Biomedical Data Analysis Using Self-Organizing Maps

Gráfová Lucie, Mareš Jan, Procházka Aleš
Institute of Chemical Technology, Faculty of Chemical Engineering
Department of Computing and Control Engineering
Technická 5, 166 28 Prague 6, Czech Republic

Abstract: Software introduces the application of self-organizing maps (SOM) to biomedical data analysis. The SOM algorithm was implemented in MATLAB environment with various optional parameters enabling the adjustment of model according to user's requirements. For easier application of SOM the graphical user interface was developed.

Keywords: biomedical image, self-organizing map

Software description

The software for biomedical data analysis using self-organizing map (SOM) has been developed, see Figure 1.

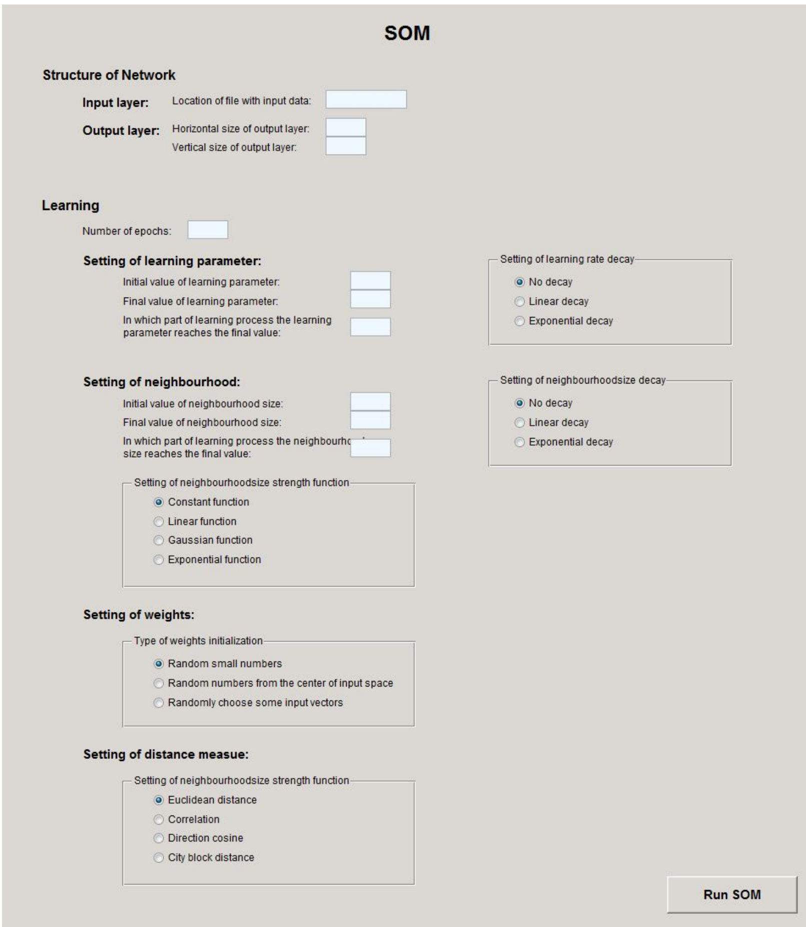

Figure 1. The software for biomedical data analysis using SOM

1 Basic settings

Firstly, the user has to set the structure of network, i.e. determine the location of the file with input data and define the size of the output layer of SOM, see Figure 2.

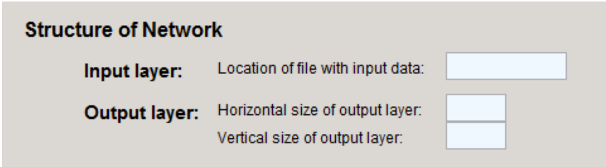

Figure 2. The basic settings of SOM

2 Learning

Secondly, the user has to set the parameters of the learning process including the number of epochs, see Figure 3.

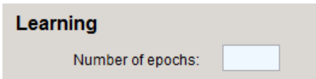

Figure 3. The selection of the number of epochs

2.1 Learning parameter

The learning parameter (learning rate, step length) is reduced during the iteration process. It decays from the initial value to the final value, which can be reached already during learning process, not only at the end of learning. There are several common forms of the decay function. The learning parameter should be in the interval <0.01, 1>.

Figure 4 shows options regarding the learning parameter.

The initial and final values of learning parameter have to be set. The initial value should be close to 1, the final value should be small, but not smaller than 0.1. Simultaneously, a point in the learning process in which the learning parameter reaches the final value has to be determined. It is represented as a number between 0 and 1.

The learning rate decay has to be set as well, for more information see Figure 5.

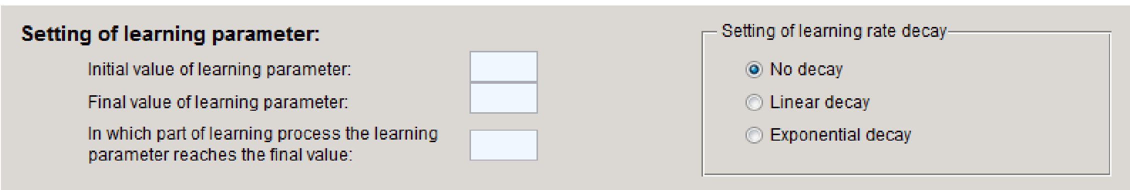

Figure 4. The settings of learning parameter

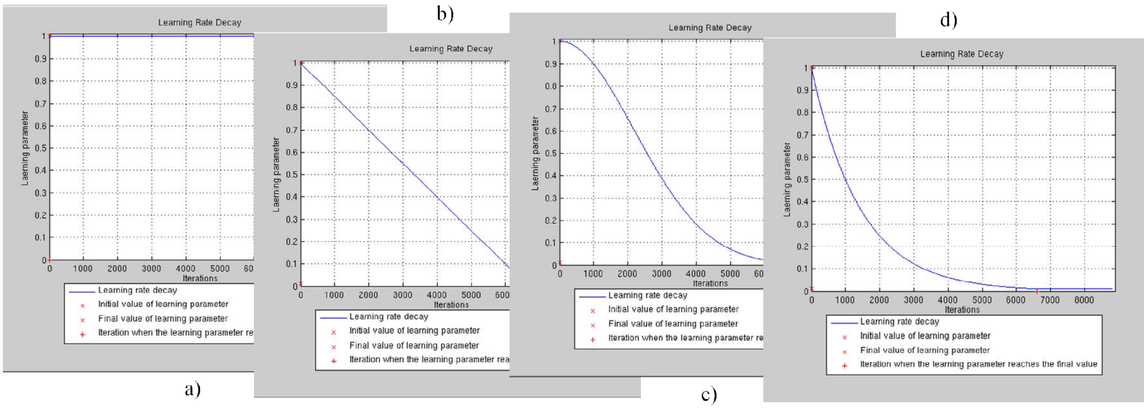

Figure 5. Learning rate decay function (dependence of the learning parameter on the number of iterations): a) No decay, b) Linear decay, c) Gaussian decay, d) Exponential decay

2.2 Neighbourhood

In SOM learning not only the winner but also the neighbouring neurons adjust their weights. It produces topology preservation. There several ways to define a neighbourhood (see Figure 6). All neighbour weight vectors are shifted towards the presented input vector, however, the winning neuron update is the most pronounced and the farther away the neighbouring neuron is, the less its weight is updated. The neighbourhood strength function determines how the weight adjustment decays with distance from the winner. The neighbourhood size function determines how the size of neighbourhood decays with increasing number of iterations.

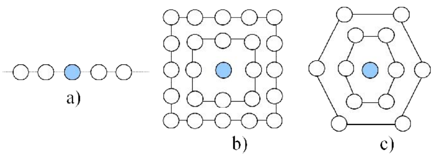

Figure 6. Types of neighbourhood: a) Linear arrangements, b) Square arrangements, c) Hexagonal arrangements

Figure 7 shows options regarding the neighbourhood.

The initial and final values of the neighbourhood size have to be set. The initial value can be up to the size of the output layer, the final must not be less than 1. That point in the learning process has to be determined, in which the neighbourhood size reaches the final value, i.e. number between 0 and 1.

The neighbourhood size decay has to be set as well, for more information see Figure 8.

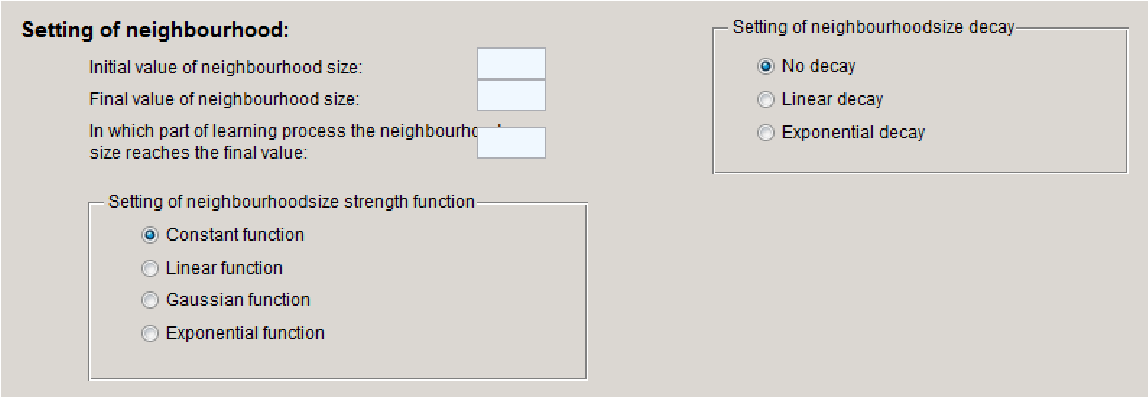

Figure 7. The settings of neighbourhood

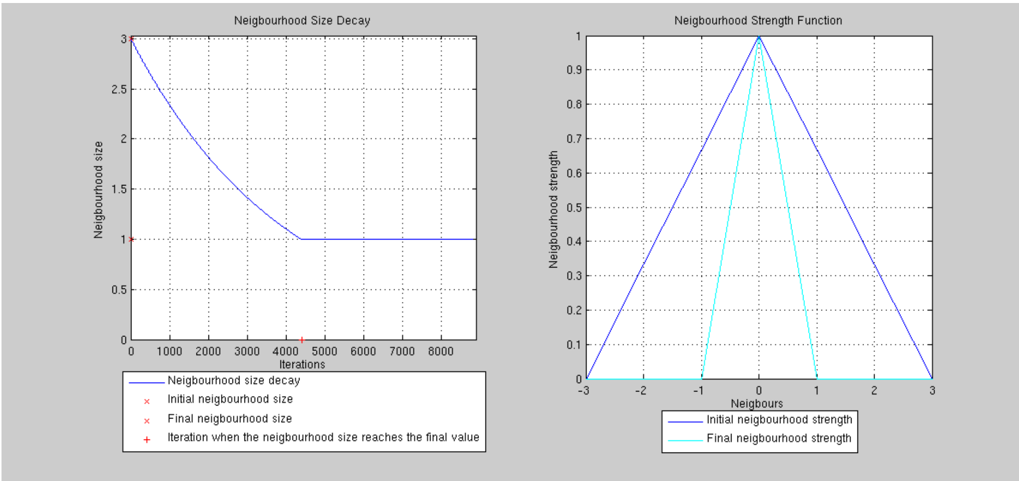

Figure 8. Neighbourhood size decay function (dependence of the neighbourhood size on the number of iterations) and neighbourhood strength decay function (dependence of the neighbourhood strength on the distance from the winner)

2.3 Weights

SOM is trained in recursive mode, i.e. the weights of the winning neurons are updated after each insertion of an input vector. The user has to choose the type of the weights initialization, see Figure 9, 10.

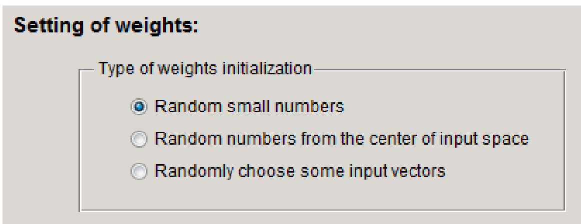

Figure 9. The Setting of weights

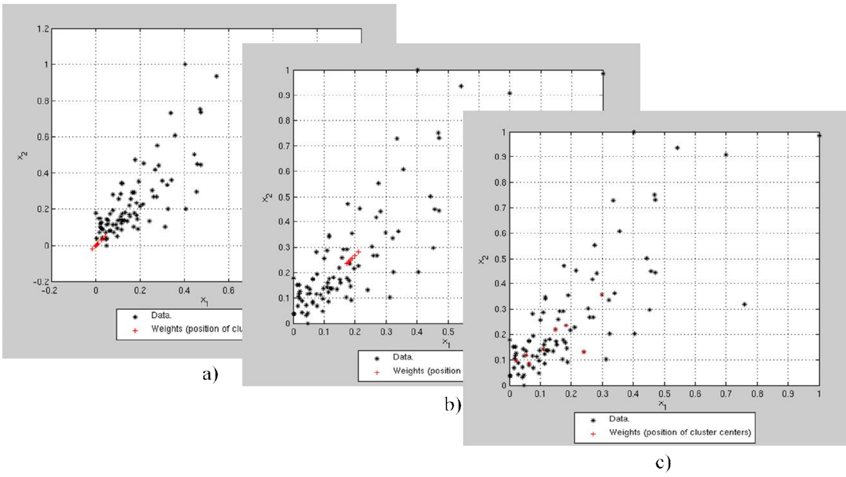

Figure 10. Weight vectors initialization: a) Random small numbers, b) Vectors near the center of gravity of inputs, c) Randomly chosen some input vectors as initial weight vectors.

2.4 Distance Measures

The learning in SOM is competitive. The nearest neuron to the presented input pattern becomes a winner. It is allowed to adjust its weight vector and weight vectors of its neighbourhood by moving weight vectors closer to that input vector. Then the winner neuron represents the input pattern. There are many measures of the closeness of a weight vector to an input vector, which the user can select (see Figure 11).

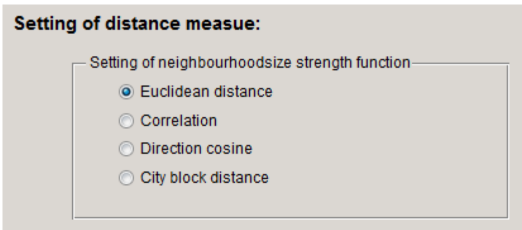

Figure 11. The Setting of distance measure

3 Results of SOM

The training criterion is the mean distance between all the inputs and their respective winning neuron weights. The weights corresponding to the smallest mean distance are the result of SOM. They represent the cluster centers.

Pressing the button 'Run SOM', the SOM starts to run, see Figure 12.

Figure 11. The 'Run SOM' button

4 Acknowledgements

The work was supported by the specific university research MSMT No. 21/2012 and grant MSM No. 6046137306.