Available online at www.sciencedirect.com

ScienceDirect

Expert Systems with Applications 36 (2009) 3336–3341

Expert Systems with Applications

www.elsevier.com/locate/eswa

A simple and fast algorithm for K-medoids clustering

Hae-Sang Park, Chi-Hyuck Jun *

Department of Industrial and Management Engineering, POSTECH, San 31 Hyoja-dong, Pohang 790-784, South Korea

Abstract

This paper proposes a new algorithm for K-medoids clustering which runs like the K-means algorithm and tests several methods for selecting initial medoids. The proposed algorithm calculates the distance matrix once and uses it for finding new medoids at every iterative step. To evaluate the proposed algorithm, we use some real and artificial data sets and compare with the results of other algorithms in terms of the adjusted Rand index. Experimental results show that the proposed algorithm takes a significantly reduced time in computation with comparable performance against the partitioning around medoids.

Keywords: Clustering; K-means; K-medoids; Rand index

1. Introduction

Clustering is the process of grouping a set of objects into clusters so that objects within a cluster are similar to each other but are dissimilar to objects in other clusters (Han, Kamber, & Tung, 2001). K-means clustering (MacQueen, 1967) and partitioning around medoids (Kaufman & Rousseeuw, 1990) are well known techniques for performing non-hierarchical clustering. K-means clustering iteratively finds the k centroids and assigns every object to the nearest centroid, where the coordinate of each centroid is the mean of the coordinates of the objects in the cluster. Unfortunately, K-means clustering is known to be sensitive to the outliers although it is quite efficient in terms of the computational time. For this reason, K-medoids clustering are sometimes used, where representative objects called medoids are considered instead of centroids. Because it is based on the most centrally located object in a cluster, it is less sensitive to outliers in comparison with the K-means clustering. Among many algorithms for K-medoids clustering, partitioning around medoids (PAM) proposed by Kaufman and Rousseeuw (1990) is known to be most powerful. However, PAM has a drawback that it works inefficiently for a large data set due to its time complexity (Han et al., 2001). This is the main motivation of this paper. We are interested in developing a new K-medoids clustering algorithm that should be simple but efficient.

There have been some efforts in developing new algorithms for K-medoids clustering. Kaufman and Rousseeuw (1990) also proposed an algorithm called CLARA, which applies the PAM to sampled objects instead of all objects. It is reported by Lucasius, Dane, and Kateman (1993) that the performance of CLARA drops rapidly below an acceptable level with increasing number of clusters. Lucasius et al. (1993) proposed a new approach of K-medoid clustering using a genetic algorithm, whose performance is reported as better than CLARA but computational burden increases as the number of clusters increases. Wei, Lee, and Hsu (2003) also compared performance of CLARA and some other variants for large data sets. Ng and Han (1994) proposed an efficient PAM-based algorithm, which updates new medoids from some neighboring objects. van der Laan, Pollard, and Bryan (2003) tried to maximize the silhouette proposed by Rousseeuw (1987) instead of minimizing the sum of distances to the closest medoid in PAM. Zhang and Couloigner (2005) suggested a K-medoid algorithm which utilizes triangular irregular network concept when calculating the total cost of the replacement in swap step of PAM to reduce the computational time. Most

* Corresponding author. Tel.: +82 54 279 2197; fax: +82 54 279 2870.
E-mail addresses: shoo359@postech.ac.kr (H.-S. Park), chjun@postech.ac.kr (C.-H. Jun).

0957-4174/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2008.01.039

of these algorithms are based on PAM, so the computational burden still remains.

The remaining parts of this paper are organized as follows: The proposed method is introduced in the next section and performance comparison is presented for two real data sets and some artificial data sets. Other methods to find initial medoids are discussed and finally conclusions are given.

2. Proposed K-medoids algorithm

Suppose that n objects having p variables each should be grouped into k ($k < n$) clusters, where k is assumed to be given. Let us define jth variable of object i as X_{ij} ($i = 1,\ldots,n$; $j = 1,\ldots,p$). The Euclidean distance will be used as a dissimilarity measure in this study although other measures can be adopted. The Euclidean distance between object i and object j is given by

$$d_{ij} = \sqrt{\sum_{a=1}^{p}(X_{ia} - X_{ja})^2} \quad i = 1,\ldots,n;\ j = 1,\ldots,n \tag{1}$$

It should be noted that the above Euclidean distance will be adopted in K-means and PAM algorithms in this study.

The proposed algorithm is composed of the following three steps.

Step 1: (Select initial medoids)
- 1-1. Calculate the distance between every pair of all objects based on the chosen dissimilarity measure (Euclidean distance in our case).
- 1-2. Calculate v_j for object j as follows:

$$v_j = \sum_{i=1}^{n} \frac{d_{ij}}{\sum_{l=1}^{n} d_{il}}, \qquad j = 1, \ldots, n \tag{2}$$

- 1-3. Sort v_j's in ascending order. Select k objects having the first k smallest values as initial medoids.
- 1-4. Obtain the initial cluster result by assigning each object to the nearest medoid.
- 1-5. Calculate the sum of distances from all objects to their medoids.

Step 2: (Update medoids)
Find a new medoid of each cluster, which is the object minimizing the total distance to other objects in its cluster. Update the current medoid in each cluster by replacing with the new medoid.

Step 3: (Assign objects to medoids)
- 3-1. Assign each object to the nearest medoid and obtain the cluster result.
- 3-2. Calculate the sum of distance from all objects to their medoids. If the sum is equal to the previous one, then stop the algorithm. Otherwise, go back to the Step 2.

The above algorithm is a local heuristic that runs just like K-means clustering when updating the medoids. In Step 1, we proposed a method of choosing the initial medoids. This method tends to select k most middle objects as initial medoids. The performance of the algorithm may vary according to the method of selecting the initial medoids. We will consider some other possibilities of choosing the initial medoids and their performance will be compared with each other through simulation study in Section 3.4.

3. Numerical experiments

In order to see the performance of the proposed method, we first applied the method to two real data sets, 'Iris' data and 'Soybean' data, whose true classes are known. Performance was measured by the accuracy, which is the proportion of objects that are correctly grouped together against the true classes. To investigate the performance more objectively, a simulation study was carried out by generating artificial data sets repetitively and calculating the average performance of the method.

3.1. Iris data

The Iris data set is available in UCI repository (ftp://ftp.ics.uci.edu/pub/machine-learning-databases/), which set includes 150 objects (50 in each of three classes – 'Setosa', 'Versicolor', and 'Virginica') having four variables ('sepal length', 'sepal width', 'petal length', and 'petal width'). We applied the proposed method and K-means with $k = 3$ to this data without using the class information. When implementing K-means, the initial centroids were chosen randomly although many other alternatives are available including Al-Daoud and Roberts (1996), Khan and Ahmad (2004), etc.

The class of an object cannot be predicted by a clustering algorithm but it may be estimated by examining the cluster result for the class-labeled data. Table 1 shows the confusion matrix by K-means clustering method, whereas Table 2 shows the result by the proposed method. The clustering accuracy against the true classes by K-means is

Table 1
Cluster result of iris data by K-means

From algorithm			
True	Setosa	Versicolor	Virginica
Setosa	50	0	0
Versicolor	0	47	3
Virginica	0	14	36

Table 2
Cluster result of iris data by the proposed method

From algorithm			
True	Setosa	Versicolor	Virginica
Setosa	50	0	0
Versicolor	0	41	9
Virginica	0	3	47