An empirical comparison of four initialization methods for the K-Means algorithm

Peña, J.M., Lozano, J.A. and Larrañaga, P.

e-mail: ccbpepaj@si.ehu.es
http://www.sc.ehu.es/isg
Intelligent Systems Group
Dept. of Computer Science and Artificial Intelligence
University of the Basque Country
P.O. Box 649, E-20080 San Sebastián, Spain

Abstract

In this paper, we aim to compare empirically four initialization methods for the K-Means algorithm: random, Forgy, MacQueen and Kaufman. Although this algorithm is known for its robustness, it is widely reported in literature that its performance depends upon two key points: initial clustering and instance order. We conduct a series of experiments to draw up (in terms of mean, maximum, minimum and standard deviation) the probability distribution of the square-error values of the final clusters returned by the K-Means algorithm independently on any initial clustering and on any instance order when each of the four initialization methods is used. The results of our experiments illustrate that the random and the Kaufman initialization methods outperform the rest of the compared methods as they make the K-Means more effective and more independent on initial clustering and on instance order. In addition, we compare the convergence speed of the K-Means algorithm when using each of the four initialization methods. Our results suggest that the Kaufman initialization method induces to the K-Means algorithm a more desirable behaviour with respect to the convergence speed than the random initialization method.

Keywords: K-Means algorithm, K-Means initialization, partitional clustering, Genetic Algorithms.

1 Introduction

One of the basic problems that arises in a great variety of fields, including pattern recognition, machine learning and statistics, is the so-called *clustering problem* [1, 2, 5, 8, 14, 18]. The fundamental data clustering problem may be defined as discovering groups in data or grouping similar objects together. Each of these groups is called a *cluster*, a region in which the density of objects is locally higher than in other regions.

In this paper, data clustering is viewed as a data partitioning problem. Several approaches to find groups in a given database have been developed, but we focus on the *K-Means* algorithm [1, 11, 12, 14, 17, 20, 28] as it is one of the most used iterative partitional clustering algorithms and because it may also be used to initialize more expensive clustering algorithms (e.g., the EM algorithm) [3, 6, 22]. However, it is well-known that the K-Means algorithm suffers from *initial starting conditions* effects (initial clustering and instance order effects).

The main purpose of this paper is to compare four classical initialization methods for the K-Means algorithm according to two criteria: quality of the final clustering performed by the K-Means algorithm when each concrete initialization method is used (effectiveness) and sensitivity of the K-Means algorithm with each initialization method to initial starting conditions (robustness). A secondary objective is to compare the speedup of the convergence of the K-Means algorithm when using each concrete initialization method (efficiency). In order to reach such a set of conclusions, for each initialization method we draw up (mean and stardard deviation) the probability distribution of the *square-error* values of the final clusters that the K-Means algorithms returns, approach its extremes by *Genetic Algorithms* [13, 16] and compute the number of iterations that the K-Means algorithm needs to converge.

The remaining part of this paper is laid out as follows. In Section 2, we describe the K-Means algorithm and point out some of its drawbacks that motivate our work. Furthermore, we take a look at Genetic Algorithms. Section 3 introduces the four initialization methods and the three databases involved in the comparison. The experimental results are also summarized in this section. In Section 4, we draw conclusions.

2 Background

2.1 K-Means algorithm

Almost all partitional clustering methods are based upon the idea of optimizing a function F referred to as *clustering criterion* which, hopefully, translates one's intuitive notions on cluster into a reasonable mathematical formula. The function value usually depends on the current partition of the database $\{C_1, ..., C_K\}$. That is:

$$F : \mathcal{P}_K(\Omega) \to I\!R \tag{1}$$

where $\mathcal{P}_K(\Omega)$ is the set of all the partitions of the database $\Omega = \{w_1, ..., w_M\}$ in K non-empty clusters. Each w_i of the M intances of the database Ω is a N-dimensional vector. Concretely, the K-Means algorithm finds locally optimal solutions using as clustering criterion F the sum of the L^2 distance between each element and its nearest cluster center (*centroid*). This criterion is sometimes referred to as square-error criterion. Therefore, it follows that:

$$F(\{C_1, ..., C_K\}) = \sum_{i=1}^{K} \sum_{j=1}^{K_i} \|w_{ij} - \overline{w_i}\| \tag{2}$$

where K is the number of clusters, K_i the number of objects of the cluster i, w_{ij} is the j-th object of the i-th cluster and $\overline{w_i}$ is the centroid of the i-th cluster which is defined as:

$$\overline{w_i} = \frac{1}{K_i} \sum_{j=1}^{K_i} w_{ij}, \;\; i = 1, ..., K.$$

As can be seen in Figure 1 where the pseudo-code is presented, the K-Means algorithm is provided somehow with an initial partition of the database and the centroids of these

```
Step 1. Select shomehow an initial partition of the database in $K$
        clusters $\{C_1, ..., C_K\}$
Step 2. Calculate cluster centroids $\overline{w_i} = \frac{1}{K_i}\sum_{j=1}^{K_i} w_{ij},\ i = 1, ..., K$
Step 3. FOR every $w_i$ in the database and following the instance order DO
        Step 3.1. Reassign instance $w_i$ to its closest cluster centroid,
                  $w_i \in C_s$ is moved from $C_s$ to $C_t$ if $\|w_i - \overline{w_t}\| \leq \|w_i - \overline{w_j}\|$
                  for all  $j = 1, ..., K,\ j \neq s$
        Step 3.2. Recalculate centroids for clusters $C_s$ and $C_t$
Step 4. IF cluster membership is stabilized THEN stop
        ELSE go to Step 3.
```

Figure 1: The pseudo-code of the K-Means algorithm.

initial clusters are calculated. Then, the instances of the database are relocated to the cluster represented by the nearest centroid in an attempt to reduce the square-error. This relocation of the instances is done following the instance order. If an instance in the relocation step (Step 3) changes its cluster membership, then the centroids of the clusters C_s and C_t and the square-error should be recomputed. This process is repeated until convergence, that is, until the square-error cannot be further reduced which means no instance changes its cluster membership.

2.2 Drawbacks of the K-Means algorithm

Despite being used in a wide array of applications, the K-Means algorithm is not exempt of drawbacks. Some of these drawbacks have been extensively reported in literature. The most important are listed below:

- as many clustering methods, the K-Means algorithm assumes that the number of clusters K in the database is known beforehand which, obviously, is not necessarily true in real-world applications
- as an iterative technique, the K-Means algorithm is especially sensitive to initial starting conditions (initial clusters and instance order)
- the K-Means algorithm converges finitely to a local minima. The running of the algorithm defines a deterministic mapping from the initial solution to the final one.

To overcome the lack of knowledge on the real value in the database of the input parameter K, we adopt a rough but usual approach: to try clustering with several values of K.

The problem of initial starting conditions is not exclusive to the K-Means algorithm but shared with many clustering algorithms that work as a hill-climbing strategy whose deterministic behaviour leads to a local minima dependent on initial solution and on instance order. Although there is no guarantee of achieving a global minima, at least the convergence of K-Means algorithm is ensured [26].

Milligan in [24] shows the strong dependence of the K-Means algorithm on initial clustering and suggests that good final cluster structures can be obtained using Ward's hierarchical method [29] to provide the K-Means algorithm with initial clusters. Fisher in [10] propose creating the initial clusters by constructing an initial hierarchical clustering based upon the work [8]. Higgs et al. in [15] and Snarey et al. in [27] suggest using a MaxMin algorithm in order to select a subset of the original database as the initial centroids to establish the initial clusters. In a recent paper [22], Meilă and Heckerman present some experimental results of an instance of the EM algorithm reminiscent of the K-Means with three different initialization methods (being one of them a hierarchical agglomerative clustering method).

Most of the initialization methods that we have mentioned above do not constitute only initialization methods. They are clustering methods themselves and when used with the K-Means algorithm result in a hybrid clustering algorithm. Thus, these initialization methods suffer from the same problem as the K-Means algorithm and they have to be provided with an initial clustering. For the remaining part of this paper, we focus on much simpler and more inexpensive initialization methods that constitute the first initialization of any other more complex clustering method (see Section 3). This is the reason that motivates Bradley and Fayyad [4] to develop an algorithm for refining the initial seeds for the K-Means algorithm.

To overcome the possible bad effects of instance order, Fisher et al. in [9] present a procedure to order the instances of the database. They show that ordering instances, so that consecutive observations are dissimilar based on L^2, lead to good clusterings. Roure and Talavera in [25] propose a local strategy to reduce the effect of the instance ordering problem. Although they focus on incremental clustering procedures, their strategy is not coupled to any particular procedure and may be adapted to the K-Means algorithm.

2.3 Genetic Algorithms

As a part of our main objective, we aim to find the best and the worst set of initial starting conditions to approach the extremes of the probability distributions of the square-error values. Due to the computational expense of performing an exhaustive search, we tackle the problem using Genetic Algorithms.

Roughly speaking, we can say that Genetic Algoritms (GAs) are kinds of *evolutionary algorithms*, that is, probabilistic search algorithms which simulate natural evolution [13, 16]. GAs are used to solve combinatorial optimization problems following the rules of natural selection and natural genetics. They are based upon the survival of the fittest among string structures together with a structured yet randomized information exchange. Working in this way and under certain conditions, GAs evolve to the global optima with probability arbitrarily close to 1.

When dealing with GAs, the search space of a problem is represented as a collection of *individuals*. The individuals are represented by character strings. Each individual is coding a solution to the problem. In addition, each individual has associated a fitness measure. The part of the space to be examined is called the *population*. The purpose of the use of a GA is to find the individual from the search space with the best "genetic material".

Figure 2 shows the pseudo-code of the GA that we use. First, the initial population is chosen and the fitness of each of its individuals is determined. Next, in every iteration, two parents are selected from the population. This parental couple produces children

```
BEGIN AGA
  Choose initial population at random
  Evaluate initial population
  WHILE NOT convergence criterion DO
  BEGIN
    Select two parents from the current population
    Produce children by the selected parents
    Mutate the children
    Evaluate the children
    Replace the worst individual of the population by the best child
  END
  Output the best individual found
END AGA.
```

Figure 2: The pseudo-code of our Genetic Algorithm.

which, with a probability near zero, are mutated, i.e., their hereditary distinctions are changed. After the evaluation of the children, the worst individual of the population is replaced by the fittest of the children. This process is iterated until a convergence criterion is satisfied.

The operators which define the children production process and the mutation process are the *crossover* operator and the *mutation* operator respectively. Both operators are applied with different probabilities and play different roles in the GA. Mutation is needed to explore new areas of the search space and helps the algorithm avoid local optima. Crossover is aimed to increase the average quality of the population. By choosing adequate crossover and mutation operators as well as an appropriate reduction mechanism, the probability that the GA reaches a near-optimal solution in a reasonable number of iterations increases.

3 Experimental results

As we have mentioned above, our main purpose is to classify four classical initialization methods according to two criteria: quality of the final clustering returned by the K-Means algorithm when using each of these four initialization methods and sensitivity of the K-Means algorithm with each initialization method to initial starting conditions. In addition to our main objective, we are also interested in the convergence speed of the K-Means algorithm when each of the four compared initialization methods is used.

The initialization methods that we compare in this paper are:

- RANDOM, divide the database into a partition of K clusters at random. This is the most usual initialization method
- FORGY APPROACH (FA) proposed by Forgy in 1965, see [1], choose K instances of the database (seeds) at random and assign the rest of the instances to the cluster represented by the nearest seed

```
Step 1. Select as the first seed the most centrally located instance
Step 2. FOR every nonselected instance $w_i$ DO
        Step 2.1. FOR every nonselected instance $w_j$ DO
                  Calculate $C_{ji} = max(D_j - d_{ji}, 0)$ where $d_{ji} = \|w_i - w_j\|$ and
                  $D_j = min_s d_{sj}$ being $s$ one of the selected seeds
        Step 2.2. Calculate the gain of selecting $w_i$ by $\sum_j C_{ji}$
Step 3. Select the not yet selected instance $w_i$ which maximizes $\sum_j C_{ji}$
Step 4. IF there are $K$ selected seeds THEN stop
        ELSE go to Step 2.
Step 5. For having a clustering assign each nonselected instance to
        the cluster represented by the nearest seed.
```

Figure 3: The pseudo-code of the KA initialization method.

- MACQUEEN APPROACH (MA) proposed by MacQueen in [20], choose K instances of the database (seeds) at random. Assign, following the instance order, the rest of the instances to the cluster with nearest centroid. After each assignment a recalculation of the centroids has to be carried out

- KAUFMAN APPROACH (KA) proposed by Kaufman and Rousseeuw in [18]. In this case, the initial clustering is obtained by the successive selection of representative instances until K instances have been found. The first representative instance is the most centrally located instance in the database. The rest of representative instances are selected according to the heuristic rule of choosing the instances that promise to have around them a higher number of the rest of intances. See Figure 3 for the pseudo-code.

Some interesting differences between the four initialization methods are that (i) only KA is deterministic, (ii) RANDOM and FA generate an initial partition independently on the instance order and (iii) MA generates and initial partition dependently on instance order. Also, obvious differences between the computational expenses of the four initialization methods exist.

To carry out the experiments we use three well-known real-world databases from the Machine Learning Repository [23]:

- the Iris database which has 150 instances, 4 attributes and 3 clusters

- the Ruspini database which has 75 instances, 2 attributes and 4 clusters

- the Glass database which has 214 instances and 9 attributes. There are 7 clusters that can be grouped in 2 bigger clusters.

As we have already said, one of the disadvantages of the K-Means algorithm is that it assumes the number of clusters K is known or given as input. We use $K = 3, 4$ for the Iris database, $K = 4, 5$ for the Ruspini database and $K = 2, 7, 10$ for the Glass database.

3.1 Drawing up the probability distributions

Note that the K-Means algorithm with any initialization method is completely deterministic given concrete initial starting conditions. Thus, as our objective is to measure the effectiveness and the robustness of the K-Means algorithm when used with each of the four proposed initialization method independently on any concrete initial starting conditions, we aim to draw up (mean, maximum, minimum and standard deviation) the probability distributions of the square-error values marginalizing out the influence of initial partition and of instance order.

Due to the enormous size of the space of initial starting conditions, it is not possible to carry out the marginalizing process within a reasonable time. We propose sampling the space of initial starting conditions for each initialization method in order to approach the probability distributions. Since this sampling process is carried out in a different way for each initialization method, we dedicate the following subsections to explain the sampling proccess for each of the four methods.

3.1.1 Sampling process for RANDOM and FA

Since RANDOM and FA can generate an initial partition independently on instance order, we carried out the sample of the space of initial starting conditions in two steps. Firstly, we sampled the space of initial partitions at random obtaining $\{P_1, ..., P_m\}$. Secondly, we sampled the space of instance orders for each individual of the sample of the space of initial partitions at random obtaining $\{O_{i1}, ..., O_{in}\}$ for the i-th individual of the m initial partitions $\{P_1, ..., P_m\}$. By doing this, we constructed the sample of the space of initial starting conditions $\{(P_1, O_{11}), ..., (P_1, O_{1n}), ..., (P_m, O_{m1}), ..., (P_m, O_{mn})\}$. Each of the initial partitions was generated at random as follows: for RANDOM K clusters were generated at random, while for FA K seeds were generated at random and then, FA was performed. The K-Means algoritm was run for each individual of the sample of the space of initial starting conditions (P_i, O_{ij}) where $i = 1, ..., m$ and $j = 1, ..., n$. We marginalized out the influence of instance order when running the K-Means algorithm starting from P_i approaching:

$$F_i = \frac{1}{n} \sum_{j=1}^{n} F((P_i, O_{ij})) \qquad i = 1, ..., m \tag{3}$$

where F_i is mean of the square-error values when starting from P_i independently on instance order and $F((P_i, O_{ij}))$ is the square-error value when starting from P_i and the instance order is O_{ij}. Then, the probability distribution of the square-error values was approached using the values F_i for $i = 1, ..., m$. We used $m = 1,000$ (number of initial partitions) and $n = 1,000$ (number of instance orders). We repeated this process for the different values of K and the different databases that we have referred to aboved.

3.1.2 Sampling process for MA

We cannot generate an initial partition for MA independently on instance order (as we did with RANDOM and FA). Therefore, for this initialization method, we generated $m \cdot n$ instance orders at random. We divided this sample into m groups of n instance orders at random resulting in $\{\{O_{11}, ..., O_{1n}\}, ..., \{O_{m1}, ..., O_{mn}\}\}$. The MA initialization method was applied to each group $\{O_{i1}, ..., O_{in}\}$, $i = 1, ..., m$, with the same K seeds generated at random obtaining the set of initial partitions $\{P_{i1}, ..., P_{in}\}$ respectively, for $i = 1, ..., m$. Thus, we were able to compose the sample of the space of initial starting conditions as the set $\{(P_{11}, O_{11}), ..., (P_{1n}, O_{1n}), ..., (P_{m1}, O_{m1}), ..., (P_{mn}, O_{mn})\}$. The K-Means algorithm was run for each individual of the sample of the space of initial starting conditions (P_{ij}, O_{ij}) where $i = 1, ..., m$ and $j = 1, ..., n$. We marginalized out the influence of instance order when running the K-Means algorithm starting from the K seeds used to generate $\{P_{i1}, ..., P_{in}\}$ approaching:

$$F_i = \frac{1}{n} \sum_{j=1}^{n} F((P_{ij}, O_{ij})) \qquad i = 1, ..., m \tag{4}$$

where F_i is mean of the square-error values when starting from the K seeds used to generate $\{P_{i1}, ..., P_{in}\}$ independently on instance order and $F((P_{ij}, O_{ij}))$ is the square-error value when starting from P_{ij} and the instance order is O_{ij}. Then, the probability distribution of the square-error values was approach using the values F_i for $i = 1, ..., m$. We used $m = 1,000$ (number of initial partitions) and $n = 1,000$ (number of instance orders). We repeated this process for the different values of K and the different databases that we have referred to aboved.

3.1.3 Sampling process for KA

As KA always returns the same initial partition P for a given number of clusters and a database, we needed to sample the space of initial starting conditions in only one dimension, the space of instance orders. Thus, we generated a sample of the space of initial starting conditions $\{(P, O_1), ..., (P, O_n)\}$ where P is the initial partition and O_j, $j = 1, ..., n$, an instance order generated at random. The probability distribution of the square-error values was approached using the values $F_j = F(P, O_j)$ for $j = 1, ..., n$, where $F(P, O_j)$ is the square-error value when starting from P and the instance order is O_j. We used $n = 1,000$ (number of instance orders). We repeated this process for the different values of K and the different databases that we have referred to in the previous section.

3.1.4 Results

Figure 4-10 show the shape of the drawn up probability distributions. These figures are histograms which illustrates the hit counts of the different square-error values obtained when the K-Means algorithm is used with each of the four initialization methods for each concrete database and each concrete value of K. Note the presence of more than one local optima in most of the histograms. Table 1 summarizes the probability distributions using their sample mean $(\overline{F})$ and their sample standard deviation (S_{n-1}). From this table, we can conclude that RANDOM and KA initialization methods outperform the rest of the compared initialization methods. In addition to induce a more effective behaviour to the K-Means algorithm than the other two initialization methods, RANDOM and KA make the K-Means algorithm exhibit a more robust behaviour (note that the sample standard deviation is shorter for RANDOM and KA). This feature is illustrating the low sensitivity of the K-Means algorithm to initial starting conditions when RANDOM or KA are used. In [22] a similar result can be found for the random initialization method.

Despite the fact that KA performs slightly better than RANDOM for all the cases except for the Ruspini database with $K = 2, 4$, it is hard to choose KA over RANDOM.

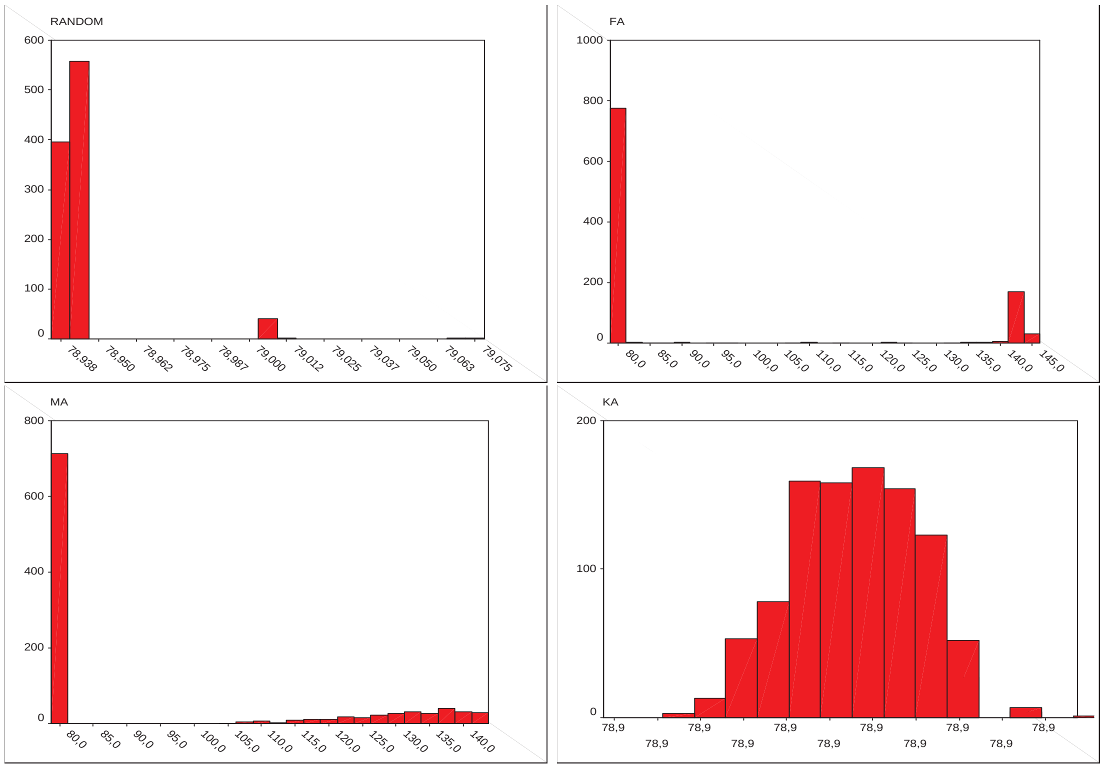

Figure 4: Histograms for the Iris database with $K = 3$. Hit counts (Y axis) for the different square-error values (X axis) when each of the four initialization methods is used.

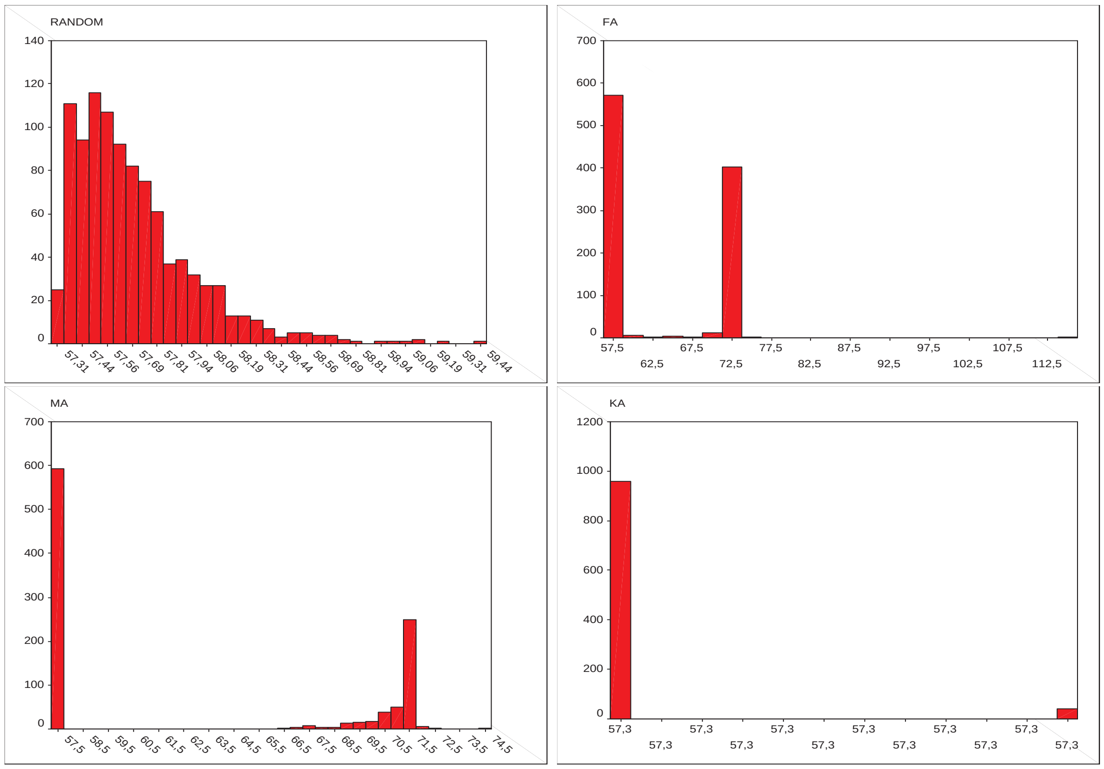

Figure 5: Histograms for the Iris database with $K = 4$. Hit counts (Y axis) for the different square-error values (X axis) when each of the four initialization methods is used.

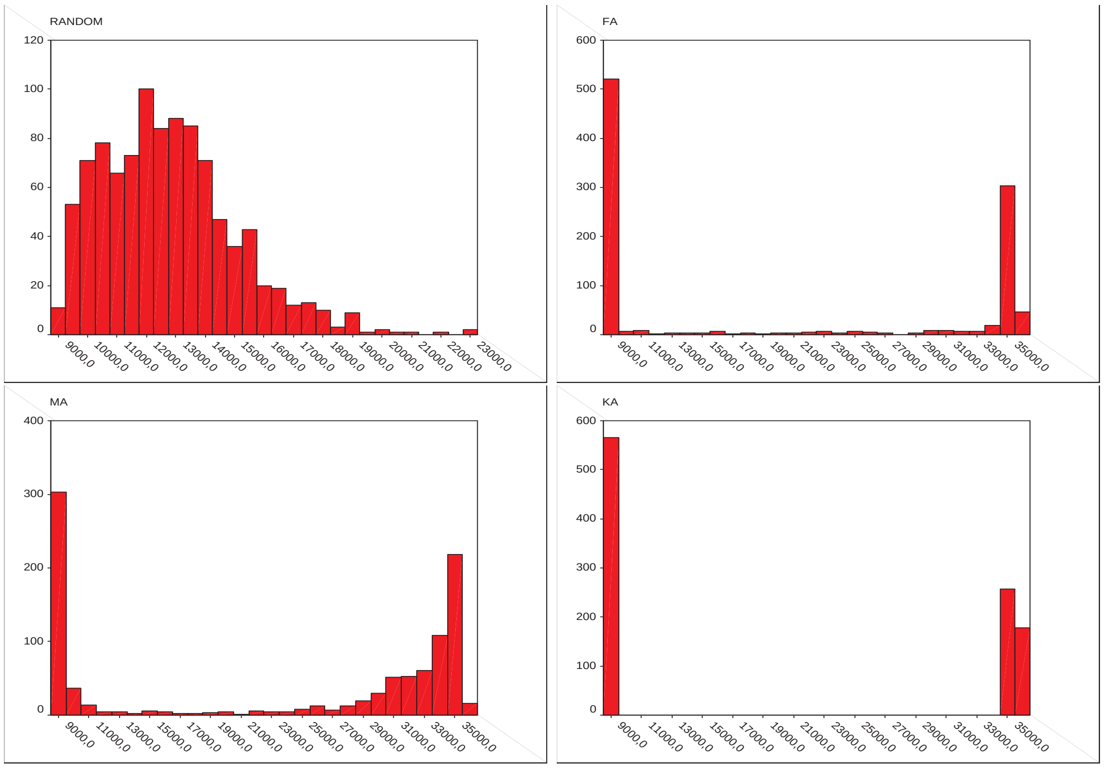

Figure 6: Histograms for the Ruspini database with $K = 4$. Hit counts (Y axis) for the different square-error values (X axis) when each of the four initialization methods is used.

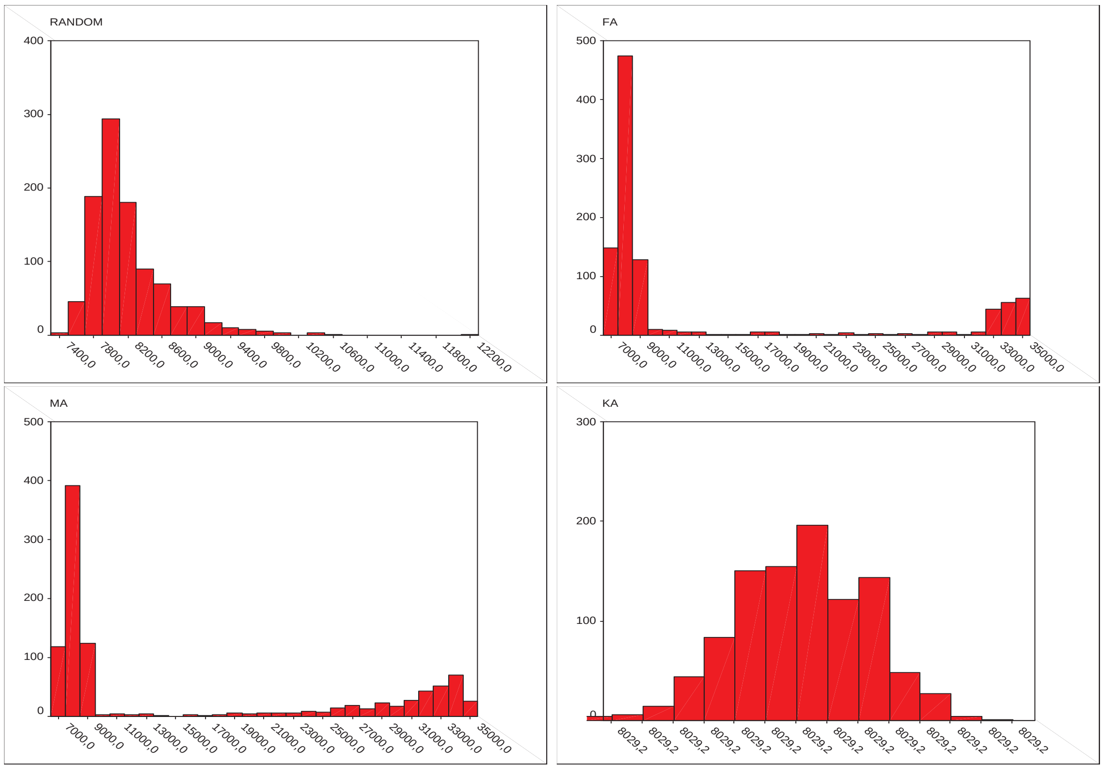

Figure 7: Histograms for the Ruspini database with $K = 5$. Hit counts (Y axis) for the different square-error values (X axis) when each of the four initialization methods is used.

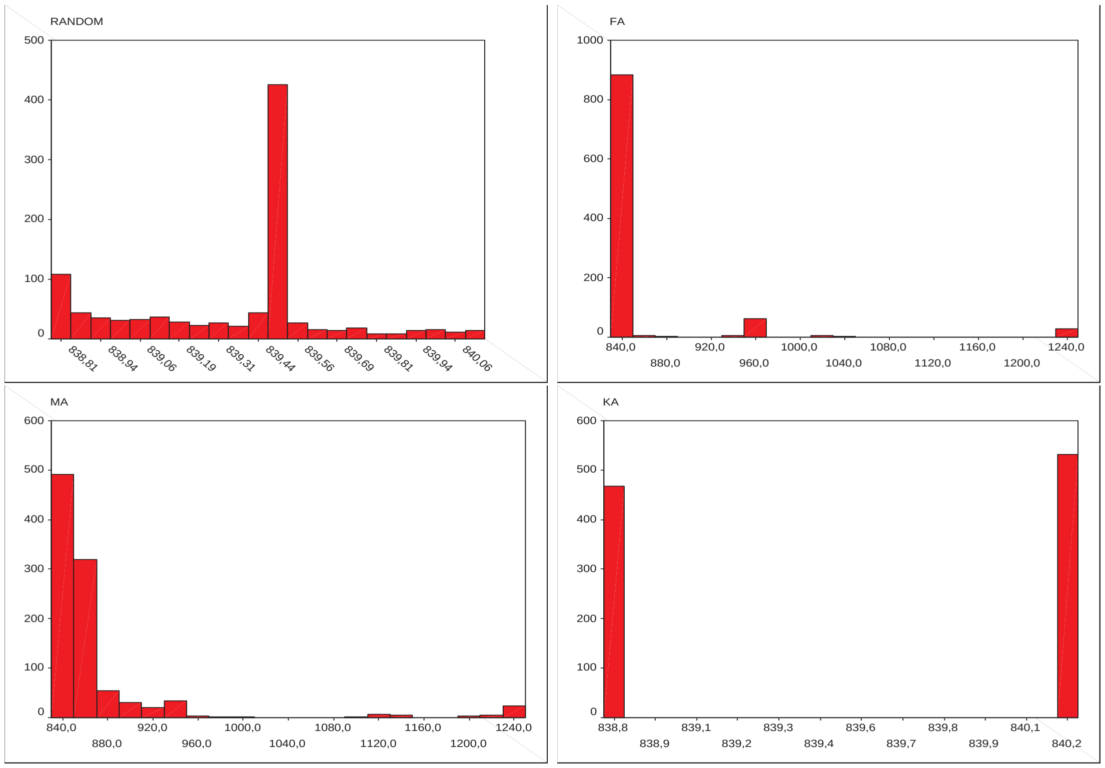

Figure 8: Histograms for the Glass database with $K = 2$. Hit counts (Y axis) for the different square-error values (X axis) when each of the four initialization methods is used.

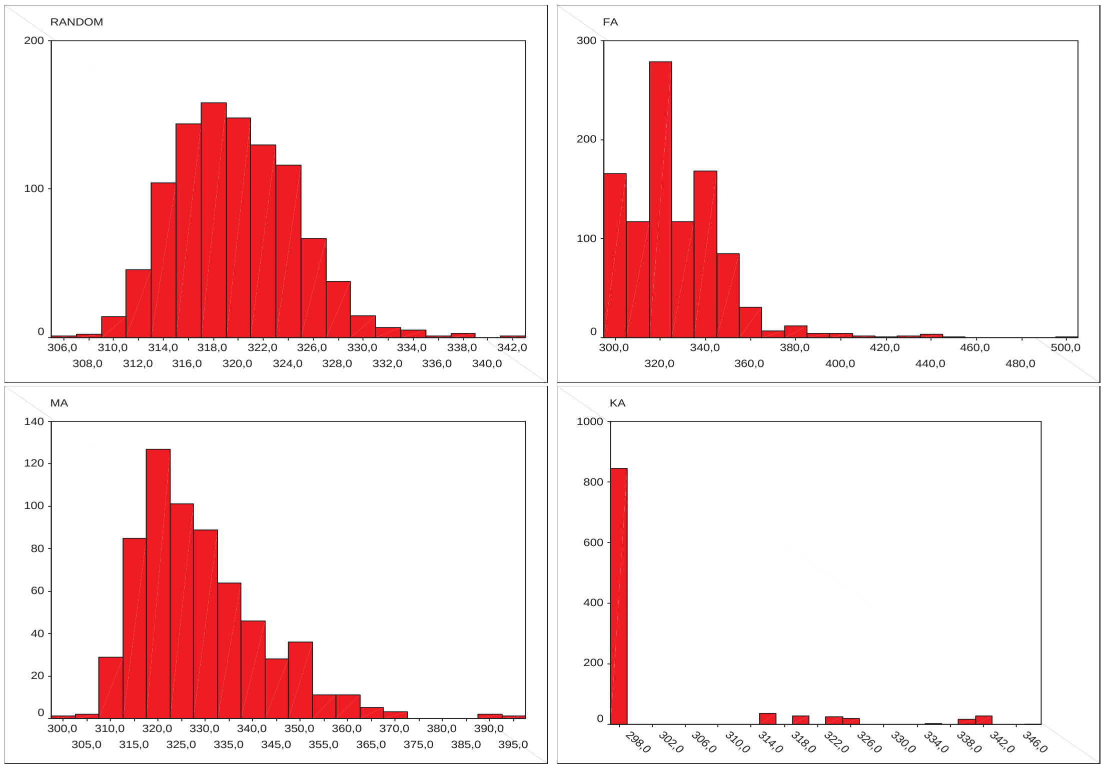

Figure 9: Histograms for the Glass database with $K = 7$. Hit counts (Y axis) for the different square-error values (X axis) when each of the four initialization methods is used.

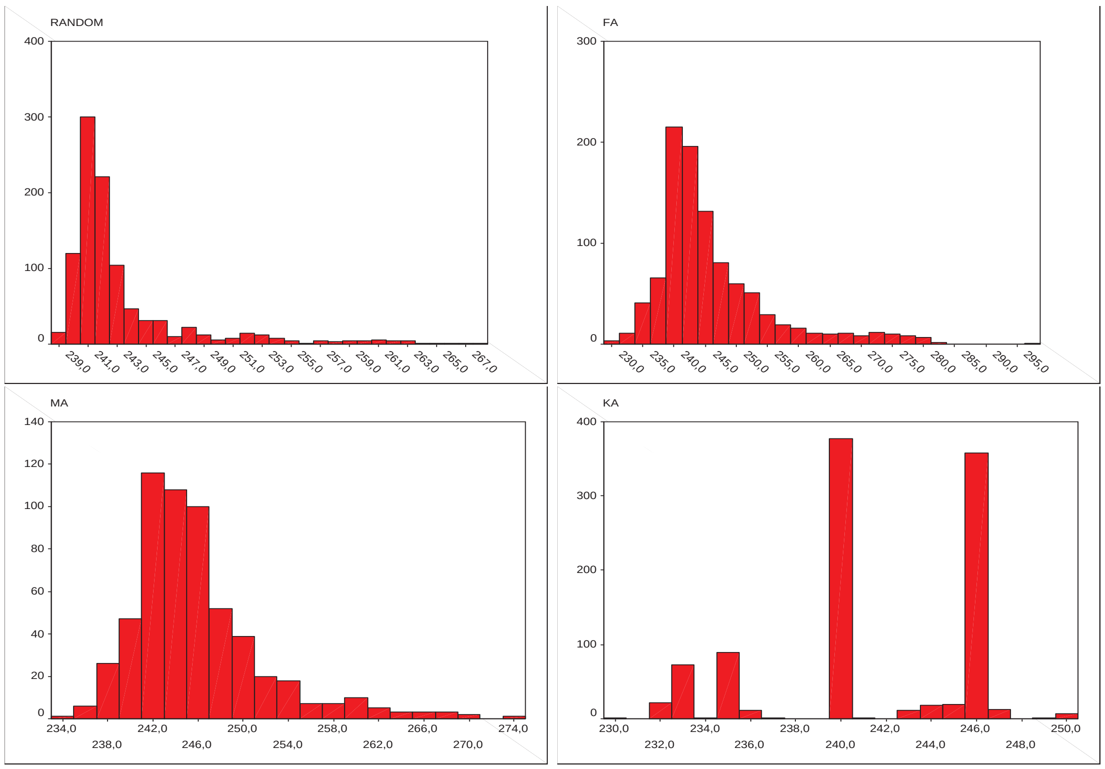

Figure 10: Histograms for the Glass database with $K = 10$. Hit counts (Y axis) for the different square-error values (X axis) when each of the four initialization methods is used.

So, we defer the selection of KA over RANDOM until more evidence appears.

We conducted seven nonparametric tests with a Kruskal-Wallis variance analysis [19] (one for each concrete database and each considered value of K). The purpose of each of these tests was to see whether the four independent samples of the space of final clustering values (each of them corresponding to one of the four initialization methods given a database and a concrete value for K) are from the same population. The results were the same for the seven nonparametric tests that we performed: the four distributions were not significantly similar ($\alpha = 0.05$).

3.2 Looking for the extremes of the probability distributions

Although the samples of the space of initial starting conditions that we have used in the previous section were of considerable size, we are not sure that the best and the worst set of initial starting conditions were included in those samples. In order to have more information on the extremes of the probability distributions of the square-error values, we aim to find the best and the worst set of initial starting conditions. Due to the computational expense of performing an exhaustive search, a GA is used.

Our GA works with a population of 50 individuals. Each of them encoding an instance order, i.e., each individual is a permutation of the $\{1, ..., M\}$ set (M instances in the database). The initial population is generated at random. The fitness of each individual of the population is the square-error value returned by a running of the K-Means algorithm. For each of these runnings the instance order encoded in the individual is used. The initial partition for each of these runnings is generated using the initialization

		Iris		Ruspini		Glass		
		$K=3$	$K=4$	$K=4$	$K=5$	$K=2$	$K=7$	$K=10$
RANDOM	$\overline{F}$	78.95	57.7	12,763.7	8,206.2	839.35	319.8	243.3
	S_{n-1}	0.01	0.3	2,255.16	468.95	0.32	4.86	4.48
FA	$\overline{F}$	92.8	63.3	20,101.6	13,290.9	860.7	326.9	246.1
	S_{n-1}	26.13	7.07	12,343.24	10,086.96	73.35	22.2	9.42
MA	$\overline{F}$	93.8	62.89	23,901.1	15,892.91	872.7	328.9	245.9
	S_{n-1}	23.9	6.69	11,463.42	10,920.31	74.71	13.47	5.94
KA	$\overline{F}$	78.94	57.32	20,682.6	8,029.17	839.53	302.8	241.4
	S_{n-1}	0	0.01	13,210.82	0	0.7	11.3	4.64

Table 1: Experimental results, mean ($\overline{F} = \frac{1}{n}\sum_{i=1}^{n} F_i$) and sample standard deviation ($S_{n-1} = \sqrt{\frac{\sum_{i=1}^{n}(F_i - \overline{F})^2}{n-1}}$).

method whose extremes we are looking for. The parental couple is selected by means of a biased range selection process. To produce the children, we use the order based crossover operator [7]. The mutation probability is 0.01, and the mutation operator is the swapping of a couple of elements of the permutation encoded in each of the children strings. Only the best of the two children strings is added to the population obtaining the intermediate population. The worst of the 51 individuals is removed from this intermediate population obtaining the new population. We iterate this process until no further improvement is found in the last 500 iterations.

To obtain the results that we resume in Table 2, we run the GA described above 10 times for each combination of one the four databases that we use, one of the considered values for K for this database and one of the compared initialization methods. Each entry of Table 2 resumes these 10 runnings using the best square-error value that the GA reaches (positive optimization, i.e., left extreme of the probability distribution). In order to complete Table 3, we follow the same process but this time we want the worst square-error value (negative optimization, i.e., right extreme of the probability distribution).

Table 2 shows that the four initialization methods almost always make the K-Means algorithm able to reach what is supposed to be the best final partition. Table 3 summarizes the worst square-error values that the GA reaches. Every initial partition that can be obtained with FA and MA can also be reached by RANDOM. Therefore, RANDOM can behave at least as bad as FA and MA. We should conclude from Table 3 that even when it is known that both extremes are reacheable by RANDOM, it is quite difficult to fall into the worst extreme when using RANDOM initialization method, since our GA is not able to do so.

Table 3 also shows that KA in the worst case is better than the rest, i.e., it makes the K-Means algorithm unable to reach as bad final partitions as the rest of methods make. This might be a property of KA (instead of assuming that our GA is unable to reach the real extreme as it happens with RANDOM) as not every initial partition reached by FA and MA can be reached by KA and therefore, we cannot assume that KA should be at least as bad as MA and FA.

	Iris		Ruspini		Glass		
	$K = 3$	$K = 4$	$K = 4$	$K = 5$	$K = 2$	$K = 7$	$K = 10$
RANDOM	78.94	57.32	9,098.35	7,133.44	838.78	298.16	230.07
FA	79.94	57.32	9,098.45	7,133.52	838.78	298.16	230.07
MA	79.94	57.32	9,098.48	7,133.54	838.78	298.16	230.07
KA	78.94	57.32	9,098.44	8,029.16	838.79	298.16	230.07

Table 2: Best square-error values over 10 runnings of the GA per entry, i.e., left extremes of the probability distributions (best instance order).

	Iris		Ruspini		Glass		
	$K = 3$	$K = 4$	$K = 4$	$K = 5$	$K = 2$	$K = 7$	$K = 10$
RANDOM	78.95	71.66	36,160.5	35,242.69	840.19	571.6	321.31
FA	145.23	144.1	36,160.6	35,311.39	1,243.83	590.91	504.41
MA	145.23	144.1	36,160.19	35,311.38	1,243.83	590.38	503.73
KA	78.94	67.47	36,160.21	8,032.4	840.19	375.78	287.02

Table 3: Worst square-error values over 10 runnings of the GA per entry, i.e., right extremes of the probability distributions (worst instance order).

3.3 Convergence speed

As Table 4 reveals, MA is the initialization method that makes the K-Means algorithm reach an earlier convergence. On the other hand, RANDOM is the initialization method that induces to the K-Means algorithm the slowest convergence speed.

We are interested in comparing RANDOM and KA initialization methods as they induce to the K-Means algorithm the best performance. Hence, we can conclude from Table 4 that KA makes the K-Means algorithm need less number of iterations to converge than RANDOM in four of the seven cases. In these four cases, the K-Means algorithm initialized with RANDOM may need up to almost five more iterations (on average) to converge than when initialized with KA. On the other hand, for the remaining three cases where RANDOM induces to the K-Means algorithm a more efficient behaviour than KA, the use of KA implies the need of less than one more iteration (on average) of the K-Means algorithm to converge than when using RANDOM. Thus, KA is preferred over RANDOM with respect to the convergence speed of the K-Means algorithm.

4 Conclusions

It is widely reported that the K-Means algorithm suffers from initial starting conditions effects. Keeping in mind this idea, our main purpose was to compare empirically four classical initialization methods for the K-Means algorithm. This comparison was done based upon the effectiveness, the robustness and the convergence speed of the K-Means algorithm when using each of the four initialization methods. We concluded that RANDOM and KA initialization methods outperformed the other two methods with respect to the effectiveness and the robustness of the K-Means algorithm when these two initialization methods were used. In addition, KA exhibited a more desirable behaviour as it made the K-Means algorithm unable to reach as bad partitions as the rest of the

		Iris		Ruspini		Glass		
		$K=3$	$K=4$	$K=4$	$K=5$	$K=2$	$K=7$	$K=10$
RANDOM	$\overline{ite}$	6.75	6.08	3.15	3.68	4.35	8.73	8.71
	sd_{ite}	0.13	0.19	0.18	0.08	0.20	0.92	0.28
FA	$\overline{ite}$	4.25	5.57	2.47	2.91	3.60	8.99	8.59
	sd_{ite}	1.34	2.18	0.72	0.73	1.03	2.41	1.56
MA	$\overline{ite}$	3.23	4.54	2.00	2.41	3.16	7.81	7.82
	sd_{ite}	0.89	1.93	0.48	0.56	0.48	0.42	1.65
KA	$\overline{ite}$	2.00	2.08	3.64	3.16	3.09	9.32	8.81
	sd_{ite}	0.00	0.28	0.79	0.48	0.31	1.73	2.70

Table 4: Mean ($\overline{ite}$) and sample standard deviation (sd_{ite}) of the number of iterations of the K-Means algorithm with each initialization method.

compared methods.

From the study of the convergence speed induced by each of the four initialization methods, we were able to conclude that KA showed its capability to induce to the K-Means algorithm a more desirable behaviour than RANDOM.

Note that RANDOM is usually considered as the standard initialization method for the K-Means algorithm and, as our empirical comparison shows, this is also a good election because of its good performance. A similar result with respect to RANDOM can be found in [4] for the majority of the databases of the Machine Learning Repository [23].

5 Acknowledgments

The authors would like to thank the two anonymous referees for their useful comments related to this work.

This research was supported by the spanish Ministerio de Educación y Cultura under AP97 44673053 grant.

References

[1] Anderberg, M.R. (1973). *Cluster Analysis for Applications.* Academic Press, New York, NY.

[2] Banfield, J. and Raftery, A. (1993). Model-based Gaussian and non-Gaussian Clustering. *Biometrics*, 49, 803-821.

[3] Bishop, C. (1995). *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford.

[4] Bradley, P.S. and Fayyad, U.M. (1998). Refining Initial Points for K-Means Clustering. *Proceedings of the Fifteenth International Conference on Machine Learning*, 91-99. Morgan Kaufmann Publishers, Inc., San Francisco, CA.

[5] Chandon, J.L. and Pinson, S. (1980). *Analyse Typologique. Théories et applications.* Masson, Paris.

[6] Cheeseman, P. and Stutz, J. (1995). Bayesian classification (AutoClass): Theory and results. *Advances in Knowledge Discovery and Data Mining*, 153-180. AAAI Press, Menlo Park, CA.

[7] Davis, L. (1985). Applying adaptive algorithms to epistatic domains. *Proceedings of the International Joint Conference on Artificial Intelligence*, 162-164.

[8] Fisher, D. (1987). Knowledge Acquisition via Incremental Conceptual Clustering. *Machine Learning*, 2, 139-172.

[9] Fisher, D., Xu, L. and Zard, N. (1992). Ordering effects in clustering. *Proceedings of the Ninth International Conference on Machine Learning*, 163-168. Morgan Kaufmann Publishers, Inc., San Mateo, CA.

[10] Fisher, D. (1996). Iterative Optimization and Simplification of Hierarchical Clusterings. *Journal of Artificial Intelligence Research*, 4, 147-179.

[11] Forgy, E. (1965). Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications. *Biometrics*, 21, 768.

[12] Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition.* Academic Press, San Diego, CA.

[13] Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Massachussets.

[14] Hartigan, J.A. (1975). *Clustering Algorithms.* John Wiley & Sons, Inc., Canada.

[15] Higgs, R.E., Bemis, K.G., Watson, I.A. and Wikel, J.H. (1997). Experimental designs for selecting molecules from large chemical databases. *J. Chem. Inf. Comput. Sci.*, 37, 861-870.

[16] Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Michigan.

[17] Jain, A.K. and Dubes, R.C. (1988). *Algorithms for Clustering Data.* Prentice Hall, Englewood Cliffs.

[18] Kaufman, L. and Rousseeuw, P.J. (1990). *Finding Groups in Data. An Introduction to Cluster Analysis.* John Wiley & Sons, Inc., Canada.

[19] Kruskal, W.H. and Wallis, W.A. (1952). Use of ranks in one-criterion analysis of variance. *J. Amer. Statist. Assoc.*, 47, 583-621; errata, ibid., 48, 907-911.

[20] MacQueen, J.B. (1967). Some Methods for Classification and Analysis of Multivariate Observations. *Proc. Symp. Math. and Probability, 5th., Berkeley, 1, 281-297, AD 669871.* University of California Press, Berkeley, CA.

[21] McLachlan, G.J. and Basford, K.E. (1988). *Mixture Models.* Marcel Dekker, Inc., New York, NY.

[22] Meilă, M. and Heckerman, D. (1998). An Experimental Comparison of Several Clustering and Initialization Methods. *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, 386-395. Morgan Kaufmann Publishers, Inc., San Francisco, CA.

[23] Merz, C., Murphy, P. and Aha, D. (1997). UCI repository of Machine Learning databases. Department of Information and Computer Science, University of California, Irvine. `http://www.ics.uci.edu/mlearn/MLRepository.html`.

[24] Milligan, G.W. (1980). An examination of the effect of six types of error perturbation on fifteen clustering algorithms. *Psychometrika*, 45, 325-342.

[25] Roure, J. and Talavera, L. (1998). Robust incremental clustering with bad instance orderings: a new strategy. *Proceedings of the Sixth Iberoamerican Conference on Artificial Intelligence*, 136-147. Helder Coelho, eds., Lisbon.

[26] Selim, S.Z. and Ismail, M.A. (1984). K-Means-Type Algorithms: A Generalized Convergence Theorem and Characterization of Local Optimality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6, 81-87.

[27] Snarey, M., Terrett, N.K., Willet, P. and Wilton D.J. (1997). Comparison of algorithms for dissimilarity-based compound selection. *J. Mol. Graphics & Modelling*, 15, 372-385.

[28] Tou, J. and González, R. (1974). *Pattern Recognition Principles.* Addison Wesley, Reading, MA.

[29] Ward, J.H. (1963). Hierarchical grouping to optimize an objective function. *Journal of American Statistical Association, 58, 236-244.*