On the Optimal Partitioning of Data with K-Means, Growing K-Means, Neural Gas, and Growing Neural Gas

M. Daszykowski, B. Walczak,* and D. L. Massart

ChemoAC, VUB, FABI, Laarbeeklaan 103, B-1090 Brussels, Belgium

Received May 27, 2002

In this paper, the performance of new clustering methods such as Neural Gas (NG) and Growing Neural Gas (GNG) is compared with the K-means method for real and simulated data sets. Moreover, a new algorithm called growing K-means, GK, is introduced as the alternative to Neural Gas and Growing Neural Gas. It has small input requirements and is conceptually very simple. The GK leads to nearly optimal values of the cost function, and, contrary to K-means, it is independent of the initial data set partition. The incremental property of GK additionally helps to estimate the number of "natural" clusters in data, i.e., the well-separated groups of objects in the data space.

INTRODUCTION

The studies of the data structure have always inspired chemometricians in the same way as experiments inspire chemists. Since the data structure is a source of essential information, during the past few years many new techniques to analyze it have been developed. Not surprisingly, the early proposed approaches, due to their conceptual simplicity, have gained more attention and popularity than the later ones. The fundamental assumption of the data structure studies is that objects with similar physicochemical properties form groups (classes, clusters) in n-dimensional property space. These groups can be revealed by clustering techniques. There are two main groups of clustering techniques, namely hierarchical and nonhierarchical ones. Depending on the problem at hand, they can be applied for exploratory analysis, i.e., to find "natural" clusters, or to partition the data set into a desired number of groups.

For exploratory data analysis, hierarchical methods are often preferred, because they do not require any assumptions about the data distribution and have good visualization properties. The result of clustering is given in the form of a tree or a dendrogram. The dendrogram allows one to gain a fundamental knowledge about the data structure and at the same time to determine the density of clusters. The partitions can be constructed by agglomeration (merging) or dividing objects into groups. The partitioning is performed gradually, until all data objects are processed. Merging or splitting objects implies creation of the new branches in the dendrogram and is always based on the previous structures found. In other words, hierarchical methods do not assume repartitioning, because once linked (or disjoint) objects are never taken into account again. This fact is considered as the main drawback of the hierarchical methods.

The nonhierarchical methods partition the data set into a desired number of classes, no matter if the data reveal clustering tendency or not. The classical cost function, minimized during partitioning procedure is the sum of squared distances, E, between the data objects and the centers of classes

$$E = \sum_{j=1}^{k}\left(\sum_{i=1}^{m}(\mathbf{x}_i - \mathbf{w}_j)^2\right) \tag{1}$$

where k is the number of clusters, m is the number of experimental objects, $\mathbf{x}_i$ denotes the i*th* experimental object, and $\mathbf{w}_j$ is the j*th* cluster center.

The best known nonhierarchical method is K-means.[1−4] It should be stressed that sequential use of nonhierarchical methods, i.e., to generate {2, 3, ..., k} partitions, contains an element of hierarchy. By tracing the membership of objects when a new cluster is introduced, it is possible to judge the "natural" clustering tendency of the data. If nonhierarchical techniques are used in this way, they became an efficient tool for clustering. This approach, which contains hierarchical and nonhierarchical elements, was originally proposed by Massart et al.[5] However, the data partition with the K-means can be considered as sub-optimal only.

In this paper, we focus our attention on K-means,[1−4] Neural Gas (NG)[6,7] and Growing Neural Gas (GNG),[8] which can be applied for data set partition. Although NG and GNG offer new concepts, compared to K-means, they are much less known than K-means. All methods minimize the same cost function but NG and GNG can in some cases achieve better results. The advantages and shortcomings of each method will be presented, and eventually a new algorithm, growing K-means, which avoids some disadvantages of the other methods is proposed.

THEORY

K-Means. MacQueen's K-Means. The aim of K-means is to divide the data set into k clusters.[1−4] The number of clusters, k, is specified by the user. Objects that belong to the same cluster are more similar to each other than to objects in the other clusters. The similarity between objects is measured, for instance, by the Euclidean distance. The main steps of the K-means according to MacQueen's algorithm[1] can be summarized as follows:

1. select the number of clusters, k;
2. partition randomly data objects into k-clusters;

* Corresponding author phone: +32-2-477-4737; fax: + 32-2-477-4735; e-mail: beata@fabi.vub.ac.be. On leave from Silesian University, 9 Szkolna Street, 40−006 Katowice, Poland.

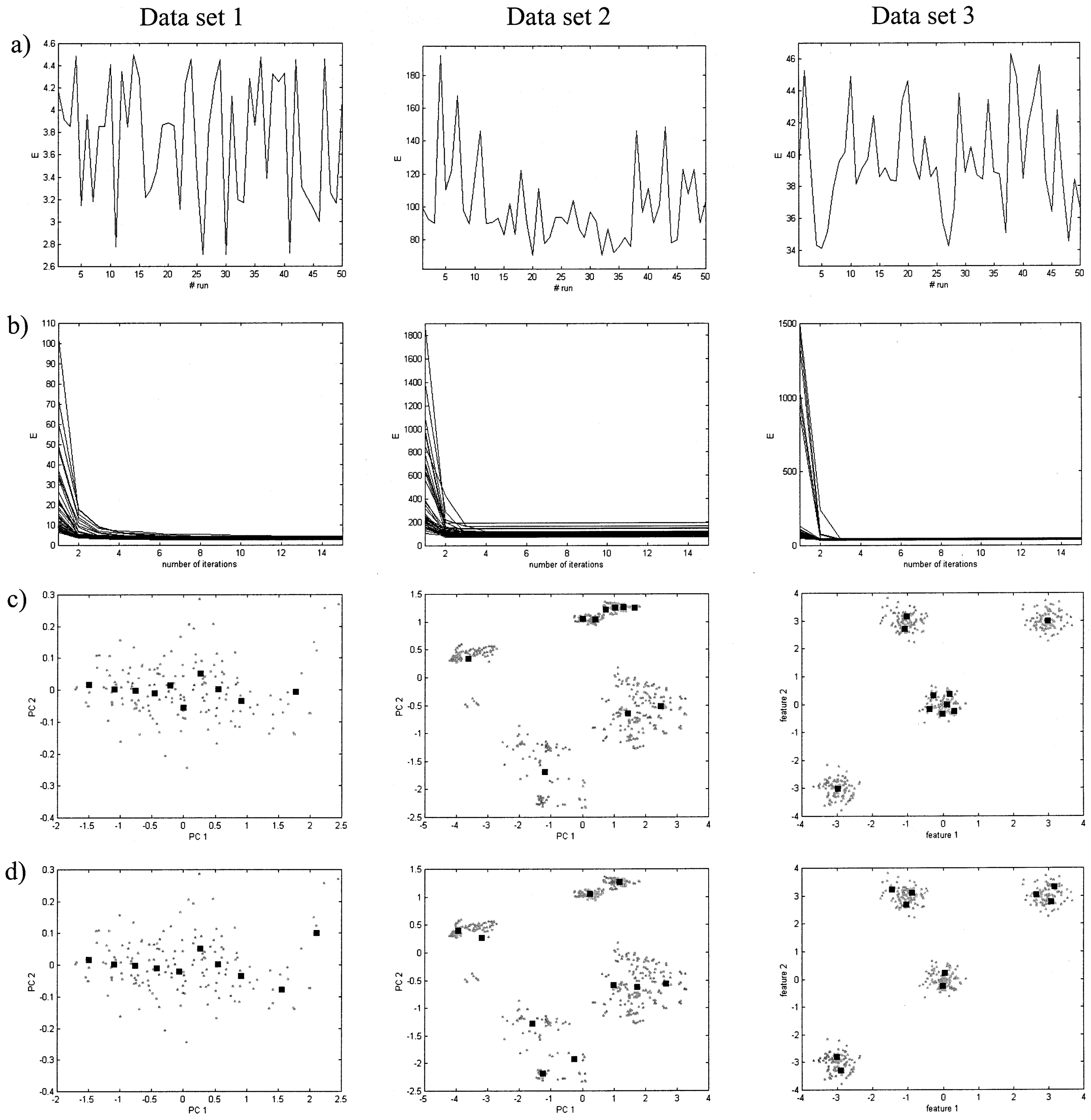

Figure 1. The performance of MacQueen's K-means for data sets 1, 2 and 3; (a) oscillations of the cost function, (b) cost function value, E, as a function of the number of iterations, (c) the partition with the highest value of the cost function, and (d) the partition with the smallest value of the cost function.

3. calculate the center of each cluster, $\mathbf{w}_j$;

4. assign each data object to the closest cluster center and calculate the sum of distances between the center of each cluster and the data objects belonging to it, E, according to eq 1;

5. repeat steps 3 and 4 till E convergence.

Neural Gas (NG). The Neural Gas algorithm, NG, proposed by Martinez et al.,[6] is a neural network consisting of k nodes (cluster centers), which independently move over the data space while learning. The name, "Neural Gas", emphasizes the similarity between the movement of nodes in the data space and the movement of gas molecules in a closed container. The position of the j*th* node in the data space is defined by the weight vector, $\mathbf{w}_j(1, n)$, where n is the number of variables. During the algorithm's run, a randomly selected object $\mathbf{x}_i(1, n)$ is presented to the network, and the weights of the nodes are adjusted with respect to the distance from $\mathbf{x}_i$ (this is considered as one iteration of the algorithm). The largest modifications of weights are introduced for the closest node to $\mathbf{x}_i$ (a winner) and the smallest ones for the most distant node. A similar approach to the concept of neighborhood was proposed with Kohonen Self-Organizing Maps.[9−10] With the number of iterations the neighborhood function shrinks. This concept was applied to avoid random partition of the data set. During the learning stage, the weights of the nodes (i.e., the position of nodes)

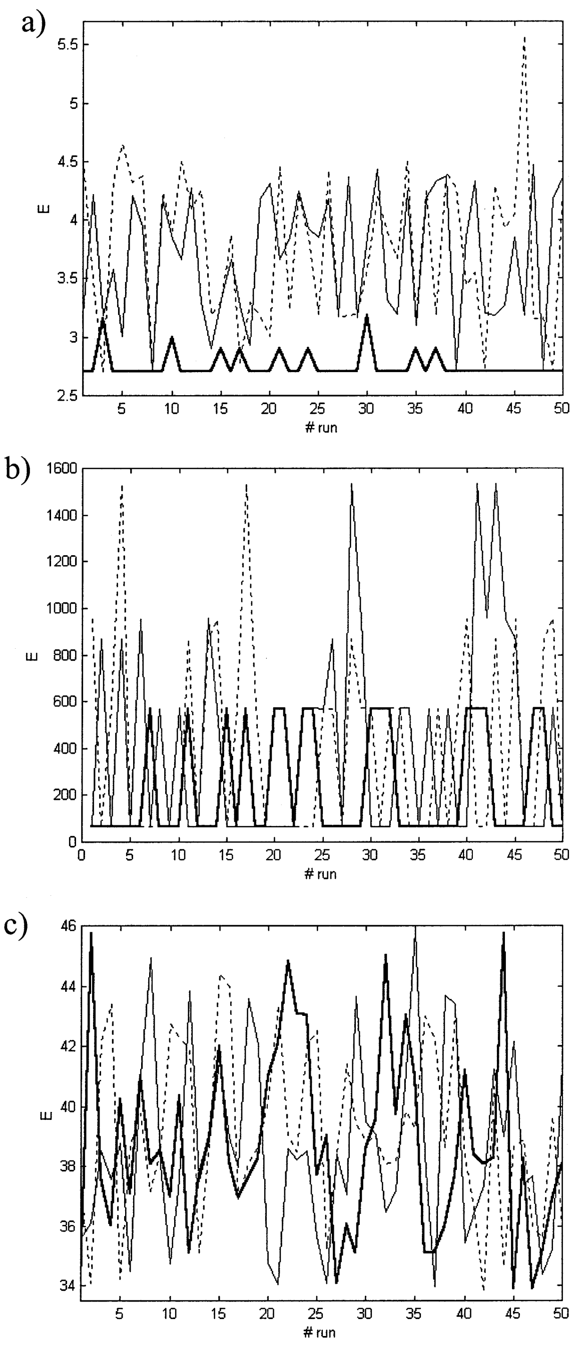

Figure 2. The results of online K-means (bold line), Forgy (dotted line), and MacQueen's K-means (line) for the following: (a) data set 1 ($k = 4$), (b) data set 3 ($k = 4$), (c) data set 3 ($k = 10$) (settings of online K-means: $t_{max} = 40000$, $lr_i = 0.5$, $lr_f = 0.005$).

are adjusted to minimize the cost function, described by eq 1, as in K-means clustering. The neighborhood concept allows more flexible assignment of the objects to nodes. If the neighborhood of the winner is limited only to itself (i.e. only a winner is updated), NG is equal to the online version of K-means. This happens when the value of the neighborhood size (decay constant) becomes small. Once the network is trained, the objects are assigned to the nearest nodes, which leads to final clustering. The algorithm of the NG can be presented as follows:[6]

Settings:

• select the number of nodes, k, equal to the number of clusters

• for each node j initialize the weight vector, $\mathbf{w}_j(1, n)$ by setting its elements to small random numbers

• set training parameters: maximal number of iterations, t_{max}, initial and final learning rate (lr_i, lr_f), initial and final decay constant, determining the neighborhood size, (d_i, d_f).

1. present to the network a randomly selected object, $\mathbf{x}_i$, from data set, $\mathbf{X}(m, n)$, where m denotes the number of objects and n the number of variables;

2. determine the distances between $\mathbf{x}_i$ and the weights, $\mathbf{w}_j$, of each node, where $j = 1, 2, ..., k$;

3. sort nodes according to their distance from the object $\mathbf{x}_i$, to find a winning node and its consecutive neighbors;

4. at each iteration t, adjust weights of a winner and its neighbors according to equation

$$\mathbf{w}_j(t + 1) = \mathbf{w}_j(t) + lr(t)\cdot h_d(ind_i)\cdot(\mathbf{x}_i - \mathbf{w}_j(t))$$

where t is the current iteration, $t = 1, 2, 3, ..., t_{max}$

$$\text{learning rate: } lr(t) = lr_i\left(\frac{lr_f}{lr_i}\right)^{t/t_{max}}, \; lr \in [0\ 1]$$

$$\text{neighborhood function: } h_d(ind_i) = e^{-(k_i/d)}$$

$$\text{decay constant: } d(t) = d_i\left(\frac{d_f}{d_i}\right)^{t/t_{max}}, \; d \in [0\ \infty)$$

if $t < t_{max}$, go to step 1;

5. assign each object of the data set to its closest node.

Growing Neural Gas (GNG). The Growing Neural Gas, GNG, is an incremental neural network. It can be described as a graph consisting of k nodes, each of which has an associated weight vector, $\mathbf{w}_j$, defining the node's position in the data space and a set of edges between the node and its neighbors.[8] During learning, new nodes are introduced into the network till a maximal number of nodes is reached. The GNG starts with two nodes, randomly localized in the data space, connected by an edge. Adaptation of weights, i.e., the nodes' position, is performed iteratively. For each data object the closest node (the winner), s1, and the closest neighbor of a winner, node s2, are determined. These two nodes are connected by an edge. With each edge an age variable is associated. At each learning step (presentation of a randomly selected object to the network), the ages of all edges emanating from the winner are increased by 1. When the edge between s1 and s2 is created or it already exists, its age is set to 0. By tracing the changes of the age variable, it is possible to detect nodes that are not active, it means they are not winners. Edges exceeding a maximal age, α, and any nodes having no emanating edges are removed. In this way topology of the network is modified. In GNG, the neighborhood of the winner is limited to its topological neighbors, i.e., nodes connected by edges with a winner. The winner and its topological neighbors are moved in the data space toward the presented object by a constant fraction of the distance, defined separately for the winner and its topological neighbors. In GNG there is no neighborhood function, and no ranking concept. Thus, all topological neighbors are updated in the same way. Simultaneously, the squared distance to the input object is accumulated for the winner. This information is stored in so-called dispersion variable, DV, which represents the data dispersion around

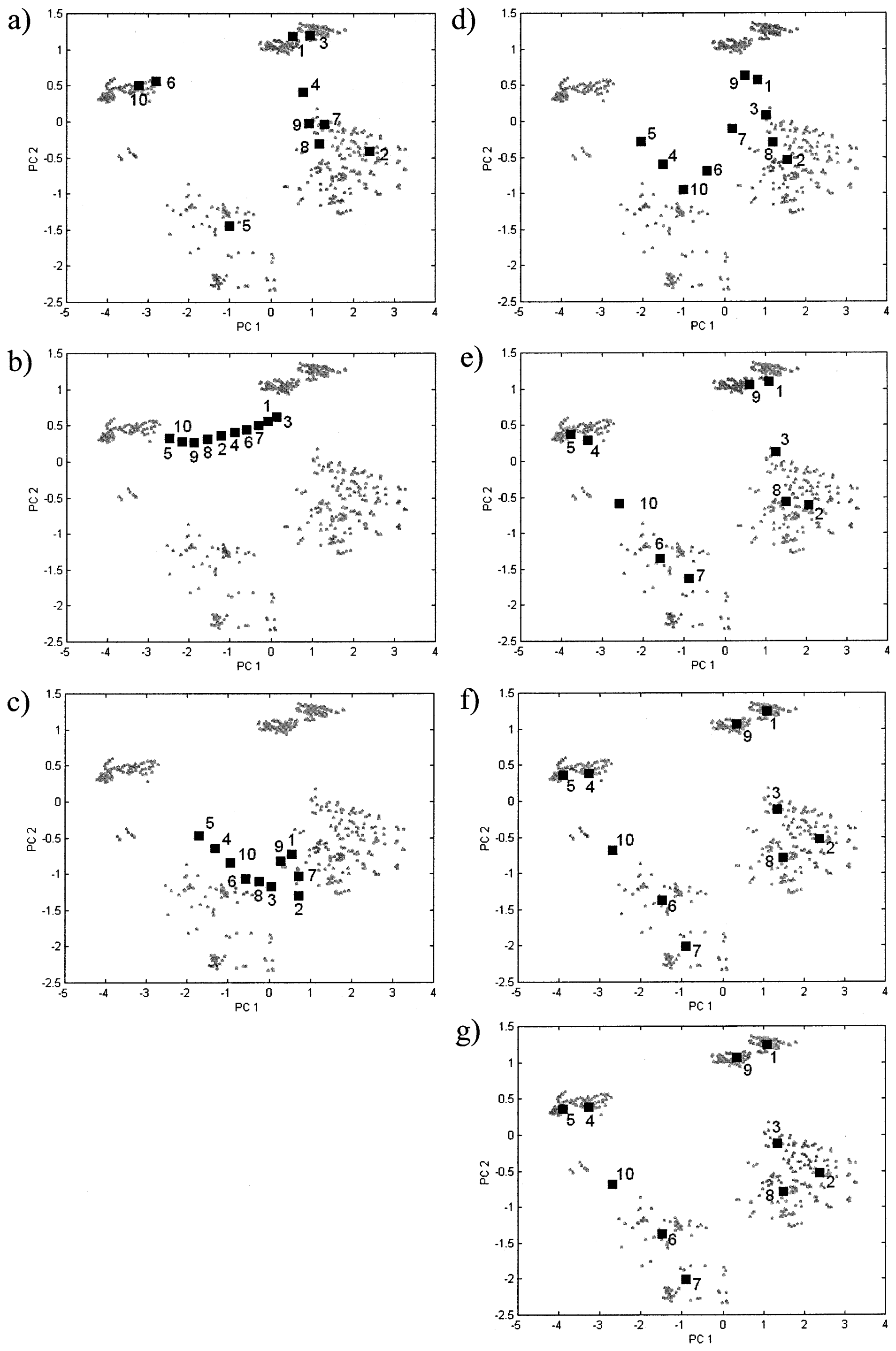

Figure 3. NG at work, run for data set 2, with the following input parameters. Results displayed after the following: (a) 0, (b) 1000, (c) 2000, (d) 5000, (e) 10000, (f) 15000, (g) 30000 iterations (settings: $k = 10$, $t_{max} = 40000$, $lr_i = 0.5$, $lr_f = 0.005$, $d_i = 10$, $d_f = 0.01$).

the nodes at certain learning stage. The dispersion variable is further used to find the optimal position of a new node.

A new node is introduced into a network after a predefined number of iterations, χ, defined by a user. It should be born in mind that the new node cannot be introduced too quickly into the network structure. It is important to give the network the time to redistribute nodes over the data space. The number of iterations after a new node is inserted, χ, plays a similar role to maximal age, defining the rate of the network's modifications. The position of the new node is the arithmetic mean of the positions of nodes q and f, having maximal DV and connected by an edge. The original edge between q and f is replaced by two edges from a new node to q and to f. Once a new node is inserted, the DV of f and q are reduced by 50%, and the DV of a new node equals the mean of the reduced DV of q and f. As a stopping criterion, λ, a maximal number of nodes is used, but other criteria can be applied as well.[11]

The main steps of the GNG algorithm are as follows:[8]

1. start with a set A of two nodes a and b at random positions $\mathbf{w}_a$ and $\mathbf{w}_b$ in the data space

$$A = \{a,b\}$$

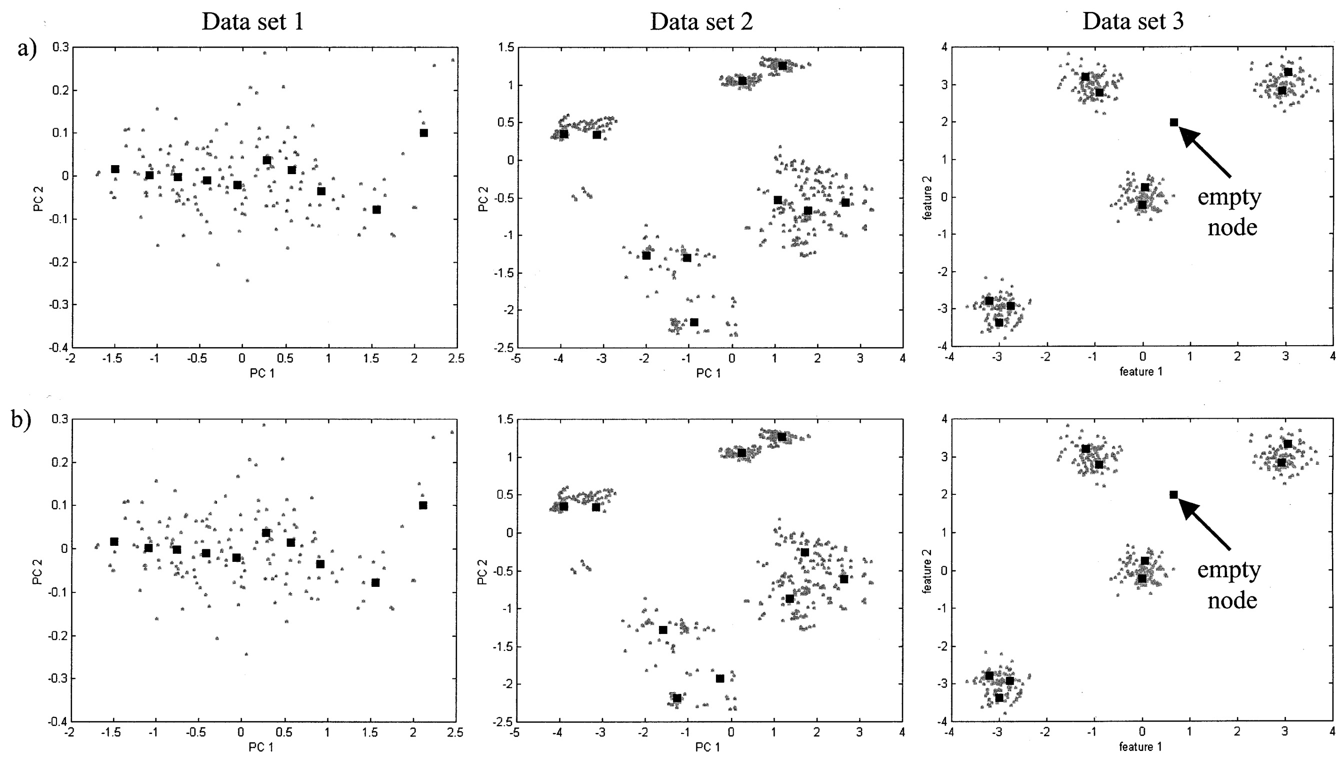

Figure 4. Results of NG obtained for data sets 1, 2 and 3, with the following settings: (a) $t_{max} = 40000$, $lr_i = 0.5$, $lr_f = 0.005$, $d_i = 10$, $d_f = 0.01$ and (b) $t_{max} = 40000$, $lr_i = 0.5$, $lr_f = 0.005$, $d_i = 10$ $d_f = 0.00001$.

2. initialize the set of edges, C, which contains an edge between a and b and set the age of this edge to zero:

$$C = \{(a,b)\};\ age_{(a,b)} \leftarrow 0$$

3. select randomly an object **x**

4. determine nodes s1 and s2 (s1, s2 ∈ A) such that

$$||\mathbf{w}_{s1} - \mathbf{x}|| \le ||\mathbf{w}_c - \mathbf{x}|| \text{ for all } c \in k$$

and

$$||\mathbf{w}_{s2} - \mathbf{x}|| \le ||\mathbf{w}_c - \mathbf{x}|| \text{ for all c except } s1 \in k$$

5. if the edge does not already exist, add an edge between s1 and s2 to C:

$$C = C \cup \{(s1,s2)\}$$

In any case set the age of the edge between s1 and s2 to zero

$$age_{(s1,s2)} \leftarrow 0$$

6. add the squared distance between the input signal and s1 to DV:

$$DV_{s1} = \Sigma||\mathbf{w}_{s1} - \mathbf{x}||^2$$

7. move s1 and s2 toward an object **x**:

$$\Delta\mathbf{w}_{s1} = lr_w\,(\mathbf{x} - \mathbf{w}_{s1})$$

$$\Delta\mathbf{w}_{s2} = lr_n\,(\mathbf{x} - \mathbf{w}_{s2})$$

8. increment the age of all edges emanating from s1

9. remove edges with age larger than α. If this results in nodes having no emanating edges, remove them as well;

10. after χ iterations, insert a new node as follows:

- determine the node q with the maximum DV and its topological neighbor, node f
- interpolate a new node r with nodes q and f, and remove the original edge between q and f:

$$C = C \cup \{(r,q), (r,f)\}$$

$$\text{remove } C\ \{(q,f)\}$$

- decrease the DV of q and f by multiplying them by constant e_w:

$$\Delta DV_q = -\,e_w DV_q$$

$$\Delta DV_f = -\,e_w DV_f$$

- interpolate the DV of r between q and f:

$$DV_r = 0.5(DV_q + DV_f)$$

11. decrease the DV of all nodes c ∈ k by multiplying them by constant e_n:

$$\Delta DV_c = -\,e_n DV_c$$

12. if the number of nodes is lower than λ, go to step 3.

Default values of parameters are as follows: $e_w = 0.5$, $e_n = 0.0005$, $\alpha = 30$, $\chi = 300$, $lr_w = 0.05$, $lr_n = 0.0006$.

Growing K-Means. Growing K-means is a new incremental network model with undefined topology, which is presented here for the first time. During the learning process, the nodes are redistributed over the data space to minimize the cost function, described by eq 1. The algorithm combines the desirable properties of the previously described methods such as, decreasing learning rate from NG, incremental character of GNG, and online presentation of data set objects

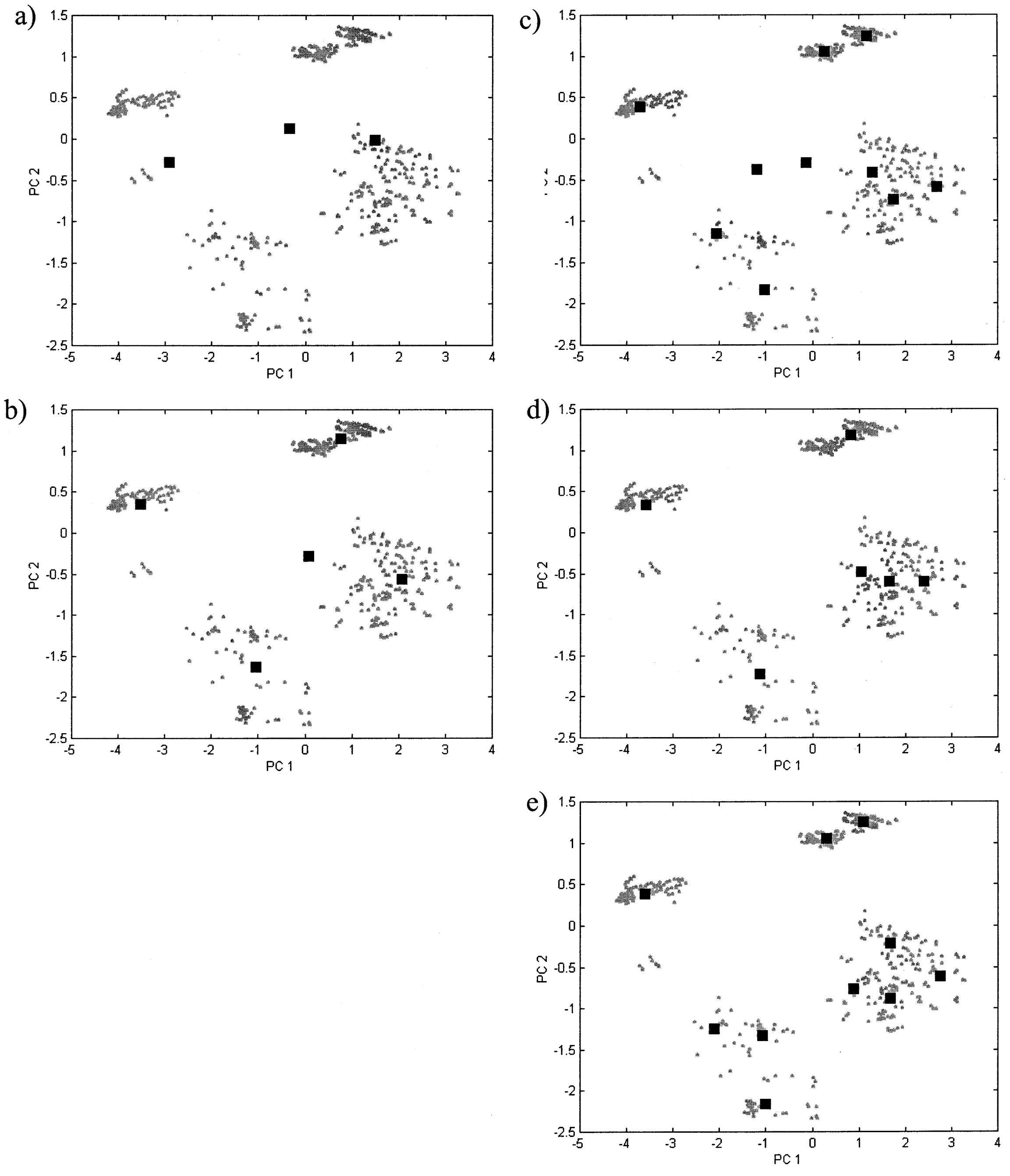

Figure 5. GNG at work, run for data set 2. Results displayed after the following: (a) 400, (b) 1000, (c) 2000, (d) 5000, (e) 10000, and (f) 12000 iterations (settings: $k = 10$, $\alpha = 10$, $\chi = 300$, $lr_w = 0.05$, $lr_n = 0.0006$, $e_n = 0.5$, $e_w = 0.0005$, $t_{max} = 40000$).

(the position of the nodes is updated after presenting an object to the network). The procedure starts with two nodes, randomly localized in the experimental space. During learning, the node closest to the presented object, is moved toward it by a constant fraction of the distance (so-called learning rate), defined by a user. Once the whole data set has been presented to the net, the objects are assigned to the closest nodes and for each node the sum of squared distances to all objects belonging to it is calculated. A new node is inserted at half the distance between the node with the largest within group variance, DV, and the furthest object belonging to this node. The network grows until the maximal number of nodes, k, is introduced into the network, and then the learning rate is decreased with the number of iterations left, as in NG. The learning of the network is terminated once k nodes are inserted and a predefined number of iterations, t_{max}, is reached. The main steps of the algorithm can be summarized as follows:

(1) specify the maximal number of nodes (clusters), k, the initial and final learning rates;

(2) start with two nodes randomly localized in the data space;

(3) present a randomly selected data set object, $\mathbf{x}_i$, to the network;

(4) determine the closest node (a winner) to the presented object;

(5) move the winner toward the presented object by a fraction (learning rate) of a Euclidean distance to it;

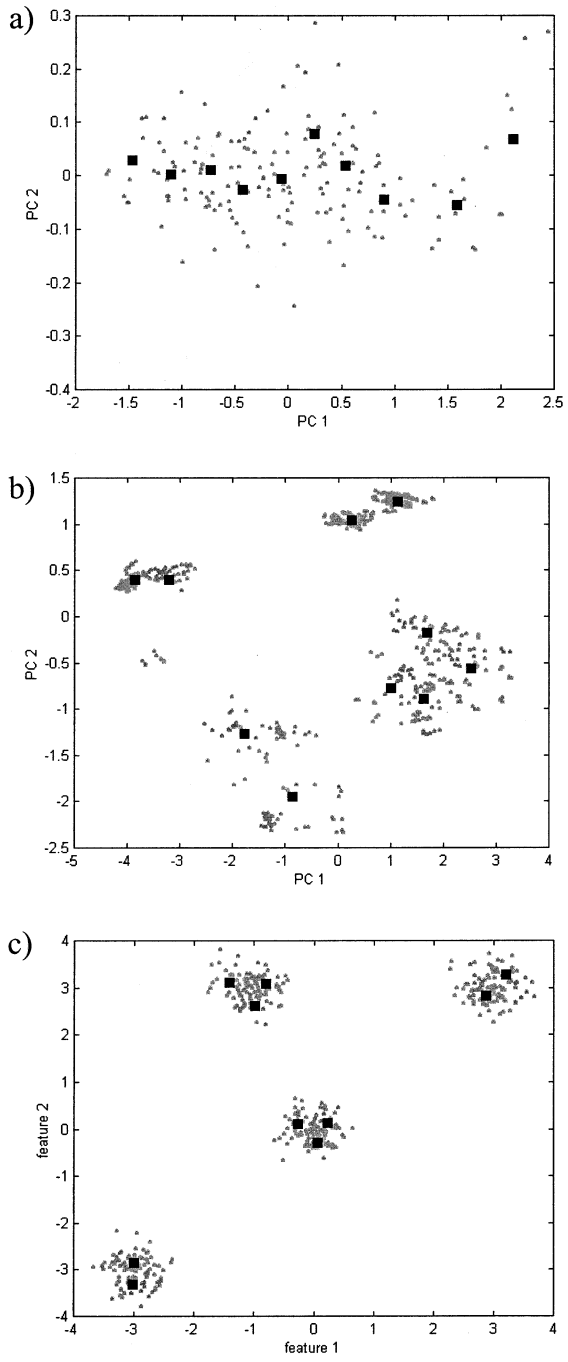

Figure 6. The results of GNG for k = 10, t_{max} = 40000, e_w = 0.5, e_n = 0.0005: (a) data set 1: $\alpha = 100$, $\chi = 160$, $lr_w = 0.05$, $lr_n = 0.0006$, (b) data set 2: $\alpha = 300$, $\chi = 400$, $lr_w = 0.05$, $lr_n = 0.0006$, and (c) data set 3: $\alpha = 300$, $\chi = 400$, $lr_w = 0.05$, $lr_n = 0.0006$.

(6) when the whole data has been presented to the network, assign each object to its nearest node and add the squared distance between an object and its closest node to dispersion variable of nodes, DV;

(7) insert a new node at half the distance between the node with the highest DV and the furthest object belonging to this node;

(8) if k nodes are introduced, decrease the learning rate to conform with the number of the remaining iterations.

DATA

Data set 1 consists of 160 NIR spectra, measured for 80 corn samples with two different spectrometers. The spectra were collected at 2 nm intervals within the spectral range 1100−2498 nm. The data set is available via Internet.[12] With PCA the data set was compressed to two significant principal components, PCs, describing over 99% of the data variance. These two PCs were used for further studies.

Data set 2 contains 536 NIR spectra of three creams with three different concentrations of an active drug. These spectra were measured in the range 1100−2500 nm at two different temperatures (20 and 30 °C) and two ways of cup filling ("bad" and "good").[13] After PCA two significant PCs explaining over 99% of the data variance were used as an input to clustering methods.

Data set 3 is a simulated 2-dimensional data set. It consists of 200 objects, grouped into four classes.

RESULTS AND DISCUSSION

The K-means method is usually applied for partitioning data into k clusters. MacQueen's K-means seems to be the most popular. It is known that K-means suffers from the initial random selection of cluster centers, which has a crucial impact upon the final results.[1] Moreover, the convergence to an optimal solution of the cost function is not guaranteed.[1] In other words, when K-means is rerun, it may provide different values of the cost function. The oscillations of the cost function values for data sets 1, 2, and 3 are presented in Figure 1a. The algorithm was run 50 times to find 10 clusters.

The partitions corresponding to the highest and the lowest values of the cost function are presented in Figure 1 c and d, respectively. The results for data set 3 in Figure 1c can be regarded as far from optimal, since six out of 10 clusters centers are situated in one "natural" cluster. One possible solution to overcome unstable results of K-means is to run the algorithm several times and to select the partition with the smallest value of the cost function. As shown in Figure 1b, after a few iterations, the value of the cost function converges. It is an advantage of this algorithm, and it can be rerun many times without a serious computational effort.

There are other algorithms of K-means clustering as well, for instance, online K-means and Forgy's K-means.[1−4] In Figure 2 the values of the cost function for each algorithm, run 50 times, for data sets 1 and 3 are presented. Figure 2 shows that MacQueen's and Forgy's K-means perform similarly. Both algorithms assign data objects to the nearest cluster center, found in the previous iteration of the algorithm, and then redefine the cluster centers. This step can lead to new clusters, if their centers are different from those found in the previous iteration. A cluster center is defined by a group of objects, whereas in the online version of K-means, the clusters center (nodes) are updated at each iteration by moving them toward a presented data object by a constant fraction of a distance (learning rate). Online K-means, for nonclustered data, often reaches low values of the cost function, due to its dynamic character, i.e., after presentation of a randomly selected object the nearest cluster center is moved toward the object (see Figure 2 a). Decreasing the learning rate with the iterations provides near-optimal solutions of the cost function. Lower fluctuations of the cost function values than those observed for MacQueen's and Forgy's K-means are obtained for k equal to the number of "natural" clusters (see Figure 2b). Otherwise, as shown in

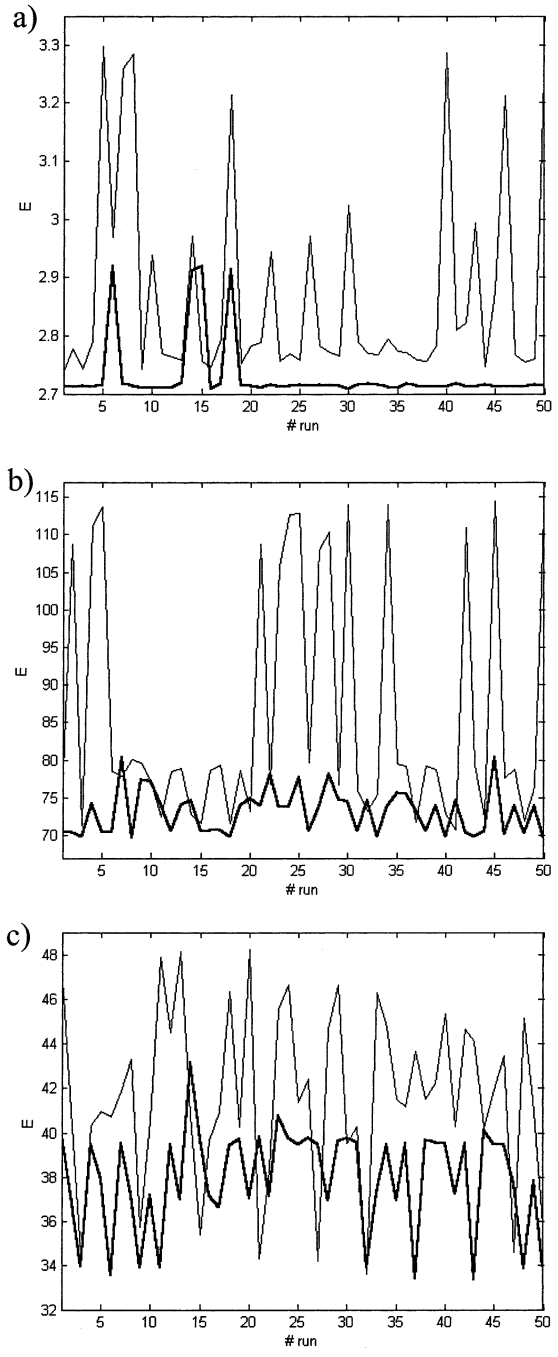

Figure 7. Oscillations of the cost function for GNG (line) and GNG* (solid line) run for data sets 1, 2 and 3, with k = 10, t_{max} = 40000, e_w = 0.5, e_n = 0.0005 and (a) α = 100, χ = 160, lr_w = 0.05, lr_n = 0.0006, (b) α = 300, χ = 400, lr_w = 0.05, lr_n = 0.0006, and (c) α = 300, χ = 400, lr_w = 0.05, lr_n = 0.0006.

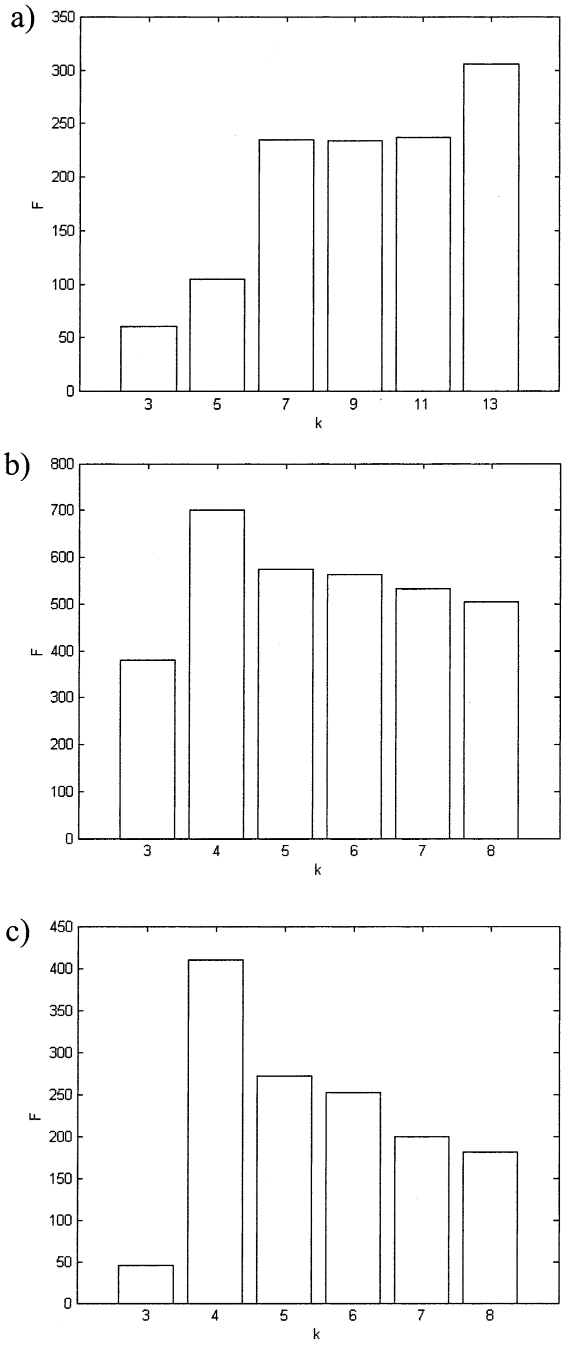

Figure 8. Values of within to between-group variance, F, for different k: (a) data set 1, (b) data set 2, and (c) data set 3.

Figure 2c, there is no difference between the discussed K-means algorithms. Although the results of online K-means are more promising for nonclustered data, similar results are achieved with the other two algorithms, rerun several times (see Figure 2a-c). The two other algorithms have lower computational time than online K-means. As shown in Figures 1 and 2, the performance of K-means depends on to the data distribution and the number of clusters to be selected. Usually the value of the cost function is higher than the optimal one.

To overcome the drawbacks of K-means such as the initial random partition and unstable results, the Neural Gas (NG) was proposed. The main concept in NG is the ranked neighborhood of the winner (the closest node to the presented data object). The consecutive subplots of Figure 3 show the performance of NG for data set 2. During 5000 iterations, nodes are moved over the experimental space very rapidly (compare the positions of the nodes in Figure 3a-d). Because the neighborhood function shrinks with the number of iterations, fewer nodes are updated together with the winner and the nodes start to split more and more (compare Figure 3b-c). After 10000 iterations, the nodes oscillate only around the optimal solutions (see Figure 3e-g) due to the small value of learning rate and shrunken neighborhood around the winner. Further training does not change the position of the nodes (see Figure 3g). An online demo of NG, which can be run for other types of data distribution and with different settings, can be found on the Internet.[14]

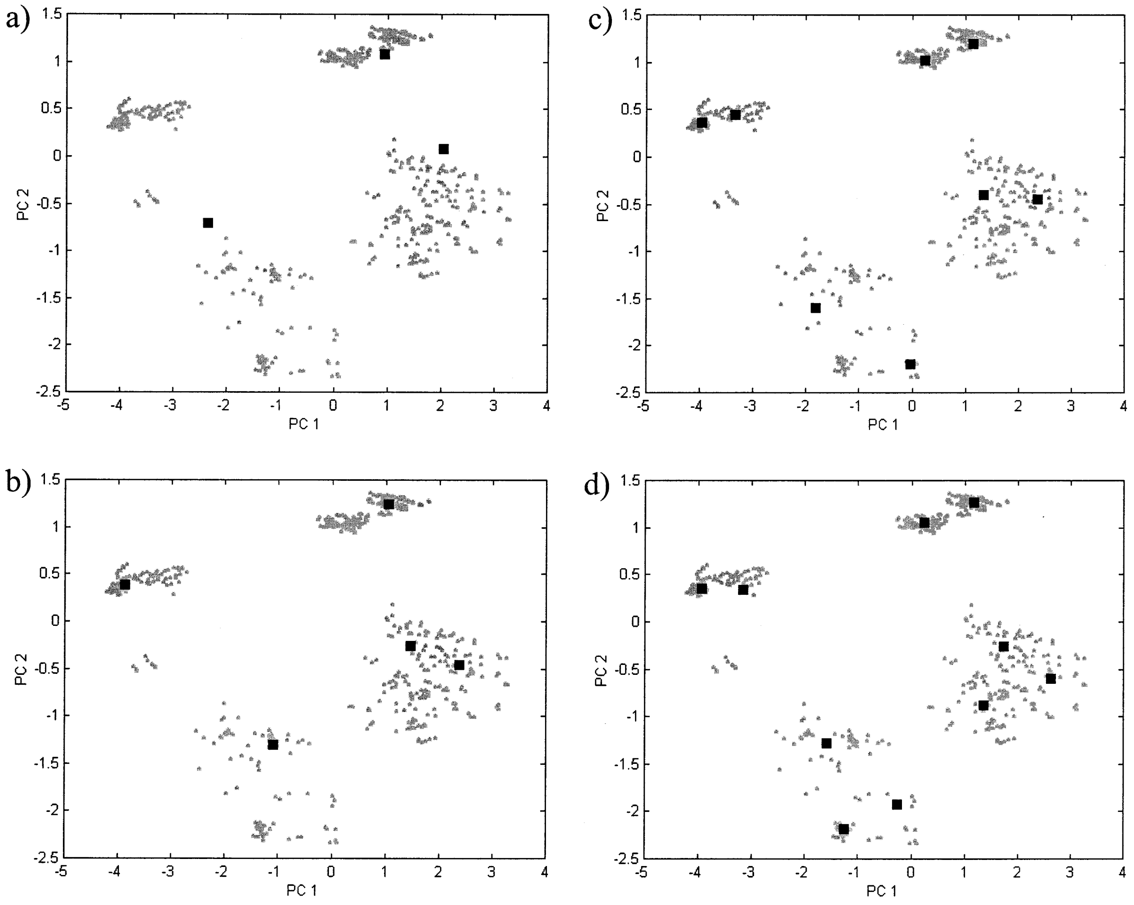

Figure 9. GK run for data set 2. Results displayed after insertion of the following: (a) 3rd node, (b) 5th node, (c) 8th node, and (d) final position of nodes (settings: k = 10, t_{max} = 40000, lr_i = 0.5, lr_f = 0.005).

Unfortunately, NG, contrary to any K-means algorithm, can provide empty nodes, i.e., no objects are assigned to them. Therefore, the desired number of partitions is not always obtained. For some types of data distributions, the values of the cost function are usually smaller than those obtained with K-means algorithms e.g. ref 6. For nonclustered data or data with similar clusters (i.e., similar distribution and number of objects in each cluster) NG performs very well. When the number of nodes exceeds the number of natural clusters, it can happen that nodes are not properly distributed and/or there are empty nodes. For certain types of distributions it can be impossible to find a nearly optimal value of the cost function, even when NG is run with various input parameters (see Figure 4a,b). The number of iterations, final learning rate and decay constant (a neighborhood size) have a serious influence on the results. The data set should be presented at least several hundred times to the network. The more iterations, the better, because nodes can be redistributed over the data space more efficiently. For low final value of learning rate, better final approximation of cluster centers can be observed, than for larger one. The rate at which the neighborhood function shrinks can also have a large impact upon the final results. NG was tested intensively for many types of data distributions and in wide ranges of input parameters. In this paper, the discussion is based on three types of data distributions, frequently observed in chemical problems. The results of NG run with different settings for data sets 1, 2 and 3 are presented in Figure 4.

For nonclustered data, such as data set 1, the input parameters do not influence significantly the results, and in particular, the partitions given in Figure 4a,b have the same value of the cost function, equal to 2.71. For data set 2, modification of input parameters can fundamentally improve results. The value of the cost function for the partition of Figure 4a equals 72.27, whereas for partition of Figure 4b, equals 69.59. However, sometimes, as for data set 3, even when input parameters are changed in a wide range, an empty node is present (see Figure 4a,b). For data set 3, lower value of the cost function, 33.31, can be obtained with K-means, whereas using NG it equals 37.64 (compare Figures 1d and 4a,b).

The Growing Neural Gas (GNG), contrary to NG, is a neural network with well-defined topology, the nodes being connected by edges. The topology of the network has a dynamic character, and while training, new nodes are introduced into the network and new connections (edges) between nodes are constructed or removed if necessary. Figure 5 illustrates the dynamic character of the GNG network.

Before a new node is inserted into the network, existing nodes are moved over the data space toward near-optimal partitions. An online demonstration of GNG, for different types of data distribution, can be found on the Internet.[15]

The results of GNG for data sets 1, 2 and 3 are summarized in Figure 6. The distribution of the cluster centers properly covers the data distribution. However the values of the cost function for data sets 1, 2 and 3 are higher than for partitions obtained with K-means and NG (compare Figures 1, 4 and 6) and equal 2.81, 72.71 and 34.77, respectively.

In the original GNG, the learning rate is constant during learning of the network. In our study, the original GNG was

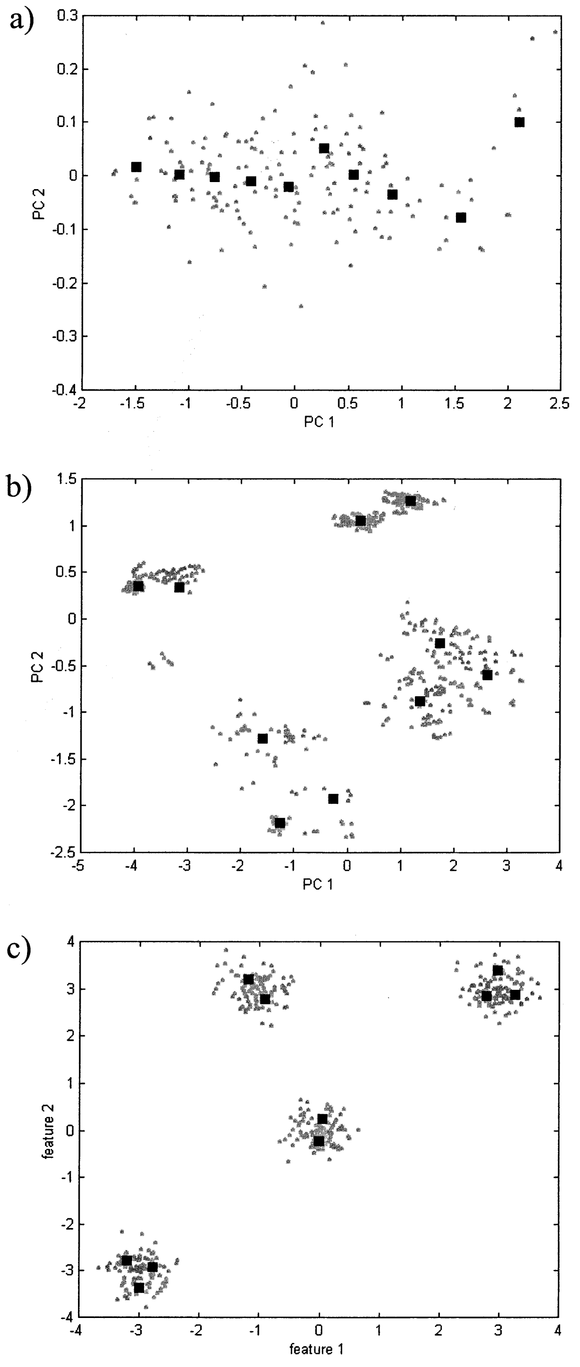

Figure 10. Results of GK, obtained for the following: (a) data set 1, (b) data set 2, and (c) data set 3 (settings: $k = 10$, $t_{max} = 40000$, $lr_i = 0.5$, $lr_f = 0.005$).

modified by introducing decreasing learning rate with the number of iterations. This modified version of GNG, abbreviated GNG*, is further used in a comparison algorithms presented in the paper. It is shown that GNG* allows more frequently reaching lower values of the cost function. If the desired number of nodes is reached, the modification of the net topology is stopped, and the nodes are redistributed over the data space. The comparison between the classical GNG (line) and a modified version, GNG* (a solid line), is presented in Figure 7. For GNG and GNG* the optimal setting (i.e., settings leading to small value of the cost function) were found by trial and error. GNG and GNG* were run 50 times, with the optimal settings, for data sets 1, 2 and 3. In both cases 40000 iterations were performed. In most cases GNG* gives better results than GNG (see Figure 7).

The input parameters, used in GNG, such as the age of edges, learning rates and DV regularization (see theory part) are not easy to estimate. Unfortunately, no rules are known how to estimate them, and their selection is based on the user's intuition or on trial and error.

GNG and Natural Clusters. Nonhierarchical methods, used sequentially, can be applied to detect the number of natural clusters in data. However, with GNG it can be achieved during one run of the algorithm. This is due to the fact that GNG has some hierarchical characteristics. A new node is always inserted in such a way that the cost function is minimized. Because the new node is always inserted between two nodes, this is similar to the creation of a new branch in hierarchical methods. Insertion of a new node in the data space implies another data set partition; therefore, GNG creates a set of partitions {2, 3, ..., k}, where k is the maximal number of nodes. In this respect, GNG combines properties of hierarchical and nonhierarchical methods. Usually, the network grows till a specified number of nodes is introduced but the growth process can be controlled to reveal more "natural" clusters. This is an advantage of the method. One possible criterion, which helps to approximate the number of clusters in the data (optimal number of nodes) is tracing the changes of F value (between-group variance to within-group variance), calculated according to eq 2, for different k[1]

$$F = \frac{\sum_{i=1}^{k} m_i B_i}{\sum_{i=1}^{k} W_i} \cdot \left(\frac{m-k}{k-1}\right) \qquad (2)$$

where B_i is the sum of squared distances between the objects of the i*th* cluster and its center, W_i is the squared distance between the center of the i*th* cluster and the data center, m_i is the number of objects in the i*th* cluster, k is the number of nodes (clusters), and m is the number of objects in a data set.

If F reaches a maximum for certain k, the difference between cluster means is large and it suggests the presence of k clusters. This concept is not specific for GNG, but it can be applied to all nonhierarchical methods for which a set of partitions is created. Once the optimal number of clusters is estimated, the insertion of new nodes and network topology modifications can be stopped. The existing nodes are redistributed over the data space in order to minimize the cost function, and then GNG turns into online K-means. Figure 8 shows the changes of F, used to estimate the optimal number of nodes in a GNG run. The results are obtained using optimal settings of GNG for data sets 1, 2 and 3, described in the text earlier.

For nonclustered data sets, such as data set 1 (see Figure 6a), F usually reaches a maximum for high values of k (see Figure 8a). In this example, F reaches maximum for $k = 13$, which is considered to be larger than the acceptable number of clusters. For clustered data, e.g. data set 2 and 3, we assume that there are less than 10 clusters. The maximum

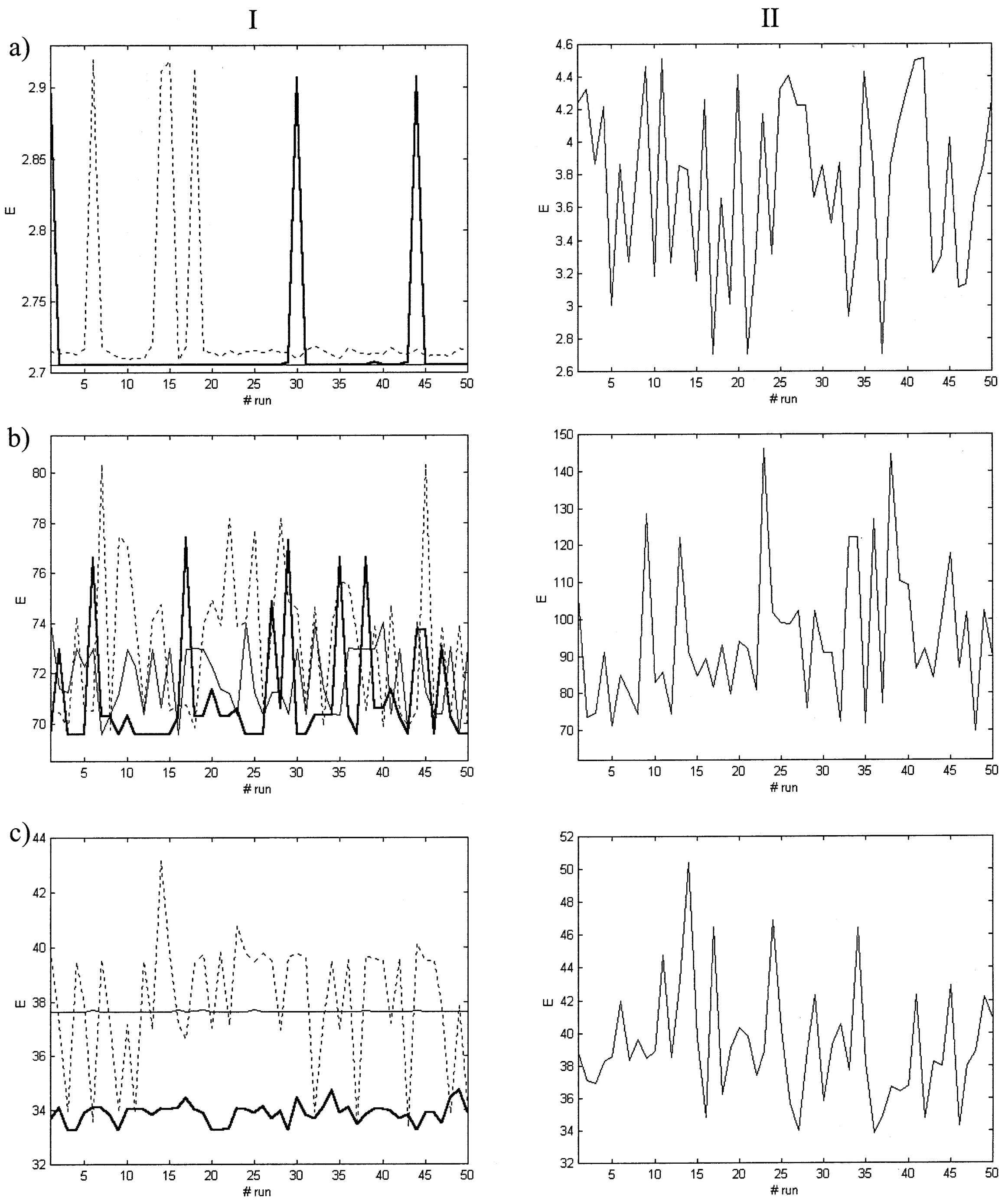

Figure 11. (I) comparison of GK (bold line), NG (line), modified GNG (dotted line) illustrated on the following: (a) data set 1, (b) data set 2, (c) data set 3; (II) oscillations of the cost function for K-means run with k = 10 for the same data sets.

of the F is observed for k = 4, i.e., for four clusters (see Figure 8b,c), which is a correct conclusion (see Figure 6b,c).

Growing K-Means. Growing K-means combines the advantages of algorithms discussed earlier such as online performance (updating the position of the node after each presentation of a randomly selected data object) and incremental character. During learning, the learning rate decreases with the number of iterations, using the same rule as in NG to reach the final value after a desired number of iterations. This guarantees solutions close to the optimal one. In practice, the algorithm requires only one input parameter, i.e., the number of iterations or a convergence criterion of weight changes. It is usually sufficient to present the data set to the network several hundred times to achieve good results. The initial learning rate can be analytically estimated based on the distance to the nearest neighbor in the simulated data set uniformly distributed within the range of experimental variables, e.g. ref 16. Intuitively, the final learning rate should be small, for instance 1000 times smaller than the initial learning rate. The performance of the method is illustrated on the data set 2. For comparison purposes, the initial learning rate and the final learning rate have the same values as used in NG and GNG. The new nodes are introduced into the network after a whole data set is presented, till a maximal number of nodes is not reached. The nodes are put in the optimal positions, i.e., to minimize the cost function as in GNG. Figure 9 shows the incremental character of the net, and more specifically, the position of nodes after insertion the 3rd, 5th, and 8th. The final position of 10 nodes after training is given in Figure 9d.

The results obtained with GK for data sets 1, 2, and 3 are presented in Figure 10 a-c, respectively. For data sets 2 and 3 the results are better than those obtained with NG, GNG and K-means for k = 10. The value of the cost function for data set 2 is 69.56 whereas for data set 3, it is 31.11.

Comparison of K-Means, NG, GNG, and GK. Each algorithm was run 50 times with the optimal settings found for the corresponding data sets, to select 10 clusters. The results are shown in Figure 11. In the first column of Figure 11 the NG, GNG and GK are compared, whereas in the second column, corresponding results of K-means are given.

Figure 11, part II clearly show that the results of K-means are not stable. The variations of the cost function values are larger than for the other approaches. GK can provide good results, better than the other methods, i.e., and more frequently leads to low values of the cost function. Only in the first case (see Figure 11, part Ia), NG performs better than GK. Although the level of the cost function reached by NG can be higher than the optimal one, the oscillations of the cost function values for each run are very small. This can be regarded as a good property of the method. For nonclustered data, NG outperforms the other methods. The GNG* shows higher oscillations of the cost function values than GK and NG but performs better than K-means (compare Figure 11, parts I-II).

CONCLUSIONS

Data set partition can be performed by all presented here methods. K-means, although it often leads to a partition far from the optimal one, is the fastest method. Thus, it can be repeated several times, and the partition with the smallest value of the cost function can be stored as the optimal one. In most cases NG provides better partitions than those obtained by K-means, because it is independent of the initial data set partition. The most practical feature of NG is that the oscillations of the cost functions are small. On the other hand, it can happen that the empty nodes are present and different input settings do not improve results. The incremental character of GNG makes the algorithm independent of the initial partition and gives the possibility to verify the number of "natural" clusters in the data. The GK, with respect to required input parameters, conceptual simplicity and small oscillations close to optimal solution of the cost function, seems to be a very elegant alternative to GNG, NG and K-means. It can perform exactly the same tasks as GNG does but in a simpler and faster way. As a clustering technique, it allows estimating optimal number of clusters, i.e., reflecting more the "natural" clusters and it leads to partitions with values of the cost function close to optimal one. The fields of its applications seem to be exactly the same as for K-means, GNG or NG, i.e., for vector quantization, for subset selection, for construction of local models, for instance Radial Basis Function Networks, Neural-Fuzzy Systems, etc. Another application of these methods is cluster-based selection.[17] Once the centers of clusters are found, the closest objects to the centers can be considered as the most representative ones.

REFERENCES AND NOTES

(1) Vogt, W.; Nagel, D.; Sator, H. Cluster Analysis in Clinical Chemistry; A Model, John Wiley & Sons: New York, 1987.
(2) Lloyd, S. P.; Least squares quantization in PCM. *IEEE Trans. Inform. Theory* IT-28 **1982**, 2.
(3) Massart, D. L.; Kaufman, L. The Interpretation of Analytical Chemical data by the use of Cluster Analysis, John Wiley & Sons: New York, 1983.
(4) Kaufman, L.; Rousseeuw, P. J. Finding Groups in Data. An Introduction to Cluster Analysis, Wiley: New York, 1990.
(5) D. L. Massart, L. Kaufman, Hierarchical nonhierarchical clustering strategy and application to classification of iron meteorites according to their trace element patterns. *Anal. Chem.* **1982**, *54*, 911−917.
(6) Martinez, T. M.; Berkovich, S. G.; Schulten, K. J. "Neural-Gas" network for vector quantization and its applications to time-series prediction. *IEEE Trans. Neural Networks* **1993**, *4*, 4, 558−569.
(7) Questier, F.; Guo, Q.; Walczak, B.; Massart, D. L.; Boucon, C.; de Jong, S. The Neural Gas network for classifying analytical data. *Chemom. Intel. Lab. Sys.* **2002**, *61*, 105−121.
(8) Fritzke, B. A Growing Neural Gas network learns topologies, Advances in Neural Information Proceedings Systems 7, MIT Press: Cambridge MA, 1995.
(9) Kohonen, T. Self-Organizing Maps, Springer-Verlag: Berlin, 1997.
(10) Zupan, J.; Gasteiger, J. Neural networks in chemistry and drug design, Wiley-Vch Verlag: Winheim, 1999.
(11) Fritzke, B. Be busy and unique ... or be history, the utility criterion for removing units in Self-Organizing networks, KI-99. *Adv. Artificial Intelligence* **1999**, *1701*, 207−218.
(12) http://www.eigenvector.com/Data/Corn/corn.mat.
(13) Luypaert, J.; Heuerding, S.; de Jong, S.; Massart, D. L. An evaluation of Direct Orthogonal Signal Correction and other preprocessing methods for the classification of clinical study lots of a dermatological cream. *J. Pharm. Biomed. Analysis* (in press).
(14) http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn/DemoGNG/NG_2.html.
(15) http://www.neuroinformatik.ruhr-unibochum.de/ini/VDM/research/gsn/DemoGNG/GNG.
(16) Daszykowski, M.; Walczak, B.; Massart, D. L. Looking for Natural Patterns in Analytical Data. 2. Tracing Local Density with OPTICS. *J. Chem. Inf. Comput. Sci.* **2002**, *42*, 500−507.
(17) Dunbar, J. B., Jr. Cluster based selection. *Perspectives Drug Discovery Design* **1997**, *7/8*, 51.

CI020270W