

K-means algorithm with a novel distance measure

Shadi I. ABUDALFA,* Mohammad MIKKI

Department of Computer Engineering, The Islamic University of Gaza, Gaza City, Palestine

Received: 26.10.2010 • Accepted: 24.04.2012 • Published Online: 02.10.2013 • Printed: 28.10.2013

Abstract: In this paper, we describe an essential problem in data clustering and present some solutions for it. We investigated using distance measures other than Euclidean type for improving the performance of clustering. We also developed an improved point symmetry-based distance measure and proved its efficiency. We developed a k-means algorithm with a novel distance measure that improves the performance of the classical k-means algorithm. The proposed algorithm does not have the worst-case bound on running time that exists in many similar algorithms in the literature.

Experimental results shown in this paper demonstrate the effectiveness of the proposed algorithm. We compared the proposed algorithm with the classical k-means algorithm. We presented the proposed algorithm and their performance results in detail along with avenues of future research.

Key words: Data clustering, distance measure, point symmetry, *k*d-tree, k-means

1. Introduction

Clustering [1] is a division of data into groups of similar objects. Each group, called a cluster, consists of objects that are similar within the cluster and dissimilar to objects of other clusters.

The clustering problem has been addressed in many contexts and by researchers in many disciplines. This reflects its broad appeal and usefulness as one of the steps in exploratory data analysis.

From a machine learning perspective clusters correspond to hidden patterns, the search for clusters is unsupervised learning [2,3], and the resulting system represents a data concept. Therefore, clustering is unsupervised learning of a hidden data concept.

There are numerous clustering techniques one can encounter in the literature. Most of the existing data clustering algorithms can be classified as hierarchical or partitioning. Within each class, there exists a wealth of subclasses, which includes different algorithms for finding the clusters.

While hierarchical algorithms [4] build clusters gradually (as crystals are grown), partitioning algorithms [5] learn clusters directly. In doing so, they either try to discover clusters by iteratively relocating points between subsets or try to identify clusters as areas highly populated with data.

Density-based algorithms [6] typically regard clusters as dense regions of objects in the data space that are separated by regions of low density. The main idea of the density-based approach is to find regions of high density and low density, with high-density regions being separated from low-density regions. These approaches can make it easy to discover arbitrary clusters. Many other clustering techniques are developed that either have theoretical significance or do not fit in previously outlined categories.

*Correspondence: sabudalfa@ucas.edu.ps

Clustering is a difficult problem, and differences in assumptions and contexts in different communities have made the transfer of useful generic concepts and methodologies slow to occur.

Many algorithms in the literature, like k-means, suffer from some faults of using Euclidean distance for calculating the symmetry measure between data clusters. Using Euclidean distance is improper for clustering overlapped and arbitrary shaped clusters. Therefore, many other distance measures were developed in the literature for improving the calculating of symmetry measure for clustering complex data sets.

Some algorithms calculate the connectivity of each data point to its cluster by depending on density reachability. A cluster, which is a subset of the points of the data set, satisfies 2 properties: all points within the cluster are mutually density-connected, and if a point is density-connected to any point of the cluster, it is part of the cluster as well. These algorithms can find arbitrarily shaped clusters, but they require parameters that are mostly sensitive to clustering performance. From the other side, these algorithms need to detect the nearest neighborhood of each data point, which is time-consuming.

We tackled previous defects in clustering algorithms and concluded that we can improve the performance of data clustering by using other distance measures instead of the Euclidean type and testing the connectivity of each data point with its cluster by using a suitable method. We tried to prevent increasing time complexity or using additional parameters. By using a suitable distance measure and checking the density reachability of data points within its cluster, we can cluster complex data sets that have overlapped and arbitrarily shaped clusters.

The contribution of the paper is that we developed an improved point symmetry-based distance measure by using a k -d-tree for clustering complex data sets and improving the performance of the classical k-means algorithm. Experimental results shown in this paper demonstrate the effectiveness of the proposed algorithm. We compared the proposed algorithm with the classical k-means algorithm. We presented the proposed algorithm and their results in detail along with avenues of future research.

The rest of the paper is organized as follows: Section 2 describes the review of the literature and related studies. Section 3 illustrates our contribution by using a k -d-tree for developing an improved point symmetry (PS)-based distance measure and developing a k-means algorithm with a novel distance measure. Section 4 shows experimental results to demonstrate the effectiveness of the proposed algorithm. Finally, Section 5 concludes the paper and presents suggestions for future work.

2. Review of the literature and related studies

2.1. K-means algorithm

K-means uses a 2-phase iterative algorithm to minimize the sum of point to centroid distances, summed over all k clusters. The first phase is “batch” updates, where each iteration consists of reassigning points to their nearest cluster centroid, all at once, followed by recalculation of cluster centroids. The 2nd phase uses “online” updates, where points are individually reassigned. By doing so it will reduce the sum of distances, and cluster centroids are recomputed after each reassignment. Each iteration during this 2nd phase consists of one pass through all the points. K-means can converge to a local optimum, which is a partition of points in which moving any single point to a different cluster increases the total sum of distances [7]. Thus, k-means has a hard membership function. Furthermore, k-means has a constant weight function, i.e. all data points belonging to a cluster have equal influence in computing the centroid of the cluster.

The k-means has 2 main advantages [8]: it is very easy to implement, and the time complexity is only $O(n)$ (n being the number of data points), which makes it suitable for large data sets. However, the k-means suffers from the following disadvantages: the user has to specify the number of classes in advance, the performance of

the algorithm is data-dependent, and the algorithm uses a greedy approach and is heavily dependent on the initial conditions. This often leads k-means to converge to suboptimal solutions.

Redmond and Heneghan [9] presented a method for initializing the k-means clustering algorithm using a *kd*-tree. The proposed method depends on the use of a *kd*-tree to perform a density estimation of the data at various locations. They used a modification of Katsavounidis' algorithm, which incorporates this density information, to choose *K* seeds for the k-means algorithm.

Mumtaz and Duraiswamy [10] proposed a novel density-based k-means clustering algorithm to overcome the drawbacks of density-based spatial clustering of applications with noise (DBSCAN) and k-means clustering algorithms. The result is an improved version of the k-means clustering algorithm. This algorithm performs better than DBSCAN [11] while handling clusters of circularly distributed data points and slightly overlapped clusters. However, there is a limitation for this algorithm. It requires a prior specification of some parameters, and the clustering performance is affected by these parameters.

2.2. A new point symmetry-based distance measure

This section presents a new point symmetry-based distance measure, which is described in the literature for improving point symmetric distance measure.

Symmetry is considered a preattentive feature that enhances recognition and reconstruction of shapes and objects [12]. Almost every interesting area around us consists of some generalized form of symmetry. As symmetry is so common in the natural world, it can be assumed that some kind of symmetry exists in the clusters also. Based on this, Su and Chou [13] proposed a symmetry-based clustering technique. They assigned points to a particular cluster if they present a symmetrical structure with respect to the cluster center. However, this work has some limitations; it is insufficient for classifying complex data sets that have clusters of irregular and unsymmetrical shapes.

Bandyopadhyay and Saha used a new point symmetry-based distance measure with an evolutionary clustering technique [14]. This algorithm is able to overcome some serious limitations of the earlier PS-based distance proposed by Su and Chou. This algorithm is therefore able to detect both convex and nonconvex clusters. Bandyopadhyay and Saha [15] offered certain improvements of this point symmetric distance measure and used it to cluster overlapping and arbitrarily shaped clusters.

Let a point be \bar{x} . The symmetrical (reflected) point of \bar{x} with respect to a particular center \bar{c} is

$$\bar{x}^* = 2 \times \bar{c} - \bar{x} \tag{1}$$

Let *knear* unique nearest neighbors of \bar{x}^* be at Euclidean distances of d_i s, $i = 1, 2, \dots, knear$. The new point symmetry-based distance measure [14] is then:

$$d_{ps}(\bar{x}, \bar{c}) = d_{sym}(\bar{x}, \bar{c}) \times d_e(\bar{x}, \bar{c}) \tag{2}$$

where $d_{sym}(\bar{x}, \bar{c}) = \frac{\sum_{i=1}^{knear} d_i}{knear}$, a symmetry measure of \bar{x} with respect to \bar{c} , and $d_e(\bar{x}, \bar{c})$ is the Euclidean distance between the points \bar{x} and \bar{c} . We used this distance measure instead of Euclidean distance with the k-means algorithm. We used Eq. (2) by estimating *knear* = 2.

The most limiting aspect of the measures from [13] is that they require a prior specification of a parameter θ , based on whether the assignment of points to clusters is done on the basis of the PS-based distance or the

Euclidean distance. Point \bar{x}_i is assigned to cluster k such that the PS-based distance between \bar{x}_i and the center of cluster k is the minimum, and provided that the total “symmetry” with respect to it is less than some threshold θ . Otherwise, the assignment is done based on the minimum Euclidean distance criterion. Therefore, the clustering performance is significantly affected by the choice of θ , and its value is dependent on the data characteristics.

Chou and Su in [16] chose θ to be equal to 0.18. In [15], the authors proposed to keep the value of θ equal to the maximum nearest neighbor distance among all the points in the data set. Thus, the computation of θ is automatic and does not require user intervention.

It is evident that the symmetrical distance computation is very time-consuming because it involves the computation of the nearest neighbors. The authors of [15] described that the computation of $d_{ps}(\bar{x}, \bar{c})$ is of complexity $O(nD)$, where D is the dimension of the data set and n is the total number of points present in the data set. Hence, for K clusters, the time complexity of computing PS-based distance between all points to different clusters is $O(n^2KD)$. In order to reduce the computational complexity, an approximate nearest neighbor search using the k d-tree approach was adopted in their paper.

From the aforementioned introduction, we can conclude the problems of using PS-based distance as follows: this measure is suitable only to classify clusters of symmetrical shape; using PS-based distance for data clustering requires a prior specification of a parameter θ ; the clustering performance is significantly affected by the choice of θ , and its value is dependent on the data characteristics; and the symmetrical distance computation is very time-consuming because it involves the computation of the nearest neighbors.

2.3. Kd-tree-based nearest neighbor computation

This section presents k d-tree which is an important multidimensional structure for storing a finite set of data points from k -dimensional space.

A K -dimensional tree, or k d-tree, [17] is a space-partitioning data structure for organizing points in a K -dimensional space. The k d-tree is a top-down hierarchical scheme for partitioning data. Consider a set of n points, $(x_1 \dots x_n)$ occupying an m -dimensional space. Each point x_i has associated with it m coordinates $(x_{i1}, x_{i2}, \dots, x_{im})$. There exists a bounding box, or bucket, which contains all data points and whose extrema are defined by the maximum and minimum coordinate values of the data points in each dimension. The data are then partitioned into 2 subbuckets by splitting the data along the longest dimension of the parent bucket. These partitioning processes may then be recursively repeated on each subbucket until a leaf bucket is created, at which point no further partitioning will be performed on that bucket. A leaf bucket is a bucket that fulfills a certain requirement, such as only containing one data point.

The k d-tree is the most important multidimensional structure for storing a finite set of data points from k -dimensional space. It decomposes a multidimensional space into hyperrectangles. A k d-tree is a binary tree with both a dimension number and a splitting value at each node. Each node corresponds to a hyperrectangle. A hyperrectangle is represented by an array of minimum coordinates and an array of maximum coordinates (e.g., in 2 dimensions ($k = 2$), (x_{min}, y_{min}) and (x_{max}, y_{max})). When searching for the nearest neighbor we need to know if a hyperrectangle intersects with a hypersphere. The contents of each node are depicted in Table 1.

An interesting property of the k d-tree is that each bucket will contain roughly the same number of points. However, if the data in a bucket are more densely packed than some other bucket we would generally expect the volume of that densely packed bucket to be smaller.

Table 1. The fields of k d-tree node.

Field	Description
Type	Type of node tree (node or leaf)
Parent	The index of parent node in k d-tree
Splitdim	The splitting dimension number
Splitval	The splitting value
Left k d-tree	A k d-tree representing those points to the left of the splitting plane
Right k d-tree	A k d-tree representing those points to the right of the splitting plane
Hyperrect	The coordinates of hyperrectangle
Numpoints	The number of points contained in hyperrectangle

Each node splits the space into 2 subspaces according to the splitting dimension of the node and the node's splitting value. Geometrically this represents a hyperplane perpendicular to the direction specified by the splitting dimension.

Searching for a point in the data set that is represented in a k d-tree is accomplished in a traversal of the tree from root to leaf, which is of complexity $O(\log(n))$ (if there are n data points). The first approximation is initially found at the leaf node, which contains the target point.

3. Methodology

In this section we illustrate our original work for improving the efficiency of data clustering and tackling the problem presented in Section 1. This section illustrates the usage of a k d-tree for developing an improved PS-based distance measure and describes our contribution for improving the efficiency of the k-means algorithm. We developed a k-means algorithm with a novel distance measure. We used an improved PS-based distance measure for developing the proposed algorithm. We used selected nodes from the k d-tree to develop this algorithm.

3.1. Selecting dense points

We proposed to use the k d-tree for checking the connectivity of each data point with its cluster. We used the k d-tree to determine the collections of dense regions in dimensional space. Using the k d-tree will reduce computation costs in determining the dense regions. We selected some points of the k d-tree that denote the dense centers of dense regions in the data set. We called these points dense points (DPs).

Selecting leaf nodes as DPs is not suitable because each leaf node in the k d-tree is a bucket containing only one data point and will cause the selecting of all data points in the data set. Therefore, selecting leaf nodes as DPs will not form dense centers of dense regions in the data set. Selecting the parent of leaf nodes in the k d-tree as DPs is not suitable also because the parent of a leaf node contains only 2 data points (2 leaf nodes) and will cause sensitivity to noise (outlying data points) in the data set.

Depending on the previous analysis, we selected DPs by searching for leaf nodes in the k d-tree and then finding the grandparent of the leaf nodes. Grandparents of the leaf nodes contain more than 2 data points, and so selecting them as DPs will reduce sensitivity to noise in the data set and will form a small number of centers for denoting dense regions in the data set and reducing the processing time of the data clustering. Figure 1 shows the structure of the k d-tree and position of DPs. We note that all DPs (which are shown as shaded nodes) denote 2nd and 3rd levels of the k d-tree. We note that more than 2 nodes fork from nodes of DPs; this indicates that DPs contain more than 2 data points.

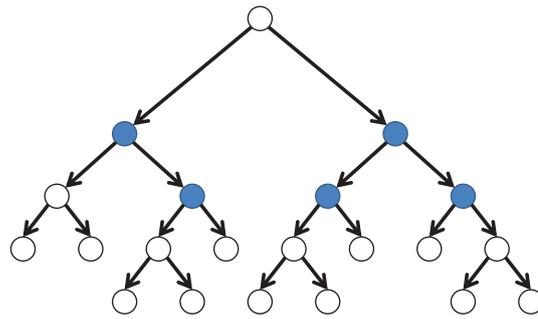


Figure 1. Selected dense points.

Figure 2a shows the position of DPs on a synthetic data set that has one cluster. We note that these points form almost the shape of a cluster with a small number of points. We note that DPs are distributed on the whole data set and they exist in dense regions.

Figure 2b shows the rectangular regions that are covered by DPs of the k d-tree. We note that these regions almost cover all the data set, and so we can conclude that DPs correspond to the dense regions of the data set.

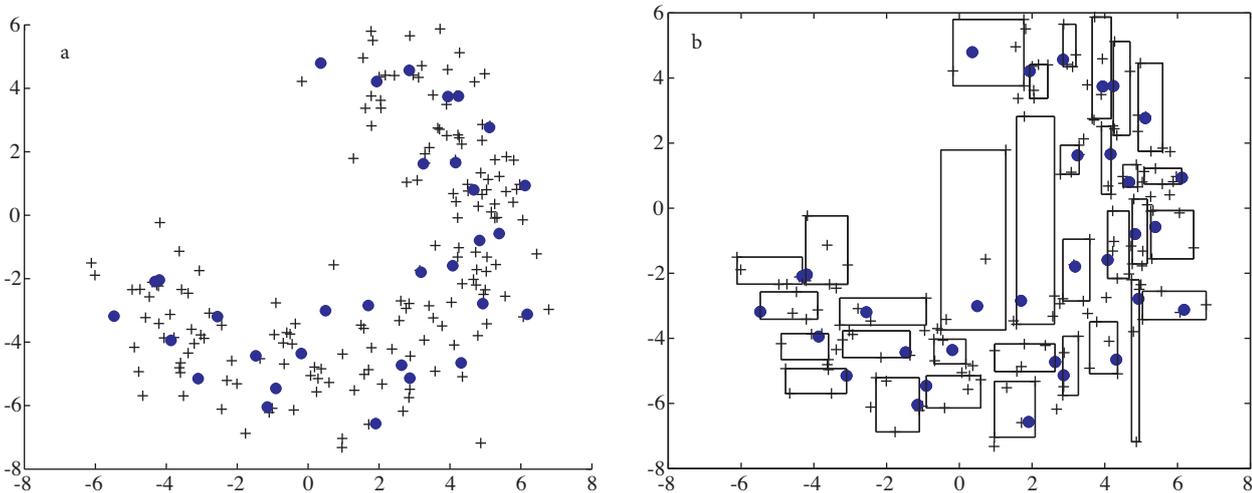


Figure 2. a) DPs of the k d-tree. b) Rectangular regions covered by DPs.

Using upper levels in the k d-tree (more than the 3rd level) for selecting DPs will decrease the number of DPs for representing dense regions, but at the same time rectangular regions will be larger and will cover some parts of the space that are empty of data points. Figure 3 shows selecting nodes from various levels in the k d-tree and the correspondence of these nodes to data points in a data set of one cluster. We note from Figure 3a that the number of nodes that denote dense regions are smaller than the number of DPs shown in Figure 3b, but the sizes of rectangular regions are increased.

These effects are increased gradually when selecting nodes from upper levels as shown in Figures 3c and 3d, and this causes covering in regions empty of data points.

Figure 3e shows that only one node represents all data points in the cluster and covers the empty space outside the cluster. Therefore, we inferred that if we use upper levels for representing DPs, the shape formed by rectangular regions for covering the cluster will then be rough, and many data points will be selected from other clusters if there are overlapped clusters in the data set.

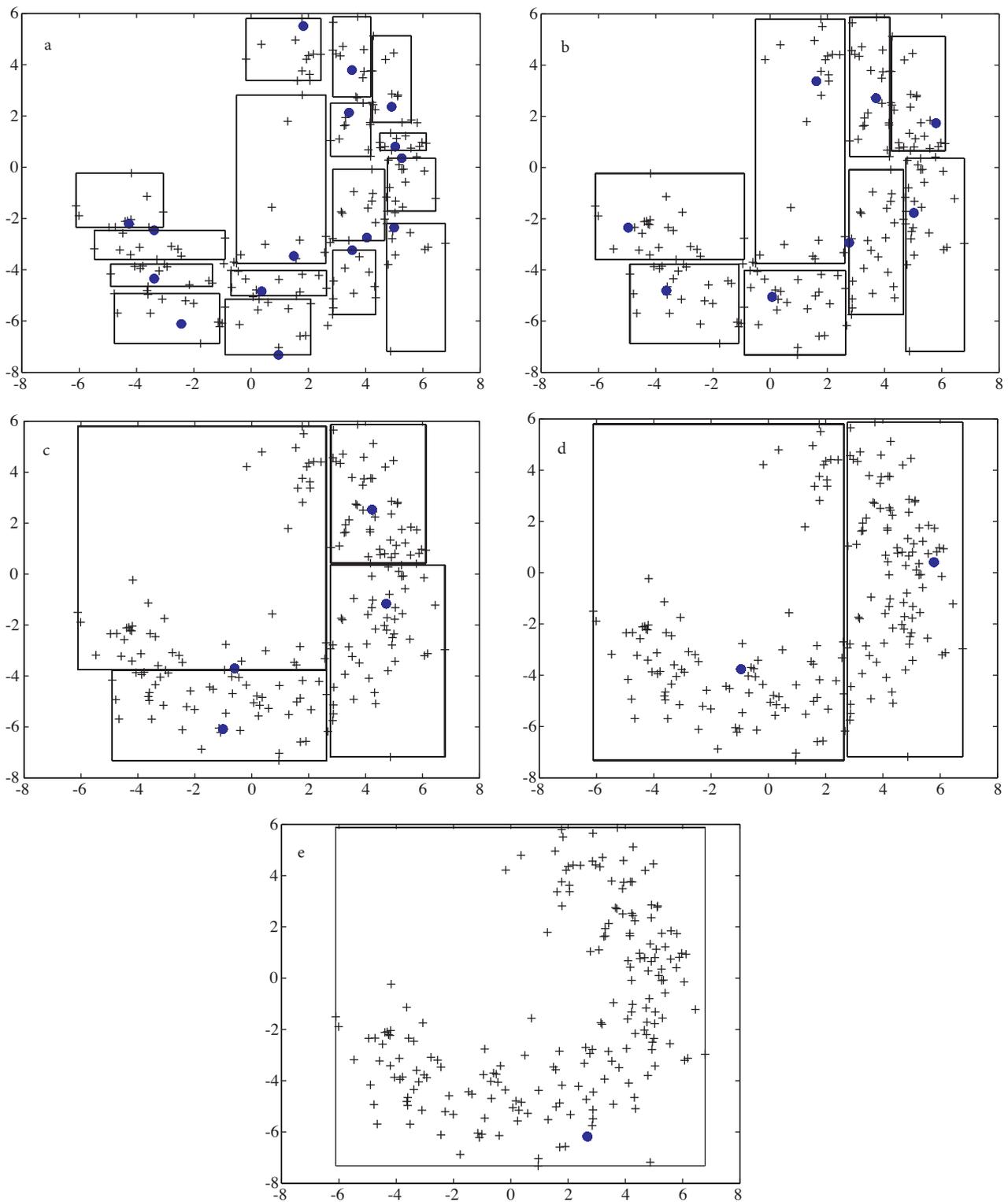


Figure 3. a) Nodes of the 4th level. b) Nodes of the 5th level. c) Nodes of the 6th level. d) Nodes of the 7th level. e) Selecting nodes of the 8th level.

We can conclude that selecting the grandparent of the leaf nodes in the k d-tree for representing DPs is the best choice to determine the collections of dense regions in dimensional space. We used this concept for selecting DPs in our experiments for increasing the performance of clustering complex data sets.

Selecting DPs has many advantages. First of all, using DPs reduces the number of data points used for data clustering, and so this method will reduce time complexity. From the other side, using DPs will reduce the effect of noise (outlying data points) on data clustering. Figure 4 shows the position of DPs (plotted as circles) in a data set having one cluster with outlying data points. We note that the outlying data points, which are denoted as + symbols in the 4 corners of Figure 4, are not selected as DPs. We note also that all DPs are concentrated in spaces that have density of data points.

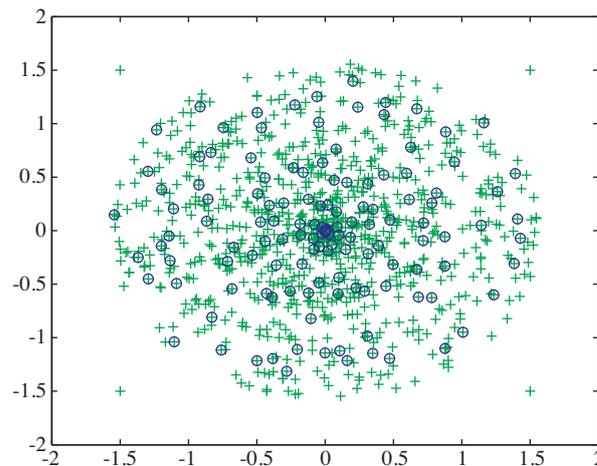


Figure 4. Selecting DPs from data set having noise.

We can thus use DPs for checking the density reachability of each data point with its cluster. Using DPs will be effective for clustering complex data sets that have overlapped and arbitrary shaped clusters. We use DPs to improve the PS-distance measure and define a new effective distance measure that can be used in k-means clustering.

3.2. Improved PS-based distance measure

In this section, we illustrate our method for enhancing PS-based distance measure by using a k d-tree. This enhancement was used for data clustering and overcoming previous limitations presented in Section 1.

When we used k-means with Euclidean distance to calculate distances between data points and centroids and then cluster data points to the nearest centroid, we noted that the results of data clustering were poor when using complex data sets.

As expected, the results will be better when we use PS-based distance measure, but some clusters were not classified correctly. We dissected this problem and deduced that we must include the density of points with the distance measure to classify this type of data set.

When using k-means with Euclidean distance and PS-based distance measure, all points are assigned to the nearest cluster despite some of them being connected to other clusters.

If we study the connectivity of these points with the nearest clusters, then we will tackle the problem and classify all clusters correctly.

Our proposed method uses DPs of the k d-tree for determining the connectivity instead of using other methods that are presented in literature like DBSCAN.

We developed a simple algorithm for selecting points that are clustered incorrectly when using k-means with Euclidean distance. We manipulated these points for developing a new distance measure.

This algorithm is as follows:

```

FOR each data point  $X$  do
    Find the nearest 2 centroids of  $X$  (i.e.  $C_1, C_2$ )
    Find the nearest 2 DPs of  $X$  (i.e.  $DP_1, DP_2$ )
    IF  $d_{ps}(DP_1, C_2) < d_{ps}(DP_1, C_1)$  OR  $d_{ps}(DP_2, C_2) < d_{ps}(DP_2, C_1)$ 
        Select  $X$ 
    END IF
END FOR
    
```

In the algorithm above, d_{ps} is the PS-based distance measure that is calculated by Eq. (2). We used PS-based distance measure instead of Euclidean distance to improve clustering performance. We used a value of 2 for k_{near} in Eq.(2), because this value gives good results when merging connectivity with distance measure. As expected, using a value greater than 2 for k_{near} will increase accuracy of data clustering, but it will also increase time complexity. Our approach merges connectivity with distance measure, so checking connectivity of each data point with its cluster will increase accuracy and using a value greater than 2 for k_{near} is not necessary.

The work described in this paper concerns crisp (hard) clustering algorithms only. Therefore, each data point will be assigned to one and only one cluster. We particularized data clustering to the nearest 2 clusters of each data point. If the data point is not followed to the nearest cluster (nearest centroid), then it will be connected to the 2nd nearest cluster.

The algorithm above finds the nearest 2 clusters (centroids) of each data point, and then finds the nearest 2 DPs of this data point. After that, the algorithm checks the connectivity of this data point with the 2nd nearest cluster by checking if any point of the nearest 2 DPs is followed to that cluster or not. If any point of the nearest 2 DPs is followed to the 2nd nearest cluster then we are sure that the data point follows to the 2nd nearest cluster despite whether it is near to the 1st nearest cluster.

We used the nearest 2 DPs to check connectivity of each data point to its cluster. As expected, using more of the nearest DPs for checking connectivity will increase the accuracy of data clustering, but will also increase time complexity. Using 2 DPs is enough for checking the connectivity of each data point with its cluster, because our study depends on calculating distances to the nearest clusters. Catching only 1 DP followed to the 2nd nearest cluster is enough to decide that the data point is connected to the 2nd nearest cluster, but we used 2 DPs instead of 1 for increasing accuracy of data clustering.

Depending on previous concepts we developed a new distance measure for improving the performance of PS-distance measure, as follows:

$$d_{IPS}(\bar{x}, \bar{c}) = \frac{d_{ps}(DP_1, \bar{c})}{d_e(DP_1, \bar{c}_2)} + \frac{d_{ps}(DP_2, \bar{c})}{d_e(DP_2, \bar{c}_2)} \tag{3}$$

where d_{ps} denotes the PS-based distance measure, which is calculated by Eq. (2), and d_e denotes the Euclidean

distance. DP_1 and DP_2 denote the 1st and 2nd DPs of the k d-tree, which are selected as demonstrated in the previous section. \bar{c}_2 denotes the 2nd nearest centroid of \bar{x} .

The first part of Eq. (3) ($\frac{d_{ps}(DP_1, \bar{c})}{d_e(DP_1, \bar{c}_2)}$) checks the connectivity of the 1st nearest DP (DP_1) to the 2nd nearest centroid of \bar{x} (\bar{c}_2). If $d_e(DP_1, \bar{c}_2) > d_{ps}(DP_1, \bar{c})$, then the value of $\frac{d_{ps}(DP_1, \bar{c})}{d_e(DP_1, \bar{c}_2)}$ will be small, and this will cause data point \bar{x} to be attached to cluster \bar{c} because clustering of data points depends on assigning each data point \bar{x} to a cluster that has the smallest distance $d_{IPS}(\bar{x}, \bar{c})$. If $d_e(DP_1, \bar{c}_2) < d_{ps}(DP_1, \bar{c})$, then the value of $\frac{d_{ps}(DP_1, \bar{c})}{d_e(DP_1, \bar{c}_2)}$ will be big, and this will cause data point \bar{x} to be attached to cluster \bar{c}_2 because the distance between \bar{x} and \bar{c} ($d_{IPS}(\bar{x}, \bar{c})$) will be bigger than the distance between \bar{x} and \bar{c}_2 ($d_{IPS}(\bar{x}, \bar{c}_2)$) when calculating all distances between data point \bar{x} and all centroids in the data set. The 2nd part of Eq. (3) ($\frac{d_{ps}(DP_2, \bar{c})}{d_e(DP_2, \bar{c}_2)}$) checks the connectivity of the 2nd nearest DP (DP_2) to the 2nd nearest centroid of \bar{x} (\bar{c}_2) as it did when checking the connectivity of DP_1 in the 1st part of Eq. (3). The 2 parts of Eq. (3) are added, which corresponds to OR operation in the above algorithm.

We used Eq. (3) to develop a new k-means algorithm with a novel distance measure for clustering complex data sets.

3.3. K-means algorithm with a novel distance measure

This section describes our contribution for improving the efficiency of the classical k-means algorithm. We present the pseudocode of the proposed algorithm as follows:

1. Initialize K center locations (C_1, \dots, C_K).
2. Select DPs of k d-tree.
3. FOR each cluster center C_j do
 - FOR each data point X_i do
 - Calculate $d_{IPS}(X_i, C_j)$ by using Eq. (3).
 - END FOR
- END FOR
4. Assign each data point X_i to its cluster center C_j by selecting the minimum distance of $d_{IPS}(X_i, C_j)$.
5. Update each cluster center C_j as the mean of all X_i that have been assigned to it.
6. Calculate $D = \sum_{i=1}^n [d_{IPS}(X_i, C_j)]^2$.
7. If the value of D has converged, then return (C_1, \dots, C_K); else go to step 3.

This algorithm has 3 main advantages. First, it is very easy to implement. Second, it does not use additional parameters like other algorithms that are proposed in the literature for improving the efficiency of k-means algorithm. Most of the parameters that are used by other algorithms are sensitive to the performance of data clustering. Third, its performance is better than the performance of the classical k-means algorithm. This algorithm clustered many data sets that were clustered incorrectly by classical k-means algorithm.

However, this algorithm suffers from the following disadvantages: the user has to specify the number of classes in advance, processing time is increased compared to using k-means with Euclidean distance, and the algorithm uses a greedy approach and is heavily dependent on the initial conditions. This often leads the results to converge to suboptimal solutions. Therefore, we propose to use [9] in step 1 of our proposed algorithm to eliminate the dependency on the initial conditions.

4. Experimental results

4.1. Performance of improved PS-based distance measure

In this section, we illustrate the performance of using improved PS-based distance measure in data clustering and overcoming previous limitations that were presented in Section 1. We use the k d-tree structure for decreasing computation cost in searching for the nearest points and improving the performance of data clustering. We use DPs of the k d-tree for checking connectivity of each data point with its cluster.

We implemented the experiment to test the efficiency of using Euclidean, a new point symmetry-based, and improved PS-based distance measures for clustering many synthetic data sets. For simplicity we fixed the value of each centroid of clusters to be the mean value of the cluster's points. We then studied the effect of these distance measures on classifying data points to the nearest centroid. In this experiment we use each distance for measuring symmetry between clusters. We measure the distance between each data point \bar{x} and all centroids, and then we assign the data point \bar{x} to the cluster that gives the minimum distance between its centroid and \bar{x} .

Figure 5a shows a synthetic data set and the initial locations of centroids that are used by the k-means algorithm.

The result of data clustering by using k-means with Euclidean distance is shown in Figure 5b. As expected, all the clusters are classified incorrectly. We can use Euclidean distance for classifying clusters with spherical shape, so this data set is not suitable. As shown in Figure 5b, the right upper cluster is classified into 3 groups, where some of the points that are denoted as triangles are assigned to the left upper cluster, and some of the other points that are denoted as stars are assigned to the lower cluster. We can conclude that using Euclidean distance is not suitable for measuring symmetry distances between clusters that are nonglobular shapes and have different sizes.

We repeated this experiment by using the new point symmetry-based distance measure instead of Euclidean distance. The new point symmetry-based distance measure is suitable for measuring symmetry between clusters of regular and symmetrical shapes.

Figure 5c shows the result of clustering the same data set by using k-means algorithm with a new point symmetry-based distance measure. We note that the data set is clustered correctly into 3 clusters. Points corresponding to each cluster are marked in specific symbols.

We repeated this experiment by using the improved PS-based distance measure. As expected, all clusters are classified correctly, as shown in Figure 5c. We note that the improved PS-based distance measure can be used for clustering data sets that have clusters of regular and symmetrical shapes.

Figure 6a shows a complex synthetic data set in 2 dimensions that contains 4 clusters, which have irregular and unsymmetrical shapes and have different sizes. The initial values of centroids are shown as shaded circles. As shown, the clusters are slightly overlapped, which causes difficulty for clustering by many algorithms that are described in the literature.

We used k-means with Euclidean, PS-based, and improved PS-based distances for clustering this data set. We assumed that the algorithm had converged when there was no change in the values of centroids.

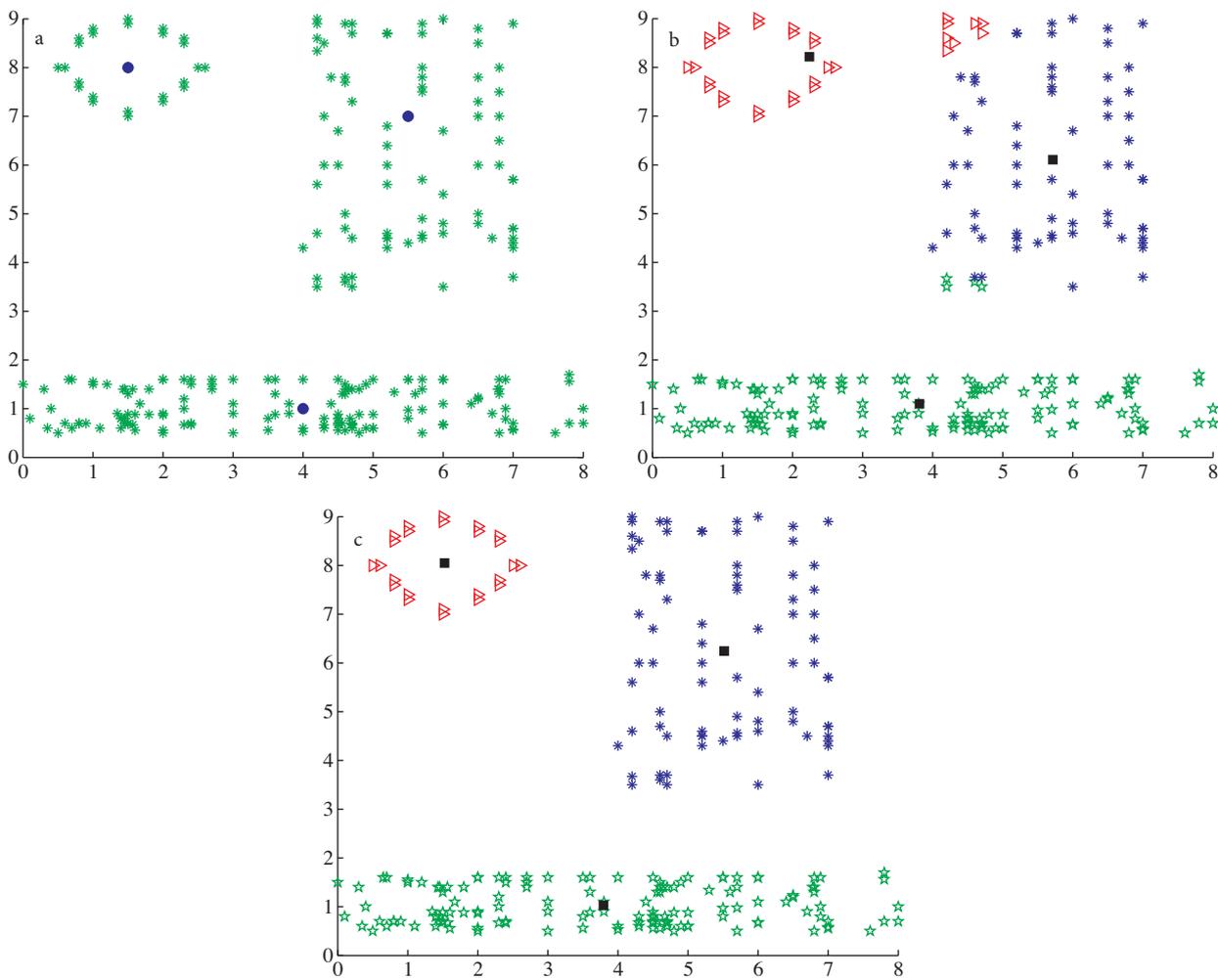


Figure 5. a) Unlabeled data set with trial cluster centroids. b) Clustering result by using k-means with Euclidean distance. c) Clustering result by using k-means with PS-based distance measure and improved PS-based distance measure.

First, we used Euclidean distance for measuring the distance between data points and centroids with the k-means algorithm and assigning each data point to the nearest centroid (corresponding to the minimum distance). The results were worse, as shown in Figure 6b, where all clusters are clustered incorrectly. We note that the largest cluster is clustered into 4 groups and most of its points are assigned to the other 3 clusters. The main explanation for this result is that the data points are assigned to the nearest centroid in spite of the fact that they are connected to another cluster. We note also that the centroid of the lower cluster is moved to a region that is empty of data points.

After that, we used PS-based distance measure (Eq. (2)) instead of Euclidean distance. As expected, the results were better, as shown in Figure 6c, but only one cluster (its points are marked as square symbols) is clustered correctly. Many of the data points are clustered incorrectly by using Euclidean distance while they are clustered correctly by using the new point symmetry-based distance measure. We used also the concept of [15] for calculating θ in our experiments, but as shown in Figure 6c, it is not effective with this data set and the clusters are clustered incorrectly.

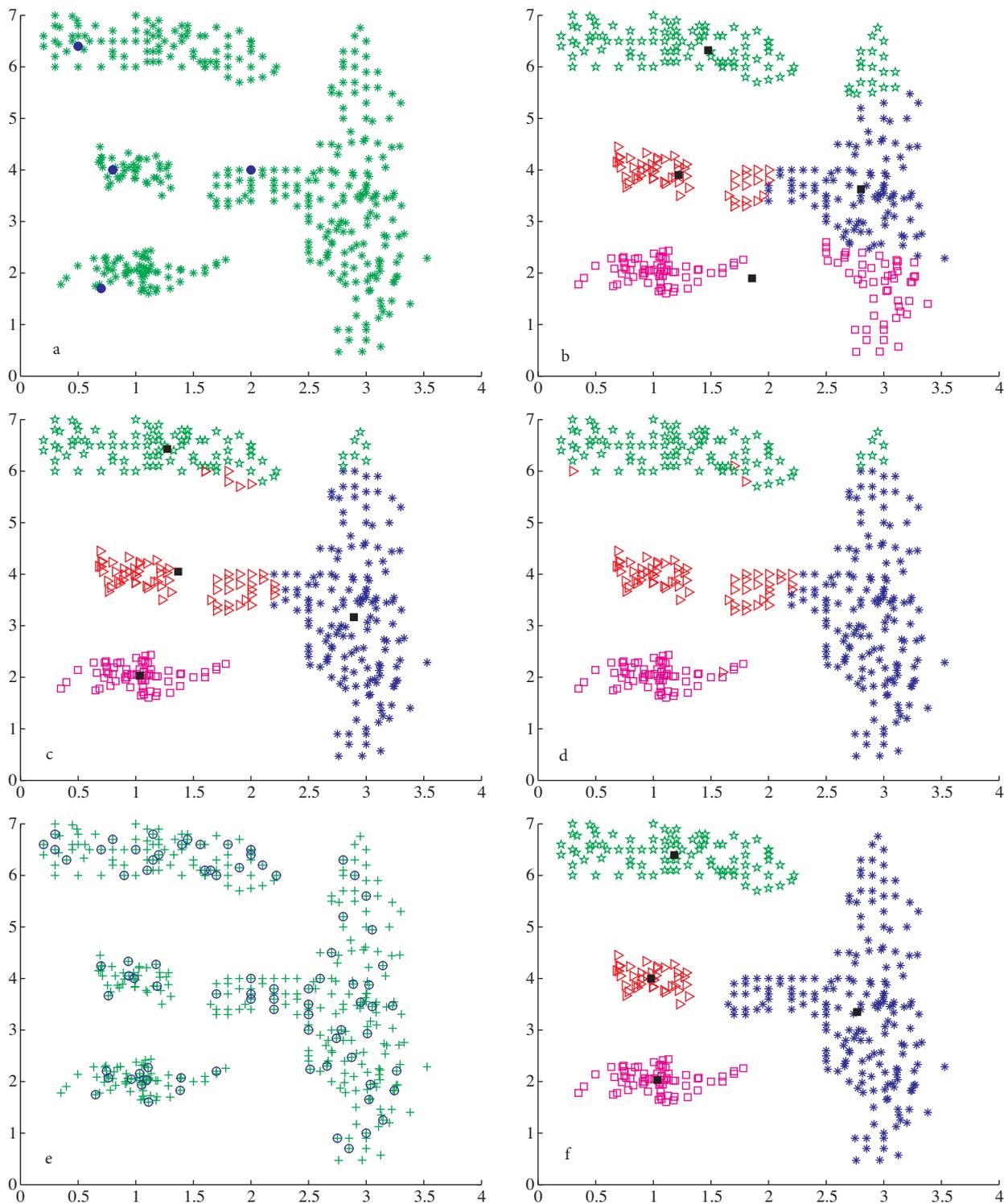


Figure 6. a) Complex synthetic data set. b) Clustering result by using k-means with Euclidean distance. c) Clustering result by using k-means with PS-based distance measure. d) The best result achieved by clustering with changing θ when using the PS-based distance. e) DPs of k d-tree. f) Clustering result by using k-means with improved PS-based distance.

We tried manually to choose different values of θ to enhance the results of data clustering by using the new point symmetry-based distance measure. Figure 6d shows the best results after changing the value of θ to 0.8.

We note that 1 of the 4 clusters (its points are marked as triangle symbols) is clustered correctly. We also noted that if θ is a small value, then the percentage of data points that are clustered incorrectly will increase. Vice versa, if θ is a large value then the percentage of data points that are clustered incorrectly will decrease.

We conclude that using the new point symmetry-based distance measure with the k-means algorithm is insufficient for clustering complex data sets that have clusters of irregular and unsymmetrical shapes. When we used this distance measure instead of Euclidean distance, we noted that it is appropriate for some data sets that have only clusters of symmetrical shapes.

Finally, we used the improved PS-based distance measure (Eq. (3)) to calculate the distance between data points and centroids, and we select the minimum distance between each data point and centroids. Figure 6e shows the selected DPs of k d-tree (marked as circles) that are used by Eq. (3).

Figure 6f shows the result of clustering by using the improved PS-based distance measure. The results were the best, as shown in Figure 6f, and we note that all the clusters are classified correctly and all the centroids, which are shown as shaded squares, are calculated correctly.

Figure 7a shows another synthetic data set. This data set consists of 2 clusters, 1 of which has a crescent shape and 1 of which has a circular shape. The initial values of centroids are shown as shaded circles.

We used k-means with Euclidean, PS-based, and improved PS-based distances for clustering this data set. We assume that the algorithm has converged when there is no change in the values of centroids. First, we used Euclidean distance for measuring the distance between the data points and centroids with k-means algorithm and assigning each data point to the nearest centroid (corresponding to the minimum distance). Figure 7b shows the result of using k-means with Euclidean distance. We note that the cluster that has a circular shape is split into 2 groups and 1 of these groups was joined to the cluster of a crescent shape.

Figure 7c shows the result of using k-means with a new point symmetry-based distance measure. We note that the cluster that has a crescent shape is split into 2 groups and the smallest group was joined to the cluster of a circular shape. We conclude that the cluster that has a crescent shape is not symmetrical and all the data points that clustered incorrectly do not have data points in the opposite side of the cluster. Thus, using k-means with a new point symmetry-based distance measure is not suitable for clustering data sets that have unsymmetrical clusters.

Finally, Figure 7d shows the result of using k-means with an improved PS-based distance measure. We note that all data points are clustered correctly, including the data points that are clustered incorrectly when using a new point symmetry-based distance. All data points are clustered correctly because the improved PS-based distance measure checks the connectivity of each data point to the corresponding cluster. Thus, using k-means with improved PS-based distance measure generates the best results.

4.2. Performance of k-means algorithm with a novel distance measure

We tested the performance of our proposed algorithm and compared it with k-means using Euclidean distance and PS-based distance measure. We measured elapsed time, percentage of data points classified incorrectly, and the number of iterations executed until the algorithm converges. The algorithms were applied to a synthetic data set, which is shown in Figure 6a. The algorithms were implemented in MATLAB 7.3 (R2006b) on a laptop Intel Core 2 CPU. The clock speed of the processors is 1.66 GHz, and the memory size is 1.00 GB of RAM. Table 2 shows the results.

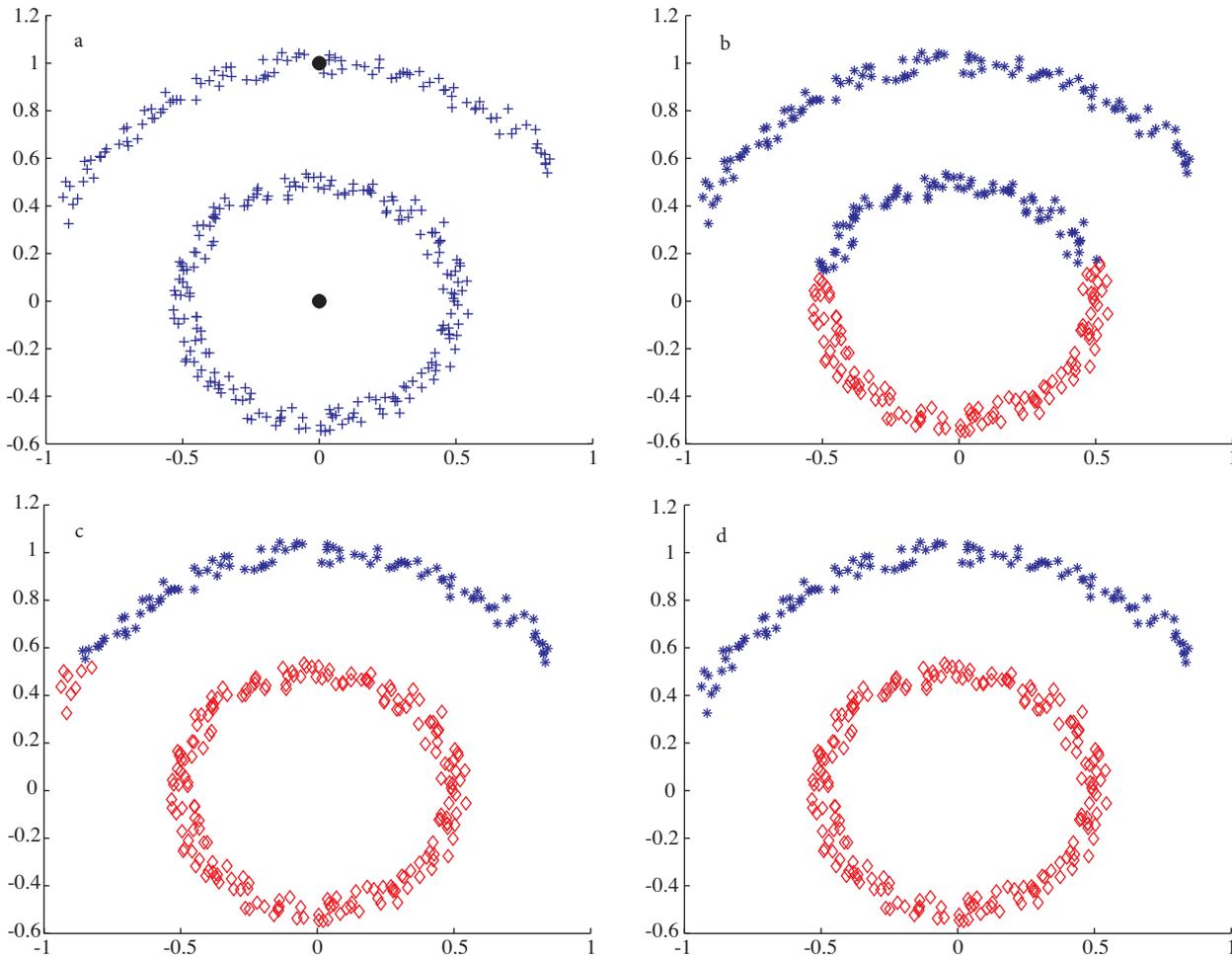


Figure 7. a) Unlabeled data set with trial cluster centroids. b) Clustering result by using k-means with Euclidean distance. c) Clustering result by using k-means with PS-based distance measure. d) Clustering result by using k-means with improved PS-based distance.

Table 2. Performance analysis of k-means and k-means with a novel distance measure.

Algorithm	Elapsed time (s)	Error (%)	Iterations (#)
k-means with Euclidean distance	0.029259	24.36	8
k-means with PS-based distance	7.113533	8.72	26
k-means with improved PS-based distance	3.383083	0.0	3

We note that using Euclidean distance with k-means takes the smallest elapsed time, but the percentage of data points (percentage of error) that are classified incorrectly is the largest, as shown in Figure 6b. The percentage of error is decreased when k-means is used with PS-based distance measure, but it takes more elapsed time. It needs a greater number of iterations until the convergence takes place. Our proposed algorithm gave the best performance. It takes the smallest number of iterations, and it classified all the data points correctly. It takes more time for clustering when comparing it with Euclidean distance and it takes less time when comparing it with PS-based distance measure.

The time elapsed for step 2 of the proposed algorithm is 0.123331 s, so the time elapsed by our proposed algorithm is 3.259752 s without calculating the time of selecting DPs of the k d-tree. We can conclude that

selecting DPs of the k d-tree did not influence the total elapsed time of using the proposed algorithm, because the k d-tree is created only once and then its nodes are used many times in the algorithm. This also matches the using of a high-dimensional data set.

We used many real-life data sets [18] for testing the performance of our proposed algorithm and compared it with k -means using Euclidean distance and PS-based distance measure. Here, n is the number of data points, d is the number of features, and K is the number of clusters.

1. Pima Indians diabetes data set ($n = 768$, $d = 8$, $K = 2$): These data were sampled from 2 clusters. The 1st cluster has 268 objects and the 2nd cluster has 500 objects. All patients here are females, at least 21 years old, and of Pima Indian heritage. The data contain 8 relevant features: 1) number of times pregnant; 2) plasma glucose concentration after 2 h in an oral glucose tolerance test; 3) diastolic blood pressure; 4) triceps skin-fold thickness; 5) 2-h serum insulin; 6) body mass index; 7) diabetes pedigree function; and 8) age.
2. Echocardiogram data set ($n = 131$, $d = 7$, $K = 2$): The problem is to predict whether or not the patient will survive at least 1 year. The most difficult part of this problem is correctly predicting that the patient will not survive. These data were sampled from 2 clusters. The 1st cluster has 43 objects and the 2nd cluster has 88 objects. The data contain 7 relevant features: 1) age when heart attack occurred; 2) pericardial effusion; 3) fractional shortening; 4) E-point septal separation; 5) left ventricular end-diastolic dimension; 6) wall motion score; and 7) wall motion index.
3. Ecoli data set ($n = 336$, $d = 7$, $K = 8$): The data were sampled from 8 different classes: 1) cp (143 objects); 2) im (77 objects); 3) imS (2 objects); 4) imL (2 objects); 5) imU (35 objects); 6) om (20 objects); 7) omL (5 objects); and 8) pp (52 objects). The data contain 7 relevant features: 1) McGeoch's method for signal sequence recognition; 2) von Heijne's method for signal sequence recognition; 3) von Heijne's signal peptidase II consensus sequence score; 4) presence of charge on N-terminus of predicted lipoproteins; 5) score of discriminant analysis of the amino acid content of outer membrane and periplasmic proteins; 6) score of the ALOM membrane spanning region prediction program; and 7) score of ALOM program after excluding putative cleavable signal regions from the sequence.
4. Hayes-Roth data set ($n = 132$, $d = 4$, $K = 3$): These data were sampled from 3 clusters. The 1st cluster has 51 objects; the 2nd cluster has 51 objects; and the 3rd cluster has 30 objects. The data set contains 4 numeric-valued attributes: 1) hobby; 2) age; 3) educational level; and 4) marital status.
5. Statlog (Heart) data set ($n = 170$, $d = 13$, $K = 2$): This data set is a heart disease database. The data set contains 13 attributes: 1) age; 2) sex; 3) chest pain type; 4) resting blood pressure; 5) serum cholesterol; 6) fasting blood sugar > 120 mg/dL; 7) resting electrocardiographic results; 8) maximum heart rate achieved; 9) exercise-induced angina; 10) ST depression induced by exercise relative to rest; 11) the slope of the peak exercise ST segment; 12) number of major vessels; and 13) thal (normal; fixed defect; or reversible defect). The objective is to classify each data vector into present (120 objects) or absent (150 objects).
6. Postoperative patient data set ($n = 90$, $d = 8$, $K = 3$): This is a data set of patient features. The classification task of this data set is to determine where patients in a postoperative recovery area should be

- sent to next. Because hypothermia is a significant concern after surgery, the attributes correspond roughly to body temperature measurements. The data set contains 8 attributes: 1) patient's internal temperature; 2) patient's surface temperature; 3) oxygen saturation; 4) last measurement of blood pressure; 5) stability of patient's surface temperature; 6) stability of patient's core temperature; 7) stability of patient's blood pressure; and 8) patient's perceived comfort at discharge. The problem is to predict the current discharge decision: 1) patient sent to the intensive care unit (2 objects); 2) patient prepared to go home (24 objects); and 3) patient sent to general hospital floor (64 objects).
7. Statlog (image segmentation) data set ($n = 2310$, $d = 19$, $K = 7$): The instances were drawn randomly from a database of 7 outdoor images. The images were hand-segmented to create a classification for every pixel. Each instance is a 3×3 region. The data set contains 19 attributes: 1) the column of the center pixel of the region; 2) the row of the center pixel of the region; 3) the number of pixels in a region; 4) the results of a line extraction algorithm that counts how many lines of a length of 5 with low contrast, less than or equal to 5, go through the region; 5) same as short-line-density-5 but counts lines of high contrast, greater than 5; 6) measure the contrast of horizontally adjacent pixels in the region. The mean is given; 7) measure the contrast of horizontally adjacent pixels in the region. The standard deviation is given; 8) measure the contrast of vertically adjacent pixels. The mean is given; 9) measure the contrast of vertically adjacent pixels. The standard deviation is given; 10) intensity of mean; 11) the average over the region of the R value; 12) the average over the region of the B value; 13) the average over the region of the G value; 14) measure the excess of red; 15) measure the excess of blue; 16) measure the excess of green; 17) value of mean; 18) saturation of mean; and 19) hue of mean. The data were sampled from 7 different classes: 1) brickface; 2) sky; 3) foliage; 4) cement; 5) window; 6) path; and 7) grass. The number of objects that belong to each cluster is 330.
 8. Glass ($n = 214$, $d = 9$, $K = 6$): The data were sampled from 6 different types of glass: 1) building windows float processed (70 objects); 2) building windows nonfloat processed (76 objects); 3) vehicle windows float processed (17 objects); 4) containers (13 objects); 5) tableware (9 objects); and 6) headlamps (29 objects). Each type has 9 features: 1) refractive index; 2) sodium; 3) magnesium; 4) aluminum; 5) silicon; 6) potassium; 7) calcium; 8) barium; and 9) iron.
 9. Statlog (vehicle silhouettes) data set ($n = 846$, $d = 18$, $K = 4$): The purpose is to classify a given silhouette as 1 of 4 types of vehicles, using a set of features extracted from the silhouette. The data were sampled from 4 different types of vehicles: 1) double decker bus (218 objects); 2) Chevrolet van (199 objects); 3) Saab 9000 (217 objects); and 4) an Opel Manta 400 (212 objects). Each type has 18 features: 1) compactness; 2) circularity; 3) distance circularity area; 4) radius ratio; 5) pr.axis aspect ratio; 6) max. length aspect ratio; 7) scatter ratio; 8) elongatedness area; 9) pr.axis rectangularity area; 10) max. length rectangularity area; 11) scaled variance major; 12) scaled variance minor; 13) scaled radius of gyration; 14) skewness about major; 15) skewness about minor; 16) kurtosis about major; 17) kurtosis about minor; and 18) hollows ratio.
 10. Yeast data set ($n = 1484$, $d = 8$, $K = 10$): The purpose is to localize the site of protein. The data contain 8 relevant features: 1) McGeoch's method for signal sequence recognition; 2) von Heijne's method for signal sequence recognition; 3) score of the ALOM membrane spanning region prediction program; 4)

score of discriminant analysis of the amino acid content of the N-terminal region of mitochondrial and non-mitochondrial proteins; 5) Presence of “HDEL” substring; 6) peroxisomal targeting signal in the C-terminus; 7) score of discriminant analysis of the amino acid content of vacuolar and extracellular proteins; and 8) score of discriminant analysis of nuclear localization signals of nuclear and nonnuclear proteins. The data were sampled from 10 classes: 1) 244 objects; 2) 429 objects; 3) 463 objects; 4) 44 objects; 5) 35 objects; 6) 51 objects; 7) 163 objects; 8) 30 objects; 9) 20 objects; and 10) 5 objects.

11. Wine ($n = 178, d = 13, K = 3$): This is a classification problem with “well-behaved” class structures. The data contain 13 relevant features: 1) alcohol; 2) malic acid; 3) ash; 4) alkalinity of ash; 5) magnesium; 6) total phenols; 7) flavonoids; 8) nonflavonoid phenols; 9) proanthocyanidins; 10) color intensity; 11) hue; 12) OD280/OD315 of diluted wines; and 13) proline. The data were sampled from 3 types of wine: 1) 59 objects; 2) 71 objects; and 3) 48 objects.

We used Waikato Environment for Knowledge Analysis (Weka) [19] version 3.6.2 for clustering data sets by k-means with Euclidean distance. Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data preprocessing, clustering, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes.

We tested the performance of k-means with Euclidean distance, a new PS-based distance, and improved PS-based distance by counting the data points that are clustered incorrectly. The data set description and the individual performance of k-means with Euclidean distance, a new PS-based distance, and improved PS-based distance are summarized in Table 3.

Table 3. The data set descriptions and the number of incorrectly clustered instances by k-means with Euclidean, PS-based, and improved PS-based distance measures.

#	Data set name	N	d	K	# data points clustered incorrectly by k-means with Euclidean distance (%)	# data points clustered incorrectly by k-means with a new PS-based distance (%)	# data points clustered incorrectly by k-means with a novel distance (%)
1	Pima Indians diabetes	768	8	2	34.9	37.24	30.73
2	Echocardiogram	131	7	2	32.82	33.59	31.3
3	Ecoli	336	7	8	38.69	17.26	17.26
4	Hayes-Roth	132	4	3	59.09	45.45	45.45
5	Statlog (heart)	270	13	2	40.74	39.26	37.78
6	Postoperative patient	90	8	3	57.78	35.56	28.89
7	Statlog (image segmentation)	2310	19	7	41.13	25.28	25.11
8	Glass	214	9	6	69.16	59.35	59.35
9	Vehicle	846	18	4	63.12	59.46	58.39
10	Yeast	1484	8	10	60.65	51.15	50.94
11	Wine	178	13	3	29.78	32.02	28.09

We observed that our proposed algorithm performed very well. We found that k-means with Euclidean algorithm failed to classify 48% of the average number of all instances in data sets while our proposed algorithm performed at 37%. We note that k-means with a new PS-based distance has an intermediate performance equal

to 39%. We can thus conclude that the k-means algorithm with a novel distance measure has better performance than k-means with Euclidean and the new PS-based distance measures.

5. Conclusion and future work

In this paper we described an essential problem in data clustering and presented some solutions for it. We investigated using distance measures other than the Euclidean type for improving the performance of clustering. We also developed a new distance measure and proved its efficiency. We developed a k-means algorithm with a novel distance measure, which improved the performance of the classical k-mean algorithm. The proposed algorithm did not have a worst-case bound on running time.

Experimental results were shown in this paper to demonstrate the effectiveness of the proposed algorithm by illustrating the time complexity and the performance of classifying some complex data sets.

The work reported in this paper may be extended in a number of ways, some of which are discussed below:

1. We used a new point symmetry-based distance measure instead of the Euclidean type for improving the performance of clustering. It would be interesting to investigate other kinds of distance measures.
2. We used a k d-tree for improving the performance of data clustering. Many optimizing search strategies in k d-trees are developed in the literature. We can use these strategies for improving the time complexity of our algorithms and to study their performance.
3. Our proposed algorithm depends on the k d-tree for improving the performance of clustering. It would be interesting to study some other kinds of trees like R+ _tree and Bkd-tree: a dynamic scalable k d-tree.

References

- [1] G. Gan, C. Ma, J. Wu, Data Clustering: Theory, Algorithms, and Applications (ASA-SIAM Series on Statistics and Applied Probability), SIAM, Philadelphia, ASA, Alexandria, 2007.
- [2] S. Theodoridis, K. Koutroumbas, Pattern Recognition, Elsevier Academic Press, Amsterdam, 2003.
- [3] K.K. Gündoğan, B. Alatas, A. Karci, "Mining classification rules by using genetic algorithms with non-random initial population and uniform operator", Turkish Journal of Electrical Engineering & Computer Sciences, Vol. 12, pp. 43–52, 2004.
- [4] T. Zhang, R. Ramakrishnan, M. Linvy, "BIRCH: an efficient method for very large databases", Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, 1996.
- [5] G. Hamerly, C. Elkan, "Alternatives to the k-means algorithm that find better clusterings", Proceedings of the Eleventh International Conference on Information and Knowledge Management, pp. 600–607, 2002.
- [6] P.S. Bradley, U.M. Fayyad, "Refining initial points for k-means clustering", ICML, Vol. 98, pp. 91–99, 1998.
- [7] J. MacQueen, "Some methods for classification and analysis of multivariate observations", Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Vol. 1, pp. 281–297, 1967.
- [8] A.K. Jain, M.N. Murty, P.J. Flynn, "Data clustering: a review", ACM Computing Surveys, Vol. 31, pp. 264–323, 1999.
- [9] S.J. Redmond, C. Heneghan, "A method for initialising the K-means clustering algorithm using k d-trees", Pattern Recognition Letters, Vol. 28, pp. 965–973, 2007.
- [10] K. Mumtaz, K. Duraiswamy, "A novel density based improved k-means clustering algorithm – Dbkmeans", International Journal on Computer Science and Engineering, Vol. 2, pp. 213–218, 2010.

- [11] M. Ester, H.P. Kriegel, J. Sander, X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise", Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Vol. 96, pp. 226–231, 1996.
- [12] F. Attneave, "Symmetry information, and memory for patterns", The American Journal of Psychology, Vol. 68, pp. 209–222, 1995.
- [13] M.C. Su, C.H. Chou, "A modified version of the k-means algorithm with a distance based on cluster symmetry", Pattern Analysis and Machine Intelligence, Vol. 23, pp. 674–680, 2001.
- [14] S. Bandyopadhyay, S. Saha, "GAPS: a clustering method using a new point symmetry-based distance measure", Pattern Recognition, Vol. 40, pp. 3430–3451, 2007.
- [15] S. Bandyopadhyay, S. Saha, "A point symmetry-based clustering technique for automatic evolution of clusters", Knowledge and Data Engineering, Vol. 20, pp. 1–17, 2008.
- [16] C.H. Chou, M.C. Su, E. Lai, "Symmetry as a new measure for cluster validity", Second WSEAS International Conference on Scientific Computation and Soft Computing, pp. 209–213, 2002.
- [17] A.W. Moore, "An introductory tutorial on *k*-d-trees", PhD, University of Cambridge, 1991.
- [18] C.L. Blake, E. Keough, C.J. Merz, "UCI repository of machine learning database", <http://www.ics.uci.edu/~mlearn/MLrepository.html>, 1998.
- [19] R. Bouckaert, E. Frank, M. Hall, R. Kirkby, P. Reutemann, A. Seewald, D. Scuse, "WEKA manual for version 3-6-2", University of Waikato, Hamilton, New Zealand, <http://www.cs.waikato.ac.nz/~ml/weka>, 2010.