Automatic Clustering Based on Cluster Nearest Neighbor Distance (CNND) Algorithm

Arghya Sur[1], Aritra Chowdhury[1], Jaydeep Ghosh Chowdhury[1], and Swagatam Das[2]

[1] Dept. of Electronics and Telecomunication Engg, Jadavpur University, Kolkata 700032, India
[2] Electronics and Computer Sciences Unit, Indian Statistical Institute, Kolkata, India
{arghyasur1991,jaydeep197}@gmail.com, arit0001@yahoo.co.in,
swagatam.das@isical.ac.in

Abstract. This article describes a simple and fast algorithm that can automatically detect any number of well separated clusters, which may be of any shape e.g. convex and/or non-convex. This is in contrast to most of the existing clustering algorithms that assume a value for the number of clusters and/or a particular cluster structure. This algorithm is based on the principle that there is a definite threshold in the intra-cluster distances between nearest neighbors in the same cluster. Promising results on both real and artificial datasets have been included to show the effectiveness of the proposed technique.

Keywords: Cluster nearest Neighbour, Clustering, Automatic Clustering, Various shaped clusters.

1 Introduction

Clustering is the process, by which a set of objects are partitioned into a number of groups, such that the similarity between objects of the same group is maximum and that between objects belonging to different groups are minimum. This measure of similarity is defined mathematically and, the objects are assigned to these groups, known as 'clusters' based on this measure. In the past few decades, cluster analysis has played a central role in diverse domains of science and engineering [1].

An important consideration with clustering algorihms is to determine the appropriate number of clusters from a given dataset, where the number of groups in the dataset is unknown *apriori.* In many clustering algorithms, this number is specified by the user, and accordingly the algorithms partition the dataset. It is a challenge to find the optimal number of clusters automatically.

Clustering algorithms can be hierarchical or partitional [2]. Hierarchical clustering algorithms recursively find nested clusters either in agglomerative mode (starting with each data point in its own cluster and merging the most similar pair of clusters successively to form a cluster hierarchy) or in divisive (topdown) mode (starting with all the data points in one cluster and recursively dividing each cluster into smaller clusters). Partitional clustering algorithms, on the other hand, attempts to partition the data set directly into a set of disjoint clusters by trying to optimize certain criteria (e.g. a squared-error function).

S.C. Satapathy et al. (Eds.): Proc. of Int. Conf. on Front. of Intell. Comput., AISC 199, pp. 189–196.
DOI: 10.1007/978-3-642-35314-7_22

Centroid based clustering algorithms are a class of partitional algorithms , such as K-means and fuzzy c-means. These algorithms attempt to iteratively find cluster centers and assign points to one of the centers. Even though K-means was first proposed over 50 years ago, it is still one of the most widely used algorithms for clustering. They require the number of clusters beforehand and hence are non-automatic. Also, they only detect spherical and similar sized clusters accurately due to the "uniform effect" [3].

In this article, we propose a simple and fast algorithm to automatically detect the appropriate number of clusters from an unlabelled dataset. The algorithm performs equally well towards both hyper-spherical shaped clusters and shell-type or solid clusters of any arbitrary shape. This algorithm is based on the principle that there is a definite threshold in the intra-cluster distances between nearest neighbors in the same cluster. Using this threshold, we can group together objects in the dataset which have their cluster nearest neighbor distance less than the threshold.

2 Proposed Algorithm

In this algorithm, we use the Euclidean Distance as distance measure. The basic idea is that, if a subset of points in the dataset belongs to the same cluster, then, if the nearest neighbor of that subset in the set of remaining un-clustered data points (points which have not been assigned to any cluster as yet) is not in that cluster, the cluster consists only of that subset of points and nothing else. Nearest neighbor of a set S' in a set S is defined as the point in S-S', whose distance from its nearest neighbor in S' is minimum. When S' is the current cluster, and S is the set of remaining un-clustered data points, we call this point as the cluster nearest neighbor (CNN) and the distance as, cluster nearest neighbor distance (cnn_dist). This algorithm runs iteratively over all points in the dataset forming clusters along the way.

First, an initial starting point in a new cluster C is randomly initialized in the set of remaining un-clustered data points. Let this set be denoted by S_REM. Then the nearest neighbor of that point in S_REM is calculated using a standard nearest neighbor search algorithm like k-d tree [4] and kNN search. Here, we have used kNN search algorithm for this purpose. Thus, the current cluster C consists of these two points now. Then, iteratively, we find the CNN of C and determine if the CNN should be in C or not. The cluster completes when either the CNN is found not to belong in C, or when S_REM becomes empty. The condition to determine if CNN should belong to C depends on the number of points in C ($clus\_len$) and a threshold. Let POP be the set of all points in the dataset and min $\_len$ be the minimum number of elements that must always be present in a cluster. Then the following conditions determine if CNN of C belongs to C.

Cond1: If $length(C) <= min\_len$,

CNN should belong to C, if nearest neighbor distance of C in POP-C is equal to cnn_dist. Else, get the cluster C', which contains the nearest neighbor of C in POP-C.

Cond2: Else,

CNN should belong to C, if $cnn\_dist <= threshold * mean(D)$, where D is the set of cnn_dist of all previous points in the current cluster and threshold is a parameter initialized externally. For our experiments, we have initialized $threshold = 1.2$.

The complete algorithm is as follows:

1. Initialize min_len=$ceil(1\%\ of\ length(pop))$ where, $length(S)$ is the number of elements in set S. Ceil function gives the smallest integer greater than or equal to the argument.
2. Initialize $i = 1$.
3. Initialize a starting point for the i^{th} cluster C_i. This can be done randomly from S_REM. Initialize $D = \emptyset$.
4. Remove that point from S_REM.
5. If S_REM not equal to $\emptyset$, Find CNN and cnn_dist of C_i. Else, *stop*.
6. Take $C = C_i$ and determine if CNN of C should belong to C according to conditions Cond1 and Cond2.
7. If $length(C) <= min\_len$ and CNN shouldn't belong to C, merge C' and C and let the merged cluster be C. This is to ensure that no fewer than min_len elements exist in a cluster.
8. If CNN shouldn't belong to C, set $min\_len = ceil(1\%\ of\ length(C))$. Go to step 9. Else, set $C = C \cup \{CNN\}$. Set $C_i = C$. Go to step 5.
9. Set $i = i + 1$. Adapt threshold by the formula $threshold_i = mean(T, cnn\_dist)$, where T is the set of $threshold_j,\ j = 1\ to\ i - 1$. If $length(S\_REM) > 0$, go to step 3. Else, go to step 5.

3 Experimental Results

3.1 Datasets

- **Artificial Datasets:**
  - *i.* *Dataset1:* This dataset consists of 788 points as represented Fig. 1(a). There are a total of 7 groups in the population of varying shapes and size.
  - *ii.* *Dataset2:* This spiral dataset consists of 312 points as represented Fig. 1(b). There are 3 groups in the population.
  - *iii.* *Dataset3:* This is a dataset consisting of 240 points spread over two groups as shown in Fig. 1(c).
  - *iv.* *Dataset4:* This is a dataset with 1000 points with two clusters- an ellipse and a circle of different radii. This is shown in Fig. 1(d).
  - *v.* *Dataset5:* This is a 3-d dataset with 500 points. There are two groups, one, a shell-shaped cluster and another, a vertical ellipsoid. The 3d representation is shown in Fig. 1(e).

- **Real life Datasets:** These 3 real-life datasets are obtained from [5].
  - *i.* *Wine Dataset:* The Wine recognition dataset consists of 178 instances having 13 features obtained from a chemical analysis of wines. The wines were grown in the same region in Italy but were derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines.
  - *ii.* *Iris Dataset:* This dataset consists of 150 points equally distributed over 3 groups, viz. *Setosa*, *Versicolor* and *Virginica*. It is represented by 4 feature values. *Versicolor* and *Virginica* overlap while *Setosa* can be linearly separated.
  - *iii.* *LiverDisorder:* This dataset consists of 345 instances with each having six features. There are 2 groups in the dataset.

The artificial datasets are shown in figure 1(a) to (e). Datasets 1-4 are 2-d while Dataset5 is a 3-d dataset.

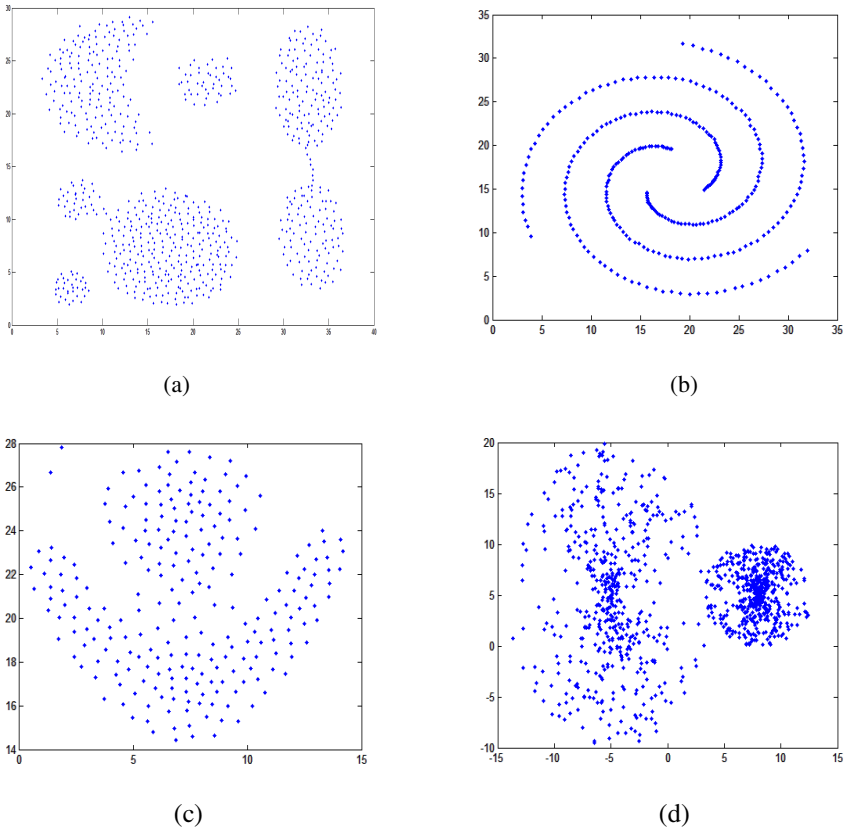
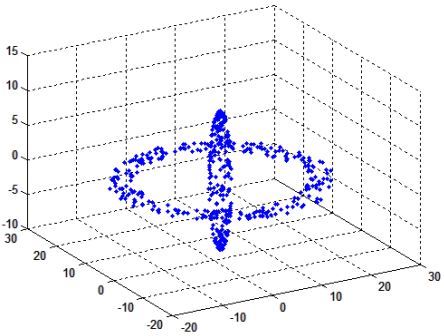


Fig. 1. (a)Dataset1 (b)Dataset2 (c)Dataset3 (d)Dataset4 (e)Dataset5

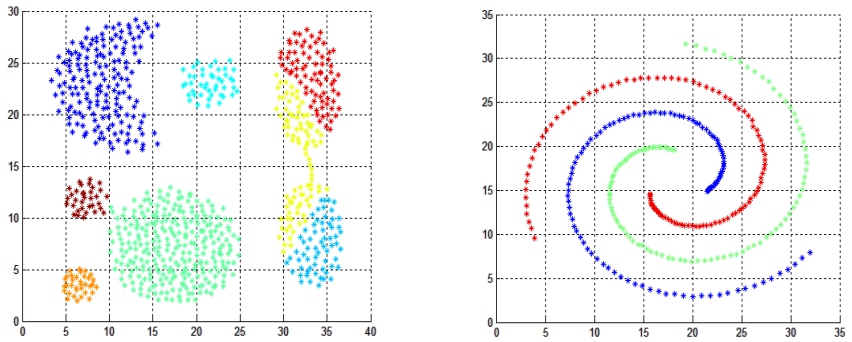


(e)

Fig. 1. (*continued*)

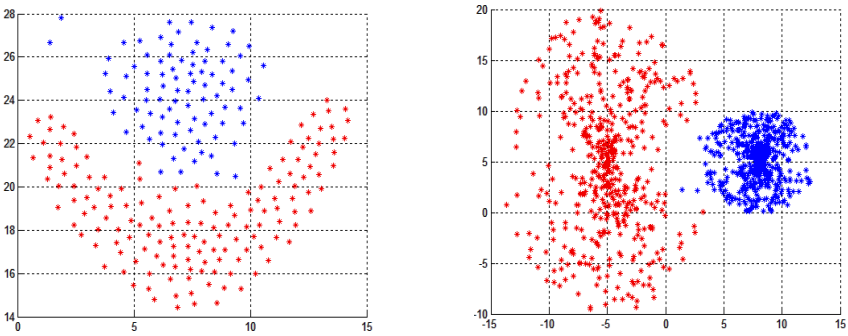
3.2 Results

This section compares the performance of the proposed clustering algorithm with a few standard clustering algorithms like k-means [6], fuzzy C-means [7] and Hierarchical Agglomerative Clustering. The contestant algorithms were mostly non-automatic. K-means and fuzzy c-means require the number of clusters beforehand while hierarchical clustering requires the maximum number of possible clusters. The figures 2(a)-(e) are the clustered representations of the figures 1(a)-(e) obtained by CNND clustering algorithm. These sets of data are used in order to represent all kinds of data adequately.

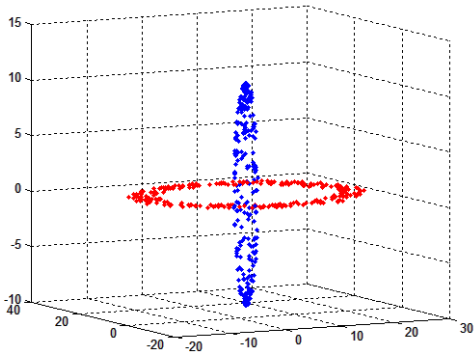


(a) (b)

Fig. 2. (a) Clustered Representation of Dataset1 (b) Clustered Representation of Dataset2 (c) Clustered Representation of Dataset3 (d) Clustered Representation of Dataset4 (e) Clustered Representation of Dataset5



(c) (d)



(e)

Fig. 2. (*continued*)

3.3 Minkowski Score

If T is the "true" solution and S is the solution obtained experimentally, then the Minkowski score [8] is defined as,

$$MS(T,S) = \sqrt{\frac{n_{01}+n_{10}}{n_{11}+n_{10}}}, \tag{1}$$

where, n_{01} represents the number of pairs of elements that are in the same cluster only in S, n_{10} represents the number of pairs of elements that are in the same cluster only in T and n_{11} represents the number of pairs of elements that are in the same cluster in both S and T.

The minkowski scores of the datasets obtained for four algorithms are shown in Table 1. The actual no. of clusters (AC) versus the obtained number of clusters (OC) in CNND algorithm is shown in Table 2. All other algorithms, being non-automatic, AC and OC are same trivially.

Table 1. Minkowski Scores of all the experimental datasets

Data Set	CNND clustering	K-means	Fuzzy C-means	Hiererchical
Dataset1	**0.3567**	0.6561	0.6129	0.5859
Dataset2	**0**	1.16	1.16	**0**
Dataset3	**0**	0.7142	0.6913	0.9259
Dataset4	**0**	0.218	0.2512	1
Dataset5	**0**	1	1	**0**
Wine Dataset	**0.5844**	0.9124	0.9255	1.373
Iris Dataset	**0.5309**	0.6047	0.6047	0.8241
Liver Disorder	**0.9786**	0.9846	0.9891	**0.9786**

Table 2. Number of actual clusters(AC) vs Number of obtained Clusters(OC) for CNND Algorithm

Data Set	AC	OC
Dataset1	7	8
Dataset2	3	**3**
Dataset3	2	**2**
Dataset4	2	**2**
Dataset5	2	**2**
Wine Dataset	3	**3**
Iris Dataset	3	4
Liver Disorder	2	**2**

4 Conclusion

From Table 1, we can see that CNND algorithm beats all the other clustering algorithms in terms of minkowski score. Also, Minkowski score of 0 obtained in 4 out of 5 artificial datasets clearly show that this algorithm is robust towards different cluster shapes and sizes, whereas k-means and fuzzy c-means give good results only in case of spherical clusters of nearly equal sizes. Hierarchical clustering is also robust towards different shaped clusters but in closely spaced or semi-overlapping clusters, it fails to give good clustering results. In this area too, however, our CNND algorithm shows quite promising results. Table 2 shows that it also finds the appropriate number of clusters in a dataset with a good accuracy. However in completely overlapped clusters, this algorithm will fail to give accurate results because it depends on finding the boundary between two clusters.

References

[1] Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice-Hall, Englewood Cliffs (1988)
[2] Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Computing Surveys 31(3), 264–323 (1999)
[3] Xiong, H., Wu, J.J., Chen, J.: K-means clustering versus validation measures: A data-distribution perspective. IEEE Trans. Systems, Man, and Cybernetics-Part B: Cybernetics 39(2), 318–331 (2009)
[4] Saha, S., Bandyopadhyay, S.: A symmetry based multiobjective clustering technique for automatic evolution of clusters. Pattern Recognition 43, 738–751 (2010)
[5] Frank, A., Asuncion, A.: UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine, CA (2010), http://archive.ics.uci.edu/ml
[6] Jain, A.K.: Data clustering: 50 years beyond K-means. Pattern Recognition Lett. (2009), doi:10.1016/j.patrec.2009.09.011
[7] Cannon, R.L., Dave, J.V., Bezdek, J.C.: Efficient implementation of the fuzzy c-means clustering algorithms. IEEE Trans. Pattern Analysis and Machine Intelligence PAMI-8, 248–255 (1986)
[8] Ben-Hur, A., Guyon, I.: Detecting Stable Clusters Using Principal Component Analysis in Methods of Molecular Biology. Humana Press (2003)