Associative-memory model based on neural networks: modification of Hopfield model

Sang-Hoon Oh, Tae-Hoon Yoon, and Jae Chang Kim

Department of Electronics Engineering, Pusan National University, Pusan, Korea

Received July 24, 1987; accepted October 13, 1987

The problem with the Hopfield associative-memory model caused by an imbalance between the number of ones and zeros in each stored vector is studied, and a modification of the Hopfield model that works well irrespective of the number of ones (or zeros) is proposed. This modified model can be implemented with no increase in memory.

Since Hopfield[1] introduced a model for associative memory based on neural networks, there has been increasing interest in associative memories.[2–12] Through the use of the Hopfield model, information can be stored and retrieved from partial information. There exist, however, some limitations[2–5] that must be overcome before the model can be realized. One of these limitations is that the Hopfield model requires each stored vector to have an approximately equal number of ones and zeros.[6] If there is an imbalance, the model has a high probability of failure to find the nearest neighbor.[5] In this Letter we propose a modification of the Hopfield model that works well irrespective of the number of ones (or zeros).

The Hopfield neural-network model[1] consists of N mutually interconnected neurons, whose current states are characterized by a binary state vector $\mathbf{v} = [v_1, v_2, \ldots, v_N]$, with v_i (1 or 0) denoting the state of neuron i. A set of M state vectors $\mathbf{v}^{(m)}$, each N bits long, can be stored in the network with a strength t_{ij} of the interconnection between neuron i and neuron j, given by

$$t_{ij} = \sum_{m=1}^{M} [2v_i^{(m)} - 1][2v_j^{(m)} - 1] - M\delta_{ij}, \qquad i, j = 1, 2, \ldots, N, \tag{1}$$

where δ_{ij} is a Kronecker delta function (if $i = j$, $\delta_{ij} = 1$; otherwise, $\delta_{ij} = 0$). The next state of neuron i is determined by the current state of other neurons as

$$v_i(\text{next state}) = \begin{cases} 1 & \text{for } \hat{v}_i > w_i \\ 0 & \text{otherwise} \end{cases}, \tag{2}$$

where

$$\hat{v}_i = \sum_{j=1}^{N} t_{ij} v_j \tag{3}$$

and w_i represents a fixed threshold value for neuron i. Through this iterative procedure of determining the next state from the current state, an initial state vector given by the input vector should converge to its nearest neighbor among the stored vectors.

In order to implement the Hopfield model, we must determine the threshold value w_i for each neuron. Its value may depend on the vectors already stored or those that are to be stored. In previous publications[1–6] w_i was chosen to be zero for all i, a choice based on the assumption[6] that the number of ones and zeros in each stored vector is nearly equal. However, in general, this assumption will not be satisfied. In such cases it is known that the convergence of the Hopfield model to the nearest neighbor is poor.

When the number of ones in each stored vector differs from vector to vector, we find that the vectors with a large number of ones are more likely to become final stable vectors. To illustrate this, we stored three vectors, **a, b,** and **c,** each 20 bits long (see Table 1). For the input vector the smallest Hamming distance from vector **c,** the state vector was expected to converge to the vector **c** since it is the nearest neighbor of the input vector. However, it converged to the vector

Table 1. Results of Computer Simulation with the Hopfield Model

Stored vectors	
a:	11110000111100001111
b:	10101000111011001000
c:	10100000100010001001
Input vector[a]	
	11000000000000000000
Retrieved vector[b]	
	11110000111100001111
Input vector[c]	
	10100000100010001001
Retrieved vector[d]	
	10100000111010001001

[a] Smallest Hamming distance from vector **c**.
[b] Vector **a**.
[c] Vector **c**.
[d] False vector.

a, which is the stored vector with the largest number of ones. In cases when the number of ones in a stored vector is much smaller than the number of ones in other stored vectors, that stored vector may be unretrievable. The table also shows that when the vector **c** is the input (6 ones), the state vector converges to some false vector.

The large difference between the number of ones and zeros also makes the convergence of the Hopfield model poor even when all vectors have the same number of ones. To illustrate this, we stored three vectors **d, e,** and **f,** each 20 bits long (see Table 2). When the vector **d** was the input, it converged to a false vector that differed from **d** by 5 bits, as shown in the table.

To overcome the problems caused by the imbalance between the number of ones and zeros, we present a modification of the Hopfield model. We double the length of the vectors to be stored by adding a complement vector to each vector. Then the extended vector $\mathbf{u} = [u_1, u_2, \ldots, u_{2N}]$ is defined as

$$u_i = \begin{cases} v_i & \text{for } 1 \le i \le N \\ 1 - v_{i-N} & \text{for } N+1 \le i \le 2N \end{cases}. \tag{4}$$

Application of the Hopfield algorithm to the vector **u** results in a new memory matrix $\mathbf{T}' = [t_{ij}']$:

$$t_{ij}' = \sum_{m=1}^{M} [2u_i^{(m)} - 1](2u_j^{(m)} - 1) - M\delta_{ij}, \qquad i, j = 1, 2, \ldots, 2N. \tag{5}$$

The next state of u_i is found from

$$u_i(\text{next state}) = \begin{cases} 1 & \text{for } \hat{u}_i > 0 \\ 0 & \text{otherwise} \end{cases}, \tag{6}$$

where

$$\hat{u}_i = \sum_{j=1}^{2N} t_{ij}' u_j. \tag{7}$$

The number of ones in the coded vector **u** will always be equal to the number of zeros. Therefore application of the Hopfield algorithm to the vector **u** instead of to the vector **v** solves the problem caused by an imbalance between the number of ones and zeros. The algorithm given by Eqs. (5)–(7) seemingly requires an increase in memory space by a factor of 4. However, it can be implemented with no increase in memory, as we now describe.

By using Eqs. (1) and (4), elements of the memory matrix $\mathbf{T}'$ defined by Eq. (5) may be represented in terms of elements of the matrix $\mathbf{T} = [t_{ij}]$ as

$$t_{ij}' = \begin{cases} t_{ij} & \text{for } 1 \le i, j \le N \\ -t_{i-N,j} - M\delta_{i-N,j} & \text{for } 1 \le i - N, j \le N \\ -t_{i,j-N} - M\delta_{i,j-N} & \text{for } 1 \le i, j - N \le N \\ t_{i-N,j-N} & \text{for } 1 \le i - N, j - N \le N \end{cases}. \tag{8}$$

Then, by using Eq. (8), Eq. (7) may be rewritten as

$$\hat{u}_i = \begin{cases} \sum_{j=1}^{N} t_{ij}(2v_j - 1) + M(v_i - 1) & \text{for } 1 \le i \le N \\ -\sum_{j=1}^{N} t_{i-N,j}(2v_j - 1) - Mv_{i-N} & \text{for } N+1 \le i \le 2N \end{cases}. \tag{9}$$

With Eqs. (6) and (9), the next state of the vector **u** can be determined by the memory matrix **T** and the current state of the vector **v.** As we know from Eq. (4), the next state of u_i, $1 \le i \le N$, will be the next state of v_i.

Based on the description given above, the model that we propose may be summarized as follows: For a given current state of the vector **v** and the memory matrix **T,** the next state of the vector **v** is given by

$$v_i(\text{next state}) = \begin{cases} 1 & \text{for } \tilde{v}_i > 0 \\ 0 & \text{otherwise} \end{cases}, \tag{10}$$

where

$$\tilde{v}_i = \sum_{j=1}^{N} t_{ij}(2v_j - 1) - M(1 - v_i). \tag{11}$$

This modified algorithm will act as if the length of the original vector were doubled so that the coded vectors had an equal number of ones and zeros. Therefore we overcome the problem in the Hopfield model caused by an imbalance between the number of ones and zeros simply by replacing Eq. (3) with Eq. (11) in the determination of the next state of neuron i. We simply replace v_j by $2v_{j-1}$ and subtract $M(1 - v_i)$, the complement of the current state of neuron i multiplied by the number of stored vectors. Thus in the modified model any neuron's next state is affected by its current state as well as by the current state of other neurons. It is also worth mentioning that since the modified algorithm is able to deal with vectors having

Table 2. Results of Computer Simulation with the Hopfield Model

Stored Vectors	
d:	00010010010000010010
e:	00100100000100100100
f:	01000001001001000001
Input vector[a]	
	00010010010000010010
Retrieved vector[b]	
	10011010110010011010

[a] Vector **d**.
[b] False vector.

Table 3. Results of Computer Simulation with the Modified Model

Input vector[a]
11000000000000000000
Retrieved vector[b]
10100000100010001001
Input vector[c]
00010010000000000000
Retrieved vector[d]
00010010010000010010

[a] Smallest Hamming distance from vector **c**; stored vectors same as in Table 1.
[b] Vector **c.**
[c] Stored vectors same as in Table 2.
[d] Vector **d.**

unequal numbers of ones and zeros, for a given memory size more information may be stored with the modified model than with the Hopfield model.

As is shown in Table 3, the modified algorithm works well even for an initial state vector that does not converge to its nearest neighbor in the Hopfield model. For example, the vector **c,** which previously could not be a final state vector even when **c** was an input, becomes a final stable vector with the modified model. An input vector of Hamming distance 6 from the vector **c** also converges to the nearest-neighbor vector **c** (see Table 3). Furthermore, for the examples shown in Table 2 in which the number of ones in each stored vector differs significantly from the number of zeros, an input vector of Hamming distance 3 from the vector **d,** as well as the vector **d** itself, converges to the nearest-neighbor vector **d** with the modified model (see Table 3).

Finally, it should be noted that we obtained the same results found in Table 3 with the memory matrix **T** clipped[2] [for $t_{ij} > 0$, t_{ij}(clipped) = 1; for $t_{ij} < 0$, t_{ij}(clipped) = -1]. This makes implementation of the model easier.

In conclusion, we have proposed a modification of the Hopfield model that works well irrespective of the number of ones in each stored vector. This modified model can, in fact, increase the amount of information stored for a given memory size.

References

1. J. J. Hopfield, Proc. Natl. Acad. Sci. USA **79,** 2554 (1982).
2. N. Farhat, D. Psaltis, A. Prata, and E. Paek, Appl. Opt. **24,** 1469 (1985).
3. B. L. Montgomery and B. V. K. V. Kumar, Appl. Opt. **25,** 3759 (1986).
4. B. Macukow and H. H. Arsenault, Appl. Opt. **26,** 34 (1987).
5. B. Macukow and H. H. Arsenault, Appl. Opt. **26,** 924 (1987).
6. D. Psaltis and N. Farhat, Opt. Lett. **10,** 98 (1985).
7. R. A. Athale, H. H. Szu, and C. B. Friedlander, Opt. Lett. **11,** 482 (1986).
8. A. Yariv, S.-K. Kwong, and K. Kyuma, Appl. Phys. Lett. **48,** 1114 (1986).
9. B. H. Soffer, G. J. Dunning, Y. Owechko, and E. Marom, Opt. Lett. **11,** 118 (1986).
10. G. J. Dunning, E. Marom, Y. Owechko, and B. H. Soffer, Opt. Lett. **12,** 346 (1987).
11. E. G. Paek and D. Psaltis, Opt. Eng. **26,** 428 (1987).
12. A. Moopenn, J. Lambe, and A. P. Thakoor, IEEE Trans. Syst. Man Cybern. **SMC-17,** 325 (1987).