Bull. Nov. Comp. Center, Comp. Science, 14 (2001), 65–77
© 2001 NCC Publisher

A method for learning of first-order cellular neural networks[*]

Sergey G. Pudov

In this paper, the first-order Cellular Neural Networks (CNN) with homogeneous weight structure are investigated and an approach to learn them is suggested. It is shown that all CNN weight templates are classified according to properties of possible stable states. As the result, the proposed learning method is based on the ideas of Perceptron learning rule. It allows to find parameters of a CNN connection template which provides the formation of patterns with preset properties. The method was applied to the 1D and the 2D CNN learning with symmetric templates and was verified by simulation.

1. Introduction

CNN were introduced in [1] and investigated as a fine-grained parallel model of computation for simulation of nonlinear phenomena both by means of conventional and special purpose CNN processor [2]. CNN can be viewed as a hybrid of Cellular Automaton (CA) and Artificial Neural Network (ANN). Like CA it consists of a huge number of simple processing elements (*cells*), usually placed in the nodes of an orthogonal or hexagonal grid, each cell being connected to a set of nearest neighbors. Like in the ANN's the connections are weighted, and each cell computes its output as a nonlinear function of its internal state, which is updated depending on the sum of weighted outputs of its neighbors. The evolution of a CNN is represented by a cell equation given either in the form of partial differential equation of reaction-diffusion type, or in its finite-difference form, when all cells calculate their next states in parallel, i.e., iteratively and synchronously. The computation starts when all cells are set in an initial state, and stops at a stable state, when no cells change their output states any more. Such a state represents a *pattern* in the form of the set of output cell states.

Investigation of stable states in CNN is motivated by the possibility to simulate and to study the processes of dissipative structures arising in active media in physics, chemistry, biology. In [3], the instances of CNN are given which form the patterns like those which occur in nature in the form of the markings on animals, marine life, insects, etc. The basic direction of

[*]Supported by the Russian Foundation for Basic Research under Grant 00-01-00026 and partially under Grant of Federal Program "Integration" № 274.

current CNN investigation is obtaining patterns formed by a given CNN and studying their properties relative to its template parameters. Particularly, in [4] the pattern formation by one- and two-dimensional CNNs is investigated, and their properties are studied for various connection templates for initial cell states randomly taken closely to zero. In some cases, the features of formed patterns were completely described, for example, for one-dimensional CNN with simplest connection template consisting of three neighbors, and for mosaic patterns in two-dimensional case with connection template called "square cross". In the latter case, the properties of CNN stable states are determined in the form of possible cell neighborhood states. For connection templates of larger size this problem has not been solved in the same way.

Here an attempt is made to find a method to solve the following task: given a pattern which is to be a CNN stable state – the template parameters of the CNN should be found. This task may not have analytical solution, so, like in all neural systems a learning method is the most appropriate. Because of the similarity of the CNN learning task to that of Cellular Neural Associative Memory (CNAM), the approach suggested in [5] and based on perceptron learning rule is used. This method was chosen according to the results of comparative analysis of the existing methods of CNAM learning and synthesis because it results in CNAM with the best capabilities to store information [6].

In this paper, autonomous CNN are considered, where next cell state depends on weighted sum of it neighbor's output state only. In Section 2 the formal model presentation and problem definition are given. In Section 3, features of possible stable states in the 1D and the 2D CNN are presented. Next, in Section 4, the main idea of learning approach and formal presentation of learning method are discussed. In Section 5, some experimental results for the 1D and the 2D CNN are given.

2. Formal problem definition

Notions in this paper are based on those suggested in [7]. We suppose that CNN consists of N cells which are enumerated in some way, for example, from 1 to N, i.e., each cell has a unique identifier or, in other words, a *name*. This numeration seems to be the most universal because it is applicable to two- and tree-dimensional CNN not only with orthogonal greed. As it was mentioned above, each cell in a CNN has weighted connections to its neighbors. Connection structure in a CNN is characterized by a *connection template* T, which for each cell i consists of the set of its neighbor names, i.e., $T(i) = \{j_1, \ldots, j_q\}$, where q is the cardinality of the cell neighborhood. It can be noted that we study CNN with spatially homogeneous connection structure, where the relative positions of cell neighbors are one and the same for all cells. A real number a_k denotes the weight value of the connection between a cell named i and its neighbor $j_k \in T(i)$, the set of all neighbors weights forms a *weight template* $A = \{a_1, \ldots, a_q\}$. Here we consider only space invariant templates that are symmetric. It is known [1] that in this case the CNN is stable, i.e., it always comes to a stable state. For each cell the set of its neighbors states forms a *neighborhood state* $X_i = \{x_1, \ldots, x_q\}$, where x_k is the state of the neighbor j_k of the cell i. In this paper, without loss of generality we suppose that the first neighbor of the cell i corresponds to the cell itself, i.e., x_1 denotes the state of the central cell i. The output state y_i of the cell i is a nonlinear function from the state x_i, i.e., $y_i = f(x_i)$. In this work, we use the following piece-wise function:

$$f(x) = \frac{1}{2}(|x+1| - |x-1|). \tag{1}$$

Thus, the cell output state is always bounded: $|y_i| \leq 1$. A cell i with $-1 \leq x_i \leq 1$ is called a *linear cell* and is represented by a gray square or a gray pixel in the figures. Otherwise it is called a *saturated cell* and is represented by a black square if $y_i = 1$ or a white one if $y_i = -1$. Further $Y_i = f(X_i)$ denotes the *neighborhood output state* of the cell i. With the above notations a weighted sum of neighbors output states can be written as follows:

$$A \otimes Y_i = \sum_{j \in T} a_j y_j. \tag{2}$$

Depending of this sum the cell changes its state x_i in time according to the equation, given by

$$\frac{dx_i}{dt}(t) = -x_i(t) + A \otimes Y_i(t), \tag{3}$$

in a continuous CNN, or according to the following synchronous updating rule:

$$x_i(t+1) = x_i(t) + \tau(-x_i(t) + A \otimes Y_i(t)) \tag{4}$$

in a discrete time CNN, where τ is a time discretization parameter. Computation stops at a stable state, when no cell output state changes in time any more, i.e., CNN forms a *pattern* which is the set of all cell's output states $\{y_i, i = 1, \ldots, N\}$. In both models (discrete and continuous), a given state is stable if and only if for all cells the following set of linear equalities and inequalities holds [4]:

$$\begin{aligned} & y_1\Big(\sum_{j \in T} a_j y_j\Big) > 0 && \text{if } |x_1| > 1, \\ & \sum_{j \in T} a_j y_j = x_1 && \text{if } |x_1| \leq 1. \end{aligned} \tag{5}$$

Now we can define the problem of CNN learning formally: given a pattern $Y = \{y_1, \dots, y_N\}$ – a weight template A should be found such that conditions (5) be satisfied. It should be noted that (5) determines a *set* of stable states with one and the same property: stability conditions depend on the state of the cell neighborhood but do not depend on its place.

3. Properties of stable states

In this section, properties of stable states, generated by one- and two-dimensional CNN are presented. Many of these results are from [3, 4] and are used in Section 5 for the choice of patterns which are input for CNN learning, and for verification of obtained weight templates.

3.1. One-dimensional CNN

At first, let us look at the known results for the 1D CNN with a weight template $A = [s, p, s]$. For this case in [4] the properties of stable states are studied for any combination of s and p, using the following parameter:

$$\mu = \frac{p-1}{2|s|}, \tag{6}$$

which determines the relation between p and the maximal influence of all neighbors ($|sy_2 + sy_3| \le 2|s|$, as $|y_1| \le 1$). Depending on the value of this parameter, three following cases are distinguished: 1) $\mu < -1$: the zero state ($x_i = 0$, $\forall\ i = 1 \dots N$) is the only equilibrium, and is therefore stable; 2) $\mu > 1$: any bipolar sequence ($|y_i| = 1$, $\forall\ i = 1 \dots N$) corresponds to an equilibrium of the CNN; 3) $|\mu| < 1$: there is always one and only one integer $B \ge 0$, for which any value that μ can take (except a set of measure zero) can be bounded:

$$-\cos\frac{\pi}{B+2} < \mu < -\cos\frac{\pi}{B+1}. \tag{7}$$

We assume that: 1) $s > 0$, since $s = 0$ is a degenerate case, and for $s < 0$ the results are easily transposed by reversing the sign of the state of every cell of even index; 2) the boundary conditions are either periodic or saturated cells (with $|y_0| = |y_{N+1}| = 1$). Then, for this CNN with $|\mu| < 1$ and $N > B$ cells, where B is given by (7), all possible stable states can be described in the following way [4, Theorem 1]: i) any linear cell belongs to a string of B linear cells surrounded by two saturated cells of opposite signs; ii) any saturated cell is bordered by at least one saturated cell of the same sign; iii) two consecutive saturated cells of opposite signs are separated by a boundary string of B linear cells.

Thus, any stable pattern is a succession of black and white strings of at least two cells long, separated by boundaries of B gray cells with B given

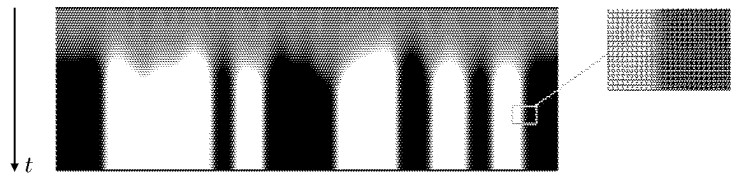


Figure 1. Example of time evolution of the 1D CNN consisting of 200 cells. Initial cells states are taken closely to zero

by (7). In Figure 1, the typical time evolution obtained by simulation of the 1D CNN with $B = 5$ is presented. In the obtained pattern, all boundaries between black and white cells must have the same length 5.

3.2. Two-dimensional CNN

For the 2D CNN with a simplest connection template of the form

$$A = \begin{bmatrix} 0 & s & 0 \\ s & p & s \\ 0 & s & 0 \end{bmatrix} \tag{8}$$

which is called "square cross", all possible stable states have been obtained in [4] for mosaic patterns only, i.e., for those patterns, where the output state of each cell is from $\{-1, 0, 1\}$. Like in the 1D case the parameter k is introduced which is an integer number satisfying the inequalities

$$p - 1 + (k+1)s > 0, \qquad p - 1 + ks < 0. \tag{9}$$

In [4], all possible mosaic neighborhood states are obtained with 8 different values of k from -5 to 4, representing all classes of mosaic patterns of a CNN. In this paper, we are not restricted by mosaic equilibrium states, we investigate CNN with output cell states from $[-1, 1]$. Hence, to show the relation between the template parameters and formed patterns, we present a diagram obtained by simulation, where one can see the typical stable states in the 2D CNNs with "square cross" connection template depending on the values of parameters p and s (Figure 2). The values p and s vary from -8 to 5 and from -2.5 to 2.5, respectively. At any restricted area on the diagram a pattern corresponds to that formed by the CNN with the weight template parameters p and s equal to the coordinates of the diagram.

3.3. Three groups of weight templates

From (5) it follows that there are an infinite set of weight templates which provide the stability of a given pattern C. It can be shown in the following way. Let A be the template which satisfies the conditions (5) for the pattern

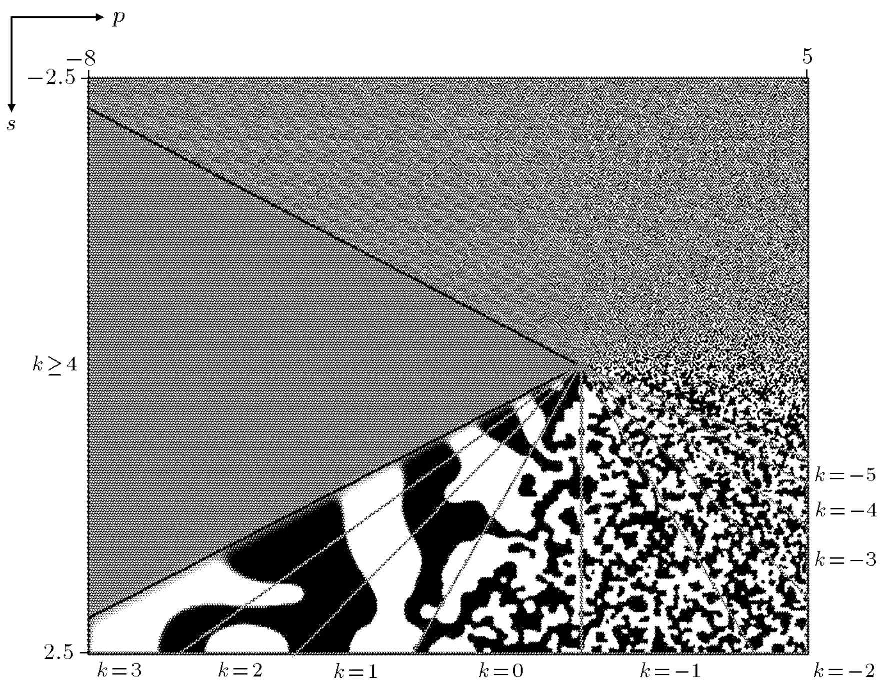


Figure 2. Diagram of possible stable states in the 2D CNN with "square cross" connection template. It consists of 400 by 500 cells, the values p and s vary from -8 to 5 and from -2.5 to 2.5 respectively. Initial cell states have been taken as random values closely to zero

C with a central cell c_1. If we multiply A by a constant $b > 0$, then these conditions may not hold for all cells. Particularly, for a linear cell i its state

$$c_1 \neq b(A \otimes C_i) = bc_1 \quad \text{when } b \neq 1. \tag{10}$$

In order to satisfy (10) it is enough to add the value $(1 - b)$ to the selfconnection weight in bA (the obtained template is further denoted as $A(b)$):

$$(A(b) \otimes C_i) = bc_1 + (1 - b)c_1 = c_1. \tag{11}$$

It is easy to show that the template $A(b)$, $b > 0$ satisfies (5) for all saturated cells. Moreover, the selfconnection weight value in it is calculated by the following formula: $p(b) = pb + (1 - b) = 1 + b(p - 1)$. As $b > 0$, then if the value p in A is greater than 1, then $p(b)$ can have any value above the 1 and vice versa, if $p < 1$, then $p(b) < 1$. Consequently, we have only three different groups of weight templates: 1) with $p < 1$; 2) with $p > 1$; and 3) with $p = 1$.

This result is useful for investigation of stable states in homogeneous CNN because it reduces the amount of independent weight parameters.

4. CNN learning

4.1. CNN learning approach

The proposed approach to CNN learning is based on the cellular modification of Perceptron Learning Rule (PLR) [8], which has been applied to Cellular Neural Associative Memory (CNAM) with bipolar stable states. This method was chosen for CNN learning because it is local, it guarantees the individual stability of prototypes, and, besides, the number L of stored patterns can be greater than the number q of cell connections. Moreover, all the existing methods of CNAM learning have been investigated and compared [6]. One of the main results is that the learning methods based on the perceptron learning rule result in CNAM with the best capabilities to store information. In order to modify CNAM learning method to that of CNN, the following should be done:

- At first, we define the set of prototypes for learning a weight template in homogeneous structure.
- After that, the PLR is modified for gray-scale prototypes.

At first, it is assumed that we have only one pattern C which is to be a stable state in CNN, and this pattern can be formed by a CNN with the connection template T of known size. The problem is to find the set of prototypes such that after learning process termination the obtained weight template A should satisfy the stability conditions (5) which are to be met for all cells. It is obvious, that we should use as prototypes, denoted as C_i, $i = 1, \ldots, N$, the set of neighborhood states of all cells (with the size of T) in the pattern C. Without loss of generality we suppose that the first L neighborhood states are pairwise different, and each one from the other $N-L$ states is equal to one of them. So, it is enough to use $C_1, \ldots, C_L$ instead of all N prototypes. In Figure 3, the example of pairwise different prototype set in the 1D CNN is presented for the connection template consisting of three cells.

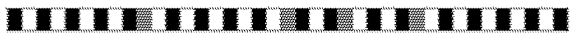
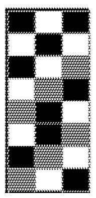
a)  b) 

Figure 3. The example of pairwise different prototypes (b) in the 1D CNN (a) for connection template consisting of three cells

Now we can present the algorithm of CNN learning. Let C be the pattern, which is to be stable in CNN. The problem to find the weight template A satisfying the conditions (5) may be solved as follows:

`Step 1.` Choose the simplest initial connection template T.

`Step 2.` Extract a set of prototypes $C_1, \ldots, C_L$ with the size of T from the pattern C.

`Step 3.` Apply the algorithm from Section 4.2 for learning the weight template A.

`Step 4.` If learning process does not terminate, then increase the size of connection template, and go to Step 2.

4.2. PLR for gray-scale prototypes

At last we present PLR for learning a weight template A using gray-scale patterns. If a central cell c_1 in a prototype C_i is saturated then to obtain (5) we should provide $c_1(A \otimes C_i) > 1$, which can be done with PLR by analogy to bipolar patterns. The case when the central cell is linear is more complicated. It is obvious that after finite number of learning iterations we cannot obtain the exact equality $A \otimes C_i = c_1$, for example, due to digital arithmetic inaccuracy. So, we should provide it approximately, i.e., $|A \otimes C_i - c_1| < \varepsilon$ for a small value $\varepsilon > 0$. For this we can use PLR with coefficients [8], where instead of $c_1^t C_i^t$, the vector $m_s c_1^t C_i^t$ is added to A^t; the coefficients m_s, $s = 1, 2, \ldots$, being chosen in the following way: $0 < m_s < 1$, $m_{s+1} \leq m_s$, $m_s \to 0$ with $s \to \infty$ (in order to make the modification of weight template more and more exact).

Now we can present the formal description of PLR for gray-scale prototypes. Let $C_1, \ldots, C_L$ be the prototypes obtained in Step 2 from Section 4.1. As in the standard PLR, an infinite sequence obtained by their recycling is organized, and a through numbering is introduced, c_1^t denotes the central cell for a prototype C^t; initial value of A^0 is chosen arbitrarily. After that, the weight template is updated according to the following iterative procedure, where $\varepsilon > 0$ is the required accuracy, s is a number of the macroiteration, m_s is a learning coefficient:

$$\begin{cases} \text{case } |c_1^t| = 1: & \\ A^{t+1} = A^t & \text{if } c_1^t(A \otimes C^t) > 1, \\ A^{t+1} = A^t \oplus m_s c_1^t C^t & \text{otherwise;} \\ \text{case } |c_1^t| < 1: & \\ A^{t+1} = A^t & \text{if } |A \otimes C^t - c_1^t| < \varepsilon, \\ A^{t+1} = A^t \oplus m_s C^t & \text{if } A \otimes C^t < c_1^t, \\ A^{t+1} = A^t \ominus m_s C^t & \text{if } A \otimes C^t > c_1^t, \end{cases} \tag{12}$$

where $A_i^t \oplus c_1^t C_i^t$ is the sum of two vectors, but the first weight in A_i^t is not changed, i.e., $a_j^{t+1} = a_j^t + c_i^t c_j^t$, $\forall\ j = 2, \ldots, q$; $\ominus$ is defined by analogy to $\oplus$. Calculation stops if the weight template does not change during one *macroiteration* (it is the learning period during which L prototypes are input to the CNN).

Note, that the learning coefficient is one and the same during each macroiteration s, and $\sum_s m_s = \infty$ (for example, $m_s = 1/s$), because otherwise (if $\sum_s m_s = G_1 < \infty$) all possible weight templates A^* obtained by this algorithm would lie near to the initial A^0, i.e., $\|A^* - A^0\| < G$ for some value of $G > 0$:

$$\|A^* - A^0\| \leq \Big(\sum_{s=1}^{\infty} m_s\Big)\Big(\sum_{t=1}^{L} \|C^t\|\Big) \leq G_1\Big(\sum_{t=1}^{L} \|C^t\|\Big) < G, \tag{13}$$

where $\|\cdot\|$ is the norm of a vector.

4.3. Special features of the weight template learning

1. The main feature of using PLR is as follows: the weight p of the central cell (it is also called the *selfconnection weight*) does not change during the learning process. It means that, for example, in the 1D CNN with a connection template consisting of three cells we fix the sign of a parameter μ for the obtained weight template, or in the 2D CNN with a "square cross" connection template we fix a vertical line on the diagram in Figure 2. It follows that because we do not know in advance what value of p is needed for the formation of the given pattern C, we should perform several learning processes for one set of prototypes but with different values of the selfconnection weight. Because all weight templates are divided into three groups (Section 3.3), we should perform three learning processes for one set of prototypes but different values of selfconnection weight (less, greater, and equal to 1).

2. If the amount of prototypes is very large then the learning process can take too much time. Sometimes it is possible to reduce the number of prototypes sufficiently (consequently, accelerate the learning) by analyzing the prototypes with a gray central cell: they should satisfy the following system of equalities

$$\sum_{j \in T} a_j c_j = c_1 \quad \text{if } |c_1| \leq 1. \tag{14}$$

Let $C_i = (c_1, \ldots, c_q)$ be the prototype used for weight template learning, and $|c_1| < 1$ is its central cell. It is assumed that the amount of such prototypes, which can be considered as elements of a vector space with the dimension q, is equal to $M > q$. It is obvious that there exist not more than q linearly independent ones among them (let them be numerated from 1 to $q_1 \leq q$), and all others are their linear combinations, i.e.,

$$C_k = \sum_{j=1}^{q_1} C_j \lambda_j \quad \text{where} \quad k > q_1. \tag{15}$$

Then, instead of M prototypes with a gray central cell it is enough to use only q_1 ones, because

$$A \otimes C_k = A \otimes \Big(\sum_{j=1}^{q_1} C_j \lambda_j \Big) = \sum_{j=1}^{q_1} c_j \lambda_j = c_k. \tag{16}$$

3. First experiments of CNN learning showed that the obtained weight template A^* is non-symmetric in more than a half of the experiments. This is not consistent with the model under investigation, because a CNN with non-symmetric weight template may not come to a stable state. So, it was decided to perform the symmetrization of the weight template during the learning process, particularly after each macroiteration. It is obvious that it does not influence the termination of the learning process. For example, in the 1D CNN with a connection template consisting of three cells, after each macroiteration the weights s_1 and s_2 in weight template $A = [s_1, p, s_2]$ are replaced by $s = (s_1 + s_2)/2$.

It can be noted that sometimes CNN with a non-symmetric weight template may also come to a stable state. In [9], the sufficient conditions for non-autonomous CNNs with non-symmetric weight templates are found, under which they come to the single stable state independently of the initial conditions. In our case of autonomous CNN, this stable state can only be zero, because otherwise there exist at least two stable states which differ in the sign of cell states only. So, this case is out of our interest, therefore the obtained weight template must be symmetric.

5. Simulation results

5.1. The goal of the experiments

All experiments are made using the system *WinALT* [10], specially developed for fine-grained structures simulation. The goal of simulation is to apply the learning procedure for obtaining weight template A^*, using as an input a stable state obtained by a known CNN with connection template A, and to compare the result with it. All experiments are performed by one and the same scheme.

`Step 1.` Using the CNN with a present weight template A a stable state C is obtained for the initial cell states randomly taken closely to zero.

`Step 2.` The size of the connection template T equal to that used in Step 1 for obtaining the stable state is chosen. From the pattern C all pairwise different prototypes $C_1, \dots, C_L$ are extracted.

`Step 3.` The weight template A^* is learned according to the proposed algorithm with symmetrization, using the set of prototypes from the previous step.

`Step 4.` After that, the obtained weight template is compared with the initial one A and is applied to the pattern C, which should not changed.

5.2. Simulation results for 1D CNN

Here we present the results of some experiments for one-dimensional circular CNN ($N = 50$ cells) with the weight template A_3 consisting of three cells. At Step 1, a stable state is obtained with a boundary of B gray cells, B vary from 1 to 5 with s both greater and less than zero. In Table 1, the basic simulation parameters and results are shown: the parameter μ of the initial weight template A, the learning time t in macroiterations, the obtained values for coefficient s^* from the obtained weight template $A^* = [s^*, 0, s^*]$, and the obtained parameter μ^*; the accuracy of learning ε is equal to 10^{-4}. It should be noted, that the initial weight template A has the weight $p < 1$ whereas in obtained template it is equal to zero.

Table 1. Experimental results for 1D CNN with weight template A_3

B	μ	s^*	t	μ^*	μ	s^*	t	μ^*
1	$-0.3(6)$	1.25	4	-0.4	$-0.3(6)$	-1.5	2	$-0.(3)$
2	-0.55	0.90917	1491	-0.5499	-0.55	-0.90913	3314	-0.5499
3	-0.75	0.66662	3059	-0.7500	-0.75	-0.66670	456	-0.7499
4	-0.82	0.60977	501	-0.8199	-0.82	-0.60974	948	-0.8200
5	-0.88	0.56814	339	-0.8800	-0.88	-0.56820	1293	-0.8799

Main results of this experiment can be formulated as follows. For the initial and obtained weight templates the parameters μ and μ^* determine stable states with the same properties, conditioned by the value of B. Moreover, if $B > 1$ then the obtained value μ^* coincides with the initial μ. It can be explained by the fact that for $B > 1$ the value of μ determines not only the width of gray cells but also their states.

5.3. Simulation results for 2D CNN

For the 2D CNNs the experiments are similar to that of the 1D CNNs. We study the CNN, consisting of ($N = 50 \times 50$ cells) with the "square-cross" weight template A_5. At Step 1, a stable state is obtained for different values of the parameter k, which vary from 0 to 3 (in all experiments we are interested in patterns with gray cells) with s both greater and less than zero. In the table the basic simulation parameters and results are shown: the parameter k and the parameters s and p of the initial weight template A, learning time t in macroiterations, the obtained values for the coefficient s^* (the value of selfconnection is chosen equal to zero); the accuracy of learning ε is equal to 10^{-4}.

From Table 2 one can see that for $k \geq 1$ the obtained weight template coincide with the initial one in correspondence with part 1 of Section 4.4. For example, if in initial weight template A the selfconnection weight p is equal to -2 and in the obtained $A^* = A(b)$ the selfconnection weight is taken equal to zero ($p^* = 0$), then from (11) the value b should be equal to 1/3, because $0 = p(b) = 1 + b(p-1) = 1 - 3b$. It is easy to see that the obtained parameter s^* is equal to a third from the initial s, so $A^* = A(1/3)$.

Table 2. Experimental results for 2D CNN with weight template A_5

k	s	p	s^*	t
0	1.1	0	1.25	2
0	2.1	0	1.25	2
0	6.1	0	2.00	2
1	0.7	0	0.7	10000
1	2.0	-2	0.667	50
2	1.2	-2	0.4	108
3	0.9	-2	0.299	81

6. Conclusion

In this paper, the approach to homogeneous CNN learning with gray-scale states is suggested. It is based on the PLR with scale coefficients, which allows to use gray-scale patterns. It is shown by simulation for the 1D and the 2D CNN that using this approach we can find a weight template of CNN which has a given stable state. The application of learning approach to patterns obtained by numerical simulation of PDE is under investigation now.

References

[1] Chua L.O., Yang L. Cellular neural networks: theory and application // IEEE Trans. Circuits and Systems. – 1988. – Vol. CAS-35. – P. 1257–1290.

[2] Roska T., Chua L.O. The CNN universal machine: an analogic array computer // IEEE Trans. on Circuits and Systems. Part II. – 1993. – Vol. 40. – P. 163–173.

[3] Chua L.O. CNN: a Paradigm for Complexity / Series on Nonlinear Science. – World Scientific, 1998. – Vol. 31.

[4] Thiran P., Crounse K., Chua L.O., Hasler M. Pattern formation properties of autonomous cellular neural networks // IEEE Trans. on Circ. and Syst. / Fund. Theory and Appl. – October, 1995. – Vol. 42, № 10.

[5] Pudov S. Cellular-neural associative memory learning // Optoelectronics, Instrumentation and Data Processing. – 1997. – Vol. 2. – P. 98–110.

[6] Pudov S.G. Comparative analysis of learning methods of cellular-neural associative memory // Lecture Notes in Computer Science. – 1999. – Vol. 1662. – P. 108–119.

[7] Bandman O.L. Cellular-neural computations, formal model and possible applications // Lecture Notes in Computer Science. – 1995. – Vol. 964. – P. 21–35.

[8] Rosenblatt F. Principles of Neurodynamics. – Washington: Spartan, 1959.

[9] Arik S., Tavsanoglu V. Equilibrium analysis of non-symmetric CNNs // Intern. J. Circuit Theory and Applications. – 1996. – Vol. 24. – P. 269–274.

[10] Beletkov D., Ostapkevich M., Piskunov S., Zhileev I. WinALT: a software tool for fine-grain algorithms and structures synthesis and simulation // Lecture Notes in Computer Science. – 1999. – Vol. 1662. – P. 491–496.