CLUSTER ANALYSIS APLICATIONS IN MATLAB USING KOHONEN NETWORK

S. Kajan, I. Sekaj, M. Lajtman

Institute of Control and Industrial Informatics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Slovak Republic

Abstract

This paper deals with the Kohonen Self-organizing maps for cluster analysis applications. The cluster analysis represents a group of methods whose aim is to classify the objects into clusters. For solving cluster analysis applications many new algorithms using neural networks have been used. This paper describes the use of an advanced method of neural network represented by Kohonen self-organizing maps. Also some examples of applications for cluster analysis in Matlab are presented.

1 Introduction

The cluster analysis represents a group of methods whose aim is to classify objects into clusters. This paper deals with the use of an advanced method of neural network (NN) represented by Kohonen self-organizing maps. The basic principle of their function is cluster analysis, i.e. ability of the algorithm, to find out certain properties and dependencies just in the offered training data without presence of any external information. The idea of just the network structure to self-organizing has been formed for the first time at the beginning of seventies by Von der Malsburg and later followed by Willshaw. All of their works from that time are characteristic by orientation to biological knowledge, mainly from the field of neuron and cerebral cortex research [4, 6, 7].

2 Kohonen neural network

The basic idea of Kohonen network emanates from knowledge, that the brain uses inner space data representation for storing information. At first, data received from the environment are transformed to vectors which are encoded to the neural network.

The extension of competitive learning rests on the principle that there are several winners permitted. The output from such NN is geometrically organized to some arrangement, e.g. abreast, or to the rectangle and thus there is a possibility of neighbour identification. This layer is called Kohonen layer. Number of inputs entering to the network is equal to the input space dimension. In Figure 1 the topology of Kohonen network with 2 inputs is depicted.

The neuron structure in Kohonen network is different from the neuron structure in perceptron network. The number of inputs entering to the neuron is equal to the number of inputs entering to the Kohonen network. The weights of these inputs serve for the encoding of patterns, which represent submitted patterns as well as in the case of perceptron. These neurons do not have actual transfer function. The only operation, which neuron executes is calculation of distance (error) d of submitted pattern from pattern encoded in the weights of given neuron according to the equation:

$$d = \sum_{i=0}^{N-1} \left[x_i(t) - w_i(t) \right]^2 \tag{1}$$

where N is a number of submitted patterns, $x_i(t)$ are individual elements of input pattern and $w_i(t)$ are appropriate weights of neuron which represent the encoded patterns [4, 6, 7].

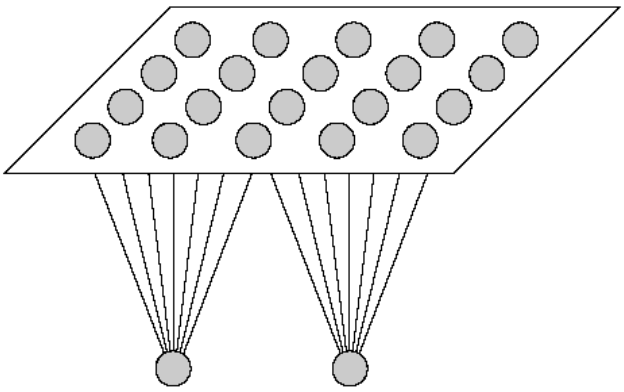

Figure 1: Topology of Kohonen network with two inputs

The learning algorithm tries to arrange the neurons in the grid to the certain areas so that they are able to classify the submitted input data. This organization of neurons can be imagined as unwrapping of primarily wrinkled paper on the plane so that it covers the whole input space.

The process of learning is autonomous, i.e. without the presence of external additional information and it's performed iteratively, i.e. weights are adapted in each learning step. This adaptation is based on comparison of input patterns and vectors embedded in each neuron (see equation 1). Whenever the vector, which best fits the input pattern is found, it is adapted as well as all the vectors of neurons situated near this neuron. The entire grid is gradually optimized to fit the input space of training data as good as possible.

So-called surroundings of neurons take a great role in the learning. The surroundings are defined for each neuron particularly. In the initialization phase, the surroundings are usually chosen to cover all the neurons in the grid, i.e. radius of the surroundings is equal to the number of neurons on the one side of the grid. The surroundings are gradually reduced and also the parameter of learning is reduced similarly. The surrounding of a neuron is calculated for a so-called winning neuron, whose vector of weights fits best the input pattern. For this neuron and its surrounding the weights are adapted. In Figure 2 a neighbourhood function (Gaussian surrounding) which represents adaptation rate of surrounding weights is depicted. This function has been recognized from the real biological neural networks and it is called neighbourhood function (sometimes also Mexican hat function). The neighbourhood function (NF) can be described by the equation:

$$\lambda\left(j^*, j\right) = h(t).\exp\left(\frac{d_E^2\left(j^*, j\right)}{r^2(t)}\right) \tag{2}$$

where $d_E(j^*,j)$ represents euclidean distance of the winner neuron j^* and another compared neuron, $r(t)$ is the radius of neighbourhood function and $h(t)$ is a height of neighbourhood function which decreases to zero during the time whereby it provides decreasing the surroundings during the learning [4, 6, 7].

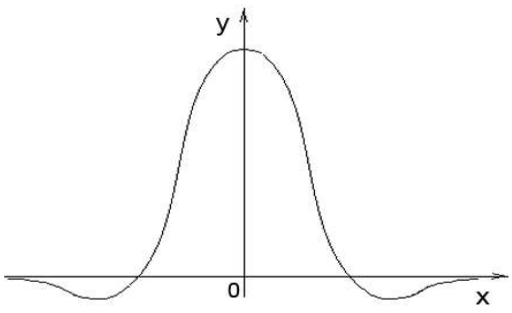

Figure 2: Gaussian neighbourhood function

In the process of learning, the best results are achieved when the size of surrounding r is discreetly decreasing during the time and also it is useful to reduce the size of the learning parameter $\eta(t)$ too. The parameter of learning is used to control the learning rate and its value is from the interval $0 \leq \eta(t) \leq 1$. At the beginning it is efficient if the parameter is maximal so that the network of neurons fans out as fast as possible and it covers the largest area of the definition range of input vectors. This parameter is gradually decreasing and it allows the network the finer setting of output neurons weights. The method which modifies the size of the surroundings radius r and also the parameter of learning η is shown in Figure 3.

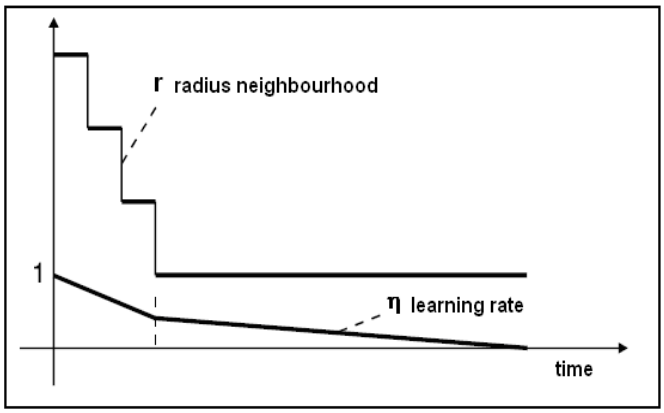

Figure 3: Time behaviour of the learning parameters

The learning algorithm is described in following steps:

Step 1: Initialization:

- Adjustment of weights w_{ij}, $0 \leq i \leq N-1$, $0 \leq j \leq M-1$, to small initial values for all the synaptic joints from N inputs to M output neurons.
- Adjustment of the learning parameter $\eta(0)$ on a value near 1. The value of this parameter is from the interval $0 \leq \eta(t) \leq 1$ and it is used to control of the learning rate.
- The initial size of the neighbourhood surroundings of output neurons has to be adjusted. It is also necessary to adjust the minimal value of surroundings (usually, the minimal surrounding is the only given neuron).

Step 2: Submission of pattern:

- Submission of training pattern $X(t) = \{ x_0(t), x_1(t),..., x_{N-1}(t) \}$ to the inputs of the neural network.

Step 3: Calculation of distances of patterns:

- Calculation of distance (similarity) d_j between the submitted pattern and all the output neurons j according to equation

$$d_j = \sum_{i=0}^{N-1} \left[x_i(t) - w_{ij}(t) \right]^2 \tag{3}$$

where $x_i(t)$ are elements of input pattern $X(t)$ and $w_{ij}(t)$ are weights between i-th input and j-th output neuron, which represents the encoded patterns.

Step 4: Selection of the winning (most similar) neuron:

- Selection of output neuron j^*, which satisfy the next condition and thus it fits the most similar neuron:

$$d_{j^*} = \min\left(d_j\right) \tag{4}$$

Step 5: Adaptation of all the weights:

- Calculation of the neighbourhood rate from the winning neuron according to equation (2).
- Adaptation of weights of neurons according to the following equation

$$w_{ij}(t+1) = w_{ij}(t) + \eta(t).\lambda(j^*, j)\left[x_i(t) - w_{ij}(t)\right] \quad (5)$$

Step 6: Continue of the learning process:

- Decreasing of the learning parameter η and the size of neighbourhood surroundings r after the submission of all the patterns.
- End of learning, when the required number of training steps is done or the required accuracy is attained, otherwise continue with the step 2.

After the phase of network learning the phase of execution is following, in which the network responds to the submitted pattern with classifying the pattern to the appropriate class. In the phase of execution only steps 2 to 4 from the learning process are executed, where the winning neuron representing the recognized cluster is received [2, 4, 6, 7].

3 Cluster analysis using Kohonen network

The aim of the classification (or cluster analysis) is the ability of the learning algorithm which is based on the Kohonen network to set neurons to clusters of the submitted patterns and thus to distribute submitted patterns into the clusters. In Figure 4 the common scheme of the Kohonen network allowing classification of inputs $X=(x_1,x_2,...,x_m)$ into classes is depicted.

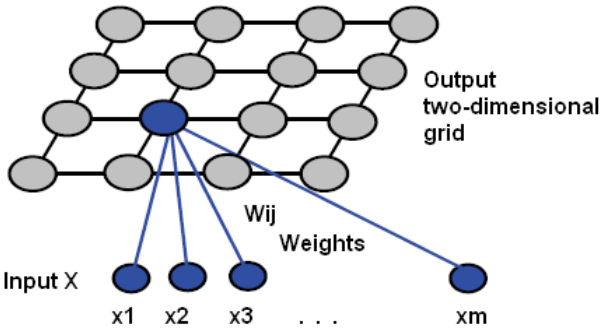

Figure 4: Kohonen network for cluster analysis

For the algorithm demonstration let us use a simple example with a Kohonen network with 2 inputs and with 9 neurons in the grid 3x3 [1, 3, 5]. The aim is to set the initial parameters: learning step η_0, size of neighbourhood surroundings r_0, height of neighbourhood function h_0, number of learning cycles *num_cycles* so that the learning algorithm which is described in the previous chapter has the best performance. Following values of the learning step η, the radius of the neighbourhood function r and the height of the neighbourhood function h in dependence on number of learning cycle t have been considered:

$$\eta(t) = \exp(\eta_0 / cycle) - 1 \quad (6)$$

$$r(t) = r_0 . \exp(-cycle) \quad (7)$$

$$h(t) = \exp((1 - h_0)/cycle) \quad (8)$$

In Figure 5 training points with optimal distribution of neurons of the Kohonen network are shown. The optimal setup of learning algorithm parameters is in Table 1. In Figure 6 courses of learning step η, size of neighbourhood r and the height of neighbourhood function h are shown. The dependence of the cluster analysis efficiency on the choice of the initial learning algorithm parameters are listed in Tables 2 to 5.

Number of cycles	Learning rate η_0	Height NF h_0	Radius NF r_0
8	0.7	1.2	10

Table 1: Initial learning algorithm parameters

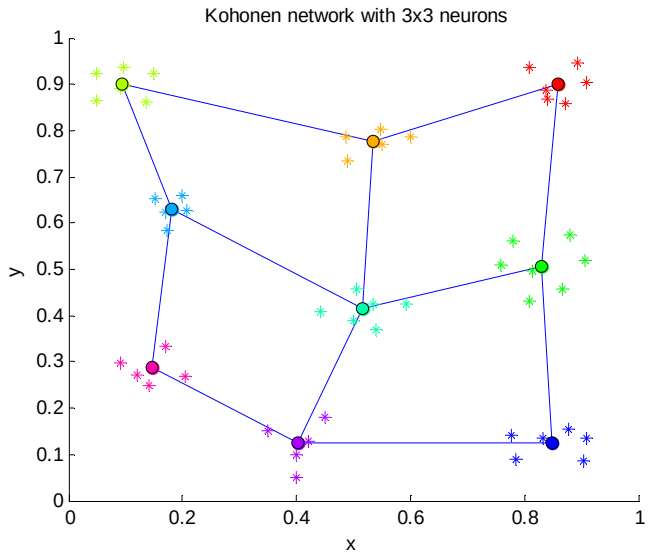

Figure 5: Result of cluster analysis using Kohonen network

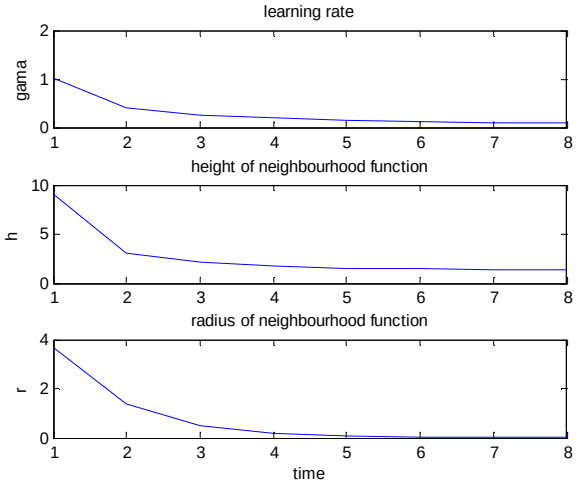

Figure 6: Trends of algorithm parameters

Number of cycles	2	4	6	8	10	12	14	16
Efficiency [%]	1	87	89	88	88	87	89	90

Table 2: Algorithm efficiency / number of cycles

Learning rate η_0	0.1	0.3	0.5	0.7	0.9	1.5
Efficiency [%]	10	93	90	95	87	77

Table 3: Algorithm efficiency / learning rate

Height NF h_0	0.2	0.4	0.6	0.8	1	1.2	1.4
Efficiency [%]	81	88	87	86	90	90	87

Table 4: Algorithm efficiency / height of NF

Radius NF r_0	1	2	4	6	8	10	12
Efficiency [%]	2	63	72	82	84	89	89

Table 5: Algorithm efficiency / radius of NF

4 Case studies

4.1 Distribution centres optimisation using Kohonen network

The aim of the first application was to use the Kohonen algorithm to the problem of the distributor location optimisation. We need to arrange the distribution centres in such a way that the distance from each centre to the surrounding customers is minimal and the total freightage of transported goods will be minimal too. For this application a graphical environment in Matlab has been created using a map of Slovakia. Neurons serve as distributional centres and the input data are used as representatives of customer on the map. The task is the distributional centre location optimisation. But this example doesn't regard the road system, not even the level of traffic or other logistic factors. The algorithm optimises only the geometrical location of the designed distribution centres considering the distance to customers. In Figure 7 graphical environment of the program is depicted. In Figure 8 the obtained results of the optimisation are presented.

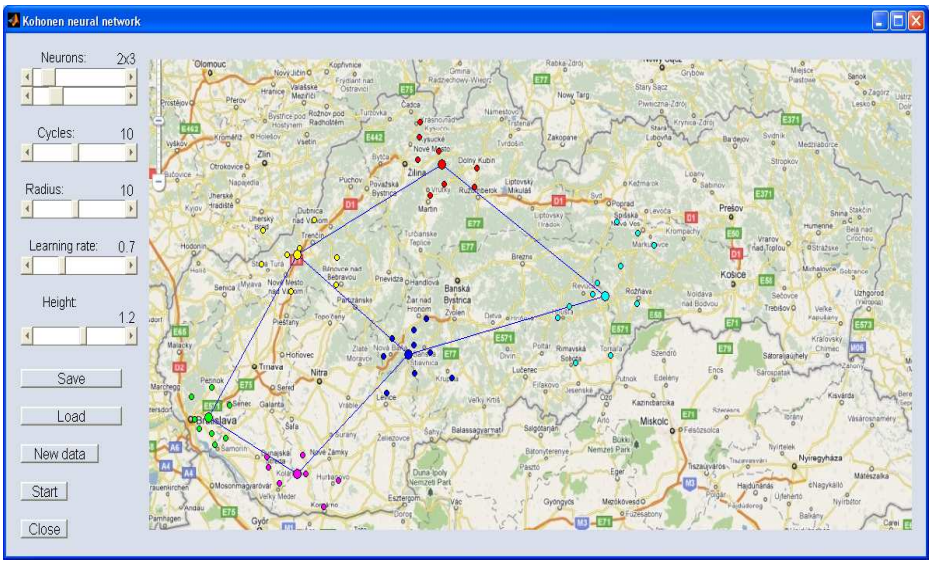

Figure 7: Program in Matlab for distribution centre optimisation using Kohonen network

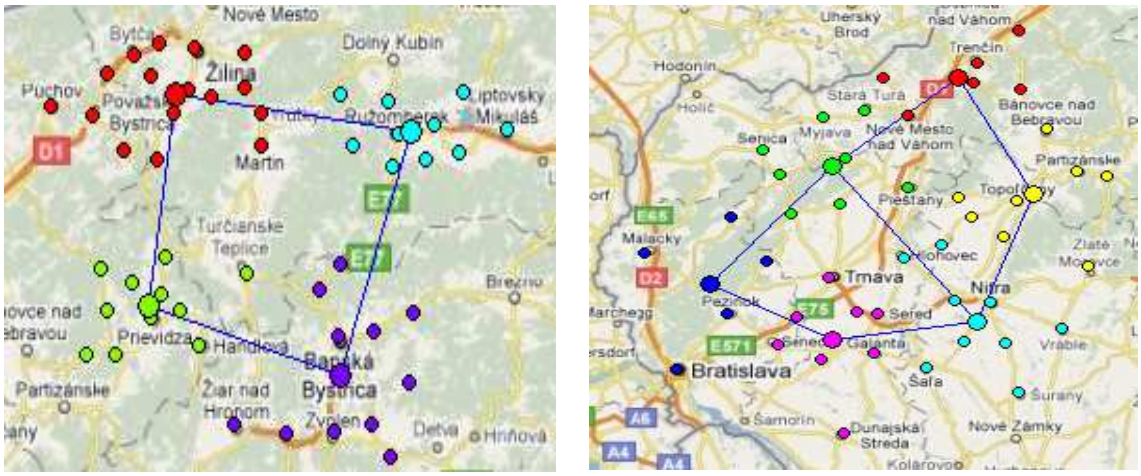

Figure 8: Result of the distribution centres optimisation

4.2 Example of databases cluster analysis

In real world applications raw data, which are also named *dirty data*, can contain errors, missing values, redundant information or they are incomplete and inconsistent. Most of data mining process needs a pre-processing stage, which objectives to carry out tasks such as data cleaning, data integration, transformation or data reduction. This important step is sometimes neglected in data mining processes. Relational database is a set of two-dimensional tables interrelated by one or more attributes. Each table is a two-dimensional structure of rows and columns where each row represents a record from the database and each column represents an attribute associated with that record. After pre-processing stage, data are usually arranged in single table known as data matrix, which must satisfy the requirements of the chosen algorithm. The data matrix D is formed by a set of *p* vectors, where each vector represents an element of the input set. Each vector has *n* components, which correspond to the set of attributes that identify it [3]. A data matrix example related to the presented children database with some parameters is in Table 6. In this example some attributes were removed, others were transformed and the whole dataset was normalized to range 0 to 1. As the example for training of the Kohonen network data from three columns: age, weight and height of Table 6 were used. Learning algorithm parameters were adjusted by the Table 1. In Figure 10 training data points with optimal distribution of neurons of the Kohonen network are shown.

No.	Name	Sex	Age	Weight	Height	....	Sport
1	Fero Mrkva	M	8	30	125	....	95
2	Anna Pekná	F	10	32	145	....	30
3	Marta Sivá	F	9	28	142	....	80
4	Peter Veľký	M	13	45	162	....	80
5	Dušan Malý	M	12	42	158	....	60
....	....	....	....	....	....	....	....
N	Pavol Ondava	M	10	38	141	....	90

Table 6: Children database with some parameters

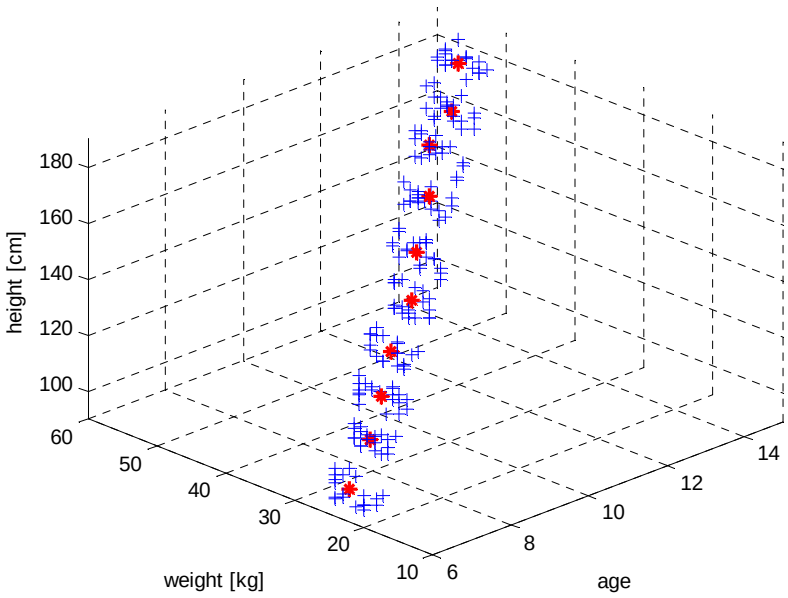

Figure 10: Result of cluster analysis for children database with three parameters

5 Conclusion

The main objective of this article was to describe the Kohonen artificial neural network principle, its training and use and to demonstrate good properties of the Kohonen network for cluster analysis applications. The created programs in the Matlab environment have been used for distribution centres optimisation and cluster analysis in database applications.

Acknowledgement

The work has been supported by the grants of the Slovak grant agency VEGA no. 1/0544/09 and no. 1/0690/09.

References

[1] H .Demuth, M. Beale. *Neural Network Toolbox, For use with Matlab.* User's guide, 2003
[2] P. Dostál, P. Pokorný. *Cluster analysis and neural network.* In: Technical Computing Prague 2009, 17th Annual Conference Proceedings. Prague, Czech Republic, 2008.
[3] F. Gorgonio, J. Costa. *Privacy-Preserving Clustering on Distributed Databases: A Review and Some Contributions*. In: book Self Organizing Maps - Applications and Novel Algorithm Design, Published by InTech, Rijeka, Croatia, january 2011, p. 33, ISBN 978-953-307-546-4
[4] V. Kvasničkа a kol. *Úvod do teórie neurónových sietí.* IRIS Bratislava, 1997. 142 s. ISBN 80-88778-30-1
[5] L. Meza, L. Neto. *Modelling with Self-Organising Maps and Data Envelopment Analysis: A Case Study in Educational Evaluation.* In: book Self Organizing Maps - Applications and Novel Algorithm Design, Published by InTech, Rijeka, Croatia, january 2011, p. 33, ISBN 978-953-307-546-4
[6] P. Sinčák, G. Andrejková. *Neurónové siete Inžiniersky prístup (1. diel).* Košice: ELFA, 1996. ISBN 80-88786-38-X
[7] M. Šnorek, M. Jiřina. *Neuronové sítě a neuropočítače.* 1. vyd. Praha: ČVUT, 1996. 39 s. ISBN 80-01-01455-X

Ing. Slavomír Kajan, PhD, E-mail: slavomir.kajan@stuba.sk
Institute of Control and Industrial Informatics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava

Doc. Ing. Ivan Sekaj, PhD, E-mail: ivan.sekaj@stuba.sk
Institute of Control and Industrial Informatics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava

Bc. Miloš Lajtman, E-mail: lajtman@gmail.com
Institute of Control and Industrial Informatics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava