

Selforganizing neural networks versus neurofuzzy networks

Stanislaw OSOWSKI* Tran Hoai LINH* Krzysztof SIWEK*

Abstract

The paper presents and compares two types of self-organizing neural networks: the hard clustering and fuzzy clustering structures. The learning algorithms as well as performances on statistically distributed data points are compared.

1 Introduction

The classical selforganizing neural networks are very well developed from learning theory and application point of view [1, 2]. Recently great attention has been paid to the fuzzy generalization of these networks, leading to the so called neurofuzzy networks. Neurofuzzy networks employ the theory of fuzzy sets [2, 3, 4] and include it into both learning algorithms and network structure. Inclusion of adaptive fuzzy inference systems into neural networks has led to the improvement of the quality of solutions delivered by neural networks. Thanks to clear interpretation of the fuzzy parameters and their strict association with the learning data it is possible to accelerate learning and obtain good generalization ability of these neurofuzzy networks.

This paper presents and compares the performance of classical and neurofuzzy selforganizing neural networks. The comparison is made on the samples of statistic data points of uniform and nonuniform distribution.

2 Hard clustering neural networks

The competitive network is usually one layer feed-forward structure, where all N inputs are connected to all M output units through the weights w_{ij} . The number of inputs is equal to the dimension of vector \mathbf{x} , while the number of outputs is equal to the number of clusters that the data are to be divided into. The cluster center's position is specified by the weight vector connected to the corresponding output neuron. The input vector $\mathbf{x} = [x_1, x_2, \dots, x_N]^T$ and the weight vector $\mathbf{w}_i = [w_{i1}, w_{i2}, \dots, w_{iN}]^T$ for i th neuron are normalized to unity length. The net activation value net_i of i th neuron is calculated as the inner product of the input and weight vectors $net_i = \sum_{j=1}^N x_j w_{ij} = \mathbf{x}^T \mathbf{w}_i$. Then the output unit with highest activation net signal is treated as the winner and selected for further processing. All

other neurons are losers with output signals equal zero. This is so called Winner Takes All (WTA) strategy. The winner for some compact data points represents the cluster of these data. The adaptation of weights of the neurons leading to the best organisation of partition of data can be performed either in on-line or off-line processes.

2.1 On-line clustering algorithms

The classical neural solutions for clustering problem, that have begun with Kohonen works [1] and usually apply the on-line approach, in which the updating of weights is done after presentation of each input data vector \mathbf{x} . In the process of learning, the neurons are selforganizing, where the self organization algorithm is formed by the sequence of the following operations:

- present the input vector \mathbf{x} to the network
- find the area in the network where the specific neuron responds most strongly to the presented vector \mathbf{x} ; the winner unit N_w is the one, whose weight vector is nearest, in the sense of assumed distance measure, to the input vector
- update the weights of the selected neurons of this area in the direction towards the vector \mathbf{x} .

Repeating these sequences many times brings the network to an organized state, in which each neuron represents one separate cluster of data. In the standard on-line algorithm we update the weights of the neurons found in the neighbourhood around the winning neuron N_w , according to the following rule

$$\mathbf{w}_k(t+1) = \mathbf{w}_k(t) + \eta_k(t)G(k, \mathbf{x})[\mathbf{x}(t) - \mathbf{w}_k(t)] \quad (1)$$

where \mathbf{w}_k is the vector of weights of k th neuron found in the neighbourhood of the winner, η is the learning coefficient, $G(k, \mathbf{x})$ is the neighbourhood function of k th neuron and t is the discrete time index.

The most powerful on-line learning algorithm is so called **neural gas**, in which the neighbourhood function is defined in terms of the distance between the input vector \mathbf{x} and the weight vector of the neuron. In this approach we arrange the neurons according to these distances, i.e., $d_0 < d_1 < d_2 \dots < d_{n-1}$ where $d_m = \|\mathbf{x} - \mathbf{w}_{m(i)}\|$ for $m = 0, 1, \dots, n-1$. The value of the neighbourhood function is then defined as $G(i, \mathbf{x}) = e^{-\frac{m(i)}{\lambda}}$ where $m(i)$ means the position of i th neuron after sorting and λ is the

*Warsaw University of Technology, Warsaw, Poland

parameter decreasing in time. The learning coefficient η in all approaches is usually decreased in time either exponentially or linearly. The algorithm of neural gas is regarded as one of the most effective methods of training the Kohonen network, allowing to obtain very good organization of network.

Another algorithm of selforganization, often used in noncontinuous function approximation is the WTA strategy. In this algorithm only the winner is updated each time using the Kohonen rule, described by (1) with the neighbourhood function equal 1 for winner and zero for other neurons. In this algorithm the problem of so called "dead" neurons that never win the competition is very important. To solve this problem the **conscience mechanism** is added, which modifies the distance between the input vector \mathbf{x} and the weight vector of the neuron appropriately to the past activity of the neuron. Neurons winning too often are punished by increasing this distance at the stage of competition. The typical modification is to multiply the real distance by the coefficient proportional to the number of victories of this neuron in the past, i.e. , $d(k, \mathbf{x}) \leftarrow V_k d(k, \mathbf{x})$ where V_k is the coefficient proportional to the number of winnings of k th neuron in the past. Thanks to such modification each neuron gets the chance to win and update its weights. The conscience mechanism is usually applied only at the first stage, allowing all neurons to participate in learning. After making all neurons active this mechanism is turned off, allowing free competition among the neurons.

2.2 Off-line clustering algorithm

The batch mode (off-line) operation of the selforganisation algorithm, often called also K-means or hard K-means algorithm, provides a simple mechanism for minimizing the sum of squared errors with K clusters, where each cluster is formed by the set of n samples that are similar to each other. The algorithm can be presented as follows:

- choose a set of initial clusters $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_K$, arbitrarily
- assign the samples to K clusters using the minimum Euclidean distance rule, according to which vector \mathbf{x} belongs to cluster C_i if $\|\mathbf{x} - \mathbf{c}_i\| < \|\mathbf{x} - \mathbf{c}_j\|$ for $j \neq i$
- compute new cluster prototypes as the mean of all vectors in group i , $\mathbf{c}_i = \frac{1}{n_i} \sum_{\mathbf{x}_j \in C_i} \mathbf{x}_j$ where n_i denotes the number of vectors \mathbf{x} assigned with i th cluster ($\sum_{i=1}^K n_i = n$)
- if any prototype changes return to step 2; otherwise stop.

At the end of iterations the cluster centers \mathbf{c}_i are represented by the weight vectors \mathbf{w}_i of the neurons ($\mathbf{w}_i = \mathbf{c}_i$). The K-means algorithm partitions

a collection of vectors \mathbf{x} into K groups and finds the cluster center in each group such that a cost function of dissimilarity or distance measure is minimized. Assuming the Euclidean distance as this measure, the overall cost function can be defined as follows

$$E = \sum_{i=1}^K \left(\sum_{\mathbf{x}_j \in C_i} \|\mathbf{x}_j - \mathbf{c}_i\|^2 \right) \quad (2)$$

For n data points and K clusters the partitioned groups can be fully described by the $K \times n$ binary membership matrix \mathbf{U} with element u_{ij} equal 1 if j th data point \mathbf{x}_j belongs to group i and 0 otherwise. In hard clustering a given data point can only be in one group, thus the membership matrix \mathbf{U} has the following properties:

$$\sum_{i=1}^K u_{ij} = 1 \quad \text{and} \quad \sum_{i=1}^K \sum_{j=1}^n u_{ij} = n \quad (3)$$

for all data points $j=1, 2, \dots, n$. The hard clustering algorithm does not take into account the relative activities of the neurons, rewarding only the winner. The loser's membership value is always zero, irrespective of its level of internal activation.

3 Fuzzy clustering networks

3.1 Clustering algorithm

Fuzzy clustering, called also C-means clustering [3] is an algorithm in which each data point belongs to the particular cluster to some degree, called the membership grade u . The vector \mathbf{x}_j belongs to the i th cluster with the membership grade described by $u_{ij} = u_i(\mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_j - \mathbf{c}_i\|^2}{\sigma^2}\right)$, where \mathbf{c}_i is the center vector and σ characterizes the width of the membership function. This time the vector \mathbf{x} can belong to several groups with different degrees of membership taking value from 0 to 1. Thus the entries of the matrix \mathbf{U} introduced in the previous section have the values between 0 and 1 with $\sum_{i=1}^K u_{ij} = 1$ for all data vectors. The overall cost function (2) can be now redefined as follows

$$E = \sum_{i=1}^K \sum_j^n u_{ij}^m \|\mathbf{c}_i - \mathbf{x}_j\|^2 \quad (4)$$

with m - the weighting exponent, $m \in [1, \infty]$. To reach the minimum of this cost function we have to take into account the constraint $\sum_{i=1}^K u_{ij} = 1$. The solution to the problem fulfills the relations

$$\mathbf{c}_i = \frac{\sum_{j=1}^n u_{ij}^m \mathbf{x}_j}{\sum_{j=1}^n u_{ij}^m} \quad (5)$$

$$u_{ij} = \frac{1}{\sum_{k=1}^K \left(\frac{d_{ij}}{d_{kj}}\right)^{2/(m-1)}} \quad (6)$$

where d_{ij} is the Euclidean distance between center \mathbf{c}_i and data vector \mathbf{x}_j . The fuzzy C-means clustering algorithm can be stated as follows:

- initialize the matrix \mathbf{U} with random values between 0 and 1
- find K fuzzy cluster centers \mathbf{c}_i using equation (5)
- calculate the cost function (4). If E is below the assumed tolerance value or its improvement over previous iteration is negligible - stop, else go to next step
- calculate new entries of the matrix \mathbf{U} using equation (6) and go to step 2.

This iterated procedure repeated many times leads to the minimum of E , which however is not necessarily the global minimum. The quality of solution is determined by the choice of the initial cluster centers following from the random values of the matrix \mathbf{U} . The centers should be concentrated in these areas where most of the multidimensional data points are distributed. Special methods of density distribution of data should be applied. The most well-known methods are the mountain clustering and subtractive clustering one[2]. The latest one has been applied in this work.

3.2 Fuzzy selforganizing neural network

The cluster centers are the most important parameters of the clusterization. If we want to find out the centers of the fuzzy rule for the premise and the consequent parts of the fuzzy inference simultaneously, we can concatenate the input and output information into one longer vector and process the clusterization on this vector. After the performed clusterization the center vectors \mathbf{c}_i are decomposed into two component vectors: \mathbf{p}_i of the input dimension and \mathbf{q}_i of the output dimension. After application of an input vector \mathbf{x} , the degree to which fuzzy rule i is fulfilled, is given by the membership function $u_i = \exp\left(-\frac{\|\mathbf{x}-\mathbf{p}_i\|^2}{\sigma^2}\right)$. Similar relation describes the membership value of the consequent part of the fuzzy rule with the input vector \mathbf{x} replaced by the output vector \mathbf{y} and \mathbf{p} replaced by vector \mathbf{q} .

After performing the clusterization process the nonlinear $\mathbf{x} \rightarrow \mathbf{y}$ mapping of the system can be finally defined. Applying for example the Wang - Mendel modification of the fuzzy inference engine [4], the response of the system at any point \mathbf{x} is defined in the following form

$$f(\mathbf{x}) = \frac{\sum_{l=1}^K q_l \exp\left(-\frac{\|\mathbf{x}-\mathbf{p}_l\|^2}{\sigma^2}\right)}{\sum_{l=1}^K \exp\left(-\frac{\|\mathbf{x}-\mathbf{p}_l\|^2}{\sigma^2}\right)} \quad (7)$$

The parameters \mathbf{p}_l and \mathbf{q}_l (for simplicity we have chosen only one output of the system - \mathbf{q} scalar) denote the centers of the respectively, premise (input) and consequent (output) part of the fuzzy rules l for $l = 1, 2, \dots, K$, adjusted in the previous stage. The parameter σ is a constant, chosen according to the specific application. From the input - output mapping point of view the fuzzy system, described by the equation (7) can be represented in a schematic neural network form as shown in Fig. 1. For simplicity we have introduced here the no-

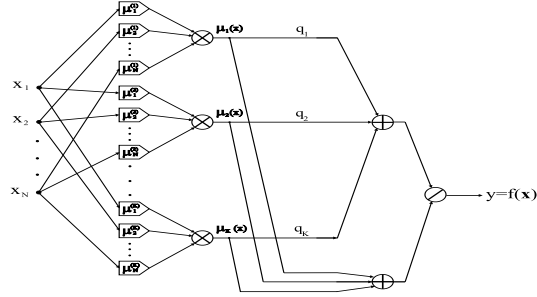


Figure 1: The general form of the feedforward fuzzy neural network

tations $\mu_i^{(j)} = \exp\left(-\frac{(x_i - c_i^{(j)})^2}{\sigma^2}\right)$ for $i = 1, 2, \dots, N$ and $j = 1, 2, \dots, K$ and $\mu_j(\mathbf{x}) = \prod_{i=1}^N \mu_i^{(j)}$ for $j=1, 2, \dots, K$. This system is called the neurofuzzy network, since it resembles the multilayer feedforward neural network structure.

4 Comparative study of results

Different strategies of hard competitive learning introduced in the previous sections have been implemented in C and checked in numerical experiments performed for the 2-dimensional data points. Different distributions of data have been tested. Fig.

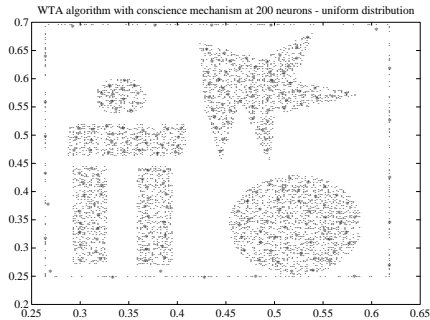


Figure 2: The organization of 200 neurons, representing the uniform distribution of data

2 presents the exemplary representation of the data

by 200 neurons trained using WTA strategy with conscience mechanism. As it is seen the neurons are placed first of all in the regions of high density of data. Similar organization of neurons have been obtained at the application of neural gas and fuzzy clustering, although in each case the placement of neurons differs a bit. Only pure Kohonen algorithm was inefficient and many neurons were dead, i.e., placed in the area deprived of data samples.

The obtained quantization errors E_q are very interesting. For the distribution of data shown in Fig. 2 after application of hard clustering algorithms at 200 neurons we got : $E_q = 0.007139$, for WTA, $E_q = 0.007050$ for neural gas and $E_q = 0.010763$ for Kohonen algorithm. In the case of 40 neurons we got respectively: $E_q = 0.017416$ (WTA), $E_q = 0.017599$ (neural gas) and $E_q = 0.02539$ (Kohonen algorithm).

Fuzzy selforganizing algorithm, supported by the subtractive clustering method, has shown better results. At the same data points for 40 fuzzy neurons the quantization errors were equal $E_q = 0.01539$ and for 200 centers we got only $E_q = 0.005739$. Fig. 3 presents the other, this time a nonuniform

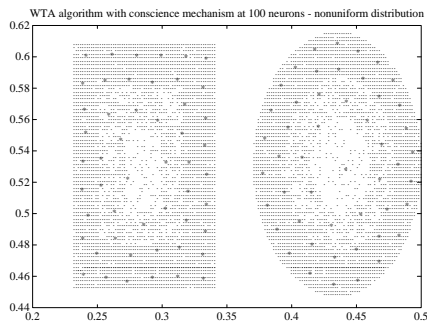


Figure 3: *The organization of 100 neurons, representing the nonuniform distribution of data*

distribution of data and placement of 100 selforganizing neurons, representing the data. The neurons are placed mainly in the regions of higher data density. The middle groups of lowest data density are represented by single neurons. This time the corresponding quantisation errors are as follows $E_q = 0.00689$, for WTA, $E_q = 0.00691$ for neural gas, $E_q = 0.0081$ for Kohonen algorithm and $E_q = 0.00012$ for fuzzy selforganization.

5 Conclusions

It should be noted that fuzzy clustering networks can be regarded as the generalization of hard clustering. In fuzzy systems each data point belongs to different clusters with some membership values

dependent on the distance from the cluster center to the input vector \mathbf{x} . Hence each cluster influences the final response of the system and this influence strongly depends on the membership value. In contrast to this hard clustering means association of the data vector with one cluster and only this cluster through its weights influences the final response of the system.

Fig. 4 presents the results of approximation of 1-dimensional sine function using the hard clustering and fuzzy clustering network. Only 20 neurons have been used in the test. The hard clustering approximates the curve in piecewise constant manner, while fuzzy clustering, corresponding to the piecewise nonlinear mapping, makes the approximation smoother and more accurate.

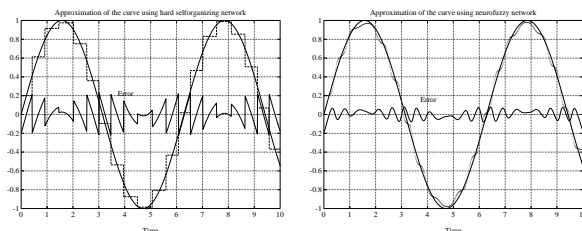


Figure 4: *Approximation of nonlinear curve using hard clustering (left figure) and fuzzy clustering (right figure) neural networks*

In hard clustering systems moving the input vector within the attraction region of the cluster center does not change the output of the system because the weights of the winning vector are not changed. We may say that hard clustering realizes the piecewise constant approximation strategy, while fuzzy clustering, where any change of \mathbf{x} changes the degree of membership of \mathbf{x} to different clusters, means piecewise nonlinear mapping from input to output space.

References

- [1] T. Kohonen, Self-organizing maps, Springer Verlag, 1995, Berlin
- [2] J. S. Jang, C. T. Sun, E. Mizutani, Neuro-fuzzy and soft computing, Prentice Hall, N. J., 1997
- [3] N. Pal, J. C. Bezdek, E. C. Tsao, Generalized clustering networks and Kohonen selforganizing scheme, IEEE Trans. Neural Networks, 1993, vol. 4
- [4] L. - X. Wang, Adaptive fuzzy systems and control, Design and stability analysis, Prentice Hall, N. J., 1994