Comparison of Four Initialization Techniques for the *K*-Medians Clustering Algorithm

A. Juan and E. Vidal

Institut Tecnològic d'Informàtica,
Universitat Politècnica de València, 46071 València, Spain
ajuan@iti.upv.es evidal@iti.upv.es

Abstract. Clustering in Metric Spaces can be conveniently performed by the so called k-medians method. It consists of a variant of the popular k-means algorithm in which cluster medians (most centered cluster points) are used instead of the conventional cluster means. Two main aspects of the k-medians algorithm deserve special attention: computing efficiency and initialization. Efficiency issues have been studied in previous works. Here we focus on initialization. Four techniques are studied: Random selection, Supervised selection, the Greedy-Interchange algorithm and the Maxmin algorithm. The capabilities of these techniques are assessed through experiments in two typical applications of Clustering; namely, Exploratory Data Analysis and Unsupervised Prototype Selection. Results clearly show the importance of a good initialization of the k-medians algorithm in all the cases. Random initialization too often leads to bad final partitions, while best results are generally obtained using Supervised selection. The Greedy-Interchange and the Maxmin algorithms generally lead to partitions of high quality, without the manual effort of Supervised selection. From these algorithms, the latter is generally preferred because of its better computational behaviour.

Key words: Clustering, Metric Spaces, K-Medians algorithm, K-Medians initialization, Greedy-Interchange algorithm, Maxmin algorithm

1 Introduction

One of the most popular clustering techniques is the so-called *k-means*, *c-means* or *basic ISODATA* algorithm [1,3]. It aims to partition the data into k clusters so that the sum of squared Euclidean distances between samples and their corresponding cluster means is minimized. Given k initial estimates of cluster means, it alternates two basic steps under an iterative scheme. These steps are the classification of samples in accordance with their nearest cluster means, and the computation of new cluster means. Each new partition decreases the sum of squared distances between samples and their corresponding cluster means. Although the k-means algorithm is suboptimal, it generally achieves good approximate solutions at the expense of a moderate computational cost.

F.J. Ferri et al. (Eds.): SSPR&SPR 2000, LNCS 1876, pp. 842–852, 2000.

It is easy to modify the k-means algorithm for the case where data cannot be adequately represented in a suitable vector space, though a metric is available to measure the dissimilarity between data points. To do this, each cluster mean is approximated by a "most centered sample" or *median*; that is, a sample whose sum of distances to all cluster samples is minimum. We call this strictly distance-based clustering technique *k-medians* or *k-centroids* [5].

Practically speaking, there are two major aspects of the k-medians algorithm which deserve special attention: its *computing cost* and its *initialization*. Computational aspects have been carefully studied in previous works [5,6]. In [5] we proposed a fast version of the k-medians algorithm which basically consists of introducing a *fast Nearest-Neighbour* search technique for efficiently computing the closest median of a sample. On the other hand, in [6] we proposed a *fast median search* technique which can be used to reduce the complexity associated with the computation of cluster medians.

Initialization issues are studied in this paper. We compare four initialization techniques for the k-medians algorithm: *random* selection, *supervised* selection, *greedy-interchange* algorithm and *maxmin* algorithm. Random selection simply consists of picking k cluster seeds at random. This is an efficient standard initialization technique, but it often selects seeds that are close together and thus low quality partitions. Supervised selection assumes that a small subset of samples can be labeled in accordance with a tentative classification scheme. If such assumption is reasonable, seeds can be selected class by class to ensure better dispersion than in the case of random selection. This is the same purpose of the greedy-interchange and maxmin algorithms, though manual effort is replaced by computing cost in this case. The former consists of two time consuming heuristics for the k-medians clustering problem which are applied consecutively. The latter is a slightly modified version of an efficient initialization technique for the k-means algorithm [8].

The greedy-interchange and maxmin algorithms are described in sections 2 and 3, respectively. Experiments are aimed at assessing the capabilities of the different initialization techniques in two typical applications of clustering; namely, exploratory data analysis and unsupervised prototype selection. Results with synthetic data (Gaussian mixtures) as well as real data (human banded chromosomes) are reported in section 4. Conclusions are summarized in section 5.

2 The Greedy-Interchange Algorithm

K-medians clustering can be properly stated as a combinatorial optimization problem. Given a metric space $(\mathbb{E},\ d)$, a finite set of data points or "prototypes" $P \subset \mathbb{E}$ and a positive integer k, we seek a subset of k cluster "representatives", $Q \subset P$, for which the following criterion is minimized:

$$z(Q) = \sum_{p \in P} \min_{q \in Q} d(p, q) \tag{1}$$

Note that there are $k^{(n-k)}$ different ways to assign $n-k$ non-representative prototypes to the same k representatives but, among them, we are only interested in the partition which results from assigning each prototype to its nearest representative. Criterion (1) measures the sum of distances associated with such minimum distance partition.

The k-medians clustering problem, well-known as a *prototype location problem* [10], is NP-Hard [7]. Despite of this negative result, there are quite a few heuristics other than the k-medians algorithm that provide good approximate solutions [10]. Most of these heuristics, however, cannot be used for clustering large data sets because of their high complexity. Two of the fastest heuristics are the *greedy* and *interchange* algorithms described below.

2.1 The Greedy Algorithm

As its name indicates, this algorithm follows a greedy strategy to draw k representatives from the prototypes. In iteration t, the set of $t-1$ previously selected representatives is enlarged with a prototype that leads to a maximum decrease of (1). That is, the set of representatives selected in iteration t, $0 \leq t \leq k$, is:

$$Q^t = \begin{cases} \emptyset & \text{if } t = 0, \\ Q^{t-1} \cup \{q_t\} & \text{if } t > 0 \end{cases} \qquad \text{where} \qquad q_t = \underset{p \in P - Q^{t-1}}{\arg\min}\ z(Q^{t-1} \cup \{p\})$$

This algorithm computes the criterion function $(n-k)k$ times approximately, and hence a direct implementation makes $(n-k)^2k^2$ distance computations. Fortunately, this computing cost can be notably reduced by simply introducing an auxiliary array of distances between prototypes and their nearest representatives (see fig. 1). Its complexity is of $(n-k)^2k$ distance computations approximately.

Algorithm greedy$(P,\ d,\ k;\ Q \subset P)$
Variable: $D \in \mathbb{R}^P$ /* auxiliary array: $D_p = \min_{q \in Q} d(p,q)$ */
Method: $Q = \emptyset$; $D = (\infty)^P$
for $t = 1$ **to** k **do** /* compute $q = q_t$ and add it to Q */
  $minz = \infty$
  $\forall p \in P - Q$ **do**
    $z' = \sum_{p' \in P-Q} \min(D_{p'}, d(p', p))$ /* $z' = z(Q^{t-1} \cup \{p\})$ */
    if $z' < minz$ **then** $minz = z'$; $q = p$
  $Q = Q \cup \{q\}$; $\forall p \in P - Q$ **do** $D_p = \min(D_p, d(p,q))$

Fig. 1. Greedy algorithm.

2.2 The Interchange Algorithm

The interchange algorithm starts from a given set of representatives and tries to improve it iteratively. In each iteration, the algorithm searches for a pair *(representative, prototype)* whose interchange leads to a minimum increase of (1),

$$(q^-, q^+) = \underset{(q,\bar{q})\in Q\times P-Q}{\arg\min} z(Q - \{q\} \cup \{\bar{q}\})$$

If (1) decreases by interchanging q^- and q^+, then such interchange is carried out and the algorithm begins a new iteration; otherwise, the algorithm stops.

There are $(n-k)k$ possible interchanges in each iteration, and each interchange requires evaluation of (1). Therefore, a straightforward implementation of this algorithm computes $(n-k)^2k^2$ distances per iteration. As in the case of the greedy algorithm, it is possible to reduce this complexity by introducing a simple algorithmic refinement (see fig. 2).

Algorithm interchange(P, d, k, $Q^0 \subset P$; $Q \subset P$)
Variable: $D \in \mathbb{R}^P$
Method: $Q = Q^0$
repeat
  $interchange$ = false; $minz = \infty$
  $\forall q \in Q$ **do** /* explore possible interchanges with q */
    $\forall p \in P - Q \cup \{q\}$ **do** $D_p = \min_{q' \in Q-\{q\}} d(p, q')$
    $\forall p \in P - Q$ **do**
      $z' = \sum_{p' \in P-Q-\{p\}\cup\{q\}} \min(D_{p'}, d(p', p))$
      if $z' < minz$ **then** $minz = z'$; $q^- = q$; $q^+ = p$
  if $z(Q - \{q^-\} \cup \{q^+\}) < z(Q)$ **then** /* advantageous interchange */
    $interchange$ = true; $Q = Q - \{q^-\} \cup \{q^+\}$
until $\neg\, interchange$

Fig. 2. Interchange algorithm.

For each representative q, the refined method efficiently explores all possible interchanges with q by introducing an auxiliary array, D. For each non-representative prototype p, the minimum distance between p and all representatives except q is computed and stored in D. In this way, evaluation of the $n-k$ possible interchanges with q can be performed by computing $(n-k)^2$ distances. In total, the number of distance computations per iteration coincides with that of the greedy algorithm (i.e. $(n-k)^2k$).

2.3 Combination: The Greedy-Interchange Algorithm

We call greedy-interchange combination (algorithm) to the consecutive application of both algorithms. Despite of the computational improvements discussed

above, the complexity of this combination is too high for clustering of a large data set. Nevertheless, it is still possible to use this combination with a (random) subset of moderate size so as to obtain an initial solution for the k-medians algorithm. In this case, the greedy-interchange algorithm can be efficiently applied by preprocessing the matrix of pairwise distances between prototypes.

3 The Maxmin Algorithm

The maxmin algorithm is a slightly modified version of an efficient initialization technique for the k-means algorithm [8]. It can also be seen as a fast approximate greedy algorithm. As this technique does, the maxmin algorithm iteratively selects one representative at a time. In iteration t, the set of $t-1$ previously selected representatives is enlarged with the prototype whose distance to its closest representative is maximum. That is, the set of representatives selected in iteration t, $1 \leq t \leq k$, is:

$$Q^t = \begin{cases} arbitrary(P) & \text{if } t = 1, \\ Q^{t-1} \cup \{q_t\} & \text{if } t > 1 \end{cases} \qquad \text{where} \qquad q_t = \underset{p \in P - Q^{t-1}}{\arg\max} \min_{q \in Q^{t-1}} d(p,q)$$

A detailed description of this algorithm is shown in fig. 3. This description includes an auxiliary array which is used just as the auxiliary array included in the greedy algorithm (fig. 1). The number of distances computed by the maxmin algorithm is approximately $(n-k)k$. Note that this number is very small in comparison with that of the greedy algorithm.

Algorithm maxmin(P, d, k; $Q \subset P$)
Variable: $D \in \mathbb{R}^P$ /* auxiliary array: $D_p = \min_{q \in Q} d(p,q)$ */
Method: $Q = \emptyset$; $D = (\infty)^P$; $q = arbitrary(P)$
for $t = 1$ **to** k **do**
  $Q = Q \cup \{q\}$; $maxmin = 0$
  $\forall p \in P - Q$ **do**
    $dpq = d(p,q)$
    if $dpq < D_p$ **then** $D_p = dpq$
    if $D_p > maxmin$ **then** $q = p$; $maxmin = D_p$

Fig. 3. Maxmin algorithm.

4 Experiments

As in the case of the k-means algorithm, the k-medians algorithm can be used both for data exploration and to provide prototypes for use in supervised classifiers. Assume that prototypes are grouped into k^* compact, well-separated clusters of similar a priori probabilities, and that each cluster can be appropriately

modeled by a single representative. Then, application of the k-medians algorithm with $k = k^*$ should discover the natural groups and provide an optimal set of representatives. Although this is a rather unrealistic assumption in many cases, much effort can be saved in early stages of classifier design if such assumption can be assessed. If not, application of the k-medians algorithm is still advantageous when a large set of unlabeled prototypes is available but only a small fraction of the prototypes is to be used for (re)training a classifier. In this case, the unconditional distribution of the whole set can be adequately approximated with the set of representatives selected by the k-medians algorithm.

The experiments reported hereafter have been designed to study the performance of the k-medians algorithm as: a) an exploratory data analysis technique; and b) a procedure for unsupervised selection of prototypes. As we will see, this performance depends very much on the initialization technique.

4.1 Exploratory Data Analysis

To study the k-medians algorithm as an exploratory data analysis technique, a simple classification problem has been chosen. It involves 10 equally-probable classes of 8-dimensional normal densities, with well-separated means, and common covariance matrix $\Sigma = 0.0307\, I$. Two sets of $10,000$ independent prototypes each were drawn from this mixture for training and test purposes.

The Bayes classifier for this problem is linear and can be implemented as a minimum Euclidean distance classifier by using the true class means as prototypes. Its empirical error rate is 2.1%. This error rate is also achieved when true class means are replaced by their empirical estimates, but it reaches 2.8% when means are approximated by class medians. If these class medians are used to initialize the k-medians algorithm, then a set of representatives is obtained whose associated minimum distance classification error rate is 3.0%. On the other hand, the nearest-neighbour rule based on the whole training set missclassifies 4.0% of the test prototypes.

Taking into account these figures, the application of the k-medians algorithm should be considered "successful" if, with $k = 10$, it provides a "quasi-optimal" set of representatives; that is, one whose associated minimum distance classification error rate is 3.0%. As it will be seen, success is closely related with good initialization: if the algorithm is not appropriately initialized, then it will fail to pick one representative of each class and hence the error rate is expected to increase dramatically. This is certainly true if values of k smaller than 10 are tried. On the other hand, this undesirable behaviour is expected to be reduced by using larger values of k.

The k-medians algorithm was executed 50 times for each one of the four initialization techniques previously discussed and each $k \in \{5, 6, \ldots, 30\}$. Although we always used the same training set of $10,000$ training prototypes in these executions, a different initial set of representatives was obtained each time: supervised selection was based on randomly selected class seeds; the greedy-interchange algorithm was tested on randomly chosen subsets of 100 prototypes; and the maxmin algorithm was started from a prototype also chosen at random. We

never took advantage of the prototypes coordinates; only Euclidean distances were used. Results are shown in figure 4. This figure encompasses eight panels distributed in four rows and two columns. Each row is associated with a different initialization technique, while the left and right columns correspond to initial and k-medians-optimized sets of representatives, respectively. Each plotted point in these panels represents the error rate (E) of the nearest neighbour classifier based on a different set of representatives. An average error rate curve is included in each panel to help with interpretation of results.

As expected, the performance of the k-medians algorithm depends very much on the initialization technique used. The better the initial solutions are, the more quasi-optimal solutions are found. This tendency is quite clear when comparing random selection with the other initialization techniques. Random selection almost always fails to provide good initial sets of representatives and, in consequence, it often leads to clearly suboptimal solutions ($E \geq 10\%$). On the contrary, the initial solutions provided by the other techniques hardly ever lead to such suboptimal results. A finer analysis reveals, however, that minor differences in quality do exist among these techniques. Although the best results correspond to supervised selection, they are very similar to those obtained by the greedy-interchange combination. This excellent outcome confirms that spending manual effort does not pay off when techniques are available that can do the same job at the expense of computing cost. In fact, even better results were observed for the greedy-interchange algorithm in further experiments (not reported here for brevity) testing this algorithm on random subsets of more than 100 prototypes. Unfortunately, its high complexity renders this technique useless for clustering large sets. In such case, the maxmin algorithm is perhaps the best choice. Despite the fact that this technique generates slightly worse results than those provided by the greedy-interchange combination, its complexity matches well with that of the k-medians algorithm. On the other hand, suboptimal solutions can be eventually circumvented by testing several sets of representatives or using values of k larger than the number of natural groups. As expected, it is observed that chances of being trapped in a suboptimal solution rapidly decrease as larger values of k are tried. However, this alternative should be used with caution since misleading results can be derived when a natural group is "dissected" which would otherwise be appropriately modeled by a single prototype. Moreover, an undesirable side effect has been also found in the experiment: the average error rate increases with k and quickly approximates that of the nearest-neighbour classifier based on the whole training set (4.0%).

Apart from using the error rate associated with a set of representatives as a measure of its quality, we also used the average Euclidean distance between the prototypes and their closest representatives; that is, a normalized version of the k-medians clustering criterion (1). As in the case of the error rate, this parameter was estimated from the $10,000$ test prototypes instead of the training prototypes to assure better statistical independence. Results, omitted here for the sake of brevity, show basically the same tendencies as those of fig. 4.

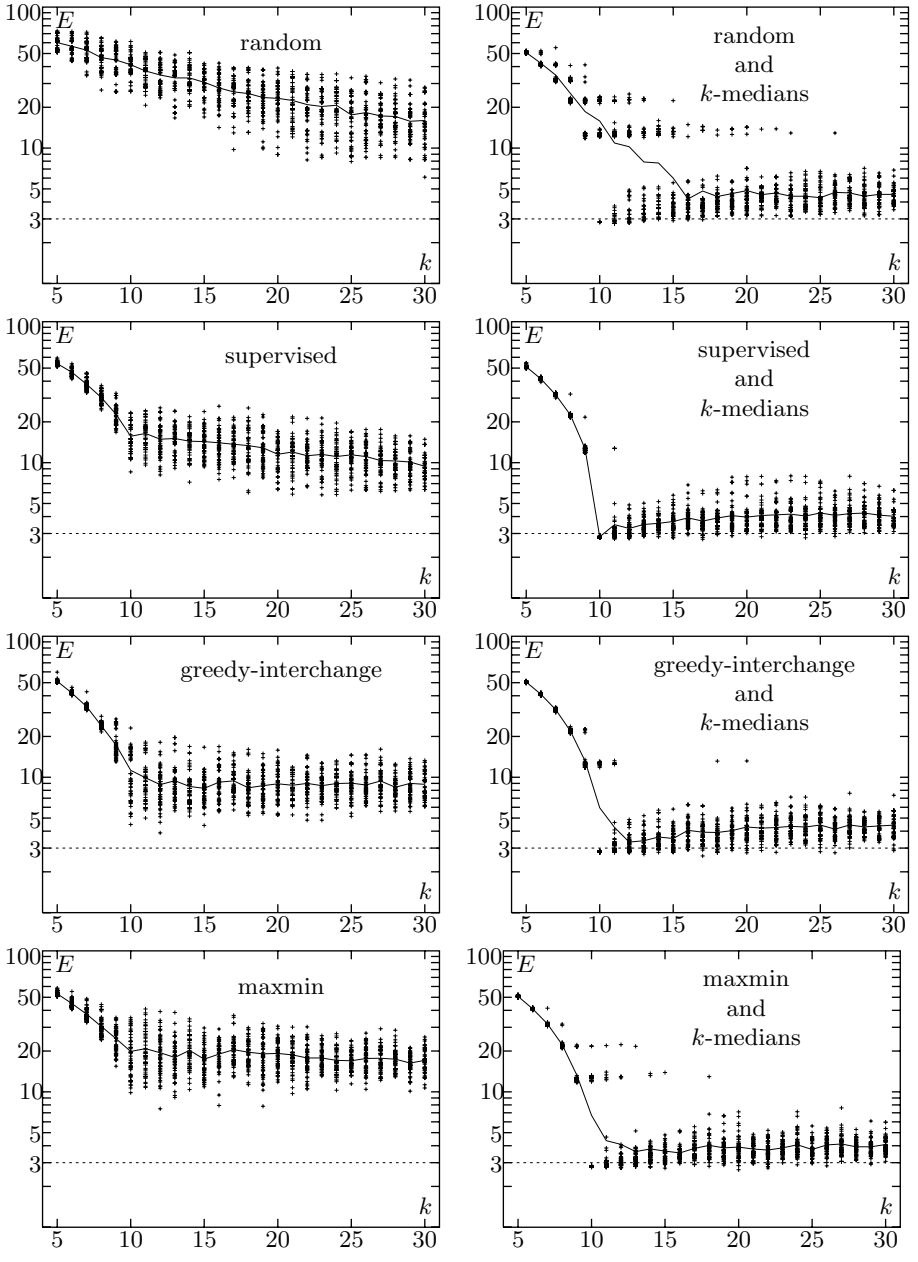

Fig. 4. Results obtained whith the k-medians algorithm tested as an exploratory data analysis technique. Each row of panels is associated with a different initialization technique; left and right columns of panels correspond to initial and optimized sets of representatives, respectively. Each plotted point in these panels represents the error rate of the nearest neighbour classifier based on a different set of representatives.

4.2 Unsupervised Selection of Prototypes

Two classification problems have been considered so as to test the k-medians algorithm as a procedure for unsupervised selection of prototypes. As in the case of the simple problem considered in the previous section, the first classification task involves 10 Gaussian equally-probable classes. Now, however, we have chosen 10-dimensional normal densities of different means and also different covariance matrices. A total of $20,000$ independent prototypes were extracted from this mixture; half for training and half for testing. Some error rates computed from these prototypes are: 9.9% for the Bayes classifier implemented as a Gaussian classifier from the true parameters of the densities; 22.3% for the minimum Euclidean distance classifier based on empirical class means; 31.5% for the same classifier based on empirical class medians; and 21.7% for the nearest neighbour classifier designed from the whole set of $10,000$ training prototypes. Clearly, this task is much more difficult than the previous simple problem. The error rate of a nearest neighbour classifier based on the prototypes selected by the k-medians algorithm will heavily depend on the value of k used. The larger value of k is used, the smaller error rate is expected. For instance, if a small value such as $k = 10$ is used, then an error rate not smaller than 31.5% should be expected. On the other hand, it is clear that rates close to 21.7% will be obtained for values of k approximating the number of available training prototypes.

The second task consists of classifying human banded chromosomes represented as strings. The data used for this task was extracted from a database of approximately $7,000$ chromosomes that where classified by cytogenetic experts [9]. Each digitized chromosome image was preprocessed through a procedure that starts obtaining an idealized, one-dimensional density profile that emphasizes the band pattern along the chromosome. The idealized profile is then mapped nonlinearly into a string composed of symbols over a certain alphabet. A total of 4400 samples was collected, 200 samples of each of the 22 non-sex chromosome types [2]. The standard procedure for estimating the error rate of a classifier applied to this task is a 2-fold cross-validation in which both sets are chosen to have 100 samples of each of the 22 non-sex chromosome types. Following this procedure, we have recently obtained an excellent error rate of 4.9% by application of the 12-nearest neighbours decision rule based on a time consuming *normalized* edit distance [4].

Three techniques have been compared in both tasks: a) random selection alone; b) random selection followed by the k-medians algorithm; and c) the k-medians algorithm initialized by the randomly started maxmin method. Supervised selection and the greedy-interchange combination have not been considered because of their high cost. For the first task, the three techniques considered have been executed 50 times for each $k \in \{10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000\}$. The same training set of $10,000$ training prototypes and the Euclidean distance were always used in these executions. The average error rate of the nearest neighbour classifier based on the representatives selected is shown in the panel at the left of figure 5 as a function of k. For the second task, values of $k \in \{22, 44, 88, 220, 440, 880, 2200\}$ have been tried. For each technique

and each of these values, an estimate of the error rate associated with the k-representatives-based 12-nearest neighbours classifier was obtained by averaging over the estimates computed from 5 executions of the 2-fold cross-validation procedure discussed above. Results are shown in the panel at the right of figure 5.

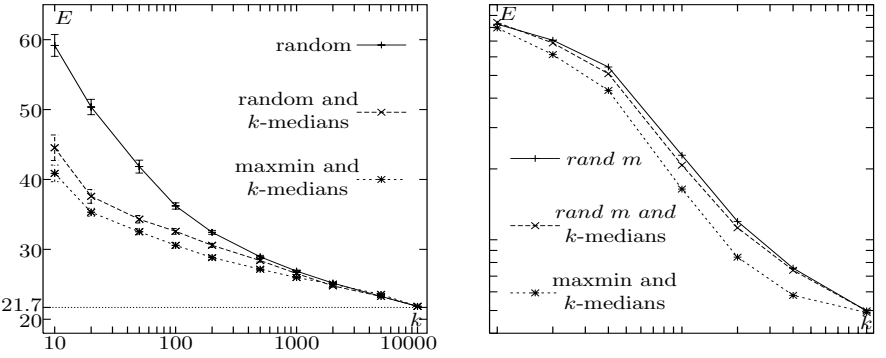

Fig. 5. Results provided by three techniques for unsupervised selection of prototypes tested on two classification tasks involving synthetic (left) and real (right) data. The average error rate of the $k'-$nearest neighbour classifier (left: $k' = 1$; right: $k' = 12$) is shown as a function of the number of representatives selected (k).

From the results of figure 5 it is clear that random selection is significantly improved by using the k-medians algorithm, and this is particularly true when the maxmin algorithm is used as initialization technique. For instance, in the second task, random selection leads to an error rate of 7.6% for $k = 880$, while the maxmin initialization reduces this figure to 5.8%. Although these results are quite satisfactory, they are not as good as one would expect. It would be nicer to see error rate curves ending with large nearly flat shapes but, in contrast, they markedly decay until the largest values of k tried. Obviously, larger sets of training prototypes would compensate for the slow rate of convergence showed by the $(k'-)$nearest neighbour(s) decision rule. On the other hand, this slow rate of convergence might be accelerated through optimization of the standard decision rule from the data.

5 Conclusions

Four initialization techniques for the *k-medians* clustering algorithm have been compared: *random* selection, *supervised* selection, the *greedy-interchange* algorithm and the *maxmin* algorithm. The capabilities of these techniques have been assessed through experiments in two typical applications of clustering; namely, exploratory data analysis and unsupervised prototype selection. Results clearly show the importance of a good initialization of the k-medians algorithm in all the

cases. Random initialization too often leads to bad final partitions, while best results are generally obtained using supervised selection. The greedy-interchange and the maxmin algorithms generally lead to partitions of high quality, without the manual effort of supervised selection. From these algorithms, the latter is generally preferred because of its better computational behaviour.

References

1. R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.
2. E. Granum and M. G. Thomason. Automatically Inferred Markov Network Models for Classification of Chromosomal Band Pattern Structures. *Cytometry*, 11, 1990.
3. A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.
4. A. Juan and E. Vidal. Application of the K-Nearest-Neighbour Rule to the Classification of Banded Chromosomes. In *ICPR'00 (accepted)*.
5. A. Juan and E. Vidal. Fast k-means-like clustering in metric spaces. *Pattern Recognition Letters*, 15(1):19–25, 1994.
6. A. Juan and E. Vidal. Fast Median Search in Metric Spaces. In A. Amin, D. Dori, P. Pudil, and H. Freeman, editors, *Advances in Pattern Recognition*, volume 1451, pages 905–912. Springer-Verlag, 1998.
7. O. Kariv and S. L. Hakimi. An Algorithmic Approach to Network Location Problems. II: The p-Medians. *SIAM Journal on Applied Math.*, 37(3):539–560, 1979.
8. I. Katsavounidis. A new initialization technique for generalized Lloyd iteration. *IEEE Signal Processing Letters*, 1(10):144–146, October 1994.
9. C. Lundsteen, J. Philip, and E. Granum. Quantitative analysis of 6985 digitized trypsin G-banded human metaphase chromosomes. *Clinical Genetics*, 18, 1980.
10. Mirchandani P.B. and Francis R.L. editors. *Discrete Location Theory*. Wiley, 1990.