Generalized Tree-Like Self-Organizing Neural Networks with Dynamically Defined Neighborhood for Cluster Analysis

Marian B. Gorzałczany, Jakub Piekoszewski, and Filip Rudziński

Department of Electrical and Computer Engineering
Kielce University of Technology
Al. 1000-lecia P.P. 7, 25-314 Kielce, Poland
{m.b.gorzalczany,j.piekoszewski,f.rudzinski}@tu.kielce.pl

Abstract. The paper presents a generalization of self-organizing neural networks of spanning-tree-like structures and with dynamically defined neighborhood (SONNs with DDN, for short) for complex cluster-analysis problems. Our approach works in a fully-unsupervised way, i.e., it operates on unlabelled data and it does not require to predefine the number of clusters in a given data set. The generalized SONNs with DDN, in the course of learning, are able to disconnect their neuron structures into sub-structures and to reconnect some of them again as well as to adjust the overall number of neurons in the system. These features enable them to detect data clusters of virtually any shape and density including both volumetric ones and thin, shell-like ones. Moreover, the neurons in particular sub-networks create multi-point prototypes of the corresponding clusters. The operation of our approach has been tested using several diversified synthetic data sets and two benchmark data sets yielding very good results.

Keywords: generalized self-organizing neural networks with dynamically defined neighborhood, multi-point prototypes of clusters, cluster analysis, unsupervised learning.

1 Introduction

Data clustering or cluster analysis is an unsupervised process that aims at grouping unlabelled data records from a given data set into an unknown in advance number of cluster or groups. Elements of each cluster should be as "similar" as possible to each other and as "dissimilar" as possible from those of other clusters. Cluster analysis belongs to fundamental issues in data mining and machine learning with wide range of applications, cf., e.g., [15], [1].

This paper presents a technique for cluster analysis based on self-organizing neural networks (SONNs) of spanning-tree-like structures and with dynamically defined neighborhood (henceforward referred to as SONNs with DDN) outlined in Kohonen's work [10]. However, we propose an essential generalization of these networks by introducing original mechanisms that, in the course of learning:

L. Rutkowski et al. (Eds.): ICAISC 2014, Part II, LNAI 8468, pp. 713–725, 2014.

a) automatically adjust the number of neurons in the network, b) allow to disconnect the tree-like structure into sub-trees, and c) allow to reconnect some of the sub-trees preserving the no-loop spanning-tree properties. All these features enable them to detect data clusters of virtually any shape and density including both volumetric clusters and thin, piece-wise linear, shell, polygonal, etc. types of clusters. Similar, to above-listed, mechanisms have been proposed by us in [7], [8], [9] ([5], [6] present their earlier versions) to obtain - for clustering purposes - the so-called dynamic SONNs with one-dimensional neighborhood; they can be treated as a special case of the presently proposed generalized SONNs with DNN. First, their details are presented. Then, an illustration of their operation using several synthetic data sets containing data concentrations of various shapes and densities is given. Finally, their application to the clustering of two benchmark data sets is presented.

An idea of evolving topological structures of self-organizing neural networks has been addressed in the literature. However, some existing solutions do not directly deal with data clustering, e.g. in [11] and [14] evolving neuron trees are used to decrease the computational complexity of Winner-Takes-Most (WTM) learning algorithm. In [16] tree structures are used to visualize the obtained results for the purpose of comparison with conventional decision trees. Among data clustering techniques (to some extent alternative to our approach), evolving topological structures are proposed in [3], [13], and in [4], [2]. However, the approaches of [3], [13] do not enable to detect, in an automatic way, the number of clusters in data sets. In turn, the results of experiments presented in [4], [2] do not provide an information on how effective the proposed approaches are in terms of the automatic detection of the number of clusters in data sets.

2 Generalized SONNs with DDN for Clustering Analysis

First, we consider the conventional SONN with one-dimensional neighborhood. Assume that the network has n inputs $x_1, x_2, \ldots, x_n$ and consists of m neurons arranged in a chain; their outputs are $y_1, y_2, \ldots, y_m$, where $y_j = \sum_{i=1}^{n} w_{ji} x_i$, $j = 1, 2, \ldots, m$ and w_{ji} are weights connecting the i-th input of the network with the output of the j-th neuron. Using vector notation ($\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^T$, $\boldsymbol{w}_j = (w_{j1}, w_{j2}, \ldots, w_{jn})^T$), $y_j = \boldsymbol{w}_j^T \boldsymbol{x}$. The learning data consists of L input vectors $\boldsymbol{x}_l$ ($l = 1, 2, \ldots, L$). The first stage of any Winner-Takes-Most (WTM) learning algorithm that can be applied to the considered network, consists in determining the neuron $j_{\boldsymbol{x}}$ winning in the competition of neurons when learning vector $\boldsymbol{x}_l$ is presented to the network. Assuming the normalization of learning vectors, the winning neuron $j_{\boldsymbol{x}}$ is selected such that

$$d(\boldsymbol{x}_l, \boldsymbol{w}_{j_{\boldsymbol{x}}}) = \min_{j=1,2,\ldots,m} d(\boldsymbol{x}_l, \boldsymbol{w}_j), \tag{1}$$

where $d(\boldsymbol{x}_l, \boldsymbol{w}_j)$ is a distance measure between $\boldsymbol{x}_l$ and $\boldsymbol{w}_j$; throughout this paper, the Euclidian distance measure will be applied

$$d_E(\boldsymbol{x}_l, \boldsymbol{w}_j) = \sqrt{\sum_{i=1}^{n} (x_{li} - w_{ji})^2}. \tag{2}$$

The WTM learning rule can be formulated as follows:

$$\boldsymbol{w}_j(k+1) = \boldsymbol{w}_j(k) + \eta_j(k) N(j, j_{\boldsymbol{x}}, k)[\boldsymbol{x}(k) - \boldsymbol{w}_j(k)], \tag{3}$$

where k is the iteration number, $\eta_j(k)$ is the learning coefficient, and $N(j, j_{\boldsymbol{x}}, k)$ is the neighborhood function. At this point, we have to address the problem of a spanning-tree-like structure of the SONN with DDN. The neighborhood of a given neuron in such a topology is defined along the arcs emanating from that neuron as illustrated in Fig. 1. Therefore, the paths between any two neurons in such a structure are the pieces of SONN with one-dimensional neighborhood. The topological distance $d_{tpl}(j, j_{\boldsymbol{x}})$ between the $j_{\boldsymbol{x}}$-th neuron and some other neurons is equal to 1 if those other neurons are direct neighbors of the $j_{\boldsymbol{x}}$-th one as shown in Fig. 1. The distance $d_{tpl}(j, j_{\boldsymbol{x}}) = 2$ for the neurons that are second along all paths starting at the $j_{\boldsymbol{x}}$-th one (see Fig. 1), etc. The topological distance measure is the basis of the neighborhood function $N(j, j_{\boldsymbol{x}}, k)$. In this paper, the Gaussian-type neighborhood function is used:

$$N(j, j_{\boldsymbol{x}}, k) = e^{-\frac{d^2_{tpl}(j, j_{\boldsymbol{x}})}{2\lambda^2(k)}} \tag{4}$$

where $\lambda(k)$ is the radius of the neighborhood (the width of the Gaussian "bell").

As already mentioned, the generalization of the above-presented SONN with DDN consists in introducing mechanisms that allow the network:

I) to automatically adjust the number of neurons in the network by removing low-active neurons from the network and adding new neurons in the areas of existing high-active neurons,
II) to automatically disconnect the network, as well as to reconnect some of the sub-networks again preserving the no-loop spanning-tree properties.

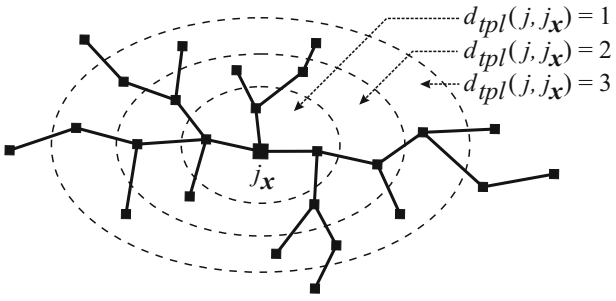

Fig. 1. Examples of neighborhood of the $j_{\boldsymbol{x}}$-th neuron

These two features enable the generalized SONN with DDN to fit in the best way the structures that are "encoded" in data sets to display them to the user. In particular, the number of disconnected sub-networks is equal to the number of clusters detected in a given data set. Moreover, the neurons in a given sub-network create a multi-point prototype of the corresponding cluster. The mechanisms I and II are implemented by activating (under some conditions) - after each learning epoch - five successive operations (cf. [7]):

1) the removal of single, low-active neurons,
2) the disconnection of the network (sub-network) into two sub-networks,
3) the removal of small-size sub-networks,
4) the insertion of additional neurons into the neighborhood of high-active neurons in order to take over some of their activities,
5) the reconnection of two selected sub-networks.

The operations 1, 3 and 4 are the components of the mechanism I, whereas the operations 2 and 5 govern the mechanism II. Based on experimental investigations, the following conditions for activating particular operations have been formulated.

Operation 1: The neuron no. j_r is removed from the network if its activity - measured by the number of its wins win_{jr} - is below an assumed level win_{min}, i.e., $win_{jr} < win_{min}$. win_{min} is experimentally selected parameter (usually, $win_{min} \in \{2, 3, ..., 7\}$). The removal of the j_r-th neuron is followed by reconfiguration of the network topology as shown in Fig. 2. If the j_r-th neuron has only two neighbors (Fig. 2a), they are now topologically connected. In the case of three or more neighbors of the j_r-th unit (Fig. 2b), one of them, say j_1, which is nearest to the j_r-th one in terms of their weight-vector distance, is selected. Then, the remaining neighbors are topologically connected to the j_1-th unit.

Operation 2: The structure of the network is disconnected into two sub-networks by removing the topological connection between two neighboring neurons j_1 and j_2 (see Fig. 3) after fulfilling the following condition: $d_{E,j_1j_2} > \alpha_{dsc} d_{E,avr}$ where $d_{E,j_1j_2} = d_E(\boldsymbol{x}_{j_1}, \boldsymbol{x}_{j_2})$ (d_E is defined in (2)), $d_{E,avr} = \frac{1}{P}\sum_{p=1}^{P} d_{E,p}$ is the average distance between two neighboring neurons for all pairs of such neurons in the network ($d_{E,p}$ is the d_E distance for the p-th pair of neighboring neurons, $p = 1, 2, \ldots, P$), and α_{dsc} is experimentally selected parameter governing the disconnection operation (usually, $\alpha_{dsc} \in [2, 4]$).

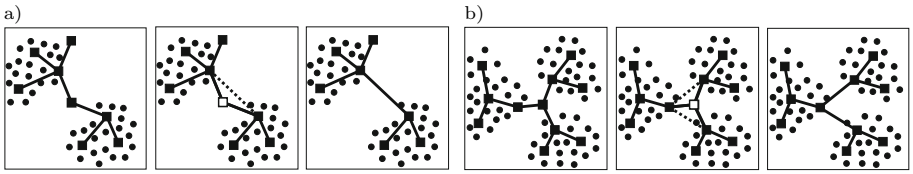

Fig. 2. Removal of single, low-active neuron connected with two (a) and three (in general, more than two) (b) neighboring neurons (illustrations of the exemplary network structure before, during, and after the operation, respectively)

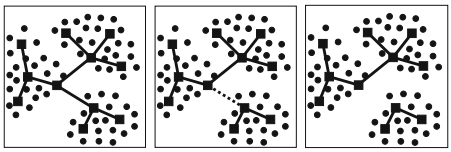

Fig. 3. Disconnection of the network into two sub-networks (illustration of the exemplary network structure before, during, and after the operation, respectively)

Operation 3: A sub-network that contains m_s neurons is removed from the system if $m_s < m_{s,min}$, where $m_{s,min}$ is experimentally selected parameter (usually, $m_{s,min} \in \{3, 4\}$).

The operation of the insertion of additional neurons into the neighborhood of high-active neurons in order to take over some of their activities covers 2 cases denoted by 4a and 4b, respectively.

Operation 4a: A new neuron, labelled as (new), is inserted between two neighboring and high-active neurons j_1 and j_2 (see Fig. 4a) if they fulfil the following conditions: $win_{j_1} > win_{max}$ and $win_{j_2} > win_{max}$, where win_{j_1} and win_{j_2} are the numbers of wins of particular neurons and win_{max} is experimentally selected parameter (usually $win_{max} \in \{4, 5, ..., 9\}$). The weight vector $\boldsymbol{w}_{(new)}$ of the new neuron is calculated as follows: $\boldsymbol{w}_{(new)} = \frac{\boldsymbol{w}_{j_1} + \boldsymbol{w}_{j_2}}{2}$.

Operation 4b: A new neuron (new) is inserted in the neighborhood of high-active neuron j_1 surrounded by low-active neighbors (see Fig. 4b) if the following conditions are fulfilled: $win_{j_1} > win_{max}$ and $win_j < win_{max}$ for j such that $d_{tpl}(j, j_1) = 1$, where win_{j_1} and win_{max} are as in Operation 4a and win_j is the number of wins of the j-th neuron. The weight vector $\boldsymbol{w}_{(new)} = [w_{(new)1}, w_{(new)2}, \ldots, w_{(new)n}]^T$ is calculated as follows: $w_{(new)i} = w_{j_1 i}(1 + \xi_i)$, $i = 1, 2, \ldots, n$, where ξ_i is a random number from the interval $[-0.01, 0.01]$. Therefore, particular components of high-active neuron j_1, after experimentally selected random modification in the range of $[-1\%, 1\%]$, give the weight vector $\boldsymbol{w}_{(new)}$ of the new neuron. It is a starting point for the new neuron in its further evolution as the learning progresses.

Operation 5: Two sub-networks S_1 and S_2 are reconnected by introducing topological connection between neurons j_1 and j_2 ($j_1 \in S_1$, $j_2 \in S_2$) - see Fig. 5 - after fulfilling condition: $d_{E,j_1 j_2} < \alpha_{con} \frac{d_{E,avr_{S_1}} + d_{E,avr_{S_2}}}{2}$. $d_{E,j_1 j_2}$ is the same

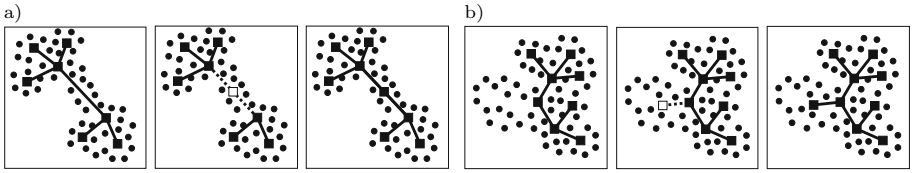

Fig. 4. Insertion of additional neuron between two high-active neighbouring neurons (a) and into the neighborhood of a single high-active neuron (b) (illustrations of the exemplary network structure before, during, and after the operation, respectively)

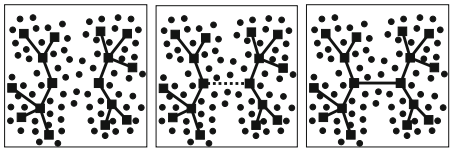

Fig. 5. Reconnection of two sub-networks (illustration of the exemplary network structure before, during, and after the operation, respectively)

as in Operation 2. $d_{E,avr_{S_1}}$ and $d_{E,avr_{S_2}}$ are calculated for sub-networks S_1 and S_2, respectively, in the same way as $d_{E,avr}$ is calculated in Operation 2 for the considered network. α_{con} is experimentally selected parameter that controls the reconnection process (usually, $\alpha_{con} \in [3, 5]$).

The conditions that govern Operations 1 through 5 are checked after each learning epoch. The condition that is fulfilled activates the appropriate operation.

In the experiments presented below, the following values of control parameters are selected: $win_{min} = 2$, $win_{max} = 4$, $m_{s,min} = 3$, $\alpha_{dsc} = 3$, and $\alpha_{con} = 4$. Moreover, the learning process is carried out through 10000 epochs, the learning coefficient $\eta_j(k) = \eta(k)$ of (3) linearly decreases over the learning horizon from $7 \cdot 10^{-4}$ to 10^{-6}, the neighborhood radius $\lambda(k) = \lambda$ of (4) is equal to 2, and the initial number of neurons in the network (at the start of the learning process) is equal to 2.

3 Cluster Analysis in Two-Dimensional Synthetic Data Sets

Fig. 6 shows the performance of the generalized SONN with DDN applied to the set of uniformly distributed data (i.e., without any clusters in them). It is a hard-to-pass test for very many clustering techniques, especially those generating a predefined number of clusters - regardless of whether any clusters exist in data or not. Our approach perfectly passes this test. After initial jump to 2, the number of sub-networks (clusters) stabilizes on 1, i.e., the system detects one big cluster in data (see Fig. 6h) and generates a multi-point prototype for it (see neurons of Fig. 6f). Fig. 6g shows the adjustment of the number of neurons in the network in the course of learning.

Figs. 7, 8, and 9 present further illustrations of the performance of our approach applied to various two-dimensional synthetic data sets. Fig. 7a presents data set used in [17]; it contains two overlapped Gaussian distributions, two concentric rings, and a sinusoidal curve with 10% noise added to data. Fig. 8a presents data set with various types of clusters in it. Both, thin piece-wise linear and two-ellipsoidal as well as volumetric of various shapes and densities clusters are considered. Finally, Fig. 9a presents a "classical" two-spiral data set. As the above-listed figures show, in all of these data sets our approach detects the correct numbers of clusters and generates multi-point prototypes for them.

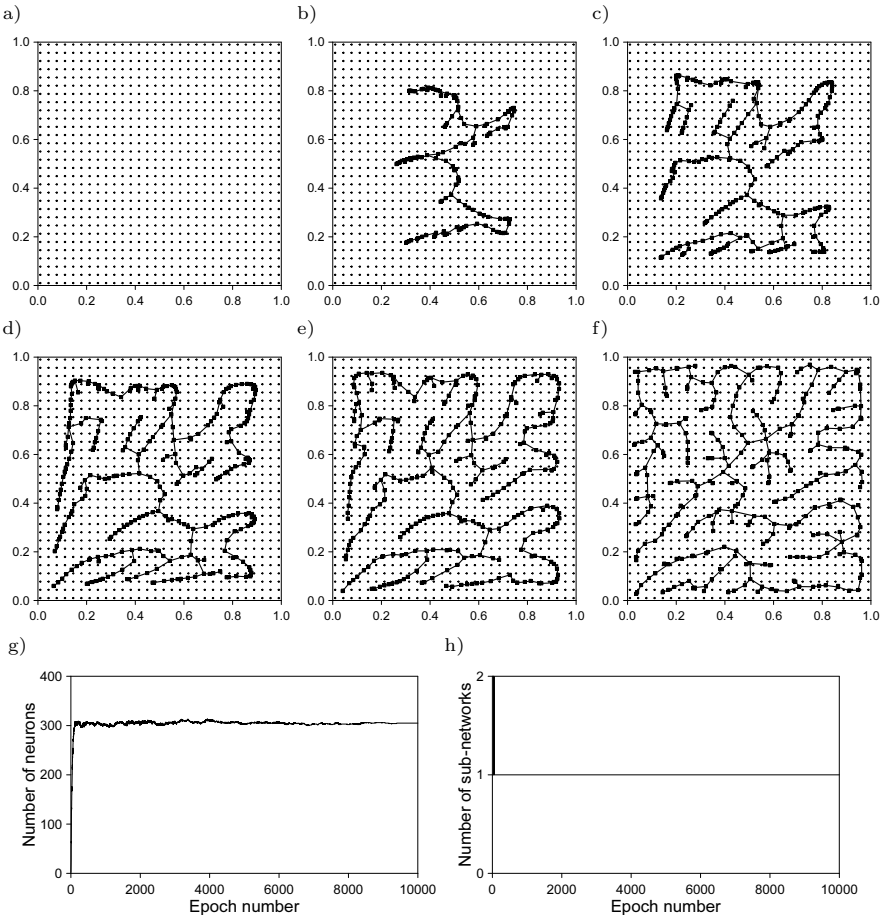

Fig. 6. Synthetic data set (a) and the evolution of the generalized SONN with DDN in it in learning epochs: b) no. 20, c) no. 50, d) no. 100, e) no. 200, and f) no. 10 000 (end of learning), as well as plots of the number of neurons (g) and the number of sub-networks (clusters) (h) vs. epoch number

4 Cluster Analysis in Selected Benchmark Data Sets

Our approach will now be tested using two multidimensional benchmark data sets such as *Breast Cancer Wisconsin* (*Diagnostic*) and *Congressional Voting Records* (*BCWD* and *CVR*, for short) [12]. *BCWD* data set has 569 records and 30 numerical attributes, whereas *CVR* data set - 435 records and 16 nominal attributes. It is essential to note that our approach does not utilize the knowledge on class assignments of particular records and on the number of classes (equal to 2

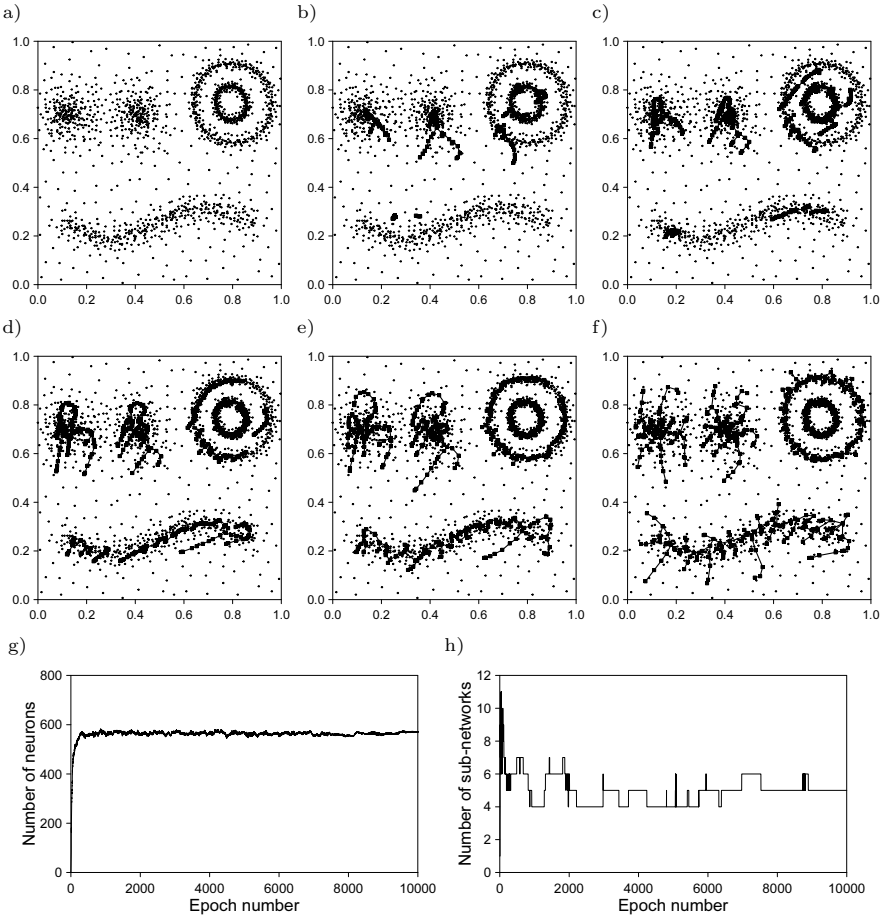

Fig. 7. Synthetic data set (a) and the evolution of the generalized SONN with DDN in it in learning epochs: b) no. 20, c) no. 50, d) no. 100, e) no. 200, and f) no. 10 000 (end of learning), as well as plots of the number of neurons (g) and the number of sub-networks (clusters) (h) vs. epoch number

in both sets). Our approach works in *a fully-unsupervised way*, i.e., it operates on *unlabelled data* and *without any predefinition of the number of clusters* (*classes*).

Figs. 10 and 11 as well as Tables 1 and 2 present the performance of our approach applied to both data sets. First, Figs. 10b and 11b show that our approach detects the correct number of clusters in both data sets. Second, since the number of classes and class assignments are known in both original data

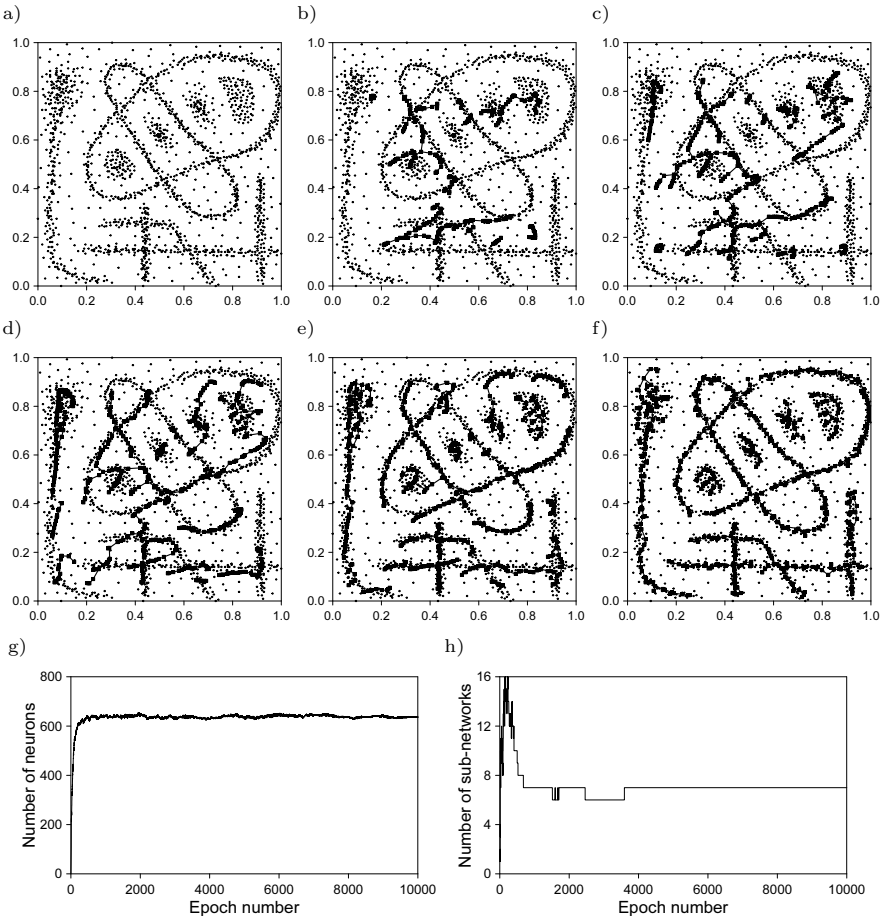

Fig. 8. Synthetic data set (a) and the evolution of the generalized SONN with DDN in it in learning epochs: b) no. 20, c) no. 50, d) no. 100, e) no. 200, and f) no. 10 000 (end of learning), as well as plots of the number of neurons (g) and the number of sub-networks (clusters) (h) vs. epoch number

sets, a direct verification of the obtained results is also possible (see Tables 1 and 2). The percentages of correct decisions, equal to 90.51% (*BCWD* data set) and 94.71% (*CVR* data set), regarding the class assignments are very high (especially, that they have been achieved by the unsupervised-learning systems operating on benchmark data sets).

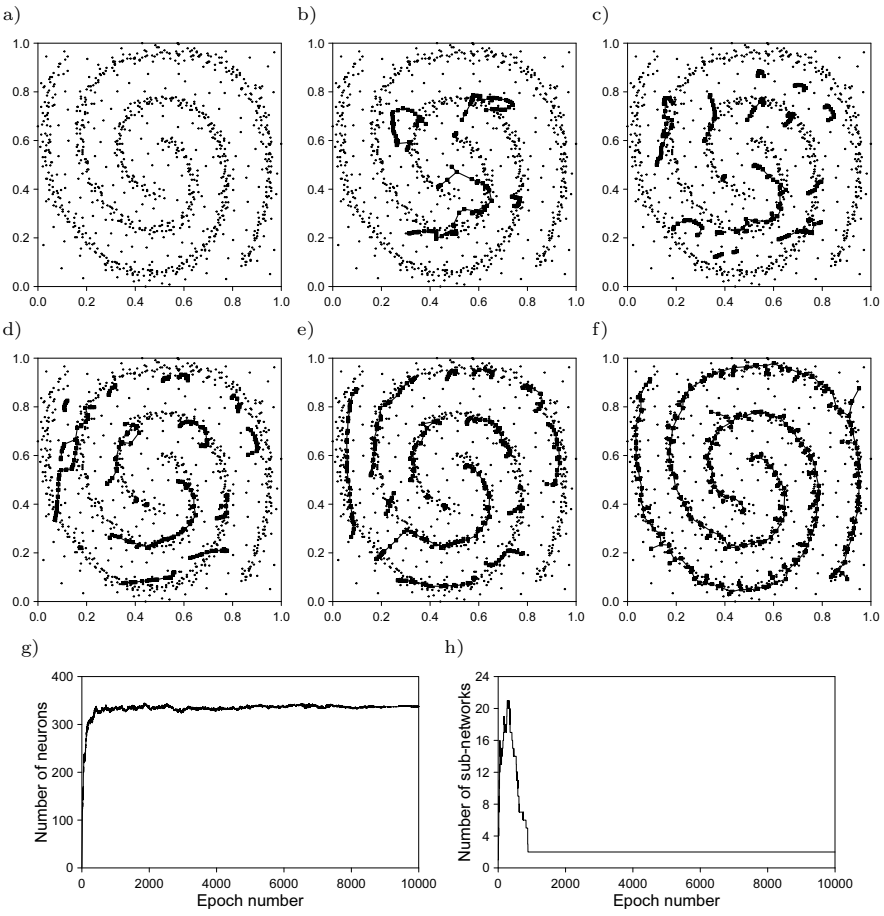

Fig. 9. Synthetic data set (a) and the evolution of the generalized SONN with DDN in it in learning epochs: b) no. 20, c) no. 50, d) no. 100, e) no. 200, and f) no. 10 000 (end of learning), as well as plots of the number of neurons (g) and the number of sub-networks (clusters) (h) vs. epoch number

Table 1. Clustering results for *BCWD* data set

Class label	Number of records	Number of decisions for sub-network labelled:		Number of correct decisions	Number of wrong decisions	Percentage of correct decisions
		Malignant	*Benign*			
Malignant	212	166	46	166	46	78.30%
Benign	357	8	349	349	8	97.76%
ALL	**569**	**174**	**395**	**515**	**54**	**90.51%**

Table 2. Clustering results for *CVR* data set

Class label	Number of records	Number of decisions for sub-network labelled:		Number of correct decisions	Number of wrong decisions	Percentage of correct decisions
		Republican	*Democrat*			
Republican	168	158	10	158	10	94.05%
Democrat	267	13	254	254	13	95.13%
ALL	**435**	**171**	**264**	**412**	**23**	**94.71%**

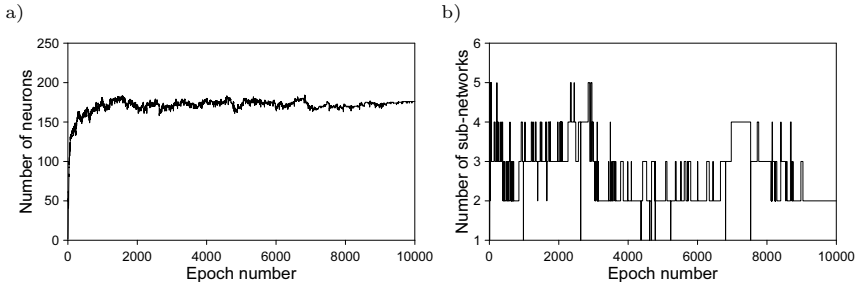

Fig. 10. Plots of the number of neurons (a) and the number of sub-networks (clusters) (b) vs. epoch number (*BCWD* data set)

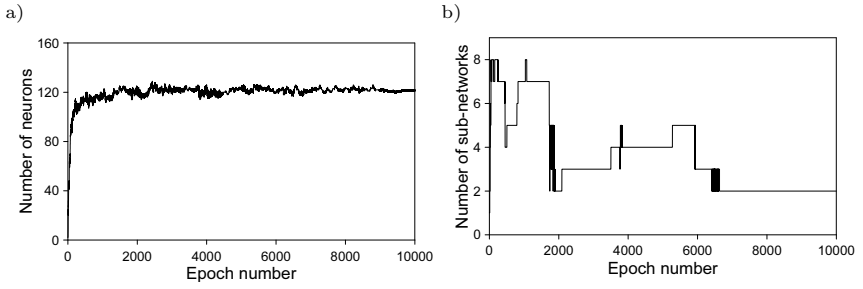

Fig. 11. Plots of the number of neurons (a) and the number of sub-networks (clusters) (b) vs. epoch number (*CVR* data set)

5 Conclusions

The generalized SONNs with DDN that can be effectively applied in complex, multidimensional cluster-analysis problems have been presented in this paper. Our approach works in a fully-unsupervised way, i.e., it operates on unlabelled data and it does not require to predefine the number of clusters in a given data set. The proposed networks, in the course of learning, are able to disconnect their neuron structures into sub-structures and to reconnect some of them again as well as to adjust the overall number of neurons in the system. These features enable them to detect data clusters of virtually any shape and density

including both volumetric ones and thin, shell-like ones. Moreover, the neurons in particular sub-networks create multi-point prototypes of the corresponding clusters. The operation of our approach has been illustrated by means of several diversified synthetic data sets and then our approach has been tested using two benchmark data sets (*Breast Cancer Wisconsin* (*Diagnostic*) and *Congressional Voting Records*) yielding very good results.

Acknowledgments. The numerical experiments reported in this paper have been performed using computational equipment purchased in the framework of the EU Operational Programme Innovative Economy (POIG.02.02.00-26-023/09-00) and the EU Operational Programme Development of Eastern Poland (POPW.01.03.00-26-016/09-00).

References

1. Everitt, B.S., Landau, S., Leese, M.: Cluster Analysis, 4th edn. A Hodder Arnold Publication, J. Willey, London (2001)
2. Forti, A., Foresti, G.L.: Growing hierarchical tree SOM: An unsupervised neural network with dynamic topology. Neural Networks 19, 1568–1580 (2006)
3. Fritzke, B.: Growing cell structures - a self-organizing network for unsupervised and supervised learning. Neural Networks 7, 1441–1460 (1994)
4. Ghaseminezhad, M.H., Karami, A.: A novel self-organizing map (SOM) neural network for discrete groups of data clustering. Applied Soft Computing 11, 3771–3778 (2011)
5. Gorzałczany, M.B., Rudziński, F.: Application of genetic algorithms and Kohonen networks to cluster analysis. In: Rutkowski, L., Siekmann, J.H., Tadeusiewicz, R., Zadeh, L.A. (eds.) ICAISC 2004. LNCS (LNAI), vol. 3070, pp. 556–561. Springer, Heidelberg (2004)
6. Gorzałczany, M.B., Rudziński, F.: Modified Kohonen networks for complex cluster-analysis problems. In: Rutkowski, L., Siekmann, J.H., Tadeusiewicz, R., Zadeh, L.A. (eds.) ICAISC 2004. LNCS (LNAI), vol. 3070, pp. 562–567. Springer, Heidelberg (2004)
7. Gorzałczany, M.B., Rudziński, F.: Cluster analysis via dynamic self-organizing neural networks. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Żurada, J.M. (eds.) ICAISC 2006. LNCS (LNAI), vol. 4029, pp. 593–602. Springer, Heidelberg (2006)
8. Gorzałczany, M.B., Rudziński, F.: Application of dynamic self-organizing neural networks to WWW-document clustering. ICAISC 2006 1(1), 89–101 (2006); (also presented at 8th Int. Conference on Artificial Intelligence and Soft Computing ICAISC 2006). Zakopane (2006)
9. Gorzałczany, M.B., Rudziński, F.: WWW-newsgroup-document clustering by means of dynamic self-organizing neural networks. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2008. LNCS (LNAI), vol. 5097, pp. 40–51. Springer, Heidelberg (2008)
10. Kohonen, T.: Self-organizing Maps, 3rd edn. Springer, Heidelberg (2000)
11. Koikkalainen, P., Oja, E.: Self-organizing hierarchical feature maps. In: Proc. of 1990 International Joint Conference on Neural Networks, San Diego, CA, vol. II, pp. 279–284 (1990)
12. Machine Learning Database Repository. University of California at Irvine, `http://ftp.ics.uci.edu`

13. Martinez, T., Schulten, K.: A "Neural-Gas" network learns topologies. In: Kohonen, T., et al. (eds.) Artificial Neural Networks, pp. 397–402. Elsevier, Amsterdam (1991)
14. Pakkanen, J., Iivarinen, J., Oja, E.: The evolving tree - a novel self-organizing network for data analysis. Neural Processing Letters 20, 199–211 (2004)
15. Pedrycz, W.: Knowledge-Based Clustering: From Data to Information Granules. J. Willey, Hoboken (2005)
16. Samsonova, E.V., Kok, J.N., IJzerman Ad, P.: TreeSOM: Cluster analysis in the self-organizing map. Neural Networks 19, 935–949 (2006)
17. Shen, F., Hasegawa, O.: An incremental network for on-line unsupervised classification and topology learning. Neural Networks 19, 90–106 (2006)