

# VARIATIONS IN ASSOCIATIVE MEMORY DESIGN

A THESIS  
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND  
ELECTRONICS ENGINEERING  
AND THE INSTITUTE OF ENGINEERING AND SCIENCES  
OF BILKENT UNIVERSITY  
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF  
MASTER OF SCIENCE

By  
Michael Akar  
August 1996

THESIS  
QA  
76.87  
.A33  
1996

# VARIATIONS IN ASSOCIATIVE MEMORY DESIGN

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND  
ELECTRONICS ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES  
OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF  
MASTER OF SCIENCE

By

Mehmet Akar

August 1996

EBSCO  
*Electronic Journals Service*

00000

QA

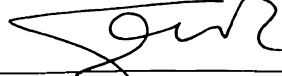
76.87

-A33

1936

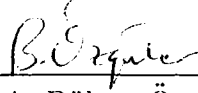
8034976

I certify that I have read this thesis and that in my opinion it is fully adequate,  
in scope and in quality, as a thesis for the degree of Master of Science.



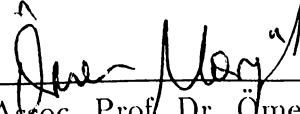
Prof. Dr. M. Erol Sezer(Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate,  
in scope and in quality, as a thesis for the degree of Master of Science.



Prof. Dr. A. Bülent Özgüler

I certify that I have read this thesis and that in my opinion it is fully adequate,  
in scope and in quality, as a thesis for the degree of Master of Science.



Assoc. Prof. Dr. Ömer Morgül

Approved for the Institute of Engineering and Sciences:



Prof. Dr. Mehmet Baray  
Director of Institute of Engineering and Sciences

## ABSTRACT

### VARIATIONS IN ASSOCIATIVE MEMORY DESIGN

Mehmet Akar

M.S. in Electrical and Electronics Engineering

Supervisor: Prof. Dr. M. Erol Sezer

August 1996

This thesis is concerned with the analysis and synthesis of neural networks to be used as associative memories. First considering a discrete-time neural network model which uses a quantizer-type multilevel activation function, a way of selecting the connection weights is proposed. In addition to this, the idea of overlapping decompositions, which is extensively used in the solution of large-scale problems, is applied to discrete-time neural networks with binary neurons. The necessary tools for expansions and contractions are derived, and algorithms for decomposition of a set equilibria into smaller dimensional equilibria sets and for designing neural networks for these smaller dimensional equilibria sets are given. The concept is illustrated with various examples.

*Keywords* : Hopfield neural network, associative memory design, multilevel activation function, overlapping decomposition

## ÖZET

### ÇAĞRIŞIMSAL BELLEK TASARIMI ÜZERİNE ÇEŞİTLEMELER

Mehmet Akar

Elektrik ve Elektronik Mühendisliği Bölümü Yüksek Lisans

Tez Yöneticisi: Dr. M. Erol Sezer

Ağustos 1996

Bu tez sinir ağlarının çağrışimsal bellek olarak kullanılması amacıyla çözümlenmesi ve tasarımı ile ilgilidir. Öncelikli olarak, ayrık zamanda nicemleyici tür fonksiyon kullanan bir sinir ağı modeli için bağlantı ağırlıklarının seçimi için bir yol önerilmiştir. Buna ek olarak, büyük ölçekli problemlerin çözümünde çokça kullanılan örtüşen parçalama tekniği, iki durumlu sinir hücreleri kullanan ayrık zaman sinir ağ modellerine uygulanmıştır. Genişletme ve büzme için gerekli kurallar çıkarılmış ve bir denge vektörleri kümesinin daha küçük boyutlu iki denge vektörleri kümesine denk olarak indirgenebilme ve bu küçük boyutlu kümeler için sinir ağları tasarımı için algoritmalar verilmiştir. Konu değişik örneklerle aydınlatılmıştır.

*Anahtar Kelimeler* : Hopfield sinir ağı, çağrışimsal bellek tasarımı, çok seviyeli hareketlendirme fonksiyonu, örtüşen parçalama

**To my family**

## ACKNOWLEDGEMENT

I would like to express my deep gratitude to my supervisor Prof. Dr. Mesut Erol Sezer for his guidance, suggestions and invaluable encouragement throughout the development of this thesis.

I would like to thank to Assoc. Dr. Ömer Morgül and Prof. Dr. Bülent Özgüler for reading and commenting on the thesis and for the honor they gave me by presiding the jury.

I would like to thank to my family for their patience and support.

And special thanks to my friends for their helpful discussions and suggestions.



# TABLE OF CONTENTS

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Review of Associative Memory Design</b>	<b>4</b>
2.1	Continuous-Time Neural Networks	6
2.1.1	The Outer Product Method . . . . .	6
2.1.2	The Projection Rule . . . . .	7
2.1.3	The Eigenstructure Method . . . . .	7
2.2	Discrete-Time Neural Networks . . . . .	8
<b>3</b>	<b>Neural nets with multilevel functions</b>	<b>14</b>
3.1	Past work on neural nets with multilevel functions . . . . .	15
3.2	Analysis and Synthesis of neural nets with multilevel functions .	16
3.3	Design of neural nets using multi-level functions . . . . .	19
<b>4</b>	<b>Decomposition Methods</b>	<b>27</b>
4.1	Disjoint Decompositions . . . . .	28
4.2	Overlapping Decompositions . . . . .	29

4.2.1	Linear Systems . . . . .	29
4.2.2	Nonlinear Systems	31
4.3	Application to Discrete-Time Neural Networks . . . . .	32
4.3.1	Overlapping Decompositions of Neural Networks . . . . .	34
<b>5</b>	<b>Examples</b>	<b>48</b>
<b>6</b>	<b>Conclusion</b>	<b>64</b>

## LIST OF FIGURES

3.1	Quantizer-type multi-level function with K levels	17
3.2	Quantizer-type multilevel with 2K levels . . . . .	19
3.3	4 level quantizer	22
4.1	Threshold function . . . . .	36
5.1	Original and decomposed systems	57
5.2	Characters to be recognized by the neural network	63

## LIST OF TABLES

3.1	Properties of the neural network for different $\tau$ , with $T = \bar{Y}\bar{Y}^\dagger - \tau(I - \bar{Y}\bar{Y}^\dagger)$ and $b = 0$	26
3.2	Properties of the neural network for different $\tau$ , with $T = \tilde{Y}\tilde{Y}^\dagger - \tau(I - \tilde{Y}\tilde{Y}^\dagger)$ and $b = \bar{y}_4 - T\bar{y}_4$	26
4.1	1st row overlapped . . . . .	45
4.2	2nd row overlapped . . . . .	45
4.3	3rd row overlapped . . . . .	45
4.4	4th row overlapped . . . . .	46
4.5	5th row overlapped . . . . .	46
4.6	6th row overlapped . . . . .	46
4.7	7th row overlapped . . . . .	47
4.8	8th row overlapped . . . . .	47
4.9	9th row overlapped . . . . .	47
5.1	Comparison of different methods for Example 1	62
5.2	Comparison of different methods for Example 2 . . . . .	62
5.3	Basin of attraction of the prototypes for Example 3 . . . . .	62

# Chapter 1

## Introduction

There are problems in nature (pattern recognition for example) which are easily solved by people and animals but which are difficult to solve with today's digital computing technology. These kinds of problems have two characteristics : they are ill-posed and their solutions need an enormous amount of computation. To overcome these difficulties, scientists have been working hard for many years to build intelligent systems that can model the highly complex, nonlinear and parallel structure of the human brain. As a result, neural networks which try to model the brain became one of the challenging fields.

Work on neural network models has a long history, but interest on neural networks has arised since Hopfield [1, 2] proposed his model, and neural networks have been used to solve many problems in various fields such as control, classification, pattern recognition and optimization.

Neural networks consist of computational elements called neurons and weighted connections between these neurons. Neurons are multi-input, single-output, nonlinear processing units which form a weighted sum of its inputs and passes the result through a nonlinear function, called activation function.

With a proper choice of the connection weights, the neural network can store some desired vectors as asymptotically stable equilibria of the network. This problem, called the associative memory design problem, has been analyzed by various researchers using both discrete-time and continuous-time neural network models. In [1, 2], Hopfield used the outer product rule to store a given

set of memory vectors for the restrictive case of orthogonal patterns. Later in [3, 4], the authors proposed the projection learning rule which guarantees any set of desired memory vectors to be stored as equilibria of the network. In [5, 6, 7], Michel and his coworkers used the eigenstructure method in which the connection matrix is synthesized so that the memory vectors become the eigenvectors of that matrix with a single positive eigenvalue. In [8], Lillo *et al* used the brain-state-in-a-box model to realize an associative memory. Later Perfetti [9], using the same model, developed some criteria to increase the basin of attraction of the desired patterns and based his synthesis procedure on this criteria. The past work on the design of associative memories is reviewed in detail in Chapter 2.

All the above work use two-level activation functions. Using multilevel activation functions help us to decrease the number of neurons used. In [10, 11], the authors used the outer product rule to design networks using multilevel activation functions. In [12], the authors based their synthesis procedure on local stability, global stability and equilibrium constraints they derived. Other work concerning neural networks using multilevel activation functions include [13, 14, 15, 16]. All this work is reviewed in detail in Chapter 3.

In this thesis, we first consider a discrete-time neural network model and analyze the associative memory problem in the case of multilevel activation functions. We propose a way to compute the connection matrix and comment on the stability issues. The advantages and disadvantages of using multilevel activation functions are illustrated with an example using the existing methods and the proposed method.

In the rest of the thesis, we employ the concept of overlapping decompositions, which is used in the large-scale system design problems, to relieve the computational work in designing associative memories. The idea of overlapping decomposition design is to obtain the global solution to a large-scale problem by dividing the system into a number of smaller subsystems sharing some common parts, and then combining the individual solutions of these subsystems. The concept of expansions and contractions are made precise, and necessary conditions are derived so that the overlapping decomposition methodology can be applied to the design of associative memories. A decomposition algorithm is given to decompose the desired set of equilibria into two smaller dimensional equilibria sets equivalently and a design algorithm is given to design neural networks for these smaller dimensional equilibria sets. Finally the concept is

illustrated with various examples.

This thesis is organized as follows. In Chapter 2, we summarize the past work on associative memory design using binary activation functions. In Chapter 3, we first review the past work on neural networks using multilevel activation functions and then analyze the associative memory design problem considering a discrete-time neural network model with a multilevel activation function. In Chapter 4, we first review some results on expansions, contractions and overlapping decompositions from large-scale system theory, and then apply the idea to the design of neural networks. In Chapter 5, we give various examples on the application of the ideas presented in Chapter 4. In Chapter 6, we give the concluding remarks.

## Chapter 2

# Review of Associative Memory

## Design

Design of associative memories has attracted great attention after Hopfield [1, 2] proposed a nonlinear continuous model which can be realized by electronic circuitry. The equation governing the electronic circuit is described by a set of first order ordinary differential equations as

$$C_i \dot{x}_i = \sum_{j=1}^n T_{ij} f_j(x_j) - \frac{x_i}{R_i} + I_i, \quad i = 1, \dots, n \quad (2.1)$$

where  $x_i$  is the input voltage of the nonlinear amplifier,  $I_i$  is a fixed bias current, and  $f_i(\cdot)$  represents the input-output characteristics of the amplifier called the activation function, which is usually a smooth, saturation type nonlinearity such as a sigmoid function.  $C_i$  and  $R_i$  are capacitor and resistor values, respectively, and  $T_{ij}$  are interconnection weights.

Letting  $x = [x_1 \ x_2 \ \dots \ x_n]^T$ ,  $f(x) = [f_1(x_1) \ f_2(x_2) \ \dots \ f_n(x_n)]^T$ ,  $b = [I_1/C_1, I_2/C_2, \dots, I_n/C_n]^T$ ,  $A = \text{diag}\{1/R_1 C_1, 1/R_2 C_2, \dots, 1/R_n C_n\}$ , and  $T = [T_{ij}/C_i]$ , the above class of neural networks can be described more compactly as

$$\dot{x} = -Ax + Tf(x) + b \quad (2.2)$$

which represents a dynamical system with state  $x \in \mathcal{R}^n$ , and a fixed input  $b \in \mathcal{R}^n$ .



The discrete counterpart of the above continuous model can be described by a first order nonlinear difference equation as

$$x(k+1) = f(Tx(k) + b) \quad (2.3)$$

where  $x$  and  $b$  are the state and the bias input respectively.

Associative memory problem is to store a desired set of patterns as stable memories of the neural network. The problem corresponds to solving for  $A$ ,  $T$  and  $b$  in the continuous-time case and solving for  $T$  and  $b$  for the discrete-time case. The desired characteristics of the resulting neural network should be [17, 18]:

1. Each prototype pattern stored as an asymptotically stable equilibrium point of the system.
2. A minimum number of asymptotically stable equilibrium points of the network which do not correspond to prototype patterns (i.e., spurious states).
3. A non-symmetric interconnection structure, which eases difficulties in the implementation of neural networks.
4. The ability to control the extent of basin of attraction about the equilibrium points corresponding to stored patterns.
5. Learning (i.e., the ability to add vectors to be stored as asymptotically stable equilibrium points to an existing set of stored vectors without affecting the existing equilibria in a given network) and forgetting (i.e., the ability to delete specified equilibrium points from a given set of stored equilibria without affecting the rest of equilibria in a given network) capabilities.
6. A high storage and retrieval efficiency, i.e., the ability to efficiently store and retrieve a large number (compared to the order  $n$  of the network) of patterns.

## 2.1 Continuous-Time Neural Networks

Now we will concentrate on the continuous-time model with sigmoidal non-linearity and summarize some of the results that appeared on the design of associative memories. We wish to store  $m$  desired patterns  $y^i$ ,  $1 \leq i \leq m$  (i.e.  $y^i = f(x^i)$ ) as stable memories of (2.2).

### 2.1.1 The Outer Product Method

A set of parameter choices determined by the Outer Product Method [1, 2] is given by

$$T = \sum_{j=1}^m y^j (y^j)^T, \quad A = I, \quad b = 0 \quad (2.4)$$

The name of this method is motivated by the fact that  $T$  consists of the sum of outer products of the patterns that are to be stored as stable memories. This method requires that  $y^i$ ,  $1 \leq i \leq m$ , be mutually orthogonal (i.e.,  $(y^i)^T y^j = 0$  when  $i \neq j$ ). Advantages of outer product rule are learning and forgetting. Learning is accomplished by modifying (2.4) as

$$T \rightarrow T + \alpha y^l (y^l)^T, \quad A = I, \quad b = 0 \quad (2.5)$$

where  $y^l$  is a new memory to be learned by the network. Forgetting is accomplished by modifying (2.4) as

$$T \rightarrow T - \alpha y^l (y^l)^T, \quad A = I, \quad b = 0 \quad (2.6)$$

where  $y^l$  is a stored memory to be forgotten by the network. In both cases,  $\alpha > 0$  is a small constant which determines the rate of learning or forgetting. Experience has shown that networks designed by this method can store effectively only up to  $0.15n$  [18] arbitrary vectors as equilibrium points where  $n$  denotes the order of the network. Moreover, design by outer product rule results in neural networks that are required to have symmetric interconnection structure, which gives rise to spurious states in addition to posing difficulties in implementations. Another important attribute of this method is that networks designed by this technique are globally stable (i.e., all trajectories of the network tend to some equilibrium point), as can be shown using a suitable energy function.

### 2.1.2 The Projection Rule

When the desired prototype patterns  $y^i$ ,  $1 \leq i \leq m$ , to be stored in (2.2) as stable memories are not mutually orthogonal, a method called the Projection Learning Rule [3, 4, 19] can be used to synthesize the interconnection parameters for (2.2). Let  $\Sigma = [y^1, \dots, y^m]$ . Then

$$T = \Sigma \Sigma^\dagger, \quad A = I, \quad b = 0 \quad (2.7)$$

is the set of parameters for (2.2) where  $\Sigma^\dagger$  is the Moore-Penrose pseudo-inverse [20] of  $\Sigma$ . We note that  $T$  given above satisfies the relation  $T\Sigma = \Sigma$ , which shows that  $T$  is an orthogonal projection of  $R^n$  onto the linear space spanned by  $y^i$ ,  $1 \leq i \leq m$  (hence the name Projection Rule). When  $y^i$ ,  $1 \leq i \leq m$ , are mutually orthogonal, the Projection Learning Rule and the Outer Product Method coincide. This method has two advantages over the Outer Product Method. First, networks designed by this method are capable of storing effectively  $0.5n$  [18] equilibrium points. Secondly, this technique guarantees that a network designed by this method will always store a given vector as an equilibrium point. However, this equilibrium point need not be asymptotically stable. Since the Moore-Penrose pseudo-inverse can be computed iteratively, there are also adaptive learning and forgetting rules [3, 4].

### 2.1.3 The Eigenstructure Method

This technique [21, 5, 18, 22] also utilizes the energy function approach, thus guarantees to store the desired set of patterns as stable memories. The patterns need not be mutually orthogonal as in the Outer Product Method. In the following, we outline Michel's algorithm [22] for the case when the desired set consists of bipolar vectors, i.e.,  $y^j \in \mathcal{B}^n = \{y \in \mathcal{R}^n : y_i = 1 \text{ or } y_i = -1, i = 1, \dots, n\}$ , and the activation functions are saturation nonlinearities. The algorithm is as follows:

#### Algorithm 2.1 (Michel's algorithm)

1. Compute the  $n \times (m - 1)$  matrix

$$Y = [y^1 - y^m, \dots, y^{m-1} - y^m] \quad (2.8)$$

2. Perform the singular value decomposition of  $Y$  as  $Y = U\Sigma V^T$  where  $U$  and  $V$  are unitary matrices and  $\Sigma$  is a diagonal matrix with the singular values of  $Y$  on its diagonal. Letting  $U = [u^1, \dots, u^n]$  we know that  $u^1, \dots, u^n$  is an orthonormal basis for  $\mathcal{R}^n$ . If we let  $k$  denote the dimension of the linear space  $\mathcal{L}$  spanned by the vectors  $y^1 - y^m, \dots, y^{m-1} - y^m$ , then  $u^1, \dots, u^k$  is an orthonormal basis for  $\mathcal{L}$  and  $u^{k+1}, \dots, u^n$  is an orthonormal basis for  $\mathcal{L}^\perp$ .
3. The parameters of the neural network are given as

$$\begin{aligned} T &= \tau_1 \sum_{i=1}^k (u^i)(u^i)^T - \tau_2 \sum_{i=k+1}^n (u^i)(u^i)^T, \\ A &= I, \quad b = \tau_1 y^m - T y^m \end{aligned} \quad (2.9)$$

where  $\tau_1 > 1$ . It is shown in [21, 5] that when  $\tau_2 > 0$  is sufficiently large, all desired patterns are stored as stable memories. In fact, all vectors in  $\mathcal{L}_a \cap B^n$  are stable memories, where  $\mathcal{L}_a$  is the affine space given by  $\mathcal{L} + y^m$ .

For the eigenstructure method, iterative learning and forgetting rules have also been worked for the above design scheme [18].

## 2.2 Discrete-Time Neural Networks

The discrete-time neural networks are described as in (2.3), where the activation functions are a saturating linearity defined as

$$f_i(x) = \begin{cases} 1 & \text{if } x \geq 1 \\ x & \text{if } -1 < x < 1 \\ -1 & \text{if } x \leq -1 \end{cases}$$

Li *et al* considered this model in [5]. In their paper they find all the solutions of the above difference equations, hence characterize all possible equilibria and asymptotically stable equilibria. Then considering a symmetric  $T$ , they define the energy function  $E(x) = -x^T(T - I)x - 2x^T b$ , and show that the energy decreases monotonically along non-equilibrium solutions of the system and each

non-equilibrium solution converges to an equilibrium, hence the neural network is globally stable, under the assumption that the eigenvalues of  $T$  are greater than  $-1$ .

The synthesis procedure they propose is the same as the method they propose for the continuous-time case. In another paper [18], they derive the learning and forgetting algorithms for the given synthesis procedure. They also show that the computational complexity of incremental learning/forgetting algorithm approaches  $O(mn^2)$  asymptotically while the complexity is  $O(n^3 + mn^2 + m^2n + m^3)$  for the classical learning.

The brain-state-in-a-box(BSB) neural model, which was first proposed by Anderson [23] in 1977, is another discrete-time neural network model which can be described by a set of first order difference equations as

$$x(k+1) = f(x(k) + \alpha W x(k)) \quad (2.10)$$

where  $x \in \mathcal{R}^n$  (denoting the neuron variables),  $\alpha > 0$  is the step size,  $W \in \mathcal{R}^{n \times n}$  (representing the interconnections of neurons) and  $f(\cdot)$  is the saturating linearity given above. The function  $f(\cdot)$  is responsible for the name given to the above equation, as the state vector  $x(k)$  lies in the “box”  $\mathcal{H}_n = [-1, 1]^n$ , which is the closed  $n$ -dimensional hypercube. Later Hui and Zak [24] generalized this model by introducing the vector  $\alpha b$ :

$$x(k+1) = f((I_n + \alpha W)x(k) + \alpha b) \quad (2.11)$$

where  $b \in \mathcal{R}^n$  (representing bias terms). Lillo *et al* [8] used this generalized brain-state-in-a-box (GBSB) model to realize an associative memory. In their paper the desired prototype patterns are mapped to the corresponding asymptotically stable vertices of the hypercube. To summarize their results, let

$$L(x) = (I_n + \alpha W)x + \alpha b \quad (2.12)$$

One can verify [25] that a vertex  $x^*$  of the hypercube  $\mathcal{H}_n$  is asymptotically stable equilibrium of the GBSB model if

$$(L(x^*))_i x_i^* > 1, \quad i = 1 \dots n$$

Their main contribution is the following theorem:

**Theorem 2.1** *Let  $Y = [y^1 \dots y^m] \in \mathcal{R}^{n \times m}$  be the matrix of prototype patterns. Assume that the prototype patterns are linearly independent so that*

$\text{rank}(Y)=m$ . Let  $B = [b \dots b] \in \mathcal{R}^{n \times m}$ . Suppose  $D \in \mathcal{R}^{n \times n}$  is a strongly row diagonal dominant matrix whose components satisfy

$$d_{ii} > \sum_{j=1, j \neq i}^n |d_{ij}|, \quad i = 1, \dots, n \quad (2.13)$$

and  $\Lambda \in \mathcal{R}^{n \times n}$  is a matrix whose components satisfy

$$\lambda_{ii} < - \sum_{j=1, j \neq i}^n |\lambda_{ij}| - b_i, \quad i = 1 \dots n \quad (2.14)$$

If

$$W = (DY - B)Y^\dagger + \Lambda(I_n - YY^\dagger) \quad (2.15)$$

where  $Y^\dagger = (Y^T Y)^{-1} Y^T$ , then all of the desired patterns will be stored as asymptotically stable equilibrium points of the hypercube  $\mathcal{H}_n$ .

They also provide a simple algorithm to select  $D$ ,  $b$ , and  $\Lambda$  so that  $W$  can be computed from the theorem. Their four step algorithm is as follows:

**Algorithm 2.2 (Lillo et al)**

1. Select a strongly row diagonal dominant matrix  $D \in \mathcal{R}^{n \times n}$ .
2. Select the components of the vector  $b$  such that

$$d_{ii} < \sum_{j=1, j \neq i}^n |d_{ij}| + b_i, \quad i = 1 \dots n$$

and

$$b = \sum_{i=1}^m c_i y^{(i)}, \quad c_i > 0, \quad i = 1 \dots n$$

Picking  $b$  to satisfy the first condition helps to ensure that the negatives of the desired memories are not stored as spurious states. Picking  $b$  to be a linear combination of the desired prototype vectors as in the second condition helps to ensure that the trajectory will be sent toward a stable vertex.

3. Pick a matrix  $\Lambda \in \mathcal{R}^{n \times n}$  such that (2.14) is satisfied.
4. Compute  $W$  with (2.15).

In the paper, the authors also show that this design procedure can also be used for signum activation function. However, the network designed by this technique is not guaranteed to be globally stable. Also, existence of learning and forgetting algorithms and storage and retrieval efficiency of the network have not been worked out yet.

Later BSB model is analyzed by Perfetti [9]. In this paper the author derives some sufficient conditions which guarantee: i) the absence of non-binary asymptotically stable equilibrium points, ii) the absence of binary equilibrium points near a desired memory vector. The main contribution in the design given in this paper is that it allows one to optimize a design parameter which controls the size of the attraction basins of the stored patterns.

The author first shows that if  $w_{ii} > 0$ ,  $i = 1, \dots, n$ , then only the vertices of the hypercube can be stable equilibria. If we also have  $w_{ii} = 0$ ,  $i = 1, \dots, n$ , then there is no equilibria at Hamming distance 1 or  $n - 1$  from the stored vector. However, the most appealing part of the work in the paper is the following theorem.

**Theorem 2.2** *None of the vertices  $\xi^*$  satisfying  $H(\xi, \xi^*) \leq k$  or  $H(\xi, \xi^*) \leq n - k$  is an equilibrium point if*

$$\sum_{j=1}^n w_{ij} \xi_i \xi_j \geq 2k \max_j |w_{ij}|, \quad i = 1, \dots, n \quad (2.16)$$

where  $H(\xi, \xi^*)$  denotes the Hamming distance between  $\xi$  and  $\xi^*$ .

For a design procedure, the conjecture the author follows is that increasing the basin of instability of the given patterns will increase their basin of attraction. Clearly, one can impose the conditions in the theorem. However, these sufficient but not necessary conditions are very strict. So, to increase the domain of attraction of the stored patterns, Perfetti's strategy is the maximization of the left-hand sums in the theorem.

According to the considerations outlined above, Perfetti's synthesis strategy can be formulated as follows: Assume  $\alpha = 1$ . Find  $W$  such that  $\delta$  is maximum, subject to the linear constraints

$$\sum_{j=1}^n w_{ij} y_i^{(k)} y_j^{(k)} \geq \delta > 0, \quad i = 1, \dots, n, \quad k = 1, \dots, m \quad (2.17)$$

$$-1 \leq w_{ij} \leq 1, \quad i, j = 1, \dots, n \quad (2.18)$$

$$w_{ij} = w_{ji} \quad , \quad i \neq j \quad , \quad i, j = 1, \dots, n \quad (2.19)$$

$$w_{ii} = 0 \quad , \quad i = 1, \dots, n \quad (2.20)$$

and to the nonlinear constraint

$$\lambda_{min}(W) > -2 \quad (2.21)$$

Without constraints (2.18) the maximization of  $\delta$  would be meaningless. Note that constraint (2.21) which is required for the global stability of the network is obtained from [5]. Due to the large number of unknowns and constraints, it is cumbersome to look for the optimal  $\delta$  using classical simplex method, therefore the author proposes the following algorithm:

### Algorithm 2.3 (Perfetti)

1. Find  $W = W^{(0)}$  so as to satisfy the linear constraints (2.17-2.20) with  $\delta = 0$ 
  - (a) If a solution exists, the vectors  $y^1 \dots y^m$  are stored as equilibrium points of the neural network. Then let  $r=1$ , choose  $\delta^{(1)} > 0$  and go to step 2.
  - (b) If a solution does not exist, it is impossible to store the vectors  $y^1 \dots y^m$  in the associative memory using a zero-diagonal connection matrix.
2. Find  $W = W^{(r)}$  so as to satisfy the linear constraints (2.17-2.20) with  $\delta = \delta^{(r)} > 0$ . If a feasible solution to (2.17-2.20) exists go to Step 3. Otherwise go to Step 4.
3. Find the minimum eigenvalue  $\lambda_{min}^{(r)}$  of  $W^{(r)}$ . If  $\lambda_{min}^{(r)} > -2$  then increase  $r$  by 1, increase  $\delta$  and go to Step 2. Otherwise go to Step 4.
4. Let  $W = W^{(r-1)}$  and  $\alpha = 1$ . The vectors  $y^1 \dots y^m$  have been stored as asymptotically stable equilibria of the neural network.

A time-consuming task of the proposed synthesis procedure is the repeated application of Step 2. Therefore, the author proposes a more efficient algorithm by defining the constraints in (2.17-2.20) as an unconstrained optimization problem.



It seems that the basic advantage of this technique over the existing ones is that there is no stable equilibria at Hamming distance one from the desired patterns provided that such a solution exists with zero-diagonal connection matrix constraint. A great disadvantage of this technique is the complexity of finding a solution. There are no learning and forgetting rules for this method and a storage capacity analysis should have to be worked out as well.

## Chapter 3

# Neural nets with multilevel functions

In this chapter, we consider the analysis and synthesis of neural networks using multilevel activation functions. We first review the literature on the subject. Then considering a discrete-time neural network model, we state the conditions for a set of desired patterns to be asymptotically stable equilibria of this model using the multi-level quantizer shown in Figure 3.1. We then show that the quantizer-type functions with the same number of saturating levels are equivalent in the sense that there is a transformation which maps the design parameters computed for one type of function to be used for another quantizer-type function. In the rest of the chapter, we deal with the analysis and synthesis problem of associative memories using multi-level activation functions. We finally conclude the chapter with an example illustrating the advantages and disadvantages of using multilevel functions.

### 3.1 Past work on neural nets with multilevel functions

In VLSI implementations of artificial feedback neural networks, reductions in the number of neurons and in the number of interconnections are highly desirable. If an  $n$ -dimensional vector with each component of  $q$ -bit length is to be stored in a neural network with binary state neurons, then an  $n \times q$  order system may be used. Alternatively, an  $n$ -dimensional neural network may be used for this purpose, provided that each neuron can represent a  $q$ -bit information, which is possible by using a  $q$ -level activation function for the neurons. In the former case, the number of interconnections will be of the order  $(n \times q)^2$ , while in latter case, the number of interconnections will be only of the order  $n^2$ .

Outer product method has been used in the design of discrete-time neural networks which make use of quantizer-type multilevel activation function [10, 11] but we know that this design technique is successful only in the case of orthogonal patterns.

In [13], the stability, capacity and design of a nonlinear continuous-time neural network are analyzed. They derive a set of sufficient conditions for the asymptotic stability of each desired equilibrium and phrase these conditions in terms of linear equations and piecewise linear inequality relations. The authors then suggest to solve these inequality relations either using methods such as Fourier elimination or using another neural network which can solve inequalities, but they do not provide specific information about this.

In [14, 15], the authors analyze a discrete-time neural network with continuous state variables updated in parallel and show that for symmetric connections, the only attractors are fixed points and period-two limit cycles. They also present a global stability criterion which guarantees only fixed-point attractors by placing limits on the gain of the sigmoid nonlinearity.

In [16], Meunier *et al* introduce networks of three-state  $(-1,0,+1)$  neurons, where the additional state embodies the absence of information. An extensive simulation study has been carried by the authors on the information processing capacity of these networks.

In [12], the authors consider a class of synchronous, discrete-time neural

networks described by first order linear difference equations. A local qualitative analysis of neural networks is conducted independent of the number of levels employed in the threshold nonlinearities. In doing so, the large scale systems methodology is used to perform a stability analysis. Next by using energy functions, the global stability is established. Finally a synthesis procedure for the neural network to store some memories as asymptotically stable equilibrium points is developed based on local stability, global stability and equilibrium constraints. In the paper they apply this synthesis procedure to a gray level image processing example, where each neuron can assume one of the sixteen values.

## 3.2 Analysis and Synthesis of neural nets with multilevel functions

Consider a discrete-time neural network model described by

$$x(k+1) = f(Tx(k) + b) \quad (3.1)$$

where  $x(k) \in \mathcal{R}^n$  is the state vector at instant  $k$ ,  $T \in \mathcal{R}^{n \times n}$  is the interconnection matrix,  $b \in \mathcal{R}^n$  is the bias term and  $f(x) = [f_1(x_1) \ f_2(x_2) \ \dots \ f_n(x_n)]^T$  with  $f_i(\cdot)$  a multi-level quantizer-type function with  $K$  levels as shown in Figure 3.1. We assume that  $f_i(\cdot)$  are right continuous. Note that the neural network in (3.1) is completely characterized by a triple  $(f, T, b)$ .

We begin by stating the equilibria conditions for the neural network model (3.1) as a theorem whose proof is trivial, and is omitted.

**Theorem 3.1** *A vector  $y_e$  is an asymptotically stable equilibrium of the neural network model (3.1) with the multi-level function given in Figure 3.1 if and only if*

- a)  $y_e \in \mathcal{Z}_d^n$ , where  $\mathcal{Z}_d = \{d_0, d_1, \dots, d_{K-1}\}$ , and
- b) *it satisfies (componentwise) the inequalities*

$$\underline{c} \leq Ty_e + b < \bar{c} \quad (3.2)$$

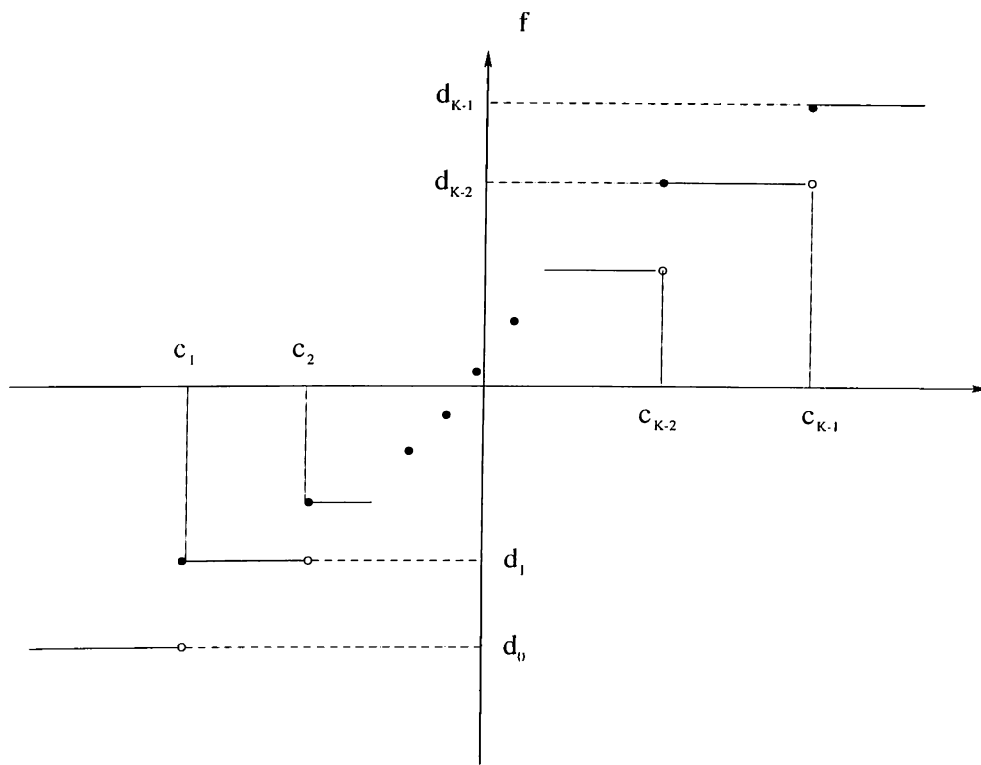


Figure 3.1: Quantizer-type multi-level function with  $K$  levels

where

$$\underline{c}_i = \begin{cases} c_l & \text{if } y_{ei} = d_l, \quad l = 1, 2, \dots, K-1 \\ -\infty & \text{if } y_{ei} = d_0 \end{cases} \quad (3.3)$$

$$\bar{c}_i = \begin{cases} c_{l+1} & \text{if } y_{ei} = d_l, \quad l = 0, 1, \dots, K-2 \\ \infty & \text{if } y_{ei} = d_{K-1} \end{cases} \quad (3.4)$$

It is clear that neural network described by (3.1) has at most  $K^n$  asymptotically stable equilibrium points.

Before going on to the synthesis problem, we will consider the following problem.

**Problem 3.1** Given a neural network, characterized by  $(f, T, b)$ , with a set  $\{y_e^{(j)}, j = 1, 2, \dots, m\}$  of equilibrium points. Let  $f'$  defined as

$$f'(x) = \alpha f(\beta x + \gamma) + \delta$$

be another  $K$ -level quantizer with quantization levels  $d'_i = \alpha d_i + \delta$ ,  $i = 0, 1, \dots, K-1$ , where  $\alpha, \beta, \gamma, \delta$  are constants and  $\alpha, \beta \neq 0$ . Find, if possible,

$T'$  and  $b'$  such that the set of equilibria of the neural network  $(f', T', b')$  is exactly  $\{y_e^{(j)} = \alpha y_e^{(j)} + \delta e, j = 1, 2, \dots, m\}$  where  $e = [1 \dots 1]^T \in \mathcal{R}^n$ .

We give the solution to the above problem in the following theorem.

**Theorem 3.2** *The choice*

$$T' = \frac{1}{\alpha\beta}T, \quad b' = \frac{1}{\beta}b - \frac{\gamma}{\beta}e - \frac{\delta}{\alpha\beta}Te \quad (3.5)$$

*solves Problem 3.1.*

**Proof:** Let  $y_e = [d_{i_1} \ d_{i_2} \ \dots \ d_{i_n}]^T$  be an equilibrium of  $(f, T, b)$  and consider  $y'_e = \alpha y_e + \delta e = [\alpha d_{i_1} + \delta \ \dots \ \alpha d_{i_n} + \delta]^T$ . Using (3.5), we have

$$T'y'_e + b' = \frac{1}{\beta}(Ty_e + b - \gamma e)$$

so that

$$\frac{1}{\beta}(\underline{c} - \gamma e) \leq T'y'_e + b' < \frac{1}{\beta}(\bar{c} - \gamma e).$$

However, the discontinuity points of  $f'$  are, by definition,  $c'_i = (c_i - \gamma)/\beta$ ,  $i = 1, 2, \dots, K - 1$ . Hence,

$$\underline{c}' \leq T'y'_e + b' < \bar{c}'$$

so that  $y'_e = \alpha y_e + \delta e$  is an equilibrium of  $(f', T', b')$ .

Conversely, if  $y'_e$  is an equilibrium of  $(f', T', b')$ , then  $y'_e = \alpha y_e + \delta e$  for some equilibrium  $y_e (= \frac{1}{\alpha}y'_e - \frac{\delta}{\alpha}e)$  of  $(f, T, b)$ .

Theorem 3.2 simply states that all  $K$ -level quantizer activation functions are equivalent in the sense that, with the parameters  $T$  and  $b$  chosen suitably, the associated neural networks have equivalent equilibria sets. This allows the quantization levels and the discontinuity points of the activation functions to be chosen as desired to simplify the analysis and design procedures.

In the next section, we will concentrate on the associative memory design using multi-level activation functions.

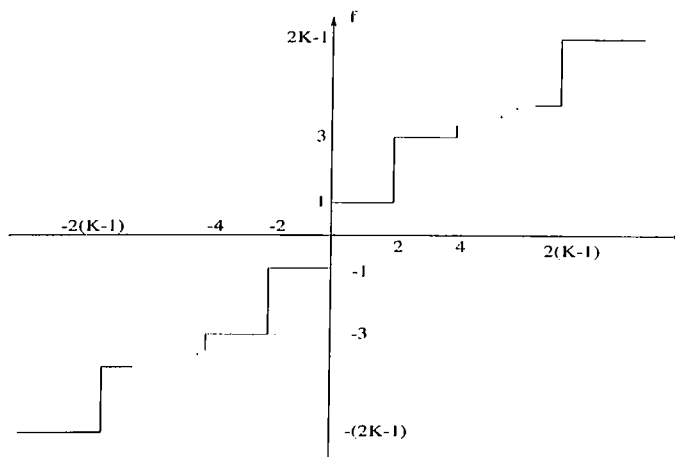


Figure 3.2: Quantizer-type multilevel with  $2K$  levels

### 3.3 Design of neural nets using multi-level functions

In this section we will consider methods of designing neural networks using multi-level quantizer type functions. We will in particular consider symmetric  $2K$ -level quantizer type functions as shown in Figure 3.2 for easier representation of equilibria constraints.

For this quantizer, the equilibria constraints for the memory vector  $y$  to be asymptotically stable equilibria of the neural network model (3.1) can be formulated as:

$$\underline{c}_i \leq (Ty + b)_i < \bar{c}_i, \quad i = 1 \dots n \quad (3.6)$$

where

$$\underline{c}_i = \begin{cases} y_i - 1 & \text{if } y_i \neq -(2K - 1) \\ -\infty & \text{if } y_i = -(2K - 1) \end{cases} \quad (3.7)$$

$$\bar{c}_i = \begin{cases} y_i + 1 & \text{if } y_i \neq 2K - 1 \\ \infty & \text{if } y_i = 2K - 1 \end{cases} \quad (3.8)$$

Based on design methods derived for two state neurons, we will now outline similar methods for the design of neural networks with multi-level functions. For convenience we repeat the problem of associative memories at this point.

**Problem 3.2** Given  $m$  vectors which are the columns of the matrix  $Y = [y^{(1)} \dots y^{(m)}]$ , find  $T$  and  $b$  such that the columns of  $Y$  are stored as fixed points in the neural network model (3.1).

From the equilibria constraints (3.6), we note that if we can find  $T$  and  $b$  such that  $Ty^{(j)} + b = y^{(j)}$ ,  $j = 1, 2, \dots, m$ , then clearly all the equilibria constraints are satisfied. So our problem reduces to finding a solution for the matrix equation  $TY + B = Y$ , where  $B = [b \dots b] \in \mathcal{R}^n$ .

If  $\text{rank}(Y) = m$  then the projection learning

$$T = YY^\dagger, \quad Y^\dagger = (Y^T Y)^{-1} Y^T, \quad b = 0 \quad (3.9)$$

can be used to synthesize the neural network. Clearly, for this choice of parameters  $Ty = y$  for  $y = y^{(j)}$ ,  $j = 1, 2, \dots, m$ , thus equilibria constraints (3.6) are satisfied. However, for this choice of system parameters we also have  $T(-y) = (-y)$  for each  $y = y^{(j)}$ ,  $j = 1, 2, \dots, m$ , which means that negatives are also stored. To get rid of the negatives, first note that the set of constraints

$$Ty^{(j)} + b = y^{(j)}, \quad j = 1 \dots m$$

is equivalent to the set of constraints

$$T(y^{(j)} - y^{(m)}) = y^{(j)} - y^{(m)}, \quad j = 1 \dots m - 1, \quad b = y^{(m)} - Ty^{(m)}$$

Therefore we may let  $\bar{Y} = [y^{(1)} - y^{(m)} \dots y^{(m-1)} - y^{(m)}]$  and again by the projection learning rule we obtain the solution

$$T = \bar{Y}\bar{Y}^\dagger, \quad \bar{Y}^\dagger = (\bar{Y}^T \bar{Y})^{-1} \bar{Y}^T, \quad b = y^{(m)} - Ty^{(m)} \quad (3.10)$$

However, if  $\text{rank}(Y) \neq m$ , it is clear that we cannot use the projection learning rule. Therefore we should look for a more general result. Again let us consider the case  $b = 0$  first. In this case, the general solution to  $TY = Y$  is given as

$$T = U_1 U_1^T + X U_2^T \quad (3.11)$$

where  $U_1 = [u_1 \dots u_k] \in \mathcal{R}^{n \times k}$  and  $U_2 = [u_{k+1} \dots u_n] \in \mathcal{R}^{n \times (n-k)}$  are orthonormal matrices,  $X \in \mathcal{R}^{n \times (n-k)}$  is an arbitrary matrix and  $k$  is the rank of the matrix  $Y$ . If we denote the space spanned by the columns of  $Y$  as  $\mathcal{L}$ , then columns of  $U_1$  form a basis for  $\mathcal{L}$  and columns of  $U_2$  form a basis for  $\mathcal{L}^\perp$ . Clearly  $U_1$  and  $U_2$  can be obtained from the singular value decomposition of



Y. We can use the matrix  $X$  for decreasing the number of spurious states. A particular choice is  $X = -\tau U_2$  with  $\tau > 0$  arbitrary. For this choice, we obtain the system parameters as

$$T = U_1 U_1^T - \tau U_2 U_2^T, \quad b = 0 \quad (3.12)$$

Now we show that with the choice of (3.12), the vectors that are in the column space of  $U_2$  are not equilibria. Assume that a vector  $v$  is in the column space of  $U_2$ , i.e., there exists some vector  $z$  such that  $v = U_2 z$ . Then

$$Tv = U_1 U_1^T U_2 z - \tau U_2 U_2^T U_2 z = -\tau U_2 z = -\tau v$$

Since  $\tau > 0$ , equilibria constraints (3.6) are not satisfied, therefore  $v$  is not an equilibrium. In this way, we decrease the number of spurious states.

Using the solution (3.12), we always store the negatives of patterns in addition to the desired patterns. To get rid of this situation, we can use the bias term. Using the same trick we did for projection learning rule, we may let  $\bar{Y} = [y^{(1)} - y^{(m)} \dots y^{(m-1)} - y^{(m)}]$  and apply the same procedure we applied above and obtain the solution

$$T = \bar{U}_1 \bar{U}_1^T - \tau \bar{U}_2 \bar{U}_2^T, \quad b = y^{(m)} - T y^{(m)} \quad (3.13)$$

where  $\bar{U}_1 = [\bar{u}_1 \dots \bar{u}_r] \in \mathcal{R}^{(n-1) \times r}$  and  $\bar{U}_2 = [\bar{u}_{r+1} \dots \bar{u}_{n-1}] \in \mathcal{R}^{(n-1) \times (n-r-1)}$  are orthonormal matrices and  $r$  is the rank of the matrix  $\bar{Y}$ . If the space spanned by the columns of  $\bar{Y}$  is denoted by  $\bar{\mathcal{L}}$ , then columns of  $\bar{U}_1$  form an orthonormal basis for  $\bar{\mathcal{L}}$  and columns of  $\bar{U}_2$  form an orthonormal basis for  $\bar{\mathcal{L}}^\perp$ , respectively. Again  $\bar{U}_1$  and  $\bar{U}_2$  can be obtained from the singular value decomposition of  $\bar{Y}$ .

Induced by the more general results given in (3.12) and (3.13), we can enhance the result of projection learning rule by adding a term similar to the second term in (3.12) and (3.13). These solutions are given as

$$T = YY^\dagger - \tau(I_n - YY^\dagger), \quad Y^\dagger = (Y^T Y)^{-1} Y^T, \quad b = 0 \quad (3.14)$$

$$T = \bar{Y}\bar{Y}^\dagger - \tau(I_n - \bar{Y}\bar{Y}^\dagger), \quad \bar{Y}^\dagger = (\bar{Y}^T \bar{Y})^{-1} \bar{Y}^T, \quad b = y^{(m)} - T y^{(m)} \quad (3.15)$$

Now we will illustrate the design procedures we proposed above and compare the results with the existing design methods for the binary-state neurons.

**Example 1** Assume that we have the columns of the following matrix to be stored in a neural network using the 4-level quantizer shown in Figure 3.3.

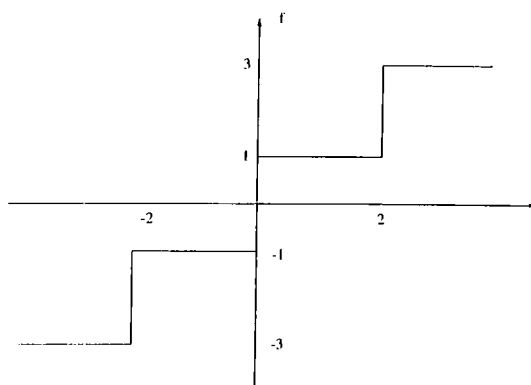


Figure 3.3: 4 level quantizer

$$Y = \begin{bmatrix} -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & -1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ -1 & 1 & 1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & 1 \end{bmatrix}$$

Since the 4-level quantizer allows each neuron output to have 4 distinct values, columns of  $Y$  can be stored in blocks of size 2. By using the code

$$\begin{bmatrix} -1 \\ -1 \end{bmatrix} \rightarrow -3 \quad \begin{bmatrix} -1 \\ 1 \end{bmatrix} \rightarrow -1 \quad \begin{bmatrix} 1 \\ -1 \end{bmatrix} \rightarrow 1 \quad \begin{bmatrix} 1 \\ 1 \end{bmatrix} \rightarrow 3$$

the memory matrix becomes

$$\bar{Y} = \begin{bmatrix} -1 & 3 & -1 & 3 \\ -1 & -3 & 3 & -1 \\ 3 & 1 & -3 & -1 \\ -1 & 1 & 1 & -1 \\ 3 & 3 & 1 & 3 \end{bmatrix}$$

Now we will apply various methods:

1. Direct solution of the equilibria constraints: This method yields the system parameters

$$T = \begin{bmatrix} 0.5394 & -0.2351 & -0.2106 & -0.0245 & 0.1372 \\ -0.2459 & 0.4303 & -0.3279 & 0.0410 & -0.0615 \\ -0.1689 & -0.2860 & 0.5811 & -0.1261 & 0.0586 \\ -0.0380 & -0.0489 & -0.0380 & 0.4892 & 0.0054 \\ 0.0598 & 0.0054 & 0.0598 & -0.0543 & 0.7772 \end{bmatrix}, \quad b = 0$$

with 136 asymptotically stable equilibria. The weight matrix is not symmetric and all the initial conditions converge to some equilibria.

2. Projection Learning Rule: Using  $T = \bar{Y}\bar{Y}^\dagger$  where  $\bar{Y}^\dagger = (\bar{Y}^T\bar{Y})^{-1}\bar{Y}^T$ , we get

$$T = \begin{bmatrix} 0.7337 & -0.3424 & -0.2663 & -0.0761 & 0.0380 \\ -0.3424 & 0.5598 & -0.3424 & -0.0978 & 0.0489 \\ -0.2663 & -0.3424 & 0.7337 & -0.0761 & 0.0380 \\ -0.0761 & -0.0978 & -0.0761 & 0.9783 & 0.0109 \\ 0.0380 & 0.0489 & 0.0380 & 0.0109 & 0.9946 \end{bmatrix}, \quad b = 0$$

This method yields 502 stable equilibria. The weight matrix is symmetric and the negatives of the desired patterns are stored automatically. Another attribute of the network is that it is globally stable.

3. Enhanced Projection Learning Rule:  $T = \bar{Y}\bar{Y}^\dagger - \tau(I - \bar{Y}\bar{Y}^\dagger)$ . For different values of  $\tau$  we analyze the network characteristics and summarize the

number of stable equilibria and the number of limit cycles in Table 3.1. Below we give the weight matrices for  $\tau = 1$  and  $\tau = 10$ . For  $\tau > 10$ , we can not decrease the number of spurious states any further.

$$\tau = 1, T = \begin{bmatrix} 0.2731 & -0.5775 & -0.4769 & -0.1006 & 0.1752 \\ -0.5883 & -0.0099 & -0.6703 & -0.0568 & -0.0126 \\ -0.4352 & -0.6284 & 0.3148 & -0.2022 & 0.0966 \\ -0.1141 & -0.1468 & -0.1141 & 0.4675 & 0.0163 \\ 0.0978 & 0.0543 & 0.0978 & -0.0434 & 0.7718 \end{bmatrix}, b = 0$$

$$\tau = 10, T = \begin{bmatrix} -2.1236 & -3.6590 & -2.8736 & -0.7854 & 0.5176 \\ -3.6698 & -3.9719 & -3.7518 & -0.9373 & 0.4276 \\ -2.8320 & -3.7099 & -2.0820 & -0.8870 & 0.4390 \\ -0.7989 & -1.0272 & -0.7989 & 0.2718 & 0.1141 \\ 0.4402 & 0.4945 & 0.4402 & 0.0544 & 0.7229 \end{bmatrix}, b = 0$$

If we use the degree of freedom in  $b$ , we hope to enhance our results. For that reason we form the new matrix by subtracting the columns of  $\bar{Y}$  from the last column. Call this new matrix  $\tilde{Y}$ . We will choose  $b$  as  $b = \bar{y}_4 - T\bar{y}_4$ . Now we will apply various design methods.

1. Projection Learning Rule: Using  $T = \tilde{Y}\tilde{Y}^\dagger$ ,  $b = \bar{y}_4 - T\bar{y}_4$ , the system parameters are computed as

$$T = \begin{bmatrix} 0.725 & -0.325 & -0.275 & -0.050 & 0.125 \\ -0.325 & 0.525 & -0.325 & -0.150 & -0.125 \\ -0.275 & -0.325 & 0.725 & -0.050 & 0.125 \\ -0.050 & -0.150 & -0.050 & 0.900 & -0.250 \\ 0.125 & -0.125 & 0.125 & -0.250 & 0.125 \end{bmatrix}, b = \begin{bmatrix} -0.20 \\ 0.40 \\ -0.20 \\ 0.60 \\ 2.00 \end{bmatrix}$$

This set of parameters yield a globally stable neural network with 134 equilibria.

2. Enhanced Projection Learning Rule:  $T = \tilde{Y}\tilde{Y}^\dagger - \tau(I - \tilde{Y}\tilde{Y}^\dagger)$ ,  $b = \bar{y}_4 - T\bar{y}_4$ . For different values of  $\tau$  we tabulate the number of stable states and the number of limit cycles in Table 3.2. Below we give the system parameters

for  $\tau = 3$ . As can be seen from Table 3.2, we can not decrease the number of spurious states below 10.

$$\tau = 3, \quad T = \begin{bmatrix} -0.10 & -1.30 & -1.10 & -0.20 & 0.50 \\ -1.30 & -0.90 & -1.30 & -0.60 & -0.50 \\ -1.10 & -1.30 & -0.10 & -0.20 & 0.50 \\ -0.20 & -0.60 & -0.20 & 0.60 & -1.00 \\ 0.50 & -0.50 & 0.50 & -1.00 & -2.50 \end{bmatrix}, \quad b = \begin{bmatrix} -0.80 \\ 1.60 \\ -0.80 \\ 2.40 \\ 8.00 \end{bmatrix}$$

Now we will carry out the design in the case of binary state neurons.

1. Projection learning rule:  $T = YY^\dagger$ , where  $Y^\dagger = (Y^TY)^{-1}Y^T$  with  $b = 0$ , yields a symmetric and globally stable neural network with 40 stable states.

2.  $T = \tau_1YY^\dagger - \tau_2(I - YY^\dagger)$ ,  $b = 0$ . With the choice of  $\tau_1 = 1$  and  $\tau_2 = 1$ , we obtain a neural network with 8 stable states, 4 of which are the negatives of the desired patterns. This network is not globally stable, i.e. there are limit cycles.

To get rid of the negatives, the only possible way is to use a bias term  $b$ . For this purpose, we form the matrix  $\hat{Y}$  by subtracting columns of the matrix  $Y$  from the last column. Then  $b$  can be computed as  $b = y_A - Ty_A$ .

$$\hat{Y}^T = \begin{bmatrix} -2 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2 & 2 & -2 & 2 & -2 & 0 & 0 \\ -2 & 0 & 2 & 0 & 0 & -2 & 2 & -2 & 0 & -2 \end{bmatrix}$$

1. Projection learning rule:  $T = \hat{Y}\hat{Y}^\dagger$  where  $\hat{Y}^\dagger = (\hat{Y}^T\hat{Y})^{-1}\hat{Y}^T$ . This method yields a globally stable neural network with 16 equilibria.

2.  $T = \tau_1\hat{Y}\hat{Y}^\dagger - \tau_2(I - \hat{Y}\hat{Y}^\dagger)$  with  $\tau_1 = 1$  and  $\tau_2 = 1$  yields a globally stable neural network with the desired memory vectors only.

As we mentioned earlier, the advantage of using multi-level functions in neural networks is to decrease the number of connections, however as we see from the example, it has a major disadvantage. Since we decreased the dimension of the state space, we gave up using some of the freedom we had, thus increased the possibility of more spurious states.

$\tau$	1	3	5	10
# of equilibria	64	28	20	14
# of cycles	0	128	246	332

Table 3.1: Properties of the neural network for different  $\tau$ , with  $T = \tilde{Y}\tilde{Y}^\dagger - \tau(I - \tilde{Y}\tilde{Y}^\dagger)$  and  $b = 0$

$\tau$	1	3	5	10
# of equilibria	37	14	14	14
# of cycles	107	223	230	282

Table 3.2: Properties of the neural network for different  $\tau$ , with  $T = \tilde{Y}\tilde{Y}^\dagger - \tau(I - \tilde{Y}\tilde{Y}^\dagger)$  and  $b = \bar{y}_4 - T\bar{y}_4$

# Chapter 4

## Decomposition Methods

Decomposition-aggregation techniques have been used extensively for the analysis and solution of large-scale problems. The conjecture of these techniques is to obtain the global solution to a large-scale problem by dividing the system into a number of smaller subsystems and then combining the individual solutions. The decomposition can be carried out in two ways, disjoint decomposition where the subsystems carry very weak interconnections that do not affect the overall system performance and overlapping decomposition where the subsystems share information with other subsystems, which may affect the overall system performance.

In this chapter, we first deal with disjoint decompositions and cases in which it can be helpful. Then we review the literature on overlapping decompositions and apply the method to the design of neural networks. Considering discrete-time neural network model, we first develop the necessary tools for expansions and contractions, then we give algorithms for decomposing a set of equilibria into two smaller sets of equilibria and for designing these smaller dimension neural networks. We finally conclude by applying the idea to the continuous-time neural networks.

## 4.1 Disjoint Decompositions

Consider the discrete-time neural network model

$$x(k+1) = f(Wx(k) + b) , \quad x(k_0) = x_0 \quad (4.1)$$

Partitioning the state vector  $x(k) \in \mathcal{R}^n$  as  $x = [x_1^T \ x_2^T \ \dots \ x_N^T]^T$ , we induce a disjoint decomposition of the matrices  $W$  and  $b$  as

$$W = \begin{bmatrix} W_{11} & W_{12} & W_{1N} \\ W_{21} & W_{22} & W_{2N} \\ \dots & \dots & \dots \\ W_{N1} & W_{N2} & W_{NN} \end{bmatrix} , \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_N \end{bmatrix} \quad (4.2)$$

and the above system can be represented as an interconnection of  $N$  subsystems:

$$x_i(k+1) = f_i(W_{ii}x_i(k) + b_i + \sum_{j=1, j \neq i} W_{ij}x_j(k)) , \quad i = 1, 2, \dots, N \quad (4.3)$$

To achieve our main goal of reducing the design of the neural network into designing lower dimension neural networks, we should somehow have the above subsystems decoupled from each other. This is possible only if the off-diagonal elements of  $W$  are weak enough not to affect the equilibria of the overall system. However, if this is the situation we can also design the neural network using  $W_{ij} = 0 \ i \neq j$ , that is we can reduce the design into  $N$  independent designs. Now let us illustrate the concept with a simple example.

Consider a stick of three pieces. Assume that each piece can either be white or black. Let our desired colored sticks be (BBB, BBW), where “B” denotes black stick piece and “W” denotes white stick piece. It is clear from the desired patterns that whatever the color of the 3rd stick piece is, the color of the first and the second piece is black, therefore all 3 pieces are independent of each other, which means that we can design a one-neuron neural network for each piece and then combine these individual solutions. Now let us add (WWW) to the desired set above. With this addition, all the stick pieces can assume both colors and we cannot have an equivalent disjoint decomposition. The best disjoint decomposition should have the first and second states in the first subsystem and third state in the second subsystem. Even in this case, our global solution tends to store (WWB) in addition to the three desired stick types mentioned above.



From this simple example, we see that we can use disjoint decomposition either in trivial cases or in finding suboptimal solutions to the overall system design.

## 4.2 Overlapping Decompositions

As we mentioned in the previous section, disjoint decomposition is not helpful other than in trivial cases. In such situations, allowing the subsystems to share information of the system provides some flexibility. In the colored stick example overlapping the information in the first or in the second state, we are able to transform the state space to a disjoint one in a larger space. Designing these disjoint systems and by back transformation, we obtain a solution with no spurious patterns.

With the motivation of having disjoint subsystems in an expanded space, the concept of overlapping decompositions has been used in several practical situations. In [26, 27], Ikeda *et al* used this scheme for constructing decentralized optimal control strategies, while Calvet in [28] applied the idea for the solution of a system of linear equations. In [29], the authors proposed a graph-theoretic decomposition procedure to decompose a large-scale system into weakly coupled overlapping components.

In this section we first review some results on overlapping decompositions of dynamic systems [26, 27, 29, 30] and in the remainder of the chapter we apply this idea to the design of neural networks.

### 4.2.1 Linear Systems

#### Expansions and Contractions

Consider two systems  $\mathcal{S}$  and  $\tilde{\mathcal{S}}$  described by

$$\mathcal{S} : \quad \dot{x} = Ax \quad (4.4)$$

and

$$\tilde{\mathcal{S}} : \quad \dot{\tilde{x}} = \tilde{A}\tilde{x} \quad (4.5)$$

where  $x \in \mathcal{R}^n$  and  $\tilde{x} \in \mathcal{R}^{\tilde{n}}$  with  $\tilde{n} > n$ . Let the solutions of (4.4) and (4.5) corresponding to the initial conditions  $x_0$  and  $\tilde{x}_0$  be denoted by  $x(t, x_0)$  and  $\tilde{x}(t, \tilde{x}_0)$ . Suppose there exist constant matrices  $U$  and  $V$  of respective dimensions  $n \times \tilde{n}$  and  $\tilde{n} \times n$ , such that  $UV = I_n$  and

$$U\tilde{x}(t, Vx_0) = x(t, x_0) \quad (4.6)$$

for all  $t \in \mathcal{R}$  and  $x_0 \in \mathcal{R}^n$ . Then  $\tilde{\mathcal{S}}$  is called an expansion of  $\mathcal{S}$  and  $\mathcal{S}$  a contraction or restriction of  $\tilde{\mathcal{S}}$ .

To derive conditions for expansions and contractions in terms of matrices we write  $\hat{A} = VAU + M$  where  $M$  is a complementary matrix of appropriate dimensions. This matrix represents a freedom in choosing an expansion. From (4.6) it is clear that  $\tilde{\mathcal{S}}$  is an expansion of  $\mathcal{S}$  if and only if  $U\hat{A}^iV = A^i$ ,  $i = 1, 2, \dots$ , or equivalently, if and only if  $UM^iV = 0$ ,  $i = 1, 2, \dots$ . Two particular cases are of special interest:

1. Type I expansions :  $MV=0$ . In this case, in addition to (4.6), we also have  $\tilde{x}(t, Vx_0) = Vx(t, x_0)$  for all  $t \in \mathcal{R}$ ,  $x_0 \in \mathcal{R}^n$ .
2. Type II expansions :  $UM=0$ . In this case, we have  $U\tilde{x}(t, \tilde{x}_0) = x(t, U\tilde{x}_0)$ , for all  $t \in \mathcal{R}$ ,  $\tilde{x}_0 \in \mathcal{R}^{\tilde{n}}$ .

## Overlapping Decompositions

Let us partition the state  $x$  of the system  $\mathcal{S}$  in (4.4) into three vector components as  $x = [x_1^T \ x_2^T \ x_3^T]^T$  the dimensions of which are such that  $n_1+n_2+n_3 = n$ . The overlapping decomposition has a representation

$$S : \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{x}_3(t) \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & \vdots & A_{13} \\ A_{21} & \boxed{A_{22}} & \vdots & A_{23} \\ \vdots & \vdots & \boxed{A_{32}} & \vdots \\ A_{31} & \vdots & A_{32} & A_{33} \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \end{bmatrix} \quad (4.7)$$

where the dotted lines indicate the portions of the system matrix  $A$  induced by the overlapping partition  $(x_1, x_2)$  and  $(x_2, x_3)$  of the state  $x$ . The decomposition of  $\mathcal{S}$  above is an overlapping decomposition into two subsystems and can easily be extended to cover any number of interconnected overlapping subsystems.

Defining the new state as  $\tilde{x} = [\tilde{x}_1^T \ \tilde{x}_2^T]^T$  of the system  $\tilde{\mathcal{S}}$  where  $\tilde{x}_1 = [x_1^T \ x_2^T]^T$  and  $\tilde{x}_2 = [x_2^T \ x_3^T]^T$ , the new state  $\tilde{x}$  is related to  $x$  as  $\tilde{x} = Vx$  where the  $\tilde{n} \times n$

transformation matrix  $V$  is

$$V = \begin{bmatrix} I_1 & 0 & 0 \\ 0 & I_2 & 0 \\ 0 & I_2 & 0 \\ 0 & 0 & I_3 \end{bmatrix} \quad (4.8)$$

and  $I_1, I_2, I_3$  are the identity matrices with dimensions compatible with the components  $x_1, x_2, x_3$ . Choosing the matrices

$$U = \begin{bmatrix} I_1 & 0 & 0 & 0 \\ 0 & \frac{1}{2}I_2 & \frac{1}{2}I_2 & 0 \\ 0 & 0 & 0 & I_3 \end{bmatrix} \quad M = \begin{bmatrix} 0 & \frac{1}{2}A_{12} & -\frac{1}{2}A_{12} & 0 \\ 0 & \frac{1}{2}A_{22} & -\frac{1}{2}A_{22} & 0 \\ 0 & -\frac{1}{2}A_{22} & \frac{1}{2}A_{22} & 0 \\ 0 & -\frac{1}{2}A_{32} & \frac{1}{2}A_{32} & 0 \end{bmatrix} \quad (4.9)$$

a possible Type I expansion system has the form

$$\tilde{S} : \begin{bmatrix} \dot{\tilde{x}}_1(t) \\ \dot{\tilde{x}}_2(t) \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & \vdots & 0 & A_{13} \\ A_{21} & A_{22} & \vdots & 0 & A_{23} \\ \hline A_{21} & 0 & \vdots & A_{22} & A_{23} \\ A_{31} & 0 & \vdots & A_{32} & A_{33} \end{bmatrix} \begin{bmatrix} \tilde{x}_1(t) \\ \tilde{x}_2(t) \end{bmatrix} \quad (4.10)$$

## 4.2.2 Nonlinear Systems

The inclusion concept can be generalized to nonlinear systems with more constraints than in the linear case [30]. Consider two dynamic systems

$$\mathcal{S} : \quad \dot{x} = f(t, x) \quad (4.11)$$

and

$$\tilde{\mathcal{S}} : \quad \dot{\tilde{x}} = \tilde{f}(t, \tilde{x}) \quad (4.12)$$

where  $x(t) \in \mathcal{R}^n$  and  $\tilde{x}(t) \in \mathcal{R}^{\tilde{n}}$  with  $\tilde{n} > n$  are the states of  $\mathcal{S}$  and  $\tilde{\mathcal{S}}$ . The functions  $f : \mathcal{R} \times \mathcal{R}^n \rightarrow \mathcal{R}^n$  and  $\tilde{f} : \mathcal{R} \times \mathcal{R}^{\tilde{n}} \rightarrow \mathcal{R}^{\tilde{n}}$  are assumed to be sufficiently smooth, so that solutions  $x(t; t_0, x_0)$  and  $\tilde{x}(t; t_0, \tilde{x}_0)$  of  $\mathcal{S}$  and  $\tilde{\mathcal{S}}$  exist and are unique for all initial conditions  $(t_0, x_0) \in \mathcal{R} \times \mathcal{R}^n$  and  $(t_0, \tilde{x}_0) \in \mathcal{R} \times \mathcal{R}^{\tilde{n}}$  and for all  $t \geq t_0$ . We use the linear transformations

$$\tilde{x} = Vx \quad , \quad x = U\tilde{x} \quad (4.13)$$

where  $V$  is an  $\tilde{n} \times n$  constant matrix with full column rank and  $U$  is an  $n \times \tilde{n}$  matrix with full row rank.

**Definition 4.1** The system  $\mathcal{S}$  is said to be included in the system  $\tilde{\mathcal{S}}$  if there exist constant matrices  $U$  and  $V$  of dimensions  $n \times \tilde{n}$  and  $\tilde{n} \times n$  such that  $UV = I_n$  and for any  $(t_0, x_0) \in \mathcal{R} \times \mathcal{R}^n$ ,  $\tilde{x}_0 = Vx_0$  implies

$$x(t; t_0, x_0) = U\tilde{x}(t; t_0, \tilde{x}_0) \quad , \quad t \geq t_0 \quad (4.14)$$

To derive conditions for inclusion, represent the function as

$$\tilde{f}(t, \tilde{x}) = Vf(t, U\tilde{x}) + \tilde{m}(t, \tilde{x}) \quad (4.15)$$

where  $\tilde{m} : \mathcal{R} \times \mathcal{R}^{\tilde{n}} \rightarrow \mathcal{R}^{\tilde{n}}$  is called a complementary function. For  $\tilde{\mathcal{S}}$  to include  $\mathcal{S}$ ,  $\tilde{m}$  is required to satisfy the restrictions stated in the following theorem:

**Theorem 4.1** [30]  $\tilde{\mathcal{S}}$  includes  $\mathcal{S}$  if either

$$i) \quad \tilde{m}(t, Vx) = 0 \quad , \quad \forall (t, x) \in \mathcal{R} \times \mathcal{R}^n, \text{ or} \quad (4.16)$$

$$ii) \quad U\tilde{m}(t, \tilde{x}) = 0 \quad , \quad \forall (t, \tilde{x}) \in \mathcal{R} \times \mathcal{R}^{\tilde{n}} \quad (4.17)$$

hold.

Moreover, in [30] it is shown that if the equilibrium points of the system  $\mathcal{S}$  are preserved under the transformation  $\tilde{x} = Vx$ , i.e.  $\tilde{m}(t, Vx) = 0$  at the equilibrium points of  $\mathcal{S}$ , then the stability of the equilibrium points of the system  $\tilde{\mathcal{S}}$  imply the stability of the equilibrium points of the system  $\mathcal{S}$ .

## 4.3 Application to Discrete-Time Neural Networks

Consider the system  $\mathcal{S}$  described by

$$\mathcal{S} : \quad x(k+1) = f(Wx(k) + b) \quad , \quad x(k_0) = x_0 \quad (4.18)$$

where  $x(k) \in \mathcal{R}^n$  is the state vector at instant  $k$ ,  $W \in \mathcal{R}^{n \times n}$  represents the interconnection structure,  $b \in \mathcal{R}^n$  is the bias term and  $f(x) =$

$[f_1(x_1) \dots f_n(x_n)]^T \in \mathcal{R}^n$  whose components use the same activation function. We associate with this system another system  $\hat{\mathcal{S}}$  described by

$$\hat{\mathcal{S}} : \quad \tilde{x}(k+1) = \tilde{f}(\tilde{W}\tilde{x}(k) + \tilde{b}) \quad , \quad \tilde{x}(k_0) = \tilde{x}_0 \quad (4.19)$$

where  $\tilde{x}(k) \in \mathcal{R}^{\tilde{n}}$ ,  $\tilde{W} \in \mathcal{R}^{\tilde{n} \times \tilde{n}}$ ,  $\tilde{b} \in \mathcal{R}^{\tilde{n}}$  with  $\tilde{n} \geq n$  and  $\tilde{f}(\tilde{x}) = [f_1(\tilde{x}_1) \dots f_{\tilde{n}}(\tilde{x}_{\tilde{n}})]^T \in \mathcal{R}^{\tilde{n}}$  whose components use the same activation function. Let  $x(k; k_0, x_0)$  and  $\tilde{x}(k; k_0, \tilde{x}_0)$  denote the solutions of the systems  $\mathcal{S}$  and  $\hat{\mathcal{S}}$ , respectively.

**Definition 4.2** The system  $\mathcal{S}$  is said to be included in the system  $\hat{\mathcal{S}}$  if there exist constant matrices  $U$  and  $V$  of dimensions  $n \times \tilde{n}$  and  $\tilde{n} \times n$  such that  $UV = I_n$  and for any initial state  $x_0$  of  $\mathcal{S}$ , we have

$$x(k; k_0, x_0) = U\tilde{x}(k; k_0, Vx_0) \quad , \quad k \geq k_0 \quad (4.20)$$

To derive conditions for expansions and contractions, we let

$$\tilde{W} = VWU + M \quad , \quad \tilde{b} = Vb + n \quad (4.21)$$

where  $M$  and  $n$  are complementary matrices.

**Theorem 4.2** *The system  $\hat{\mathcal{S}}$  includes the system  $\mathcal{S}$  if either*

$$(i) \quad MV = 0 \quad , \quad n = 0 \quad , \quad Vf(x) = \tilde{f}(Vx), \quad \text{or} \quad (4.22)$$

$$(ii) \quad UM = 0 \quad , \quad Un = 0 \quad , \quad U\tilde{f}(\tilde{x}) = f(U\tilde{x}) \quad (4.23)$$

*hold.*

**Proof :** We give only the proof of (i), as the proof of (ii) is similar. From (4.18), (4.19) and (4.22) it follows that if  $\tilde{x}(k) = Vx(k)$ , then

$$\begin{aligned} \tilde{x}(k+1) &= \tilde{f}((VWU + M)Vx(k) + (Vb + n)) \\ &= \tilde{f}(V(Wx(k) + b)) \\ &= Vf(Wx(k) + b) \\ &= Vx(k+1) \end{aligned}$$

Then, by induction on  $k$ , we have that  $\tilde{x}_0 = Vx_0$  implies  $\tilde{x}(k, k_0, \tilde{x}_0) = Vx(k, k_0, x_0)$  for all  $k \geq k_0$  and all  $x_0 \in \mathcal{R}^n$ , so that  $U\tilde{x}(k, k_0, Vx_0) = x(k, k_0, x_0)$ .

### 4.3.1 Overlapping Decompositions of Neural Networks

The purpose of using overlapping decompositions in solving the associative memory problem is to reduce the computational complexity of the design procedure at the expense of some increase in the dimensionality. In doing so, however, we must take extreme care to make sure that the solution of the expanded problem can be contracted to a solution of the original problem. This puts some limitations on the type of expansions we can use as we explain below.

Suppose that a neural network  $\mathcal{S} = (f, W, b)$  is designed to have a set  $\mathcal{X}_e$  of stable equilibria corresponding to a set of patterns to be stored. Let  $\tilde{\mathcal{S}} = (\tilde{f}, \tilde{W}, \tilde{b})$  be a Type I expansion of  $\mathcal{S}$  satisfying (4.22). Then it is easy to show that for any  $x \in \mathcal{X}_e$ ,  $Vx \in \tilde{\mathcal{X}}_e$  (set of stable equilibria of  $\tilde{\mathcal{S}}$ ). Hence,  $U\tilde{\mathcal{X}}_e \supset \mathcal{X}_e$ , that is, the desired set of patterns can be extracted from the equilibria of the expanded system. On the other hand, if we use a Type II expansion satisfying (4.23), then all we can guarantee is that  $U\tilde{\mathcal{X}}_e \subset \mathcal{X}_e$ , in which case all of the desired patterns may not be extracted from the equilibria of the expanded system. For this reason, we will use Type I expansions in the design of associative memories.

Consider the following overlapping decomposition of the neural network given in (4.18).

$$\mathcal{S} : \begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \end{bmatrix} = f \left( \begin{bmatrix} W_{11} & W_{12} & \vdots & W_{13} \\ W_{21} & W_{22} & \vdots & W_{23} \\ W_{31} & \vdots & W_{32} & W_{33} \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} + \begin{bmatrix} b_1 \\ \vdots \\ b_2 \\ \vdots \\ b_3 \end{bmatrix} \right) \quad (4.24)$$

where the dotted lines indicate the portions of the system matrix  $W$  induced by the overlapping partition  $(x_1, x_2)$  and  $(x_2, x_3)$  of the state  $x$ .

Defining the transformation matrix  $V$  as

$$V = \begin{bmatrix} I_1 & 0 & 0 \\ 0 & I_2 & 0 \\ 0 & 0 & I_3 \end{bmatrix} \quad (4.25)$$

where  $I_1, I_2, I_3$  are the identity matrices with dimensions compatible with the

components  $x_1, x_2, x_3$  and choosing

$$U = \begin{bmatrix} I_1 & 0 & 0 & 0 \\ 0 & \frac{1}{2}I_2 & \frac{1}{2}I_2 & 0 \\ 0 & 0 & 0 & I_3 \end{bmatrix}, \quad M = \begin{bmatrix} 0 & \frac{1}{2}W_{12} & -\frac{1}{2}W_{12} & 0 \\ 0 & \frac{1}{2}W_{22} & -\frac{1}{2}W_{22} & 0 \\ 0 & -\frac{1}{2}W_{22} & \frac{1}{2}W_{22} & 0 \\ 0 & -\frac{1}{2}W_{32} & \frac{1}{2}W_{32} & 0 \end{bmatrix}, \quad n = 0 \quad (4.26)$$

a possible expansion of  $\mathcal{S}$  is obtained as an interconnection of two subsystems described by

$$\tilde{\mathcal{S}}: \quad \tilde{x}_1(k+1) = \tilde{f}_1(\tilde{W}_1\tilde{x}_1(k) + \tilde{W}_{12}\tilde{x}_2(k) + \tilde{b}_1) \quad (4.27)$$

$$\tilde{x}_2(k+1) = \tilde{f}_2(\tilde{W}_2\tilde{x}_2(k) + \tilde{W}_{21}\tilde{x}_1(k) + \tilde{b}_2) \quad (4.28)$$

where  $\tilde{x}_1 = [x_1^T \ x_2^T]^T$ ,  $\tilde{x}_2 = [x_2^T \ x_3^T]^T$ ,

$$\tilde{W}_1 = \begin{bmatrix} W_{11} & W_{12} \\ W_{21} & W_{22} \end{bmatrix}, \quad \tilde{W}_{12} = \begin{bmatrix} 0 & W_{13} \\ 0 & W_{23} \end{bmatrix}, \quad \tilde{b}_1 = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \quad (4.29)$$

$$\tilde{W}_2 = \begin{bmatrix} W_{22} & W_{23} \\ W_{32} & W_{33} \end{bmatrix}, \quad \tilde{W}_{21} = \begin{bmatrix} W_{21} & 0 \\ W_{31} & 0 \end{bmatrix}, \quad \tilde{b}_2 = \begin{bmatrix} b_2 \\ b_3 \end{bmatrix} \quad (4.30)$$

and

$$\tilde{f}_1(\tilde{x}_1) = [f(\tilde{x}_{11}) \ \dots \ f(\tilde{x}_{1, n_1+n_2})]^T, \quad \tilde{f}_2(\tilde{x}_2) = [f(\tilde{x}_{21}) \ \dots \ f(\tilde{x}_{2, n_2+n_3})]^T$$

Clearly, if  $\tilde{W}_{12} = 0$  and  $\tilde{W}_{21} = 0$ , then the two subsystems are decoupled and therefore can be designed independently. This, however, puts some restrictions on the structures of the  $\tilde{W}_1$  and  $\tilde{W}_2$  matrices of the subsystems. They have to be of the form

$$\tilde{W}_1 = \begin{bmatrix} W_{11} & W_{12} \\ 0 & W_{22} \end{bmatrix}, \quad \tilde{W}_2 = \begin{bmatrix} W_{22} & 0 \\ W_{32} & W_{33} \end{bmatrix} \quad (4.31)$$

Now suppose that a set of desired memory vectors  $Y = [y^1 \ \dots \ y^m]$  where  $y^i \in \mathcal{R}^n$ ,  $i = 1, 2, \dots, m$ , are given. The following algorithm may be used for the design of the desired neural network by means of overlapping decompositions.

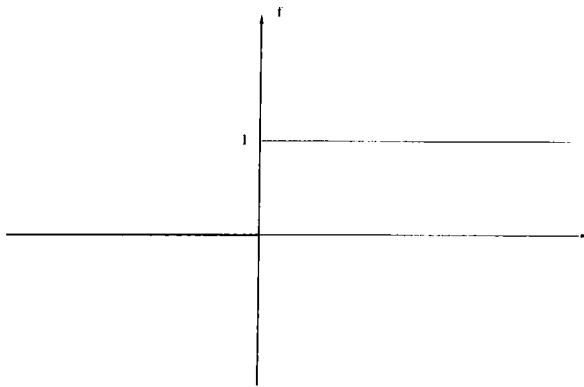


Figure 4.1: Threshold function

**Algorithm 4.1 (Divide and Design Algorithm)**

1. Find a transformation matrix  $V$  and expand the memory vectors as

$$\tilde{Y} = VY = \begin{bmatrix} \tilde{Y}_1 \\ \tilde{Y}_2 \end{bmatrix} \quad (4.32)$$

2. Design subnetworks with  $\tilde{W}_i$  as in (4.31) and  $\tilde{b}_i$  as in (4.29), (4.30) to store the memory matrices  $\tilde{Y}_i$ ,  $i = 1, 2$ .
3. Compute  $W$  and  $b$  by contraction.

**Example 1** Suppose that we want to store the columns of the following matrix as fixed attractors of the discrete-time neural network given in (4.18) using the activation function in Figure 4.1.

$$Y = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Suppose that we take  $V$  as



$$V = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ - & - & - & - & - \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

that is we overlap the 3rd row. Transforming the equilibria matrix  $Y$  by  $V$  we obtain

$$\tilde{Y} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ - & - & - & - & - \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Now the problem is reduced to solving for 2 neural networks of dimension 3 which have equilibria as the columns of the following matrices.

$$\tilde{Y}_1 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \quad \tilde{Y}_2 = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

Taking into account also the condition  $MV = 0$  with the equilibria constraints of the above subsystems, the following solution results in no spurious states for the subsystems.

$$\hat{W}_1 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}, \quad \tilde{b}_1 = \begin{bmatrix} -2.5 \\ -2.5 \\ -0.5 \end{bmatrix}, \quad \hat{W}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad \tilde{b}_2 = \begin{bmatrix} -0.5 \\ -2.5 \\ -2.5 \end{bmatrix}$$

Using

$$U = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad M = \begin{bmatrix} 0 & 0 & \frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & -\frac{1}{2} & \frac{1}{2} & 0 & 0 \end{bmatrix}$$

we can obtain the system parameters by contraction as

$$W = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} -2.5 \\ -2.5 \\ -0.5 \\ -2.5 \\ -2.5 \end{bmatrix}$$

We note that this solution results in a globally stable network with no spurious states.

Clearly solutions of Steps 1 and 2 in the “Divide and Design algorithm” (4.1) do not seem so trivial. Now we will try to supply solutions to these steps. For the solution of Step 2 we can use the following algorithm:

**Algorithm 4.2 (Design Algorithm)**

1. desiredset=tempset=Y; next=ok=1;
2. while ok=1
  - 2.1. if a solution to the constraints exist, feasible=1;
  - 2.2. else, feasible=0 , next=next+1.
  - 2.3. if feasible=1

**2.3.1.** find all equilibria  $\rightarrow$  eqset, and let unwantedset={ eqset-Y }

**2.3.2.** for i=1:length(unwantedset), for j=1:length(desiredset)

**2.3.2.1** count=1

**2.3.2.2** if  $HD^1(\text{unwantedset}(i), \text{desiredset}(j))=1$

**2.3.2.2.1** dl(count)=desiredset(j)

**2.3.2.2.2** t1(count)=unwantedset(i)

**2.3.2.2.3** count=count+1

end;

end; end;

**2.3.3.** desiredset=desiredset  $\cup$  dl(next)

**2.3.4.** tempset=tempset  $\cup$  t1(next)

**2.4.** else

**2.4.1.** desiredset=desiredset - dl(next)

**2.4.2.** tempset=tempset - t1(next)

end

**2.5.** if next=length(t1) ok=0 end

end

---

<sup>1</sup>Hamming distance

The solution to step 2.1 can be obtained using either simplex algorithm or Fourier elimination. The main idea behind the design algorithm (4.2) is as follows. First we find an initial solution for the equilibria constraints and compute the equilibria set for this solution. Then we try to send each spurious state in this equilibria set to a memory vector which is at unit Hamming distance and converges to a desired memory vector. We iterate until all the possible spurious states are sent to desired equilibria.

On the other hand, the decomposition is a critical process since the advantage of overlapping decomposition heavily depends on how large the overlapping block is. Therefore in our designs we want to decompose the equilibria set such that the overlapping block size is at most 2. Below we give an algorithm to identify the groups and the row to be overlapped.

**Input:** The desired memories matrix  $Y = [y^1 \dots y^m] \in \mathcal{R}^{n \times m}$

**Algorithm 4.3 (Decomposition Algorithm)**

For each row of  $Y$  do

1. Number the other rows from 1 to  $n - 1$
2. Let  $set = \emptyset$ ,  $otherset = \{1, 2, 3, \dots, n - 1\}$ . We assume that  $set$  and  $otherset$  with the overlapped row form the groups in the decomposition.
3.  $i = 1$ ,  $minimum = 2^n$
4. while ( $i < n - 1$ ) and ( $minimum > m$ )
  - 4.1. Set  $j = 1$
  - 4.2. while ( $j < \text{length}(\text{otherset})$ ) and ( $minimum > m$ )
    - 4.2.1.  $elt = \text{row of } Y \text{ corresponding to } j\text{-th element of otherset}$
    - 4.2.2.  $temp1 = set \cup \{elt\}$ ,  $temp2 = set - \{elt\}$
    - 4.2.3. Compute the number of equilibrium points corresponding to groups  $temp1$  and  $temp2$ .

4.2.4. If eqnum < minimum , minimum=eqnum and index=j

end

4.3. set=set  $\cup$  {index}, otherset=otherset-{index}

end

**Output:** The row to be overlapped and the groups.

It is clear that there may be no equivalent overlapping decomposition for the design of desired memories. In this case, either we take the optimal solution, or we revise the algorithm for the search of two or more overlapping rows. Now we illustrate the above algorithm with an example.

**Example 2** Assume that we want to store the columns of the following matrix as fixed attractors of the discrete-time neural network given in (4.18) using the threshold function given in Figure 4.1.

$$Y = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The algorithm is applied to each row of  $Y$  and the results are summarized in Tables (4.1)-(4.9). In these tables, the first row shows the elements of the variable “set” in the algorithm. Below the first row, the values of “templ” and number of equilibria are shown for each iteration of the inner loop in the algorithm. The indices denoted by “\*” are suboptimal solutions and those with “\*\*” are the optimal solutions. From the tables we conclude that we can decompose the equilibria set  $Y$  into 2 overlapping blocks in various ways. The groups with the overlapping block are shown below.

1. Groups:123,3456789 if row 3 is overlapped
2. Groups:12345,56789 if row 5 is overlapped
3. Groups:123456,6789 if row 6 is overlapped

Now we will design the neural network with the 5th row overlapped. Choose the expansion matrix  $V$  as

$$V = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Transforming the equilibria  $Y$  by  $V$  we obtain

$$\tilde{Y} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Now the problem is reduced to solving for 2 neural networks of dimension 5 which have equilibria as the columns of the following matrices.

$$\tilde{Y}_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}, \quad \tilde{Y}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Taking into account also the condition  $MV = 0$  with the equilibria constraints of the above subsystems, the following solution results in no spurious states for the subsystems.

$$\tilde{W}_1 = \begin{bmatrix} 1 & 0 & 2 & 0 & -1 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ -1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \tilde{b}_1 = \begin{bmatrix} -1.5 \\ -1.5 \\ -1.5 \\ -1.5 \\ -0.5 \end{bmatrix}$$

$$\tilde{W}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & -1 \\ -3 & 1 & 2 & 0 & 1 \\ -3 & 1 & 1 & 1 & 1 \\ -2 & 1 & 0 & 1 & 1 \end{bmatrix}, \quad \tilde{b}_2 = \begin{bmatrix} -0.5 \\ -0.5 \\ -1.5 \\ -1.5 \\ -1.5 \end{bmatrix}$$

Using

$$U = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, M = \begin{bmatrix} 0 & 0 & 0 & 0 & -\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & -\frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{3}{2} & -\frac{3}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{3}{2} & -\frac{3}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

we can obtain the system parameters by contraction as

$$W = \begin{bmatrix} 1 & 0 & 2 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & -3 & 1 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 & -3 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & -2 & 1 & 0 & 1 & 1 \end{bmatrix}, b = \begin{bmatrix} -1.5 \\ -1.5 \\ -1.5 \\ -1.5 \\ -0.5 \\ -0.5 \\ -1.5 \\ -1.5 \\ -1.5 \end{bmatrix} \quad (4.33)$$

The solution given in (4.33) results in a globally stable network with no spurious states.



$\emptyset$		1		16		162		1624		16245		162453	
1*	19	2	25	2*	25	3	31	3	29	3*	25	7*	21
2	21	3	33	3	31	4*	26	5*	25	7	25	8	21
3	19	4	25	4	25	5	26	7	29	8	25		
4	21	5	25	5	25	7	31	8	29				
5	21	6*	22	7	31	8	31						
6	19	7	25	8	31								
7	21	8	25										
8	21												

Table 4.1: 1st row overlapped

$\emptyset$		1		16		163		1632		16324		163245	
1*	13	2	20	2	22	2*	23	4*	23	5*	21	7	19
2	19	3	19	3*	21	4	23	5	23	7	24	8	19
3	20	4	20	4	22	5	23	7	27	8	24		
4	19	5	20	5	22	7	28	8	27				
5	19	6*	18	7	28	8	28						
6	17	7	20	8	28								
7	19	8	20										
8	19												

Table 4.2: 2nd row overlapped

$\emptyset$		2		21	
1	15	1*	13	3**	12
2*	13	3	15	3	18
3	13	4	18	4	18
4	17	5	18	5	16
5	17	6	16	7	18
6	15	7	18	8	18
7	17	8	18		
8	17				

Table 4.3: 3rd row overlapped

$\emptyset$		6		61		612		6123		61234		612345	
1	20	1 *	23	2*	26	3*	27	4*	26	5*	23	7	20
2	20	2	23	3	26	4	27	5	26	7	26	8	20
3	20	3	23	4	26	5	27	7	30	8	26		
4	20	4	23	5	26	7	32	8	30				
5	20	5	23	7	32	8	32						
6 *	18	7	30	8	32								
7	20	8	30										
8	20												

Table 4.4: 4th row overlapped

$\emptyset$		2		21		213	
1	16	1*	15	3*	14	4**	12
2*	14	3	15	4	16	5	18
3	16	4	18	5	19	6	16
4	14	5	18	6	17	7	18
5	16	6	16	7	19	8	18
6	14	7	18	8	19		
7	16	8	18				
8	16						

Table 4.5: 5th row overlapped

$\emptyset$		6		67	
1	17	1	18	1	20
2	15	2	16	2	18
3	17	3	18	3	20
4	15	4	16	4	18
5	17	5	18	5	20
6 *	13	7 *	15	8 **	12
7	15	8	15		
8	15				

Table 4.6: 6th row overlapped

$\emptyset$		2		21		213		2135		21356		213564	
1	19	1*	21	3*	23	4	27	4	24	4*	19	7	22
2*	17	3	21	4	28	5*	23	6*	21	7	25	8	22
3	19	4	27	5	23	6	23	7	28	8	25		
4	17	5	21	6	23	7	29	8	28				
5	19	6	21	7	28	8	29						
6	19	7	25	8	28								
7	22	8	25										
8	22												

Table 4.7: 7th row overlapped

$\emptyset$		8		82		827		8271		82713		827135	
1	19	1	20	1	22	1*	23	3*	23	4	24	4	19
2	17	2*	18	3	22	3	23	4	27	5*	21	6	17
3	19	3	20	4	28	4	28	5	23	6	21		
4	17	4	18	5	22	5	23	6	23				
5	19	5	20	6	22	6	23						
6	19	6	20	7*	21								
7	20	7	19										
8*	13												

Table 4.8: 8th row overlapped

$\emptyset$		2		21		213		2135		21356		213567	
1	20	1*	23	3*	26	4	32	4	30	4	26	4	20
2*	18	3	23	4	32	5*	27	6*	26	7*	23	8	18
3	20	4	30	5	26	6	27	7	26	8	23		
4	18	5	23	6	26	7	27	8	26				
5	20	6	23	7	26	8	27						
6	20	7	23	8	26								
7	20	8	23										
8	20												

Table 4.9: 9th row overlapped

# Chapter 5

## Examples

**Example 1** This example is taken from [6]. The desired memory vectors are the columns of the following matrix. We will again use the threshold function given in Figure 4.1.

$$Y = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

If we apply the decomposition algorithm (4.3) to  $Y$ , we see that there is no equivalent decomposition if we overlap only one row. However, if we apply the algorithm with two rows overlapped, we see that various equivalent decompositions exist. Among them, let us take the groups (1,3,4,5,7,8) and (2,3,4,6,9,10).

Then the expansion matrix becomes

$$V = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Transforming the equilibria matrix  $Y$  by  $V$  we obtain

$$\tilde{Y}^T = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Now the problem is reduced to solving for 2 neural networks of dimension 6 which have equilibria as the columns of the following matrices.

$$\tilde{Y}_1 = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}, \quad \tilde{Y}_2 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

Taking into account also the condition  $MV = 0$  with the equilibria constraints of the above subsystems, the following solution results in no spurious states for the subsystems.

$$\tilde{W}_1 = \begin{bmatrix} 1 & 0 & -2 & -1 & 1 & 2 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & -1.33 & -0.67 & 2 & -1 & 0.67 \\ -1 & 2 & -1 & 1 & 0 & -1 \\ -1 & -4 & 3 & 1 & -2 & -3 \end{bmatrix}, \tilde{b}_1 = \begin{bmatrix} -0.5 \\ -0.5 \\ -0.5 \\ 0.5 \\ 0.5 \\ 1.5 \end{bmatrix}$$

$$\tilde{W}_2 = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & -2 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & 0 & 0 \end{bmatrix}, \tilde{b}_2 = \begin{bmatrix} -0.5 \\ -0.5 \\ -0.5 \\ 0.5 \\ -0.5 \\ -0.5 \end{bmatrix}$$

Using

$$U = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

and

$$M = \begin{bmatrix} 0 & 0 & -1.00 & 0 & 0 & 0 & 0 & 0 & 1.00 & 0 & 0 & 0 \\ 0 & 0.50 & 0 & 0 & 0 & 0 & 0 & -0.50 & 0 & 0 & 0 & 0 \\ 0 & 0.50 & 0.50 & 0 & 0 & 0 & 0 & -0.50 & -0.50 & 0 & 0 & 0 \\ 0 & -0.665 & -0.335 & 0 & 0 & 0 & 0 & 0.665 & 0.335 & 0 & 0 & 0 \\ 0 & 1.00 & -0.50 & 0 & 0 & 0 & 0 & -1.00 & 0.50 & 0 & 0 & 0 \\ 0 & -2.00 & 1.50 & 0 & 0 & 0 & 0 & 2.00 & -1.50 & 0 & 0 & 0 \\ 0 & 0 & -0.50 & 0 & 0 & 0 & 0 & 0 & 0.500 & 0 & 0 & 0 \\ 0 & -0.50 & 0 & 0 & 0 & 0 & 0 & 0.50 & 0 & 0 & 0 & 0 \\ 0 & -0.50 & -0.50 & 0 & 0 & 0 & 0 & 0.50 & 0 & 0 & 0 & 0 \\ 0 & 1.00 & -0.50 & 0 & 0 & 0 & 0 & -1.00 & 0.5 & 0 & 0 & 0 \\ 0 & -0.50 & 0 & 0 & 0 & 0 & 0 & 0.50 & 0 & 0 & 0 & 0 \\ 0 & 0.50 & 0 & 0 & 0 & 0 & 0 & -0.50 & 0 & 0 & 0 & 0 \end{bmatrix}$$

we can obtain the system parameters by contraction as

$$W = \begin{bmatrix} 1 & 0 & 0 & -2 & -1 & 0 & 1 & 2 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & -1.33 & -0.67 & 2 & 0 & -1 & 0.67 & 0 & 0 \\ 0 & -1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 2 & -1 & 1 & 0 & 0 & -1 & 0 & 0 \\ -1 & 0 & -4 & 3 & 1 & 0 & -2 & -3 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad b = \begin{bmatrix} -0.5 \\ -0.5 \\ -0.5 \\ -0.5 \\ 0.5 \\ 0.5 \\ 0.5 \\ 1.5 \\ -0.5 \\ -0.5 \end{bmatrix} \quad (5.1)$$

The solution given in (5.1) results in a globally stable network with no spurious states. In Table (5.1) we compare the performances of various neural networks using different methods. The network parameters for the eigenstructure method and Perfetti's method can be found in [6] and [9], respectively.

For Lillo's method, we choose the matrices

$$D = \begin{bmatrix} 6.000 & -0.550 & -1.050 & 0.250 & 0.650 & 1.450 & 0.350 & -0.700 & 0.150 & -0.500 \\ -0.650 & 7.800 & 0.200 & -0.600 & 1.150 & -0.350 & 0.750 & 1.200 & -0.850 & -0.300 \\ 1.050 & 0.750 & 6.050 & 0.250 & -0.350 & -0.650 & -0.500 & -0.900 & 0.650 & 0.250 \\ -0.350 & -0.200 & -0.600 & 4.500 & -0.650 & -0.550 & -0.750 & 0.150 & 0.350 & 0.450 \\ 0.450 & 0.150 & -0.900 & 0.700 & 6.200 & -0.600 & -1.150 & 0.450 & -0.400 & 0.500 \\ 0.850 & -0.700 & 0.900 & -1.000 & 0.700 & 7.900 & 0.300 & 0.250 & 1.550 & 0.850 \\ 0.250 & -0.100 & 0.750 & -1.300 & -0.850 & -0.500 & 7.000 & 1.350 & 0.600 & -0.550 \\ -1.250 & -1.100 & 0.400 & 0.350 & -0.850 & -0.750 & 0.850 & 7.700 & 0.450 & 1.150 \\ 0.450 & 0.650 & -1.050 & 0.150 & -1.200 & 0.950 & 0.600 & 0.550 & 6.750 & 0.550 \\ -1.100 & 0.400 & 0.300 & -0.650 & 0.250 & 0.350 & 0.500 & -1.100 & 0.600 & 6.000 \end{bmatrix}$$

and

$$\Lambda = \begin{bmatrix} -8.330 & -0.120 & 0.520 & -0.940 & 0.290 & -0.430 & 0.960 & -0.230 & -1.340 & 1.040 \\ -1.080 & -15.570 & 1.070 & 0.960 & -1.430 & 1.420 & -0.670 & 0.710 & -1.200 & 0.990 \\ 0.850 & -1.460 & -12.080 & 0.320 & -1.450 & 0.140 & 0.380 & -1.410 & -0.560 & 0.230 \\ -0.540 & 0.450 & 0.940 & -10.030 & 0.710 & 0.560 & -1.110 & 0.760 & -0.550 & -1.490 \\ -1.190 & 0.210 & 1.150 & -0.860 & -9.400 & -0.780 & 1.130 & -1.230 & -0.370 & 0.280 \\ 0.910 & -0.330 & 0.440 & 0.890 & -1.310 & -11.520 & 0.280 & 1.380 & -0.510 & 1.090 \\ -0.750 & -0.080 & -1.020 & 1.110 & 0.640 & -1.380 & -12.230 & 0.870 & -0.940 & -1.380 \\ -0.790 & 0.550 & -0.920 & 0.450 & -0.470 & -0.680 & -0.550 & -11.020 & 1.440 & -0.810 \\ 0.490 & -0.840 & 1.290 & -1.310 & 0.860 & -0.630 & 0.450 & -0.350 & -12.360 & 0.720 \\ -0.730 & 0.100 & -1.430 & -0.650 & -0.050 & 1.300 & 1.400 & 0.260 & 1.160 & -9.500 \end{bmatrix}$$

to compute the system matrices as

$$W = \begin{bmatrix} 1.791 & -0.547 & -3.648 & -3.044 & -3.559 & 2.061 & 1.652 & -0.922 & -1.767 & 2.025 \\ -2.183 & -7.693 & -0.748 & 2.834 & -0.383 & -2.765 & 1.071 & -1.031 & 6.677 & 5.252 \\ -3.454 & -1.241 & -5.778 & 1.727 & -4.854 & -4.974 & 0.514 & -1.544 & -0.341 & -1.092 \\ -2.602 & 3.529 & 1.244 & -4.660 & -2.902 & 2.547 & -2.530 & 2.180 & 2.529 & -2.361 \\ -4.506 & -0.502 & -3.117 & -3.921 & 1.244 & 1.138 & 0.186 & -0.286 & -1.082 & 3.573 \\ 1.493 & -2.117 & -5.815 & 3.908 & 1.343 & -0.459 & -1.192 & 2.852 & -2.297 & -2.244 \\ 1.620 & 0.806 & -0.979 & -1.182 & 0.520 & -4.599 & -5.531 & -5.829 & -0.054 & -4.901 \\ -1.929 & -1.156 & 0.285 & 1.443 & -1.529 & 0.813 & -7.543 & -4.027 & -0.266 & 3.485 \\ 0.352 & 6.485 & -0.726 & 3.002 & -1.298 & -1.628 & 0.261 & -0.161 & -5.035 & 3.922 \\ 0.274 & 1.636 & -1.150 & -3.608 & 1.624 & -3.474 & -2.448 & 4.108 & 2.696 & -1.157 \end{bmatrix}$$

$$b = \begin{bmatrix} 1 & 3 & -3 & 1 & 1 & 1 & 1 & -1 & 3 & 1 \end{bmatrix}^T$$

We note that  $W$  and  $b$  given above are used in the generalized BSB model (2.11) with  $\alpha = 0.3$  for the simulation of the method by Lillo *et al.*



**Example 2** This example is taken from [7]. The patterns to be stored are the five vectors considered in the Example 1 along with the following vector

$$y^{(6)} = \left[ 0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1 \right]^T$$

We will again use the threshold function given in Figure 4.1. If we apply the decomposition algorithm (4.3) to this set of equilibria, we see that there is no equivalent decomposition if we overlap only one row. However, if we apply the algorithm with two rows overlapped, we see that various equivalent decompositions exist. Among them, let us take the groups (1,3,4,5,8) and (1,2,5,6,7,9,10). Then the expansion matrix becomes

$$V = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Transforming the equilibria set by  $V$  we obtain

$$\tilde{Y} = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Now the problem is reduced to solving for 2 neural networks of dimensions 5 and 7 which have equilibria as the columns of the following matrices.

$$\tilde{Y}_1 = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}, \quad \tilde{Y}_2 = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Taking into account also the condition  $MV = 0$  with the equilibria constraints of the above subsystems, the following solution results in no spurious states for

the subsystems.

$$\tilde{W}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 2 & 1 & 0 \end{bmatrix}, \quad \tilde{b}_1 = \begin{bmatrix} -0.5 \\ 0.5 \\ 1.5 \\ -0.5 \\ -2.5 \end{bmatrix}$$

$$\tilde{W}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 3 & 0.5 & 2 & -2 & -6 & 0.5 & -4 \\ 1 & -0.5 & 1 & -1 & 1 & -0.5 & 0 \\ 1 & 0.5 & -2 & 0 & -1 & 0.5 & 0 \\ -1 & 1 & 3 & 0 & -2 & 1 & 0 \end{bmatrix}, \quad \tilde{b}_2 = \begin{bmatrix} -0.5 \\ 0.5 \\ -0.5 \\ 3.5 \\ 0.5 \\ 1.5 \\ -0.5 \end{bmatrix}$$

Using

$$U = \begin{bmatrix} \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

we can obtain the system parameters by contraction as

$$W = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 3 & 0.5 & 0 & 0 & 2 & -2 & -6 & 0 & 0.5 & -4 \\ 1 & -0.5 & 0 & 0 & 1 & -1 & 1 & 0 & -0.5 & 0 \\ 1 & 0 & 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0.5 & 0 & 0 & -2 & 0 & -1 & 0 & 0.5 & 0 \\ -1 & 1 & 0 & 0 & 3 & 0 & -2 & 0 & 1 & 0 \end{bmatrix}, \quad b = \begin{bmatrix} -0.5 \\ 0.5 \\ 0.5 \\ 1.5 \\ -0.5 \\ 3.5 \\ 0.5 \\ -2.5 \\ 1.5 \\ -0.5 \end{bmatrix} \quad (5.2)$$

The solution given in (5.2) results in a network with no spurious states. However the solution has limit cycles. In Table (5.2) we compare the performances of various neural networks using different methods. The system parameters for eigenstructure method and Perfetti's method can be found in [7] and [9]. For the Lillo's method, we use the same strongly row diagonal dominant matrix  $D$  we used in Example 1 and

$$\Lambda = \begin{bmatrix} -10.060 & -1.360 & 0.540 & 0.540 & 1.300 & -0.350 & 0.060 & 0.990 & -1.400 & -1.340 \\ 0.090 & -10.150 & -1.480 & -0.350 & -1.300 & -0.250 & 0.560 & 0.270 & 1.290 & 1.040 \\ 0.080 & -1.220 & -12.370 & -0.250 & 0.600 & 1.230 & 0.790 & -0.710 & -1.360 & 0.710 \\ -0.520 & 0.400 & 0.770 & -11.620 & -0.400 & -0.760 & 1.450 & 0.670 & 0.760 & 0.450 \\ -1.280 & 0.390 & 1.150 & -0.680 & -9.140 & 0.800 & -0.070 & -0.790 & -0.680 & -0.420 \\ -1.000 & -0.040 & 1.190 & 1.230 & -1.320 & -9.760 & 0.010 & 0.050 & -0.540 & 1.460 \\ -0.020 & -0.700 & -1.230 & 1.340 & -1.280 & 0.000 & -9.770 & -0.670 & 1.240 & 0.090 \\ -0.110 & 1.320 & -1.350 & 0.780 & 0.810 & 0.980 & -1.120 & -11.540 & 0.570 & 1.100 \\ 0.390 & 0.710 & 0.680 & 1.500 & 1.170 & -0.800 & -0.580 & -0.450 & -9.130 & 0.270 \\ 1.040 & -0.260 & 1.020 & -0.690 & -0.250 & 0.110 & -0.100 & -0.640 & -0.970 & -9.660 \end{bmatrix}$$

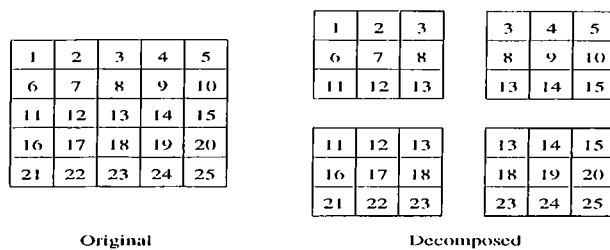


Figure 5.1: Original and decomposed systems

to compute the system parameters as

$$W = \begin{bmatrix} 2.442 & -0.253 & -4.053 & -3.161 & -2.908 & 2.004 & 0.758 & -0.993 & -0.293 & 0.201 \\ -1.971 & -4.156 & 0.602 & -1.499 & -0.171 & -0.627 & -1.949 & -1.644 & 7.284 & 1.845 \\ -3.009 & 1.281 & -3.692 & -0.830 & -4.409 & -2.333 & -1.161 & -2.857 & 1.141 & -2.454 \\ -3.989 & 0.577 & -0.989 & -2.503 & -4.289 & 0.700 & 1.863 & 4.877 & 0.937 & 0.336 \\ -4.385 & -0.254 & -2.123 & -4.807 & 1.365 & 1.011 & -0.866 & -0.205 & -1.324 & 3.438 \\ 1.499 & 0.435 & -2.876 & 0.128 & 1.349 & 3.475 & -4.604 & -0.709 & -0.065 & -3.095 \\ 0.486 & -2.709 & -1.976 & 0.914 & -0.614 & -5.461 & -1.939 & -4.606 & -0.769 & -1.534 \\ -2.158 & -2.648 & -0.791 & 4.838 & -1.758 & -1.033 & -4.829 & -3.092 & -3.398 & 6.263 \\ 1.511 & 7.480 & 0.876 & 1.491 & -0.139 & -0.185 & -2.815 & -2.010 & -2.360 & 1.694 \\ 1.099 & 1.595 & -0.496 & -1.932 & 2.449 & -4.645 & -3.014 & 3.861 & 0.885 & -0.475 \end{bmatrix}$$

$$b = \left[ 0 \quad 2 \quad -4 \quad 2 \quad 2 \quad 0 \quad 2 \quad 0 \quad 2 \quad 2 \right]^T$$

We note that  $W$  and  $b$  given above are used in the generalized BSB model (2.11) with  $\alpha = 0.3$  for the simulation of the method by Lillo *et al.*

**Example 3** In this example we will try to realize the character set given in Figure (5.2) as asymptotically stable equilibria of a 25 neuron neural network using the threshold function given in Figure 4.1. Assume that we code the prototype patterns such that black pixels are denoted by “0”s and white pixels are denoted by “1”s. Numbering and decomposition of the neurons into 4





$$\dot{W}_1 = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & -1 & 1 & -1 & 1 \\ 2 & 1 & 2 & 0 & 0 & -1 & -1 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & -4 & 1 & 1 & -0.750 & 3 & -2 & 0 \\ -1 & -1 & 0 & -1 & 1 & -0.167 & 1.833 & -1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \tilde{b}_1 = \begin{bmatrix} 0.5 \\ -0.5 \\ -0.5 \\ -0.75 \\ 1.667 \\ -0.5 \\ -0.5 \\ -0.5 \\ -0.5 \end{bmatrix}$$

$$\dot{W}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -2 & 1 & 2 & -0.75 & 1 & -1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & -2 & 0 & -3 & 1 & 0 & 1 & -1 & 0 \\ -5 & 4 & -2 & -0.75 & -1 & 3 & 1 & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \tilde{b}_2 = \begin{bmatrix} -0.5 \\ 0.5 \\ -0.75 \\ -0.5 \\ 3.5 \\ -0.75 \\ -0.5 \\ -0.5 \\ -0.5 \end{bmatrix}$$

$$\dot{W}_3 = \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & -1 & 0 & 1 & 0 & 3 & 1 & 0 & -1 \\ -1 & -2 & 1 & 0 & 2 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 3 & 0 & -1 & -2 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \tilde{b}_3 = \begin{bmatrix} -0.5 \\ -0.5 \\ -0.5 \\ -3.5 \\ 1.5 \\ -0.5 \\ -0.5 \\ -0.5 \\ -0.5 \end{bmatrix}$$



$$\hat{W}_4 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -3 & -1 & 0 & 2 & 0 & -2 & 2 \\ -1 & 0 & 2 & 0 & 0 & 1 & 0 & -1 & -1 \\ -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0.5 & 0 & -1 & 0.5 & -2 & 1 & 2 \end{bmatrix}, \quad \tilde{b}_4 = \begin{bmatrix} -0.5 \\ -0.5 \\ -0.5 \\ -0.5 \\ 3.5 \\ -0.5 \\ -0.5 \\ 0.5 \\ -0.5 \end{bmatrix}$$

The above solutions result in globally stable neural networks with 33, 62, 35 and 38 asymptotically stable equilibria, respectively. By contraction we obtain a neural network with 6769 equilibria. The number of patterns converging to each desired prototype is shown in Table (5.3).

If we try to synthesize the neural network using the methods described in Chapter 2, we will not succeed because among those methods, outer product rule will be able to store only 3 of the patterns, projection learning rule, eigen-structure method and method by Lillo *et al* [8] will store all the  $2^{25}$  patterns. For this set of patterns, Perfetti's method is not even applicable since there are prototype patterns which are at unit Hamming distance from each other (5 & 6, 8 & 9).

Method	OPM	PLR	Eigenstructure method	Lillo et al	Perfetti	overlapping
# of stable vectors	41	122	20	8	10	5
# of vectors converging to $y^{(1)}$	38	24	84	143	112	240
# of vectors converging to $y^{(2)}$	41	28	75	62	58	128
# of vectors converging to $y^{(3)}$	73	34	113	16	82	512
# of vectors converging to $y^{(4)}$	38	24	84	29	133	16
# of vectors converging to $y^{(5)}$	68	33	105	56	127	128
total # of adjacent vectors in the basins of attraction	33	30	40	9	46	35

Table 5.1: Comparison of different methods for Example 1

Method	OPM	PLR	Eigenstructure method	Lillo et al	Perfetti	overlapping
# of stable vectors	81	282	18	10	14	6
# of vectors converging to $y^{(1)}$	48	9	36	46	126	16
# of vectors converging to $y^{(2)}$	10	14	64	16	23	224
# of vectors converging to $y^{(3)}$	31	13	85	11	53	120
# of vectors converging to $y^{(4)}$	34	9	36	18	122	16
# of vectors converging to $y^{(5)}$	35	14	66	48	123	32
# of vectors converging to $y^{(6)}$	11	5	29	50	30	240
total # of adjacent vectors in the basins of attraction	29	18	52	3	51	35

Table 5.2: Comparison of different methods for Example 2

Pattern	A	B	C	D	E	F	G	H	I	J
# of vectors converging	10240	864	288	6016	120	384	768	122880	1344	2688
Pattern	P	R	U	V	Y	K	L	M	N	O
# of vectors converging	7680	368	5120	56832	39376	768	1920	79680	651392	6720
Pattern	Z	1	2	3	4	5	6	7	8	9
# of vectors converging	128	5376	24	576	56	432	144	512	216	648

Table 5.3: Basin of attraction of the prototypes for Example 3



Figure 5.2: Characters to be recognized by the neural network

# Chapter 6

## Conclusion

In this thesis, we dealt with the associative memory problem in various aspects. We first showed that quantizer multilevel functions with the same number of saturation levels are equivalent. We then presented a characterization of the connection weights for a discrete-time neural network model using quantizer type multilevel functions. With an example we illustrated that while usage of multilevel activation function decreases the number of neurons used, it increases the possibility of more spurious states as a result of decreasing the dimension of the state space.

In the rest of the thesis, we applied the idea of overlapping decompositions to the associative memory design. Considering a discrete-time neural network model, we first developed the necessary tools for expansions and contractions, then gave an algorithm to solve the associative memory problem in a larger state space. We provided the algorithm for the decomposition of the equilibria set into two smaller dimension equilibria sets equivalently and the algorithm for the design of these subsystems in a suboptimal way. We illustrated the effectiveness of the method by a pattern recognition example. We note that design by use of overlapping decompositions yields sparse or nearly sparse matrices which provides the neural networks to be more easily implemented.

At this point, generalization of overlapping decompositions to the design of continuous-time neural networks is straightforward. However, one other thing needs to be mentioned. In the design of associative memories by means of algorithm (4.1), we set  $\overline{W}_{12}$  in (4.29) and  $\overline{W}_{21}$  in (4.30) to zero, but one can

use  $W_{13}$  and  $W_{31}$  in  $\overline{W}_{12}$  and  $\overline{W}_{21}$ , as a means of decreasing the number of spurious states.

It is obvious that associative memory design problem will be continued to be analyzed in various aspects. One research topic is a good characterization of the connection weights for the multilevel activation function so that the desired characteristics of associative memories are satisfied. Another interesting research topic may be to find a better algorithm for design of neural networks which can be incorporated in the “Divide and Design” algorithm given in Chapter 4.

## REFERENCES

- [1] J.J. Hopfield “Neural networks and physical systems with emergent collective computational abilities,” *Proc. Natl. Acad. Sci. USA*, vol. 79, pp. 2554–2558, April 1982.
- [2] J.J. Hopfield “Neurons with graded response have collective computational properties like those of two-state neurons,” *Proc. Natl. Acad. Sci. USA*, vol. 81, pp. 3088–3092, May 1984.
- [3] L. Personnaz, I. Guyon and G. Dreyfus “Information storage and retrieval in spin-glass like neural networks,” *J. Physique Lett.*, vol. 46, pp. 359–365, April 1985.
- [4] L. Personnaz, I. Guyon and G. Dreyfus “Collective computational properties of neural networks: New learning mechanisms,” *Physical Review A*, vol. 34, pp. 4217–4228, November 1986.
- [5] J.H. Li, A.N. Michel and W. Porod “Analysis and synthesis of a class of neural networks: Linear systems operating on a closed hypercube,” *IEEE Trans. on Circuits and Systems*, vol. 36, pp. 1405–1422, November 1989.
- [6] J.H. Li, A.N. Michel and W. Porod “Analysis and synthesis of a class of neural networks: Variable structure systems with infinite gain,” *IEEE Trans. on Circuits and Systems*, vol. CAS-36, pp. 713–731, 1989.
- [7] 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, January 1991.
- [8] W.E. Lillo , D.C. Miller , S. Hui and S.H. Zak “Synthesis of brain-state-in-a-box (bsb) based associative memories,” *IEEE Trans. on Neural Network*, vol. 5, pp. 730–737, September 1994.

- [9] R. Perfetti “A synthesis procedure for brain-state-in-box neural networks,” *IEEE Trans. on Neural Networks*, vol. 6, pp. 1071–1080, September 1995.
- [10] W. Banzhaf “Towards continuous models of memory,” in *Proc. IEEE 1st Int. Conf. Neural Nets*, volume 2, pp. 223–230, San Diego, June 1987.
- [11] M. Fleisher “The hopfield model with multi-level neurons,” in D. Anderson, ed., *Neural Inform. Processing Syst.:AIP Conf. Proc.*, pp. 278–289, Denver,CO, 1987.
- [12] J. Si and A.N. Michel “Analysis and synthesis of a class of discrete-time neural networks with multilevel threshold neurons,” *IEEE Trans. on Neural Networks*, vol. 6, pp. 105–116, January 1995.
- [13] A. Guez, V. Protopopsecu and J. Barnhen “On the stability, storage capacity and design of nonlinear continuous neural networks,” *IEEE Trans. Syst., Man, Cybern.*, vol. 18, pp. 80–86, January–February 1988.
- [14] C.M. Marcus, F.R. Waugh and R.M. Westervelt “Associative memory in an analog iterated-map neural networks,” *Phys. Rev. A*, vol. 41, pp. 3355–3364, March 1990.
- [15] C.M. Marcus and R.M. Westervelt “Dynamics of iterated-map neural networks,” *Phys. Rev. A*, vol. 40, pp. 501–504, July 1989.
- [16] C. Meunier, D.Hansel and A. Varga “Information processing in in three-state neural networks,” *J. Statist. Phys.*, vol. 55, pp. 859–901, 1989.
- [17] A.N. Michel and J.A. Farrell “Associative memories via artificial neural networks,” *IEEE Contr. Sys. Mag.*, vol. 10, pp. 6–17, April 1990.
- [18] G. Yen and A.N. Michel “A learning and forgetting algorithm in associative memories: The eigenstructure method,” *IEEE Trans. on Circuits and Systems-II: Analog and Digital Signal Processing*, vol. 39, pp. 212–225, April 1992.
- [19] J.A. Farrell and A.N.Michel “A synthesis procedure for hopfield’s continuous-time associative memory,” *IEEE Trans. on Circuits and Systems*, vol. 37, pp. 877–884, July 1990.
- [20] G.H. Golub and C.F. Van Loan. *Matrix Computations*. John Hopkins University Press, Baltimore, 1989.

- [21] J.H. Li, A.N. Michel and W. Porod “Qualitative analysis and synthesis of a class of neural networks,” *IEEE Trans. on Circuits and Systems*, vol. 35, pp. 976–986, August 1988.
- [22] A.N. Michel, K. Wang, D. Liu and H. Ye “Qualitative limitations incurred in implementations of recurrent neural networks,” *IEEE Control Systems Mag.*, vol. 15, pp. 52–65, June 1995.
- [23] J.A. Anderson, J.W. Silverstein, S.A. Ritz and R.S. Jones “Distinctive features, categorical perception, and probability learning: Some applications of a neural model,” *Psych. Rev.*, vol. 84, pp. 413–451, September 1977.
- [24] S. Hui and S.H. Zak “Dynamical analysis of the brain-state-in-a-box (bsb) neural models,” *IEEE Trans. Neural Networks*, vol. 3, pp. 86–94, 1992.
- [25] S. Hui, W.E. Lillo and S.H. Zak “Dynamics of brain-state-in-a-box (bsb) neural models: Stability and convergence analysis,” in M. H. Hssoun, ed., *Associative Neural Memories: Theory and Implementation*, NY:Oxford University Press, 1992.
- [26] M. Ikeda and D.D. Siljak “Overlapping decompositions, expansions and contractions of dynamic systems,” *Large Scale Systems*, vol. 1, pp. 29–38, 1980.
- [27] M. Ikeda, D.D. Siljak and D.E. White “Decentralized control with overlapping information sets,” *J. of Optimization Theory and Applications*, vol. 34, pp. 279–310, 1981.
- [28] J.L. Calvet and A. Titli “Overlapping vs partitioning in block-iteration methods: Application in large-scale system theory,” *Automatica*, vol. 25, pp. 137–145, 1989.
- [29] İ.M. Arabacıoğlu, M.E. Sezer and Ö.H. Oral “Overlapping decomposition of large scale systems into weakly coupled subsystems,” in C.I. Byrnes and A. Lindquist, eds., *Computational and Combinatorial Methods in System Theory*, pp. 135–147, North-Holland, Amsterdam, 1986.
- [30] Dragoslav D. Šiljak. *Decentralized Control of Complex Systems*, volume 184. Academic Press Inc., 1990.