ELSEVIER

Discrete Applied Mathematics 89 (1998) 169–180

DISCRETE APPLIED MATHEMATICS

Linking discrete orthogonality with dilation and translation for incomplete sigma-pi neural networks of Hopfield-type

Burkhard Lenze *

Fachhochschule Dortmund, Fachbereich Informatik, Postfach 105018, D-44047 Dortmund, Germany

Received 20 February 1997; revised in revised form 30 June 1997; accepted 4 August 1997

Abstract

In this paper, we show how to extend well-known discrete orthogonality results for complete sigma-pi neural networks on bipolar coded information in presence of dilation and translation of the signals. The approach leads to a whole family of functions being able to implement any given Boolean function. Unfortunately, the complexity of such complete higher order neural network realizations increases exponentially with the dimension of the signal space. Therefore, in practise one often only considers incomplete situations accepting that not all but hopefully the most relevant information or Boolean functions can be realized. At this point, the introduced dilation and translation parameters play an essential rôle because they can be tuned appropriately in order to fit the concrete representation problem as best as possible without any significant increase of complexity. In detail, we explain our approach in context of Hopfield-type neural networks including the presentation of a new learning algorithm for such generalized networks.

Keywords: Discrete orthogonality; Dilation; Translation; Higher order networks; Sigma-pi networks; Hopfield networks; Walsh functions

1. Introduction

One of the most classical problems in discrete information processing is to efficiently implement a given Boolean function $p:\{-1,1\}^n \to \{-1,1\}^m$. Since p can be written as a function vector $p=(p_1,p_2,\ldots,p_m)$ consisting of functions $p_j:\{-1,1\}^n \to \{-1,1\}$, $1 \leqslant j \leqslant m$, it is sufficient to consider the case $m=1$. A standard situation of the above kind is the following: assume that a number of bipolar coded $(n,1)$-tuples are given,

$$(\boldsymbol{x}^{(s)}, y^{(s)}) \in \{-1,1\}^n \times \{-1,1\}, \quad 1 \leqslant s \leqslant t, \tag{1.1}$$

* E-mail: lenze@fh-dortmund.de.

PII: S0166-218X(98)00102-4

where we require that the vectors $x^{(s)}$, $1 \leqslant s \leqslant t$, are pairwise distinct. The problem is to construct a function $p: \{-1,1\}^n \to \{-1,1\}$ satisfying

$$p(x^{(s)}) = y^{(s)}, \quad 1 \leqslant s \leqslant t. \tag{1.2}$$

As it is well known, a possible solution of this problem connected with the keyword Walsh function can be obtained as follows. Define the so-called 2^n Walsh coefficients or sigma-pi coefficients by

$$w_R := 2^{-n} \sum_{s=1}^{t} y^{(s)} \prod_{k \in R} x_k^{(s)}, \quad R \subset \{1,2,\ldots,n\}, \tag{1.3}$$

(with the usual convention to identify a product over the empty set with 1) and $p: \{-1,1\}^n \to \{-1,0,1\}$ by

$$p(x) := \sum_{R \subset \{1,\ldots,n\}} w_R \prod_{k \in R} x_k. \tag{1.4}$$

Then, it is easily checked that p satisfies the interpolation conditions (1.2). The construction of p is based on the so-called sigma-pi orthogonality of bipolar coded vectors, namely,

$$\sum_{R \subset \{1,\ldots,n\}} \prod_{k \in R} x_k^{(1)} x_k^{(2)} = \begin{cases} 0, & x^{(1)} \neq x^{(2)}, \\ 2^n, & x^{(1)} = x^{(2)}, \end{cases} \tag{1.5}$$

or – in other words – the discrete orthogonality of the Walsh functions (see [3, 5, 7]) for the general theory and [1, 2, 8] for special applications in neural network context). The problem in view of solution (1.4) is that in general one has to deal with 2^n terms in the sum defining p, a complexity which is unacceptable in practice. Therefore, a number of modifications and simplifications are discussed in literature, especially in connection with neural network and associative memory design. The first essential simplification is to restrict the sum appearing in (1.4) to run only over a "small" set Γ of subsets of $\{1,\ldots,n\}$ instead of summing over all subsets of $\{1,\ldots,n\}$,

$$p(x) := \sum_{R \in \Gamma} w_R \prod_{k \in R} x_k. \tag{1.6}$$

Of course, any such restriction of the sum will probably violate some or even all interpolation conditions (1.2). To compensate this simplification, at least in part, a second modification is usually introduced. Using the so-called signum-type transfer function $T_S: \mathbb{R} \to \{-1,1\}$,

$$T_S(\xi) := \begin{cases} -1, & \xi < 0, \\ 1, & \xi \geqslant 0, \end{cases} \tag{1.7}$$

definition (1.6) is changed to

$$p(x) := T_S\left(\sum_{R \in \Gamma} w_R \prod_{k \in R} x_k \right). \tag{1.8}$$

This kind of ultimate definition can be justified because p should always yield 1 or -1 which is obviously no longer guaranteed when simply cutting the sum as in (1.6). The new definition (1.8), however, makes sure that $p(\boldsymbol{x})$ is either 1 or -1 and reduces the task of (1.6) to indicate the right sign, only. The drawback of this approach is that in many concrete applications the number of remaining weights w_R, $R \in \Gamma$, to be tuned in order to satisfy all conditions (1.2) are not sufficient; besides the problem to figure out which set Γ should be used (see [2] for a discussion of such pruning strategies) the chosen set of sigma-pi weights is often too restricted to settle all specific tasks even in case that they are closely related. Precisely at this point our ideas enter the field. We will use dilation and translation parameters as already introduced in [6] in order to increase the flexibility of our Boolean representation functions without losing the nice property of discrete orthogonality, the basis of the classical approach. In the next section we will explain this dilation and translation idea in detail and give the theoretical justification. In Section 3 we will apply our approach in the special case of incomplete sigma-pi Hopfield neural networks, we will propose a new learning algorithm based on dilation and translation, and show and discuss the results of a concrete example. In the final section we briefly summarize our main conclusions.

2. Linking discrete orthogonality with dilation and translation

In the following, we take a look at a generalization of (1.8) by introducing dilation and translation. In detail, we consider functions of type $p: \{-1,1\}^n \to \{-1,1\}$,

$$p(\boldsymbol{x}) := T_S\left(\sum_{R\in\Gamma} w_R \prod_{k\in R} (\alpha_k x_k - d_k)\right), \tag{2.1}$$

with Γ a set of subsets of $\{1,\ldots,n\}$, $w_R \in \mathbb{R}$, $R \in \Gamma$, the so-called sigma-pi weights and $\alpha_k \in \mathbb{R}$, $\alpha_k > 0$, and $d_k \in \mathbb{R}$, $1 \leqslant k \leqslant n$, some dilation and translation parameters. The general question we should keep in mind is how to choose the weights and parameters in order to make p satisfy certain interpolation conditions like (1.2) under the constraint of a reasonable small set Γ to control complexity. For theoretical reasons we start with a result dealing with the so-called complete case where Γ coincides with the set of all subsets of $\{1,\ldots,n\}$. First of all, we have the following orthogonality result.

Theorem 2.1. *Let $n \in \mathbb{N}$, $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)} \in \{-1,1\}^n$ and $d_k \in \mathbb{R}$, $1 \leqslant k \leqslant n$, be given arbitrarily. Then, for $\alpha_k \in \mathbb{R}$, $1 \leqslant k \leqslant n$, defined as*

$$\alpha_k := \sqrt{1 + d_k^2}, \quad 1 \leqslant k \leqslant n, \tag{2.2}$$

the following discrete orthogonality result holds:

$$\sum_{R\subset\{1,\ldots,n\}} \prod_{k\in R} (\alpha_k x_k^{(1)} - d_k)(\alpha_k x_k^{(2)} - d_k) \begin{cases} = 0, & \boldsymbol{x}^{(1)} \neq \boldsymbol{x}^{(2)}, \\ > 1, & \boldsymbol{x}^{(1)} = \boldsymbol{x}^{(2)}. \end{cases} \tag{2.3}$$

Proof. Let $n \in \mathbb{N}$ and $d_k \in \mathbb{R}$, $1 \leqslant k \leqslant n$, be given arbitrarily and $\alpha_k \in \mathbb{R}$, $1 \leqslant k \leqslant n$, be defined by (2.2). Moreover, we use the fact that for arbitrary $\beta_k \in \mathbb{R}$, $1 \leqslant k \leqslant n$, the identity

$$\sum_{R \subset \{1,\dots,n\}} \prod_{k \in R} \beta_k = \prod_{k=1}^{n} (1 + \beta_k) \tag{2.4}$$

holds which may be easily proved by induction. Now, for $x^{(1)}, x^{(2)} \in \{-1, 1\}^n$ given arbitrarily the above identity implies

$$\begin{aligned} &\sum_{R \subset \{1,\dots,n\}} \prod_{k \in R} (\alpha_k x_k^{(1)} - d_k)(\alpha_k x_k^{(2)} - d_k) \\ &= \prod_{k=1}^{n} (1 + \alpha_k^2 x_k^{(1)} x_k^{(2)} - \alpha_k d_k x_k^{(1)} - \alpha_k d_k x_k^{(2)} + d_k^2). \end{aligned} \tag{2.5}$$

Case 1: $(x^{(1)} \neq x^{(2)})$. In this case there exists at least one index $k \in \{1, \dots, n\}$ with $x_k^{(1)} = -x_k^{(2)}$. Therefore, the kth factor of the right-hand side of (2.5) reduces to

$$1 + \alpha_k^2 x_k^{(1)} x_k^{(2)} - \alpha_k d_k x_k^{(1)} - \alpha_k d_k x_k^{(2)} + d_k^2 = 1 - \alpha_k^2 + d_k^2. \tag{2.6}$$

However, because of (2.2) we have

$$1 - \alpha_k^2 + d_k^2 = 0. \tag{2.7}$$

Summing up, one factor of the right-hand side of (2.5) is zero, which implies that the whole product is zero and therefore yields

$$\sum_{R \subset \{1,\dots,n\}} \prod_{k \in R} (\alpha_k x_k^{(1)} - d_k)(\alpha_k x_k^{(2)} - d_k) = 0. \tag{2.8}$$

Case 2: $(x^{(1)} = x^{(2)})$. In this case the right-hand side of (2.5) reduces to

$$\begin{aligned} &\prod_{k=1}^{n} (1 + \alpha_k^2 x_k^{(1)} x_k^{(2)} - \alpha_k d_k x_k^{(1)} - \alpha_k d_k x_k^{(2)} + d_k^2) \\ &= \prod_{k=1}^{n} (1 + \alpha_k^2 - 2\alpha_k d_k x_k^{(1)} + d_k^2) \\ &= \prod_{k=1}^{n} (1 + (\alpha_k x_k^{(1)} - d_k)^2) \\ &= \prod_{k=1}^{n} (1 + (\sqrt{1 + d_k^2}\, x_k^{(1)} - d_k)^2) > 1, \end{aligned} \tag{2.9}$$

which implies

$$\sum_{R\subset\{1,\dots,n\}}\prod_{k\in R}(\alpha_k x_k^{(1)}-d_k)(\alpha_k x_k^{(2)}-d_k)>1. \qquad \square \tag{2.10}$$

Remark. For $d_k=0$, $1\leqslant k\leqslant n$, the above result reduces to the well known sigma-pi orthogonality of bipolar coded vectors, or – in other words – the orthogonality property of Walsh functions (see [3, 5, 7]).

With the above theorem we are prepared to prove the following basic result.

Theorem 2.2. *Let* $n,t\in\mathbb{N}$, $d_k\in\mathbb{R}$, $1\leqslant k\leqslant n$, *and*

$$(\boldsymbol{x}^{(s)},y^{(s)})\in\{-1,1\}^n\times\{-1,1\},\quad 1\leqslant s\leqslant t, \tag{2.11}$$

with $\boldsymbol{x}^{(r)}\neq\boldsymbol{x}^{(s)}$ *for* $r\neq s$ *be given arbitrarily. Then, with* $\alpha_k\in\mathbb{R}$, $1\leqslant k\leqslant n$, *and* $w_R\in\mathbb{R}$, $R\subset\{1,\dots,n\}$, *defined as*

$$\alpha_k:=\sqrt{1+d_k^2},\quad 1\leqslant k\leqslant n, \tag{2.12}$$

$$w_R:=\sum_{s=1}^{t}y^{(s)}\prod_{k\in R}(\alpha_k x_k^{(s)}-d_k),\quad R\subset\{1,\dots,n\}, \tag{2.13}$$

and T_S *as in* (1.7), *the function* $p:\{-1,1\}^n\to\{-1,1\}$,

$$p(\boldsymbol{x}):=T_S\left(\sum_{R\subset\{1,\dots,n\}}w_R\prod_{k\in R}(\alpha_k x_k-d_k)\right), \tag{2.14}$$

satisfies the interpolation conditions

$$p(\boldsymbol{x}^{(s)})=y^{(s)},\quad 1\leqslant s\leqslant t. \tag{2.15}$$

Proof. Let $s\in\{1,\dots,t\}$ be given arbitrarily and $\alpha_k\in\mathbb{R}$, $1\leqslant k\leqslant n$, and $w_R\in\mathbb{R}$, $R\subset\{1,\dots,n\}$, be defined by (2.12) and (2.13), respectively. Then, by means of (2.14) we have

$$\begin{aligned}
p(\boldsymbol{x}^{(s)}) &= T_S\left(\sum_{R\subset\{1,\dots,n\}}w_R\prod_{k\in R}(\alpha_k x_k^{(s)}-d_k)\right)\\
&= T_S\left(\sum_{R\subset\{1,\dots,n\}}\left(\sum_{r=1}^{t}y^{(r)}\prod_{k\in R}(\alpha_k x_k^{(r)}-d_k)\right)\prod_{k\in R}(\alpha_k x_k^{(s)}-d_k)\right)\\
&= T_S\left(\sum_{r=1}^{t}y^{(r)}\left(\sum_{R\subset\{1,\dots,n\}}\prod_{k\in R}(\alpha_k x_k^{(r)}-d_k)(\alpha_k x_k^{(s)}-d_k)\right)\right).
\end{aligned} \tag{2.16}$$

Applying Theorem 2.1 and using the fact that $x^{(r)} \neq x^{(s)}$ for $r \neq s$ we finally obtain

$$p(x^{(s)}) = T_S\left(y^{(s)} \underbrace{\sum_{R \subset \{1,\ldots,n\}} \prod_{k \in R} (\alpha_k x_k^{(s)} - d_k)^2}_{>1} \right)$$

$$= T_S(y^{(s)}) = y^{(s)}. \quad \square \tag{2.17}$$

In view of the above theorem the free translation parameters d_k, $1 \leqslant k \leqslant n$, (implying the dilation parameters α_k, $1 \leqslant k \leqslant n$, and the sigma-pi weights w_R, $R \subset \{1,\ldots,n\}$, via (2.12) and (2.13), respectively) obviously play a crucial rôle. For any choice of these parameters the function p defined by (2.14) will interpolate the given data. One idea for using the freedom of choosing the translation parameters in a clever way could be the following: Choose the translation parameters d_k, $1 \leqslant k \leqslant n$, in such a way that as much sigma-pi weights w_R as possible become zero or close to zero so that they may be dropped without violating any of the interpolaton conditions (2.15). This kind of strategy would probably reduce the complexity of p which is our general aim. Unfortunately, at the moment we do not have any simple technique to figure out in advance how to choose the translation parameters d_k, $1 \leqslant k \leqslant n$, in order to make as much sigma-pi weights superfluous as possible. Therefore, in the following section we will proceed in a way which is the standard one in the literature. We will assume that the set of interpolation points (2.11) has been analyzed in advance and a fixed set Γ of subsets of $\{1,\ldots,n\}$ has been found to be quite well suited to realize the interpolation function p. With Γ a fixed set of subsets of $\{1,\ldots,n\}$ we now still have the freedom to choose the translation parameters d_k, $1 \leqslant k \leqslant n$, therefore increasing the likelihood of still being able to fit the interpolation conditions by the reduced set of sigma-pi weights. In detail, we will now discuss this approach in context of sigma-pi Hopfield neural networks.

3. Application to incomplete sigma-pi Hopfield neural networks

We assume that the reader is quite familiar with usual higher order networks of Hopfield-type (see [4, Ch. 5], for a detailed introduction). These networks are used in order to store a set of t different bipolar coded vectors

$$x^{(1)}, x^{(2)}, x^{(3)}, \ldots, x^{(t)} \in \{-1, 1\}^n \tag{3.1}$$

and reconstruct them even in case of slightly disturbed input information. More precisely, in case that one of the stored vectors $x^{(s)}$ or a damaged version of it is entered into the network it should be able to generate the correct output $x^{(s)}$ using a kind of Gauß–Seidel iteration. In neural network theory this reconstruction iteration is usually called recall mode. In the following, we give the general definition of the recall mode

of sigma-pi Hopfield neural networks including dilation and translation, where # always stands for the number of elements of the respective set under consideration (see [6] for a more gentle introduction to these special kind of generalized Hopfield-type networks).

Definition 3.1 (*Recall mode*). Let $n \in \mathbb{N}$ and Γ be a non-empty collection of subsets of $\{1,\dots,n\}$. The generalized sigma-pi Hopfield neural network may have n neurons, $\#\Gamma$ sigma-pi weights $w_R \in \mathbb{R}$, $R \in \Gamma$, and n dilation and translation weights α_k, $d_k \in \mathbb{R}$, $\alpha_k > 0$, $k \in \{1,\dots,n\}$, which we assume to be generated by means of any reasonable learning algorithm. Now, in case that an input vector is entered,

$$\boldsymbol{x} =: \boldsymbol{x}^{[0]} = (x_1^{[0]}, x_2^{[0]}, \dots, x_n^{[0]}) \in \{-1,1\}^n, \tag{3.2}$$

the generalized sigma-pi Hopfield network generates a sequence of vectors $(\boldsymbol{x}^{[u]})_{u \in \mathbb{N}_0}$ via

$$a_j^{[u]} := \sum_{\substack{R \in \Gamma \\ j \in R}} w_R \prod_{\substack{k \in R \\ k < j}} (\alpha_k x_k^{[u+1]} - d_k) \prod_{\substack{k \in R \\ k > j}} (\alpha_k x_k^{[u]} - d_k), \tag{3.3}$$

$$x_j^{[u+1]} := \begin{cases} T_S(a_j^{[u]}) & \text{for } a_j^{[u]} \neq 0, \\ x_j^{[u]} & \text{for } a_j^{[u]} = 0, \end{cases} \tag{3.4}$$

for $1 \leqslant j \leqslant n$ and $u \in \mathbb{N}_0$. As the output vector the so-defined recall mode yields the vector $\boldsymbol{x}^{[v]} \in \{-1,1\}^n$, which for the first time satisfies

$$\boldsymbol{x}^{[v]} = \boldsymbol{x}^{[v+1]} \tag{3.5}$$

for some $v \in \mathbb{N}_0$. The sigma-pi network is called complete of order r, $r \in \{1,2,\dots, n-1\}$, in case that Γ coincides with the set of all subsets of $\{1,\dots,n\}$ with at most $r+1$ elements; if no such r exists, we say that it is incomplete.

There are at least two additional remarks which should be added in connection with the above definition. First of all, the above definition implicitly states that the recall mode terminates, i.e., that there exists a natural number $v \in \mathbb{N}_0$ satisfying

$$\boldsymbol{x}^{[v]} = \boldsymbol{x}^{[v+1]}, \tag{3.6}$$

which immediately implies

$$\boldsymbol{x}^{[v]} = \boldsymbol{x}^{[v+u]}, \quad u \in \mathbb{N}. \tag{3.7}$$

This has been proved in [6]. Secondly, as defined above the dilation, translation and sigma-pi weights (parameters) are not yet tuned in order to store the bipolar vectors given by (3.1). This is explicitly done in a so-called learning mode which we define as follows.

Definition 3.2 (*Learning mode*). Let $n \in \mathbb{N}$ and Γ be a non-empty collection of subsets of $\{1,\dots,n\}$. The generalized sigma-pi Hopfield neural network may have n neurons,

$\#\Gamma$ sigma-pi weights $w_R \in \mathbb{R}$, $R \in \Gamma$, and n dilation and translation weights $\alpha_k, d_k \in \mathbb{R}$, $\alpha_k > 0$, $k \in \{1, \ldots, n\}$. Now, in case that t bipolar coded vectors

$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \ldots, \boldsymbol{x}^{(t)} \in \{-1, 1\}^n \tag{3.8}$$

are entered into the network in order to be stored, then the algorithm which sets

$$d_k := \lambda \sum_{s=1}^{t} x_k^{(s)}, \quad 1 \leqslant k \leqslant n, \tag{3.9}$$

$$\alpha_k := \sqrt{1 + d_k^2}, \quad 1 \leqslant k \leqslant n, \tag{3.10}$$

$$w_R := \sum_{s=1}^{t} \prod_{k \in R} (\alpha_k x_k^{(s)} - d_k), \quad R \in \Gamma, \tag{3.11}$$

with $\lambda \geqslant 0$ a free learning parameter, is called generalized Hebb learning scheme.

Note that the generalized Hebb learning scheme defined above reduces to the usual Hebb learning scheme for sigma-pi Hopfield neural networks in case that $\lambda = 0$ or $d_k = 0$, $1 \leqslant k \leqslant n$ (see [4, Ch. 5, Section 5.1.2]). However, for $d_k > 0$ ($d_k < 0$) the corresponding activation factor $(\alpha_k x_k - d_k)$ lies between 0 and 1 (-1 and 0) for $x_k = 1$ ($x_k = -1$) while it is less than -1 (greater than 1) for $x_k = -1$ ($x_k = 1$). Since the sign of d_k indicates whether there are more 1 ($d_k > 0$) or more -1 ($d_k < 0$) entries in the kth component of all training patterns the proposed learning scheme makes an x_k-value with low probability enter the network in a moderately overemphasized way, therefore increasing the probability of suitable changes in network activity. On the other hand, a quite likely x_k-value is moderately damped by dilation and translation indicating the network that no dramatic changes in network state are necessary.

Let us discuss the special choice of d_k, $1 \leqslant k \leqslant n$, as fixed in (3.9) also from another point of view. As it is well known each state vector $\boldsymbol{x} \in \{-1, 1\}^n$ of the Hopfield neural network can be associated with a so-called energy value, namely,

$$E(\boldsymbol{x}) := -\sum_{R \in \Gamma} w_R \prod_{k \in R} (\alpha_k x_k - d_k). \tag{3.12}$$

The general design strategy of Hopfield neural networks is to choose the network parameters w_R, α_k and d_k in such a way that the vectors (3.8) to be stored become local minima of the energy function E (see [1, 4, 6, 8] for details). Now, let us check the implication of this general attempt in case that we fix α_k and w_R by (3.10) and (3.11) according to our orthogonality argument and only allow to vary the d_k. For simplicity, let us further only consider the case that Γ coincides with the set of all

subsets of $\{1,2,\ldots,n\}$ and let $r\in\{1,2,\ldots,t\}$ be given arbitrarily. We obtain

$$\begin{aligned} E(\boldsymbol{x}^{(r)}) &= -\sum_{R\subset\{1,\ldots,n\}}\sum_{s=1}^{t}\prod_{k\in R}(\alpha_k x_k^{(s)}-d_k)(\alpha_k x_k^{(r)}-d_k) \\ &= -\sum_{s=1}^{t}\prod_{k=1}^{n}(1+(\sqrt{1+d_k^2}x_k^{(s)}-d_k)(\sqrt{1+d_k^2}x_k^{(r)}-d_k)) \\ &= -\prod_{k=1}^{n}(1+(\sqrt{1+d_k^2}x_k^{(r)}-d_k)^2), \end{aligned} \tag{3.13}$$

where we have to apply the same arguments as used in proving Theorem 2.1. Now, depending on the choice of d_k, $1\leqslant k\leqslant n$, the following different situations can appear:

(1) $d_k=0$ for all $k\in\{1,2,\ldots,n\}$: This is the classical choice which implies that each factor in the final representation of $E(\boldsymbol{x}^{(r)})$ given by (3.13) becomes equal to 2. Therefore,

$$E(\boldsymbol{x}^{(r)}) = -2^n, \quad 1\leqslant r\leqslant t, \tag{3.14}$$

i.e. each training pattern has the same energy. Especially, this choice does not make any difference between easy-to-store and difficult-to-store training patterns.

(2) $d_k\neq 0$ for some $k\in\{1,2,\ldots,n\}$: This choice implies that for a fixed given training pattern $\boldsymbol{x}^{(r)}$ the kth factor

$$(1+(\sqrt{1+d_k^2}x_k^{(r)}-d_k)^2) \tag{3.15}$$

appearing in the final representation of $E(\boldsymbol{x}^{(r)})$ given by (3.13) becomes less than 2 for $d_k x_k^{(r)}>0$ and greater than 2 for $d_k x_k^{(r)}<0$. The total effect on the energy $E(\boldsymbol{x}^{(r)})$ is that it increases for $d_k x_k^{(r)}>0$ (decreasing the probability of obtaining a local minimum in $\boldsymbol{x}^{(r)}$) while it decreases for $d_k x_k^{(r)}<0$ (increasing the probability of obtaining a local minimum in $\boldsymbol{x}^{(r)}$). If we now can manage to fix the parameter d_k in such a way that $d_k x_k^{(r)}>0$ indicates that the kth component of $\boldsymbol{x}^{(r)}$ is easy to store while $d_k x_k^{(r)}<0$ is a hint for a difficult-to-store kth component our translation strategy should work. Indeed, in case that

$$d_k := \lambda\sum_{s=1}^{t}x_k^{(s)}\neq 0, \quad \lambda>0, \tag{3.16}$$

is defined as in (3.9) and different from zero, $d_k x_k^{(r)}>0$ implies that $x_k^{(r)}$ has the same value as most kth components of the whole set of training patterns (easy-to-store) while $d_k x_k^{(r)}<0$ indicates that $x_k^{(r)}$ has a quite rare sign in comparison with all other kth components of the training patterns (difficult-to-store). Summing up, a proper signed d_k with moderate absolute value (rôle of λ) as fixed in (3.16) fits the job: It slightly decreases the energy of difficult-to-store patterns with rare sign distributions of the components while it cautiously increases the energy of easy-to-store patterns hopefully without destroying their local minima properties.

Of course, all the above arguments are only a heuristic attempt to motivate definition (3.9), where $\lambda \geqslant 0$ is still a free learning parameter to be chosen appropriately (whatever that means). In contrast, definitions (3.10) and (3.11) are not of this experimental type since they are the precise counterparts of definition (2.12) and (2.13) in the Hopfield setting. In other words, while definition (3.9) may perhaps not be the best way to fix the translation parameters, definitions (3.10) and (3.11) for the dilation parameters and the sigma-pi weights are tuned best possible in the spirit of having discrete orthogonality in the background. In the following, we will underline the power of our new learning scheme by applying our generalized sigma-pi Hopfield neural networks to some test problems. The problems consist in learning a number of different patterns of (4×5)-dimension $(n = 20)$ composed of "." (to be identified with -1) and "X" (to be identified with 1). The patterns are either simple realizations of capital and small letters or randomly generated. In case of randomly generated patterns the ratio of the total number of "." against the total number of "X" can be chosen in order to obtain highly correlated and therefore difficult to store situations. For a number of different values of λ the program learns the respective weights (3.9)–(3.11), then successively starts the recall mode (3.3) and (3.4) for all learning patterns and, simultaneously, calculates the total error produced by the respective network on the whole training set. In detail, total error calculation means that each mismatched component on a training pattern increases the error by one. In view of the definitions given previously we finally only have to choose the set Γ which determines the set of active sigma-pi weights. Here, we choose Γ to consist of all subsets of $\{1,2,\ldots,20\}$ with precisely three elements,

$$\Gamma = \{\{1,2,3\},\{1,2,4\},\{1,2,5\},\ldots,\{18,19,20\}\}. \tag{3.17}$$

This means that we generate an incomplete situation and the introduction of dilation and translation weights especially makes sense (see [6] for a detailed argument to focus on incomplete situations in presence of dilation and translation parameters). Now, let us take a look at three typical program runs to be discussed afterwards.

1. Learning letters:

```
XXXXX  X...X  XXXXX  .X...  X...X  X...X
..X..  XXXXX  X....  .X...  .X.X.  .X.X.
..X..  X...X  X....  .X...  ..X..  ..X..
..X..  X...X  XXXXX  .XXX.  .X.X.  ..X..

..X..  .X...  .....  ..X..  .....  .....
XXXXX  .X...  ..XX.  ..X..  .X.X.  .X.X.
..X..  .XXX.  .X...  ..X..  ..X..  ..X..
..X..  .X.X.  ..XX.  ..X..  .X.X.  ..X..
lambda:       0.000 0.004 0.008 0.012 0.016 0.020 0.024 0.028 0.032 0.036 0.040 0.044
total error: 15    11    7     3     1     1     3     3     3     3     3     7
```

2. Learning correlated random patterns:

```
XXX.X  .XX.X  XXXXX  X.XXX  X..X.  XXXXX  XXXXX
XXXXX  .XXXX  .XXXX  .XXXX  X.XXX  XX.XX  ..X..
X.XXX  XXX..  XXXXX  XXXXX  XX...  .X.XX  XXXXX
XXXX.  X.XX.  .XXXX  XXX.X  XXXXX  XXXX.  XXXXX

X.XXX  X.XX.  XX.XX  XXXXX  X.XXX  XXXXX  XXXXX
.XXX.  XXXX.  XXXXX  XXXXX  XXXXX  XXXXX  .XXX.
XX..X  XXXXX  XXXX.  .XXXX  XX.XX  X..XX  XX.X.
XXXXX  XXXXX  XXX.X  .XXX.  .X.XX  XXXXX  ...X.
total number of <.>-marks: 58
total number of <X>-marks: 222
```

lambda:	0.000	0.004	0.008	0.012	0.016	0.020	0.024	0.028	0.032	0.036	0.040	0.044
total error:	36	36	36	25	19	4	4	0	0	0	6	6

3. Learning uncorrelated random patterns:

```
XX.X.  XX.X.  XX.XX  X...X  XXX.X  XX.X.  .X..X  X.X..  XXXX.  .X.XX
X.XXX  XX...  XXXX.  ...X.  XXXX.  .X.X.  X....  .....  .X.X.  XXXX.
XX...  .X.XX  .XXX.  X.X.X  .XX.X  XX...  X.XX.  X..X.  XXX.X  X.XX.
X.X..  ..XX.  XXX.X  XX...  ...X.  X.X.X  ....X  .XXXX  X...X  X.XXX

X.X.X  .X..X  ...X.  XX...  .X.XX  X.X.X  XX..X  X....  .X.XX  ....X
..X..  XXXXX  X.X.X  X.X..  .....  .XXXX  .XXXX  .XX..  .X.X.  .X.X.
.X..X  XX..X  .X.XX  XX...  X..X.  X...X  XXXXX  .XX.X  X.X.X  XXXX.
.XX..  .XXXX  ...X.  X..X.  X..XX  XXXX.  ...XX  X..XX  .XXXX  .X..X
total number of <.>-marks: 190
total number of <X>-marks: 210
```

lambda:	0.000	0.004	0.008	0.012	0.016	0.020	0.024	0.028	0.032	0.036	0.040	0.044
total error:	0	0	0	0	0	0	5	5	6	6	6	12

The first and, especially, the second example (and, of course, further non-reported experiments) show that for a specific set of positive parameters λ the total recall error on training sets of (highly) correlated patterns is in general significantly smaller than for the classical choice $\lambda = 0$. Only in the easy-to-handle situation of uncorrelated patterns ($d_k \approx 0$, $1 \leqslant k \leqslant n$) there is essentially no difference between choosing λ identically zero or close to zero. However, such completely uncorrelated situations are not very typical in practise. Moreover, similar results are obtained when Γ is replaced by any other set of subsets of $\{1, 2, \ldots, 20\}$ generating a so-called incomplete situation. Finally, let us note that quite recently some studies have been started in order to investigate the behaviour of our dilation and translation networks in case of synchronous recall updates instead of asynchronous ones. Detailed results on this topic may appear elsewhere.

4. Concluding remarks

In this paper, we again considered dilation and translation parameters for sigma-pi Hopfield neural networks in order to increase their flexibility while keeping the number of allowed sigma-pi weights fixed. In order to find a reasonable strategy to fix the free additional parameters we proved a new discrete orthogonality condition for Walsh-type functions with dilation and translation. Based on the restrictions of dilation and translation parameters implying this new kind of discrete orthogonality we proposed a new learning algorithm for so-called generalized sigma-pi Hopfield neural networks and studied its power by considering some concrete applications. The basic conclusion from these (and other) experiments was that at least in case of highly correlated information our choice of dilation and translation parameters significantly increases the capacity of the networks while leaving their complexity almost unaffected.

References

[1] H.H. Chen, Y.C. Lee, T. Maxwell, G.Z. Sun, H.Y. Lee, C.L. Giles, High order correlation model for associative memory, in: J.S. Denker (Ed.), AIP Conference Proceedings, American Institute of Physics, New York, 1986, pp. 86–99.

[2] G. Fahner, R. Eckmiller, Structural adaptation of parsimonious higher-order neural classifiers, Neural Networks 7 (1994) 279–289.

[3] E. Gauß, Walsh-Funktionen für Ingenieure und Naturwissenschaftler, B.G. Teubner, Stuttgart, 1994.

[4] Y. Kamp, M. Hasler, Recursive Neural Networks for Associative Memory, Wiley, Chichester, 1990.

[5] M.G. Karpovsky, Finite Orthogonal Series in the Design of Digital Devices, Wiley, New York, 1976.

[6] B. Lenze, Introducing dilation and translation for incomplete sigma-pi neural networks of Hopfield-type, in: F. Fontanella, K. Jetter, P.-J. Laurent (Eds.), Advanced Topics in Multivariate Approximation, World Scientific, Singapore, 1996, pp. 197–210.

[7] K. Niederdrenk, Die endliche Fourier- und Walsh-Transformation mit einer Einführung in die Bildverarbeitung, Vieweg & Sohn, Braunschweig, 1982.

[8] D. Psaltis, C.H. Park, J. Hong, Higher order associative memories and their optical implementations, Neural Networks 1 (1988) 149–163.