OPTICAL COMPUTING AND HIGHER ORDER ASSOCIATIVE MEMORIES

Thesis by

Cheol Hoon Park

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1990

(Submitted April 27, 1990)

Acknowledgements

First of all, I would like to show my appreciation for the tender care and guidance of my advisor, Professor Demetri Psaltis. From his deep thought and insight I have learned a lot not only about research itself but also about the way of doing research.

I also deeply appreciate the benefits from Drs. Eung Gi Paek, Hyuk Lee, and John Hong. Lots of discussions with Dr. Paek excited my insight to the ideas; the optical implementation of cubic memories came from discussions with Dr. Lee; and Dr. Hong contributed a lot to suggest the optical implementations of quadratic memories. My thanks should go to my colleagues (some of them are now doctors) for their help and discussions, Ken Hsu, David Brady, Xiang-Guang Gu, Scott Hudson, Nabeel Riza, Alan Yamamura, Mark Neilfeld, Charlie Stirk, Drs. Santosh Venkatesh, Kelvin Wagner, Mike Haney, Jeff Yu, Fai Mok, Robert Snapp, and recently Sidney Li, Chuanyi Ji, Subrata Rakshit, Francis Ho, Steve Lin, Seiji Kobayashi, Yong Qiao, and Dr. David Kagan. Discussions with Charlie Stirk on the noise sensitivity of outer product memories made Chapter 5 possible. Also, thanks to Dr. Snapp for his fruitful discussions. I am indebted to Alan Yamamura for his initial help with our computers.

Continuous encouragement and support have come from my family and my friends. My greatest happiness is to have had a chance to go through the Bible recently and to learn the love of God and the patience of Job.

"God saw that it was good."

Genesis 1

"Be joyful always,

pray at all times,

be thankful in all circumstances.

This is what God wants from you

in your life in union with Christ Jesus."

Thessalonians I 5, 16–18

Meanwhile these three remain:

faith, hope, and love;

and the greatest of these is love.

Corinthians I 13, 13

Abstract

Beginning with a discussion of the relationship between degrees of freedom and capacity of the system, the original work on higher order associative memories is described in three aspects, *Learning, Capacity,* and *Generalization* for pattern recognition and neural networks with the orthogonalization of binary vectors and the ternarization of weights, and their optical implementations using volume holograms are suggested for optical computing. Selection of terms is considered to satisfy the given conditions. When a simple sum of outer product learning rule is applied, higher order memories become higher order Hopfield–type memories. Their capacities are derived from SNR analyses for both nonzero diagonal and zero diagonal memories. Especially in the case of quadratic and cubic memories, optical implementations are suggested in three elegant ways due to the three-dimensional property of volume holograms. Robustness of higher order associative memories is discussed as a generalization property with consideration of dynamic range in terms of robustness of errors in input (error tolerance) and noise in the system (noise sensitivity). In the case of autoassociation or bidirectional association the energy functions are used to investigate the dynamics that provides a mechanism of escaping the local minima to find global minima.

Algorithmic aspects and architectures of optical computing are discussed in terms of deterministic and random algorithms.

Contents

List of Figures

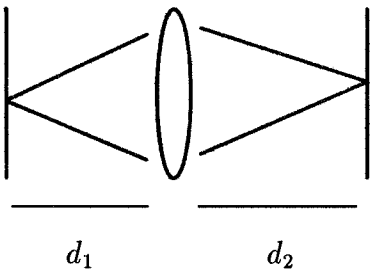
Chapter 1

Introduction

New aspects of computational algorithms and architectures will be introduced in hopes of constructing a general computer. We can think of a general computer as a black box that implements the operations yielding a desired output for any given input. We would like to build an optimum computer to achieve not only Boolean operations but also self–organization[21] and generalization[13, 35], which are not accomplished by present computers. And optics has merits over electronics due to high connectivity[15, 39], three-dimensional analog processing[32], and high speed, inherently and algorithmically due to parallelism[28], which will be discussed in a separate chapter. So optics is a suitable candidate for the black box.

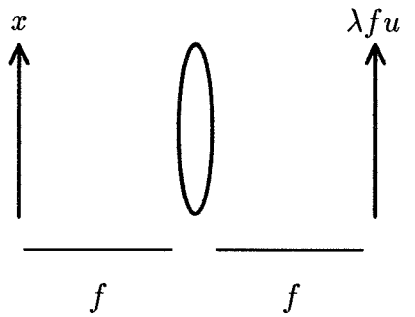
Optics has characteristics of global operations and parallelism from the input to the output such as the imaging system and the Fourier transform system, which will match the characteristics of neural networks. For example, for 1000×1000 pixels of input and output, there are 10^6 parallel channels[28] as in Fig. 1.1. The human brain has superiority to digital computers, especially with recognition problems such as pattern recognition and speech recognition[3, 41] in a sense of generalization and with the global operations such as dynamic systems.

An associative memory can be thought of as a global mapping $Y = f(X)$ from an input vector space $X = \{\underline{x}^m\}$ to an output vector space $Y = \{\underline{y}^m\}$ so that it produces $\underline{y}^m$ as its associated output when $\underline{x}^m$ becomes an input for $m = 1, \ldots, M$. We denote by N and N_0 the dimensionalities of the input and output vectors, respectively. If the mapping also works inversely, it is called a bidirectional associative memory[22]. When the output vectors are stored as binary N_0-tuples, the associative memory can be implemented as an array of discriminant functions, each dichotomizing the input



$$\frac{1}{d_1} + \frac{1}{d_2} = \frac{1}{f}$$

a



$$F(u) = \int f(x) e^{j2\pi ux} dx$$

b

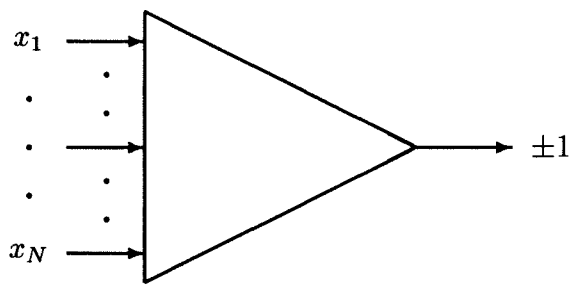
Figure 1.1 Optical systems: a. Imaging system; b. Fourier transform system.

vectors into two classes. This type of associative memory is shown schematically in Fig. 1.2. In evaluating the effectiveness of a particular associative memory, we are concerned with its ability to store a large number of associations (capacity), the ease with which the parameters of the memory can be set to realize the prescribed mappings (learning), and the way in which the memory responds to input that did not constitute its training set (generalization). We will describe associative memories in this respect. The relationship between the number of independent parameters or degrees of freedom of a memory and its ability to store associations[2, 10, 38] is fundamental to this work and it will be stated in the following chapter as a theorem.

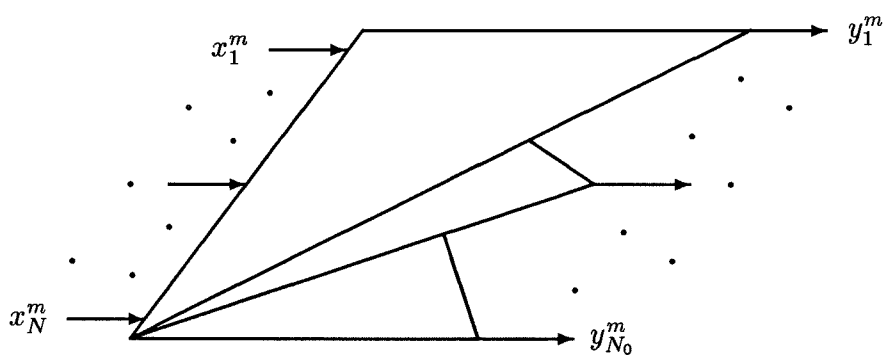
Chapter 2 concerns theoretical background for the rest of the work. It starts with the concepts of degrees of freedom and capacity of the system. The properties of higher order memories are analyzed with orthogonalization of binary feature vectors and ternarization of weights. The motivation is the increase in storage capacity that results from the increase in the number of degrees of freedom that is needed to describe a higher order associative mapping.

In Chapter 3 two algorithmic aspects of computing are discussed in terms of *deterministic* and *random* algorithms. The former stems from orthogonalization and ternarization introduced in Chapter 2 and the latter is related to dynamic systems, which explains the algorithm of the optical system for the high speed overcomming the limitation of the digital computer using the generation of a fractal pattern[23] as an example. The method of choosing terms that are necessary for given requirements of a problem is also discussed and is relevant to the ternarization algorithm.

The higher order associative memory[6, 7, 13, 24, 31, 36, 37, 38, 40] is discussed with optical implementations in Chapter 4. Higher order memories can be characterized as a generalization of the linear memories using nonlinearities and have both properties of high capacity and easy learning. They absorb the information out of the input and utilize the correlations of the information as well as the infor-



a



b

Figure 1.2 a. Discriminant function; b. Associative memory constructed as an array of discriminant functions.

mation itself to memorize the input–output data and carry out the task. It turns out that higher order memories offer higher storage capacity and efficient algorithms for generalizations such as invariances. The simple sum of outer product learning rule can be applied to the memory to make it a higher order Hopfield–type memory. Another way of describing this kind of higher order memory is that its interconnection weight w_{ij} between the i-th and the j-th neurons is affected not only by the states of the i-th and the j-th neurons themselves but also by the states of the other neurons. And we suggest the optical implementations of higher order associative memories using volume holograms and compare them with those using planar holograms. Especially, in the case of quadratic and cubic memories, the memory tensors w_{ijk} and w_{ijkl} required to describe the memories are supplied in optics by volume holograms because of the three-dimensional properties they have[39], and their optical implementations are suggested in three alternative ways with separate interpretations.

In Chapter 5 robustness of higher order associative memories is discussed as a generalization property in two respects, robustness for the errors in the input compared to the stored data (error tolerance), and robustness for the noise in the memory (noise sensitivity). Dynamic range of the system is considered for the real computation.

This is mainly based on references [28], [36], [37], and [38].

Chapter 2

Characteristics of Higher Order Memories

2.1 Degrees of freedom and storage capacity

Let D be the number of independent variables (degrees of freedom) we have under our control to specify input-output mappings and let each parameter have K separate levels or values that it can assume. As an example, given vector spaces X_1, X_2, ... , X_r, and Y with dimensionalities N_1, N_2, ... , N_r, and N_0, consider the cartesian product space $X_1 \times X_2 \times \cdots \times X_r$ and an r-linear mapping $W^r : X_1 \times X_2 \times \cdots \times X_r \rightarrow Y$, which is linear in each argument. Then, the dimensionality of the space $X_1 \times X_2 \times \cdots \times X_r$ becomes $N_1 N_2 \cdots N_r$ and therefore the number of degrees of freedom D is given by $N_1 N_2 \cdots N_r N_0$[43].

We define the storage capacity C to be the maximum number of arbitrary associations that can be stored and recalled without error. Throughout the text, N and N_0 represent the dimensionalities of the input and output vectors, respectively. Let K_0 be the number of distinct levels that the output components can assume.

Theorem 1

$$C \leq \frac{D \log_{K_0} K}{N_0}. \tag{2.1}$$

Proof The number of different states of the memory is given by K^D and the total number of outputs that a given set of M input patterns can be mapped to is $K_0^{N_0 M}$. If the number of mappings was larger than the number of distinct states of memory, then mappings would exist that are not implementable. The requirement that all mappings be done leads to the relationship of the theorem.

The equality in Eq. 2.1 is achieved by Boolean circuits such as programmable logic arrays for $K_0 = 2$. When the equality holds, resetting any one bit in any

one of the parameters of the memory gives a different mapping. The way to get generalization when $C = D \log_{K_0} K/N_0$ is to impose on it the overall structure of the memory before learning begins. One of the appealing features of neural architectures is the considerable redundancy in the degrees of freedom that is typically available. Therefore, there is hope that while a memory learns specific input–output correspondences it can also discover the underlying structure that may exist in the problem and learn to respond correctly for a set of inputs much larger than the training set. Moreover, the same redundancy is responsible for the error tolerance that is evident in many neural architectures. Higher order memories are generally redundant and they can provide us with a methodology for selecting the degree of redundancy along with the number of degrees of freedom and the associated capacity to store random problems.

It is important to keep in mind that Eq. 2.1 holds for *arbitrary* mappings. If the input and output vectors are restricted in some way that happens to be matched to the architecture of a particular associative memory, then it may be possible to exceed this limit. However, selecting the architecture of the associative memory such that it optimally implements only a subset of all possible associations is basically equivalent to choosing the architecture so that it generalizes in a desirable way. For instance, suppose that we design an associative memory so that it is shift invariant (i.e., the output is insensitive to a translation of the input)[24, 35]. Then this system will respond predictably to all the shifted versions of the patterns that were used to train it. We can equivalently think of this system as having a larger storage capacity than the limit of Eq. 2.1 over the set of shift invariant mappings. If we can identify *a priori* the types of generalization that we wish the memory to exhibit and we can find ways to impose these on the architecture, then this is certainly a sensible thing to do. Higher order memories can also provide a convenient framework within which this can be accomplished.

The penalty we must pay for the increase in the storage capacity that is afforded by the increase in the degrees of freedom in a higher order associative memory is the increase in implementation complexity. The computer that implements a higher order memory must have sufficient storage capacity to store a very large number of parameters. Moreover, it must be capable of addressing the stored information with a high degree of parallelism in order to produce an output quickly. We will discuss in Chapter 4 optical implementations of higher order memories and we will show a remarkable compatibility between the computational requirements of these memories and the ability of optics to store information in three dimensions. But in Chapter 3 the possibility of selecting terms for given problems will be shown to reduce the complexity or the number of parameters in implementation.

2.2 Discriminant functions and associative memories

We will consider as a precursor the most familiar associative memories that are constructed as arrays of linear discriminant functions[21]. A linear discriminant function is a mapping from the sample space X, a subset of R^N, to 1 or -1:

$$\begin{aligned} y &= \operatorname{sgn}\{\underline{w}^t \cdot \underline{x} + w_0\} \\ &= \operatorname{sgn}\{w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_N x_N\} \end{aligned} \tag{2.2}$$

where

$$\operatorname{sgn} x = \begin{cases} +1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0, \end{cases}$$

$\underline{w}$ is a weighting vector and w_0 is a threshold value. In this case, the capacity is upperbounded by $(N+1)\log_2 K$ according to our definition of capacity. In this relatively simple case, the exact capacity is known to be equal to $C = N+1$ that is given by degrees of freedom assuming the input points are in general position (points in an N-dimensional space are called to be in general position if no subset of $N+1$

points falls in any $(N-1)$ dimensioal subspace) and $K = \infty$[10]. An associative memory is constructed by simply forming an array of linear discriminant functions each mapping the same input to a different binary variable. Several algorithms exist for training such memories including the perceptron, Widrow-Hoff, sum of outer products, pseudoinverse, simplex methods, and so on[11, 17, 21, 31, 48, 50]. This memory can be thought of as the first order of the broader class of higher order memories that contain not only a linear expansion of the input vector but also quadratic and higher order terms. We will see in Chapter 4 that the learning methods that are applicable to the linear memories generalize directly to the higher order memories. First, however, we will describe the properties of the mappings that are implementable with higher order memories.

2.3 Properties of higher order memories

A Φ-function is defined to be a *fixed* mapping of the input vector $\underline{x}$ to an L-dimensional vector $\underline{z}$ followed by a linear discriminant function:

$$\begin{aligned} y &= \mathrm{sgn}\{\underline{w}'^{t} \cdot \underline{z}(\underline{x}) + w_0\} \\ &= \mathrm{sgn}\{w'_1 z_1 + w'_2 z_2 + \cdots + w'_L z_L + w_0\} \end{aligned} \tag{2.3}$$

where $\underline{z}(\underline{x}) = (z_1(\underline{x}), z_2(\underline{x}), \ldots, z_L(\underline{x}))$, $\underline{w}'$ is an L-dimensional weighting vector and $\underline{z}(\underline{x})$ is an L-dimensional vector derived from $\underline{x}$. So it is basically a two–layer network with the first layer fixed. The storage capacity in this case is equal to the capacity of the second layer $L+1$[10] that is given by the degrees of freedom of the second layer if the samples $\underline{z}$ are in general position whereas the upper bound on the capacity from Eq. 2.1 is $(L+1)\log_2 K$. The inefficiency in this case is $\log_2 K$ bits, the same as for the linear discriminant function even though the capacity can be raised by increasing L. It is not known what the exact relationship between L and K is. That is, we do not know whether for higher dimensions we need better resolution for the

values of the weights to be capable of implementing a fixed fraction of the linear mappings (it will be shown that K depends not on L but on the number of data stored in the memory when we have binary samples and adapt simply the outer product learning rule).

A higher order associative memory is an array of Φ-functions with polynomial neurons in the middle, that is, the mappings $\underline{z}(\underline{x})$ being polynomial expansions of the vector $\underline{x}$. Shown in Fig. 2.1 is the schematic diagram of a higher order associative memory whose capacity is also $L+1$. When the polynomial expansion is of the r-th order in $\underline{x}$, then the output vector $\underline{y}$ is given by

$$y_l = \mathrm{sgn}\{W_l^r(\underline{x}, \underline{x}, \ldots, \underline{x}) + W_l^{r-1}(\underline{x}, \ldots, \underline{x}) + \cdots + W_l^2(\underline{x}, \underline{x}) + W_l^1\underline{x} + w_{l0}\} \quad (2.4)$$

where $l = 1, \ldots, N_0$, W_l^k is a k-linear mapping or a $(k+1)$-th order tensor and W_l^1 is equivalent to $\underline{w}^t$ in Eq. 2.2. According to Eq. 2.3,

$$z_j(\underline{x}) = x_{p_{j1}}^{n_1} x_{p_{j2}}^{n_2} \cdots x_{p_{jr}}^{n_r} \quad (2.5)$$

where $j = 1, 2, \ldots, L$, $p_{ji} \in \{1, 2, \ldots, N\}$ for $i = 1, \ldots, r$ such that all z_j are different and $n_1, n_2, \ldots, n_r = 0, 1$. Then L is $\binom{N+r}{r} - 1$[10], and hence the capacity bound is $\binom{N+r}{r} \log_2 K$ as before. For example, if $r = 2$, the function becomes quadratic and has the form $y_l = \underline{x}^t W_l^2 \underline{x} + W_l^1 \underline{x} + w_{l0}$ and the number of nonredundant terms in the quadratic expansion is $(N+1)(N+2)/2$.

The components of the vector $\underline{z}$ are binary-valued if $\underline{x}$ is binary. In this case, the samples cannot be assumed to be in general position, which results in the decrease in the capacity.

Claim 2 *There can be G binary vectors in N-dimensional space that lie in general position where $N + 2 \leq G \leq 2N$ for $N \geq 2$.*

Proof The number of binary vectors in general position in N-dimensional space can not exceed $2N$, since an N-dimensional cube consists of two $(N-1)$-dimensional

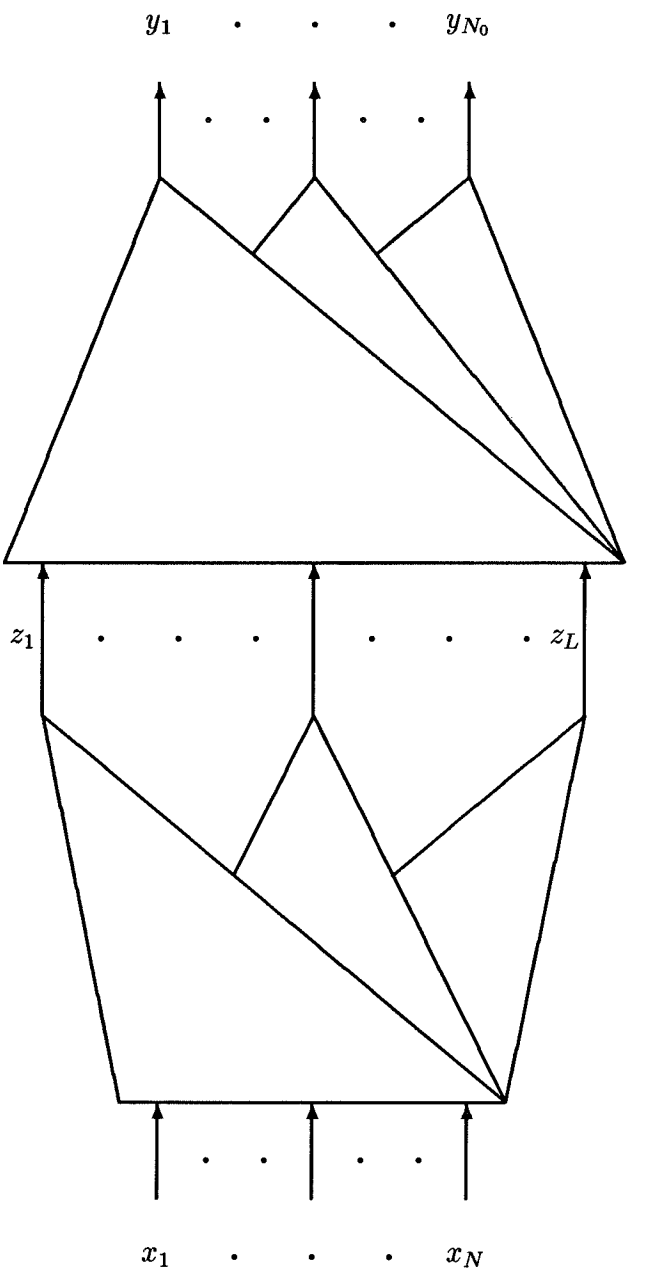


Figure 2.1 Higher order associative memory.

cubes and we can pick up at most N points in general position out of each $(N-1)$-dimensional cube according to the definition of general position[11].

The lower bound is derived by observing that $N+2$ binary points can be chosen to lie in general position as shown in Fig. 2.2 for three- and four-dimensional cubes and it can be generalized for an arbitrary N. For example, $G = 5$ for $N = 3$.

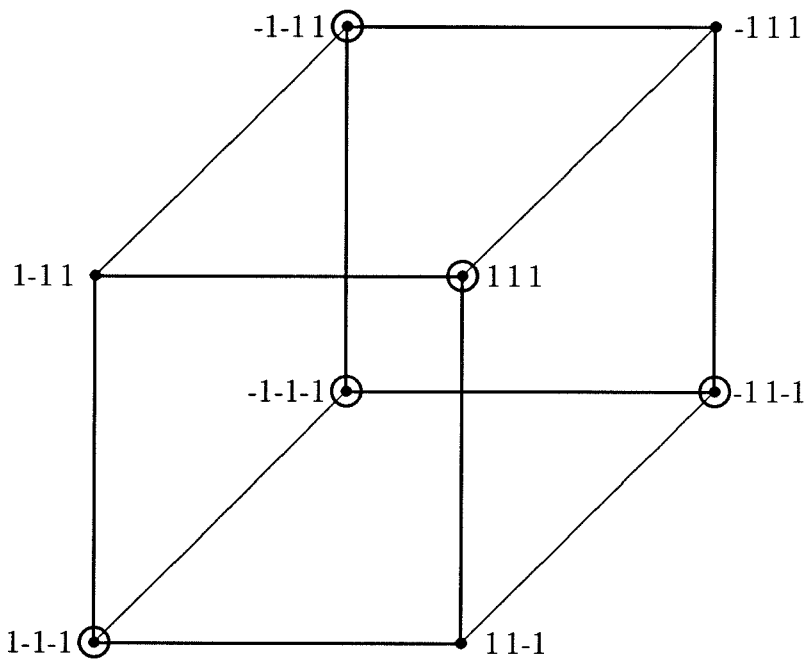
We will evaluate the effectiveness of higher order mappings in producing representations $\underline{z}(\underline{x})$ that are separable by the second layer of weights by calculating the Hamming distance between $\underline{z}$ vectors given the Hamming distance between the corresponding $\underline{x}$ vectors. We expect that if the Hamming distance between two binary vectors is large or if $\underline{z}$ vectors are uncorrelated with each other (or more orthogonal to each other) then they are easy to distinguish from one another.

2.3.1 Complete polynomial expansion of binary vectors

There are at most 2^N nonredundant terms in any polynomial expansion (Eq. 2.4) of a binary vector $\underline{x}$ in N dimensions. First, we will consider the following N-th order expansion (or equivalently bit production) for the bipolar vectors $\underline{x}$ in N dimensional binary space $\{1,-1\}^N$:

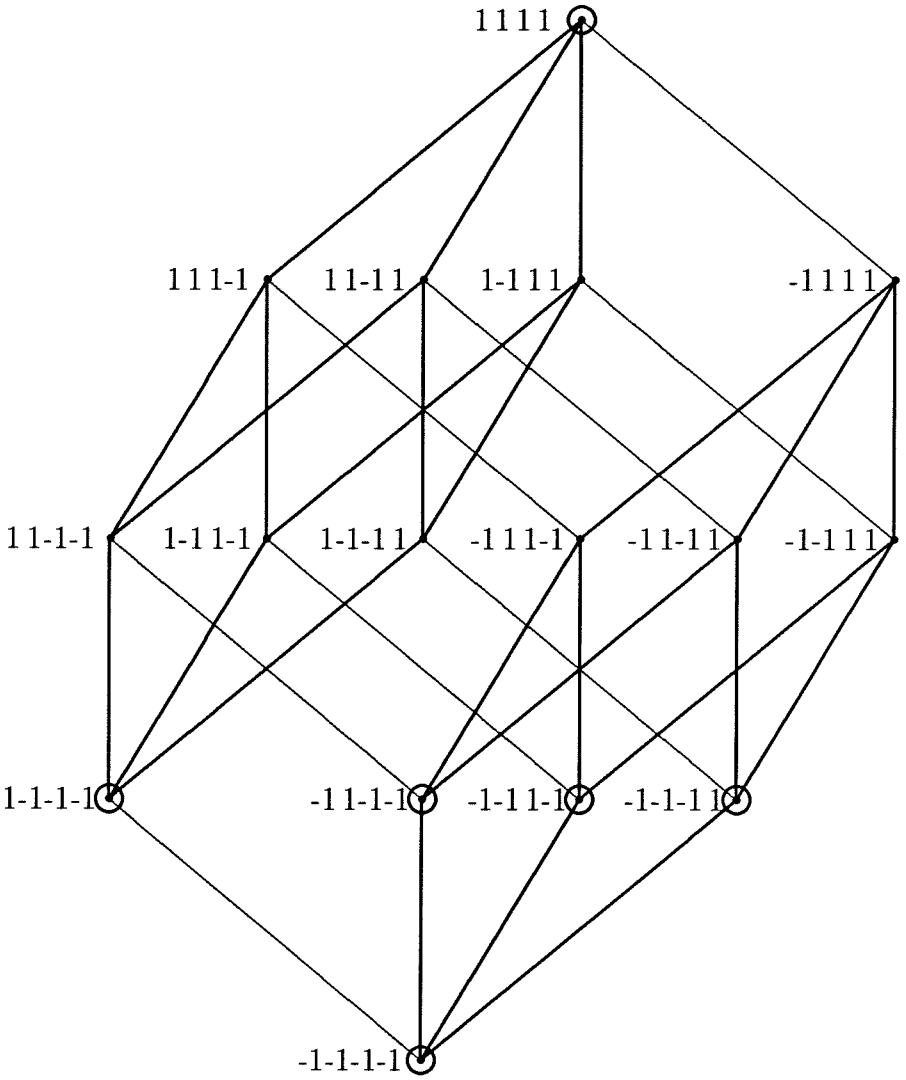
$$\underline{z} = \underline{z}(\underline{x}) = (1, x_1, x_2, \ldots, x_N, x_1x_2, \ldots, x_1x_2\ldots x_N)^t. \tag{2.6}$$

If we apply a linear discriminant function to the new vectors $\underline{z}$, then the capacity becomes 2^N, which is equal to the total number of possible input vectors[36]. In other words this memory is capable of performing *any* mapping of N binary variables to any binary output vector $\underline{y}$. Of course the number of weights that are needed to implement this memory grows to 2^N times N_0, the number of bits at the output. In what follows we show that in this extreme case the vectors $\underline{z}$ become orthogonal to each other.



a

Figure 2.2 a. Three–dimensional cube; b. Four–dimensional cube.


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
-1-1 1 1
1-1-1-1
-1 1-1-1
-1-1 1-1
-1-1-1 1
-1-1-1-1

b

Theorem 3 *If we expand binary vectors* $\underline{x}^m$ $(m = 1, 2, \ldots, 2^N)$ *in* $\{1, -1\}^N$ *to* 2^N*-dimensional binary vectors* $\underline{z}^m$ *according to Eq. 2.6, then the following hold:*

(a) $< \underline{z}^{m_1}, \underline{z}^{m_2} >= 2^N \delta_{m_1 m_2}$ *where* $< \cdot, \cdot >$ *is an inner product.*

(b) $\sum_i z_i^m = 0 \ (m \neq 1)$.

(c) $\sum_m z_{j_1}^m z_{j_2}^m = 2^N \delta_{j_1 j_2}$ *and* $\sum_m z_j^m = 0 \ (j \neq 1)$.

Example The following table is for the case of $N = 3$. Note the orthogonality and the numbers of 1's and -1's in the new vectors and the set of each component of them except the first vector and the set of the first components.

x_1	x_2	x_3	1	x_1	x_2	x_3	x_1x_2	x_2x_3	x_3x_1	$x_1x_2x_3$
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	-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	1	-1	-1	-1	1	1	1	-1

Table 2.1 Full expansion for $N = 3$.

Proof (a) Let us consider any two different binary vectors in the binary space of $\{1, -1\}^N$ whose Hamming distance is n $(1 \leq n \leq N)$. When they are expanded to two 2^N-dimensional binary vectors, the number of k-th order terms that have opposite signs in the two expansions is

$$\begin{aligned} & \binom{n}{1}\binom{N-n}{k-1} + \binom{n}{3}\binom{N-n}{k-3} + \binom{n}{5}\binom{N-n}{k-5} + \cdots\cdots\cdots \\ = & \sum_{i=odd} \binom{n}{i}\binom{N-n}{k-i}. \end{aligned} \tag{2.7}$$

Notice that two polynomials have different values if, and only if, they have an odd number of terms whose signs are opposite. The Hamming distance between the two fully expanded (up to order 2^N) vectors can be calculated by adding the number of terms that have different signs over all the orders of the expansion:

$$
\begin{aligned}
& \binom{n}{1}\binom{N-n}{0} + \binom{n}{1}\binom{N-n}{1} + \left\{\binom{n}{1}\binom{N-n}{2} + \binom{n}{3}\binom{N-n}{0}\right\} \\
+ & \left\{\binom{n}{1}\binom{N-n}{3} + \binom{n}{3}\binom{N-n}{1}\right\} + \cdots\cdots\cdots\cdots \\
+ & \left\{\binom{n}{1}\binom{N-n}{N-n} + \binom{n}{3}\binom{N-n}{N-n-2} + \binom{n}{5}\binom{N-n}{N-n-4} + \cdots\right\} \\
+ & \left\{\binom{n}{3}\binom{N-n}{N-n-1} + \binom{n}{5}\binom{N-n}{N-n-3} + \binom{n}{7}\binom{N-n}{N-n-5} + \cdots\right\} \\
+ & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
= & \sum_{i=odd}\binom{n}{i}\sum_{j=0}^{N-n}\binom{N-n}{j} = 2^{n-1}2^{N-n} \\
= & \ 2^{N-1}. \qquad (2.8)
\end{aligned}
$$

The fact that the Hamming distance is 2^{N-1} for any two expanded vectors (for any n) proves that all of the 2^N vectors become orthogonal and that $< \underline{z}^{m_1}, \underline{z}^{m_2} >= 2^N \delta_{m_1 m_2}$.

(b) Just think of the cases where one of the two vectors is $(1, 1, \ldots, 1)$. Then, all the other vectors $\underline{z}$ have equal numbers of 1's and -1's because their Hamming distances are all 2^{N-1} from the $(1, 1, \ldots, 1)$ vector.

(c) See reference [11], page 109.

Slepian has discussed this orthogonalization property as a method for designing orthogonal codes and has given a different proof for it[42]. The proof presented here is useful for characterizing higher order memories because it allows us to trace the contribution of each order of the expansion to the orthogonalization and immediately derive results about the properties of quadratic and cubic memories.

Each component of output vector $\underline{y}$ in this case is given by

$$y_l = \mathrm{sgn}\Big\{\sum_{i=1}^{2^N} w_{li} z_i\Big\} \tag{2.9}$$

where $l = 1, \ldots, N_0$ and w_{li} is an $N_0 \times 2^N$-dimensional weight matrix. The matrix w_{li} that can implement the mapping $\underline{x}^m \rightarrow \underline{y}^m$ for $m = 1$ to 2^N can be formed simply as the sum of outer products of $\underline{y}^m$ and $\underline{z}^m$ from the above theorem:

$$w_{li} = \sum_{m=1}^{2^N} y_l^m z_i^m. \tag{2.10}$$

2.3.2 Single order expansion

The orthogonalization property of the full expansion is interesting because it shows that higher order memories provide a complete framework that takes us from the simplest "neuron," the linear discriminant function, to the full capability of a Boolean look-up table. Higher order memories can indeed provide a valuable tool for designing digital programmable logic arrays. Since associative memories are capable of accepting input with large N (e.g., if $N = 10^3$, then $2^N \approx 10^{300}$), considering a full expansion of the input data is completely out of the question. In such a case, we are really interested in an expansion that contains a large enough number of terms to provide the capacity needed to learn the problem at hand. In this subsection, we analyze the properties of partial expansions that include all the terms of one order.

We will first consider the memory consisting of all the terms of a quadratic expansion with binary input vectors:

$$\begin{aligned} y_i &= \mathrm{sgn}\Big\{\sum_j \sum_k w_{ijk} x_j x_k\Big\} \\ &= \mathrm{sgn}\Big\{\sum_{n=1}^{L} w'_{in} z_n\Big\}. \end{aligned} \tag{2.11}$$

The number of nonredundant terms in a quadratic expansion of a binary vector is $L = N(N-1)/2$. Let two input vectors have a Hamming distance n. The angle

θ_1 between these two vectors is given by the relation $\cos\theta_1 = 1 - (2n/N)$. The angle θ_2 between the corresponding $\underline{z}(\underline{x})$ vectors derived from quadratic expansions can be readily calculated since we know their Hamming distance from the proof of Theorem 3(a):

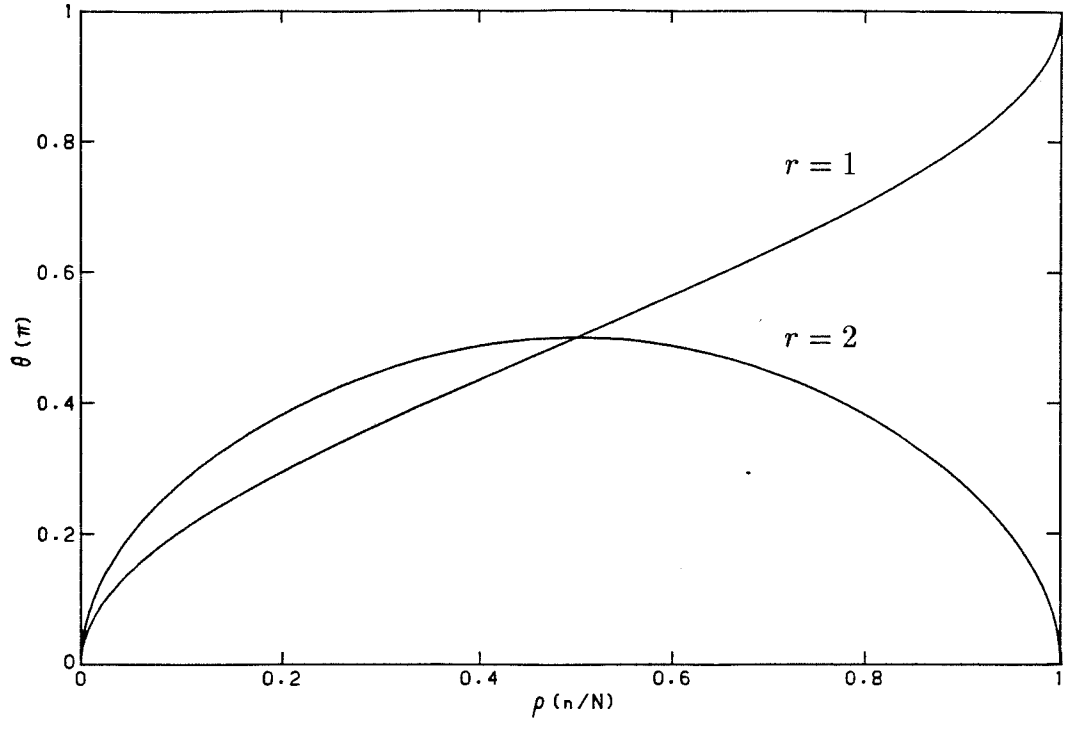
$$\begin{aligned} \cos\theta_2 &= 1 - \frac{4n(N-n)}{N(N-1)} \\ &\approx 1 - 4\rho + 4\rho^2 = (1-2\rho)^2 \end{aligned} \tag{2.12}$$

where $\rho = n/N$. Plotted in Fig. 2.3a are θ_2 and θ_1 versus ρ. For $\rho < .5$, θ_2 is always larger than θ_1. Specifically for $\rho \ll 1$, $\theta_2 = \sqrt{2} \times \theta_1$. For $\rho > .5$ the quadratically expanded vectors are closer to each other than the original vectors and in the extreme case $n = N$, θ_2 becomes zero. We see, therefore, that the quadratic mapping not only expands the dimensionality that provides capacity but also spreads the input samples apart, a generally desirable property, making them more orthogonal to each other and so more uncorrelated. The insensitivity of the quadratic mapping to a change in sign of all the bits is a property that is shared by all even order expansions. Next, we consider a cubic memory

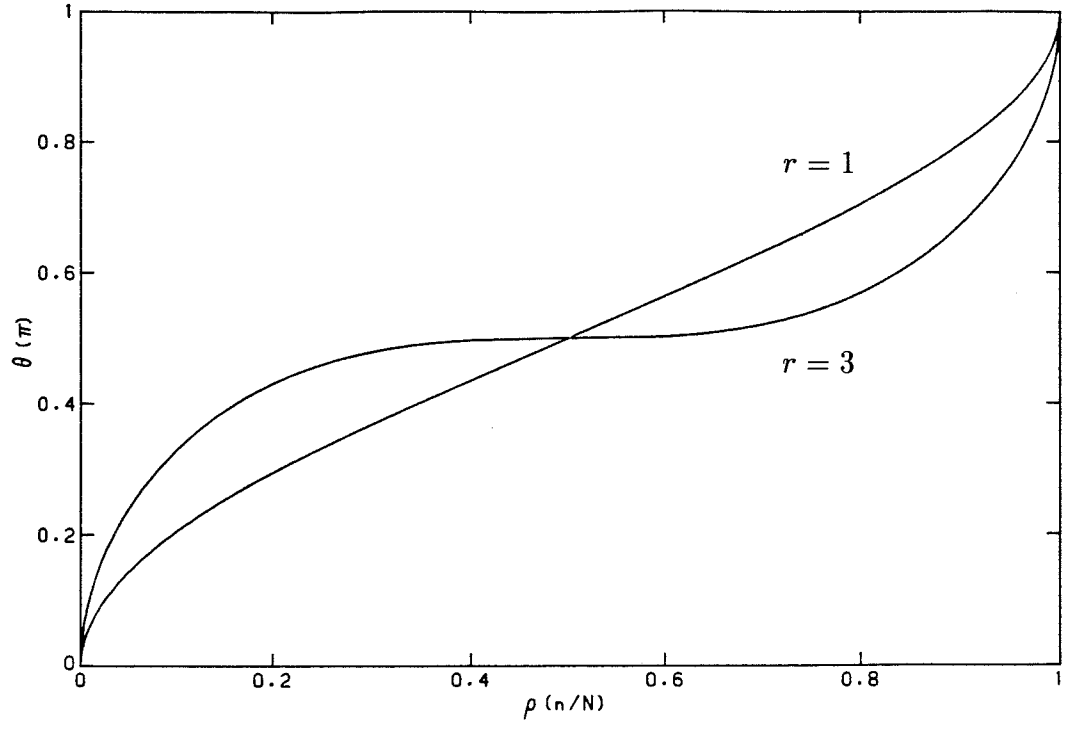
$$\begin{aligned} y_i &= \mathrm{sgn}\Big\{\sum_j \sum_k \sum_l w_{ijkl} x_j x_k x_l\Big\} \\ &= \mathrm{sgn}\Big\{\sum_{n=1}^{L} w'_{in} z_n\Big\} \end{aligned} \tag{2.13}$$

where $L = \binom{N}{3}$. In Fig. 2.3b, we plot θ_3, the angle between two cubically expanded binary vectors as a function of ρ. For convenience, θ_1 is also plotted in the same figure. In this case, θ_3 increases faster with ρ for $\rho < .5$ while for $\rho > .5$, θ_3 remains smaller than θ_1 and in the limit $\rho = 1, \theta_3 = \pi$. Thus the cubic memory discriminates between a vector and its complement. For $\rho \ll 1$, $\theta_3 = \sqrt{3} \times \theta_1$. At $\rho \approx .4$ to .6 the cubic expansion gives essentially orthogonal vectors.

The basic trends that are evident in the quadratic and cubic memories generalize to any order $r(\ll N)$. The number of independent terms in the r-th order expansion



a



b

Figure 2.3 a. Angles between linearly and quadratically expanded vectors as functions of input Hamming distance; b. Angles between linearly and cubically expanded vectors as functions of input Hamming distance.

of a binary vector is $\binom{N}{r}$, which has the maximum value ($\approx \frac{2^N}{\sqrt{2\pi N}}$) for $r \approx N/2$ by Stirling's formula[30]. Again this is not of practical importance because the number of terms in a full expansion of this sort is prohibitively large. What is of interest, however, is the effectiveness with which relatively small order expansions can orthogonalize a set of input vectors. The angle θ_r between two vectors that have been expanded to the r-th order only is given by the following relation according to Eq. 2.7:

$$\cos\theta_r = \frac{\binom{N}{r} - 2\sum_{i=odd}\binom{n}{i}\binom{N-n}{r-i}}{\binom{N}{r}}. \tag{2.14}$$

We can obtain a simpler expression for the interesting case $r \ll N$ and for small ρ, $\theta_r \approx \sqrt{r} \times \theta_1$.

Proposition 4 *For $r \ll N$,*

$$\cos\theta_r \approx (1-2\rho)^r \tag{2.15}$$

where $\rho = n/N$. Moreover, for small ρ,

$$\theta_r \approx \sqrt{r}\theta_1 \tag{2.16}$$

where $\theta_1 \approx 2\sqrt{\rho}$.

Proof For a small r, we can make the approximations $\binom{N}{r} \approx N^r/r!$, $\binom{n}{i} \approx n^i/i!$, and $\binom{N-n}{r-i} \approx (N-n)^{r-i}/(r-i)!$. Then, $\cos\theta_r$ is approximated as follows:

$$\begin{aligned}\cos\theta_r &\approx 1 - 2\sum_{i=odd}\frac{r!}{i!(r-i)!}\rho^i(1-\rho)^{r-i} \\ &= (1-2\rho)^r\end{aligned}$$

because of these relationships:

$$\sum_{i=odd} + \sum_{i=even} = (1-\rho+\rho)^r = 1,$$

$$\sum_{i=odd} - \sum_{i=even} = -(1-\rho-\rho)^r = -(1-2\rho)^r.$$

When $\rho \ll 1$, $\cos\theta_r$, which is approximately $1 - \theta_r^2/2!$, is approximated by $1 - 2r\rho$ directly from Eq. 2.14 or from Eq. 2.15. Therefore, it is followed by Eq. 2.16 that $\theta_r \approx 2\sqrt{r\rho}$.

We plot θ_r versus ρ for selected orders in Fig. 2.4 using Eq. 2.15. It is evident that increasing r results in better separated feature vectors in the sense that they are becoming less correlated to be dichotomized with ease. Polynomial mappings act as an effective mechanism for increasing the dimensionality of the space in which input is classified because they guarantee a very even distribution of the samples in this new space.

2.4 Ternarization of weights for orthogonal vectors

We have discussed the orthogonalization algorithm for the binary samples and turn to the weights. For the orthogonal vectors derived by full expansions the simulations up to $N = 4$ ($2^N = 16$) and selected simulations for $N = 5$ showed that the weights might be made ternary. The algorithm used in this case will be discussed in the following chapter.

Moreover, it may be possible to make a generalization for arbitrary orthogonal vectors without requiring them to be binary and of 2^N dimensions for some integer N. In fact, there are N orthogonal vectors in N-dimensional space.

Conjecture 5 *The weights of LDFs can be ternarized* $(+1, 0, -1)$ *for arbitrary dichotomies of any set of N-dimensional orthogonal vectors (at least with high probability).*

It can be explained informally as follows: in the case of N N-dimensional orthonormal vectors, $\underline{v}^m$ $(m = 1, \ldots, N)$, there are 2^N disjoint solution cones that have the same shape filling the whole space with different orientations. The cone is decided

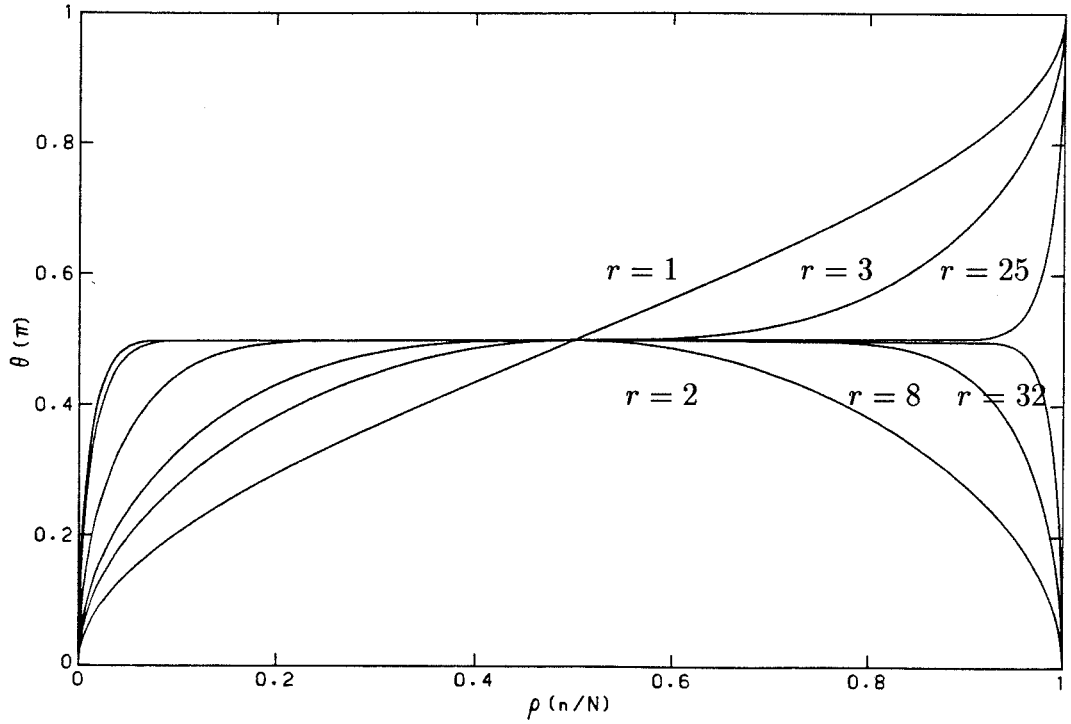


Figure 2.4 Angles between expanded vectors for selected orders.

by N vectors, $y_l^m \underline{v}^m$, for each l-th output component, as a set

$$\Big\{ \sum_{m=1}^{N} \alpha_m y_l^m \underline{v}^m \,|\, \alpha_m > 0 \Big\}. \tag{2.17}$$

Its center axis is in the direction of the vector sum of such vectors $\sum_m y_l^m \underline{v}^m$, the lines through each of which from the origin are all perpendicular to each other and determine the convex set that is the cone called an orthant. There are two kinds of orientations: one is the orientation of the center axis and the other is the rotation of the cone around the center axis. Thus the complexity of the cone orientation gets larger as the dimensionality of the space becomes bigger. It is not sufficient for each cone to include at least one weight vector with weights binary-valued (i.e., even if the 2^N binary weight vectors are evenly spaced in N-dimensional space and the ratio between the number of weight vectors and that of disjoint solution cones is equal to 1, the orientation of the sample vectors may distribute the weight vectors to the solution cones unevenly, which means that some cones may include no binary weight vectors inside). In the case of ternary valued weights the ratio becomes $3^N/2^N$ except that the weight vectors are not spaced evenly. The fact that the ratio is greater than 1 and grows exponentially as N gets large implies that it may be possible for at least one weight vector to be included inside of each cone regardless of the uneven distributions.

Up to five-dimensional cases, we found a proof for this conjecture using geometric pictures.

Lemma 6 *For the orthant of N-dimensional space, a circular cone can be made to contact the orthant inside with the angle θ_N of the following value:*

$$\cos\theta_N = \frac{N-2}{N} \tag{2.18}$$

where θ_N is twice the angle between the center axis and the line of the circular cone contacting the octant.

Proof Because it does not depend on the orientation of the octant, Fig. 2.5 is taken into consideration for simplicity in the case of $N = 3$. Then the unit vector of the center axis is given by $(\frac{1}{\sqrt{N}}, \frac{1}{\sqrt{N}}, \ldots, \frac{1}{\sqrt{N}})^t$ and the unit vector of one of the contacting lines by $(\frac{1}{\sqrt{N-1}}, \frac{1}{\sqrt{N-1}}, \ldots, \frac{1}{\sqrt{N-1}}, 0)^t$. Because the angles between the center axis and the contacting lines are all the same, the angle is obtained by the inner product between them. So we get

$$\cos\frac{\theta_N}{2} = \frac{N-1}{\sqrt{N}\sqrt{N-1}} = \sqrt{\frac{N-1}{N}}. \tag{2.19}$$

Eq. 2.18 follows from the relationship, $\cos 2\theta = 2\cos^2\theta - 1$.

Lemma 7 *The convex set determined by the N vectors $\underline{v}^1 = (1, 0, \ldots, 0)^t$, $\underline{v}^2 = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, 0, \ldots, 0)^t$, $\underline{v}^3 = (\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, 0, \ldots, 0)^t$, $\cdots$, and $\underline{v}^N = (\frac{1}{\sqrt{N}}, \frac{1}{\sqrt{N}}, \ldots, \frac{1}{\sqrt{N}})^t$ on the N-dimensional unit sphere has $(N-1)N/2$ arcs and the vector $\underline{u}^N$ on this convex set that has the same angle ϕ_N from all the N vectors $\underline{v}^n$ is given by*

$$\underline{u}^N = \cos\phi_N(1, \sqrt{2}-1, \sqrt{3}-\sqrt{2}, \ldots, \sqrt{N}-\sqrt{N-1})^t \tag{2.20}$$

where $0 < \phi_N < \pi/2$ and

$$\cos^2\phi_N = \frac{1}{\sum_{n=1}^{N}(\sqrt{n}-\sqrt{n-1})^2}. \tag{2.21}$$

Proof For $N = 2$ there is one arc between $\underline{v}^1 = (1, 0)^t$ and $\underline{v}^2 = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})^t$. Assume that there are $(n-1)n/2$ arcs for $N = n$. Then, for $N = n+1$, the number of arcs decided by $n+1$ vectors is the sum of the number of arcs decided by the first n vectors, $(n-1)n/2$, and the number of arcs between $\underline{v}^{n+1}$ and the n vectors, n, which is in total $(n-1)n/2 + n = n(n+1)/2$.

Let the vector $\underline{u}^N$ be $(x_1, x_2, \ldots, x_N)^t$ such that $x_1^2 + x_2^2 + \cdots + x_N^2 = 1$ since it is on the unit sphere. Since the angle between two vectors on the unit sphere is

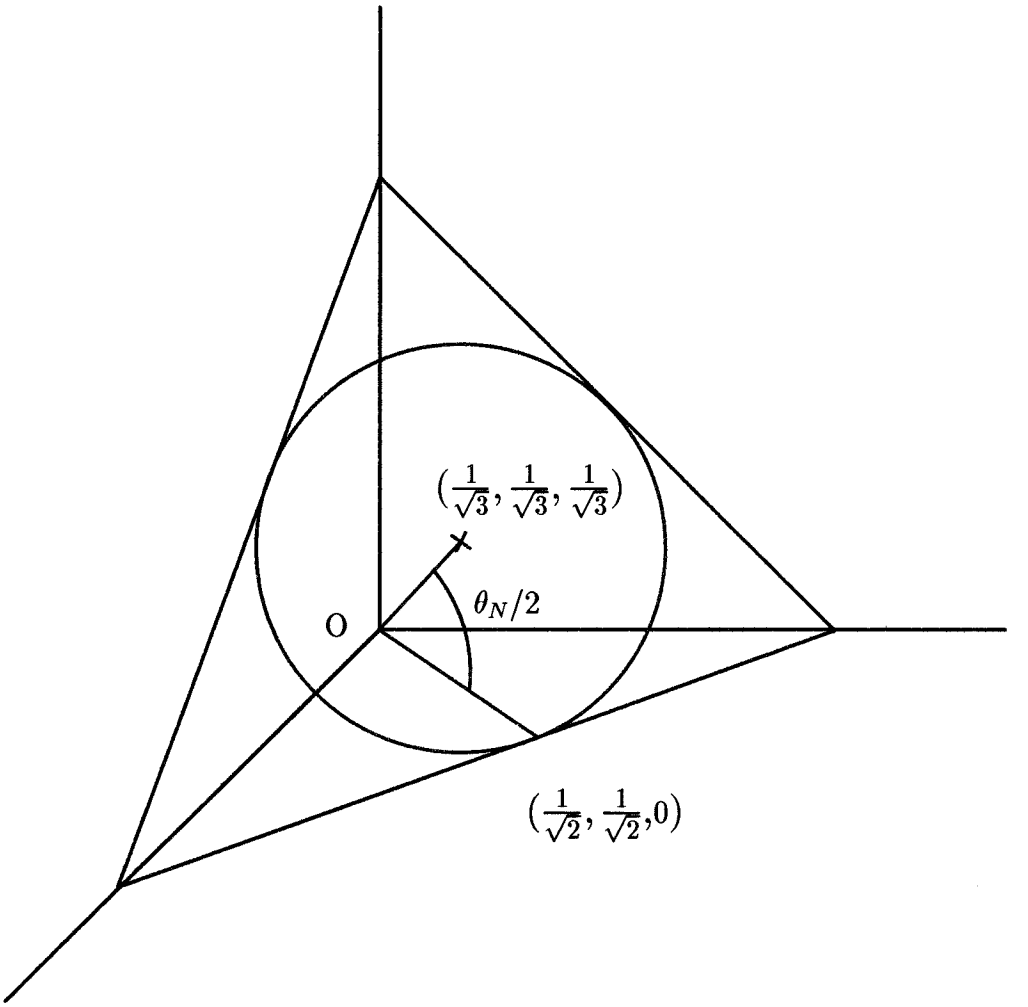


Figure 2.5 An orthant for $N = 3$.

directly related to the inner product the following relationship is obtained:

$$\cos\phi_N = x_1 = \frac{1}{\sqrt{2}}(x_1 + x_2) = \frac{1}{\sqrt{3}}(x_1 + x_2 + x_3) = \cdots = \frac{1}{\sqrt{N}}(x_1 + x_2 + \cdots + x_N). \tag{2.22}$$

Solving x_n $(2 \leq n \leq N)$ in terms of x_1 and substituting them into $\sum_{n=1}^{N} x_n^2 = 1$ we get immediately

$$\cos^2\phi_N = x_1^2 = \frac{1}{\sum_{n=1}^{N}(\sqrt{n} - \sqrt{n-1})^2} \tag{2.23}$$

from which the vector $\underline{u}^N$ is derived.

In two-dimensional case, consider Fig. 2.6. The solution cones whose boundaries contain the vectors and their complements and are perpendicular to the adjacent ones include at least one weight vector up to two inside depending on their orientation because the adjacent ternary weight vectors make an angle of 45 degrees (binary weights are not enough since they may be all on the boundaries).

Fig. 2.7 is for three-dimensional case. First of all, we normalize the weight vectors so that they are imbedded on the unit sphere. Now the orientation of the center axis is considered. Because of the symmetry of the weight vectors on the unit sphere the unit vector along the center axis can be restricted to lie on the convex set that is one sixth of the first octant of unit sphere determined by the vectors $\underline{v}^1$, $\underline{v}^2$, and $\underline{v}^3$ that is illustrated in Fig. 2.7. The question is whether the solution cone will include any ternary weight inside regardless of the rotation of the cone around the center axis whose orientation is also varying. We draw a circular cone of angle $\theta_3/2$ around the center axis and claim that it must contain at least any one of three weight vectors, which is a sufficient condition for the question. It is equivalent to show that the circular cone of angle $\theta_3/2$ around the vector $\underline{u}^3$ defined in Lemma 7 contains all the vectors $\underline{v}^i$ and so covers the entire region we are considering in Fig. 2.7 because of the convexity of the cone. In other words, if this is the case, then when the unit vector along the center axis of the circular cone of angle $\theta_3/2$ is at any point of the

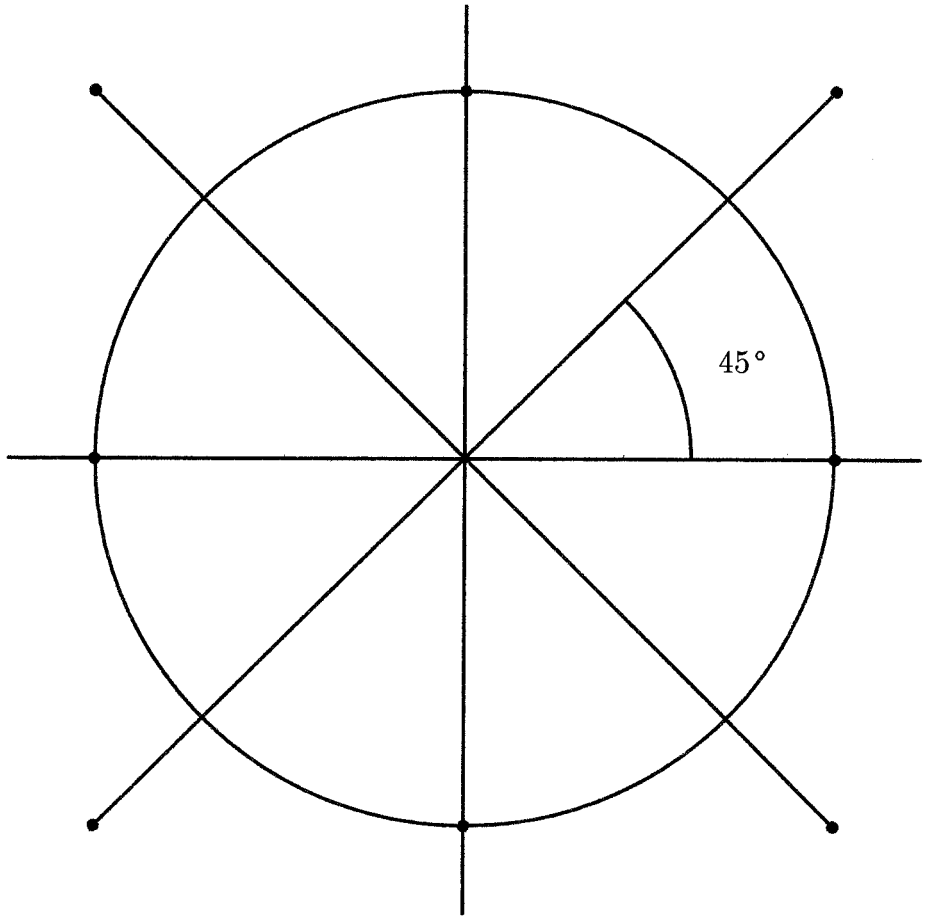


Figure 2.6 Two–dimensional ternary weight space.

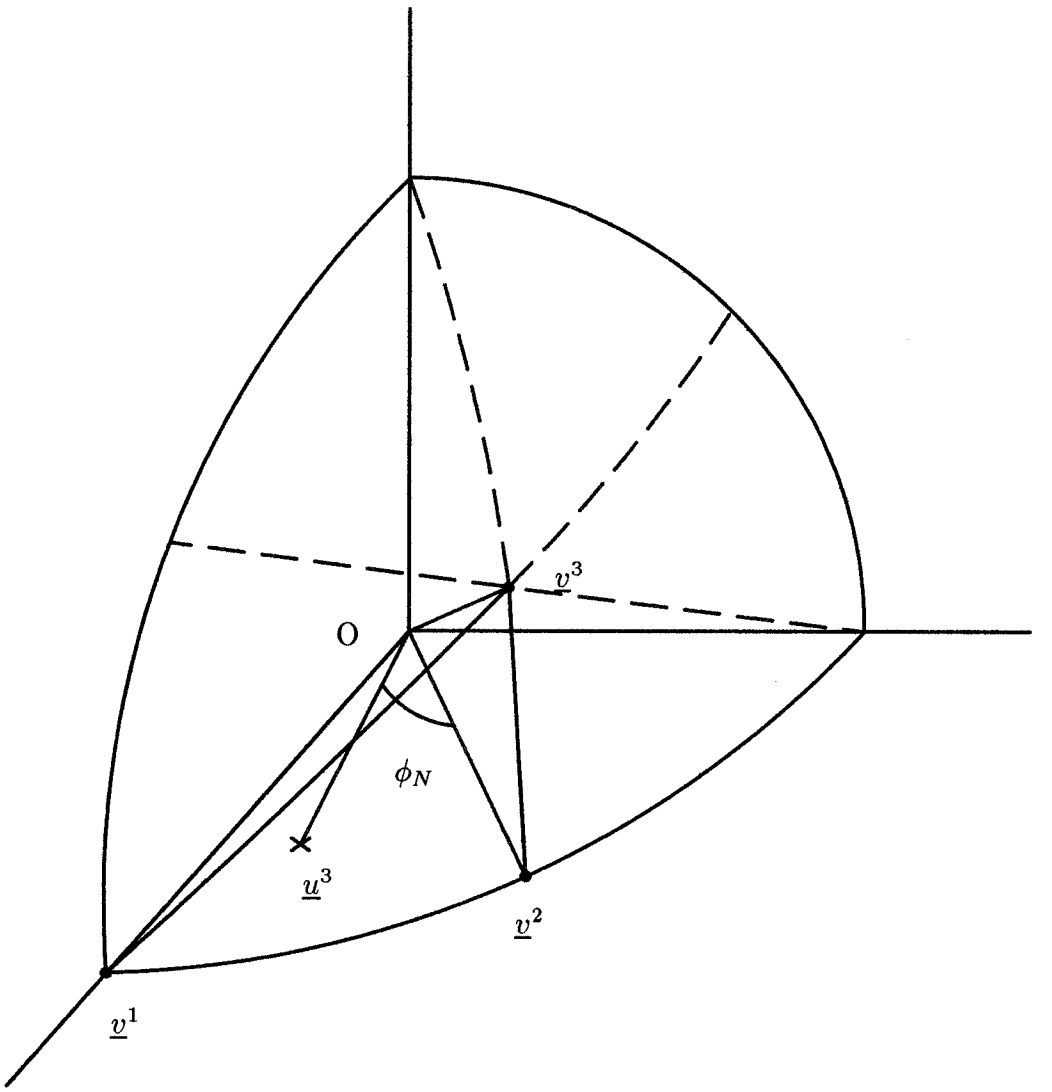


Figure 2.7 Three–dimensional ternary weight space.

convex set but $\underline{u}^N$ the circular cone always contains at least one of the vectors $\underline{v}^1$, $\underline{v}^2$, and $\underline{v}^3$ since at least one of the angles between the unit vector along the center axis and the vectors $\underline{v}^1$, $\underline{v}^2$, and $\underline{v}^3$ becomes smaller than ϕ_3. And it is true since $\cos^2\theta_3/2 = 2/3 < \cos^2\phi_3 \approx .786$, that is, $\theta_3/2 > \phi_3$ by Lemmas 6 and 7.

In general case we normalize the N-dimensional weight vectors into the unit sphere. Then, there are $N!$ regions to be checked to be covered each of which is decided by N unit weight vectors on N-dimensional unit sphere because of the symmetry. One such region is decided by the vectors $\underline{v}^1$, $\underline{v}^2$, ... , and $\underline{v}^N$ as defined in Lemma 7. In order that the circular cone around $\underline{u}^N$ contains the N vectors $\underline{v}^i$, it is required that $\theta_N/2 > \phi_N$ or equivalently $\cos^2\theta_N/2 < \cos^2\phi_N$, which is satisfied for $N \leq 5$. Therefore, we proved the following theorem.

Theorem 8 *N-dimensional orthogonal vectors can be dichotomized arbitrarily by ternary-valued weights for $N \leq 5$.*

Actually, what was used to prove this theorem is a sufficient condition because the circular cone in Lemma 6 is diminishing as the dimensionality increases. It means that the ratio between the volume occupied by the cone and that of the orthant becomes smaller, approaching zero as the dimensionality gets bigger. Therefore, we need another approach to prove the conjecture for the higher dimensional case. But the proof up to five-dimensional cases and the fact that the average number of ternary weights included in one orthant increases exponentially as $3^N/2^N$ are encouraging to the conjecture (e.g., for $N = 5$, 6, 10, 20, and 100, $3^N/2^N \approx 7$, 11, 57, 3.3×10^3, and 4×10^{17}, respectively).

Chapter 3

Algorithmic Optical Computing

3.1 Selection of terms in polynomial expansion

Explained in this section is the application of selecting the terms used in the polynomial expansion algorithmically to meet a given requirement.

> **Motivation** Is there a way of finding a maximum number of terms of r bit products of the original N-dimensional input binary vectors out of $\binom{N}{r}$ terms such that these terms, which constitute the components of the coded vectors, do not share the same two bits with one another and the input bits occur equally in the new vector?

Let N be the dimensionality of the input vector $\underline{x}$ and L be the dimensionality of the coded vector $\underline{z}$ and let m be the number of occurrences of each component of the input vector in its coded vector. Then, the following relationship is obtained:

$$Lr = Nm. \tag{3.1}$$

Intuitively, m can not exceed $(N-1)/(r-1)$ and we will show that when r is a prime number (e.g., $2, 3, 5, 7, 11, \cdots$) and N is a power of r this limit can be achieved and so flipping any one bit of the input vector results in flipping m bits of the corresponding coded vector flipping any two bits resulting in flipping $2m-1$ bits (it is clear that in the special case of $r = 2$ there always exist $\binom{N}{2}$ terms in total that amount to the limit). That is, when $N = r^l$,

$$\begin{aligned} m &= \frac{N-1}{r-1} = \frac{r^l - 1}{r-1} \\ &= r^{l-1} + r^{l-2} + \cdots + r + 1. \end{aligned} \tag{3.2}$$

And

$$L = \frac{Nm}{r} = \frac{N(N-1)}{r(r-1)} = \frac{r^{l-1}(r^l-1)}{r-1}. \tag{3.3}$$

Let n ($n \ll N$) be the Hamming distance between two input vectors and let θ_1 and θ_r be the angles between these two input vectors and between the corresponding two coded vectors, respectively. Then, $\cos\theta_1 = 1-n/N$ and $\cos\theta_r \approx (L-2nm)/L = 1-2r\rho$ for $\rho \ll 1$ where $\rho = n/N$, which correspond to Proposition 4. Thus, they are reduced to a relationship $\theta_r \approx \sqrt{r}\theta_1$ where $\theta_1 \approx 2\sqrt{\rho}$.

As an example, consider the following tables of the index set for $N = 3^2$:

1	2	3
4	5	6
7	8	9

(a)

1	4	7
2	5	8
3	6	9

(b)

1	5	9
2	6	7
3	4	8

(c)

1	6	8
2	4	9
3	5	7

(d)

Table 3.1 Bit selections by index set.

Each row in the tables represents the numbers of the input components that constitute the bit product. For example, 1 2 3 in Table 3.1(a) denotes the term $x_1x_2x_3$. Then, there are 12 bit products satisfying the requirement stated in Motivation and each x_i occurs four times, which coincides with Eqs. 3.2 and 3.3 for $r = 3$ and $N = 9$.

We will prove Eqs. 3.2 and 3.3 using an equivalence relation.

Theorem 9 *Let r be a prime number and let p and n be integers satisfying $0 \le p \le r-1$ and $1 \le n \le r$. Given an integer q_i, let q^i_{np} and m_i be integers satisfying*

$$(n-1)p + q_i = m_i r + q^i_{np} \tag{3.4}$$

where $1 \le q_i \le r$ and $0 \le q^i_{np} \le r-1$. In other words, $q^i_{np} = \{(n-1)p + q_i\} \bmod r$. Then, the following hold:

(a) Given n and p, all q^i_{np} are different for all q_i.

(b) Given p, n_1 and n_2, all $(q^i_{n_1p} - q^i_{n_2p})$ mod r are equal for all q_i.

(c) Given q_i and different n_1 and n_2, all $(q^i_{n_1p} - q^i_{n_2p})$ mod r are different for all p.

Proof (a) Suppose that q^i_{np} and $q^{i'}_{np}$ are the same for different q_i and $q_{i'}$. Then, $q^i_{np} - q^{i'}_{np} = 0$ and $(q^i_{np} - q^{i'}_{np})$ mod $r = \{(n-1)p + q_i - (n-1)p - q_{i'}\}$ mod $r = (q_i - q_{i'})$ mod r. Since $1 \leq q_i, q_{i'} \leq r$, $|q_i - q_{i'}|(\geq 1)$ is less than $r-1$. Therefore, $(q_i - q_{i'})$ mod r is not zero, which is a contradiction.

(b) It is easy to see this because $(q^i_{n_1p} - q^i_{n_2p})$ mod $r = (n_1 - n_2)p$ mod r is independent of q_i.

(c) Suppose that $q^i_{n_1p_1} - q^i_{n_2p_1}$ and $q^i_{n_1p_2} - q^i_{n_2p_2}$ are the same for different p_1 and p_2. Then, it follows that $(n_1 - n_2)p_1 - (m_1 - m_2)r = (n_1 - n_2)p_2 - (m'_1 - m'_2)r$, which can be reduced to

$$(n_1 - n_2)(p_1 - p_2) = \{(m_1 - m_2) - (m'_1 - m'_2)\}r. \tag{3.5}$$

The left–hand side differs from zero by the assumption and therefore r must be a divisor of $(n_1 - n_2)(p_1 - p_2)$. And $1 \leq |n_1 - n_2| \leq r-1$ and $1 \leq |p_1 - p_2| \leq r-1$ since $0 \leq p_1, p_2 \leq r-1$ and $1 \leq n_1, n_2 \leq r$. But r being a prime number gives a contradiction.

This explains the coding method when $N = r^2$ for a prime number r as follows: Every nearest r bits are multiplied together to form a component of $\underline{z}$ vector as shown in Table 3.1(a). Now we are considering the ways of forming bit products of r input components according to Motivation each of which comes from each term constructed already. We shift circularly to the left the second row by p $(0 \leq p \leq r-1)$ bits, the third row by $2p$ bits, and, in general, the n-th $(2 \leq n \leq r)$ row by $(n-1)p$ bits and combine each column to one term as shown in Table 3.1(b) to (d). Then, they all together cover all the possibilities of bit products satisfying the

condition of Motivation (Theorem 9(a)) and Theorem 9(b) and (c) tells us that when r is a prime number they are satisfying the condition and therefore each bit appears exactly r times and in total $r+1$ times that equals an upperbound $(N-1)/(r-1)$, where $q^i_{np}+1$ represents the original number of column that each index in n-th row and q_i-th ($1 \leq q_i \leq r$) column belongs to before it is shifted (the first row is not shifted, i.e., $q^i_{1p} = q_i$ for all p). In order that all of these terms may not share the same two bits with one another the relative bit shifts, $(q^i_{n_1 p} - q^i_{n_2 p}) \bmod r$, must be all different for all p, which holds only when r is a prime number according to Theorem 9(c).

We are ready to generalize this result for $N = r^l$. For $l = 3$ we can make r blocks out of the index set that consists of r^3 indexes such that each of them has r sub–blocks of the nearest r indexes explained as above for $N = r^2$. Relabel the r^2 sub–blocks from 1 to r^2, repeat the same procedure as above and then each new index appears $r+1$ times satisfying the condition of Motivation when r is a prime number. Because one of its occurrences is equivalent to the original r blocks of the nearest r^2 indexes and each of the rest can have r bit products there are $r \cdot r + (r+1)$ terms of r bit products per each indexi. In the general case (i.e., for arbitrary l) r blocks out of the index set can be made to have r^{l-1} indexes and equivalently to have r sub–blocks of the nearest r^{l-2} indexes. Relabel the r^2 sub–blocks and repeat the coding procedure again where the number of occurrence of each input component in each coding that satisfies the condition of Motivation is given by $r \cdot r^{l-2} + (r^{l-2} + \cdots + r + 1)$ when r is prime number. Then, we get $r \cdot r^{l-1} + (r^{l-1} + r^{l-2} + \cdots + r + 1)$ bit products, which is consistent with Eq. 3.2, because one of the codings of the new index set is the original r blocks of the nearest r^{l-1} indexes. Therefore, we proved the following theorem.

Theorem 10 *When r is a prime number and N is a power of r there can exist $N(N-1)/r(r-1)$ terms of r bit products out of N input bits that satisfy the following conditions:*

(a) They do not share two same bits with one another.

(b) The input bits occur equally in those products.

3.2 Deterministic orthogonalization algorithm

The orthogonalization of binary vectors by complete polynomial expansion and the ternarization of weights that were discussed in the previous chapter provide us with a useful tool for a deterministic algorithm.

A direct implementation with a memory of size N_0 times 2^N using a simple outer product rule can be done according to Eq. 2.10 as shown in Fig. 3.1,

$$w_{li} = \sum_{m=1}^{M} y_l^m z_i^m \quad (1 \leq M \leq 2^N). \tag{3.6}$$

Generalization can be achieved by only pre-imposing the overall structure on the memory in a deterministic way according to the designer's purpose and this kind of memory deals with the inputs in a direct way. But as mentioned before, this is out of the question for large N because the memory size is exceedingly huge. The number of separate levels of w_{li}, K, can be $2M+1$ from $-M$ to M that does not depend on the input dimensionality and so L. The expectation value of w_{li} is zero assuming that the outputs y_l^m are selected randomly and independently of the stored data $\underline{x}^m$, and its standard deviation is given by $M^{1/2}$ that is not negligible for implementation when the full capacity is expended. This dynamic range problem may be solved by ternarizing the weight values as discussed in the previous chapter, which will be explained in more detail in this section. Then, according to Theorem 1, the inefficiency is $\log_2 3$ bits, which therefore is the smallest one that we can achieve with linear discriminant functions. The role of $\log_2 3$ can be considered as choice of

necessary terms out of 2^N terms.

Then how can the "2^N" problem be resolved? One of the possible solutions is introducing feature extraction[1, 11]. If it is possible to extract $\log_2 L$ binary features from N-dimensional binary pattern vectors where L is the order of power of N, then 2^N reduces to L, which is a reasonable value, when the orthogonalization algorithm is used. But this is still an open problem that depends on the training samples. Another possible way is using orthogonalization by numbering. When all of the 2^N samples are not required to be stored in the memory and only a part of them whose number M is the polynomial of N are under consideration given arbitrary M samples, we label them by numbers from 1 to M that can be represented as binary vectors of $\lceil \log_2 M \rceil$ bits where $\lceil \cdot \rceil$ denotes the smallest integer greater than or equal to the argument inside. Regardless of the input dimensionality, these binary vectors in turn become $2^{\lceil \log_2 M \rceil}$-dimensional orthogonal binary vectors $\underline{z}^m$ when they are fully expanded where $2^{\lceil \log_2 M \rceil}$ is approximately M since $M \leq 2^{\lceil \log_2 M \rceil} < 2M$. Now we apply the learning rule in Eq. 3.6 resulting in the reduced memory size that can be handled.

These are easy learnings like higher order memories with outer product rule in a sense that the learning process can be finished in a polynomial time in terms of the input dimensionality with the samples whose number is also a polynomial of the input dimensionality.

We will discuss the algorithm for the ternarization of weights based on the argument given in the previous chapter where the possibility of ternarizing the weights for the orthogonal vectors was shown. A simple but brute force way of ternarizing weights for arbitrary orthogonal vectors is trying all the ternary weights one by one until we find one that fits the set of training samples. There are 3^N ternary weights to be considered for the N-dimensional orthogonal vectors.

Alternatively, a simple ternarization algorithm is presented with geometric illus-

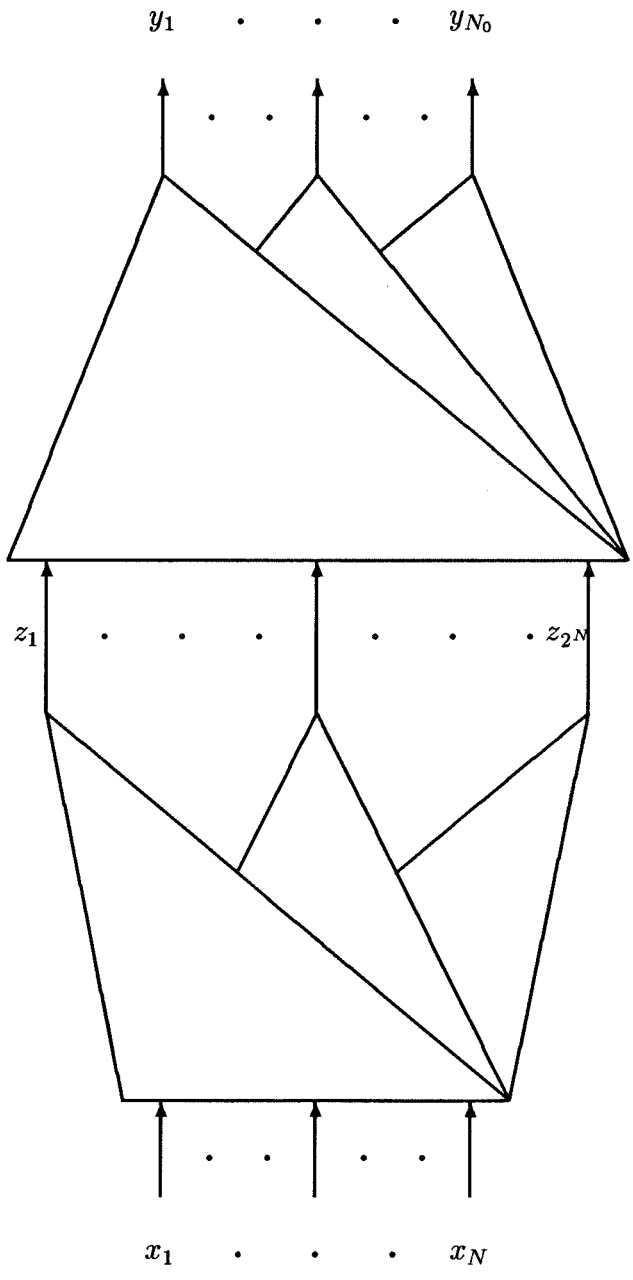


Figure 3.1 Full expansion.

trations. We can modify Eq. 3.6 without changing the overall result as follows:

$$w_{li} = \sum_{m=1}^{2^{N_1}} \alpha_m y_l^m z_i^m \tag{3.7}$$

where α_m is a learning coefficient for the m-th training sample. As long as each α_m is greater than zero it lies within the solution cone. Take a look at the following ternarization:

$$w_{li} = \text{sgnn}\Big\{\sum_{m=1}^{2^{N_1}} \alpha_m y_l^m z_i^m\Big\} \tag{3.8}$$

where for $n \geq 0$

$$\text{sgnn } x = \begin{cases} +1 & \text{if } x > n \\ -1 & \text{if } x < -n \\ 0 & \text{otherwise.} \end{cases}$$

In the case of binary samples, the expanded orthogonal vectors, $\underline{z}^m$, of 2^{N_1} dimensions are also binary that decide the solution cones of the weights. The vectors $y_l^m \underline{z}^m$ are themselves part of the ternary weight vectors. The geometry of the distribution of ternary weights from the center axis of the solution cone to each of those weights that decide the solution cone is symmetric and therefore α_m may be set to be all one. We start with α_m being all one and n zero, test the weights derived by Eq. 3.8 whether they satisfy the training samples, otherwise increment n by one and do this procedure until a solution is found where n can not exceed 2^{N_1}, which means that when 2^{N_1} is a polynomial of N this procedure will be completed in a polynomial time in terms of N. A simulation shows that all of the 2^4-dimensional expanded binary orthogonal vectors can be stored with n not exceeding two.

But for the arbitrary N-dimensional orthogonal vectors the distribution of the ternary weights is not symmetric inside the solution cone. Thus we may need learning iterations. Recall that each solution cone includes on average ternary weights as many as about $3^{N_1}/2^{N_1}$ that is exponentially huge (e.g., $3^{N_1}/2^{N_1} \approx 57$ for $N_1 = 10$ and $3^{N_1}/2^{N_1} \approx 3.3 \times 10^3$ for $N_1 = 20$). We start with α_m being all one and n zero,

test the weights and if there are some vectors that are not stored correctly, then add those to the sum,

$$\begin{aligned} w_{li} &= \mathrm{sgn0}\Big\{\sum_{m=1}^{M} y_l^m z_i^m + \sum_{m'} y_l^{m'} z_i^{m'}\Big\} \\ &= \mathrm{sgn0}\Big\{\sum_{m=1}^{M} \alpha_m y_l^m z_i^m\Big\} \end{aligned} \tag{3.9}$$

where each α_m is either one or two and $1 \leq M \leq N$. If it does not give the solution in some finite steps, say, M steps then increment n by one and do the same procedure until a solution is found where n can be at most M. It can be written in general

$$w_{li} = \mathrm{sgnn}\Big\{\sum_{m=1}^{M} \alpha_m y_l^m z_i^m + \sum_{m'} y_l^{m'} z_i^{m'}\Big\}. \tag{3.10}$$

Therefore, this procedure will be completed in a polynomial time in terms of input dimensionality N, again.

The algorithm we have shown may provide a method of choosing terms in polynomial expansion in the first layer for generalization and is closely related to data compression.

3.3 Random iterational algorithm

The advantages of optical computing are known to be high interconnectivity and three-dimensional (analog) processing due to volume holograms[32]. The purposes of the computers may be large capacity of memory and high speed that comes from algorithmic power of the architecture[2]. Keeping this in mind we can ask how high a speed we can achieve with optical computing whereas high storage capacity can be achieved by higher order memories. With the above aspects, the speed, defined by the number of operations per unit cycle, reaches at most a constant times the speed of conventional digital computers. That is,

$$T_o = C \times T_d \tag{3.11}$$

where T_o is the number of operations per unit cycle achievable by optical computing with the above qualities, T_d that by conventional digital computers and C is at most the number of interconnections afforded by volume holograms, which is approximately 10^9 as will be discussed in Chapter 4. In a sense of algorithmic computation the speed-up by constant-times does not mean much because it can not resolve the limitation of the digital computers resulting from computational problems such as computability[1], NP-completeness[19] and so on.

Then let's see what would happen if we can achieve the following relationship:

$$T_o = C^n \times T_d \tag{3.12}$$

where n is the number of iterations (or cycles). Local interconnections are also possible in digital computers but full interconnections that can be provided by optics are out of the question[15, 32]. In order for general computing, capability of full interconnections is required.

Consider the optimization problems that require lots of iterations(or feedbacks)[18] such as those shown in Fig. 4.7. It consists of matrix tensor multiplication, which has 10^6 multiplication and 10^6 summation operations per each of 10^3 channels, and feedback.

Another simple example is shown in Fig. 3.2 that explains how to generate by iterations the Vicsek pattern[23] whose dimensionality is $\log_3 5$ and how to decide its dimensionality, which is related to data decompression.

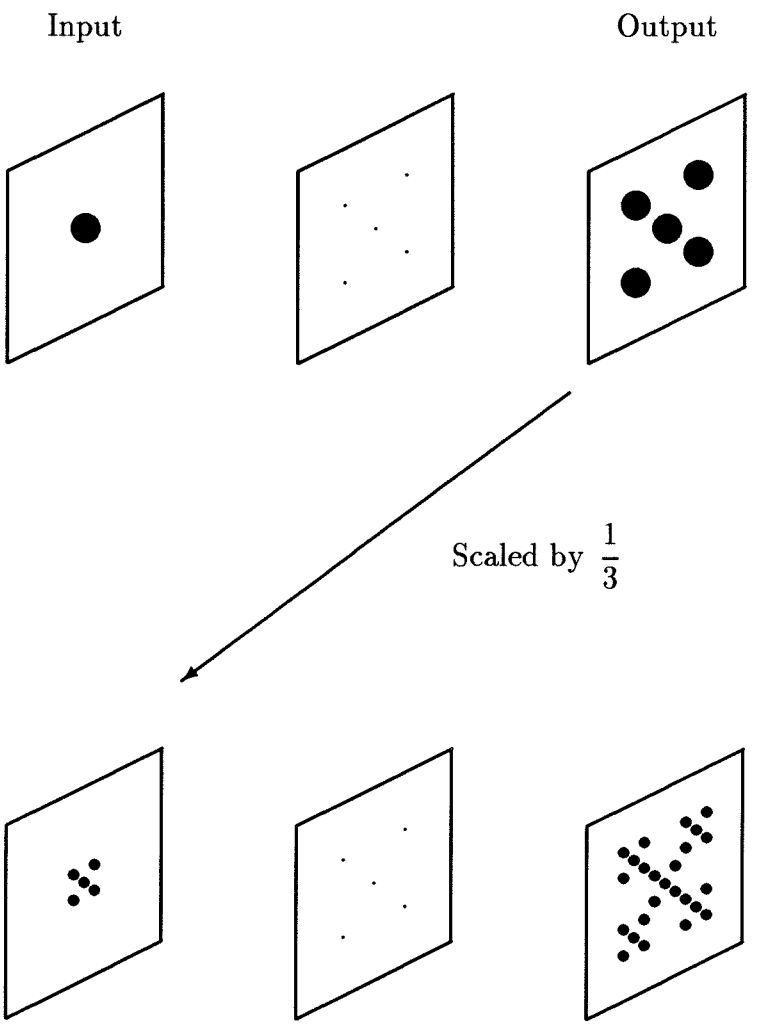


Figure 3.2 Generation of Vicsek pattern.

Chapter 4

Higher Order Associative Memories

4.1 Training of higher order memories using outer product rule

Consider again the r-th order expansion given in Eq. 2.4 in a different form,

$$y_l = \mathrm{sgn}\Big\{\sum_{k=1}^{r}\sum_{j_1\cdots j_k} w_{lj_1\cdots j_k} x_{j_1}\cdots x_{j_k} + w_0\Big\}. \tag{4.1}$$

Then the capacity was given by

$$C = \sum_{k=0}^{r}\binom{N+k-1}{k} = \binom{N+r}{r} \tag{4.2}$$

which can be approximated by $\binom{N+r-1}{r}$ that originates from the highest order, that is, r-th order terms only for $r \ll N$ and a large N. Therefore, we can replace the original expansion by the single expansion of the highest order with the same order of capacity.

Once the initial polynomial mapping has been selected, the rest of the system in a higher order memory is simply a linear discriminant function. As such it can be trained by any of the existing methods for training linear discriminant functions. For instance, the pseudoinverse[21, 31, 48] can be used to calculate the set of weights that will map a set of L-dimensional expanded vectors $\underline{z}^m$ to the associated output vectors $\underline{y}^m$. Alternatively, error driven algorithms such as the perceptron or adaline can be used to iteratively train the memory by repeatedly presenting the input vectors to the system, monitoring the output to obtain an error signal, and modifying the weights to gradually decrease the error. The relative ease with which higher order memories can be trained is a very important advantageous feature of this approach. A higher order memory is basically a multilayered network where the first layer is

selected *a priori*. In terms of capacity alone, there is no advantage whatsoever in having multiple layers with modifiable weights. From Theorem 1 we know that at best the capacity is determined by the number of modifiable weights. For a higher order memory we get the full advantage of the available degrees of freedom whereas if we put the same number of weights in multiple layers the resulting degeneracies will decrease the capacity. The relative advantage of trainable multiple layers is the potential for generalization that emerges through the learning process.

The sum of outer products algorithm that has been used extensively for training linear associative memories can also be used for training the higher order memories and this algorithm generalizes to the higher order case in particularly interesting ways. In addition, this particular learning algorithm is predominantly used for the holographic optical implementations that are described in the following section. Therefore, we will discuss in some detail the properties of higher order memories that are trained using this rule.

Let us consider associative memories constructed as an expansion of the r-th order only with input samples in N-dimensional binary space and $r \geq 1$. Then

$$y_l = \text{sgn}\Big\{ \sum_{j_1 j_2 \cdots j_r} w_{l j_1 j_2 \cdots j_r} x_{j_1} x_{j_2} \cdots x_{j_r} \Big\} \tag{4.3}$$

where $1 \leq j_1, j_2, \cdots, j_r \leq N$, and $1 \leq l \leq N_0$. The number of independent terms L in the r-th order expansion is approximately $N^r/r!$ for $r \ll N$.

The expression for the weights of the r-th order expansion using the sum of outer products algorithm[7, 36, 38] is

$$w_{l j_1 j_2 \cdots j_r} = \sum_{m=1}^{M} y_l^m x_{j_1}^m x_{j_2}^m \cdots x_{j_r}^m \tag{4.4}$$

where M is the number of vectors stored in the memory, $\underline{y}^m$ is an output vector associated with a memory vector $\underline{x}^m$ as before. It is interesting to notice that each weight in this case can have $2M + 1$ integral levels from $-M$ to M independently

of the order of expansion or equivalently L. It has a binomial distribution whose expectation value and standard deviation are given by zero and $M^{1/2}$, respectively, with the assumption that the input and the output components are selected randomly and independently. Shown in Fig. 4.1 is the implementation with this weight tensor for $r = 2$. The outer product $x_j x_k$ is formed and it is interconnected to each output y_i by weight tensor w_{ijk}; since the diagonal terms x_j^2 in this case are all 1, only nondiagonal terms contribute to form the output. With the above expression for the weight tensor Eq. 4.3 can be rewritten as follows:

$$y_l = \mathrm{sgn}\Big\{\sum_{m=1}^{M} y_l^m (\sum_{j=1}^{N} x_j^m x_j)^r\Big\}. \tag{4.5}$$

The above equation suggests an alternative implementation for higher order memories that are trained using the outer product rule. This is shown schematically in Fig. 4.2. The inner products between the input vector and all the stored vectors $\underline{x}^m$ are formed first, then raised to the r-th power, and the signal from the m-th unit is connected to the output through interconnective weights y_l^m instead of using an $(r+1)$-th order tensor $w_{l j_1 j_2 \cdots j_r}$.

The second alternative implementation is to use nonlinear interconnections as shown in Fig. 4.3 for a quadratic memory. The basic idea behind this scheme is that the interconnection weight w_{ij} between the i-th and the j-th neurons is determined by not only the states of these neurons themselves (Hebbian learning)[17, 21] but also the states of the other neurons as a sum of the interactions of r neurons:

$$\begin{aligned} y_i &= \mathrm{sgn}\Big\{\sum_{j_1}(\sum_{j_2 \cdots j_r} w_{i j_1 j_2 \cdots j_r} x_{j_2} \cdots x_{j_r}) x_{j_1}\Big\} \\ &= \mathrm{sgn}\Big\{\sum_{j} w_{ij} x_j\Big\}, \end{aligned} \tag{4.6}$$

where

$$w_{ij} = \sum_{j_2 \cdots j_r} w_{i j j_2 \cdots j_r} x_{j_2} \cdots x_{j_r}. \tag{4.7}$$

If $\underline{y}^m = \underline{x}^m$, then the memory is autoassociative, and in this case the output can be

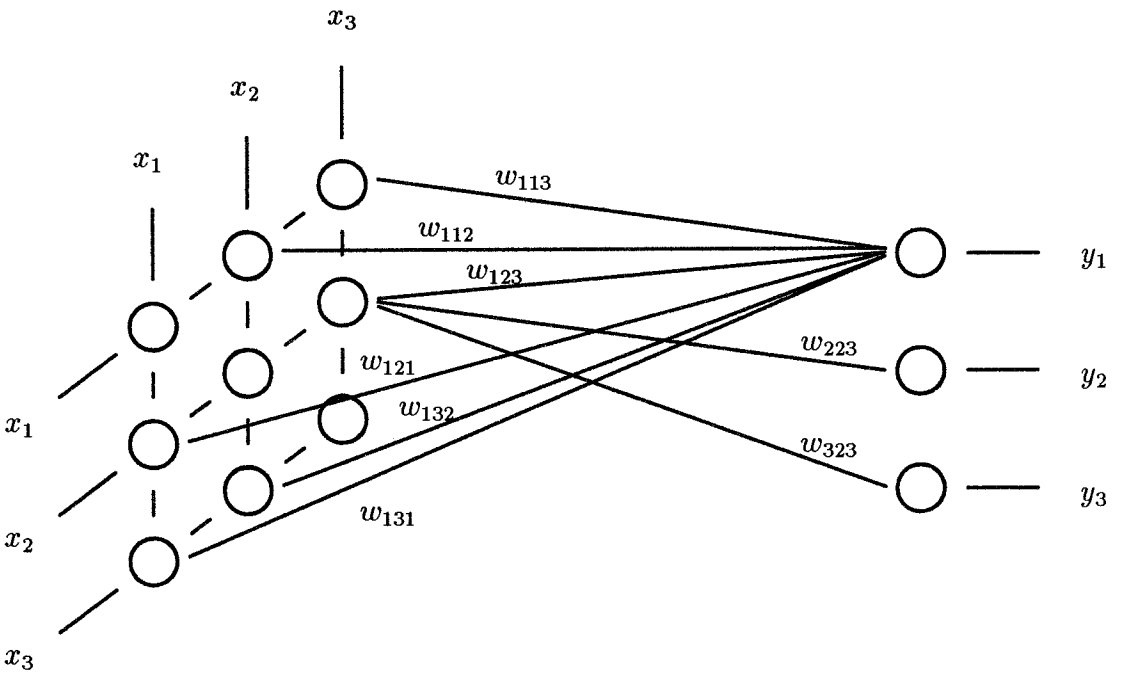


Figure 4.1 Quadratic associative memory for $N = 3$.

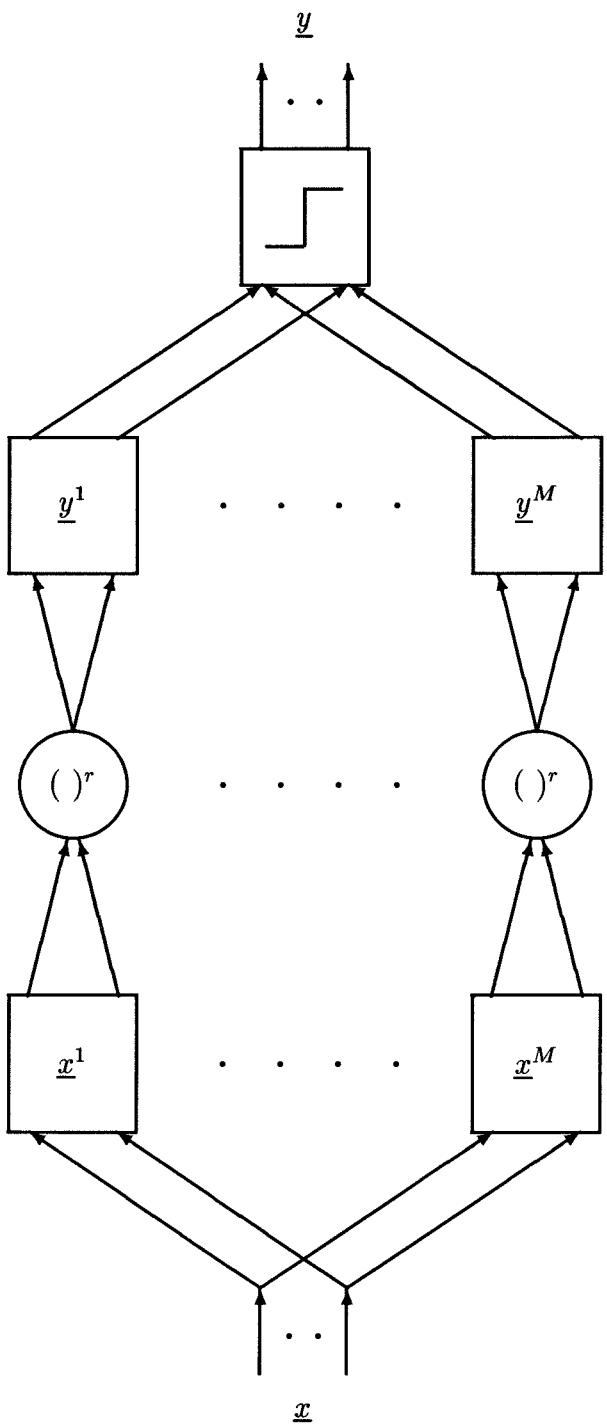


Figure 4.2 Outer product, r-th order associative memory.

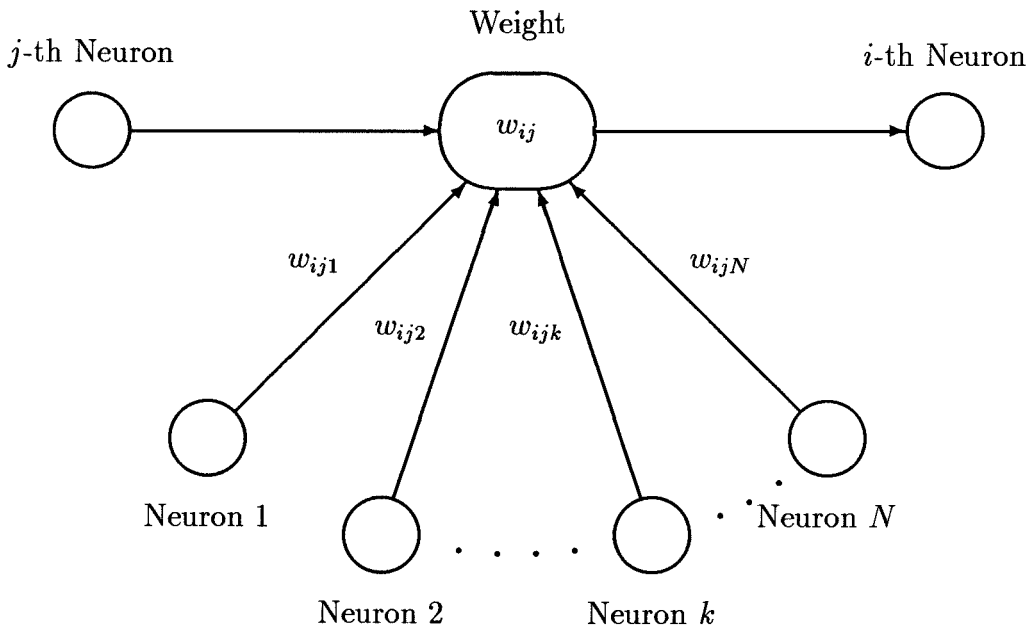


Figure 4.3 Quadratic mappings implemented as nonlinear interconnections.

fed back to the input resulting in a system whose stable states are programmed to be the vectors $\underline{x}^m$. This becomes a direct extension of the Hopfield network[4, 17, 26] to the higher order case.

4.2 Calculation of signal-to-noise ratio

Assuming that $\underline{x} = \underline{x}^n$ is one of the stored vectors, y_l becomes

$$\begin{aligned} y_l &= \mathrm{sgn}\Big\{N^r y_l^n + \sum_{m\neq n} y_l^m \Big(\sum_{j=1}^{N} x_j^m x_j^n\Big)^r\Big\} \\ &= \mathrm{sgn}\Big\{N^r y_l^n + n_l(\underline{x}^n)\Big\} \end{aligned} \tag{4.8}$$

where the first term is the desired signal term, n_l is a noise term, and thresholding weight is set to zero.

The expectation value of $n_l(\underline{x}^n)$ is zero if the bits that comprise the stored binary input and output vectors are drawn randomly and independently having equal probabilities of being $+1$ or -1. If this is the case, then the term $\sum_{j=1}^{N} x_j^m x_j^n$ for $m \neq n$ will have a binomial distribution with zero average and a variance of N, since

$$\mathrm{E}\Big(\sum_{tt'} x_t^m x_{t'}^m\Big) = \sum_{tt'} \delta_{tt'}, \quad \mathrm{E}\Big(\sum_{mm'} x_t^m x_t^{m'}\Big) = \sum_{mm'} \delta_{mm'} \tag{4.9}$$

where δ_{ij} is the Kronecker delta function. The variance of n_l is calculated as a $2r$-th moment of the binomial distribution,

$$\begin{aligned} \mathrm{E}(n_l^2) &= (M-1)\sum_{k=0}^{N}(N-2k)^{2r}\binom{N}{k}\Big(\frac{1}{2}\Big)^N \\ &= (M-1)\sum_{j=0}^{2r} 2^j N^{2r-j}\Big(\frac{1}{2}\Big)^N \sum_{k=0}^{N} k^j \binom{N}{k}, \end{aligned} \tag{4.10}$$

where $\left(\frac{1}{2}\right)^N \sum_{k=0}^{N} k^j \binom{N}{k}$ is the j-th moment of binomial distribution. The variances are, thus, given by $(M-1)N$, $(M-1)(3N^2-2N)$, $(M-1)(15N^3-30N^2+16N)$, and so on for the cases r=1, 2, 3, $\cdots$. We can estimate easily the variance for a large r by approximating the binomial distribution as a Gaussian distribution

using DeMoivre–Laplace Theorem[30] such that the variance is a $2r$-th moment of Gaussian distribution of zero average and a variance N. Then the variance becomes asymptotically $(M-1)N^r(2r)!/(2^r r!)$.

The following is another approach to calculate the noise variance:

$$\begin{aligned} \mathrm{E}(n_l^2) &= \mathrm{E}\Big(\sum_{m\neq n}\sum_{m'\neq n} y_l^m y_l^{m'} \sum_{j_1 j_2\cdots j_r}\sum_{s_1 s_2\cdots s_r} x_{j_1}^m x_{j_2}^m \cdots x_{j_r}^m x_{j_1}^n x_{j_2}^n \cdots x_{j_r}^n \\ &\qquad\qquad x_{s_1}^{m'} x_{s_2}^{m'} \cdots x_{s_r}^{m'} x_{s_1}^n x_{s_2}^n \cdots x_{s_r}^n\Big) \\ &= \mathrm{E}\Big(\sum_{m\neq n}\sum_{j_1 j_2\cdots j_r}\sum_{s_1 s_2\cdots s_r} x_{j_1}^m x_{j_2}^m \cdots x_{j_r}^m x_{s_1}^m x_{s_2}^m \cdots x_{s_r}^m \\ &\qquad\qquad x_{j_1}^n x_{j_2}^n \cdots x_{j_r}^n x_{s_1}^n x_{s_2}^n \cdots x_{s_r}^n\Big). \end{aligned} \tag{4.11}$$

In the above, we used the facts that different stored vectors are uncorrelated (i.e., for $m \neq m'$) and $y_l^2 = 1$. Then, the variance becomes $(M-1)\mathrm{Q}(N,r)$, where $\mathrm{Q}(N,r)$ is the number of possible permutations such that

$$\delta_{i_1 t_1}\delta_{i_2 t_2}\cdots\delta_{i_r t_r} = 1 \tag{4.12}$$

where the index set $\{i_1,\ldots,i_r,t_1,\ldots,t_r\}$ spans all the possible combinations produced by the index set $\{j_1,\ldots,j_r,s_1,\ldots,s_r\}$ in Eq. 4.11. The variances driven from this approach are the same as those from the previous approach. We will derive lower and upper bounds that for large N provide us with a good estimate of the variance for any order r, which will justify our previous argument.

Proposition 11 *The total number of permutations, $Q(N,r)$, for which Eq. 4.12 holds, satisfies the following relationship:*

$$P(N,r)\frac{(2r)!}{2^r r!} + \binom{2r}{4} P(n,r-1)\frac{(2r-4)!}{2^{r-2}(r-2)!} \leq Q(N,r) \leq N^r \frac{(2r)!}{2^r r!} \tag{4.13}$$

where $P(m,n) \equiv m!/(m-n)!$.

Proof The number of ways of making r pairs of $2r$ items is $(2r-1)(2r-3)\cdots(3)(1) = (2r)!/2^r r!$. The items that we are concerned with are the indexes i_j, t_j, and each of

these indexes can take one of N values. We can only select the values of half these variables (N^r possibilities) and for each of these choices we can create r pairs. Hence the upperbound is $N^r(2r)!/2^r r!$ that is the same as one derived from the Gaussian approximation. This is an upper bound because we have overcounted for different pairings of variables that have the same value.

The initial lower bound is derived if each pair has a different value from all others, which eliminates the possibility of overcounting. The number of possible ways to satisfy Eq. 4.12 with the indexes in any two pairs not taking the same values is $\mathrm{P}(N,r)(2r)!/2^r r!$. This is an underestimate because all pairs that contain indexes taking the same value should be counted once. We can thus improve the lower bound by counting the number of ways these degenerate pairings occur and adding them into the previous bound. For example, when two pairs out of r have the same values with $\binom{2r}{4}$ choices, there are $\binom{2r}{4}N\mathrm{P}(N-1,r-2)(2r-4)!/2^{r-2}(r-2)!$ possible permutations where $(2r-4)!/2^{r-2}(r-2)!$ is the number of ways of making $r-2$ pairs of $2r-4$ items. Therefore, $\mathrm{Q}(N,r)$ is lower bounded by $\mathrm{P}(N,r)(2r)!/2^r r! + \binom{2r}{4}\mathrm{P}(N,r-1)(2r-4)!/2^{r-2}(r-2)!$, since $N\mathrm{P}(N-1,r-2) = \mathrm{P}(N,r-1)$.

We can get a very good approximation to the SNR using the approximations of $M-1 \approx M$ and $\mathrm{Q}(N,r) \approx N^r(2r)!/2^r r!$ which are very nearly true for the interesting case $r \ll N$:

$$\begin{aligned} \mathrm{SNR} &\approx \frac{N^r}{\{MN^r(2r)!/2^r r!\}^{1/2}} \\ &= \left\{\frac{N^r}{M}\frac{2^r r!}{(2r)!}\right\}^{1/2}. \end{aligned} \tag{4.14}$$

For example, the linear memory, $r=1$, has an SNR $\approx (N/M)^{1/2}$, the quadratic memory, $r=2$, an SNR of $N/(3M)^{1/2}$ and the cubic memory, $r=3$, an SNR of $(N^3/15M)^{1/2}$.

The diagonal terms in a high order memory $w_{lj_1j_2\cdots j_r}$ can be defined as those

of which all the indexes j are not different. We form the weight tensor with zero diagonal as follows:

$$w_{lj_1j_2\cdots j_r} = \begin{cases} \sum_m y_l^m x_{j_1}^m x_{j_2}^m \cdots x_{j_r}^m & \text{if } j\text{s are all different,} \\ 0 & \text{otherwise.} \end{cases} \tag{4.15}$$

When the input is one of the stored vectors $\underline{x}^n$ and the weight tensor has zero diagonal, the output y_l becomes

$$\begin{aligned} y_l &= \operatorname{sgn}\Big\{ \sum_{\text{different } j} w_{lj_1j_2\cdots j_r} x_{j_1}^n x_{j_2}^n \cdots x_{j_r}^n \Big\} \\ &= \operatorname{sgn}\Big\{ \mathrm{P}(N,r) y_l^n + \sum_{m\neq n} y_l^m \sum_{\text{different } j} x_{j_1}^m x_{j_2}^m \cdots x_{j_r}^m x_{j_1}^n x_{j_2}^n \cdots x_{j_r}^n \Big\} \end{aligned} \tag{4.16}$$

where the first term is a signal term and the second a noise term as before. The variance of the noise term is easily shown to be $(M-1)\mathrm{P}(N,r)r!$ using Eq. 4.9. Therefore, the SNR becomes

$$\mathrm{SNR} = \left\{ \frac{\mathrm{P}(N,r)}{(M-1)r!} \right\}^{1/2} \approx \left\{ \binom{N}{r} / M \right\}^{1/2} \tag{4.17}$$

which can be approximated as $(N^r/Mr!)^{1/2}$ for $r \ll N$.

4.3 Capacity

The SNRs that are calculated in the previous section are ready to be used for deriving capacities of higher order associative memories of two cases[25, 48]: one is the case that most of the stored data are recalled correctly and another is the case that almost all of them are recalled correctly.

The idea for the former case is

$$\mathrm{P}(\text{correct}) = \left(1 - \mathrm{P}_e\right)^N = 1 - \epsilon \tag{4.18}$$

since $1 - N\mathrm{P}_e \approx (1-\mathrm{P}_e)^N \approx \exp(-N\mathrm{P}_e)$ for $N\mathrm{P}_e \ll 1$, whereas for the latter

$$\mathrm{P}(\text{correct}) = \left(1 - \mathrm{P}_e\right)^{MN} = 1 - \epsilon, \tag{4.19}$$

where $(0 <)\epsilon \ll 1$ and P_e is the probability of error per each bit under the assumption that noise has Gaussian distribution according to the central limit theorem[30]:

> Let $\{\mathbf{x}_i\}$ be a sequence of mutually independent random variables and let $\mathbf{x} = \mathbf{x}_1 + \cdots + \mathbf{x}_n$. Then the density of $\mathbf{x}$ properly normalized tends to a normal distribution as $n \to \infty$.

In the first case we can easily obtain an estimate for the capacity of an r-th order memory by equating the signal to noise ratios of the linear and r-th order memories and solving for M_r, the number of stored vectors that will yield the equality, since P_e is a function of SNR. For r small compared to N, comparing its value with the capacity M_1 of a linear memory[25, 48] we can obtain the relationship between the capacities,

$$\frac{M_r}{M_1} = N^{r-1}\frac{2^r r!}{(2r)!}. \tag{4.20}$$

And therefore the capacity is given by

$$M_r = \frac{N^r 2^r r!}{2\log N\,(2r)!}. \tag{4.21}$$

For example M_2 of a quadratic memory is $M_1 N/3$ and M_3 of a cubic memory is $M_1 N^2/15$. For large N increases in capacity are huge.

The capacity of the r-th order memory with zero diagonal can be shown compared with that of a linear memory to be

$$\frac{M_r}{M_1} = \frac{\binom{N}{r}}{N} \tag{4.22}$$

which is approximately $N^{r-1}/r!$ for $r \ll N$ and a large N. And therefore

$$M_r = \frac{\binom{N}{r}}{2\log N} \approx \frac{N^r}{2r!\log N}. \tag{4.23}$$

It turns out that this approaches more closely the capacity bound than the first scheme.

In the second case we need a little more algebra. From the assumption of Gaussian distribution the probability of error can be derived

$$\mathrm{P}_e = Q(\mathrm{SNR}) \tag{4.24}$$

where

$$Q(u) = \frac{1}{\sqrt{2\pi}} \int_u^\infty e^{-t^2/2} dt. \tag{4.25}$$

With the approximation that $Q(u) \approx \frac{1}{\sqrt{2\pi}u} e^{-u^2/2}$ the capacity of an r-th order memory can be derived from Eq. 4.19

$$M_r = \frac{N^r 2^r r!}{2(1+r)\log N\,(2r)!}. \tag{4.26}$$

For the zero diagonal memories, the capacity can be derived from the same procedure,

$$M_r = \frac{\binom{N}{r}}{2(1+r)\log N} \approx \frac{N^r}{2(r+1)!\log N}. \tag{4.27}$$

4.4 Higher order bidirectional associative memories

In what follows we will consider briefly the case that interconnection weights of higher order memories are bidirectional so that the output in the previous sections also works as an input inversely[22, 45]. Let us take a look at Fig. 4.2 with modification that the output $\underline{y}$ is also applied to the system as an input in the reverse direction and the input $\underline{x}$ can be obtained by thresholding the value in the way that $\underline{y}$ was obtained. Then, $\underline{x}$ and $\underline{y}$ are related as follows:

$$x_i = \mathrm{sgn}\Big\{ \sum_{m=1}^{M} x_i^m \Big(\sum_{j=1}^{N_0} y_j^m y_j\Big)^r \Big\}, \tag{4.28}$$

$$y_j = \mathrm{sgn}\Big\{ \sum_{m=1}^{M} y_j^m \Big(\sum_{i=1}^{N} x_i^m x_i\Big)^r \Big\}. \tag{4.29}$$

More generally, multiple associations can be taken into account as follows[7]: as a simplest case,

$$w_{ijk} = \sum_m x_i^m y_j^m z_k^m. \tag{4.30}$$

Then,

$$x_i = \mathrm{sgn}\Big\{\sum_{jk} w_{ijk} y_j z_k\Big\}, \tag{4.31}$$

$$y_j = \mathrm{sgn}\Big\{\sum_{ik} w_{ijk} x_i z_k\Big\}, \tag{4.32}$$

$$z_k = \mathrm{sgn}\Big\{\sum_{ij} w_{ijk} x_i y_j\Big\}. \tag{4.33}$$

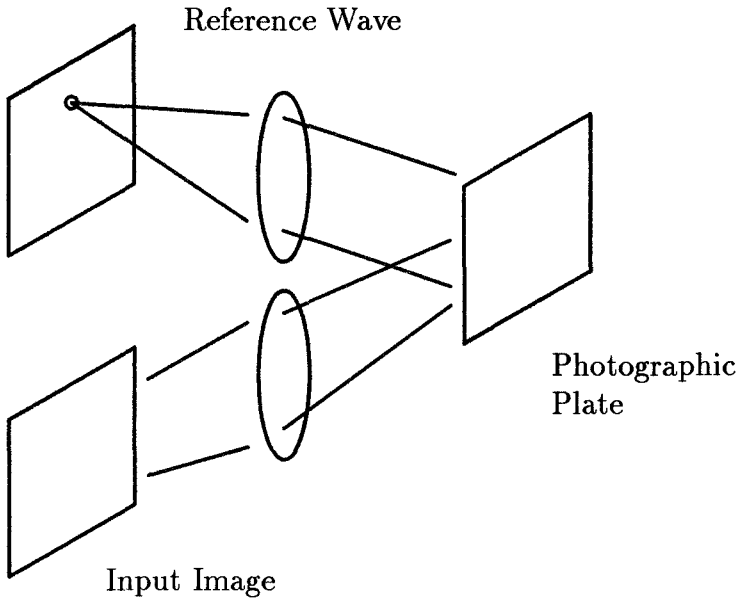
This kind of association is bidirectional and the SNR can be easily derived from the analysis similar to one that was done previously. This is useful for storing multiple associations and page modes in data storage. We can generalize this multiple association to higher orders and introduce nonlinearities by making some of the data appear multiply.

The convergence of the recall process of higher order bidirectional memories will be discussed in the following chapter.

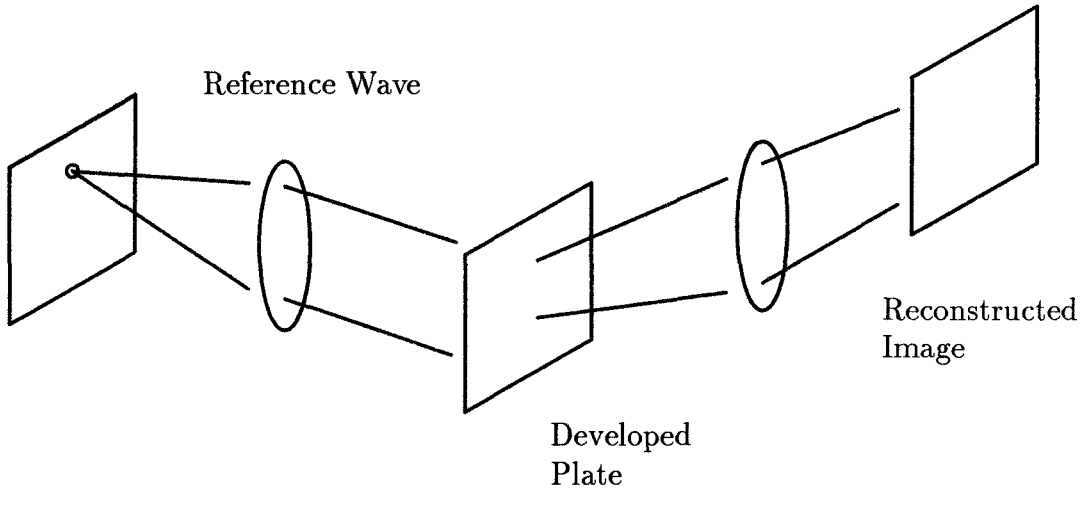
4.5 Optical implementations of higher order associative memories

The outer product higher order associative memories described in the previous section require basic components for their implementation: interconnection weights, r-th power devices, and threshold nonlinearities[36, 37, 38]. In this section, we present a variety of optical implementations using volume holograms to provide the interconnection weights and optical or electro-optical devices to provide the required nonlinearities.

We will first briefly discuss holography[9] and in particular the distinction between the use of planar versus volume holograms. The holographic process is shown schematically in Fig. 4.4. In the recording stage with planar hologram (Fig. 4.4a) the interference pattern between the reference plane wave created by collimating the light from a point source using a lens and the wave originating from the object "A"



a



b

Figure 4.4 Holographic recording and reconstruction. a. Recording; b. Reconstruction.

is recorded on a planar light sensitive medium such as a photographic plate. When the developed plate is illuminated with the same reference wave, the field that is diffracted by the recorded interference pattern gives a virtual image of the original object, which can be converted to a real image with a lens. The reconstruction of the hologram is thus equivalent to interconnecting the single point from which the reference plane wave is derived to all the points that comprise the reconstructed image. The weight of each interconnection is specified by the interference pattern stored in the hologram. Volume holograms are used in the same way except that whereas a planar hologram records the interference pattern as a two-dimensional pattern on a plane, a volume hologram records the interference pattern throughout the volume of a three-dimensional medium. The disparity in the dimensionalities of the two storage methods results in marked differences in the capabilities of the two processes.

One of the differences is explained with the aid of Figs. 4.5a and 4.5b where the reconstructions of both a planar and a volume hologram are shown. Each hologram is prepared to store the two images "A" and "O" by double exposures with each image being associated with a reference plane wave that is incident on the hologram at a different angle. Each reference plane wave is generated by a separate point source and thus the reconstruction of a hologram with the two reference waves is equivalent to interconnecting two input points to all the points on the reconstruction plane. In the case of the planar hologram, however, when either one of the reference waves is incident both images are reconstructed in a way that the corresponding image is overlapped by the shifted image of another one unless the images are multiplexed in some way[29]. This implies that we cannot in this case independently specify how each of the input points is connected to the output. In contrast, because of the interaction of the fields in the third dimension[20] the volume hologram is able to distinguish the differences in the angle of incidence of the reference beam and when

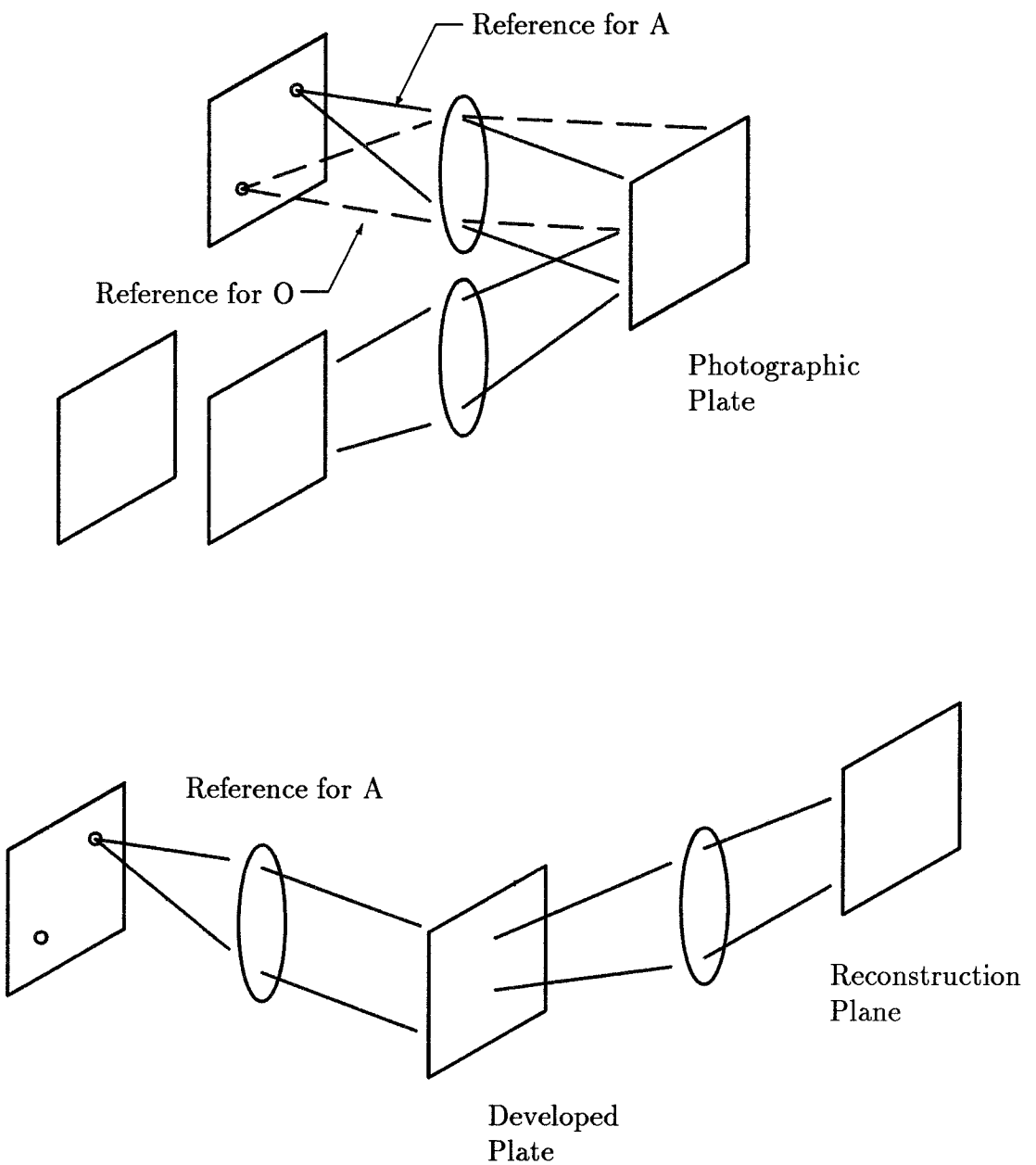
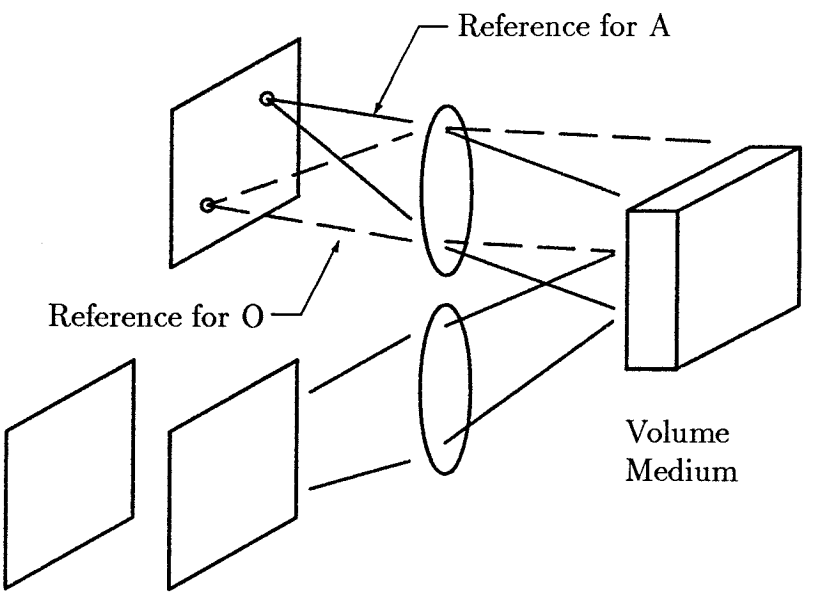
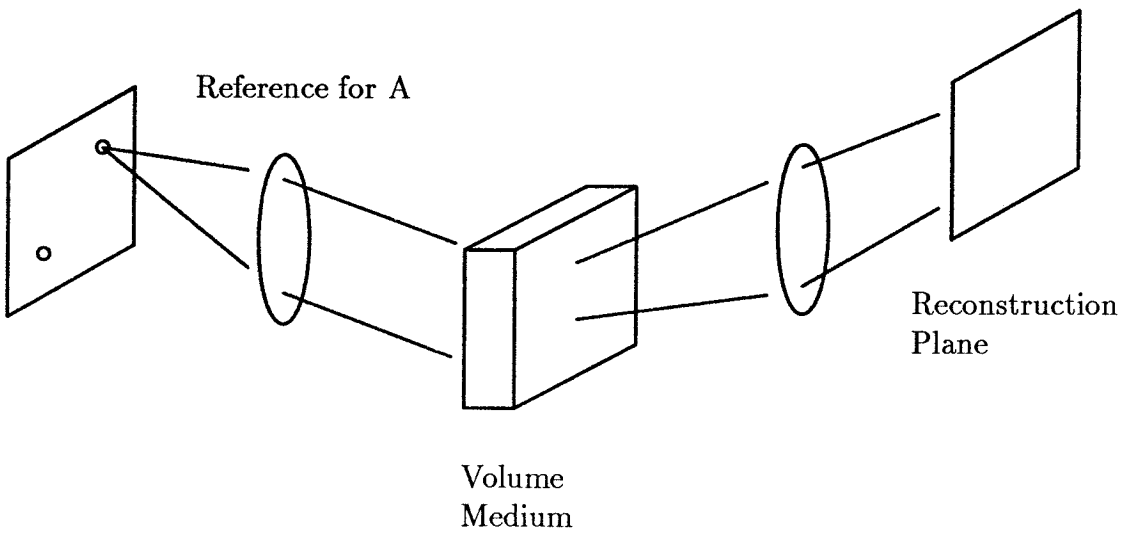


Figure 4.5 Holographic interconnection using a. planar versus b. volume holograms.


Reference for A
Reference for O
Volume
Medium


Reference for A
Reconstruction
Plane
Volume
Medium

b

the reference for "A" illuminates the medium, only "A" is reconstructed and similarly for the second pattern. When both input points are on simultaneously, each is interconnected to the corresponding output independently. It can be generalized for multiple input points. Thus volume holograms provide more flexibility for implementing arbitrary interconnections, which results in efficient three-dimensional storage of the interconnection weights needed to specify the higher order memory.

Another way of drawing the distinction between planar and volume holograms is in terms of the degrees of freedom. As an instance, the implementation of a quadratic memory whose dimensionality of input is N requires approximately N^3 interconnection weights for the third order interconnection tensor. The number of degrees of freedom of the planar hologram of area A is upper bounded by A/δ^2 while that of a volume hologram is limited to V/δ^3, where V is the volume of the crystal and δ is the minimum detail that can be recorded in any one dimension[34, 39, 46]. Equating the number of degrees of freedom required to provide the interconnection weights to those that are available, the crystal volume is determined to be at least $V = N^3\delta^3$ whereas a planar hologram to do the same job would require the area A of $N^3\delta^2$. For comparison, a network with $N = 10^3$ can in principle be implemented using a cubic crystal with the length of each side being $l_v = N\delta = 1$ cm, but a square planar hologram is required to have the length of each side of at least $l_p = N^{3/2}\delta = 0.33$ m at $\delta = 10$ μm. Thus, the volume hologram offers a more compact means of implementing large memory systems.

4.5.1 Quadratic associative memories

There are several schemes for fully utilizing the interconnection capability of volume holograms[39]. For the implementation of quadratic memories we use volume holograms to fully interconnect a 2-D pattern to a 1-D pattern ($N^2 \mapsto N$ mappings) and also the reverse ($N \mapsto N^2$). The geometry for recording the weights for both cases

is shown in Fig. 4.6a and the reconstruction geometries are illustrated in Figs. 4.6b and 4.6c. The circles represent the resolvable spots at the various planes in the system. The waves emanating from input points are transformed into plane waves by the Fourier transform lenses L_1 and L_2 and interfere within the crystal, creating volume gratings.

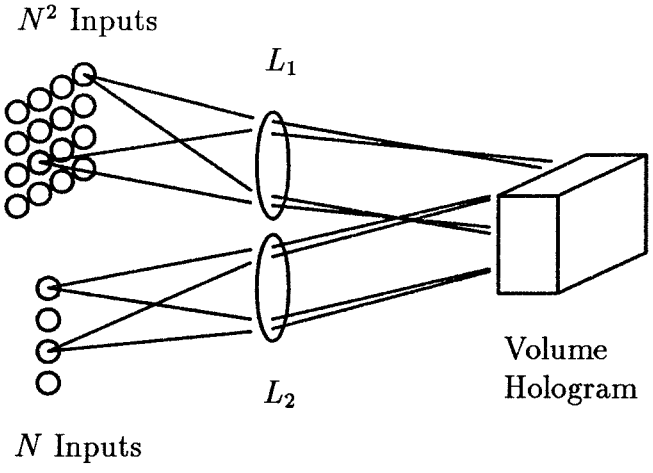
The weights are loaded into the volume hologram with multiple holographic exposures in the system of Fig. 4.6a that has an $N \rightarrow N^2 \rightarrow N^3$ storage operation. For the $N \mapsto N^2$ mapping (Fig. 4.6b) that has an $N^3 \rightarrow N^2 \rightarrow N$ recall operation as will be illustrated later, in reading out the stored information, a single source in the input array reconstructs 2-D images consisting of N^2 pixels that it is associated with. The rest of the images, which are associated with the other input points, are not read out because of the angular separability of volume holograms. The counterpart to this scheme, shown in Fig. 4.6c, implements an arbitrary $N^2 \mapsto N$ mapping. This setup is basically the same as that of Fig. 4.6b except that the roles of the input planes have been interchanged or equivalently the direction in which light propagates has been reversed.

$N^2 \mapsto N$ Schemes First, we consider a method by which the full third order interconnection tensor is implemented directly with a volume hologram. Recall that if the weight tensor is trained using the sum of outer products then it is given by

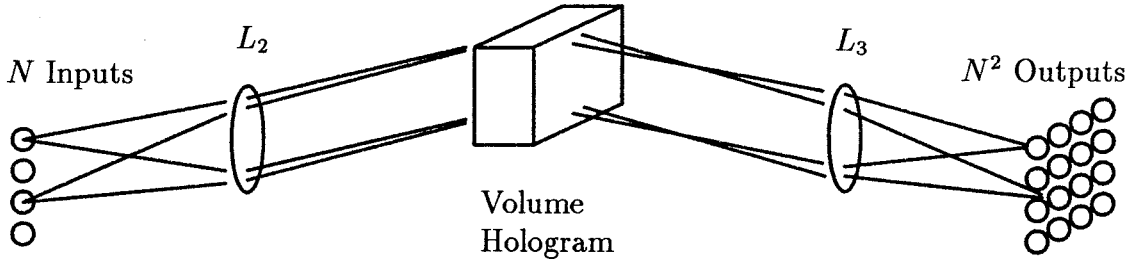
$$w_{ijk} = \sum_{m=1}^{M} y_i^m x_j^m x_k^m, \tag{4.34}$$

where x_j^m is an input vector and y_i^m its associated output vector. Such a memory is accessed by creating an outer product of the input vector and then multiplying it with w_{ijk} as shown in Fig 4.1,

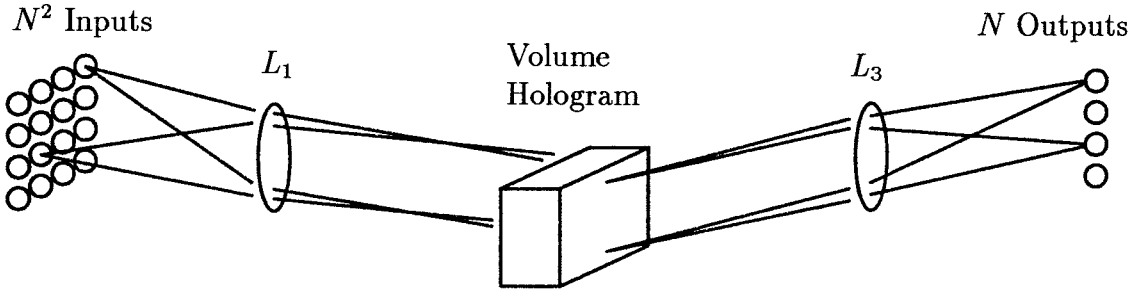
$$y_i = \mathrm{sgn}\Big\{\sum_{j=1}^{N}\sum_{k=1}^{N} w_{ijk} x_j x_k\Big\}. \tag{4.35}$$



a



b



c

Figure 4.6 Optical interconnections using volume holograms. a. Recording apparatus; b. $N \mapsto N^2$ mapping; c. $N^2 \mapsto N$ mapping.

The volume hologram is prepared using the set-up in Fig. 4.6a. First, the outer product matrix of the input vector, $x_j^m x_k^m$, is formed on a two-dimensional spatial light modulator (SLM)[49]. Another one-dimensional SLM whose transmittance represents the output vector y_i^m is placed on the other input plane, and the two SLMs are illuminated by coherent light. The transmitted waves are then Fourier transformed by lenses L1 and L2 to interfere within the crystal volume to create index gratings. This procedure is repeated for all M associated input–output pairs so that a sum of M interference patterns is created in the crystal. For the quadratic associative memory with outer product rule whose capacity is fully used, this involves M_2 exposures as given in Eq. 4.20.

We will now describe another method of recording the weight tensor in the volume hologram that has fewer exposures and can be used not only for the outer product scheme but also for recording any given weight tensor. The same basic recording architecture of Fig. 4.6a is used in this case, too. In the first exposure, the top light source in the linear array is turned on while the 2-D SLM contains the matrix w_{1jk}. When the SLM is illuminated with light coherent with that of the point source, the crystal records the interference pattern between the Fourier transform of the image w_{1jk} and a reference beam that is the plane wave generated from the top light source. In the next step, the second source is turned on while the SLM stores the matrix w_{2jk} and so on. The connectivities for all the points in the linear array at the input are sequentially specified and the memory training is completed when all N exposures have been made. The disadvantage of this method relative to the direct outer product recording scheme is the need to precalculate the interconnection weights but it has the advantage of fewer exposures (N versus M) and greater flexibility in choosing the training method.

The architecture in Fig. 4.6c is used to access the data stored in the hologram by either one of the recording methods described above. The 2-D SLM is placed at

the input plane to contain the outer product matrix $x_j x_k$ of the input vector. The light from the N^2 input points is interconnected with the N output points via the recorded interconnection kernel w_{ijk}. A linear array of N photodetectors read out the output.

It is important to restate at this moment that this particular implementation achieves the quadratic interconnections by first transforming the N input features (i.e., the N components of the input vector $\underline{x}$) into a set of N^2 features via the outer product operation. The result is that although the interconnections are quadratic with respect to the N original features they are linear with respect to the N^2 transformed features. This allows the application of error driven learning algorithms for linear networks such as the Adaline[50] where the interconnections are developed by an iterative training process. The operation of such a learning scheme is illustrated in Fig. 4.7, which shows the same basic architecture as Fig. 4.6c with feedback from the output into the input. Each iteration consists of a reading and a writing stage. During the reading stage, the interconnection weights stored in the crystal are tested with a particular item to be memorized by illuminating the 2-D SLM, which contains the outer product matrix $\underline{x}^m \underline{x}^{mt}$ and the output is formed on the detector array. In the subsequent writing stage, the error pattern generated by subtracting the desired output from the actual output pattern, $\epsilon_i = y_i^m - \sum_{jk} w_{ijk} x_j^m x_k^m$, is loaded into the 1-D SLM and both SLMs (the 2-D SLM still contains $\underline{x}^m \underline{x}^{mt}$) are illuminated with coherent light, forming a set of gratings in addition to the previously recorded gratings. The procedure is iteratively repeated for each item to be memorized until the output error is sufficiently small. This algorithm is a descent procedure designed to minimize the mean squared cost function $E_i = \frac{1}{M} \sum_{m=1}^{M} (\sum_{j=1}^{N} \sum_{k=1}^{N} w_{ijk} x_j^m x_k^m - y_i^m)^2$ by iteratively updating the interconnection values.

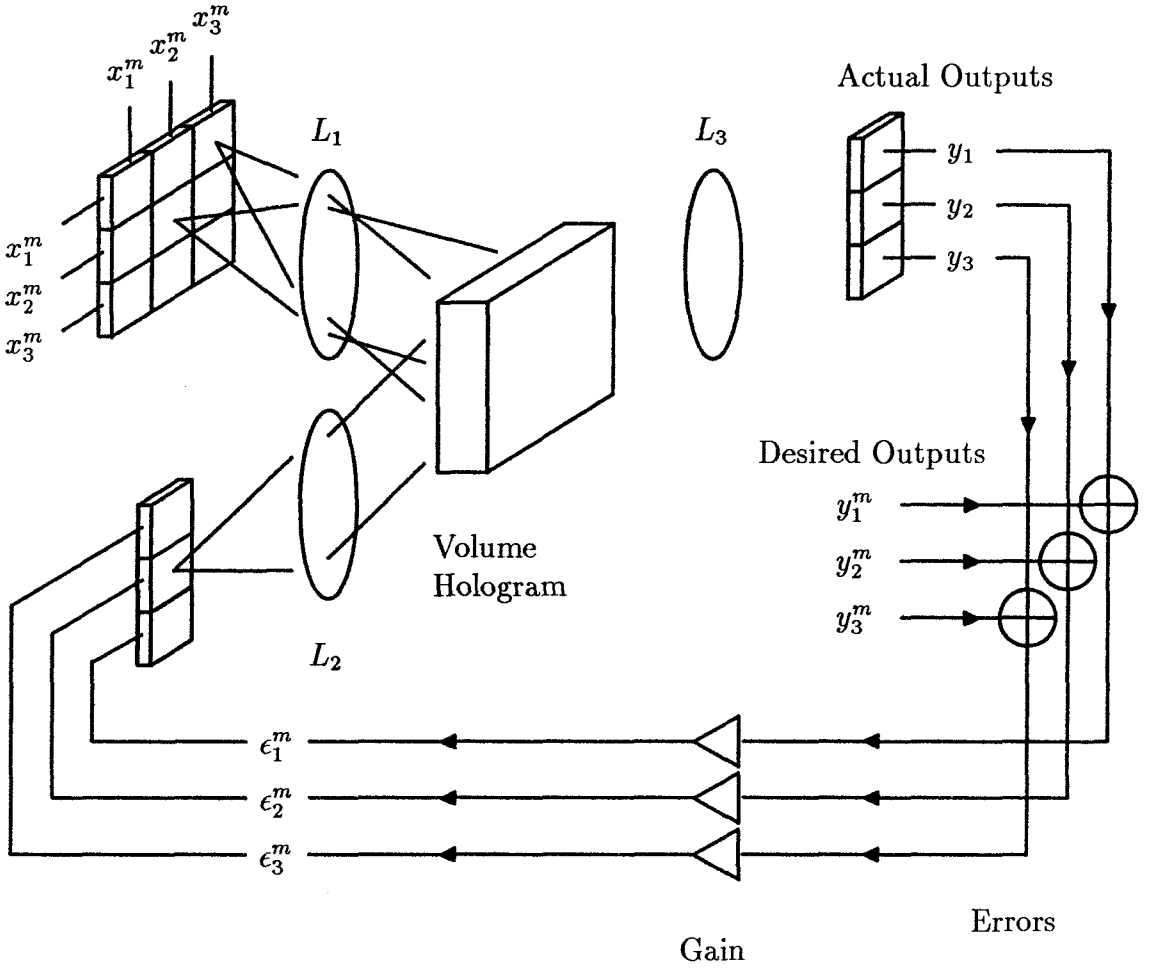


Figure 4.7 Optical system for performing error driven learning in a higher order memory.

$N \mapsto N^2$ Schemes The $N \mapsto N^2$ mapping capability of the volume hologram that is the inverse of that required for the architectures just described can also be used to implement quadratic memories. The basic idea behind this scheme is illustrated in Fig. 4.3, which shows the interconnection between the i-th and the j-th neurons whose weight w_{ij} is decided by the states of the other neurons as well as the states of the i-th and the j-th neurons themselves as a linear combination of all of them,

$$w_{ij} = \sum_{k=1}^{N} w_{ijk} x_k. \tag{4.36}$$

The overall result is exactly the same as above except for the order of operations,

$$y_i = \Big\{ \sum_j (\sum_k w_{ijk} x_k) x_j \Big\}. \tag{4.37}$$

As implied by the equation, it consists of two stages; interconnection of an input vector $\underline{x}$ to a matrix w_{ij} by the weight w_{ijk}, and vector matrix multiplication[14] of the input vector $\underline{x}$ and the weight matrix w_{ij} to form an output vector $\underline{y}$ whose optical implementation is shown in Fig. 4.8. The input vector on the left is the transpose of the input vector on the right placed in a one-dimensional SLM. The portion of the system on the left side of the SLM is the vector matrix multiplier and it works as follows: Light from each input point is imaged horizontally but spread out vertically so that each source illuminates a narrow, vertical area on the 2-D SLM whose reflectance corresponds to the weight matrix w_{ij} in Eq. 4.36. The reflected light from the SLM travels back towards the input and a portion of it is reflected by a beam splitter and imaged horizontally but focused vertically onto a 1-D detector array. The output that represents the vector matrix product between the input vector and the matrix represented by the 2-D reflectance of the SLM can be fedback as an input.

The weight matrix, in this case, is not fixed but rather computed from the input via a volume hologram by exposing the right-hand side of the SLM as shown in the

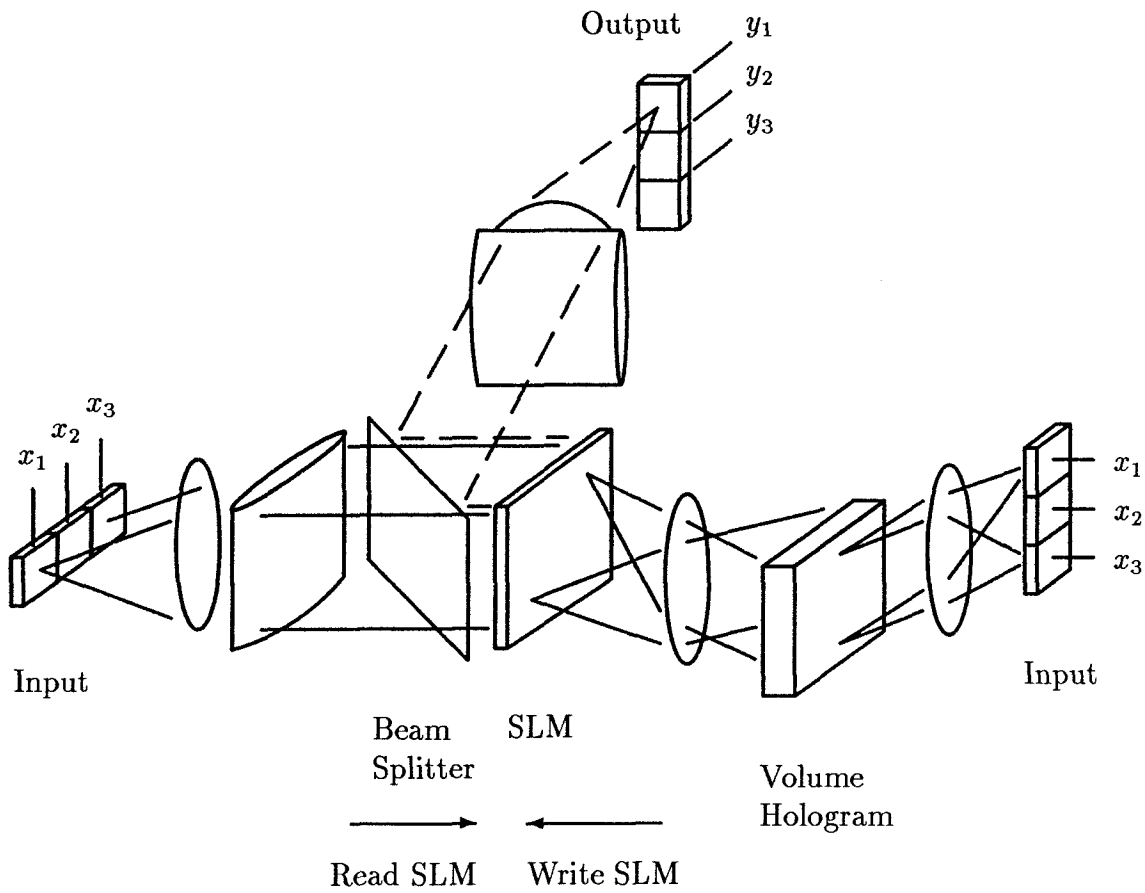


Figure 4.8 Optical architecture for the implementation of the nonlinear interconnection.

figure. The optical system to the right of the 2-D SLM in Fig. 4.8 is the same as the $N \mapsto N^2$ system of Fig. 4.6b. The volume hologram that has been prepared to perform the appropriate dimension increasing operation ($N \mapsto N^2$) transforms the light distribution of its one-dimensional SLM into the input dependent matrix of weights given by Eq. 4.36. This system is functionally equivalent to the previous system except it does not require the use of a 2-D input SLM. The 1-D devices utilized in this architecture are easier and faster to use in practice.

4.5.2 Cubic associative memories

For cubic memories, the fourth order tensor w_{ijkl} is required to provide the interconnection weights,

$$y_i = \mathrm{sgn}\Big\{\sum_{jkl} w_{ijkl} x_j x_k x_l\Big\} \tag{4.38}$$

where

$$w_{ijkl} = \sum_{m=1}^{M} y_i^m x_j^m x_k^m x_l^m \tag{4.39}$$

where x_j^m and y_i^m are associated input–output pairs as before. Alternatively, this equation can be rearranged to give the idea of implementation,

$$\begin{aligned} y_i &= \mathrm{sgn}\Big\{\sum_j (\sum_{kl} w_{ijkl} x_k x_l) x_j\Big\} \\ &= \mathrm{sgn}\Big\{\sum_j w_{ij} x_j\Big\}, \end{aligned} \tag{4.40}$$

where the interconnection weight w_{ij} between the i-th and the j-th neurons is decided as a sum of the interactions of two neurons,

$$w_{ij} = \sum_{kl} w_{ijkl} x_k x_l. \tag{4.41}$$

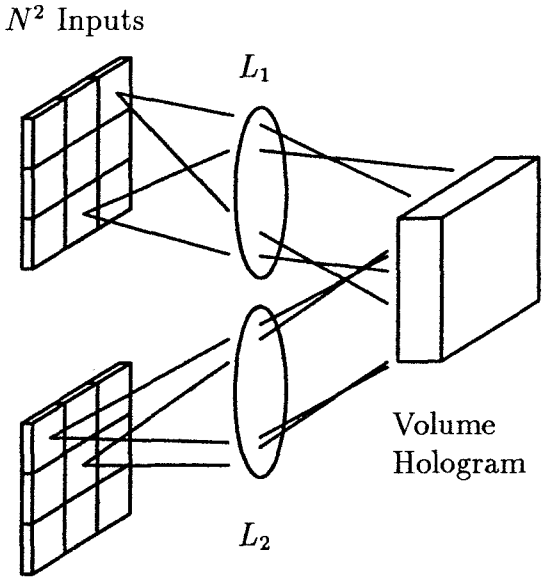
According to Eqs. 4.40 and 4.41 there are two operations; interconnection of the outer product matrix $x_k x_l$ of an input vector and a weight matrix w_{ij} by the weight tensor w_{ijkl}, and vector matrix multiplication of the input vector and this

weight matrix to form an output. It is analogous to the implementation of quadratic memories using $N \mapsto N^2$ mapping except that it needs the $N^{3/2} \mapsto N^{3/2}$ mapping instead to match the available degrees of freedom of the volume hologram[39].

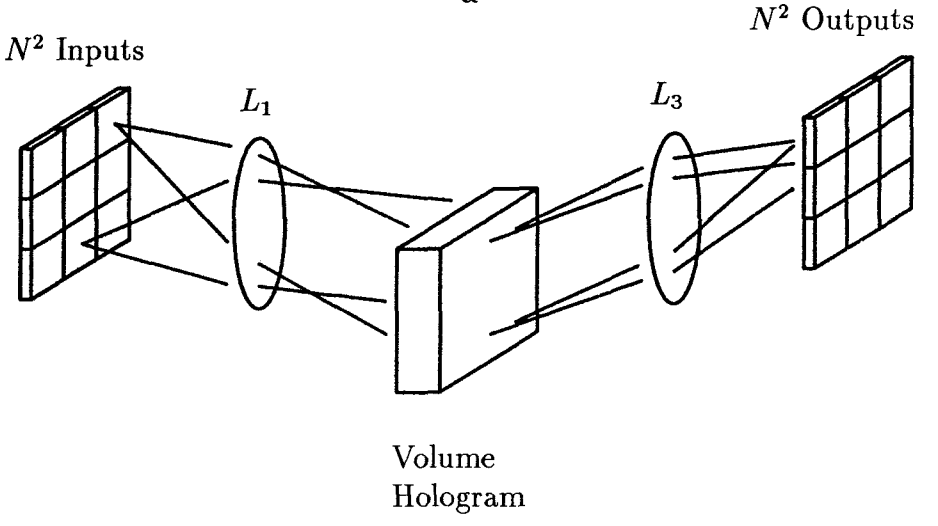
In the recording stage, two outer product matrices $x_k^m x_l^m$ and $y_i^m x_j^m$ are formed on two 2-D input SLMs as shown in Fig. 4.9. The two outputs of the SLMs that are illuminated with coherent light are Fourier transformed by lenses L1 and L2 to make an interference pattern in the volume hologram, which is equivalent to a dimension increasing operation $N \rightarrow N^2 \rightarrow N^4$. This procedure is repeated for all M pairs of associated inputs and outputs. The reconstruction process shown in Fig. 4.10 is basically the same as one in Fig. 4.8 except that it makes use of an $N^{3/2} \mapsto N^{3/2}$ mapping method with volume hologram instead of an $N \mapsto N^2$ mapping method for quadratic associative memories and that therefore it needs a 2-D input SLM. The reconstruction is a dimension decreasing operation $N^4 \rightarrow N^2 \rightarrow N$.

4.5.3 Power law implementation

The $N \mapsto N^2$ mapping technique can be used in conjunction with its inverse, the $N^2 \mapsto N$ mapping, to implement the higher order associative memory with outer product rule using two volume holograms, a 1-D SLM, and a 2-D SLM. Shown in Fig. 4.11 is a schematic diagram of such a system. The first hologram is prepared with the multiple exposure scheme discussed earlier (Fig. 4.6a) where for each exposure, a memory vector $\underline{x}^m$ in the one-dimensional input array and one point in the two-dimensional $(\sqrt{M} \times \sqrt{M})$ input training array are turned on simultaneously. The second hologram is prepared by a similar procedure except that the associated output vectors are recorded in correspondence to each point in the two-dimensional training plane. After the holograms are thus trained, an input vector is loaded into the one-dimensional input array and the correlations between it and the M stored vectors are displayed in the middle output plane[5, 27, 29, 36, 37, 38]. A 2-D SLM



a



b

Figure 4.9 Optical interconnection for cubic associative memories. a. Recording; b. Reconstruction.

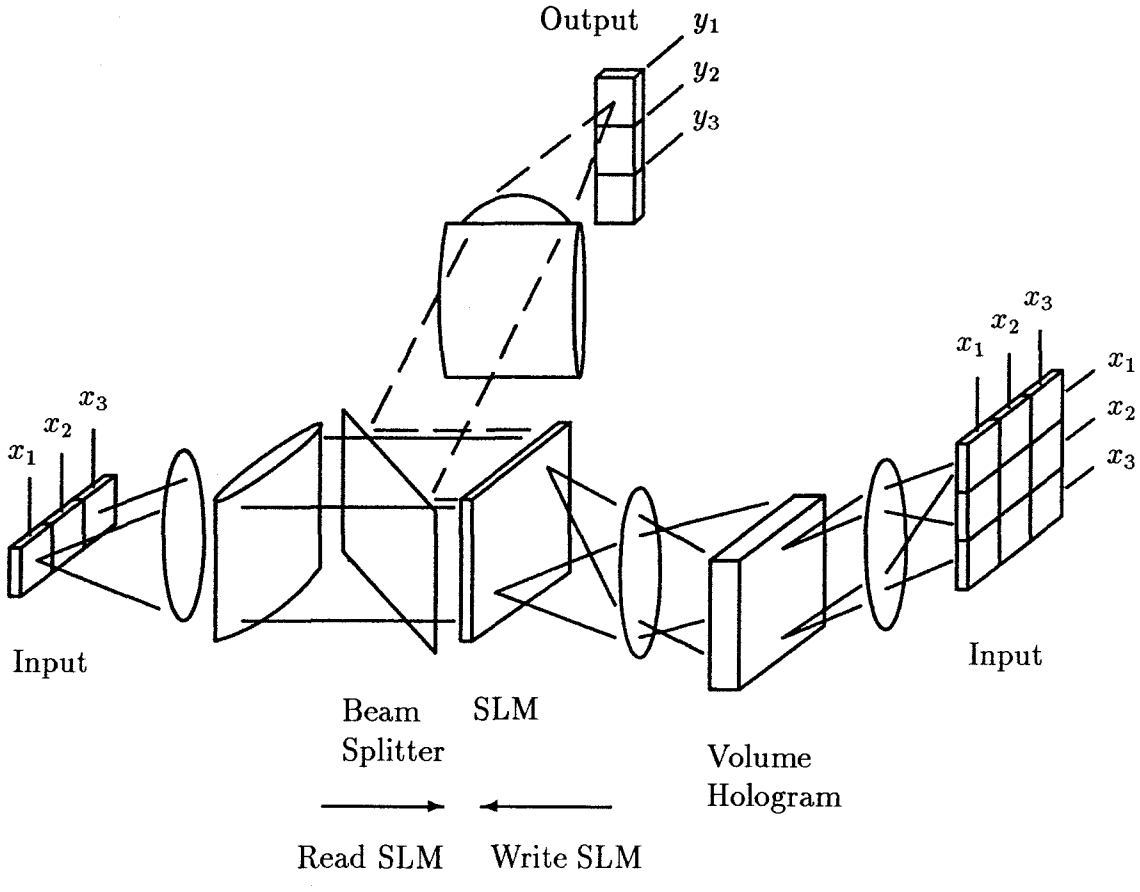


Figure 4.10 Optical architecture for cubic associative memories.

can be used to produce an amplitude distribution that is the square of the incident correlation amplitudes for the quadratic memories or the r-th power for the r-th order memory in general. The processed light then illuminates the second hologram, which serves as an $M \mapsto N$ interconnection, each correlation peak in the middle SLM plane reading out its corresponding output vector and forming a weighted sum of the stored data on the one-dimensional output detector array. This is the alternative optical implementation of the system shown in Fig. 4.2, according to Eq. 4.5 with the 2-D SLM performing the r-th power law nonlinearity at the middle plane and the two volume holograms providing the interconnections between the input and output. If the 2-D SLM is removed from the correlation plane this system reduces to the Hopfield–type linear associative memory.

The implementations of bidirectional memories are basically the same as those that were described already except that the input and the output are bidirectional. The system in Fig. 4.11 can be used to implement r-th order bidirectional associative memories described by Eqs. 4.28 and 4.29 assuming that light goes in both directions.

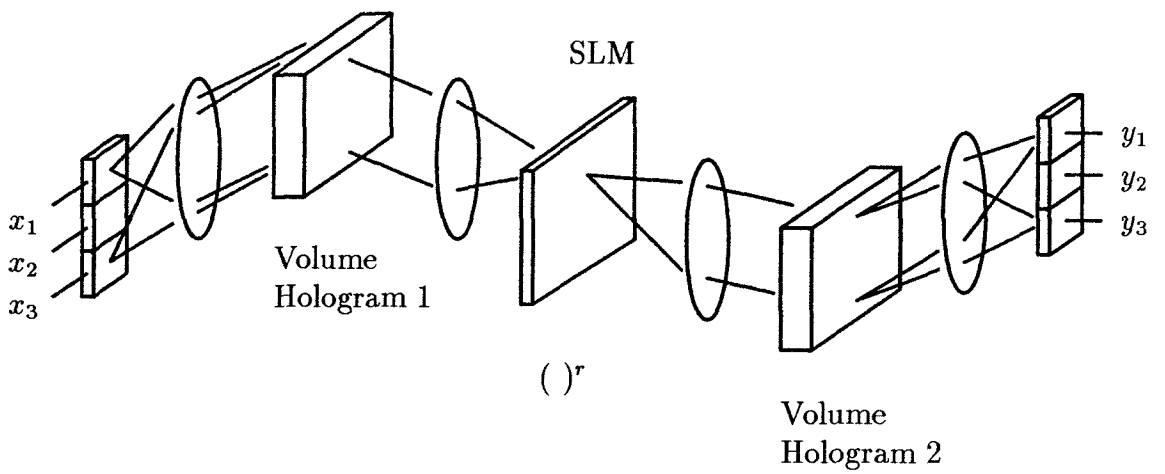


Figure 4.11 Optical higher order associative memory implemented with volume holograms.

Chapter 5

Robustness of Higher Order Memories

The performance of any practical implementation of the higher order memories, discussed in the previous chapter, will be degraded by the unavoidable presence of internal or external noise, which will affect the capacity and dynamic range of the system. The "robustness" of associative memories can be considered not only robustness for errors in the input (error tolerance)[25, 47] but also robustness for noise inherent in the memory components (noise sensitivity)[16, 44]. It can be explained in a way that the capacity and robustness of the memory are complementary by introducing redundancies given the number of degrees of freedom[38] as will be shown in SNRs we are going to derive as a measure of performance of the network. There are several types of noise that can be considered and they depend on the learning algorithm and the architecture of the systems that are to be implemented. When a simple sum of outer product learning rule is employed, there are two kind of noise: one is cross talk noise among stored data that does not depend on the system noise; the other is system noise that depends on architecture. For example, if the input is buried in noise its effect may be reduced by threshold operation on each input component before being applied to the network, called pre-thresholding, and system noise limits the dynamic range of the network weights and other components.

In the case of autoassociation and bidirectional associations, an energy function can be used to describe the convergence of the memory recall process and error correcting properties of associative memories. We will describe the noise sensitivity of higher order memories in general, and when they are trained with the sum of outer product learning rule we will derive their characteristics on several types of noise that depends on the architecture, including the effect of dynamic range of the

memory. Without loss of generality the distribution of noise will be assumed to be normal according to the central limit theorem[30] as in the previous chapter.

Dynamic range of the system is bounded by characteristics of devices that are used, which puts a limitation on accuracies of computation.

5.1 Error tolerance of higher order memories

In an autoassociative or bidirectional associative memory, robustness against input errors as well as convergence property of the recall process can be explained by using an energy function of the network.

5.1.1 Autoassociative memories

Chen and his co-workers[7] introduced an energy function[8, 17, 18] for the r-th order autoassociative memory with feedback and outer products as follows:

$$\mathrm{E}_r = -\sum_{m=1}^{M} < \underline{x}^m, \underline{x} >^{r+1} \tag{5.1}$$

where $< \cdot, \cdot >$ denotes an inner product of two vectors. The change in the energy due to a change $\delta\underline{x}$ in the state of the network was shown to be monotonically decreasing for odd r:

$$\begin{aligned} \Delta \mathrm{E}_r &\equiv \mathrm{E}_r(\underline{x} + \delta\underline{x}) - \mathrm{E}_r(\underline{x}) \\ &= -(r+1)\sum_l \delta x_l \sum_{j_1 \cdots j_r} w_{l j_1 j_2 \cdots j_r} x_{j_1} x_{j_2} \cdots x_{j_r} - R_r \end{aligned} \tag{5.2}$$

where

$$R_r \equiv \sum_m \sum_{j=2}^{r+1} \binom{r+1}{j} < \underline{x}^m, \underline{x} >^{r+1-j} < \underline{x}^m, \delta\underline{x} >^j . \tag{5.3}$$

The first term in Eq. 5.2 is always nonpositive because of the specification of the update rule: $\delta x_l \geq 0$ if $\sum_{j_1 \cdots j_r} w_{l j_1 j_2 \cdots j_r} x_{j_1} x_{j_2} \cdots x_{j_r} \geq 0$ and vice versa. Chen et al.[7] showed that the second term is also nonpositive by showing that R_r is an increasing function of r for r odd and $R_1 \geq 0$.

For an even order autoassociative memory, convergence can be proved only if it is updated in an asynchronous way, even though in simulations in the following section quadratic autoassociative memories consistently converge as well when using synchronous updating[36]. The fact that the energy is not always decreasing when r is even may actually be helpful for getting out of spurious local minima and settling in one of the programmed stable states, which are global minima in a region of the energy surface. A descent procedure for which the energy is always decreasing cannot escape from local minima since there is no mechanism for climbing out of them. As an example, consider a quadratic memory, i.e., r=2 (even), whose energy function is given by

$$\mathrm{E}_2 = -\sum_{ijk} w_{ijk} x_i x_j x_k. \tag{5.4}$$

And therefore the change in the energy becomes

$$\begin{aligned} \Delta \mathrm{E}_2 &= -3\sum_{ijk} w_{ijk} x_j x_k \delta x_i - 3\sum_{ijk} w_{ijk} x_k \delta x_i \delta x_j - \sum_{ijk} w_{ijk} \delta x_i \delta x_j \delta x_k \\ &= -3\sum_{i} \delta x_i \Big(\sum_{m} x_i^m < \underline{x}^m, \underline{x} >^2 \Big) - 3\sum_{m} < \underline{x}^m, \underline{x} >< \underline{x}^m, \delta \underline{x} >^2 \\ &\quad -\sum_{m} < \underline{x}^m, \delta \underline{x} >^3 . \end{aligned} \tag{5.5}$$

The first term is nonincreasing by the updating rule, but the second and third terms can be increasing. If the state vector $\underline{x}$ is close to one of the stored vectors $\underline{x}^n$, then the first term becomes dominant, and thus the energy will be nonincreasing, causing the system to settle at $\underline{x} = \underline{x}^n$. If $\underline{x}$ is not close to any of the stored vectors, then all three terms in the above equations are on the average comparable to each other. Since two of them are not nondecreasing, the energy function may increase and it may be possible to escape from local minima.

The following is the analysis for the above argument by calculating the SNR between the decreasing signal term and the other terms for one step convergence[25]:

Let the input be one of the stored vectors with random errors, $\delta\underline{x}$, such that

$$< \underline{x}^n, \underline{x} >=< \underline{x}^n, \underline{x}^n - \delta\underline{x} >= (1-2\rho)N \tag{5.6}$$

where $< \underline{x}^m, \delta\underline{x} >= 2\rho N\delta_{mn}$ and $0 \leq \rho < 0.5$ where δ_{mn} is one only for $m = n$ and zero. Then, Eq. 5.5 can be written

$$\begin{aligned}\Delta E_2 &= -3(2\rho N)(1-2\rho)^2N^2 - 3\sum_i \delta x_i\Big(\sum_{m\neq n} x_i^m < \underline{x}^m, \underline{x} >^2\Big) \\ &\quad -3(2\rho N)^2(1-2\rho)N - 3\sum_{m\neq n} < \underline{x}^m, \underline{x} >< \underline{x}^m, \delta\underline{x} >^2 \\ &\quad -(2\rho N)^3 - \sum_{m\neq n} < \underline{x}^m, \delta\underline{x} >^3 .\end{aligned} \tag{5.7}$$

The expectation value of ΔE_2 becomes

$$E\{\Delta E_2\} = -\{3(2\rho N)(1-2\rho)^2N^2 + 3(2\rho N)^2(1-2\rho)N + (2\rho N)^3\} \tag{5.8}$$

assuming that $\underline{x}^m$ and $\delta\underline{x}$ are uncorrelated for $m \neq n$, that is, $E\{< \underline{x}^m, \delta\underline{x} >\} = E\{< \underline{x}^m, \underline{x} >\} = 0$ for $m \neq n$. The noise term is

$$\begin{aligned}n(\underline{x}) &= -3\sum_i \delta x_i\Big(\sum_{m\neq n} x_i^m < \underline{x}^m, \underline{x} >^2\Big) - 3\sum_{m\neq n} < \underline{x}^m, \underline{x} >< \underline{x}^m, \delta\underline{x} >^2 \\ &\quad - \sum_{m\neq n} < \underline{x}^m, \delta\underline{x} >^3 .\end{aligned} \tag{5.9}$$

Then $n(\underline{x})^2$ has three squared terms and one cross term having nonvanishing expectation value. We will calculate $E\{n^2\}$ one by one with the help of Eqs. 4.9, 4.11, 4.10, 4.12, and 4.13 as follows:

$$\begin{aligned}&E\Big\{\Big(\sum_i \delta x_i \sum_{m\neq n} x_i^m < \underline{x}^m, \underline{x} >^2\Big)^2\Big\} \\ &= E\Big\{\sum_{i_1 i_2} \delta x_{i_1}\delta x_{i_2} \sum_{mm'\neq n} x_{i_1}^m x_{i_2}^{m'} \sum_{j_1 j_2 j_3 j_4} x_{j_1}^m x_{j_2}^m x_{j_3}^{m'} x_{j_4}^{m'} x_{j_1} x_{j_2} x_{j_3} x_{j_4}\Big\} \\ &= 2^2\rho N E\Big\{\sum_{m\neq n}\sum_{j_1 j_2 j_3 j_4} x_{j1}^m x_{j2}^m x_{j3}^m x_{j4}^m x_{j_1} x_{j_2} x_{j_3} x_{j_4}\Big\} \\ &= 2^2\rho N(M-1)(3N^2-2N),\end{aligned} \tag{5.10}$$

$$
\begin{aligned}
&\mathrm{E}\Big\{\Big(\sum_{m\neq n} <\underline{x}^m,\underline{x}><\underline{x}^m,\delta\underline{x}>^2\Big)^2\Big\} \\
&= \mathrm{E}\Big\{\sum_{mm'\neq n}\sum_{i_1 i_2} x_{i_1}^m x_{i_2}^{m'} x_{i_1} x_{i_2} \sum_{j_1 j_2 j_3 j_4} x_{j_1}^m x_{j_2}^m x_{j_3}^{m'} x_{j_4}^{m'} \delta x_{j_1}\delta x_{j_2}\delta x_{j_3}\delta x_{j_4}\Big\} \\
&= N\mathrm{E}\Big\{\sum_{m\neq n}\sum_{j_1 j_2 j_3 j_4} x_{j_1}^m x_{j_2}^m x_{j_3}^m x_{j_4}^m \delta x_{j_1}\delta x_{j_2}\delta x_{j_3}\delta x_{j_4}\Big\} \\
&= N(M-1)2^4\{3(\rho N)^2 - 2(\rho N)\}, \qquad (5.11)
\end{aligned}
$$

$$
\begin{aligned}
&\mathrm{E}\Big\{\Big(\sum_{m\neq n} <\underline{x}^m,\delta\underline{x}>^3\Big)^2\Big\} \\
&= \mathrm{E}\Big\{\sum_{mm'\neq n}\sum_{j_1 j_2 j_3 j_4 j_5 j_6} x_{j_1}^m x_{j_2}^m x_{j_3}^m x_{j_4}^{m'} x_{j_5}^{m'} x_{j_6}^{m'} \delta x_{j_1}\delta x_{j_2}\delta x_{j_3}\delta x_{j_4}\delta x_{j_5}\delta x_{j_6}\Big\} \\
&= (M-1)2^6\{15(\rho N)^3 - 30(\rho N)^2 + 16\rho N\}, \qquad (5.12)
\end{aligned}
$$

$$
\begin{aligned}
&\mathrm{E}\Big\{\sum_i \delta x_i\Big(\sum_{m\neq n} x_i^m <\underline{x}^m,\underline{x}>^2\Big)\sum_{m'\neq n} <\underline{x}^{m'},\delta\underline{x}>^3\Big\} \\
&= \mathrm{E}\Big\{\sum_{mm'\neq n}\sum_{i_1 i_2} x_{i_1}^m x_{i_2}^m x_{i_1} x_{i_2} \sum_{j_1 j_2 j_3 j_4} x_{j_1}^m x_{j_2}^{m'} x_{j_3}^{m'} x_{j_4}^{m'} \delta x_{j_1}\delta x_{j_2}\delta x_{j_3}\delta x_{j_4}\Big\} \\
&= N(M-1)2^4\{3(\rho N)^2 - 2(\rho N)\}. \qquad (5.13)
\end{aligned}
$$

Therefore, the signal-to-noise ratio is given by

$$
\begin{aligned}
\mathrm{SNR} &\approx \frac{(2\rho N)^{1/2}\{3(1-2\rho)^2N^2 + 3\cdot 2\rho N(1-2\rho)N + (2\rho N)^2\}}{\Big\{M\{18(3N^2-2N) + 15N2^3(3\rho N-2) + 2^5\{15(\rho N)^2 - 30\rho N + 16\}\}\Big\}^{1/2}} \\
&= \frac{(2\rho)^{1/2}(3-6\rho+4\rho^2)N^{5/2}}{\Big\{M\{(54+360\rho+480\rho^2)N^2 - (276+960\rho)N + 512\}\Big\}^{1/2}}. \qquad (5.14)
\end{aligned}
$$

As an example, consider the case of $\rho = 0.1$. Then, for $N \approx 10^3$,

$$
\mathrm{SNR} \approx \frac{1.09N^{5/2}}{\Big\{M(94.8N^2 - 372N + 512)\Big\}^{1/2}} \approx 0.11\Big(\frac{N^3}{M}\Big)^{\frac{1}{2}} \qquad (5.15)
$$

while, for $\rho \ll 1$ and $\rho = .5$, $\mathrm{SNR} \approx \sqrt{\frac{\rho}{3}}\Big(\frac{N^2}{M}\Big)^{1/2}$ and $0.05\Big(\frac{N^2}{M}\Big)^{1/2}$, respectively.

For a comparison, consider a linear memory,

$$
\begin{aligned}
\Delta \mathrm{E}_1 &= -2\sum_{ij} w_{ij} x_i \delta x_j - \sum_{ij} w_{ij}\delta x_i \delta x_j \\
&= -2(2\rho N)(1-2\rho)N - 2\sum_{m\neq n} <\underline{x}^m,\underline{x}><\underline{x}^m,\delta\underline{x}> \\
&\quad -(2\rho N)^2 - \sum_{m\neq n} <\underline{x}^m,\delta\underline{x}>^2 . \qquad (5.16)
\end{aligned}
$$

Then, the SNR is easily derived as

$$\text{SNR} \approx \frac{(\rho N)^{1/2}(1-\rho)N}{\left\{M\{(1+3\rho)N-2\}\right\}^{1/2}}. \tag{5.17}$$

In Fig. 5.1 are two SNRs plotted as functions of ρ when both memories store full capacity, $M_2 = M_1 N/3$, and SNRs are normalized by $\sqrt{N^2/M_1}$.

For several step convergence, in general, we assume the following

$$\begin{aligned} < \underline{x}^m, \underline{x} > &= (1-2\rho)N\delta_{mn} \\ < \underline{x}^m, \delta\underline{x} > &= 2dN\delta_{mn} \end{aligned} \tag{5.18}$$

where ρ is the ratio of errors compared with the dimensionality of data, N, δ_{mn} is one only for $m = n$, otherwise zero, and $< \delta\underline{x}, \delta\underline{x} >= 2^2\delta N$ where δN is the number of flipped bits at each iteration and so $0 \leq \delta \leq 1$. We expect d to be positive and close to δ. Then

$$\text{E}\{\Delta \text{E}_2\} = -\{3(2dN)(1-2\rho)^2N^2 + 3(2dN)^2(1-2\rho)N + (2dN)^3\} \tag{5.19}$$

and

$$\begin{aligned} \text{E}\{n^2\} &= 9\cdot 2^2\delta N(M-1)(3N^2-2N) + 9N(M-1)2^4\{3(\delta N)^2 - 2(\delta N)\} \\ &+ (M-1)2^6\{15(\delta N)^3 - 30(\delta N)^2 + 16\delta N\} \\ &+ 6N(M-1)2^4\{3(\delta N)^2 - 2(\delta N)\}. \end{aligned} \tag{5.20}$$

Therefore, the SNR is given by

$$\begin{aligned} \text{SNR} &\approx \frac{(2dN)^{1/2}\{3(1-2\rho)^2N^2 + 3\cdot 2dN(1-2\rho)N + (2dN)^2\}}{\left\{M\{18(3N^2-2N) + 15N2^3(3\delta N-2) + 2^5\{15(\delta N)^2 - 30\delta N + 16\}\}\right\}^{1/2}} \\ &= \frac{(2d)^{1/2}\{3(1-2\rho)^2 + 2d(1-2\rho) + 4d^2\}N^{5/2}}{\left\{M\{(54+360\delta+480\delta^2)N^2 - (276+960\delta)N + 512\}\right\}^{1/2}}. \end{aligned} \tag{5.21}$$

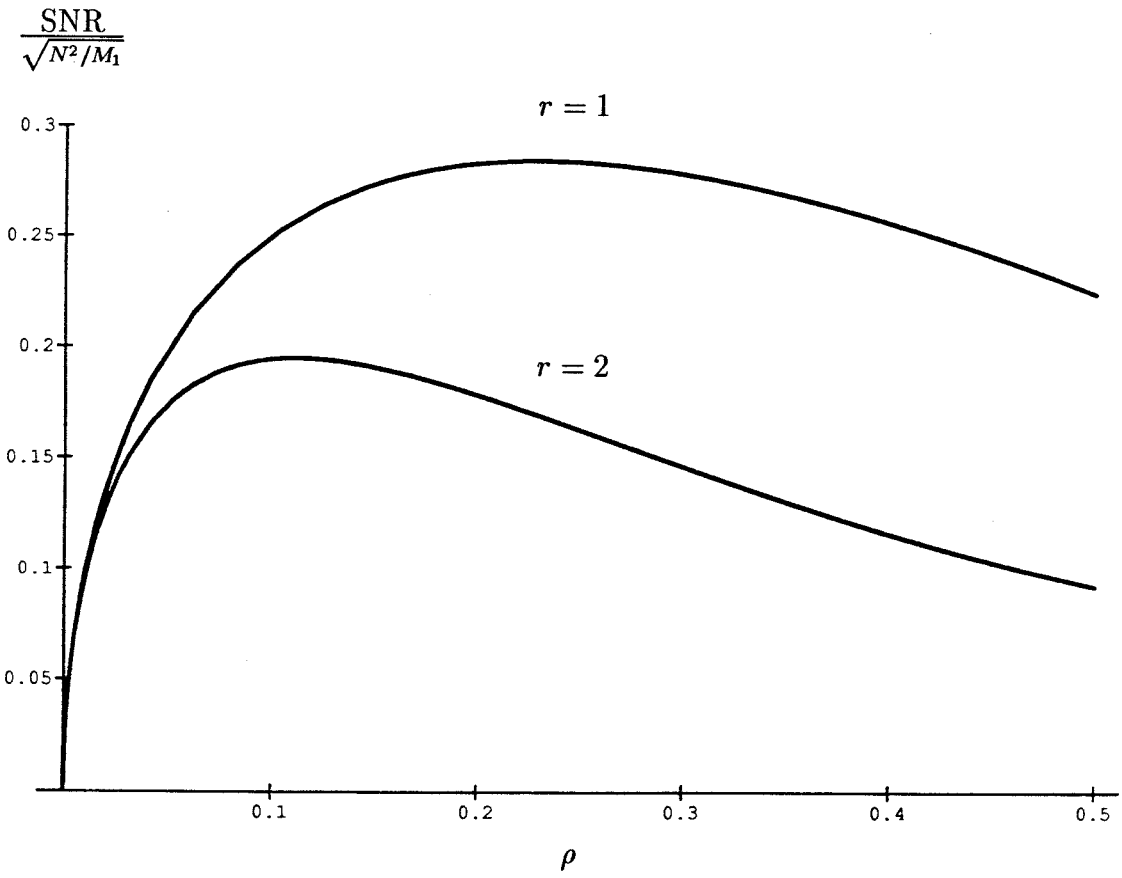


Figure 5.1 Normalized SNRs of energy functions for quadratic and linear memories when full capacities are expended.

5.1.2 Bidirectional associative memories

We will define the energy function of this case according to one in reference [22],

$$\mathrm{E}_r = -\sum_{m=1}^{M} < \underline{y}^m, \underline{y} >< \underline{x}^m, \underline{x} >^r - \sum_{m=1}^{M} < \underline{x}^m, \underline{x} >< \underline{y}^m, \underline{y} >^r . \tag{5.22}$$

Then, changes in the energy due to changes $\delta\underline{x}$ and $\delta\underline{y}$ in the states of the network are given by,

$$\Delta \mathrm{E}_r^x \equiv \mathrm{E}_r(\underline{x} + \delta\underline{x}) - \mathrm{E}_r(\underline{x}), \quad \Delta \mathrm{E}_r^y \equiv \mathrm{E}_r(\underline{y} + \delta\underline{y}) - \mathrm{E}_r(\underline{y}). \tag{5.23}$$

It is enough to see the behavior of one of them and the argument for the convergence of higher order bidirectional memories will be similar to the previous one. The idea is described briefly.

Let us see a quadratic case as a simple one,

$$x_i = \mathrm{sgn}\Big\{\sum_{m=1}^{M} x_i^m (\sum_{j=1}^{N_0} y_j^m y_j)^2\Big\}, \tag{5.24}$$

$$y_j = \mathrm{sgn}\Big\{\sum_{m=1}^{M} y_j^m (\sum_{i=1}^{N} x_i^m x_i)^2\Big\}. \tag{5.25}$$

Then, energy E_2 and its change $\Delta \mathrm{E}_2^x$ are given by,

$$\mathrm{E}_2 = -\sum_{m=1}^{M} < \underline{y}^m, \underline{y} >< \underline{x}^m, \underline{x} >^2 - \sum_{m=1}^{M} < \underline{x}^m, \underline{x} >< \underline{y}^m, \underline{y} >^2, \tag{5.26}$$

$$\Delta \mathrm{E}_2^x = -\Big(\sum_m \underline{x}^m < \underline{y}^m, \underline{y} >^2\Big)\cdot\delta\underline{x} - 2\Big(\sum_m < \underline{y}^m, \underline{y} >< \underline{x}^m, \underline{x} > \underline{x}^m\Big)\cdot\delta\underline{x} \tag{5.27}$$

$$-\sum_m < \underline{y}^m, \underline{y} >< \underline{x}^m, \delta\underline{x} >^2 . \tag{5.28}$$

The first term is nonincreasing because of the updating rule. The second term can be thought to come from multiple associations discussed in the previous chapter, and the updated value of x_i is given by $\mathrm{sgn}\{\sum_{jk} w_{ijk} x_j y_k\}$ where $w_{ijk} = \sum_m x_i^m x_j^m y_k^m$. Therefore, it will be nonincreasing. And the third term can be increasing. Around the state close to stored associations the first and the second terms are dominant and energy function will decrease. Just like the previous argument it may help

the state to reach global minima by providing the mechanism to escape from local minima that the energy function may increase at the states far from any of stored data and decrease around the states near stored data.

This can be generalized to higher orders with the same argument,

$$\begin{aligned}\Delta \mathrm{E}_r^x &= \mathrm{E}_r(\underline{x}+\delta\underline{x}) - \mathrm{E}_r(\underline{x}) \\ &= -\Big(\sum_m \underline{x}^m < \underline{y}^m, \underline{y} >^r\Big)\cdot\delta\underline{x} - r\Big(\sum_m < \underline{y}^m, \underline{y} >< \underline{x}^m, \underline{x} >^{r-1} \underline{x}^m\Big)\cdot\delta\underline{x} \\ &\quad -R_r \end{aligned} \tag{5.29}$$

where

$$R_r = \sum_m < \underline{y}^m, \underline{y} > \sum_{j=2}^{r} \binom{r}{j} < \underline{x}^m, \underline{x} >^{r-j} < \underline{x}^m, \delta\underline{x} >^j . \tag{5.30}$$

The first term is nonincreasing, the second term can be considered as a term of multiple associations, and the term R_r only may be increasing.

5.2 Simulation results of quadratic memories

Numerical simulations of the quadratic memory using outer product storage were performed on Eqs. 4.3 and 4.4 for $r = 2$. For an autoassociative memory (i.e., $\underline{y}^m = \underline{x}^m$), feedback can be used in a fashion completely analogous to a Hopfield-type network[12, 17]. The simulations were done for autoassociative case with feedback. Binary valued vectors in $\{1, -1\}^N$ were randomly selected. Two sets of experiments are displayed in Table 5.1. The first column in Table 5.1 lists the number of 63-bit vectors that were stored by sum of outer products and the second column lists the number of vectors that were correctly recalled in each case. Correct recall means that the vector obtained by implementing Eq. 4.3 exactly matched the input that is one of the stored vectors. The value in the parenthesis is the number of incorrect bits in the case of incorrect recalls. Notice for instance that when 140 vectors were stored only three were not correctly recalled. This represents a dramatic increase in the

$N = 63$			$N = 32$		
Stored vectors	Stable vectors		Stored vectors	Stable vectors	Attraction radius
60	60		8	8	9.0
70	70		16	16	8.3
80	79	(1)	24	24	8.3
90	89	(1)	32	32	8.0
100	98	(1)	40	39	7.0
110	108	(1)	48	45	5.0
120	119		56	49	4.8
130	128				
140	137				
150	140				

Table 5.1 Computer simulations.

capacity compared to the linear associative memory[12]. Also shown in Table 5.1 are the results of an experiment with 32-bit vectors. In this case the radius of attraction (i.e., the average number of bits that the initial input differs from a stored vector and still converges to it) was also measured. The decrease in the radius of attraction that accompanies the increase in the number of stored vectors is very gradual.

The computer simulations are consistent with the capacity result, $M_2 = M_1 \frac{N}{3}$, given by Eq. 4.20. For instance, for $N = 63$, we expect an increase in storage capacity by a factor of approximately 20. If we take as 140 the number of vectors that were correctly stored in this case, then we calculate the number of vectors that can be stored in the linear memory to be 7, which is about right. For $N = 32$, the number of vectors that can be recalled correctly is about 40, which implies that the increase in the storage capacity is a factor of 10.

5.3 Sensitivity of higher order associative memories

5.3.1 Analysis

Consider, first, the single r-th order expansion with noise added to both the inputs and the weights,

$$y_l = \mathrm{sgn}\Big\{ \sum_{j_1 \cdots j_r} (w_{lj_1 \cdots j_r} + n^w_{lj_1 \cdots j_r})(x_{j_1} \cdots x_{j_r} + n^x_{j_1 \cdots j_r}) + (w_0 + n^w_0) \Big\}, \tag{5.31}$$

where $n^w_{lj_1 \cdots j_r}$ is a collection of channel noise for the desired weight $w_{lj_1 \cdots j_r}$ of zero average and a variance σ^2_w, $n^x_{j_1 \cdots j_r}$ a collection of input noise for the desired input $x_{j_1} \cdots x_{j_r}$ whose average and variances are given by zero and σ^2_x, and n^w_0 a threshold noise of zero average and a variance σ^2_t. In what follows, a variety of architectures (or implementations) of higher order memories will be discussed, which introduce different types of input and channel noise.

Architecture I As the simplest case, the first order simple sum of outer product learning rule is discussed with the assumption that noise is mutually independent and interconnection(or channel) noise, which depends on the training method as well as the architecture, is independent of the stored data. When the input is set to be one of the stored data, Eq. 5.31 is reduced to

$$y_i = \mathrm{sgn}\Big\{ \sum_{j=1}^{N} (w_{ij} + n^w_{ij})(x^n_j + n^x_j) + n^w_0 \Big\}, \tag{5.32}$$

where $w_{ij} = \sum_{m=1}^{M} y^m_i x^m_j$. After expanding the sum, we have

$$y_i = \mathrm{sgn}\Big\{ N y^n_i + < \underline{x}^n, \underline{n}^x > y^n_i + \sum_{m \neq n} y^m_i < \underline{x}^m, \underline{x}^n + \underline{n}^x > + \sum_j n^w_{ij}(x^n_j + n^x_j) + n^w_0 \Big\}, \tag{5.33}$$

where the desired signal term is $N y^n_i$ and the other terms constitute a noise term whose average is zero; the third term describes cross talk noise not related to system noise. The variance of noise term can be shown to be $(M-1)N(1+\sigma^2_x) + N\sigma^2_x +$

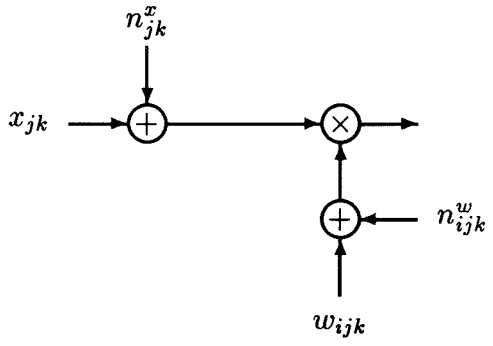


Figure 5.2 Architecture I for $r = 2$.

$N\sigma_w^2(1+\sigma_x^2)+\sigma_t^2$ using the assumption. Therefore, the SNR becomes approximately

$$\text{SNR} \approx \sqrt{\frac{N}{M(1+\sigma_x^2)+\sigma_w^2(1+\sigma_x^2)+\sigma_t^2/N}}. \tag{5.34}$$

When each input $x_{j_1}\cdots x_{j_r}$ is pre-thresholded before it is applied to the system we can also get the SNR of r-th order memory with sum of outer product learning rule in general as shown in Eq. 5.31 with the same assumption on noise and $w_{lj_1\cdots j_r}$ being $\sum_{m=1}^{M} y_l^m x_{j_1}^m \cdots x_{j_r}^m$: when the input $x_{j_1}\cdots x_{j_r}$ is not pre-thresholded but applied directly to the system (Fig. 5.2 when $r = 2$), noise will be added componentwise to be comparable in magnitude to the desired signal as will be discussed later. Then, when the input is one of the stored samples,

$$\begin{aligned} y_l \;=\; & \operatorname{sgn}\Big\{N^r y_l^n + \Big(\sum_{j_1\cdots j_r} x_{j_1}^n \cdots x_{j_r}^n n^x_{j_1\cdots j_r}\Big) y_i^n + \sum_{m\neq n} y_l^m < \underline{x}^m, \underline{x}^n >^r \\ & + \sum_{m\neq n} y_l^m \sum_{j_1\cdots j_r} x_{j_1}^m \cdots x_{j_r}^m n^x_{j_1\cdots j_r} + \sum_{j_1\cdots j_r} n^w_{lj_1\cdots j_r}(x_{j_1}^n \cdots x_{j_r}^n + n^x_{j_1\cdots j_r}) + n_0^w\Big\} \end{aligned} \tag{5.35}$$

where the first term in the right-hand side is the desired signal term and the other terms constitute a noise term n_l of zero average; the third term is cross talk as

before. The variance of the noise term can be derived with the help of Eqs. 4.9, 4.10, and 4.13,

$$\mathrm{E}\{n_l^2\} \approx (M-1)N^r\Big\{\frac{(2r)!}{2^r r!}+\sigma_x^2\Big\}+N^r\sigma_x^2+N^r\sigma_w^2(1+\sigma_x^2)+\sigma_t^2, \tag{5.36}$$

assuming interconnection(or channel) weights are independent of each other, that is, different permutations of the same index set result in separate channels,

$$\mathrm{E}\{n^w_{lj_1\cdots j_r}\cdots n^w_{ls_1\cdots s_r}\} = \mathrm{E}\{n^x_{j_1\cdots j_r}\cdots n^x_{s_1\cdots s_r}\} = \delta_{j_1 s_1}\cdots\delta_{j_r s_r}, \tag{5.37}$$

whereas the case where the channel weights of different permutations of an index set add up to one will also be considered, ending in a different architecture (Architecture II). Therefore, the SNR is given by

$$\mathrm{SNR} \approx \Big\{\frac{N^r}{M\{\frac{(2r)!}{2^r r!}+\sigma_x^2\}+\sigma_w^2(1+\sigma_x^2)+\sigma_t^2/N^r}\Big\}^{\frac{1}{2}}. \tag{5.38}$$

When the diagonal terms are zero as in Eq. 4.15 and the input is one of the stored vectors, Eq. 5.31 is written

$$\begin{aligned} y_l \;=\; & \mathrm{sgn}\Big\{\mathrm{P}(N,r)y_l^n+\Big(\sum_{\text{different } j} x^n_{j_1}\cdots x^n_{j_r}n^x_{j_1\cdots j_r}\Big)y_l^n \\ & + \sum_{m\neq n} y_l^m \sum_{\text{different } j} x^m_{j_1}\cdots x^m_{j_r}(x^n_{j_1}\cdots x^n_{j_r}+n^x_{j_1\cdots j_r}) \\ & + \sum_{\{\text{different}\}\, j} n^w_{lj_1\cdots j_r}(x^n_{j_1}\cdots x^n_{j_r}+n^x_{j_1\cdots j_r})+n_0^w\Big\} \end{aligned} \tag{5.39}$$

depending on whether the input $x_{j_1}\cdots x_{j_r}$ has diagonal components where $\mathrm{P}(m,n)$ is defined to be $m!/(m-n)!$ as before. The first term is a signal, and the variance of the noise term, the average of which is zero, is given by either

$$\sigma^2 = (M-1)\mathrm{P}(N,r)(r!+\sigma_x^2)+\mathrm{P}(N,r)\sigma_x^2+\mathrm{P}(N,r)\sigma_w^2(1+\sigma_x^2)+\sigma_t^2 \tag{5.40}$$

or

$$\sigma^2 = (M-1)\mathrm{P}(N,r)(r!+\sigma_x^2)+\mathrm{P}(N,r)\sigma_x^2+N^r\sigma_w^2(1+\sigma_x^2)+\sigma_t^2, \tag{5.41}$$

where Eq. 5.40 is for the case of having no input diagonal terms and Eq. 5.41 is not. Then, the approximation $\mathrm{P}(N,r) \approx N^r$ for $r \ll N$ gives the SNR of both cases,

$$\mathrm{SNR} \approx \left\{ \frac{N^r}{M(r!+\sigma_x^2)+\sigma_w^2(1+\sigma_x^2)+\sigma_t^2/N^r} \right\}^{\frac{1}{2}}. \tag{5.42}$$

Therefore, it can be concluded from Eqs. 5.38 and 5.42 that higher order memories of this architecture are less sensitive to system noise than lower order ones because system noise has less influence on their performance as the order increases. In optical implementations the dynamic ranges of interconnection weights are bounded because of device characteristics. Moreover, the weighted sum of input vectors to a neuron can have bounded values. It is due to the noise in channels and in devices. And it will be discussed in the next section.

Architecture II In the case where the channel weights of different permutations of the same index set add up to one, the calculation of the variance of noise is rather simple for the memories with zero diagonal terms and the input $x_{j_1} \cdots x_{j_r}$ with no diagonals, and then the SNR can be shown to be

$$\mathrm{SNR} \approx \left\{ \frac{N^r/r!}{M(1+\sigma_x^2)+\sigma_w^2(1+\sigma_x^2)+\sigma_t^2/N^r r!} \right\}^{\frac{1}{2}}. \tag{5.43}$$

When the input has diagonals, with the help of Lemma 12, the SNR can be derived for $r \ll N$ from

$$\sigma^2 = M\mathrm{P}(N,r)r!(1+\sigma_x^2)+\mathrm{R}(N,r)r!\sigma_w^2(1+\sigma_x^2)+\sigma_t^2. \tag{5.44}$$

Lemma 12 *For all the possible index sets* $\{j_1, j_2, \cdots, j_r\}$ $(1 \le j_1, j_2, \cdots, j_r \le N)$ *the total number of permutations,* $R(N,r)$*, of indexes* $s_1, s_2, \cdots, s_r$*, such that*

$$\{s_1, s_2, \cdots, s_r\} = \{j_1, j_2, \cdots, j_r\}, \qquad 1 \le s_1, s_2, \cdots, s_r \le N \tag{5.45}$$

is given by

$$P(N,r)r! \le R(N,r) \le N^r r!. \tag{5.46}$$

Proof The lower bound is derived by considering the case of all indexes j having different values and the upper bound by ordering the indexes k for any case of indexes j.

Since it is a good approximation for $r \ll N$ that $\mathrm{P}(N,r) \approx N^r$, the SNR is given by Eq. 5.43, again.

With the memory having non-zero diagonals like Eq. 5.35 the variance of noise becomes, according to Lemma 12

$$\mathrm{E}\{n_l^2\} = (M-1)N^r\frac{(2r)!}{2^r r!} + M\mathrm{R}(N,r)\sigma_x^2 + \mathrm{R}(N,r)\sigma_w^2(1+\sigma_x^2) + \sigma_t^2. \tag{5.47}$$

The SNR in this case is given by

$$\mathrm{SNR} \approx \left\{\frac{N^r}{M\{\frac{(2r)!}{2^r r!} + r!\sigma_x^2\} + r!\sigma_w^2(1+\sigma_x^2) + \sigma_t^2/N^r}\right\}^{\frac{1}{2}}. \tag{5.48}$$

Therefore, it follows that this architecture renders the effect of system noise comparable to that of cross talk between the stored samples even for the higher order case.

In the following we will discuss the cases where the input is not pre-thresholded (preprocessed) (III,IV), where the input is partially pre-thresholded (V,VI), where memory noise depends on each of memory data (VII,VIII,$\cdots$), and where the higher order memory is implemented using a power law (XIII).

Architecture III We first take into consideration the case where input noise is added componentwise to the input with the other same conditions given for the first architecture and only one input array is used to perform outer product operation so that input noise terms are added coherently. As an example, quadratic memories with the outer product learning rule are considered (Fig. 5.3),

$$y_i = \mathrm{sgn}\Big\{\sum_{jk}(w_{ijk} + n_{ijk}^w)(x_j + n_j^x)(x_k + n_k^x) + n_0^w\Big\}, \tag{5.49}$$

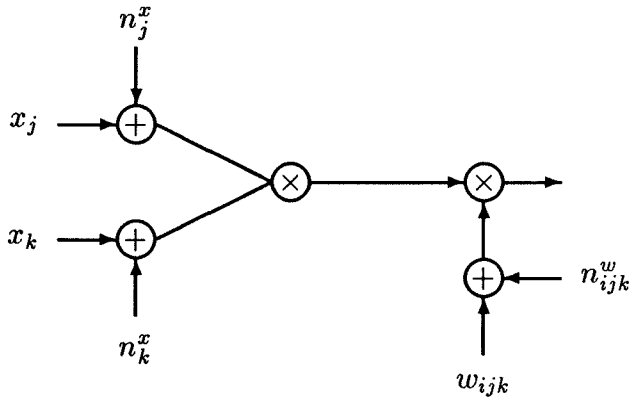


Figure 5.3 Architecture III for $r = 2$.

If the input is one of the stored data, and w_{ijk} is $\sum_m y_i^m x_j^m x_k^m$, the output y_i becomes

$$\begin{aligned} y_i &= \mathrm{sgn}\Big\{N^2 y_i^n + (2N < \underline{x}^n, \underline{n}^x > + < \underline{x}^n, \underline{n}^x >^2) y_i^n \\ &+ \sum_{m \neq n} y_i^m < \underline{x}^m, \underline{x}^n + \underline{n}^x >^2 + \sum_{jk} n_{ijk}^w (x_j^n + n_j^x)(x_k^n + n_k^x) + n_0^w \Big\}, \quad (5.50) \end{aligned}$$

where the first term $N^2 y_i^n$ is the only desired signal term and the signal value will be statistically $(N^2 + N\sigma_x^2) y_i^n$ due to the contribution of the term $< \underline{x}^n, \underline{n}^x >^2 y_i^n$. The variance of noise can be calculated with the help of Eqs. 4.9, 4.10, 4.11, 4.12, and 4.13 as follows: using the fact that the average and variance of $< \underline{x}^m, \underline{x}^n + \underline{x}^x >$ are given by zero and $N(1 + \sigma_x^2)$ when $m \neq n$,

$$\begin{aligned} &\mathrm{E}\Big\{\Big(\sum_{m \neq n} y_i^m < \underline{x}^m, \underline{x}^n + \underline{n}^x >^2\Big)^2\Big\} \\ &= (M-1) 3N^2 (1 + \sigma_x^2)^2, \quad (5.51) \end{aligned}$$

$$\begin{aligned} &\mathrm{E}\Big\{\Big(2N < \underline{x}^n, \underline{n}^x > + < \underline{x}^n, \underline{n}^x >^2) y_i^n\Big)^2 - (N\sigma_x^2)^2\Big\} \\ &= 4N^3 \sigma_x^2 + 2N^2 \sigma_x^2, \quad (5.52) \end{aligned}$$

$$
\begin{aligned}
&\mathrm{E}\Big\{\Big(\sum_{jk} n^w_{ijk}(x^n_j + n^x_j)(x^n_k + n^x_k)\Big)^2\Big\} \\
&= \mathrm{E}\Big\{\sum_{j_1 j_2 k_1 k_2} n^w_{ij_1k_1} n^w_{ij_2k_2}(x^n_{j_1} + n^x_{j1})(x^n_{j_2} + n^x_{j2})(x^n_{k_1} + n^x_{j1})(x^n_{k_2} + n^x_{j1})\Big\} \\
&= \sigma_w^2 \mathrm{E}\Big\{\sum_{jk}(x^n_j + n^x_j)^2(x^n_k + n^x_k)^2\Big\} \\
&= \sigma_w^2 \mathrm{E}\Big\{\sum_{jk}(1 + 2x^n_j n^x_j + n^{x2}_j)(1 + 2x^n_k n^x_k + n^{x2}_k)\Big\} \\
&= \sigma_w^2(N^2 + 2N^2\sigma_x^2 + 4N\sigma_x^2 + (N(N-1) + 3N)\sigma_x^4) \\
&\approx N^2\sigma_w^2(1 + \sigma_x^2)^2
\end{aligned} \tag{5.53}
$$

Thus, the SNR is

$$
\mathrm{SNR} \approx \left\{\frac{N^2}{3M(1+\sigma_x^2)^2 + 4N\sigma_x^2 + \sigma_w^2(1+\sigma_x^2)^2 + \sigma_t^2/N^2}\right\}^{\frac{1}{2}}. \tag{5.54}
$$

This can be extended to the r-th order memories,

$$
y_l = \mathrm{sgn}\Big\{\sum_{j_1 \cdots j_r}(w_{lj_1\cdots j_r} + n^w_{lj_1\cdots j_r})(x_{j_1} + n^x_{j_1}) \cdots (x_{j_r} + n^x_{j_r}) + n^w_0\Big\}. \tag{5.55}
$$

With the sum of outer product learning rule and an input being one of memory data it can be written

$$
\begin{aligned}
y_l &= \mathrm{sgn}\Big\{N^r y^n_l + y^n_l \sum_{k=1}^{r}\binom{r}{k} N^{r-k} < \underline{x}^n, \underline{n}^x >^k + \sum_{m \neq n} y^m_l < \underline{x}^m, \underline{x}^n + \underline{n}^x >^r \\
&+ \sum_{j_1 \cdots j_r} n^w_{lj_1\cdots j_r}(x^n_{j_1} + n^x_{j_1}) \cdots (x^n_{j_r} + n^x_{j_r}) + n^w_0\Big\}
\end{aligned} \tag{5.56}
$$

where the first term is a desired signal as before and the other terms constitute a noise term with an average of $\left(\binom{r}{2} N^{r-1}\sigma_x^2 + \cdots\right) y^n_l$ due to the even order terms of $< \underline{x}^n, \underline{n}^x >^k$ that is negligible compared with $N^r y^n_l$ for large N. For $r \ll N$ the variance of noise can be obtained as a sum of the following terms:

$$
\begin{aligned}
&\mathrm{E}\Big\{\Big(\sum_{m \neq n} y^m_l < \underline{x}^m, \underline{x}^n + \underline{n}^x >^r\Big)^2\Big\} \\
&= \mathrm{E}\Big\{\Big(\sum_m < \underline{x}^m, \underline{x}^n + \underline{n}^x >^{2r}\Big)^2\Big\} \\
&= (M-1)N^r \frac{(2r)!}{2^r r!}(1 + \sigma_x^2)^r,
\end{aligned} \tag{5.57}
$$

$$\begin{aligned}
&\mathrm{E}\Big\{\Big(y_l^n \sum_{k=1}^{r}\binom{r}{k} N^{r-k} < \underline{x}^n, \underline{n}^x >^k\Big)^2 - \Big(\binom{r}{2} N^{r-1}\sigma_x^2 + \cdots\Big)^2\Big\} \\
&= N^{2r-1} r^2 \sigma_x^2 + O(N^{2r-2} r^4 \sigma_x^4), \qquad (5.58)
\end{aligned}$$

$$\begin{aligned}
&\mathrm{E}\Big\{\Big(\sum_{j_1 \cdots j_r} n^w_{lj_1 \cdots j_r}(x^n_{j_1} + n^x_{j_1}) \cdots (x^n_{j_r} + n^x_{j_r})\Big)^2\Big\} \\
&= \sigma_w^2 \mathrm{E}\Big\{\Big(\sum_j (x_j^n + n_j^x)^2\Big)^r\Big\} \\
&\approx N^r \sigma_w^2 (1+\sigma_x^2)^r, \qquad (5.59)
\end{aligned}$$

The SNR, thus, follows

$$\mathrm{SNR} \approx \left\{\frac{N^r}{M\frac{(2r)!}{2^r r!}(1+\sigma_x^2)^r + N^{r-1} r^2 \sigma_x^2 + \sigma_w^2(1+\sigma_x^2)^r + \sigma_t^2/N^r}\right\}^{\frac{1}{2}}. \qquad (5.60)$$

When the diagonal terms of the memory are zero and the input is one of the memory data, the signal term will be $\mathrm{P}(N,r) y_l^n$, the variance of noise can be calculated for $r \ll N$ by the similar procedures, and the SNR can be obtained with the approximation of $\mathrm{P}(N,r) \approx N^r$,

$$\mathrm{SNR} \approx \left\{\frac{N^r}{Mr!(1+\sigma_x^2)^r + N^{r-1} r^2 \sigma_x^2 + \sigma_w^2(1+\sigma_x^2)^r + \sigma_t^2/N^r}\right\}^{\frac{1}{2}}. \qquad (5.61)$$

The effect of input noise, as shown in Eqs. 5.60 and 5.61, becomes outstanding in this architecture.

When one input array is used per each component of outer product operation, e.g., r distinct arrays for r-th order memory, so that the input noise terms are added in an incoherent way, Eq. 5.55 is modified to

$$y_l = \mathrm{sgn}\Big\{\sum_{j_1 \cdots j_r} (w_{lj_1 \cdots j_r} + n^w_{lj_1 \cdots j_r})(x_{j_1} + n^{x1}_{j_1}) \cdots (x_{j_r} + n^{xr}_{j_r}) + n_0^w\Big\}, \qquad (5.62)$$

where $\underline{n}^{x1}$, $\cdots$, and $\underline{n}^{xr}$ are mutually independent of each other. With the same input and memory, signal is $N^r y_l^n$ and noise variance is given by

$$\sigma^2 \approx N^{2r-1} r \sigma_x^2 + (M-1) N^r \Big\{\frac{(2r)!}{2^r r!} + r\frac{(2(r-1))!}{2^{r-1}(r-1)!}\sigma_x^2\Big\} + N^r \sigma_w^2 (1+\sigma_x^2)^r + \sigma_w^2. \qquad (5.63)$$

And the SNR, therefore, is given by

$$\mathrm{SNR} \approx \left\{ \frac{N^r}{M\frac{(2r)!}{2^r r!}(1+\frac{r}{2r-1}\sigma_x^2) + N^{r-1} r\sigma_x^2 + \sigma_w^2(1+\sigma_x^2)^r + \sigma_t^2/N^r} \right\}^{\frac{1}{2}}. \tag{5.64}$$

When the memory has zero diagonals, the SNR is

$$\mathrm{SNR} \approx \left\{ \frac{N^r}{Mr!(1+\sigma_x^2) + N^{r-1} r\sigma_x^2 + \sigma_w^2(1+\sigma_x^2)^r + \sigma_t^2/N^r} \right\}^{\frac{1}{2}}. \tag{5.65}$$

Compared with Eqs. 5.60 and 5.61, input noise effect is reduced by $1/r$ in this case.

Architecture IV If the channels of different permutations of the same indexes j are to add up to one, when input noise is accumulated in a coherent way, then Eq. 5.59 only will be modified for SNR,

$$\begin{aligned} &\mathrm{E}\Big\{\Big(\sum_{j_1\cdots j_r} n^w_{lj_1\cdots j_r}(x^n_{j_1}+n^x_{j_1})\cdots(x^n_{j_r}+n^x_{j_r})\Big)^2\Big\} \\ &\quad\approx \mathrm{R}(N,r)\sigma_w^2 + N\mathrm{R}(N,r-1)r\sigma_w^2\sigma_x^2 + O(N^r r^2(r-2)!\sigma_w^2\sigma_x^4). \end{aligned} \tag{5.66}$$

Therefore, the SNR is immediately

$$\mathrm{SNR} \approx \left\{ \frac{N^r}{M\frac{(2r)!}{2^r r!}(1+\sigma_x^2)^r + N^{r-1} r^2\sigma_x^2 + r!\sigma_w^2(1+\sigma_x^2) + \sigma_t^2/N^r} \right\}^{\frac{1}{2}}. \tag{5.67}$$

For the memory with zero diagonal terms, the SNR is easily obtained

$$\mathrm{SNR} \approx \left\{ \frac{N^r/r!}{M(1+\sigma_x^2)^r + \frac{N^{r-1}}{(r-1)!} r\sigma_x^2 + \sigma_w^2(1+\sigma_x^2) + \sigma_t^2/N^r r!} \right\}^{\frac{1}{2}}. \tag{5.68}$$

From Eqs. 5.67 and 5.68 it follows that memory noise as well as input noise have a non-negligible effect on the capacity of the memory.

When input noise is added incoherently, Eq. 5.59 will be

$$\begin{aligned} &\mathrm{E}\Big\{\Big(\sum_{j_1\cdots j_r} n^w_{lj_1\cdots j_r}(x^n_{j_1}+n^{x1}_{j_1})\cdots(x^n_{j_r}+n^{xr}_{j_r})\Big)^2\Big\} \\ &\quad= \mathrm{R}(N,r)\sigma_w^2 + N\mathrm{R}(N,r-1)r\sigma_w^2\sigma_x^2. \end{aligned} \tag{5.69}$$

The SNRs are, thus, given by

$$\mathrm{SNR} \approx \left\{ \frac{N^r}{M\frac{(2r)!}{2^r r!}(1+\frac{r}{2r-1}\sigma_x^2) + N^{r-1} r\sigma_x^2 + r!\sigma_w^2(1+\sigma_x^2) + \sigma_t^2/N^r} \right\}^{\frac{1}{2}}, \tag{5.70}$$

$$\mathrm{SNR} \approx \left\{ \frac{N^r/r!}{M(1+\sigma_x^2) + \frac{N^{r-1}}{(r-1)!}\sigma_x^2 + \sigma_w^2(1+\sigma_x^2) + \sigma_t^2/N^r r!} \right\}^{\frac{1}{2}}. \tag{5.71}$$

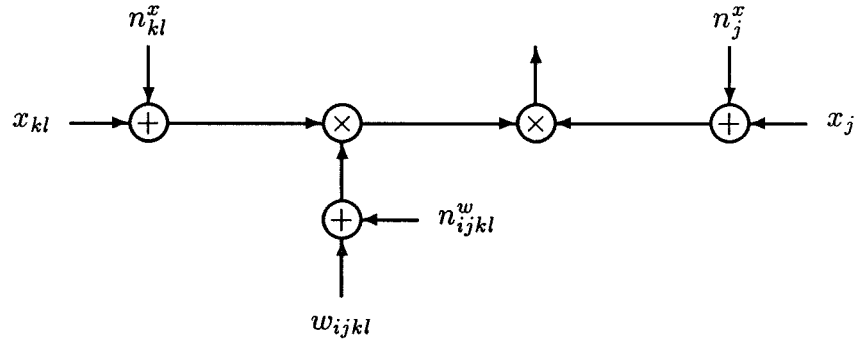


Figure 5.4 Architecture V for $r = 3$.

Architecture V The architectures III and IV can be improved by partial preprocessing when $r \geq 3$. Take into account an alternative description of a higher order memory whose interconnection weight between two neurons is affected not only by the states of these neurons themselves but also by the states of the other neurons as shown in Fig. 4.3. The input components that the weights are dependent on can be pre-thresholded. Then, for $r \geq 3$ (Fig. 5.4 for cubic memory), the output is given by

$$y_l = \mathrm{sgn}\Big\{\sum_{j_1}\Big(\sum_{j_2\cdots j_r}(w_{lj_1\cdots j_r} + n^w_{lj_1\cdots j_r})(x_{j_2}\cdots x_{j_r} + n^x_{j_2\cdots j_r})\Big)(x_{j_1} + n^x_{j_1}) + n^w_0\Big\}, \quad (5.72)$$

whereas the overall operation in the case of quadratic memory($r = 2$) is the same as that in architectures III and IV. With the outer product learning rule and the input being one of stored data,

$$\begin{aligned} y_l \;=\; & \mathrm{sgn}\Big\{N^r y_l^n + \Big(N\sum_{j_2\cdots j_r} x^n_{j_2}\cdots x^n_{j_r} n^x_{j_2\cdots j_r} + N^{r-1}\sum_j x^n_j n^x_j \\ & + \sum_{j_1\cdots j_r} x^n_{j_1}\cdots x^n_{j_r} n^x_{j_1} n^x_{j_2\cdots j_r}\Big) y^n_l + \sum_{m\neq n} y^m_l < \underline{x}^m, \underline{x}^n >^r \end{aligned}$$

$$+ \sum_{m\neq n} y_l^m \sum_{j_2\cdots j_r} < \underline{x}^m, \underline{x}^n > x_{j_2}^m \cdots x_{j_r}^m n_{j_2\cdots j_r}^x + \sum_{m\neq n} y_l^m \sum_j < \underline{x}^m, \underline{x}^n >^{r-1} x_j^m n_j^x$$
$$+ \sum_{m\neq n} y_l^m \sum_{j_1\cdots j_r} x_{j_1}^m \cdots x_{j_r}^m n_{j_1}^x n_{j_2\cdots j_r}^x + \sum_{j_1\cdots j_r} n_{lj_1\cdots j_r}^w (x_{j_1}^n + n_{j_1}^x)(x_{j_2}^n \cdots x_{j_r}^n + n_{j_2\cdots j_r}^x)$$
$$+ n_0^w \Big\}, \tag{5.73}$$

where $N^r y_l^n$ is a signal term and the other terms have zero average. The variance of noise and the SNR for $r \geq 3$ are respectively

$$\sigma^2 = (M-1)N^r \Big\{ \frac{(2r)!}{2^r r!} + \Big(1 + \frac{(2(r-1))!}{2^{r-1}(r-1)!}\Big)\sigma_x^2 + \sigma_x^4 \Big\} + (N^{r+1} + N^{2r-1})\sigma_x^2$$
$$+ N^r \sigma_x^4 + N^r \sigma_w^2 (1 + 2\sigma_x^2 + \sigma_x^4) + \sigma_t^2, \tag{5.74}$$

and

$$\text{SNR} \approx \Big\{ \frac{N^r}{M\Big\{\frac{(2r)!}{2^r r!} + \{1 + \frac{(2(r-1))!}{2^{r-1}(r-1)!}\}\sigma_x^2\Big\} + (N + N^{r-1})\sigma_x^2 + \sigma_w^2(1+\sigma_x^2)^2 + \sigma_t^2/N^r} \Big\}^{\frac{1}{2}}, \tag{5.75}$$

whereas the variance and SNR of quadratic memory are the same as those of architecture III.

For the memory having no diagonal terms ($r \geq 3$) the signal becomes $\mathrm{P}(N,r)y_l^n$ and the variance of noise is either

$$\sigma^2 = (M-1)\mathrm{P}(N,r)\Big\{r! + \{1 + (r-1)!\}\sigma_x^2 + \sigma_x^4\Big\} + \Big\{\mathrm{P}(N,r-1)(N-r+1)^2$$
$$+ N\mathrm{P}(N-1,r-1)^2\Big\}\sigma_x^2 + \mathrm{P}(N,r)s_x^4 + N^r\sigma_w^2(1 + 2\sigma_x^2 + \sigma_x^4) + \sigma_t^2, \tag{5.76}$$

or

$$\sigma^2 = (M-1)\mathrm{P}(N,r)\Big\{r! + \{1 + (r-1)!\}\sigma_x^2 + \sigma_x^4\Big\} + \Big\{\mathrm{P}(N,r-1)(N-r+1)^2$$
$$+ N\mathrm{P}(N-1,r-1)^2\Big\}\sigma_x^2 + \mathrm{P}(N,r)s_x^4 + \mathrm{P}(N,r)\sigma_w^2(1 + 2\sigma_x^2 + \sigma_x^4) + \sigma_t^2, \tag{5.77}$$

depending on whether the input $x_{j_1}\cdots x_{j_r}$ has diagonal components. The approximation $\mathrm{P}(N,r) \approx N^r$ for $r \ll N$ gives the SNR

$$\text{SNR} \approx \Big\{ \frac{N^r}{M\{r! + (1 + (r-1)!)\sigma_x^2\} + (N + N^{r-1})\sigma_x^2 + \sigma_w^2(1+\sigma_x^2)^2 + \sigma_t^2/N^r} \Big\}^{\frac{1}{2}}. \tag{5.78}$$

Quadratic memory has the same values as in architecture IV.

Architecture VI In the case that the channels of different permutations of the same indexes j are to add up to one, then the variance of noise is given by

$$\sigma^2 \approx (M-1)N^r\Big\{\frac{(2r)!}{2^r r!}+\Big((r-1)!+\frac{(2(r-1))!}{2^{r-1}(r-1)!}\Big)\sigma_x^2+\sigma_x^4\Big\}+\{N^{r+1}(r-1)! + N^{2r-1}\}\sigma_x^2+N^r(r-1)!\sigma_x^4+N^r\sigma_w^2\{r!+2(r-1)!\sigma_x^2\}+\sigma_t^2, \tag{5.79}$$

It follows that the SNR is

$$\text{SNR} \approx \Big\{\frac{N^r}{M\Big\{\frac{(2r)!}{2^r r!}+\{(r-1)!+\frac{(2(r-1))!}{2^{r-1}(r-1)!}\}\sigma_x^2\Big\}+\{N(r-1)!+N^{r-1}\}\sigma_x^2 + \frac{}{+r!\sigma_w^2(1+\frac{2}{r}\sigma_x^2)+\sigma_t^2/N^r}\Big\}^{\frac{1}{2}}. \tag{5.80}$$

For the memory with zero diagonals, the SNR can be easily shown to be

$$\text{SNR} \approx \Big\{\frac{N^r/r!}{M(1+\frac{2}{r}\sigma_x^2)+(\frac{N}{r}+\frac{N^{r-1}}{r(r-1)!})\sigma_x^2+\sigma_w^2(1+\frac{2}{r}\sigma_x^2)+\frac{\sigma_t^2}{N^r r!}}\Big\}^{\frac{1}{2}}. \tag{5.81}$$

Architecture VII From now on, we will consider the case where memory noise is dependent on each memory pattern but each memory noise is independent of other memory noise: it is expected in this case that the signal is degraded more by the memory noise. There may be three possible types of memory noise like those of input noise. One is the case when the training samples are preprocessed (or pre-thresholded), one is not, and the other is partially. The first case is discussed in this and the following two subsubsections with various types of input noise as presented in the preceding subsubsections, and then the others in separate subsubsections. Each subsubsection will cover two architectures that are described in terms of input noise earlier.

Consider Eq. 5.31 with the sum of outer product learning rule (Fig. 5.5 when $r=2$),

$$y_l = \text{sgn}\Big\{\sum_m \sum_{j_1\cdots j_r}(y_l^m x_{j_1}^m\cdots x_{j_r}^m + n_{lj_1\cdots j_r}^m)(x_{j_1}\cdots x_{j_r}+n_{j_1\cdots j_r}^x)+n_0^w\Big\} \tag{5.82}$$

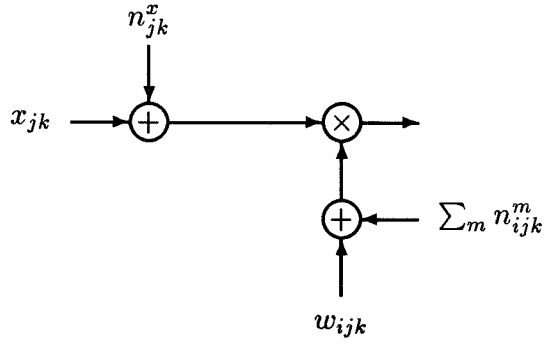


Figure 5.5 Architecture VII for $r = 2$.

where $n^m_{l j_1 \cdots j_r}$ is a noise of zero average and a variance σ_w^2 that is added to the m-th training sample, $n^x_{j_1 \cdots j_r}$ a collection of input noise for the input $x_{j_1} \cdots x_{j_r}$ whose average and variance are given by zero and σ_x^2, and n_0^w a threshold noise of zero average and a variance σ_t^2, as before.

With the assumptions given in architecture I and an input being one of stored data, the signals are the same and the variances for non-zero diagonal and zero diagonal cases are, respectively,

$$\mathrm{E}\{n_l^2\} \approx MN^r \Big\{ \frac{(2r)!}{2^r r!} + \sigma_x^2 + \sigma_w^2(1+\sigma_x^2) \Big\} + \sigma_t^2 \tag{5.83}$$

and

$$\mathrm{E}\{n_l^2\} \approx M\mathrm{P}(N,r)\{r! + \sigma_x^2 + \sigma_w^2(1+\sigma_x^2)\} + \sigma_t^2 \tag{5.84}$$

Therefore, the SNRs for non-zero diagonal and zero diagonal cases are, respectively, given by

$$\mathrm{SNR} \approx \left\{ \frac{N^r}{M\Big\{ \frac{(2r)!}{2^r r!} + \sigma_x^2 + \sigma_w^2(1+\sigma_x^2) \Big\} + \sigma_t^2/N^r} \right\}^{\frac{1}{2}} \tag{5.85}$$

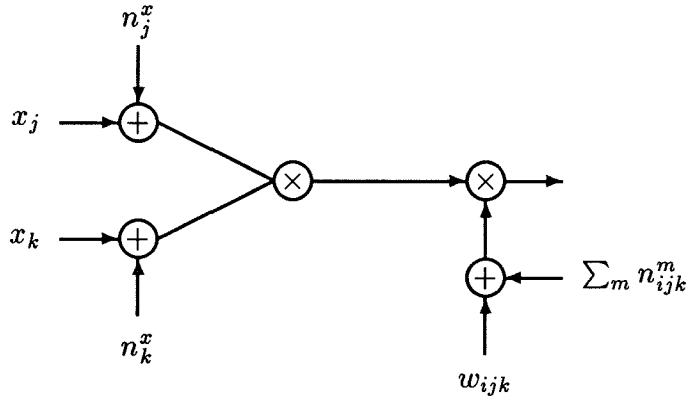


Figure 5.6 Architecture VIII for $r = 2$.

and

$$\text{SNR} \approx \left\{ \frac{N^r}{M\{r! + \sigma_x^2 + \sigma_w^2(1+\sigma_x^2)\} + \sigma_t^2/N^r} \right\}^{\frac{1}{2}}. \tag{5.86}$$

Using the assumptions in architecture II gives approximate SNRs for nonzero diagonal and zero diagonal cases, respectively, by exclusively replacing σ_x^2 or σ_w^2 by σ_x^2 or σ_w^2

$$\text{SNR} \approx \left\{ \frac{N^r}{M\left\{\frac{(2r)!}{2^r r!} + r!\sigma_x^2 + r!\sigma_w^2(1+\sigma_x^2)\right\} + \sigma_t^2/N^r} \right\}^{\frac{1}{2}} \tag{5.87}$$

and

$$\text{SNR} \approx \left\{ \frac{N^r/r!}{M\{1 + \sigma_x^2 + \sigma_w^2(1+\sigma_x^2)\} + \sigma_t^2/N^r r!} \right\}^{\frac{1}{2}}. \tag{5.88}$$

It will turn out that the capacity of the memory will be reduced considerably when dynamic range is considered.

Architecture VIII When input noise is added componentwise to the input in addition (Fig. 5.6 when $r = 2$), the situation will be much worse. Then, σ_w^2 will be switched by $M\sigma_w^2$ as shown in the previous subsubsection. The SNRs, with the

assumptions in architecture III and coherent input noise, become

$$\text{SNR} \approx \left\{ \frac{N^r}{M\left\{\frac{(2r)!}{2^r r!}(1+\sigma_x^2)^r + \sigma_w^2(1+\sigma_x^2)^r\right\} + N^{r-1}r^2\sigma_x^2 + \sigma_t^2/N^r} \right\}^{\frac{1}{2}} \tag{5.89}$$

and

$$\text{SNR} \approx \left\{ \frac{N^r}{M\left\{r!(1+\sigma_x^2)^r + \sigma_w^2(1+\sigma_x^2)^r\right\} + N^{r-1}r^2\sigma_x^2 + \sigma_t^2/N^r} \right\}^{\frac{1}{2}}. \tag{5.90}$$

With incoherent input noise the SNRs are

$$\text{SNR} \approx \left\{ \frac{N^r}{M\left\{\frac{(2r)!}{2^r r!}(1+\frac{r}{2r-1}\sigma_x^2) + \sigma_w^2(1+\sigma_x^2)^r\right\} + N^{r-1}r\sigma_x^2 + \sigma_t^2/N^r} \right\}^{\frac{1}{2}}, \tag{5.91}$$

$$\text{SNR} \approx \left\{ \frac{N^r}{M\left\{r!(1+\sigma_x^2) + \sigma_w^2(1+\sigma_x^2)^r\right\} + N^{r-1}r\sigma_x^2 + \sigma_t^2/N^r} \right\}^{\frac{1}{2}}. \tag{5.92}$$

The assumptions used in architecture IV give the SNRs

$$\text{SNR} \approx \left\{ \frac{N^r}{M\left\{\frac{(2r)!}{2^r r!}(1+\sigma_x^2)^r + r!\sigma_w^2(1+\sigma_x^2)\right\} + N^{r-1}r^2\sigma_x^2 + \sigma_t^2/N^r} \right\}^{\frac{1}{2}}, \tag{5.93}$$

$$\text{SNR} \approx \left\{ \frac{N^r/r!}{M\{(1+\sigma_x^2)^r + \sigma_w^2(1+\sigma_x^2)\} + \frac{N^{r-1}}{(r-1)!}r\sigma_x^2 + \sigma_t^2/N^r r!} \right\}^{\frac{1}{2}}, \tag{5.94}$$

$$\text{SNR} \approx \left\{ \frac{N^r}{M\left\{\frac{(2r)!}{2^r r!}(1+\frac{r}{2r-1}\sigma_x^2) + r!\sigma_w^2(1+\sigma_x^2)\right\} + N^{r-1}r\sigma_x^2 + \sigma_t^2/N^r} \right\}^{\frac{1}{2}},$$

and

$$\text{SNR} \approx \left\{ \frac{N^r/r!}{M\left\{(1+\sigma_x^2) + \sigma_w^2(1+\sigma_x^2)\right\} + \frac{N^{r-1}}{(r-1)!}\sigma_x^2 + \sigma_t^2/N^r r!} \right\}^{\frac{1}{2}}. \tag{5.95}$$

Architecture IX This subsubsection will cover two assumptions as given in architectures V and VI that have improved the architectures III and IV for $r \geq 3$ (Fig. 5.7 when $r = 3$). It is straightforward since only σ_w^2 is required to be modified to $M\sigma_w^2$. Therefore, the SNRs under the assumptions in architecture V are

$$\text{SNR} \approx \left\{ \frac{N^r}{M\left\{\frac{(2r)!}{2^r r!} + \{1 + \frac{(2(r-1))!}{2^{r-1}(r-1)!}\}\sigma_x^2 + \sigma_w^2(1+\sigma_x^2)^2\right\} + (N + N^{r-1})\sigma_x^2 + \sigma_t^2/N^r} \right\}^{\frac{1}{2}} \tag{5.96}$$

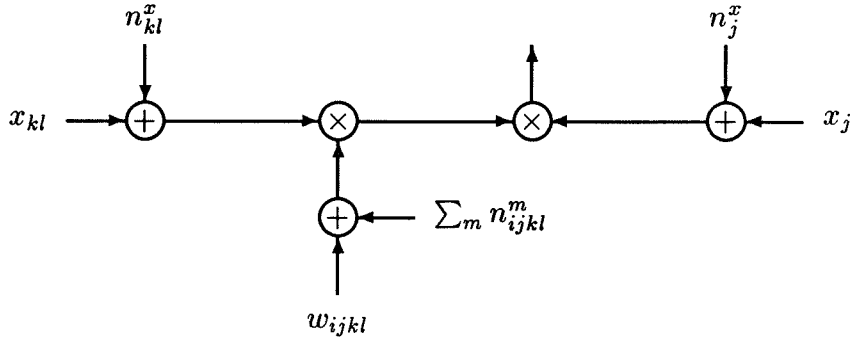


Figure 5.7 Architecture IX for $r = 3$.

and

$$\text{SNR} \approx \left\{ \frac{N^r}{M\{r! + (1 + (r-1)!)\sigma_x^2 + \sigma_w^2(1+\sigma_x^2)^2\} + (N + N^{r-1})\sigma_x^2 + \sigma_t^2/N^r} \right\}^{\frac{1}{2}}. \tag{5.97}$$

And with those in architecture VI

$$\text{SNR} \approx \left\{ \frac{N^r}{M\left\{\frac{(2r)!}{2^r r!} + \{(r-1)! + \frac{(2(r-1))!}{2^{r-1}(r-1)!}\}\sigma_x^2 + r!\sigma_w^2(1 + \frac{2}{r}\sigma_x^2)\right\} + \{N(r-1)! + N^{r-1}\}\sigma_x^2 + \sigma_t^2/N^r} \right\}^{\frac{1}{2}} \tag{5.98}$$

and

$$\text{SNR} \approx \left\{ \frac{N^r/r!}{M\{1 + \frac{2}{r}\sigma_x^2 + \sigma_w^2(1 + \frac{2}{r}\sigma_x^2)\} + (\frac{N}{r} + \frac{N^{r-1}}{r(r-1)!})\sigma_x^2 + \frac{\sigma_t^2}{N^r r!}} \right\}^{\frac{1}{2}}. \tag{5.99}$$

Architecture X When the training samples are not pre-thresholded and training sample and input noise is accumulated in a coherent manner, then applying the sum of outer product learning rule as a learning algorithm will produce the output with

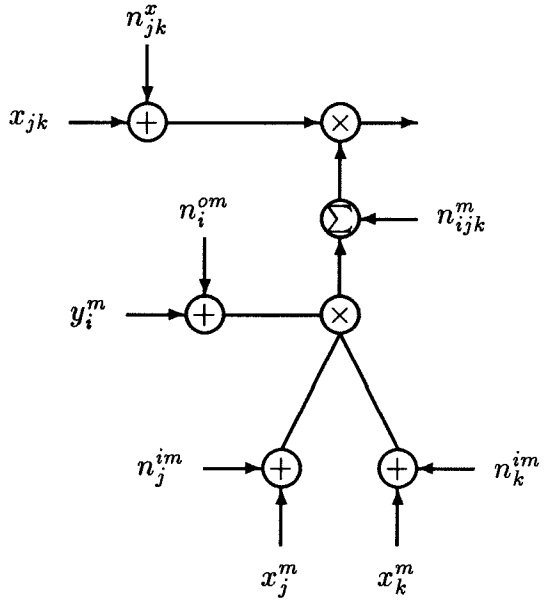


Figure 5.8 Architecture X for $r = 2$.

noise of the form (Fig. 5.8 when $r = 2$),

$$\begin{aligned} y_l &= \mathrm{sgn}\Big\{\sum_m \sum_{j_1 \cdots j_r} \{(y^m_l + n^{om}_l)(x^m_{j_1} + n^{im}_{j_1}) \cdots (x^m_{j_r} + n^{im}_{j_r}) + n^m_{lj_1 \cdots j_r}\} \\ &\qquad (x_{j_1} \cdots x_{j_r} + n^x_{j_1 \cdots j_r}) + n^w_0\Big\} \\ &= \mathrm{sgn}\Big\{\sum_m \sum_{j_1 \cdots j_r} (y^m_l + n^{om}_l)(x^m_{j_1} + n^{im}_{j_1}) \cdots (x^m_{j_r} + n^{im}_{j_r})(x_{j_1} \cdots x_{j_r} + n^x_{j_1 \cdots j_r}) \\ &\quad + \sum_m \sum_{j_1 \cdots j_r} n^m_{lj_1 \cdots j_r}(x_{j_1} \cdots x_{j_r} + n^x_{j_1 \cdots j_r}) + n^w_0\Big\} \end{aligned} \tag{5.100}$$

where $n^m_{lj_1 \cdots j_r}$ is a system noise of zero average and a variance σ^2_w that is added to the m-th training sample, n^{om}_l and n^{im}_j are training output and input noise with zero average and variance σ^2_T, $n^x_{j_1 \cdots j_r}$ a collection of input noise for the input $x_{j_1} \cdots x_{j_r}$ whose average and variance are given by zero and σ^2_x, respectively, and n^w_0 a threshold noise of zero average and a variance σ^2_t, where all the noise is independent *statistically* of each other and the input and memory samples.

In this subsubsection we will deal with the assumptions given in architectures I and II to derive the SNRs with the input being one of stored data. Then, Eq. 5.100 becomes

$$\begin{aligned} y_l &= \mathrm{sgn}\Big\{N^r y_l^n + y_l^n \sum_{k=1}^{r} \binom{r}{k} N^{r-k} < \underline{n}^{in}, \underline{x}^n >^k \\ &+ y_l^n \Big\{ \sum_{k=1}^{r} \binom{r}{k} \sum_{j_1 \cdots j_r} x_{j_1}^n \cdots x_{j_{r-k}}^n n_{j_{r-k+1}}^{in} \cdots n_{j_r}^{in} n_{j_1 \cdots j_r}^{x} + \cdots \Big\} \\ &+ \sum_{m \neq n} y_l^m \sum_{j_1 \cdots j_r} (x_{j_1}^m + n_{j_1}^{im}) \cdots (x_{j_r}^m + n_{j_r}^{im})(x_{j_1}^n \cdots x_{j_r}^n + n_{j_1 \cdots j_r}^{x}) \\ &+ \sum_{m} n_l^{om} \sum_{j_1 \cdots j_r} (x_{j_1}^m + n_{j_1}^{im}) \cdots (x_{j_r}^m + n_{j_r}^{im})(x_{j_1}^n \cdots x_{j_r}^n + n_{j_1 \cdots j_r}^{x}) \\ &+ \sum_{m} \sum_{j_1 \cdots j_r} n_{l j_1 \cdots j_r}^{m} (x_{j_1}^n \cdots x_{j_r}^n + n_{j_1 \cdots j_r}^{x}) + n_0^w \Big\} \end{aligned} \tag{5.101}$$

where all terms but the first that is a desired signal $N^r y_l^n$ constitute noise of average $\left(\binom{r}{2} N^{r-1}\sigma_x^2 + \cdots\right) y_l^n$ as in architecture III. Its variance is derived with the assumptions given in architecture I as follows:

$$\begin{aligned} &\mathrm{E}\Big\{\Big(y_l^n \sum_{k=1}^{r} \binom{r}{k} N^{r-k} \sum_{j_1 \cdots j_k} n_{j_1}^{in} \cdots n_{j_k}^{in} x_{j_1}^n \cdots x_{j_k}^n\Big)^2 - \Big(\binom{r}{2} N^{r-1}\sigma_x^2 + \cdots\Big)^2\Big\} \\ &= N^{2r-1} r^2 \sigma_T^2 + O(N^{2r-2} r^4 \sigma_T^4), \end{aligned} \tag{5.102}$$

$$\begin{aligned} &\mathrm{E}\Big\{\Big(y_l^n \Big\{ \sum_{k=1}^{r} \binom{r}{k} \sum_{j_1 \cdots j_r} x_{j_1}^n \cdots x_{j_{r-k}}^n n_{j_{r-k+1}}^{in} \cdots n_{j_r}^{in} n_{j_1 \cdots j_r}^{x} + \cdots \Big\}\Big)^2\Big\} \\ &= N^r r^2 \sigma_T^2 \sigma_x^2 + O(N^r r^4 \sigma_T^4 \sigma_x^2), \end{aligned} \tag{5.103}$$

$$\begin{aligned} &\mathrm{E}\Big\{\Big(\sum_{m \neq n} y_l^m \sum_{j_1 \cdots j_r} (x_{j_1}^m + n_{j_1}^{im}) \cdots (x_{j_r}^m + n_{j_r}^{im}) x_{j_1}^n \cdots x_{j_r}^n\Big)^2\Big\} \\ &= \mathrm{E}\Big\{\Big(\sum_{m \neq n} y_l^m < \underline{x}^m + \underline{n}^{im}, \underline{x}^n >^r\Big)^2\Big\} \\ &\approx (M-1) N^r \frac{(2r)!}{2^r r!} (1 + \sigma_T^2)^r, \end{aligned} \tag{5.104}$$

$$\begin{aligned} &\mathrm{E}\Big\{\Big(\sum_{m \neq n} y_l^m \sum_{j_1 \cdots j_r} (x_{j_1}^m + n_{j_1}^{im}) \cdots (x_{j_r}^m + n_{j_r}^{im}) n_{j_1 \cdots j_r}^{x}\Big)^2\Big\} \\ &= \sigma_x^2 \mathrm{E}\Big\{\Big(\sum_{m} \sum_{j_1 \cdots j_r} (x_{j_1}^m + n_{j_1}^{im})^2 \cdots (x_{j_r}^m + n_{j_r}^{im})^2\Big)^2\Big\} \\ &\approx (M-1) N^r (1 + \sigma_T^2)^r \sigma_x^2, \end{aligned} \tag{5.105}$$

$$\mathrm{E}\Big\{\Big(\sum_m n_l^{om} < \underline{x}^m + \underline{n}^{im}, \underline{x}^n >^r\Big)^2\Big\}$$
$$\approx \sigma_T^2\Big\{N^{2r} + N^{2r-1}r^2\sigma_T^2 + (M-1)N^r\frac{(2r)!}{2^r r!}(1+\sigma_T^2)^r\Big\}, \tag{5.106}$$

$$\mathrm{E}\Big\{\Big(\sum_m n_l^{om}\sum_{j_1\cdots j_r}(x_{j_1}^m + n_{j_1}^{im})\cdots(x_{j_r}^m + n_{j_r}^{im})n_{j_1\cdots j_r}^x\Big)^2\Big\}$$
$$\approx MN^r\sigma_T^2(1+\sigma_T^2)^r\sigma_x^2, \tag{5.107}$$

$$\mathrm{E}\Big\{\Big(\sum_m\sum_{j_1\cdots j_r} n_{lj_1\cdots j_r}^w(x_{j_1}^n\cdots x_{j_r}^n + n_{j_1\cdots j_r}^x)\Big)^2\Big\}$$
$$= MN^r\sigma_w^2(1+\sigma_x^2). \tag{5.108}$$

Therefore, the SNR is for $r \ll N$

$$\mathrm{SNR} \approx \Big\{\frac{N^r}{M\Big\{\frac{(2r)!}{2^r r!}+\sigma_x^2\Big\}(1+\sigma_T^2)^{r+1} + M\sigma_w^2(1+\sigma_x^2) + N^r\sigma_T^2 + N^{r-1}r^2(\sigma_T^2+\sigma_T^4) + \sigma_t^2/N^r}\Big\}^{\frac{1}{2}}. \tag{5.109}$$

For the memory with zero diagonal terms, it is not difficult to show that the SNR is given by

$$\mathrm{SNR} \approx \Big\{\frac{N^r}{M(r!+\sigma_x^2)(1+\sigma_T^2)^{r+1} + M\sigma_w^2(1+\sigma_x^2) + N^r\sigma_T^2 + N^{r-1}r^2(\sigma_T^2+\sigma_T^4) + \sigma_t^2/N^r}\Big\}^{\frac{1}{2}}. \tag{5.110}$$

When both training sample and input noise are added incoherently, Eq. 5.100 is modified into

$$y_l = \mathrm{sgn}\Big\{\sum_m\sum_{j_1\cdots j_r}\{(y_l^m + n_l^{om})(x_{j_1}^m + n_{j_1}^{im1})\cdots(x_{j_r}^m + n_{j_r}^{imr}) + n_{lj_1\cdots j_r}^m\}$$
$$(x_{j_1}\cdots x_{j_r} + n_{j_1\cdots j_r}^x) + n_0^w\Big\}$$
$$= \mathrm{sgn}\Big\{\sum_m\sum_{j_1\cdots j_r}(y_l^m + n_l^{om})(x_{j_1}^m + n_{j_1}^{im1})\cdots(x_{j_r}^m + n_{j_r}^{imr})(x_{j_1}\cdots x_{j_r} + n_{j_1\cdots j_r}^x)$$
$$+\sum_m\sum_{j_1\cdots j_r} n_{lj_1\cdots j_r}^m(x_{j_1}\cdots x_{j_r} + n_{j_1\cdots j_r}^x) + n_0^w\Big\} \tag{5.111}$$

where n^{im1}, $\cdots$, and n^{imr} are mutually independent training input noise. With an input being one of stored data, $\underline{x}^n$, the signal is given by $N^r y_l^n$, and a variance of noise whose average is zero given by

$$\begin{aligned}\sigma^2 \approx\ & N^{2r-1}r\sigma_T^2 + N^r r\sigma_T^2\sigma_x^2 + (M-1)N^r\Big\{\frac{(2r)!}{2^r r!} + r\frac{(2(r-1))!}{2^{r-1}(r-1)!}\sigma_T^2\Big\} \\ +\ & (M-1)N^r(1+\sigma_T^2)^r\sigma_x^2 + \sigma_T^2\Big\{N^{2r} + N^{2r-1}r\sigma_T^2 + (M-1)N^r\Big(\frac{(2r)!}{2^r r!} \\ +\ & r\frac{(2(r-1))!}{2^{r-1}(r-1)!}\sigma_T^2\Big)\Big\} + MN^r\sigma_T^2(1+\sigma_T^2)^r\sigma_x^2 + MN^r\sigma_w^2(1+\sigma_x^2) + \sigma_t^2. \quad (5.112)\end{aligned}$$

The SNR follows,

$$\begin{aligned}\text{SNR} \approx\ & \Bigg\{\frac{N^r}{M\Big\{\frac{(2r)!}{2^r r!}(1+\frac{r}{2r-1}\sigma_T^2) + (1+\sigma_T^2)^r\sigma_x^2\Big\}(1+\sigma_T^2) + M\sigma_w^2(1+\sigma_x^2)\Big\} + N^r\sigma_T^2} \\ & \overline{+N^{r-1}r(\sigma_T^2+\sigma_T^4) + \sigma_t^2/N^r}\Bigg\}^{\frac{1}{2}}. \quad (5.113)\end{aligned}$$

With memory having zero diagonal terms the SNR becomes

$$\begin{aligned}\text{SNR} \approx\ & \Bigg\{\frac{N^r}{M\{r!(1+\sigma_T^2) + (1+\sigma_T^2)^r\sigma_x^2\}(1+\sigma_T^2) + M\sigma_w^2(1+\sigma_x^2) + N^r\sigma_T^2} \\ & \overline{+N^{r-1}r(\sigma_T^2+\sigma_T^4) + \sigma_t^2/N^r}\Bigg\}^{\frac{1}{2}}. \quad (5.114)\end{aligned}$$

Under the assumptions given in architecture IV the SNRs for the cases of non-zero diagonal and zero diagonal memories are approximately given by

$$\begin{aligned}\text{SNR} \approx\ & \Bigg\{\frac{N^r}{M\Big\{\frac{(2r)!}{2^r r!}(1+\sigma_T^2)^r + r!\sigma_x^2(1+\sigma_T^2)\Big\}(1+\sigma_T^2) + Mr!\sigma_w^2(1+\sigma_x^2) + N^r\sigma_T^2} \\ & \overline{+N^r r^2(\sigma_T^2+\sigma_T^4) + \sigma_t^2/N^r}\Bigg\}^{\frac{1}{2}} \quad (5.115)\end{aligned}$$

$$\begin{aligned}\text{SNR} \approx\ & \Bigg\{\frac{N^r/r!}{M\{(1+\sigma_T^2)^r + \sigma_x^2(1+\sigma_T^2)\}(1+\sigma_T^2) + M\sigma_w^2(1+\sigma_x^2) + \frac{N^r}{r!}\sigma_T^2} \\ & \overline{+r\frac{N^{r-1}}{(r-1)!})(\sigma_T^2+\sigma_T^4) + \sigma_t^2/N^r r!}\Bigg\}^{\frac{1}{2}}, \quad (5.116)\end{aligned}$$

$$\begin{aligned}\text{SNR} \approx\ & \Bigg\{\frac{N^r}{M\Big\{\frac{(2r)!}{2^r r!}(1+\frac{r}{2r-1}\sigma_T^2) + r!(1+\sigma_T^2)\sigma_x^2\Big\}(1+\sigma_T^2) + Mr!\sigma_w^2(1+\sigma_x^2) + N^r\sigma_T^2} \\ & \overline{+N^{r-1}r(\sigma_T^2+\sigma_T^4) + \sigma_t^2/N^r}\Bigg\}^{\frac{1}{2}}, \quad (5.117)\end{aligned}$$

and

$$\text{SNR} \approx \left\{ \frac{N^r/r!}{M\{(1+\sigma_T^2)+(1+\sigma_T^2)\sigma_x^2\}(1+\sigma_T^2)+M\sigma_w^2(1+\sigma_x^2)+\frac{N^r}{r!}\sigma_T^2 \atop +\frac{N^{r-1}}{(r-1)}(\sigma_T^2+\sigma_T^4)+\sigma_t^2/N^r r!} \right\}^{\frac{1}{2}}. \tag{5.118}$$

As the equations show, memory noise due to the training method in which noise is added componentwise becomes significant.

Architecture XI The worst situation for both performance and analysis of higher order memories with outer product learning rule occurs when both input and memory noise exists componentwise in each bit of input and stored data and are accumulated in a coherent way such that the learning and recall procedures have the same type of process as follows and are very similar to those of power law implementation in analysis (Fig. 5.9 when $r = 2$):

$$\begin{aligned} y_l &= \text{sgn}\Big\{\sum_m \sum_{j_1\cdots j_r} \{(y_l^m + n_l^{om})(x_{j_1}^m + n_{j_1}^{im})\cdots(x_{j_r}^m + n_{j_r}^{im}) + n_{lj_1\cdots j_r}^m\} \\ &\qquad (x_{j_1} + n_{j_1}^x)\cdots(x_{j_r} + n_{j_r}^x) + n_0^w\Big\} \\ &= \text{sgn}\Big\{\sum_m (y_l^m + n_l^{om}) \sum_{j_1\cdots j_r} (x_{j_1}^m + n_{j_1}^{im})\cdots(x_{j_r}^m + n_{j_r}^{im})(x_{j_1} + n_{j_1}^x)\cdots(x_{j_r} + n_{j_r}^x) \\ &\quad + \sum_m \sum_{j_1\cdots j_r} n_{lj_1\cdots j_r}^m (x_{j_1} + n_{j_1}^x)\cdots(x_{j_r} + n_{j_r}^x) + n_0^w\Big\}. \end{aligned} \tag{5.119}$$

With input being one of stored samples the output becomes

$$\begin{aligned} y_l &= \text{sgn}\Big\{N^r y_l^n + y_l^n \sum_{k=1}^{r} N^{r-k} \sum_{i=0}^{k} \binom{r}{i}\binom{r}{k-i} \sum_{j_1\cdots j_k} x_{j_1}^n \cdots x_{j_k}^n n_{j_1}^{in}\cdots n_{j_i}^{in} n_{j_{i+1}}^x \cdots n_{j_k}^x \\ &+ \sum_{m\neq n} y_l^m \sum_{j_1\cdots j_r} (x_{j_1}^m + n_{j_1}^{im})\cdots(x_{j_r}^m + n_{j_r}^{im})(x_{j_1}^n + n_{j_1}^x)\cdots(x_{j_r}^n + n_{j_r}^x) \\ &+ \sum_m n_l^{om} \sum_{j_1\cdots j_r} (x_{j_1}^m + n_{j_1}^{im})\cdots(x_{j_r}^m + n_{j_r}^{im})(x_{j_1}^n + n_{j_1}^x)\cdots(x_{j_r}^n + n_{j_r}^x) \\ &+ \sum_m \sum_{j_1\cdots j_r} n_{lj_1\cdots j_r}^m (x_{j_1}^n + n_{j_1}^x)\cdots(x_{j_r}^n + n_{j_r}^x) + n_0^w\Big\}. \end{aligned} \tag{5.120}$$

The variance of noise whose average is given by $\left(\binom{r}{2}N^{r-1}(\sigma_T^2+\sigma_x^2)+\cdots\right)y_l^n$ that is negligible in comparison with $N^r y_l^n$ for large N is calculated with the help of

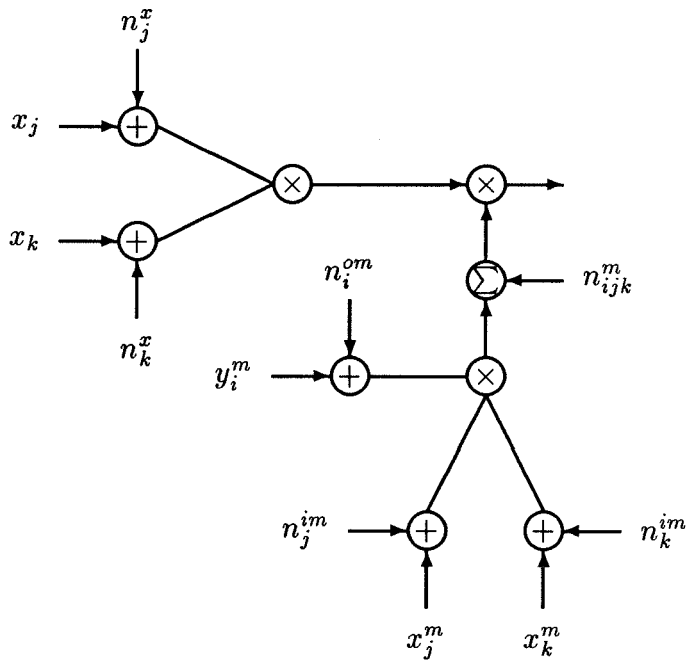


Figure 5.9 Architecture XI for $r = 2$.

Eqs. 5.57, 5.58, 5.59, 5.102, 5.103, 5.104, 5.105, 5.106, and 5.107 according to the assumptions in architecture III in the following way:

$$\begin{aligned}
&\mathrm{E}\Big\{\Big(y_l^n \sum_{k=1}^{r} N^{r-k} \sum_{i=0}^{k} \binom{r}{i}\binom{r}{k-i} \sum_{j_1\cdots j_k} <\underline{x}^n, \underline{n}^{in}>^i <\underline{x}^n, \underline{n}^x>^{k-i}\Big)^2 \\
&\quad - \Big(\binom{r}{2} N^{r-1}(\sigma_T^2+\sigma_x^2)+\cdots\Big)^2\Big\} \\
&\quad = N^{2r-1}r^2(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2) + O\Big(N^{2r-2}r^4(\sigma_T^4+\sigma_T^2\sigma_x^2+\sigma_x^4)\Big), \qquad (5.121)
\end{aligned}$$

$$\begin{aligned}
&\mathrm{E}\Big\{\Big(\sum_{m\neq n}\sum_{j_1\cdots j_r} y_l^m (x_{j_1}^m + n_{j_1}^{im})\cdots(x_{j_r}^m + n_{j_r}^{im})(x_{j_1}^n + n_{j_1}^x)\cdots(x_{j_r}^n + n_{j_r}^x)\Big)^2\Big\} \\
&\quad = \mathrm{E}\Big\{\sum_{m\neq n} <\underline{x}^m + \underline{n}^{im}, \underline{x}^n + \underline{n}^x>^{2r}\Big\} \\
&\quad \approx (M-1)N^r \frac{(2r)!}{2^r r!}(1+\sigma_T^2)^r(1+\sigma_x^2)^r, \qquad (5.122)
\end{aligned}$$

$$\mathrm{E}\Big\{\Big(\sum_m n_l^{om} \sum_{j_1\cdots j_r} (x_{j_1}^m + n_{j_1}^{im})\cdots(x_{j_r}^m + n_{j_r}^{im})(x_{j_1}^n + n_{j_1}^x)\cdots(x_{j_r}^n + n_{j_r}^x)\Big)^2\Big\}$$
$$\approx \sigma_T^2\Big\{N^{2r} + N^{2r-1}r^2(\sigma_T^2 + \sigma_x^2 + \sigma_T^2\sigma_x^2)$$
$$+ (M-1)N^r\frac{(2r)!}{2^r r!}(1+\sigma_T^2)^r(1+\sigma_x^2)^r\Big\}, \tag{5.123}$$

$$\mathrm{E}\Big\{\Big(\sum_m \sum_{j_1\cdots j_r} n_{lj_1\cdots j_r}^m (x_{j_1}^n + n_{j_1}^x)\cdots(x_{j_r}^n + n_{j_r}^x)\Big)^2\Big\}$$
$$\approx MN^r\sigma_w^2(1+\sigma_x^2)^r. \tag{5.124}$$

Therefore, the SNR follows from summing the above,

$$\mathrm{SNR} \approx \Bigg\{\frac{N^r}{M\Big\{\frac{(2r)!}{2^r r!}(1+\sigma_T^2)^{r+1}(1+\sigma_x^2)^r + \sigma_w^2(1+\sigma_x^2)^r\Big\} + N^r\sigma_T^2}$$
$$\frac{}{+N^{r-1}r^2(1+\sigma_T^2)(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2) + \sigma_t^2/N^r}\Bigg\}^{\frac{1}{2}}. \tag{5.125}$$

With memories having zero diagonal terms the SNR becomes

$$\mathrm{SNR} \approx \Bigg\{\frac{N^r}{M\{r!(1+\sigma_T^2)^{r+1}(1+\sigma_x^2)^r + \sigma_w^2(1+\sigma_x^2)^r\} + N^r\sigma_T^2}$$
$$\frac{}{+N^{r-1}r^2(1+\sigma_T^2)(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2) + \sigma_t^2/N^r}\Bigg\}^{\frac{1}{2}}. \tag{5.126}$$

When both memory and input noise are added incoherently, Eq. 5.119 is modified into

$$y_l = \mathrm{sgn}\Big\{\sum_m \sum_{j_1\cdots j_r} \{(y_l^m + n_l^{om})(x_{j_1}^m + n_{j_1}^{im1})\cdots(x_{j_r}^m + n_{j_r}^{imr}) + n_{lj_1\cdots j_r}^m\}$$
$$(x_{j_1} + n_{j_1}^{x1})\cdots(x_{j_r} + n_{j_r}^{xr}) + n_0^w\Big\}$$
$$= \mathrm{sgn}\Big\{\sum_m (y_l^m + n_l^{om}) \sum_{j_1\cdots j_r} (x_{j_1}^m + n_{j_1}^{im1})\cdots(x_{j_r}^m + n_{j_r}^{imr})(x_{j_1} + n_{j_1}^{x1})\cdots(x_{j_r} + n_{j_r}^{xr})$$
$$+ \sum_m \sum_{j_1\cdots j_r} n_{lj_1\cdots j_r}^m (x_{j_1} + n_{j_1}^{x1})\cdots(x_{j_r} + n_{j_r}^{xr}) + n_0^w\Big\}. \tag{5.127}$$

When the input is $\underline{x}^n$, the signal is $N^r y_l^n$, and a variance of noise, the average of which is zero, is given by

$$\sigma^2 \approx N^{2r-1}r(\sigma_T^2 + \sigma_x^2 + \sigma_T^2\sigma_x^2) + (M-1)N^r\Big\{\frac{(2r)!}{2^r r!} + \frac{(2(r-1))!}{2^{r-1}(r-1)!}r$$

$$(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)\Big\}\sigma_T^2\Big\{N^{2r}+N^{2r-1}r(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)+(M-1)N^r\Big(\frac{(2r)!}{2^r r!}$$
$$+\ \frac{(2(r-1))!}{2^{r-1}(r-1)!}r(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)\Big)\Big\}+MN^r\sigma_w^2(1+\sigma_x^2)^r. \tag{5.128}$$

The SNR becomes

$$\text{SNR}\ \approx\ \Big\{\frac{N^r}{M\Big\{\frac{(2r)!}{2^r r!}\Big(1+\frac{r}{2r-1}(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)\Big)(1+\sigma_T^2)+\sigma_w^2(1+\sigma_x^2)^r\Big\}+N^r\sigma_T^2}$$
$$\frac{}{+N^{r-1}r(1+\sigma_T^2)(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)+\sigma_t^2/N^r}\Big\}^{\frac{1}{2}}. \tag{5.129}$$

For the memory with zero diagonal terms, the SNR becomes

$$\text{SNR}\ \approx\ \Big\{\frac{N^r}{M\{r!(1+\sigma_T^2)^2(1+\sigma_x^2)+\sigma_w^2(1+\sigma_x^2)^r\}+N^r\sigma_T^2}$$
$$\frac{}{+N^{r-1}r(1+\sigma_T^2)(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)+\sigma_t^2/N^r}\Big\}^{\frac{1}{2}}. \tag{5.130}$$

When the channels of different permutations of an index set are all equivalent, only Eq. 5.124 is to be modified according to Lemma 12 and Eq. 5.66,

$$\text{E}\Big\{\Big(\sum_m\sum_{j_1\cdots j_r}n^m_{lj_1\cdots j_r}(x^n_{j_1}+n^x_{j_1})\cdots(x^n_{j_r}+n^x_{j_r})\Big)^2\Big\}\approx M\text{R}(N,r)\sigma_w^2(1+\sigma_x^2). \tag{5.131}$$

It gives the SNRs

$$\text{SNR}\ \approx\ \Big\{\frac{N^r}{M\Big\{\frac{(2r)!}{2^r r!}(1+\sigma_T^2)^{r+1}(1+\sigma_x^2)^r+r!\sigma_w^2(1+\sigma_x^2)\Big\}+N^r\sigma_T^2}$$
$$\frac{}{+N^{r-1}r^2(1+\sigma_T^2)(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)+\sigma_t^2/N^r}\Big\}^{\frac{1}{2}}, \tag{5.132}$$

$$\text{SNR}\ \approx\ \frac{N^r/r!}{M\{(1+\sigma_T^2)^{r+1}(1+\sigma_x^2)^r+\sigma_w^2(1+\sigma_x^2)\}+\frac{N^r}{r!}\sigma_T^2}$$
$$\frac{}{+\frac{N^{r-1}}{(r-1)!}r(1+\sigma_T^2)(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)+\sigma_t^2/N^r r!}\Big\}^{\frac{1}{2}} \tag{5.133}$$

$$\text{SNR}\ \approx\ \Big\{\frac{N^r}{M\Big\{\frac{(2r)!}{2^r r!}\Big(1+\frac{r}{2r-1}(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)\Big)(1+\sigma_T^2)+r!\sigma_w^2(1+\sigma_x^2)^r\Big\}+N^r\sigma_T^2}$$
$$\frac{}{+N^{r-1}r(1+\sigma_T^2)(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)+\sigma_t^2/N^r}\Big\}^{\frac{1}{2}}, \tag{5.134}$$

and

$$\mathrm{SNR} \approx \left\{ \frac{N^r/r!}{M\{(1+\sigma_T^2)^2(1+\sigma_x^2)+\sigma_w^2(1+\sigma_x^2)^r\}+\frac{N^r}{r!}\sigma_T^2 + \frac{N^{r-1}}{(r-1)!}(1+\sigma_T^2)(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)+\sigma_t^2/N^r r!} \right\}^{\frac{1}{2}}. \tag{5.135}$$

It can be concluded that this architecture that allows noise to exist component-wise both in input and in memory yields the worst performance in the presence of noise.

Architecture XII We will discuss the situation in which both training and recall processes have weight between two neurons depending on other neurons as well as themselves. Then, the output for $r \geq 3$ can be improved by partial preprocessing,

$$y_l = \mathrm{sgn}\Big\{\sum_{j_1}\Big(\sum_{j_2\cdots j_r} w_{lj_1\cdots j_r}(x_{j_2}\cdots x_{j_r}+n^x_{j_2\cdots j_r})\Big)(x_{j_1}+n^x_{j_1})+n_0^w\Big\}, \tag{5.136}$$

where $w_{lj_1\cdots j_r}$ is given by either

$$w_{lj_1\cdots j_r} = \sum_m (y_l^m x_{j_1}^m + n_{lj_1}^{m1})(x_{j_2}^m\cdots x_{j_r}^m + n_{j_2\cdots j_r}^{m2}) + n^w_{lj_1\cdots j_r} \tag{5.137}$$

or

$$w_{lj_1\cdots j_r} = \sum_m \Big\{(y_l^m x_{j_1}^m + n_{lj_1}^{m1})(x_{j_2}^m\cdots x_{j_r}^m + n_{j_2\cdots j_r}^{m2}) + n^m_{lj_1\cdots j_r}\Big\}. \tag{5.138}$$

Since it is straightforward to get the SNR for the memory of Eq. 5.138 from that of Eq. 5.137, the memory given by Eq. 5.137 is taken into consideration. As an example, cubic memory is considered (Fig. 5.10 when $r = 3$),

$$\begin{aligned} y_i &= \mathrm{sgn}\Big\{\sum_m\sum_{jkl}(y_i^m x_j^m + n_{ij}^{m1})(x_k^m x_l^m + n_{kl}^{m2})(x_j+n_j^x)(x_k x_l + n_{kl}^x) \\ &+ \sum_{jkl} n^w_{ijkl}(x_j+n_j^x)(x_k x_l + n_{kl}^x) + n_0^w\Big\} \end{aligned} \tag{5.139}$$

where when it is applied by input that is one of stored data the signal becomes $N^3 y_l^n$ and the variance of noise is obtained in the following way:

$$\mathrm{E}\Big\{\Big(\sum_m y_i^m \sum_{jkl} x_j^m (x_k^m x_l^m + n_{kl}^{m2})(x_j^n + n_j^x)(x_k^n x_l^n + n_{kl}^x) - N^3 y_l^n\Big)^2\Big\}$$

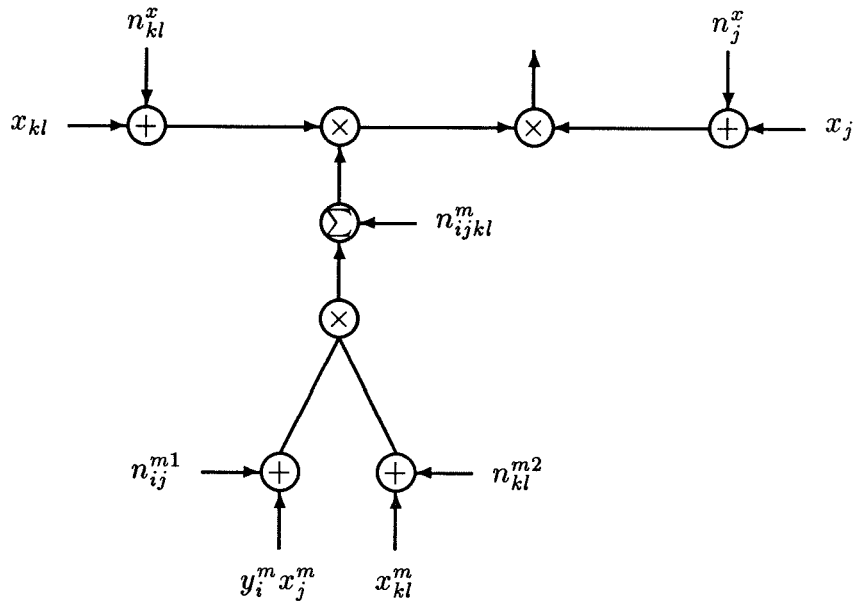


Figure 5.10 Architecture XII for $r = 3$.

$$
\begin{aligned}
&= (M-1)(15N^3 - 30N^2 + 16N) + N^2\sigma_x^2\{N^2 + (M-1)N\} \\
&+ N\sigma_x^2\{N^4 + (M-1)(3N^2 - 2N)\} + MN^3\sigma_x^4 + \sigma_T^2\Big\{N^2\{N^2 + (M-1)N\} \\
&+ N^2\sigma_x^2\{N^2 + (M-1)N\} + MN^3\sigma_x^2 + MN^3\sigma_x^4\Big\}, \qquad (5.140)
\end{aligned}
$$

$$
\begin{aligned}
&\mathrm{E}\Big\{\Big(\sum_m \sum_{jkl} n_{ij}^{m1}(x_k^m x_l^m + n_{kl}^{m2})(x_j^n + n_j^x)(x_k^n x_l^n + n_{kl}^x)\Big)^2\Big\} \\
&\quad = N\sigma_T^2\{N^4 + (M-1)(3N^2 - 2N)\} + MN^3\sigma_T^2\sigma_x^2 + N\sigma_T^2\sigma_x^2\{N^4 \\
&\qquad +(M-1)(3N^2 - 2N)\} + MN^3\sigma_T^2\sigma_x^4 + MN^3\sigma_T^4(1+\sigma_x^2)^2, \qquad (5.141)
\end{aligned}
$$

$$
\begin{aligned}
&\mathrm{E}\Big\{\Big(\sum_{jkl} n_{ijkl}^{w}(x_j^n + n_j^x)(x_k^n x_l^n + n_{kl}^x)\Big)^2\Big\} \\
&\quad = N^3\sigma_w^2(1+\sigma_x^2)^2. \qquad (5.142)
\end{aligned}
$$

The SNR, thus, follows from summing the above

$$\mathrm{SNR} \approx \left\{ \frac{N^3}{M\{15+4(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)+(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)^2\}} \right. \\ \left. \frac{}{+(N+N^2)(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)+\sigma_w^2(1+\sigma_x^2)^2+\sigma_t^2/N^3} \right\}^{\frac{1}{2}}. \quad (5.143)$$

It can be easily extended to r-th order memories. When one of stored data is applied as an input, the signal term in Eq. 5.136 becomes $N^r y_l^n$ and the variance of noise is derived with the help of Eqs. 5.73 and 5.74 as follows:

$$\begin{aligned} &\mathrm{E}\Big\{\Big(\sum_m \sum_{j_1\cdots j_r} y_l^m x_{j_1}^m (x_{j_2}^m \cdots x_{j_r}^m + n_{j_2\cdots j_r}^{m2})(x_{j_1}^n + n_{j_1}^x)(x_{j_2}^n \cdots x_{j_r}^n + n_{j_2\cdots j_r}^x) - N^r y_l^n\Big)^2\Big\} \\ &\approx (M-1)N^r \frac{(2r)!}{2^r r!} + N^{r-1}\sigma_x^2\{N^2+(M-1)N\} + N\sigma_x^2\Big\{N^{2(r-1)} \\ &+ (M-1)N^{r-1}\frac{(2(r-1))!}{2^{r-1}(r-1)!}\Big\} + MN^r\sigma_x^4 + \sigma_T^2\Big\{N^{r-1}\{N^2+(M-1)N\} \\ &+ N^{r-1}\sigma_x^2\{N^2+(M-1)N\} + MN^r\sigma_x^2 + MN^r\sigma_x^4\Big\}, \end{aligned} \quad (5.144)$$

$$\begin{aligned} &\mathrm{E}\Big\{\Big(\sum_m \sum_{j_1\cdots j_r} n_{lj_1}^{m1}(x_{j_2}^m \cdots x_{j_r}^m + n_{j_2\cdots j_r}^{m2})(x_{j_1}^n + n_{j_1}^x)(x_{j_2}^n \cdots x_{j_r}^n + n_{j_2\cdots j_r}^x)\Big)^2\Big\} \\ &= N\sigma_T^2\Big\{N^{2(r-1)} + (M-1)\frac{(2(r-1))!}{2^{r-1}(r-1)!}\Big\} + MN^r\sigma_T^2\sigma_x^2 + N\sigma_T^2\sigma_x^2\Big\{N^{2(r-1)} \\ &+ (M-1)\frac{(2(r-1))!}{2^{r-1}(r-1)!}\Big\} + MN^r\sigma_T^2\sigma_x^4 + MN^r\sigma_T^4(1+\sigma_x^2)^2, \end{aligned} \quad (5.145)$$

$$\begin{aligned} &\mathrm{E}\Big\{\Big(\sum_{j_1\cdots j_r} n_{lj_1\cdots j_r}^w (x_{j_1}^n + n_{j_1}^x)(x_{j_2}^n \cdots x_{j_r}^n + n_{j_2\cdots j_r}^x)\Big)^2\Big\} \\ &= N^r\sigma_w^2(1+\sigma_x^2)^2. \end{aligned} \quad (5.146)$$

Therefore, the SNR is given by

$$\mathrm{SNR} \approx \left\{ \frac{N^r}{M\Big\{\frac{(2r)!}{2^r r!} + \{1+\frac{(2(r-1))!}{2^{r-1}(r-1)!}\}(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)+(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)^2\Big\}} \right. \\ \left. \frac{}{+(N+N^{r-1})(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)+\sigma_w^2(1+\sigma_x^2)^2+\sigma_t^2/N^r} \right\}^{\frac{1}{2}}. \quad (5.147)$$

Notice existence of "beautiful" symmetricity of σ_T^2 and σ_x^2. When s_T^2 is zero and σ_x^4 is ignored, Eq. 5.147 is reduced to Eq 5.75.

With the memory having zero diagonal terms the SNR can be immediately shown to be approximately

$$\text{SNR} \approx \left\{ \frac{N^r}{M\Big\{r! + \{1+(r-1)!\}(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2) + (\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)^2\Big\} + (N+N^{r-1})(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2) + \sigma_w^2(1+\sigma_x^2)^2 + \sigma_t^2/N^r} \right\}^{\frac{1}{2}}. \quad (5.148)$$

The assumptions given in architecture IV give the SNRs for non-zero diagonal and zero diagonal memories, respectively,

$$\text{SNR} \approx \left\{ \frac{N^r}{M\Big\{\frac{(2r)!}{2^r r!} + \{(r-1)! + \frac{(2(r-1))!}{2^{r-1}(r-1)!}\}(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2) + (r-1)!(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)^2\Big\} + \{N(r-1)! + N^{r-1}\}(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2) + r!\sigma_w^2 + 2(r-1)!\sigma_w^2\sigma_x^2 + r!\sigma_w^2 + 2(r-1)!\sigma_w^2\sigma_x^2 + \sigma_t^2/N^r} \right\}^{\frac{1}{2}}, \quad (5.149)$$

and

$$\text{SNR} \approx \left\{ \frac{N^r/r!}{M\Big\{1 + \frac{2}{r}(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2) + \frac{1}{r}(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)^2\Big\} + \{\frac{N}{r} + \frac{N^{r-1}}{r(r-1)!}\}(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2) + \sigma_w^2 + \frac{2}{r}\sigma_w^2\sigma_x^2 + \sigma_t^2/N^r} \right\}^{\frac{1}{2}}. \quad (5.150)$$

With the memory given by Eq. 5.138, the only change in the variance of noise is Eq. 5.146

$$\begin{aligned} &\text{E}\Big\{\Big(\sum_m \sum_{j_1\cdots j_r} n^m_{l j_1\cdots j_r}(x^n_{j_1} + n^x_{j_1})(x^n_{j_2}\cdots x^n_{j_r} + n^x_{j_2\cdots j_r})\Big)^2\Big\} \\ &= MN^r\sigma_w^2(1+\sigma_x^2)^2. \end{aligned} \quad (5.151)$$

Therefore, four SNRs are given in order by

$$\text{SNR} \approx \left\{ \frac{N^r}{M\Big\{\frac{(2r)!}{2^r r!} + \{1 + \frac{(2(r-1))!}{2^{r-1}(r-1)!}\}(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2) + (\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)^2 + \sigma_w^2(1+\sigma_x^2)^2\Big\} + (N+N^{r-1})(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2) + \sigma_t^2/N^r} \right\}^{\frac{1}{2}}, \quad (5.152)$$

$$\text{SNR} \approx \left\{ \frac{N^r}{M\Big\{r! + \{1+(r-1)!\}(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2) + (\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)^2 + \sigma_w^2(1+\sigma_x^2)^2\Big\} + (N+N^{r-1})(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2) + \sigma_t^2/N^r} \right\}^{\frac{1}{2}}, \quad (5.153)$$

$$\text{SNR} \approx \left\{ \frac{N^r}{M\Big\{\frac{(2r)!}{2^r r!} + \{(r-1)! + \frac{(2(r-1))!}{2^{r-1}(r-1)!}\}(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2) + (r-1)!(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)^2 + \sigma_w^2\{r! + 2(r-1)!\sigma_x^2 + (r-1)!\sigma_x^4\}\Big\} + \{N(r-1)! + N^{r-1}\}(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2) + \sigma_t^2/N^r} \right\}^{\frac{1}{2}}, \quad (5.154)$$

and

$$\text{SNR} \approx \left\{ \frac{N^r/r!}{M\Big\{1 + \frac{2}{r}(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2) + \frac{1}{r}(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)^2 + \sigma_w^2(1+\frac{2}{r}\sigma_x^2+\frac{1}{r}\sigma_x^4)\Big\} + \{\frac{N}{r} + \frac{N^{r-1}}{r(r-1)!}\}(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2) + \sigma_t^2/N^r r!} \right\}^{\frac{1}{2}}. \quad (5.155)$$

Architecture XIII Finally, power law implementation is discussed under noise (Fig. 5.11):

$$y_l = \text{sgn}\Big\{\sum_m (y_l^m + n_l^{om})\Big(< \underline{x}^m + \underline{n}^{im}, \underline{x} + \underline{n}^x >^r + n_r^m\Big) + n_0^w\Big\}, \quad (5.156)$$

where n_r^m is a noise added to the signal in the r-th power law device for m-th stored vector and its variance is assumed to be given by σ_r^2. When the input is one of stored data, $\underline{x}^n$, the output becomes

$$\begin{aligned} y_l &= \text{sgn}\Big\{N^r y_l^n + \Big(\sum_{k=1}^{r} N^{r-k} \sum_{i=0}^{k} \binom{r}{i}\binom{r}{k-i} < \underline{x}^n, \underline{n}^x >^i < \underline{n}^{in}, \underline{x}^n >^{k-i}\Big) y_l^n \\ &+ \sum_{m \neq n} y_l^m < \underline{x}^m + \underline{n}^{im}, \underline{x}^n + \underline{n}^x >^r + \sum_m n_l^{om} < \underline{x}^m + \underline{n}^{im}, \underline{x}^n + \underline{n}^x >^r \\ &+ \sum_m (y_l^m + n_l^{om}) n_r^m + n_0^w\Big\} \end{aligned} \quad (5.157)$$

where the signal term is $N^r y_l^n$ and the variance of noise term whose average is given

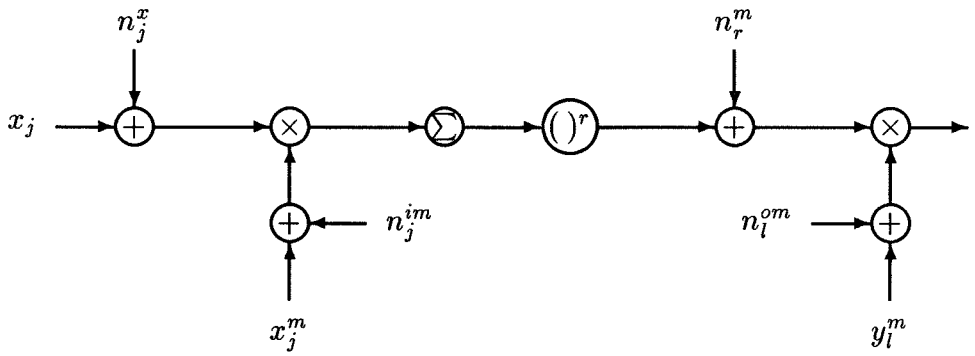


Figure 5.11 Architecture XIII for power law implementation.

by $\left(\binom{r}{2}N^{r-1}(\sigma_T^2+\sigma_x^2)+\cdots\right)y_l^n$ can be calculated as follows:

$$\begin{aligned}
&\mathrm{E}\Big\{\Big(y_l^n\sum_{k=1}^{r}N^{r-k}\sum_{i=0}^{k}\binom{r}{i}\binom{r}{k-i}<\underline{x}^n,\underline{n}^x>^i<\underline{n}^{in},\underline{x}^n>^{k-i}\Big)^2\\
&\quad-\Big(\binom{r}{2}N^{r-1}(\sigma_T^2+\sigma_x^2)+\cdots\Big)^2\Big\}\\
&\quad\approx\ N^{2r-1}r^2(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)+O\Big(N^{2r-2}r^4(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)\Big),
\end{aligned}\tag{5.158}$$

$$\begin{aligned}
&\mathrm{E}\Big\{\Big(\sum_{m\neq n}y_l^m<\underline{x}^m+\underline{n}^{im},\underline{x}^n+\underline{n}^x>^r\Big)^2\\
&\quad\approx\ (M-1)N^r\frac{(2r)!}{2^r r!}(1+\sigma_T^2)^r(1+\sigma_x^2)^r,
\end{aligned}\tag{5.159}$$

$$\begin{aligned}
&\mathrm{E}\Big\{\Big(\sum_m n_l^{om}<\underline{x}^m+\underline{n}^{im},\underline{x}^n+\underline{n}^x>^r\Big)^2\\
&\quad\approx\ \sigma_T^2\Big\{N^{2r}+N^{2r-1}r^2(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)\\
&\quad+\ (M-1)N^r\frac{(2r)!}{2^r r!}(1+\sigma_T^2)^r(1+\sigma_x^2)^r\Big\},
\end{aligned}\tag{5.160}$$

$$\begin{aligned}
&\mathrm{E}\Big\{\Big(\sum_m(y_l^m+n_l^{om})n_r^m\Big)^2\\
&\quad=\ M(1+\sigma_T^2)\sigma_r^2.
\end{aligned}\tag{5.161}$$

Thus, the SNR is given by

$$\text{SNR} \approx \left\{ \frac{N^r}{M\left\{\frac{(2r)!}{2^r r!}(1+\sigma_T^2)^{r+1}(1+\sigma_x^2)^r + \frac{1}{N^r}(1+\sigma_x^2)\sigma_r^2\right\} + N^r\sigma_T^2} \right.$$
$$\left. \frac{}{+N^{r-1}r^2(1+\sigma_T^2)(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)+\sigma_t^2/N^r} \right\}^{\frac{1}{2}}. \qquad (5.162)$$

5.3.2 Dynamic range consideration

In analog computation, dynamic range of the system that has bounded levels due to finite value that can be assumed and noise that exists in the system makes a limitation on the accuracy of performance[33]. Higher order memories with the outer product learning rule have higher values in input to the output unit. These values are approximately as big as N^r for r-th order but have their weights whose value is within order of $\pm\sqrt{M}$ with very high probability (99% within $\pm 3\sqrt{M}$), even though its maximum can be $\pm M$ where M is the number of stored data as shown earlier. When the full capacity is utilized, weights may have values of order of $\sqrt{M_r}$, that is, $\{N^r/\log N\}^{1/2}$. Therefore, it is required that weights and input to output unit be normalized. For instance, the simplest case given in Eq. 5.32 is modified,

$$y_i = \text{sgn}\left\{\frac{Y}{N}\frac{\sqrt{M}}{W}\sum_{j=1}^{N}\left(\frac{W}{\sqrt{M}}w_{ij}+n_{ij}^w\right)(x_j^n+n_j^x)+n_0^w\right\}, \qquad (5.163)$$

where Y and W are normalized values in input to output unit and weights within some orders, respectively, and w_{ij} is given by $\sum_{m=1}^{M} y_i^m x_j^m$ as before.

Eq. 5.163 can be rewritten,

$$y_i = \text{sgn}\left\{\frac{Y}{N}\left\{\sum_{j=1}^{N}\left(w_{ij}+\frac{\sqrt{M}}{W}n_{ij}^w\right)(x_j^n+n_j^x)+\frac{N}{Y}n_0^w\right\}\right\}. \qquad (5.164)$$

It is possible to think that the terms $\frac{\sqrt{M}}{W}n_{ij}^w$ and $\frac{N}{Y}n_0^w$ are system and threshold noise of zero averages and variances $\frac{M}{W^2}\sigma_w^2$ and $\frac{N^2}{Y^2}\sigma_t^2$, respectively. Then, when the input is one of stored data, the SNR becomes immediately

$$\text{SNR} \approx \sqrt{\frac{N}{M(1+\frac{1}{W^2}\sigma_w^2)(1+\sigma_x^2)+\frac{N}{Y^2}\sigma_t^2}}. \qquad (5.165)$$

Therefore, it is straightforward to find SNRs for architectures that were described, and the general one out of every four SNRs will be stated one by one with some examples.

Eq. 5.38, which was the best performance (architecture I) is modified,

$$\mathrm{SNR} \approx \left\{ \frac{N^r}{M\{\frac{(2r)!}{2^r r!} + \sigma_x^2 + \frac{1}{W^2}\sigma_w^2(1+\sigma_x^2)\} + \frac{N^r}{Y^2}\sigma_t^2} \right\}^{\frac{1}{2}}. \tag{5.166}$$

Consider Eq. 5.49 for quadratic memories with normalizations and coherent addition of input noise,

$$y_i = \mathrm{sgn}\left\{ \frac{Y}{N^2}\left\{ \sum_{jk}(w_{ijk} + \frac{\sqrt{M}}{W}n_{ijk}^w)(x_j + n_j^x)(x_k + n_k^x) + n_0^w \right\} \right\}. \tag{5.167}$$

Then the SNR given in Eq. 5.54 is modified with approximation

$$\mathrm{SNR} \approx \left\{ \frac{N^2}{M(3 + \frac{1}{W^2}\sigma_w^2)(1+\sigma_x^2)^2 + 4N\sigma_x^2 + \frac{N^2}{Y^2}\sigma_t^2} \right\}^{\frac{1}{2}}. \tag{5.168}$$

This is expanded to r-th order case instead of Eq. 5.60 (architecture III),

$$\mathrm{SNR} \approx \left\{ \frac{N^r}{M\{\frac{(2r)!}{2^r r!} + \frac{1}{W^2}\sigma_w^2\}(1+\sigma_x^2)^r + N^{r-1}r^2\sigma_x^2 + \frac{N^r}{Y^2}\sigma_t^2} \right\}^{\frac{1}{2}}. \tag{5.169}$$

In the case of incoherent addition of input noise the SNR in Eq. 5.64 is modified to

$$\mathrm{SNR} \approx \left\{ \frac{N^r}{M\left\{\frac{(2r)!}{2^r r!}(1 + \frac{r}{2r-1}\sigma_x^2) + \frac{1}{W^2}\sigma_w^2(1+\sigma_x^2)^r\right\} + N^{r-1}r\sigma_x^2 + \frac{N^r}{Y^2}\sigma_t^2} \right\}^{\frac{1}{2}}. \tag{5.170}$$

When the input is partially preprocessed like Eq. 5.72, the SNR that was given by Eq. 5.75 (architecture V) is modified,

$$\mathrm{SNR} \approx \left\{ \frac{N^r}{M\left\{\frac{(2r)!}{2^r r!} + \{1 + \frac{(2(r-1))!}{2^{r-1}(r-1)!}\}\sigma_x^2 + \frac{1}{W^2}\sigma_w^2(1+\sigma_x^2)^2\right\} + (N + N^{r-1})\sigma_x^2 + \frac{N^r}{Y^2}\sigma_t^2} \right\}^{\frac{1}{2}}. \tag{5.171}$$

Take a look at Eq. 5.82 (architecture VII) with normalization

$$y_l = \mathrm{sgn}\left\{ \frac{Y}{N^r}\left\{ \sum_m \sum_{j_1 \cdots j_r} (y_l^m x_{j_1}^m \cdots x_{j_r}^m + \frac{\sqrt{M}}{W}n_{lj_1\cdots j_r}^m)(x_{j_1}\cdots x_{j_r} + n_{j_1\cdots j_r}^x) + \frac{N^r}{Y}n_0^w \right\} \right\}. \tag{5.172}$$

Then the SNR given by Eq. 5.85 is modified

$$\mathrm{SNR} \approx \left\{ \frac{N^r}{M\left\{\frac{(2r)!}{2^r r!} + \sigma_x^2\right\} + M^2 \frac{1}{W^2}\sigma_w^2(1+\sigma_x^2) + \frac{N^r}{Y^2}\sigma_t^2} \right\}^{\frac{1}{2}}. \tag{5.173}$$

As the equation implies, the capacities of this and the following architectures with normalizations, which have memory noise depending on each stored data, may be dramatically reduced in order by half. This may imply that pre-calculation of weights or parallel learning of training samples is needed instead of serial learning.

When input noise is componentwise added to the input in addition (architecture VIII) and accumulated in a coherent manner, the SNR given by Eq. 5.89 is modified,

$$\mathrm{SNR} \approx \left\{ \frac{N^r}{M\frac{(2r)!}{2^r r!}(1+\sigma_x^2)^r + M^2 \frac{1}{W^2}\sigma_w^2(1+\sigma_x^2)^r + N^{r-1}r^2\sigma_x^2 + \frac{N^r}{Y^2}\sigma_t^2} \right\}^{\frac{1}{2}}. \tag{5.174}$$

In the case of incoherent accumulation of input noise the SNR in Eq. 5.91 is modified to

$$\mathrm{SNR} \approx \left\{ \frac{N^r}{M\frac{(2r)!}{2^r r!}(1+\frac{r}{2r-1}\sigma_x^2) + M^2 \frac{1}{W^2}\sigma_w^2(1+\sigma_x^2)^r + N^{r-1}r\sigma_x^2 + \frac{N^r}{Y^2}\sigma_t^2} \right\}^{\frac{1}{2}}. \tag{5.175}$$

With input partially preprocessed, the SNR given by Eq. 5.96 (architecture IX) is modified

$$\begin{aligned} \mathrm{SNR} \approx & \left\{ \frac{N^r}{M\left\{\frac{(2r)!}{2^r r!} + \{1 + \frac{(2(r-1))!}{2^{r-1}(r-1)!}\}\sigma_x^2\right\} + M^2 \frac{1}{W^2}\sigma_w^2(1+\sigma_x^2)^2} \right. \\ & \left. \frac{}{+(N+N^{r-1})\sigma_x^2 + \frac{N^r}{Y^2}\sigma_t^2} \right\}^{\frac{1}{2}}. \end{aligned} \tag{5.176}$$

When training samples are not preprocessed but the input is pre-thresholded (architecture X) and training sample noise is accumulated coherently, then the output is from Eq. 5.100,

$$\begin{aligned} y_l = \mathrm{sgn}\Big\{ \frac{Y}{N^r}\Big\{ \sum_m \sum_{j_1 \cdots j_r} \{(y_l^m + n_l^{om})(x_{j_1}^m + n_{j_1}^{im}) \cdots (x_{j_r}^m + n_{j_r}^{im}) + \frac{\sqrt{M}}{W} n_{lj_1 \cdots j_r}^m \} \\ (x_{j_1} \cdots x_{j_r} + n_{j_1 \cdots j_r}^x) + \frac{N^r}{Y} n_0^w \Big\} \Big\}. \end{aligned} \tag{5.177}$$

The SNR that was given by Eq. 5.109 is modified to

$$\text{SNR} \approx \left\{ \frac{N^r}{M\left\{\frac{(2r)!}{2^r r!}+\sigma_x^2\right\}(1+\sigma_T^2)^{r+1} + M^2\frac{1}{W^2}\sigma_w^2(1+r\sigma_x^2) + N^r\sigma_T^2 \atop +N^{r-1}r^2(\sigma_T^2+\sigma_T^4) + \frac{N^r}{Y^2}\sigma_t^2} \right\}^{\frac{1}{2}}. \tag{5.178}$$

When training sample noise is accumulated incoherently, the SNR in Eq. 5.113 is modified into

$$\text{SNR} \approx \left\{ \frac{N^r}{M\left\{\frac{(2r)!}{2^r r!}(1+\frac{r}{2r-1}\sigma_T^2)+(1+\sigma_T^2)^r\sigma_x^2\right\}(1+\sigma_T^2) + M^2\frac{1}{W^2}\sigma_w^2(1+\sigma_x^2)\right\} \atop +N^r\sigma_T^2 + N^{r-1}r(\sigma_T^2+\sigma_T^4) + \frac{N^r}{Y^2}\sigma_t^2} \right\}^{\frac{1}{2}}. \tag{5.179}$$

When both learning and recall procedures are not preprocessed (architecture XI) and both training sample and input noise are accumulated in an incoherent way, the SNR given in Eq. 5.125 is modified,

$$\text{SNR} \approx \left\{ \frac{N^r}{M\frac{(2r)!}{2^r r!}(1+\sigma_T^2)^{r+1}(1+\sigma_x^2)^r + M^2\frac{1}{W^2}\sigma_w^2(1+\sigma_x^2)^r + N^r\sigma_T^2 \atop +N^{r-1}r^2(1+\sigma_T^2)(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2) + \frac{N^r}{Y^2}\sigma_t^2} \right\}^{\frac{1}{2}}. \tag{5.180}$$

In the case of incoherent accumulation of both training sample and input noise the SNR in Eq. 5.129

$$\text{SNR} \approx \left\{ \frac{N^r}{M\frac{(2r)!}{2^r r!}\left(1+\frac{r}{2r-1}(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)\right)(1+\sigma_T^2) + M^r\frac{1}{W^2}\sigma_w^2(1+\sigma_x^2)^r \atop +N^r\sigma_T^2 + N^{r-1}r(1+\sigma_T^2)(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2) + \frac{N^r}{Y^2}\sigma_t^2} \right\}^{\frac{1}{2}}. \tag{5.181}$$

When the learning rule that has weights adapted to the states of all the neurons (architecture XII) is used, the SNR in Eq. 5.147 is modified,

$$\text{SNR} \approx \left\{ \frac{N^r}{M\left\{\frac{(2r)!}{2^r r!}+\{1+\frac{(2(r-1))!}{2^{r-1}(r-1)!}\}(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)+(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2)^2\right\} \atop +M^2\frac{1}{W^2}\sigma_w^2(1+\sigma_x^2)^2 + (N+N^{r-1})(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2) + \frac{N^r}{Y^2}\sigma_t^2} \right\}^{\frac{1}{2}}. \tag{5.182}$$

Finally, when power law implementation is used, Eq. 5.156 is modified to

$$y_l = \mathrm{sgn}\Big\{\frac{Y}{W}\Big\{\sum_m (y_l^m + n_l^{om})\Big(\frac{W}{N^r} < \underline{x}^m + \underline{n}^{im}, \underline{x} + \underline{n}^x >^r + n_r^m\Big) + \frac{N^r}{Y} n_0^w\Big\}\Big\}. \quad (5.183)$$

And the SNR in Eq. 5.162 is changed to

$$\mathrm{SNR} \approx \Bigg\{\frac{N^r}{M\Big\{\frac{(2r)!}{2^r r!}(1+\sigma_T^2)^{r+1}(1+\sigma_x^2)^r + \frac{N^r}{W^2}(1+\sigma_x^2)\sigma_r^2\Big\} + N^r\sigma_T^2 + N^{r-1}r^2(1+\sigma_T^2)(\sigma_T^2+\sigma_x^2+\sigma_T^2\sigma_x^2) + \frac{N^r}{Y^2}\sigma_t^2}\Bigg\}^{\frac{1}{2}}. \quad (5.184)$$

The most significant in this case is noise that exists in the output of power law device, e.g., nonuniformity, which will give a limitation on capacity.

References

[1] Y.S. Abu-Mostafa, "The Complexity of Information Extraction," *IEEE Trans. Info. Th.*, IT-32, p. 513, 1986.

[2] Y.S. Abu-Mostafa and D. Psaltis, "Computation Power of Parallelism in Optical Architectures," *IEEE Comp. Soc. Workshop on Comp. Arch. for Patt. Anal. and Im. Database Manag.*, Miami Beach, Fl, p. 42, Nov. 1985.

[3] Y.S. Abu-Mostafa and D. Psaltis, "Optical Neural Computers," *Scientific American*, 256(3), p. 88, 1987.

[4] J.A. Anderson, "Cognitive and Psychological Computation with Neural Models," *IEEE Trans. Sys., Man, and Cyber.*, SMC-13, p. 799, 1983.

[5] R.A. Athale, H.H. Szu, and C.B. Friedlander, "Optical Implementation of Associative Memory with Controlled Nonlinearity in the Correlation Domain," *Opt. Lett.*, 11(7), p. 482, 1986.

[6] P. Baldi, and S.S. Venkatesh, "Number of Stable Points for Spin-Glasses and Neural Networks of Higher Orders," *Phys. Rev. Lett.*, 58(9), p. 913, 1987.

[7] H.H. Chen, Y.C. Lee, T. Maxwell, G.Z. Sun, H.Y. Lee, and C.L. Giles, "High Order Correlation Model for Associative Memory," J. Denker, ed., *AIP Confer. Procs.*, 151, p. 86, Snowbird, UT, 1986.

[8] M. Cohen and S. Grossberg, "Absolute Stability of Global Pattern Formation and Parallel Memory Storage by Competitive Neural Networks," *IEEE Trans. Sys., Man, and Cyber.*, SMC-13, p. 815, 1983.

[9] R.J. Collier, C.B. Burkhardt, and L.H. Lin, *Optical Holography*, Academic Press, N-Y, 1971.

[10] T.M. Cover, "Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition," *IEEE Trans. Elec. Comp.*, EC-14, p. 326, 1965.

[11] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*, Wiley, N–Y, 1973.

[12] N.H. Fahrat, D. Psaltis, A. Prata, and E.G. Paek, "Optical Implementation of the Hopfield Model," *Appl. Opt.*, 24(10), p. 1469, 1985.

[13] C.L. Giles and T. Maxwell, "Learning and Generalization in Higher Order Networks," *Appl. Opt.*, 26(23), p. 4972, 1987.

[14] J.W. Goodman, R.A. Dias, and L.M. Woody, "Fully Parallel, High Speed Incoherent Optical Method for Performing Discrete Fourier Transforms," *Opt. Lett.*, 2(1), p. 1, 1978.

[15] J.W. Goodman, F.J. Frederick, J. Leonberger, S. Kung, and R.A. Athale, "Optical Interconnections for VLSI Systems," *Proc. IEEE*, 72(7), p. 850, 1984.

[16] J.W. Goodman, M.S. Song, "Performance Limitations of an Analog Method for Solving Simultaneous Linear Equations," *Appl. Opt.*, 21(3), p. 502, 1982.

[17] J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," *Proc. Natl. Acad. Sci. USA*, 79, p. 2554, 1982.

[18] J.J. Hopfield, "'Neural' Computation of Decisions in Optimization Problems," *Biol. Cyber.*, 52, p. 141, 1985.

[19] J.S. Judd, "Complexity of Connectionist Learning with Various Node Functions," COINS Technical Report 87-60, University of Massachusetts.

[20] H. Kogelnik, "Coupled Theory for Thick Hologram Gratings," *Bell Sys. Tech. Jour.*, 48, p. 2909, 1969.

[21] T. Kohonen, *Self Organization and Associative Memory*, Springer–Verlag, N–Y, 1984.

[22] B. Kosko, "Bidirectional Associative Memories," *IEEE Trans. Sys., Man, and Cyber.*, 18(1), p. 49, 1988.

[23] B.B. Mandelbrot, *The Fractal Geometry of Nature*, W.H. Freeman & Co., S–F, 1982.

[24] T. Maxwell, C.L. Giles, Y.C. Lee, and H.H. Chen, "Nonlinear Dynamics of Artificial Neural Systems," J. Denker, ed., *AIP Confer. Procs.*, 151, p. 299, Snowbird, UT, 1986.

[25] R.J. McEliece, E.C. Posner, E.R. Rodemich, and S.S Venkatesh, "The Capacity of the Hopfield Associative Memories," *IEEE Trans. Infor. Th.*, IT-33, p. 461, 1987.

[26] K. Nakano, "Association–A Model of Associative Memory," *IEEE Trans. Sys., Man, and Cyber.*, SMC-2(3), 1972.

[27] Y. Owechko, G.J. Dunning, E. Marom, and B.H. Soffer, "Holographic Associative Memory with Nonlinearities in the Correlation Domain," *Appl. Opt.*, 26(10), p. 1900, 1987.

[28] E.G. Paek, C.H. Park, F. Mok, and D. Psaltis, "Acoustooptic Image Correlators," *SPIE*, 638-05, p. 25, Orlando, FL, 1986.

[29] E.G. Paek and D. Psaltis, "Optical Associative Memory Using Fourier Transform Holograms," *Opt. Engr.*, 26(5), p. 428, 1987.

[30] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, N–Y, 1984.

[31] T. Poggio, "On Optimal Nonlinear Associative Recall," *Biol. Cyber.*, 19, p. 201, 1975.

[32] D. Psaltis, "Optical Realization of Neural Network Models," *IEEE Fall Joint Computer Conference*, Dallas, TX, p. 428, Nov. 1986.

[33] D. Psaltis and R.A. Athale, "High Accuracy Computation with Linear Analog Optical Systems: A Critical Study," *Appl. Opt.*, 25(18), p. 3071, 1986.

[34] D. Psaltis, D. Brady, and K. Wagner, "Adaptive Optical Networks Using Photorefractive Crystals," *Appl. Opt.*, 27(9), p. 1752, 1988.

[35] D. Psaltis and J. Hong, "Shift-Invariant Optical Associative Memories," *Opt. Engr.*, 26(1), p. 10, 1987.

[36] D. Psaltis and C.H. Park, "Nonlinear Discriminant Functions and Associative Memories," J. Denker, ed., *AIP Confer. Procs.*, 151, p. 370, Snowbird, UT, 1986.

[37] D. Psaltis, C.H. Park, and J. Hong, "Quadratic Optical Associative Memories," *JOSA A*, 3(13), p. 32, 1986.

[38] D. Psaltis, C.H. Park, and J. Hong, "Higher Order Associative Memories and their Optical Implementations," *Neural Networks*, 1(2), p. 149, 1988.

[39] D. Psaltis, J. Yu, X.G. Gu, and H. Lee, "Optical Neural Nets Implemented with Volume Holograms," *OSA Second Topical Meeting on Optical Computing*, Incline Village, NV, p. 129, Mar. 1987.

[40] T.J. Sejnowski, "High-Order Boltzmann Machines," J. Denker, ed., *AIP Confer. Procs.*, 151, p. 398, Snowbird, UT, 1986.

[41] T.J. Sejnowski and C.R. Rosenberg, "NETtalk: A Parallel Network that Learns to Read Aloud," Johns Hopkins Univ. Technical Report, JHU/EECS-86/01.

[42] D. Slepian, "A Class of Binary Signaling Alphabets," *Bell Sys. Tech. Jour.*, 35, p. 203, 1956.

[43] M. Spivak, *Differential Geometry I*, Publish or Perish, Inc., Houston, 1979.

[44] C.W. Stirk, S.M. Rovnyak, and R.A. Athale, "Effects of System Noise on an Optical Implementation of an Additive Lateral Inhibitory Network," *IEEE First Int. Conf. on Neural Networks*, III, San Diego, CA, p. 615, 1987.

[45] H.M. Tai, C.H. Wu, and T.L. Jong, "High-Order Bidirectional Associative Memories," *Electron. Lett.*, 25(1), p. 1424, 1989.

[46] P.J. Van Heerden, "Theory of Optical Information Storage in Solids," *Appl. Opt.*, 2(4), p. 393, 1963.

[47] S.S. Venkatesh, "Epsilon Capacity of Neural Networks," J. Denker, ed., *AIP Confer. Procs.*, 151, p. 440, Snowbird, UT, 1986.

[48] S.S. Venkatesh and D. Psaltis, "Linear and Logarithmic Capacities in Associative Memories," *IEEE Trans. Infor. Th.*, 35(3), p. 558, 1989.

[49] C. Warde and A.D. Fisher, "Spatial Light Modulators: Applications and Functional Capabilities," J.L. Horner, ed., *Optical Signal Processing*, p. 477, Academic Press, S-D, 1987.

[50] B. Widrow and M.E. Hoff, "Adaptive Switching Circuits," *IRE Wescon Conv. Rec.*, Pt. 4, p. 96, 1960.