A Fast Approximated k–Median Algorithm

Eva Gómez–Ballester, Luisa Micó, Jose Oncina *

Universidad de Alicante, Departamento de Lenguajes y Sistemas Informáticos
{eva, mico,oncina}@dlsi.ua.es

Abstract. The *k-means* algorithm is a well–known clustering method. Although this technique was initially defined for a vector representation of the data, the set median (the point belonging to a set P that minimizes the sum of distances to the rest of points in P) can be used instead of the mean when this vectorial representation is not possible. The computational cost of the set median is $O(|P|^2)$. Recently, a new method to obtain an approximated median in $O(|P|)$ was proposed. In this paper we use this approximated median in the *k–median* algorithm to speed it up.

1 Introduction

Given a set of points P, the k-clustering of P is defined as the partition of P into k distinct sets (clusters)[2]. The partition must have the property that the points belonging to each cluster are *most similar*.

Clustering algorithms may be divided in different categories [9], although in this work we are interested in clustering algorithms based on cost function optimization. Usually, in this type of clustering, the number of clusters k is kept fixed.

A well–known algorithm based on cost function optimization is the *k–means* algorithm described in figure 1.

In particular, the *k–means* algorithm finds locally optimal solutions using as cost function the sum of the distances between each point and its nearest cluster center (the mean). This cost function can be formuled as

$$J(C) = \sum_{i \in k} \sum_{p \in C_i} ||p - m_i||^2, \tag{1}$$

where $C = \{C_1, C_2 \ldots, C_k\}$ and $m_1, m_2 \ldots m_k$ are, respectively, the clusters and the means of each cluster.

Although the *k–means* algorithm was developed for a vector representation of data (the defined cost function uses the euclidean distance), a more general definition can be used for data where a no vector representation is possible, or is not a good alternative. For example, in character recognition, speech recognition or any application of syntactic recognition, data can be represented using strings.

* The authors thank the Spanish CICyT for partial support of this work through project TIC2000–1703–CO3–02

algorithm k–Means Clustering
input: P : set of points; k : number of classes;
output: $m_1, m_2 \ldots m_k$
select the initial cluster representatives $m_1, m_2 \ldots m_k$
do
  classify the $|P|$ points according to nearest m_i
  recompute $m_1, m_2 \ldots m_k$
until no change in $m_i \ \forall i$
end algorithm

Fig. 1. The *k–means algorithm*

For this type of problems, the median string can be used to be the representative of each class.

The median can be obtained from a constrained point set P (*set median*) or can be extended to the whole space U where the points are extracted (*generalized median*).

Given a set of points P, the *generalized median string* m is defined as the point (in the whole space U) that minimizes the sum of distances to P:

$$m = argmin_{p \in U} \sum_{p' \in P} d(p, p') \tag{2}$$

When the points are strings and the edit distance is used, this is a NP-Hard problem [4]. A simple greedy algorithm to compute an approximation to the median string was proposed in [6].

When the median is selected among the points belonging to the set, the *set median* is obtained. In this case the median is defined as the point (in the set P) that minimizes de sum of distances to P. In this case, the search of the median is constrained to a finite set of points:

$$m = argmin_{p \in P} \sum_{p' \in P} d(p, p') \tag{3}$$

Unlike the *generalized median string*, the cost of computing the *set median* (using points or strings) is for the best known algorithm $O(|P|^2)$.

When the set median is used in *k–means* algorithm instead of the mean, the new cost function is:

$$J(C) = \sum_{i \in k} \sum_{p \in C_i} d(p, m_i), \tag{4}$$

where $m_1, m_2 \ldots m_k$ are the k medians obtained with the equation (3) leading to the *k–median* algorithm.

Recently, a fast algorithm to obtain an approximated median was proposed [7]. The main characteristics of this algorithm are: 1) no assumption about the structure of the points or the distance function are made and 2) it has a linear time complexity. The experiments showed that very accurate medians can be obtained using appropriate parameters.

In this paper this approximated median algorithm has been used in the *k–median* algorithm instead of the median. So this new approximation can be used in problems, as exploratory data analysis, where data are represented by strings.

Depending on the initialization of the *k–median* algorithm the behaviour is different. Some different initializations of the algorithm have been proposed in the literature ([1][8]). In this work we have used the most simple initialization that consists in selecting randomly k cluster representatives.

In the next section this approximated median algorithm is described. Some experiments using synthetic and real data to compare the approximated and the exact set median are shown in section 3 and finally conclusions are drawn in section 4.

2 The approximated median search algorithm

Given a set of points P, the algorithm selects as median the point that minimizes the sum of distances of a subset of the complete set ([7]).

The algorithm has two main steps: 1) a subset of n_r points (*reference points*) from the set P is randomly selected. It is very important to select randomly the reference points because it is expected that they have a similar behaviour of the whole set (some experiments were made to support this conclusion in [7]). The sum of distances from each point in P to the reference points are calculated and stored. 2) The n_t points of P whose sum of distances is lowest are selected (*test points*). For each test point, the sum of distances to every point belonging to P is calculated and stored. The test point that minimizes this sum is selected as the median of the set. The algorithm is described in figure 2. The algorithm needs two parameters: n_r and n_t. Experiments reported in [7] show that the choice $n_r = n_t$ is reasonable. Moreover, the use of a small number of n_r and n_t points is enough to obtain accurate medians.

3 Experiments

A set of experiments was made to know the behaviour of the *approximated k–median* in relation to the *k–median* algorithm.

As the objective in the *k–median* algorithm is the minimization of the cost function, the cost function and the expended time have been studied with both algorithms.

Two main groups of experiments with synthetic and real data were made. All experiments (real and synthetic) were repeated 30 times using different random initial medians. Deviations were always below 4% and are not plotted in the figures.

algorithm approximated median
input:
  P : set of points; $d(\cdot,\cdot)$: distance function;
  n_r : number of reference points; n_t : number of test points
output:
  $m \in P$: median
var:
  U : used points (reference and test); T : test points;
  PS : array of $|P|$ partial sums; FS : array of $|P|$ full sums
// **Initialization**
$U = \emptyset$
$\forall p \in P$
  $PS[p] = 0$
// **Selecting the reference points**
repeat n_r times
  $u =$ random point in $P - U$
  $U = U \cup \{u\}$
  $FS[u] = PS[u]$
  $\forall p \in P$
    $d = d(p, u)$
    $PS[p] = PS[p] + d$
    $FS[u] = FS[u] + d$
// **Selecting the test points**
$T = n_t$ points in $P - U$ that minimize $PS[\cdot]$
// **Calculating the full sums**
$\forall t \in T$
  $FS[t] = PS[t]$
  $U = U \cup \{t\}$
  $\forall p \in P - U$
    $d = d(t, p)$
    $FS[t] = FS[t] + d$
    $PS[p] = PS[p] + d$
// **Selecting the median**
$m =$ the point in U that minimizes $FS[\cdot]$
end algorithm

Fig. 2. The approximate median search algorithm.

3.1 Experiments with synthetic data

To generate synthetic clustered data, the algorithm proposed in [5] has been used. This algorithm generates random synthetic data from different classes (clusters) with a given maximum overlap between them. Each class follows a Gaussian distribution with the same variance and different randomly chosen means. For the experiments presented in this work, syntethic data from 4 and 8 classes was generated with dimensions 4 and 8. The overlap was set to 0.05 and the variance to 0.03.

The first set of experiments was designed to study the evolution of the cost function in each iteration for *k-median* and *approximated k–median* algorithms. 2048 points were generated using 4 and 8 clusters from spaces of dimension 4 and 8 (512 and 256 points respectively for each class) (see figure 3). In the *approximated k–median* three different sizes of the *used points* ($n_r + n_t$) were used (40, 80 and 320). As shown in this figure, the behaviour of the cost function is similar in all the cases. It is important to note that the *approximated k–median* algorithm computes much less distances than the *k–median*. For example, in the experiment with 4 classes, the number of distance computations in one iteration is around 500,000 for the *k–median* and 80,000 for the *approximated k–median* algorithm with 40 *used points*.

Note that the cost function decreases quickly and, afterwards, stays stable until the algorithm finishes. Then, the stop criterion can be relaxed to stop the iteration when the cost function changes less than a threshold. Using a 1% threshold in the experiments of figure 3, all the experiments would be stopped around the tenth iteration.

Table 1. Expended time (in seconds) by the *k–median* and the *approximated k–median* algorithm for a set of 2048 prototypes using the 1% threshold stop criterion.

Dimensionality 4				
Number of classes	*k–m*	*ak–m (40)*	*ak–m (80)*	*ak–m (320)*
4	2.86	0.29	0.54	1.82
8	3.11	0.54	1.01	1.92

Dimensionality 8				
Number of classes	*k–m*	*ak–m (40)*	*ak–m (80)*	*ak–m (320)*
4	2.09	0.25	0.44	1.46
8	2.16	0.46	0.86	1.74

In table 1 the expended time by both algorithms are represented for a set of 2048 points. In the *approximated k–median* three different numbers of *used points* were used (40, 80 and 320) with the 1% threshold stop criterion. The time was measured on a Pentium III running at 800 MHz under a Linux system.

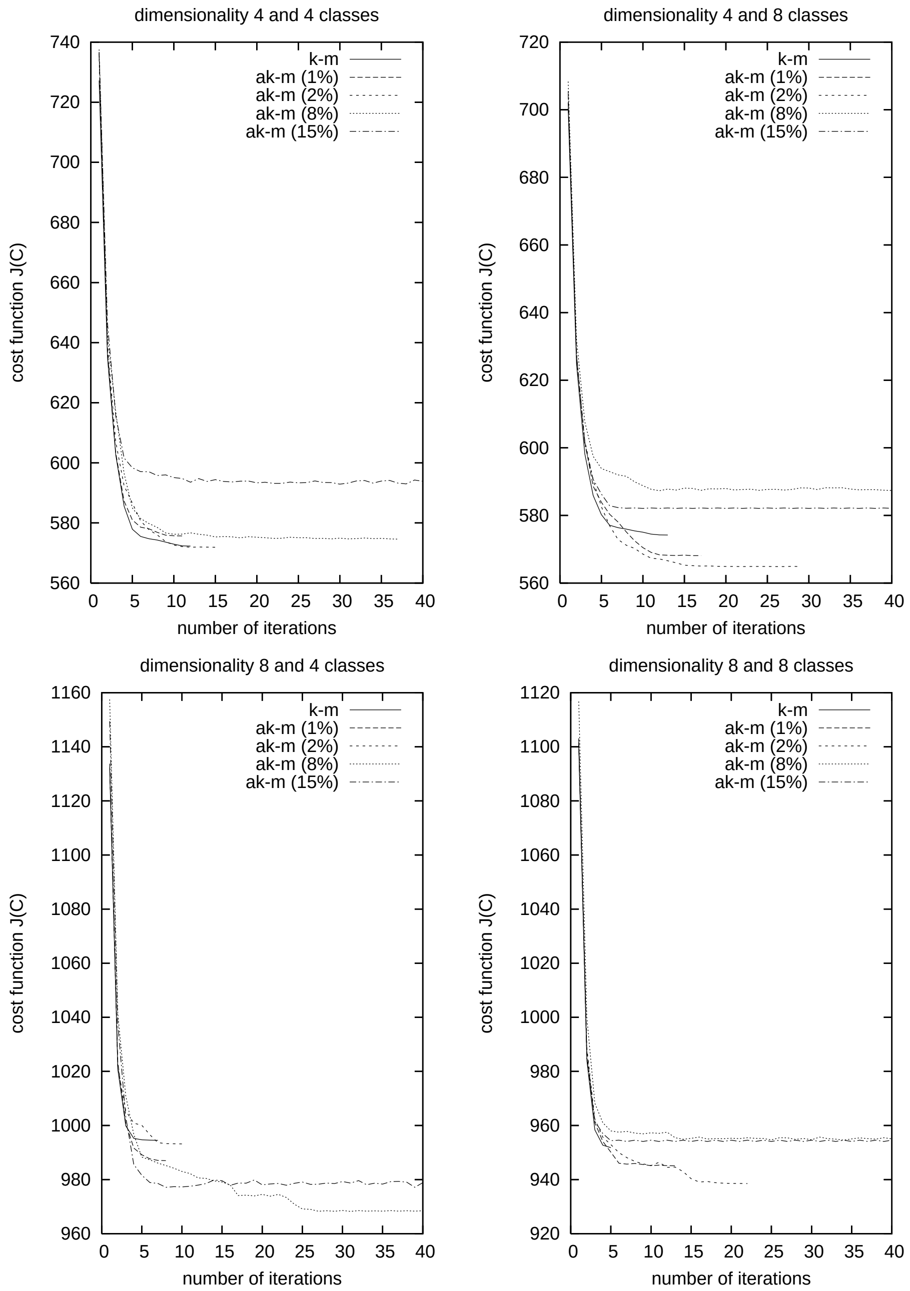

Fig. 3. Comparison of the cost function using 4 and 8 classes for a set of 2048 points with dimensionality 4 and 8.

Table 1 shows that the time expended by the *approximated k–median* using any of the three different sizes of *used points*, is much lower than the *k–median* algorithm.

In the last experiment with synthetic data, the expended time was measured with different set sizes. Figure 4 illustrates that the use of the *approximated k–median* reduces drastically the expended time when the set size increases.

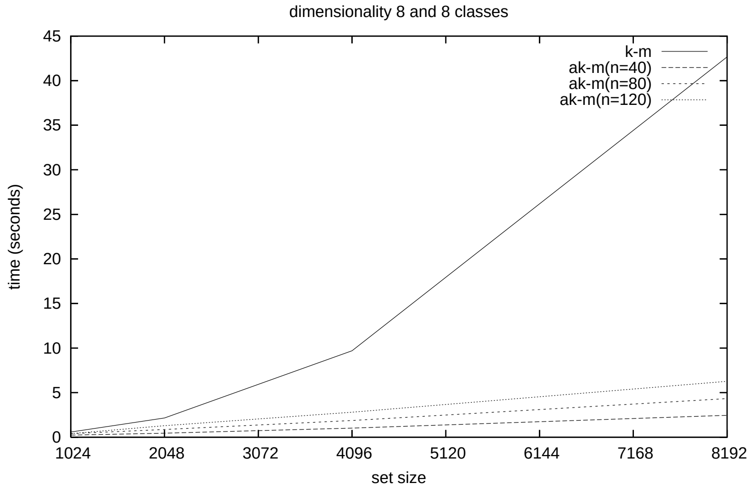

Fig. 4. Expended time (in seconds) by the *k–median* and the *approximated k–median* algorithm when the size of the set increases using the 1% threshold stop criterion.

3.2 Experiments with real data

For real data, a chain code description of the handwritten digits (10 writers) of the NIST Special Database 3 (National Institute of Standards and Technology) was used. Each digit has been coded as an octal string that represents the contour of the image.

The edit distance [3] with deletion and insertion cost set to 1 was used. The substitution costs are proportional to the relative angle between the directions (in particular 1, 2, 3 and 4 were used).

As can be seen in figure 5, the results are similar to the synthetic data experiments. Moreover, as it was said previously for synthetic data, the process for the *approximated k–median* can also be stopped when the value of the cost function between two consecutive iterations is lower than a threshold. These results are showed in figure 6.

As figure 6 shows, a relatively low number of used points (20 and 40) is enough to obtain a similar behaviour of the cost function with different set sizes.

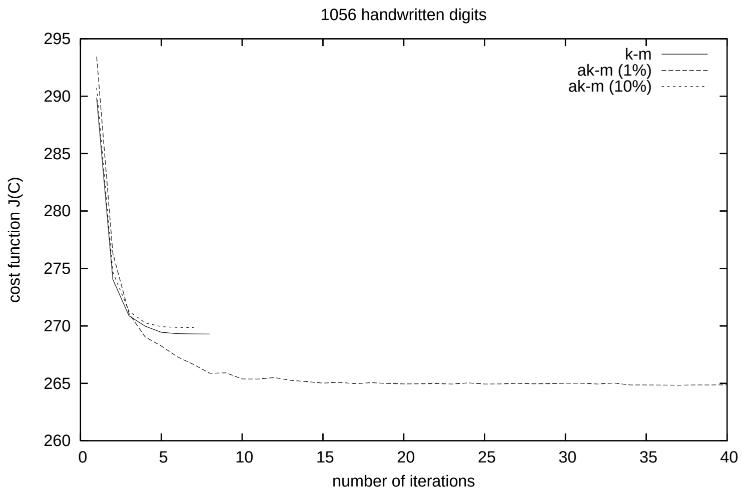

Fig. 5. Comparison of the cost function using 1056 digits.

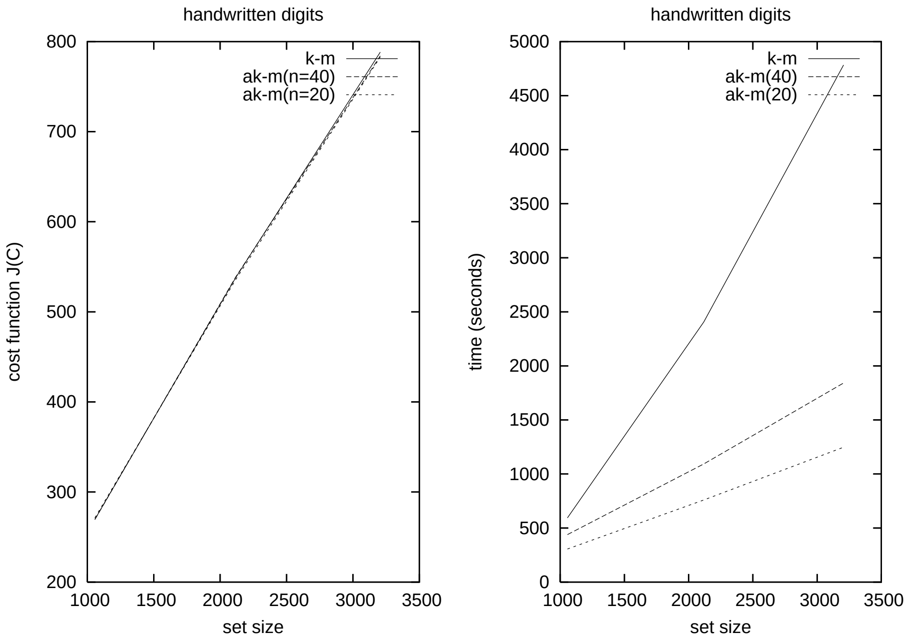

Fig. 6. . Cost function and expended time (in seconds) when the size of the set increases using the 1% threshold stop criterion.

Moreover, the expended time grows linearly with the set size in the *approximated k–median*, while this increase is quadratic for the *k–median* algorithm.

4 Conclusions

In this work an effective fast *approximated k–median* algorithm is presented. This algorithm has been obtained using an approximated set median instead of the set median in the *k–median* algorithm. As the computation of approximated medians is very fast, its use speeds the *k–median* algorithm. The experiments show that a low number of *used points* in relation to the complete set is enough to obtain similar value for the cost function in the *approximated k–median*.

In future works we will develop some ideas related to the stop criterion and the initialization of the algorithm.

Acknowledgement: The authors wish to thank to Francisco Moreno–Seco for permit us the use of his synthetic data generator.

References

1. Bradley, P.S., Fayyad, U.M.: Refining Initial Points for K–Means Clustering. Proc. 15th International Conf. on Machine Learning (1998).
2. Duda, R., Hart, P., Stork, D.: Pattern Classification. Wiley (2001).
3. Fu, K.S.: Syntactic Pattern Recognition and Applications. Prentice–Hall, Englewood Cliffs, NJ (1982).
4. de la Higuera, C., Casacuberta, F.: The topology of strings: two NP–complete problems. Theoretical Computer Science **230** 39–48 (2000).
5. Jain, A.K., Dubes, R.C.: Algorithms for clustering data. Prentice-Hall (1988).
6. Martínez, C., Juan, A., Casacuberta, F.: Improving classification using median string and nn rules. In: Proceedings of IX Simposium Nacional de Reconocimiento de Formas y Anlisis de Imgenes, 391–394 (2001).
7. Micó, L., Oncina, J.: An approximate median search algorithm in non–metric spaces. Pattern Recognition Letters **22** 1145–1151 (2001).
8. Peña, J.M., Lozano, J.A., Larrañaga, P.: An empirical comparison of four initialization methods for the K–means algorithm. Pattern Recognition Letters **20** 1027–1040 (1999).
9. Theodoridis, S., Koutroumbas, K.: Pattern Recognition. Academic Press (1999).