Efficiency of k-Means and K-Medoids Algorithms for Clustering Arbitrary Data Points

Dr. T. VELMURUGAN
Associate professor,
PG and Research Department of Computer Science,
D.G.Vaishnav College, Chennai-600106, India.
E-Mail: velmurugan_dgvc@yahoo.co.in

Abstract

There are number of techniques proposed by several researchers to analyze the performance of clustering algorithms in data mining. All these techniques are not suggesting good results for the chosen data sets and for the algorithms in particular. Some of the clustering algorithms are suit for some kind of input data. This research work uses arbitrarily distributed input data points to evaluate the clustering quality and performance of two of the partition based clustering algorithms namely k-Means and k-Medoids. To evaluate the clustering quality, the distance between two data points are taken for analysis. The computational time is calculated for each algorithm in order to measure the performance of the algorithms. The experimental results show that the k-Means algorithm yields the best results compared with k-Medoids algorithm.

Key Words: k-Means Algorithm, K-Medoids Algorithm, Cluster Analysis, Arbitrary data points.

1. Introduction

Data Mining (DM) is the extraction of information from large amounts of data to view the hidden knowledge and facilitate the use of it to the real time applications. DM has a wide variety of algorithms for data analysis. Some of the major DM techniques used for analysis are Clustering, Association, Classification and etc. Clustering is an effective technique for exploratory data analysis, and has found applications in a wide variety of areas. Most existing methods of clustering can be categorized into three: partitioning, hierarchical, grid-based and model-based methods. Partition based clustering generates a partition of the data such that objects in a cluster are more similar to each other than they are to objects in other clusters. The k-Means[1, 5], EM[5], and k-medoids[6] are examples of partitional methods. Partitional algorithms have the advantage of being able to incorporate knowledge about the global shape or size of clusters by using appropriate prototypes and distance measures in the objective function [7, 8, 12,13].

Recently, the advent of World Wide Web search engines, the problem of organizing massive multimedia databases, and the concept of "data mining" large databases has led to renewal of interest in clustering and the development of new algorithms[9]. In many applications, clustering is used as an intermediate compression tool. First, the data is clustered, and then only the clusters' representatives are used for the analysis part. Some times the number of data objects in each and every cluster is used for analysis. But this research work discusses about the computational complexity of k-Means and k-Medoids algorithms. Arbitrarily distributed data points are given as input for analysis. The organization of the rest of the paper is as follows. In section 2, the two clustering algorithms and its basic concepts are discussed. In section 3 experimental results of the taken algorithms are tabulated and discussed. Summary of the experimental results are described in section 4. Finally, section 5 contains the conclusions.

2. Methodology

Cluster analysis groups data objects based on only information found in the data that describes the objects and their relationships. The goal is that objects within a group be similar (or related) to one another and different from (or unrelated to) the objects in other groups. The greater the similarity (or homogeneity) within a group and the greater the difference between the groups are, the better or more distinct the clustering. A large number of clustering algorithms have been developed in a variety of domains for different types of applications [2, 5]. None of these algorithms is suitable for all types of applications. This research work is carried out to compare the performance of k-Means and k-Medoids clustering algorithms based on the clustering result.
quality. The basic ideas and its concepts are explored in the following sections.

2.1. The k-Means Algorithm

The k-Means is one of the simplest unsupervised learning algorithms that solve the well-known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori [10, 11]. The main idea is to define k centroids, one for each cluster. These centroids should be placed in a cunning way because of different location causes different result. So, the better choice is to place them as much as possible far away from each other. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early group age is done. At this point it is necessary to re-calculate k new centroids as bar centers of the clusters resulting from the previous step. After obtaining these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this loop, one may notice that the k centroids change their location step by step until no more changes are done. In other words centroids do not move any more. Finally, this algorithm aims at minimizing an objective function

\[ J = \sum_{j=1}^{k} \sum_{i=1}^{n} \| x_i^{(j)} - c_j \|^2, \]

where \( \| x_i^{(j)} - c_j \| \) is a chosen distance measure between a data point \( x_i^{(j)} \) and the cluster centre \( c_j \), is an indicator of the distance of the \( n \) data points from their respective cluster centers. The algorithm is composed of the following steps:

1. Place \( k \) points into the space represented by the objects that are being clustered. These points represent initial group centroids.
2. Assign each object to the group that has the closest centroid.
3. When all objects have been assigned, recalculate the positions of the \( k \) centroids.
4. Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

The algorithm is significantly sensitive to the initial randomly selected cluster centers. The k-Means algorithm can be run multiple times to reduce this effect. k-Means is a simple algorithm that has been adapted to many problem domains and it is a good candidate to work for a randomly generated data points. One of the most popular heuristics for solving the k-Means problem is based on a simple iterative scheme for finding a locally minimal solution [3, 4, 10]. This algorithm is often called the k-Means algorithm.

2.2. The k-Medoids Algorithm

The k-Means algorithm is sensitive to outliers since an object with an extremely large value may substantially distort the distribution of data [11]. Instead of taking the mean value of the objects in a cluster as a reference point, a medoid can be used, which is the most centrally located object in a cluster. Thus, the partitioning method can still be performed based on the principle of minimizing the sum of the dissimilarities between each object and its corresponding reference point. This forms the basis of the k-Medoids method. The basic strategy of k-Medoids clustering algorithms is to find \( k \) clusters in \( n \) objects by first arbitrarily finding a representative object (the medoids) for each cluster. Each remaining object is clustered with the medoid to which it is the most similar. The k-Medoids method uses representative objects as reference points instead of taking the mean value of the objects in each cluster is the key point of this method. The algorithm takes the input parameter \( k \), the number of clusters to be partitioned among a set of \( n \) objects.

A typical k-Medoids algorithm for partitioning based on medoid or central objects is as follows:

**Input:** \( k \): The number of clusters  
D: A data set containing \( n \) objects

**Output:** A set of \( k \) clusters that minimizes the sum of the dissimilarities of all the objects to their nearest medoid.

**Method:** Arbitrarily choose \( k \) objects in \( D \) as the initial representative objects;

**Repeat** assign each remaining object to the cluster with the nearest medoid;  
randomly select a non medoid object \( O_{\text{random}} \);  
compute the total points \( S \) of swapping object \( O_i \) with \( O_{\text{random}} \);  
if \( S < 0 \) then swap \( O_i \) with \( O_{\text{random}} \) to form the new set of \( k \) medoid;

**Until** no change;
It attempts to determine k partitions for n objects. After an initial random selection of k medoids, the algorithm repeatedly tries to make a better choice of medoids [5, 11]. Therefore, the algorithm is often called as representative object based algorithm.

2.3. Distance Measure

An important step in most clustering is to select a distance measure, which will determine how the similarity of two elements is calculated. This will influence the shape of the clusters, as some elements may be close to one another according to one distance and farther away according to another. For example, in a 2-dimensional space, the distance between the point \((x = 1, y = 0)\) and the origin \((x = 0, y = 0)\) is always 1 according to the usual norms, but the distance between the point \((x = 1, y = 1)\) and the origin can be 2, \(\sqrt{2}\) or 1 if you take respectively the 1-norm, 2-norm or infinity-norm distance. Another important distinction is whether the clustering uses symmetric or asymmetric distances. Many of the distance functions listed above have the property that distances are symmetric (the distance from object A to B is the same as the distance from B to A). In other applications, this is not the case. A true metric gives symmetric measures of distance. The symmetric and 2-norm distance measure is used in this research work. In the Euclidean space \(R^n\), the distance between two points is usually given by the Euclidean distance (2-norm distance). The formula for 2-norm distance is

\[
2\text{-norm distance} = \left(\sum_{i=1}^{n}|x_i - y_i|^2\right)^{1/2}
\]

The 2-norm distance is the Euclidean distance, a generalization of the Pythagorean Theorem to more than two coordinates. It is what would be obtained if the distance between two points were measured with a ruler; the "intuitive" idea of distance. Based on this idea of finding the distance, the clustering qualities of the proposed algorithms are analyzed here.

3. Experimental results

Having introduced both the clustering algorithms and their basic mathematical foundations, now it is the time to discuss the technique on the basis of experimental approach. This approach involves the implementation of the k-Means and k-Medoids algorithm introduced previously and testing its performance on the basis of its quality of clustering. The program is written in JAVA programming language via applet windows. The data points are created by three ways in the applet window. The data points are created arbitrarily by pressing the mouse buttons in the applet window for this research work. The other two types are statistical distributions (Uniform and Normal), they are not used here[12, 13]. After the creation of data points, the user has to specify the number of clusters by fixing the k value (the number of clusters). The number of clusters is restricted to a maximum of 20 by the program. Before executing the program, the input data points are displayed in red color. After the execution, the results are displayed by different colors. In the right side of the output window, below the 'Cluster Results' the number of data points for each cluster and the time for the execution of the same cluster is displayed in milliseconds (ms). One of the samples of 200 arbitrarily distributed input data points is shown in fig. 1 for k-Means algorithm. The resulting clusters of the same data points are displayed in fig. 2 after the execution of the program. The center point of each cluster is displayed in white color. The number of clusters and the data points are given by the user during the execution of the program as input. The number of data points is 200 and the number of clusters is 5 (k = 5) in this case. The algorithm is repeated 200 times (one iteration for each data point) to get efficient output. The cluster centers (centroids) are fixed by the algorithm for each cluster and clusters are formed depending upon the distance between data points. For different types of input data points, the algorithm gives different outputs.

![Figure 1. Input Data Points for k-Means algorithm](image-url)
is the size of the cluster. In figure 2, C1 means cluster one and the number 43 is the number of data points in C1. T1 means execution time for the first cluster and it is found that 31 ms. If the numbers of data points are 200 then the algorithm repeats the same two hundred times. For each data point, the algorithm executes once. The total elapsed time to cluster all the 200 data points and 5 clusters is 172 ms for the arbitrary distribution of input data points. The sum of the individual cluster time is 172 ms. The difference between these two is zero ms in this case. This is the execution time for the other codes in the program except the algorithmic steps. Sometimes it is zero by the behavior of the JAVA threats. Due to page limitation, the same algorithm is executed five times only and the results are tabulated in table 1. The total elapsed time for all clusters is given at the end of the row ‘Size’ under the column ‘TOT’. The sum of the execution time for each cluster (individual execution time for each cluster) is given at the end of the row ICT. The difference between these two is listed in the last column named as ‘DT’ (Difference Time). Like, k-Medoids algorithm is executed and one of the results is shown in fig. 3. Five times the algorithm is executed and the results are shown in table 2 for the same algorithm.
4. Summary

The results of both the algorithms are analyzed based on the number of data points and the computational time of each algorithm. The behavior of the algorithm is analyzed by observations. The number of data points is clustered by the algorithm as per the distribution of arbitrary shapes of the data points. Time complexity analysis is a part of computational complexity theory that is used to describe an algorithm’s use of computational resources; in this case, the best case and the worst case running time expressed. From table 1, the maximum and minimum time taken by the k-Means algorithm is 172 and 156 respectively. Like, from table 2, 221 and 196 are the maximum and minimum time taken by the k-Medoids algorithm. The performance of the algorithms have been analyzed for several executions by considering different data points (for which the results are not shown) as input (300 data points, 400 data points etc.) and the number of clusters are 10 and 15 (for which also the results are not shown), the outcomes are found to be highly satisfactory. Figure 4 shows that the graph of the average results of the distribution of data points. The average execution time is taken from the tables 1 and 2. It is easy to identify from the figure 4 that there is a difference between the times of the algorithms. Here, it is found that the average execution time of the k-Means algorithm is very less by comparing the k-Medoids algorithm.

Table 1. K-Means Results

<table>
<thead>
<tr>
<th>Run \ Cluster</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>TOT</th>
<th>DT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>43</td>
<td>39</td>
<td>49</td>
<td>30</td>
<td>39</td>
<td>172</td>
<td>0</td>
</tr>
<tr>
<td>ICT</td>
<td>31</td>
<td>15</td>
<td>32</td>
<td>32</td>
<td>62</td>
<td>172</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>28</td>
<td>46</td>
<td>29</td>
<td>46</td>
<td>51</td>
<td>156</td>
<td>8</td>
</tr>
<tr>
<td>ICT</td>
<td>31</td>
<td>31</td>
<td>10</td>
<td>16</td>
<td>60</td>
<td>148</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>37</td>
<td>44</td>
<td>41</td>
<td>36</td>
<td>42</td>
<td>165</td>
<td>9</td>
</tr>
<tr>
<td>ICT</td>
<td>31</td>
<td>15</td>
<td>31</td>
<td>32</td>
<td>47</td>
<td>156</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>52</td>
<td>45</td>
<td>28</td>
<td>37</td>
<td>38</td>
<td>162</td>
<td>0</td>
</tr>
<tr>
<td>ICT</td>
<td>31</td>
<td>64</td>
<td>15</td>
<td>16</td>
<td>46</td>
<td>162</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Size</td>
<td>33</td>
<td>33</td>
<td>38</td>
<td>43</td>
<td>53</td>
<td>163</td>
<td>10</td>
</tr>
<tr>
<td>ICT</td>
<td>26</td>
<td>37</td>
<td>31</td>
<td>16</td>
<td>43</td>
<td>153</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. K-Medoids Results
5. Conclusions

From the experimental approach, for the proposed two algorithms in this research work, the obtained results are discussed. The choice of clustering algorithm depends on both the type of data available and on the particular purpose and chosen application. Usually the time complexity varies from one processor to another processor, which depends on the speed and the type of the system. The partitioning based algorithms work well for finding spherical-shaped clusters in small to medium-sized data points. The efficiency of the algorithms for the arbitrary distributions of data points is analyzed by various executions of the programs. Finally, this research work concludes that the computational time of k-Means algorithm is less than the k-Medoids algorithm for the chosen application. Hence, the efficiency of k-Means algorithm is better than the k-Medoids algorithm.

References


