The global k-means clustering algorithm

Aristidis Likas[a,*] Nikos Vlassis[b] Jakob J. Verbeek[b]

[a]Department of Computer Science
University of Ioannina
45110 Ioannina
Greece
arly@cs.uoi.gr

[b]Computer Science Institute
University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam
The Netherlands
{vlassis,jverbeek}@science.uva.nl

Abstract

We present the global k-means algorithm which is an incremental approach to clustering that dynamically adds one cluster center at a time through a deterministic global search procedure consisting of N (with N being the size of the data set) executions of the k-means algorithm from suitable initial positions. We also propose modifications of the method to reduce the computational load without significantly affecting solution quality. The proposed clustering methods are tested on well-known data sets and they compare favorably to the k-means algorithm with random restarts.

Keywords: Clustering; K-means algorithm; Global optimization; k-d trees, Data mining.

1 Introduction

A fundamental problem that frequently arises in a great variety of fields such as pattern recognition, image processing, machine learning and statistics is the clustering problem [1]. In its basic form the clustering problem is defined as the problem of finding homogeneous groups of data points in a given data set. Each of these groups is called a cluster and can be defined as a region in which the density of objects is locally higher than in other regions.

The simplest form of clustering is partitional clustering which aims at partitioning a given data set into disjoint subsets (clusters) so that specific clustering criteria are optimized. The most

*corresponding author

widely used criterion is the clustering error criterion which for each point computes its squared distance from the corresponding cluster center and then takes the sum of these distances for all points in the data set. A popular clustering method that minimizes the clustering error is the k-means algorithm. However, the k-means algorithm is a local search procedure and it is well-known that it suffers from the serious drawback that its performance heavily depends on the initial starting conditions [2]. To treat this problem several other techniques have been developed that are based on stochastic global optimization methods (eg. simulated annealing, genetic algorithms). However, it must be noted that these techniques have not gained wide acceptance and in many practical applications the clustering method that is used is the k-means algorithm with multiple restarts [1].

In this paper we propose the global k-means clustering algorithm, which constitutes a deterministic effective global clustering algorithm for the minimization of the clustering error that employs the k-means algorithm as a local search procedure. The algorithm proceeds in an incremental way: to solve a clustering problem with M clusters, all intermediate problems with $1, 2, \ldots, M-1$ clusters are sequentially solved. The basic idea underlying the proposed method is that an optimal solution for a clustering problem with M clusters can be obtained using a series of local searches (using the k-means algorithm). At each local search the $M-1$ cluster centers are always initially placed at their optimal positions corresponding to the clustering problem with $M-1$ clusters. The remaining M-th cluster center is initially placed at several positions within the data space. Since for $M = 1$ the optimal solution is known, we can iteratively apply the above procedure to find optimal solutions for all k-clustering problems $k = 1, \ldots, M$. In addition to effectiveness, the method is deterministic and does not depend on any initial conditions or empirically adjustable parameters. These are significant advantages over all clustering approaches mentioned above.

In the following section we start with a formal definition of the clustering error and a brief description of the k-means algorithm and then we describe the proposed global k-means algorithm. Section 3 describes modifications of the basic method that require less computation at the expense of being slightly less effective. Section 4 provides experimental results and comparisons with the k-means algorithm with multiple restarts. Finally Section 5 provides conclusions and describes directions for future research.

2 The global k-means algorithm

Suppose we are given a data set $X = \{x_1, \ldots, x_N\}$, $x_n \in R^d$. The M-clustering problem aims at partitioning this data set into M disjoint subsets (clusters) $C_1, \ldots, C_M$, such that a clustering criterion is optimized. The most widely used clustering criterion is the sum of the squared Euclidean distances between each data point x_i and the centroid m_k (cluster center) of the subset C_k which contains x_i. This criterion is called clustering error and depends on the cluster centers $m_1, \ldots, m_M$:

$$E(m_1, \ldots, m_M) = \sum_{i=1}^{N} \sum_{k=1}^{M} I(x_i \in C_k) ||x_i - m_k||^2 \tag{1}$$

where $I(X) = 1$ if X is true and 0 otherwise.

The k-means algorithm finds locally optimal solutions with respect to the clustering error. It is a fast iterative algorithm that has been used in many clustering applications. It is a point-based clustering method that starts with the cluster centers initially placed at arbitrary positions and proceeds by moving at each step the cluster centers in order to minimize the clustering error. The main disadvantage of the method lies in its sensitivity to initial positions of the cluster centers. Therefore, in order to obtain near optimal solutions using the k-means algorithm several runs must be scheduled differing in the initial positions of the cluster centers.

In this paper, the *global k-means* clustering algorithm is proposed, which constitutes a deterministic global optimization method that does not depend on any initial parameter values and employs the k-means algorithm as a local search procedure. Instead of randomly selecting initial values *for all cluster centers* as is the case with most global clustering algorithms, the proposed technique proceeds in an incremental way attempting to optimally add one new cluster center at each stage.

More specifically, to solve a clustering problem with M clusters the method proceeds as follows. We start with one cluster ($k = 1$) and find its optimal position which corresponds to the centroid of the data set X. In order to solve the problem with two clusters ($k = 2$) we perform N executions of the k-means algorithm from the following initial positions of the cluster centers: the first cluster center is always placed at the optimal position for the problem with $k = 1$, while the second center at execution n is placed at the position of the data point x_n ($n = 1, \ldots, N$). The best solution obtained after the N executions of the k-means algorithm is considered as the solution for the clustering problem with $k = 2$. In general, let $(m_1^*(k), \ldots, m_k^*(k))$ denote the final solution for k-clustering problem. Once we have found the solution for the $(k-1)$-clustering problem, we try to find the solution of the k-clustering problem as follows: we perform N runs of the k-means algorithm with k clusters where each run n starts from the initial state $(m_1^*(k-1), \ldots, m_{(k-1)}^*(k-1), x_n)$. The best solution obtained from the N runs is considered as the solution $(m_1^*(k), \ldots, m_k^*(k))$ of the k-clustering problem. By proceeding in the above fashion we finally obtain a solution with M clusters having also found solutions for all k-clustering problems with $k < M$.

The latter characteristic can be advantageous in many applications where the aim is also to discover the 'correct' number of clusters. To achieve this, one has to solve the k-clustering problem for various numbers of clusters and then employ appropriate criteria for selecting the most suitable value of k [3]. In this case the proposed method directly provides clustering solutions for all intermediate values of k, thus requiring no additional computational effort.

In what may be a concern of computational complexity, the method requires N executions of the k-means algorithm for each value of k ($k = 1, \ldots, M$). Depending on the available resources and the values of N and M, the algorithm may be an attractive approach, since, as experimental results indicate, the performance of the method is excellent. Moreover, as we will show later, there are several modifications that can be applied in order to reduce the computational load.

The rationale behind the proposed method is based on the following assumption: an optimal clustering solution with k clusters can be obtained through local search (using k-means) starting from an initial state with

- the $k-1$ centers placed at the optimal positions for the $(k-1)$-clustering problem and

- the remaining k-th center placed at an appropriate position to be discovered.

This assumption seems very natural: we expect that the solution of the k-clustering problem to be *reachable* (through local search) from the solution of $(k-1)$-clustering problem, once the additional center is placed at an appropriate position within the data set. It is also reasonable to restrict the set of possible initial positions of the k-th center to the set X of available data points. It must be noted that this is a rather computational heavy assumption and several other options (examining fewer initial positions) may also be considered. The above assumptions are also verified experimentally, since in all experiments (and for all values of k) the solution obtained by the proposed method was at least as good as that obtained using numerous random restarts of the k-means algorithm. In this spirit, we can cautiously state that the proposed method is *experimentally optimal* (although it is difficult to prove theoretically).

3 Speeding-up execution

Based on the general idea of the global k-means algorithm, several heuristics can be devised to reduce the computational load without significantly affecting the quality of the solution. In the following subsections two modifications are proposed, each one referring to a different aspect of the method.

3.1 The fast global k-means algorithm

The fast global k-means algorithm constitutes a straightforward method to accelerate the global k-means algorithm. The difference lies in the way a solution for the k-clustering problem is obtained, given the solution of the $(k-1)$-clustering problem. For each of the N initial states $(m_1^*(k-1), \ldots, m_{(k-1)}^*(k-1), x_n)$ we do not execute the k-means algorithm until convergence to obtain the final clustering error E_n. Instead we compute an *upper* bound $E_n \leq E - b_n$ on the resulting error E_n for all possible allocation positions x_n, where E is the error in the $(k-1)$-clustering problem and b_n as defined in Eq. (2). We then initialize the position of the new cluster center at the point x_i that minimizes E_n, or equivalently that maximizes b_n, and execute the k-means algorithm to obtain the solution with k clusters. Formally we have

$$b_n = \sum_{j=1}^{N} \max(d_{k-1}^j - ||x_n - x_j||^2, 0), \tag{2}$$

$$i = \arg\max_n b_n \tag{3}$$

where d_{k-1}^j is the squared distance between x_j and the closest center among the $k-1$ cluster centers obtained so far (ie., center of the cluster where x_j belongs). The quantity b_n measures the *guaranteed* reduction in the error measure obtained by inserting a new cluster center at position x_n.

Suppose the solution of the $(k-1)$-clustering problem is $(m_1^*(k-1), \ldots, m_{(k-1)}^*(k-1))$ and a new cluster center is added at location x_n. Then the new center will allocate all points x_j whose squared distance from x_n is smaller than the distance d_{k-1}^j from their previously closest center. Therefore, for each such data point x_j the clustering error will decrease by $d_{k-1}^j - |x_n - x_j|^2$. The summation over all such data points x_j provides the quantity b_n for a specific insertion location x_n. Since the k-means algorithm is guaranteed to decrease the clustering error at each step, $E - b_n$ upper bounds the error measure that will be obtained if we run the algorithm until convergence after inserting the new center at x_n (this is the error measure used in the global k-means algorithm).

Experimental results (see next section) suggest that using the data point that minimizes this bound leads to results almost as good as those provided by the global k-means algorithm. Moreover, the cluster insertion procedure can be efficiently implemented by storing in a matrix all pairwise squared distances between points when the algorithm starts, and using this matrix for directly computing the upper bounds above. A similar 'trick' has been used in the related problems of greedy mixture density estimation using the EM algorithm [4] and principal curve fitting [5].

Finally, we may still apply this method as well as the global k-means algorithm when we do not consider every data point x_n $(n = 1, \ldots, N)$ as possible insertion position for the new center, but use only a smaller set of appropriately selected insertion positions. A fast and sensible choice for selecting such a set of positions based on k-d trees is discussed next.

3.2 Initialization with k-d trees

A k-d tree [6, 7] is a multi-dimensional generalization of the standard one-dimensional binary search tree, that facilitates storage and search over k-dimensional data sets. A k-d tree defines a recursive partitioning of the data space into disjoint subspaces. Each node of the tree defines a subspace of the original data space and, consequently, a subset containing the data points residing in this subspace. Each nonterminal node has two successors, each of them associated with one of the two subspaces obtained from the partitioning of the parent space using a cutting hyperplane. The k-d tree structure was originally used for speeding up distance-based search operations like nearest neighbors queries, range queries, etc.

In our case we use a variation of the original k-d tree proposed in [7]. There, the cutting hyperplane is defined as the plane that is perpendicular to the direction of the principal component of the data points corresponding to each node, therefore the algorithm can be regarded as a method for *nested* (recursive) principal component analysis of the data set. The recursion usually terminates if a terminal node (called bucket) is created containing less than a prespecified number of points b (called bucket size) or if a prespecified number of buckets have been created. It turns out that, even if the algorithm is not used for nearest neighbor queries, merely the construction of the tree provides a very good preliminary clustering of the data set. The idea is to use the bucket centers (which are fewer than the data points) as possible insertion locations for the algorithms presented previously.

In Figure 1 average performance results are shown on 10 data sets each one consisting of 300 data points drawn from the same mixture of 15 Gaussian components. The components of the Gaussian mixture are well separated and exhibit limited eccentricity.

We compare the results of three methods to the clustering problem with $k = 15$ centers: (i) The dashed line depicts the results when using the fast global k-means algorithm with all data points constituting potential insertion locations. The average clustering error over the 10 data sets is 15.7 with standard deviation 1.2. (ii) The solid line depicts results when the standard k-means algorithm is used: one run for each data set was conducted. At each run the 15 cluster centers were initially positioned to the centroids of the buckets obtained from the application of the k-d tree algorithm until 15 buckets were created. The average clustering error over the 10 data sets is 24.4 with standard deviation 9.8. (iii) The solid line (with error bars indicating the standard deviation from the mean value) shows the results when using the fast global k-means algorithm with the potential insertion locations constrained by the centroids of the buckets of a k-d tree. On the horizontal axis we vary the number of buckets for the k-d tree of the last method.

We also computed the 'theoretical' clustering error for each data set, ie., the error computed by using the true cluster centers. The average error value over the 10 data sets was 14.9 with standard deviation 1.3. These results were too close to the results of the standard fast global k-means to include them in the figure.

We can conclude from this experiment that (a) the fast global k-means approach gives rise to performance significantly better than when starting with all centers at the same time initialized using the k-d tree method, and (b) restricting the insertion locations for the fast global k-means to those given by the k-d tree (instead of using all data points) does not significantly degrade performance if we consider a sufficiently large number of buckets in the k-d tree (in general larger than the number clusters).

Obviously, it is also possible to employ the above presented k-d tree approach with the global k-means algorithm.

4 Experimental results

We have tested the proposed clustering algorithms on several well-known data sets, namely the iris data set [8], the synthetic data set [9] and the image segmentation data set [8]. In all data sets we conducted experiments for the clustering problems obtained by considering only feature vectors and ignoring class labels. The iris data set contains 150 four-dimensional data points, the synthetic data set 250 two-dimensional data points and the for the image segmentation data set we consider 210 six-dimensional data points obtained through PCA on the original 18-dimensional data points. The quality of the obtained solutions was evaluated in terms of the values of the final clustering error.

For each data set we conducted the following experiments:

- one run of the global k-means algorithm for $M = 15$.
- one run of the fast global k-means algorithm for $M = 15$.
- the k-means algorithm for $k = 1, \ldots, 15$. For each value of k, the k-means algorithm was executed N times (where N is the number of data points) starting from random initial positions for the k centers, and we computed the minimum and average clustering error as well as its standard deviation.

For each of the three data sets the experimental results are displayed in Figures 2, 3 and 4 respectively. Each figure plot displays the clustering error value as a function of the number of clusters. It is clear that the global k-means algorithm is very effective providing in all cases solutions of equal or better quality with respect to the k-means algorithm. In terms of the fast version of the algorithm, it is very encouraging that, although executing significantly faster, it provides solutions of excellent quality, comparable to those obtained by the original method. Therefore, it constitutes a very efficient algorithm, both in terms of solution quality and computational complexity and can run much faster if k-d trees are employed as explained in the previous section.

In the following subsections we provide extensive experimental results comparing the fast versions of the algorithm to the conventional k-means algorithm with random initialization.

4.1 Texture segmentation

In this clustering experiment the objective is to cluster 16×16 pixel image patches extracted from a set of 37 Brodatz texture images [10]. Each complete texture image consists of 256×256 pixels and 500 patches per texture were extracted by randomly selecting 16×16 windows. It is expected that patches originating from the same texture image to form an individual cluster (or clusters).

For every number of textures $k = 2, \ldots, 6$, we randomly constructed 100 data sets. Each data set was created by first randomly selecting k of the 37 textures and then selecting 200 patches for each texture, resulting in $200k$ patches per data set. All reported results are averages over the 100 data sets.

We compared performance of the three algorithms: (1) k-means initialized using a uniformly selected random subset of the data, (2) fast global k-means, (3) fast global k-means with the insertion locations limited to the *top* $2k$ *nodes* of the corresponding kd-tree. To evaluate the different methods, we considered the mean squared clustering error (MSE) of the patches to their closest mean. In a first experiment the number clusters was considered equal to the number of textures selected to create a data set. The results are given in Figure 5, where we also provide the corresponding execution times (in seconds). Bold values in the Tables indicate the method with best performance in terms of the clustering error.

We also conducted a second series of experiments where we used twice as many clusters as textures. The results are displayed in Figure 6.

It can be observed that (on average) the k-means algorithm with random initialization gives the worst results for almost every number of clusters and textures. It is also interesting to note that the fast global k-means that uses the top $2k$ nodes of the kd-tree as insertion candidates is not only much faster than the generic fast global k-means algorithm, but it also provides slightly better results in terms of the clustering error.

4.2 Artificial data

In this subsection we provide more extensive comparative experimental results using artificially created data sets. The purpose is to compare the randomly initialized k-means algorithm with the fast global k-means algorithm that uses a the top $2k$ nodes of the corresponding kd-tree as candidate insertion locations.

The data have been drawn from randomly generated Gaussian mixtures [11] and we varied the number of sources (mixture components) k, the dimensionality d of the data space and the 'separation' c between the sources of the mixture. Following [11], the separation c of a Gaussian mixture satisfies:

$$\forall_{i \neq j}: \quad \| \mu_i - \mu_j \|^2 \geq c \cdot \max_{\{i,j\}} \{\text{trace}(\mathbf{C}_i), \text{trace}(\mathbf{C}_j)\}, \tag{4}$$

where the μ's and $\mathbf{C}$'s denote respectively the means and covariance matrices of the mixture components. In our experiments we considered mixtures having a separation in the range between 0.5 and 1 and these values correspond exclusively to weakly separated clusters. The number of data points in each data set was $50k$, where k is the number of sources. Some example data sets are shown in Figure 7.

We have considered 120 problems corresponding to any of the following combination of values: $k = \{2, 7, 12, 17, 22\}$, $d = \{2, 4, 6, 8\}$, $c = \{0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. For each problem 10 data sets were created. The results are displayed in Figures 8 and 9.

On every data set, the 'greedy' (fast global k-means with k-d tree initialization) algorithm was applied first. Then the randomly initialized k-means algorithm was applied as many times as possible in the run time of the greedy algorithm. To evaluate the quality of the solutions found for a specific data set, first the mean clustering error μ is computed for the runs performed using the k-means algorithm. The minimum clustering error value corresponding to the runs with the k-means algorithm is provided in the 'min.' columns as $(1 - \min/\mu) \times 100$, and the standard deviation in the σ column as $\sigma/\mu \times 100$. The 'gr.' column provides the clustering error E_{gr} obtained with the fast global k-means algorithm in the form: $(1 - E_{gr}/\mu) \times 100$. Finally, the 'trials' column indicates how many runs of the k-means algorithm were allowed in the run time of the greedy algorithm. It must be noted that each row of the table displays the averaged results over 10 data sets constructed for the specific values of k, d and c. We used bold values in the Tables to indicate whether the clustering error of the greedy algorithm or the minimum error achieved for the runs with the k-means algorithm was the smallest.

It is clear from the experiments that the benefit of the greedy method becomes larger if there are more clusters, the seperation becomes larger and the dimensionality gets smaller. It is striking that in almost all cases the greedy algorithm gives better results. In cases where the greedy method is not superior, both methods yield results relatively close to the average k-means result.

One thing to note is that the clusters in the created data sets are poorly seperable. It is also interesting to note that the number of trials allowed for the random k-means algorithm grows only slowly as the number of clusters increases. From top to bottom, although the number of clusters has increased by a factor 10 the number of trials however has not even doubled on average. The inspection of the number of trials as a function of the k, d and c indicates dependence on the number of clusters only. As Figure 10 indicates, both the average number of trials as well as the corresponding variance increase as the number of clusters increases.

Matlab implementations of the fast global k-means and the k-d tree building algorithms can be downloaded from `http://www.science.uva.nl/research/ias`.

5 Discussion and conclusions

We have presented the global k-means clustering algorithm, which constitutes a deterministic clustering method providing excellent results in terms of the clustering error criterion. The method is independent of any starting conditions and compares favorably to the k-means algorithm with multiple random restarts. The deterministic nature of the method is particularly important in cases where the clustering method is used either to specify initial parameter values for other methods (for example RBF training) or constitutes a module in a more complex system. In such a case we can be almost certain that the employment of the global k-means (or any of the fast variants) will always provide sensible clustering solutions. Therefore, one can evaluate the complex system and adjust critical system parameters without having to worry for dependence of system performance on the clustering method employed.

Another advantage of the proposed technique is that in order to solve the M-clustering problem, all intermediate k-clustering problems are also solved for $k = 1, \ldots, M$. This may prove useful in many applications where we seek for the actual number of clusters and the k-clustering problem is solved for several values of k. We have also developed the fast global k-means algorithm, which significantly reduces the required computational effort, while at the same time providing solutions of almost the same quality.

We have also proposed two modification of the method that reduce the computational load without significantly affecting solution quality. These methods can be employed to find solutions to clustering problems with thousands of high-dimensional points and one of our primary aims is to test the techniques on large scale data mining problems.

Another direction of future work is related with the use of parallel processing for accelerating the proposed methods since, for every k, the N executions of the k-means algorithm are independent and can be performed in parallel. Another research direction concerns the application of the proposed method to other types of clustering (for example fuzzy clustering), as well as to topographic methods like SOM. Moreover, an important issue that deserves further study is related with the possible development of theoretical foundations for the assumptions behind the method. Finally, it is also possible to employ the global k-means algorithm as a method for providing effective initial parameter values for RBF networks and data modeling problems using Gaussian mixture models and compare the effectiveness of the obtained solutions with other training techniques for Gaussian mixture models [12, 13].

Acknowledgements

N. Vlassis and J. J. Verbeek are supported by the Dutch Technology Foundation STW project AIF 4997.

References

[1] M. N. M. A. K. Jain and P. J. Flynn, "Data clustering: A review," *ACM Computing Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

[2] J. A. L. J. M. Pena and P. Larranaga, "An empirical comparison of four initialization methods for the k-means algorithm," *Pattern Recognition Letters*, vol. 20, pp. 1027–1040, 1999.

[3] G. W. Milligan and M. C. Cooper, "An examination of procedures for determining the number of clusters in a data set," *Psychometrika*, vol. 50, pp. 159–179, 1985.

[4] N. Vlassis and A. Likas, "A greedy EM algorithm for Gaussian mixture learning," *Neural Processing Letters*, vol. 15, pp. 77–87, 2002.

[5] J. Verbeek, N. Vlassis, and B. Kröse, "A k-segments algorithm to find principal curves," tech. rep., Computer Science Institute, University of Amsterdam, The Netherlands, Nov. 2000. IAS-UVA-00-11.

[6] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[7] R. F. Sproull, "Refinements to nearest-neighbor searching in k-dimensional trees," *Algorithmica*, vol. 6, pp. 579–589, 1991.

[8] C. L. Blake and C. J. Merz, "UCI repository of machine learning databases." University of California, Irvine, Dept. of Information and Computer Sciences, 1998.

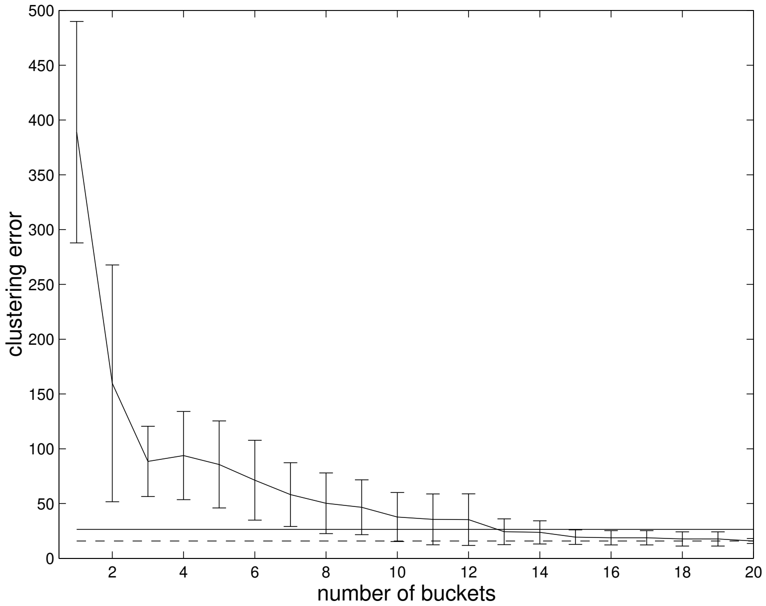
[9] B. D. Ripley, *Pattern Recognition and Neural Networks.* Cambridge, U.K.: Cambridge University Press, 1996.

[10] P. Brodatz, *Textures: A photographic album for artists and designers.* New York, U.S.A.: Dover, 1966.

[11] S. Dasgupta, "Learning mixtures of Gaussians," in *Proc. IEEE Symp. on Foundations of Computer Science*, (New York), Oct. 1999.

[12] G. J. McLachlan and D. Peel, *Finite Mixture Models.* New York: Wiley, 2000.

[13] N. Vlassis and A. Likas, "A kurtosis-based dynamic approach to Gaussian mixture modeling," *IEEE Trans. Systems, Man, and Cybernetics, Part A*, vol. 29, pp. 393–399, July 1999.



500
450
400
350
300
250
200
150
100
50
0
clustering error
2
4
6
8
10
12
14
16
18
20
number of buckets

Figure 1:

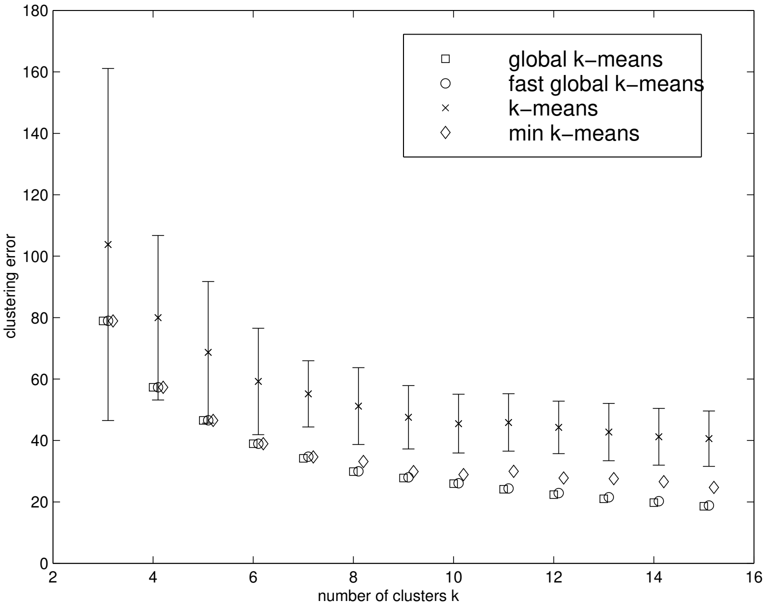
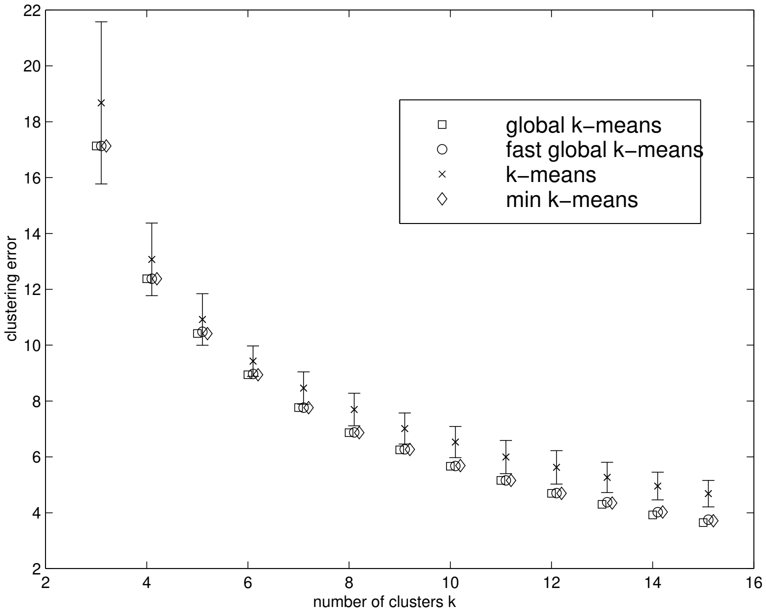
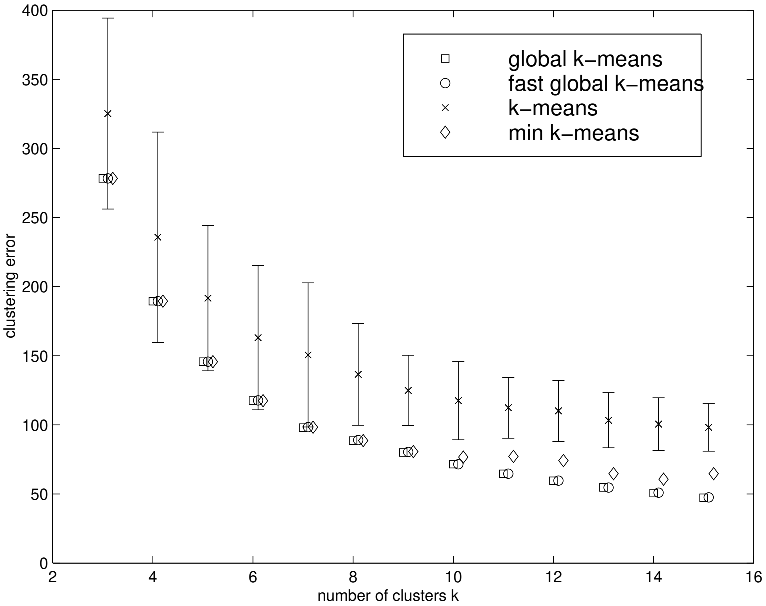

global k−means
fast global k−means
k−means
min k−means
clustering error
number of clusters k
0
20
40
60
80
100
120
140
160
180
2
4
6
8
10
12
14
16

Figure 2:



global k−means
fast global k−means
k−means
min k−means
clustering error
number of clusters k
22
20
18
16
14
12
10
8
6
4
2
2
4
6
8
10
12
14
16

Figure 3:



global k−means
fast global k−means
k−means
min k−means
clustering error
number of clusters k
0
50
100
150
200
250
300
350
400
2
4
6
8
10
12
14
16

Figure 4:

MSE:

method, # clusters	2	3	4	5	6
random k-means	4.01	3.55	3.58	3.06	2.98
fast global	3.98	3.49	3.26	2.94	2.91
fast global, kd	**3.87**	**3.40**	**3.21**	**2.88**	**2.85**

Run time:

method, # clusters	2	3	4	5	6
random k-means	0.03	0.08	0.16	0.21	0.35
fast global	0.21	0.63	1.37	2.5	4.12
fast global, kd	0.06	0.18	0.34	0.54	0.81

Figure 5:

MSE:

method, # clusters	4	6	8	10	12
random	3.63	3.23	3.10	2.80	2.78
fast global	3.74	3.22	3.06	2.79	2.80
fast global, kd	**3.51**	**3.07**	**2.91**	**2.70**	**2.69**

Run time:

method, # clusters	4	6	8	10	12
random k-means	0.073	0.14	0.22	0.32	0.44
fast global	0.38	1.15	2.44	4.31	7.15
fast global, kd	0.20	0.42	0.71	0.97	1.45

Figure 6:

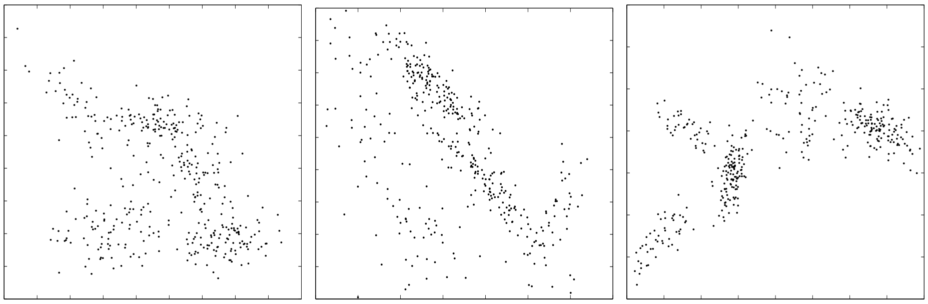


Figure 7:

		$d=2$				$d=4$			
k	sep.	gr.	min.	σ	trials	gr.	min.	σ	trials
2	0.5	0.8	**1.0**	1.9	4.3	1.6	**2.2**	3.0	3.4
2	0.6	1.1	**1.4**	1.7	2.7	0.4	**0.7**	1.0	2.9
2	0.7	-0.8	**0.1**	0.1	3.5	-0.6	**0.2**	0.2	2.7
2	0.8	0.2	**0.5**	0.8	2.9	0.5	0.5	0.9	2.9
2	0.9	0.3	**0.5**	1.0	3.4	1.2	**2.1**	4.2	3.0
2	1.0	-0.5	**0.3**	0.4	3.9	0.5	**0.7**	1.4	3.2
7	0.5	**2.7**	2.1	2.4	3.6	2.3	**2.7**	3.3	3.9
7	0.6	**16.9**	14.7	17.0	4.3	1.7	**2.3**	2.6	4.1
7	0.7	**3.8**	3.7	4.4	3.4	5.3	5.3	6.6	4.1
7	0.8	**11.2**	9.9	13.9	4.2	**4.9**	4.7	6.5	4.3
7	0.9	**17.1**	15.9	18.2	4.5	**6.6**	6.3	7.4	3.7
7	1.0	**9.9**	8.1	8.6	4.4	**7.6**	7.2	9.6	4.1
12	0.5	**8.5**	5.9	6.5	4.3	**3.3**	2.4	2.4	4.4
12	0.6	**14.5**	12.2	10.3	4.5	**4.1**	2.8	2.4	4.3
12	0.7	**15.7**	13.4	15.4	4.5	**12.7**	7.1	7.4	4.4
12	0.8	**25.1**	15.4	21.3	4.6	**11.1**	8.5	7.5	4.9
12	0.9	**20.6**	14.2	19.0	4.6	**16.4**	12.3	13.3	4.6
12	1.0	**24.9**	18.1	19.4	4.1	**20.8**	15.8	14.9	5.8
17	0.5	**16.4**	11.1	11.4	4.1	**5.3**	3.6	3.9	4.3
17	0.6	**17.5**	12.6	14.4	4.1	**4.4**	3.6	3.5	4.2
17	0.7	**25.4**	18.7	18.8	4.9	**8.5**	6.3	6.6	4.4
17	0.8	**24.2**	17.6	14.3	5.0	**10.7**	6.3	6.5	4.9
17	0.9	**46.9**	29.0	27.3	5.4	**16.8**	10.7	11.4	5.0
17	1.0	**53.2**	37.2	49.0	5.8	**25.0**	19.1	17.2	5.6
22	0.5	**23.8**	18.5	24.4	4.7	**4.8**	3.5	3.3	4.6
22	0.6	**23.9**	14.3	19.6	5.1	**10.0**	6.3	5.3	4.9
22	0.7	**29.1**	18.6	20.0	5.1	**12.5**	8.9	9.6	5.1
22	0.8	**41.6**	29.0	28.8	5.2	**8.5**	5.1	4.9	5.2
22	0.9	**42.4**	30.5	28.3	5.9	**23.0**	14.4	13.6	5.2
22	1.0	**47.7**	34.7	36.4	5.8	**26.3**	16.7	15.9	5.3

Figure 8:

		$d = 6$				$d = 8$			
k	sep.	gr.	min.	σ	trials	gr.	min.	σ	time
2	0.5	0.5	**0.9**	1.2	3.2	0.1	**0.6**	0.9	2.7
2	0.6	0.4	**0.5**	0.8	3.3	1.5	1.5	2.5	3.2
2	0.7	**0.9**	0.4	0.7	2.5	2.2	**2.6**	3.7	2.6
2	0.8	0.5	**0.6**	0.8	3.0	1.8	1.8	3.0	3.7
2	0.9	0.3	0.3	0.6	3.1	-0.0	**0.1**	0.1	2.6
2	1.0	0.8	**0.9**	1.8	5.1	0.7	**1.1**	2.1	3.3
7	0.5	2.3	**2.4**	2.3	3.8	**3.4**	3.1	3.1	3.9
7	0.6	**4.4**	3.5	3.9	4.4	**2.9**	2.3	2.4	3.5
7	0.7	**4.9**	3.9	4.4	3.9	**3.8**	3.3	4.1	4.1
7	0.8	**5.7**	5.5	7.3	4.6	**5.0**	4.6	4.9	5.2
7	0.9	**4.6**	4.3	4.5	4.3	6.9	**7.3**	6.8	4.3
7	1.0	**6.9**	7.3	8.2	4.7	**9.2**	8.6	7.8	4.9
12	0.5	**3.6**	2.9	3.2	4.1	**2.5**	1.6	1.9	3.8
12	0.6	**4.8**	3.0	2.8	4.0	**5.7**	3.1	2.9	4.4
12	0.7	**5.1**	2.7	2.8	4.3	**4.5**	2.7	3.1	4.1
12	0.8	**5.3**	4.2	4.7	4.4	**5.7**	4.8	4.7	5.0
12	0.9	**8.6**	6.7	7.2	4.2	**10.0**	7.5	6.7	5.1
12	1.0	**12.3**	8.1	7.2	5.3	**9.2**	6.2	6.3	5.5
17	0.6	**5.0**	2.8	2.6	3.9	**4.9**	3.8	3.2	4.2
17	0.7	**5.4**	3.2	3.2	4.9	**6.3**	3.4	3.1	5.1
17	0.8	**8.8**	5.2	5.3	5.0	**8.4**	5.6	5.5	5.1
17	0.9	**11.6**	7.0	6.0	5.2	**14.4**	8.8	9.5	5.6
17	1.0	**22.2**	12.6	10.1	5.9	**15.9**	8.7	7.9	5.6
22	0.5	**2.8**	1.8	1.6	4.7	**2.2**	1.2	1.1	4.3
22	0.6	**4.8**	2.5	2.2	4.7	**3.0**	2.0	1.9	4.8
22	0.7	**9.3**	4.6	5.1	4.9	**9.3**	6.2	5.1	5.5
22	0.8	**13.0**	7.4	6.4	5.5	**11.5**	7.2	6.2	5.3
22	0.9	**14.7**	7.1	6.7	5.4	**19.5**	11.2	10.0	6.1
22	1.0	**21.6**	12.6	10.4	5.8	**16.2**	9.4	8.0	5.3

Figure 9:

k	2	7	12	17	22
mean	4.22	5.01	5.28	5.34	6.01
var.	1.63	2.34	1.79	2.09	2.12

Figure 10:

Figure 1: Performance results for data drawn from a Gaussian mixture with 15 components.

Figure 2: Performance results for the Iris data set.

Figure 3: Performance results for the Synthetic data set.

Figure 4: Performance results for the Image Segmentation data set.

Figure 5: Results for the Texture Segmentation problem using as many clusters as textures.

Figure 6: Results for the Texture Segmentation problem using twice as many clusters as textures.

Figure 7: Some example 2D data sets with five sources and separation (from left to right) 0.5, 0.75 and 1.

Figure 8: Experimental results on artificial data sets with d=2 and d=4.

Figure 9: Experimental results on artificial data sets with d=6 and d=8.

Figure 10: The number of allowed trials for the randomly initialized k-means as a function of the number of clusters.

Aristidis Likas received the Diploma degree in electrical engineering and the Ph.D. degree in electrical and computer engineering both from the National Technical University of Athens. Since 1996, he has been with the Department of Computer Science, University of Ioannina, Greece, where he is currently an assistant professor. His research interests include neural networks, pattern recognition, machine learning and automated diganosis.

Nikos Vlassis received the M.Sc. degree in electrical and computer engineering (1993) and the Ph.D. degree in artificial intelligence (1998) from the National Technical University of Athens, Greece. In 1998 he joined the Intelligent Autonomous Systems Group at the University of Amsterdam, The Netherlands, where he is now holding an assistant professor position. In 1999 he was a visiting researcher at the Electrotechnical Laboratory (currently AIST), Japan, with a scholarship from the Japan Industrial Technology Association MITI. He holds the Dimitris N. Chorafas Foundation prize (Luzern, Switzerland) for young researchers in Engineering and Technology. His research focusses on probabilistic and statistical learning methods for intelligent systems.

Jakob J. Verbeek (1975) received in 1998 a cum laude Masters degree in Artificial Intelligence and a cum laude Master of Logic degree in 2000, both at the University of Amsterdam, the Netherlands. Since September 2000 he is a PhD student in the Advanced School for Computing and Imaging, working in the Intelligent Autonomous Systems group at the University of Amsterdam. Current research activities take place in a project on nonlinear feature extraction. Research interests include principal curves and surfaces, non parametric density estimation and machine learning in general.

The global k-means clustering algorithm

Aristidis Likas, Nikos Vlassis and Jakob J. Verbeek

Summary

In this paper the *global k-means algorithm* is presented which is an *incremental approach to clustering* that dynamically adds one cluster center at a time through a deterministic global search procedure consisting of N (with N being the size of the data set) executions of the k-means algorithm from suitable initial positions. The basic idea underlying the proposed method is that an optimal solution for a clustering problem with M clusters can be obtained using a series of local searches (using the k-means algorithm). At each local search the $M-1$ cluster centers are always initially placed at their optimal positions corresponding to the clustering problem with $M-1$ clusters. The remaining M-th cluster center is initially placed at several positions within the data space. Since for $M=1$ the optimal solution is known, we can iteratively apply the above procedure to find optimal solutions for all k-clustering problems $k=1,\ldots,M$. The proposed method is deterministic, does not depend on any initial positions for the cluster center and does not contain any empirically adjustable parameters, thus eliminating all problems characterizing the k-means algorithm and its stochastic extensions.

In addition we present modifications of the method to reduce the computational load without significantly affecting solution quality. The first modification called the *fast global k-means algorithm* defines a fast computed bound on the clustering error that is used instead of local searches. The second modification is related with the partitioning of the data space using a k-d tree structure, in order to reduce the number of examined insertion positions for a new cluster center.

The proposed clustering methods are tested on well-known data sets and they compare favorably to the k-means algorithm with random restarts. In addition, the modified versions lead to implementations that are very fast and exhibit almost equal performance.