Fast Median Search in Metric Spaces

A. Juan and E. Vidal
Departamento de Sistemas Informáticos y Computación,
Universidad Politécnica de Valencia, 46071 Valencia, Spain.
E-mail: ajuan@dsic.upv.es

Abstract. Searching for a *median* of a set of patterns is a well-known technique to model the set or to aid searching for more accurate models such as a *generalized median* or a *k-median*. While medians and generalized medians are (relatively) easy to compute in the case of Euclidean representation spaces, this is no longer true when more complex distance measures are to be used. In these cases, a direct method to perform median search is not feasible for large sets of patterns. To cope with this computational problem, we proposed a technique which is significantly faster than the direct approach, both in terms of distance computations and overhead (time not alloted to distance computing). We also proposed another technique which is even faster than its predecessor in terms of distance computations, though it involves significantly more overhead. In this paper we introduce a generalized algorithm for fast median search which includes these two techniques as particular cases. We also develop a new particular case of this generalized algorithm.

Key Words: Set Median, Fast Search, Distance, Metric Spaces.

1 Introduction

Assume that we are given a set of patterns in an arbitrary (non-vector) representation space and a distance function to measure the dissimilarity between any pair of them. A well-known technique for modelling the given set consists of finding a pattern of the set whose sum of distances to all patterns is minimum. Such a pattern is called a *set median* (simply *median* in what follows).

The concept of median is related to a more general concept known as *generalized median*. This generalization arises when the search is not constrained to the given set, but extended to the whole pattern representation space. Although a generalized median constitutes a more accurate model than a median, finding such a model is often a difficult task that requires complex and specific (approximate) algorithms [1]. Obviously,

here we are excluding trivial cases like that of computing the generalized median of a set of patterns in a Euclidean representation space. In these cases where such computation is not trivial, a median can be used as an approximate generalized median or as a starting point for finding better ones [5].

Another generalization of the concept of median is known as *k-median*. As its name suggests, searching for a k-median consists of finding a subset of k patterns (models) such that the sum of distances between all the patterns and their corresponding nearest models is minimum. This problem, well-known as a *prototype location problem*, is NP-Hard [6]. On the other hand, it can be seen as a clustering problem since each possible subset of k patterns induces a partition of the set into k clusters and a measure of its quality. Good partitions are generally found using a simple *k-means-like* algorithm which starts from a given initial partition and then loops searching for a median of each cluster and reclassifying patterns according to their nearest medians [3]. Thus, the concept of median also turns out to be useful in finding more accurate models.

Searching for a median of a set of n patterns is a relatively easy task since it suffices to compute $\frac{1}{2}n(n-1)$ pairwise distances between patterns. However, this naive approach is no longer applicable when large sets of patterns or computationally expensive distance functions are involved. To cope with this problem, we proposed a technique called *Fast Median Search Algorithm (FMSA)*, which is significantly faster then the direct approach, both in terms of distance computations and *overhead* (time not alloted to distance computing) [2]. Recently, we have proposed a new FMSA version which is even faster than its predecessor *in terms of distance computations*, though it involves significantly more overhead [4]. In this paper we introduce a *generalized* FMSA which includes previous FMSA versions as particular cases. We also develop a new particular case of this generalized FMSA which is tested through computer simulations.

2 The Fast Median Search Algorithm

Given a metric space (E, d) and a set of n patterns $P \subset E$, searching for a median consists of finding a prototype $p^* \in P$ that minimizes

$$f(p) = \sum_{p' \in P} d(p, p')$$

for all $p \in P$. The (generalized) Fast Median Search Algorithm (FMSA) efficiently solves this problem as is formally described in Figure 1 and discussed below.

Assume that a lower bound function g has been defined for f. The FMSA partitions P into sets U (*Used*), A (*Alive*) and E (*Eliminated*). The set A is used to keep track of those patterns whose sums have not been calculated and are candidates for being a median (initially $A = P$). Patterns in this set are selected in turn using a g-guided strategy which chooses the pattern having minimum g first. Once a pattern s is selected, $f(s)$ is computed and it is transferred from A to U. Notice that the FMSA carries out this computation using a linear array F which stores $\sum_{u \in U} d(\bar{u}, u)$ for each unused pattern $\bar{u}$. If $f(s)$ is smaller than those sums previously computed, both the *current median*, p^*, and its associated sum, f^*, are updated. Then, g is computed for all alive patterns and those whose lower bounds are not smaller than f^* are transferred from A to E; that is, they will no further be considered as candidates for being a median. This process ends when no patterns remain candidates for being a median ($A = \emptyset$).

For any alive pattern a, define

$$g_1(a) = \sum_{u \in U_t} d(a, u) + \sum_{\bar{u} \in P - U_t} \max_{u \in U_t} |d(a, u) - d(u, \bar{u})| \tag{1}$$

$$g_2(a) = \max_{1 \le i \le t} \left(\sum_{u \in U_i} d(a, u) + \sum_{\bar{u} \in P - U_i} |d(a, s_i) - d(s_i, \bar{u})| \right) \tag{2}$$

$$g_3(a) = \max_{1 \le i \le t} \left(\sum_{u \in U_i} d(a, u) + \left| |P - U_i| d(a, s_i) - \sum_{\bar{u} \in P - U_i} d(s_i, \bar{u}) \right| \right) \tag{3}$$

where t denotes the current iteration and $U_i = \{s_1, s_2, \ldots, s_i\}$ is the set of used (selected) patterns until iteration i, $1 \le i \le t$. Then,

$$g_3(a) \le g_2(a) \le g_1(a) \le f(a) \tag{4}$$

To see this, we first note that, by the triangle inequality,

$$\max_{u \in U_i} |d(a, u) - d(u, \bar{u})| \le d(a, \bar{u}) \tag{5}$$

Algorithm FMSA

Input: $d : E \times E \rightarrow \mathbf{R}$; $P \subset E, n = |P|$ /* d is a *metric* on E */

Output: $p^* \in P$; $f^* \in \mathbf{R}$ /* a median and its sum of distances */

Variables: $U, A, E \subset P$; $F \in \mathbf{R}^n$; $s, s', a, p \in P$; $dsp, g^*, ga \in \mathbf{R}$

Function: $g : P \rightarrow \mathbf{R}$ /* lower bound function (to be defined) */

Method:

$A = P; U = E = \emptyset; f^* = \infty; F = \mathbf{0}; s' = rand(A)$

while $|A| > 0$ **do**

$s = s'; A = A - \{s\}; U = U \cup \{s\}$ /* s is the *selected* pattern */

$\forall p \in A \cup E$ **do** /* computing of the sum of distances of s */

$dsp = d(s, p); F_s = F_s + dsp; F_p = F_p + dsp$

end$\forall$

if $F_s < f^*$ **then** /* updating of p^* and f^* */

$p^* = s; f^* = F_s$

endif

$g^* = \infty$

$\forall a \in A$ **do**

$ga = g(a)$ /* computing of lower bounds */

if $ga \geq f^*$ **then** /* elimination */

$A = A - \{a\}; E = E \cup \{a\}$

else /* selection */

if $ga < g^*$ **then** $s' = a; g^* = ga$ **endif**

endif

end$\forall$

endwhile

Fig. 1. Fast Median Search Algorithm (FMSA)

for any $\bar{u} \in P - U_i$, $1 \leq i \leq t$. Thus for any i we have

$$\sum_{u \in U_i} d(a,u) + \left| |P - U_i| d(a,s_i) - \sum_{\bar{u} \in P - U_i} d(s_i, \bar{u}) \right|$$

$$\leq \sum_{u \in U_i} d(a,u) + \sum_{\bar{u} \in P - U_i} |d(a,s_i) - d(s_i,\bar{u})|$$

(by induction on the size of $P - U_i$)

$$\leq \sum_{u \in U_i} d(a,u) + \sum_{\bar{u} \in P - U_i} \max_{u \in U_i} |d(a,u) - d(u,\bar{u})| \quad \text{(since } s_i \in U_i\text{)}$$

$$\leq \sum_{u \in U_t} d(a,u) + \sum_{\bar{u} \in P - U_t} \max_{u \in U_t} |d(a,u) - d(u,\bar{u})| \quad (U_i \subseteq U_t\text{; by (5)})$$

$$\leq f(a) \quad \text{(by (5))}$$

and this chain of inequalities implies (4).

FMSA versions based on (1) and (3) are described in [4] and [2], respectively. That based on (3) is significantly faster than the direct method, both in terms of distance computations and overhead. On the other hand, (1) leads to a FMSA version which is even faster in terms of distances computations, though its space complexity is $\Theta(n^2)$ and its overhead ranges from $\Omega(n^2)$ to $O(n^3)$. Here we will focus on a new version based on (2), which will be referred to as FMSA-g_2.

Although (2) can be incrementally computed from one iteration to the following, direct calculation of this function is too expensive since it entails $O(n^2)$ steps. Fortunately, this overhead can be reduced as follows. Assume that $\bar{U}_t = \{\bar{u}_1, \bar{u}_2, \ldots, \bar{u}_{|\bar{U}_t|}\}$ is an arrangement of the set of unused patterns in ascending order of their distances to the selected pattern s_t in iteration t. Then, the bound of each alive pattern $\bar{u}_i$ in this iteration can be updated by computing

$$\sum_{u \in U_t} d(\bar{u}_i, u) + \sum_{j=1}^{|\bar{U}_t|} |d(\bar{u}_i, s_t) - d(s_t, \bar{u}_j)|$$

$$= \sum_{u \in U_t} d(\bar{u}_i, u) + \sum_{j=1}^{i} (d(\bar{u}_i, s_t) - d(s_t, \bar{u}_j)) + \sum_{j=i}^{|\bar{U}_t|} (d(\bar{u}_j, s_t) - d(s_t, \bar{u}_i))$$

$$= \sum_{u \in U_t} d(\bar{u}_i, u) + (2i - |\bar{U}_t|) d(\bar{u}_i, s_t) + \sum_{j=1}^{|\bar{U}_t|} d(\bar{u}_j, s_t) - 2 \sum_{j=1}^{i} d(\bar{u}_j, s_t)$$

As partial sums $\sum_{j=1}^{i} d(\bar{u}_i, s_t)$, $1 \leq i \leq |\bar{U}_t|$, can be incrementally precomputed in $O(n)$, updating of a bound only requires $O(1)$ steps. Complete computation of bounds, however, requires $O(n \log n)$ steps since sorting unused patterns becomes the dominating cost.

It is easy to show that the FMSA-g_2 can not stop after calculation of a single sum, but it can end after two iterations when, for instance, patterns are numbers compared through the usual distance, and the first pattern selected for sum calculation is either the minimum or the maximum of these numbers. Therefore, it is worth noting that the algorithm can find a median by computing as little as $2n - 3$ distances in $O(n \log n)$ steps. On the other hand, a worst case arises when the discrete metric is used ($d(p, q) = 1$ if $p = q$; 0 otherwise). In this case, (2) reduces to sums on used patterns and elimination has no effect. Thus the process ends computing as many distances as the direct method ($\frac{1}{2}n(n-1)$) and the overhead becomes $O(n^2 \log n)$. On the average, as (2) is tighter than (3) but not as tighter as (1), the FMSA-g_2 is expected to compute a number of distances between those of the previous FMSA versions.

3 Experiments

The FMSA-g_2 was tested on sets of n artificial patterns ($n = 16, 32, 64, \ldots, 1024$) randomly drawn from a uniform distribution in the unit d-dimensional hypercube ($d = 2, 4, 6, 8, 10$), and compared through the Euclidean distance. Although the algorithm is not intended to deal with patterns in a Euclidean space, following a time-honored tradition, we chose such kind of patterns since they are easy to generate and results can be compared with those of many other related problems and techniques. On the other hand, the FMSA-g_2 never took advantage of the point coordinates, but only the distances were used.

For each value of n and d, 100 sets of patterns were independently generated and the FMSA-g_2 was applied to search for a median of each of them. Figure 2 shows the mean number of distances computed the FMSA-g_2, given as a percentage of the number of distances computed by the direct method and accompanied with its corresponding 95% confidence interval.

Although for large dimensions only modest savings are obtained, these savings tend to increase as n increases and they are substantial for low dimensions. As expected, the number of distances computed by

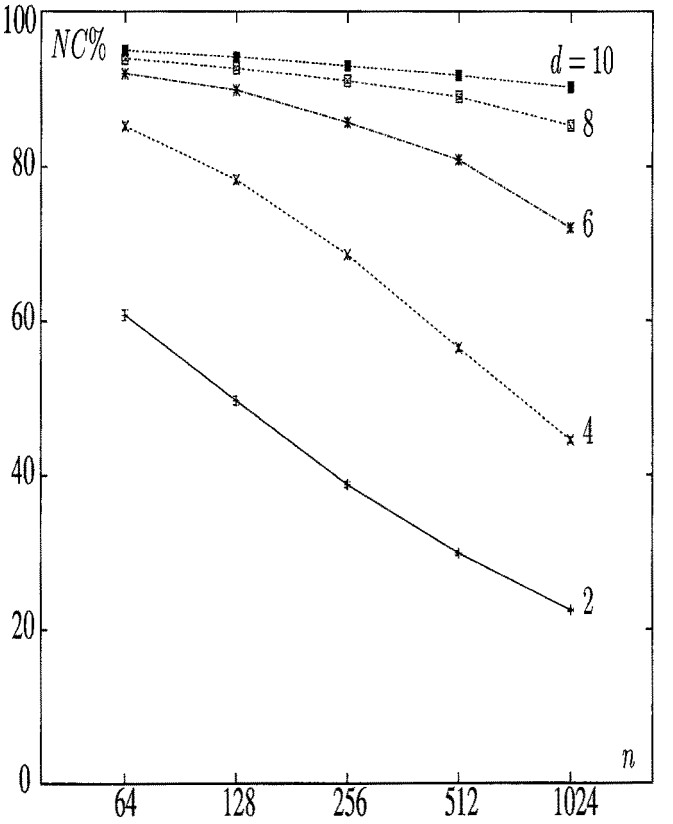

Fig. 2. Mean number of distances computed by the FMSA-g_2, given as a percentage of the number of distances computed by the direct method ($NC\%$), for several number of patterns (n) and dimensions (d).

the FMSA-g_2 is lower than that of the FMSA-g_3 [2], and greater than that associated with the FMSA-g_1 [4]. Although in terms of number of computed distances the FMSA-g_1 is in fact better than the here proposed FMSA-g_2, the overhead of the FMSA-g_1 ranges from $\Omega(n^2)$ to $O(n^3)$, which renders this algorithm useless for very large data sets.

4 Conclusions and Future Work

A generalized Fast Median Search Algorithm (FMSA) has been presented which includes two previously proposed algorithms for fast me-

dian search. A new particular case of this generalized FMSA has been proposed and tested through computer simulations.

The FMSA can be used to reduce the computational cost of a *k-means-like* clustering procedure which starts from a given initial partition, and then loops searching for a median of each cluster and reclassifying patterns according to their nearest medians. In fact, an optimized version of this procedure has been already developed by direct introduction of the simplest FMSA particular case, the FMSA-g3 [2]. This optimized version was tested on a set of speech data comprising ten repetitions of the Spanish digit vocabulary uttered by five different speakers. A computationally expensive (*Dynamic Time Warping*) distance function was used to compare utterances. For small values of the number of clusters ($k \leq 10$), it was observed that the optimized version runs from 3 to 6 times faster than the direct procedure [2]. While better results in terms of distance computations can be easily obtained by using the FMSA-g2 (or FMSA-g1) instead of the FMSA-g3, we think that even better results are possible by taking advantage of the fact that clusters (and medians) tend to become "stable" after some iterations of the procedure [3]. To this end, we are presently developing appropriate versions of the FMSA-g1, FMSA-g2 and FMSA-g3.

References

1. F. Casacuberta and M. D. de Antonio. A greedy algorithm for computing approximate Median Strings. In A. Sanfeliu, J. J. Villanueva, and J. Vitrià, editors, *VII National Symposium on Pattern Recognition and Image Analysis*, volume 1, pages 193–198, Bellaterra, Barcelona (Spain), 1997.
2. A. Juan and E. Vidal. An optimized K-means-like algorithm. In *4th Portuguese Conference on Pattern Recognition*, pages 11–16, Coimbra, Portugal, 1992.
3. A. Juan and E. Vidal. Fast k-Means-like Clustering in Metric Spaces. *Pattern Recognition Letters*, 15(1):19–25, 1994.
4. A. Juan and E. Vidal. An Algorithm For Fast Median Search. In A. Sanfeliu, J. J. Villanueva, and J. Vitrià, editors, *VII National Symposium on Pattern Recognition and Image Analysis*, volume 1, pages 187–192, Bellaterra, Barcelona (Spain), 1997.
5. T. Kohonen. Median Strings. *Pattern Recognition Letters*, 3:309–313, 1985.
6. P. B. Mirchandani and R. L. Francis, editors. *Discrete Location Theory*. Wiley, 1990.