Global Journal of Pure and Applied Mathematics.
ISSN 0973-1768 Volume 12, Number 6 (2016), pp. 5039-5046

http://www.ripublication.com/gjpam.htm

Neural Networks Neighborhood Models

Anatoliy Shmyrin and Irina Sedykh

Department of Mathematics, Lipetsk State Technical University, Lipetsk, 398600, Russia.

Abstract

A generalized definition of a neighborhood model is given. It is shown that a neural network can also be represented as a neighborhood model. An example of neural network neighborhood modeling is considered.

Keywords: systems theory, neighborhood models, neural networks, perceptron, activation function.

INTRODUCTION

Today the neighborhood systems theory [1-3] is a universal means of modeling a large class of discrete distributed systems: stationary and dynamic, fuzzy and non-fuzzy, with linear and non-linear bonds. Neighborhood models develop the general approaches of the systems theory and the control theory, as well as generalize such traditional discrete models as finite and cellular automata, Petri nets, difference equations etc.

A GENERALIZED DEFINITION OF THE NEIGHBORHOOD MODEL

A further development in the neighborhood systems theory is the introduction of a generalized definition of a neighborhood model in [4]. In its general case, a static neighborhood model is described by the set $NS_G = (N, X, V, Y, G, F)$, where:

1) $N = (A, O_x, O_v, O_y)$ – neighborhood model structure, $A = \{a_1, a_2, ..., a_n\}$ – set of nodes, O_x – neighborhoods of node connections by state, O_v – neighborhoods of node connections by control, O_y – neighborhoods of node connections by output impact. For each node $a_i \in A$, its neighborhood is determined by state

$O_x[a_i] \subseteq A$, by control $O_v[a_i] \subseteq A$ and by output $O_y[a_i] \subseteq A$; $O_x = \bigcup_{i=1}^{n} O_x[a_i]$, $O_v = \bigcup_{i=1}^{n} O_v[a_i]$; $O_y = \bigcup_{i=1}^{n} O_y[a_i]$;

2) $X \in R^p$ – neighborhood model state vector in real time;
3) $V \in R^m$ – neighborhood model control vector in real time;
4) $Y \in R^l$ – neighborhood model output vector in real time;
5) $G: X_{O_x} \times V_{O_v} \to X$ – recalculation function of neighborhood model states (generally non-deterministic), where X_{O_x} – number of node states entering the neighborhood O_x; V_{O_v} – number of node controls entering the neighborhood O_v;
6) $F: X_{O_x} \times V_{O_v} \to Y$ – recalculation function of neighborhood model outputs (generally non-deterministic).

In particular cases, for different discrete models there may be no separate elements of a neighborhood model.

Functions *G* and *F* may be arbitrary, e.g. linear, bilinear, quadratic, polynomial etc. In their linear case, *G* and *F* can be represented as a system of linear equations:

$$\begin{cases} \sum\limits_{x \in O_{x'}[a_i]} w_{x'}^x[a_i, \alpha] x'[\alpha] = \sum\limits_{x \in O_x[a_i]} w_x^x[a_i, \alpha] x[\alpha] + \\ + \sum\limits_{\beta \in O_v[a_i]} w_v^x[a_i, \beta] v[\beta], \\ \sum\limits_{y \in O_y[a_i]} w_y^y[a_i, \gamma] y[\gamma] = \sum\limits_{x \in O_x[a_i]} w_x^y[a_i, \alpha] x[\alpha] + \\ + \sum\limits_{\beta \in O_v[a_i]} w_v^y[a_i, \beta] v[\beta], \end{cases} \tag{1}$$

where $O_x[a_i]$ – neighborhood of node a_i by *x*; $O_v[a_i]$ – neighborhood of node a_i by *v*; $O_y[a_i]$ – neighborhood of node a_i by *y*; $a_i \in A$; $x[a_i] \in R^p$, $x'[a_i] \in R^p$ – states in node a_i of the model; $v[a_i] \in R^m$ – input in node a_i of the model; $y[a_i] \in R^l$ – output in node a_i; $w_{x'}^x[a_i, \alpha] \in R^{c \times p}$, $w_x^x[a_i, \alpha] \in R^{c \times p}$, $w_x^y[a_i, \alpha] \in R^{c \times p}$, $w_v^x[a_i, \beta] \in R^{c \times m}$, $w_v^y[a_i, \beta] \in R^{c \times m}$, $w_y^y[a_i, \gamma] \in R^{c \times l}$ – matrices-parameters; $\alpha, \beta, \gamma \in A$.
In its matrix form, the model (1) is given by:

$$\begin{cases} W_{x'}^x \cdot X' = W_x^x \cdot X + W_v^x \cdot V, \\ W_y^y \cdot Y = W_x^y \cdot X + W_v^y \cdot V. \end{cases} \tag{2}$$

When the functions *G* and *F* are non-linear, the model (2) is reduced to:

$$\begin{cases} W_{x'}^x \cdot X' = G(X, V), \\ W_y^y \cdot Y = F(X, V). \end{cases} \tag{3}$$

Changing the elements of the general description of a neighborhood model, it is possible to obtain different classes of discrete distributed models, e.g. Petri nets, transport systems and other discrete models.

A NEURAL NETWORK NEIGHBORHOOD MODEL

It will be shown that a neural network can be represented as a neighborhood model [4-6]. Let a neural network consist of m layers, in each j layer ($j=1,...,m$) there are n_j neurons. The i neuron of the j layer ($i=1,...,n_j$) will be designated as $i^{(j)}$.

The following definitions will be introduced. The neuron $i^{(j)}$ neighborhood by input impact $O_v[i^{(j)}]$ is a group of neurons of the previous $(j-1)$ layer with which this neuron is connected, including the neuron itself.

It is obvious that the input impact neighborhoods of the input layer neurons coincide with neurons themselves, i.e. $O_v[i^{(1)}]=\{i^{(1)}\}$, $i=1,...,n_1$.

The input impact neighborhoods of the input layer neurons will be called first-level neighborhoods. The input impact neuron neighborhoods of the *n*-layer (hidden or output) of a neural network will be called *n*-level neighborhoods. It is worth noting that neuron neighborhoods may meet or coincide (excluding the neuron itself).

The neuron neighborhood by state coincides with the neuron itself: $O_x[i^{(j)}]=\{i^{(j)}\}$, where j, ($j=1,...,m$) – layer number; i, ($i=1,...,n_j$) – neuron number in the layer.

Let us consider the representation of a neural network as a neighborhood model:

$$f\left(\sum_{\alpha\in O_v[a^j]} w_v^y[\alpha^{j-1},a^j]v[\alpha^{j-1}]\right) = y[a^j], \qquad (4)$$

where $v[\alpha^j]\in R$ – input impact in neuron α of the j ($j=2,...,m$) layer of the neural network; $y[a^j]\in R$ – output in neuron a of the j layer; $w_v^y[\alpha^{j-1},a^j]\in R$ – weighting factor of a bond of neurons α of the $(j-1)$ and a of the j layer; $f: R\to R$ – activation function.

Thus, a neural network is a neighborhood model $NS_{NN}=(N, ,V,Y, ,f)$.

If the weighted sum for the node a of the j layer of the neural network is designated as $s[a^j]=\sum_{\alpha\in O_v[a^j]} w_v^y[\alpha^{j-1},a^j]v[\alpha^{j-1}]$, then the formula (4) is given by:

$$f\left(s[a^j]\right) = y[a^j]. \qquad (5)$$

For the neuron a^j the received signal $s[a^j]$ is converted by the activation function f into the output neural signal $y[a^j]$.

Activation functions may be linear and non-linear. In the case of a linear activation function for a neural network we obtain a symmetrical linear neighborhood system [4-6]:

$$\sum_{\alpha \in O_v[a^j]} w_v^y[\alpha^{j-1}, a^j] v[\alpha^{j-1}] \quad = y[a^j]. \tag{6}$$

The formula (6) in its matrix form is given by:

$$\left(W_v^{y,j-1,j}\right)^T \cdot V^{j-1} = Y^j, \tag{7}$$

where $W_v^{y,j-1,j} \in R^{n_{j-1} \times n_j}$ – weighting matrix of the j layer; $V^j \in R^{n_j}$, $Y^j \in R^{n_j}$ – input and output vectors of the j layer, $(j = 2,..., m)$. It should be noted that after each iteration $V^j = Y^j$.

The formula (4) in its matrix form:

$$f^j\left(\left(W_v^{y,j-1,j}\right)^T \cdot V^{j-1}\right) = Y^j, \tag{8}$$

where $f^j : R^{n_{j-1}} \to R^{n_j}$ – activation function of the j layer.

In the case of a non-linear activation function, taking into account several summands in expanding this function brings to polinomial neighborhood systems [2].

From [2] it is known that the m-linear expression in polynomial neighborhood systems can be represented as the sum of elements containing flat matrices as the final result, i.e. its consideration will be reduced to a bilinear case and a combined control problem will be solved [1,2].

The total number of network neurons equals to $N = \sum_{j=1}^{m} n_j$. Let us compile the adjacency matrix R for an arbitrary neural network. In order to achieve that, we assign the natural number $k = 1,..., N$ to each neuron. Thus, k is the neuron's counting number in the network.

Matrix R can be represented as a number of blocks consisting of incidence matrices of each layer:

$$R = \begin{bmatrix} E & R^{1,2} & \mathrm{O} & \mathrm{O} & ... & \mathrm{O} \\ \mathrm{O} & E & R^{2,3} & \mathrm{O} & ... & \mathrm{O} \\ \mathrm{O} & \mathrm{O} & E & R^{3,4} & ... & \mathrm{O} \\ ... & ... & ... & ... & ... & R^{m-1,m} \\ \mathrm{O} & \mathrm{O} & \mathrm{O} & \mathrm{O} & ... & E \end{bmatrix},$$

where $R^{j-1,j}$ – adjacency matrix of the nodes of the $(j-1)$ and the j layers of the neighborhood model of a neural network $(j = 2,..., m)$; E – identity matrices of the corresponding shapes; O – zero matrices of the corresponding shapes.

Let w_{ik}^{j-1} be the weighting factor of the bond between the i neuron of the $(j-1)$ layer with the k neuron of the j layer $(i = 1,..., n_{j-1}; k = 1,..., n_j; j = 2,..., m)$. Let us denote

as $W^{j-1,j} = (w_{ik}^{j-1})$ the matrixes of the weighting factors of the bond between the $(j-1)$ and the j layers' nodes of the neighborhood model of the neural network $(j = 2,...,m)$.

Then the block matrix of weighting factors $W = (w_{ik})$, $(i,\ k = 1,...,N)$ of a neural network is determined in the following way:

$$W = \begin{bmatrix} E & W^{1,2} & O & O & ... & O \\ O & E & W^{2,3} & O & ... & O \\ O & O & E & W^{3,4} & ... & O \\ ... & ... & ... & ... & ... & W^{m-1,m} \\ O & O & O & O & ... & E \end{bmatrix}.$$

Then the neural network (7) in the block-matrix form is given by:

$$W^T \cdot V = Y, \qquad (9)$$

where $W \in R^{N \times N}$ – block matrix of weighting factors; $V \in R^N$, $Y \in R^N$ – input and output vectors; $N = \sum_{j=1}^{m} n_j$ – total number of network neurons.

AN EXAMPLE OF NEURAL NETWORK NEIGHBORHOOD MODELING

A partial 3-layer perceptron will be considered as an example of a neural network (Fig. 1). The bonds of the input layer neurons of the given neural network with the neurons of the first hidden layer following it, as well as the bonds between the neurons of the hidden and the output layers, are generally of a sampling character, with the weighting values adjusted on the training stage.

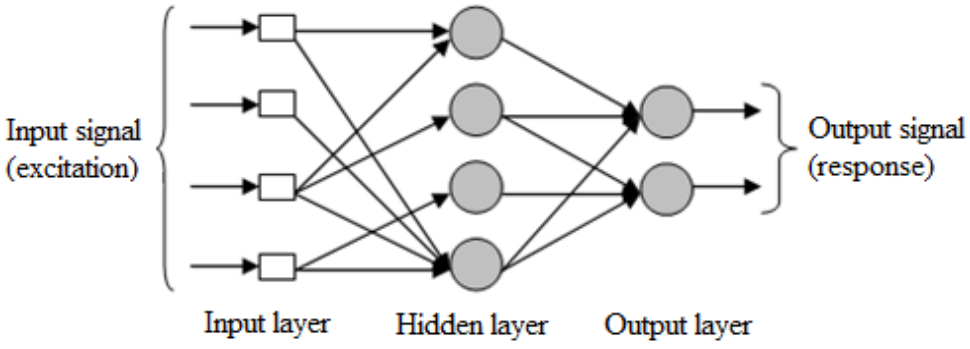

Figure 1. Partial 3-layer perceptron

The neural network shown in Fig.1 has the following second-level neighborhoods by input impact:

$O_v[1^{(2)}]=\{1^{(1)},3^{(1)},1^{(2)}\}$, $O_v[2^{(2)}]=\{3^{(1)},2^{(2)}\}$,

$O_v[3^{(2)}]=\{4^{(1)},3^{(2)}\}$, $O_v[4^{(2)}]=\{1^{(1)},2^{(1)},3^{(1)},4^{(1)},4^{(2)}\}$.

Third-level neighborhoods by input impact:

$O_v[1^{(3)}]=\{1^{(2)},2^{(2)},4^{(2)},1^{(3)}\}$, $O_v[2^{(3)}]=\{2^{(2)},3^{(2)},4^{(2)},2^{(3)}\}$.

Neuron neighborhoods by state: $O_x[i^{(j)}]=\{i^{(j)}\}$.

A three-layer neural network (see Fig. 1) will be represented as a neighborhood model:

$$f\left(\sum_{\alpha\in O_v[a^j]} w_v^y[\alpha^{j-1},a^j]v[\alpha^{j-1}]\right) = y[a^j], \qquad (10)$$

where $j=2,3$.

The total number of network neurons equals to $N=\sum_{j=1}^{3} n_j = 4+4+2=10$. Let us compile the adjacency matrix R for a neural network (see Fig. 1). In order to achieve that, the natural number $k=1,...,10$ will be assigned to each neuron.

The block adjacency matrix of the neural network shown in Fig. 1 is given by:

$$R=\left[\begin{array}{cccc:cccc:cc}
1&0&0&0&1&0&0&1&0&0\\
0&1&0&0&0&0&0&1&0&0\\
0&0&1&0&1&1&0&1&0&0\\
0&0&0&1&0&0&1&1&0&0\\
\hdashline
0&0&0&0&1&0&0&0&1&0\\
0&0&0&0&0&1&0&0&1&1\\
0&0&0&0&0&0&1&0&0&1\\
0&0&0&0&0&0&0&1&1&1\\
\hdashline
0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&1
\end{array}\right].$$

The matrix R can be represented as blocks consisting of adjacency matrices of each layer:

$$R^{1,2} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \; R^{2,3} = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}.$$

Then:

$$R = \begin{bmatrix} E & R^{1,2} & 0 \\ 0 & E & R^{2,3} \\ 0 & 0 & E \end{bmatrix},$$

where E – identity matrix of the corresponding shape.

Weighting matrices of each layer for a neural network (see Fig. 1) equal to:

$$W^{1,2} = \begin{bmatrix} w_{11}^1 & 0 & 0 & w_{14}^1 \\ 0 & 0 & 0 & w_{24}^1 \\ w_{31}^1 & w_{32}^1 & 0 & w_{34}^1 \\ 0 & 0 & w_{43}^1 & w_{44}^1 \end{bmatrix}, \; W^{2,3} = \begin{bmatrix} w_{11}^2 & 0 \\ w_{21}^2 & w_{22}^2 \\ 0 & w_{32}^2 \\ w_{41}^2 & w_{42}^2 \end{bmatrix}.$$

Then the block matrix of weighting factors $W = (w_{ik})$, $(i, \; k = 1,...,10)$ of a neural network (see Fig. 1) is determined in the following way:

$$W = \left[\begin{array}{cccc:cccc:cc} 1 & 0 & 0 & 0 & w_{11}^1 & 0 & 0 & w_{14}^1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & w_{24}^1 & 0 & 0 \\ 0 & 0 & 1 & 0 & w_{31}^1 & w_{32}^1 & 0 & w_{34}^1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & w_{43}^1 & w_{44}^1 & 0 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & w_{11}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & w_{21}^2 & w_{22}^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & w_{32}^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & w_{41}^2 & w_{42}^2 \\ \hdashline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \text{ or}$$

$$W = \begin{bmatrix} E & W^{1,2} & 0 \\ 0 & E & W^{2,3} \\ 0 & 0 & E \end{bmatrix},$$

where E – identity matrix of the corresponding shape.

Then the neural network (7) in the block-matrix form is given by:

$$W^T \cdot V = Y, \qquad (11)$$

where $W \in R^{10\times10}$ – block matrix of weighting factors; $V \in R^{10}$, $Y \in R^{10}$ – input and output vectors.

ACKNOWLEDGMENTS

The work is supported by the Russian Fund for Basic Research (project 16-07-00854 a).

REFERENCES

[1] Blyumin, S.L., Shmyrin, A.M. Neighborhood systems. Lipetsk: LEGI, 2005.

[2] Blyumin, S.L., Shmyrin, A.M., Shmyrina, O.A. Bilinear neighborhood systems. Lipetsk: LEGI, 2006.

[3] Blyumin, S.L., Shmyrin, A.M., Sedykh, I.A., Filonenko, V.Yu. Petri nets neighborhood modeling. Lipetsk: LEGI, 2010.

[4] Shmyrin, A.M., Sedykh, I.A. Discrete models in the neighborhood systems class // Bulletin of Tambov University. Vol.17, № 3. Tambov, 2012. – pp.867-871.

[5] Kornienko, N.A., Shmyrin, A.M., Sedykh, I.A., Shmyrina, T.A. Partial neural networks modeling by neighborhood systems // Management of large-scale system development (MLSD´2010): Proceedings of the 4th international conference. – Vol.1. Moscow: Institute of Control Sciences of Russian Academy of Sciences. – pp. 325-327.

[6] Shmyrin, A.M., Sedykh, I.A., Kornienko, N.A., Shmyrina, T.A. Discrete models generalization by neighborhood systems // Hardware and software of instrumentation, control and management systems: Proceedings of an international conference. Moscow: Institute of Control Sciences of Russian Academy of Sciences**,** 2010. – pp. 207-208.