Sādhanā, Vol. 19, Part 2, April 1994, pp. 239–255. © Printed in India.

Convergence of higher-order two-state neural networks with modified updating

M VIDYASAGAR

Centre for AI & Robotics, Raj Bhavan Circle, High Grounds, Bangalore 560 001, India

E-mail: sagar@yantra.ernet.in

MS received 9 September 1993; revised 25 February 1994

Abstract. The Hopfield network is a standard tool for maximizing a *quadratic* objective function over the discrete set $\{-1, 1\}^n$. It is well-known that if a Hopfield network is operated in an *asynchronous* mode, then the state vector of the network converges to a local maximum of the objective function; if the network is operated in a *synchronous* mode, then the state vector either converges to a local maximum, or else goes into a limit cycle of length two. In this paper, we examine the behaviour of *higher-order* neural networks, that is, networks used for maximizing objective functions that are not necessarily quadratic. It is shown that one can assume, without loss of generality, that the objective function to be maximized is *multilinear*. Three methods are given for updating the state vector of the neural network, called the asynchronous, the best neighbour and the gradient rules, respectively. For Hopfield networks with a quadratic objective function, the asynchronous rule proposed here reduces to the standard asynchronous updating, while the gradient rule reduces to synchronous updating; the best neighbour rule does not appear to have been considered previously. It is shown that both the asynchronous updating rule and the best neighbour rule converge to a local maximum of the objective function within a finite number of time steps. Moreover, under certain conditions, under the best neighbour rule, each global maximum has a nonzero radius of direct attraction; in general, this may not be true of the asynchronous rule. However, the behaviour of the gradient updating rule is not well-understood. For this purpose, a *modified* gradient updating rule is presented, that incorporates both *temporal as well as spatial* correlations among the neurons. For the modified updating rule, it is shown that, after a finite number of time steps, the network state vector goes into a limit cycle of length m, where m is the degree of the objective function. If $m = 2$, i.e., for quadratic objective functions, the modified updating rule reduces to the synchronous updating rule for Hopfield networks. Hence the results presented here are "true" generalizations of previously known results.

Keywords. Neural dynamics; Hopfield networks; higher-order networks; modified updating.

1. Introduction

A vast majority of the current research into feedback neural networks has been focused on the so-called *Hopfield networks*, whose dynamics are described by the equation

$$x_i(t+1) = \text{sign}\left[\sum_{j=1}^{n} w_{ij}x_j(t) + \theta_i\right], \quad i = 1, \ldots, n, \tag{1}$$

where n is the number of neurons, $x_i(t) \in \{-1, 1\}$ is the state of neuron i at time t, w_{ij} is the weight of the interconnection from neuron i to neuron j, and $-\theta_i$ is the firing threshold of neuron i. Hopfield (1982) defines the energy of the network as[1]

$$E = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}x_i x_j + \sum_{i=1}^{n} x_i\theta_i, \tag{2}$$

and proves the following property: Suppose $w_{ji} = w_{ij}$ for all i,j (symmetric interactions), and $w_{ii} = 0$ for all i (no self-interactions). Finally, suppose the neural states are updated *asynchronously*, as follows: At each (discrete) instant of time t, select an integer $i \in \{1, \ldots, n\}$ at random, compute $x_i(t+1)$ in accordance with (1), but leave $x_j(t)$ unchanged for all $j \neq i$. In this mode of operation, it is true that

$$E[\mathbf{x}(t+1)] \geqslant E[\mathbf{x}(t)], \tag{3}$$

where $\mathbf{x} = [x_1 \cdots x_n]^t$. Thus, in an asynchronous mode of operation, the neural network will eventually reach a fixed point of the network, that is, a vector $\mathbf{x}_0$ with the property that

$$\mathbf{x}(t) = \mathbf{x}_0 \rightarrow \mathbf{x}(t+1) = \mathbf{x}_0, \tag{4}$$

irrespective of which neuron is updated at time t. Goles *et al* (1985) prove that, if the network is updated *synchronously*, that is, at time t the states of *all* neurons are updated according to (1), then the network either converges to a fixed point, or else goes into a limit cycle of length two. See Bruck & Goodman (1988) for a unification of both convergence results, and Kamp & Hasler (1990) for a book-length treatment of the topic.

Therefore, in the case where it is desired to optimize a quadratic function of the form (2) over a finite set $\{-1, 1\}^n$, the behaviour of the corresponding neural network (1) is well-understood. It is now known (see, e.g., Hopfield & Tank 1985) that several NP-complete problems can be formulated as the minimization of a quadratic function of the type (2). However, there are situations in which it is more natural to use an objective function which is a polynomial of degree three or higher. One such example is given in Masti & Vidyasagar (1991), wherein the problem of checking whether or not there exists a truth assignment on a set of Boolean variables that makes each of a set of formulas true (commonly known as the "satisfiability problem" and the "original" NP-complete problem), is formulated as a minimization problem over the set $\{0, 1\}^n$ where n is the number of literals, and the degree of the objective function

[1] Actually, the energy function defined by Hopfield is the *negative* of this E, but this is a minor difference.

is the length of the longest clause in the set of formulas. Another example is given in Bruck & Blaum (1989), wherein the problem of algebraic block-coding is formulated as that of maximizing a polynomial over $\{-1,1\}^n$, where the number of neurons n equals the length of the encoded words, and the degree of the objective function is equal to the number of information bits.

In this paper, we examine the use of neural networks for maximizing objective functions that are not necessarily quadratic, over the discrete set $\{-1,1\}^n$. It is shown that one can assume, without loss of generality, that the objective function to be maximized is *multilinear*. Three methods are given for updating the state vector of the neural network, called the asynchronous, the best neighbour and the gradient rules, respectively. For Hopfield networks with a quadratic objective function, the asynchronous rule proposed here reduces to the standard asynchronous updating, while the gradient rule reduces to synchronous updating; the best neighbour rule does not appear to have been considered previously. It is shown that both the asynchronous updating rule and the best neighbour rule converge to a local maximum of the objective function within a finite number of time steps. Moreover, under certain conditions, under the best neighbour rule, each global maximum has a nonzero radius of direct attraction; in general, this may not be true of the asynchronous rule. However, the behaviour of the gradient updating rule is not well-understood. For this purpose, a *modified* gradient updating rule is presented, that incorporates both *temporal as well as spatial* correlations among the neurons. For the modified updating rule, it is shown that, after a finite number of time steps, the network state vector goes into a limit cycle of length m, where m is the degree of the objective function. If $m = 2$, i.e., for quadratic objective functions, the modified updating rule reduces to the synchronous updating rule for Hopfield networks. Hence the results presented here are "true" generalizations of previously known results.

2. Updating rules, fixed points and local maxima

In this section, we briefly study the problem of maximizing a polynomial $E(\mathbf{x})$ as $\mathbf{x}$ varies over the discrete set $\{-1,1\}^n$. Three types of updating rules are introduced, namely: asynchronous, best neighbour, and gradient. The relationship between fixed points (under each type of updating rule) and local maxima is explored.

(1) *Asynchronous updating.* If $\mathbf{x}\in\{-1,1\}^n$ is the current state, choose an index $i\in\{1,\ldots,n\}$ at random. Define $\mathbf{y}\in\{-1,1\}^n$ by

$$y_i = -x_i, \quad y_j = x_j, \quad \text{for } j \neq i. \tag{5}$$

Note that $\mathbf{y}$ is at a Hamming distance of one from $\mathbf{x}$. Set the next state equal to $\mathbf{y}$ if $E(\mathbf{y}) > E(\mathbf{x})$, and equal to $\mathbf{x}$ if $E(\mathbf{y}) \leqslant E(\mathbf{x})$. Note that if the next state is set equal to $\mathbf{y}$ with a probability

$$p = 1/[1 + \exp(-\Delta E/T)], \tag{6}$$

where $\Delta E = E(\mathbf{y}) - E(\mathbf{x})$ and T is the "temperature" of the network, then we get the simulated annealing algorithm (see e.g. van Laarhoven & Aarts 1987).

(2) *Best neighbour updating.* Suppose $\mathbf{x}\in\{-1,1\}^n$ is the current state. Let $N(\mathbf{x})$

denote the set of nearest neighbours of $\mathbf{x}$ in the sense of the Hamming distance; that is, let $N(\mathbf{x})$ consist of all vectors in $\{-1,1\}^n$ that differ from $\mathbf{x}$ in exactly one component. If $E(\mathbf{y}) \leqslant E(\mathbf{x})$ for all $\mathbf{y} \in N(\mathbf{x})$, then set the next state equal to $\mathbf{x}$. Otherwise, set the next state to be a "best neighbour" of $\mathbf{x}$, that is, a vector $\mathbf{y}_0 \in N(\mathbf{x})$ such that

$$E(\mathbf{y}_0) \geqslant E(\mathbf{y}), \quad \text{for all } \mathbf{y} \in N(\mathbf{x}). \tag{7}$$

The difference between asynchronous updating and best neighbour updating is that in the latter case, E is evaluated at *all* nearest neighbours of $\mathbf{x}$ before the next state in determined, whereas in the former case, E is evaluated at a *single randomly chosen* neighbour of $\mathbf{x}$.

(3) *Gradient updating.* Let $\mathbf{x} \in \{-1,1\}^n$ be the current state. Let $\nabla E(\mathbf{x}) \in \mathfrak{R}^n$ denote the gradient of E evaluated at $\mathbf{x}$. Define the next state by

$$x_i(t+1) = \operatorname{sign}[\nabla E(\mathbf{x})]_i, \quad i = 1, \ldots, n, \tag{8}$$

or, more compactly,

$$\mathbf{x}(t+1) = \operatorname{sign}[\nabla E(\mathbf{x})]. \tag{9}$$

If some component of $\nabla E(\mathbf{x})$ equals zero, say $[\nabla E(\mathbf{x})]_i = 0$, then define $x_i(t+1) = x_i(t)$.

This brings up a subtle point that is worth stating explicitly. In both asynchronous and best neighbour updating rules, the expression for $E(\mathbf{x})$ is unimportant, and only the set of values of E at the 2^n points in $\{-1,1\}^n$ is important. However, in the gradient updating rule, the *form* of the function $E(\mathbf{x})$ is also important. To illustrate this point, let $E(\mathbf{x})$ be some function defined on $\mathfrak{R}^n$, and define

$$E_1(\mathbf{x}) = x_1^2 E(\mathbf{x}). \tag{10}$$

Since $x_1^2 = 1$ whenever $x_1 = \pm 1$, it follows that both E and E_1 have the same values over $\{-1,1\}^n$. Hence both the asynchronous and the best neighbour updating rules give identical updates irrespective of whether E or E_1 is used as the objective function. However,

$$\frac{\partial E_1}{\partial x_1} = x_1^2 \frac{\partial E}{\partial x_1} + 2x_1 E = \frac{\partial E}{\partial x_1} + 2x_1 E \neq \frac{\partial E}{\partial x_1} \tag{11}$$

in general. Therefore, in the case of gradient updating, it *does* matter whether E or E_1 is used as the objective function.

The next result shows a way around this difficulty by showing that, given the values of the objective function on $\{-1,1\}^n$, there exists a *unique multilinear polynomial* that attains the same values. Recall that a function $E(\mathbf{x})$ is said to be *multilinear* if, for each fixed index $i \in \{1, \ldots, n\}$, $E(\mathbf{x})$ is an affine function of x_i. In other words, if $E(\mathbf{x})$ is written out as a polynomial in the n variables $x_1, \ldots, x_n$, no variable x_i appears with an exponent greater than one. This is a reasonable assumption, because $x_i^2 = 1$ whenever $\mathbf{x} \in \{-1,1\}^n$. If $E(\mathbf{x})$ is a quadratic function of the form (2), then E is multilinear if and only if $w_{ii} = 0$ for all i. More generally, a polynomial E is multilinear if and only if

$$\partial^2 E / \partial x_i^2 = 0, \quad \text{for all } i. \tag{12}$$

Thus, assuming that the objective function is a multilinear polynomial is a generalization of the "no self-interactions" assumption for Hopfield networks.

Lemma 1. *Suppose* $E:\mathfrak{R}^n \to \mathfrak{R}$ *is an arbitrary function. Then there exists a unique multilinear polynomial* $F:\mathfrak{R}^n \to \mathfrak{R}$ *such that*

$$E(\mathbf{z}) = F(\mathbf{z}), \quad \text{for all } \mathbf{z} \in \{-1, 1\}^n. \tag{13}$$

Proof. Given the polynomial $E(\mathbf{z})$, perform the substitution

$$z_i = 2x_i - 1, \quad x_i = (z_i + 1)/2. \tag{14}$$

Then it is easy to see that $x_i = 1$ or 0, if $z_i = 1$ or -1. By making this substitution, the multilinear polynomial $E(\mathbf{z})$ gets transformed into another polynomial $\bar{E}(\mathbf{x})$, which is also multilinear. Also, as $\mathbf{z}$ varies over the 2^n bipolar vectors in $\{-1, 1\}^n$, the corresponding vector $\mathbf{x}$ varies over the 2^n binary vectors $\{0, 1\}^n$. Hence, in order to prove the lemma, it is enough to establish that there exists a multilinear polynomial $\bar{F}(\mathbf{x})$ such that

$$\bar{E}(\mathbf{x}) = \bar{F}(\mathbf{x}), \quad \text{for all } \mathbf{x} \in \{0, 1\}^n. \tag{15}$$

Then, by back-substituting for $\mathbf{x}$ in terms of $\mathbf{z}$, one can recover the desired multinear polynomial $F(\mathbf{z})$. Hence attention is focused on establishing the relationship (15). For convenience, the "bars" on the symbols E and F are dropped.

Every multilinear polynomial in the variables $x_1, \ldots, x_n$ can be written in the form

$$F(\mathbf{x}) = \sum_{\mathbf{i} \in \{0,1\}^n} c_{\mathbf{i}} x_1^{i_1} \cdots x_n^{i_n}, \tag{16}$$

where $\mathbf{i} = \{i_1, \ldots, i_n\}$ and 0° is taken as 1. Given the function E, determine the family of 2^n (not necessarily distinct) values $(E(\mathbf{x}), \mathbf{x} \in \{0, 1\}^n)$. It is first shown that it is possible to select the coefficients $c_{\mathbf{i}} \mathbf{i} \in \{0, 1\}^n$, such that (15) holds. First, let $\mathbf{x} = \mathbf{0}$. Then, from (16),

$$F(\mathbf{0}) = c_{\mathbf{0}}. \tag{17}$$

Hence, set $c_{\mathbf{0}} = E(\mathbf{0})$. Next, let $\mathbf{x} = \{0 \cdots 0\,1\,0 \cdots 0\}$, where the 1 is in the jth position. Then

$$F(\mathbf{x}) = c_{\mathbf{0}} + c_{\{0 \cdots 0\,1\,0 \cdots 0\}}. \tag{18}$$

Thus set

$$c_{\{0 \cdots 0\,1\,0 \cdots 0\}} = E(\{0 \cdots 0\,1\,0 \cdots 0\}) - c_{\mathbf{0}}. \tag{19}$$

In this way, the n coefficients $c_{\{1\,0 \cdots 0\}}$ through $c_{\{0 \cdots 0\,1\}}$ can be determined. Next, choose $\mathbf{x}$ to have exactly two nonzero components, and so on. At the end of the process, we will have chosen a set of coefficients $c_{\mathbf{i}} \mathbf{i} \in \{0, 1\}^n$ such that the function F defined by (16) satisfies (15). To show that this choice is unique, suppose $\bar{F}$ is another multilinear polynomial that also satisfies (15) (with F replaced by $\bar{F}$ of course). Then $\Delta F(\mathbf{x}) = F(\mathbf{x}) - \bar{F}(\mathbf{x})$ is also a multilinear polynomial, and $\Delta F(\mathbf{x}) = 0$ for all $\mathbf{x} \in \{0, 1\}^n$. Now a repetition of the above argument shows that all coefficients of ΔF are zero; that is, $\Delta F(\mathbf{x}) = 0$ for all $\mathbf{x} \in \mathfrak{R}^n$. □

Example 1. Suppose $n = 2$, and let

$$E(z_1, z_2) = z_1 \exp(z_2).$$

It is desired to find a multilinear polynomial $F(z_1, z_2)$ such that (13) holds. For this purpose, we first make the transformation

$$z_1 = 2x_1 - 1, \quad z_2 = 2x_2 - 1.$$

This gives

$$\bar{E}(x_1, x_2) = (2x_1 - 1)\exp(2x_2 - 1).$$

Now the function $\bar{E}$ is evaluated at the four vectors $(x_1, x_2) = (0,0), \ldots, (1,1)$. This gives

$$\bar{E}(0,0) = E(-1,-1) = -e^{-1}, \quad \bar{E}(1,0) = E(1,-1) = e^{-1},$$
$$\bar{E}(0,1) = E(-1,1) = -e, \quad \bar{E}(1,1) = E(1,1) = e.$$

Now set up the multilinear form

$$\bar{F}(x_1, x_2) = c_{00} + c_{10}x_1 + c_{01}x_2 + c_{11}x_1x_2.$$

In order for (15) to hold, it is necessary that

$$c_{00} = \bar{E}(0,0) \Rightarrow c_{00} = -e^{-1},$$
$$c_{00} + c_{10} = \bar{E}(1,0) \Rightarrow c_{10} = 2e^{-1},$$
$$c_{00} + c_{01} = \bar{E}(0,1) \Rightarrow c_{01} = -e + e^{-1},$$
$$c_{00} + c_{10} + c_{01} + c_{11} = \bar{E}(1,1) \Rightarrow c_{11} = 2e - 2e^{-1}.$$

This uniquely determines $\bar{F}(x_1, x_2)$. The desired multilinear polynomial $F(z_1, z_2)$ that satisfies (13) can now be determined by substituting

$$x_1 = (2z_1 + 1)/2, \quad x_2 = (2z_2 + 1)/2.$$

The details are omitted as they are rather simple. □

If the function E is already a polynomial, then it is routine to determine the corresponding *multilinear* polynomial. F such that (13) holds. It is only necessary to replace each term of the form x_i^{2k} by 1, and each term of the form x_i^{2k+1} by x_i, because $x_i^2 = 1$ whenever $x_i \in \{-1, 1\}^n$. For example, consider the polynomial $E: \mathfrak{R}^4 \to \mathfrak{R}$ defined by

$$E(\mathbf{x}) = x_1^2x_2x_3 + x_1x_2^3x_4 + x_2x_4 + x_3.$$

Then

$$F(\mathbf{x}) = x_2x_3 + x_1x_2x_4 + x_2x_4 + x_3.$$

COROLLARY 2

Define the $2^n \times 2^n$ matrix M as follows: Order the 2^n elements of $\{-1, 1\}^n$ in some order, and do the same for the 2^n elements of $\{0, 1\}^n$. Let

$$m_{\mathbf{i},\mathbf{x}} = x_1^{i_1} \cdots x_n^{i_n}, \quad \mathbf{i} \in \{0,1\}^n, \quad \mathbf{x} \in \{-1,1\}^n. \tag{20}$$

Then M is nonsingular.

With the aid of lemma 1, we can find a *unique* multilinear polynomial E with prescribed values on $\{-1,1\}^n$. If E is chosen as this polynomial, then the gradient updating rule is unambiguous.

Next, the concepts of fixed point and local maximum are introduced.

DEFINITION 1

A vector $\mathbf{x}_0 \in \{-1,1\}^n$ is said to be a *fixed point* of the neural network with respect to the objective function E and a particular updating rule, if

$$\mathbf{x}(0) = \mathbf{x}_0 \Rightarrow \mathbf{x}(1) = \mathbf{x}_0, \tag{21}$$

under that updating rule.

Note that, if the asynchronous updating rule is used, then a vector $\mathbf{x}_0$ must satisfy (21) irrespective of *which* index i is chosen, in order to qualify as a fixed point.

DEFINITION 2

A vector $\mathbf{x} \in \{-1,1\}^n$ is said to be a *local maximum* of the objective function E if

$$E(\mathbf{x}) \geqslant E(\mathbf{y}), \quad \text{for all } \mathbf{y} \in N(\mathbf{x}), \tag{22}$$

where $N(\mathbf{x})$ denotes the set of n vectors that lie at a Hamming distance of one from $\mathbf{x}$. It is said to be a *strict local maximum* if

$$E(\mathbf{x}) > E(\mathbf{y}), \quad \text{for all } \mathbf{y} \in N(\mathbf{x}). \tag{23}$$

As a prelude to the main result of this section, we give an alternate interpretation of asynchronous updating.

Lemma 3. *Define an updating rule as follows: Given the current state $\mathbf{x}(t)$ and an objective function $E(\mathbf{x})$, choose an index $i \in \{1,\ldots,n\}$ at random, and define*

$$x_i(t+1) = \operatorname{sign}\{\nabla E[\mathbf{x}(t)]_i\}, \quad x_j(t+1) = x_j(t), \quad j \neq i. \tag{24}$$

If the objective function E is multilinear, then the above updating rule is identical to the asynchronous updating rule.

Note. The assumption that E is multilinear is essential; the lemma is *not* true for arbitrary objective functions.

Proof. Let $\bar{\mathbf{x}}_i$ denote the $(n-1)$-dimensional vector obtained by *omitting* the ith component of $\mathbf{x}$; that is,

$$\bar{\mathbf{x}}_i = [x_1 \cdots x_{i-1} x_{i+1} \cdots x_n]^t. \tag{25}$$

The multilinearity of the function E implies that there exist functions $\bar{E}_i: \mathfrak{R}^{n-1} \to \mathfrak{R}$ and $c_i: \mathfrak{R}^{n-1} \to \mathfrak{R}$ such that

$$E(\mathbf{x}) = x_i \bar{E}_i(\bar{x}_i) + c_i(\bar{\mathbf{x}}_i). \tag{26}$$

Notice also that $\bar{E}_i(\bar{\mathbf{x}}_i) = [\nabla E(\mathbf{x})]_i$. Therefore, if the ith component of $\mathbf{x}(t)$ is replaced by $-x_i(t)$, then the resulting change in E is given by

$$\Delta E = -2x_i(t)\{\nabla E[\mathbf{x}(t)]_i\}. \tag{27}$$

In the asynchronous updating rule, $x_i(t+1)$ is set equal to $-x_i(t)$ if and only if $\Delta E > 0$. It is now clear that the proposed rule is identical to the asynchronous updating rule. □

Now we come to the main result of this section.

Theorem 1. *Suppose E is a multilinear polynomial on $\mathfrak{R}^n$. Suppose $\mathbf{x} \in \{-1, 1\}^n$, and that no component of $\nabla E(\mathbf{x})$ is zero. Under these conditions, the following statements are equivalent:*

1. $\mathbf{x}$ *is a strict local maximum of* E.
2. $\mathbf{x}$ *is a fixed point under asynchronous updating.*
3. $\mathbf{x}$ *is a fixed point under best neighbour updating.*
4. $\mathbf{x}$ *is a fixed point under gradient updating.*

Proof. Suppose $\mathbf{y} \in N(\mathbf{x})$ is obtained by replacing x_i by $-x_i$. Let i be an arbitrary index from the set $\{1, \ldots, n\}$. Define $\bar{x}_i$, $\bar{E}_i$ and c_i as in the proof of lemma 3. Then, from (27),

$$E(\mathbf{y}) - E(\mathbf{x}) = -2x_i\bar{E}_i(\bar{\mathbf{x}}_i) = -2x_i[\nabla E(\mathbf{x})]_i.$$

Now $\mathbf{x}$ is a strict local maximum of E is and only if $E(\mathbf{y}) - E(\mathbf{x}) < 0$ for all $\mathbf{y} \in N(\mathbf{x})$, or equivalently

$$x_i = \text{sign}[\nabla E(\mathbf{x})]_i, \quad i = 1, \ldots, n. \tag{29}$$

Clearly, (29) is also a necessary and sufficient condition for $\mathbf{x}$ to be a fixed point under each of the three types of updating. □

At this point the reader may wonder why a distinction is made between asynchronous updating and best neighbour updating. The reason is brought out next.

Suppose $\mathbf{x}_0$ is a fixed point under some updating rule. Then the *domain of direct attraction* of $\mathbf{x}_0$, denoted by DDA $(\mathbf{x}_0)$, is defined as

$$\text{DDA}(\mathbf{x}_0) = \{\mathbf{x} \in \{-1, 1\}^n : \mathbf{x}(0) = \mathbf{x} \Rightarrow \mathbf{x}(1) = \mathbf{x}_0\}. \tag{30}$$

In other words, DDA$(\mathbf{x}_0)$ is the set of states that get mapped into $\mathbf{x}_0$ *in a single time step.* The *radius of direct attraction* of $\mathbf{x}_0$, denoted by RDA $(\mathbf{x}_0)$, is defined as the largest number r such that

$$H(\mathbf{x}, \mathbf{x}_0) \leqslant r \Rightarrow \mathbf{x} \in \text{DDA}(\mathbf{x}_0), \tag{31}$$

where $H(\mathbf{x}, \mathbf{x}_0)$ is the Hamming distance between $\mathbf{x}$ and $\mathbf{x}_0$, that is, the number of components in which $\mathbf{x}$ and $\mathbf{x}_0$ differ. In other words, the radius of direct attraction is the radius of the largest "sphere" centered at $\mathbf{x}_0$ contained in the set DDA$(\mathbf{x}_0)$.

Theorem 2. *Suppose $\mathbf{x}_0$ is a global maximum of the function E; that is, $E(\mathbf{x}_0) \geqslant E(\mathbf{y})$*

for all $\mathbf{y} \in \{-1, 1\}^n$. *Suppose also that all other global maxima of* E *are at a distance of at least three from* $\mathbf{x}_0$. *Then*

1. $\mathbf{x}_0$ *is a fixed point under best neighbour updating.*
2. RDA $(\mathbf{x}_0) \geqslant 1$.

Proof. The proof is almost obvious. The hypotheses on $\mathbf{x}_0$ imply that $E(\mathbf{x}_0) > E(\mathbf{x})$ for all $\mathbf{x} \in N(\mathbf{x}_0)$. Hence $\mathbf{x}_0$ is a fixed point under best neighbour updating. Next, suppose $\mathbf{x} \in N(\mathbf{x}_0)$, and that $\mathbf{y} \in N(\mathbf{x})$. Then $H(\mathbf{y}, \mathbf{x}_0) \leqslant 2$. By assumption, this implies that $E(\mathbf{x}_0) > E(y)$ (unless $\mathbf{y} = \mathbf{x}_0$), because there is no global maximum of E within a Hamming distance of 2 from $\mathbf{x}_0$. Hence $\mathbf{x}_0$ is the best neighbour of $\mathbf{x}$, which means that if $\mathbf{x}(0) = \mathbf{x}$, then $\mathbf{x}(1) = \mathbf{x}_0$ under best neighbour updating. □

The only reason for mentioning this obvious result is that it is *not* true under asynchronous updating.

Example 2. As an illustration, consider the block-coding problem studied in Bruck & Blaum (1989). For the (7, 4) Hamming code, the integer n equals 7, and the objective function is

$$E(\mathbf{x}) = x_1 x_2 x_4 x_5 + x_1 x_3 x_4 x_6 + x_1 x_2 x_3 x_7. \tag{32}$$

It is easy to see that $E(\mathbf{x}) \leqslant 3$ for all $\mathbf{x} \in \{-1, 1\}^7$. Moreover, $E(\mathbf{x}) = 3$ if and only if

$$x_1 x_2 x_4 x_5 = x_1 x_3 x_4 x_6 = x_1 x_2 x_3 x_7 = 1. \tag{33}$$

There are exactly $2^4 = 16$ vectors that satisfy (33), and these are the global maxima of E. Moreover, the minimum Hamming distance between any pair of global maxima is 3. For details, see Bruck & Blaum (1989). Thus, by theorem 2, each of these global maxima has a radius of direct attraction of at least 1, if best neighbour updating is used.

To show that this is not true if asynchronous updating is used, let $\mathbf{x}_0$ equal the vector of all 1's. It is clear that $\mathbf{x}_0$ is a global maximum of E, because (33) is satisfied. Now consider the sequence

$$\mathbf{x}(0) = [-1\,1\,1\,1\,1\,1\,1], \tag{34}$$

$$\mathbf{x}(1) = [-1\,1\,1\,1\,-1\,1\,1], \tag{35}$$

$$\mathbf{x}(2) = [-1\,1\,1\,1\,-1\,-1\,1], \tag{36}$$

$$\mathbf{x}(3) = [-1\,1\,1\,1\,-1\,-1\,-1]. \tag{37}$$

Then

$$E(0) = -3, \quad E(1) = -1, \quad E(2) = 1, \quad E(3) = 3. \tag{38}$$

This is a valid sequence under asynchronous updating, because the objective function is strictly increased at each time step. Moreover, $\mathbf{x}(3)$ is a global maximum of E, because $E(3) = 3$. Hence $\mathbf{x}(3)$ is a fixed point under asynchronous updating. Note that $\mathbf{x}(1) \in N(\mathbf{x}_0)$. Therefore, starting at a neighbour of $\mathbf{x}_0$, we have constructed a sequence of transitions that converge to *another* fixed point. This shows that, under asynchronous updating, the radius of direct attraction is *zero*.

It can be shown that, with the objective function suggested in Bruck & Blaum (1989), the same phenomenon occurs with *every* choice of the coding matrix and *every* equilibrium. □

In view of theorem 1, the next result, which discusses the convergence of the neural network under asynchronous and best neighbour updating, is almost obvious.

Theorem 3. *Suppose the network is operated under the asynchronous updating rule or the best neighbour updating rule. Then the network converges to a local maximum in a finite number of time steps.*

3. Modified synchronous updating

Theorem 3 shows that, under both the asynchronous as well as the best-neighbour updating rules, the trajectory of the neural network (9) converges to a local maximum of the objective function within a finite number of time steps. But what happens if the network is updated according to the gradient rule? This corresponds to *synchronous* updating of all neurons according to (9). In the case of a Hopfield network of the form (1), it is known that the network trajectory either converges to a local maximum or else goes into a limit cycle of length two (see Goles *et al* (1985)). But in the case where the objective function is not necessarily quadratic, the behaviour of the network under gradient updating is not well-understood. In this section, it is shown that, by *modifying* the gradient updating formula, it is possible to say something about the convergence of the network.

The traditional Hopfield network described by (1) is characterized by *linear* interconnections; that is, the right side of (1) is an affine function of the neural state vector $\mathbf{x}(t)$. In case where a neural network is to be used to optimize a polynomial of degree three or more, some authors propose the use of higher-order interconnections among neurons. The gradient updating rule (9) is an example of higher-order interconnections, because if E is a polynomial of degree three or more, then each component of $\nabla E(\mathbf{x})$ is a polynomial of degree two or more. Higher-order interconnections can be said to reflect *spatial* correlation between neural states. However, as mentioned in the preceding paragraph, the behaviour of a neural network under gradient updating is not well-understood. In this section, we propose a modification of gradient updating for higher-order neural networks. In the modified formula, the state of the network at time $t+1$ depends on the state of the network at times $t, t-1, \ldots, t-(m-2)$, where m is the degree of the objective function. If the objective function is quadratic, then $m=2$ and $m-2=0$. In this case, the state of the network at time $t+1$ depends only on its state at time t, and the updating formula reduces to (1). For higher-order objective functions, however, the updating formula makes use of the past states as well. Moreover, the interaction terms are products of up to $m-1$ terms. Hence the updating formula makes use of *temporal as well as spatial* correlation between neurons. With this updating formula, it is shown that the network trajectory either converges to an equilibrium, or else goes into a limit cycle of length m, where m is the degree of the objective function. Hence theorem 5 below is a "true" generalization of the results in Goles *et al* (1985).

As a prelude to presenting the updating formula, we give a brief discussion on symmetric multilinear forms. Suppose we are given an objective function E, which is a multilinear polynomial of degree m. Then we can write

$$E(\mathbf{x}) = \sum_{k=1}^{m} E_k(\mathbf{x}, \ldots, \mathbf{x}), \tag{39}$$

where E_k is a homogeneous, symmetric, multilinear polynomial. Homogeneity means simply that E_k comprises all terms in E that are a product of *exactly* k terms of the form $x_{i_1}, \ldots, x_{i_k}$. Multilinearity in the present context means that $E_k(\mathbf{x}_1, \ldots, \mathbf{x}_k)$ is a linear function of $\mathbf{x}_i$ for each $i \in \{1, \ldots, k\}$. Symmetry means that

$$E_k(\mathbf{x}_1, \ldots, \mathbf{x}_k) = E_k(\mathbf{x}_{\pi(1)}, \ldots, \mathbf{x}_{\pi(k)}), \tag{40}$$

for all permutations π of $\{1, \ldots, k\}$ into itself. For instance, if $k = 2$, then

$$E_2(\mathbf{x}, \mathbf{y}) = \mathbf{x}^t M \mathbf{y} \tag{41}$$

for some $n \times n$ matrix M. Now E_2 is symmetric if and only if the matrix M is symmetric.

If E_k is a multilinear homogeneous polynomial of degree k, it is easy to see that there exists a function $\mathbf{e}_k : \mathfrak{R}^{n(k-1)} \to \mathfrak{R}^n$ such that

$$E_k(\mathbf{x}_1, \ldots, \mathbf{x}_k) = \mathbf{x}_1^t \mathbf{e}_k(\mathbf{x}_2, \ldots, \mathbf{x}_k), \tag{42}$$

where each component of the vector $\mathbf{e}_k(\mathbf{x}_2, \ldots, \mathbf{x}_k)$ is a homogeneous, multilinear polynomial of degree $k-1$. Moreover, if E_k is symmetric, so is $\mathbf{e}_k$, in the sense that

$$\mathbf{e}_k(\mathbf{x}_2, \ldots, \mathbf{x}_k) = \mathbf{e}_k(\mathbf{x}_{\pi(2)}, \ldots, \mathbf{x}_{\pi(k)}), \tag{43}$$

for all permutation π of $\{2, \ldots, k\}$ into itself.

For every homogeneous multilinear polynomial E_k of degree k, there is an equivalent homogeneous *symmetric* polynomial E_{sk} such that

$$E_{sk}(\mathbf{x}, \ldots, \mathbf{x}) = E_k(\mathbf{x}, \ldots, \mathbf{x}). \tag{44}$$

In fact, we can define

$$E_{sk}(\mathbf{x}_1, \ldots, \mathbf{x}_k) = \frac{1}{k!} \sum_{\pi} E_k(\mathbf{x}_{\pi(1)}, \ldots, \mathbf{x}_{\pi(k)}), \tag{45}$$

where the summation is over all permutations π of $\{1, \ldots, k\}$ into itself. For instance, if $k = 2$ and

$$E_2(\mathbf{x}, \mathbf{y}) = \mathbf{x}^t M \mathbf{y}, \tag{46}$$

then

$$E_{s2}(\mathbf{x}, \mathbf{y}) = \frac{1}{2}[E(\mathbf{x}, \mathbf{y}) + E(\mathbf{y}, \mathbf{x})] = \frac{1}{2}\mathbf{x}^t[M + M^t]\mathbf{y}, \tag{47}$$

which is the familiar method of obtaining a *symmetric* quadratic form. The procedure in the case of higher values of k is illustrated through an example.

Example 1. (Continued) Consider once again the objective function in connection with the (7, 4) Hamming code, namely

$$E(\mathbf{x}) = x_1 x_2 x_4 x_5 + x_1 x_3 x_4 x_6 + x_1 x_2 x_3 x_7. \tag{48}$$

In this case, $E(\mathbf{x})$ is a homogeneous polynomial of degree 4. If we define

$$E_4(\mathbf{x}, y, \mathbf{z}, \mathbf{v}) = x_1 y_2 z_4 v_5 + x_1 y_3 z_4 v_6 + x_1 y_2 z_3 v_7, \tag{49}$$

then E_4 is not symmetric. It can be replaced by an equivalent symmetric polynomial as follows: Because of the complexity, the details are given only for the first term, namely $x_1 y_2 z_4 v_5$. There are $4! = 24$ ways of assigning subscripts 1, 2, 4, 5 to the symbols **x**, **y**, **z**, **v**. Write down all 24 possible combinations, add them up, and divide by 24. Thus

$$E_{s4}(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{v}) = (1/24)[x_1 y_2 z_4 v_5 + x_2 y_1 z_4 v_5 + \cdots x_5 y_4 z_2 v_1]. \tag{50}$$

Symmetric versions of the other terms can be constructed in an entirely analogous manner.

Now we are ready to state the updating rule. Suppose the objective function $E(\mathbf{x})$ is a multilinear polynomial of degree m in $x_1, \ldots, x_n$. Then, without loss of generality, we can write

$$E(\mathbf{x}) = \sum_{k=1}^{m} E_k(\mathbf{x}, \ldots, \mathbf{x}), \tag{51}$$

where E_k is a homogeneous, symmetric, multilinear polynomial.[2] Define $\mathbf{e}_k$ as before, namely by

$$E_k(\mathbf{x}_1, \ldots, \mathbf{x}_k) = \mathbf{x}_1^t \mathbf{e}_k(\mathbf{x}_2, \ldots, \mathbf{x}_k), \tag{52}$$

where each component of $\mathbf{e}_k$ is a homogeneous, symmetric, multilinear polynomial of degree $k-1$. Note that

$$\nabla E_k(\mathbf{x}) = k\mathbf{e}_k(\mathbf{x}, \ldots, \mathbf{x}). \tag{53}$$

The idea behind the modified updating rule is rather simple, though the notation can become somewhat cumbersome. To state it, it is helpful to introduce the symbol $S(k, m)$. Given integers $m > k > 0$, let $S(k, m)$ denote the set of all k-tuples $(i_1, \ldots, i_k)$ such that

$$0 \leqslant i_1 < i_2 < \cdots < i_k \leqslant m-1. \tag{54}$$

Note that this is a minor modification of the symbol $S(m, n)$ in Vidyasagar (1985, p. 391). For example, $S(3, 5)$ equals

$$S(3,5) = \{(0,1,2), (0,1,3), (0,1,4), (0,2,3), (0,2,4), (0,3,4), (1,2,3), (1,2,4), (1,3,4), (2,3,4)\}. \tag{55}$$

It is easy to see that the number of elements in $S(k, m)$ equals $m!/k!(m-k)!$.

Given an objective function of the form (51), we first define the constands α_k, $k = 1, \ldots, m$, by

$$\alpha_k = [(m-k)!k!]/(m-1)!. \tag{56}$$

Note that $\alpha_k = m/|S(k, m)|$, where $|S(k, m)|$ is the cardinality of the set $S(k, m)$, i.e., the number of elements in the set $S(k, m)$. For each k in $\{1, \ldots, m\}$, define the function

[2]Note that we use E_k instead of E_{sk}, which would be consistent with earlier notation.

$\mathbf{f}_k(t)$ as follows: Let π vary over all elements of the set $S(k-1, m-1)$. For each π, define

$$\mathbf{f}_{k,\pi}(t) = \mathbf{e}_k[\mathbf{x}(t-\pi_1), \ldots, \mathbf{x}(t-\pi_{k-1})], \tag{57}$$

and define

$$\mathbf{f}_k(t) = \sum_{\pi \in S(k-1, m-1)} \mathbf{f}_{k,\pi}(t). \tag{58}$$

Now the updating rule is as follows:

$$\mathbf{x}(t+1) = \operatorname{sign}\left[\sum_{k=1}^{m} \alpha_k \mathbf{f}_k(t)\right]. \tag{59}$$

Example 2. Suppose E is of degree 4. Express $E(\mathbf{x})$ as

$$E(\mathbf{x}) = E_4(\mathbf{x}, \mathbf{x}, \mathbf{x}, \mathbf{x}) + E_3(\mathbf{x}, \mathbf{x}, \mathbf{x}) + E_2(\mathbf{x}, \mathbf{x}) + E_1(\mathbf{x}) + E_0. \tag{60}$$

Define the vectors $\mathbf{e}_4, \mathbf{e}_3, \mathbf{e}_2, \mathbf{e}_1$ in accordance with (3.4). The sets $S(k-1, 3)$ are given as

$$\begin{aligned} &S(3,3) = \{(0,1,2)\}, \\ &S(2,3) = \{(0,1),(0,2),(1,2)\}, \\ &S(1,3) = \{0,1,2\}. \end{aligned} \tag{61}$$

The set $S(0, 3)$ is undefined (see below). The vectors $\mathbf{f}_4, \mathbf{f}_3, \mathbf{f}_2, \mathbf{f}_1$ are given respectively by

$$\begin{aligned} \mathbf{f}_4 &= \mathbf{e}_4[\mathbf{x}(t), \mathbf{x}(t-1), \mathbf{x}(t-2)], \\ \mathbf{f}_3 &= \mathbf{e}_3[\mathbf{x}(t), \mathbf{x}(t-1)] + \mathbf{e}_3[\mathbf{x}(t), \mathbf{x}(t-2)] + \mathbf{e}_3[\mathbf{x}(t-1), \mathbf{x}(t-2)], \\ \mathbf{f}_2 &= \mathbf{e}_2[\mathbf{x}(t)] + \mathbf{e}_2[\mathbf{x}(t-1)] + \mathbf{e}_2[\mathbf{x}(t-2)], \\ \mathbf{f}_1 &= \mathbf{e}_1, \end{aligned} \tag{62}$$

where we take advantage of the fact that, because $E_1(\mathbf{x})$ is a linear functional of $\mathbf{x}$, the gradient $\mathbf{e}_1$ is independent of $\mathbf{x}$. The constants α_i are given from (56) as

$$\alpha_4 = 4, \quad \alpha_3 = 1, \quad \alpha_2 = 2/3, \quad \alpha_1 = 1. \tag{63}$$

Finally, the updating rule is given by (59).

It is left to the reader to verify that, if $m = 2$ so that the objective function is quadratic and has the form

$$E(\mathbf{x}) = \tfrac{1}{2}\mathbf{x}^t \mathbf{W}\mathbf{x} + \mathbf{x}^t \theta, \tag{64}$$

then the updating rule becomes

$$\mathbf{x}(t+1) = \operatorname{sign}[\mathbf{W}\mathbf{x}(t) + \theta], \tag{65}$$

which is the familiar rule. Hence (59) is a "true" generalization of the Hopfield network.

What is the advantage gained by using the complicated updating rule (59) in place of the simpler gradient rule

$$\mathbf{x}(t+1) = \operatorname{sign}\{\nabla E[\mathbf{x}(t)]\}? \tag{66}$$

The principal advantage is that the dynamics of the system (59) can be analysed precisely. This is the main result of this section, and indeed, of the paper. Before stating this result, a preliminary issue is resolved.

DEFINITION 3.

A vector $\mathbf{x}_0 \in \{-1, 1\}^n$ is said to be a *fixed point* of the system (59) if

$$\mathbf{x}(0) = \mathbf{x}(1) = \cdots = \mathbf{x}(m-2) = \mathbf{x}_0 \Rightarrow \mathbf{x}(t) = \mathbf{x}_0 \text{ for all } t \geqslant m-1. \tag{67}$$

This is just the standard definition of a fixed point, adapted for the fact that the right side of (59) contains some delayed terms.

Theorem 4. *A vector $\mathbf{x}_0 \in \{-1, 1\}^n$ is a fixed point of* (59) *if and only if it is a fixed point of the gradient updating rule, that is, if and only if*

$$\mathbf{x}_0 = \text{sign}[\nabla E(\mathbf{x}_0)]. \tag{68}$$

Theorem 4 means that, even though there are time delays present in the updating rule (59), the fixed points are the same as those of the gradient updating rule. Combined with theorem 1, this result means that, should the state vector of the network converge to a fixed point, this fixed point will be a local maximum of the objective function E, even though time delays are introduced into the updating rule.

Proof. Suppose $\mathbf{x}_0$ satisfies (67), and suppose $\mathbf{x}(0) = \mathbf{x}(1) = \cdots = \mathbf{x}(m-2) = \mathbf{x}_0$. Then it follows from (57) that

$$\mathbf{f}_{k,\pi} = \mathbf{e}_k(\mathbf{x}_0, \ldots, \mathbf{x}_0). \tag{69}$$

Therefore, from (58),

$$\mathbf{f}_k = \frac{(m-1)!}{(k-1)!(m-k)!} \mathbf{e}_k(\mathbf{x}_0, \ldots, \mathbf{x}_0), \tag{70}$$

where the coefficient on the right side is the number of elements of the set $S(k-1, m-1)$. Now combining (56) and (70) shows that

$$\alpha_k \mathbf{f}_k = k\mathbf{e}_k(\mathbf{x}_0, \ldots, \mathbf{x}_0) = \nabla E_k(\mathbf{x}_0) \tag{71}$$

Finally,

$$\sum_{k=1}^{m} \alpha_k F_k = \sum_{k=1}^{m} \nabla E_k(\mathbf{x}_0) = \nabla E(\mathbf{x}_0). \tag{72}$$

Therefore (67) is satisfied if and only if

$$\mathbf{x}_0 = \text{sign}[\nabla E(\mathbf{x}_0)], \tag{73}$$

that is, if and only if $\mathbf{x}_0$ is a fixed point under the gradient updating rule. □

Now we come to the main result.

Theorem 5. *With the updating rule* (59), *there exists a finite integer N such that*

$$\mathbf{x}(t) = \mathbf{x}(t-m), \quad \text{for all } t \geqslant N. \tag{74}$$

Remark. Theorem 5 states that, with the modified updating rule, each trajectory of the neural network eventually satisfies (74). This means that the possible behaviours of the network are: (i) it settles into a fixed point, or (ii) it goes into a limit cycle whose length is *divisor* of m. For example, if the objective function has degree 6, the network trajectories either converge to a fixed point, or else go into a limit cycle whose length is 2, 3, or 6.

Proof. Define a function $G(t)$ as follows:

$$G(t) = \sum_{k=1}^{m} \alpha_k G_k(t), \tag{75}$$

where α_k is defined in (56), and $G_k(t)$ is defined as follows: Let π vary over $S(k,m)$, and define

$$G_{k,\pi}(t) = E_k[\mathbf{x}(t-\pi_1),\ldots,\mathbf{x}(t-\pi_k)], \tag{76}$$

$$G_k(t) = \sum_{\pi \in S(k,m)} G_{k,\pi}(t). \tag{77}$$

Now let us compute the quantity

$$\Delta G(t) = G(t+1) - G(t). \tag{78}$$

It is claimed that

$$\Delta G(t) = [\mathbf{x}(t+1) - \mathbf{x}(t-m+1)]^t \left[\sum_{k=1}^{m} \alpha_k \mathbf{f}_k(t) \right], \tag{79}$$

where $\mathbf{f}_k(t)$ is defined in (58). Suppose for the moment that (79) is true. Then, in view of (59), it follows that if $\mathbf{x}(t+1) \neq \mathbf{x}(t-m+1)$, then $\Delta G(t) > 0$. Moreover, the increase $\Delta G(t)$ can be bounded from below. In fact, $\Delta G(t)$ is at least equal to the magnitude of the smallest component of $\Sigma_k \alpha_k \mathbf{f}_k(t)$. Since $\mathbf{x}(t)$ varies over a finite set, this quantity itself can be bounded from below, say by ε. Similarly, $G(t)$ can be bounded from above, say by M. Thus it follows that $\mathbf{x}(t+1) \neq \mathbf{x}(t-m+1)$ cannot happen more than $[M/\varepsilon]$ times, that is, a finite number of times.

Thus the proof is complete if it can be shown that (79) holds. What we show instead is that

$$\Delta G_k(t) = G_k(t+1) - G_k(t) = [\mathbf{x}(t+1) - \mathbf{x}(t-m+1)]^t \mathbf{f}_k(t). \tag{80}$$

Combined with (75), this is enough to establish (79).

To prove (80), we proceed as follows: Note that

$$\Delta G_k(t) = \sum_{\theta \in S(k,m)} E_k[\mathbf{x}(t-\theta_1+1),\ldots,\mathbf{x}(t-\theta_k+1)] - \sum_{\pi \in S(k,m)} E_k[\mathbf{x}(t-\pi_1),\ldots,\mathbf{x}(t-\pi_k)]. \tag{81}$$

If $\theta_1 \neq 0$, then there exists a π in $S(k,m)$ such that

$$(\theta_1 - 1,\ldots,\theta_k - 1) = (\pi_1,\ldots,\pi_k), \tag{82}$$

so that corresponding terms in the two summations cancel out. Similarly, if $\pi_k \neq m$, then there exists a θ in $S(k,m)$ such that (82) holds. So once again the corresponding terms in the two summations cancel out. Thus we are left with

$$\begin{aligned}\Delta G_k(t) = &\sum_{\theta_1=0} E_k[\mathbf{x}(t-\theta_1+1), \mathbf{x}(t-\theta_2+1), \ldots, \mathbf{x}(t-\theta_k+1)] \\ &- \sum_{\pi_k=m} E_k[\mathbf{x}(t-\pi_1), \ldots, \mathbf{x}(t-\pi_{k-1}), \mathbf{x}(t-\pi_k)] \\ = &[\mathbf{x}(t+1)]^t \sum_{(\theta_2-1,\ldots,\theta_k-1)\in S(k-1,m-1)} \mathbf{e}_k[\mathbf{x}(t-\theta_2+1), \ldots, \mathbf{x}(t-\theta_k+1)] \\ &- [\mathbf{x}(t-m+1)]^t \sum_{(\pi_1,\ldots,\pi_{k-1})\in S(k-1,m-1)} \mathbf{e}_k[\mathbf{x}(t-\pi_1), \ldots, \mathbf{x}(t-\pi_{k-1})].\end{aligned} \tag{83}$$

But both summations are the same, and equal $\mathbf{f}_k(t)$ (see (58)). Hence we get

$$\Delta G_k(t) = [\mathbf{x}(t+1) - \mathbf{x}(t-m+1)]^t \mathbf{f}_k(t), \tag{84}$$

which is precisely (80). This completes the proof.

4. Conclusions

In this paper, we have formulated the problem of maximizing a general objective function over the hypercube $\{-1, 1\}^n$ as that of maximizing a *multilinear polynomial* over $\{-1, 1\}^n$. Three modes of operation have been considered: asynchronous updating, gradient updating, and a new mode known as best neighbour updating. In the case of a quadratic objective function, the best neighbour updating does not appear to have been considered previously. We have shown that the asynchronous and the best neighbour updating rules converge to a local maximum of the objective function in a finite number of time steps. In the gradient mode, the behaviour of the network is not well-understood. For this purpose, we have *modified* the gradient updating rule in such a way that it incorporates both *temporal as well as spatial* correlations among the neurons. For the modified updating rule, we have shown that after a finite number of time steps, the network state vector goes into a limit cycle of length m, where m is the degree of the objective function. This does not preclude the possibility that the trajectory converges to a fixed point (which is a degenerate form of a limit cycle). If so, any such fixed point is a local maximum of the objective function.

While the modified updating formula presented here has the advantage that its dynamics are well-understood under gradient updating, it is natural to ask whether this advantage is enough to offset the added complexity of computing and implementing the modified updating rule. There can be no clear-cut answer to this question, as it is a matter of one's taste.

References

Hopfield J J 1982 Neural networks and physical systems with emergent collective computational capabilities. *Proc. Natl. Acad. Sci. USA* 79: 2554–2558

Goles E, Fogelman E, Pellegrin 1985 Decreasing energy functions as a tool for studying threshold networks. *Discuss. Appl. Math.* 12: 261–277

Bruch J, Goodman J W 1988 A generalized convergence theorem for neural networks. *IEEE Trans. Info. Theor* 34: 1089–1092

Kamp Y, Hasler M 1990 *Recursive neural networks for associative memory* (Chichester: John Wiley)

Hopfield J J, Tank D W 1985 'Neural' computation of decision optimization problems. *Biol. Cybern.* 52: 141–152

Masti C L, Vidyasagar M 1991 A stochastic high-order connectionist network for solving inferencing problems. *Proc. Int. Joint. Conf. Neural Networks* Singapore, pp. 911–916

Bruck J, Blum M 1989 'Neural networks, error-correcting codes, and polynomials over the binary *n*-cube. *IEEE Trans. Info. Theor.* 35: 976–987

van Laarhoven P, Aarts E 1987 *Simulated annenling: Theory and applications* (Dordrecht: Reidel)

Vidyasagar M 1985 *Control system synthesis: A factorization approach* (Cambridge, MA: MIT Press)