
ELSEVIER

Pattern Recognition Letters 22 (2001) 1145–1151

Pattern Recognition Letters

www.elsevier.nl/locate/patrec

An approximate median search algorithm in non-metric spaces

Luisa Micó *, Jose Oncina

Dept. Lenguajes y Sistemas Informáticos, Universidad de Alicante, Apdo. 99, E-03071 Alicante, Spain

Received 21 July 2000; received in revised form 28 February 2001

Abstract

Given a set of data points and a distance function, the median point is defined as the point (in the set) that minimizes the sum of the distances to the remaining points of the set. In the general case, the median computation has an $O(n^2)$ time cost, where n is the number of points. Nevertheless, for most tasks an approximate median is enough. In this paper a very fast algorithm (linear in time) is presented that finds a point that is a very good approximation of the exact median. This algorithm is independent of the distance function and does not degrade as the dimensionality of the data increases. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Median; Non-metric spaces; Pattern recognition

1. Introduction

Given a set of n points P and a distance function, the *median* is defined as the point (in the set) that minimizes the sum of the distances to the remaining points of the set. If points are real numbers and the distance function is the absolute value of their difference, then this definition of median reduces to the scalar median that is the one normally used in statistics (if the numbers are sorted the median is the number that appears in the middle of the list). There are several works in the literature treating this special case, as the classical Hoare (1961) "find" algorithm that has an $O(n)$ time complexity. The scalar median is widely used in image processing as an edge-preserving impulsive-noise reduction filter (Alvarez et al., 1992). The basic idea is that pixel values are replaced by the median of the pixels contained in a window around it. Then the median computation is a bottleneck for large images.

An extension of the previous case is the *vector median* in which the points are real (or integer) vectors and the distances are (usually) the L_2 (Euclidean) and the L_1 (absolute value) distances. The vector median is widely used in multiband image processing tasks (Astola et al., 1990) as color images or vector fields obtained by optic flow computation (Bartolini et al., 1993). There are no algorithms to solve this problem in lower than an $O(n^2)$ time but there are some approximate algorithms such as (Barni et al., 1995) whose emphasis is in avoiding the calculation of the square root of the Euclidean distance (then, the complexity remains quadratic but they report a 50% time reduction) and Barni et al.'s (1992), based on the idea that the calculation of the vector median using L_1 is equivalent to calculating the scalar

* Corresponding author. Tel.: +34-96-590-3400; fax: +34-96-590-3464.

E-mail addresses: mico@dlsi.ua.es (L. Micó), oncina@dlsi.ua.es (J. Oncina).

PII: S0167-8655(01)00062-9

median componentwise. As this vector does not necessarily belong to the original set its nearest neighbor is proposed as the approximate vector median. This algorithm works in linear time and is clearly faster than our algorithm, but only works with the L_1 distance.

In the general case, the points can represent data structures such as strings and the distance function can be any dissimilarity measure such as the edit distance (or weighted Levenshtein distance) (Fu, 1982). Just in case the distance function is a metric: $d(x,y) = 0$ iff $x = y$, $d(x,y) = d(y,x)$ (symmetric), $d(x,y) < d(x,z) + d(z,y)$ (triangle inequality), there are some algorithms based on avoiding distance computations (Juan and Vidal, 1998; Juan, 1999) but have quadratic time and space requirements. If the triangular inequality does not hold but the distance is symmetric, the only fast algorithm consists in avoiding to calculate two times the same distance ($d(x,y)$ and $d(y,x)$). No approximate algorithm has seen reported so far.

This general-case median is a well-known technique for modeling a given set and appears as a subtask in some tasks such as data clustering (Fu, 1982). Moreover, most recent works in computer vision compare images using measures of similarity that are complex and non-metric, since they do not obey the triangle inequality (Jacobs et al., 2000). This can occur because the triangle inequality is difficult to enforce in complex matching algorithms that are statically robust. Also, when matching probability distributions, the Kullback–Leibler measure of cross entropy is used, which is asymmetric and does not obey the triangle inequality.

Therefore, it is interesting to develop fast techniques to find the median of a set of points in nonmetric spaces. Moreover, often it is either enough to have an approximate median, or the median is needed in problems that are solved with approximate methods and, therefore calculating an exact median instead of an approximate one may not be worth the effort.

A related definition is the *generalized median* that arises when the search is not constrained to the point set, but extended to the whole space where the points are extracted. Finding the generalized median is a more difficult task that requires complex and specific algorithms (Casacuberta and de Antonio, 1997).

In this paper an approximate general search algorithm for nonmetric spaces is presented. This algorithm does not make any assumption about the structure of the points or about the distance function and has a linear time complexity. An easy modification is also presented to avoid duplicate distance computation when the distance is symmetric.

The algorithm can also be used for the vector median computation. As discussed only in the case that the distance is L_1, there is a faster algorithm, Barni et al. (1992).

2. The approximate median search algorithm

The idea of the algorithm is simple: instead of computing the sum of each point to all the other points to select the point that minimizes this sum, only a subset of all the points is used to obtain an estimation of this sum. Obviously, the larger this subset is, the lower the error interval of the sum estimation will be. The algorithm first calculates such estimation and then calculates the exact distance sum for the points that have a lower distance sum estimation.

The algorithm works in two main steps. In the first step a random subset of n_r points is selected (*reference points*), and, for each point, the sum of distances from the point to the reference points is calculated and stored (*partial sum*). In the second step, the n_t points (*test points*) whose partial sum is lowest are selected and, for each of those points, the sum of distances from the point to the remaining points is calculated and stored (*full sum*). The point that minimizes the full sum is selected as the approximate median. Obviously, if the distance is symmetric, as the full sum is also known for the reference points, these points are also taken into account at the time of selecting the best approximate median. The algorithm, with some tricks to avoid the double distance computation when the distance is symmetric, can be seen in Fig. 1.

It is easy to see that the time complexity of the algorithm is $O(n_u|P|)$, where n_u is the number of

```
algorithm approximate median
input:
  P : set of points
  d(·,·) : distance function
  n_r : number of reference points
  n_t : number of test points
output:
  m ∈ 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 = ∅
∀p ∈ P
  PS[p] = 0
// Selecting the reference points
repeat n_r times
  u = random point in P − U
  U = U ∪ {u}
  FS[u] = PS[u]
  ∀p ∈ 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[·]
// Calculating the full sums
∀t ∈ T
  FS[t] = PS[t]
  U = U ∪ {t}
  ∀p ∈ 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[·]
end algorithm
```

Fig. 1. The approximate median algorithm.

used points ($n_u = n_r + n_t$) and $|P|$ the number of data points, instead of $O(|P|^2)$ for the exact algorithm. However, the space complexity is $O(|P|)$ instead of $O(1)$.

The algorithm has two parameters, the number of reference points (n_r) and the number of test points (n_t). In the experiment section it is shown that the choice $n_r = n_t$ seems reasonable and that with a small number of used points very good median candidates can be obtained.

One might think that the test points could be selected incrementally. Then, as each time that a test point is selected the partial sum of the remaining points is updated, this information can be used (at no cost) to select the next test point. One expects that the more information is used in order to select the test points, the more accurate medians will be obtained. The only change needed in the algorithm is to change the code in sections "Selecting the test points" and "Calculating the full sums" for the code shown in Fig. 2.

Anyway, this is not a good idea (as it is shown in Section 3) because the mission of the reference points is to represent the whole set of points and is expected that the reference points have a similar behavior, as a predictor of the median, as that of the whole set. If more points, not randomly selected, are used they are going to produce a bias and worse results will result.

3. Experiments

In order to show the behavior of the proposed algorithm a set of experiments was prepared with both synthetic and real data. The synthetic data were extracted from a uniform distribution in the unit cube for different dimensionalities and the Euclidean distance was used. In these experiments the algorithm for symmetric distances was used. The real data were a chain code (Fu, 1982; Thedoridis and Koutroumbas, 1999) description of a fraction of the handwritten digits (10 writers) of

```
      ⋮
// Selecting test points and calculating their full sums
repeat
  t = the point in P − U that minimizes PS[·]
  FS[t] = PS[t]
  U = U ∪ {t}
  ∀p ∈ P − U
    d = d(t, p)
    FS[t] = FS[t] + d
    PS[p] = PS[p] + d
until |U| == n_r + n_t
      ⋮
```

Fig. 2. A section of the bad algorithm.

the NIST Special Database 3 (National Institute of Standards and Technology). The edit distance (Fu, 1982) with deletion, insertion and substitution errors equal to 1, 1.5 and 2, respectively, was used in order to make the distance non-symmetric and therefore non-metric.

To measure the adequacy of the approximate median, the normalized difference between the sum of distances from the exact median and the sum of distances from the approximate median was used. That is, if P is the set of data points, $d(\cdot,\cdot)$ is the distance function, m_e is an exact median (the median has not to be unique) and m_a is the approximate median, the *sum error* e of the approximate median is measured as

$$e = \frac{\sum_{p\in P} d(m_a, p) - \sum_{p\in P} d(m_e, p)}{\sum_{p\in P} d(m_e, p)}.$$

As the exact median minimizes the sum of distances then the sum error is always positive and is zero if and only if the approximate median is exact.

The first set of experiments was designed to study the behavior of the sum error as the size of the used points increases. In these experiments the number of reference points was set equal to the number of test points. Each of the experiments was repeated 10 times using a different random choice of reference points.

In the experiments with synthetic data, ten data sets of 1000 points were used. The experiment was repeated for spaces of dimensionality 6, 12 and 18. The number of used points was increased from 5 to 60 in steps of 5. Fig. 3 shows the results. From 60 to 100 used prototypes (not shown in the plot) all the sum errors were zero (an exact median was found). It can be shown that using 20 points, sum errors lower that 1% can be reached and time savings of 98% are obtained.

It is surprising to observe that the algorithm works better (lower sum error for the same number of used points) as the dimensionality increases. This effect is probably due to the fact that as the dimension increases the sum of distances from a point to the rest becomes similar for all the points (Bayer et al., 1999), and then it is easier to find a point similar to the optimal.

In the experiments with handwritten digits a similar approach was used. In this case all handwritten digits from 10 writers were used (ranging from 901 samples for the digit 5 to 1203 for the digit 1). The number of used points was increased from 5 to 95 in steps of 5. Fig. 3 shows the result for digits 0, 1, 2 and the average for all digits. In this case, for 95 used points sum errors of 0.03% are obtained and with 20 used points the average sum error is lower than 1.7%.

In order to study, for a given number of used points, which is the best proportion of reference points and test points a second set of experiments was made. In these experiments three different numbers of used points were chosen (25, 50 and 100) and the number of reference points increased from 1 to the number of used points. Each experiment was repeated 10 times using different

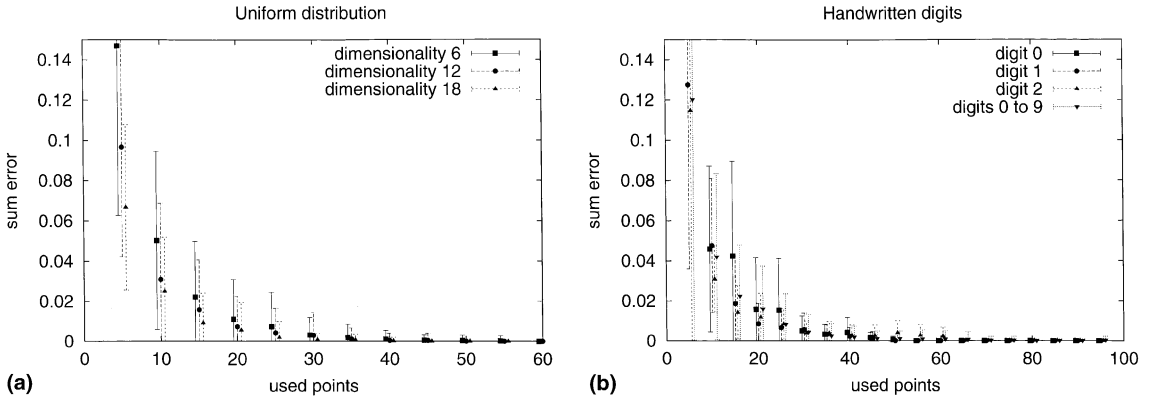

Fig. 3. Sum error as the number of used points increases.

random reference points. In the experiments with synthetic data each experiment was repeated with ten data sets of 1000 points in a 6-dimensional space. The experiments with handwritten digits show the average over all the digits.

As it can be observed in Fig. 4 the plots form a very wide valley. Then the relation between the number of reference points and the number of test points is not too critical and making both parameters equal seems a reasonable option.

Two sets of experiments were made in order to study the dependence of the sum error, for a fixed number of used points, on the number of data points. In these experiments, two different numbers of used points were chosen (25 and 50) and $n_r = n_t$ was set. The number of points increased from 50 to 900 in steps of 50 and each experiment was repeated 10 times using different random reference points. The experiments with synthetic data were repeated for 10 random data sets of points in a 6-dimensional space. The experiments with handwritten digits were repeated 10 times for different random subset of the total number of data points. The plot shows the average for all the digits.

As can be seen in Fig. 5 the sum error grows very slowly with the data point set size. This growth is slower as the number of used points increases.

This last experiment was repeated in order to compare the expended time of the "brute force" algorithm and our algorithms (Fig. 6). This time

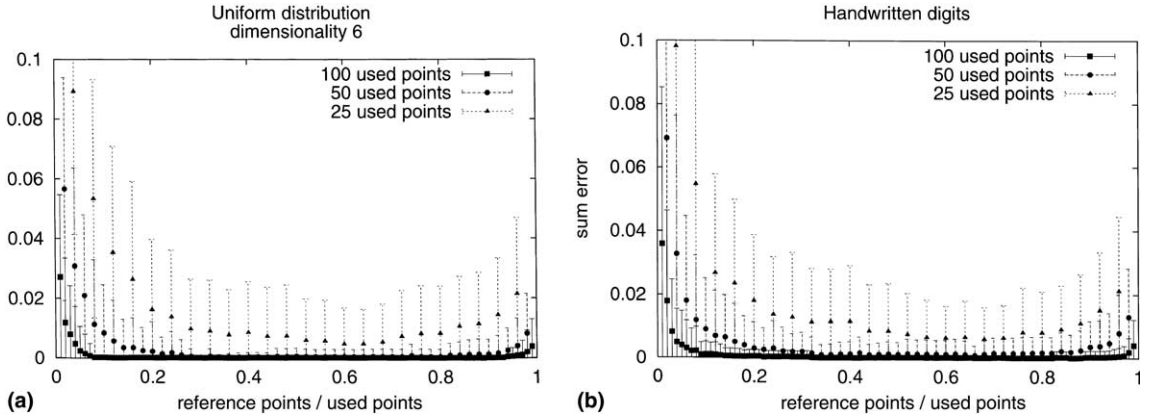

Fig. 4. Sum error for 100, 50 and 25 used points, when the proportion of reference points change from 0 to 1.

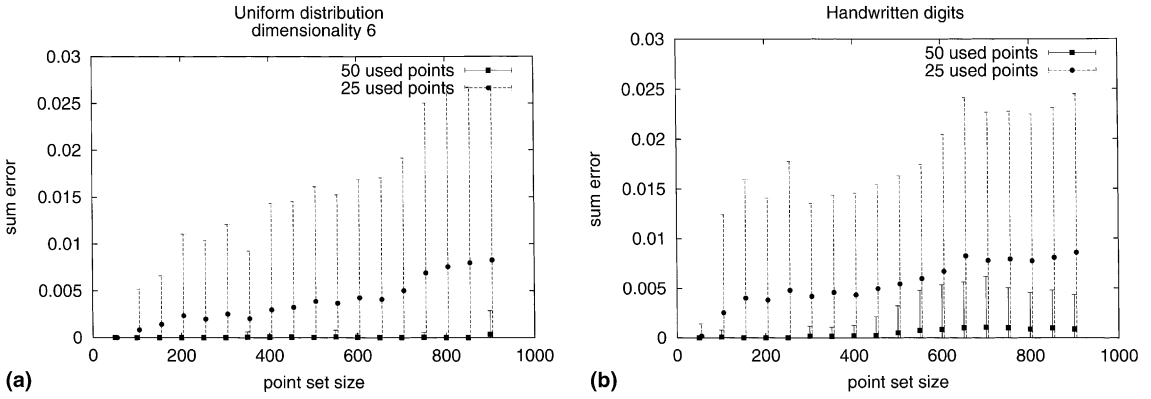

Fig. 5. Sum error when the point set size increases, for a fixed size of used points.

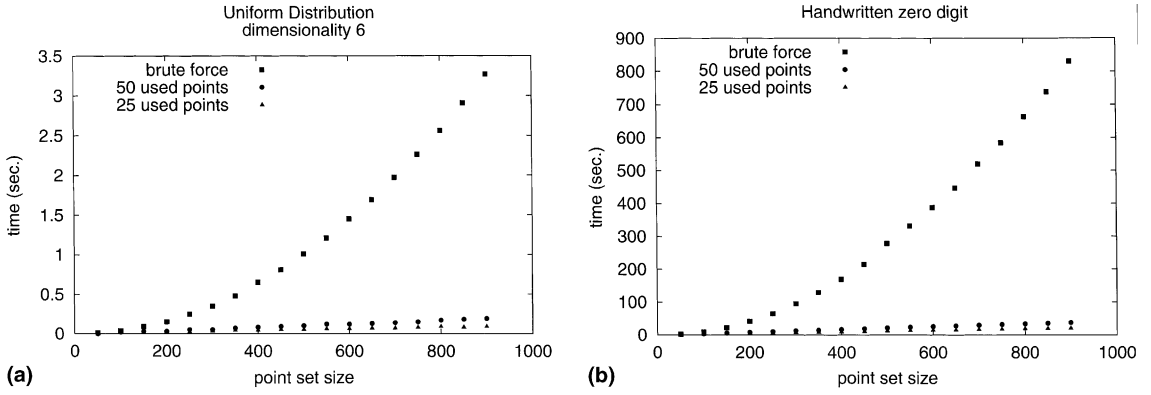

Fig. 6. Comparison of the expended time with the brute force algorithm.

no repetitions were made because the time expended does not depend of the values of the distances. The handwritten digit experiment was made only with the data for the "0" digit. The times where measured in a Pentium MMX running at 200 MHz under a Linux system. As was expected the algorithm proves to be linear and faster than the "brute force" even for small point set sizes.

In order to compare the good and bad algorithms mentioned in Section 2, Fig. 7 shows the result of experiments (synthetic data in a 6-dimensional space) made in the same way as that of the first set of experiments of this section. In both cases it can be observed that the incremental use of test points as reference points slightly increases the sum error.

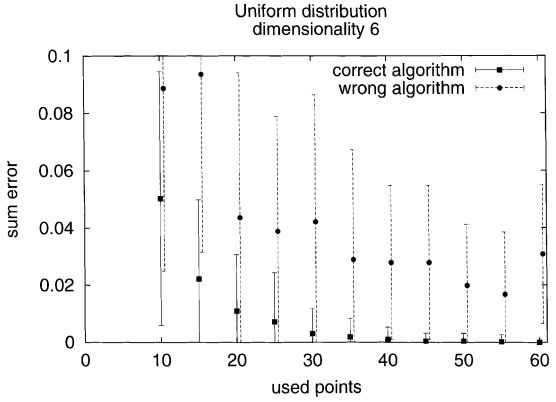

Fig. 7. Sum error of the wrong and correct algorithms as the number of used points increases.

4. Conclusions

In this work an effective fast approximate median computation algorithm is presented. The algorithm makes no assumptions on the used distance and therefore it can be used on non-metric spaces. An improvement of the algorithm for symmetric distances is also presented.

The algorithm has two parameters, the number of reference points (n_r) and the number of test points (n_t). The complexity of the algorithm is $O(n_u|P|)$ where n_u (used points) is the sum of both parameters and P is the set of points. The experiments (with synthetic and real data) show that very approximate medians can be obtained with a small number of used points and that the error depends very weakly on the number of the used points. This makes the algorithm linear in practical situations.

The experiments also show that setting both parameters to equal values is a reasonable choice and that its behavior does not degrade as the dimensionality of the data increases.

Acknowledgements

The authors thank Mikel L. Forcada and José M. Iñesta for their help in writing this paper. The authors thank the Spanish CICyT for partial support of this work through project CICYT TIC97-0941. The authors thank the Spanish CICyT for partial support of this work through project TIC2000-1703-CO3-02.

References

Alvarez, L., Lions, P., Morel, J., 1992. Image selective smoothing and edge detection by nonlinear diffusion. SIAM J. Numer. Anal. 29, 845–866.

Astola, J., Haavisto, P., Neuvo, Y., 1990. Vector median filters. In: Proc. IEEE, Vol. 78.

Barni, M., Bartolini, F., Buti, F., Cappellini, V., October 1995. Optimum linear approximation of the Euclidean norm to speed up vector median filtering. In: Proc. ICIP'95 Conf., Washington, DC, 1995.

Barni, M., Capellini, V., Mecocci, A., 1992. The use of different metrics in vector median filtering: application to fine arts and paintings. In: Proc. EUSIPO '92.

Bartolini, F., Cappellini, V., Colombo, C., Mecocci, A., June 1993. Enhancement of local optic flow techniques. In: Proc. 4th Internat. Workshop on Time Varying Image Processing and Moving Object Recognition, Florency, Italy.

Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U., January 1999. When is nearest neighbor meaningful? In: Proc. 7th Internat. Conf. on Database Theory (ICDT).

Casacuberta, F., de Antonio, M., 1997. A greedy algorithm for computing approximate median strings. In: VII National Symposium on Pattern Recognition and Image Analysis, Vol. 1, Bellaterra, Barcelona, Spain.

Fu, K.S., 1982. Syntactic Pattern Recognition and Applications. Prentice-Hall, Englewood Cliffs, NJ.

Hoare, C., 1961. Find (algorithm 65). Commun. ACM 4 (7), 321–322.

Jacobs, D.W., Weinshall, D., Gdalyahu, Y., 2000. Classification with nonmetric distances: Image retrieval and class representation. IEEE Trans. Pattern Anal. Machine Intell. 22 (6), 583–600.

Juan, A., 1999. Optimización de presentaciones en técnicas de aprendizaje no supervisado y su aplicación al reconocimiento de formas. Ph.D. Thesis, Universidad Politécnica de Valencia, Valencia, Spain.

Juan, A., Vidal, E., 1998. Fast median search in metric spaces, In: Advances in Pattern Recognition, Lecture Notes in Computer Science, Vol. 1451. Springer, Berlin.

Thedoridis, S., Koutroumbas, K., 1999. Pattern Recognition. Academic Press, New York.