IMPROVEMENT OF HOPFIELD ASSOCIATIVE MEMORY BY CONTOUR ENHANCEMENT IN LIBRARY PATTERNS

Jacek Mańdziuk
Institute of Mathematics, Warsaw University of Technology,
Plac Politechniki 1, 00-661 Warsaw, POLAND
(INVITED PAPER)

ABSTRACT: In this paper a modification of binary Hopfield Associative Memory based on contour enhancement in library patterns is presented. The intuition behind the proposed improvement is the following: in the classical Hopfield's approach all bits in stored patterns are treated evenly under the Hebb storage rule, regardless of their "relative importance". On the other hand it is intuitively clear that the "importance" of a particular bit in a binary pattern correlates with a number of its neighbours with opposite sign. Therefore, it is proposed that in the storage process one should focus on bits lying on a contour of a pattern rather than on spots composed of ON bits or OFF bits only. In order to enhance the contour, patterns are transformed from a two-state (binary) representation to a four-(or more)-state one.

Results of computer simulations performed on various sets of randomly generated patterns support the idea of contour enhancement in case of sparse (dense) library patterns. High quality results are already obtained when a four-state representation is used, with further improvement observed along with the increase of the number of states.

Keywords - Associative Memory, Hopfield Model, Pattern Recognition, Contour Enhancement

1. INTRODUCTION

In the early eighties a neural network model based on the idea similar to the spin glass model in quantum mechanics was introduced (Hopfield, 1982; Hopfield 1984). One of the major application areas of this model are Associative Memories.

The capacity of Hopfield Associative Memory (HAM) under the Hebb storage rule - measured as the number of uniformly generated patterns which can be correctly recognized by HAM - is asymptotically equal to $0.15N$, where N is the length of patterns (Hopfield, 1982). In several applications this level of performance is not satisfactory. Moreover, in case of sparse, dense or correlated

patterns the storage capacity of HAM degrades rapidly (Jagota & Mańdziuk, 1998).

This paper presents a modification of binary HAM based on contour enhancement in library (stored) patterns. The intuition behind the proposed improvement is the following: in the classical Hopfield's approach all bits in stored patterns are treated evenly under the Hebb storage rule, regardless of their "relative importance". On the other hand it is intuitively clear that the "importance" of a particular bit in a binary pattern correlates with the number of its neighbours with opposite sign. Hence, it is proposed in the paper that in the storage process one should focus on bits lying on a contour of a pattern rather than on spots composed of all ON or all OFF bits. In order to enhance the contour, patterns are transformed from a two-state (binary) representation to a four-(or more)-state one.

In order to verify the idea of contour enhancement computer simulations were performed for several sets of patterns. In each data set, each library pattern of length N was divided into $\lceil \frac{N}{B} \rceil$ blocks of size B composed of subsequent pixels. All bits in a block were equal to one another and were equal to 1 with probability p, and to -1 with probability $1-p$. Hence, the degree of correlation between patterns in the simulations was governed by parameter B - the block size, whereas sparseness was defined by parameter p - the probability of all bits in a particular block to be ON.

Results of computer simulations performed for various sets of randomly generated patterns support the idea of contour enhancement in case of sparse (dense) library patterns. High quality results are already obtained when a four-state representation is used, with further improvement observed along with the increase of the number of states.

The paper is organized as follows: in the next Section the classical Hopfield Associative Memory is briefly reminded. In Section 3 proposed modifications are introduced and their properties discussed. Computer simulation results are placed in Section 4. Closing remarks and conclusions are presented in the last Section.

2. HOPFIELD ASSOCIATIVE MEMORY - A BRIEF DESCRIPTION

In this Section, HAM is briefly described and its main features are outlined. For more details please refer to Hopfield's papers (Hopfield, 1982; Hopfield, 1984).

Consider the set on M binary patterns $X^i = \{x_1^i, \ldots, x_N^i\}, x_j^i \in \{1, -1\}, i = 1, \ldots, M, j = 1, \ldots, N$. HAM used for storing and retrieving these patterns is composed of N neurons $y_1, \ldots, y_N$. The connection matrix $T \in \mathcal{M}_{N \times N}$ is computed by the Hebb rule, i.e.

$$t_{ij} = \sum_{s=1}^{M} x_i^s x_j^s \qquad i, j = 1, \ldots, N \tag{1}$$

where t_{ij} denotes connection weight from output of neuron y_j to input of neuron y_i.

In the recognition procedure, for unknown input pattern $Z = [z_1, \ldots, z_N]$ each neuron $y_i, i = 1 \ldots, N$ computes its output value according to the following rule[1]:

$$y_i = g(\sum_{j=1}^{N} t_{ij} z_j) \qquad i, j = 1, \ldots, N \tag{2}$$

where g is the threshold function:

$$g(x) = \begin{cases} 1 & \text{for} \quad x \geq 0 \\ -1 & \text{for} \quad x < 0 \end{cases} \tag{3}$$

HAM model is an excellent tool for storing and retrieving patterns when the number of them is relatively small and patterns are pairwise sufficiently distinct.

More precisely, it is shown in (Mc Ellice et al., 1987) that under some conditions imposed on stored patterns, asymptotical capacity of bipolar HAM is equal to $\frac{N}{4logN}$, or in case the demand for a rigorous recognition is slightly lessened the estimated capacity equals $\frac{N}{2logN}$. In rough estimation, asymptotical capacity of bipolar HAM reported in (Hopfield, 1982) is equal to $0.15N$.

Estimation of model's capacity and an attraction radius by the use of a single parameter is presented in (Wang et al., 1990). Some other estimations for a generalized Hopfield's network are given in (Xu & Tsai, 1990).

Unfortunately, the more patterns are stored in the network the poorer its average performance is. Moreover, performance degrades for sparse and correlated patterns (Jagota & Mańdziuk, 1998). There are several ways to improve HAM's capacity. Some of improved methods rely on other than Hebb one storage rules, e.g. pseudoinverse rule (Personnaz et al., 1986), perceptron-type rule (Kamp & Hasler, 1990; Jagota & Mańdziuk, 1998), projection rule (Sezan et al., 1990), unlearning procedure (Wimbauer et al., 1994), rules which include antiassociation (anticorrelation) terms (Jagota, 1994; Schultz, 1995), and others. Another approach is based on design of individual thresholds in the input-output transformation function of particular neurons. This can be done either in one-step (Schultz, 1995) or iteratively (Mańdziuk, 1996).

Some of these modifications lead to significant improvements although usually paying the price of being *non-local*. Moreover, many of the effective methods lack the *intuitive simplicity* and *biological relevance*. These three features are, on the other hand, emphasized as strong points of the Hebb storage rule.

3. PROPOSED MODIFICATIONS OF HAM

In this paper three methods that attempt to alleviate HAM's drawbacks by paying relatively more attention to contours than to interior spots in stored patterns are presented.

In order to achieve the above goal the set of library patterns is transformed to the set of "enhanced" patterns and then stored according to the Hebb rule. Transformation operation together with the storage procedure will be called *Contour Enhancement Hopfield Associative Memory* (CEHAM).

[1]The original recognition procedure proposed by Hopfield (Hopfield, 1982) was asynchronous and iterative. The one-step synchronous recognition scheme described here is one of the well-known alternatives.

Transformation procedure for each pixel in a pattern checks the number of its neighbours which are different from this pixel and based on that information computes a new value for this pixel. This new pixel's value is usually different from the original one, since the transformation converts a two-state (binary) representation into multi-state one. In the simplest case considered, each pixel after transformation can take one out of four possible values.

The following notation will be used in description of CEHAM. Denote by $\mathcal{Z}$ the set of all rectangular, binary patterns composed of NR rows and NC columns, and let $N = NR \cdot NC$.

For any given pattern $Y \in \mathcal{Z}$ and $i = 1, \ldots, NR, j = 1, \ldots, NC$ denote by D_{ij} the number of pixels adjacent to pixel $Y[i][j]$ different than $Y[i][j]$, by S_{ij} the sum of $Y[i][j]$ and all its neighbours, by N_{ij} the number of pixels adjacent to $Y[i][j]$ plus one, and by SUM the sum of all pixels in Y[2].

Suppose, that the set of library patterns $\mathcal{X} \subset \mathcal{Z}$ is composed of M elements $X^k, k = 1, \ldots, M$. In the simplest case, denoted by CEHAM-A, transformation of pattern X^k can be described as follows:

CEHAM-A

- compute S_{ij} for all pixels $X^k[i][j]$
- put

$$X^k[i][j] := \begin{cases} \alpha \cdot X^k[i][j] & \text{if } S_{ij} \cdot X^k[i][j] < 0 \\ X^k[i][j] & otherwise \end{cases} \tag{4}$$

  where $\alpha > 1$ is a predefined coefficient.

Informally, pixels which have more than half (in case of interior pixels), or more than half plus one (in case of side pixels) of their neighbours different than themselves scale their values so as to become more relevant compared to unmodified pixels.

In other words, pixels which are relatively distinct from their neighbours are enhanced and become even more distinct.

Another transformation procedure, denoted by CEHAM-B, takes additionally into account the sign of the sum of all bits in a library pattern.

CEHAM-B

- compute D_{ij} and N_{ij} for all pixels $X^k[i][j]$ and compute SUM
- put

$$X^k[i][j] := \begin{cases} X^k[i][j] - \gamma \frac{D_{ij}}{N_{ij}} & \text{if } SUM \geq 0 \\ X^k[i][j] + \gamma \frac{D_{ij}}{N_{ij}} & otherwise \end{cases} \tag{5}$$

  where γ is a positive constant.

The intuition behind CEHAM-B is the following. In case of $SUM \geq 0$, the number of positive pixels in a pattern is greater or equal to the number of negative ones. Therefore, the information about the location of negative pixels

[2]In the notation of D_{ij}, S_{ij}, N_{ij}, and SUM the dependence on pattern Y is omitted for the sake of clarity. This dependence will be clearly indicated by the context.

is (in average) more relevant for the recognition procedure than the information about positive pixels. Therefore, since $0 \leq D_{ij}/N_{ij} < 1$, all negative pixels are enhanced or remain unchanged, and the degree of an enhancement correlates with a number of positive neighbours. For the same reason the magnitude of positive pixels is either decreased or remains unchanged. Similar intuitive reasoning can be applied in case of $SUM < 0$. In this case positive pixels are enhanced or remain unchanged, and again the degree of a change depends on the number of their negative neighbours. The magnitude of negative pixels is lessened or remains unchanged.

CEHAM-B works efficiently for sparse (dense) patterns, that is when the number of negative (positive) pixels is significantly greater than the number of positive (negative) ones. Some problems may occur when p - the probability of all bits in a block being ON - is close to 0.5. Consider for example, the case of $SUM \geq 0$. From eq. (5), as mentioned above, it is clear that not only negative pixels are enhanced, but also positive ones are suppressed. This suppression is harmful when the ratio between the number of negative pixels and the number of positive ones is close to 1, because the enhancement of negative pixels is "unbalanced". Similar observation is true in the symmetric case of $SUM < 0$.

One of possible remedies to this problem is the last method proposed in this paper, denoted by CEHAM-C, in which in case of p close to 0.5 the identity transformation is applied. The description of CEHAM-C is the following:

CEHAM-C

- compute D_{ij} and N_{ij} for all pixels $X^k[i][j]$ and compute SUM
- put

$$X^k[i][j] := \begin{cases} X^k[i][j] - \gamma\frac{D_{ij}}{N_{ij}} & \text{if } \frac{SUM}{N} \geq \beta \\ X^k[i][j] + \gamma\frac{D_{ij}}{N_{ij}} & \text{if } \frac{SUM}{N} \leq -\beta \\ X^k[i][j] & otherwise \end{cases} \tag{6}$$

where β and γ are positive constants.

In all three methods, rectangular patterns after transformation are converted to a 1-dimensional vector by scanning row by row and then stored with the Hebb rule.

Recognition procedure in all three CEHAM methods is the same as the one in HAM. In particular, *test patterns are not transformed* in the recognition phase.

3.1 Two illustrative examples

As previously mentioned, the capacity of bipolar HAM degrades in case of sparse or dense patterns. In such situations modified methods perform significantly better than HAM. Before presentation of main results let us discuss two simple examples that illustrate potential benefits of proposed modifications.

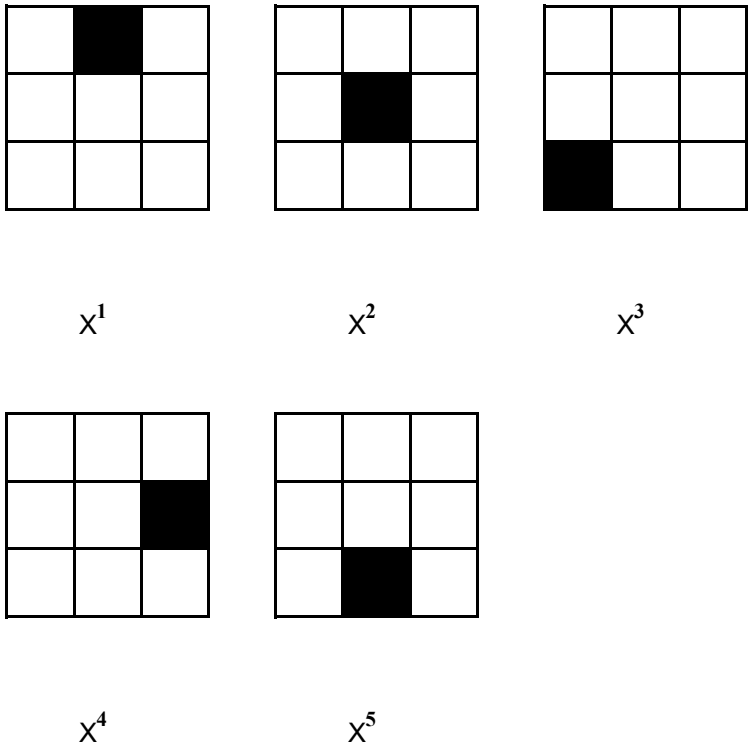

Figure 1. Set of sparse patterns from EXAMPLE #1.

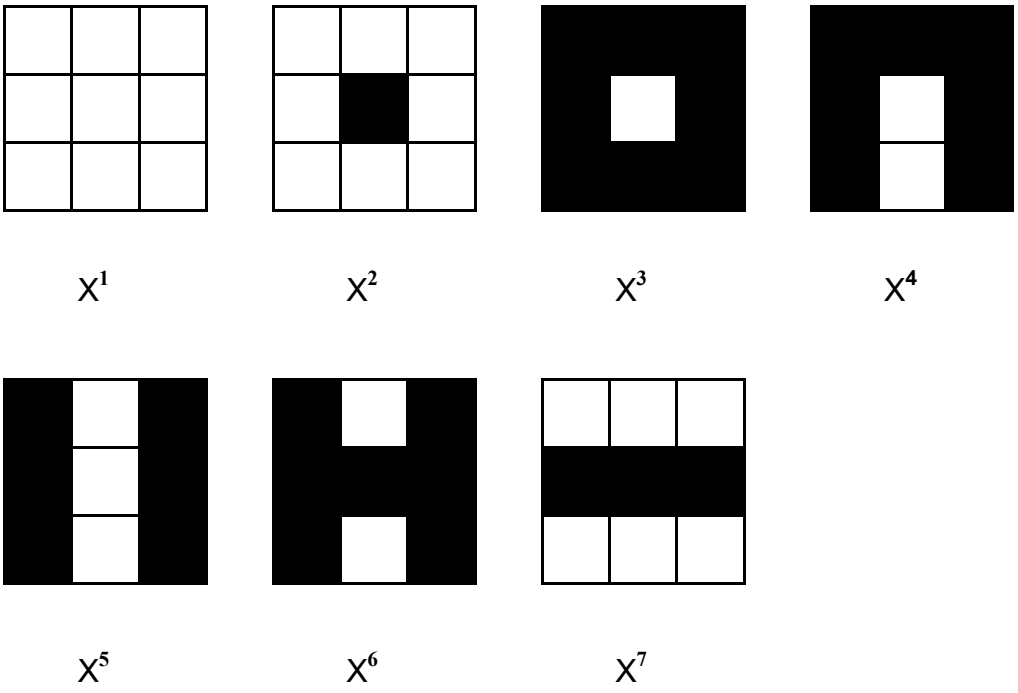

Figure 3. Set of corerlated patterns from EXAMPLE #2.

5,0	3,0	5,0	5,0	3,0	3,0	3,0	3,0	5,0
3,0	5,0	3,0	3,0	1,0	1,0	1,0	1,0	3,0
5,0	3,0	5,0	5,0	3,0	3,0	3,0	3,0	5,0
5,0	3,0	5,0	5,0	3,0	3,0	3,0	3,0	5,0
3,0	1,0	3,0	3,0	5,0	1,0	1,0	1,0	3,0
3,0	1,0	3,0	3,0	1,0	5,0	1,0	1,0	3,0
3,0	1,0	3,0	3,0	1,0	1,0	5,0	1,0	3,0
3,0	1,0	3,0	3,0	1,0	1,0	1,0	5,0	3,0
5,0	3,0	5,0	5,0	3,0	3,0	3,0	3,0	5,0

HAM

5,0	2,0	5,0	5,0	2,0	2,0	2,0	2,0	5,0
2,0	**8,0**	2,0	2,0	**-1,0**	**-1,0**	**-1,0**	**-1,0**	2,0
5,0	2,0	5,0	5,0	2,0	2,0	2,0	2,0	5,0
5,0	2,0	5,0	5,0	2,0	2,0	2,0	2,0	5,0
2,0	-1,0	2,0	2,0	8,0	-1,0	-1,0	-1,0	2,0
2,0	-1,0	2,0	2,0	-1,0	8,0	-1,0	-1,0	2,0
2,0	**-1,0**	2,0	2,0	**-1,0**	**-1,0**	**8,0**	**-1,0**	2,0
2,0	-1,0	2,0	2,0	-1,0	-1,0	-1,0	8,0	2,0
5,0	2,0	5,0	5,0	2,0	2,0	2,0	2,0	5,0

CEHAM-A

4,1	2,1	3,9	3,9	1,9	1,2	1,3	1,2	3,8
2,1	**6,7**	1,9	1,7	**-0,7**	**-0,5**	**-1,4**	**-1,4**	1,2
3,9	1,9	3,7	3,7	1,7	1,7	1,1	1,0	3,6
3,9	1,7	3,7	3,8	1,5	1,1	1,6	1,5	3,7
1,9	-0,7	1,7	1,5	6,7	-0,8	-0,5	-0,8	1,7
1,2	-0,5	1,7	1,1	-0,8	6,4	-1,5	-0,7	1,7
1,3	**-1,4**	1,1	1,6	**-0,5**	**-1,5**	**6,2**	**-0,4**	1,1
1,2	-1,4	1,0	1,5	-0,8	-0,7	-0,4	6,4	1,7
3,8	1,2	3,6	3,7	1,7	1,7	1,1	1,7	3,7

CEHAM-B

4,1	2,1	3,9	3,9	1,9	1,2	1,3	1,2	3,8
2,1	**6,7**	1,9	1,7	**-0,7**	**-0,5**	**-1,4**	**-1,4**	1,2
3,9	1,9	3,7	3,7	1,7	1,7	1,1	1,0	3,6
3,9	1,7	3,7	3,8	1,5	1,1	1,6	1,5	3,7
1,9	-0,7	1,7	1,5	6,7	-0,8	-0,5	-0,8	1,7
1,2	-0,5	1,7	1,1	-0,8	6,4	-1,5	-0,7	1,7
1,3	**-1,4**	1,1	1,6	**-0,5**	**-1,5**	**6,2**	**-0,4**	1,1
1,2	-1,4	1,0	1,5	-0,8	-0,7	-0,4	6,4	1,7
3,8	1,2	3,6	3,7	1,7	1,7	1,1	1,7	3,7

CEHAM-C

Figure 2. Matricies corresponding to sparse patterns from EXAMPLE #1

3.1.1 Example # 1. Sparse patterns

Consider the set of 5 patterns of size 3×3 presented in *Figure* 1. In the *Figure* positive pixels are denoted by black squares and negative ones are represented by white squares. Patterns are certainly sparse and therefore an advantage is expected from applying modified methods, compared to HAM. This is actually the case, since for each of these library patterns presented as the input, HAM responds with all-white pattern, yielding a 1-bit error.

On the other hand, all three modified methods correctly recognize library patterns. The reason for this behaviour becomes clear when comparing weight matricies generated by the methods. These matricies are presented in *Figure* 2. Parameters α, β and γ are set to $2, 0.2$ and 1, respectively.

In HAM's matrix, since all weights are positive and in a relatively small range, input vectors composed of a small number of positive pixels (like library patterns) produce negative response on all bits. On the contrary, in CEHAM-A, CEHAM-B and CEHAM-C matricies there exist some negative weights, and these weights support positive output on particular pixels in case of sparse input patterns. Moreover, the range of weights is relatively bigger in these matricies, compared to HAM.

Consider for example the 2nd row in CEHAM-A, CEHAM-B and CEHAM-C matricies. In each case:

- the element on the main diagonal has a large, positive value,
- there are four negative elements introduced by patterns X^2, X^4, X^3 and X^5, respectively. These elements are emphasized on *Figure* 2.

Both, the self-feedback diagonal element and those negative elements are crucial in preserving the **positive 2nd bit** in recognition of pattern X^1.

Similarly, for example, the large self-feedback diagonal weight along with the four negative elements in the 7th row are crucial in proper recognition of pattern X^3, which has the 7th bit ON. Again, these negative weights (in all three matricies) are introduced by the other library patterns, i.e. X^1, X^2, X^4 and X^5, respectively in positions $2, 5, 6$ and 8 of the 7th row.

Finally, note that for sparse patterns used in this example and for $\beta = 0.2$ there are no differences between CEHAM-B and CEHAM-C matricies.

3.1.2 Example # 2. Correlated patterns

In the second example the set of correlated patterns depicted in *Figure* 3 is considered. Some of the patterns are pairwise very similar, e.g. X^1 and X^2 or X^3 and X^4. Pattern X^2 is the complement of X^3. Moreover, X^1 and X^2 are sparse, whereas X^3, X^4 and X^6 are dense. *Figure* 4 presents matricies obtained in four methods.

Again, for the choice of $\alpha = 2, \beta = 0.2$ and $\gamma = 1$, there are no differences between CEHAM-B and CEHAM-C methods.

In the recognition tests some of potential advantages and drawbacks of the methods are clearly visible.

First, none of the methods is able to recognize X^1, which is mistaken with X^2. Output value on the 5th bit of X^1 equals $11.00, 18.00$ and 2.17 for HAM,

7,0	3,0	7,0	5,0	-3,0	5,0	7,0	1,0	7,0
3,0	7,0	3,0	**1,0**	-3,0	**1,0**	3,0	5,0	3,0
7,0	3,0	7,0	5,0	-3,0	5,0	7,0	1,0	7,0
5,0	1,0	5,0	7,0	-1,0	7,0	5,0	-1,0	5,0
-3,0	-3,0	-3,0	-1,0	**7,0**	-1,0	-3,0	-1,0	-3,0
5,0	1,0	5,0	7,0	-1,0	7,0	5,0	-1,0	5,0
7,0	3,0	7,0	5,0	-3,0	5,0	7,0	1,0	7,0
1,0	5,0	**1,0**	-1,0	**-1,0**	-1,0	**1,0**	7,0	**1,0**
7,0	3,0	7,0	5,0	-3,0	5,0	7,0	1,0	7,0

HAM

7,0	1,0	7,0	4,0	-8,0	4,0	7,0	-2,0	7,0
1,0	13,0	1,0	**-2,0**	-5,0	**-2,0**	1,0	10,0	1,0
7,0	1,0	7,0	4,0	-8,0	4,0	7,0	-2,0	7,0
4,0	-2,0	4,0	10,0	-2,0	10,0	4,0	-5,0	4,0
-8,0	-5,0	-8,0	-2,0	**22,0**	-2,0	-8,0	1,0	-8,0
4,0	-2,0	4,0	10,0	-2,0	10,0	4,0	-5,0	4,0
7,0	1,0	7,0	4,0	-8,0	4,0	7,0	-2,0	7,0
-2,0	10,0	**-2,0**	-5,0	**1,0**	-5,0	**-2,0**	16,0	**-2,0**
7,0	1,0	7,0	4,0	-8,0	4,0	7,0	-2,0	7,0

CEHAM-A

3,8	0,9	3,8	2,7	-4,2	2,7	3,6	-1,0	3,6
0,9	9,5	0,9	**0,0**	-3,1	**0,0**	0,7	7,4	0,7
3,8	0,9	3,8	2,7	-4,2	2,7	3,6	-1,0	3,6
2,7	0,0	2,7	6,3	-0,9	6,3	2,5	-1,6	2,5
-4,2	-3,1	-4,2	-0,9	**17,5**	-0,9	-3,8	1,3	-3,8
2,7	0,0	2,7	6,3	-0,9	6,3	2,5	-1,6	2,5
3,6	0,7	3,6	2,5	-3,8	2,5	3,4	-0,5	3,4
-1,0	7,4	**-1,0**	-1,6	**1,3**	-1,6	**-0,5**	11,6	**-0,5**
3,6	0,7	3,6	2,5	-3,8	2,5	3,4	-0,5	3,4

CEHAM-B

3,8	0,9	3,8	2,7	-4,2	2,7	3,6	-1,0	3,6
0,9	9,5	0,9	**0,0**	-3,1	**0,0**	0,7	7,4	0,7
3,8	0,9	3,8	2,7	-4,2	2,7	3,6	-1,0	3,6
2,7	0,0	2,7	6,3	-0,9	6,3	2,5	-1,6	2,5
-4,2	-3,1	-4,2	-0,9	**17,5**	-0,9	-3,8	1,3	-3,8
2,7	0,0	2,7	6,3	-0,9	6,3	2,5	-1,6	2,5
3,6	0,7	3,6	2,5	-3,8	2,5	3,4	-0,5	3,4
-1,0	7,4	**-1,0**	-1,6	**1,3**	-1,6	**-0,5**	11,6	**-0,5**
3,6	0,7	3,6	2,5	-3,8	2,5	3,4	-0,5	3,4

CEHAM-C

Figure 4. Matricies corresponding to correlated patterns from EXAMPLE #2

CEHAM-A and CEHAM-B(C), respectively. CEHAM-B(C) is closest to the negative value, but nethertheless the output on the 5th bit is positive also in this method.

Second, X^6 is mistaken with X^5 in HAM and CEHAM-A. In case of HAM, the reason for this error is too small self-feedback diagonal element, which is unable (together with the 2nd and the 8th ones) to suppress negative contributions from the other six elements. In CEHAM-A, the diagonal element is much larger, but due to the transformation procedure, which favors negative bits (there are only two of them in X^6) the negative elements in the 5th row are also enhanced. On the contrary, in CEHAM-B(C) the diagonal element is also much larger than in HAM, but the enhancement of negative pixels is kept in a reasonable range. As a result the output on the 5th bit of X^6 equals to $-3.00, -10.00, 1.38$ in HAM, CEHAM-A, CEHAM-B(C), respectively.

Third, HAM does not recognize patterns X^4, X^5 and X^7, which are mistaken with X^3, X^4 and X^2, respectively. Here, for all three patterns the same observations are valid. As an example, consider pattern X^4, which produces uncorrect output on the 8th bit, yielding pattern X^3. In HAM matrix there are six elements in the 8th row which contribute to errorneous positive output on the 8th bit of pattern X^4. On the contrary, in CEHAM-A and CEHAM-B(C) matricies all elements but one support negative output on this bit. Moreover, diagonal self-feedback element is enhanced in these matricies. Consequently, the output value on the 8th bit of X^4 equals $1.00, -25.00, -11.72$, respectively in HAM, CEHAM-A, CEHAM-B(C).

3.1.3 Summary

The examples reveal that results of enhancement transformation in CEHAM-A, CEHAM-B and CEHAM-C methods can be twofold.

First, in case of crucial pixels self-feedback diagonal elements are larger in these methods than in HAM. The highest values are obtained in CEHAM-A method. This can be clearly seen in the second example, in which diagonal elements in rows number $5, 8$ and 2 are magnified, in order to preserve proper recognition of bits number $5, 8$ and 2. These bits are the only distinctions between X^1 and X^2 as well as X^5 and X^6 for the 5th bit, X^3 and X^4 for the 8th bit and X^4 and X^5 for the 2nd one.

Moreover, in case of "unimportant" pixels diagonal elements are usually smaller in CEHAM-B(C) methods than in HAM or CEHAM-A. In the first example there are four pixels which are "less important" than the others, i.e. pixels number $1, 3, 4$ and 9. Therefore, in respective rows of CEHAM-B(C) matricies diagonal elements are smaller than in HAM and CEHAM-A matricies.

Second, in CEHAM-A and CEHAM-B(C) matricies there are several elements with reversed contribution to the output pixel value, that is elements with opposite sign compared to the ones in HAM matrix, e.g. five elements in the 8th row of matricies from Example 2 (emphasized in *Figure* 4).

Generally, the enhancement in CEHAM-B(C) is more selective and not so radical as in CEHAM-A. First of all, diagonal elements in CEHAM-A are stronger than in CEHAM-B(C). Additionally, several elements in CEHAM-A

matricies are stronger than their counterparts in CEHAM-B(C) or HAM matricies, e.g. $[4, 8]$, $[2, 8]$ or $[5, 7]$ in Example 2.

An interesting illustration of differences between methods are elements $[2, 4]$ and $[2, 6]$ in Example 2, which are positive in HAM, equal 0 in CEHAM-B(C), and are negative in CEHAM-A. Consequently, the 2nd and the 4th bits as well as the 2nd and the 6th ones are correlated in HAM, non-correlated in CEHAM-B(C) and anti-correlated in CEHAM-A.

3.2 Mutual relations between methods

CEHAM-A and CEHAM-C can be regarded as extensions of HAM. CEHAM-A uses *local* information about the *relative importance* of a particular bit in a stored pattern, and based on that information either enhances this pixel or leaves it unchanged. Enhancement coefficient α is *global* and therefore, in spite of the locality of checking neighbourhood procedure, a magnitude of the enhancement is equal for all "relevant" pixels. Setting $\alpha = 1$ in (4) leads to HAM, in which all bits are treated evenly. Consequently, CEHAM-A is a generalization of HAM in the above sense.

CEHAM-C, uses parameter β to separate the region in which HAM storage procedure is used from the region in which contour enhancement is applied. By varying parameter β one may either extend or shrink the region in which HAM formulation is applied. In particular, for $\beta > 1$ in (6) the formulation of CEHAM-C is the same as the one of HAM. Therefore, HAM can be regarded as a special case of CEHAM-C with $\beta > 1$, i.e. without contour enhancement.

4. SIMULATION RESULTS

The effectiveness of proposed modifications was tested on several sets of square patterns for various N. In each data set, all patterns were generated randomly with respect to two generic parameters: B and p. Namely,

- each pattern was divided into $\lceil \frac{N}{B} \rceil$ blocks composed of B subsequent pixels[3]. All bits in a block were either equal to $+1$ or to -1,
- p defined the probability of a particular block to be a $+1$-block.

Blocks in rectangular patterns were defined in a linear manner, that is patterns were scanned row by row starting from the top, downwards. The degree of correlation between patterns in a data set depended on B. Correlation increased along with an increase of B. Sparseness of patterns was defined by p and increased along with a decrease of $p(1 - p)$.

For example, a set of uniformly distributed patterns was generated with $< B, p >=< 1, 0.5 >$.

The efficacy of CEHAM-A, CEHAM-B and CEHAM-C will be presented for the following choice of parameters: $NC = NR = 10, M = 10, 20, 30, 40,$ and 50. For each choice of M, three values of p are tested: $p = 0.1, 0.3$ and 0.5[4].

[3]Due to the end effects, the last block could be smaller.

[4]Values of $p > 0.5$ are not tested because of symmetry of HAM and CEHAM methods with respect to p.

0 bits	M=10			M=20			M=30			M=40			M=50		
HAM	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**
0.1	9,40	9,47	9,00	9,88	8,97	9,42	9,71	9,50	9,11	9,70	9,70	9,08	9,70	9,50	9,40
0.3	6,43	6,90	4,33	16,08	11,17	8,42	22,50	14,79	10,17	25,26	17,38	13,92	26,82	19,87	13,27
0.5	0,00	0,80	1,50	0,32	1,45	3,67	0,70	2,48	2,78	1,45	2,62	1,92	1,71	1,89	1,10
CEHAM-A	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**
0.1	8,40	8,23	6,40	9,65	8,78	9,35	9,71	9,50	8,99	9,70	9,70	9,08	9,70	9,50	9,40
0.3	0,70	2,77	2,93	2,13	5,33	4,40	3,64	5,38	3,39	4,75	5,84	4,15	5,07	6,42	4,50
0.5	0,27	2,53	3,30	2,48	3,77	5,62	3,28	4,32	4,81	4,24	4,36	4,08	4,75	4,40	4,71
CEHAM-B	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**
0.1	8,37	8,63	7,20	9,80	8,93	9,25	9,71	9,50	9,11	9,70	9,70	9,08	9,70	9,50	9,40
0.3	1,20	2,80	3,47	2,48	4,88	2,98	3,66	3,77	2,21	4,18	3,25	1,63	4,27	2,71	1,53
0.5	1,30	3,67	4,10	2,15	3,93	5,57	2,90	4,08	4,44	3,45	3,91	2,95	4,09	3,09	2,53
CEHAM-C	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**
0.1	8,37	8,63	7,20	9,80	8,93	9,25	9,71	9,50	9,11	9,70	9,70	9,08	9,70	9,50	9,40
0.3	0,97	2,67	2,67	2,60	3,53	3,10	3,50	3,21	1,98	4,15	2,88	1,75	4,15	2,46	1,08
0.5	0,10	1,30	2,23	0,50	2,07	4,92	0,82	3,23	4,28	1,61	3,23	2,52	1,85	2,58	1,69

Table 1. Stability results for different choices of parameters B, p and M.

Block-size B is equal to $1, 3$ or 5. Coefficients α, β and γ are set to $2, 0.2$ and 1, respectively. Coefficients' values were assigned in a reasonable way, but *no coefficient optimization was performed*. Remarks on more efficient choices of α, β and γ are presented in Section 4.3.

Numerical results of computer simulations are presented in *Tables* $1, 2$ and 3.

4.1 Stability results

Table 1 describes *stability results*, obtained in recognition of a set of uncorrupted library patterns. Values in *Table* 1 denote *average numbers of incorrect bits* per pattern in respective pattern sets. Each value is averaged over three independent data sets. For each triple $< M, B, p >$, the same data sets are used for all four methods.

For example, value 2.48 in section CEHAM-A/M=20 in row denoted by 0.5 and column denoted by 1 should be interpreted in the following way: in the method CEHAM-A, in case of $M = 20$, for three independent data sets generated with $< B, p >=< 1, 0.5 >$, the average number of incorrect bits per pattern in the recognition phase computed among all library patterns is equal to 2.48.

The main qualitative observations from *Table* 1 and the rest of experimental data (not reported in the paper) are the following.

- **In case of $p = 0.5$, HAM performs slightly better than CEHAM-C, and outperforms CEHAM-B and CEHAM-A**.

From definitions of HAM and CEHAM-C, it is expected that in case of $p = 0.5$ both methods perform in the same way. This statement is true, however, only when the *actual value of* p is close to 0.5. Since the value of threshold β in the experiment was equal to 0.2 and γ was equal to 1, from eq. (6) it is clear that CEHAM-C performs as HAM if and only if $0.4 \leq p_{act}(X) \leq 0.6$, for all patterns in all data sets used in simulations, where $p_{act}(X)$ denotes the ratio between the number of positive pixels and the number of all pixels in pattern X.

In the experimental data $p_{act}(X)$ was outside the above limits for several patterns X. Consequently, for these patterns one of the first two cases of eq. (6) was applied in CEHAM-C method. Certainly, this difference between HAM and CEHAM-C will disappear for greater values of β.

- **In case of $p = 0.3$, the improvement in all three CEHAM methods is significant compared to HAM**.

There are mainly two reasons for such an improvement. First, HAM degrades on sparse patterns (cf. Jagota & Mańdziuk, 1998). Second, contour enhancement on the contrary, works efficiently for sparse patterns, in which the distinction between "important" pixels and "background" ones is especially visible and more relevant for the proper recognition than in the case of uniformly distributed patterns.

The difference between three CEHAM methods in case of $p = 0.3$ is more subtle. For small pattern sets ($M = 10, 20$) composed of uncorrelated patterns

1 bit	M=10			M=20			M=30			M=40			M=50		
HAM	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**
0.1	9,40	9,47	9,00	9,88	8,97	9,42	9,71	9,50	9,11	9,70	9,70	9,08	9,70	9,50	9,40
0.3	6,70	6,60	5,00	15,95	10,87	8,25	22,47	14,68	10,44	25,02	17,13	14,21	26,77	19,48	13,70
0.5	0,00	0,90	1,67	0,37	1,55	3,83	0,94	2,34	2,61	1,65	2,58	2,00	2,17	2,13	1,10
CEHAM-A	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**
0.1	8,37	8,23	6,40	9,65	8,78	9,35	9,71	9,50	8,99	9,70	9,70	9,08	9,70	9,50	9,40
0.3	0,83	2,97	3,27	2,37	5,17	4,47	3,80	5,34	3,31	4,88	5,90	4,18	5,11	6,77	4,37
0.5	0,47	2,40	3,20	2,68	3,72	5,80	3,91	4,57	5,04	4,44	4,63	4,31	5,26	4,82	5,02
CEHAM-B	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**
0.1	8,43	8,60	7,23	9,80	8,93	9,25	9,71	9,50	9,11	9,70	9,70	9,08	9,70	9,50	9,40
0.3	1,17	2,93	3,43	3,00	4,72	2,90	3,78	4,06	2,34	4,56	3,53	1,70	4,55	2,90	1,60
0.5	1,50	3,57	3,93	2,15	4,18	5,52	3,21	4,22	4,54	3,73	3,95	3,06	4,63	3,33	2,65
CEHAM-C	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**
0.1	8,43	8,60	7,23	9,80	8,93	9,25	9,71	9,50	9,11	9,70	9,70	9,08	9,70	9,50	9,40
0.3	0,97	2,83	2,93	3,00	3,78	3,13	3,67	3,37	2,07	4,45	3,23	1,87	4,59	2,70	1,26
0.5	0,10	1,57	2,40	0,53	2,18	5,15	1,04	3,31	4,42	1,76	3,28	2,67	2,27	2,93	1,85

Table 2. Error correction results in case of 1-bit distortion for each of the library patterns.

10 bits	M=10			M=20			M=30			M=40			M=50		
HAM	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**
0.1	9,40	9,47	9,00	9,88	8,97	9,42	9,71	9,50	9,11	9,70	9,70	9,08	9,70	9,50	9,40
0.3	8,13	6,80	5,50	16,37	11,13	8,42	21,33	13,86	10,94	24,30	17,82	14,21	25,73	18,70	13,63
0.5	0,23	0,80	1,50	1,72	3,17	4,50	3,61	4,66	4,61	5,13	4,82	3,58	6,43	4,56	3,37
CEHAM-A	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**
0.1	8,60	8,13	6,93	9,70	8,75	9,22	9,71	9,50	8,94	9,70	9,70	9,08	9,70	9,50	9,39
0.3	2,30	3,33	4,97	4,52	6,23	5,42	6,49	6,58	5,06	7,74	7,69	5,81	8,23	8,25	5,76
0.5	1,80	3,53	2,77	4,97	5,85	5,68	7,30	6,91	6,96	9,10	7,19	6,29	10,01	7,28	6,22
CEHAM-B	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**
0.1	8,57	8,47	7,07	9,77	8,92	9,38	9,71	9,50	9,11	9,70	9,70	9,08	9,70	9,50	9,40
0.3	3,27	3,50	4,70	6,05	5,77	4,05	7,18	5,83	4,16	8,68	5,98	3,84	9,10	5,93	3,51
0.5	2,90	4,37	5,03	5,75	6,45	6,88	7,62	6,24	6,71	9,38	6,70	5,16	11,01	5,87	4,17
CEHAM-C	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**	**1**	**3**	**5**
0.1	8,57	8,47	7,07	9,77	8,92	9,38	9,71	9,50	9,11	9,70	9,70	9,08	9,70	9,50	9,40
0.3	2,70	3,33	4,73	5,97	4,80	4,58	7,04	4,99	3,54	8,59	5,83	3,55	9,09	5,37	2,89
0.5	0,27	1,73	2,37	1,98	4,67	5,07	3,82	5,26	6,07	5,33	5,65	4,46	6,55	5,37	4,14

Table 3. Error correction results in case of 10-bit distortion for each of the library patterns.

($B = 1$), CEHAM-A performs a little better than CEHAM-B and CEHAM-C. Moreover, for $M = 10$, $B = 3, 5$, CEHAM-A outperforms CEHAM-B, but not CEHAM-C. In the other cases with $p = 0.3$ reported in *Table* 1, CEHAM-C usually performs better than CEHAM-B, and CEHAM-B is usually superior to CEHAM-A.

- **In case of $p = 0.1$ there are no relevant differences among four methods**.
- **For $M > 10,\ p = 0.1$ performance of HAM increases in comparison with $p = 0.3$.**

The two last observations can be explained as follows (cf. Jagota & Mańdziuk, 1998). When $p = 0.1$ there are very few attractors, perhaps only two: $(-1)^N$ and its complement 1^N. Therefore, for each library pattern approximately $0.1N$ of its bits should be unstable (all and only bits equal to 1). *Table* 1 reveals that this is indeed the case.

For HAM, in case of $p = 0.3$ there is probably only a few attractors as well, and therefore between $p = 0.1$ and $p = 0.3$ performance decreases. However, this reasoning cannot be extended to the case of $p = 0.5$, since as p increses new attractors emerge. Unless $\frac{M}{N}$ is too large, these attractors are closer to library patterns, yielding smaller average bit-error.

Results presented in *Table* 1 can be improved by more judicious choice of coefficients α, β and γ. In particular, the improvement can easily be achieved in case of $p = 0.1$. A discussion on more efficient choices of α, β and γ is presented in Section 4.3.

4.2 Error correction results

The layout of *Table* 2 and *Table* 3 is exactly the same as the one of *Table* 1. *Table* 2 presents results for the case, in which 1 randomly chosen bit in each library pattern is flipped in the recognition phase, whereas *Table* 3 concerns the case of flipping 10 randomly chosen bits in the recognition phase. Errorneous input bits are chosen independently for each library pattern. Again, results are averaged over three data sets - the same ones that were used to evaluate stability performance in Section 4.1.

Error correction results reported in *Table* 2 (1 bit distortion case) and *Table* 3 (10 bit distortion case) and the rest of experimental data (not reported in the paper) permit the following qualitative conclusions.

- **The four main observations presented in the previous Section are generally true also for error correction results.**

The explanations presented in Section 4.1 also remain valid.

- **For $p = 0.3,\ M > 10$ performance of HAM remains on the same level in the error correction case as in the stability case.**

The possible explanation of this phenomenon is the fact that for relatively sparse patterns ($p = 0.3$) there exist strong attractors and consequently the distortion is suppressed by the magnitude of these attractors.

CEHAM-A	B=1	B=3	B=5
M=10	4,26	3,16	4,46
α	4	4	5
M=20	6,00	5,06	4,10
α	4	5	5
M=30	6,05	4,42	3,47
α	5	6	6
M=40	5,31	4,45	3,51
α	6	6	6
M=50	4,30	4,11	3,59
α	6	7	7
CEHAM-C	**B=1**	**B=3**	**B=5**
M=10	4,23	4,06	4,43
γ	3	3	3
M=20	5,68	4,98	5,11
γ	3	4	4
M=30	6,38	4,32	4,00
γ	3	5	5
M=40	5,76	4,09	4,31
γ	4	5	5
M=50	5,51	5,02	5,13
γ	4	5	5

Table 4. Enhanced results in case of p=0.1,
for CEHAM-A and CEHAM-C(B) methods.

4.3 Tuning of α, β and γ

In the simulations reported so far it was assumed that the distribution of library patterns was unknown, and therefore α, β and γ were fixed for all test sets to values $2, 0.2$ and 1, respectively.

In order to present potential improvements, which can be achieved by more judicious choice of α, β and γ some effort was devoted to experimental establishing of more appropriate values for them. One of the goals was to find efficient α, β, γ for different choices of B and M, in case of $p = 0.1$. In the experimental evaluation, integer values between 2 and 10 were checked as candidates for α and γ. Value of β remained unchanged, since for $p = 0.1$ CEHAM-C should be simplified to CEHAM-B, i.e. without HAM subcase in (6).

For CEHAM-A method it was found that $4 \leq \alpha \leq 7$ was a better choice than $\alpha = 2$. Stability results for best choices of α among tested candidates, averaged over the same three data sets as the ones used in the main experiment are presented in *Table* 4.

The average bit-error over all values of M is approximately equal to $5.2, 4.2$, and 3.8 for $B = 1, 3$, and 5, respectively.

For CEHAM-C (CEHAM-B) method, efficient values of γ are equal to $3, 4$ and 5. Stability results in this case are presented in *Table* 4.

The average bit-error over all choices of M is approximately equal to $5.5, 4.5$ and 4.6, respectively for $B = 1, 3$ and 5.

For all three methods efficient value of the parameter (α or γ) increases along with the increase of B and independently along with an incresase of M (for B fixed). The improvement is about the factor of 2 compared to previous choices of α and γ.

For the clarity of the paper, error correction results in case of $p = 0.1$ as well as results for other values of p are not presented. The general conclusion is that efficient values of α, β and γ depend on the range of p, B and M. Results presented in the paper can serve as guidelines for appropriate choices of coefficients in case of other combinations of parameters p, B, M.

5. CONCLUSIONS

In the paper, three variants of a modification of Hopfield Associative Memory are presented. Suggested modification is based on contour enhancement in library patterns. In short, it is proposed to emphasize pixels which lie on the contour of a (content of a) pattern. Consequently, interior and background pixels (spots composed of all ON or all OFF bits) are relatively suppressed.

Results of computer simulations presented in the paper show that for sparse patterns the improvement is evident even in the simplest method (CEHAM-A). Further improvement is observed for more complex transformations (CEHAM-B and CEHAM-C). In case of uniformly generated patterns, modified methods perform slightly worse than HAM.

Our current research is devoted to theoretical analysis of performance of proposed methods, and in particular to deriving analytical bounds for efficient values of α, β, γ with respect to the degree of correlation between patterns - B, sparseness of patterns - p, and the number of them - M.

REFERENCES

1. Hopfield, J. J., (1982). Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA*, v.79, pp. 2554-2558.
2. Hopfield, J. J., (1984). Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. Natl. Acad. Sci. USA*, v.81, pp. 3088-3092.
3. Jagota, A., (1994). A Hopfield-Style Network with a Graph-Theoretic Characterisation. *Journal of Artificial Neural Networks*, v.1(1), pp. 145-166.
4. Jagota, A., & Mańdziuk, J., (1998). Experimental study of Perceptron-type local learning rule for Hopfield associative memory. *Information Sciences*, v.111(1-4), pp. 65-81.
5. Kamp, Y., & Hasler, M., (1990). *Recursive Neural Networks for Associative Memory*, John Wiley & Sons, New York.
6. Mańdziuk, J., (1996). Improving performance of the bipolar Hopfield network by supervised learning. *Proc. of the World Congr. on Neural Networks (WCNN'96), San Diego, CA, USA*, pp. 267-270.
7. Mc Ellice R.J., et al., (1987). The Capacity of the Hopfield Associative Memory. *IEEE Trans. on Inf. Theory IT-33*, v.4, pp. 461-482.
8. Personnaz, L., Guyon, I., & Dreyfus, G., (1986). Collective computational properties of neural networks: new learning mechanism. *Phys. Rev., A*, v.34(5), pp. 4217-4228.
9. Schultz, A., (1995). Five variations of Hopfield Associative Memory Network. *Journal of Artificial Neural Networks*, **2(3)**, pp. 285-294.
10. Sezan, M. I., Stark, H., & Yeh, S-J., (1990). Projection method formulations of Hopfield-type associative memory neural networks. *Applied Optics*, v.29(17), pp. 2616-2622.
11. Wang, J-H, et al., (1990). Determination of Hopfield Associative Memory Characteristic Using A Single Parameter. *Neural Networks*, v.3, pp. 319-331.
12. Wimbauer, S., Klemmer, N., & van Hemmen, J. L., (1994). Universality of Unlearning. *Neural Networks*, v.7(2), pp. 261-270.
13. Xu, X., & Tsai, W. T., (1990). Constructing Associative Memories Using Neural Networks. *Neural Networks*, v.3, pp. 301-309.

Published in
Neural, Parallel & Scientific Computations 7(3), 359-377, (1999)