Optik - International Journal for Light and Electron Optic*, vol. 127, issue 20, (2016), pp. 8513-8519.

[20] Kuhn, H.W., The Hungarian Method for Assignment Problems, *Naval Research Logistic Quarterly* 2, (1955), pp. 83–97.

[21] Müller, E. et al., Evaluating Clustering in Subspace Projections of High Dimensional Data, *In Proc. 35th International Conference on Very Large Data Bases (VLDB 2009), Lyon, France*, (2009), pp. 1270-1281.

[22] Gabriela Moise, Jörg Sander and Martin Ester P3C: A Robust Projected Clustering Algorithm In Proc. 6th IEEE International Conference on Data Mining (2006).

[23] UCI Machine Learning Repository. [Online] http://archive.ics.uci.edu/ml/.