A Neural Architecture for Content as well as Address-Based Storage and Recall: Theory and Applications

Chun-Hsien Chen & Vasant Honavar
Artificial Intelligence Research Group
Computer Science Department
Iowa State University
Ames, Iowa 50011-1040, U.S.A.
chen@cs.iastate.edu, honavar@cs.iastate.edu

September 26, 1995

Keywords: Associative memory, content-addressed memory, fault-tolerance, multiple associative recall, adjustable precision memory, binary mappings, information retrieval, address-based memory, high-capacity memory, interference.

Running Head: Neural Memories

Corresponding Author:

Vasant Honavar
Department of Computer Science
226 Atanasoff Hall
Iowa State University
Ames, Iowa 50011-1040. U.S.A.
phone: 515 294-1098
fax: 515 294-0258
e-mail: honavar@cs.iastate.edu

Abstract

This paper presents an approach to design of a neural architecture for both associative (content-addressed) and address-based memories. Several interesting properties of the memory module are mathematically analyzed in detail. When used as an associative memory, the proposed neural memory module supports recall from partial input patterns, (sequential) multiple recalls and fault tolerance. When used as an address-based memory, the memory module can provide working space for dynamic representations for symbol processing and shared message-passing among neural network modules within an integrated neural network system. It also provides for real-time update of memory contents by one-shot learning without interference with other stored patterns.

1 Introduction

Artificial neural networks (ANNs), due to their inherent parallelism and potential for fault tolerance, offer an attractive computational model for a variety of applications in pattern classification, language processing, complex systems modelling, control, optimization, prediction and reasoning.

It is generally agreed that ANNs have demonstrated success in *low-level* perceptual tasks (e.g., signal processing, pattern recognition) [Haykin, 1994; Kung, 1993; McKenna, 1994; Meyerowitz, 1991]. However, despite their generality (as computational models) and despite the potential advantages of using them as components in general–purpose artificial intelligence systems [Goonatilake and Khebbal, 1995; Honavar, 1994; Honavar and Uhr, 1994; 1995; Levine and Apariciov, 1994; Sun and Bookman, 1994], detailed design and performance tradeoffs in integrated systems of this sort are yet to be fully understood and working prototypes of such systems are only beginning to be developed. Towards this end, a careful analysis of neural memories with emphasis on problems and prospects of integrating them into larger systems that combine the advantages of both traditional symbol processing and neural network approaches to artificial intelligence is needed.

This paper explores a particular class of neural memories built from threshold logic units (Perceptrons or McCulloch-Pitts neurons) from a geometrical/mathematical perspective. This analysis provides a better understanding of several interesting properties of such memories including: fault-tolerance, auto as well as hetero-associative recall from partially specified patterns, (sequential) recall of multiple patterns with different degrees of match with a stored pattern, incremental learning; and address-based storage and recall (mimicking the behavior of memories used in conventional digital computers. The mathematical analysis also suggests efficient hardware realizations of such memories. The paper is organized as follows:

- Section 1 reviews address-based memory, content-addressable memory, and key properties of multi-layer Perceptrons which form the basis of the neural memories proposed in this paper.
- Section 2 develops the theoretical foundations and examines the storage capacity of the proposed binary/bipolar neural memories through an investigation of the spatial distribution and linear separability of vertices in binary/bipolar hypercubes from a geometric perspective.
- Section 3 explores several interesting properties of the proposed memory modules including: recall from partially specified input patterns, (sequential) multiple recalls, and fault tolerance by examining and extending the physical meanings of the settings of connection weights and neuron thresholds in the proposed neural memories.
- Section 4 concludes with a summary of the paper and a brief discussion of related researches.

1.1 Information Retrieval and Binary Mapping

In general, most classification and information retrieval problems using discrete input/output values can be viewed in terms of a binary random mapping f_I, where f_I is rigidly defined from a set U of k binary input vectors $u_1,...,u_k$ of dimension n to a set V of k binary output vectors $v_1,...,v_k$ of dimension m such that

$$f_I : U \rightarrow V$$
$$f_I(u_i) = v_i$$

Note that f_I is a partial function.

1.2 Content-addressed Memory (Associative Memory)

The term *associative memory* (AM) or *content-addressed memory* refers to a memory system where recall of a stored pattern is accomplished by providing a noisy or partially specified input pattern. Examples of such memory models include Hopfield networks [Hopfield, 1982], correlation matrix memories [Kohonen, 1972], bidirectional associative memories [Kosko, 1988], among others [Amari, 1977; Hao and Vandewalle, 1994, Nakano, 1972, Kumagai, 1993]. A precise definition of associative binary/bipolar memories follows:

Let $D_H(u, u')$ denote the Hamming distance between binary (bipolar) vectors u and u'. *Hamming distance* is the number of bits that differ between two binary (bipolar) vectors. Suppose we are given a set U of k binary input vectors $u_1,...,u_k$ of dimension n and a set V of k desired binary output vectors $v_1,...,v_k$ of dimension m. Then the task is to design an associative memory that can store each of the input-output pattern pairs.

In many applications, it is useful to be able to control the degree of mismatch that is tolerated during information retrieval. This is accomplished by introducing the concept of *precision control* in associative memory as follows: Define $U_i^n(p_i) = \{u | u \in \mathbf{B}^n \; \& \; D_H(u, u_i) \leq p_i\}$, $1 \leq i \leq k$, i.e., $U_i^n(p_i)$ is the set of n-dimensional binary vectors which have Hamming distance less than or equal to p_i away from the given n-dimensional binary vector u_i, where $\mathbf{B}^n$ is the *universe* of n-dimensional binary vectors, and p_i is called *allowable precision level* and is an adjustable integer parameter (see Section 2.2 for details).

Information retrieval in a binary associative memory can be specified in terms of a binary *associative mapping* f_A as follows:

$$f_A : U^n \rightarrow V$$
$$f_A(x) = v_i;\text{ if } x \in U_i^n(p_i),\ 1 \leq i \leq k$$

where $U^n = \cup_{i=1}^k U_i^n(p_i) = U_1^n(p_1) \cup U_2^n(p_2) \ldots \cup U_k^n(p_n)$ and conventionally $U_i^n(p_i) \cap U_j^n(p_j) = \emptyset$ for $i \neq j$, $1 \leq i, j \leq k$. For example, if such a memory is used to store and recall uppercase English characters, $U = V$ and $u_i = v_i$, $1 \leq i \leq 26$. Suppose the allowable precision levels (i.e., each of the p_is) are set equal to (Hamming distance) 4. Then in Figure 1, both the top two noisy input patterns would result in the recall of the stored memory pattern **T**. Multiple recalls are possible in the proposed neural memory when $\exists i \neq j$ such that $U_i^n(p_i) \cap U_j^n(p_j) \neq \emptyset$ in which case f_A is a *one-to-many* mapping. Most conventional associative memory models seldom tackle the problem of multiple recalls.

Note that $f_I \subseteq f_A$ if functions f_I (the binary random mapping f_I defined in Section 1.1) and f_A are viewed as sets of input-output ordered pairs of the functions f_I and f_A respectively. That is,

$$f_I \equiv \{(x, f_I(x)) | x \in U\}$$
$$f_A \equiv \{(x, f_A(x)) | x \in U^n\}$$

The partial function f_A may be extended to a full function $\hat{f}_A$ (information retrieval) for binary associative memory as follows:

$$\hat{f}_A : \mathbf{B}^n \rightarrow (V \cup \{<0^m>\})$$
$$\hat{f}_A(x) = \begin{cases} f_A(x) & \text{if } x \in U^n \\ <0^m> & \text{if } x \in (\mathbf{B}^n - U^n) \end{cases}$$

where $<0^m>$ is the m-dimensional binary vector of all zeros and denotes a value which is *undefined*.

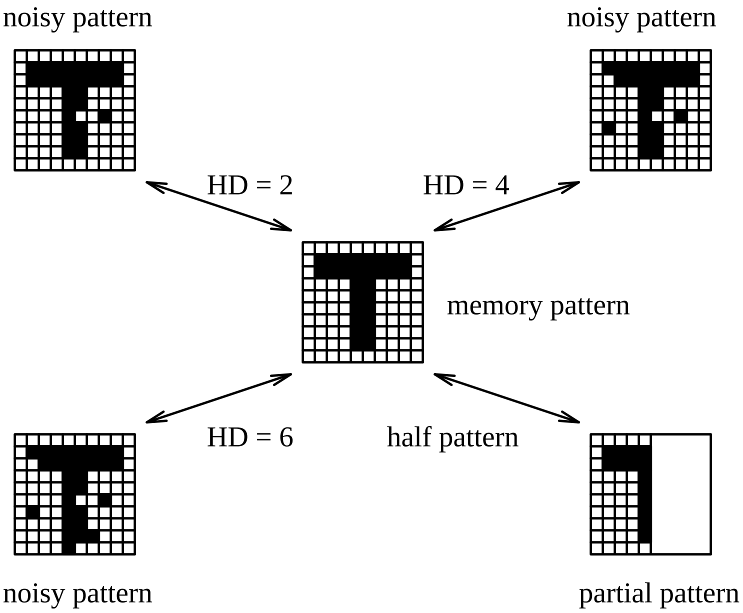

Figure 1. Examples of memory pattern, noisy patterns, and partial pattern, where HD denotes Hamming distance.

Content-addressed memories can be divided into two categories: *auto-associative memories* (used primarily for reconstructing a pattern from a noisy or partially specified pattern) and *hetero-associative* memories which can be used to store associated pattern pairs so that when an input pattern is provided, the associated pattern is retrieved. The types of pattern associations that can be stored depends on various factors such as: the choice of neural network architecture, the choice of functions computed by the neurons, and the algorithm used to set up the parameters (thresholds and weights) associated with the neurons and connections. Thus, a *lin-*

ear associative memory with n input neurons can store and recall perfectly at most n pattern associations. Similar storage capacity results are known for several content-addressed memory models such as the *Hopfield* network [Hopfield, 1982; McEliece et al., 1987], *bi-directional associative memories* [Kosko, 1988], correlation matrix memories [Kohonen, 1972], etc. The interested reader is referred to [Kung, 1993; Haykin, 1994] for discussion of a variety of associative memory models. As already pointed out, many simple content-addressed memory models studied in the literature are incapable of stable storage and recall of associations between arbitrary pairs of patterns (except under very restricted circumstances). In such models, whether a pattern can be associated with another critically depends on how the two patterns are coded as bit vectors as well as on all the other pattern associations that have already been stored in memory. The ability to reliably store and recall associations between arbitrary patterns is regarded by many to be a prerequisite for higher level cognitive activity (e.g., logical inference) [Fodor and Pylyshyn, 1988]. The associative memory model examined in this paper is designed to store and recall associations between arbitrary pairs of patterns.

1.3 Address-based Memory

Address-based memory is extensively used for storing both data as well as programs in current computer systems. In cognitive models and artificial intelligence programs based on the *Von Neumann* model of computation, i.e., models within the so–called symbolic paradigm [Newell 1980], address-based memory often serves as the working memory (or scratch-pad) for storing intermediate results during the execution of a program. On the surface, storage and recall of patterns using addresses appear to be very different in spirit from the recall of patterns based on their content (as judged by its similarity to a stored pattern). Indeed, many authors have suggested this to be a primary difference between neural networks (or connectionist models) and traditional artificial intelligence systems. However, this perceived difference is rather superficial given the demonstrable Turing-equivalence of sufficiently powerful neural network models [Honavar, 1994; Honavar and Uhr, 1995]. Therefore, it is rather straightforward to design neural memories capable of address-based storage and recall of patterns as the following discussion illustrates.

A mathematical model for information retrieval in address-based memory can be formulated in terms of a binary random mapping f_I that was defined in Section 1.1 by extending the partial function f_I to a full function $\hat{f}_I$ (information retrieval) for address-based memory as follows:

$$\hat{f}_I : \mathbf{B}^n \rightarrow (V \cup \{<0^m>\})$$
$$\hat{f}_I(x) = \begin{cases} f_I(x) & \text{if } x \in U \\ <0^m> & \text{if } x \in (\mathbf{B}^n - U) \end{cases}$$

$\hat{f}_I$ maps from the set of n-bit binary addresses to the set of m-bit binary values. The retrieved value (or content of a memory address) is undefined if no pattern has been stored at the corresponding address.

It is well known (in the literature on the design of memory systems for digital computers) that this approach to address-based memory design is not necessarily the most efficient for large address spaces. In this case, hierarchical memory organization using multiple levels of address decoding and multiple memory modules of the type specified above is a more practical alternative [Sloan, 1976].

1.4 Perceptrons

A 1-layer Perceptron has n input neurons, m output neurons and one layer of connection weights. The output y_i of output neuron i is given by $y_i = f_h(\sum_{j=1}^{n} w_{ij}x_j - \theta_i)$. w_{ij} denotes the weight on the link from input neuron j to output neuron i, θ_i is the threshold of output neuron i, x_j is input value at input neuron j, and f_h is *binary hardlimiter* function.

$$f_h(x) = \begin{cases} 1 & \text{if x} \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

It is well known that such a 1-layer Perceptron can implement only linearly separable functions from $\mathbf{R}^n$ to $\{0,1\}^m$ [Minsky, 1969]. We can see the connection weight vector $w_i =< w_{i1}, ..., w_{in} >^T$ as defining a linear hyperplane H_i which linearly separates all the n-dimensional vectors into two sets, where $[\cdot]^T$ denotes the *transpose* of a vector or a matrix.

A 2-layer Perceptron has one layer of k hidden neurons (and hence two layers of connection weights with each hidden neuron being connected to every input neuron as well as every output neuron). In this paper, we use 2-layer Perceptron in which each hidden neuron uses binary hardlimiter function f_h as activation function. The output of output neuron i is given by

$y_i = f(\sum_{l=1}^{k} w_{il} z_l - 1)$; where z_l is the output of hidden neuron l, f is binary hardlimiter function f_h in the model using binary output, and f is bipolar hardlimiter function $\breve{f}_h$ in the model using bipolar output. (The thresholds of all output neurons are set to 1). The *bipolar hardlimiter* function $\breve{f}_h$ is defined as

$$\breve{f}_h(x) = \begin{cases} 1 & \text{if x} \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

2 Multi-layer Perceptron as A Binary Memory

This section describes the synthesis of a binary address-based memory or a binary associative memory using a 2-layer network of simple Perceptrons described in Section 1. The binary address-based memory has a storage capacity of $N = 2^n$ while the binary associative memory has a storage capacity $N = \lfloor 2^n / \sum_{i=0}^{p} C(n,i) \rfloor$, where n is the number of input neurons and p is the adjustable precision level (allowable noise level) measured in terms of Hamming distance (as defined in Section 1.2). A hidden neuron is used for each stored memory pattern. The numbers of input and output neurons are fixed in these models. In the case of associative memory, this amounts to fixing the dimensionality of input and output patterns; while in the address based memory, it is tantamount to fixing the maximum size of the address space and the dimensionality of the patterns stored in memory.

2.1 Linear Separability of Binary Vertices and Its Application in Pattern Classification Using Perceptron

Note that every n-dimensional binary vector is a binary vertex of an n-dimensional hypercube. Hereafter, we will use the terms *binary vertex* and *binary vector* interchangeably.

Theorem 1: Let u be a binary vector of dimension n, i.e., $u =< u_1, ..., u_n >^T$ where $u_i \in \{0,1\}$ for $1 \leq i \leq n$. Let $\overline{u} =< \overline{u}_1, ..., \overline{u}_n >^T$ be the complement of the binary vector u. That is, $u_i + \overline{u}_i = 1$ for $1 \leq i \leq n$. Let $u - \overline{u} = u^{ref_u} =< u_1^{ref_u}, ..., u_n^{ref_u} >^T$. Note that $u_i^{ref_u} \in \{1, -1\}$ for $1 \leq i \leq n$. Let us call

u^{ref_u} the *reference vector*. Let S_p^u be the set of n-dimensional binary vertices which are at a Hamming distance p away from vertex u, $0 \leq p \leq n$. Then every binary vertex $x \in S_p^u$ falls on a hyperplane H_p^u which is perpendicular to the reference vector u^{ref_u}. Furthermore, if $H^u = \{H_p^u; 0 \leq p \leq n\}$, the hyperplanes in H^u are mutually parallel.

Theorem 1 is proved in Appendix A. The expression defining H_p^u in an n-dimensional binary hypercube is derived in Appendix B as follows :

$$H_p^u \equiv (\sum_{i=1}^{n}(2u_i - 1)x_i) - (\|u\|^2 - p) = 0$$

Every hyperplane H_p^u, where $0 \leq p \leq n$, can serve as a linear separating hyperplane to partition all n-dimensional binary vertices into two sets. Such a linear separating hyperplane can be efficiently implemented for 2-class pattern classification in a 1-layer Perceptron whose output neuron has a threshold of $\|u\|^2 - p$ and the connection weight on the link from input neuron i is given by $2u_i - 1$ for $1 \leq i \leq n$, where n is the number of input neurons. Note that $\|u\|^2$ equals the number of 1's in binary vector u.

Since each binary vertex of dimension n on hyperplane H_p^u is Hamming distance p away from vertex u, there are $N_p = C(n,p) = \frac{n!}{(n-p)!p!}$ such binary vertices of dimension n on hyperplane H_p^u, where $0 \leq p \leq n$. The *separating hyperplane* H_p^u partitions all the binary vertices of a binary n-hypercube into two sets. One set contains $N_A = \sum_{i=0}^{p} N_i = \sum_{i=0}^{p} C(n,i)$ binary vertices that are at a Hamming distance less than or equal to p away from vertex u, and the other contains $N_B = \sum_{i=p+1}^{n} N_i = \sum_{i=p+1}^{n} C(n,i)$ binary vertices that are at a Hamming distance more than p away from vertex u. Let us call the former partition the *associative partition* (denoted by α_p^u) of u, the vertex u the *center* of that associative partition, and p the *radius* of the associative partition. Note that both H_p^u and α_p^u are defined by the given binary memory vector u and its precision level p. When $x \in \alpha_p^u$ is fed into the 1-layer Perceptron, the output neuron that corresponds to the separating hyperplane H_p^u is activated to produce an output of 1.

In theory, an n-hypercube can be almost equally partitioned by $N = \lfloor 2^n / \sum_{i=0}^{p} C(n,i) \rfloor$ such associative partitions as α_p^u with each associative partition containing $\sum_{i=0}^{p} C(n,i)$ n-dimensional binary vertices which are all at a Hamming distance less than or equal to p away from their corresponding partition center. The partition centers correspond to the given binary patterns to be recognized.

When the separating hyperplane H_p^u is realized by a 1-layer Perceptron, the value of $2u_i - 1$ is either 1 or -1, x_i is either 1 or 0, and $(2u_i - 1)x_i$ can therefore be 1, 0 or -1; and $\|u\|^2$ is integer. Also note that the maximum activation of the 1-layer Perceptron is p and minimum value is $-(n-p)$; since the separating hyperplane H_p^u is defined as $(u - \overline{u})(x - \overline{u})^T - (n - p) = 0$, the maximum value of $(u - \overline{u})(x - \overline{u})^T$ is n when $x = u$, and the minimum value of $(u - \overline{u})(x - \overline{u})^T$ is 0 when $x = \overline{u}$.

2.2 Recall with Adjustable Precision Using Associative Partitions

We say that an associative partition $\alpha_{p_i}^{\nu_i}$ is not *isolated* from another associative partition $\alpha_{p_j}^{\nu_j}$ $(i \neq j)$ if $\alpha_{p_i}^{\nu_i} \cap \alpha_{p_j}^{\nu_j} \neq \emptyset$. Thus, if two associative partitions are not isolated from each other, they overlap and as a result, there is at least one binary vector that is a member of both partitions. The separating hyperplanes (or equivalently, associative partitions) can be implemented in a 1-layer Perceptron with N output neurons to recognize N patterns with precision level (allowable noise level) up to Hamming distance p_i for memory pattern ν_i, $1 \leq i \leq N$, provided the precision levels (p_is) are chosen to ensure that each associative partition is *isolated* from every other. In order to ensure that the associative partitions corresponding to two memory patterns ν_i and ν_j are isolated from each other, $D_H(\nu_i, \nu_j)$ has to be greater than $(p_i + p_j)$ where p_i and p_j are the allowable precision levels. Otherwise, the associative partitions of ν_i and ν_j would overlap with each other, and when an input pattern x, where $D_H(x, \nu_i) \leq p_i$ and $D_H(x, \nu_j) \leq p_j$, is fed into the 1-layer Perceptron, the output neurons for both the memory patterns ν_i and ν_j will produce 1 as their outputs. In this case, the input pattern x cannot be unambiguously classified as it falls in the region of overlap between the associative partitions $\alpha_{p_i}^{\nu_i}$ and $\alpha_{p_j}^{\nu_j}$.

2.3 Capacity for Recognition of Noisy Digitized Printed Patterns

Suppose input patterns are 10×10 arrays of binary pixels (see Figure 1). Then 100 input neurons are required to implement such a 1-layer Perceptron. The number of possible input patterns is $2^{100} \approx 10^{30}$. An output neuron is needed for each distinct pattern to be stored. Table 1 shows the corresponding maximal storage capacity of the 1-layer Perceptrons designed for a range of different allowable noise levels. Table 1 also suggests that a 1-layer Perceptron with n input neurons has very high storage capacity for efficiently recognizing digitized binary printed patterns and that the allowable precision (noise) levels of less than 30% are desirable for reliable storage and recall.

allowable noise	maximal capacity
0%	$N = \lfloor 2^{100} / \sum_{i=0}^{100\times 0} C(100,i) \rfloor \approx 1.0 \times 10^{30}$
10%	$N = \lfloor 2^{100} / \sum_{i=0}^{100\times 0.1} C(100,i) \rfloor \approx 5.0 \times 10^{16}$
20%	$N = \lfloor 2^{100} / \sum_{i=0}^{100\times 0.2} C(100,i) \rfloor \approx 1.4 \times 10^{9}$
30%	$N = \lfloor 2^{100} / \sum_{i=0}^{100\times 0.3} C(100,i) \rfloor \approx 2.4 \times 10^{4}$
40%	$N = \lfloor 2^{100} / \sum_{i=0}^{100\times 0.4} C(100,i) \rfloor \approx 38$
50%	$N = 2$

Table 1 shows the corresponding maximal storage capacity of a 1-layer Perceptron with 100 input neurons for recognizing digitized binary printed patterns for a range of allowable noise levels.

2.4 Synthesis of Associative and Address-Based Memories Using a 2-layer Perceptron

Given a set U of k distinct binary input vectors $u_1,...,u_k$ of dimension n, where $u_i =< u_{i1}, ..., u_{in} >^T$ and $u_{ig} \in \{0,1\}$ for $1 \leq i \leq k$ & $1 \leq g \leq n$; and a set V of k desired binary output vectors $v_1,...,v_k$ of dimension m, where $v_i =< v_{i1}, ..., v_{im} >^T$ and $v_{ih} \in \{0,1\}$ for $1 \leq i \leq k$ & $1 \leq h \leq m$. Assume the Hamming distance between any two binary vectors in U is at least $2p+1$, where $p \in \mathbf{N}$. This ensures that all associative partitions would be isolated with the precision level being set at p.

We can now design a neural architecture for information retrieval using address-based memory, denoted by function $\hat{f}_I$, or associative memory, denoted by function $\hat{f}_A$. For this purpose, a memory module of 2-layer Perceptron (see figure 2) can be synthesized using the 1-layer Perceptrons, defined in Section 2.1, as follows:

The *memory module* has n input, k hidden, and m output neurons. For each associative ordered pair (u_i, v_i), where $1 \leq i \leq k$, we create a hidden neuron i with threshold $\|u_i\|^2 - p_i$, where $p_i \in \mathbf{N}$ and $p_i \leq p$ is the adjustable precision level for that associative pair. The connection weight from input neuron g to hidden neuron i is $2u_{ig} - 1$ $(= u_{ig} - \overline{u}_{ig})$ and that from hidden neuron i to output neuron h is v_{ih}. The threshold for each of the output neurons is set to 1. The activation function at hidden neuron is binary hardlimiter function f_h, defined in Section 1.4, and that at output neuron can be binary hardlimiter function f_h or identity function I, i.e., $I(x) = x$, depending on the particular hardware implementation employed.

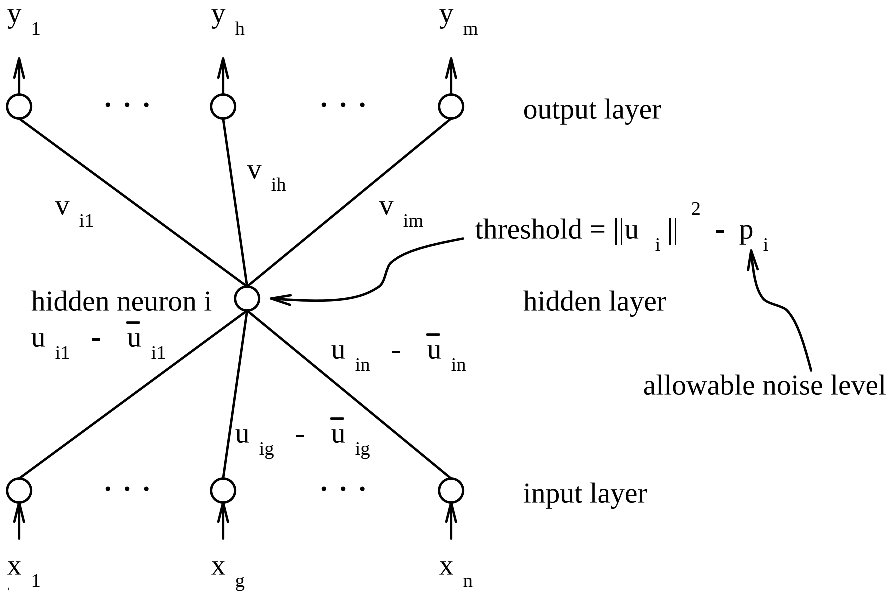

Figure 2 shows the setting of connection weights and hidden node threshold in the proposed 2-layer Perceptron for an associated binary memory pair (u_i, v_i) with adjustable precision level up to Hamming distance p_i. For an address-based memory, p_i is set as 0.

Since all the associative partitions are isolated from each other, when the memory module is presented with a binary input vector $x \in \alpha_{p_i}^{u_i}$, only the hidden neuron i produces an output of 1 and the output values from all other hidden neurons produce 0. So the value at output neuron j is v_{ij}, and hence the output binary vector will be $< v_{i1}, ..., v_{im} >^T = v_i$. Since for each memory association pair a hidden neuron is created and its creation or deletion is independent of other stored memory association pairs, this particular design of associative memory lends itself to rapid *one-shot incremental learning* with no interference with previously stored associations.

It is also worth pointing out that exactly the same network architecture can be used to realize both associative as well as address-based memory. If p_i is set as 0, $\mid \alpha_{p_i}^{u_i} \mid = 1$ and the memory module functions as an address-based memory when 2^n hidden neurons are used to resolve all possible addresses; and if $1 \leq p_i \leq p$, $\mid \alpha_{p_i}^{u_i} \mid > 1$ and it can be used as an adjustable-precision content-addressable or associative memory. ($\mid A \mid$ denotes the cardinality of a set A). Address-based memory, extensively used in current computer system, can serve as working space of dynamic representations for symbol processing and shared message-passing space among neural network modules in an integrated neural network system. As working space for symbol manipulation, neural memories have to allow run-time update without learning and do not degrade when the number of stored memory patterns increases. Note that the proposed neural address-based memory has these two properties.

2.5 Conversion Between Bipolar and Binary Memory Modules

Much of the analysis in the previous subsections assumed binary input patterns. It turns out that the use of *bipolar* instead of *binary* input patterns simplifies the implementation of the associative memory design described in Section 2.4 — especially when recall from partially specified input patterns is desired (see Section 3.1 and 3.2 for details). This subsection explores the relationship between memory models using binary and bipolar input patterns and the conversion between the two.

The spatial distribution and geometrical characteristics of bipolar vertices in a bipolar hypercube is very similar to those of binary vertices in a binary hypercube, except that the distance between any two bipolar vertices is 2 times of that between their corresponding binary vertices. Given a bipolar vertex u, there also exist n+1 mutually parallel linear hyperplanes which have similar features described in Theorem 1. The expression defining H_p^u in an n-dimensional bipolar hypercube is derived in Appendix C as follows :

$$H_p^u \equiv (\sum_{i=1}^{n} u_i x_i) - (n - 2p) = 0$$

Every hyperplane H_p^u, where $0 \leq p \leq n$, can serve as a linear separating hyperplane to partition all n-dimensional bipolar vertices into two sets. Such a linear separating hyperplane can be efficiently implemented for pattern classification in a 1-layer Perceptron whose output neuron has a threshold of $n - 2p$ and the connection weight on the link from input neuron i is given by u_i for $1 \leq i \leq n$, where n is the number of input neurons.

Since input is bipolar, the connection weight in the 1-layer Perceptron is either 1 or -1. The connection weight of 1 matches the corresponding bit of an input pattern if it is *on* while a connection weight of -1 matches the corresponding bit of an input pattern if it is *off*. A match contributes 1 unit to the activation of the corresponding hidden neuron while a mismatch contributes -1 unit. Each hidden neuron sums up the contributions to its activation from each of its input links, compares it with its threshold and activates the corresponding output neuron if the degree of match (similarity measurement) for the entire pattern exceeds or equals the threshold.

Note that the value passed from each connection is either 1 or -1 , compared to the three values $\{1, 0, -1\}$ in the binary model (see Section 2.1). This property can further simplify the hardware implementation requirement for a 1-layer Perceptron using bipolar (as opposed to binary) input.

Based on this 1-layer Perceptron and the method described in Section 2.4 for setting the weights of the second layer connections, the synthesis of a memory module of 2-layer Perceptron is rather straightforward given a set of desired pattern associations. It is worth pointing out that the bipolar associative neural memory model derived here turns out to be exactly equivalent to the memory model proposed by [Hao and Vandewalle, 1994] which uses real-value neuron thresholds and proves the effectiveness of the bipolar memory model by algebra based on using Hamming distance as difference measurement between input pattern and memory patterns. In this paper, the spatial distribution and linear separability of bipolar vertices in a bipolar hypercube is examined from a geometrical perspective to locate a set of mutually parallel linear hyperplanes which respectively separate nicely all the bipolar vertices into two sets to facilitate the application of bipolar neural memories. The linear separating hyperplanes can be efficiently implemented in a 1-layer Perceptron with connection weights and neuron thresholds of integer values.

Some notable differences between the binary and bipolar associative memory models developed above are:

- The binary model uses binary hardlimiter as the activation function at both hidden and output neurons, so does the bipolar model if the associated output is binary. If the associated output is bipolar, binary and bipolar hardlimiters (respectively) are used as activation functions at hidden and output neurons.

- Threshold setting for a hidden neuron in the binary model equals the number of *on* bits of the corresponding memory pattern minus the desired precision level (measured in Hamming distance), which is not independent of the corresponding memory pattern; whereas threshold setting for a hidden neuron in the bipolar model equals the number of input neurons minus twice the value of the desired precision level, which is independent of all the memory patterns. This has a special advantage when the associative memory is used to recall a pattern based on a partially specified input (see Section 3.1 for details).

3 Properties of the Proposed Neural Associative Memory

The following three subsections explore and develop mathematical models for several interesting properties of the proposed bipolar neural associative memory including: recall from partially specified patterns, (sequential) multiple recalls, and fault tolerance. A set U of k bipolar input vectors $u_1,...,u_k$ of dimension n and a set V of k desired binary/bipolar output vectors $v_1,...,v_k$ of dimension m are given. In the discussion that follows, we will assume that m-dimensional null pattern (a vector of 0s in the binary case or a vector of -1s in the bipolar case) is excluded from V.

3.1 Associative Recall from a Partially Specified Input Pattern

This section examines the problem of recall from a partially specified bipolar input pattern. The analysis that follows assumes that the unavailable components of an input pattern vector have a default value of 0. Thus, a partial input pattern is *completed* by filling in a 0 for each of the unavailable components (the 0s as a whole also can serve as noise mask or filter). This makes it possible to handle the problem of associative recall from partially specified input pattern in a manner that is analogous to that of recall from completely specified input pattern. The 1st-layer connections of the neural memory module perform similarity measurements on the available components of a partial pattern, ignore the similarity measurements on the unavailable components, and pass the similarity measurements to the corresponding hidden neurons to decide whether to activate a corresponding hidden neuron.

Let $\dot{u}$ be a partially specified n-dimensional bipolar pattern with the values of some of the components being unknown. Define

- $bits(\dot{u})$: a function which counts the number of components with known values (+1 or -1) of partial pattern $\dot{u}$
- $pad0(\dot{u})$: a function which pads the unavailable bits of bipolar partial pattern $\dot{u}$ with 0's.
- $u \odot v$: a binary predicate which tests whether "*u is a partial pattern of v*", where "*u is a partial pattern of v*" means that the values of available components of u are same as those of their corresponding components in v.

For example, let $\dot{u} = <?, -1, 1, 1, ?>$ be a 5-dimensional partial pattern whose first and fifth components have unspecified values. Then $bits(\dot{u}) = 3$, $pad0(\dot{u}) = <0, -1, 1, 1, 0>$ and $\dot{u} \odot <1, -1, 1, 1, 1>$ is *true*. (Note that this definition of a partial pattern respects the positions of the components and does not accommodate shifts or translation).

Let $\dot{D}_H(\dot{u}, \dot{v})$ denotes the Hamming distance between two bipolar partial patterns, with same corresponding unavailable components, $\dot{u}$ and $\dot{v}$ respectively. If $bits(\dot{u}) = j$, $pad0(\dot{u})$ is a *padded* j-bit partial pattern derived from partially specified pattern $\dot{u}$. Define $\dot{U}^j_{Pi} = \{\dot{u} \mid bits(\dot{u}) = j \ \& \ \dot{u} \odot u_i\}$, $1 \leq j \leq n \ \& \ 1 \leq i \leq k$, i.e., $\dot{U}^j_{Pi}$ is the set of partial patterns, with j available bits, of a bipolar pattern u_i. Define $\ddot{U}^j_{Pi}(p_i) = \{pad0(\ddot{u}) \mid \exists \dot{u},\ \dot{u} \in \dot{U}^j_{Pi} \ \& \ \dot{D}_H(\ddot{u}, \dot{u}) \leq \lfloor j/n \rfloor \times p_i\}$, $1 \leq j \leq n \ \& \ 1 \leq i \leq k$, i.e., $\ddot{U}^j_{Pi}(p_i)$ is the set of padded j-bit partial patterns which are at Hamming distance less than or equal to $\lfloor j/n \rfloor \times p_i$ to any one of the padded j-bit partial patterns of full pattern u_i.

Practical applications may place limits on the range of usable settings of p_i (allowable noise level) (see Section 2.3 for details). It may also be necessary to limit recall of partial patterns to cases in which a sufficiently large number of bits in the pattern say $j \geq c$, have known values. For instance, when dealing with patterns of 10×10 pixels, we may set $p_i = 0.3 \times 100 = 30$ and require that at least 40% of the pixels be available in the input pattern.

To simplify matters in what follows, we use the same precision level p for each stored pattern. That is, $p_i = p, 1 \leq i \leq k$. However, note that particular applications may require the use of different values of p_i under different circumstances. For example, punctuation symbols and letters of the alphabet may need different values of p_i for successful recognition in recognizing printed English characters.

Let $\ddot{U}_{Pi}^{c\sim n}(p) = \cup_{j=c}^{n} \ddot{U}_{Pi}^{j}(p)$; and $\ddot{U}_{P}^{c\sim n}(p) = \cup_{i=1}^{k} \ddot{U}_{Pi}^{c\sim n}(p)$. Let f_P denote the function of recall from padded bipolar partial pattern. Then f_P is defined as follows:

$$f_P : \ddot{U}_{P}^{c\sim n}(p) \rightarrow V$$
$$f_P(x) = v_i;\ \text{if } x \in \ddot{U}_{Pi}^{c\sim n}(p),\ 1 \leq i \leq k$$

f_P is a partial function and is extended to a full function $\hat{f}_P$ for recall from padded bipolar partial pattern using associative memory as follows:

$$\hat{f}_P : \ddot{\mathbf{B}}^{c\sim n} \rightarrow (V \cup \{<(-1)^m>\})$$
$$\hat{f}_P(x) = \begin{cases} f_P(x) & \text{if } x \in \ddot{U}_{P}^{c\sim n}(p) \\ <(-1)^m> & \text{if } x \in (\ddot{\mathbf{B}}^{c\sim n} - \ddot{U}_{P}^{c\sim n}(p)) \end{cases}$$

where $\ddot{\mathbf{B}}^{c\sim n}$ is the *universe* of n-dimensional vectors each of whose components is 1, 0, or −1 and which have at least c non-zero components (corresponding to the available bits) and thus at most $(n - c)$ zeros for the unavailable bits (as a result of padding).

It is easy to see that if the 1st-layer connection weights in the bipolar neural memory (described in Section 2.5) were set up using only a part of a memory pattern, the connection weights set for the available components of the partial pattern would be same as those that would have been obtained if the complete pattern were used in establishing the weights. The threshold setting for a hidden neuron in the bipolar memory equals the number of input neurons minus twice the value of the desired precision level, which is independent of all stored memory patterns but depends on the dimensionality of input pattern. Hence the bipolar neural memory module designed for recall from a fully specified input pattern can be used for associative recall from a partially specified input pattern by only adjusting the thresholds of the hidden neurons as follows: *Multiply the threshold of each hidden neuron by the ratio of the number of available components of a partial input pattern* *to that of a complete pattern. That is, reduce the threshold of each hidden neuron i from $(n - 2p_i)$ to $(n - 2p_i) \times n_a/n$, where $n_a \leq n$ is the number of the available bits of a partial input pattern.*

Note that p_i is the precision level for memory pattern i in the problem of recall from full pattern and $(n - 2p_i) \times n_a/n = n_a - 2(p_i \times n_a/n)$ is the new threshold for recall from a partial pattern. The expression for the new threshold is similar to that for old threshold. In the new threshold n_a equals the number of available bits of a partial input pattern and $p_i \times n_a/n$ is the new precision level. In the interest of efficiency of a hardware realization, it is desirable to use $\lceil p_i \times n_a/n \rceil$ or $\lfloor p_i \times n_a/n \rfloor$ as the new precision level.

3.2 Multiple Associative Recalls

The memory retrieval process in the neural memory described in Section 2 can be viewed as a two-stage process: *identification* and *recall*. During identification of an input pattern, the 1st-layer connections perform similarity measurements and sufficiently activate zero or more hidden neurons so that they produce outputs of 1. The actual choice of hidden neurons to be turned on is a function of the 1st-layer weights, the input pattern, and the threshold settings of the hidden neurons. During recall, if only one hidden neuron is turned on, one of the stored memory patterns will be *recalled* by that hidden neuron along with its associated 2nd-layer connections. Without any additional control, if multiple hidden neurons are enabled, the corresponding output pattern will be a superposition of the output patterns associated with each of the activated hidden neurons. With the addition of appropriate control circuitry, this behavior can be modified to yield sequential recall of more than one stored pattern. This has a number of practical applications such as information retrieval from databases, knowledge-based diagnosis systems, etc. (The interested reader is referred to [Chen and Honavar, 1995b] for a detailed discussion of a neural architecture for information retrieval from databases). This has the effect of searching through memory for patterns that are sufficiently close to a given input pattern and then recall them one after another.

Multiple recalls are possible if some of the associative partitions realized in the memory module are *not isolated* (see Section 2.2 for details). An input pattern (a bipolar vertex) located in a region of overlap among several partitions is close enough to the corresponding partition centers (stored memory patterns) at the same time and hence can turn on more than one hidden neuron. The following explores this phenomenon in more detail.

Define $\dot{U}_i^n(p_i) = \{u \mid u \in \dot{\mathbf{B}}^n \ \& \ D_H(u, u_i) \leq p_i\}$, $1 \leq i \leq k$, where $\dot{\mathbf{B}}^n$ is the universe of n-dimensional bipolar vectors; i.e., $\dot{U}_i^n(p_i)$ to be the set of n-dimensional bipolar vectors which have Hamming distance less than or equal to p_i away from the given n-dimensional bipolar vector u_i, where p_i is a specified precision level. Let $p_i = p, 1 \leq i \leq k$.

Define f_M as follows:

$$f_M : \dot{U}^n(p) \rightarrow (2^V - \emptyset)$$
$$f_M(x) = \{v_i \mid x \in \dot{U}_i^n(p),\ 1 \leq i \leq k\}$$

where $\dot{U}^n(p) = \dot{U}_1^n(p) \cup \dot{U}_2^n(p) \ldots \cup \dot{U}_k^n(p)$, $\dot{U}_i(p) \cap \dot{U}_j(p) \neq \emptyset$ for some $i \neq j$, and 2^V is the *power set* of V (i.e., the set of all subsets of V). The output of f_M is a set of bipolar vectors that correspond to the set of patterns that should be recalled given the bipolar input vector x.

f_M is a partial function and is extended to a full function $\hat{f}_M$ to describe multiple recall in the associative ANN memory as follows:

$$\hat{f}_M : \dot{\mathbf{B}}^n \rightarrow (2^V \cup \{<(-1)^m>\} - \emptyset)$$
$$\hat{f}_M(x) = \begin{cases} f_M(x) & \text{if } x \in \dot{U}^n(p) \\ \{<(-1)^m>\} & \text{if } x \in (\dot{\mathbf{B}}^n - \dot{U}^n(p)) \end{cases}$$

Recall of multiple patterns is likely to be all the more useful when the input pattern is only partially specified. The following extends the mathematical model for multiple recall outlined above to deal with recall from a partially specified bipolar input pattern.

Let f_{MP} be a function defined as follows:

$$f_{MP} : \ddot{U}_P^{c\sim n}(p) \rightarrow (2^V - \emptyset)$$
$$f_{MP}(x) = \{v_i \mid x \in \ddot{U}_{Pi}^{c\sim n}(p),\ 1 \leq i \leq k\}$$

where $\dot{U}_{Pi}^h(p) \cap \dot{U}_{Pj}^h(p) \neq \emptyset$ for some h's and $i \neq j$'s, $c \leq h \leq n$ and $1 \leq i,\ j \leq k$.

f_{MP} is a partial function and is extended to a full function $\hat{f}_{MP}$ for multiple recalls from padded bipolar partial patterns in associative memory as follows:

$$\hat{f}_{MP} : \ddot{\mathbf{B}}^{c\sim n} \rightarrow (2^V \cup \{<(-1)^m>\} - \emptyset)$$
$$\hat{f}_{MP}(x) = \begin{cases} f_{MP}(x) & \text{if } x \in \ddot{U}_P^{c\sim n}(p) \\ <(-1)^m> & \text{if } x \in (\ddot{\mathbf{B}}^{c\sim n} - \ddot{U}_P^{c\sim n}(p)) \end{cases}$$

Another interesting property of the proposed ANN memory is that it allows *sorted multiple recall* described as follows: If the input pattern is held constant and the thresholds of all hidden neurons are decremented at each time step, then gradually more and more hidden neurons will be turned on. Decrementing the threshold of a hidden neuron results in an enlargement of the corresponding associative partition in a geometrical sense, and hence more and more partitions will overlap at the input vector (vertex) from iteration to iteration. In the absence of any other control circuits, the recalled pattern will be a superposition of the outputs resulting from all of the hidden neurons that are enabled at any time step. However, in many applications, we need different patterns to be recalled individually. This can be accomplished by adding a habituation mechanism that forces a hidden neuron to turn itself off automatically after it has been on for one time step unless a new input pattern is presented. This results in a serialized or sequential recall of patterns in increasing order of dissimilarity (as measured by the Hamming distance) from the input pattern. Alternatively, one can perform a *set difference* operation on the hidden neuron outputs from every pair of consecutive time steps before allowing the hidden neurons to influence the 2nd-layer connections and the output neurons. It is rather straightforward to implement such set-theoretic operations using neural networks [Chen and Honavar, 1995a].

As already pointed out, the ability to perform multiple recalls is more likely to be useful when dealing with partially specified input patterns (see Section 3.1 for details). Such information retrieval applications of practical interest include: database lookup using keywords [Chen and Honavar, 1995b], diagnosis of diseases or faults from a partially specified set of symptoms or test results, and DNA sequence recognition from available DNA segments.

3.3 Fault Tolerance

This section discusses the performance of the neural memory developed in Sections 2 and 3 in the presence of two basic types of faults — *neuron fault* and *connection fault*. In the discussion that follows, it is assumed that:

- When a connection fails and stops passing a value, it is assumed that *default value 0* is passed from that faulty connection.
- When an input or hidden neuron fails and stops functioning, it is assumed that *default value 0* is passed along each of its outgoing connections.

- When an output neuron fails, it is assumed that *default value* -1 (or 0 in binary model) is produced by that output neuron.

First we note that a single fault in a 1st-layer connection has less deleterious effect on the performance of the memory than that caused by a noisy bit in an input pattern. This is because a faulty 1st-layer connection will adversely affect only one of the similarity measurements between the input pattern and the stored memory patterns whereas a noisy bit of an input pattern affects all the similarity measurements.

For example, if $u_i = <1,-1,1,-1,1>^T$ and $u_j = <1,1,1,1,1>^T$ are two of the memory patterns stored by hidden neurons i, j and their respective connections in an auto-associative memory. Note that $D_H(u_i, u_j) = 2$. Let w_i^1 denote the weight vector of the 1st-layer connections connected to hidden neuron i. Suppose the precision levels $p_i = p_j = 1$. Then $w_i^1 = < 1,-1,1,-1,1>^T$, $w_j^1 = <1,1,1,1,1>^T$, and the thresholds at hidden neurons i and j are $\theta_i = \theta_j = 3$ in the neural memory according to expression (27) (see Appendix C). Suppose a noisy input pattern $x = < 1,-1,1,1,1 >^T$ is fed into the neural memory module. Then both hidden neurons i and j are activated and as a result, the output is a superposition of memory patterns u_i and u_j. Suppose the connection from hidden neuron i to the second input neuron is faulty, which causes $w_i^1 = <1,0,1,-1,1>^T$ equivalently. Assumes that w_j^1 is unaffected by the connection fault. When the noisy input pattern x is fed into the neural memory module, the summation values at hidden neuron i and j are -1 and 0 respectively. Only hidden neuron j is activated and memory pattern u_j is recalled.

Since each of the 2nd-layer connections emanating from a hidden unit stores 1 bit of the corresponding stored memory pattern, a faulty 2nd-layer connection corrupts at most 1 bit of the recalled memory pattern. Suppose the connection from hidden neuron j to the third output neuron is faulty. When only hidden neuron j is activated to recall memory pattern u_j, a default value 0 is passed from that faulty connection to the third output neuron under the assumption of connection fault. The recalled output is $<1,1,-1,1,1>^T$ which is one Hamming distance from memory pattern u_j. When only hidden neuron i is activated to recall memory pattern u_i with a connection fault from hidden neuron i to the second output neuron, the recalled output is $<1,-1,1,-1,1>^T$ which equals u_i

A fault in one of the input neurons has less of an adverse effect on the performance of the memory than 1 bit of noise in the input pattern. However, it is easy to see that an input neuron fault is more serious than a fault in a single 1st-layer connection. This is because a fault in one of the input neuron adversely affects each of the stored memory patterns. For example, suppose u_i, u_j and x are as before. Let us consider following three cases:

- **case 1**: one of the first, third, or fifth input neurons is faulty. When x is fed into the neural memory module, the summation values at hidden neurons i and j are both -1. No memory pattern is recalled.
- **case 2**: the second input neuron is faulty. Then the summation values at hidden neuron i and j are -1 and 1 respectively. The hidden neuron j is activated to recall memory pattern u_j.
- **case 3**: the fourth input neuron is faulty. Then the summation values at hidden neuron i and j are 1 and -1 respectively. The hidden neuron i is activated to recall memory pattern u_i.

If a hidden neuron is faulty, the memory pattern associated with that hidden neuron can not be recalled. If an output neuron is faulty, the corresponding bit of all recalled memory patterns will have value -1. So the recalled memory patterns having value 1 at that corresponding bit are corrupted by one bit of noise.

4 Summary and Discussion

This paper has discussed the analysis and synthesis of a neural memory for both address-based as well as associative (content-based) storage and recall of patterns. When used as content-addressed memory, the proposed ANN memory allows adjustable precision and sorted extraction of all stored memory patterns, has high potential storage capacity, and exhibits several interesting properties: recall from partial pattern, multiple recall and fault tolerance. It also lends itself to one-shot incremental learning without interference with previously memorized patterns. A detailed mathematical analysis of the properties of the proposed neural memory architecture is presented. Address-based memory can serve as working space of dynamic representations for symbol processing and shared message-passing space among neural network modules in an integrated neural network system. It provides for reliable content modification in real time, a necessary feature for symbol processing applications.

Memory retrieval can be viewed as a process of table lookup which is used in a wide variety of computing applications. Examples of such lookup tables include: *key-based organizations* used in record search of relational databases, *routing tables* used in routing of messages in communication networks, *symbol tables* used in compiling computer programs written in high level languages, and *lexicons* or dictionaries used in natural language processing. Usually, such applications build key-based organizations including *hashing, index-sequential access structure* and *B-trees* to allow quickly locating a desired data item using the memory (or disk) of Von Neumann computer.

The pattern matching and retrieval process of the proposed content-addressed memory in which all memory patterns are compared with input pattern in parallel within one step can be far more efficient (in terms of computation time) than searching for data in a key-based organization of the sort used in conventional computer systems [Chen and Honavar, 1995b]. With the need for real-time response in language translation and with the increased number of users as well as increased use of large networked databases over the Internet, efficient architectures for high-speed database lookup, message routing and query processing have assumed great practical significance. Extensions of the proposed ANN memory architecture to efficiently handle database queries and syntax analysis are proposed in [Chen and Honavar, 1995b; 1995c]. Further geometrical and mathematical analysis of binary mappings to facilitate the design of neural architectures for more complex symbol processing tasks such as structured pattern matching and reasoning (e.g., parallel inference) is currently in progress.

It is worth mentioning that the proposed neural memory supports realization many-to-one binary random mappings (see Section 1 & 2) which is extensively used in the design of in digital logic devices such as `AND/OR` (or `NAND/NOR`) gates. The design and optimization of such circuitry has been always one of the key research problems for the VLSI research community and industry. The hardware realization of the proposed neural architecture provides same flexibility as `PLA` (programmable logic array) and thus higher abstraction level than `AND/OR` gates for logic functions (many-to-one binary mappings). The interested reader is refered to [Chen and Honavar, 1995b] for an evaluation of anticipated performance of hardware realizations of the proposed memory architecture. If the hardware realization provides for run-time loading of connection weights and neuron thresholds under software control, it provides for an efficient, time-saving, and error-preventing alternative to the implementation of `PLA` and combinational circuitry of `AND/OR` gates for logic circuitry.

Acknowledgements

The authors would like to acknowledge the support of the National Science Foundation (through the grant IRI-9409580) and the Iowa State University College of Liberal Arts and Sciences during this research. They would also like to thank the members of Artificial Intelligence Research Group at Iowa State University as well as the anonymous referees for their constructive feedback on earlier drafts of the paper.

Appendix

Appendix A

Appendix A proves Theorem 1.

Proof:

Let x be a binary vertex in S_p^u, $x - \overline{u} = x^{ref_u} =< x_1^{ref_u}, ..., x_n^{ref_u} >^T$ and l_x^u be the length of the projection of x^{ref_u} onto the reference vector u^{ref_u}. Note that $x_i^{ref_u} = 0$ or 1 if $u_i^{ref_u} = 1$, and $x_i^{ref_u} = 0$ or -1 if $u_i^{ref_u} = -1$ for $1 \leq i \leq n$. Note also that there are p components $x_i^{ref_u}$ of x^{ref_u} such that $x_i^{ref_u} = 0$ and $(n-p)$ components $x_j^{ref_u}$ of x^{ref_u} such that $x_j^{ref_u} = 1 \; or -1$, where $1 \leq i,\ j \leq n$. Let $\| \cdot \|$ denote the length of a vector. Then

$$l_x^u = \frac{1}{\|u^{ref_u}\|}(u^{ref_u})^T x^{ref_u} \quad (1)$$

$$= \frac{1}{\|u^{ref_u}\|}\sum_{i=1}^{n} u_i^{ref_u} x_i^{ref_u} \quad (2)$$

$$= \frac{1}{\|u^{ref_u}\|}\left(\sum_{x_j^{ref_u}=1 or -1}^{n-p} u_j^{ref_u} x_j^{ref_u} + \sum_{x_i^{ref_u}=0}^{p} u_i^{ref_u} x_i^{ref_u}\right) \quad (3)$$

$$= \frac{1}{\|u^{ref_u}\|}(n-p) \quad (4)$$
$$= \frac{1}{\sqrt{n}}(n-p) \quad (5)$$

Thus, $\forall x \in S_p^u$, the length of the projection of x^{ref_u} onto the reference vector u^{ref_u} is $(n-p)/\sqrt{n}$. That is, all binary vertices in S_p^u lie on the same hyperplane H_p^u which is perpendicular to the reference vector u^{ref_u} and located at a distance of $(n-p)/\sqrt{n}$ from the vertex $\overline{u}$, that is, a distance $p/\sqrt{n}$ to vertex u. Hence, every hyperplane $H_p^u \in H^u$, $0 \leq p \leq n$, is parallel to every other hyperplane in H^u. There are n+1 such mutually parallel linear hyperplanes H_p^u's. Among them, u is on H_0^u and $\overline{u}$ is on H_n^u. .... $\square$

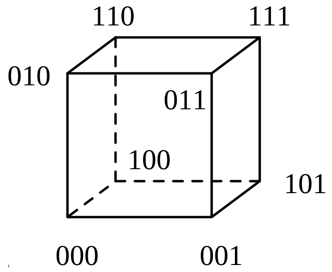

Figure 3. 3-dimensional hypercube.

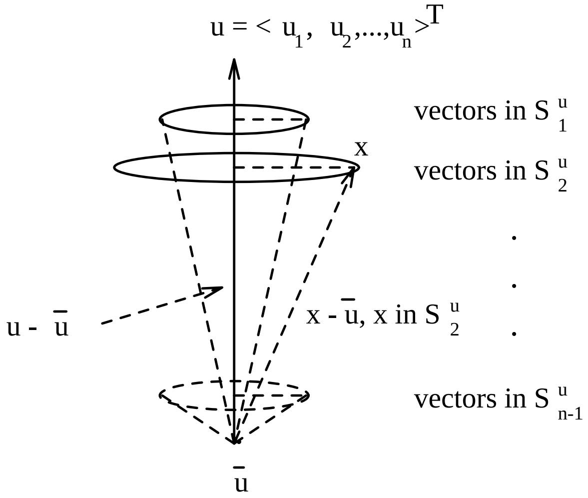

Figure 4. Given an n-dimensional binary vertex u, all other vertices of an n-dimensional binary hypercube are located on n mutually parallel hyperplanes. The hyperplanes have same normal vector $u - \overline{u}$.

Appendix B

Appendix B derives the expression defining hyperplane H_p^u among binary vertices in an n-dimensional binary hypercube.

Let us consider the defining expression of a hyperplane H_p^u, $1 \leq p \leq n$. Let x be a binary vertex on hyperplane H_p^u. From equations (1) and (5), we have:

$$l_x^u = \frac{1}{\|u^{ref_u}\|}(u^{ref_u})^T x^{ref_u} = \frac{1}{\sqrt{n}}(u - \overline{u})^T(x - \overline{u}) = \frac{1}{\sqrt{n}}(n - p)$$

Thus,

$$(u - \overline{u})^T(x - \overline{u}) = (n - p)$$

So the defining expression of the hyperplane H_p^u is given by:

$$\begin{aligned}
H_p^u &\equiv (u - \overline{u})^T(x - \overline{u}) = (n - p) && (6)\\
&\equiv (u - \overline{u})^T x - (u - \overline{u})^T\overline{u} - (n - p) = 0, \quad note \;\; u^T\overline{u} = 0 && (7)\\
&\equiv (u - \overline{u})^T x - (n - \|\overline{u}\|^2 - p) = 0, \quad note \;\; \|u\|^2 + \|\overline{u}\|^2 = n && (8)\\
&\equiv (u - \overline{u})^T x - (\|u\|^2 - p) = 0 && (9)\\
&\equiv (u_1 - \overline{u}_1)x_1 + ... + (u_n - \overline{u}_n)x_n - (\|u\|^2 - p) = 0 && (10)\\
&\equiv (2u_1 - 1)x_1 + ... + (2u_n - 1)x_n - (\|u\|^2 - p) = 0 && (11)\\
&\equiv (\sum_{i=1}^{n}(2u_i - 1)x_i) - (\|u\|^2 - p) = 0 && (12)
\end{aligned}$$

From above, it is known that the expressions defining the n+1 mutually parallel linear hyperplanes H_p^u's in an n-dimensional binary hypercube, where $0 \leq p \leq n$, have same coefficients but different constant terms.

Appendix C

Appendix C derives the expression defining hyperplane H_p^u among bipolar vertices in an n-dimensional bipolar hypercube.

All notations here are same as those in Appendices A and B, with one exception: bipolar vector (vertex) is used in place of binary vector (vertex). A set U of k bipolar input vectors u_1,...,u_k of dimension n and a set V of k desired binary/bipolar output vectors v_1,...,v_k of dimension m are given.

In this case $u_i \in \{-1, 1\}$; $u_i + \overline{u}_i = 0$; $u_i^{ref_u} \in \{2, -2\}$; $x_i^{ref_u} = 0$ or 2 if $u_i^{ref_u} = 2$, and $x_i^{ref_u} = 0$ or -2 if $u_i^{ref_u} = -2$; there are p components $x_i^{ref_u}$ of x^{ref_u} such that $x_i^{ref_u} = 0$, and $(n-p)$ components $x_j^{ref_u}$ of x^{ref_u} such that $x_j^{ref_u} = 2 \; or -2$. Then

$$
\begin{aligned}
l_x^u &= \frac{1}{\|u^{ref_u}\|}(u^{ref_u})^T x^{ref_u} && (13)\\
&= \frac{1}{\|u^{ref_u}\|}\sum_{i=1}^{n} u_i^{ref_u} x_i^{ref_u} && (14)\\
&= \frac{1}{\|u^{ref_u}\|}\Big(\sum_{x_j^{ref_u}=2or-2}^{n-p} u_j^{ref_u} x_j^{ref_u} + \sum_{x_i^{ref_u}=0}^{p} u_i^{ref_u} x_i^{ref_u}\Big) && (15)\\
&= \frac{1}{\|u^{ref_u}\|} \times 4(n-p) && (16)\\
&= \frac{1}{2 \times \sqrt{n}} \times 4(n-p) && (17)\\
&= \frac{2}{\sqrt{n}}(n-p) && (18)
\end{aligned}
$$

From equations (13) and (18) we have:

$$l_x^u = \frac{1}{\|u^{ref_u}\|}(u^{ref_u})^T x^{ref_u} = \frac{1}{2\sqrt{n}}(u - \overline{u})^T (x - \overline{u}) = \frac{2}{\sqrt{n}}(n-p)$$

Thus,

$$(u - \overline{u})^T (x - \overline{u}) = 4(n-p)$$

So the hyperplane H_p^u is given by

$$
\begin{aligned}
H_p^u &\equiv (u - \overline{u})^T (x - \overline{u}) = 4(n-p) && (19)\\
&\equiv (u - \overline{u}^T)x - (u - \overline{u})^T \overline{u} - 4(n-p) = 0 && (20)\\
&\equiv (u - \overline{u})^T x - (u^T \overline{u} - \|\overline{u}\|^2 + 4n - 4p) = 0, \; note \; u^T \overline{u} = -n && (21)\\
&\equiv (u - \overline{u})^T x - (-n - n + 4n - 4p) = 0 && (22)\\
&\equiv (u - \overline{u})^T x - (2n - 4p) = 0 && (23)\\
&\equiv (u_1 - \overline{u}_1)x_1 + ... + (u_n - \overline{u}_n)x_n - (2n - 4p) = 0 && (24)\\
&\equiv 2u_1 x_1 + ... + 2u_n x_n - (2n - 4p) = 0 && (25)
\end{aligned}
$$

$$
\begin{aligned}
&\equiv \quad u_1x_1 + ... + u_nx_n - (n-2p) = 0 && (26)\\
&\equiv \quad (\sum_{i=1}^{n} u_ix_i) - (n-2p) = 0 && (27)
\end{aligned}
$$

References

[1] S. Amari, Neural Theory of Association and Concept-Formation, *Biol. Cyber.* 26, pp. 175-185, 1977.

[2] H. H. Chen et al., Higher Order Correlation Model for Associative Memory, *American Institute of Physics Conference Proceedings*, no. 151: Neural Networks for Computing, pp. 398-403, Snowbird, Utah, 1986.

[3] C. Chen & V. Honavar, Neural Network Automata, *Proc. of World Congress on Neural Networks*, vol. 4, pp. 470-477, San Diego, June 1994.

[4] C. Chen & V. Honavar, A Neural Architecture for Parallel Set Operations, *paper in preparation*, 1995a.

[5] C. Chen & V. Honavar, A Neural Architecture for a High-Speed Database Query System. Accepted by *International Journal of Microcomputer Applications.* Preliminary version available as Iowa State University Dept. of Computer Science Tech. Rep. ISU-CS-TR 95-11, 1995b.

[6] C. Chen and V. Honavar, A Neural Network Architecture for Syntax Analysis. *Submitted.* Preliminary version available as Iowa State University Dept. of Computer Science Tech. Rep. ISU-CS-TR 95-18, 1995c.

[7] J. Fodor & Z. Pylyshyn, Connectionism and Cognitive Architecture: A Critical Analysis, In: *Connections and Symbols*, Pinker, S. & Mehler, J. (Ed.), *MIT Press*, Cambridge, MA, 1988.

[8] S. I. Gallant, Neural Network Learning and Expert Systems, Chapter 2 & 5, *MIT Press*, Massachusetts, 1993.

[9] Goonatilake, S. and Khebbal, S. (Ed.), Intelligent Hybrid Systems. *Wiley*, London, 1995.

[10] K. Haines & R. Hecht-Nielsen, A BAM with Increased Information Storage Capacity, *Proc. of IJCNN*, vol. 1, pp. 181-190, 1988.

[11] J. Hao & J. Vandewalle, A New Model of Neural Associative Memories, *International Journal of Neural Systems*, vol. 5, no. 1, pp. 39-47, March 1994.

[12] S. Haykin, Neural Networks, *MacMillan*, New York, 1994.

[13] V. Honavar, Symbolic Artificial Intelligent and Numeric Artificial Neural Networks: Towards A Resolution of the Dichotomy, In: *Computational Architectures Integrating Symbolic and Neural Processes*, Sun, R. & Bookman, L. (Ed.), *Kluwer*, New York, 1994.

[14] V. Honavar & L. Uhr (Ed.), Artificial Intelligence and Neural Networks: Steps Toward Principled Integration. *Academic Press*, New York, 1994.

[15] V. Honavar & L. Uhr, Integrating Symbol Processing Systems and Connectionist Networks, In: *Intelligent Hybrid Systems*, Goonatilake, S. & Khebbal, S. (Ed.), *Wiley*, London, 1995.

[16] J. J. Hopfield, Neural Networks and Physical Systems with Emergent Collective Computational Abilities, *Proc. Natl. Acad. Sci. USA*, vol. 79, pp. 2554-2558, April 1982.

[17] T. Kohonen, Correlation matrix memories, *IEEE Transactions on Computers*, vol. c-21, no. 4, pp. 353-359, April 1972.

[18] B. Kosko, Adaptive Bidirectional Associative Memories, *Applied Optics*, vol. 26, no. 23, pp. 4947-4960, Dec. 1987.

[19] B. Kosko, Bidirectional Associative Memories, *IEEE Trans. Syst. Man Cybern.*, vol. 18, no. 1, pp. 49-60, Jan./Feb. 1988.

[20] Y. Kumagai, J. Kamruzzaman & H. Hikita, Further Cross Talk Reduction of Associative Memory and Exact Data Retrieval, *Proc. of IJCNN*, vol 3, pp. 1371-1378, San Francisco, 1993.

[21] S. Y. Kung, Digital Neural Networks, *Prentice Hall*, New York, 1993.

[22] D. S. Levine & M. Apariciov (Ed.), Neural Networks for Knowledge Representation, *Lawrence Erlbaum*, New York, 1994.

[23] R. J. McEliece, E. C. Posner, E. R. Rodemich & S. S. Venkatesh, The Capacity of the Hopfield Associative Memory, *IEEE Trans. Inform. Theory*, vol. IT-33, no. 4, pp. 461-482, July 1987.

[24] T. M. McKenna, The Role of Interdisciplinary Research Involving Neuroscience in the Development of Intelligent Systems, In: *Artificial Intelligence and Neural Networks: Steps Towards Principled Integration*, V. Honavar & L. Uhr (Ed.), Academic Press, San Diego, CA, 1994.

[25] A. L. Meyerowitz, Neural Networks: A Computer Science Perspective, *Naval Research Review*, vol. 43, No. 2. pp. 13-18.

[26] M. Minsky and S. Papert, Perceptrons: An Introduction to Computational Geometry, MIT Press, Massachusetts, 1969.

[27] K. Nakano, Associatron - A Model of Associative Memory, *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-2, no. 3, pp. 380-388, July 1972.

[28] A. Newell. Symbol Systems, *Cognitive Science* Vol. 4, pp. 135-183, 1980.

[29] C. H. Séquin & R. D. Clay, Fault Tolerance in Artificial Neural Networks, *Proc. IJCNN*, vol. 1, pp. 703-708, San Diego, 1990.

[30] M. E. Sloan, Computer Hardware and Organization, Science Research Associates, Inc., Chicago, 1976.

[31] R. Sun & L. Bookman (Ed.), Computational Architectures Integrating Symbolic and Neural Processes. *Kluwer*, New York, 1994.

[32] G. Swaminathan, S. Srinivasan, S. Mitra, J. Minnix, B. Johnson & R. Iñigo, Fault Tolerance of Neural Networks, *Proc. of IJCNN*, vol 2, pp. 699-702, Washington DC, 1990.