

DNN: A NEW NEURAL NETWORK ARCHITECTURE OF  
ASSOCIATIVE MEMORY WITH PRUNING AND  
ORDER-SENSITIVE LEARNING AND ITS APPLICATIONS

THESIS SUBMITTED FOR  
THE AWARD OF THE DEGREE OF

**Doctor of Philosophy**

M. SREENIVASA RAO



DEPARTMENT OF COMPUTER & INFORMATION SCIENCES  
SCHOOL OF MATHEMATICS & COMPUTER/INFORMATION SCIENCES  
**UNIVERSITY OF HYDERABAD**  
HYDERABAD, INDIA

**1998**

**Department of Computer & Information Sciences  
School of Mathematics & Computer/Information Sciences**


**UNIVERSITY OF HYDERABAD**

This is to certify that I, M. Sreenivasa Rao, have carried out the research embodied in the present thesis for the full period prescribed under the Ph.D. ordinances of the University.


I declare, to the best of my knowledge, that, no part of this thesis was earlier submitted for the award of any research degree of any University.



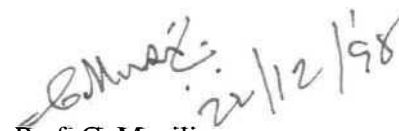
M. Sreenivasa Rao



Prof. Arun Kumar Pujari  
Supervisor.



Prof. Arun Agarwal  
Head of the Department.



Prof. C. Musili  
Dean of the School.

## **ACKNOWLEDGEMENTS**

I express my sincere thanks and gratitude to my research supervisor Prof. Arun Kumar Pujari for his able guidance, constant support, expert advice, encouragement and constructive criticism throughout the course of this research work. I am extremely grateful to him for suggesting this interesting problem and for his untiring interest he has shown in my work which is vital for the successful completion of this thesis. I am also grateful to the members of his family.

I thank all other faculty members of the Department of Computer Science, University of Hyderabad, Hyderabad, for their encouragement. I thank the staff of the AI lab for their help extended to do this research work. I am also thankful to the office staff of Department of Computer Science for their help and cooperation.

I am grateful to the authorities of JNT University, Hyderabad and JNTU College of Engineering, Hyderabad for granting permission to carry out this research.

I wish to express my gratitude to all faculty members of the Department of Computer Science, JNT University for their cooperation and encouragement.

I gratefully acknowledge the encouragement of Prof. A. Venugopal Reddy, Osmania University, Prof. I. V. Ramana, JNT University and Prof. J. A. Chandulal, Andhra University.

I am grateful to the valuable suggestions of Prof. B. Sreenivasan, University of Monash, Australia, Prof. G. Uma Maheswara Rao and Prof. C. K. Mitra, University of Hyderabad, Hyderabad.

I express my everlasting gratitude to my friend Dr. Bh. R. M. Phaneendra and my colleagues Smt. G. Vijaya Kumari and Mr. OBV Ramanaiah for their frequent discussion and valuable suggestions. I shall always remain indebted to them for their kind help.

I express my sincere thanks to my wife Indira and children Prema and Anusha for their patience and support during the period of my research work. I am also thankful to everyone who has been directly and indirectly associated with my work.

**M. SREENIVASA RAO**

# TABLE OF CONTENTS

<b>List of Figures</b>	iv
<b>CHAPTER</b>	
<b>1. Introduction</b>	1
1.1 Basic Concepts of ANN	2
1.2 Learning Rules	3
1.3 Models of Neural Networks	5
1.4 Overview of the Present Work	9
1.5 Layout of the Thesis	11
<b>2. DNN: A New Neural Network Architecture</b>	15
2.1 Introduction	15
2.2 Hopfield Network	16
2.3 Capacity of the Hopfield Network	19
2.4 A Survey of Multiple Neural Network Architectures	21
2.5 Dynamical Neural Network	22
2.5.1 The Network Structure	23
2.5.2 Training of the Network	24
2.6 New Learning Rule	28
2.6.1 Earlier Learning Rules	28
2.6.2 Polyhedral Combinatorics	30
2.6.3 The Synaptic Matrix	31
2.7 Correctness of the Learning Rule	33
2.7.1 User-specified Local Minima	33
2.7.2 Recall Efficiency of Learning Rule	37
2.8 The Efficiency of DNN	38
2.9 Conclusions	41
<b>3. The Special Features of DNN</b>	
3.1 Introduction	43
3.2 Order-Sensitive Learning	44
3.3 Experimental Results	50
3.4 Relative Pruning	51
3.5 Conclusions	54

<b>4. Close Proximity Match for Large Databases Using DNN</b>	<b>55</b>
4.1 Introduction	55
4.2 Text Retrieval Methods	56
4.3 Signature	58
4.3.1 Signature Extraction Methods	59
4.3.1.1 Superimposed Coding Technique	59
4.3.1.2 Hashing Method	60
4.4 Library Retrieval System	63
4.4.1 Experimental Results of Library Retrieval System	66
4.5 Protein Database	68
4.5.1 Experimental Results of Protein Database	70
4.6 Discussion	71
4.7 Conclusions	74
<b>5. Word Sense Disambiguation Using DNN</b>	<b>75</b>
5.1 Introduction	75
5.2 Word Sense Disambiguation	76
5.2.1 What is WSD?	76
5.2.2 Different Approaches	76
5.2.2.1 AI-based Approaches	77
5.2.2.2 Knowledge-based Approaches	78
5.2.2.3 Corpus-based Approaches	79
5.2.3 Context	79
5.3 Earlier Corpus-based Methods	80
5.4 DNN for WSD	82
5.5 Context Extraction	84
5.5.1 Stop-words	85
5.6 Methodology	87
5.7 Experimental Results	91
5.8 Discussion	96
5.9 Conclusions	97
<b>6. Conclusions</b>	<b>99</b>
<b>Appendix-1</b>	<b>103</b>
<b>Appendix-2</b>	<b>105</b>
<b>Appendix-3</b>	<b>106</b>
<b>Appendix-4</b>	<b>107</b>
<b>References</b>	<b>108</b>

# **LIST OF FIGURES**

## **Chapter 2**

- 2.1 The Architecture of DNN
- 2.2(a) First Level Grouping and Leader Nodes
- 2.2(b) Second Level Grouping after Pruning First Level
- 2.2(b) Final Node after Pruning Second Level
- 2.3 The Algorithm for Training and Pruning of DNN
- 2.4 Example of Training and Pruning of DNN
- 2.5 Convergence Transition Following the New Learning Rule
- 2.6(a) Recall Efficiency of Hebbian Rule
- 2.6(b) Recall Efficiency of New Learning Rule
- 2.7 Performance Trade-off of NET-A, NET-B, NET-C and DNN

## **Chapter 3**

- 3.1 Colour Experiment

## **Chapter 4**

- 4.1 The Algorithm for Hashing Method
- 4.2 The Algorithm for Superimposed Coding Technique
- 4.3 Example of Superimposed Coding
- 4.4 The Close Proximity Match Algorithm for Library Database
- 4.5(a) Initiation with Training and Test Patterns
- 4.5(b) Hopfield Stable States, Pruning and Feedback (Phase1)
- 4.5(c) Training of Phase-1 Pruned Network with Feedback Patterns
- 4.5(d) Stable States of the Phase-1 Pruned Network (Phase 2)
- 4.5(e) Training of Phase-2 Pruned Network with Feedback Patterns
- 4.5(f) Final Output Pattern
- 4.6 Library Retrieval System
- 4.7 Protein Database with Substitutions
- 4.8 Protein Database with Deletions
- 4.9 Protein Database with Insertions
- 4.10 Protein Database with Bunching

## **Chapter 5**

- 5.1 The Level-wise Algorithm
- 5.2 The Algorithm for Word Sense Disambiguation

## **Chapter 6**

6.1 VLSI Implementation of DNN

6.2 The Algorithm for Training and Pruning of VLSI Implemented DNN

# **Chapter 1**

## **Introduction**

Artificial Neural Networks are abstract representations of brain information processing. The hope to reproduce at least some of the flexibility and power of human brain by artificial means has lead to the subject of study known as Neural Networks, Neural Computation or Brain like computation [Anderson 92].

The recent resurgence of interest in neural networks has its roots in the recognition that the brain performs computations in a different manner than do conventional computers. Conventional computers are extremely fast and precise at executing sequence of instructions that have been precisely formulated for them. But human brain is more efficient than conventional computers at computationally complex tasks such as vision, speech, information retrieval and pattern recognition, though human brain is composed of neurons which are million times slower than computer gates.

Unfortunately the understanding of biological neural systems is not developed sufficiently enough to address the functional similarities that may exist between the biological and man made neural systems. As a result, many major potential gains derived from such functional similarities, if they exist have yet to be exploited. It is not necessary that the architecture of brain is copied as it is to the extent it is understood. Implementation of the functionality of the brain in any manner is the guiding force in neurocomputing.

Neurocomputing, a functionally new and different approach for information processing,



is concerned with parallel, distributed and adaptive information processing systems that develop information processing capabilities in adaptive response to an information environment [Hecht-Nielsen 91]. In other words Artificial Neural Systems(ANS) are physical cellular systems which can acquire, store and utilize experiential knowledge.

## 1.1 Basic Concepts of ANN

The potential of Artificial Neural Networks (ANN) relies on massively parallel architecture composed of many simple computational elements connected by edges called weights. The basic computational element in an artificial neural network is called neuron and is also known as node or processing element. Brain researchers have identified over 100 different kinds of biological neurons. Processing elements also come in a variety of types. McCulloch and Pitts [McCulloch 43] proposed a binary threshold unit as computational model for a neuron. As a unit, the neural network can be represented with threshold and weight functions.

In an artificial neural network, inputs are fed to neurons through synapses (connection weights). Basically, the output of a neuron in a neural network is a weighted sum of its inputs, but a threshold function is also used to determine the final value or the output. For a detailed exposure for threshold functions, refer [Kosko 92].

The state of a neuron is nothing but the activity of a neuron at any time instance. This attribute may be discrete or continuous valued. An update rule or transition rule or activation rule is a rule for evaluating the state of one or more neurons under the existing conditions and changing them if necessary.

Topology of ANN is the pattern of connectivity in such a way that it can be viewed as weighted directed graph in which artificial neurons are nodes and directed edges (with

weights) are connections between neuron outputs and neuron inputs. Basing on the connection pattern (architecture), ANNs may be grouped into two categories: *feed-forward networks (non-recurrent)* are those in which graphs have no loops, while *feed back (recurrent)* networks are those in which loops occur because of feedback connections. Broadly speaking feed forward networks are static, where as recurrent or feedback networks are dynamic systems. When a new input pattern is presented, the neuron outputs are computed. Because of the feedback paths, the inputs to each neuron are then modified, which leads the network to enter a new state.

The state of all neurons in a feedback network at any instance of time is the *state of the neural network*. *Stable states* of a feedback network are the states of the network which do not change under usual disturbances in the states of the neurons of the network. *Learning* in a neural network is the process of making certain set of states of network as stable states. The function in which one can substitute the state values of the neurons that represents the energy of the network (at that instance) is called its *energy function*. During the updation of the network, the value of the energy function decreases and eventually reaches a minimum. In view of the above definition this state of the network is referred to as *stable state* and this minimum is called *local minimum*. When this state is reached, the network is said to be stable.

Since this thesis addresses a new neural network architecture with a new learning rule, some of the learning rules known in the literature are recalled in the next section.

## 1.2 Learning Rules

The ability to learn is a fundamental trait of intelligence. ANNs' ability to automatically learn from examples makes them attractive and exciting. A learning process in the ANN

context can be viewed as the problem of updating network architecture and connection weights so that a network can efficiently perform a specific task [Jain 96].

A learning algorithm refers to a procedure in which learning rules are used for adjusting the weights. There are three main learning paradigms: *supervised*, *unsupervised* and *hybrid*.

In *supervised learning*, or learning with a "teacher", the network is provided with a correct answer (output) for every input pattern. Weights are determined to allow the network to produce answers as close as possible to the known correct answers.

In contrast, *unsupervised learning*, or learning without a teacher, does not require a correct answer associated with each input pattern in the training data set. It explores the underlying structure in the data, or correlates patterns in the data and organizes patterns into categories from these correlations.

*Hybrid learning* combines supervised and unsupervised learning.

There are four basic types of learning rules: error-correction, Boltzmann, competitive learning and Hebbian learning.

Error-correction learning [Rosenblatt 58]: In this learning paradigm, the network is given a desired output for each input pattern. During the learning process the actual output  $y$  generated by the network may not be equal to the desired output  $d$ . The basic principle of error-correction learning rules is to use the error signal  $(d - y)$  to modify the connection weights to gradually reduce this error.

Boltzmann learning [Anderson 88]: It is a stochastic learning rule derived from information theoretic and thermodynamic principles. The objective of Boltzmann learning is to adjust the connection weights so that the states of visible units satisfy a particular

desired probability distribution.

Competitive learning [Haykin 94]: In this method of learning output units compete among themselves for activation. As a result, only one output unit is active at any given time. This phenomenon is known as winner-take-all. Competitive learning is found to exist in biological neural networks.

Hebbian learning [Hebb 49]: The oldest learning rule is Hebb's postulate of learning. Hebb's rule is based on the following observation from neurobiological experiments: If neurons on both sides of a synapse are activated synchronously and repeatedly, the synapse's strength is selectively increased. Mathematically, the Hebbian rule can be described as

$$w_{ij}(T + 1) = w_{ij}(r) + \eta y_i(T) x_j(r) \quad (1.2.1)$$

where  $x_i$  and  $y_j$  are the output values of neurons  $i$  and  $j$ , respectively, which are connected by synapse  $w_{ij}$ , and  $\eta$  is the learning rate. Note that  $x_i$  is the input to the synapse. The important property of this rule is that learning is done locally, that is, the change in the synaptic weight depends only on the activities of the two neurons connected by it. This significantly simplifies the complexity of the learning circuit in a VLSI implementation.

Some of these learning rules are recalled as and when required in subsequent chapters.

### 1.3 Models of Neural Networks

Researchers from many scientific disciplines are designing artificial neural networks to solve a variety of problems in pattern recognition, prediction, optimization, associative memory and control. In this direction the premier work is due to McCulloch and Pitts, later known as Perception. Afterwards many models like Multilayer feedforward, Kohonen's self-organizing map, ART and Hopfield have played vital role in the development.

This section deals with different trendsetting models of neural networks.

**Perceptron** [McCulloch 43; Rosenblatt 58; Rumelhart 86]: Perceptron is the first precisely specified, computationally oriented neural network. In this network processing units may be of binary or of continuous type. Simple reinforcement kind of learning rules are used.

**Multilayer perceptron** [Rumelhart 86; Minsky 69]: The most popular class of multilayer feed-forward networks is multilayer perceptrons with one or more layers of nodes between the input and the output nodes. The first layer is the input layer and the last the output layer. The layers that are placed between the first and the last layers are the hidden layers. Multilayer perceptrons can form arbitrarily complex decision boundaries and can represent any Boolean function. The development of the back-propagation learning algorithm [Rumelhart 86] for determining weights in a multilayer perceptron has made these networks the most popular among researchers and users of neural networks. The back-propagation algorithm enables one to determine the errors in the hidden layer outputs which are used as a basis for adjustment of connection weights between input and hidden layers.

**Kohonen Self-organizing Map** [Kohonen 84]: Many parts of the brain are organized in such a way that the aspects of the sensory environment can be represented in the form of two dimensional maps. Self-organizing network is an attempt to construct this aspect by artificial means. The synaptic weights to a neuron represent prototype vectors defining the classes or clusters. The fundamental fact that must hold true for a topographically organized system is that nearby neurons respond similarly. The essential mechanism of this model is to cause the neural network to modify itself to achieve this character. Kohonen's self-organizing map can be used for projection of multivariant data, density

approximation and clustering.

Adaptive Resonance Theory model (ART) [Carpenter 88]: ART is introduced to solve the stability-plasticity dilemma i.e., learning new things (plasticity) and yet retain the existing knowledge (stability). The networks designed by this approach are better suited to adapt to unexpected changes as biological neural networks are geared to do. Carpenter and Grossberg [Carpenter 91] contend that adaptive resonance, defined to be a state of collective activity of the behavioral system as a whole, arises when feedforward and feedback computations are consonant. Accordingly, they propose the ART1 and ART2 networks to deal with the stability and plasticity dilemma.

The network has a sufficient supply of output units, but they are not used until deemed necessary. A unit is said to be *committed* (*uncommitted*) if it is (is not) being used. The learning algorithm updates the stored prototypes of a category only if the input vector is sufficiently similar to them. An input vector and a stored prototype are said to resonate when they are sufficiently similar. The extent of similarity is controlled by a *vigilance parameter*,  $\rho$  with  $0 < \rho < 1$ , which also determines the number of categories. When the input vector is not sufficiently similar to any existing prototype in the network, a new category is created and an uncommitted unit is assigned to it with the input vector as the initial prototype.

Hopfield network [Hopfield 82, 84]: Hopfield used the network energy function as a tool for designing recurrent networks and for understanding their dynamic behavior. Hopfield's formulation made explicit the principle of storing information as dynamically stable attractors and popularized the use of recurrent networks for associative memory and for solving combinatorial optimization problems. A Hopfield network with  $n$  units has two versions: binary and continuous valued. Let  $v_i$  be the state or output of the  $i^{th}$

neuron. For binary networks  $v_i$  is 1 or 0, for bipolar  $v_i$  is +1 or -1, but for continuous networks  $v_i$  could be any real value. Let  $w_{ij}$  be the synaptic weight connecting neurons  $i$  and  $j$ . The network dynamics for the binary Hopfield network are as follows:

If the state of  $i$  th neuron at time  $t$  is denoted by  $x_i(t)$ , then the neuron at the next time step  $t + 1$  is computed as

$$x_i(t + 1) = \text{sgn}\left(\sum_{j=1}^n w_{ij}x_j(t) - t_i\right) \quad (1.3.1)$$

where the  $\text{sgn}(x)$  is signum function that produces 1 if  $x > 0$  and 0 otherwise. The central feature of the Hopfield network is that each state can be associated with a quantity called energy  $E$ . The energy of the network at a particular state is given by [Hopfield 82].

$$E = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij}x_i x_j + \sum_{i=1}^n t_i x_i \quad (1.3.2)$$

The energy function is of Lyapunov type which maps system state variables to real numbers and monotonically decreases with time [Kosko 92]. The central property of the energy function is that as network state evolves according to the network dynamics, the network energy always decreases and eventually reaches a local minimum point (attractor) where the network stays with a constant energy.

Associative memory [Hopfield 82; Kohonen 72]: When a set of patterns is stored as local minima (attractors or stable states) in a network, it can be used as an associative memory. Any pattern present in the basin of attraction of stored pattern can be used as index to retrieve it. The set of all initial state vectors, that converges to a stable state is called its basin of attraction. An associative memory usually operates in two phases: memory storage and information retrieval. In the storage phase, the weights in the network are determined (using a learning rule like Hebb rule) so that the attractors of the network memorize a set of patterns to be stored. In the retrieval phase, the input pattern is used as the initial state of the network, and the network evolves according to

its dynamics. A pattern is produced or retrieved when the network reaches equilibrium. The number of patterns stored in a network is called *capacity of the network*. It is finite because a network with  $n$  binary neurons has a maximum of  $2^n$  distinct states and not all of them are attractors. Some attractors are called spurious attractors (or spurious states) when they store patterns different from those in the training input.

#### 1.4 Overview of the Present Work

Associative memory concept is related to the association of stored information for a given input patterns. High capacity and accurate recall are the most desired properties of a neural network, which is being classified as an associative memory network. Models for improving and characterizing the theoretical storage capacity of associative memories have been suggested by many researchers in the recent past. The most significant work related to associative memory networks is attributed to Hopfield model [Hopfield 82, 84]. However, there are certain limitations in the models known in the literature including Hopfield, namely the practical storage capacity is  $0.15n$ ,  $n$  is the number of neurons, the stability of the patterns decreases as the number of patterns stored in the memory approaches the maximum capacity [Muller 90] and the accuracy of recall falls with the increase in the number of stored patterns [Han 89]. Various learning rules including the popular Hebb rule proposed for Hopfield network, suffer from the presence of spurious states.

These limitations have prompted us for an alternative, which has resulted in the new network termed here as Dynamical Neural Network (DNN for short) with the properties such as associative memory, pruning and order sensitive learning. It is called Dynamical Neural Network in the sense that its architecture gets modified dynamically over time as training progresses. The proposed network is a massively parallel and distributed



prompted us to propose this new learning algorithm. This has not yet been explored in the context of artificial neural networks. Another novel capability of the network, allows pruning of basic nodes in binary order as it progressively carries out associative memory retrieval and enables a part of the network parallelly reusable for another task. Mathematical proofs and experimental results are presented to substantiate the above inferences.

The efficiency of the network is demonstrated with the presentation of applications in various fields. These different application areas vary from Library Database, Protein Structure Database to Natural Language Understanding.

## 1.5 Layout of the Thesis

This thesis is organized as follows:

Chapter 1, this chapter deals with the introduction and background for this work. The motivation for the present work is also explained in detail.

In Chapter 2, a detailed description of the proposed model is presented. The architecture of the proposed network has a composite structure wherein each node of the network is a Hopfield network by itself. The Hopfield network employs the new order-sensitive learning technique and converges to user-specified stable states but not to any spurious states. The survey on multiple neural network architectures and on learning rules is presented. The mathematical tools required for the learning rule (synaptic matrix) is presented in detail. The correctness of the learning rule is validated with the help of energy function of the network. Experimental results are presented in this chapter, which confirm the superiority of this learning rule in recall over Hebb rule. The recall efficiency of this DNN is found to be better than other single and multiple neural networks (of same configuration) for some known learning strategies.

In Chapter 3, the special characteristics of DNN, i.e., order-sensitivity and relative pruning are described. Let  $X$  and  $Y$  be two distinct training patterns. A learning rule is said to be order-sensitive if network behaves differently for different order, of presentation of the training patterns. In the present context of Hopfield network (the basic block of DNN), it is identified that the learning is order-sensitive, as the learning rule generates a different synaptic matrix when  $X$  precedes  $Y$  in the training compared to the matrix when  $X$  follows  $Y$ . It is emphasized here that order-sensitivity is one of the important features in human learning process and has not yet been explored in the context of artificial neural networks. Moreover, the natural instinct of human learning is to have longer impression of patterns that are learnt earlier.

Basing on another biological phenomenon a new notion called relative pruning is introduced for the network. Relative pruning is different from the standard notion of pruning of neural networks, which aims at removing the neurons or weights that are not participating in training and without loss of generality of the training algorithm. On the other hand, in our proposed network, the network gets pruned relatively to an extent that half the number of basic nodes are relieved in each iteration and this finally results in leaving one node in the network. The advantage of this relative pruning is that the network structure employs the hardware most optimally when it is implemented and the relatively pruned nodes can be used for parallel processing of the next series of data. In other words this capability enables a part of the network parallelly reusable for another task.

Chapter 4, describes the applicability of the DNN to close-proximity match in large databases like library and protein databases. In a library retrieval system it is required to retrieve a book using inexact keywords. Similarly, in the protein sequence database determining approximate match is required to discover the relationship between newly

sequenced protein that resulted due to evolution and the various classes of proteins already available in the database. In both the cases it is required to perform approximate retrieval from large databases. The DNN is not trained by the actual data, rather with signatures which are obtained by superimposed coding on partitions of the input data. With the proposed network, this problem can be solved with an accuracy of 95% and above.

Word sense disambiguation (WSD) is the task of assigning sense labels to the occurrence of an ambiguous word. WSD is one of the hard tasks to be accomplished in Natural Language Understanding. Our premise is that the context in which the word is used, plays a major role in disambiguation and the context is largely determined by co-occurring words that are present in the sentence. In Chapter 5, it is demonstrated that DNN is suitable to handle WSD. Manually sense-tagged corpora is used for obtaining training sentences and an algorithm is used to extract the context of a particular sense of the word. The disambiguation of the word takes place by matching (associating) the signatures of the contexts of training and test sentences using the DNN.

Though the Hopfield model in its conventional form is suitable for associative memory, it is not suitable for word-sense disambiguation due to the existence of spurious stable states. The spurious stable states may lead to an invalid word-sense of a given word. But the DNN with a new learning rule to avoid any spurious states and with an enhanced retrieval capacity is suitable for word sense disambiguation. The experimental results reported confirms the theoretical findings. The disambiguator is tested on Telugu words. The performance of the disambiguator is quite good and it has an average accuracy of 83%.

In Chapter 6, a primitive hardware implementation of DNN is presented along with the

concluding remarks for the thesis.

In a nutshell, the results embodied in this thesis explain about a new architecture. This architecture betters in many aspects over its predecessors like, avoiding spurious states, converging only to user specified states and associative memory with 100% perfect recall. The advantage in this architecture is that it embeds relative pruning, a new interesting feature. The learning rule is order-sensitive, also relatively a very recent phenomenon in neural networks. The claims are corroborated with substantial experimental data.

# Chapter 2

## DNN: A New Neural Network Architecture

### 2.1 Introduction

Many models of neural networks are proposed to solve the problems of classification, vector quantization, self-organization, associative memory. Associative memory concept is related to the association of stored information for a given input pattern. High storage capacity, fast learning rate and the ability to recall accurately are the most desired properties of a neural network, which is being classified as an associative memory network. Many models by different researchers [Hilberg 97; Smith 96; Sukhaswami 93; Kang 93] for improving and characterizing the theoretical storage capacity of associative memories have been appeared in the recent past. One of the major contribution to the associative memory networks relates Hopfield [Hopfield 82, 84]. However there are certain limitations in the models known in the literature including Hopfield, namely the practical storage capacity is  $0.15n$ ,  $n$  is the number of neurons; the stability of the patterns decrease as the number of patterns stored in the memory approaches the maximum capacity [Muller 90] and the accuracy of recall falls with the increase in the number of stored pattern [Han 89]. Various learning rules including the popular Hebb rule proposed for Hopfield network, suffer from the presence of spurious states. These limitations have prompted us for an alternative, which has resulted in the network known as DNN with

The major contents of this chapter are communicated to "International Journal of Neural Systems" with the title "A new neural network architecture as associative memory with pruning and order-sensitive learning" and is accepted for publication.

the following properties. The proposed network is based on the principle of associative memory but unlike other associative memories it does not allow any spurious stable states. The network has a composite structure wherein each node of the network is a Hopfield network by itself. A new learning rule is proposed for the Hopfield network (a node in DNN) which converges only to user-specified stable states, not to any spurious states. It demonstrates a novel idea of order-sensitive learning which gives preference to chronological order of presentation of the exemplar patterns. The proposed network prunes nodes as it progressively carries out associative memory retrieval. It may be noted that pruning is also one of the desired properties of neural computing. These notions are explained in the subsequent chapters.

The mathematical derivation of this learning technique is based on geometrical structure of the network and that of the energy function. In this work besides the introduction of new architecture, the performance of the same is evaluated with different learning algorithms. The proposed architecture is designed in such a way that the architecture and the learning rule do match the best combination. This resulted due to the different experiments conducted with different combinations of architectures and learning rules.

Since the basic node in DNN is the Hopfield network, for the sake of continuity, a brief introduction to Hopfield network (in addition to the earlier description) and its capacity is presented in the following sections.

## **2.2 Hopfield Network**

A Hopfield neural network [Hopfield 82, 84] is a feed-back neural network of  $n$  neurons and is defined by the pair  $(W, T)$ , where  $W$  is  $n \times n$  synaptic matrix ( $w_{ij}$ ,  $i, j = 1, 2, \dots, n$ ) and  $T$  is a  $n \times 1$  threshold vector ( $t_i$ ,  $i = 1, 2, \dots, n$ ). The state of each neuron can take

one of two possible binary values: 1 and 0. However, in literature the different kinds of continuous and discrete valued neurons have been associated with Hopfield network. If the state of  $i^{th}$  neuron at time  $r$  is denoted by  $x_i(r)$ , then the neuron at the next time step  $r + 1$  is computed as

$$X_{i,(t+1)} = \text{sgn}\left(\sum_{j=1}^n W_{ij}x_j(t) - t_i\right) \quad (2.2.1)$$

The state vector  $X = (x_1, x_2, \dots, x_n)$  is called stable state if  $X = \text{sgn}(\mathbf{W}^t \mathbf{X} - \mathbf{T})$ . The set of all state vectors, that converges to a stable state  $X$  is called the basin of attraction of  $X$ . The order of updating of neurons at each time step can be synchronous or asynchronous. In synchronous updating all the neurons are updated for each time step. In asynchronous updating, for each time step one neuron is selected for updating. The selection of a neuron can be either random or deterministic. In random asynchronous updating, at a particular time  $r$  a neuron is selected randomly for updating. Using update rule (Eqn. 2.2.1) the state of the selected neuron at the next time step  $r+1$  is computed. If there is a change in the state of the selected neuron, in the next step any neuron can be randomly selected for updating. If there is no change in the state of the selected neuron, another neuron in the remaining list is selected randomly. If the exclusion list includes all the neurons then the network is said to have reached a stable state. Similarly in the deterministic asynchronous update, the selection of neuron is done based on some deterministic mechanism like fixed sequence or neurons receiving maximum local field  $h_i(r)$ .

$$h_i(r) = \sum_{j=1}^n W_{ij}x_j(r) - t_i \quad (2.2.2)$$

In the present work the order of updating of neurons is deterministic and asynchronous.

Update rule: Let  $X$  be the state vector of the network at a particular time  $r$ . The selection of the neuron i.e. to be updated at the next time step  $r + 1$  is as follows:

For each  $i$ ,  $1 < i < n$

do

Compute  $X^i(\tau + 1) = (x_1^i, x_2^i, \dots, x_n^i)$  as

$$x_j^i = x_j(\tau), j \neq i \text{ and}$$

$$x_i^i = 1 - x_i(\tau)$$

end do

For each  $i$ ,  $1 < i < n$ , compute energy (Eq. 1.3.2) at  $X^i(\tau + 1)$  as  $E_{X^i}(\tau + 1)$

For each  $i$ ,  $1 < i < n$ ,  $\Delta E^i = E_{X(\tau)} - E_{X^i(\tau+1)}$

Compute

$$\Delta E_X = \min_i \Delta E^i = \Delta E^k \quad (2.2.3;$$

The neuron that is to be updated is  $k^{th}$  neuron.

The stable states that are different from the fundamental memories are called *spurious state* vectors. The patterns which are presented to the network initially are called *exemplar or training patterns*. The input pattern for which the associated pattern is retrieved by the network is called *test pattern*. The main task of designing a Hopfield network is the specification of the synaptic matrix and the threshold vector keeping certain desired states in mind as the stable states. The steps involved to compute the synaptic matrices is generally known as learning rule. The effectiveness of learning rule is determined by the number of state vectors that it can make stable in the Hopfield network without producing spurious states [Farrell 90].

Associative or content-addressable memory is a device by which storage and retrieval of information is done by association with other information. The recall of information is based on partial knowledge of its content without the knowledge of its storage



location. Thus, it has the property to recall the correct associated stored pattern of unknown pattern presented which is noisy, distorted or incomplete. Hopfield network has this associative memory property which makes the network suitable for real-life retrieval systems.

### 2.3 Capacity of the Hopfield Network

The number of state vectors that can be made stable in a Hopfield network is known as capacity of the Hopfield network [Sharma 94]. Theoretically, Hopfield network with  $n$  binary neurons has the capacity  $2n$ . But practically it depends on many other factors like the number of neurons in the network, learning rule, nature of synaptic weights, nature of values that associated with neuron states, relation between state vectors and update rules.

The estimation of the capacity of a Hopfield network is given in [Hopfield 82]. To store a randomly selected  $p$  state vectors  $X_i^m (m = 1, 2, \dots, p; i = 1, 2, \dots, n)$  in the Hopfield network of  $n$  binary neurons, the synaptic weights  $w_{ij} (i, j = 1, 2, \dots, n)$  are obtained as

$$w_{ij} = \frac{1}{p} \sum_{m=1}^p (x_i^m - 1)(x_j^m - 1) \text{ for } i \neq j \text{ and } w_{ii} = 0 \quad (2.3.1)$$

By using random and asynchronous update mechanism computer simulations are conducted and estimated the capacity as  $0.15n$  [Hopfield 82]. That is only  $0.15n$  states can be simultaneously memorized by Hopfield network before an error in recall is severe. But even this is true only when all the patterns are orthogonal to each other, which is not likely for randomly generated patterns [Baram 89]. A binary pattern is said to be orthogonal to another if there are less than or equal to 50% pixels which are same in both of them. The practical capacity of a Hopfield network is also affected by the learning rule used to construct the synaptic matrix. The widely used learning rule is the Hebb rule.

The lower and upper bounds of the capacity of Hopfield network when Hebb learning rule is used are

$$P_{\text{lower}} = 0.75(n - 1)^{\frac{1}{2}} + 2r \quad (2.3.2)$$

$$P_{\text{upper}} = 1.25(n - 1)^{\frac{1}{2}} + 2r \quad (2.3.3)$$

Where  $r$  represents the number of stable state vectors that are aligned with exemplar.

It is observed that for applications like character recognition the Hebb rule is inadequate when the number of patterns stored in a network are more than  $0.05n$  [Sukhaswami 92]. Even the modified Hebb rules [Abbott 89; Gardner 89; Forrest 89; Muller 90] do not ensure a convergence due to the presence of spurious states.

The stability of patterns decreases as number of patterns stored in a memory approaches the maximum capacity, because the number of basins of attraction increases [Muller 90]. Also, accuracy of recall falls and learning time increases with the increase in the number of stored patterns in the network [Han 89].

For the associative memory to be of any practical use it should be able to store a large number of patterns with perfect recall. Therefore, it is important to construct the Hopfield networks with ability to differentiate very similar but distinct state vectors and avoid any spurious states. Hence, a Hopfield model is proposed here with new learning rule which has user-specified states as stable states and avoids spurious states, so that the model is robust for retrieval. This model is able to memorize any non orthogonal pattern, but has limited capacity i.e.,  $p=2$ . In order to overcome the storage capacity limitation a novel neural network architecture has been reported.

## 2.4 A Survey of Multiple Neural Network Architectures

In the foregoing discussion, it is observed that the associative memory model has certain inherent limitations in terms of perfect recall and capacity. In other words, more is the number of patterns to be memorized, there would be more chance of having spurious states [Muller 90]. There have been many attempts to overcome this limitation by exploring several composite structures for associative memory. This section outlines the major work reported in literature in this direction. Characterization of associative memory models for fully coupled networks, asymmetric networks, Ising bonds, sign constrained synapses and multiconnected networks have been reported [Muller 90].

In [Baram 89] a fractal approach to associative memory design is proposed. In this model the network topology is to connect self-similar Hopfield units similar to fractals. In this model, small sub networks are interconnected in a layered hierarchy. The sub networks corresponding to different layers have the same geometric forms and different sizes, which may be related to different spatial frequencies in the pattern field. Such interconnection is reported to process high error correction capability.

A study to resolve the important issues like exact recall and capacity in multilayer associative neural network was made [Kang 93]. It is a triple layered neural network consists of two one-shot associative memories as interfaces to both ends of a BAM. The proposed network not only maximizes the total number of stored images but also completely relaxes any code-dependent conditions of the learning pairs.

A hybrid neural network is presented in [Smith 96] which combines both Hopfield neural network and Kohonen's principles of self-organization to solve difficult optimization problems. It is demonstrated that many of the traditional problems associated with each of these approaches can be resolved when they are combined into a hybrid model.

Association performance of Cross-Coupled Hopfield Nets with Many-to-Many Mapping Internetworks (CCHN-MMMI) for correlated patterns is studied in [Ozawa 93]. CCHN-MMMI is composed of plural Hopfield-type neural networks which are mutually connected via multilayered networks. The results show that storage capacity of CCHN-MMMI is greatly increases as compared with that of Hopfield-type associative memory.

Another novel neural network structure in higher levels of abstraction is proposed in [Hilberg 97] to understand or generate natural language texts. This concept leads to a hierarchy of network layers which extract and store local details in every layer and transfer the remaining non local context information to higher levels. At the same time data compression is provided from layer to layer. Possible applications of this network are storage, transmission, understanding, generating and translation of texts.

A Multiple Neural Network associative memory model to overcome the storage limitation of associative memories reported in [Sukhaswami 93,95]. This network is used to recognize printed and handwritten Telugu characters. The network consists multiple number Hopfield networks which are divided into different layers arranged in a hierarchical manner. Possible applications are to automate the sorting of mail, cheque reading etc..

In the present study, a new structure with a new learning rule is proposed.

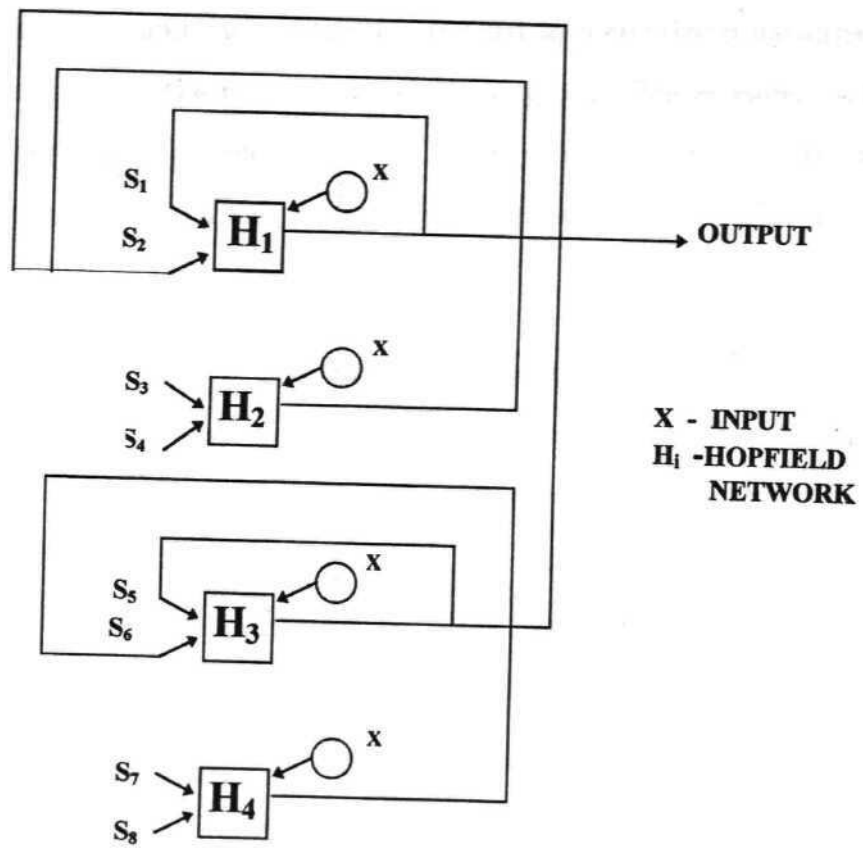
## 2.5 Dynamical Neural Network

To improve the capacity of a associative memory, and to have robust recall a new architecture called Dynamical Neural Network (DNN) is proposed. It is called Dynamical Neural Network in the sense that its architecture gets modified dynamically over time as training progresses. The architecture of the proposed network has a composite structure wherein each node of the network is a Hopfield network by itself. The Hopfield network

employs the new learning technique and converges to user-specified stable states without having any spurious states. The capabilities of the new architecture are as following. The proposed network is a massively parallel and distributed network. It is based on the principle of associative memory but unlike other associative memories it does not allow spurious stable states. It demonstrates a novel idea of order-sensitive learning which gives preference to chronological order of presentation of the exemplar patterns. The proposed network prunes nodes as it progressively carries out associative memory retrieval. It may be noted that pruning is also one of the desired properties of neural computing.

### 2.5.1 The Network Structure

In this section the architecture of Dynamical Neural Network is described. DNN has an composite structure consisting of several basic nodes. The basic nodes are completely connected Hopfield networks and they are similar to each other. In other words, all basic nodes have same number of neurons, but these nodes differ from each other in terms of synaptic weights and thresholds as they memorize different patterns. However, for convenience, in the present work it is assumed that all basic nodes memorize same number of patterns (say  $p$ ). The connections among the basic nodes are shown in Figure 2.1. The underlined idea behind this network structure is the following. If each basic node memorize  $p$  patterns then  $p$  basic nodes are grouped together. When a test pattern is presented to the DNN, assume that it is presented to all the basic nodes simultaneously and all basic nodes are activated at the same time to reach the respective stable states. Within a group of basic nodes one of them is designated as the leader of the group. For simplicity consider the first node as the leader. After the nodes in a group reach the respective stable states these nodes transmit their stable states to the leader in that group. The corresponding connections among the nodes are shown in Figure 2.1 where  $p$  is taken to be 2.



**Figure 2.1 The Architecture of DNN**

At this stage DNN adopts a relative pruning strategy and it returns only the leader of each group and ignores all other basic nodes within a group. In the next pass the DNN consists of lesser number of nodes, but the structure is retained. These leader nodes are treated as basic nodes and each of them is trained to memorize  $p$  patterns corresponding to  $p$  stable states of the member nodes of the group. These leader nodes are again grouped together taking  $p$  nodes at a time. This process is repeated till a single node remains. The process of grouping, pruning, and the leader nodes in the groups are shown in Figure 2.2(a), 2.2(b) and 2.2(c).

It is clear that in the DNN if each basic node memorizes  $p$  patterns then each group memorizes  $p^2$  patterns. Thus one leader node representing a portion of the DNN memorizes  $p^2$  patterns. When  $p$  such leader nodes are grouped together then  $p^3$  patterns can be memorized. If the process runs for  $i$  iterations to arrive at a single basic nodes then the DNN can memorize  $p^i$  patterns. On the other hand if it is required that the DNN to memorize  $K$  patterns, then we must start with  $K/p$  basic nodes.

### 2.5.2 Training of the Network

The DNN described in the previous section can be a very powerful structure for associative memory. It is shown in the subsequent sections that DNN achieves a perfect recall with a substantially large capacity when the network structure of the DNN is coupled with the proposed learning rule. In this section, it is demonstrated that DNN can work as associative memory.

Let us assume that there are  $K$  exemplar patterns that are required to be memorized by the network and when a test pattern is given the DNN as an associative memory model is expected to return the closest of the  $K$  exemplar patterns. These  $K$  patterns are divided

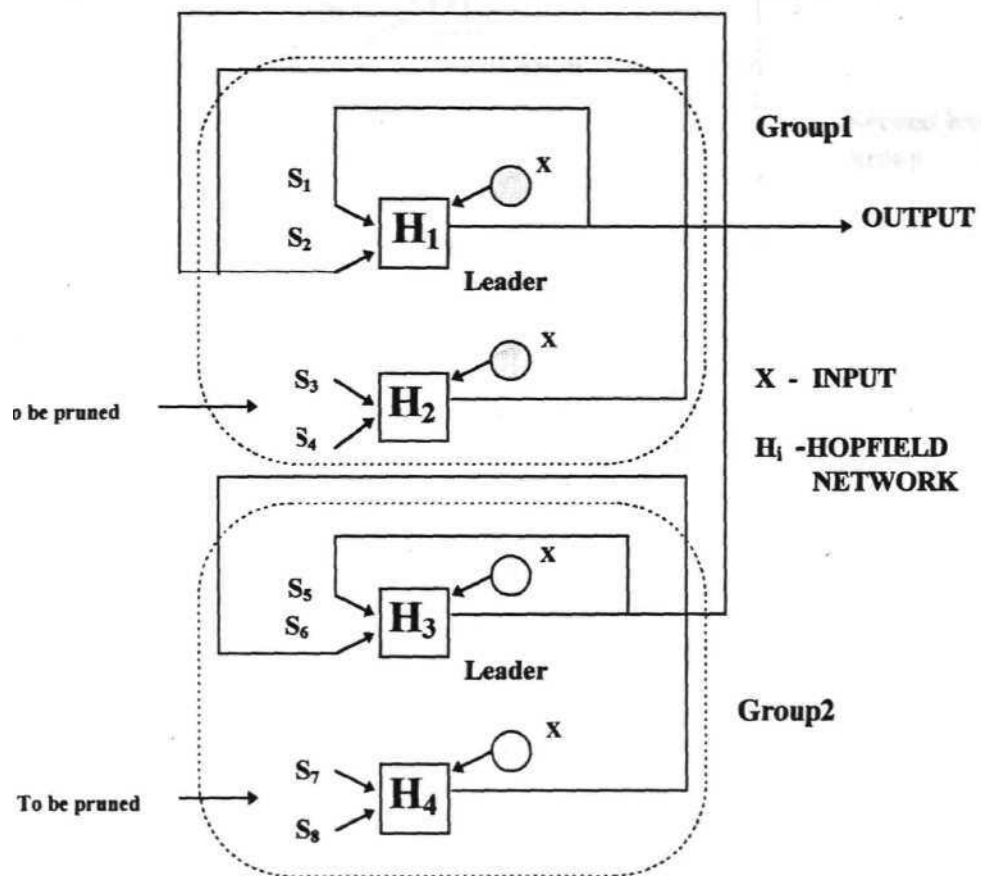
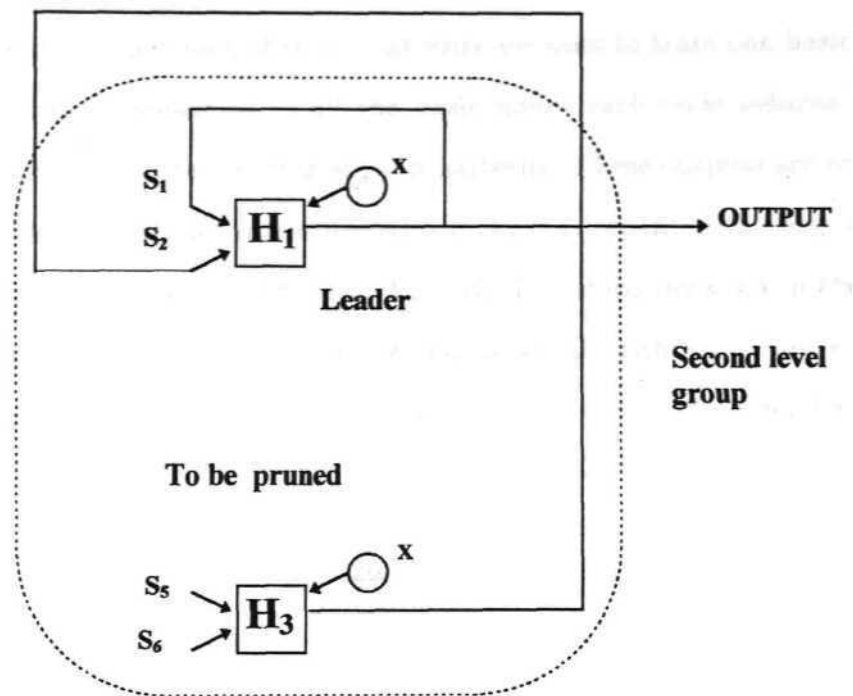
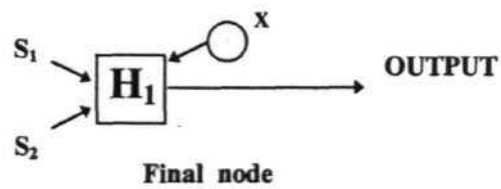


Figure 2.2(a) First Level Grouping and Leader Nodes





**Figure 2.2 (b) Second Level Grouping after Pruning First Level**



**Figure 2.2 (c) Final Node after Pruning Second Level**

in to groups of  $p$  patterns and each of this  $p$  patterns are used to train one basic network. When the test pattern is presented to all the basic nodes each node returns a pattern which is closest among the corresponding set of  $p$  patterns. These outputs are represented as training patterns to the leader nodes and this process is repeated as described above till a pattern closest to the test pattern is returned. The steps involved in the training and pruning of the network when  $p=2$  are given as an algorithm in Figure 2.3. The complete details of the steps (Figure 2.3) in the iterative process of associative memory retrieval are given as below.

Let  $H_i$ , for  $i = 1, \dots, m$  be the basic nodes of the network. Since  $p$  is taken as two, each pair of nodes  $H_i, H_{i+1}$  is considered as one group and  $H_i$  is considered as leader of that group. Let  $S_j$ , for  $j = 1, \dots, K$  be the exemplars to be memorized in the network. Each exemplar pattern  $S_j$  is a binary vector of size  $n$ , where  $n$  is the number of neurons in each basic Hopfield node. These patterns are memorized in DNN that works as associative memory, using a certain recording or learning algorithm. Memorizing the patterns in DNN produces changes in the synaptic weight matrices. Retrieving of the memorized patterns from associative memory, is called as recall. Throughout the retrieving process, the input pattern  $X = (x_1, x_2, \dots, x_n)$  is constantly supplied to all the basic nodes.

In each iteration (steps 5 to 15), two major tasks are performed. Firstly (steps 5 to 10), the function  $train-network(H_i, S_{2i-1}, S_{2i})$  is used to train the basic Hopfield node  $H_i$  with exemplars  $S_{2i-1}, S_{2i}$  to determine the weights and threshold. According to the dynamics of the Hopfield network each  $H_i$  with  $X$  as input converges to one of  $S_{2i-1}, S_{2i}$ . Of course, the learning rule may admit spurious states and the network may converge to a state other than  $S_{2i-1}, S_{2i}$ . The function  $Hopfield(H_i, X, O_i)$  returns the stable state  $O_i$  by  $H_i$ , that corresponds to the input state  $X$ .

**Input:**  $S_1, S_2, \dots, S_{2m}$  *Exemplar patterns*,  $X$  *Test pattern*, ( where  $m$  is the number of basic nodes in the network).

**Output:**  $O$  *Output pattern*.

**procedure** DNN( $S_1, S_2, \dots, S_{2m}, X : in; O : out$ )

1.  $j = 1; k = 2;$
2. *for* ( $i = 1; i \leq m; i = i + j$ ) *do*
3.     constantly present the input pattern  $X$  to each node  $H_i$
4. *end do*
5. *for* ( $i = 1; i \leq m; i = i + j$ ) *do*
6.     train-network( $H_i, S_{2i-1}, S_{2i}$ ) /\* the node  $H_i$  is trained with  $S_{2i-1}$  and  $S_{2i}$  exemplar patterns.\*/
7. *end do*
8. *for* ( $i = 1; i \leq m; i = i + j$ ) *do*
9.     Hopfield( $H_i, X, O_i$ ) /\* each node  $H_i$  stabilizes at a stable state  $O_i$ .\*/
10. *end do*
11. *for* ( $i = 1; i < m; i = i + k$ ) *do*
12.      $S_{2i-1} \leftarrow O_i$
13.      $S_{2i} \leftarrow O_{i+j}$
14.     prune the node  $H_{i+j}$
15. *end do*
16.  $j = j + j; k = k + k; m = m - 1$
17. *repeat* step 5 through step 16 *until*  $m = 1$
18. **END.**

Figure 2.3 The Algorithm for Training and Pruning of DNN

In the second task (steps 11 to 15), the output of the member nodes in each group  $H_i, H_{i+1}$  are feedback to its leader node  $H_i$  so that  $H_i$  is freshly trained with these exemplar patterns. Once  $H_{i+1}$  passes its output to  $H_i$ , it is pruned. In the next iteration, the output of  $H_{i+2}$  is feedback to  $H_i$  and so on. As this process progresses iteratively, in the first iteration  $m/2$  basic nodes are pruned and in the second iteration  $m/4$  nodes are pruned and this finally results in leaving one node in the entire network whose output is the closest pattern to the input pattern. So the proposed network works as associative memory with pruning i.e, at any iteration it can eliminate half the number of basic nodes employed in the previous iteration. The process is illustrated using an example with eight exemplar patterns and depicted in Figure 2.4.

Example 2.1: Let us consider the exemplar patterns as following:

Exemplar patterns are

$$\{S_1 = 11111110$$

$$S_2 = 11111010$$

$$S_3 = 10111001$$

$$S_4 = 00001011$$

$$S_5 = 00011001$$

$$S_6 = 11100000$$

$$S_7 = 00000010$$

$$S_8 = 11011010\}$$

Test pattern  $X = [11111000]$

Since eight exemplar patterns are to be memorized in the network, the network need four basic nodes. That is, each node  $H_i, i = 1, \dots, 4$ , is trained by set  $\{S_{2j+1}, S_{2j+2}\}$



$j = 0, \dots, 3$ , exemplar patterns. Using a learning rule the weight matrices  $W_{ij}, i = 1, \dots, 4$ , of the four basic nodes  $H_i, i = 1, \dots, 4$ , are computed and given in the figure. The test pattern  $X$  is presented to all the basic nodes  $H_i, i = 1, \dots, 4$ . The dynamics of the Hopfield model stabilizes at a stable pattern for each of the basic nodes and each basic node retrieves one of the memorized patterns that is close to the test input pattern  $X$ .

As explained above, outputs of  $H_1$  and  $H_2$  are feedback to  $H_1$  and the outputs of  $H_3$  and  $H_4$  are to  $H_3$ . The nodes  $H_2$  and  $H_4$  are pruned. The nodes  $H_1$  and  $H_3$  are freshly trained. The pattern  $X$  is presented to  $H_1$  and  $H_3$  as test pattern. The dynamics of the Hopfield model stabilizes at a stable pattern for each of the basic nodes  $H_1$  and  $H_3$ . Now these outputs of  $H_1$  and  $H_3$  are feedback to  $H_1$ . The node  $H_3$  is pruned. The node  $H_1$  is freshly trained with feedback patterns and the test pattern  $X$  is presented. Finally the node  $H_1$  retrieves one which in turn the closest pattern to  $X$  among the eight patterns.

The storage capacity of the DNN is  $mp$  where  $m$  is the number of basic networks in the first layer and  $p$  is the capacity of a single basic network. In other words it is also equal to  $K$  i.e the total number of training patterns. The architecture of DNN is feasible for both serial and parallel hardware implementations.

## 2.6 New Learning Rule

The proposed new learning rule is discussed in the following sections.

### 2.6.1 Earlier Learning Rules

There have been several learning rules for Hopfield model and these learning rules differ from each other in adapting different techniques such as Linear programming, Fuzzy logic and Combinatorics and in emphasizing different capabilities of the network such as improving the capacity, computational effects, suitability of VLSI implementation,

avoiding spurious states and providing parallelism. Some of them are [Athithan 95; Verleysen 89; Steven 93; Kam 90; Michel 89; Deiderich 87; Forrest 89; Gardner 89; Abbott 89; Krauth 87].

One of the major disadvantages from which most of the learning rules, including Hebb rule, suffer is the likely convergence to spurious states. The different variations of Hebb rule, such as Diederich and Oppen rule [Deiderich 87], Gardener and Forrest rule [Forrest 89; Gardner 89], Abbott and Kepler rule [Abbott 89] and Krauth and Mezard rule [Krauth 87] also suffer from this drawback. From a practical perspective, it is highly desirable to have a network with a user-specified state vectors as the only stable states and avoid any spurious states. Besides having the spurious states, Hebb rule also has other difficulties of having very limited capacity of  $0.15n$ . Thus for a reasonably large number of patterns to be memorized, it needs an insurmountably large number of neurons in the network. Moreover, two exemplar patterns which are not orthogonal (or nearly orthogonal), it may be difficult to memorize these patterns simultaneously. Keeping these factors in mind, in the present work a new learning rule is proposed for Hopfield network to provide user-specified states as stable states and avoid any spurious states. It also improves the capacity of the Hopfield network substantially and has an order-sensitive learning capability.

It is to be noted that with the proposed learning rule, the basic node can overcome the constraint of limited capacity which is a problem in small size networks. That is, though the theoretical value of the storage capacity of the network is  $2n$ , the practical results of Hopfield [Hopfield 82] yield, this as  $0.15n$  only. This bound holds tight only if all the patterns are mutually orthogonal, which is unlikely for randomly generated patterns. With the present learning rule, even smaller networks can have more storage capacity, which is independent of its size. For example, if the user-specified states are 3, capacity

of the network is 3 for all values of  $n$  and for all randomly generated patterns. Though the capacity of each Hopfield node is two, the overall capacity of the network is  $2m$  where  $m$  is the number of basic nodes in the network.

The learning rule is based on a geometrical interpretation of the set of state vectors and the energy function. The results of polyhedral combinatorics are used for this purpose.

### 2.6.2 Polyhedral Combinatorics

The set of all  $2^n$  state vectors form a binary hypercube in  $n$ -dimensional space and the energy function is a quadratic function defined on the hypercube. Any quadratic function of  $n$  binary variables can be equivalently written as a linear function of  $n(n+1)/2$  binary variables.

Consider the energy function (Eqn. 1.3.2) which is a quadratic function in  $n$  binary variables. Since the state  $x_i = 0$  or  $1$ , by substituting  $x_i^2 = x_i$  and introducing a new variable  $x_{ij} = x_i x_j$ , the equation 1.3.2 can be written as a linear function in  $\frac{n(n+1)}{2}$  binary variables.

$$E = - \sum_{i=1}^n \sum_{j < i} w_{ij} x_{ij} + \sum_{i=1}^n (t_i - \frac{w_{ii}}{2}) x_i \quad (2.6.1)$$

The set of  $2^n$  binary  $n$ -vectors can be equivalently visualized as  $2^n$  binary  $n(n+1)/2$ -vectors which is a proper subset of the unit hypercube of  $n(n+1)/2$  dimension. Putting it formally,

Let  $B_2^n = \{X | X = (x_1, x_2, \dots, x_n), x_i = 0, 1\}$

and define  $a : B_2^n \longrightarrow B_2^{\frac{n(n+1)}{2}}$  such that,

$$\sigma(X) = X = \{x_1, x_2, \dots, x_n, x_{12}, x_{13}, \dots, x_{n-1n}\}$$

Denote  $\bar{B} = \sigma(B_2^n) = \{X : \sigma(X) = \bar{X} \text{ for } X \in B_2^n\}$

Thus  $B \subset B_2^{\frac{n(n+1)}{2}}$



The energy function  $E_x$  for  $X \in B_2^n$  is numerically equal to the linear function  $L(\sigma(X))$  (Eqn. 2.6.1). Finding the minimizing point of a quadratic energy function over  $B_2^n$  is same as finding the minimizing point of linear function  $L$  over  $B_2^{\frac{n(n+1)}{2}}$ . The learning of Hopfield model is a reverse process to that of the above one, where given a set of  $n$ -binary vectors, it is to determine a quadratic energy function which has the given points as minimizing points. In other words, given a set of elements in  $B_2^{\frac{n(n+1)}{2}}$ , it is to determine a linear function which has the specified set of points as minimizing points. If  $C$  is a convex polytope of given set of  $2^n$  points which are its extreme points, this process involves determining a supporting hyperplane which touches  $C$  at the specified set of points. A supporting hyperplane of a polytope is a hyperplane which touches the boundary of the polytope but does not cut across the polytope [Nemhauser 88]. In other words, this hyperplane corresponds to a linear function such that the specified set of extreme points are its minimum points.

As per the design, the hyperplane has to pass through the given stable states only. But it may even pass through some more additional points. These additional points are also extreme points but not the user-specified points. Hence they can be termed as "spurious stable points". It is possible to determine a supporting hyperplane that passes only through the specified extreme points as every extreme point is an adjacent point to every other extreme point [Pujari 83]. Determining the supporting hyperplane with specified points as adjacent extreme points is equivalent to determining the synaptic matrix of the Hopfield network with user-specified states as local minimum points.

### 2.6.3 The Synaptic Matrix

In this section a formulation is proposed for construction of Hopfield network having any two given binary state vectors as the only stable states. This formulation is based on the

concept mentioned above that, for any two vectors  $X$  and  $Y$ , it is possible to construct a supporting hyperplane for convex hull touching it at  $X$  and  $Y$  only.

Considering the above equations (1.3.2) and (2.6.1) the set of  $2^n$  binary  $n$ -vectors  $X$  of the hypercube is now visualized as  $2^n$  binary  $n(n+1)/2$ -vectors  $X$  of the polyhedron and the energy function (Eqn. 2.6.1) defines a hyperplane. It is shown in [Pujari 83] that each of the  $2^n$  binary state vectors is an extreme point convexhull and each pair of extreme points are adjacent. In other words, for any given pair of extreme points, there is a supporting hyperplane which passes through this pair.

Thus, for a given  $X$  and  $K$ , to be as the user-specified stable states of Hopfield network, the process of constructing the weight matrix  $W$  is described below. The Section 2.7 gives the proof that the above Hopfield network  $(W,T)$  has  $X$  and  $Y$  as only local minimum points. Hence  $X$  and  $Y$  are the user-specified stable states of the Hopfield network  $(W,T)$ .

Let  $X, Y$  be the user specified states to be trained by the Hopfield network. The construction of  $W$  using the proposed learning rule is as follows:

Let  $X = (x_1, x_2, \dots, x_n)$  and  $G(X) = \{i : x_i = 1\}$ , Define

$$S_1 = G(X) \cap G(Y), |S_1| = s_1$$

$$S_2 = G(X) \setminus G(Y), |S_2| = s_2$$

$$S_3 = G(Y) \setminus G(X), |S_3| = s_3$$

$$S_4 = \text{otherwise } |S_4| = s_4$$

The synaptic matrix  $W$  is constructed as described below. The diagonal elements are given by

$$w_{ii} = \begin{cases} 1 & \text{if } i \in G(X), \\ 1 - s_3 + s_2/s_3 & \text{if } i \in S_3, \\ -n^3 & \text{Otherwise} \end{cases}$$

and the off-diagonal elements  $w_{ij}$ ,  $i \neq j$  are given by

$$w_{ij} = \begin{cases} 1/2 & \text{if } i, j \in G(X), \\ \frac{s_2}{2s_3} & \text{if } i \in S_1 \text{ and } j \in S_3; \text{ or } i \in S_3 \text{ and } j \in S_1, \\ 1 + \frac{s_2(s_2-1)}{2s_3(s_3-1)} & \text{if } i, j \in S_3, \\ -n^3 & \text{otherwise} \end{cases}$$

The threshold vector  $T$  is given by  $t_i = -0.5 \quad \forall i$ .

## 2.7 Correctness of the Learning Rule

### 2.7.1 User-specified Local Minima

The previous section illustrates the design of a Hopfield network, with two user-specified stable states. In this section, it is proved that these are the only stable state vectors or only local minima, which the network admits.

Let  $Z$  be a state of Hopfield network.  $Z^k$  is a state obtained by changing the  $k^{th}$  bit.  $E_Z, E_{Z^k}$  energy values of the network at  $Z$  and  $Z^k$ , respectively are as follows.

$$E_Z = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij} z_i z_j + \sum_{i=1}^n t_i z_i \quad (2.7.1)$$

and

$$E_{Z^k} = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n w_{ij} z_i^k z_j^k + \sum_{i=1}^n t_i z_i^k \quad (2.7.2)$$

If  $Z^k$  is obtained by changing  $k^{th}$  bit in  $Z$  from 0 to 1. By subtracting equation (2.7.2) from (2.7.1), we get

$$\Delta E = E_Z - E_{Z^k} = \frac{1}{2} w_{kk} + 2 \sum_{i=1, i \neq k}^n w_{ik} z_i - t_k \quad (2.7.3)$$

Similarly, if  $Z^k$  is obtained by changing  $k^{th}$  bit in  $Z$  from 1 to 0 then

$$\Delta E = -\frac{1}{2} w_{kk} + 2 \sum_{i=1, i \neq k}^n w_{ik} z_i + t_k \quad (2.7.4)$$

In order to prove that  $X$  and  $Y$  are the only local minima of energy of Hopfield network  $W(X, Y)$ , it is sufficient to prove that  $\Delta E$  is negative only when  $Z = X$  or  $Z = Y$ .

Similar to the notation of Section 2.6.3, define

$$S_{12} = S_1 \cup S_2 = \{i : X_i = 1\}$$

$$S_{34} = S_3 \cup S_4 = \{i : x_i = 0\}$$

$$S_{13} = S_1 \cup S_3 = \{i : y_i = 1\}$$

$$S_{24} = S_2 \cup S_4 = \{i : y_i = 0\}$$

First, it is shown that,  $X, Y$  are local minima,

Case 1: In order to show that  $X$  is a local minimum, it is to show that

$$\Delta E = E_X - E_{X^k} < 0, \quad \forall k.$$

Depending on  $k$ , consider the following two cases

Case 1a:  $k \notin S_{34}$ , when one of 0 bits of  $X$  is changed to 1 bit to obtain  $X^k$ .

$$\Delta E = \frac{1}{2} w_{kk} + 2 \sum_{i \in S_{12}, k \in S_{34}} w_{ik} - t_k \quad (2.7.5)$$

$$\Delta E = \begin{cases} \frac{1}{2} [w_{kk} + 2 \sum_{i \in S_{12}, k \in S_3} w_{ik}] - t_k, & k \in S_3. \\ \frac{1}{2} [w_{kk} + 2 \sum_{i \in S_{12}, k \in S_4} w_{ik}] - t_k, & k \in S_4. \end{cases}$$

By substituting the values of  $w_{kk}$  and  $w_{ik}$  from Section 2.6.3 in the above equations,

$$\Delta E = \begin{cases} \frac{1}{2} [1 - s_3 + \frac{s_2}{s_3} + 2(s_1 \frac{s_2}{2s_3}) - s_2 n^3] + \frac{1}{2}, & k \in S_3 \\ \frac{1}{2} [-n^3 - 2(s_1 + s_2)n^3] + \frac{1}{2}, & k \in S_4. \end{cases}$$

Since  $n^3$  is the dominating factor and has negative sign, hence the right hand side is negative for either case.

Case 1b:  $k \in S_{12}$ , when one of 1 bit of  $X$  is changed to 0 bit to obtain  $X^k$ .

$$\Delta E = -\frac{1}{2} w_{kk} + 2 \sum_{i \in S_{34}, k \in S_{12}} w_{ik} + t_k \quad (2.7.6)$$

By substituting the values of  $w_{kk}$  and  $w_{ik}$  from Section 2.6.3 in the above equation

$$\Delta E = \begin{cases} -\frac{1}{2} [s_1 + s_2] - \frac{1}{2}, & k \in S_1 \\ -\frac{1}{2} [s_1 + s_2] - \frac{1}{2}, & k \in S_2. \end{cases}$$

$\Delta E$  is negative for either case.

Case 2: Similar to the above case in order to show that  $Y$  is a local minimum, it is to show that

$$\Delta E = E_Y - E_{Y^k} < 0, \forall k.$$

Depending on  $k$ , consider the following two cases

Case 2a:  $fc \in S_{24}$ , when one of 0 bit of  $Y$  is changed to 1 bit to obtain  $Y^k$ .

$$\Delta E = \frac{1}{2} w_{kk} + 2 \sum_{i \in S_{13}, k \in S_{24}} w_{ik} - t_k \quad (2.7.7)$$

$$\Delta E = \begin{cases} \frac{1}{2} [w_{kk} + 2 \sum_{i \in S_{13}, k \in S_2} w_{ik}] - t_k, & k \in S_2. \\ \frac{1}{2} [w_{kk} + 2 \sum_{i \in S_{13}, k \in S_4} w_{ik}] - t_k, & k \in S_4. \end{cases}$$

By substituting the values of  $w_{kk}$  and  $w_{ik}$  from Section 2.6.3 in the above equations

$$\Delta E = \begin{cases} \frac{1}{2} [1 + 2 (\frac{s_1}{2} - s_3 n^3)] + \frac{1}{2}, & k \in S_2 \\ -\frac{1}{2} [-n^3 - 2(s_1 + s_2)n^3] + 1, & fc \in S_4. \end{cases}$$

Since  $n^3$  is the dominating factor and has negative sign, hence the right hand side is negative for either case.

Case 2b:  $k \notin S_{13}$ , when one of 1 bit of  $Y$  is changed to 0 bit to obtain  $Y^k$ .

$$\Delta E = -\frac{1}{2} w_{kk} + 2 \sum_{i \in S_{24}, k \in S_{13}} w_{ik} + t_k \quad (2.7.8)$$

$$\Delta E = \begin{cases} -\frac{1}{2} [1 + 2 (s_3 \frac{s_2}{2s_3} + \frac{s_1-1}{2})] - \frac{1}{2}, & k \in S_1 \\ -\frac{1}{2} [1 - s_3 + \frac{s_2}{s_3} + 2 (\frac{s_2}{2s_3} s_1 + s_3 - 1 (\frac{s_2(s_2-1)}{2s_3(s_3-1)}))] - \frac{1}{2}, & fce5_3 \end{cases}$$

$\Delta E$  is negative for either case.

Next, it is shown that  $X$  and  $Y$  are the only local minima of the Hopfield network  $(W, T)$ . Let  $P$  be a state of Hopfield network such that  $P \neq X$  and  $P \neq Y$ .  $P^k$  is a state obtained from  $P$  by changing the  $k$  th bit of  $P$  from 1 to 0 or 0 to 1. In order to show that  $X$  and  $Y$  are the only local minima it is to show that  $P$  is not a local minima, that is there exists a  $k$  such that

$$\Delta E = E_{P^k} - E_P < 0.$$

Now consider the relations between  $X$ ,  $Y$  and  $P$  as follows:

It may be noted that  $S_1$  is also  $G(X) \cap G(Y) \cap G(P)$ .

$$\begin{aligned} K_1 &= \{i : i \in G(X) \text{ A } i \notin G(Y) \text{ A } i \notin G(P)\}, \quad |K_1| = k_1 \\ K_2 &= \{i : i \in G(X) \text{ A } i \notin G(Y) \text{ A } i \in G(P)\}, \quad |K_2| = k_2 \\ K_3 &= \{i : i \notin G(X) \text{ A } i \in G(Y) \text{ A } i \in G(P)\}, \quad |K_3| = k_3 \\ K_4 &= \{i : i \notin G(X) \text{ A } i \in G(Y) \text{ A } i \notin G(P)\}, \quad |K_4| = k_4 \\ K_5 &= \text{otherwise} \end{aligned}$$

The Hamming distances of  $P$  from  $X$  and  $Y$  are  $k_1 + k_3$ ,  $k_2 + k_4$  respectively and let  $k_1 + k_3 \neq 0$ ,  $k_2 + k_4 \neq 0$ . In order to show that  $P$  is not a local minimum it is to show that

$$\text{If } K_2 \neq 0, k \in K_3, \Delta E = E_{P^k} - E_P < 0$$

$$\text{If } K_3 \neq 0, k \in K_2, \Delta E = E_{P^k} - E_P < 0$$

$$\text{If } K_3 \text{ and } K_2 = 0, k \in K_1, \Delta E = E_{P^k} - E_P < 0$$

$$\text{If } K_3, K_2 \text{ and } K_1 = 0, k \in K_4, \Delta E = E_{P^k} - E_P < 0$$

$$\begin{aligned} \Delta E = & \frac{1}{2} \left[ 1 - s_3 + \frac{s_2}{s_3} + 2 \left( \frac{s_1 s_2}{2 s_3} + k_2 (-n^3) + (k_3 - 1) \left[ 1 + \frac{s_2 (s_2 - 1)}{2 s_3 (s_3 - 1)} \right] \right) \right] + \frac{1}{2}, & \text{fee*} \\ & \frac{1}{2} \left[ 1 + 2 \left( \frac{s_1}{2} + \frac{k_2 - 1}{2} + k_3 (-n^3) \right) \right] + \frac{1}{2}, & \text{feeX}_3 \\ & - \frac{1}{2} \left[ 1 + 2 \frac{s_1 + s_2 - 1}{2} \right] - \frac{1}{2}, & \text{fceATi} \\ & - \frac{1}{2} \left[ 1 + 2 \frac{s_1 + s_2 - 1}{2} \right] - \frac{1}{2}, & k \in K_4 \end{aligned}$$

$\Delta E$  is negative for the four cases.

Hence it is proved that any state, except  $X$  and  $Y$ , can not be the local minimum of Hopfield network  $W(X,Y)$ . Hence the proposed learning rule has only the user-specified states as stable states.

To provide a practical proof for the proposed learning rule experiments were carried out with different values of input vectors of different sizes. The results depicted here show that the proposed learning rule will converge for any input (from  $2^n$  states) to only one of the user-specified stable states. It is also shown that the specified states can be random and realistic values of any size. For example let  $n = 4$  and  $X = 0111$  and  $Y = 1001$ . Now the Hopfield network  $(W,T)$  can be derived using the set of equations mentioned in Section 2.6.3. As input state is varying from 0000 to 1111 i.e for all  $2^4$  states the network converges to any one of the specified states i.e either to  $X$  or  $Y$  only. This phenomena is being depicted in Figure 2.5, where  $X$  and  $Y$  are shown with thick boundaries. The transitions are shown with arrow lines. The energy of the network at that state is shown in brackets. One can observe that the energy of the network is continuously decreasing and reaches a minimal value as the transitions are taking place from initial state to stable state. Here the capacity of the network is two. The *order of updation* used is deterministic and asynchronous (Eqn. 2.2.3). Thus the experimental results are complying with theoretical observations.

### 2.7.2 Recall Efficiency of Learning Rule

In this section it is demonstrated that the learning rule presented in the previous section has better recall capability than that of Hebb rule. It is well known that if we train a network using Hebb rule with two training patterns having smaller Hamming distance,

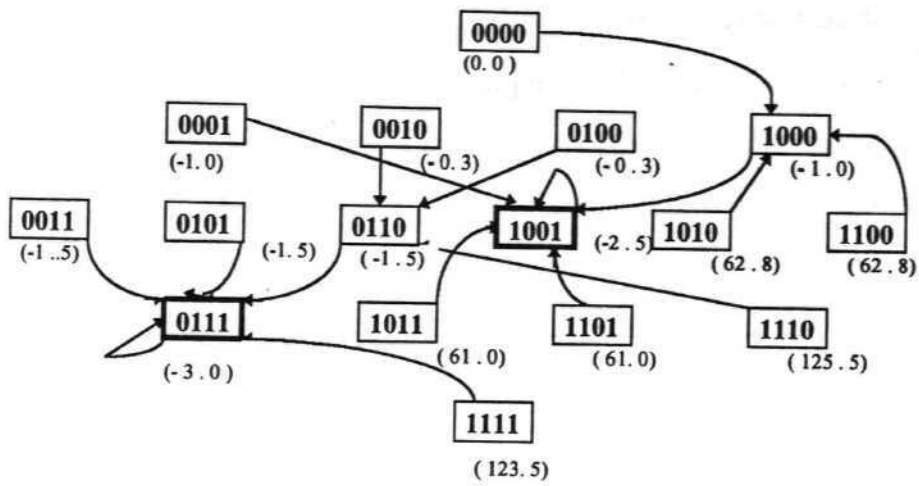


Figure 2.5 Convergence Transition Following the New Learning Rule



the likely-hood of a test patterns converging to spurious states increases substantially. It is shown here that the present learning rule provides a perfect recall capability irrespective of the Hamming distance. In order to demonstrate this, the following experiment was carried out. Two patterns are randomly generated for different values of Hamming distance. The synaptic matrices and threshold are computed using Hebb rule and the present learning rule. By presenting all possible patterns as test patterns the percentage of perfect recall and spurious recall is computed. A recall is said to be a perfect recall if the network converges to one of the training patterns and otherwise it is a spurious recall as it converges to a spurious state. It is observed that even for the smaller Hamming distance the present learning rule results in hundred percent perfect recall whereas, on the contrary, for Hebb rule high percentage of perfect recall is achieved only for higher values of Hamming distance. This observation is depicted graphically in Figure 2.6(a) and 2.6(b), when the patterns are 11-bit length.

## 2.8 The Efficiency of DNN

It is observed that the learning rule has the perfect recall capability independent of the size of the exemplar patterns and Hamming distance between the patterns. The DNN has several basic nodes each having capacity two. If there are  $m$  such nodes DNN has capacity  $2m$ . Since the learning rule can memorize randomly generated patterns of any size without admitting spurious states, eventually the DNN memorizes  $2m$  randomly generated patterns without spurious states. Thus as an associative memory, DNN can overcome all major limitations like low capacity, presence of spurious states, small Hamming distance etc. Hence the DNN works as an ideal associative memory with substantially high capacity. Henceforth the term DNN is associated with both the network structure and the learning rule.

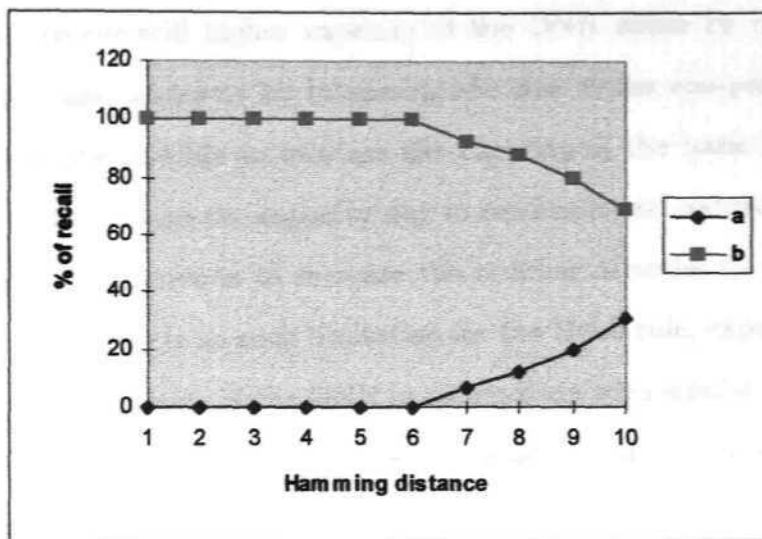


Figure 2. 6(a) Recall Efficiency of Hebbian Rule

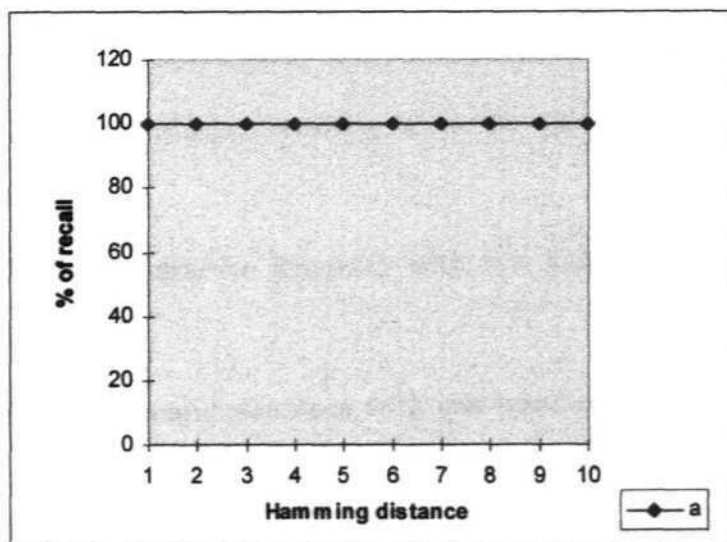


Figure 2. 6(b) Recall Efficiency of New Learning Rule

Graph (a) - Perfect recall      Graph (b) - Spurious recall

One can achieve still higher capacity of the DNN either by increasing the capacity of individual basic nodes or by increasing the size of the composite structure. Though it is theoretically possible to increase the capacity of the basic node, it is impractical to substantially increase the capacity due to combinatorial nature of the learning rule. The only other alternative is to increase the number of nodes in the DNN. Acknowledging the fact that there is no such limitation for the Hebb rule, experimentations are designed to study the efficiency of the DNN in comparison with similar composite structures with the basic nodes (Hopfield nodes) being trained by Hebb rule with two or more than two training patterns.

The efficiency of the DNN with other networks which have similar structures with that of DNN has been compared in the following. For this, three different network structures where all are trained by eight randomly generated patterns with Hebbian rule have been considered.

- i) NET-A: a composite structure with four basic nodes each memorizing two training patterns.
- ii) NET-B: a composite structure with two basic nodes each memorizing four training patterns.
- iii) NET-C: a atomic structure with one basic node which memorizes, all eight training patterns.

The underlying idea of choosing these three structures for comparison with DNN is due to the following:

In these three networks the basic nodes are Hopfield nodes. Even though the Hopfield network with Hebbian rule can memorize  $0.15n$  patterns but for large  $n$  Hopfield network can memorize two patterns easily. Hence with a composite structure the NET-A may

do well. As mentioned that when Hebbian rule is used there is no limit for number of patterns to be trained. In order to learn eight patterns less number of basic nodes is sufficient. In the structure NET-B, two nodes are used to train eight patterns if each node takes four patterns, which is the merit of this structure. In the structure NET-C, since each node can memorize eight patterns only one node is used.

The experimentations conducted on DNN and other network structures NET-A, NET-B and NET-C are described in the following. The relation between  $n$  and percentage of perfect recall for each network structure is depicted graphically.

These four networks are trained with the same eight randomly generated patterns of size  $n=8$ . Four sets of two hundred test patterns of same size are generated randomly. By presenting each test pattern it has been observed the type of recall whether perfect or spurious. That is by presenting 200 randomly generated patterns the percentage of perfect recall is calculated by using the equation

$$\text{percentage of perfect recall} = \frac{\text{no. of. perfect recalls}}{\text{no. of test patterns}} * 100$$

The experimentation is repeated for four sets of 200 patterns and the average performance is computed, which is depicted in the following Table 2.1. The same experiment is repeated by varying the  $n$  between 8 to 60 and each time average percentage of perfect recall is computed. The relation between  $n$  and percentage of perfect recall is shown graphically in Figure 2.7. From the graph it has been observed that DNN has 100% perfect recall for any randomly generated test and training patterns of any size, where as in case of other networks it is continuously increasing starting from a low value and never reaching 100%.

In all the three networks namely NET-A, NET-B and NET-C spurious recalls are present, irrespective of the size of the patterns trained. It is concluded from the graph, that in case

Size of the pattern n = 15		DNN		NET-A		NET-B		NET-C	
Set no.	No. of patterns	No. of perfect recall	% of perfect recall	No. of perfect recall	% of perfect recall	No. of perfect recall	% of perfect recall	No. of perfect recall	% of perfect recall
Set 1	200	200	100.0	24	12.0	40	20.0	4	2.0
Set 2	200	200	100.0	38	19.0	26	13.0	8	4.0
Set 3	200	200	100.0	28	14.0	26	13.0	6	3.0
Set 4	200	200	100.0	38	19.0	24	12.0	12	6.0
Avg.		200	100.0	32	16.0	29	14.5	9.0	4.5
Max			100	38	19.0	40	20.0	12	6.0

TABLE - 2.1

Treshold	Percentage of perfect recall			
	NET-A		NET-B	
-6.0	6.0	1.0	8.0	
-3.0	14.0	4.0	4.0	
-2.0	14.0	20.0	4.0	
-1.0	19.0	20.0	3.0	
-0.5	19.0	20.0	3.0	
0.0	19.0	20.0	3.0	
+0.5	19.0	20.0	4.0	
+1.0	19.0	20.0	1.0	
+2.0	19.0	20.0	1.0	
+3.0	25.0	14.0	0.0	
+6.0	13.0	16.0	3.0	
Max	3.0	25.0	0.5	20.0

TABLE - 2.2

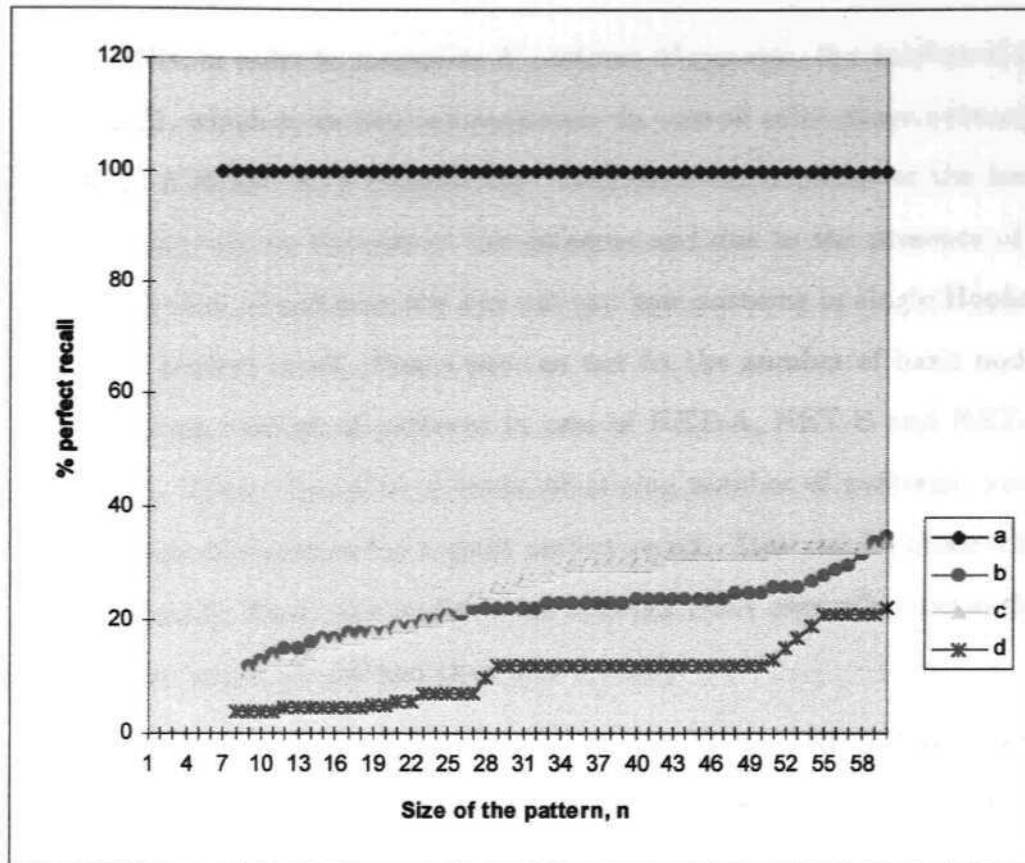


Figure 2.7 Performance Trade-off of NET-A, NET-B, NET-C and DNN

of DNN, in order to memorize  $K$  patterns of any size, the number of basic nodes required is  $K/2$ , which is an easy consequence. In case of other three networks it is theoretically  $K/2$ ,  $K/4$  and  $K/8$  respectively. But, since the capacity of the basic node with Hebb rule depends on the size of the patterns and due to the presence of the spurious states for all sizes of patterns one can not say how patterns in single Hopfield network do have 100% perfect recall. Hence one can not fix the number of basic nodes corresponding to the given number of patterns in case of NET-A, NET-B and NET-C for 100% perfect recall. There should be a trade-off among number of patterns, number of basic nodes and size of patterns for a good perfect recall. This can be observed from the graphs in Figure 2.7. Hence the merit of the network DNN over other networks is that it does not require any trade-off and thus user friendly.

In the above experimentation NET-A, NET-B and NET-C are trained using Hebb rule with fixed threshold. However, the convergence of Hopfield model with Hebb rule depends critically on threshold. In order to fix the optimal value of threshold a next set of experimentations were carried out with variable threshold, in the sense that for every value of  $n$  the threshold value is changed till the best value of recall is attained for each of the network. It is observed for a given value of  $n$ , there may be three different threshold values corresponding to NET-A, NET-B and NET-C. It is observed during the experiments that even the optimal threshold value is given the recall efficiency of NET-A, NET-B and NET-C is much lower than that of DNN. As an illustration, the experimental observations for  $n = 15$  are presented in Table 2.2.

## 2.9 Conclusions

In this chapter, a new neural network architecture, DNN is proposed. The network works as associative memory with 100% perfect retrieval. The Hopfield network (the

basic node in DNN) employs the new learning technique and converges to user-specified stable states without having any spurious states. The survey on multiple neural network architectures and on learning rules is presented. The mathematical tools required for the learning rule (synaptic matrix ) are presented in detail. The correctness of the learning rule is validated with the help of energy function of the network. Experimental results are presented in this chapter, which confirm the betterness of this learning rule in recall over Hebb rule. It also increases the storage capacity of the small size networks substantially and provides a perfect recall capability irrespective of the Hamming distance between the training patterns. Even though the design is shown only for two states the method can be generalized to more number of stable states. The recall efficiency of this Dynamical Neural Network (DNN) is found to be better than other single and multiple neural networks (of same configuration) for some known learning strategies. This architecture can find ready practical applications in the areas of Natural Language Processing, Protein structure database and Library database.



# Chapter 3

## The Special Features of DNN

### 3.1 Introduction

The DNN has some interesting features besides being efficient for associative recall. The DNN architecture exhibits a novel characteristic, termed here as *relative pruning*, wherein the nodes are optionally and dynamically used (and reused) during the process of training. The dynamically organizing features has led us to term the network as DNN. The DNN has a composite structure wherein all nodes are similar and each node of the network is a Hopfield network by itself. During the training all the nodes may not participate constantly. Hence optimal utility of the nodes will definitely improve the efficiency of the network. This enabled us to introduce a new feature in DNN, i.e. *relative pruning* which provides the parallel reusability of the nodes in the DNN.

The learning rule of DNN exhibits another interesting feature of being sensitive to the order of presentation of training patterns. Artificial Neural Networks are biologically inspired and are introduced with the hope to reproduce some of the flexibility and power of human brain by artificial means. Researchers usually think of the functionality of the brain while considering new network configurations and their learning algorithms. Human brain is biased in memorizing and retrieving certain memories. One can find this characteristic in the learning rule of DNN, which is order-sensitive. The natural instinct of human learning is to have longer impression of patterns that are learnt earlier, which

in general represents order-sensitivity. In this chapter, these characteristics of the DNN namely, the *order-sensitivity* and *relative pruning* are discussed.

### 3.2 Order-Sensitive Learning

The human brain continuously learns from environment and memorizes the information, most probably in chronological order. When new memories are stored it may lose the old memories. But this process is not always like a pipeline, there are many exceptions to it. A priority is given to or bias is shown for certain memories, that is certain memories have longer impression and biased while retrieving. Emphasizing the same, the natural instinct of human learning is to have longer impression of patterns that are learnt earlier, "first impression is the best impression".

This biological phenomena is supported by many day to day observations. One can observe while a child is trying to learn words and recognize the words he is biased towards the words he has learnt first. In other words when he has to recognize a misspell word he may be inclined to match it with words he knew in the order he learnt them. For example a child is usually taught the word 'apple' for the letter 'a' first. He might learnt, many more words later starting with the letter 'a' like 'ape', 'ant' and 'ass' etc.. But when he is given a misspell word such as 'aple', he is more inclined to match it with 'apple' rather than with some other word like 'ape'. It should be noted that both 'apple' and 'ape' are equally likely corrections.

Order sensitivity can also be observed when solving Crossword puzzles or Scrambled words. When forming words with the given clues in the puzzle we often try the missing letter in an alphabetical order.

Colour recognition stands another example for order-sensitiveness. A child who is taught

colours in particular order say for example blue, green would correctly classify a true blue object and a true green object. But when an object with an intermediary colour is given he is biased to classify it as blue rather than green.

This clearly shows that human brain may be biased with order of learning things when recognizing ambiguous objects. This prompted us to the development of order-sensitive learning rule. It is emphasized here that order-sensitivity is one of the important features in human learning process and has not yet been explored in the context of artificial neural networks. Moreover, the natural instinct of human learning is to have longer impression of patterns that are learnt earlier.

Putting things more formally, let  $X$  and  $Y$  be two distinct training patterns. A learning rule is said to be *order-sensitive* if the network behaves differently for different order of presentation of the training patterns. In the present context of Hopfield network, it is identified that the learning is order-sensitive, if the learning rule generates different synaptic matrices when  $X$  precedes  $Y$  in the training compared to the matrix when  $X$  follows  $Y$ .

The learning rule, proposed is truly an order-sensitive one in the sense that

- i) the learning rule generates different synaptic matrices for the same set of training patterns presented in different order.
- ii) the basins of attractions are biased towards earlier-learnt-pattern.

Let us consider the following example for illustration.

Let  $P = 11101100001$ ,  $Q = 11010011100$

Let  $W_{PQ}$  be the synaptic matrix computed as per derivation given in Section 2.6.3 considering  $P \rightarrow X$ ,  $Q \rightarrow Y$  and  $W_{QP}$  is synaptic matrix obtained by  $P \rightarrow Y$ ,  $Q \rightarrow X$ .

In other words,  $W_{PQ}$  corresponds to training of the network when  $P$  is presented first, followed by  $Q$  and  $W_{QP}$  corresponds to the training when the patterns are presented in the reverse order. The synaptic matrix  $W_{PQ}$  and  $W_{QP}$  are given below.

$$\begin{aligned}
 WPQ = & \begin{array}{cccccccccccc}
 1.0 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & -n^3 & 0.5 \\
 0.5 & 1.0 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & -n^3 & 0.5 \\
 0.5 & 0.5 & 1.0 & -n^3 & 0.5 & 0.5 & -n^3 & -n^3 & -n^3 & -n^3 & 0.5 \\
 0.5 & 0.5 & -n^3 & -2.0 & -n^3 & -n^3 & 1.5 & 1.5 & 1.5 & -n^3 & -n^3 \\
 0.5 & 0.5 & 0.5 & -n^3 & 1.0 & 0.5 & -n^3 & -n^3 & -n^3 & -n^3 & 0.5 \\
 0.5 & 0.5 & 0.5 & -n^3 & 0.5 & 1.0 & -n^3 & -n^3 & -n^3 & -n^3 & 0.5 \\
 0.5 & 0.5 & -n^3 & 1.5 & -n^3 & -n^3 & -2.0 & 1.5 & 1.5 & -n^3 & -n^3 \\
 0.5 & 0.5 & -n^3 & 1.5 & -n^3 & -n^3 & 1.5 & -2.0 & 1.5 & -n^3 & -n^3 \\
 0.5 & 0.5 & -n^3 & 1.5 & -n^3 & -n^3 & 1.5 & 1.5 & -2.0 & -n^3 & -n^3 \\
 -n^3 & -n^3 & -n^3 & -n^3 & -n^3 & -n^3 & -n^3 & -n^3 & -n^3 & -n^3 & -n^3 \\
 0.5 & 0.5 & 0.5 & -n^3 & 0.5 & 0.5 & -n^3 & -n^3 & -n^3 & -n^3 & 1.0
 \end{array} \\
 WQP = & \begin{array}{cccccccccccc}
 1.0 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & -n^3 & 0.5 \\
 0.5 & 1.0 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & -n^3 & 0.5 \\
 0.5 & 0.5 & -2.0 & -n^3 & 1.5 & 1.5 & -n^3 & -n^3 & -n^3 & -n^3 & 1.5 \\
 0.5 & 0.5 & -n^3 & 1.0 & -n^3 & -n^3 & 0.5 & 0.5 & 0.5 & -n^3 & -n^3 \\
 0.5 & 0.5 & 1.5 & -n^3 & -2.0 & 1.5 & -n^3 & -n^3 & -n^3 & -n^3 & 1.5 \\
 0.5 & 0.5 & 1.5 & -n^3 & 1.5 & -2.0 & -n^3 & -n^3 & -n^3 & -n^3 & 1.5 \\
 0.5 & 0.5 & -n^3 & 0.5 & -n^3 & -n^3 & 1.0 & 0.5 & 0.5 & -n^3 & -n^3 \\
 0.5 & 0.5 & -n^3 & 0.5 & -n^3 & -n^3 & 0.5 & 1.0 & 0.5 & -n^3 & -n^3 \\
 0.5 & 0.5 & -n^3 & 0.5 & -n^3 & -n^3 & 0.5 & 0.5 & 1.0 & -n^3 & -n^3 \\
 -n^3 & -n^3 & -n^3 & -n^3 & -n^3 & -n^3 & -n^3 & -n^3 & -n^3 & -n^3 & -n^3 \\
 0.5 & 0.5 & 1.5 & -n^3 & 1.5 & 1.5 & -n^3 & -n^3 & -n^3 & -n^3 & -2.0
 \end{array}
 \end{aligned}$$

Thus, the learning rule generates different weight matrices and hence yields different behavior of the network. Let us take the input pattern 11000000000 as initial state, the Hopfield network with  $W_{PQ}$  as the synaptic weights, converges to  $P$  as the stable state. Whereas if the same is presented to the network with  $W_{QP}$  as synaptic matrix the stable state to which it converges is  $Q$ . Thus we observe that when 11000000000 is the initial state, the network is biased towards the pattern which it learnt earlier. And also this input state has equi-Hamming-distance from  $P$  and  $Q$  i.e. 6. As explained above in case of Hopfield network with synaptic matrix  $W_{PQ}$  the output state is  $P$  because it is the earlier-learnt-pattern. Similarly in case of  $W_{QP}$  the output state is  $Q$ .

Let  $P, Q$  be two user-specified stable states and  $W_{PQ}$  be the synaptic matrix corresponding to the training of the network when  $P$  is presented first, followed by  $Q$  and  $W_{QP}$  represents the synaptic matrix when the training patterns are presented in the reverse order. Let  $Z$  be the input state which is equidistant from  $P$  and  $Q$ . The learning rule is said to be order-sensitive when  $Z$  converges to  $P$  in case of the network  $W_{PQ}$  and to  $Q$  in case of the network  $W^{QP}$ .

Let  $P = (p_1, p_2, \dots, p_n)$  and  $G(P) = \{i : p_i = 1\}$ . Define

$$\begin{aligned} S_1 &= G(P) \cap G(Q), |S_1| = s_1 \\ S_2 &= G(P) \setminus G(Q), |S_2| = s_2 \\ S_3 &= G(Q) \setminus G(P), |S_3| = s_3 \end{aligned}$$

It may be noted that  $S_1$  is also  $G(P) \cap G(Q) \cap G(Z)$ .

$$\begin{aligned} K_1 &= \{i : i \in G(P) \text{ \& } i \notin G(Q) \text{ \& } i \notin G(Z)\}, |K_1| = k_1 \\ K_2 &= \{i : i \in G(P) \text{ \& } i \notin G(Q) \text{ \& } i \in G(Z)\}, |K_2| = k_2 \\ K_3 &= \{i : i \notin G(P) \text{ \& } i \in G(Q) \text{ \& } i \notin G(Z)\}, |K_3| = k_3 \\ K_4 &= \{i : i \notin G(P) \text{ \& } i \in G(Q) \text{ \& } i \in G(Z)\}, |K_4| = k_4 \\ K_5 &= \text{otherwise} \end{aligned}$$

**Theorem:** The learning rule given in Section 2.6 is order-sensitive in the following sense.

1. For  $W_{PQ}$ ,  $Z$  converges to  $P$  if it is nearer to  $P$  or to  $Q$ . For  $W_{QP}$ ,  $Z$  converges to  $Q$  if it is nearer to  $Q$ .
2. If  $Z$  is equidistant from  $P$  and  $Q$ , for  $W_{PQ}$ ,  $Z$  converges to  $P$  when  $k_2 > k_3$  or  $\frac{s_3+1}{2} \geq k_3$ .

**Proof:** The case 1 is already proved in Section 2.7.

The proof for case 2 is given as following.

$Z$  is equi-Hamming distant from  $P$  and  $Q$  only when

$$k_1 + k_3 = k_2 + k_4.$$

Using the update rule (Eq. 2.2.3),  $Z^k$  is the state obtained from  $Z$  in one transition. If  $Z^k$  is nearer to  $P$  than to  $Q$ , then by the earlier analysis (Section 2.7.1)  $Z^k$  converges to  $P$ , else  $Z^k$  converges to  $Q$ . So it is sufficient to check only one change of transition from  $Z$ . This can be studied by identifying the value of  $k$  such that  $k^{th}$  bit is updated according to the update rule. In case of the network  $W_{PQ}$ , the values  $\Delta E^i = E_{Z^i} - E_Z$  for  $i \in K_1, K_2, K_3$  and  $K_4$  are as follows (Section 2.7.1):

$$\Delta E^i = \begin{cases} -\frac{1}{2} \left[ 1 + 2 \left( \frac{s_1}{2} + \frac{k_2}{2} + k_3(-n^3) \right) \right] - 0.5, & i \in K_1. \\ \frac{1}{2} \left[ 1 + 2 \left( \frac{s_1}{2} + \frac{k_2-1}{2} + k_3(-n^3) \right) \right] + 0.5, & i \in K_2. \\ \frac{1}{2} \left[ 1 - s_3 + \frac{s_2}{2} + 2 \left( \frac{s_1 s_2}{2 s_3} + k_2(-n^3) + (k_3 - 1) \left[ 1 + \frac{s_2(s_2-1)}{2 s_3(s_3-1)} \right] \right) \right] + 0.5, & i \in K_3. \\ -\frac{1}{2} \left[ 1 - s_3 + \frac{s_2}{s_3} + 2 \left( \frac{s_1 s_2}{2 s_3} + k_2(-n^3) + (k_3) \left[ 1 + \frac{s_2(s_2-1)}{2 s_3(s_3-1)} \right] \right) \right] - 0.5, & i \in K_4. \end{cases}$$

So the bit  $k$  that is the candidate of update is

$$\Delta E^k = \min \Delta E^i \quad (3.2.1)$$

Clearly, for  $i \in K_1$  and  $i \in K_4$ ,  $\Delta E^i > 0$  and for  $i \in K_3$  and  $i \in K_2$ ,  $\Delta E^i < 0$ .

Since the dominating factors in the above equations for  $i \in K_3$  and  $i \in K_2$  are  $-$  and  $-k_3 n^3$  respectively, so  $\Delta E^i_{(i \in K_3)} < \Delta E^i_{(i \in K_2)}$  only when  $fc_2 > k_3$ . So the bit  $k$  that is subjected to change in  $Z$  is in  $K_3$ , which results in moving  $Z^k$  nearer to  $P$  than to  $Q$ , and hence the  $Z^k$  converges to  $P$ . So it is proved that in case of the network  $W_{PQ}$  a equi-Hamming distance pattern  $Z$  is converged to  $P$  when  $fc_2 > k_3$ .

Similarly, when  $k_2 = k_3$ , then  $s_2 = 53$  and  $k_1 = fc_4$ , because  $k_1 + k_3 = k_2 + k_4$ . Then it is clear from the above equations that  $\Delta E^i_{(i \in K_3)} < \Delta E^i_{(i \in K_2)}$  only when  $\frac{s_3+1}{2} > k_3$ . So the bit  $k$  that is subjected to change in  $Z$  is in  $K_3$ , which results in moving  $Z^k$  nearer to  $P$  than to  $Q$ , and hence the  $Z^k$  converges to  $P$ . So it is proved that in case of the network  $WPQ$  a equi-Hamming distance pattern  $Z$  is converged to  $P$ .

Similarly in case of  $W_{QP}$ , it can be shown that a equi-Hamming distance pattern  $Z$  is going nearer to  $Q$  in the next transition, when  $k \in K_2$ . In order to show this the synaptic matrix  $W_{QP}$  is used and is given as follows:

$$w_{ii} = \begin{cases} 1 & \text{if } i \in G(P), \\ 1 - s_2 + s_3/s_2 & \text{if } i \in S_2, \\ -n^3 & \text{Otherwise} \end{cases}$$

and the off-diagonal elements  $W_{ij}$ ,  $i \neq j$  are given by

$$w_{ij} = \begin{cases} 1/2 & \text{if } i, j \in G(P), \\ \frac{s_3}{2s_2} & \text{if } i \in S_1, j \in S_2; \text{ or } i \in S_2, j \in S_1, \\ 1 + \frac{s_3(s_3-1)}{2s_2(s_2-1)} & \text{if } i, j \in S_2, \\ -n^3 & \text{otherwise} \end{cases}$$

### 3.3 Experimental Results

While discussing the realistic examples of order-sensitive learning, color recognition is considered as an example. That is, a child who is taught colors in particular order say for example blue, green would correctly classify a true blue object and a true green object. But when an object with an intermediary color is given, the child is biased to classify it as blue (first learnt) rather than green.

Basing on the above phenomenon an experiment is conducted as a practical application to the proposed order-sensitive learning rule. The aim of the experiment is the following: If the two user-specified stable states ( $P$  and  $Q$ ) are represented by two different colors and the input state ( $Z$ ) which is at equi-Hamming distance from the two stable states is represented by a third color then there will be an ambiguity over to which of the first two colors the third color is closer. Then the order-sensitive learning will resolve this ambiguity by matching the third color to the earlier of the first two.

The steps involved in the experiment are: 1. Selecting two patterns randomly that represent  $P$  and  $Q$ . 2. Mapping the  $P$  and  $Q$  (binary numbers) to colors. 3. Generating all the patterns that are at the same hamming distance from the selected two patterns and mapping them to colors. 4. Selecting a color from the colors generated in the step-3 which satisfies our aim (ambiguity over to which of the first two colors the third color is closer).

Mapping of binary pattern to color is achieved by converting binary patterns into RGB values. That is dividing the pattern in to three parts, first part will denote the RED value, second part will denote the BLUE value and the last part will show the GREEN value. The colors are generated by using the "PAINT BRUSH" in MS-Windows. The results are depicted in Figure 3.1.



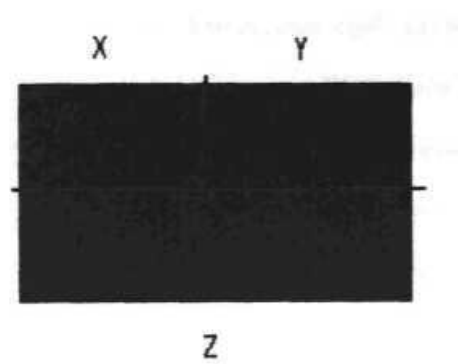


Figure 3.1 Colour Experiment

Hence the order-sensitive property of the learning rule is quite useful in automating the resolving of ambiguity in matching a color which is equally closer to two colors. For this, the experiment is to be conducted in a reverse fashion, that is color is to be mapped to binary patterns and order-sensitive learning rule has to be applied.

### 3.4 Relative Pruning

A major problem in the applications of neural networks is the choice of a topology. A considerable amount of architectures and methods for the construction of optimal neural networks have been proposed in the literature. Some of these approaches try to improve well-known architectures by training a network which is expected to be big enough to solve the given problem and subsequently remove its units (prune units). The topology of DNN is well organized such that the nodes that are not currently participating in the training can be pruned and reused for parallel processing of the next series of data. This pruning and parallel reusing of nodes will improve the efficiency of the network.

Basing on another biological phenomena a new notion called *relative pruning* is introduced in the network. Consider the biological situation wherein, a person being trained to drive a vehicle. In the initial stages of this training, all the units or the neurons in a particular layer are engaged in the processing of information (controlling this phenomenon). After a certain stage, when the trainee has achieved certain efficiency, some of the neurons can be relaxed for this activity. Processing in a similar fashion, we arrive at a situation that some of the neurons or units may be assigned relatively a different job. Thus as far as this information processing is concerned one may think, that these neurons have undergone a sort of *relative pruning*.

This biological phenomena is a motivation for our architecture. In our architecture too, a

concept some what similar to the above, gives a scope for *relative pruning*. In processing a particular task some of the neurons may be assigned relatively a different job and as and when required these neurons may be brought back to this particular task. One may even observe a sort of periodic phenomena as far as the sharing of information among neurons is concerned. Thus by looking at this aspect a new word namely, '*relative pruning*' is introduced.

**Relative pruning:** Relative engagement of neurons for a particular task in a network is called Relative pruning.

Relative pruning is different from the standard notation of *pruning of the network* [Karnin 90; Reed 93; Sieltma 91; Thimm 95] which aims at removing the neurons or weights that are not participating in training and without loss of generality of the training algorithm. On the other hand, in our proposed network, the network gets relatively pruned to the extent that half the number of basic nodes are relieved in each iteration and this finally results in leaving one node in the network. The advantage of this relative pruning is that the network structure employs the hardware most optimally when it is implemented and the relatively pruned nodes can be reused for parallel processing of the next series of data.

To further emphasize the differences between pruning and relative pruning, the concept of pruning is explained first. Pruning methods optimize both size and the generalization capabilities of neural networks. Most of this work is related to feed forward networks. Pruning is done as network trimming within the assumed initial architecture. The network is trimmed by removal of unimportant weights. This can be accomplished by estimating the sensitivity of the total error to the exclusion of each weight in the network [Karnin 90]. The weights which are insensitive to error changes can be discarded after

each step of incremental training. Unimportant neurons can also be removed [Siettema 91]. In either case it is advisable to retain the network with the modified architecture. The trimmed network is of smaller size and is likely to be trained faster than before its trimming.

The concepts of *pruning* in general and *relative pruning* in particular context are explained above. However for sake of completeness the process of *relative pruning* is presented as the following.

DNN has a composite structure consisting of several basic nodes which are similar to each other. It is assumed in the present work that all basic nodes memorize same number of patterns (say  $p$ ). As explained in Chapter 2 we reemphasize that when each basic node memorizes  $p$  patterns then  $p$  basic nodes are grouped together. Within a group of basic nodes designate one of them as the leader of the group. For simplicity consider the first node as the leader. After the nodes in a group reach the respective stable states these nodes transmit their stable states to the leader in that group. At this stage DNN adopts a relative pruning strategy and it retains only the leader of each group and ignores all other basic nodes within a group. In the next pass the DNN consists of lesser number of nodes, but the structure is retained. These leader nodes are treated as basic nodes and each of them is trained to memorize  $p$  patterns corresponding to  $p$  stable states of the member nodes of the group. These leader nodes are again grouped together taking  $p$  nodes at a time. This process is repeated till a single node remains.

Thus in one cycle, the nodes carry out state-transitions in parallel, keeping the weights unchanged and in the next cycle, the nodes communicate among themselves to change the weights. At this stage half of the network is pruned and the remaining half is available for the next iteration of two cycles. In this process, the network eventually converges to

the closest pattern for any given pattern. The Hopfield nodes are connected to each other in a fashion similar to interconnection network and taking advantage of such structure, the active communication between subsets of nodes is determined and unnecessary nodes are pruned. If one can make use of the pruned nodes in the subsequent process, similar to the above mentioned biological phenomenon, the present pruning process becomes a relative pruning. Thus reiterating, one can use the pruned nodes to relatively a different assignment in case of this architecture.

### **3.5 Conclusions**

In this chapter the major characteristics of DNN are introduced. The concept of order-sensitivity is explained with some live examples from children psychology and how this important concept achieved in DNN is explained. The notion of relative pruning and the difference between the concepts of pruning and relative pruning are explained. The need of relative pruning is strengthened with examples from biology. Thus this chapter adds strength to the need of introducing a new architecture.

# Chapter 4

## Close Proximity Match for Large Databases Using DNN

### 4.1 Introduction

Retrieving information from a very large database using approximate queries has been of prime interest of researchers in the recent years. The studies in this area particularly for unformatted databases like library databases, protein databases and image databases have been investigated with special considerations. And methods like Full text scanning, Inversion, Clustering and Signature files are reported [Faloutsos 85]. In unformatted databases one of the important features is that the user tends to retrieve information based on approximate queries, which are quite common in practice and sometimes unintentional. For example, in the case of library database a user may not be sure of the exact set of keywords to retrieve a specific book, but may have an idea of the subjects that the book deals with. In such cases, query keywords may not match with all the keywords stored in the database. In another case the user may be aware of the keywords but uses them in the abbreviated form or with different spelling, with a spelling mistake or typographical mistakes. In the protein sequence database, determining approximate match is required to discover the relationships between newly sequenced protein that resulted due to evolution and the various classes of proteins already available in the database. Thus close proximity match in such large databases is one of the very important operations

<sup>1</sup>The contents of this chapter appeared in the Proceedings of the Eighth International Workshop on DEXA, France, IEEE Computer Society Press, 1997, with the title " A new neural network architecture for efficient close proximity match of large databases"

and there are many research reports in this context [Hall 80; Shang 96].

Associative memory concept is related to the association of stored information for given input patterns. The distinct traits of ANN-based implementations as associative memory are their use of content-based retrieval instead of address based retrieval of stored data, approximate instead of exact matching. Hence the present task is represented as problem of association and can be accomplished by associative memory. The DNN is more suitable to accomplish the present task as it is having the properties like associative memory with 100% perfect recall, large storage capacity, avoiding spurious states and converging only to user specified states.

Since the DNN operates with binary data while the information in library database and protein database are in textual form, it is required to map the textual data to a binary string called *signature*. The network is not trained by the actual data, rather by signatures which are obtained by superimposed coding. These signatures are used as exemplar patterns to store them in the DNN by training. The signature of the query is presented to the network as a test pattern. The associative memory property of the network makes it to recall a closest pattern of the given query pattern among the memorized patterns. The paradigm is tested on two different databases having totally different requirements, viz., library retrieval systems and protein databases. The experimental results are reported to corroborate that high level of precision can be obtained for efficient recall of inexact queries using the proposed neural network.

## 4.2 Text Retrieval Methods

An information retrieval system is concerned with the representation, storage, organization and accessing of information items. In principle no restriction is placed on the

type of items handled in information retrieval. But retrieval systems are normally used to handle textual data. Hence in the present context we would like to describe the text retrieval problem, whose main concern is with the type of queries and the operational requirements. For the sake of completeness, a brief overview of some of the general methods used in text retrieval is presented below.

Faloutsos [Faloutsos 85] has presented a review of access methods applicable to both formatted and unformatted data. He classifies text retrieval methods into the following four classes:

(1) *Full Text Scanning* [Knuth 77; Hollaar 83]: In this method the full text database is scanned for matching records. No extra storage overheads are incurred, but the method is relatively slow for large databases.

(2) *Inversion* [Lesk 78]: In this method each document can be represented by a list of (key) words, which describe the contents of the document for retrieval purpose. Fast retrieval can be achieved if we invert the keywords. This method uses an inverted file index, where key words are stored alphabetically. It is implemented in many commercial text retrieval systems. It provides relatively efficient query speeds, but can be very expensive of storage. In addition another disadvantage is that the insertion times are slow in this method.

(3) *Clustering* [Salton 83]: In this method, similar documents are grouped together to form clusters. Clustered documents can be stored physically together, facilitating efficient retrieval of related documents. A descriptor is stored for each cluster and a search correlates, typically using a vector similarity function, these descriptors with the query descriptor to retrieve relevant documents. The main disadvantage of this method is the slow insertion time.



(4) *Signature Files* [Tsichritzis 83; Larson 83]: In this method, a descriptor or a signature is associated with each record of document, the descriptor being a bit encoding of the values used to retrieve the record. When a query is processed, the file of descriptors, rather than the data records, is examined for possible matches. A query descriptor is formed using the same encoding technique that is used for forming record descriptors. The possible record matches are those records whose descriptors contain bits set in each position for which a bit is set in the query descriptor.

The signature file approach seems most promising for large databases. Signature file methods have good retrieval properties and require small storage overheads [Sacks 87]. Notice that there is no restriction on the number of key words for any document, because documents can be divided into "logical blocks" (pieces of text that contain a constant number of key words). In other words the advantages are simplicity of implementation, efficiency in handling insertions, ability to handle queries on parts of words, ability to support a growing file and tolerance for typographical and spelling errors [Faloutsos 85]. Since DNN uses only a binary data and in view of the above advantages, in the present context signature file method is more suitable.

### 4.3 Signature

As mentioned in the previous section, signature file method is one of the efficient methods for text retrieval. In this section, the method of computing signature by superimposed coding technique is discussed.

The basic networks in DNN are trained by the signatures of the records, rather than the actual records. The signatures are obtained by superimposed coding technique (Section 4.3.1.1). The signatures of the records in the database are used as exemplar patterns

for DNN and the signature of the query is used as a test pattern. The associative memory property of the network makes it to recall one of the closest pattern of the given query pattern among the memorized patterns.

#### 4.3.1 Signature Extraction Methods

As mentioned before, the signature file approach seems to be a promising method for text retrieval. In this section, some methods of signature extraction are described. They are *Word signature method*, *Compression technique* and *Superimposed coding technique*.

*Word signature method [Tsichritzis 83]:* In this method, each word of the document is hashed into a bit pattern of length  $n$ . These patterns (word signatures) are concatenated to form the document signature.

*Compression technique [McIlroy 82]:* In this method the document is split into logical blocks (a piece of text that contains a constant number of words). The idea is to use a (large) bit vector of zero's of size  $n$  for each block and hash that block into one bit positions, which are set to 1. The resulting bit vector is sparse and can therefore be compressed. Then the resulting bit vectors are logically ORed to get the signature of the document.

In the present context the superimposed coding technique for computing the signature is used because it gives good retrieval performance and efficient storage [Faloutsos 85, Rabbitt 84]. In this method all signatures are of same size, which is one of the requirements of DNN.

##### 4.3.1.1 Superimposed Coding Technique

Superimposed coding technique [Christodoulakis 84] is similar to compression technique but each word is mapped into an individual signature. For the sake of completeness, certain terms used in the present work are explained as follows. In any database, *Keyword*

is the basic component and *Record* is a collection of individual components (keyword-s). *Signatures of Keyword, Record and Query* are encoded binary representations, which characterize the essence of them. In case of Library database, record is a document and individual component is a keyword. Using the superimposed coding technique, the signature for a record is computed by superimposing the signatures of its individual components. If the keyword consists of more than one word, the signature of the keyword is superimposing of the signatures of the words. The signature of the keyword is obtained by hashing technique.

Input:  $n$  size of the signature,  $r$  number of bits set to 1,  $W[]$  word whose signature is to be computed.

Output:  $t$  signature of  $W[]$ .

procedure Hash ( $n, r, w[]$ : in;  $t$ :out)

1.  $H(w) = 0$ ;  $/ =$  length of the word  $W[]$ ;  $p = {}^nC_r$ ;
2. for  $i = 1$  to 1 do
3.    $H(w) = H(w) * 2 + \text{ASCII}(w[i])$ ;
4. end do
5.  $t = H(w) \bmod p$ ;
6. End.

Figure 4.1 The Algorithm for Hashing Method

#### 4.3.1.2 Hashing Method

In order to get the signature of a keyword hashing technique is used [Knuth 73; Knott 75]. Let the signature be a  $n$  bit pattern, in which  $r$  bits are set to 1. Then it is one among the  ${}^nC_r$  bit patterns that can be generated using  $n$  bits in which  $r$  bits are set to 1. The hashing function  $H(w)$  maps the keyword into one of the above patterns generated by

computing a hash value of the keyword. The hash function uses shift and add strategy. The ASCII values of the characters in the keyword are added and shifted by  $H(w)$  in order to compute the hash value. The final hash value is obtained by module operation with  ${}^nC_r$ .

To further ease the understanding of the hashing method, steps involved in it are compiled in the form an algorithm and is presented in the above Figure 4.1. The step 3 in Figure 4.1, denotes the shift and add strategy in calculating the hash value.

The following example illustrates the above algorithm. Let us suppose that  $n = 4$  and  $r = 2$ . Then all the possible ( ${}^4C_2$ ) combinations are:

Hash Value	Signature
0	0011
1	0101
2	0110
3	1001
4	1010
5	1100

The signature for the word "DATA" is calculated as follows :

The ASCII (D) = 68

The ASCII (A) = 65

The ASCII (T) = 84

Initially  $H(w) = 0$

$$H(w) = 0 * 2 + \text{ASCII (D)} = 0 * 2 + 68 = 68$$

$$H(w) = 68 * 2 + \text{ASCII (A)} = 136 + 65 = 201$$

$$H(w) = 201 * 2 + \text{ASCII (T)} = 402 + 84 = 486$$

$$H(w) = 486 * 2 + \text{ASCII (A)} = 972 + 65 = 1037$$

The table size is  ${}^nC_r$ , here  $n = 4$ ,  $r = 2$ . So table size =  ${}^4C_2 = 6$

The Hash value of "DATA" =  $1037 \bmod 6 = 5$

So the signature of the word "DATA" = 1100.

Now the computational steps involved in the superimposed coding technique are presented as an algorithm in Figure 4.2.

Input:  $D$  document consists of  $k$  words  $w_1, w_2, \dots, w_k$ .

Output:  $S$  signature of the document.

procedure superimposed-coding( $D : \text{in}; S : \text{out}$ )

1. *for*  $i = 1$  to  $k$  *do*
2.    $t_i \leftarrow \text{Hash}(n, r, w_{i[]})$ ;
3. *end do*
4.  $S = t_1 \vee t_2 \vee \dots \vee t_k$
5. *End.*

Figure 4.2 The Algorithm for Superimposed Coding Technique

An example of this process is given in Figure 4.3

Example 4.1: If a book by title "Computer Applications " consists of three keywords, 1. *computers* 2. *applications* 3. *mathematics*. If  $n = 12$  and  $r = 4$  the signatures of the keywords are as shown in the Figure 4.3. The signature of the book is obtained by superimposing the signatures of the keywords with OR operation.

<i>computer</i>	1100	1000	0100
<i>applications</i>	0001	0101	0100
<i>mathematics</i>	0011	0001	1001
signature of the book	1111	1101	1101

Figure 4.3 Example of Superimposed Coding

In the similar line, the signature for a protein sequence is obtained by dividing the protein sequence string into blocks of fixed size of Amino acid residues. For example a protein sequence **FLGRVDNPQNW** is divided into four blocks each of size three as **FLG**, **RVD**, **NPQ**, **NW**. For each block, the signature is computed and then these are superimposed to get the signature of the protein sequence.

It is possible that a record signature matches a query signature but the corresponding record does not satisfy the query. Such occurrence is referred to as false drop. The probability of a false drop can be made arbitrarily small by appropriate choice of the size of the signature.

#### 4.4 Library Retrieval System

The need to find an approximate match to a string arises in many practical problems. A Library database is such a system wherein documents are stored as a set of keywords that describes the content of the document. The query from a user consists a set of keywords which may or may not match exactly with that of the document. Sometimes the user may be aware of the keywords but uses them in an abbreviated form or may use with some typographical errors. In this context even if the query from the user matches partially or fully with a stored pattern, the information related to that should be retrieved.

The library retrieval system uses the signature file method. In the signature file method a signature or descriptor is associated with each record in the database. The signature is a bit encoding of the values used to retrieve the record. The signature of the record is obtained by using superimposed coding technique. These signatures are memorized in DNN by training. A query signature is formed using the same encoding technique that is used for forming record signature. The DNN retrieves the signature of the document

which approximately matches with that of the query.

The computational steps involved in the library database system are given as an algorithm and is presented in Figure 4.4.

Input:  $L$  Library database consists of documents  $D_1, D_2, \dots, D_m$   $Q$  query.

Output:  $B$  book corresponding to query  $Q$

procedure Library ( $D_1, D_2, \dots, D_m, Q : in; B : out$ )

1. for  $i = 1$  to  $m$  do
2.    $S_i = \text{superimposed-coding}(D_i)$
3. end do
4.  $X = \text{superimposed-coding}(Q)$
5.  $O = Dm(S_1, S_2, \dots, S_m, X)$
6. Look up in Library database  $L$  for a book  $B$  (document) whose signature matches with  $O$ .
7. End.

Figure 4.4 The Close Proximity Match Algorithm for Library Database

The following example illustrates the network behaviour for approximate retrieval in case of Library database.

#### Example 4.2

Let us consider a sample library database consists of eight books. Each book in the library database has on average three keywords. The keywords of eight books are given below:

Book1: computer networks, computer architecture, parallel processing.

Book2: information theory, artificial intelligence, linguistics, data processing, semantics.

Book3: proceedings, computer network, usage experiences.

Book4: computer networks, queuing theory, performance evaluation.

Book5: computer networks, protocols, standards and interfaces.

BookG: data transmission systems, distributed computer networks.

Book7: electronic data processing, distributed processing, computer networks.

Book8: computer science, artificial intelligence, lecture notes.

The signature of the book,  $Book_i$  is given below as  $S_i$ :

$S_1 = 111110100001000100000011101010$

$S_2 = 111100010100000101011000101111$

$S_3 = 111001100001110000100010111111$

$S_4 = 111100101000110100000111111111$

$S_5 = 111101100000000000000011001111$

$S_6 = 011111110000101111100011111011$

$S_7 = 111100110000101101000011101011$

$S_8 = 111100101000000100111111011011$

Let us assume that the user wants to retrieve a book based on the keywords *computer architecture*, *parallel processing*. Thus the signature of the query  $X$  is computed by superimposing the signatures of both the keywords.

$X = 111110100001000100000010101010.$

The step by step process of obtaining the closest pattern to the test pattern from the library database is explained with the help of Figures 4.5(a) to 4.5(f). Initially the signatures of the records in database are presented as initial states or training patterns to the basic nodes. Let us consider that each basic node can memorize two patterns. Since the sample database has eight records, to memorize them the network need four basic nodes. That is, each node  $H_i$ ,  $i = 1, \dots, 4$ , is trained by set  $\{S_{2j+1}, S_{2j+2}\}$ ,  $j = 0, \dots, 3$ , exemplar patterns. The query pattern  $X$  is presented to all the basic nodes  $H_i$ ,  $i = 1, \dots, 4$  [Figure 4.5(a)].

The dynamics of the Hopfield model stabilizes at a stable pattern for each of the basic nodes  $H_i$ ,  $i = 1, \dots, 4$ . Each basic node retrieves one of the memorized patterns that is close to the test input pattern  $X$  [Figure 4.5(b)]. In the present example the outputs of



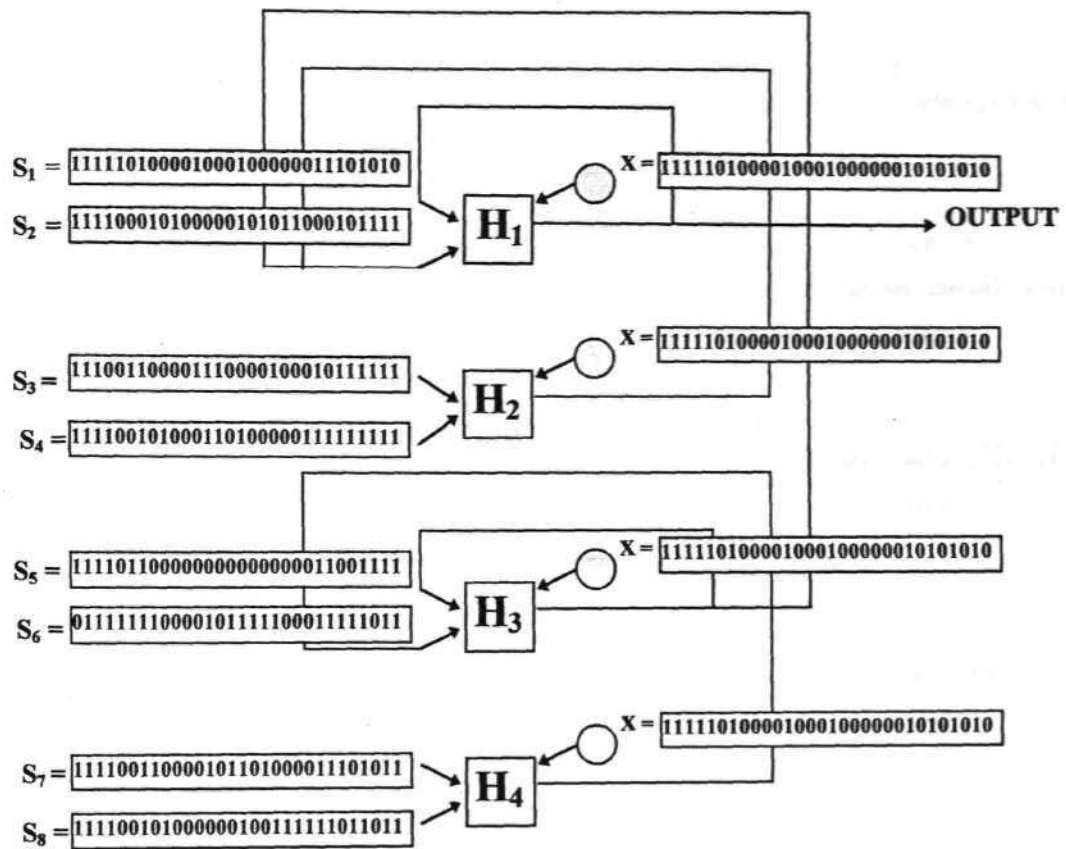


Figure 4.5 (a) Initiation with Training and Test Patterns

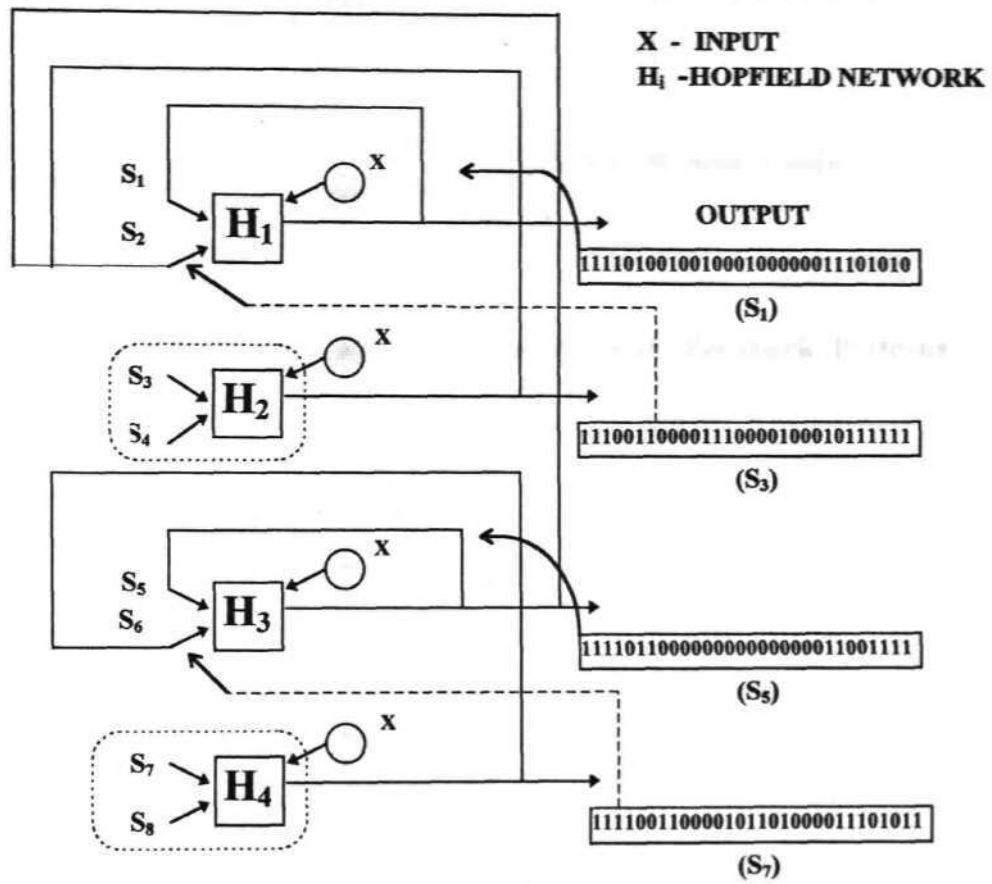


Figure 4.5(b) Hopfield Stable States, Pruning and Feedback (Phase 1)

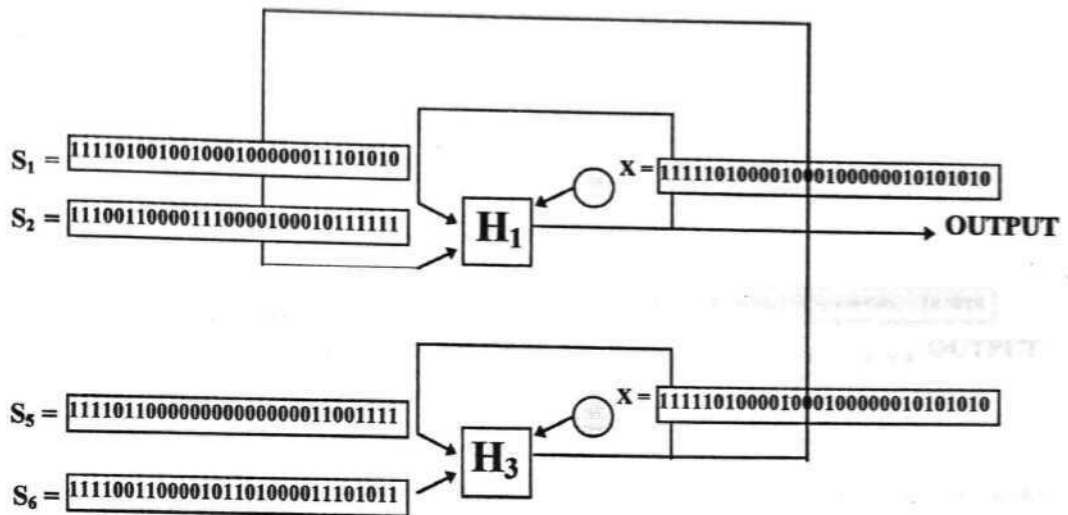


Figure 4.5(c) Training of Phase-1 Pruned Network, with Feedback Patterns

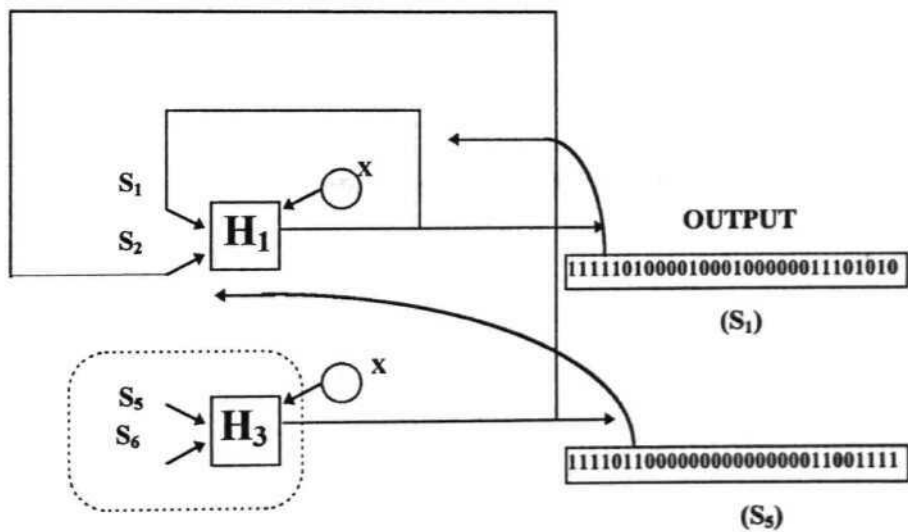


Figure 4.5(d) Stable States of the Phase-1 Pruned Network (Phase 2)

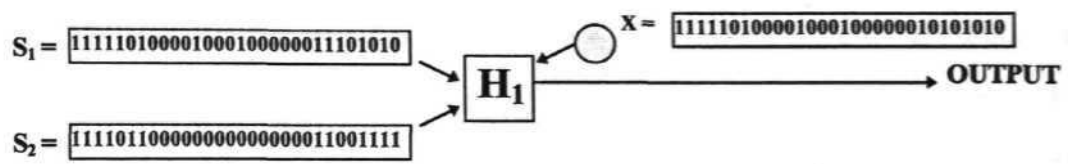


Figure 4.5(e) Training of Phase-2 Pruned Network with Feedback Patterns

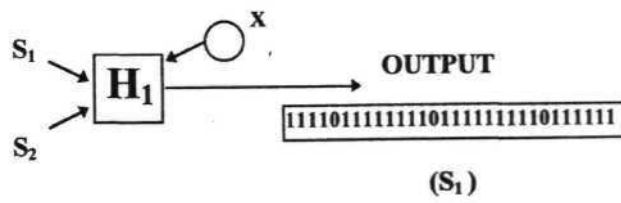


Figure 4.5(l) Final Output Pattern

$H_1$ ,  $H_2$ ,  $H_3$  and  $H_4$  are  $S_1$ ,  $S_3$ ,  $S_5$  and  $S_7$  respectively. As explained in Section 2.5.2 the outputs of  $H_1$  and  $H_2$  are feedback to  $H_1$  and the outputs of  $H_3$  and  $H_4$  are to  $H_3$ . The nodes  $H_2$  and  $H_4$  are pruned [Figure 4.5(b)].

Now the nodes  $H_1$  and  $H_2$  are freshly trained with the feedback patterns. The query pattern  $X$  is presented as test pattern [Figure 4.5(c)]. Similar to the Figure 4.5(b) the dynamics of the Hopfield model stabilizes at a stable pattern for each of the basic nodes  $H_1$  and  $H_3$  [Figure 4.5(d)]. The outputs of  $H_1$  and  $H_3$  are  $S_1$  and  $S_5$  respectively. These  $S_1, S_5$  are feedback to  $H_1$ . The node  $H_3$  is pruned. The node  $H_1$  is freshly trained with patterns  $S_1$ ,  $S_5$  and the test pattern  $X$  is presented [Figure 4.5(e)]. Finally the node  $H^l$  retrieves one among  $S_1$  and  $S_5$  which in turn closest pattern to  $X$  in the whole database [Figure 4.5(f)]. In the present case it is  $S_1 = 111110100001000100000011101010$ . This is the signature of the book namely *computer networks, computer architecture, parallel processing* which is required.

Due to false drops it is possible that record signature matches a query signature but the corresponding record does not satisfy the query. It may degrade the performance of DNN. Solution to this is discussed in Section 4.6.

#### 4.4.1 Experimental Results of Library Retrieval System

The performance of the DNN critically depends on the efficiency of signature computation and the efficiency of signature computation is also depends on the appropriate choice of two parameters  $n$  and  $r$ . In the event of occurrence of false drop, it may happen that two distinct records may correspond to same signature. It is observed that for appropriate values of  $n$  and  $r$ , false drop can be avoided. Keeping the foregoing discussion in view experiments were carried out to evaluate the performance of DNN in the context of library retrieval. The aim is to study the accuracy of retrieval for varying values of  $n$ .

For the library application, the collection of books of University of Hyderabad library is considered for a case study. This library has over 2 lakh volumes. In this case study, the search is domain based in the sense that the collection of books falls in the domain of Computer Science. A part of this domain consisting of 200 samples of records is considered and each record on average consists of 3 keywords. The signature for each keyword is computed using hashing technique. The signature of the text book is determined by superimposing these signatures.

These signatures are memorized in DNN, using them as training patterns. The process of close proximity match starts when the signature of a query is presented as test pattern. The experiment was carried out for three cases depending on the number of query keywords. The first case (case i) deals with the case in which one keyword differ (even deleted) from the input record, the second case (case ii) deals with that of two keywords differing in the above sense and the third case (case iii) deals with when there is no change in case of keywords from the input record.

Experiments were carried out by varying the values of N (size of the signature) from 8 to 50. The percentage of retrieval is calculated in each case. The relations between percentage of retrieval and signature size are depicted graphically. The graphs (a), (b) and (c) in Figure 4.6 represents the cases (iii), (i) and (ii) respectively. The formula for percentage of retrieval is as follows:

$$\text{percentage of retrieval} = \frac{\text{no. of accurate retrievals}}{\text{no. of input queries}} * 100$$

Observations based on the graphical relations are as follows:

(1) In the initial range of N between 8 and 20, the growth in the percentage of retrieval is phenomenal, though with varying degrees in cases (i), (ii) and (iii). This is due to the decrease in the false drop as signature size is increasing.

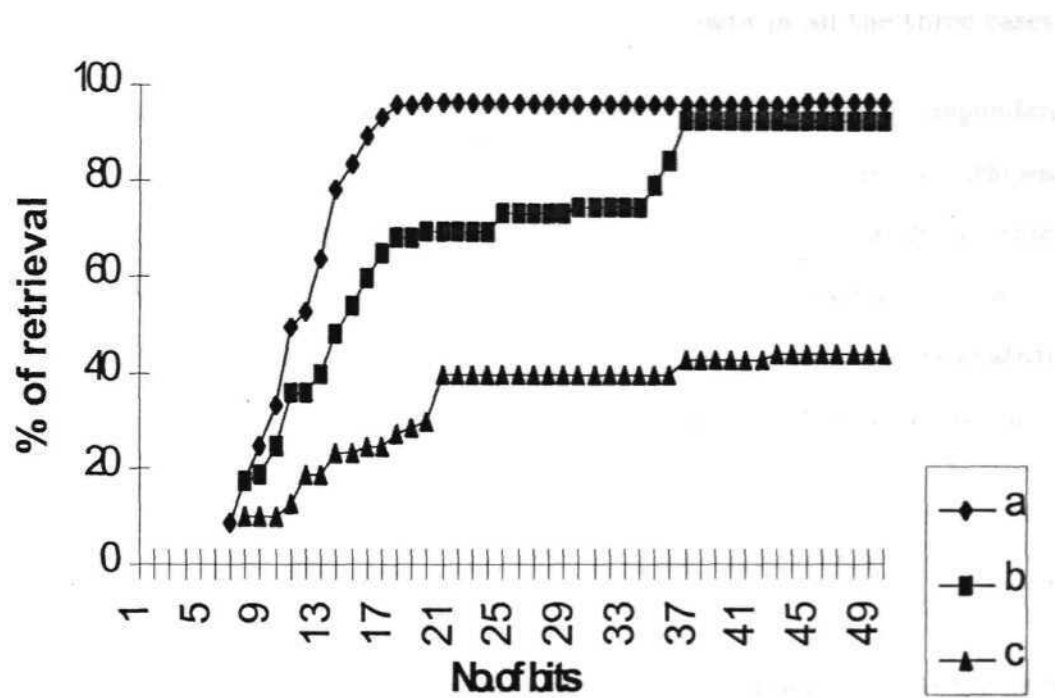


Figure 4.6 Library Retrieval System

(2) Further increase in the values of  $N$  in all cases is causing only marginal changes in the growth of percentage of retrieval. In this stage the retrieval rate has reached the stability as false drop can not be decreased further.

(3) The percentage of retrieval approaches a constant with the further increase of  $N$  and there is an optimum value for  $N$  beyond which there is no growth in all the three cases.

Based on the observations, it is concluded that the percentage of retrieval is dependent on the signature size. Hence it is concluded that the DNN can perform an efficient approximate search if appropriate technique can be developed to find the signature. Since the present study aims at establishing the suitability of neural network for retrieval, no attempt is made here to investigate the most efficient method for computing the signature. However, if proper signature computation scheme is employed the retrieval performance is believed to be improved.

#### 4.5 Protein Database

Molecular Biologists are quite often required to search for the similarities between newly determined amino acid sequences and sequences already available in the databases. This has become more useful because of the availability of large protein databases and increasing relationships that have been discovered between newly sequenced protein and various classes of proteins in the databases. There have been many research reports in the context of protein sequence similarity search [Altschul 94; Cantalloube 95]. The DNN is used to perform a sensitive similarity search. Sensitive similarity search is required because new protein sequences are resulted from already existing protein sequences due to substitution, insertion and deletion of amino acids.

The process involves i) dividing the protein sequence string into blocks of fixed size



amino acid residues (ii) the protein sequence string is converted into binary signature by using superimposed coding technique that results in a binary sequence database. The signature of the query consists of unknown protein sequence is presented as test pattern. As explained above the network retrieves a protein sequence similar or nearest to the query from the database.

It is to be noted that since protein sequence length is varying approximately from 3 to 5000 amino acid residues, but the signature size is constant for all records, probability of false drop increases. In general amino acid replacements occur more frequently than insertions and deletions. So any similar pairs of protein sequence may not have a drastic difference in length. Keeping this in view, the database is divided into smaller sets according to length so that the number of records in the search reduces and so is false drop. As an example, all the sequences whose length is from 30 to 50 (with  $\pm 10\%$  variation) are placed in one set. The following examples will clearly illustrate the approximate search with the network on protein database.

### Example 4.3

Consider a sample database of protein sequences of 1-50 length:

- VAP FPH GKL VTY KYI ADV KAG VDP SLV
- MKF GLF FQN FLS ENQ SSE
- IKD
- . SKM KKCEFA KIA KEQHMDGYHGVSLADWVCLVNNESDFNTKA INRNKGI
- . MKKAKAIFL FILIVSGFLLVACQANYIRDVQGGTVAPSSSSELTGIAVQ
- . MKKITG IIL LLLAVIILS ACQANYIRDVQGGTVSPSSTAEVTGLATQ
- AIG YQI YVR SFR DGN LDG VGD FR
- . MKKAWWKEGVVYQIY
- . SNQAKADAYKEAFQNGWGAXAVDALXTAVIMGV LHPEVIESFY
- . ZPFATMRYPSDDSE

The signatures of the records in the database are as follows.

- 000101111000000111110110000101

- 000101001010100000101100000101
- 00000000100000000000100000101
- 010111111000000011111110010111
- 111001111010101010111111100111
- 10001111011011111101101000111
- 011101010000010101111110000111
- 000101000100000011101100001111
- 101000100111110101111111100111
- 001000100000010001001100000111

As mentioned in the beginning a new protein sequence is resulted due to evolution. In other words a new protein sequence is resulted due to substitution or insertion or deletion of aminoacid residues in already available protein in the database. These newly sequenced protein is presented as query for approximate match by network to discover the relation between this and already existing one in the database. Example for the case of substitution is as follows:

*Due to substitution*

*Query input:* VAP FPH HLP VTY KYI ADV KAG VDP SLV

*Signature of the input:* 0001011110000000111111110000101

*Signature of the output:* 0001011110000000111110110000101

*Output:* VAP FPH GKL VTY KYI ADV KAG VDP SLV

The examples related to insertion and deletion are similar to that of substitution. By observing the results above, it is concluded that the network can perform efficient approximate search on protein database if appropriate technique can be provided to find the signature.

#### 4.5.1 Experimental Results of Protein Database

Similar to the library database, experiments were done on protein database to study the relation between percentage of accurate retrieval and the length of the signature  $N$ .

Protein sequences are collected from Swiss-Prot protein sequence database. The third parameter is taken to be number of changes due to substitutions, deletions and insertions between the query and actual protein sequence. In all the above three cases almost similar results are obtained to that of the library retrieval system which are depicted in Figures 4.7, 4.8 and 4.9. In all the figures, graph (a) shows the relation between % of accurate retrieval and TV with no changes between queries and actual records. In Figure 4.7 graphs (b) and (c) shows the relation between  $N$  and percentage of retrieval when two and four amino acid residues are substituted in the corresponding queries respectively. In Figure 4.8 graph (b) shows the relation when two amino acid residues are deleted in the corresponding queries. The variation of the relation when two and four amino acids are substituted are shown in graph (b) and (c) of Figure 4.9. In Figure 4.10 graphs (a), (b) and (c) show the relation when protein sequence are bunched with two, four and six amino acid residues respectively and no change between query and actual record.

## 4.6 Discussion

In this section, some of the cases that bring out the merits and demerits of DNN for close proximity match are discussed. Due to false drops it is possible that record signature matches a query signature but the corresponding record does not satisfy the query. An example for this type of situation is as follows.

*Query for a book:* protocols, standards and interfaces.

*Signature of the query:* 111100110000101101000011101011

*Signature of the retrieved book:* 111100110000101101000011101011

*Retrieved book:* electronic data processing, distributed processing, computer networks.

The probability of a false drop can be made arbitrarily small by appropriate choice of the size of the signature. The probability of the false drop will decrease with the increase of the signature size [Faloutsos 85]. Since the maximum value of the percentage of retrieval

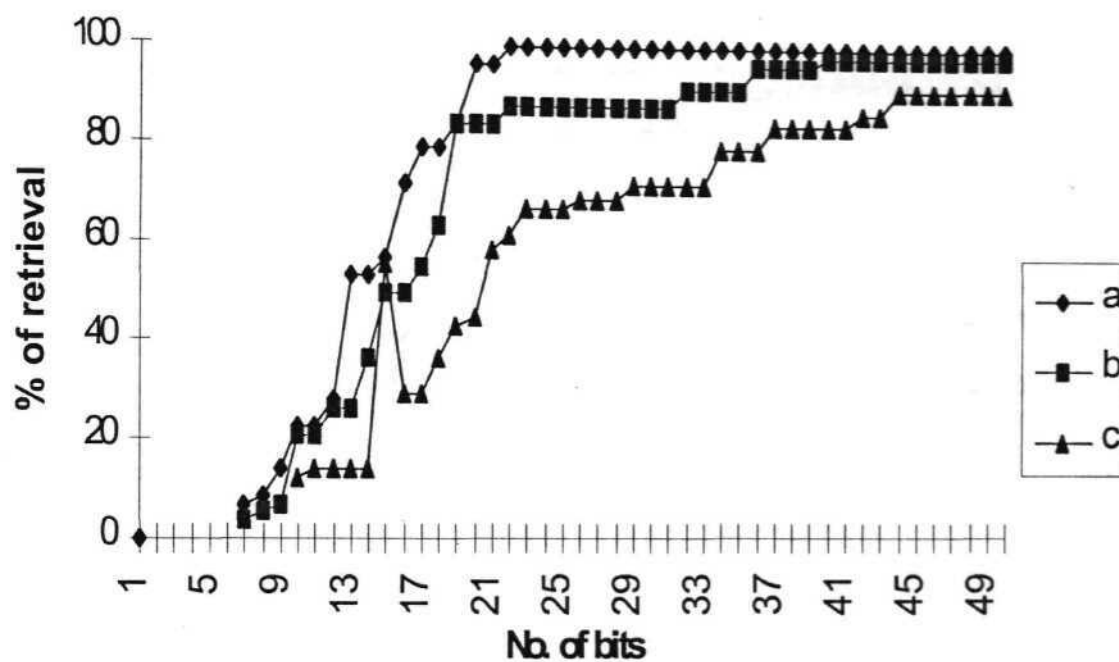


Figure 4.7 Protein Database with Substitutions.

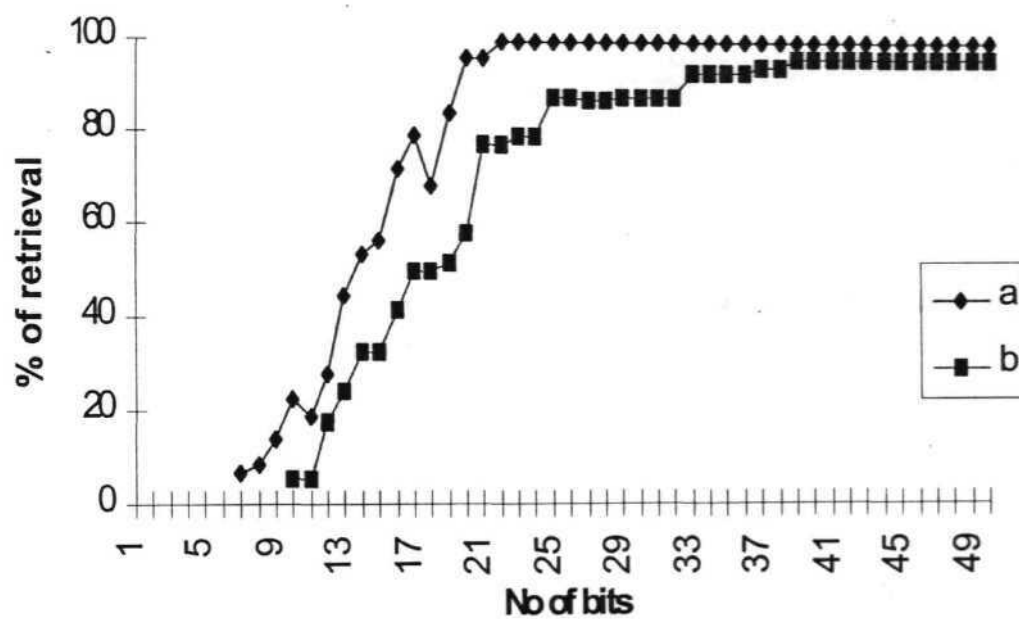


Figure 4.8 Protein Database with Deletions.

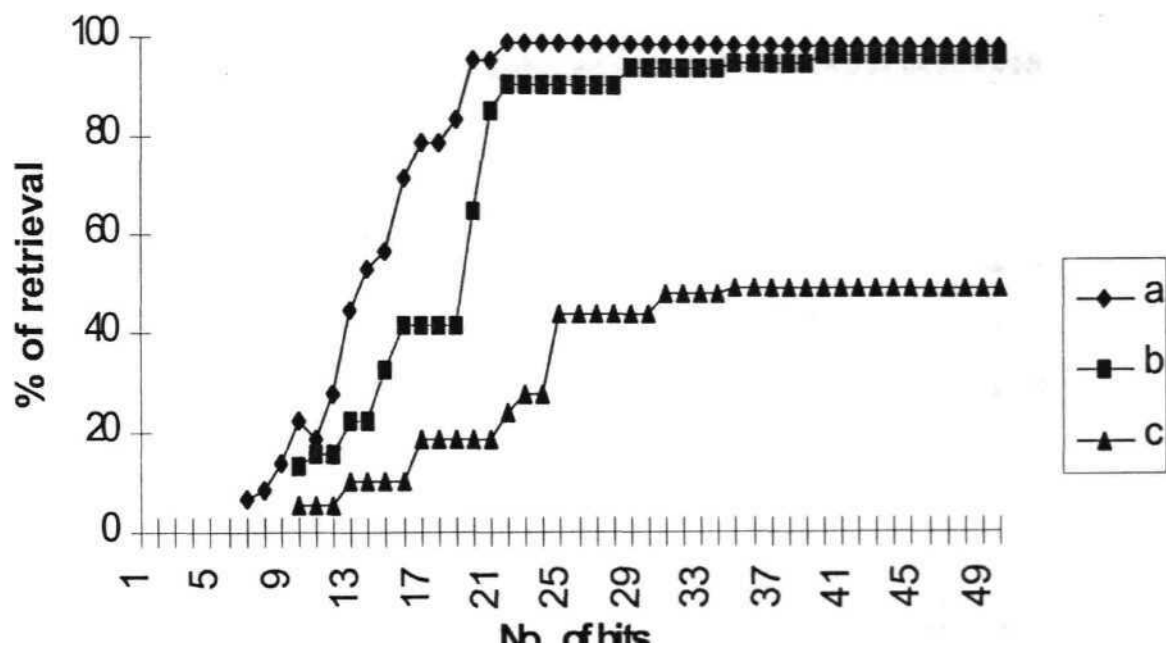


Figure 4.9 Protein Database with Insertions

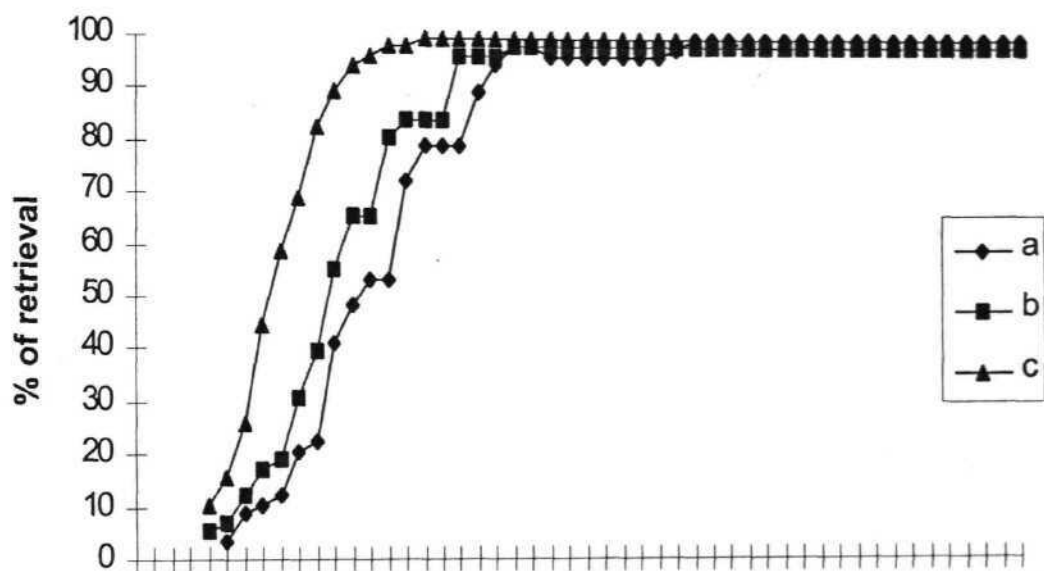


Figure 4.10 Protein Database with Bundling.

is around 95, the number of false drops with the network is very small.

The performance of the DNN in case of close proximity match is very good. That is even the query from the user differs: due to spelling mistakes, change in order of keywords, change in order of words in a keyword, deletion of keywords and addition of keywords, the DNN performs well. These cases are illustrated as follows.

*Case 1:*

If the signature of the record "parallel processing, computer architecture, computer networks" which is resulted due to swapping the position of the keywords "computer networks, computer architecture, parallel processing", is presented as a query then the result is as follows.

*Query input:* parallel processing, computer architecture, computer networks

*Signature of the input:* 01000 00100 00010 01000 11000 11000 00001 11010

*Signature of the output:* 01000 00100 00010 01000 11000 11000 00001 11010

*Retrieved book:* computer networks, computer architecture, parallel processing

In the above example the query is succeeded because the signature generation method, superimposed coding technique, generates the same signature even though the position of the keywords are changed in the records.

*Case 2:*

If the signature of the record "designing, ethernet implementing " which is resulted due to swapping the position of the words in the keywords "designing, implementing ethernet" is presented as a query then the result is as follows.

*Query input:* designing, ethernet implementing

*Signature of the input:* 01010 00011 00000 00110 10001 00111 00000 00011

*Signature of the output:* 01010 00011 00000 00110 10001 00111 00000 00011



*Retrieved book:* designing, implementing ethernet

In the above example the query is succeeded because signature generation method, superimposed coding technique, generates the same signature even though the position of the words in the keyword are changed in the records.

*Case 3:*

If the signature of the record "neural computers, neural networks, concepts " which is resulted due to adding a new keyword to the record "neural computers, neural networks" is presented as a query then the result is as follows.

*Query input:* neural computers, neural networks, concepts

*Signature of the input:* 00110 10010 11001 01100 00000 01100 00110 00111

*Signature of the output:* 00110 00000 11001 01100 00000 01100 00110 00111

*Retrieved book:* neural computers, neural networks

*Case 4:*

If the signature of the record "network computing, computer architecture, processing parallelly" which is resulted due to modifying the record "computer networks, computer architecture, parallel processing " is presented as a query then the result is as follows.

*Query input:* network computing, computer architecture, processing parallelly

*Signature of the input:* 01001 00100 10010 01000 11100 11000 00001 11010

*Signature of the output:* 01000 00100 00010 01000 11000 11000 00001 11010

*Retrieved book:* computer networks, computer architecture, parallel processing

*Case 5:*

The process of close proximity match using DNN is not considering meaning of the keywords for retrieval. Due to this reason, even though the keywords *image processing* and *vision* are related DNN may not do well. To deal with the cases of this type one should also take the *knowledge base* into consideration.

## **4.7 Conclusions**

In this chapter, close proximity matching capability of DNN is discussed. Experiments were carried out on the performance of the network by considering two case studies, namely library database and protein database. It is shown that the neural network based retrieval scheme works efficiently even for very large databases. Though the performance is degraded minutely due to false drop, around 95% of retrieval makes this method a novel one.

# Chapter 5

## Word Sense Disambiguation Using DNN

### 5.1 Introduction

This chapter, deals with the solving of the problem of word sense disambiguation using DNN. Word sense disambiguation is the problem of assigning a sense to an ambiguous word. Our premise is that the context in which the word is used, plays a major role in disambiguation and the context is largely determined by co-occurring words that are present in the sentence. It is emphasized earlier too, that the knowledge of co-occurring words is important for disambiguation [Guthrie 91; Leacock 93; Leacock 96; Smadja 89; Tsutsumi 91; Yarowsky 92]. The sense-tagged corpora and Level-wise [Agrawal 93] algorithm are made use to identify the frequent co-occurring word combinations. These word-sets are representative of a sense of the word having multiple senses. These co-occurring word combinations are used by a superimposed coding technique to determine a signature of a sense of the word. All the possible signatures are used as the training patterns to be memorized by a Hopfield model of neural network. The disambiguation process involves the association of the signature of the context of the instance of the word that is to be disambiguated with one of the memorized training patterns by the Hopfield model of neural network. Though the Hopfield model in its conventional form is

<sup>1</sup> The contents of this chapter appeared in ACS'97 International Conference Proceedings, Technical university of Szczecin, Poland, 1997, with the title "Word sense disambiguation using artificial neural networks"

<sup>2</sup> An improved version of this paper is to appear in Journal of Intelligent Systems, with the title "Word sense disambiguation using Hopfield neural networks"

suitable for associative memory, it is not suitable for word-sense disambiguation due to the existence of spurious stable states. The spurious stable states may lead to an invalid word-sense of a given word. In the present chapter, the DNN with new learning rule, that avoids spurious states with an enhanced retrieval capacity is used for word sense disambiguation. The experimental results presented confirms the theoretical findings.

## 5.2 Word Sense Disambiguation

### 5.2.1 What is WSD?

Word sense disambiguation (WSD) is the task of assigning sense labels to occurrences of an ambiguous word. For example, consider two senses of the word *suit*, listed in a dictionary; *an action in court*, and *suit of clothes*. Given the sentence *The union's lawyers are reviewing the suit*, we would like the system to decide automatically that *suit* is used here in its court-related sense. Word sense disambiguation is very useful at one level or another in many Natural Language Processing (NLP) applications, including machine translation [Brown 91; Dagan 94], information retrieval [Krovetz 92; McRoy 92], speech synthesis [Yarowsky 92] and text processing [Yarowsky 94]. WSD has received increased attention in recent literature on computational linguistics [Lesk 86; Yarowsky 92; Yarowsky 95; Bruce 94; Ng 96; Chang 98; Collier 97].

### 5.2.2 Different Approaches

All approaches to WSD rely on the fundamental principle that word sense disambiguation involves the association of a given word in a text or a discourse with a definition or meaning (sense) which is distinguishable from other meanings potentially attributable to that word [Ide 98]. The task, therefore, necessarily involves two steps: (1) the determination of all different senses for every word relevant (at least) to the text or discourse under consideration, and (2) a means to assign each occurrence of a word to the appropriate sense.

A precise definition of a sense is, however a considerable debate within the community. Variety of approaches in this regard has raised concern about the comparability of much WSD work, and given the difficulty of the problem of sense definition, no definitive solution is likely to be found soon [Ide 98]. The present study addresses 2. The assignment of senses to words, is accomplished by the reliance on two major sources of information, namely, (a) the *context* of the word to be disambiguated, in broad sense this includes information contained within the text in which the word appears and (b) the *external knowledge source* such as lexical, encyclopedic, etc., resources. The work on WSD can be categorized into different classes depending on the external knowledge sources and the method of extracting the context. These approaches are broadly classified as AI-based approach, Knowledge-based approach and Corpus-based approach.

#### 5.2.2.1 AI-based Approaches

AI-based approaches [Hayes 78; Small 80; Dahlgren 88; Wilks 90] use detailed knowledge about syntax and semantics to perform the task. Knowledge representation schemes, such as *semantic network* and *frames* are used to represent the word meanings and other relevant knowledge. In the semantic network [Masterman 62], the concepts and the dictionary meanings of the word are represented as nodes and the links among the words (tokens) and concepts (types) represent various semantic relations and the association between words. The disambiguation is implicit as only one concept node associated with a given input word is likely to be involved in the most direct path found between two input words. Some AI-based approaches exploit the use of frames containing information about words and their roles and relations to other words in individual sentences [Hayes 78]. Besides utilizing the knowledge representation techniques, there are also approaches where the uses of expert system technology and reasoning process are reported in literature [Small 80; Dahlgren 88]. However, all these AI-based methods heavily rely on large

amounts of hand-crafted knowledge and not at all practical for language understanding in any but extremely limited domain. A turning point in WSD occurred in 1980s when it became possible to have large scale lexical resources such as machine readable dictionaries and corpora. As a result, a variety of statistical methods that range **from** dictionary-based (knowledge based) methods that rely upon definitions to corpus-based methods that use context extracted from large corpora have been put forward.

#### 5.2.2.2 Knowledge-based Approaches

As Machine-readable-dictionaries (MRDs) became a popular resource of knowledge for language processing tasks, automatic extraction of lexical and semantic knowledge bases from MRDs has become a primary activity in the area of WSD. Unfortunately, the use of MRDs as knowledge sources for sense disambiguation leads to some problems [Ide 98]. Despite its shortcomings, the MRDs provide a ready-made source of information about word senses and became a staple of WSD research. Lesk [Lesk 86] describes the first MRD-based WSD method that relies on the extent of overlap between words in a dictionary definition and words in the local context of the words to be disambiguated. The other attempts regarding MRD based WSD are [Wilks 90; Veronis 90; Yarowsky 95; Cheng 98]. Thesauri provide information about relationship among words, most notably synonyms. The *Roget's* and the other thesauri are used in WSD by [Patrick 85; Yarowsky 92].

WordNet [Miller 90], a lexical Knowledge base , has become the best-known and mostly available knowledge base for WSD in English. WordNet combines the features that are commonly exploited in disambiguation work. It includes definitions for individual senses of words within it, as in a dictionary and it defines "syn-sets" of synonymous words representing a single lexical concept. Some of the attempts to exploit WordNet for sense

disambiguation are reported in [Voorhees 93; Resnik 95].

### 5.2.2.3 Corpus-based Approaches

In recent years, text corpora have been the main source of information for WSD [Gale 92]. A corpus provides a bank of samples that enable the development of numerical language models, and thus the use of corpora goes hand-in-hand with empirical methods in WSD [Ide 98]. A typical corpus-based algorithm constructs a training set from all contexts of a polysemous word in the corpus, and uses it to learn a classifier that maps instances of this word (each supplied with its context) into the senses. The corpus-based approach involves matching the context of the instance of the word to be disambiguated with information about the context of previously disambiguated instance of the word derived from corpora [Gale 92; Leacock 93; Bruce 94; Ng 96; Karov 98]. Besides matching of the context, this approach also depends critically on *manual sense tagging*, a laborious and time-consuming process that has to be repeated for every word, in every language, and more likely than not, for every topic of discourse or source of information. Generally, sense information is not readily available in the corpus explicitly.

### 5.2.3 Context

The context of the word includes information contained within the text or discourse in which the word appears [Ide 98] and it plays a crucial role in understanding the appropriate sense of the word. The context can be of different types; *local context*, *topical context* and *domain context*. The *local context* includes information on word order, distance, collocation and some information about syntactic structure; it includes all tokens (words and punctuation marks) in the immediate vicinity of the target word. The *topical context* consists of substantive words that are likely to co-occur with a given sense of the target

The context of the current script or domain represents the *domain context*.

### 5.3 Earlier Corpus-based Methods

The recent emphasis on corpus based NLP has resulted in much work on WSD of unconstrained real world texts.

Gale, Church and Yarowsky [Gale 92] develop a topical classifier based on Bayesian decision theory. The information the classifier uses is an unordered list of words that co-occur with the target in the training examples. The work of McRoy [McRoy 92] uses hand crafted lexicons and it is pointed out that a diverse set of knowledge sources are necessary to achieve WSD. Yarowsky [Yarowsky 93] uses local collocation knowledge that provides important clues to WSD. It is demonstrated only on performing binary (or very <sup>x</sup> coarse) sense disambiguation.

Bruce and Wiebe [Bruce 94] develop a probabilistic classifier that uses supervised learning from tagged sentences. The work of Bruce and Wiebe uses parts of speech (POS) and morphological form, in addition to surrounding words. Similar approach is followed by Ng and Lee [Ng 96] but they integrate a diverse set of knowledge sources including part of speech of neighboring words, morphological form, the unordered set of surrounding words, local collocations, and verb-object syntactic relation. The work of Miller [Miller 94], Yarowsky [Yarowsky 92] and Leacock [Leacock 93] uses the unordered set of surrounding words and they use statistical classifiers, neural networks, or IR-based techniques.

Yarowsky [Yarowsky 95] uses an unsupervised learning procedure to perform WSD. He proposes automatic augmenting of a small set of experimenter-supplied seed collocation (e.g., *manufacturing plant* and *plant life* for two different senses of the noun *plant*) into a much larger set of training materials. He resolves the problem of sparseness of the collocations by iteratively bootstrapping the acquisition of training materials from a seed collocations for each sense of homograph.



Since the present work is concerned with a corpus-based neural network approach for WSD, it is appropriate to view the previous neural network approaches for WSD. The use of neural networks to disambiguate word sense is proposed by Cottrell and Smal-1 [Cottrell 83] and Waltz and Pollack [Waltz 85]. However these networks are created manually and are therefore necessarily limited in size (no more than few dozen nodes) [Ide 90]. It is not clear that these models will scale up to realistic dimensions. Veronis and Ide [Ide 90] use large scale neural networks to resolve word sense. They describe a method to automatically create a neural network using machine readable dictionaries. The connections in the network represents the knowledge of semantic relations between a word and the words used to define it. They use this knowledge to disambiguate the ambiguous words.

Leacock, Towell and Voorhees [Leacock 93] compare the Bayesian classifier with a content vector classifier as used in information retrieval and a neural network with backpropagation. They show that neural networks are more effective than several other methods of sense disambiguation.

Mooney [Mooney 96] describes an experimental comparison of seven different learning algorithms on the problem of learning to disambiguate the meaning of a word from context. In this comparative study the statistical, neural network, decision-tree, rule-based, and case-based classification techniques are tested. Ample empirical evidence are given which indicates that neural networks are at least as effective as other learning systems.

Voorhees [Voorhees 98] describes a neural network based classifier for WSD that improve the effectiveness of information retrieval system. The classifier combines output of a neural network that learns the topical context with the output of a network that learns local

context to distinguish among the senses of highly ambiguous words. The disambiguator is a particular formulation of feed-forward neural networks which demonstrates that contextual representation of both local and topical components are effective for resolving word senses.

In view of the above discussion it is evident that neural networks are suitable for word sense disambiguation. In the present context the DNN is used as a supervised classifier for WSD. The classifier uses generally the topical context and sometimes local context in case of test sentences to represent the sense. The Level-wise algorithm is used for efficient context extraction, which identify the frequently co-occurring words. It is shown here that DNN performs very efficiently for WSD. The disambiguation for Telugu words with a sense-tagged corpus is attempted using DNN and very encouraging results are obtained. Thus this chapter reflects the importance of DNN for WSD.

#### 5.4 DNN for WSD

In general, the problem of context matching can also be viewed as associative recall in the sense that with every sense of a word, there is a set of contexts that can be associated with it. Given a partial or imprecise context of this word, the context matching is to find the correct context as extracted from the corpus. Thus, this problem can apparently be solved using any associative memory scheme such as Hopfield model of neural network. However, the limitations of Hopfield network such as presence of spurious states, lack of large storage capacity, it cannot be trivially applied to WSD. One of the most important desirable properties that an associative memory model should possess to solve WSD is that it must not have any spurious recall. Otherwise, the recall will be a spurious sense of a word which can lead to an altogether confused disambiguation. However, any attempt to remove spurious states generally affects adversely the capacity of the network. This

observation motivated us to justify that DNN is very appropriate for WSD.

The present work uses manually sense-tagged corpus to construct the training and testing examples for the classifier. But efforts are made to automatically sense-tag a training corpus using testing examples.

Let  $W$  be a word with multiple word-sense  $s_1, s_2, \dots, s_m$ . In the sentences where the word  $W$  is used with sense  $s_i$ , there are some words (co-occurring) that represent the sense. The set of most frequent co-occurring word combinations for a sense  $s_i$  of  $W$  is called *word-set*. This word-set  $a_{i1}, a_{i2}, \dots, a_{in}$  is the representative of the topical context of the particular sense  $s_i$  of  $W$ . The word-sets  $E_i = \{a_{i1}, a_{i2}, \dots, a_{in}\}$  ( $i = 1, \dots, m$ ) for all the sense  $s_i$  ( $i = 1, \dots, m$ ) of  $W$  are extracted from the corpus using Level-wise algorithm. These word-sets  $E_i$  ( $i = 1, \dots, m$ ) are used as the training sets for the disambiguation system. When a new sentence is presented to the system, the sentence contains the context as a word-set  $\{t_1, t_2, \dots, t_j\}$  which is representative of the meaning of the word in the sentence. The problem is to determine the closest context among the training set for the given context  $\{t_1, t_2, \dots, t_j\}$ .

Thus the major steps of the disambiguation system are (1) determination of the context from the sense-tagged corpus and (2) matching of the context of the new sentence with the set of context extracted from the corpus. In addition to these two problems, there are other issues that are to be resolved such as finding the context of the test sentence, preprocessing of the corpus for context extraction and automatic sense tagging of the corpus.

## 5.5 Context Extraction

In this section an algorithm to extract the context from the sense-tagged corpus is presented. The underlying principle of the algorithm is based on that of rule discovery in Data Mining. It is to compute the frequently occurring (co-occurring) word-set in the set of sentences specific to a sense. That is the algorithm extracts co-occurring words (word-sets) that are exceeding a user specified frequency threshold. Assuming that the corpus contains reasonably large number of sentences, the algorithm attempts to make optimal number of passes through the corpus. In this process it generates the frequent word-sets based on a user-specified *threshold frequency* ( $a$ ) or the *minimum support count*. The Level-wise algorithm [Agrawal 93] is adopted here for this purpose.

The Level-wise algorithm ensures the selection of words that co-occur more frequently with sense  $s$ , of word  $W$ , over those co-occurring less frequently. In other words, it prefers words that co-occur more frequently if there is a large number of eligible keywords. And it also reduces the possibility of selecting a word based on spurious occurrence.

Level-wise algorithm is an efficient breadth-first or level-wise method for generating candidate sets, i.e., potentially frequent word-sets. This algorithm make use of a binary database (a relational table) where the columns are characterized as word-sets. A set of definitions that are used in this algorithm are explain below.

### Definitions

Let  $R$  is a  $m \times n$  matrix of 0,1 entries. A row of  $R$  corresponds to a sentence in the corpus. A column of  $R$  corresponds to a word in the corpus.

$$R_{ij} = \begin{cases} 1 & \text{if the } i \text{ th sentence has the } j \text{ th word} \\ 0 & \text{otherwise} \end{cases}$$

Let  $X$  be a *word-set*

*Frequent set* is an word-set  $X$  in the corpus  $R$  appearing in at least a number of statements. If  $X$  is a frequent set all its subsets are frequent.

*Border set* is an word-set in  $R$ , all proper subsets of which are frequent sets, but it is not a frequent set by itself.

*Maximal frequent set* is an word-set, which is frequent set but no superset of which is frequent set or a border set.

*Example 5.1:* Let  $C = \{\{a,b,c\}, \{a, b,d\}, \{a,c\}, \{b,c\}\}$  be the sentences in a corpus where  $a, b, c$  and  $d$  are unique words. If  $a = 2$ , then the frequent sets are  $\{a\}, \{b\}, \{c\}, \{a, b\}, \{b, c\}, \{a,c\}$ , the border sets are  $\{d\}, \{a,b, c\}$  and the maximal frequent set is  $\emptyset$ . The computational steps involved in the Level-wise algorithm are given in Figure 5.1.

### 5.5.1 Stop-words

The words, which do not contribute to the disambiguation process are called *stop-words*. The initial step in the process of disambiguation is to remove these stop-words from both the training sentences and the test sentences. Basically the stop-words are classified as *domain independent* and *domain dependent* words.

*Domain independent stop-words:* The domain independent stop-words are independent of words to be disambiguated and corpus.

For example, high frequency words such as *that, the, by, you, a, etc.* are domain independent stop-words.

*Domain dependent stop-words:* The domain dependent stop-words depend on the sense of the word. So there is a separate domain *dependent* stop-word list for each sense of the word  $W$ .

Input:  $R$   $m \times n$  matrix of 0,1 entries, a threshold frequency.

Output:  $F$  Frequent sets,  $B$  Border sets and  $M$  Maximal frequent sets

procedure Level-wise( $R$ ,  $s$  : in;  $F.B.M$  : out)

1.  $C_1$  set of word-sets of size 1,  $L_1$  set of frequent sets of size 1,  $B_1$  set of word-sets of size 1 which are not in  $L_1$ ,  $k = 2$
2. while ( $L \neq \emptyset$ ) do
3.     for all word-sets  $l_1 \in L_{k-1}$  do
4.         for all word-sets  $l_2 \in L_{k-1}$  do
5.             if  $l_1[1] = l_2[1] \wedge l_1[2] = l_2[2] \wedge \dots \wedge l_1[k-1] = l_2[k-1]$  then
6.                  $c = l_1[1], l_1[2], \dots, l_1[k-1], l_2[k-1]$
7.                 for all  $(k-1)$ -subsets  $s$  of  $c$  do
8.                     if  $s \in L_{k-1}$  then
9.                          $C_k = C_k \cup \{c\}$
10.                 end do
11.             end do
12.         end do
13.     for all sentences  $t \in R$  do
14.         Increment the count of all word-sets in  $C_k$  that are contained in  $t$ ;
15.     end do
16.      $L_k =$  All word-sets in  $C_k$  with minimum support ( $\sigma$ ) ;
17.      $B_k =$  All word-sets in  $C_k$  whose count less than minimum support ( $\sigma$ );
18.      $fc = fc + 1$ ;
19. end do
20.  $F = \bigcup L_k$ ,  $B = \bigcup B_k$
21.  $M =$  all word-sets  $w_i \in F$  A superset of  $w_i \notin F$  or  $B$
22. End.

**Figure 5.1 The Level-wise Algorithm**

For example let us consider a sentence in bank corpus, " *An account holder wearing a black suit enter the bank to deposit money*". Here '*account holder*', '*deposit*' and '*money*' are co-occurring words, but '*black suit*', '*wearing*' and '*enter*' do not contribute towards the disambiguation of word '*bank*'. Thus the later collection consists of domain dependent stop words. Though these are the domain dependent stop-words for the word '*bank*' in its '*financial bank*' sense, they may contribute towards disambiguation of the word '*suit*' in its '*suit of clothes*' sense.

Another advantage of removing these domain dependent stop-words from a test sentence is in extracting local context with a window. Suppose to extract local context with a window of  $\pm 3$  from the above sentence there is a possibility of missing the extraction of co-occurring words '*account holder*'. If the domain dependent stop-words '*black suit*', '*wearing*' and '*enter*' are removed from the sentence, the required co-occurring words will move into the  $\pm 3$  window and leads to the extraction of exact context. The question is how to extract these *domain dependent* stop-words list. The Level-wise algorithm is useful in extracting the domain dependent stop-words list. These are the words that are present in *frequent set* with frequency one. So by using Level-wise algorithm the domain dependent stop-words list for each sense are extracted. The function  $dd-stopwords(F)$  (Ref. Figure 5.2) is used for this purpose where  $F$  is the frequent set.

## 5.6 Methodology

In this section the methodology adopted by DNN for disambiguation is presented along with an algorithm (Figure 5.2) concerning the same. The task of disambiguating the word  $W$  into one of its possible senses  $s_1, s_2, \dots, s_m$ , consists of two phases. 1) training phase 2) testing phase. In the training phase a set of sentences  $C_W = S_{s_1} \cup S_{s_2} \cup \dots \cup S_{s_m}$

with the occurrence of word  $W$  is collected from the corpus. Here the set  $C_w$  is a collection of sentences,  $S_{si}$  for each sense  $s_i$  ( $i = 1, \dots, m$ ). The *domain-independent* stop-word list  $DI$  is used to discard the domain independent stop-words from the sentences of the corpus using the function  $discard(C_w, DI)$ . The next step is the extraction of context for each sense  $S_i$  of  $W$ , from these sentences using the level-wise algorithm.

Input:  $C_w$  sense-tagged corpora for the word  $W$ ,  $DI$  domain independent stop-word list,  $T$  test sentence,  $s$  threshold frequency .

Output:  $Y$  the test sentence tagged with correct sense.

procedure  $WSD(C_w, DI, T: in; Y: out)$

1.  $S = discard(C_w, DI)$
2. Partition  $S$  to  $S_{s1}, S_{s2}, \dots, S_{sm}$
3. for  $i = 1$  to  $m$  do
4.      $R_{Si} = convert(S_{Si}); F = level-wise(R_{Si}, s);$
5.      $DD = dd-stopwords(F); S_{si}^d = discard(S_{Si}, DD);$
6.      $R_{Si} = convert(S_{si}^d); E_i = level-wise(R_{Si}, s);$
7. end do
8. for each word-set  $W_i \in \bigcup_{m=1}^m E_j$  do
9.      $R_i = superimposed-coding (w_i)$
10. end do
11.  $T_{di} = discard(T, DI); T_{dd} = discard(T_{di}, DD);$
12.  $t = context(T_{dd}); X = superimposed-coding(t);$
13.  $O = DNN(-R_1, R_2, \dots, R_m, X);$
14. Look up the signatures of the word-sets for signature that matches with  $O$  and attach its label to test sentence.
15. End.

Figure 5.2 The Algorithm of Word Sense Disambiguation



The set of sentences  $S_{s_i}$  for a sense  $s_i$  of  $W$  is used to extract the context which is represented by the set of word-sets  $E_i = a_{i1}, a_{i2}, \dots, s_i$ . Before this, the level-wise them expects a binary input, the sentences  $S_{s_i}$  for each sense  $s_i$  are converted into a matrix  $R$  of 1, 0 entries by using the function  $convert(S_{s_i})$ . The collection of contexts i.e., the set of word-sets  $\cup E_i(1, \dots, m)$  for each sense  $s_i$  is used as training set for the disambiguator DNN.

Since the basic building block of DNN is Hopfield network and it uses only binary form of information, to learn, the training set (word-sets) has to be converted into binary string set (*signature*). The signature is computed by using superimposed coding technique (see Section 4.3). These signatures are memorized in DNN by using them as training patterns.

In the testing phase, the actual process of disambiguation starts, when a new sentence (*test sentence*  $T$ ) containing the word  $W$  (with sense  $s_i$ ) is presented as input. First, the domain dependent and independent stop-words are removed from the test sentence by using the functions  $discard(T, DD)$  and  $discard(T, DI)$  respectively. In order to extract the context from the test sentence, a window of size  $\pm 3$  (local context) is used. The function  $context(T)$  used for this purpose. The signature of the new sentence (for which the word-sense is required to be determined) is calculated by the same superimposed coding technique. This signature is presented as the test pattern to the trained network and the network converges to one of the stable states which corresponds to  $s_i$ . Now this methodology of disambiguation is illustrate with an example.

## Example 5.2

Let us consider some sentences in *bank corpus* with *financial bank* sense.

- A bank services on the cash deposits it possesses.
- When money is credited to an account of a bank, it is called cash deposits.
- Cash deposits have flexibility for easy withdrawal in a bank.
- Cash deposits are easily transferable from one bank to another.
- In a bank clerks are appointed to handle cash.
- The handling of cash by clerks in a bank is supervised by the manager.
- Managers help people in their cash transactions through clerks.
- In a bank clerk accepts/disburses cash.

As explained in the algorithm of WSD the first step is to discard the *domain independent* stop-words from the corpus. In this example *domain independent* stop-words are

a, are, the, on, is, in, it, when, to, an,  
of, have, for, from, one, by, their, another

Next, using the Level-wise algorithm the context (word-sets) is extracted. In the present example the context is represented by the word-sets 'cash-deposits-bank' and 'clerk-cash-bank', if the threshold frequency  $a$  is 2 (out of eight sentences the word-set is appearing in four). The *domain dependent* stop-words are also extracted, using Level-wise algorithm (words in frequent set occur with frequency one). In the present example, domain dependent stop-words are

service, posses, account, called, easy, withdrawal,  
easily, transferable, transaction, help, accept,  
disburses, supervised, handle, appointed.

The signatures of the word-sets are used as training patterns for the network. The disambiguation starts when a new sentence with a word *bank* is presented to DNN for disambiguation. For example consider the following sentence as test sentence.

- Cash deposits are accepted in a bank only when the transaction is for legal purpose.

After removing the *domain dependent* and *domain independent* stop-words from the test sentence the resulted sentence is

- Cash deposits accepted bank only legal purpose.

In order to extract the context from the test sentence, a window of size  $\pm 3$  (local context) is used. The result is

- Cash deposits bank only legal purpose.

As explained above, DNN associates the signatures of the contexts of training and test sentences in order to disambiguate.

## 5.7 Experimental Results

Experiments were carried out for a set of Telugu words to demonstrate the applicability of DNN for disambiguation. Telugu is the most extensively used language in India after Hindi. The input to the WSD program consists of unrestricted, real-world Telugu sentences, where each sentence is manually pre-tagged with correct sense of the ambiguous word. The output is tagged with the correct sense of the ambiguous word (in the form of a sense as the definition of the word, given in the dictionary). The choice on which sense of definitions (Dictionary dependent) to use is agreed upon in advance. In the present work, the sense definitions that are given in Telugu dictionary (Telugu Nighamtuvu [Nighamtuvu 79]) are used.

The present work initially uses manually sense-tagged corpus of Centre for Applied Linguistics & Translation Studies, University of Hyderabad to construct the training and testing examples for the classifier. But efforts are being made to automatically sense-tag a training corpus using testing examples. The corpus consists of 3 million words, which is consider to be a small corpus for the present task. Experiments are conducted to disambiguate three Telugu words '*pannu*', '*adugu*' and '*pattu*'. The word '*pannu*' has two senses (tax and teeth), the word '*adugu*' also has two senses (ask and foot), where as '*pattu*' has three senses (silk, catch and hold). The number of sentences for each word in the training and testing corpora are given in the following table.

Table1

Word	Sens	Training sample size	Testing sample size
<i>pannu</i>		<i>tax</i>	<i>fin</i> 13
పి	<i>teeth</i>		45 9
	<i>total</i>	105	22
<i>adugu</i>	<i>ask</i>		19 6
అడుగు	<i>foot</i>	73	20
	<i>total</i>	92	26
<i>pattu</i>	<i>catch</i>	34	10
పట్టు	<i>silk</i>	33	10
	<i>hold</i>	7	3
	<i>total</i>	74	23

The sample training corpora for the word *pannu* is given in Appendix-1. The disambiguates a set of about 800 high frequency domain dependent stop-word list (see Appendix-2) to remove from the training sentences. From the training sentences it extracts the word-sets for each sense of the word using Level-wise algorithm. The word-sets for the two senses (teeth and tax) of the word *pannu* is given in Appendix-3. The domain dependent stop-word list for the two senses (teeth and tax) are given in Appendix-4. The signatures of the word-sets for all senses of the word are used as training (exemplar) patterns to the DNN, they are memorized in DNN. The signature size for each pattern is fixed as 50-bits. The disambiguation process starts when the signature of a new sentence contain this word is presented to the system as a test pattern. Before converting the test sentence into signature it is preprocessed as follows: i) The domain dependent and independent stop-word are removed from it, ii) a window of  $\pm 3$  is used to extract the context. The DNN converges to one of the stored pattern which corresponds to the sense

Table1

Word	Sens	Training sample size	Testing sample size
<i>pannu</i>		<i>tax</i>	<i>fin</i> 13
పి	<i>teeth</i>		45 9
	<i>total</i>	105	22
<i>adugu</i>	<i>ask</i>		19 6
అడుగు	<i>foot</i>	73	20
	<i>total</i>	92	26
<i>pattu</i>	<i>catch</i>	34	10
పట్టు	<i>silk</i>	33	10
	<i>hold</i>	7	3
	<i>total</i>	74	23

The sample training corpora for the word *pannu* is given in Appendix-1. The bigram is a set of about 800 high frequency domain dependent stop-word list (see Appendix-2) to remove from the training sentences. From the training sentences it extracts the word-sets for each sense of the word using Level-wise algorithm. The word-sets for the two senses (teeth and tax) of the word *pannu* is given in Appendix-3. The domain dependent stop-word list for the two senses (teeth and tax) are given in Appendix-4. The signatures of the word-sets for all senses of the word are used as training (exemplar) patterns to the DNN, they are memorized in DNN. The signature size for each pattern is fixed as 50-bits. The disambiguation process starts when the signature of a new sentence contain this word is presented to the system as a test pattern. Before converting the test sentence into signature it is preprocessed as follows: i) The domain dependent and independent stop-word are removed from it, ii) a window of  $\pm 3$  is used to extract the context. The DNN converges to one of the stored pattern which corresponds to the sense

<b>easel</b>					
<i>Sense</i>	<i>Training sample size</i>	<i>Word — set size</i>	<i>Testing sample size</i>	Result A <i>Average</i> number correct	Result B <i>Average</i> number correct
<i>tax</i>	60	46	13	12	9
<i>teeth</i>	45		9	8	6
case2					
<i>tax</i>	56	43	13	11	9
<i>teeth</i>	40		9	8	6
case3					
<i>tax</i>	46	43	13	11	8
<i>teeth</i>	33		9	8	6
case4					
<i>tax</i>	46	43	16	14	10
<i>teeth</i>	33		11	9	7
case8					
<i>tax</i>	39	36	13	11	8
<i>teeth</i>	29		9	7	6
case6					
<i>tax</i>	39	36	16	13	10
<i>teeth</i>	29		11	8	7
caseT					
<i>tax</i>	18	16	13	11	8
<i>teeth</i>	14		9	6	6
case8					
<i>tax</i>	18	16	16	13	10
<i>teeth</i>	14		11	8	7

Table 4

adugu	అడుగు				
easel					
<i>Sense</i>	<i>Training sample size</i>	<i>Word - set size</i>	<i>Testing sample size</i>	<i>Result A Average number correct</i>	<i>Result B Average number correct</i>
<i>ask</i>	19	21	6	5	4
<i>foot</i>	73		20	17	14
case2					
<i>ask</i>	13	13	6	6	5
<i>foot</i>	53		20	16	14
case3					
<i>ask</i>	7	12	6	5	4
<i>foot</i>	33		20	15	13

Table 5

pattu	పట్టు				
easel					
<i>Sense</i>	<i>Training sample size</i>	<i>Word — set size</i>	<i>Testing sample size</i>	<i>Result A Average number correct</i>	<i>Result B Average number correct</i>
<i>catch</i>	34	23	10	7	7
<i>silk</i>	33		10	9	8
<i>hold</i>	7		3	1	1
case2					
	24	17	10	6	6
<i>silk</i>	23		10	8	8
<i>hold</i>	4		3	2	1

## 5.8 Discussion

In this section, some of the issues that effect the performance of the present disambiguation process are discussed.

*Data sparseness:* In case of the present disambiguator, the problem of data sparseness is severe due to small corpus. Even though the DNN does not have any limitation in storage capacity, it could not memorize large number of training samples (which effects the performance of the diambiguator) because of the small corpus. While some of the possible co-occurrences are not observed in the corpus, and disambiguation is typically rely on the statistics of the co-occurrence of the words, then the performance of the disambiguator is poor. The disambiguator initially uses manually sense-tagged corpus to construct the training examples for the classifier. After disambiguating a new test sentence (sense will be tagged to the sentence) the output will be augmented to the corpus. This solves the problem of manually sense tagging and the corpus will become incrementally complete.

*Completeness:* The disambiguator is a supervised learning classifier. It is a complete system since it is able to disambiguate all possible word-senses of the word (listed in the dictionary). The maximum percentage of correct disambiguation is 90 and the average is 83. The percentage can be improved if a robust corpus is available (as explained above). Another possibility of improving the percentage of disambiguation is incorporating the information of word definitions and relations in signature computing technique. The superimposed coding technique that is used to compute the signature uses the ASCII values of the characters occurred in the words. Even though the add-shift strategy is adopted to take the position of the character occurrences into account, there is a possibility that two words, with all together different meaning may mapped to a same signature. This



is called false drop. The probability of a false drop can be made arbitrarily small by appropriate choice of the size of the signature. Still the probability of a false drop can be made minimum if one incorporate the information of word definitions and relations in the signature computing technique.

*Domain dependent stop-word:* As explained above domain dependent stop-words improve the context extraction process from the test sentence. The Level-wise algorithm is used in extracting the domain dependent stop-word list. These are the words that are present in frequent set with frequency one. But there is a possibility of listing actual co-occurring words as domain dependent stop-words if they occur in only one training sentence(see *bank corpus example*). In that particular case it may effect performance of the disambiguator. But it is a rare possibility.

*False drops:* It is possible that a record signature matches a query signature but the corresponding record does not satisfy the query. Such occurrence is referred to as false drop. The probability of a false drop can be made arbitrarily small by appropriate choice of the size of the signature.

## 5.9 Conclusions

The aim of designing a disambiguating system which is sound and incrementally achieve completeness, is succeeded. The disambiguator uses a corpus based neural network approach and works on context matching. A novel neural network architecture (DNN) together with a new learning rule is used for sense disambiguation. The Level-wise algorithm is used for the extraction of context and domain dependent stop-words. The disambiguator is tested on Telugu words. The performance of the disambiguator is quite

Good and it has an average accuracy of 83%. The experimental results are quite encouraging. But more experimentation is needed to fully evaluate the system and hence it is proposed to extend the system to different and wider corpora in future

# Chapter 6

## Conclusions

The availability of a Neural Network as an associative memory with high storage capacity, the perfect recall without spurious states is not so common. This has prompted us for an alternative, which has resulted in the network, Dynamical Neural Network (DNN for short) with the properties such as associative memory, pruning and order-sensitive learning. Thus this thesis addresses this new neural network architecture called DNN. The network works as an associative memory with 100% perfect retrieval. A new learning rule for Hopfield network (the basic node in DNN) is proposed and evaluated. With this learning rule the fundamental problem of designing a network with desired states as stable states without spurious states has been successfully resolved. And it also increases the storage capacity of the small size networks substantially. Even though the design is shown only for two states the method can be generalized to any number of stable states. Though the Hopfield network memorizes two states, one can have large capacity in DNN due to its composite structure. The suitability of this DNN for practical applications is tested for two practical applications, viz., word sense disambiguation [Sreenivasa 97b] and the second one is close proximity match of large databases (like library and protein database) [Sreenivasa 97a].

In addition to the above mentioned advantages, the DNN is more suitable for hardware implementation. The proposed architecture has additional advantage, that is the network has a regular structure which is a very important feature for hardware implementation

of the network through VLSI.

The advantages of inherent neuronal properties like parallel and distributed processing, can not be realized without specialized hardware. Unfortunately, the large number of neurons and their highly interconnected nature make the construction of neural networks challenging. But this fully interconnected network has the most regular structure and small number of well defined operations, which make the network particularly suitable for VLSI implementation. A majority of implementation rely upon analog VLSI to provide fast and compact neurons, processing the required computational primitives [Mead 89]. The use of digital VLSI technology offers the advantages like high precision, low cost, programmability and availability [Zurada 97]. The use of hybrid approach for the VLSI implementation of neural networks provides the merits of both analog and digital technologies [Murray 91]. In particular, the networks use digital signals to carry information and control analog circuitry, where as analog circuitry is used for storing analog information. So our proposed architecture can be implemented using hybrid VLSI.

As a legacy to the present work, an attempt is made for the hardware realization of DNN. Since the design is of very primitive nature in this direction, a detailed work can be carried out in future.

The network structure consists of fully connected, multiple number of basic networks each of which is a Hopfield network with a new learning rule. The number of basic nodes depends on the number of training patterns to be stored in the DNN. In each basic network the weight matrix elements are computed by the combinational circuit 'C'. The equations required to compute the elements in the weight matrix are given in Section 2.6.3. The weight matrix is loaded in the electronic circuit 'W'. The sequential enabler and terminator (SET) provides the required timing and data (the patterns) transfer signals.

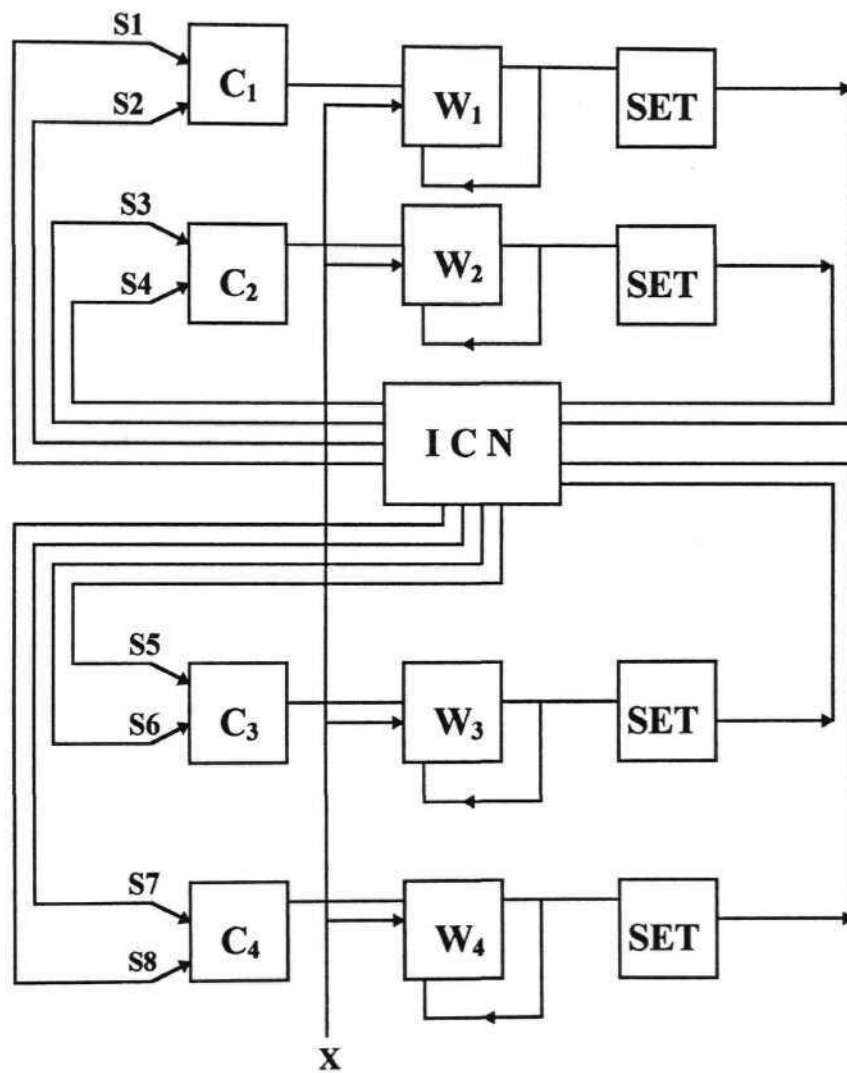


Figure 6.1 VLSI Implementation of DNN

**Input:**  $S_1, S_2, \dots, S_{2m}$  Exemplar patterns,  $X$  Test pattern, ( where  $m$  is the number of basic nodes in the network).

**Output:**  $0$  Output pattern.

**procedure** DNN( $S_1, S_2, \dots, S_{2m}, X : in; 0 : out$ )

1.  $j = 1; k = 2;$
2. *parallel for* ( $i = 1; i \leq m; i = i + j$ ) *do*
3.     constantly present the input pattern  $X$  to each node  $H_i$
4. *end do*
5. *parallel for* ( $i = 1; i \leq m; i = i + j$ ) *do*
6.     elements of weight matrix  $W_i$  is computed by  $C_i$  with  $S_{2i-1}$  and  $S_{2i}$  exemplar patterns.
7. *end do*
8. *parallel for* ( $i = 1; i \leq m; i = i + j$ ) *do*
9.     dynamics of Hopfield stabilizes each node  $H_i$  at a stable state
10. *end do*
11. *parallel for* ( $i = 1; i < m; i = i + k$ ) *do*
12.     output of  $H_i$  connected to  $C_i$
13.     output of  $H_{i+j}$  connected to  $C_i$
14.     prune the node  $H_{i+j}$
15. *end do*
16.  $j = j + j; k = k + k; m = \frac{m}{2}$
17. *repeat* step 5 through step 16 *until*  $m = 1$
18. END.

**Figure 6.2 The Algorithm for Training and Pruning of VLSI Implemented DNN**

In the DNN the basic networks are fully connected using a cube network as shown in Figure 6.1.

Initially each basic network in the DNN is trained to store two patterns. Outputs of successive pairs of basic networks are computed and one basic network in each pair is pruned. This is possible by way of supplying the output from a basic network (within each pair) to the other basic network as shown in the Figure 6.1. When this process is repeated, it finally results in leaving only one basic network in the entire layer, whose output exactly matches to one of the originally supplied patterns. As a conclusion the proposed DNN will take all the states supplied by the user as stable states and retrieves the desired pattern that exactly matches with it. At any current stage, the architecture is such that it can eliminate half the number of basic networks employed in the previous iteration.

The steps involved in the training and pruning of the VLSI implemented DNN are given as an algorithm and is shown in Figure 6.2.

It appears that DNN is more suitable for image processing applications as well. This would be one of the interesting works to be taken up. One can work in this direction.

## APPENDIX-1 (PANNU)

1. maMwrlulu I Alocana bAgunnaxani marnAde koVwwa pannu guriMci cAtiMpu veyiMcAru
2. koVwwa pannu vaccipadina kAraNaMcewa A Xanikulu yuvakudi jlwaM waggiMcAdu
3. wAnu Xanikulu mlxane pannu veSAnani Ayana anukunnAdu
4. kAni koVwwa pannu BAraNni Xanikulu moyaka sAManya prajala mlxike woseSAru
5. pannu kattakapowe poVlAnnuMci woVlagiMcinAru
6. e kAraNamu cewaniyiNa pannu wlyiMcina warvAwa rakwa srAvamu AgakuMdA vuMte
7. eVmamalis, 200 kriyosotu 200 mArcucu gaMtakoVKasAri iwavaleVnu
8. xunni paMdiMci bAbugAriki cAlArojulu pannu kUdA katti vunnAnaMdi
9. nAnnaku nuwu weVlusa pannu kattakapowe oVKasAri pilipiMcAraMdi
9. prajalalo ArWika samAnawvAnni wlsukoVni rAvadAniki praBuwvaM cepattina vixAnAla
10. valla AxAyapu pannu prAmuKyAM mariMwa peVrigiMxi
10. parokRa pannu prawyakRa pannu vyakwini uxxeSiMci viXiswAro
11. kAni parokRa panAxAyapu pannu prawyakRa pannula kovaku ceVMxinaxi
12. airanakapu pannu kastams dyUti xigumawi suMkaM eVksEj suMkaM moVxalEnavi parokRa
12. pannulaku uxAharaNalu
13. keMxra praBuwvaM viXiMce pannulalo AxAyapu pannu awi muKyamEnaxi
14. lo AxAyapu pannu samBaMXiMcina aMSAlanu pUrwigA savariMci, AxAyapupannu cattaM,
14. 1918 gA rUpoVMxiMcAru
15. I cattaMlo muKyaviSeRaM AxAyAnni yZaArjiMcina saMvawsaraMlone pannu viXiMcadaM I
15. cattaM vyApArarlwyA, vqwwirlwyA laBiMce yAxqcCika vasULyYanu pannu viXiMce
15. AxAyaMga parigaNiMciMxi
16. va saMvawsaraMlo praBuwvaMvAru AxAyapu pannu cattAnni sulaBaM ceyadAniki "1A
16. kamiRan" ku bAXyawa appagiMcAru
17. aseVsIlaku kalige ibbaMxulaku woVlagiMcadaM pannu eVgavewa moVxalEna
17. muKyaviRayAla guriMci wlsukovalasina caryalanu sUciMcamani praBuwvaM I vicAraNa
17. saMGAnni koriMxi
18. ArWika cattaMlo pannu minahAyimpu parimiwi
19. A niRpawwini nirNayiMcadAniki rARtrapawi prawi ayixu saMvawsaraMla koVKasAri oVka
19. kamitlni niyamiMci vAri siPArsula meraku AxAyaM pannu viBajimcadaM jaruguwuMxi
20. pannu nirXaraNa saMvawsaraM seVkrAn 2(9) prawi saMvawsaraM epril Iva wexl nuMci
20. prArAMBamayye 12 neVlala kAlAnni pannu nirXaraNa saMvawsaramani aMtAru
21. uxAharaNaku praswuwa pannu nirXaraNa saMvawsaraM epril 1,1988 lo prArAMBamayi
21. mArCi 31, 1989 lo aMwamavuwuMxi
22. pannu nirXaraNa saMvawsarAnni pannu saMvawsaraM ArXika saMvawsaraM anikUda uMtAru
23. aseVs! gawa saMvawsaraMlo saMpaXiMcina AxAyaM mlxa AxAyapu pannu aXikArulu pannu
23. nirXaraNa saMvawsaraMlo pannu viXiMci vasUlu ceswAru
24. I cattaMlo AxAyapu pannu retlu nirNayiswAru
25. uMte vAtini kUdA leVkkaloki wlsukoVni pannu viXiswAru
26. sUcana @Sec3(4) prakAraM oVKasAri nirNayiMcina gawa saMvawsarAnni AxAyapu pannu
26. aXikAri anumawi lekuMdA mArcarAxu
27. javAbu I vyApArAlaku pannu nirWArana saMvawsaraM 1988-89 ki sari ayina gawa
27. saMvawsarAlu I viXaMga uMtAyi
28. ravANA ceyadAniki vasUlu cesina soVmmulo 7.5% awani AxAyaMga parigaNiMci pannu
28. vasUlu ceswAru
29. varaku mAWrame fAxAyAnni ArjiMcinappatiki BARawaxeSaM vaxili veVlylYe muMxu A
29. AxAyaMpE pannu #ceVlliMcamani AxAyapu pannu aXikAri AxeSiMcavaccu
30. uxyogula jIwAlalo nuMci yajamAni AxAyaM pannu waggiMci praBuwva KajAnAlo dipAjit
30. ceVyyAli
31. AxAyaM seVkrAn2 AxAyapu pannu oVka aseVs! AxAyaM mlxa AXArapadi viXiswAru
32. pannu viXiMpu xqRtyA AxAyaM ane paxAniki eVMwa prAmuKyZyamunnaxi
33. AxAyaM pannu cattaM seVkrAn2 lo kiMxi aMSAlanu AxAyaM ane paxaM kiMxa cercAru
34. seVkrAn 28 lexA seVkrAn 41 lexA seUkrAn 59 kiMxa AxAyapu pannu viXiMce AxAyAlu
35. oVka vyakwiki AxAyaM svlkariMce hakkuuMte, A AxAyaM vasUlu cesukokapoyina,
35. hakkulexA saMciwa prAwipaxikape xAni mlxa pannu viXiMcavaccu
36. catta sammawaM kAni AxAyaM mlxa kOdA pannu viXiswAru
37. AxAyaM hakkupE vivAxaM AxAya vanaru guriMci vivAxaM exEnA unnappudu pannu
37. ceVlliMpu kAlaM poVdagiMcadaM jaragaxu
38. varaku pannu minahAyimpu uMdexi
39. I AxAyAlanu iwara vanarula AxAyaM SiRika kiMxa pannu viXiswAru
40. pannu nirWArana saMvawsaraM nuMci vAti mlxa pannu minahAyimpu parimiwi
41. vyavasAya AxAyaM seVkrAn 2(1) 1961 AxAyaM pannucattaM seVkrAn 10 (1) prakAraM
41. vyavasAyAxAyaMpE fpannu minahAyimpu vunnaxi
42. AxAyaMpannu BARawaxeSaMlo praveSa peVttinappati nuMci vyavasAyAxAyAniki pannu
42. minahAyimpu laBiswunnaxi



- 13 I koVwwa viZAnaM PrakAraM vyavasAya AxAyAnni pannu retu koVraku #mAwrame  
leVkkalokiwlisukoVni aseVsIki pannu vixiMcadaM jarugunu
- 44 anagA aseVs! yoVvka vyavasAyewara AxAyaM pannu viXiMpu pariXilo vunnappudu  
vyavasAya AxAyAnni retu koVraku mAwraM leVkkaloki wIsukuMtAru
- 45, aMxZavalla alAMti AxAyAniki pannu minahAyimpu lexu
- 46, iMxuku AxAyapu pannu nibaMXanalu 1962 loni 7,8 niyamAlu varwiswAy
- 47, aMte A moVwwaM vyavasAya AxAyaM kanaka xAni mlxa pannu lexanna mAta
- 48, vyavasAya yZAxAyaM mlxa AxAyapu pannu lenappatikl A AxAyaM unna fvAri iwara  
AxAyaM mlxa eVkkuva pannu viXiMce uxxeSyamwo PEnAns cattAniki 1973 #lo cesina  
savaraNa prakAraM 1974-75 aseVnmeVMt saMvawsaraM nuMci vyavasAya AxAyAnni moVwwaM  
AxAyaMlo cerci pannu retu nirNayiswAru
49. A aXikaretu prakAraM iwara AxAyaM mlxa mAwrame pannu viXiswAru
50. kanlsa pannu viXiMpu AxAyAnni miMcina vyavasAyewara AxAyaM uMtene vyavasAya  
AxAyAnni pannuretu kosaM kalupuwunnAru
51. xlni mlxa AxAyaM pannuretla prakAraM pannu guNiMcAli
52. vyavasAyewara AxAyaM eVMwuMte pannu asalu uMdaxo AmoVwwAniki vyavasAya AxAyAnni  
kalapAli
53. xlnimlxa AxAyapu pannuretla prakAraM pannu viXiMcAli
54. A nilava aseVs! ceVllimcavalasina pannu
55. AxAyaM vanarunu svlkariswunna vyakwi pannu ceVllimcavalasina bAXyawa uMtumXi
56. xAniki pannu noVppi ayiwe xAnini plki pAreswe powuMxi
57. pannu kattakapowe poVlAnnuMci woVlagiMcinaAru
58. e kAraNam cewaniyinaA pannu wlyiMcina warvAwa rakwa srAvamu AgakuMdA vuMte  
eVmamalis, 200 kriyosotu 200 mArcucu gaMtakoVkasAri iwavaleVnu
59. pannu xAni svarUpaM pannuxavada eVmukalo nuMci moVlucukuni vaswuMxi
60. pannu kriMxi BagaMlo cinna raMXraM uMtumXi
61. pEki kanipiMce pannu aMwA eVnAmil ane paxArXaM xvArA kappi uMcabaduwuMxi
62. gatti AhAraM winnappudu pannu arigipokuMdA kApAduwuMxi
63. puttukawone pannu walli garBaMlo uMdagAne SiSuvuku xaMwAlu moVlakeVwwuwAy
64. ayiwe puttukawone pannu unnappudu walliki pAlu iwadAniki bAXaga uMtumXi
65. pannu glsukupowuMxi
66. aMxukani A pannu plki veyadaM avasaraM puttinappude pannu plkiwe exo avuwuMxani  
BayapadanavasaraM lexu
67. pannu plklNa rAvalasina samayAniki malyH pannu vaswuMxi
68. pannu vubbi povadAnne kerls tUw lexA pippi pannu AmTaRU
69. pippi pannu cinnapillallo wvaragA erpaduwuMxi
70. pannu muKyaMga mUdu poVrala sammelyanaM
71. pannu pEpoVra eVnAmil, xAni kriMxa poVra deVMtin
72. paMti pEpoVra eVnAmil xeVbbawinadaMwo pippi pannu erpaduwuMxi
73. sUkRmakrimula valla wayArEna lAktik yAsid eVsitik yAsidle pippi pannu  
erpadatAniki praxAna kAraNaM
74. pippi pannu xuRpaliwAlu pippi pannuni nirlakRyaM ceswe bAktlriyA krimulu paMti  
lopalaki praveSiswAy
75. aMxukane pippi pannu koVMxariki xavada vAcipovadaM, xavada eVmukalo clmu ceradaM  
jaruguwuMxi
76. pippi pannu moVxati xASalo callati nllYlaki, pulupuki palyLu jiwumanadame uMte  
nixAnaMga palp kuharaM vAci bAgA noVppi kaluguwuMxi
77. pippi pannu vaswe veVMtane dAktaruki cUpiMcukovAli
78. pippi pannu potu uMte Asprin eVnAljin EbrUPeVn vaMti bilYlalu wadavaki oVkati  
coVppuna rojuki 3-4 sArlu vAdAli
79. A krimula valla pippi pannu erpadiMxani nammakaM kaligiswAru
80. pippi pannu waggadAniki paMtilo pasarlu piMduwAru
81. pippi pannu uMte kUda xurvAsana rAvaccu
82. cigulYla vApu, pippi pannu, notilo puMdu uMte vAtiki wagina cikiwsa ceyAli
83. xavada eVmukalanuMci pannu pEki vaccina cota ciguru AkramiMcukuni uMtumXi
84. nixAnaMga pannu kaxuluwuMxi
85. koVMwakAlAniki pannu UdipowuMxi
86. ciguru vApu xOda eVmukalo pannu sWiraMga, gattiga uMdatAniki paMti ciguru cAla  
avasaraM
87. ciguru vyAWivalla pannu kaxuluwuMxi
88. SASvawa pannu Udiwe A sWalaMlo kroVwwa pannu moVlakeVwwadaM uMdaxu
- 89 xAniwo paMti patuwvaM poyi pannu kaxaladaM movxalavuwukaMxi
- 90, PaliwaMga pannu kaxaladaM UdipovadaM jaruguwuMxi
- 91 pippi pannu erpadina vArilo bAktlriyAkrimulu paMti kuxurulo uMde peVriyodeVMtal  
poVrani kRINiMpajesi paMtini balahlnaparuswAyu
- 92 xAniPaliwaMga pannu UgisalAdataM UdipovadaM avuwuMxi
- 93 ciguru vApuwopAtu pannu unna xOda eVmuka BagaMlo clmu ceruwuMxi
- 94 pippi pannu aXikaMga unna cotla wrAgenltini visleRiMci aMxulo
- 95 piccixAna I edu munsipAltiki nllYlYa pannu katta vaxxu

## APPENDIX-2

### Domain Independent Stop-words

MtAwi ani reVMdu ixi kAni uMtuMxi koVnni wana axi lo e ane cAla lexA nenu nA aMte ml MtAyi unna lexu mana nuMdi aMtAru eVkkuva mlxa vAru guriMci maXya nuMci xvAra kAxu ayiwe uMxi xAni avi vAri mAwrame ivi waroa prawi leka pani aneka viXaMga waruvAwa gala Ayana kAbatti peVxxa valla mUdu mlru va viviXa yoVka manaM peji nAku awani weVlugu mA cinna nl valana anni Sri vAtini iwara gAni mariyu kUda muMxu vAti eVMwa ami warvAwa oVke pE AmeV xlnini annAdu cesina vuMxi koVMwa kAni xAnini baGa awadu saMbaMXiMcina cesi ila varaku moVxati vaccina mAwraM vuMtuMxi jarigiMxi vAriki eVkkuvaga kaMpeVnl oVkati viRayaM unnAyi peru axe wakkuva xAniki uMde vacciMxi ala vyakwi maMci ippudu xlni moVxalEna iMka vacci wayAru batti ceVMxina yl eVla avasaraM vaMti kosaM viluva Sakwi wirigi paxXawi ru nlru cese manaku eVMwo jaruguwuMxi nAlugu appudu arWam xlniki pUrwiga vunna aMwa o ga ikkada aMtU lekuMdA moVwwaM prawyeka ayina eM ayina veVMtane koVMxaru malYLYI akkada nannu exo koVwwa kriMxi unnaxi cAla aMxulo praBuwwaM alAge asalu aMwa mAta uMdAli ceyadaM uMte veru prakAraM lanu avuwuMxi ArWika krl viRayaMlo reVMdava kaxA oVka uxA aBivqxXi vaswuMxi rAjaklya vltini anaga ki lu xAnni kAlaMlo vacce saMKya mlku vaxxa pEna walli kAvalasina maMxi mari kanuka kAraNaM Sa BUmi erpAtu waravAwa uMdunu BAra maroVka anexi aMxuvalla sAXAraNaMga BARawa eVMxuku si iMxulo awi muKya prajalu uwpawwi vAdu aMxuru moVxata anna cOsi maniRi AyA xaggara vyavasWa uMdi praBuwwa kalisi wAnu muKyaMga sAhiwya AXunika ni eVnno vuMtAyi rAwri kAka mAtalu pEki migilina vaccunu leni kakuMdA ku gAli bi wappa uxAharaNaku kiMxi nlku mArpu oVkoVka annAru rU ceSAru ika nlti kAlaM dabbu kaligi uMdaxu saMbaMXaM ituvaMti sAmAjika Srama weVlusu kalipi kevalaM pillalu lABaM paxXawilo jarigina muKyamEna nuwu ixe wagina di avakASaM swrl xeSaMlo praXana pUrw prajala awaniki wuwpawwi mArpulu vyApAra aMxi BARA nlru aXika vArini veVLYLYi iMgllRu vALyLYu la awanu kAMwi cariwra kiMxa saMsWa gAni kAvuna eVnni waMdri kriMxa eV nltilo ceswAru vaswuvu maro levu sAMGika inAnava AXArapadi emiti i jAnapaxa goVppa kaWa kaMteV sAXAraNa mAnavudu sumAru ceyadAniki erpaduwuMxi pariSoXana prAcIna vi wala samAjaMlo vlti wo ceswuMxi rAju loni weVlugulo lalo vixyuv roju carya vaccu ceyAli aMxaru moVxalagu viRayAlu jarige wanu muMxuku ceVppina vltilo vAtiki eml AMXra yuMdunu gUrci Karcu varakU uwara atuvaMti vunnAyi xAxApu manamu saMgawi viRayAnni samayaMlo vlru rakAla anni laku mahA peVtti cUwe aMxuke rUpaMlo BagaM xqRtilo wanaku aMwe koVMceVM sAManya sariga axanapu lekapowe BARalo prapaMca peVttubadi mEna pi Ata moVxalEnavi iMwa cUci eVppudU SAwaM InAdu memu Bagamu ceSAdu erpadina xlnilo itlu vuMte parisWiwi nannaya vALyLYa uMtAru wuMxi muMxuga keV saruku paxi moVxalu maMcixi pAtu jAwi cuttU a ravANA viXanaM xakRiNa eVvaru rAjyaM cesiMxi cewa seVM BARawIya kAVali rAjanlwi ceswe bayataku saMskgwa avunu moVkkalu uMdexi vesi ceVppi gawa ceVppadu ayiMxi kaMte gAru kavi end. uMdavaccu ameVrika BArya nUwana vlri prajalaku keMxra iLAMti uxayaM prakgwi uRNograwa sare vaswuvulu iMtiki baXyawa viXamuga rojulu reVMdo saMvawsaram gAka vAni vAdi spaRtaMga BOwika dA pannu vAtilo pari panulu ti cUswU vuMdAli wlsukoVni wlsi lakRaNaLu iccina oVkasAri erpadawAyi moVkkala poyiMxi amalU praBAvaM kqRi aMxuvallana aMxuku yaMxu prema mUla rediyo pra weVliyaxu ceswU vaswAyI raMgu aMxu exEna migawa gopi kaligina pAwra saM iMko anl vEpu mulu vyAXi ceVppavaccu ixaru sArlu veVnuka vArilo mAku BARawaxeSaMlo nltini avasaramEna marl rAjya mawa eVkkada Aru reVMdu jIviwaM kavu kalamu unnAru rasAyanika kAvaccu kalupu ne AhAra paxArWalu xASalo Bavana cesulu pataM iMtI wakkuvaga itti paxAlu eVlaktrAn xlnine alAMti lopala xAnilo kAswa weVluswuMxi moVka wvaraga kaWalU ninnu xeSa vini rARtra viRayAlanu poyi jAwIya neti nirmANaM vivarAlu kanlsAM unnappudu vela vaccadu wappu eVwu gattiga kAryAlaya vAnini Sakwini nedu ganuka aroma praSna kalavu SASwra vuMde ceyataM kalYLYu pEga ceVMxi javAbu erpadi munu xlnni aMxarikl SarlraM wirupawi iMtlo saMvawsarAla valeV baMWini panicese leci lilll koVni unnavi viBajana mAnasika prayawnaM Akulu sarEna na xAniwo E Sarlra paxXawini jAgrawwaga kannA mAmUlu yani jIviwa kaNaLu exi aMxucewa dAktaru mAwramu akRarAlu samasya ceyavaccu kalaxu AnaMxaM adigadu ceswAyI prapaMcaMlo paxaM ceVppu mani prayANaM marikoVnni BagaMlo bayata APISu vaccAyI kramaMga kaRtaM vedi nunna ceri saMskgwi vaswe naxi ceVppAiru samAcAraM SabxaM vrAyumu rAjyam mariMwa pAlu vaswuvula paxXawulu kUdina appude gUda ayi vuMdaxu xaggaraku waragawi civaraku cakkaga samAcArAnni BayaM oVkate unnawa SAMwi eVppudu mAnavuni peVLYIYi dAniki mUdava ceya awyaMwa weVlupu samAXAnaM rojuku perlu padi ceyunu xUraMga caryalu aMxukani powuMxi wedA SASwraM Barwa poVdavu erpadunu itla pAriSrAmika rasAyana xUraM galavu uMdataM amna pAwa wAmu sixXAMwaM baHya praSnalu cattaM illu kanaka panini paMwulu cUdaMdi AxAyAM sUkRma sulaBaMga pUrvaM nyAya ivannl cota iwadu sAhiwyaM prAMwaMlo nAti sa ti caxuvu aBiprAyaM muMxe marala prAWamika jilla coVppuna gaxA hiMxi lexani rakRaNa nirvahaNa ceVppiMxi Alocana Xara koVxxi AhAraM Amlamu baXA pollsu civari kroVwwa AnAti rakaM neVnraiaxiga cakkani rAlexu ji uwarama avasaramu vere mikkili nijAniki xeSaM unnatlu ayixu saMgaM maMwri peVrigi Aksijan samAjaM menejmeVMT koVxxiga aByAsamu BARanu poVMxina rakarakAla sarigga rojululo pAtalu SAKa prawyekaMga sAMkewika puswakaM xAri atu vaswuvulanu ceVppAli grahiMci aXikaMga rAvadaM praswuwam aXyayanaM raMga jIviwaMlo saMsWalU kanipiswuMxi aNuvulu rakaMga moVwwamu Exu Exu

### APPENDIX-3 (Word-sets)

1. AxAyapu pannu
2. aseVs! pannu
3. minahAyimpu pannu
4. nirXAraNa pannu
5. pannu saMvawsaraM
6. pannu viXiMce
7. AxAyAnni pannu vyavasAya
8. AxAyaM pannu viXiswAru
9. AxAyaM pannu vyavasAya
10. pannu AxAyAnni AxAyaM
11. pannu AxAyaM AxAyapu
12. pannu AxAyaM aseVs!
13. pannu AxAyapu aseVs!
14. pannu AxAyapu prakAraM
15. pannu AxAyapu viXiMce
16. pannu nirXAraNa saMvawsaraM
17. pannu prakAraM vyavasAya
18. pannu AxAyaM pannu prakAraM
19. UdipovadaM pannu
20. clmu pannu
21. eVmukalo pannu
22. kaxaladaM pannu
23. kaxuluwuMxi pannu
24. nixAnaMga pannu
25. pEpoVra pannu
26. pannu xavada
27. ciguru pannu xOda
28. erpaduwuMxi pannu pippi
29. paMti pannu pippi
30. pannu pippi uMte
31. pannu UdipovadaM kaxaladaM
32. pannu clmu ciguru
33. pannu clmu eVmukalo
34. ciguru eVmukalo
35. pannu ciguru kaxuluwuMxi
36. pannu ciguru paMti
37. pannu eVmukalo xOda
38. eVmukalo xavada
39. pannu erpaduwuMxi pEpoVra
40. pannu kaxaladaM paMti
41. pannu nixAnaMga pippi
42. pannu pEpoVra paMti
43. pannu pippi xavada

## APPENDIX 4

### Domain dependent stop-words ( Teeth)

3-4 AhAraM AmTaRU BayapadanavasaraM PaliwaMgA SiSuvuku Udiwe aMwA aMxukani arigipokuMdA avuwuMxi bAktlriyA balahlnaparushAyu cUpiMcukovAli ceradaM ceswe ceyAli cikiwsa coVppuna coda deVMtin eVmuka eVnAljin eVsitik erpadatAniki erpadina gaMtakoVkasAri gatti iwadAniki jiwumanadame kAraNaM kRJNiMpajesi kaluguwuMxi kappi kattakapowe koVMwakAJAniki kriMxa krimulu kroVwwa kuxurulo lopalaki malYII moVlakeVwwwuAyi moVxalavuwukaMxi munsipAltiki nllYlaki, nirlakRyaM pAlu plknA paMtilo palYlu pannu, pannuxavada patuwvaM peVriyodeVMtal piccixAnA poVra poVrani povadAnne poyi praxAna pulupuki rAvaccu raMXraM rojuki sUkRmakrimula sWiraMgA, sammelYanaM svarUpaM uMcabaduwuMxi uMdatAniki unnappudu vAcipovadaM, vApu vApuwopAtu vAtiki vaxxu viSleRiMci vubbi wlyiMcina waggadAniki walli wayArEna woVlagiMcinAru wvaragA xAniwo xaSalo xuRpaliwAlu yAsid yAsidle

### Domain dependent stop-words ( Tax)

#AxAyAnni #/o #pannu 1,1988 1000 1918 1961 1973 1981 1987-88 19892 200 31,59 7.5% AXArapadi Alocana ArXika AxAyAlanu AxAyAniki AxAyaMpE AxAyapupannu BArAnni BArawaxeSaMlo PEnAns aMwamavuwuMxi aXikAmlu alAMti anagA anukunnAdu appagiMcAru aseVnmeVMt aseVsllaku ayixu bAgunnaxani cAtiMpu cattAniki catta ceVMxinaxi ceVyyAli cercAru cesukokapoyinA, dipAjit eVMwuMte eVksEj exEnA guNiMcAli hakkupE iMxuku iwavaleVnu jIwaM jarugunu kALAnni kAraNaMcewa kalapAJi kalupuwunnAru kamitlni kanaka katti ki koVkasAri kovaku ku laBiswunnaxi lexanna mArcarAxu mArcucu mlxane maMwrulu mamAde miMcina moyaka muKyaviRayAla nAnnaku niRpawwini nilava nirNayiMcina niyamiMci paMdiMci pannucattaM pannulaku pannuretu parigaNiMci paxAniki peVrigiMxi pilipiMcAraMdi poVLAnnuMci prAmuKyaM prAraMBamayye praBuwwa prajala praswuwa rARtrapawi rUpoVMxiMcAru ravANA AmAnyu sUciMcamani saMciwa saMva saMvawsarAlu saMvawsaramani sammawaM sari savariMci, soVmmulo suMkaM svlkariMce uMdaxo uMtAru uMtene uxAharaNaku uxxeSiMci uxyogula vanaru vanarunu vasUIYIYanu veSAnani veyiMcAru viXAnaM viXiMcavaccu viXiswAro vivAxaM vqwwirlwyA vunnAnaMdi vunnaxi vyApAralwyA, vyakwini vyavasAyAxAyaMpE wlsukovalasina wlyiMcina waggiMci wexl woVlagiMcinAru xlnimlxa xqRtyA yAxqcCika yZArjiMcina

## References

- [Abbott 89] Abbott, L. P., Kepler, T. B. Universality in space of interactions for network models. *Journal of Physics A Math. Gen.*, **22**, 2031-2038, 1989.
- [Agrawal 93] Agrawal, R., Imielinski, T., Swami, A. Mining association rules between sets of items in large databases. *Proceedings of ACM SIGMOD Conference on Management of Data (SIGMOD'93)*, 207-216, 1993.
- [Altschul 94] Altschul S. F., Boguski M. S., Gish W., Wootton J. C. Issues in searching molecular sequence databases. *Nat. Genet.*, **6**, 119-129, 1994.
- [Anderson 88] Anderson, J. A., Rosenfield, E. Neurocomputing: Foundations of Research. *MIT press*, Cambridge, Mass, 1988.
- [Anderson 92] Anderson, J. A. Foreword, in Kosko, B. Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence. *Prentice-Hall International*, New Jersey, 1992.
- [Athithan 95] Athithan, G. A comparative study of two learning rules for associative memory. *Pramana Journal of Physics*, **45**, 569-582, 1995.
- [Baram 89] Baram, Y. Associative memory in fractal networks. *IEEE Transactions on SMC*, **19**, 5, 1989.
- [Brown 91] Brown, P. Della-Pietra, S., Della-Pietra, V., Mercer, R. Word sense disambiguation using statistical methods. *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, 264-270, 1991.
- [Bruce 94] Bruce, R., Wiebe, J. Word sense disambiguation using decomposable models. *Proceedings of the 32nd Annual Meeting of Association for Computational Linguistics, Las Cruets, NM*, 139-145, 1994.
- [Carpenter 88] Carpenter, G. A., Grossberg, S. The ART of adaptive pattern recognition by self-organizing neural networks. *IEEE Computer*, 77-88, 1988.
- [Cantalloube 95] Cantalloube H., Labesse G., Chomilier J., Nahurn C., Cho Y.Y., chams V., Achour A., Lachgar A., Mbika J. P., Issing W., Moron J. P., Bizzini B.; Zagury D. Automat and BLAST: comparison of two protein sequence similarity search programs. *Comput. Appl. Biosci.*, 1995.

- [Changd 98] Change, J.S., Chen, J. N. Topical clustering of MRD senses based on information retrieval techniques. *Computational Linguistics*, 24, 61-97, 1998.
- [Christodoulakis 84] Christodoulakis, S., Faloutsos, C. Design considerations for a message server. *IEEE Transactions on Software Engineering*, 2, 201-210, 1984.
- [Collier 97] Collier, N. Convergence time characteristics of an associative memory for Natural Language Processing. *Proceedings of IJCAI 97, 15th Joint Conference on AI*, Nagoya, Japan, 1106-1111, 1997.
- [Cottrell 83] Cottrell, G. W., Small, S. L. A connectionist scheme for modelling word sense disambiguation. *Cognition and Brain Theory*, 6, 89-120, 1983.
- [Dagan 94] Dagan, L, Itai, A. Word sense disambiguation using a second language monolingual corpus. *Computational Linguistics*, 20, 4, 563-596, 1994.
- [Dahlgren 88] Dahlgren, Kathleen G. Naive Semantics for Natural Language Understanding. *Kluwer Academic Publishers*, Boston, 1988.
- [Deiderich 87] Deiderich, S., Oppen, M. Learning of correlated patterns in spin-Glass networks by local learning rules. *Journal of Physics A Math. Gen.*, 58, 9, 949-952^1987.
- [DeRose 88] S. J. DeRose. Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 14, 31-39, 1988.
- [Faloutsos 85] Faloutsos, C. Access methods for text. *ACM Computing Surveys*, 17, 1, 49-74, 1985.
- [Farrell 90] Farrell, J. A., Michel, A. N. A synthesis procedure for Hopfield's continuous-time associative memory. *IEEE Transactions on Circuits and Systems*, 37, 877-884, 1990.
- [Forrest 89] Forrest, B. M. Content addressability and learning in neural networks. *Journal of Physics A Math. Gen.*, 21, 245, 1989.
- [Gale 92] Gale, W., Church, K., Yarowsky, D. A method for disambiguating word senses in a large corpus. *Computers and the Humanities*, 26, 415-439, 1992.
- [Gardner 89] Gardner, E., Gutfreund, H., Yekutieli, I. The phase space of interactions in neural networks with definite symmetry. *Journal of Physics A Math. Gen.*, 22, 1995-2008, 1989.
- [Guthrie 91] Guthrie, J. A., Guthrie, L. Wilks, Y., Aidinejad, H. Subject-dependent co-occurrence and word sense disambiguation. *Proceedings of the 29th Annual Meeting of the ACL*, 146-152, 1991.

- [Hall 80] Hall, P. A. V. Dowling, G. R. Approximate string matching. *ACM Computing Surveys*, 12, 4, 381-402, 1980.
- [Han 89] Han, J. Y., Sayeh, M. R., Zhang, J. Convergence and limit points of neural network and its application to pattern recognition. *IEEE Transactions on SMC*, 19, 1217-1222, 1989.
- [Hayes 78] Hayes, Philip, J. Mapping input into schemas. *Technical Report 29, Department of Computer Science, University of Rochester*, 1978.
- [Haykin 94] Haykin, S. Neural Networks: A Comprehensive Foundation. *Macmillan College Publishing Co.*, New York, 1994.
- [Hebb 49] Hebb, D. O. The Organisation of Behavior. *John Wiley and Sons, New York*, 1949.
- [Hecht-Nielsen 91] Hecht-Nielsen, R. Neurocomputing. *Addison-Wesley Publishing Company*, MA, 1991.
- [Hilberg 97] Hilberg, W. Neural networks in higher levels of abstraction. *Biological Cybernetics*, 76, 23-40, 1997.
- [Hollaar 83] Hollaar, L. A., Smith, K. F., Chow, W. H., Emrath, P. A., Haskin, R. L. Architecture and operation of a large, full-text information retrieval system. *Advanced Database Machine Architecture*, D. K. Hsiao, Ed. *Prentice Hall, Englewood Cliffs, New Jersey*, 256-299, 1983.
- [Hopfield 82] Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of National Academy of Sciences, USA*, 79, 2554-2558, 1982.
- [Hopfield 84] Hopfield, J. J. Neurons with graded response have collective computational abilities like those of two state neurons. *Proceedings of National Academy of Sciences, USA*, 81, 3088-3092, 1984.
- [Ide 90] Ide, N., Veronis, J. Very large neural networks for word sense disambiguation. *proceedings of the 9th European Conference on Artificial Intelligence, ECAI'90, Stockholm*, 366-368, 1990.
- [Ide 98] Ide N., Veronis, J. Introduction to the special issue on word sense disambiguation: The state of the art. *Computational Linguistics*, 24, 1-40, 1998.
- [Jain 96] Jain, A. K., Mao, J., Mohiuddin, K. M. Artificial neural networks: A tutorial. *IEEE Computer*, 29, 31-44, 1996.

- [Kam 90] Kam, M., Jeng-chieh, C., Robert, F., Design of the fully connected binary neural network via linear programming. *IEEE International Symposium on circuits and Systems*, USA, New Jersey, 2, 1094-1097, 1990.
- [Rang 93] Rang, H. Multilayer Associative Neural Networks (M. A. N. N): storage capacity Vs noise-free recall. *International Jt. conference on Neural Networks (IJCNN)* 93, 901-907, 1993.
- [Karnin 90] Karnin, E. D. A simple procedure for pruning backpropagation trained neural networks. *IEEE Transactions on Neural Networks*, 1, 2, 239-242, 1990.
- [Karov 98] Karov, Y. Shimon, E. Similarity-based word sense disambiguation. *Computational Linguistics*, 24, 41-59, 1998.
- [Knott 75] Knott, G. D. Hashing functions. *Journal of Computer*, 18, 3, 265-278, 1975.
- [Knuth 73] Knuth, D. E. The Art of Computer Programming Vol.3: Sorting and Searching. *Addison Wesley*, Reading, Mass, 1973.
- [Knuth 77] Knuth, D. E., Morris, J. H., Pratt, V. R. Fast pattern matching in strings, *SIAM Journal of Computing*, 6, 2, 323-350, 1977.
- [Kohonen 84] Kohonen, T. Self-organization and Associative Memory. *Springer-Verlag, Berlin*, 1994.
- [Kohonen 88] Kohonen, T. An introduction to neural computing. *Neural Networks*, 1, 3-16, 1988.
- [Kosko 92] Kosko, B. Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence. *Prentice-Hall International*, New Jersey, 1992.
- [Krauth 87] Krauth, W., Mezard, M. Learning algorithms with optimal stability in neural networks. *Journal of Physics A*, 20, 745, 1987.
- [Krovetz 92] Krovetz, R., Bruce C. W. Lexical ambiguity and information retrieval. *ACM Transactions on Information Systems*, 10, 2, 115-141, 1992.
- [Larson 83] Larson, P. A. A method for speeding up text retrieval. *Proceedings of the ACM SIGMOD Conference*, ACM, New York, 1983.
- [Leacock 93] Leacock, C., Towell, G., Voorhees, E. Corpus-based statistical sense resolution. *Proceedings of the ARPA Workshop on Human Language Technology*, 1993.



- [Leacock 96] Leacock, C., Towell, G., Voorhees, E. M. Towards building contextual representations of word senses using statistical models. *Branimir Boguraev and James Pustejovsky, editors, Corpus Processing for Lexical Acquisition*, MIT Press, Cambridge, MA, 97-113, 1996.
- [Lesk 78] Lesk, M. E. Some applications of inverted indexes on the UNIX system. *UNIX programmers manual, Bell Laboratories*, Murray Hill, New Jersey, 1978.
- [Lesk 86] Lesk, M. Automatic sense disambiguation: How to tell a pine cone from, an ice cream cone. *Proceedings of the 1986 ACM SIGDOC Conference*, 24-26, 1986.
- [Masterman 62] Masterman, M. Semantic message detection for machine translation, using an interlingua. *1961 International Conference on Machine Translation of Languages and Applied Language Analysis*, Her Majesty's Stationery Office, London, 437-475, 1962.
- [McCulloch 43] McCulloch, W. S., Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biosciences*, 5, 115-133, 1943.
- [McIlroy 82] McIlroy, M. D. Development of a spelling list. *IEEE Transactions of Communications COM-30*, 1, 91-99, 1982.
- [McRoy 92] McIlroy, S. W. Using multiple knowledge sources for word sense discrimination. *Computational Linguistics*, 18, 1, 1-30, 1992.
- [Mead 89] Mead, C. Analog VLSI and Neural Systems. *Addison-Wesley, Reading, MA*, 1989.
- [Michel 89] Michel, V., Bruno, S., Andre, V., J. Paul, G. A. High-storage capacity content-addressable memory and its learning algorithm. *IEEE Transactions on circuits and systems*, 762-766, 1989.
- [Miller 90] Miller, G. A., Ed. WordNet: An on-line lexical database. *International Journal of Lexicography*, 3, 4, 235-244, 1990.
- [Miller 94] Miller, G. A., Martin, C., Landes, S., Leacock, C., Robert, G. T. Using a semantic concordance for sense identification. *Proceedings of the ARPA Workshop on Human Language Technology*, Plainsboro, New Jersey, 240-243, 1994.
- [Mooney 96] Mooney, R. J. Comparative experiments on discriminating word senses: an illustration of the role of bias in machine learning techniques. *Proceedings of the 2nd conference on Empirical methods in Natural Language Processing*, 1996
- [Muller 90] Muller, B., Reinhardt, J. Neural Networks: An Introduction. *Springer-Verlag, Heidelberg*, 1990.

- [Murray 91] Murray, A. F. Silicon implementation of neural networks, *IEE Proceedings PartF*, **138**, 1,3-12, 1991.
- [Nemhauser 88] Nemhauser, G. L., Wosely, L. A. Integer and Combinatorial Optimization. *John Wiley & Sons Ltd*, New York, 1988.
- [Nigharntuvu 79] Sree Suryandhra Nighamtuvu. *Reprint of 1939 edition, published by A.P. Sahitya Academy*, 1979.
- [Ng 96] Ng, H. T., Lee, H. B. Integrating multiple knowledge sources to disambiguate word sense and examplar based approach. *Proceedings of the 34th Annual Meeting (ACL)*, University of California, Santa Cruz, CA, 1996.
- [Ozawa 93] Ozawa, S., Tsustsumi, K. Association performance of cross-coupled Hopfield nets for correlated patterns. *Proceedings of 1993 International Joint Conference on Neural Networks*, pub. *IEEE*, New Jersey, USA, 2, 2335-2338, 1993.
- [Patrick 85] Patrick, A. B. An exploration of abstract thesaurus instantiation. *M.Sc. thesis*, *University of Kansas*, Lawrence, KS, 1985.
- [Pujari 83] Pujari, A. K., Mittal, A. K., Gupta, S. K. A convex polytope of diameter one. *Discrete Applied Mathematics*, 5, 241-242, 1983.
- [Rabitti 84] Rabitti, F., J. Zizka. Evaluation of access methods to text documents in office systems. *Proceedings of the 3rd joint ACM-BCS symposium on research and development in information retrieval*, Cambridge, Mass, 21-40, 1984.
- [Reed 93] Reed, R. Pruning algorithms-a survey. *IEEE Transactions on Neural Networks*, 4, 5, 740-747, 1993.
- [Resnik 95] Resnik, P. Disambiguating noun groupings with respect to WordNet senses. *Proceedings of the Third Workshop on Very Large Corpora*, Cambridge, MA, 54-68, 1995.
- [Rosenblatt 58] Rosenblatt, F. The Perceptron: A probabilistic model for information storage and organization in the Brain. *Psychological Review*, **65**, 386-408, 1958.
- [Rumelhart 86] Rumelhart, D. E., Hinton, G. E., McClelland, J. L. A General Framework for Parallel Distributed Processing. *D.E. Rumelhart and J. L. McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Cambridge, MA:MIT Press, 1, 45-76, 1986.
- [Sacks 87] Sack-Davis, R., Kent, A., K. Ramamohanarao. Multikey access methods based on superimposed coding technique. *ACM Transactions on Database System*, 12, 4, 655-696, 1987.

- [Salton 83] Salton, G, Fox, E. A., Wu, H. Extended Boolean information retrieval. *Commun. ACM*, 26, 11, 1022-1036, Nov, 1983.
- [Shang 96] Shang H., Merrettal T. H. Tries for approximate string matching. *IEEE Transactions on Knowledge and Data Engineering*, 8, 4, 540-547, 1996.
- [Sharma 94] Sharrna, R. Investigation of capacity and dynamics of Hopfield model of neural networks. *Ph.D. Thesis, Dept. of CIS, University of Hyderabad, India*, 1994.
- [Sietsma 91] Sietsma, J., Dow, R. J. F. Creating artificial neural networks that generalize. *Neural Networks*, 4, 1, 67-69, 1991.
- [Smadja 89] Smadja. F. A. Lexical co-occurrence: The missing link. *Literary and Linguistic Computing*, 4, 3, 1989.
- [Small 80] Small, S. L. Word expert parsing: A theory of distributed word-based natural language understanding. *Ph.D. Thesis, Department of Computer Science, University of Maryland*, 1980.
- [Smith 96] Smith, K., Palaniswami, M. Krishnamoorthy, M. Hybrid neural approach to combinatorial optimization. *Computers & Operations Reassert*, 23, 6, 597-610, 1996.
- [Sreenivasa 97a] Sreenivasa Rao, M., Pujari, A. K., Sreenivasan, B. A new neural network architecture for efficient close proximity match of large databases. *IEEE Computer Society Press, Proceedings of the Eighth International Workshop on DEXA, France, Edited by R. R. Wanger*, 444-449, 1997.
- [Sreenivasa 97b] Sreenivasa Rao, M., Pujari, A. K. Word sense disambiguation using artificial neural networks. *Proceedings of the International Conference of ACS'97, Technical University of Szczecin, Poland*, 375-384, 1997.
- [Steven 93] Barber, S. M., Delgado-Frias, J. G., Vassiliadis, Stamatis, Pechanek, Gerald, G. Spin-L: Sequential pipelined neuroemulator with learning capabilities. *Proceedings of 1993 International joint Conference on Neural Networks*, pub. IEEE, New Jersey, USA, 2, 1927-1930, 1993.
- [Sukhaswami 92] Sukhaswami, M. B., Seetharamulu, P., Pujari, A. K. Recognition of Telugu characters using associated memory. *Proceedings of the International Workshop on Pattern Recognition Image Understanding and Artificial Neural Networks (Shanghai, China)*, 32-38, 1992.
- [Sukhaswami 93] Sukhaswami, M. B. Investigations on some applications of artificial neural networks. *Ph.D. Thesis, Dept. of CIS, University of Hyderabad, Hyderabad, India*, 1993.

- [Sukhaswami 95] Sukhaswami, M. B., Seetharamulu, P., Pujari, A. K. Recognition of Telugu characters using neural networks. *International Journal of Neural Systems* 6,3,317-357, 1995.
- [Thimm 95] Thimm, G., Fiesler, E. Evaluating pruning methods. *1995 International Symposium on Artificial Neural Networks*, National Chia-Tung University, Hsinchu, Taiwan, ROC, 20-25, 1995.
- [Tsichritzis 83] Tsichritzis, D., Christodoulakis, S. Message files, *ACM Transactions on Office Information Systems*, 1,1, 88-98, 1983.
- [Tsutsumi 91] Tsutsumi, T. Word-sense disambiguation by examples. *Proceedings of International Conference on Issues in Computational Linguistics*, Malaysia, 1991.
- [Verleysen 89] Verleysen, M., Paul, G., Jespers, A. An analog VLSI implementation of Hopfield neural network. *IEEE Micro*, 46-55, 1989.
- [Veronis 90] Veronis, J., Ide, N. Word sense disambiguation with very large neural networks extracted from machine readable dictionaries. *Proceedings of the 13th International Conference on Computational Linguistics*, COLING'90, Helsinki, Finland, 2, 389-394, 1990.
- [Voorhees 93] Voorhees, E. M. Using WordNet to disambiguate word senses for text retrieval. *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Pittsburgh, PA, 171-180, 1993.
- [Voorhees 98] Voorhees, E. M., Towell, G. Disambiguating highly ambiguous words. *Computational Linguistics*, 24, 125-145, 1998.
- [Waltz 85] Waltz, D. L., Pollack, J. B. Massively parallel parsing: A strongly interactive model of natural language interpretation. *Cognitive Science*, 9, 51-74, 1985.
- [Wilks 90] Wilks, Y. A., Dan, F., Cheng-Ming, G., James E. MacDonald, Tony, P., Slaton, B. A. Providing machine tractable dictionary tools. *James Pustejovsky, editor, Semantics and the Lexicon*. MIT Press, Cambridge, MA, 1990.
- [Yarowsky 92] Yarowsky, D. Word-sense disambiguation using statistical models of Roger's categories trained on large corpora. *Proceedings of the Fourteenth International Conference on Computational Linguistics*, Nantes, France, 454-460, 1992.
- [Yarowsky 93] Yarowsky, D. One sense per collocation. *Proceedings of the ARPA Human Language Technology Workshop*, Princeton, New Jersey, 266-271, 1993.

- [Yarowsky 94] Yarowsky, D. Decision lists for lexical ambiguity resolution: Application to accent restoration in Spanish and French. *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, Las Cruces, New Mexico, 88-95, 1994.
- [Yarowsky 95] Yarowsky, D. Unsupervised word sense disambiguation rivaling supervised methods. *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, Cambridge, Massachusetts, 189-196, 1995.
- [Zurada 97] Zurada, J. M. Introduction to Artificial Neural Systems. *Jaico Publishing House*, Mumbai, India, 1997.