Visualization of multidimensional data taking into account the learning flow of the self-organizing neural network

GINTAUTAS DZEMYDA
Institute of Mathematics and Informatics
Akademijos St.4.
Vilnius 2600, Lithuania
Dzemyda@ktl.mii.lt

OLGA KURASOVA
Institute of Mathematics and Informatics
Akademijos St.4.
Vilnius 2600, Lithuania
Kurasova@ktl.mii.lt

ABSTRACT

In the paper, we discuss the visualization of multidimensional vectors taking into account the learning flow of the self-organizing neural network. A new algorithm realizing a combination of the self-organizing map (SOM) and Sammon's mapping has been proposed. It takes into account the intermediate learning results of the SOM. The experiments have showed that the algorithm gives lower mean projection errors as compared with a consequent application of the SOM and Sammon's mapping. This is the essential advantage of the new algorithm, i.e. we succeed to eliminate the influence of the "magic factor" α $(0<\alpha\leq 1)$ on Sammon's mapping results. For larger values of α $(\alpha>1)$, the mean projection error grows. However, in this case the new algorithm operates more stable and gives smaller values of the mean projection error.

Keywords

Sammon's mapping, self-organizing maps, neural networks, projection error, visualization

1. INTRODUCTION

Visualization of multidimensional data is a complicated problem followed by extensive researches. There exist a lot of methods that can be used for reducing the dimensionality of data, and, particularly, for visualizing the *n*-dimensional vectors. A deep review of the methods is performed, e.g., in [Kas97a] and [Koh01a]. However there is no universal method. The self-organizing map [Koh01a] and Sammon's algorithm (mapping, projection) [Sam69a] are the methods often used for the visaulization of multidimensional vectors. When a multidimensional space is projected onto a plane, the projection errors are inevitable. It is necessary to create methods that minimize these errors or that allow to increase the comprehension of multidimensional data. It is shown in [Dze01a] experimentally that a combination of the SOM and Sammon's mapping is an effective method of visualization. The so-called vectors-winners, obtained after neural network training, are analyzed and visualized here by using Sammon's algorithm. However, the results of Sammon's algorithm are dependent on the initial data. In this paper, we have proposed a new combination of the SOM and Sammon's mapping. Here the multidimensional data are projected onto the plane by using Sammon's algorithm, taking into account the process of SOM training. The experiments have showed that the new algorithm gives lower mean Sammon's projection errors as compared with the applicaton of Sammon's algorithm after the SOM training is complete. Moreover, the dependance of Sammon's projection error on the so-called "magic factor" has been reduced.

Journal of WSCG, Vol.11, No.1, ISSN 1213-6972
WSCG'2003, February 3-7, 2003, Plzen, Czech Republic.

2. BASIC ALGORITHMS

Sammon's algorithm. Sammon's projection [Sam69a] is a nonlinear projection method to map a high-dimensional space onto a space of lower dimensionality. In our case, the initial dimensionality is *n*, and the resulting one is 2.

Let us have vectors $X_i=(x_{i1},x_{i2},...,x_{in})$, $i=1,...,s$ from an *n*-dimensional space R^n. The pending

problem is to visualize (get the projection) these *n*-dimensional vectors $X_i, i=1,...,s$ onto the plane R^2. Two-dimensional vectors $Y_1, Y_2, ..., Y_s \in R^2$ will correspond to them. Here $Y_i = (y_{i1}, y_{i2}),\ i=1,...,s$. Denote the distance between the vectors X_i and X_j by d_{ij}^*, and the distance between the corresponding vectors in the projected space (Y_i and Y_j) by d_{ij}. Sammon's algorithm tries to minimize the distortion of projection:

$$E_s = \frac{1}{\sum\limits_{\substack{i,j=1 \\ i<j}}^{n} d_{ij}^*} \sum_{\substack{i,j=1 \\ i<j}}^{n} \frac{(d_{ij}^* - d_{ij})^2}{d_{ij}^*} \qquad (1)$$

The coordinates y_{ik}, $i=1,...,s,\ k=1,2$ of the two-dimensional vectors $Y_i \in R^2$ are computed by the iteration formula:

$$y_{ik}(m'+1) = y_{ik}(m') - \alpha \frac{\dfrac{\partial E_s(m')}{\partial y_{ik}(m')}}{\left|\dfrac{\partial^2 E_s(m')}{\partial y_{ik}^2(m')}\right|} \qquad (2)$$

Here m' denotes the iteration, α is named a "magic factor", because the error of projection depends on it. In fact, the error depends both on α and the initial values $Y_1^0, Y_2^0, ..., Y_s^0$ of vectors $Y_1, Y_2, ..., Y_s$. It is found completely experimentally that $\alpha \in [0.3, 0.4]$ guarantees fairly good convergence [Koh01a].

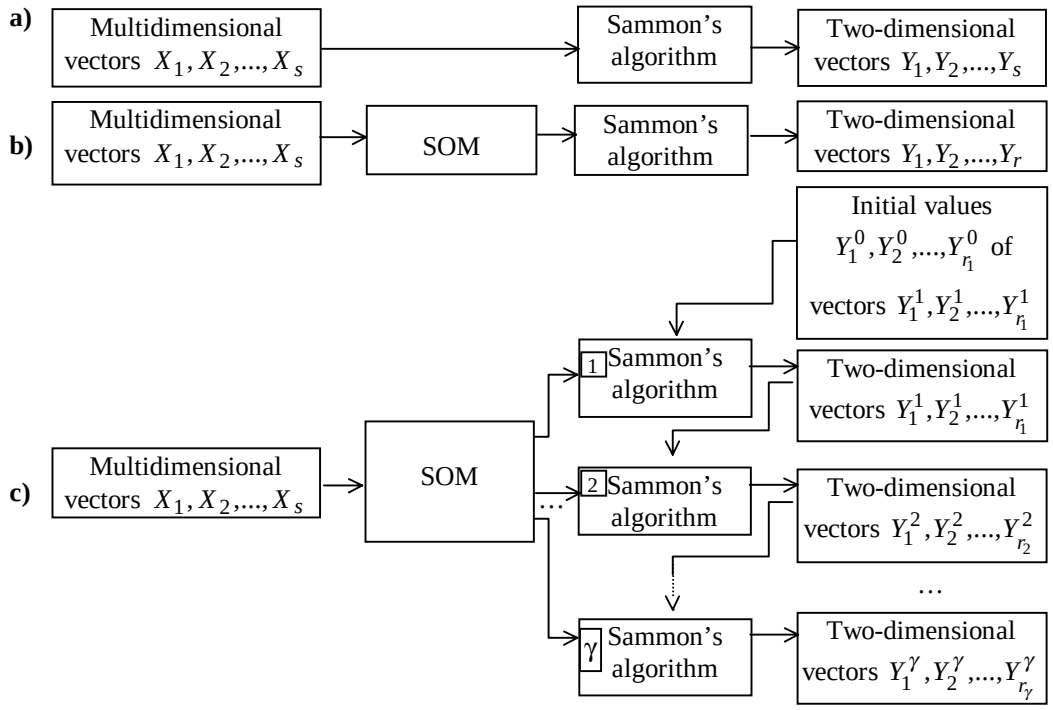


Figure 1. Three scenarios of the projecting multidimensional vectors onto the plane

Self-organizing map (SOM). The self-organizing map (SOM) [Koh01a] is a class of neural networks that are trained in an unsupervised manner, using competitive learning. It is a well-known method for mapping a high-dimensional space onto a low-dimensional one. We present here some general details on the SOM. We consider here a mapping onto a two-dimensional grid of neurons. Let $X_1, ..., X_s \in R^n$ be a set of *n*-dimensional vectors for mapping. Usually, the neurons are connected to each other via a rectangular or hexagonal topology. Let us consider an example of the rectangular case, because all ideas can be easily extended to the hexagonal one. The rectangular SOM is a two-dimensional array of neurons $M = \{m_{ij}, i=1,...,k_x, j=1,...,k_y\}$. Here k_x is the number of rows, and k_y is the number of columns. The total number of neurons is equal to $k_x \times k_y$. All neurons adjacent to a given neuron can be defined as its neighbours of a first order, then the neurons adjacent to a first-order neighbour, excluding those already considered, as neighbours of a second order, etc. The dimension of

the vectors, that will be presented as inputs to train the network, is *n*. Each component of the input vector is connected to every individual neuron. Thus, there is a connection between the neuron of the network and every component of the input vector. The weights of these connections form an *n*-dimensional synaptic weight vector (the codebook vector, also called a reference, model, or parameter vector [Koh01a]). Thus, any neuron is entirely defined by its location on the grid (the number of row *i* and column *j*) and by the codebook vector, i.e., we can consider a neuron as an *n*-dimensional vector $m_{ij} = (m_{ij}^1, m_{ij}^2, ..., m_{ij}^n) \in R^n$. In this way, each vector (neuron) m_{ij} represents a part of R^n.

The learning starts from the vectors m_{ij} initialized randomly (other ways of initializing the vectors m_{ij} are possible, too). At each learning step, an input vector X is drawn from the training set $\{X_1, ..., X_s\}$ and passed to the neural network. The Euclidean distance from this input vector to each vector m_{ij} is calculated and the vector (neuron) $m_c \in \{m_{ij}, i = 1, ..., k_x, j = 1, ..., k_y\}$ with the minimal Euclidean distance to X is designated as a winner. Denote the row, where m_c is located, by i_c, and the column by j_c, i.e., c is a combination of two numbers: i_c and j_c. The components of the vector m_{ij} are adapted according to the rule $m_{ij} \leftarrow m_{ij} + h_{ij}^c (X - m_{ij})$, where h_{ij}^c is the learning rate, which is maximal for the winning neuron, and decreases with an increase in the neighbourhood order and learning steps.

Let us introduce a term "learning epoch". The epoch consists of *s* learning steps: the input vectors from X_1 to X_s are passed to the neural network in consecutive or random order. The consecutive order was used in [Dze01a]. Both the orders were examined in [Dze01b]. In this paper we use the random order, because we try to eliminate the influence of numeration of the input vectors on the learning process. The whole learning process consists of *v* epochs.

After a large number of learning steps, the network has been organized and *n*-dimensional input vectors $X_1, ..., X_s$ have been mapped – each input vector is related to the nearest neuron, i.e., the vectors are distributed among the elements of the map during training. Some elements of the map may remain unrelated with any vector from $\{X_1, ..., X_s\}$, but there may occur elements related with some vectors.

Using the SOM-based approach above we can draw a table with cells corresponding to the neurons. The cells corresponding to the neurons-winners are filled with the numbers of vectors $X_1, ..., X_s$. Some cells may remain empty. One can decide visually on the distribution of vectors $X_1, ..., X_s$ in the *n*-dimensional space R^n in accordance with their distribution among the cells of the table.

Combining Sammon's mapping with the self-organizing maps. The way of integration of Sammon's mapping and the SOM is presented in [Dze01a]. The self-organizing map provides structured information about the set of the analysed vectors: several elements (neurons) of the two-dimensional rectangular grid are activated (become winners), while the remaining elements are not activated. The activated elements of the grid may be considered as points on the plane. The number of row and column characterizes any of these elements, i.e., the location of these elements is fixed on the plane by the nodes of a rectangular grid. However, the elements are characterized by *n*-dimensional vectors, too. A natural idea arises to apply the distance-preserving projection method to additional mapping of vectors-winners $Z_1, Z_2, ..., Z_r$, corresponding to the neurons-winners, in the SOM [Kas97a]. Sammon's mapping may be used for such purposes. The combined algorithm is as follows: all input vectors $X_1, ..., X_s$ are first processed using the SOM; then the vectors-winners, whose number *r* is less or equal to *s*, are displayed using Sammon's mapping. In [Dze01a] such a combination of mapping methods has been examined and grounded experimentally by comparing the results of Sammon's mapping of the vectors, that correspond to some parameters characterized by their correlation matrix, and Sammon's mapping of the vectors-winners in the SOM.

Therefore, two scenarios of visualizing the *n*-dimensional vectors were analysed in [Dze01a] and [Dze01b]. They are given in Figures 1a and 1b: the original Sammon's algorithm and its consequent combination with the SOM.

The third scenario is presented in Figure 1c. It is a new combination of the self-organizing map and Sammon's algorithm. The experiments have showed that namely this combination of SOM and Sammon's mapping is very good in search for a more precise projection of multidimensional vectors

in the sense of criterion E_S (1), when vectors, corresponding to the neurons-winners of the SOM, are analysed.

3. A NEW ALGORITHM

We suggest the following way of integrating the SOM and Sammon's algorithm:

1. The training set consists of *s* *n*-dimensional vectors $X_1, X_2, ..., X_s$. The neural network will be trained using *e* learning epochs.

2. All the *e* epochs are divided into equal training parts – blocks. Before the training of the neural network starts, we choose the number γ of blocks into which the training process will be divided. Each block contains *p* training epochs ($p = e \ div \ \gamma$). Denote by *q* a block of the training process consisting of *p* epochs. $q = 1, \ldots, \gamma$.

3. Denote vectors-winners obtained by the *q*-th block of the training process by $Z_1^q, Z_2^q, ..., Z_{r_q}^q$ and two-dimensional projections of these vectors-winners, calculated using Sammon's algorithm, by $Y_1^q, Y_2^q, ..., Y_{r_q}^q$ ($Y_i^q = (y_{i1}^q, y_{i2}^q)$, $i = 1, ..., r_q$). Note that the number r_q of vectors-winners will be smaller or equal to *s*. The vectors-winners $Z_1^1, Z_2^1, ..., Z_{r_q}^1$, obtained after the first block of the training process (q=1), are analyzed by using Sammon's algorithm. However, there is a unique relation between a vector-winner and the corresponding vector (or several vectors) from the training set $\{X_1, X_2, ..., X_s\}$. The initial coordinates of two-dimensional vectors $Y_i^0 = (y_{i1}^0, y_{i2}^0)$, $i = 1, ..., r_1$, for Sammon's algorithm are set as follows: $y_{i1}^0 = i + \frac{1}{3}$, $y_{i2}^0 = i + \frac{2}{3}$. Two-dimensional projections $Y_1^1, Y_2^1, ..., Y_{r_1}^1$ of vectors-winners are calculated using Sammon's algorithm.

4. The vectors-winners obtained after the *q*-th block of the training process are analyzed by using Sammon's algorithm. The initial coordinates of two-dimensional vectors $Y_1^q, Y_2^q, ..., Y_{r_q}^q$ for Sammon's algorithm are selected taking into account the result of the (*q*-1)-st block. Note that $r_q \neq r_{q-1}$ in general. A way of the selection is presented below. We must determine the initial coordinates of each two-dimensional vector Y_i^q correspondent to the neuron-winner Z_i^q, $i = 1, ..., r_q$. The sequence of steps is as follows. Determine vectors from $\{X_1, X_2, ..., X_s\}$ that are related with Z_i^q. Denote these vectors by $X_{i_1}, X_{i_2}, ...$ ($X_{i_1}, X_{i_2}, ... \in \{X_1, X_2, ..., X_s\}$). Determine neurons-winners of the (*q*-1)-st block that were related with $X_{i_1}, X_{i_2}, ...$ Denote these neurons-winners by $Z_{j_1}^{q-1}, Z_{j_2}^{q-1}, ...$ ($Z_{j_1}^{q-1}, Z_{j_2}^{q-1}, ... \in \{Z_1^{q-1}, Z_2^{q-1}, .., Z_{r_{q-1}}^{q-1}\}$), and their two-dimensional projections, obtained in a result of Sammon's algorithm, by $Y_{j_1}^{q-1}, Y_{j_2}^{q-1}, ...$ ($Y_{j_1}^{q-1}, Y_{j_2}^{q-1}, ... \in \{Y_1^{q-1}, Y_2^{q-1}, ..., Y_{r_{q-1}}^{q-1}\}$). The initial coordinates of Y_i^q are set to be equal to the mean value of the set of vectors $\{Y_{j_1}^{q-1}, Y_{j_2}^{q-1}, ...\}$. Afterwards, two-dimensional projections $Y_1^q, Y_2^q, ..., Y_{r_q}^q$ ($Y_i^q = (y_{i1}^q, y_{i2}^q)$, $i = 1, ..., r_q$) of the vectors-winners are calculated using Sammon's algorithm.

5. The training of the neural network is continued untill $q=\gamma$. After γ-th block we get two-dimensional projections $Y_1^\gamma, Y_2^\gamma, ..., Y_{r_\gamma}^\gamma$ of the *n*-dimensional vectors-winners $Z_1^\gamma, Z_2^\gamma, ..., Z_{r_\gamma}^\gamma$ that are uniquely related with vectors $X_1, X_2, ..., X_s$ (see Section 2).

4. STRATEGY OF INVESTIGATION

Our purpose was to examine the mean projection error by the new algorithm compared with that obtained by the algorithm 1b [Dze01a] in dependance on the "magic factor" α (2). Therefore, the experiments were carried out using different values of the "magic factor" α.

The projection error E_S is minimized by a gradient method (2). With an increase in the order number of the iteration, the projection error decreases. But sometimes the error may vary, i.e., it decreases, increases, and decreases again [Apo99a]. It may increase in the last iteration. Therefore, we fix the least projection error over all the iterations as a final result.

The projection errors can differ for various sets of the initial values of neurons, because they are generated at random. Thus, much higher or much lower projection error, that is of a random nature, can be obtained both by the new algorithm and by

algorithm 1b. To avoid that, the experiments have been carried out 200 times with different, randomly generated sets of the initial values of neurons. Then the results have been averaged. Therefore, we get the so-called mean projection error for a fixed value of α.

5. RESULTS OF ANALYSIS

The advantages of the new algorithm, proposed in Section 3 and given in Figure 1c, in comparison with algorithm 1b [Dze01a] have been shown analyzing the data on coastal dunes and their vegetation in Finland [Hel98a]. The following parameters a_1 - a_{16} characterize the dunes: a_1 is the distance from the water line; a_2 is the height above the sea level; a_3 is the soil PH; a_4, a_5, a_6, and a_7 are the contents of calcium (CA), phosphorous (P), potassium (K), magnesium (Mg); a_8 and a_9 are the mean diameter and sorting of sand; a_{10} is the northernness in the Finnish coordinate system; a_{11} is the rate of land uplift; a_{12} is the sea level fluctuation; a_{13} is the soil moisture content; a_{14} is the slope tangent; a_{15} is the proportion of bare sand surface; a_{16} is the tree cover.

The correlation matrix $R = \{r_{a_i a_j}, i, j = 1,...,s\}$ of these sixteen parameters is given in [Hel98a]. Using the method developed by Dzemyda [Dze01a], sixteen vectors $X_1, X_2, ..., X_s$, s=16, of unit length have been computed. Their dimension n is equal to 16. The values of the vectors are presented in [Dze02a]. These vectors correspond to the parameters a_1 - a_{16}. Namely, they are used during the experiments.

Cases with various parameters of the proposed algorithm and its constituent parts have been analyzed:

- size of neural network (2x2, 3x3, 4x4, 5x5, 6x6);
- number of training epochs e (100, 200, 300);
- number γ of training blocks and number p of epochs per each training block (e= $p\gamma$);
- values of the "magic factor" α in Sammon's mapping (0.1; 0.11;…; 1.99; 2).

Under the same initial conditions, the errors of projection have been calculated for all the parameters referred above by using both 1b and the new algorithm. As mentioned above, the experiments have been repeated 200 times with different (random) initial values of the components of the neurons-vectors. The ratio between the mean projection errors, obtained by both 1b and the new algorithm, has been calculated. It appears from Table 1 and Figure 2, that this ratio is always greater than one. Thus, the mean projection errors, obtained by the new algorithm, are smaller. When increasing the number γ of the training blocks (Figure 2), this ratio increases: essentially when the SOM is of a smaller size. The ratio decreases with an increase in the network size.

Figures 3 and 4 show that the mean projection error, obtained by the new algorithm, depends much less on the value of the "magic factor" α in comparison with algorithm 1b. Figures 3 and 4 illustrate four cases, however, the similar results are observed in the most of remaining cases. This is the essential advantage of the new algorithm, i.e., if we need to visualize the neurons-winners of the SOM, we succeed to eliminate the influence of the "magic factor" α on Sammon's mapping results. This conclusion is true for $0 < \alpha \le 1$. For larger values of α, the mean projection error grows. However, in this case the new algorithm operates more stable than algorithm 1b, and it gives smaller values of the mean projection error.

Figure 5 illustrates output of both the algorithms (projection of the multidimensional test data on a plane). We do not present scales of variables, because we are interested in observing the interlocation of points on a plane. Dimension of the SOM is 6x6, number of epochs is 200, number γ of training blocks is 40. Following the recommendation in [Koh01a], the value of α has been selected equal to 0.35. The visually presented distributions of the points are quite different. This proves the necessity to make every effort for minimization of the distortion of projection E_s (1).

6. CONCLUSIONS

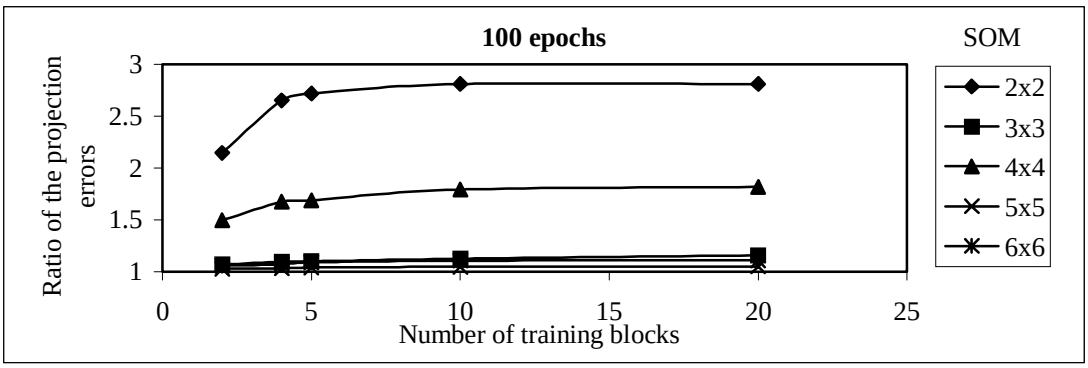
When comparing the mean projection error, obtained by using the combination of the SOM and Sammon's mapping (algorithm 1b), with that by the new algorithm that takes into account the learning flow of the self-organizing neural network, we see lower projection errors in the results got by the new algorithm. A larger number γ of training blocks decreases the mean projection error. However, that needs much more computing time.

The main result of this paper is that, if we need to visualize the neurons-winners of the SOM, we have a strategy how to eliminate to a certain extent the influence of the "magic factor" α on Sammon's mapping results, i.e., the mean projection error, obtained by the new algorithm, depends much less on the value of the "magic factor" than that obtained by algorithm 1b.

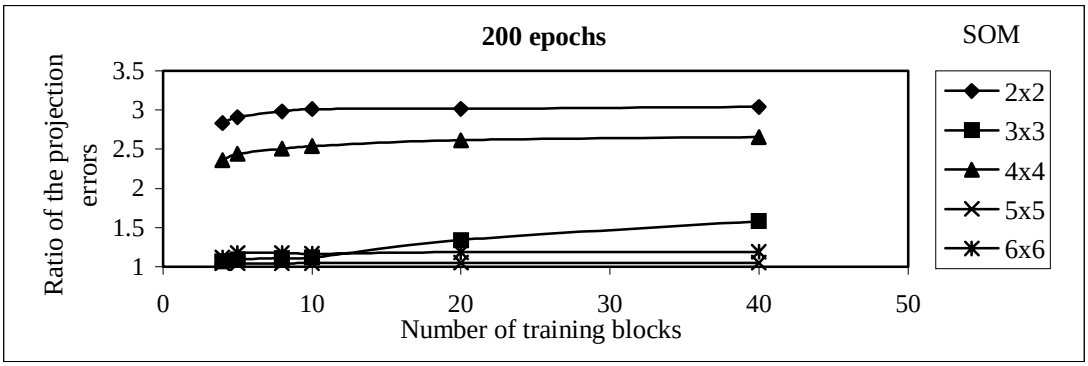
e	100					200						300				
p	**50**	**25**	**20**	**10**	**5**	**50**	**40**	**25**	**20**	**10**	**5**	**50**	**25**	**20**	**10**	**5**
γ	**2**	**4**	**5**	**10**	**20**	**4**	**5**	**8**	**10**	**20**	**40**	**6**	**12**	**15**	**30**	**60**
2x2	2.15	2.65	2.72	2.81	2.81	2.84	2.91	2.99	3.01	3.02	3.04	3.09	3.16	3.17	3.16	3.19
3x3	1.07	1.1	1.1	1.13	1.16	1.07	1.09	1.11	1.11	1.34	1.58	1.09	1.11	1.12	1.14	1.16
4x4	1.5	1.67	1.69	1.79	1.82	2.36	2.44	2.51	2.54	2.62	2.66	3.32	3.47	3.5	3.58	3.59
5x5	1.03	1.03	1.04	1.05	1.05	1.04	1.04	1.04	1.05	1.05	1.05	1.04	1.05	1.04	1.04	1.05
6x6	1.06	1.07	1.09	1.1	1.11	1.12	1.17	1.18	1.17	1.19	1.19	1.27	1.29	1.29	1.3	1.3

Table 1. The ratio between the projection errors obtained by algorithm 1b and the new algorithm

a)



b)



c)

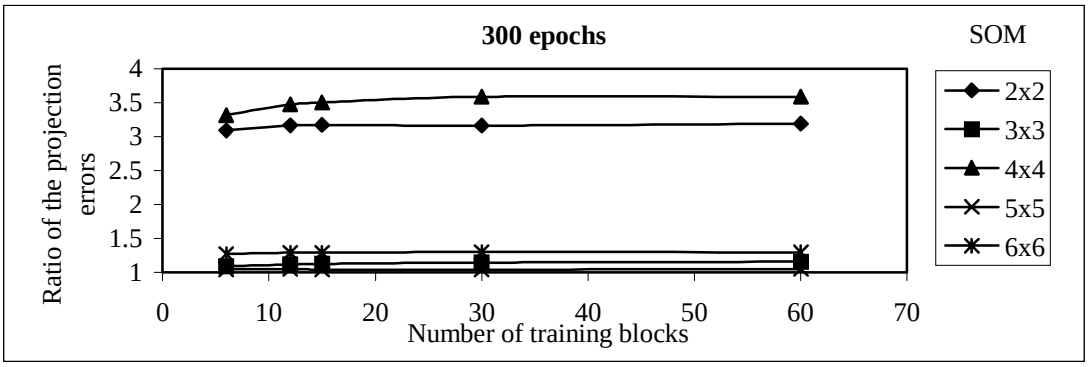


Figure 2. Ratio of the projection errors for different number of training blocks *γ*

7. REFERENCES

[Apo99a] Apostal, I., and Szpankowski, W. Indexing and mapping of proteins by Sammon's projection algorithm. Journal of Computational Chemistry, 20, pp.1049-1059, 1999.

[Dze01a] Dzemyda, G. Visualization of a set of parameters characterized by their correlation matrix. Computational Statistics and Data Analysis, Vol. 36, No. 1, pp. 15-30, 2001.

[Dze01b] Dzemyda, G., Tiešis, V. Visualization of multidimensional objects and the socio-economical impact to activity in EC RTD databases. Informatica, No. 12(2), pp. 239-262, 2001.

[Dze02a] Dzemyda, G., Kurasova, O. Comparative analysis of the graphical result presentation in the SOM software. Informatica, No. 13(3), pp. 275-286, 2002.

[Hel98a] Hellemaa, P. The Development of Coastal Dunes and Their Vegetation in Finland. Dissertation. Fenia 176:1, Helsinki. ISSN 0015-0010, 1998. http://ethesis.helsinki.fi/julkaisut/mat/maant/vk/hellemaa/index.html

[Kas97a] Kaski, S. Data exploration using self-organizing maps. In Acta Polytechnica Scandinavica, Mathematics, Computing and Management in Engineering Series, No. 82 Espoo: Finish Academy of Technology, 1997. http://www.cis.hut.fi/~sami/thesis/thesis_tohtml.html

[Koh01a] Kohonen, T. Self-Organizing Maps, 3nd ed., Springer Series in Information Sciences, Vol. 30, Springer-Verlag, 2001.

[Sam69a] Sammon, J.W. A nonlinear mapping for data structure analysis. IEEE Transactions on Computers, Vol. C-18, pp. 401–409, 1969.

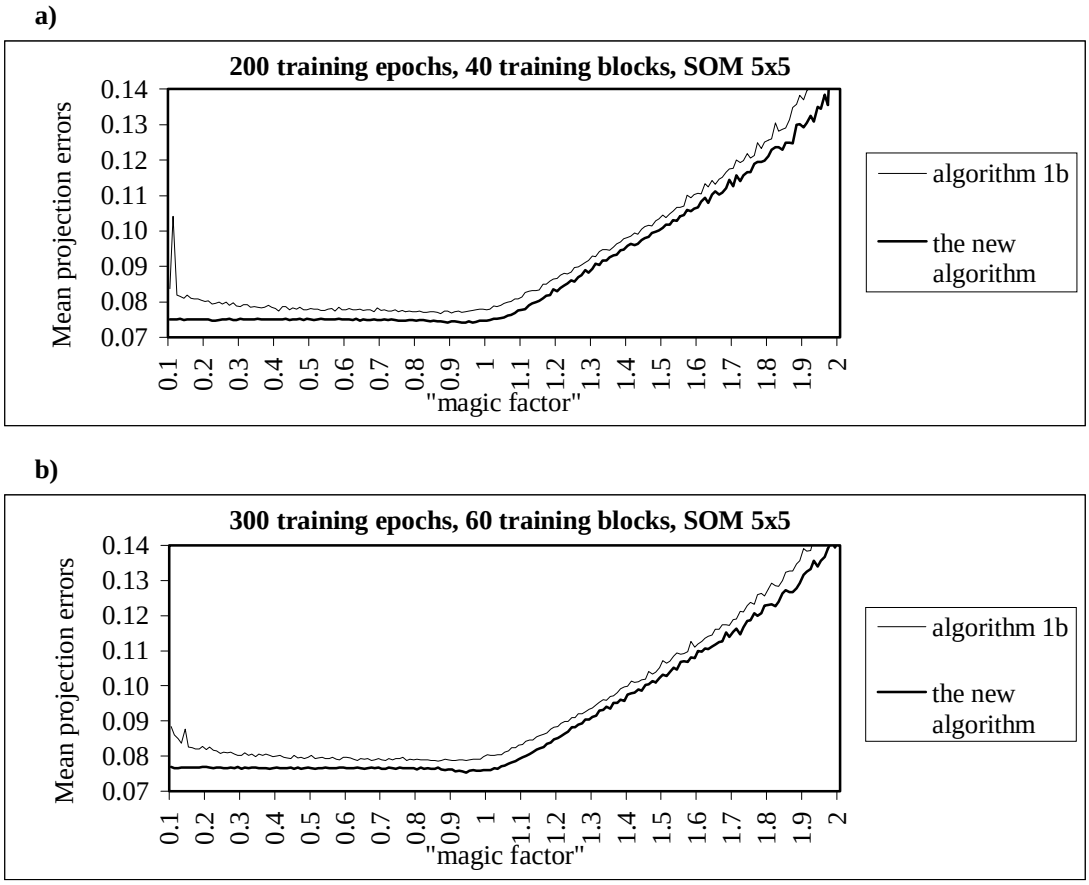
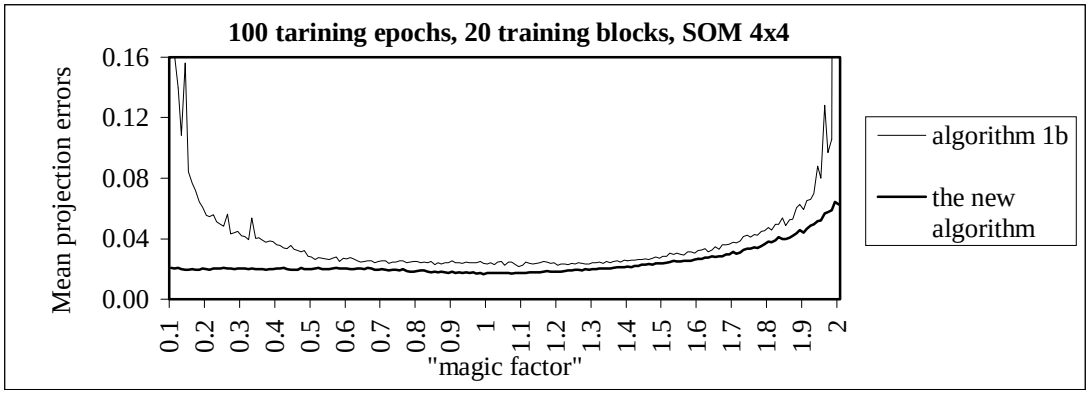


Figure 3. Dependence of the projection error on the "magic factor" α

a)



b)

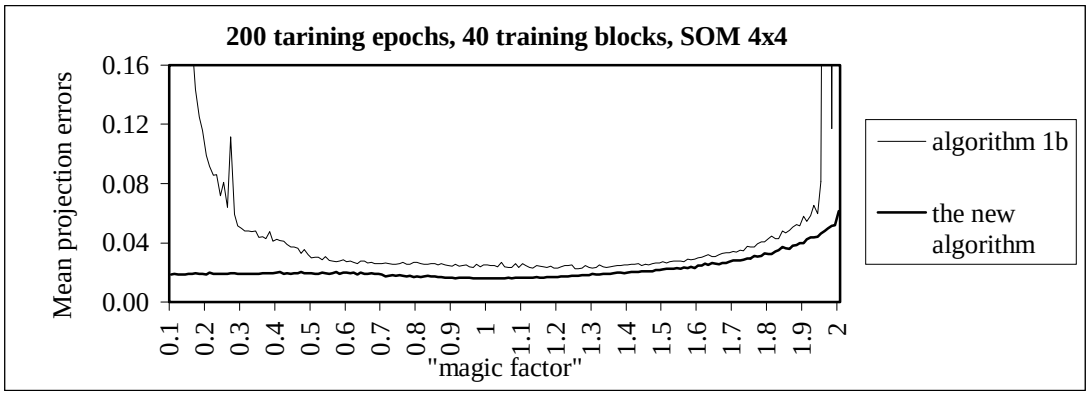
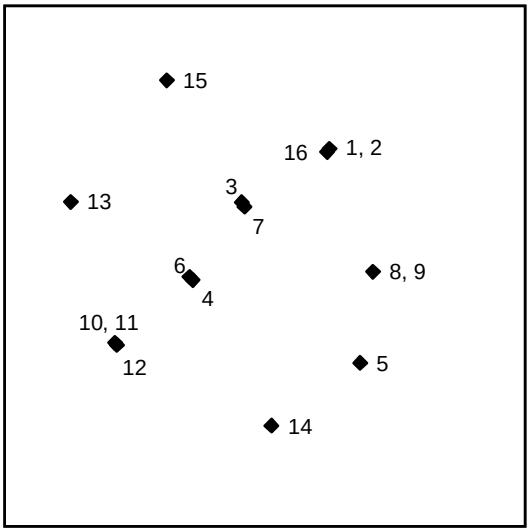


Figure 4. Dependence of the mean projection error on the "magic factor" α

a)



b)

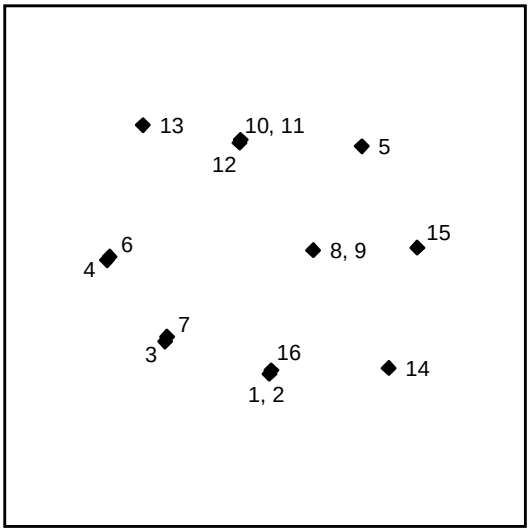


Figure 5. Examples of visualization:
a) algorithm 1b (α **=0.35,** E_S **= 0.0890), b) the new algorithm (** α **=0.35,** E_S **= 0.0764)**