A New Synthesis Approach for Feedback Neural Networks Based on the Perceptron Training Algorithm

Derong Liu, *Senior Member, IEEE,* and Zanjun Lu, *Student Member, IEEE*

***Abstract*— In this paper, a new synthesis approach is developed for associative memories based on the perceptron training algorithm. The design (synthesis) problem of feedback neural networks for associative memories is formulated as a set of linear inequalities such that the use of perceptron training is evident. The perceptron training in the synthesis algorithms is guaranteed to converge for the design of neural networks without any constraints on the connection matrix. For neural networks with constraints on the diagonal elements of the connection matrix, results concerning the properties of such networks and concerning the existence of such a network design are established. For neural networks with sparsity and/or symmetry constraints on the connection matrix, design algorithms are presented. Applications of the present synthesis approach to the design of associative memories realized by means of other feedback neural network models are studied. To demonstrate the applicability of the present results and to compare the present synthesis approach with existing design methods, specific examples are considered.**

***Index Terms*— Associative memory, neural networks, perceptron.**

I. Introduction

SINGLE-LAYER fully connected feedback neural networks are a special class of nonlinear dynamical systems which are endowed with many asymptotically stable equilibrium points (stable memories) as well as unstable equilibria. The study of such systems has been of great interest to many researchers in recent years (see, e.g., [1], [4]–[9], [11]–[15], [17], [21]–[23], [26]–[29], [31], and [33]). These works are concerned with the qualitative analysis of neural networks [1], [4], [5], [7]–[9], [11]–[15], [17], [22], [23], [31], and design methodologies for such networks [6], [8], [9], [11], [14], [15], [17], [21]–[23], [26]–[29], [33]. The study of single-layer sparsely interconnected feedback neural networks has also been of recent interest [2], [3], [17]–[20], [24], [32].

In the present paper, the realization of *associative memories* via a class of neural networks is considered. The goal of associative memories is to store a set of desired patterns as stable memories such that a stored pattern can be retrieved when the input pattern (or the initial pattern) contains sufficient information about that stored pattern. In practice the desired memory patterns are usually represented by bipolar vectors (or binary vectors).

There are several well-known synthesis methods available in the literature including the *outer product method* [11], the *projection learning rule* [28], [29], and the *eigenstructure method* [15], [23] (for an overview of these synthesis methods, see [21]). The outer product method requires that desired patterns be mutually orthogonal in order for all the desired patterns to be stored in the network. The projection learning rule does not require that prototype patterns be mutually orthogonal; but this method cannot guarantee that an equilibrium corresponding to a given desired memory is asymptotically stable. The eigenstructure method appears to be the most effective. It can guarantee to store any set of bipolar patterns as stable memories which need not be mutually orthogonal and which correspond to asymptotically stable equilibria of a neural network. The eigenstructure method has also been generalized for the synthesis of neural networks with predetermined constraints on the interconnecting structure [17]–[20]. In these synthesis methods, a set of linear *equations* is formulated and solved for the design of neural networks. In the design method advanced in [8] and [9], a set of linear *inequalities* is formulated and solved for the optimal mean-square-error (MSE) solution using the Ho–Kashyap method [10]. In the design method developed in [27], a set of linear inequalities is formulated and solved using optimization techniques. In the design method presented in [32], a set of linear inequalities is formulated and solved using linear programming.

This paper makes contributions to feedback neural networks for associative memories. In particular, a new synthesis approach will be developed based on the perceptron training algorithm. The synthesis approach of the present paper is developed by formulating and solving a set of linear *inequalities*. The inequalities are solved by training a set of perceptrons to obtain the connection matrix, and the perceptron training is guaranteed to converge for the design of neural networks with no constraints on the connection matrix. For networks with sparsity and/or symmetry constraints on the connection matrix and with constraints on the diagonal elements of the connection matrix, conditions under which a design solution exists will be established and design algorithms will be provided.

Manuscript received July 15, 1996; revised January 27, 1997 and April 2, 1997. This work was supported by Stevens Institute of Technology.

The authors are with the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ 07030-5991 USA.

Publisher Item Identifier S 1045-9227(97)07515-2.

The class of feedback neural networks considered in the present paper is described by equations of the form

$$\begin{aligned}\dot{x} &= -Ax + T\,\mathrm{sat}\,(x) + I\\ y &= \mathrm{sat}\,(x)\end{aligned} \tag{1}$$

where $x \in \Re^n$ is the state vector, $\dot{x}$ denotes the derivative of x with respect to time $t, y \in D^n \overset{\Delta}{=} \{x \in \Re^n : -1 \le x_i \le 1, i = 1, \cdots, n\}$ is the output vector, $A = \mathrm{diag}\,[a_1, \cdots, a_n]$ with $a_i > 0$ for $i = 1, \cdots, n, T = [T_{ij}] \in \Re^{n \times n}$ is the coefficient (or connection) matrix, $I = [I_1, \cdots, I_n]^T \in \Re^n$ is a bias vector, and $\mathrm{sat}\,(x) = [\mathrm{sat}\,(x_1), \cdots, \mathrm{sat}\,(x_n)]^T$ represents the activation function, where

$$\mathrm{sat}\,(x_i) = \begin{cases} 1, & x_i > 1\\ x_i, & -1 \le x_i \le 1\\ -1, & x_i < -1.\end{cases}$$

It is assumed that the initial states of (1) satisfy $|x_i(0)| \le 1$ for $i = 1, \cdots, n$. System (1) is a variant of the analog Hopfield model [12] with activation function $\mathrm{sat}\,(\cdot)$.

The design (synthesis) problem of neural networks (1) for associative memories will be formulated such that the use of perceptron training is evident. Preliminaries will be first introduced in Section II concerning the perceptron training algorithm and its convergence theorem. The rest of the present paper is organized as follows.

- In Section III, a new synthesis algorithm for associative memories will be developed based on the perceptron training algorithm.
- In Section IV, results concerning the properties of neural networks with constraints on the diagonal elements of the connection matrix and concerning the existence of such a network design will be established.
- In Section V, design algorithms for neural networks with connectivity constraints (sparsity and/or symmetry) will be presented.
- In Section VI, applications of the present synthesis approach to the design of associative memories realized by means of several other feedback neural-network models will be studied.
- In Section VII, specific examples will be considered to demonstrate the applicability of the present results and to compare the present synthesis approach with existing design methods.
- In Section VIII, the present paper will be concluded with several pertinent remarks.

II. Preliminaries

This section introduces necessary preliminaries including the perceptron training algorithm and its convergence theorem.

A number of different types of perceptrons are described in [25] and [30]. The one which will be utilized in the present paper is described by

$$z = \mathrm{sgn}\,(Wu) \tag{2}$$

where $u = [u_1, u_2, \cdots, u_n, 1]^T, W = [w_1, w_2, \cdots, w_n, \theta]$, and

$$\mathrm{sgn}\,(\xi) = \begin{cases} 1, & \xi \ge 0\\ -1, & \xi < 0.\end{cases}$$

This simple perceptron can perform pattern classification (between two classes denoted by X_1 and X_2). The weight vector W can be obtained by the following perceptron training algorithm (cf. [25] and [30]).

Perceptron Training Algorithm: Given m training patterns $\alpha^k, k = 1, 2, \cdots, m,$ which are known to belong to class X_1 (corresponding to $z = 1$) or X_2 (corresponding to $z = -1$), the weight vector W can be obtained by the following algorithm.

1) Initialize the weight vector $W(l)$ for $l = 0$.
2) For $l = 0, 1, 2, \cdots$
   a) if $W(l)u(l) \ge 0$ and $u(l) \in X_2$, then update $W(l+1) = W(l) - \eta u(l)$;
   b) if $W(l)u(l) < 0$ and $u(l) \in X_1$, then update $W(l+1) = W(l) + \eta u(l)$;
   c) otherwise, $W(l{+}1) = W(l)$, where $u(l) = \alpha^k$ for some $k, 1 \le k \le m,$ and $\eta > 0$ is the perceptron learning rate.
3) Stop the training when no more updates for the weight vector W are needed, i.e., stop the training when all the training patterns can be correctly classified by W.

The following result is well known [25], [30].

Perceptron Training Convergence Theorem: The perceptron training algorithm will be convergent if and only if the two classes X_1 and X_2 are linearly separable. ■

Remark 2.1: It is noted that one can always continue the training of a perceptron until a weight vector W is obtained such that

$$W\alpha^k > 0 \quad \text{if} \quad \alpha^k \in X_1$$

and

$$W\alpha^k < 0 \quad \text{if} \quad \alpha^k \in X_2$$

for $k = 1, 2, \cdots, m.$ ■

In the sequel, the perceptron training algorithm will be used to develop a new synthesis approach for associative memories realized by neural network (1).

III. A New Synthesis Algorithm

A vector α will be called a (*stable*) *memory vector* (or simply, a *memory*) of system (1) if $\alpha = \mathrm{sat}\,(\beta)$ and if β is an asymptotically stable equilibrium point of system (1). Recall that the equilibrium x_e of system (1) is *asymptotically stable* if 1) it is *stable* in the sense of Lyapunov; i.e., for every $\varepsilon > 0$ there is a $\delta = \delta(\varepsilon)$ such that $||x(t) - x_e|| < \varepsilon$ for all $t \ge 0$, whenever $||x(0) - x_e|| < \delta$ ($||\cdot||$ denotes any vector norm) and 2) it is *locally attractive*; i.e., $x(t) \to x_e$ as $t \to \infty$ whenever $x(0) \in D(x_e)$, where $D(x_e)$ is an open neighborhood of $\Re^n$ containing x_e. The largest set $D(x_e)$ for which the preceding property is true is called the *domain of attraction* or the *basin of attraction* of x_e.

Use B^n to denote the set of n-dimensional *bipolar vectors*, i.e., $B^n \triangleq \{x \in \Re^n: x_i = 1 \text{ or } -1, i = 1, \cdots, n\}$. For $\alpha = [\alpha_1, \alpha_2, \cdots, \alpha_n]^T \in B^n$, define $C(\alpha) = \{x \in \Re^n: x_i\alpha_i > 1, i = 1, \cdots, n\}$. A result for bipolar memories established in [17] and [19] states the following.

Lemma 3.1: If $\alpha \in B^n$ and if

$$\beta = A^{-1}(T\alpha + I) \in C(\alpha) \tag{3}$$

then (α, β) is a pair of stable memory vector and asymptotically stable equilibrium point of (1). ■

The proof of this result uses the fact that if $\alpha \in B^n$, then $\text{sat}(x) = \alpha$ for all $x \in C(\alpha)$. For $\alpha \in B^n$ and $x \in C(\alpha)$, the first equation of (1) can be written as

$$\dot{x} = -Ax + T\alpha + I. \tag{4}$$

System (4) has a unique equilibrium at $x_e = A^{-1}(T\alpha + I)$ and $x_e = \beta \in C(\alpha)$ by assumption. Clearly, this equilibrium is also asymptotically stable, since in system (4) all eigenvalues $\lambda_i(-A)$ of $-A$ are negative (since $\lambda_i(A) = a_i > 0$).

The following synthesis problem concerns the design of (1) for associative memories.

Synthesis Problem: Given m vectors $\alpha^1, \alpha^2, \cdots, \alpha^m$ in B^n, choose $\{A, T, I\}$ in such a manner that

1) $\alpha^1, \alpha^2, \cdots, \alpha^m$ are stable memory vectors of system (1);
2) the system has no oscillatory and chaotic solutions;
3) the total number of spurious memory vectors (i.e., memory vectors which are not desired) is as small as possible, and the domain (or basin) of attraction of each desired memory vector is as large as possible. ■

Item 1) of the synthesis problem can be guaranteed by choosing the $\{A, T, I\}$ such that every α^i satisfies condition (3) of Lemma 3.1. Item 2) can be achieved by designing a neural network with symmetric connection matrix T and will be addressed in Section V. Item 3) can be partly ensured by constraining the diagonal elements of the connection matrix and will be discussed in Section IV.

In this paper, it will be shown that the above synthesis problem can be solved by applying the perceptron training algorithm summarized in Section II.

To solve the synthesis problem, one needs to determine A, T and I from (3) with $\alpha = \alpha^k, k = 1, 2, \cdots, m$, i.e., one needs to determine A, T and I from

$$A^{-1}(T\alpha^k + I) \in C(\alpha^k) \quad \text{for} \quad k = 1, 2, \cdots, m. \tag{5}$$

Condition (5) can be equivalently written as

$$\begin{cases} T_i\alpha^k + I_i > a_i & \text{if} \quad \alpha_i^k = 1 \\ T_i\alpha^k + I_i < -a_i & \text{if} \quad \alpha_i^k = -1 \end{cases} \tag{6}$$

for $k = 1, 2, \cdots, m$ and for $i = 1, 2, \cdots, n$; where T_i represents the ith row of T, I_i denotes the ith element of I, and α_i^k is the ith entry of α^k. From (6) [or equivalently, from (5)], the following synthesis algorithm (design method) based on the perceptron training algorithm can now be obtained.

Synthesis Algorithm 3.1: Using the perceptron training algorithm, obtain n perceptrons $W^i = [w_1^i, w_2^i, \cdots, w_{n+1}^i], i = 1, 2, \cdots, n,$ such that

$$\begin{aligned} W^i\overline{\alpha}^k \geq 0 \quad &\text{if} \quad \alpha_i^k = 1 \\ W^i\overline{\alpha}^k < 0 \quad &\text{if} \quad \alpha_i^k = -1 \end{aligned} \tag{7}$$

and for $k = 1, 2, \cdots, m,$ where

$$\overline{\alpha}^k = \begin{bmatrix} \alpha^k \\ \cdots \\ 1 \end{bmatrix}.$$

Choose $A = \text{diag}[a_1, a_2, \cdots, a_n]$ with $a_i > 0$. For $i, j = 1, 2, \cdots, n,$ choose $T_{ij} = w_j^i$ if $i \neq j, T_{ii} = w_i^i + a_i\mu_i$ with $\mu_i > 1$, and $I_i = w_{n+1}^i$. ■

The next result addresses the validity of the above synthesis algorithm.

Theorem 3.1:

1) The A, T and I obtained in Synthesis Algorithm 3.1 satisfies the condition (5); i.e., (1) with A, T and I obtained in Synthesis Algorithm 3.1 does indeed satisfy the item 1) of the synthesis problem.
2) The perceptron training in Synthesis Algorithm 3.1 will always converge.

Proof:

1) Considering the definition of $\overline{\alpha}^k$ and the choice for T_{ij} and I_i in Synthesis Algorithm 3.1 with $a_i > 0$ and $\mu_i > 1$, one can see that for $i = 1, 2, \cdots n$ and $k = 1, 2, \cdots, m, W^i\overline{\alpha}^k \geq 0$ (when $\alpha_i^k = 1$) implies

$$\begin{aligned} &T_{i1}\alpha_1^k + T_{i2}\alpha_2^k + \cdots + T_{ii}\alpha_i^k + \cdots + T_{in}\alpha_n^k + I_i \\ &\quad \geq a_i\mu_i\alpha_i^k > a_i \end{aligned} \tag{8}$$

and $W^i\bar{\alpha}^k < 0$ (when $\alpha_i^k = -1$) implies

$$\begin{aligned} &T_{i1}\alpha_1^k + T_{i2}\alpha_2^k + \cdots + T_{ii}\alpha_i^k + \cdots + T_{in}\alpha_n^k + I_i \\ &\quad < a_i\mu_i\alpha_i^k < -a_i. \end{aligned} \tag{9}$$

The above two expressions can be written in a matrix-vector form as in (5). Therefore, $\alpha^k, k = 1, 2, \cdots, m,$ are indeed stable memory vectors of system (1) according to Lemma 3.1.

2) Consider the ith perceptron's training in Synthesis Algorithm 3.1. The training patterns $\overline{\alpha}^k$ in the algorithm are labeled according to the ith element α_i^k of α^k for $k = 1, 2, \cdots, m$. Since every $\alpha^k \in B^n$, it is clear that the two classes in the present case are linearly separable and this is true for every $i, i = 1, 2, \cdots, n$. ■

Remark 3.1: Note that it is always required to choose $\mu_i > 1$ in Synthesis Algorithm 3.1 from Lemma 3.1 and the proof of Theorem 3.1. ■

Remark 3.2: Condition (5) is also equivalent to

$$A^{-1}(T\alpha^k + I) = E_k\alpha^k \quad \text{for} \quad k = 1, 2, \cdots, m \tag{10}$$

where $E_k = \operatorname{diag}[e_1^k, e_2^k, \cdots, e_n^k]$ with $e_i^k > 1$ (to be determined) for $i = 1, 2, \cdots, n, k = 1, 2, \cdots, m$. From (8) and (9), it is clearly observed that Synthesis Algorithm 3.1 results in $e_i^k \geq \mu_i$. This implies that large μ_i in Synthesis Algorithm 3.1 will result in large e_i^k for (10). If one chooses A as the identity matrix and

$$T\alpha^k + I = \mu\alpha^k \quad \text{for} \quad k = 1, 2, \cdots, m \tag{11}$$

with every $e_i^k = \mu > 1$, then (5) is satisfied. Equation (11) has been employed in [17]–[20]. ■

Remark 3.3: It is clear from Synthesis Algorithm 3.1 that one has the freedom to choose matrix $A = \operatorname{diag}[a_1, a_2, \cdots, a_n]$ with $a_i > 0$ for system (1). It is without loss of generality that one can usually choose $a_i = 1$, i.e., choose A as the identity matrix in Synthesis Algorithm 3.1. ■

Remark 3.4: The learning rate η in the perceptron training algorithm can be any positive real number [25], [30]. If $\eta = 1$ or any other positive integer and if one chooses the initial W^i to be the zero vector or any vector with integer and zero components, then, the matrix T and the vector I obtained from Synthesis Algorithm 3.1 will have only integer components. It is noted that in very large scale integration (VLSI) implementations of neural networks, certain weights (e.g., integers) can be implemented *more* accurately than others (e.g, numbers with many decimal digits). The learning rate η specifies the step size of every update for the weight vector during the perceptron training. A big η gives large step size which implies a coarse searching in the solution space of W^i. In most cases, choosing η to be $0 < \eta < 1$ is desirable for the perceptron training to converge quickly. ■

Remark 3.5: From Synthesis Algorithm 3.1 and Theorem 3.1, one can see that there are no restrictions on how many bipolar vectors can be stored as stable memories in system (1) designed by the present approach. This implies that the storage capacity (maximum allowable number m for the desired patterns) can be very large. It will be seen in Section IV, however, that there are certain constraints on the desired memory patterns if one wants to design a neural network (1) with some prespecified constraints on the diagonal elements of the connection matrix T (cf. Remark 4.7 of the next section). A more sophisticated definition of associative memory capacity has also been used in the literature [8], [9]. The capacity used in [8] and [9] is defined as a measure of the ability of an associative memory to store a set of *unbiased random* binary patterns at a given error correction and recall accuracy level (e.g., 99%). Using this definition of capacity, the present method and the Ho–Kashyap recording [8], [9] will be compared (cf. Example 7.4). ■

Remark 3.6: If one wishes that the above synthesis algorithm result in a system of form (1) with $I = 0$, one can modify (7) in Synthesis Algorithm 3.1 as follows:

$$W^i\alpha^k \geq 0 \quad \text{if} \quad \alpha_i^k = 1$$

and

$$W^i\alpha^k < 0 \quad \text{if} \quad \alpha_i^k = -1$$

where $W^i = [w_1^i, w_2^i, \cdots, w_n^i]$. Choose A and T as in Synthesis Algorithm 3.1, and choose $I = 0$. Theorem 3.1 is still valid for this case. ■

IV. Neural Networks with Constraints on the Diagonal Elements of the Connection Matrix

The key questions in evaluating associative memory designs are

- storage capacity;
- spurious memories;
- basins of attraction of desired memories.

As pointed out in Remark 3.5, the storage capacity (maximum allowable number m for the desired patterns) in the present case can be very large. When the number of desired memory patterns becomes too large, the connection matrix T obtained from many existing synthesis methods (see, e.g., [6], [11], [14], [15], [17]–[23], [28], and [29]) will be diagonally dominant or nearly diagonally dominant. A diagonally dominant connection matrix T will make almost every bipolar pattern (every corner of the hypercube) become a stable memory vector which results in many spurious memories (stable memories which are not desired) and which results in a useless network. In practice, one has to consider a tradeoff between storage capacity and the number of spurious memories.

Generally speaking, large positive diagonal elements in the connection matrix T will result in a large number of spurious memories (cf. Example 7.3). The basins of attraction of desired memories will get smaller as the total number of spurious memories in a network gets larger. One of the goals of the synthesis approach in the present paper is to design neural networks with constraints on the diagonal elements of the connection matrix (e.g., lower and/or upper bounded) so that the number of spurious memories may be reduced and that the basins of attraction of desired memories may be increased.

The following corollary provides some guidelines on what *lower bounds* can be used for the diagonal elements of the connection matrix T when it is desired that neural network (1) has only bipolar stable memory vectors. This result is a generalization of Theorem 5 of [3] and Theorem 1 of [27] (for a different model) to the present case.

Corollary 4.1: Assume that in system (1), $T_{ii} \geq a_i$ for $i = 1, 2, \cdots, n$, where $a_i > 0$ is the ith diagonal element of matrix A. Then, all the stable memory vectors of system (1) will be in B^n, i.e., the system will have no stable memories in $D^n \backslash B^n$, where $D^n = \{x \in \Re^n: -1 \leq x_i \leq 1, i = 1, \cdots, n\}$ represents the n-dimensional hypercube.

Proof: The proof can be done by considering the notation used in [19] and following similar steps as in [27] to argue that at least one of the eigenvalues of every linear subsystem of (1) has nonnegative real part (cf. Theorem 1 of [19] and Theorem 1 of [27]). It is noted that the similar result obtained in [3] applies to system (1) only when matrix T is symmetric and that the proof in [3] is different from the present one. ■

For $\alpha, \gamma \in B^n$, define the Hamming distance between α and γ as

$$H(\alpha,\gamma) = \tfrac{1}{2}\sum_{i=1}^{n} |\alpha_i - \gamma_i|$$

i.e., $H(\alpha,\gamma) =$ the number of bits for which α and γ differ.

The next result summarizes some important properties for neural networks (1) with the connection matrix T having *upper bounds* on the diagonal elements.

Theorem 4.1: Suppose that α is a stable memory vector of system (1) with $A = \operatorname{diag}[a_1, a_2, \cdots, a_n]$, where $a_i > 0$ for $i = 1, 2, \cdots, n$.

1) None of the vectors $\gamma \in B^n$ such that $H(\alpha,\gamma) = 1$ can become memory vectors of system (1) if $T_{ii} \le a_i$ for $i = 1, 2, \cdots, n$.
2) Assume that $T_{ii} \le a_i$ for some $i, \gamma \in B^n, H(\alpha,\gamma) = 1$, and α and γ differ in the ith bit. Then, the ith component of the state of system (1) will move in the direction toward α if the system starts from $x(0) = \gamma$.

Proof:

1) From Lemma 3.1, $\alpha = [\alpha_1, \alpha_2, \cdots, \alpha_n]^T \in B^n$ is a stable memory vector if (3) is satisfied. Assume that $\gamma = [\gamma_1, \gamma_2, \cdots, \gamma_n]^T \in B^n, H(\alpha,\gamma) = 1$, and α and γ differ in the ith bit, $1 \le i \le n$, i.e., $\alpha_j = \gamma_j$ for $j \ne i$ and $\alpha_i = -\gamma_i$. According to (3), if $T_{ii} \le a_i$, one has [cf. (6)]

$$\begin{cases} T_i\gamma + I_i = T_i\alpha + I_i + 2T_{ii}\gamma_i > a_i - 2T_{ii} \\ \qquad \ge -a_i \quad \text{if} \quad \gamma_i = -\alpha_i = -1 \\ T_i\gamma + I_i = T_i\alpha + I_i + 2T_{ii}\gamma_i < -a_i + 2T_{ii} \\ \qquad \le a_i \quad \text{if} \quad \gamma_i = -\alpha_i = 1 \end{cases} \tag{12}$$

where T_i and I_i represent the ith row of T and the ith component of I, respectively. It is clear that there will be no equilibrium points of (1) in $\overline{C(\gamma)}$ since (12) implies that $A^{-1}(T\gamma + I) \notin \overline{C(\gamma)}$, where $\overline{C(\gamma)}$ denotes the closure of $C(\gamma)$. Therefore, γ cannot be a stable memory vector of system (1). This conclusion is true for all $\gamma \in B^n$ such that $H(\alpha,\gamma) = 1$ if $T_{ii} \le a_i$ for $i = 1, 2, \cdots, n$.

2) Assume that $T_{ii} \le a_i, \gamma \in B^n, H(\alpha,\gamma) = 1$, and α and γ differ in the ith bit. If system (1) starts from $x(0) = \gamma$, one can obtain for the ith component of the state of system (1)

$$\begin{aligned} \dot{x}_i(0) &= -a_i x_i(0) + T_i \operatorname{sat}(x(0)) + I_i \\ &= -a_i\gamma_i + T_i\gamma + I_i \end{aligned} \tag{13}$$

In view of (12), (13) implies that

$$\dot{x}_i(0) > -a_i\gamma_i - a_i = 0 \quad \text{when} \quad \gamma_i = -1\ (\alpha_i = 1)$$

and

$$\dot{x}_i(0) < -a_i\gamma_i + a_i = 0 \quad \text{when} \quad \gamma_i = 1\ (\alpha_i = -1).$$

Clearly, immediately after $t = 0$, the ith component $x_i(t)$ of the state $x(t)$ will evolve in the direction toward α if $x(0) = \gamma$. ■

Remark 4.1: Results in Theorem 4.1 make it very likely that every corner of the hypercube which is in the immediate neighborhood (with Hamming distance 1) of a stable memory vector is in the domain of attraction of that stable memory vector. Theorem 4.1 also implies that none of the two desired patterns can have Hamming distance 1 if the diagonal elements of T are required to satisfy $T_{ii} \le a_i, i = 1, 2, \cdots, n$, where $a_i > 0$ is the ith diagonal element matrix A. ■

Remark 4.2: Clearly, it is desired in practice to design neural networks with $T_{ii} = a_i$ for $i = 1, 2, \cdots, n$ in order to take advantage of the results in Corollary 4.1 and Theorem 4.1, where a_i is the ith diagonal element of matrix A. The constraints on the diagonal elements of matrix T given by $T_{ii} = a_i$ for $i = 1, 2, \cdots, n$ will be referred to as the *optimal constraints* in the present paper. Therefore, under the optimal constraints,

- a neural network will have only bipolar stable memory vectors;
- every corner of the hypercube which is in the immediate neighborhood of a stable memory vector cannot become a stable memory and is very likely to be in the domain of attraction of that stable memory vector.

Experimental results show that neural networks (1) which satisfy the optimal constraints usually have less spurious memories and larger basins of attraction for desired memories than networks which do not satisfy the optimal constraints. ■

Remark 4.3: When $I = 0$ in system (1), one can show that $\alpha \in B^n$ is a stable memory if and only if $-\alpha$ is a stable memory. This implies that Part 1 of Theorem 4.1 will be true for all $\gamma \in B^n$ such that $H(\alpha,\gamma) = 1$ or $H(\alpha,\gamma) = n - 1$ if $I = 0$ in system (1). A similar result for a different model has been obtained in [27]. ■

Remark 4.4: The adaptive Ho–Kashyap recording techniques can lead to *optimal designs* in terms of the size of basin of attraction [8], [9]. As can be seen from the previous remarks, the present approach can also optimize the size of basin of attraction by employing constraints on the diagonal elements of the connection matrix. ■

Remark 4.5: From Synthesis Algorithm 3.1, one knows that $T_{ii} = w_i^i + a_i\mu_i$ where μ_i is chosen such that $\mu_i > 1$. When the w_i^i obtained from Synthesis Algorithm 3.1 satisfies that $w_i^i < 0$, one can choose $a_i > 0, T_{ii} = a_i$, and $\mu_i = 1 - w_i^i/a_i > 1$, or choose $a_i > 0$ and $T_{ii} < a_i$ such that $\mu_i = (T_{ii} - w_i^i)/a_i > 1$. It can be easily proved that a neural-network design with the ith diagonal element of the connection matrix T satisfying $T_{ii} \le a_i$ where $a_i > 0$ is the ith diagonal element of matrix A, can be obtained *if and only if* $w_i^i < 0$ from Synthesis Algorithm 3.1. ■

Remark 4.6: If the algorithm ends up with $w_i^i \ge 0$ for some i, then one can repeat the perceptron training with an initial weight W^i chosen to be

$$W^i = [w_1^i, \cdots, w_{i-1}^i, -\xi_i, w_{i+1}^i, \cdots, w_{n+1}^i]$$

with $\xi_i > 0$. Experiments show that this will usually end up with $w_i^i < 0$ if such a solution exists. ■

Most of the existing synthesis methods [6], [11], [14], [15], [17]–[23], [26], [28], [29], [33] cannot be applied if some or all of the diagonal elements of a connection matrix are constrained (e.g., $T_{ii} \le a_i$ for some $a_i > 0$) except the methods developed in [27] and [32] where optimization techniques and linear programming methods are used. The next result provides a necessary and sufficient condition for the existence of a neural-network design with diagonal elements of the connection matrix to be upper bounded.

Theorem 4.2: A neural network design with the ith diagonal element of T satisfying $T_{ii} \le a_i$, where $a_i > 0$ is the ith diagonal element of matrix A, can be achieved if and only if the desired patterns $\alpha^1, \alpha^2, \cdots, \alpha^m$ with the ith entry eliminated are linearly separable, where the two classes are determined according to the ith element of the desired patterns.

Proof: Sufficiency: From Remark 2.1, it is always possible to obtain perceptrons W^i from Synthesis Algorithm 3.1 such that

$$\begin{cases} W^i\overline{\alpha}^k > 0 & \text{if} \quad \alpha_i^k = 1 \\ W^i\overline{\alpha}^k < 0 & \text{if} \quad \alpha_i^k = -1 \end{cases} \tag{14}$$

and for $k = 1, 2, \cdots, m$ and for $i = 1, 2, \cdots, n$.

Assume that the desired patterns $\alpha^1, \alpha^2, \cdots, \alpha^m$ with the ith entry eliminated are linearly separable for some $i, 1 \le i \le n$, where the two classes are determined according to the ith element. One can now apply Synthesis Algorithm 3.1 to train the perceptron W^i without using the ith element of the desired patterns (training patterns). Equation (14) in this case implies that when $\alpha_i^k = 1$

$$w_1^i\alpha_1^k + \cdots + w_{i-1}^i\alpha_{i-1}^k + w_{i+1}^i\alpha_{i+1}^k + \cdots + w_{n+1}^i > 0 \tag{15}$$

and when $\alpha_i^k = -1$

$$w_1^i\alpha_1^k + \cdots + w_{i-1}^i\alpha_{i-1}^k + w_{i+1}^i\alpha_{i+1}^k + \cdots + w_{n+1}^i < 0 \tag{16}$$

for $k = 1, 2, \cdots, m$. Choose ε_i such that

$$0 < \varepsilon_i < \min_{1 \le k \le m} \left\{ \left| \sum_{j=1, j\ne i}^{n} w_j^i\alpha_j^k + w_{n+1}^i \right| \right\}. \tag{17}$$

Let $w_i^i = -\varepsilon_i$. Then, it is clear from (15)–(17) that, when $\alpha_i^k = 1$

$$\begin{aligned} \sum_{j=1}^{n} w_j^i\alpha_j^k + w_{n+1}^i &= \sum_{j=1, j\ne i}^{n} w_j^i\alpha_j^k + w_{n+1}^i + w_i^i\alpha_i^k \\ &= \left| \sum_{j=1, j\ne i}^{n} w_j^i\alpha_j^k + w_{n+1}^i \right| - \varepsilon_i > 0 \end{aligned}$$

and when $\alpha_i^k = -1$

$$\sum_{j=1}^{n} w_j^i\alpha_j^k + w_{n+1}^i = -\left| \sum_{j=1, j\ne i}^{n} w_j^i\alpha_j^k + w_{n+1}^i \right| + \varepsilon_i < 0.$$

This implies that one can always obtain W^i with $w_i^i < 0$ and hence, to obtain the ith diagonal element of matrix T satisfying $T_{ii} \le a_i$, where $a_i > 0$ is the ith diagonal element of matrix A (cf. Remark 4.5).

Necessity: If the ith diagonal element of T can be chosen such that $T_{ii} \le a_i$, where a_i is the ith diagonal element of A; i.e., if Synthesis Algorithm 3.1 ends up with $w_i^i < 0$, from (7), one obtains

$$\begin{aligned} w_1^i\alpha_1^k + \cdots + w_{i-1}^i\alpha_{i-1}^k + w_{i+1}^i\alpha_{i+1}^k + \cdots + w_{n+1}^i \\ \ge -w_i^i\alpha_i^k > 0 \quad \text{when} \quad \alpha_i^k = 1 \end{aligned}$$

and

$$\begin{aligned} w_1^i\alpha_1^k + \cdots + w_{i-1}^i\alpha_{i-1}^k + w_{i+1}^i\alpha_{i+1}^k + \cdots + w_{n+1}^i \\ < -w_i^i\alpha_i^k < 0 \quad \text{when} \quad \alpha_i^k = -1 \end{aligned}.$$

The above two expressions constitute a perceptron with the ith entry α_i^k of α^k deleted. Therefore, the desired patterns $\alpha^1, \alpha^2, \cdots, \alpha^m$ with the ith entry eliminated are linearly separable into two classes determined according to the ith element. ■

In order to use the result in Theorem 4.2, one has to train the perceptrons of Synthesis Algorithm 3.1 to see whether the training will converge since in general there are no simple criteria for testing linear separability of two classes of patterns. It turns out, however, that a simple criterion is possible for the present case in which all the training patterns are represented in a bipolar space. The next result provides a sufficient condition (simple criterion) for the existence of neural network design with the connection matrix T having upper bounds on the diagonal elements.

Theorem 4.3: Let

$$Y = [\alpha^1 \vdots \alpha^2 \vdots \cdots \vdots \alpha^m] \tag{18}$$

and $Y^i = [Y$ with the ith row deleted]. If

$$\text{rank}\,(Y) = \text{rank}\,(Y^i) \tag{19}$$

for some $i, 1 \le i \le n$, then Synthesis Algorithm 3.1 can lead to a neural-network design with the ith diagonal element of T satisfying $T_{ii} \le a_i$, where $a_i > 0$ is the ith diagonal element of matrix A.

Proof: If (19) is satisfied, the ith row of Y can be expressed as a linear combination of all the other rows of Y; i.e.,

$$y_i = \sum_{j=1, j\ne i}^{n} \lambda_j y_j$$

for some real numbers λ_j, where $y_j = [\alpha_j^1, \alpha_j^2, \cdots, \alpha_j^m]$ represents the jth row of Y. The componentwise expression of the above equation gives

$$\alpha_i^k = \sum_{j=1, j\ne i}^{n} \lambda_j\alpha_j^k \quad \text{for} \quad k = 1, 2, \cdots, m. \tag{20}$$

From Theorem 3.1, Synthesis Algorithm 3.1 will always find perceptrons with weight W^i for $i = 1, 2, \cdots, n$. Consider the

ith perceptron W^i for which (7) is true and for which $w_i^i \geq 0$. From (20), one can obtain

$$\begin{aligned} W^i\overline{\alpha}^k &= \sum_{j=1,j\neq i}^{n} w_j^i \alpha_j^k + (w_i^i + \varepsilon_i)\alpha_i^k - \varepsilon_i \alpha_i^k + w_{n+1}^i \\ &= \sum_{j=1,j\neq i}^{n} w_j^i \alpha_j^k + (w_i^i + \varepsilon_i) \sum_{j=1,j\neq i}^{n} \lambda_j \alpha_j^k - \varepsilon_i \alpha_i^k \\ &\quad + w_{n+1}^i \end{aligned}$$

where $\varepsilon_i > 0$. One can choose a new weight vector $\underline{W}^i = [\underline{w}_1^i, \underline{w}_2^i, \cdots, \underline{w}_{n+1}^i]$ with $\underline{w}_j^i = w_j^i + \lambda_j(w_i^i + \varepsilon_i)$ for $i \neq j$ and $\underline{w}_i^i = -\varepsilon_i < 0$. Since $\underline{W}^i\overline{\alpha}^k = W^i\overline{\alpha}^k$, one can see that (7) is also satisfied for $\underline{W}^i$ by assumption. Based on $\underline{W}^i$, it is clear that diagonal elements of the connection matrix T satisfying $T_{ii} \leq a_i$, where $a_i > 0$ is the ith diagonal element of matrix A, can be achieved since $\underline{w}_i^i < 0$ (cf. Remark 4.5). ■

Remark 4.7: It is clear from Theorem 4.3 that if $\text{rank}(Y) = n$, then condition (19) in the theorem can never be satisfied. That is to say, if one wishes to design a neural network (1) with some prespecified constraints on the diagonal elements of the connection matrix T so that less spurious memories and bigger basins of attraction for desired memories may be achieved, it is desirable that the prototype patterns satisfy that $\text{rank}(Y) < n$, where Y is defined in (18). ■

V. Neural Networks with Sparsity and Symmetry Constraints on the Interconnecting Structure

Two connectivity constraints will be discussed in this section, namely the *sparsity* and *symmetry* constraints.

Sparsity constraints on the interconnecting structure of (1), in general, can be expressed as prespecified zero elements in the connection matrix T at given locations [19], [24]. A matrix $S = [S_{ij}] \in \Re^{n\times n}$ is said to be an *index matrix,* if it satisfies $S_{ij} = 1$ or 0. The restriction of matrix $F = [f_{ij}] \in \Re^{n\times n}$ to an index matrix S, denoted by $F|S$, is defined by $F|S = [h_{ij}]$, where

$$h_{ij} = \begin{cases} f_{ij}, & \text{if } S_{ij} = 1 \\ 0, & \text{otherwise.} \end{cases}$$

System (1) is said to be a *neural network with a sparse coefficient matrix* if $T = T|S$ for some given index matrix $S \in \Re^{n\times n}$. Note that the index matrix S specifies the interconnecting structure of a feedback neural network. It is also noted that by special choice of the index matrix S, neural network (1) will have the same structure as that of cellular neural neworks [3], [19].

The following sparse design problem has been considered in [17]–[20].

Sparse Design Problem: Given an $n \times n$ index matrix $S = [S_{ij}]$ with $S_{ii} = 1$ for $i = 1, \cdots, n$, and m vectors $\alpha^1, \cdots, \alpha^m$ in B^n, choose $\{A, T, I\}$ with $T = T|S$ in such a manner that $\alpha^1, \cdots, \alpha^m$ are memory vectors of system (1). ■

Synthesis Algorithm 3.1 developed in Section III can be modified to solve the sparse design problem.

Sparse Design Algorithm 5.1: Using the perceptron training algorithm, obtain n perceptions $W^i = [w_1^i, w_2^i, \cdots, w_{n+1}^i], i = 1, 2, \cdots, n$, where w_j^i is preset to zero if $S_{ij} = 0$, such that

$$W^i\overline{\alpha}^k \geq 0 \quad \text{if} \quad \alpha_i^k = 1$$

and

$$W^i\overline{\alpha}^k < 0 \quad \text{if} \quad \alpha_i^k = -1$$

for $k = 1, 2, \cdots, m$, where $\overline{\alpha}^k$ is defined as before.

A, T and I will be chosen in the same way as in Synthesis Algorithm 3.1. ■

Sparse Design Algorithm 5.1 is a modification of Synthesis Algorithm 3.1, in which one needs to preset $w_j^i = 0$ if $S_{ij} = 0$. The convergence of the perceptron training in the sparse design algorithm can also be proved following similar lines as the proof of Theorem 3.1 (considering $S_{ii} = 1$ for $i = 1, 2, \cdots, n$).

If one wants to achieve a sparse neural-network design with constraints on the diagonal components of the connection matrix T, the following results provide a necessary and sufficient condition and a simple criterion for the existence of such a design. Their proofs follow similar lines as the proofs of Theorems 4.2 and 4.3.

Corollary 5.1: Suppose that in the index matrix $S, S_{ii} = 1$ for $i = 1, 2, \cdots, n$. A sparse neural-network design with the ith diagonal element of T satisfying $T_{ii} \leq a_i$, where $a_i > 0$ is the ith diagonal element of matrix A, can be achieved if and only if the patterns $\alpha^1|S_i^T, \alpha^2|S_i^T, \cdots, \alpha^m|S_i^T$ with the ith entry eliminated are linearly separable; where S_i is the ith row of the index matrix S and the two classes are determined according to the ith element of the desired patterns. ■

Corollary 5.2: Suppose that in the index matrix $S, S_{ii} = 1$ for $i = 1, 2, \cdots, n$. Let S_i be the ith row of the index matrix $S, Q = [\alpha^1|S_i^T \vdots \alpha^2|S_i^T \vdots \cdots \vdots \alpha^m|S_i^T]$, and $Q^i = [Q$ with the ith row deleted]. If

$$\text{rank}(Q) = \text{rank}(Q^i)$$

for some $i, 1 \leq i \leq n$, then Sparse Design Algorithm 5.1 can lead to a sparse design with the ith diagonal element of T satisfying $T_{ii} \leq a_i$, where $a_i > 0$ is the ith diagonal element of matrix A. ■

Parameter perturbations (e.g., due to implementation errors) can be represented by $\Delta A = \text{diag}[\Delta a_1, \cdots, \Delta a_n], \Delta T \in \Re^{n\times n}$ and $\Delta I \in \Re^n$. System (1) with parameter perturbations can be written as

$$\dot{x} = -(A + \Delta A) + (T + \Delta T)\,\text{sat}(x) + (I + \Delta I) \qquad (21)$$

where A, T, I, and $\text{sat}(\cdot)$ are defined as in (1).

For the perturbation model described by (21) the following robustness analysis result has been established in [17] and [20]. The notation $\delta(x) = \min_{1\leq i\leq n} \{|x_i|\}$ for $x \in \Re^n$ will be used in the sequel.

Lemma 5.1: Suppose that $\alpha^1, \cdots, \alpha^m \in B^n$ are desired stable memory vectors of system (1), and suppose that $\beta^1, \cdots, \beta^m$ are asymptotically stable equilibrium points of system (1) corresponding to $\alpha^1, \cdots, \alpha^m$, respectively; i.e.,

$$\beta^k = A^{-1}(T\alpha^k + I) \quad \text{for} \quad k = 1, \cdots, m. \tag{22}$$

Let

$$\mu = \min_{1 \le k \le m} \{\delta(\beta^k)\} > 1. \tag{23}$$

Then $\alpha^1, \cdots, \alpha^m$ are also stable memory vectors of system (21) provided that

$$||A^{-1}\Delta A||_\infty + ||A^{-1}\Delta T||_\infty + ||A^{-1}\Delta I||_\infty < \mu - 1 \tag{24}$$

where $||\cdot||_\infty$ denotes the matrix norm induced by the l_∞ vector norm. ■

Lemma 5.1 implies that the asymptotically stable equilibrium points $\overline{\beta}^k$ corresponding to α^k after parameter perturbations satisfy

$$\overline{\beta}^k = (A + \Delta A)^{-1}((T + \Delta T)\alpha^k + (I + \Delta I)) \in C(\alpha^k)$$

for $k = 1, 2, \cdots, m$, provided that (24) is satisfied. Recall that the matrix norm induced by the l_∞ vector norm for a matrix $F = [f_{ij}] \in \Re^{m \times n}$ is defined by

$$||F||_\infty = \max_{1 \le i \le m} \left\{ \sum_{j=1}^{n} |f_{ij}| \right\}.$$

It is clear from Lemma 5.1 that Synthesis Algorithm 3.1 and Sparse Design Algorithm 5.1 guarantee that $\alpha^1, \cdots, \alpha^m$ are also stable memory vectors of (21) provided that (24) is satisfied, where μ is given in (23) and β^k's are given in (22) Note that Synthesis Algorithm 3.1 and Sparse Design Algorithm 5.1 guarantee that

$$\mu = \min_{1 \le k \le m, 1 \le i \le n} \{e_i^k\} > 1 \tag{25}$$

according to Remark 3.2, where $e_i^k \ge \mu_i$. Also, note that the procedure described in Remark 4.6 for obtaining negative w_i^i can be utilized repeatedly with large ξ_i to arrive at a perceptron which allows to choose $T_{ii} \le a_i$ and to choose large μ_i in Synthesis Algorithm 3.1 and Sparse Design Algorithm 5.1.

The above enables one to achieve *an upper bound* for the parameter inaccuracies encountered in the implementation of a given design to store a desired set of bipolar patterns in system (1). This bound can be controlled by the designer in the design procedure. Specifically, the synthesis algorithms advocated above incorporates two features which are very important in the VLSI implementation of artificial neural networks.

1) It allows the designer to choose a suitable interconnecting structure for the neural network.
2) It takes into account inaccuracies which arise in the realization of the neural network by hardware.

For the T and I determined by Synthesis Algorithm 3.1 with $\mu_i > 1$, choose $\Delta T = (T^T - T)/2$. Then $T_s \triangleq T + \Delta T = (T + T^T)/2$ is a symmetric matrix. From Lemma 5.1, one notes that if $||A^{-1}\Delta T||_\infty = ||A^{-1}(T^T - T)||_\infty/2 < \mu - 1$, where μ is obtained in (23) and where it is assumed that $\Delta A = 0$ and $\Delta I = 0$, the neural network (1) will also store all the desired patterns as memories, with a symmetric connection matrix $T + \Delta T = T_s$.

The above observation gives rise to the possibility of designing a neural network (1) with *prespecified interconnecting structure* and *with a symmetric interconnection matrix*. (Note that in this case, it is required that $S = S^T$.) Such a capability is of *great* interest since the neural network (1) will be *globally stable* when T is symmetric [3], [19]. (Global stability means that for every initial state, the network will converge to some asymptotically stable equilibrium point and periodic and chaotic solutions do not exist.) Sparse Design Algorithm 5.1 presented above will usually result in a nonsymmetric matrix T. The symmetric design procedure developed in [17] can be used to determine a *symmetric* connection matrix T_s which also satisfies that $T_s|S = T_s$ for $S = S^T$. For the sake of completeness, the symmetric design procedure is summarized below.

Symmetric Design Algorithm 5.2: Given m vectors $\alpha^1, \cdots, \alpha^m$ in B^n which are to be stored as stable memory vectors for system (1) and an index matrix $S = S^T$ with $S_{ii} = 1$ for $i = 1, 2, \cdots, n$, one can proceed as follows to get a symmetric design.

1) According to Sparse Design Algorithm 5.1, determine $A, T = T|S$ and I for system (1) with $\mu_i > 1$.
2) Compute $\beta^k = A^{-1}(T\alpha^k + I)$ for $k = 1, 2, \cdots, m$ and let $\mu = \min_{1 \le k \le m} \{\delta(\beta^k)\}$.
3) If $T = T^T$ or $\mu \le 1 + \eta$, where η is a small positive number (e.g., $\eta = 0.01$), stop. Otherwise, go to Step 4).
4) Compute $\Delta T = (T^T - T)/2$. If $||A^{-1}\Delta T||_\infty < \mu - 1$, choose $\lambda = 1$. Otherwise, choose
$$\lambda = \frac{\mu - 1}{||A^{-1}\Delta T||_\infty} - \varepsilon$$
where ε is a small positive number (e.g., $\varepsilon = 0.001$). Compute $\overline{T} = T + \lambda \Delta T$.
5) Compute $\overline{\beta}^k = A^{-1}(\overline{T}\alpha^k + I)$ for $k = 1, 2, \cdots, m$, and compute
$$\nu = \min_{1 \le k \le m} \{\delta(\overline{\beta}^k)\} > 1.$$
6) Replacing μ by ν and replacing T by $\overline{T}$, go to Step 3).

If one ends up with $T = T^T$, a solution has been found for the symmetric design problem. If one ends up with $\mu < 1 + \eta$ and $T \ne T^T$, the symmetric design procedure is not successful in solving a symmetric T for the given problem. ■

VI. Applications to Other Neural-Network Models

The analog Hopfield neural-network model [12] is described by

$$\begin{aligned} \dot{x} &= -Ax + Tg(x) + I \\ y &= g(x) = [g_1(x_1), g_2(x_2), \cdots, g_n(x_n)]^T \end{aligned} \tag{26}$$

where g_i: $R \to (-1, 1)$ is called the activation function and is continuously differentiable, strictly monotonically increasing in x_i, and $x_i g(x_i) > 0$ for all $x_i \neq 0$. The neural network (1) considered in the previous sections is a variation of neural networks described by (26).

In addition to (1) and (26), there are many other feedback neural-network models in the literature which have been used for the realization of associative memories. For example, neural networks described by

$$x(k+1) = \mathrm{sign}\,(Tx(k) + I) \tag{27}$$

are considered in [11], [22], [28], and [29], where $\mathrm{sign}\,(\cdot)$ is the signum function. Also, neural networks described by

$$\dot{x} = h(Tx + I) \tag{28}$$

are investigated in [15], where $h(\cdot)$ represents the continuous-time saturation function (for notation, see [16]). Moreover, neural networks described by

$$x(k+1) = \mathrm{sat}\,(Tx(k) + I) \tag{29}$$

are considered in [23]. In the literature, (27) is called the *discrete-time Hopfield model*, and (28) and (29) are called *neural networks defined on hypercubes*.

The synthesis algorithms as well as the analysis results established in the present paper can also be applied to all of the above neural network models. In the following, assume that $\alpha \in B^n$ is a desired memory pattern to be stored in a network. Similar conditions to (3) will be derived for each of the above models so that the applicability of the synthesis approach developed in the previous sections becomes evident. For each of the above models, modifications to the robustness analysis result (cf. Lemma 5.1) and optimal constraints for the diagonal elements of the connection matrix (cf. Remark 4.2) will also be discussed briefly.

Neural Network (26): For (26), assume that there exist $\lambda_i > 0$ such that

$$g_i(x_i) \approx 1 \quad \text{when} \quad x_i > \lambda_i$$

and

$$g_i(x_i) \approx -1 \quad \text{when} \quad x_i < -\lambda_i$$

for $i = 1, 2, \cdots, n$. Without loss of generality, one can assume that $\lambda_i \geq 1$ for $i = 1, 2, \cdots, n$. This is possible if one chooses

$$g_i(x_i) = \frac{2}{\pi} \arctan\,(x_i)$$

or

$$g_i(x_i) = \mathrm{arctanh}\,(x_i) = \frac{e^{x_i} - e^{-x_i}}{e^{x_i} + e^{-x_i}}.$$

In this case, the design objective has to be modified to approximately store a set of desired bipolar patterns using (26) due to the nature of functions $g_i(\cdot)$. Assume that $\alpha \in B^n$ is such a desired memory pattern to be stored in (26). Denote $\Lambda = \mathrm{diag}\,[\lambda_1, \lambda_2, \cdots, \lambda_n]$. For $x \in \Re^n$ such that $\Lambda^{-1}x \in C(\alpha)$, (26) can be written as

$$\dot{x} \approx -Ax + T\alpha + I \tag{30}$$

since $g(x) \approx \alpha$ when $\Lambda^{-1}x \in C(\alpha)$. The unique equilibrium point of (30), $\beta = A^{-1}(T\alpha + I)$, is asymptotically stable [cf. (4)] and satisfies $g(\beta) \approx \alpha$ whenever $\Lambda^{-1}\beta \in C(\alpha)$. Thus, stable memories must satisfy the condition that

$$\Lambda^{-1}A^{-1}(T\alpha + I) \in C(\alpha). \tag{31}$$

All the results established in the previous sections can easily be modified according to (31). In particular, if one chooses $A = \Lambda^{-1}$, (31) becomes (3).

The robustness result in Lemma 5.1 needs to be modified for the present case by replacing (24) by $||A^{-1}\Delta A||_\infty + ||A^{-1}\Delta T||_\infty + ||A^{-1}\Delta I||_\infty < \nu$ where

$$\nu = \min_{1 \leq k \leq m} \{\delta(\beta^k - [\lambda_1, \lambda_2, \cdots, \lambda_n]^T)\} > 0$$

and β^k's are given in (22). Due to the nature of functions $g_i(\cdot)$, optimal constraints cannot be easily identified in general for the diagonal elements of matrix T in system (26). However, constraints given by $a_i \leq T_{ii} \leq \lambda_i a_i$ for $i = 1, 2, \cdots, n$ can be considered to be nearly optimal in this case.

Neural Network (27): One needs (cf. [11], [22], [28], and [29]) $\alpha = \mathrm{sign}\,(T\alpha + I)$, which is equivalent to

$$T\alpha + I \in E(\alpha) \tag{32}$$

where $E(\alpha) = \{x \in \Re^n\colon\ x_i\alpha_i > 0, i = 1, \cdots, n\}$. Similar synthesis algorithms to the ones developed in the present paper using perceptron training can be formulated based on (32). For example, Synthesis Algorithm 3.1 can be modified for (27) as follows.

Synthesis Algorithm 6.1: Suppose that $\alpha^1, \alpha^2, \cdots, \alpha^m$ are given vectors in B^n which are to be stored as memory vectors for system (27). Using the perceptron training algorithm, obtain n perceptrons $W^i = [w_1^i, w_2^i, \cdots, w_{n+1}^i], i = 1, 2, \cdots, n,$ such that

$$W^i \overline{\alpha}^k \geq 0 \quad \text{if} \quad \alpha_i^k = 1$$

and

$$W^i \overline{\alpha}^k < 0 \quad \text{if} \quad \alpha_i^k = -1$$

for $k = 1, 2, \cdots, m$, where $\overline{\alpha}^k$ is defined as before.

For $i, j = 1, 2, \cdots, n$, choose $T_{ij} = w_j^i$ if $i \neq j$ and $T_{ii} = w_i^i + \mu_i$ with $\mu_i > 0$, and choose $I_i = w_{n+1}^i$. ■

For the robustness analysis result in Lemma 5.1, (24) has to be replaced by

$$||\Delta T||_\infty + ||\Delta I||_\infty < \nu \tag{33}$$

where

$$\nu = \min_{1 \leq k \leq m} \{\delta(\beta^k)\} > 0$$

and $\beta^k = T\alpha^k + I$ for $k = 1, 2, \cdots, m$. The optimal constraints are given by $T_{ii} \leq 0$ for $i = 1, 2, \cdots, n$ in this case.

Neural Network (28): One needs (cf. [15]) $(T_i\alpha + I_i)\alpha_i > 0$ for $i = 1, 2, \cdots, n$, which is equivalent to (32). In this case, (33) applies and the optimal constraints are given by $T_{ii} = 0$ for $i = 1, 2, \cdots, n$.

Neural Network (29): It is required that (cf. [23]) $(T_i\alpha + I_i)\alpha_i > 1$ for $i = 1, 2, \cdots, n$, which is equivalent to (3). In this case, condition (24) in Lemma 5.1 must be replaced by

$$||\Delta T||_\infty + ||\Delta I||_\infty < \mu - 1$$

where μ is given by (22) with $\beta^k = T\alpha^k + I$ for $k = 1, 2, \cdots, m$ and the optimal constraints are given by $T_{ii} = 1$ for $i = 1, 2, \cdots, n$.

Remark 6.1: Neural network (29) can also be written as

$$x(k+1) = \text{sat}\,(x(k) + T'x(k) + I) \tag{34}$$

which is called the brain-state-in-a-box (BSB) model [7], [13], [27] in the literature. The design problem for the BSB model studied in the literature is for the design of matrix T' and vector I. It is noted that the zero diagonal design problem studied in [27] for the BSB model concerns the zero diagonal design of matrix T' in (34) which corresponds to the design with optimal constraints of the present case (cf. [27], where $I = 0$ in (34) is considered). It should be noted that the zero diagonal design problem considered in [27] for the BSB model is solved using a different approach from the present one. ■

VII. Examples

To demonstrate the applicability and versatility of the analysis and synthesis results presented in the preceding sections and to compare with existing results, four specific examples will be considered. All simulations for the present paper are performed on a Sun SPARC Station using MATLAB.

Example 7.1: A neural network with 12 neurons $(n = 12)$ is considered with the objective of storing the 12 $(m = 12)$ patterns $\alpha^1, \alpha^2, \cdots, \alpha^{12}$ shown in Fig. 1 as memories. As indicated in this figure, 12 boxes are used to represent each pattern (in R^{12}), with each box corresponding to a vector component which is allowed to assume values between -1 and one. For purpose of visualization, -1 will represent white, one will represent black, and the intermediate values will correspond to appropriate grey levels. In this example, a neural network (1) with A to be the identity matrix and satisfying the optimal constraints, i.e., with every diagonal element of T equal to one $(T_{ii} = a_i = 1)$, will be designed.

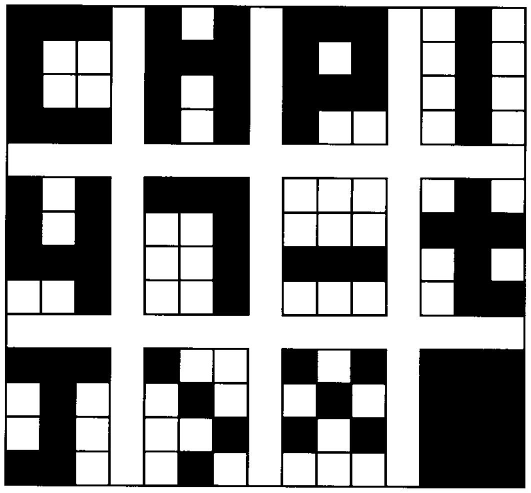
Fig. 1. The 12 desired memory patterns.

The objective is to utilize Synthesis Algorithm 3.1 to design a nonsymmetric neural network with all the diagonal elements to be one. In Synthesis Algorithm 3.1, the initial weight W^i is chosen to be the zero vector and the perceptron learning rate is chosen to be $\eta = 0.1$. It turns out that the desired patterns does not satisfy the rank condition of Theorem 4.3 when $i = 2, 5, 8, 10, 11, 12$. Still, all the w_i^i obtained from Synthesis Algorithm 3.1 with Remark 4.6 are less than zero. Therefore, a connection matrix T with every diagonal element to be one can be obtained. A is obtained as the identity matrix, T is obtained as shown in the matrix at the bottom of the page, and I is obtained as

$$I = [6, 5.5, 2.5, -0.7, 12.8, -2.8, 2.3, 17.6, 4, -9.3, 2.7, -1.5]^T.$$

Using Lemma 3.1, one can determine that neural network (1) with the $\{A, T, I\}$ obtained as above has eight spurious memory points in B^n. From Corollary 4.1, one can see that these eight vectors are the only spurious memory vectors in D^n in this case since $T_{ii} \geq a_i$ for $i = 1, 2, \cdots, n$ are satisfied. To see whether the neural network in this case has other stable memories in D^n, 5000 simulation runs with randomly chosen

$$T = \begin{bmatrix} 1 & -1.2 & 9.2 & 0.4 & 2.2 & 0 & -1 & -4.4 & 4.6 & 2.4 & 2.2 & 0.2 \\ -0.3 & 1 & 5.9 & -3.3 & -4.9 & 5.5 & -9.3 & 3.9 & -6.7 & 7.7 & 6.3 & -0.9 \\ 8.9 & 3.9 & 1 & -0.1 & 0.3 & 0.9 & 4.5 & -0.5 & -0.9 & 5.7 & -5.1 & 1.3 \\ -1.5 & 0.5 & -0.9 & 1 & -0.3 & 6.7 & 7.1 & 3.3 & -2.3 & 4.7 & 2.1 & 5.3 \\ 1.8 & -19 & 0.4 & -2.6 & 1 & 11 & -10.2 & -6.2 & 4.8 & 4.6 & 24 & -10.6 \\ 0 & 2.8 & 1 & 6 & 2 & 1 & -4.6 & 3.6 & 5.2 & -0.6 & -5.2 & 5 \\ -0.3 & -5.7 & 3.5 & 7.3 & -2.3 & -3.9 & 1 & 2.1 & 4.1 & 4.1 & -3.7 & -0.1 \\ -15.4 & 7.8 & 4.6 & 8.6 & -1.6 & 20 & 6 & 1 & -2.6 & -5 & 7.8 & -23.4 \\ 6.2 & -5.8 & 0.6 & -1.4 & 2.4 & 6 & 4.6 & 0 & 1 & -2.8 & -4.2 & -1 \\ 5.3 & 5.5 & 7.3 & 7.7 & 3.1 & 0.9 & 4.3 & -3.5 & -4.5 & 1 & 4.3 & -2.3 \\ 2.9 & 6.1 & -4.3 & 1.7 & 6.3 & -5.1 & -4.1 & 1.9 & -3.7 & 2.3 & 1 & 1.7 \\ -1.5 & -0.7 & 5.1 & 7.3 & 0.7 & 10.5 & -1.3 & -6.5 & -4.7 & -6.9 & 6.7 & 1 \end{bmatrix}$$

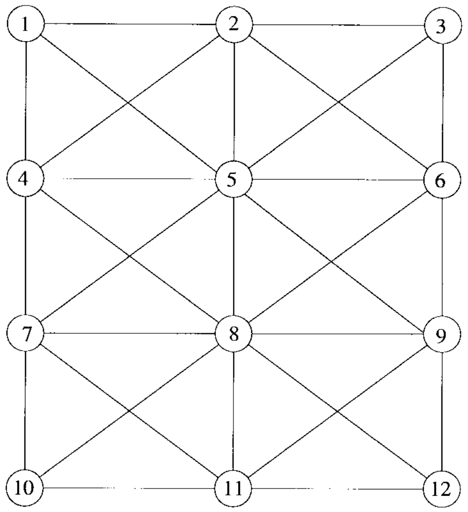

Fig. 2. Interconnecting structure of a cellular neural network.

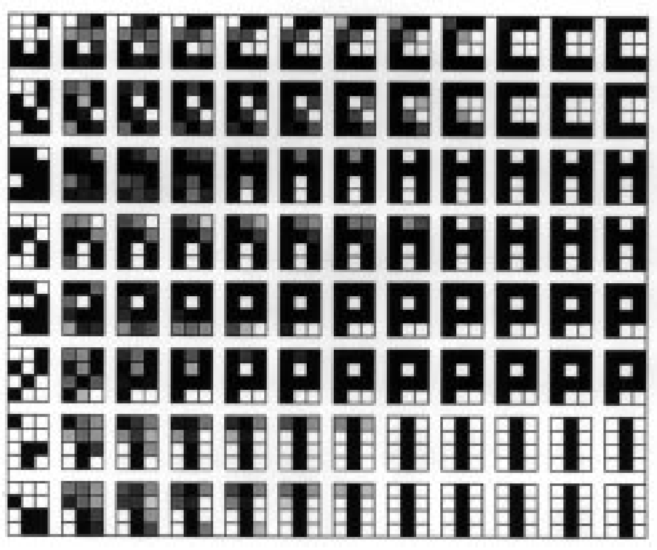
Fig. 3. Typical evolutions of the four patterns in Example 7.2.

initial states in D^n are performed. From these simulation runs, no more spurious memory vectors in D^n are discovered. Simulation results also show that every $\gamma \in B^n$ such that $H(\alpha^k, \gamma) = 1$ for some $k, 1 \le k \le 12$, is indeed in the domain of attraction of α^k. Using Lemma 5.1, one can determine as in (23) $\mu = 7$, which implies that the allowable upper bound for parameter perturbations is given by

$$\begin{aligned}&||A^{-1}\Delta A||_\infty + ||A^{-1}\Delta T||_\infty + ||A^{-1}\Delta I||_\infty \\ &\quad = ||\Delta A||_\infty + ||\Delta T||_\infty + ||\Delta I||_\infty < \mu - 1 = 6. \quad \blacksquare\end{aligned}$$

Example 7.2: In this example, the design of neural network with the configuration given in Fig. 2 is considered with the objective to store the first four patterns $(\alpha^1, \alpha^2, \alpha^3, \alpha^4)$ in Fig. 1 as memories. In both cases, a neural network (1) with A to be the identity matrix and satisfying the optimal constraints, i.e., with every diagonal element of T equal to one ($T_{ii} = a_i = 1$), will be part of the design objectives.

Case I—Sparse Design: The index matrix for this interconnecting structure is as follows, where "0" represents no connection and "1" represents a connection

$$S = S^T = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

Note that neural network with structure in Fig. 2 is also called a cellular neural network [3].

Sparse Design Algorithm 5.1 is utilized with Remark 4.6. After each perceptron training, choose $T_{ii} = 1$ and $\mu_i = 1 - w_i^i$ whenever $w_i^i < 0$. Matrix T is obtained as shown at the bottom of the page, and bias vector I is obtained as

$$I = [3.2, 6.4, 6.1, 2.7, 6.5, 0, 2.4, 2.1, -0.1, 3.5, 1.3, 0]^T.$$

$$T = \begin{bmatrix} 1 & -2.2 & 0 & 8.6 & -3.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0.6 & 1 & -0.6 & -0.6 & -6.2 & -4.4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1.1 & 1 & 0 & -6.1 & 4.9 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5.7 & -0.3 & 0 & 1 & -2.7 & 0 & 5.7 & -2.5 & 0 & 0 & 0 & 0 \\ -2.1 & -4.7 & -2.1 & -2.1 & 1 & 0.9 & -2.1 & -1.7 & 0.9 & 0 & 0 & 0 \\ 0 & -3.2 & 3.2 & 0 & 0 & 1 & 0 & 2.2 & 8.6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4.8 & -2.4 & 0 & 1 & -2.4 & 0 & 4.8 & -2.4 & 0 \\ 0 & 0 & 0 & -1.7 & -2.1 & 1.1 & -1.7 & 1 & 1.1 & -1.7 & -1.1 & -8.7 \\ 0 & 0 & 0 & 0 & 0.1 & 6.5 & 0 & 0.1 & 1 & 0 & -6.5 & -0.1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 7.9 & -3.7 & 0 & 1 & -3.5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1.9 & -1.3 & -9.7 & -1.9 & 1 & 1.3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -9.8 & 0 & 0 & 0 & 1 \end{bmatrix}$$

TABLE I

	Number of spurious memories in B^n	Number of spurious memories in $D^n \backslash B^n$
$T_{ii} = 4$	4084 ($\mu = 3.6$), 357 ($\mu = 4$), 64 ($\mu = 7$) 20 ($\mu = 14$), 16 ($\mu = 28$), 12 ($\mu = 56$)	0
$T_{ii} = 3$	4084 ($\mu = 2.6$), 95 ($\mu = 4$), 32 ($\mu = 7$) 16 ($\mu = 14$), 12 ($\mu = 28$), 10 ($\mu = 56$)	0
$T_{ii} = 2$	3876 ($\mu = 1.6$), 30 ($\mu = 4$), 18 ($\mu = 7$) 12 ($\mu = 14$), 11 ($\mu = 28$), 9 ($\mu = 56$)	0
$T_{ii} = 1$ (optimal constraints)	6 ($\mu = 1.2$), 9 ($\mu = 4$), 8 ($\mu = 7$) 8 ($\mu = 14$), 9 ($\mu = 28$), 9 ($\mu = 56$)	0
$T_{ii} = 0$	3 ($\mu = 4$) , 5 ($\mu = 14$)	≥ 6 ($\mu = 4$), ≥ 4 ($\mu = 14$)
$T_{ii} = -1$	0 ($\mu = 4$), 3 ($\mu = 14$)	≥ 10 ($\mu = 4$), ≥ 8 ($\mu = 14$)
$T_{ii} = -2$	0 ($\mu = 4$), 2 ($\mu = 14$)	≥ 10 ($\mu = 4$), ≥ 8 ($\mu = 14$)
$T_{ii} = -3$	0 ($\mu = 4$), 1 ($\mu = 14$)	≥ 10 ($\mu = 4$), ≥ 11 ($\mu = 14$)
eigenstructure method [15], [23]	1382 ($\mu = 1.6$), 874 ($\mu = 4$) 874 ($\mu = 7$), 874 ($\mu = 14$) 874 ($\mu = 28$), 874 ($\mu = 56$)	0
method of [32] with $T_{ii} = 1$	15 ($\mu = 1.7$), 17 ($\mu = 4$) 14 ($\mu = 7$), 13 ($\mu = 14$) 12 ($\mu = 28$), 13 ($\mu = 56$)	0
method of [32] with $T_{ii} = 3$	644 ($\mu = 4$), 183 ($\mu = 5$) 69 ($\mu = 7$), 24 ($\mu = 14$) 18 ($\mu = 28$), 17 ($\mu = 56$)	0

In the above, the same η $(= 0.1)$ as in Example 7.1 is used. Even though the rank condition in Theorem 4.3 is not satisfied for $i = 2, 3, 5$, it is still possible to obtain a neural-network design with every diagonal element of T equal to one using Sparse Design Algorithm 5.1 with Remark 4.6. Using Lemma 3.1, one can determine that system (1) in this case has total of eight spurious memories in D^n and from Lemma 5.1, one can determine that $\mu = 7$.

Case II—Sparse and Symmetric Design: Starting from T and I obtained in Case I, one can obtain a neural network with symmetric interconnection matrix T for the same network structure using Symmetric Design Algorithm 5.2. T is obtained as shown at the bottom of the page, and I is the same as in the previous case. The network in this case has eight spurious memories and one can determine that $\mu = 3.9$.

The performance of this network is illustrated by means of typical simulation runs of (1), shown in Fig. 3. In this figure, the desired memory pattern is depicted in the last column. The initial states, shown in the first column, is generated by randomly reversing 4 to 5 b of each desired pattern. The iteration of the simulation evolves from left to right. The desired pattern is recovered in less than 11 steps in all cases with a step size $h = 0.06$ in the simulation of (1). ■

Example 7.3: The objective of this example is to study the effects of the diagonal elements of connection matrix T on the spurious memories in a neural network, and to compare the present synthesis approach with the eigenstructure method [15], [23] and the method of [32]. The same example as in Example 7.1 will be used. As seen in Example 7.1, under the optimal constraints ($T_{ii} = a_i = 1$ for $i = 1, 2, \cdots, n$) with $\mu = 7$, the network has total of eight spurious memories in D^n. Table I shows the results of neural network designed using Synthesis Algorithm 3.1 which stores the $m = 12$ patterns in Fig. 1 as stable memories. Different constraints for the diagonal elements of matrix T given by $T_{ii} = -3, -2, -1, 0, 1, 2, 3, 4$ are considered in the design examples (cf. Remark 4.5 for $T_{ii} < a_i$).

The value of μ is also computed as in (22) for each case considered. From Table I, one can see that the total number of spurious memories in a network which satisfies the

$$T = \begin{bmatrix}
1 & -1.4 & 0 & 7.15 & -2.65 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1.4 & 1 & -0.85 & -0.45 & -5.45 & -3.8 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -0.85 & 1 & 0 & -4.1 & 4.05 & 0 & 0 & 0 & 0 & 0 & 0 \\
7.15 & -0.45 & 0 & 1 & -2.4 & 0 & 5.25 & -2.1 & 0 & 0 & 0 & 0 \\
-2.65 & -5.45 & -4.1 & -2.4 & 1 & 0.45 & -2.25 & -1.9 & 0.5 & 0 & 0 & 0 \\
0 & -3.8 & 4.05 & 0 & 0.45 & 1 & 0 & 1.65 & 7.55 & 0 & 0 & 0 \\
0 & 0 & 0 & 5.25 & -2.25 & 0 & 1 & -2.05 & 0 & 6.35 & -2.15 & 0 \\
0 & 0 & 0 & -2.1 & -1.9 & 1.65 & -2.05 & 1 & 0.6 & -2.7 & -1.2 & -9.25 \\
0 & 0 & 0 & 0 & 0.5 & 7.55 & 0 & 0.6 & 1 & 0 & -8.1 & -0.05 \\
0 & 0 & 0 & 0 & 0 & 0 & 6.35 & -2.7 & 0 & 1 & -2.7 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -2.15 & -1.2 & -8.1 & -2.7 & 1 & 0.65 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -9.25 & -0.05 & 0 & 0.65 & 1
\end{bmatrix}$$

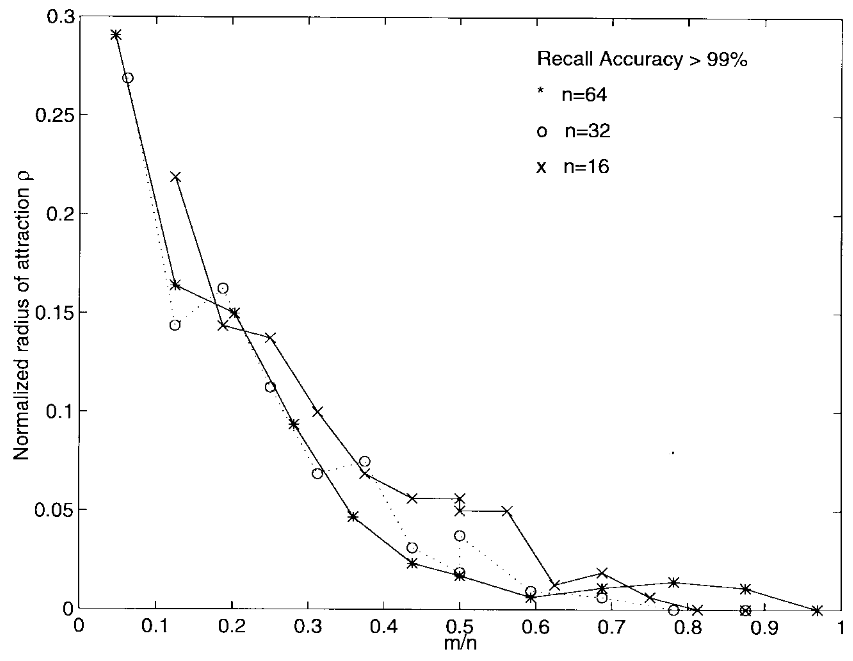

Fig. 4. Capacity performance for Synthesis Algorithm 3.1.

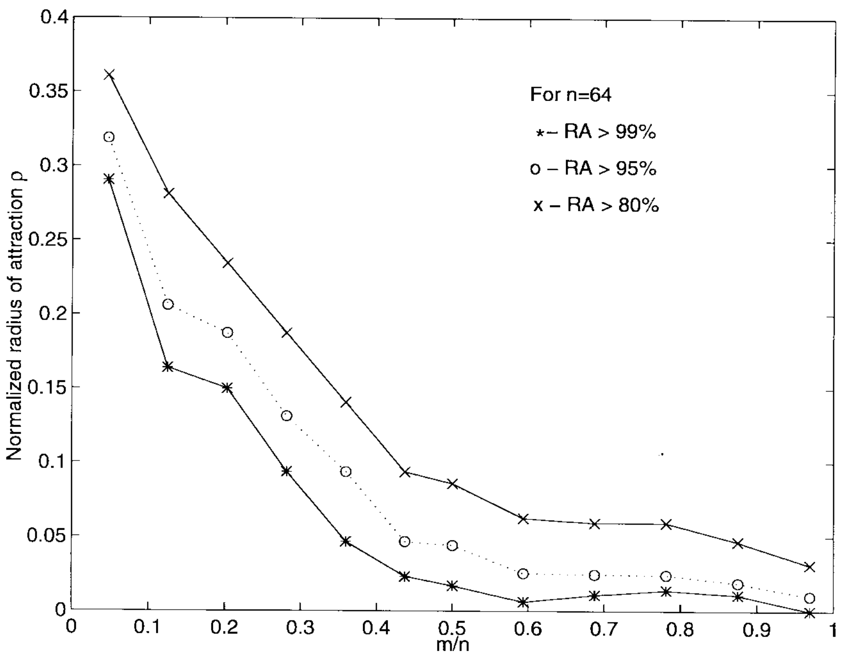

Fig. 5. Capacity performance for Synthesis Algorithm 3.1 for various RA ranges $(n = 64)$.

optimal constraints are generally smaller than the total number of spurious memories in a network which does not satisfy the optimal constraints. Also, the total number of spurious memories in a network change little with respect to the value of μ when the optimal constraints are satisfied. However, when the optimal constraints are not satisfied, the total number of spurious memories depends heavily on the value of μ (especially when $T_{ii} = 2, 3, 4$).

For the same design problem, results for neural networks designed using the eigenstructure method [15], [23] and the method of [32] are also shown in the table. Since the eigenstructure method does not have any control over the diagonal elements of the connection matrix T, the network designed by the eigenstructure method will usually have more spurious memories than the network designed using the approach developed herein and using the method of [32], while the later two design methods are comparable to each other in terms of the total number of spurious memories. It is noted that when the number of desired patterns is small (e.g., when $m \leq 5$), the eigenstructure method is also comparable to the approach developed herein and the method of [32]. It is also noted that the present synthesis approach is simpler to implement and usually takes less computational time than the method of [32]. ■

Example 7.4: The objective of this example is to study the capacity of associative memories designed using the present approach and to compare with the study in [8] and [9]. In the present example, capacity is defined as a measure of the ability of an associative memory to store a set of unbiased random bipolar patterns at a given error correction and recall accuracy level (cf. [8] and [9]). For each test, ten sets of m random bipolar patterns $(2 < m \leq n)$ are generated with memory size $n = 16, 32$, and 64. The capacity results are based on ensemble averages over ten training sets. The empirical data show that the capacity of system (1) with high recall accuracy (RA $>$ 99%) is approximately linear to the pattern dimension (i.e., $m \approx n$). When the number of stored patterns exceeds n, system (1) looses its error correction ability (given RA $>$ 99%). The results shown in Figs. 4 and 5 are comparable to the Ho–Kashyap recording [9] with certain improvement. For example, it is concluded in [9] that the Ho–Kashyap recording requires $m/n < 0.5$, while in the present case, m/n can be very close to one. ■

VIII. Conclusions

In this paper, the design problem of associative memories realized by feedback neural networks is considered. A new synthesis approach is developed based on the perceptron training algorithm. It is proved in the present paper that the perceptron training in the synthesis algorithm will always be convergent when there are no constraints on the connection matrix of neural networks (Theorem 3.1). Results concerning the properties of networks with constraints on the diagonal elements of the connection matrix are established (Corollary 4.1 and Theorem 4.1). For networks with constraints on the connection matrix (i.e., sparsity/symmetry constraints and constraints on the diagonal elements), conditions under which a design solution exists are established (Theorems 4.2 and 4.3 and Corollaries 5.1 and 5.2) and design algorithms are provided. The present synthesis approach provides the following features for a neural-network design.

- The network has a predetermined sparse or full interconnecting structure, with or without symmetry constraints on the interconnection weights.
- The design takes into account inaccuracies which arise in the realization of neural networks by hardware.
- The design of neural networks with certain constraints on the diagonal elements in the connection matrix can significantly reduce the total number of spurious memories and makes it very likely that every corner of the hypercube in the immediate neighborhood (with Hamming distance 1) of a stable memory vector is in the domain of attraction of the stable memory vector.

Four examples are provided to demonstrate the applicability and versatility of the present results.

References

[1] S. V. B. Aiyer, M. Niranjan, and F. Fallside, “A theoretical investigation into the performance of the Hopfield model,” *IEEE Trans. Neural Networks*, vol. 1, pp. 204–215, June 1990.

[2] M. Brucoli, L. Carnimeo, and G. Grassi, “Discrete-time cellular neural networks for associative memories with learning and forgetting capabilities,” *IEEE Trans. Circuit Syst. I*, vol. 42, pp. 396–399, July 1995.

[3] L. O. Chua and L. Yang, “Cellular neural networks: Theory,” *IEEE Trans. Circuit Syst.*, vol. CAS-35, pp. 1257–1272, Oct. 1988.

[4] M. A. Cohen and S. Grossberg, “Absolute stability of global pattern formation and parallel memory storage by competitive neural networks,” *IEEE Trans. Syst, Man, Cybern.*, vol. SMC-13, pp. 815–826, Sept./Oct. 1983.

[5] M. Cottrell, “Stability and attractivity in associative memory networks,” *Biol. Cybern.*, vol. 58, pp. 129–139, 1988.

[6] S. R. Das, “On the synthesis of nonlinear continuous neural networks,” *IEEE Trans. Syst., Man, Cybern*, vol. 21, pp. 413–418, Mar./Apr. 1991.

[7] R. M. Golden, “The ‘brain-state-in-a-box’ neural model is a gradient descent algorithm,” *J. Math. Psych.*, vol. 30, pp. 73–80, Mar. 1986.

[8] M. H. Hassoun, *Associative Neural Memories: Theory and Implementation*. Oxford, U.K.: Oxford Univ. Press, 1993.

[9] M. H. Hassoun and A. M. Youssef, “Associative neural memory capacity and dynamics,” in *Proc. Int. J. Conf. Neural Networks*, San Diego, CA, vol. 1, June 1990, pp. 763–769.

[10] Y. Ho and R. L. Kashyap, “An algorithm for linear inequalities and its applications,” *IEEE Trans. Electron. Comput.*, vol. EC-14, pp. 683–688, 1965.

[11] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities,” in *Proc. Nat. Academy Sci. USA*, vol. 79, Apr. 1982, pp. 2554–2558.

[12] ——, “Neurons with graded response have collective computational properties like those of two-state neurons,” in *Proc. Nat. Academy Sci. USA*, vol. 81, May 1984, pp. 3088–3092.

[13] S. Hui and S. H. Żak, “Dynamical analysis of the brain-state-in-a-box (BSB) neural models,” *IEEE Trans. Neural Networks*, vol. 3, pp. 86–94, Jan. 1992.

[14] J.-H. Li, A. N. Michel, and W. Porod, “Qualitative analysis and synthesis of a class of neural networks,” *IEEE Trans. Circuits Syst.*, vol. CAS-35, pp. 976–986, Aug. 1988.

[15] ——, “Analysis and synthesis of a class of neural networks: Linear systems operating on a closed hypercube,” *IEEE Trans. Circuits Syst.*, vol. 36, pp. 1405–1422, Nov. 1989.

[16] D. Liu and A. N. Michel, “Asymptotic stability of systems operating on a closed hypercube,” *Syst. Contr. Lett.*, vol. 19, pp. 281–285, Oct. 1992.

[17] ——, *Dynamical Systems with Saturation Nonlinearities: Analysis and Design*. New York: Springer-Verlag, 1994.

[18] ——, “Cellular neural networks for associative memories,” *IEEE Trans. Circuits Syst. II*, vol. 40, pp. 119–121, Feb. 1993.

[19] ——, “Sparsely interconnected neural networks for associative memories with applications to cellular neural networks,” *IEEE Trans. Circuit Syst. II*, vol. 41, pp. 295–307, Apr. 1994.

[20] ——, “Robustness analysis and design of a class of neural networks with sparse interconnecting structure,” *Neurocomputing*, vol. 12, pp. 59–76, June 1996.

[21] A. N. Michel and J. A. Farrell, “Associative memories via artificial neural networks,” *IEEE Contr. Syst. Mag.*, vol. 10, pp. 6–17, Apr. 1990.

[22] A. N. Michel, J. A. Farrell, and H.-F. Sun, “Analysis and synthesis techniques for Hopfield type synchronous discrete-time neural networks with application to associative memory,” *IEEE Trans. Circuits Syst.*, vol. 37, pp. 1356–1366, Nov. 1990.

[23] A. N. Michel, J. Si, and G. Yen, “Analysis and synthesis of a class of discrete-time neural networks described on hypercubes,” *IEEE Trans. Neural Networks*, vol. 2, pp. 32–46, Jan. 1991.

[24] A. N. Michel, K. Wang, D. Liu, and H. Ye, “Qualitative limitations incurred in implementations of recurrent neural networks,” *IEEE Contr. Syst. Mag.*, vol. 15, no. 3, pp. 52–65, June 1995.

[25] M. L. Minsky and S. A. Papert, *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA: MIT Press, 1969 and 1988 (expanded version).

[26] R. Perfetti, “A neural network to design neural networks,” *IEEE Trans. Circuits Syst.*, vol. 38, pp. 1099–1103, Sept. 1991.

[27] ——, “A synthesis procedure for brain-state-in-a-box neural networks,” *IEEE Trans. Neural Networks*, vol. 6, pp. 1071–1080, Sept. 1995.

[28] L. Personnaz, I. Guyon, and G. Dreyfus, “Information storage and retrieval in spin-glass like neural networks,” *J. Phys. Lett.*, vol. 46, pp. L359–365, Apr. 1985.

[29] L. Personnaz, I. Guyon, and G. Dreyfus, “Collective computational properties of neural networks: New learning mechanisms,” *Phys. Rev. A*, vol. 34, pp. 4217–4228, Nov. 1986.
[30] F. Rosenblatt, *Principles of Neurodynamics*. New York, NY: Spartan, 1962.
[31] F. M. A. Salam, Y. Wang, and M.-R. Choi, “On the analysis of dynamic feedback neural nets,” *IEEE Trans. Circuits Syst.*, vol. 38, pp. 196–201, Feb. 1991.
[32] G. Seiler, A. J. Schuler, and J. A. Nossek, “Design of robust cellular neural networks,” *IEEE Trans. Circuit Syst. I*, vol. 40, pp. 358–364, May 1993.
[33] Z.-B. Xu, G.-Q. Hu, and C.-P. Kwong, “Some efficient strategies for improving the eigenstructure method in synthesis of feedback neural networks,” *IEEE Trans. Neural Networks*, vol. 7, pp. 233–245, Jan. 1996.

Derong Liu (S’91–M’94–SM’96) received the Ph.D. degree in electrical engineering from the University of Notre Dame, Notre Dame, IN, in 1993.

From 1993 to 1995, he was with General Motors Research and Development Center, Warren, MI, where he worked on automotive engine diagnostics. Currently, he is Assistant Professor in the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ. He is coauthor of the book *Dynamical Systems with Saturation Nonlinearities: Analysis and Design* (New York: Springer-Verlag, 1994). His research interests include artificial neural networks, systems and control theory, and digital signal/image/video processing.

Dr. Liu was a member of the Program Committee of the 11th IEEE International Symposium on Intelligent Control (1996). He is a member of the Conference Editorial Board of the IEEE Control Systems Society, and he is the Local Arrangements Chair for the 6th IEEE Conference on Control Applications (1997). He was recipient of the Michael J. Birck Fellowship from the University of Notre Dame. He is a member of Eta Kappa Nu.

Zanjun Lu (S’97) received the B.S. degree in electrical engineering from Tianjin University, P.R. China, in 1992, and the M.S. degree in electrical engineering from the Institute of Automation, Chinese Academy of Sciences, in 1995. He has been working toward the Ph.D. degree in electrical engineering at Stevens Institute of Technology, Hoboken, NJ, since 1996.

From 1995 to 1996, he was a Faculty Member with the Department of Electrical Engineering at the Graduate School of Chinese Academy of Sciences, Beijing, China. His research interests include neural networks, system and control theory, information theory, and computer networks.