Baltic J. Modern Computing, Vol. 2 (2014), No. 2, 45-55

Investigation on Learning Parameters of Self-Organizing Maps

Pavel STEFANOVIČ, Olga KURASOVA

Vilnius University, Institute of Mathematics and Informatics,
Akademijos St. 4, LT 08663, Vilnius, Lithuania

pavel.stefanovic@mii.vu.lt, olga.kurasova@mii.vu.lt

Abstract: In the paper, the influence of learning parameters on self-organizing map (SOM) is analyzed, when both numerical data and text documents are investigated. Three neighboring functions (bubble, Gaussian, and heuristic) and four learning rates (linear, inverse-of-time, power series, and heuristic) have been investigated. The learning rates are changed according to epochs or iterations. The quality of self-organizing map is measured not only by quantization error, but also by two other measures, which are suitable when the classified data are analyzed.

Keywords: self-organizing map, neighboring function, learning rate, text document matrix, SOM quality estimators.

1. Introduction

Past more than thirty years when self-organizing map (SOM) was introduced (Kohonen, 2001), but SOM has been still widely used for data classification, clustering, and visualization. Over the past decade, a lot of self-organizing map modifications has been developed, but mostly all of them are created for specific tasks. For example, the environmental self-organizing maps (EnvSOM) are used for a large dataset analysis, where self-organizing maps are influenced by environment conditions (Alonso *et al.*, 2011). The batch-learning self-organizing maps (BLSOM) are used for the large data analysis in bioinformatics area (Iwasaki *et al.*, 2013). There are many other modifications and extensions of the traditional SOM algorithm, such like the recursive self-organizing map (Voegtlin, 2002), the growing hierarchical self-organizing map (Dittenbach *et al.*, 2000), the merge self-organizing map (Hammer *et. al*, 2004), WEBSOM (Lagus *et al.*, 2004), the self-organizing map for structured data (Hagenbuchner *et al.*, 2003), and more other. For a long-time, the self-organizing map has been used just for a statistical data analysis, but nowadays it is used for different kinds of data: audio (Mayer, 2011), video (Barecke *et al.*, 2006), images (Ishtiaq *et al.*, 2009), text information (Kohonen and Xing, 2011), etc.

The results of self-organizing maps depend on learning parameters, such like neighboring functions, learning rates, map size, etc. The main goal of this research is to analyze how the self-organizing map learning parameters affect the results of the trained SOM, when both numerical data and text documents are investigated. The quality of SOM is measured according not only quantization error, but also the measures purposed in (Stefanovič and Kurasova, 2014). In our preview work (Stefanovič and Kurasova, 2011a), we has analyzed the influence of neighboring functions and learning rates on SOM results, too. However, there the experiments have been carried out just with numerical datasets and only a quantization error has been estimated. In this paper, we analyze and compare results when different kinds of datasets are used: text documents and numerical datasets. Particularity of data from text documents, comparing to numerical data, is that usually dimensionality of the data is high, moreover, the number of data features is larger than that of data items. Therefore, it is important to explore the influence of learning parameters on SOM results, when both numerical data and text documents are analyzed.

2. Self-organizing map

The self-organizing map is a type of artificial neural network, which is learned by an unsupervised manner. The self-organizing map (Kohonen, 2001) is a set of nodes, connected to each other via a rectangular or hexagonal topology. The connections between the inputs and the nodes have weights, so a set of weights corresponds to each node. The set of weights forms a vector $M_{ij}, i = 1, \dots, k_x, j = 1, \dots, k_y$ that is usually called a neuron or a codebook vector, where k_x is the number of rows, and k_y is the number of columns (in the case of a rectangular topology). The main aim of the SOM is to preserve the topology of multidimensional data when they are transformed into lower dimensional space. Usually, the self-organizing maps are used to cluster, classify, and visualize different kinds of datasets. The self-organizing maps can deal only with numerical data, so first, we have to transform any kinds of datasets to numerical expression. Suppose, we have a dataset $X = \{X_1, X_2, \dots, X_N\}$, where each data item is described by the features $x_1, x_2, \dots, x_n$, i. e. $X_p = (x_{p1}, x_{p2}, \dots, x_{pn})$. So, X_p is a point (or a vector) in n-dimensional space, $X_p \in R^n$. All datasets are given to SOM as a matrix:

$$\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N1} & x_{N2} & \cdots & x_{Nn} \end{pmatrix} \tag{1}$$

Here x_{pl} is the value of the lth component of the vector $X_p, p = 1, \dots, N,\ l = 1, \dots, n$. N is the number of analyzed input vectors, and n is the number of components. The learning process of the SOM algorithm starts from initialization of the components of the vectors (neurons) M_{ij}. They can be initialized at random (usually these values are random numbers from the interval (0, 1)) or by the principal components. At each learning step, an input vector $X_p \in \{X_1, X_2, \dots, X_N\}$ is passed to the SOM. The vector X_p is compared with all neurons M_{ij}. Usually the Euclidean distance between this input vector X_p and each neuron M_{ij} are calculated. The vector (neuron) M_w with the minimal Euclidean distance to X_p is designated as a neuron winner (best match unit). All neurons components are adapted according to the learning rule:

$$M_{ij}(t+1) = M_{ij}(t) + h_{ij}^{w}(X_p - M_{ij}(t)) \tag{2}$$

Here t is the number of learning step, h_{ij}^{w} is a neighboring function, w is a pair of indexes of the neuron winner of vector X_p. The learning is repeated until the maximum number of learning step T is reached.

2.1. Learning parameters

The results of self-organizing map depend on the selected learning parameters. So, it is important to choose the best learning parameters to get the better results. The results mostly are affected by different neighboring functions h_{ij}^{w} (Table 1) and learning rates $\alpha(t)$ (Table 2). Usually, two neighboring functions – bubble (3) and Gaussian (4) are used. In our research, we analyze one more neighboring function, so-called heuristic (5), introduced in (Stefanovič and Kurasova, 2011a).

Table 1. Neighboring functions

Bubble (3)	**Gaussian (4)**	**Heuristic (5)**
$h_{ij}^{w} = \begin{cases} \alpha(t), (i,j) \in N_w \\ 0, (i,j) \in N_w \end{cases}$	$h_{ij}^{w} = \alpha(t) \cdot \exp\left(\frac{-\lVert R_w - R_{ij} \rVert^2}{2\left(\eta_{ij}^{w}(t)\right)^2}\right)$	$h_{ij}^{w} = \frac{\alpha(t)}{\alpha(t) \cdot \eta_{ij}^{w} + 1}$

In Table 1, N_w is the index set of neighboring nodes around the node with indexes w. Two-dimensional vectors R_w and R_{ij} consist of indexes of M_w and M_{ij}, respectively. The indexes show a place of the neuron-winner, the codebook vector of which is M_w, for the vector X_p and that of the neuron, the codebook vector of which is M_{ij}, in SOM. The parameter η_{ij}^{w} is the neighboring rank of M_{ij} according to M_w. As it was mentioned before, the learning rate $\alpha(t)$ also influences the results of self-organizing map. Usually linear (6), inverse-of-time (7), and power series (8) learning rates are used for SOM training. In our investigation, we analyze one more learning rate, so-called heuristic (9). Four variants of learning rates are presented in Table 2.

Table 2. Learning rates

Linear (6)	**Inverse-of-time (7)**	**Power series (8)**	**Heuristic (9)**
$\alpha(t) = \frac{1}{t}$	$\alpha(t) = \left(1 - \frac{t}{T}\right)$	$\alpha(t) = (0.005)^{\frac{t}{T}}$	$\alpha(t) = max\left(\frac{T+1-t}{T}, 0.01\right)$

In this paper, two cases are investigated:

- When the learning rate $\alpha(t)$ depends on the iteration number (in this case, t is the order number of the current iteration and T is the total number of iterations). One iteration is part of the training process, when one input vector is passed to the network and the neurons are changed.
- When the learning rate $\alpha(t)$ depends on the epoch number (in this case, t is the order number of the current epoch and T is the total number of epochs). An epoch is part of the training process, when all vectors of the training dataset are passed to the network at once.

2.2. SOM quality estimators

One of self-organizing map advantages is visualization of datasets, so after SOM is trained, it is important to evaluate the quality of the map. Usually a quantization error E_{QE} (10) (Table 3) is calculated. E_{QE} is the averaged distance between the vectors X_p and the codebook vectors $M_{w(p)}$ of their neurons-winners. It shows how well the codebook vectors of neurons of the trained SOM adapt to the input vectors X_p, $p = 1, \dots, N$.

Table 3. Quality estimators of self-organizing map

Estimated errors	**Formula**
Quantization error (10)	$E_{\mathrm{QE}} = \frac{1}{N}\sum_{p=1}^{N}\lVert X_p - M_{w(p)}\rVert$
Error between same class members (11)	$E_c = \frac{1}{N_c}\sum_{i=1}^{n_c-1}\sum_{j=1}^{n_c}\left(\lVert Z_i^c - Z_j^c\rVert k_i^c k_j^c + b\right)$, where $b = \frac{l_i^{c'}}{k_i} + \frac{l_j^{c'}}{k_j}$
Error between different class centers (12)	$E_{\mathrm{center}} = \frac{1}{m}\sum_{c=1}^{m-1}\sum_{d=c+1}^{m}\lVert Y^c - Y^d\rVert$, where $Y^c = \frac{1}{n_c}\sum_{i=1}^{n_c} Z_i^c$

In the paper, SOM quality is estimated not only by the quantization error, but also by two other measures, proposed in (Stefanovič and Kurasova, 2014) and used for classified data. The measures shows how well classified data correspond to clusters on SOM. The first proposed measure E_c (11) is the Euclidean distance between indexes of all the SOM cells, corresponding to the data from same class. The measures E_c are suitable, when we want to see which classes are closer to each other in the maps and how classes match to them clusters. Here c is a class label, $c = 1, 2, \dots, m$, m is the number of classes; N_c is the number of data items from the cth class; n_c is the total number of neurons corresponding to the data from the cth class; $Z^c = \{Z_1^c, Z_2^c, \dots, Z_{n_c}^c\}$ is a vector, consisting of indexes of the SOM cells, corresponding to the data from the cth class; k_i^c is the number of the data items from the cth class in the SOM cell, the indexes of which are Z_i^c. b is a penalty. k_i is the number of the data items in the SOM cell, the indexes of which are Z_i^c; $l_i^{c'}$ is the numbers of data items from other classes than the cth class in the SOM cell, the indexes of which are Z_i^c. In this case, the smaller value of measure E_c mean the better results, it means that all same class members are closer to each other.

The measure E_{center} (12) is a distance between the centers of indexes of SOM cells, corresponding to data items from each class. Y^c is the center of indexes of SOM cells corresponding to the data items from the cth class. In this case, the bigger value of measure E_{center} means the better results, i.e. all the different class centers are far from each other, so on the map they are separated.

3. Experimental investigation

It is important to investigate how various learning parameters affect the results of SOM with different kinds of datasets. In this research, we analyze two datasets: text documents and numerical datasets. To analyze text document dataset by self-organizing map, first, we have to convert text information to numerical expression.

3.1. Text document conversation to numerical expression

At first, so-called text document dictionary has to be created. In this dictionary, all the words are included from the document dataset according to the control factors: word length, word frequency, remove alphanumeric and numbers, etc. It is important to choose proper control parameters. For example, if we choose to big word frequency, we can lose some documents, in the case, if the document do not have such words. When we have the text document dictionary, we can create text document matrix, the form of which is as in (1). One raw of the matrix corresponds to a document. x_{pl} is the frequency of the lth word in the pth text document, $p = 1, \ldots, N$, $l = 1, \ldots, n$. N is the number of the analyzed text documents, and n is the number of words in the text document dictionary. Dimensionality n of the vectors $X_p = (x_{p1}, x_{p2}, \ldots, x_{pn}), p = 1, \ldots, N$ depends on the number of the words in the dictionary.

There are different tools, which help to construct the text document matrix from documents dataset. Such popular systems like KNIME (Berthold et al, 2007) also can be used for text document analysis. In our research, we use a text to matrix generator (TMG) toolbox for Matlab to get a bag-of-words and create a text document matrix (Zeimpekis and Gallopoulos, 2005).

3.2. Dataset analyzed

The experiments have been carried out using five text document datasets, which we take from the document database of Seimas of the Republic of Lithuania (LRS, 2013). Fifteen documents of similar length from four different areas have been selected at random: the Ministry of Health, the Ministry of Education, the Ministry of Internal Affairs, and the Ministry of Agriculture.

All 60 documents are converted to numerical expressions. When creating text document matrix, some control factors are fixed. The numbers and alphanumeric are removed, because this information is not informative and do not define the documents. Primary research has showed that the total number of frequencies for this dataset is five, because, if we use the bigger number, some documents do not have such words, which repeated five times, and the documents simply rejected. Therefore, we create five different text document matrixes which have 60 rows and the different number of columns: 3812 (when frequency equal to 1), 1494 (when frequency equal to 2), 769 (when frequency equal to 3), 446 (when frequency equal to 4), and 287 (when frequency equal to 5).

Two numerical datasets are also analyzed: glass and zoological (zoo). The glass dataset was collected by a scientist, which wanted to help criminalists to recognize glass slivers found (Asuncion and Newman, 2007). Nine-dimensional vectors $X_1, X_2, \ldots, X_{214}$ are formed, where $X_p = (x_{p1}, x_{p2}, \ldots, x_{p9})$, $p = 1, \ldots, 214$. The dataset has been separated to five classes. The zoological dataset consists of 16 Boolean values. Sixteen-dimensional vectors $X_1, X_2, \ldots, X_{92}$ are formed, where $X_p = (x_{p1}, x_{p2}, \ldots, x_{p16}), p = 1, \ldots, 92$. The dataset has five different classes.

3.3. Experimental results

The system, in which the SOM algorithm is implemented in Matlab (Stefanovič and Kurasova, 2011b), is used in the experimental investigation. In this system, we can choose any neighboring functions from Table 1 and any learning rate from Table 2. The possibility to choose according to what (epochs or iterations) learning rate will be changed is implemented, too. The primary research has shown, that the size of the map and bigger epoch or iteration number do not affect the results essentially, so in all experiments, the map size is 10×10 and the epoch numbers are equal to 50. The number of epochs multiplied by the number of data items N corresponds to the number of the iterations. Each experiment is repeated 10 times with different initial values of neurons M_{ij}. The averages of the quantization error and all the other measures are calculated. The self-organizing map is trained using 80% of all the dataset, and the rest 20% of the dataset is used for testing in order to see how well the testing dataset adapts to the trained SOM.

First of all, an experiment was carried out using five text document datasets. The results of all the five text document matrix are averaged (when the frequency is from 1 to 5) and presented in Fig. 1. The numbers in brackets under columns correspond to the number of the learning rates in Table 2. The results, obtained in both cases of changing learning rate (according to epochs and iterations), using three neighboring functions, are presented in Fig. 1. As we can see in Fig. 1, the smallest quantization error was obtained, when we use the Gaussian neighboring function and the inverse-of-time learning rate (7) according to iterations (except for the linear learning rate (6) according to iterations) is used. The quantization error is also small when the Gaussian neighboring function and the heuristic learning rate (9) is used according to epochs and iterations. The worst results are obtained, when the bubble neighboring function is used. When the heuristic neighboring function is used, the quantization errors close to that, when the Gaussian neighboring function is used.

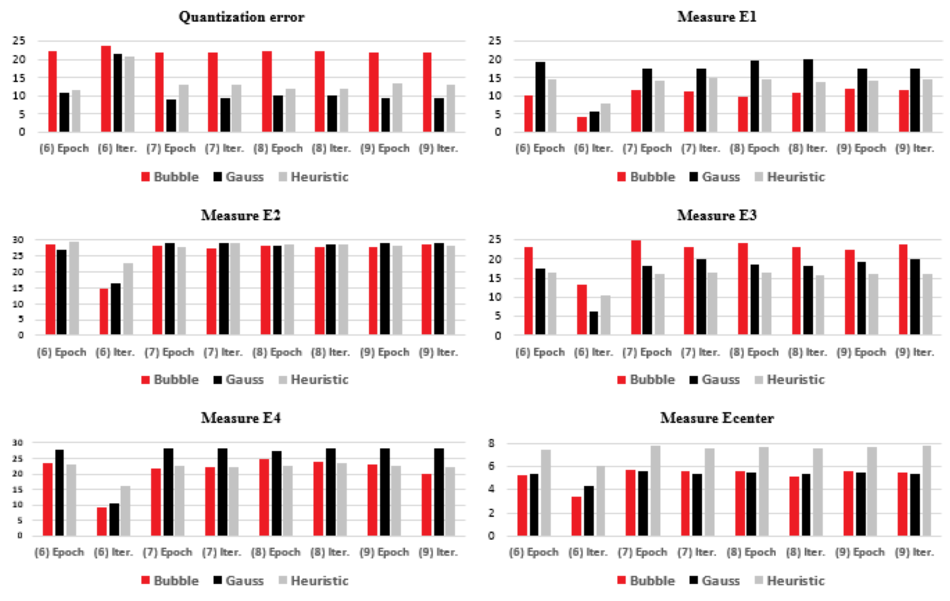

Figure 1. The averaged values of measures for text document training dataset

If we look to the results of measures E_c, we can see that, in the case of the first class (E_1), mostly all the smallest values of measure are obtained, when the bubble neighboring function is used, it means that the same class members are placed near in the map (the smaller results mean the better results). The values are also small, when the heuristic neighboring function is used. As we can see, the smallest values are obtained, when the linear learning rate (6) is used according to iterations. It is difficult to see the tendency of the E_2 values, because the results are similar using all the neighboring functions and learning rates. In the case of measure E_3, the smallest values are get, when the heuristic neighboring function is used (except for the linear learning rate (6) according to iterations). The results of measure E_4 are almost similar, when the bubble and heuristic neighboring function are used. The smallest results are also obtained, when the linear learning rate (6) according to iterations is used.

The biggest value of measure E_{center} is obtained, when the heuristic neighboring function and the heuristic learning rate (9) according to iterations is used. It means that the different class centers are far away from each other, so the different classes separate better. The results obtained, when the bubble and Gaussian neighboring functions are used, are very similar. The smallest results are obtained, when the heuristic (9) and inverse-of-time (7) learning rates are used according to epochs.

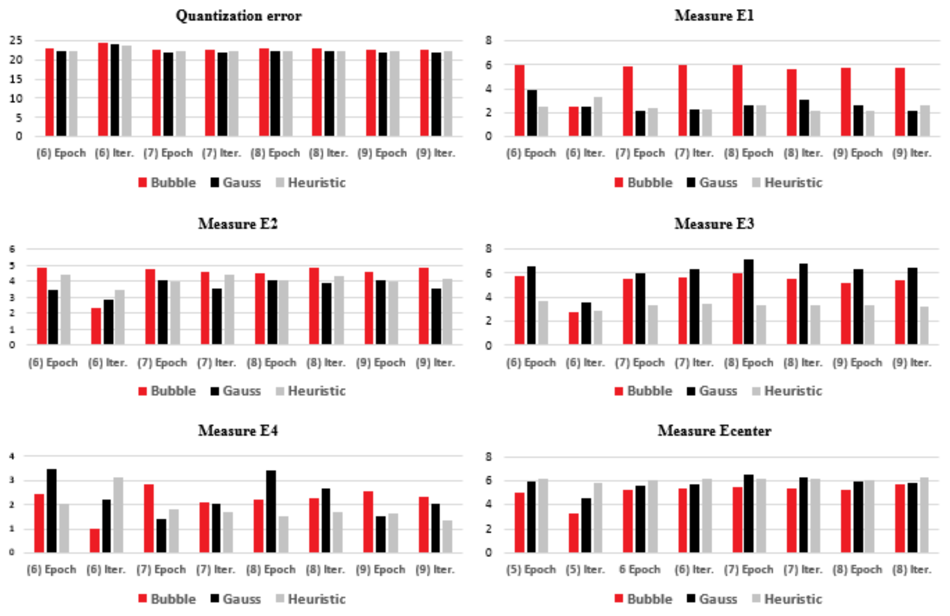

Figure 2. The averaged values of measures for text document testing dataset

The results of the testing data are presented in Fig. 2. The quantization error values are similar, so it is difficult to see some difference. In the cases of measures E_1 and E_2, the Gaussian and heuristic neighboring function also provide similar results, but in the most cases, the heuristic neighboring function gives slightly better results. The values of measure E_3 are smallest, when the heuristic function is used (except for the linear learning rate (6) according to iterations). The values of measure E_4 are various, so it is difficult to say which learning parameters give the better results. Almost in all the cases, the value of measure E_{center} are better, when the heuristic neighboring function is used. The best result is obtained, when the heuristic learning rate (9) according to iterations is used.

Concluding the results of the text document analysis, we can say that the best quantization error is obtained, when the Gaussian neighboring function and any learning rates according to epochs or iterations are used. In the case of measure E_{c}, the best results are get, when the heuristic neighboring function and the linear learning rate (6) according to iterations is used. In many cases, the worse results are obtained, when the Gaussian neighboring function is used. The results of measure E_{center} are the best, when the heuristic neighboring function and the heuristic learning rate (9) according to iterations is used.

Similar experiments have been made using glass and zoo datasets. The results of the quantization error and all the measures also are averaged and presented in Fig. 3. Almost in all the cases, the smallest quantization error is obtained, when the Gaussian neighboring function is used (except for the linear rate (6) according to iterations and the power series learning rate (8) according to epochs). The worst quantization error results are obtained, when the bubble neighboring function is used.

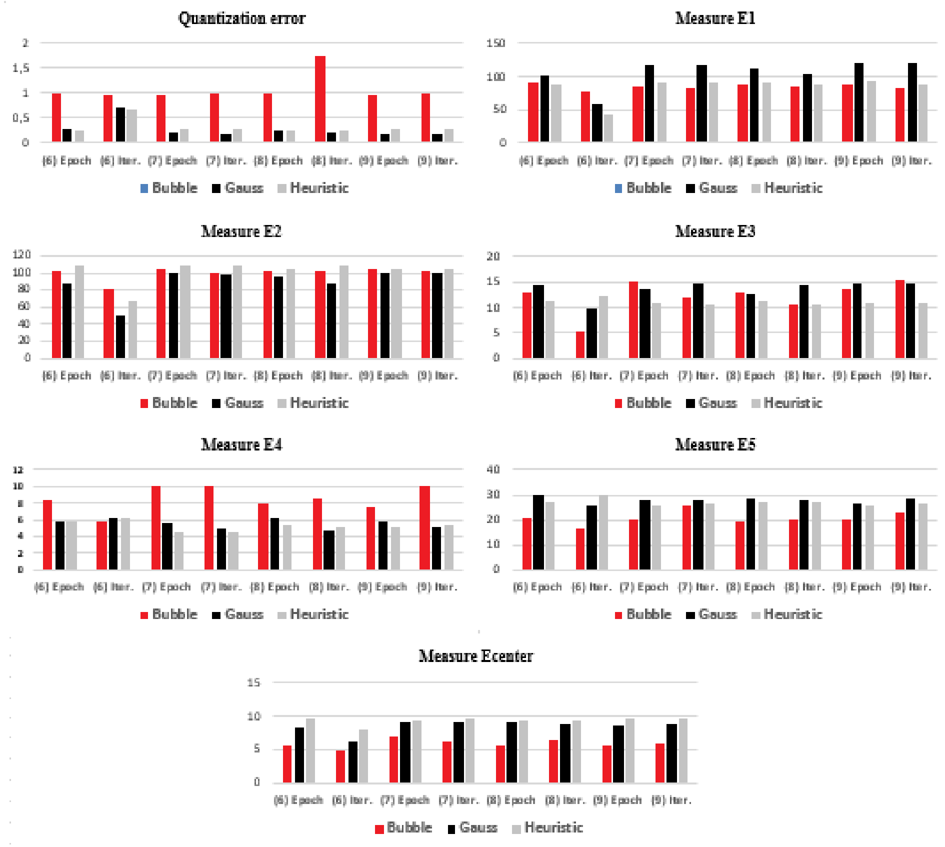

Figure 3. The averaged values of measures for glass and zoo training datasets.

The results for the first class E_1 are better, when the bubble and heuristic neighboring functions are used. The Gaussian neighboring function gives worse results. The values of measure E_2 are smallest, when the Gaussian neighboring function is used. The smallest value is obtained, when linear learning rate (6) is used according to iterations. The other neighboring functions give worse results. The best value of E_3 obtained, when the heuristic neighboring function is used (except for the linear learning rate (6) according to iterations). The bubble and Gaussian neighboring functions give worse results. In the case of measure E_4, the results obtained by the bubble and heuristic neighboring functions are almost same. The smallest measure is obtained, when the inverse-of-time learning rate (7) according to iterations or epochs are used. According to the measure E_5, the best results are get, when the bubble neighboring function is used. The values of the measure between different class centers E_{center} are better, when the heuristic neighboring function is used. The highest value is obtained, when the heuristic learning rate (9) according to iterations is used. The worst results are obtained, when the bubble neighboring function is used.

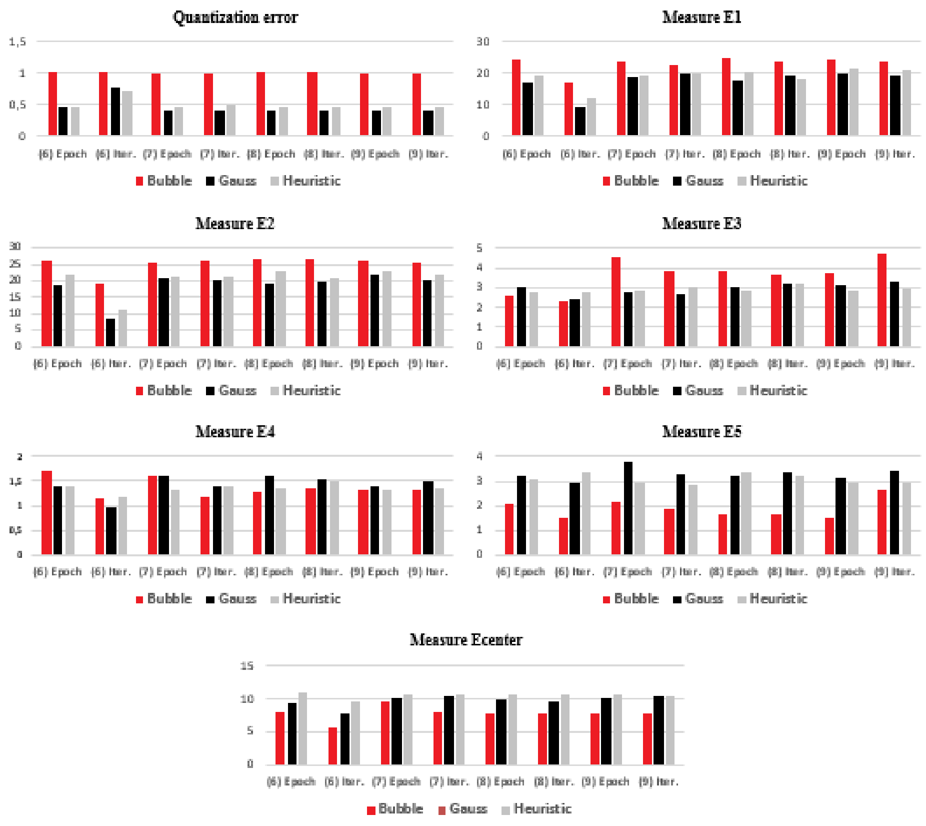

Figure 4. The averaged values of measures for glass and zoo testing datasets.

The result of the testing data presented in Fig. 4. In this case, the quantization errors are also the smallest, when the Gaussian neighboring function is used. The bubble neighboring function gives the highest value of measures E_1, E_2, E_3, and the smallest value of measure E_{center}. The Gaussian neighboring function gives the best results according to measures E_1 and E_2, but the worst ones are according to value of measures E_4 (except for the linear (6) and inverse-of-time (7) learning rates according to epochs) and E_5 (except for the power series learning rate (8) according to epochs). The heuristic neighboring function gives the best results with all learning rates in the case of measure E_{center}. So, like in other experiments, this heuristic function results show that the usage of this neighboring function better separate different class members in SOM.

The results of the glass and zoo data analysis also show that the best quantization error is obtained, when the Gaussian neighboring function is used. The heuristic neighboring function gives the best results, in the case of measure E_{center}. It is difficult to see differences in the case of measure E_c, because, in the different cases, various neighboring functions give various results.

4. Conclusions

In the paper, the influence of the learning parameters on the results of the self-organizing maps has been investigated, when two kinds of the datasets (numerical and text) are analyzed. Usually text document datasets differ from numerical ones in the number of features, characterizing the data. In the case of text document datasets, the number of the features can be very huge subject to the number of words in the dictionary. Moreover, the text document matrix is sparse, because not all the words from the dictionary are in every document. Therefore, it is purposeful to investigate SOM learning parameters for both numerical and text datasets. Moreover, it is purposeful to evaluate SOM results not only by quantization error, but also by the measures, which indicate how well the classified data corresponds to clusters in the SOM. The experimental investigation, when text documents, glass, and zoo datasets are analyzed by the self-organizing maps, confirms that learning parameters (neighboring functions and learning rates) impact on the SOM results. The results show that ones learning parameters are more suitable, when text datasets are analyzed and other ones, when numerical datasets are investigated.

In most cases, the quantization errors are smallest, when the Gaussian neighboring function and any learning rate is used. The quantization errors, obtained when the heuristic neighboring function is used, are close to that obtained when the Gaussian neighboring function is applied. The experimental investigation shows that the heuristic neighboring function is the most suitable, in many cases, not only for numerical data, but also for text document dataset, in sense of the measures, which evaluate how well the classified data corresponds to clusters in the SOM. When the heuristic neighboring function is used, it is appropriate to use the heuristic learning rate according to iterations. In this case, the values of E_{center} mostly always are higher, so it means that different classes are separated better for training and testing data. The smallest values of measure E_{center} are obtained, when the bubble neighboring function is used. The results of the Gaussian neighboring function are similar to the bubble neighboring function results, when text documents are analyzed, but in the case of numerical data, they are similar to the heuristic neighboring function results. It is difficult to make conclusions, which learning parameters give the best results of measure E_c, because the results are various and depend on the class number. The results weakly depend on how the learning rule is changed (according to epochs or iterations).

5. References

Asuncion, A., Newman, D. J. (2007). UCI Machine Learning Repository, Irvine, CA: University of California, School of Information and Computer. http://www.ics.uci.edu/~mlearn/MLRepository.html

Alonso, S., Sulkava, M., Prada, M. A., Domínguez, M., and Hollmén, J. (2011). EnvSOM: A SOM Algorithm Conditioned on the Environment for Clustering and Visualization. *WSOM 2011, LNCS 6731*, pp. 61–70. Springer-Verlag Berlin Heidelberg.

Barecke, T., Kijak, E., Nurnberger, A., and Detyniecki, M. (2006). Summarizing video information using self-organizing maps, *Proc. IEEE Int. Conf. Fuzzy Syst.*, pp. 540-546.

Berthold, M., Cebron, N., Dill, F., Kotter, T., and Meinl, T. (2007). KNIME: The Konstanz Information Miner. Studies in Classification, Data Analysis, and Knowledge Organization (GFKL 2007). Springer.

Dittenbach, M., Merkl, D., Rauber, A. (2000). The growing hierarchical self-organizing map. *IEEE - INNS - ENNS International Joint Conference on Neural Networks 6*, 6015.

Hagenbuchner, M., Sperduti, A., and Tsoi, A. C. (2003). A self-organizing map for adaptive processing of structured data. *IEEE Transactions on Neural Networks 14* (3), 491-505.

Hammer, B., Micheli, A., Sperduti, A., Strickert, M. (2004) A general framework for unsupervised processing of structured data. *Neurocomputing* 57, 3-35.

Ishtiaq, M., Jaffar, A., Hussain, A., Basit, A., Mirza, A. M. (2009). Wavelet Based Video Segmentation Using Self Organizing Map Neural Network, *Computer Science and Information Technology - Spring Conference, 2009. IACSITSC '09. International Association of,* pp. 122-125

Iwasaki, Y., Abe, T.,Wada, Y.,Wada, K., Ikemura, T. (2013). Novel bioinformatics strategies for prediction of directional sequence changes in influenza virus genomes and for surveillance of potentially hazardous strains. *BMC Infectious Diseases* 13(386).

Kohonen, T. (2001). *Self-organizing Maps, 3rd ed*., *Springer Series in Information Sciences*. Berlin: Springer-Verlag.

Kohonen, T., Xing, H. (2011). Contextually Self-Organized Maps of Chinese Words. In: J. Laaksonen, T. Honkela (Eds.) *Advances in Self-Organizing Maps – WSOM 2011, Lecture Notes in Computer Science*, Vol. 6731, Springer-Verlag, pp. 16–29.

Lagus, K., Kaski, S., Kohonen, T. (2004). Mining massive document collections by the WEBSOM method. *Information Sciences* 163(1-3), 135-156.

Mayer, R. (2011). Analysing the Similarity of Album Art with Self-organizing maps. In: J. Laaksonen, T. Honkela (Eds.) *Advances in Self-Organizing Maps – WSOM 2011, Lecture Notes in Computer Science*, Vol. 6731, Springer-Verlag, pp. 357–366.

Seimas of the Republic of Lithuania (2013). http://www3.lrs.lt/dokpaieska/forma_l.htm

Stefanovič, P., Kurasova, O. (2011a). Influence of Learning Rates and Neighboring Functions on Self-Organizing Maps. *WSOM 2011, LNCS 6731*, pp. 141-150. Springer-Verlag Berlin Heidelberg.

Stefanovič, P., Kurasova, O. (2011b). Visual analysis of self-organizing maps. *Nonlinear Analysis: Modeling and Control*, 16(4), p. 488–504.

Stefanovič, P., Kurasova, O. (2013). Similarity analysis of text documents by self-organizing maps and k-means. Informacijos mokslai. T. 65, p. 24–33.

Stefanovič, P., Kurasova, O. (2014). Creation of text document matrices and visualization by self-organizing map. *Information Technology and Control*, T. 43, Nr. 1, p. 37-46.

Voegtlin, T. (2002). Recursive self-organizing maps. *Neural Networks* 15 (8-9), 979-992.

Zeimpekis, D., Gallopoulos, E. (2005). TMG: A Matlab Toolbox for Generating Term-Document Matrices from Text Collections, Technical Report HPCLAB-SCG 1/01-05, University of Patras, GR-26500, Patras, Greece.

Received April 3, 2014, revised May 22, 2014, accepted May 29, 2014