NONLINEAR ENGINEERING AND ROBOTICS

MSC 2010: 37B15, 37B35, 37N35, 68T10, 92B20, 93D30

Autoassociative Hamming Neural Network

E. S. Antipova, S. A. Rashkovskiy

An autoassociative neural network is suggested which is based on the calculation of Hamming distances, while the principle of its operation is similar to that of the Hopfield neural network. Using standard patterns as an example, we compare the efficiency of pattern recognition for the autoassociative Hamming network and the Hopfield network. It is shown that the autoassociative Hamming network successfully recognizes standard patterns with a degree of distortion up to 40% and more than 60%, while the Hopfield network ceases to recognize the same patterns with a degree of distortion of more than 25% and less than 75%. A scheme of the autoassociative Hamming neural network based on McCulloch–Pitts formal neurons is proposed. It is shown that the autoassociative Hamming network can be considered as a dynamical system which has attractors that correspond to the reference patterns. The Lyapunov function of this dynamical system is found and the equations of its evolution are derived.

Keywords: autoassociative Hamming network, Hopfield network, iterative algorithm, pattern recognition, dynamical system, neurodynamics, attractors, stationary states

Received January 29, 2021

Accepted May 17, 2021

This work was done within the framework of the state assignment No. AAAA-A20-120011690135-5.

Ekaterina S. Antipova

antipovaes@live.ru

The State University of Management

Ryazansky prosp. 99, Moscow, 109542 Russia

Sergey A. Rashkovskiy

rash@ipmnet.ru

Ishlinsky Institute for Problems in Mechanics RAS

prosp. Vernadskogo 101/1, Moscow, 119526 Russia

1. Introduction

One of the impressive types of neural networks is the Hopfield network [1], which demonstrates that the process of information recovery (pattern recognition) can be carried out on the basis of the internal dynamics of the system itself rather than with the help of some logical scheme. In this case, the neural network is a dynamical system, while each pattern stored in the memory of the neural network corresponds to an attractor of this system. In this case the pattern presented for recognition determines the initial conditions for the further evolution of such a neurodynamical system, which ends when the system reaches an attractor and this is interpreted as pattern recognition. The invention of the Hopfield network spurred the further development of neurodynamical models of associative memory [2]. In addition, the Hopfield network showed one of the possible ways that nature could follow when creating biological neural networks in the process of evolution: the processes of memory and information processing in biological neural networks are associated with the internal dynamics of the nervous system, consisting of a large number of simple interacting elements — neurons.

Despite undeniable success, the Hopfield network has a significant drawback: it has a very small memory capacity, which forces us to look for other neurodynamical models that have a larger memory capacity.

The Hopfield network refers to the so-called autoassociative memory, in which the output (resulting) signal has the same dimension as the input signal.

Shortly after Hopfield published his pioneering paper [1] on an autoassociative neural network designed for pattern recognition, Lippmann published a paper [3] in which he described the heteroassociative network based on the Hamming distance and called the Hamming network. This network was also intended for pattern recognition. Although Hopfield's paper [1] was the impetus for the development of the Lippmann idea, only the iterative principle of finding a solution is common to the Hopfield and Hamming networks. If the autoassociative Hopfield network directly produces (as a result of its internal evolution) the recognized (right or wrong) pattern itself, the Hamming heteroassociative network [3] outputs, in fact, only the number of the reference pattern that is closest (from the point of view of the Hamming distance) with respect to the presented pattern. In order to “see” the pattern found by the Hamming heteroassociative network, an external interpreter is additionally required, which, using the number of the found pattern, would find it in the database and display it “on the screen” using the algorithms external to the Hamming network.

From this point of view, the heteroassociative Hamming network, in fact, solves the problem of classification (when each reference pattern is a separate class), and is a species of a Kohonen layer [2].

The ideas of [3] related to the Hamming network were developed in [4–13] for various applications. In all these studies, the different variants of the heteroassociative Hamming network were considered.

At the same time, it is of interest to construct an autoassociative Hamming network which would also be based on the calculation of Hamming distances, but which would produce directly the pattern found as a result of its internal evolution.

The goal of this work is to develop and study such an autoassociative Hamming network. As will be shown, the principle of operation of such a network is in many respects similar to the principle of operation of the Hopfield network [1].

2. Hamming distance and correlation coefficient

Consider the discrete binary space $X = (x_1, x_2, \dots, x_N)$, where $x_i = \pm 1$, and N is the dimension of space. Each point of this space corresponds to a specific sequence of ± 1 which can encode certain information, for example, a flat black and white pattern.

Let us determine the distance between two points $X = (x_1, x_2, \dots, x_N)$ and $Y = (y_1, y_2, \dots, y_N)$ of this space. This distance is naturally defined as the Hamming distance.

Calculation of Hamming distance is a logical operation, but when implementing a neural network, which is an analog device, it is desirable to avoid logical operations, preferring algebraic operations.

Consider the function

$$R(X, Y) = \frac{1}{4} \sum_{i=1}^N (x_i - y_i)^2. \quad (2.1)$$

Obviously, the function (2.1) is associated with the Euclidean distance $LE(X, Y)$ between these points:

$$LE(X, Y) = 2\sqrt{R(X, Y)}. \quad (2.2)$$

It is easy to see that, in the discrete binary space considered, the function (2.1) is equal to the Hamming distance between points X and Y .

Opening the parentheses in (2.1) and taking into account that $x_i^2 = y_i^2 = 1$, one obtains

$$R(X, Y) = \frac{N}{2} [1 - K(X, Y)] \quad (2.3)$$

where

$$K(X, Y) = \frac{1}{N} \sum_{i=1}^N x_i y_i \quad (2.4)$$

is the correlation coefficient of the vectors X and Y , which varies in the range $-1 \leq K(X, Y) \leq 1$. The smaller the correlation coefficient (2.4), the greater the Hamming distance between points X and Y .

If $K(X, Y) = 1$, then $R(X, Y) = 0$ and the vectors X and Y coincide: $X = Y$. If $K(X, Y) = -1$, then $Y = -X$, i.e., vectors X and Y are inverse with respect to each other. In this case, the Hamming distance between points X and Y is maximum: $R(X, Y) = N$. If the purpose of black-and-white pattern recognition is to understand which object is depicted on the presented pattern, then the direct and inverse pattern can be considered coincident, since they correspond to the same object. In this case, it does not matter which of the patterns, direct or inverse, was recognized by the neural network. If the goal is to find a complete coincidence of the recognized pattern with a certain reference pattern, then the direct and inverse patterns should be considered as different. Further, we will be interested in both these cases. If the vectors X and Y differ in random components (i.e., there are no regularities in the mismatching components), and at the same time $K(X, Y) = 0$, then the vectors X and Y (and corresponding patterns) do not correlate. This case is the most difficult from the point of view of pattern recognition.

So the larger $|K(X, Y)|$, the less differences between the patterns X and Y , while the smaller $|K(X, Y)|$, the less correlation between the patterns X and Y , and the less they have in common.

This property can be used for pattern recognition: the correlation coefficient $R(X, X^{(r)})$ is calculated for the presented pattern X and each of the reference patterns $X^{(1)}, X^{(2)}, \dots, X^{(L)}$, where $s = 1, 2, \dots, L$, and a reference pattern r is found for which $|K(X, X^{(r)})|$ either $K(X, X^{(r)})$ is maximum or $R(X, X^{(r)})$ is minimum. In the first case, such an algorithm does not distinguish between direct and inverse patterns.

This algorithm is implemented as a heteroassociative Hamming network [3], which is a Kohonen layer. The correlation coefficients (2.4) are calculated at the output of the network and then fed to the interpreter. In this case, the output neurons of the Kohonen layer contain only an adder and do not have an activation function. The heteroassociative Hamming network can be configured to calculate and analyze $|K(X, Y)|$. In this case, the output neurons of the Kohonen layer, in addition to the adder, have the activation function $F(z) = |z|$, or, more conveniently, in the analog implementation of the network, $F(z) = z^2$, where for the output neuron of the Kohonen layer with the number s : $z^{(s)} = \sum_{i=1}^N x_i x_i^{(s)}$ is the result of the operation of the adder.

The Hamming network can also be configured to calculate the Hamming distance; in this case, the activation function is chosen to be linear: $F(z) = N - z$.

The interpreter (filter) processes the signals from the outputs of the Kohonen layer and determines the number of the output neuron with the highest signal. The interpreter can be implemented in various ways, for example, in the form of a logic circuit, or, as in [3], using an additional self-recursive layer of neurons that implements the iterative process. Other interpreter schemes are also possible, which, however, do not change the principle of pattern recognition in the heteroassociative Hamming network.

3. Autoassociative Hamming neural network

Suppose there is a set of reference patterns $X^{(s)} = (x_1^{(s)}, x_2^{(s)}, \dots, x_N^{(s)})$, where $x_i^{(s)} = \pm 1$; $s = 1, 2, \dots, L$. It is necessary to determine which of them corresponds to the presented pattern X' .

In this case, the result should not be the number s of the corresponding reference pattern, but the pattern itself as a vector $X^{(s)}$.

Our goal is to develop an algorithm that is not reduced to logical procedures, but is a certain sequence of algebraic calculations, in the same way as this is done when recognizing patterns in the Hopfield network [1, 2].

We introduce a function

$$\Phi_0(X, Y) = \begin{cases} A \neq 0, & X = Y, \\ 0, & X \neq Y, \end{cases} \quad (3.1)$$

which we call the perfect filter, where $A > 0$ is some number. Then we can write the obvious identity

$$x_i^{(s)} = \operatorname{sgn} \sum_{r=1}^L x_i^{(r)} \Phi_0(X^{(s)}, X^{(r)}). \quad (3.2)$$

Although the perfect filter $\Phi_0(X, Y)$ is a logical function, it can be implemented, for example, using the recursive layer of the heteroassociative Hamming network [3]. This allows constructing the autoassociative Hamming network by adding two new layers to the heteroassociative Hamming network [3] (Fig. 1).

Fig. 1. The autoassociative Hamming network obtained by modifying the heteroassociative Hamming network [3].

Layers I, II and III form the usual heteroassociative Hamming network [3], the output signals of which are designated as $\Phi_0^{(r)}$. Layer IV consists of the formal neurons shown in Fig. 2, which work according to the algorithm

$$z_i = \sum_{r=1}^L w_{ir} \Phi_0^{(r)}, \quad y_i = \text{sgn}(z_i), \quad (3.3)$$

where $w_{ir} = x_i^{(r)}$ are the weights of the channels connecting the r th neuron of the third layer with the i th neuron of the fourth layer; the activation function is $F(z) = \text{sgn}(z)$.

The layer V plays the role of an interface and is intended only to display bits coming from the outputs of the fourth layer. Layers II, III and IV play the role of hidden layers.

The schematic diagram (Fig. 1) contains, in fact, simply a superstructure to the heteroassociative Hamming network [3]. However, this is not the only way to implement the autoassociative Hamming network. Consider a variant similar in its ideology to the Hopfield network.

Fig. 2. Schematic diagram of the i th formal neuron for the hidden layer IV.

We introduce a monotonically increasing function $\Phi(K)$ varying within $[-1, 1]$. Then the function $\Phi(K(X, Y))$, depending on the arguments X and Y , will be called a filter. The faster the function $\Phi(K)$ changes with a change in the parameter K , the closer the filter $\Phi(K(X, Y))$ is to the perfect filter (3.1).

In this case, the greatest contribution to the sum $\sum_{r=1}^L x_i^{(r)} \Phi(K(X^{(s)}, X^{(r)}))$ is made by those patterns $X^{(r)}$ that are closest (in terms of the Hamming distance) to the pattern $X^{(s)}$ and, therefore, this sum will be close to $x_i^{(s)}$.

As a filter, we can use any monotonically increasing function. We consider the power function

$$\Phi(K) = K^m, \quad (3.4)$$

where $m = 1, 2, 3, \dots$. The larger m , the closer the filter (3.4) to the perfect filter (3.1).

These considerations allow constructing an iterative process

$$x_i[n] = \operatorname{sgn} \left(\sum_{s=1}^L x_i^{(s)} K^m \left(X[n-1], X^{(s)} \right) \right) \quad (3.5)$$

where $n = 0, 1, 2, \dots$ is the iteration number. At each step n of the iteration, only one selected bit with the number i_n is adjusted by the formula (3.5).

As the initial condition, the vector presented for recognition is specified:

$$X[0] = X'. \quad (3.6)$$

Numerical experiments show that in asynchronous mode (i.e., when the bit i_n which will be corrected is chosen randomly at each iteration step), the algorithm (3.5), (3.6) converges to the nearest (from the point of view of the Hamming distance) reference pattern.

If the exponent m in the filter (3.4) is even ($m = 2, 4, \dots$), then the function $\Phi(K)$ is even: $\Phi(K) = \Phi(-K)$. In this case, the recognized pattern $X[\infty]$ (at the error-free recognition) will coincide with the corresponding reference pattern $X^{(s)}$, even if the original pattern X' presented for recognition is closer to the inverse pattern $-X^{(s)}$. This means that in this case, for the algorithm (3.5), (3.6), it does not matter how the presented pattern X' is “colored”; all that matters is which real object is behind it.

If the exponent m in the filter (3.4) is odd ($m = 1, 3, \dots$), then the function $\Phi(K)$ is odd: $\Phi(K) = -\Phi(-K)$. In this case, the recognized pattern $X[\infty]$ (at the error-free recognition) will coincide with the corresponding reference pattern $X^{(s)}$ if the pattern X' presented for recognition is closer to the pattern $X^{(s)}$ and will coincide with the inverse pattern $-X^{(s)}$ if the original pattern X' presented for recognition is closer to the inverse pattern $-X^{(s)}$. This means that for algorithm (3.5), (3.6) it matters not only what real object is behind the presented pattern X' , but also how it is “colored”.

Numerical experiments show that, starting with $m = 3$, this algorithm copes well with the problem of pattern recognition (see Section 5) even if the presented pattern is significantly distorted compared to the corresponding reference pattern. Calculations show that it suffices to choose $m = 3$ or $m = 4$, depending on whether the direct and inverse images are considered to be the same or not.

When $m = 1$, equation (3.5), taking (2.4) into account, can be transformed to

$$\begin{aligned} x_i[n] &= \operatorname{sgn} \left(\sum_{s=1}^L x_i^{(s)} \sum_{j=1}^N x_j[n-1] x_j^{(s)} \right) = \\ &= \operatorname{sgn} \left(\sum_{j=1}^N x_j[n-1] \sum_{s=1}^L x_i^{(s)} x_j^{(s)} \right) = \operatorname{sgn} \left(\sum_{j=1}^N x_j[n-1] w_{ij} \right) \end{aligned} \quad (3.7)$$

where

$$w_{ij} = \sum_{s=1}^L x_i^{(s)} x_j^{(s)}. \quad (3.8)$$

The iterative procedure (3.7) coincides with the algorithm of the Hopfield network in the recognition mode, and relation (3.8) can be considered as a learning rule for this network (Hebbian learning rule). It is slightly different from the Hopfield network learning rule, for which artificially it is assumed that $w_{ij} = 0$ for $i = j$, while according to (3.8) $w_{ij} = L$ for $i = j$.

Figure 3 shows a schematic diagram of an autoassociative Hamming neural network which operates in accordance with algorithm (3.4)–(3.6).

The first, the sensory layer, transmits without any changes the signals that have come to it to the neurons of the second layer, and simultaneously plays the role of an interface: it displays the pattern specified by the input vector X . Thus, the neurons of the first layer do not perform any calculations. The pattern X' , which is required to be recognized, is also fed to the neurons of the first layer. Neurons of hidden layers (the second and the third) are common formal neurons (Fig. 4).

Neurons of the second layer have the activation function $F_{II}(K) = K^m$ and work according to the algorithm

$$K_s = \sum_{i=1}^N w_{si} x_i, \quad \Phi_s = K_s^m, \quad (3.9)$$

where $w_{si} = x_i^{(s)}$ are the weights of the channels connecting the neuron i of the first layer with the neuron s of the second layer.

Fig. 3. Schematic diagram of an autoassociative Hamming neural network, which operates in accordance with algorithm (3.4)–(3.6).

Fig. 4. Formal neurons for hidden layers: a) neuron s of layer II; b) neuron i of layer III.

Neurons of layer III have an activation function $F_{\text{III}}(z) = \text{sgn}(z)$ and work according to the algorithm

$$z_i = \sum_{s=1}^L u_{is} \Phi_s, \quad x_i = \text{sgn}(z_i), \quad (3.10)$$

where $u_{si} = x_i^{(s)}$ are the weights of the channels connecting the neuron s of the second layer with the neuron i of the third layer. Thus, $w_{si} = u_{si}$.

The last layer of the autoassociative Hamming network (Fig. 3) provides a time delay and consists of interconnected gates: only one randomly selected gate opens at each time, which passes the signal from layer III to sensory layer I, thus updating the randomly selected bit.

In the future, speaking of the autoassociative Hamming network, we will only have in mind the network shown in Fig. 3.

The autoassociative Hamming network (Fig. 3) is trained by adding a new neuron to the second layer, establishing its connections with all the neurons of the first and third layers, and assigning weights to these connections.

4. Comparison of autoassociative Hamming and Hopfield neural networks

To demonstrate the effectiveness of the autoassociative Hamming network (Fig. 3), we compare it with the conventional Hopfield network [1, 2], as well as with the Hopfield network trained using a nonlocal algorithm [14], which, although it complicates the learning process, demonstrates better results in pattern recognition than the conventional (local) Hopfield learning algorithm.

Pattern recognition algorithms for a conventional Hopfield network and for a Hopfield network trained by a nonlocal algorithm [14] are the same and are described by equation (3.7). The nonlocal learning algorithm of the Hopfield network is described by equations [14]:

$$\sigma_i^{(M)} = x_i^{(M)} - \frac{1}{N} \sum_{j=1}^N w_{ij}^{(M-1)} x_j^{(M)} \quad (4.1)$$

$$w_{ij}^{(M)} = w_{ij}^{(M-1)} + \sigma_i^{(M)} \sigma_j^{(M)} \text{ for } i \neq j \quad \text{and} \quad w_{ij}^{(M)} = 0 \text{ for } i = j \quad (4.2)$$

where $M = 1, 2, \dots$ is the learning step number, and $X^{(M)}$ is the reference pattern selected at the M th learning step.

The learning process (4.1), (4.2) is iterative. Initially, it is assumed that $w_{ij}^{(0)} = 0$ for all i and j . Then, at each learning step, one of the reference patterns is randomly selected, and the matrix w_{ij} is corrected by formulas (4.1) and (4.2). The learning process is repeated many times, so that each reference pattern takes part in correcting the matrix w_{ij} a specified number of times (on average). For this, the number of learning cycles is chosen to be equal to kL , where k is the average multiplicity of participation of each pattern in the learning (usually, $k = 5 \dots 10$) and L is the number of reference patterns.

Comparison of pattern recognition algorithms was carried out on the reference patterns [2] shown in Fig. 5. The correlation coefficients of these patterns are given in Table 1. Note that in this paper we consider simple reference patterns [2] (Fig. 5) to demonstrate the operability of the proposed autoassociative Hamming network (Fig. 3). In the future, we plan to use the much more

Fig. 5. Reference patterns [2] used in numerical experiments.

Table 1. Correlation coefficients (2.4) of reference patterns shown in Fig. 5

Pattern	0	1	2	3	4	6	9	■
0	1	0.2	0.04	0.08	0.16	-0.28	-0.28	-0.06
1	0.2	1	0.08	0.08	-0.28	0.12	0.12	0.1
2	0.04	0.08	1	0.36	0.16	0.36	-0.2	0.02
3	0.08	0.08	0.36	1	0.52	-0.08	0.36	-0.02
4	0.16	-0.28	0.16	0.52	1	-0.12	0.08	0.18
6	-0.28	0.12	0.36	-0.08	-0.12	1	-0.28	0.38
9	-0.28	0.12	-0.2	0.36	0.08	-0.28	1	-0.18
■	-0.06	0.1	0.02	-0.02	0.18	0.38	-0.18	1

complex data sets, e.g., MNIST, CIFAR, etc. for comprehensive testing of the autoassociative Hamming network under consideration.

Before starting the calculations for each algorithm, the network was trained. The Hopfield network was trained using the standard algorithm (3.8) or the nonlocal algorithm (4.1), (4.2). In both cases, it was assumed that $w_{ij} = 0$ for $i = j$.

For the autoassociative Hamming network (Fig. 3), training consisted in memorizing reference patterns in the form of vectors $X^{(1)}, \dots, X^{(L)}$.

Obviously, the autoassociative Hamming network (Fig. 3) requires significantly less memory for storing patterns than the Hopfield network. For the Hamming network, only the vectors of the reference patterns need to be stored, which takes $L \times N$ bits. For the Hopfield network, we need to store the matrix w_{ij} , which, taking into account symmetry and zero diagonal elements, takes the memory size $V_1 N(N-1)/2$, where V_1 is the number of bits needed to encode one element of the matrix w_{ij} . Taking into account that usually $L \ll N$, we conclude that the Hopfield network requires $\frac{V_1(N-1)}{2L} \gg 1$ times more memory than the Hamming network (Fig. 3).

Calculations for each neural network were carried out as follows.

1. One of the many reference patterns is selected and distorted using a random number generator. The degree of distortion is set by the parameter

$$\varphi = \frac{H}{N} 100\%, \quad (4.3)$$

where H is the number of changed bits and N is the total number of bits in the pattern (the dimension of the pattern). Thus, for the selected pattern, using a random number generator, $[(\varphi/100)N]$ bits are selected, the values of which are changed to the opposite, where $[\dots]$ means the integer part of the number.

2. The resulting distorted pattern is used as the initial condition for one of the three recognition algorithms.

3. The recognition algorithm starts: for the Hopfield network (for both training algorithms) this is algorithm (3.7), for the autoassociative Hamming network this is algorithm (3.5). In all cases, the asynchronous method of recognition is used, when at each step only one randomly selected bit of the vector X is changed.

4. The iterations continued, while during the last kN steps at least once there was a change in the vector X . In the calculations, $k = 5 \dots 10$ was taken.

For each pattern and each degree of distortion φ , from 40 to 50 numerical experiments were carried out, and the probability of recognition of this pattern with a given degree of distortion for the used recognition algorithm was estimated.

Note that the parameter φ describes the degree of distortion with respect to the direct (reference) pattern $X^{(s)}$, while the parameter $\varphi' = 100 - \varphi$ describes the degree of distortion with respect to the inverse pattern $-X^{(s)}$. In particular, $\varphi = 100\%$ ($\varphi' = 0$) corresponds to an inverse pattern. If $\varphi < 50\%$, then the distorted pattern is closer to the direct pattern; if $\varphi > 50\%$, then the distorted pattern is closer to the inverse pattern. In our calculations, direct and inverse patterns were considered to be the same, therefore, if the network recognized the presented pattern as inverse to the selected reference pattern, this was considered as correct recognition.

Some results of calculations are shown in Figs. 6 and 7.

For clarity, the results in Fig. 6 are presented for two limiting cases: for the pattern “0”, which has the smallest correlation with the other patterns (see Table 1), and for the pattern “4”, which correlates more strongly than others, and therefore is worse recognized by the Hopfield network.

Figure 6 shows that the pattern “4” is recognized by the Hopfield network much worse (even for the nonlocal learning algorithm (4.1), (4.2)) than the pattern “0”. At the same time, both patterns with almost the same probability are recognized by the autoassociative Hamming network over the whole range of distortion degree φ . This indicates a high stability in operation of the autoassociative Hamming network. As follows from Fig. 6, the probability of pattern recognition for “0” and “4” patterns using the autoassociative Hamming network (3.5) practically does not depend on the exponent m for $m \geq 3$.

Figure 7 compares the pattern recognition probabilities for different reference patterns shown in Fig. 5 using the autoassociative Hamming network. We can see that the autoassociative Hamming algorithm (3.5), in contrast to the Hopfield network, stably recognizes all reference patterns considered.

Fig. 6. Comparison of pattern recognition probabilities for “0” (upper) and “4” (lower) using different algorithms. For clarity, the probability is given as a percentage.

Fig. 7. Comparison of pattern recognition probabilities for different reference patterns (see Fig. 5) using the autoassociative Hamming network for $m = 3$.

5. Autoassociative Hamming neural network as a dynamical system

Consider the function

$$U(X) = - \sum_{s=1}^L K^{m+1} (X, X^{(s)}) \quad (5.1)$$

defined in the discrete binary space X .

We show that under certain conditions this function has local minima at the points $X = X^{(s)}$.

Consider the point $X = X^{(r)}$. At this point, the function (5.1) is

$$U(X^{(r)}) = - \sum_{s=1}^L K^{m+1} (X^{(r)}, X^{(s)}). \quad (5.2)$$

If we change the bit with the number k in the vector $X^{(r)}$ to the inverted one: $x_k^{(r)} \rightarrow -x_k^{(r)}$, the correlation functions change as follows:

$$K(X^{(r)}, X^{(s)}) \rightarrow \begin{cases} K(X^{(r)}, X^{(s)}) - \frac{2}{N} x_k^{(s)} x_k^{(r)}, & s \neq r, \\ 1 - \frac{2}{N}, & s = r. \end{cases} \quad (5.3)$$

The corresponding change in the function (5.1) is

$$\Delta U(X^{(r)}) = \sum_{\substack{s=1 \\ s \neq r}}^L \left[K^{m+1}(X^{(r)}, X^{(s)}) - \left(K(X^{(r)}, X^{(s)}) - \frac{2}{N} x_k^{(s)} x_k^{(r)} \right)^{m+1} \right] + \frac{2}{N}. \quad (5.4)$$

We use the well-known relation

$$a^{m+1} - b^{m+1} = (a - b)f_m(a, b), \quad (5.5)$$

where

$$f_m(a, b) = a^m + a^{m-1}b + a^{m-2}b^2 + \dots + b^m \quad (5.6)$$

for any a, b and natural $m = 1, 2, \dots$

Obviously, for small β

$$f_m(a, a + \beta) = (m + 1)a^m + \frac{m(m - 1)}{2}a^{m-1}\beta + \dots \quad (5.7)$$

Taking (5.6) into account, we write relation (5.4) as

$$\Delta U(X^{(r)}) = \frac{2}{N} \left(1 + x_k^{(r)} \sum_{\substack{s=1 \\ s \neq r}}^L x_k^{(s)} f_m \left[K(X^{(r)}, X^{(s)}), K(X^{(r)}, X^{(s)}) - \frac{2}{N} x_k^{(s)} x_k^{(r)} \right] \right). \quad (5.8)$$

Since $x_k^{(s)} x_k^{(r)} = \pm 1$, it is easy to verify by direct calculation that with $N \geq 100$ one can write

$$f_m \left[K(X^{(r)}, X^{(s)}), K(X^{(r)}, X^{(s)}) - \frac{2}{N} x_k^{(s)} x_k^{(r)} \right] \approx f_m \left[K(X^{(r)}, X^{(s)}), K(X^{(r)}, X^{(s)}) \right].$$

Then, taking (5.7) into account, we can rewrite (5.8) in the form

$$\Delta U(X^{(r)}) = \frac{2}{N} \left(1 + (m + 1)x_k^{(r)} \sum_{\substack{s=1 \\ s \neq r}}^L x_k^{(s)} K^m(X^{(r)}, X^{(s)}) \right). \quad (5.9)$$

According to (5.7), the error of the expression in parentheses in equation (5.9) when replacing (5.8) by (5.9) is of the order of

$$\frac{m(m - 1)}{N} \left| \sum_{\substack{s=1 \\ s \neq r}}^L x_k^{(s)} K^{m-1}(X^{(r)}, X^{(s)}) \right|.$$

Replacing (5.8) by (5.9) means that the condition

$$\frac{m(m - 1)}{N} \left| \sum_{\substack{s=1 \\ s \neq r}}^L x_k^{(s)} K^{m-1}(X^{(r)}, X^{(s)}) \right| \ll (m + 1) \left| \sum_{\substack{s=1 \\ s \neq r}}^L x_k^{(s)} K^m(X^{(r)}, X^{(s)}) \right|$$

is satisfied. This condition is satisfied if

$$N \left| K \left(X^{(r)}, X^{(s)} \right) \right| \gg 1. \quad (5.10)$$

Suppose that for any k and r the condition

$$(m+1) \left| \sum_{\substack{s=1 \\ s \neq r}}^L x_k^{(s)} K^m \left(X^{(r)}, X^{(s)} \right) \right| < 1. \quad (5.11)$$

Then for any k and r

$$\Delta U \left(X^{(r)} \right) > 0. \quad (5.12)$$

Thus, we have proved that, if condition (5.11) is satisfied, the function (5.1) has local minima at the points $X = X^{(s)}$ ($s = 1, 2, \dots, L$) of the discrete binary space. Around each point $X = X^{(s)}$ there is a basin of attraction. Being in this basin of attraction, one can always find a path to the point $X = X^{(s)}$ such that at each elementary step the function (5.1) will not increase, i. e., the following condition will be satisfied:

$$\Delta U(X) \leq 0. \quad (5.13)$$

It is easy to see that at the points $X = -X^{(s)}$, corresponding to the inverse patterns, function (5.1) has local maxima. In this case, around each point $X = -X^{(s)}$ there is also a basin of attraction. Being in this basin of attraction, one can always find a path to the point $X = -X^{(s)}$ such that at each elementary step, the function (5.1) will not decrease, i. e., the following condition will be satisfied:

$$\Delta U(X) \geq 0. \quad (5.14)$$

Consider an arbitrary point X of a discrete binary space.

If we change one of the bits (bit k) of the vector X :

$$x'_k = -x_k, \quad x'_i = x_i, \quad i \neq k, \quad (5.15)$$

then

$$\Delta U(X) = U(X') - U(X) = \sum_{s=1}^L \left[K^{m+1} \left(X, X^{(s)} \right) - K^{m+1} \left(X', X^{(s)} \right) \right]. \quad (5.16)$$

Obviously,

$$\begin{aligned} K \left(X', X^{(s)} \right) &= \frac{1}{N} \sum_{\substack{i=1 \\ i \neq k}}^N x_i x_i^{(s)} + \frac{1}{N} x'_k x_k^{(s)} = \frac{1}{N} \sum_{i=1}^N x_i x_i^{(s)} + \frac{1}{N} (x'_k - x_k) x_k^{(s)} = \\ &= K \left(X, X^{(s)} \right) + \frac{1}{N} (x'_k - x_k) x_k^{(s)}. \end{aligned} \quad (5.17)$$

Then, taking (5.5) and (5.6) into account, we rewrite (5.16) and (5.17) in the form

$$\Delta U(X) = -\frac{1}{N} (x'_k - x_k) \sum_{s=1}^L x_k^{(s)} f_m \left[K \left(X, X^{(s)} \right), K \left(X', X^{(s)} \right) \right]. \quad (5.18)$$

We require that the change (5.15) of the vector X does not lead to an increase in the function (5.1), that is, that condition (5.13) is satisfied. From relation (5.18) it follows that this is possible if at each step we take

$$x'_k = \operatorname{sgn} \sum_{s=1}^L x_k^{(s)} f_m \left[K \left(X, X^{(s)} \right), K \left(X', X^{(s)} \right) \right]. \quad (5.19)$$

Moving in the discrete binary space according to algorithm (5.19) guarantees that we arrive in the fastest way at the nearest local minimum of the function (5.1). If the initial point X lies in the basin of attraction of the point (pattern) $X^{(s)}$, then, as a result of movement (5.19), we will reach the point $X^{(s)}$ and remain there indefinitely until an external influence is exerted on it.

It is easy to see that, if a point (pattern) X is in the basin of attraction of some inverse pattern $-X^{(s)}$, then moving in accordance with algorithm (5.19) leads to an increase in the function (5.1), i. e., this guarantees that we arrive in the fastest way at the nearest local maximum of the function (5.1), which corresponds to the inverse pattern $-X^{(s)}$.

Equation (5.19) describes a discrete dynamical system that has attractors at the points $X = X^{(s)}$ and $X = -X^{(s)}$ ($s = 1, 2, \dots, L$) of a discrete binary space. Obviously, the function (5.1) is the Lyapunov function of this dynamical system.

From relation (5.17) one obtains

$$K \left(X', X^{(s)} \right) - K \left(X, X^{(s)} \right) = \frac{1}{N} (x'_k - x_k) x_k^{(s)}. \quad (5.20)$$

The right-hand side of the expression (5.20) can take the values $\left\{ -\frac{2}{N}; 0; \frac{2}{N} \right\}$.

As in the previous case, it is easy to verify that, with $N \geq 100$, one can write

$$f_m \left[K \left(X, X^{(s)} \right), K \left(X', X^{(s)} \right) \right] \approx f_m \left[K \left(X, X^{(s)} \right), K \left(X, X^{(s)} \right) \right] = (m+1) K^m \left(X, X^{(s)} \right).$$

Then, taking (5.7) into account, we write equation (5.19) in the form

$$x'_k = \operatorname{sgn} \sum_{s=1}^L x_k^{(s)} K^m \left(X, X^{(s)} \right). \quad (5.21)$$

Comparing (5.21) with (3.5), we conclude that this dynamical system is the autoassociative Hamming neural network, which was introduced in Section 3, based on intuitive considerations.

In the process of pattern recognition using an autoassociative Hamming network, the Lyapunov function (5.1) changes randomly. This is due to the randomness of the recognition algorithm itself, as well as to the random nature of the initial distortions. At the same time, in the process of pattern recognition according to algorithm (3.5), the Lyapunov function decreases monotonically with decreasing degree of distortion, as shown in Fig. 8.

6. Pattern recognition probability using Hamming distance

Let us calculate the theoretical probability of error-free pattern recognition using the Hamming distance.

Each reference pattern $X^{(s)}$ corresponds to a point in the discrete binary space X which has dimension N . Thus, the system of L reference patterns is represented by L points in this

Fig. 8. Change in the Lyapunov function (5.1) of the pattern “4” with a decrease in the degree of distortion of the pattern in the process of recognition. The initial distortion is 40%.

space. We assume that the patterns are randomly and uniformly distributed in this discrete binary space. In this case, the probability of location of any pattern at any point in space is the same.

Choose some reference pattern $X^{(s)}$ and randomly change exactly H different bits. The degree of distortion of the image $X^{(s)}$ is determined by relation (4.3).

As a result, we obtain a point X' of a discrete binary space, located at Hamming distance H from the pattern $X^{(s)}$.

The number of all points in a discrete binary space located at a distance H from the point X' is equal to

$$n(H) = C_N^H = \frac{N!}{H!(N-H)!}. \quad (6.1)$$

Then the number of points of the discrete binary space that are located at distances not exceeding H from the point X' is equal to $\sum_{k=0}^H \frac{N!}{k!(N-k)!}$. The probability that some reference pattern is inside or on the surface of the hypersphere of radius H with center at the point X' is $2^{-N} \sum_{k=0}^H \frac{N!}{k!(N-k)!}$. The probability that a reference pattern is not in this area is $\left(1 - 2^{-N} \sum_{k=0}^H \frac{N!}{k!(N-k)!}\right)$. Then the probability that none of the remaining $(L-1)$ reference

patterns are at a distance less than or equal to H from the point X' is equal to

$$p(H) = \left(1 - 2^{-N} \sum_{k=0}^H \frac{N!}{k!(N-k)!} \right)^{L-1}. \quad (6.2)$$

Thus, the expression (6.2) describes the probability that the pattern that is the closest to the point X' is the reference pattern $X^{(s)}$, from which the pattern X' was obtained by changing H different bits. Therefore, the expression (6.2) determines the probability of error-free recognition of the pattern $X^{(s)}$ upon presentation of the pattern X' .

In deriving the expression (6.2), it was implicitly assumed that $H < N/2$. If direct and inverse patterns are considered as the same object, then

$$p(N - H) = p(H). \quad (6.3)$$

The dependence (4.3), (6.2) and (6.3) is shown in Figs. 6 and 7.

Obviously, other things being equal, the pattern recognition algorithm, based on a comparison of Hamming distances, has the highest probability of recognition as compared to any other algorithms. Thus, the dependence (4.3), (6.2), (6.3) determines the maximum possible probability of pattern recognition by any algorithms. From Figs. 6 and 7, we can see that the autoassociative Hamming network (3.5) (shown in Figs. 3 and 4) has a pattern recognition probability close to its maximum possible theoretical value (6.2), (6.3) for all degrees of distortion (4.3).

7. Concluding remarks

Thus, we have shown that it is possible to create an autoassociative Hamming network operating on a principle that is similar to that of the Hopfield network, but has a much higher pattern recognition probability. Such a Hamming network is a dynamical system, while the reference patterns are attractors of this system.

From the above analysis, it follows that, to increase the probability of pattern recognition, it is necessary to reduce the correlation coefficients of the reference patterns. This can be achieved, for example, by excluding the similar areas from the reference patterns or by considering (comparing) only the most significant for recognition parts of the patterns. In the latter case, the correlation coefficient of two patterns X and Y can be determined as

$$K(X, Y) = \frac{1}{N} \sum_{i=1}^N u_i x_i y_i,$$

where $u_i \geq 0$ are the coefficients of significance of bits that satisfy the condition $\sum_{i=1}^N u_i = 1$. The larger the role of a bit (or group of bits) in the recognition process, the greater the coefficient u_i for this bit.

Numerical experiments have shown that the autoassociative Hamming network (at least with $m \geq 3$ for the reference patterns shown in Fig. 5) has very few false patterns compared to the Hopfield network: if it makes a mistake, then most often it gives one of the reference patterns. This is apparently due to the fact that the dynamic system (5.1), (5.21), at least for the examples considered, has a small number of attractors that differ from the reference patterns or are inverse to them. At the same time, the Hopfield network, even at a small degree of distortion of the

selected reference pattern ($> 20\%$), begins to make mistakes, and in many cases the mistakenly found pattern is a false one, because it does not match any of the existing reference patterns or those inverse to them. The false patterns can be of two types: (i) a compilation of existing reference patterns, when a false pattern consists of parts of two or more reference patterns and (ii) patterns that do not consist of parts of reference patterns, but are fundamentally new objects.

From the point of view of pattern recognition, any errors, including the generation of false patterns, are a drawback of the network, which should be eliminated. However, from the point of view of artificial intelligence, the generation of false patterns can be considered as the ability of a network to generate new knowledge based on existing ones. In fact, the Hopfield network and, to a lesser extent, the autoassociative Hamming network can generate patterns that they have never seen before. This can be considered as the generation of new knowledge by a neural network.

It is necessary to note that this is the basis of the human creativity. A human can invent, create and discover new things in two ways: by compiling existing knowledge (examples include images such as “mermaid” and “centaur”) and by inventing fundamentally new things which do not consist of elements of already known objects.

From this point of view, the less powerful Hopfield network is more “inventive” (more “creative”) than the more powerful Hamming network, which better recognizes patterns, but practically does not generate new knowledge (false patterns).

This property can be used to create neural networks that can perform creative work, including that of making inventions and discoveries. Thus, the neural networks can act as knowledge generators which create new knowledge based on the accumulated knowledge. However, it should be kept in mind that knowledge generation alone is not enough to create inventions and discoveries, since not all newly generated knowledge will be useful. In this case, the “struggle for survival” of ideas comes into play, which is similar to natural selection in nature, as a result of which it becomes clear which new knowledge is useful and will “survive” and which is useless and will be forgotten.

This question will be considered in forthcoming papers.

Conflict of interest

The authors declare that they have no conflict of interest.

References

- [1] Hopfield, J. J., Neural Networks and Physical Systems with Emergent Collective Computational Abilities, *Proc. Natl. Acad. Sci. USA*, 1982, vol. 79, no. 9, pp. 2554–2558.
- [2] Haykin, S., *Neural Networks: A Comprehensive Foundation*, New York: Macmillan, 1994.
- [3] Lippmann, R. P., An Introduction to Computing with Neural Nets, *IEEE ASSP Magazine*, 1987, vol. 4, no. 2, pp. 4–22.
- [4] Ikeda, N., Watta, P., Artiklar, M., and Hassoun, M. H., A Two-Level Hamming Network for High Performance Associative Memory, *Neural Netw.*, 2001, vol. 14, no. 9, pp. 1189–1200.
- [5] Yongsheng, H., HuaMei, D., and Zhongbin, T., A Logistical Model Based on the Hamming Competitive Neural Network Algorithm, *J. Appl. Sci.*, 2014, vol. 14, no. 2, pp. 129–136.
- [6] Fan, L., Revisit Fuzzy Neural Network: Demystifying Batch Normalization and ReLU with Generalized Hamming Network, in *31st Conf. on Neural Information Processing Systems (Long Beach, Calif., 2017)*, pp. 1923–1932.

- [7] Koutroumbas, K. and Kalouptsidis, N., Generalized Hamming Networks and Applications, *Neural Netw.*, 2005, vol. 18, no. 7, pp. 896–913.
- [8] Schmid, A., Leblebici, Y., and Mlynek, D., Hardware Realization of a Hamming Neural Network with On-Chip Learning, in *Proc. of the 1998 IEEE Internat. Symp. on Circuits and Systems (ISCAS'98): Vol. 3*, Cat. No. 98CH36187, pp. 191–194.
- [9] Norouzi, M., Fleet, D. J., and Salakhutdinov, R. R., Hamming Distance Metric Learning, in *NIPS'12: Proc. of the 25th Internat. Conf. on Neural Information Processing Systems: Vol. 1*, pp. 1061–1069.
- [10] Khristodulo, O. I., Makhmutov, A. A., and Sazonova, T. V., Use Algorithm Based at Hamming Neural Network Method for Natural Objects Classification, *Procedia Comput. Sci.*, 2017, vol. 103, pp. 388–395.
- [11] Kovačević, V. B., Gavrovska, A. M., and Paskaš, M. P., High-Speed Implementation of Hamming Neural Network, in *Proc. of the 10th Symp. on Neural Network Applications in Electrical Engineering (Belgrade, Serbia, Sept 2010)*, pp. 167–170.
- [12] Lu, W., Li, Z., and Shi, B., A Modified Hamming Neural Network, in *Proc. of the 4th Internat. Conf. on Solid-State and Integrated Circuit Technology (Beijing, China, Oct 24–28, 1995)*, pp. 694–696.
- [13] Klimov, V. S., Klimov, A. S., and Mkrtychev, S. V., Computer Diagnostics of Resistance Spot Welding Based on Hamming Neural Network, *J. Phys. Conf. Ser.*, 2019, vol. 1333, no. 4, 042015, 6 pp.
- [14] Denker, J. S., Neural Network Models of Learning and Adaptation, *Phys. D*, 1986, vol. 22, nos. 1–3, pp. 216–232.