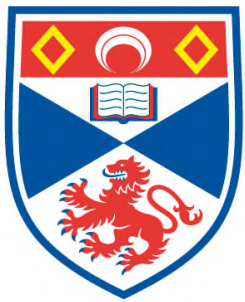
ADAPTIVE PATTERN RECOGNITION IN A REAL-WORLD ENVIRONMENT

Dimitrios Bairaktaris

A Thesis Submitted for the Degree of PhD
at the
University of St Andrews



1991

ADAPTIVE PATTERN RECOGNITION IN A REAL-WORLD ENVIRONMENT

A thesis submitted to the University

of St.Andrews for the degree of

Doctor of Philosophy

by

Dimitrios Bairaktaris

Department of Mathematical and
Computational Sciences

University of St.Andrews

November 1990

Date of admission as a Research Student: October 1987
Date of admission for the Doctor of Philosophy degree: 27th September 1990

I, Dimitrios Bairaktaris, hereby certify that this thesis has been composed by myself, that it is a record of my own work, and that it has not been accepted in partial or total fulfilment of any other degree or professional qualification.

In submitting this thesis to the University of St.Andrews I understand that I am giving permission for it to be made available for use in accordance with the regulations of the University Library for the time being in force, subject to any copyright vested in the work not being affected thereby. I also understand that the title and abstract will be published, and that a copy of the work may be made available to any bona fide library or research worker.

Signed : Date : 9/11/90

I hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulation appropriate to the Degree of PhD.

Signature of Supervisor: Date: 13/11/90

Acknowledgements

I would like to express my deepest gratitude to my supervisor Dr. M.J.Livesey for his invaluable assistance towards the completion of this thesis.

Αφιερωμένο στον πατέρα μου και την μητέρα μου

Χωρίς την βοήθεια τών οποίων δεν θα είχε
πραγματοποιηθεί ποτέ.

(Dedicated to my father and mother)

ABSTRACT

This thesis introduces and explores the notion of a *real-world* environment with respect to adaptive pattern recognition and neural network systems. It then examines the individual properties of a real-world environment and proposes *Continuous Adaptation*, *Persistence* of information and *Context-sensitive* recognition to be the major design criteria a neural network system in a real-world environment should satisfy.

Based on these criteria, it then assesses the performance of Hopfield networks and Associative Memory systems and identifies their operational limitations. This leads to the introduction of *Randomized Internal Representations*, a novel class of neural network systems which although stores information in a fully distributed way yet is capable of encoding and utilizing context.

It then assesses the performance of Competitive Learning and Adaptive Resonance Theory systems and again having identified their operational weakness, it describes the *Dynamic Adaptation Scheme* which satisfies all three design criteria for a real-world environment.

CONTENTS

CHAPTER 1 1
INTRODUCTION 1
1.1 General 1
1.2 Connectionist Systems 1
1.3 Pattern Recognition 2
1.4 Real-World Environment 6
1.4.1 Introduction 6
1.4.2 Continuous Adaptation 7
1.4.3 Integrity of Information 10
1.5 Context-Sensitive Pattern Recognition 10
1.7 Summary 12
CHAPTER 2. 13
CONNECTIONIST SYSTEMS 13
2.1 Introduction 13
2.2 History - The early days 13
2.2.1 McCulloch & Pitts 13
2.2.2 Introducing Learning Rules 15
2.2.3 Embedding Fields Theory 18
2.2.4 Introducing Thresholds and Lateral Inhibition 20
2.2.5 A Simple System for Pattern Classification 22
2.3 History - Recent Developments 23
2.3.1 The Correlograph 24
2.3.2 On-Center Off-Surround Systems 25
2.3.3 The Cognitron 26
2.4 Architectural Issues 29
2.4.1 Pattern of Connectivity 29
2.4.2 Knowledge Representation 31
2.5 Self-Organization 31
2.6 Summary 34

CHAPTER 3 35
AUTO-ASSOCIATIVE MEMORY SYSTEMS 35
3.1 Introduction 35
3.2 The Data Set 35
3.3 Hopfield Networks 36
3.3.1 Introduction 36
3.3.2 Architecture 37
3.3.3 Adaptation Scheme 37
3.3.4 Evaluation 39
3.3.5 The Capacity Issue 44
3.4 Matching the pattern to the size of a Hopfield network 45
3.4.1 The Transformation Method 45
3.4.2 Simulation Results 48
3.5 Associative Memory Systems 49
3.5.1 Introduction 49
3.5.2 Training and Evaluation of the Network 49
3.5.3 Bidirectional Associative Memory (BAM) Systems 53
3.5.4 Simulation Results 54
3.6 Auto-Associative Memories in a Real-World Environment 56
3.6.1 Introduction 56
3.6.2 Self-Organization 56
3.6.3 Continuous Adaptation 57
3.7 Summary 59

CHAPTER 4 60
RANDOMIZED INTERNAL REPRESENTATION (RIR) 60
4.1 Introduction 60
4.2 A combination of BAM and Hopfield Networks 60
4.2.1 Architecture 60
4.2.3 Adaptation Scheme 62
4.2.4 Evaluation 63
4.2.5 Simulation Results 65
4.3 A Shared Content-Addressable Memory (SCAM) 67
4.3.1 Pattern Classification Using Contextual Information 67
4.3.2 Architecture 68
4.3.3 Dynamic Binding and Synchronized Recall 69
4.4 A Three-Layer Bidirectional Auto-Associative Memory 73
4.4.1 Introduction 73
4.4.2 BAM vs. Hopfield Cluster 74
4.4.4 Simulation Results 75
4.4.5 Extensions 77
4.5 Summary 77

CHAPTER 5 78
COMPETITIVE LEARNING SYSTEMS 78
5.1 Introduction 78
5.2 Competitive Learning Models 78
5.2.1 Introduction 78
5.2.2 Architecture 79
5.2.3 Evaluation 81
5.2.4 Winner-take-all Clusters 83
5.2.5 Adaptation 84
5.2.6 Stability in Competitive Learning Systems 85
5.3 Adaptive Resonance Theory (ART) 90
5.3.1 Introduction 90
5.3.2 Architecture 90
5.3.3 Evaluation 92
5.3.4 Adaptation 94
5.3.5 Limitations of ART Systems 94
5.4 Summary 96

CHAPTER 6 - DYNAMIC ADAPTATION SCHEME (DAS) 98
6.1 Introduction 98
6.2 Adaptation Rule 98
6.3 Subsets, Supersets and the All-on pattern 101
6.4 Modifiable Thresholds 102
6.5.Dynamic Adaptation Scheme (DAS) 103
6.5.1 The Scheme 103
6.5.2 Evaluation 104
6.6 Simulation Results 109
6.7 Context Sensitive Classification 113
6.8 Summary 120

CHAPTER 7 - CONCLUSION 121
7.1 Introduction 121
7.2 DAS vs. RIR 121

APPENDIX A 127
APPENDIX B 131
REFERENCES 134

CHAPTER 1

INTRODUCTION

1.1 General

In recent years there has been a revival of interest in a particular class of adaptive systems described under the general term *connectionist systems* or *neural networks*. Although, connectionist systems were firstly introduced during the 40's (see Chapter 2), it was recent developments in computer hardware and some very interesting research work performed during the 70's, that brought connectionist systems under the spotlight. Motivated by their fascinating property to *adapt internally and encode information which can be recalled later from only part of the initial information*, two particular classes of connectionist systems were investigated with respect to *pattern recognition* in a *real-world* environment. This thesis reports the results of this investigation and describes the development of two new classes of connectionist systems engineered to meet the requirements of a real-world environment.

In contrast to the fact that the early connectionist systems were aimed at modelling parts of the human brain rather than being used as computing devices, a purely engineering approach is taken here. In this respect the new systems presented in this thesis claim no biological or psychological plausibility whatsoever.

1.2 Connectionist Systems

A connectionist system consists of a number of units and connections in a network structure. An adaptation process enables the system to obtain information about the contents of an environment. For the particular case of

pattern recognition investigated here, the environment is described in terms of *patterns*. The system is expected to obtain information about the *categories* of patterns in the environment.

A brief account of some early pattern recognition connectionist systems is provided in Chapter 2, and emphasis is given in those issues which are related to the design constraints of a real-world environment. Four contemporary connectionist systems have been under close investigation:

1. *Auto-associative Memories* (Chapter 3).
2. *Associative Memories* (Chapter 3)
3. *Competitive Learning systems* (CL) (Chapter 5)
4. *Adaptive Resonance Theory* (ART) (Chapter 5)

Auto-associative memories are examined in conjunction with associative memory systems and together they are used to form an new class of auto-associative memory system described under the general term *Randomized Internal Representations* (RIR) (Chapter 4). CL and ART systems had a direct influence on the development of *Dynamic Adaptation Scheme* (DAS) presented in Chapter 6. The above systems (1-4) were chosen amongst several other systems because a primary investigation showed that they satisfied the following two criteria which are a necessary for a real-world system:

- *Incremental adaptation rule* (Section 1.4.2)
- *Self-organization* (Section 2.6)

1.3 Pattern Recognition

In recent years pattern recognition has been one of the favourite areas of application for connectionist systems [002][003][004][005][006][007].

Pattern recognition should not be confused with Visual Pattern Recognition (VPR). The systems investigated here are aimed at addressing a wide area of applications not necessarily restricted to VPR. The more general term *signal processing* is probably more appropriate to describe the type of pattern recognition performed by connectionist systems.

In the standard approach taken for connectionist systems, two separate processing phases can be identified:-

Adaptation. This is the phase where the system "learns" about the environment. A set of patterns, hereafter called the *training data*, is selected from the environment for storage in the system. The distribution of patterns in the training data is taken to be a *faithful* representation of the distribution in the environment. However, because the training data is fixed they can only faithfully represent a stationary environment.

The training patterns are taken to be representatives of the categories of patterns present in the environment, and at least one representative, from every category has to be included in the training data. Hereafter, the category representative pattern will be called the *archetype*. The system modifies its weight values according to the archetypes as defined by an adaptation rule. Once the adaptation process is completed the weight values remain fixed and the system is used for recognition.

Given the above conditions, connectionist systems in their standard form cannot be considered as *truly* adaptive pattern recognition devices. The term *parametric* is probably more appropriate. We argue that only a system which is capable of real-world applications can be considered to be truly adaptive.

Evaluation. Once adaptation is completed, a pattern, hereafter called the *probe*

pattern, is presented to the system for recognition. Note that using a probe pattern is the only way to extract information from the system. Two forms of recognition can be identified:

- *Pattern Recall* where the system responds with the archetype closest to the probe pattern.
- *Pattern Classification* where a unit of the system associated with a particular category of patterns, becomes active as an indication of the category of the probe pattern. This unit will be called the *winner-unit.*

There is a one to one correspondence between the two types of recognition mentioned above and the ways in which information is represented in a connectionist system:

- *Local* representation, where a pattern is represented by a unit and its connections.
- *Distributed* representation, where every pattern is represented by all the units and the connections in the system.

Local and distributed representations are the two extreme cases. There are other forms of representation [001] which are not consider here for reasons explained in Section 2.5. Associative memory systems support distributed representations while CL, ART and DAS systems support local representations. The pattern representation format is the decisive feature of a connectionist system which defines what type of pattern recognition is performed by the system.

Distributed representation systems are only capable of pattern recall, while local representation systems are capable of pattern classification. A pattern classification system can be trivially extended to a pattern recall device (Exhibit 1.1). This is achieved simply by enabling the winner-unit to reproduce the archetype pattern. In some cases this is feasible at no extra cost (ART and CL

systems) simply by reversing the direction in which the system is evaluated. A pattern recall device however cannot be extended, in any trivial way, to a pattern classification device.

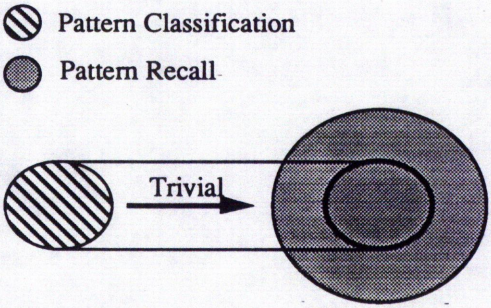


Exhibit 1.1 There is a one to one correspondence between the way information is represented in a connectionist system and the type of pattern recognition it is capable of performing. Note that all pattern classification devices can be trivially extended to pattern recall devices.

Consider a surveillance system which recognizes human faces. The system is responsible for opening a security door when it sees a familiar face. In this case a classification system is needed because *action* is required. In a naive implementation, the door locking mechanism may be hardwired to the winner-unit.

Now consider the case where a human guard is responsible for opening the door. In this case a pattern recall system is sufficient to *assist* the guard in the recognition task by responding with the closest archetype face on a screen. The guard's visual system will use this information to do the final

classification. In this way the guard's visual system converts a pattern recall device into a pattern classification device but in a non-trivial manner.

For either mode of recognition, performance relies on the training set. For the systems presented in Chapters 3 and 5, a fixed, and in some cases constrained, environment guarantees successful performance. The conditions under which a systems will operate in a real-world environment are discussed next.

1.4 Real-World Environment

1.4.1 Introduction

A real-world environment will be one that has the following characteristics:

(a) *Continuous changes*, which are subdivided into two categories:

- *Global*, changes which can be either the appearance of a new category of patterns or the disappearance of an existing category of patterns
- *Local*, changes which occur within an existing category of patterns

(b) *Unexpected events*

Consider the design criteria a connectionist system should meet in order to recognize patterns in such an environment.

Characteristic (a) suggests that the environment is non-stationary. It follows that no finite set of training data can be found. Any training data set available at one stage will not be sufficient to describe the environment over a long period of time. To enable a connectionist system to obtain a *faithful* representation of the environment it must operate under a *continuous adaptation* regime (Section 1.4.2). Under such a regime the system is capable of adapting at every change in the environment.

Characteristic (b) suggests that the environment may provide hazardous or irrelevant input. Such input may force the system to lose information already in store. Because no external assistance is available to protect the system from such losses, it is essential that the system itself guarantees *integrity* of the information already in store.

1.4.2 Continuous Adaptation

A continuous adaptation regime implies that the system no longer separates processing into distinct adaptation and evaluation phases. Instead there is a continuous change of the system's status between the following two modes:

- *Plastic*, when the system adapts to input
- *Stable*, when the system remains unchanged

In the simple case, a system is said to be plastic when the adaptation rule is in force. These two modes should not be confused or interpreted as a substitute for the adaptation-evaluation couple. In the ideal case a system must always be plastic and as shown in the diagram of Exhibit 1.2 plasticity should be made proportional to the degree of change in the environment. In this case the stable state is equivalent to zero plasticity which corresponds to zero changes in the environment.

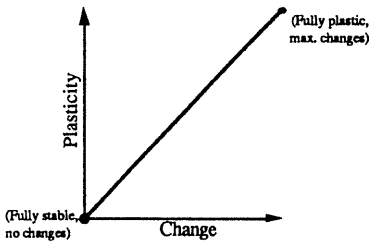


Exhibit 1.2 Plasticity is proportional to the degree of change in the environment.

Under a continuous adaptation regime the system is required to perform pattern recognition whichever mode it is in. Next we present the three main problems facing the designer of a continuously adapting system.

Stability-plasticity control. A system under a continuous adaptation regime is expected to be plastic every time changes in the environment are reported and to be stable when no changes are reported. The stability-plasticity control mechanism should enable the system to:

(a) Sense and identify changes in the environment

(b) Appropriately instruct itself to store these changes

Carpenter and Grossberg [008] where the first to introduce the stability-plasticity problem. However, their discussion of the problem was restricted to global changes, and their solution is of limited use (Section 5.3).

Incremental adaptation rule. This is required for the system to be capable of encoding every instance of change in the environment in real-time. An incremental adaptation rule allows the system to adapt only on the basis of information conveyed by the current probe pattern.

There are adaptation rules, such as back-propagation [001], which are strictly non-incremental since they require the complete set of data in order to adapt successfully. There are also incremental rules, such as Weber's law [009], which do not guarantee integrity of information. This problem is discussed briefly in Section 1.4.3 and further in Section 5.2.6.

Dynamic Allocation and Optimization of Resources. Continuous adaptation raises another issue with respect to the size of a connectionist system. It is

generally accepted that the information storage capacity of a connectionist system is related to the number of its units. However, the exact relationship depends upon the particular system. For example in competitive learning models [009] one would have to add the number of environmental categories to the size of the patterns in order to estimate the number of the processing elements required by the system. On the other hand, the size of Hopfield [010] networks is simply determined by the number of the categories. What is important in both cases is that determining the size of the system requires numerical knowledge of the environment. From a engineering standpoint an *a priori* decision about the size of the system means that computing resources could be wasted. An oversized network can be constructed to accommodate extremely rich in classes environments. To avoid this, it is necessary to develop a mechanism that allocates computing resources dynamically. There are two levels where this dynamic allocation has to be applied. The conceptual level, at which a resource must be incorporated in the system; the physical level at which the conceptually unbounded system is implemented in hardware. Here we only considered the conceptual level. Dynamic allocation is really another manifestation of the stability-plasticity control problem in the sense that encoding of global changes may require for additional computer resources.

Furthermore, because computing resources are always limited, re-use of hardware is necessary. When some of the environmental categories no longer exist it seems wise to release the hardware they use, for representing new classes. It is necessary to state here that re-use of computing resources can be implemented only for local representations. In distributed representations, for reasons explained in Chapter 3 it is very difficult to implement such an optimization mechanism.

There is an analogy here between the above optimization problem and the problems related to dynamic allocation and garbage collection of computer memory.

1.4.3 Integrity of Information

Depending on the particular type of connectionist system a variety of reasons may threaten integrity of the information in store. Auto-associative memories, for example, function properly only when certain statistical constraints on the input patterns are satisfied. If these constraints are violated, then information is lost and quite often there is no way to recover it. CL systems can also be very sensitive to a particular kind of input. For example, a change in the sequence of patterns may result in permanent loss of information (see Section 5.2.6).

A system intended to perform successfully in a real-world environment should be able to handle any form of input in a consistent way. For all the systems investigated here the cause of integrity problems is attributed to their adaptation rule, which often leads the systems into unstable behaviour. Therefore, it is of major importance to identify the exact reasons for this unstable behaviour and make an attempt to find an adaptation scheme which guarantees a stable behaviour. From an engineering point of view a stable behaviour is really the complement of the continuous adaptation regime.

1.5 Context-Sensitive Pattern Recognition

Real-world environments are quite often rich in contextual information which is extensively used in every day pattern recognition tasks. In general, it is much easier for us to recognize a pattern within a context than it is to recognize it by itself. Quite often a visual stimulus is *bound* to the corresponding

auditory stimulus, thus improving recognition performance. For example, a pelican crossing uses both visual (green, red light) and auditory patterns (changing tone of a bell) to indicate about the safety of the crossing. Another example of contextual information can be found in documents. A document may be viewed as a "flat" collection of letters but one easily discovers a *hierarchical organization* which can be used as a source of contextual information. Letters are part of words, and it has been demonstrated [051] that humans are better at recognizing letters within the context of a word than just the letters themselves. Furthermore, words are part of sentences, sentences are part of paragraphs and so on.

A system capable of performing in a real-world environment must be able to take advantage of contextual information. From an engineering point of view, context sensitive recognition is even more important. There are cases where recognition may fail for various reasons (eg. exceedingly large noise levels, ambiguous classification etc.). In such cases, performance may be improved by the use of context (Sections 4.3 and 6.7).

Three basic problems are related to the utilization of contextual information in a connectionist system:

(a) *Identification* of contextual information

(b) *Encoding* of contextual information

(a) Under the standard adaptation-evaluation regime, identification of contextual information could be part of a pre-processing stage where patterns are selected for the training data. In a real-world environment such a pre-processing stage cannot be implemented in real-time. In this case an interactive system may allow the user to identify context related

patterns and inform the system (Section 4.3.1). This solution requires explicit synchronization of the system which may affect its response time. Alternatively, an implicit indication of contextual relations, such as simultaneous arrival of patterns, can be utilized. In this case synchronization is handled by the environment.

(b) The problem here lies in the fact that adaptation rules were made to store patterns and not semantic relations. It is therefore essential that the adaptation rule, and the system itself, should be extended in some way to accommodate storage of contextual information. There is no unique solution to the problem of encoding contextual information. However there is one solution which is applicable to all systems, namely the use of a *common* representation for contextually related patterns.

We require that any extensions made to utilize context, should not by any means affect the recognition properties of the basic system.

1.7 Summary

In this chapter an outline of the following chapters was followed by an introduction to connectionist systems with respect to pattern recognition. The definition of a real-world environment was followed by a discussion on the design criteria a connectionist system should meet in order to operate in such an environment.

CHAPTER 2.
CONNECTIONIST SYSTEMS

2.1 Introduction

The term *connectionist* originates from the key feature of these systems, namely the connections that enable elementary computing agents (units) to communicate. The term neural networks also used to describe these systems originates in their architectural and functional similarities to neuronal structures in the brain. In the following two sections we provide a brief historical review of the early days of neural networks and a more detailed description of recent neural models focusing attention on the issues relevant to this thesis. This is followed by a discussion of specific architectural issues in connectionist systems together with an extended discussion on the self-organization of these systems.

2.2 History - The early days

2.2.1 McCulloch & Pitts

In 1908 Ramon y Cajal first identified the nerve cells or neurons in the human brain (Exhibit 2.1). A neuron consists of a cell body which receives electrical signals from several dendrites, originating from other neurons, by means of, varying strength, synapses.

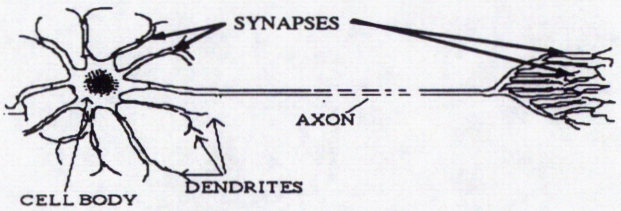


Exhibit 2.1 A nerve cell

The cell body performs an operation over the incoming signals and emits an output signal on its axon which in turn delivers the signal to other neurons. This observation raised the question of whether a collection of simple neurons could solve any interesting problems.

In 1943 McCulloch and Pitts [011] observed that the all-or-none character of neural activity should allow the treatment of neuronal structures by means of propositional logic. They suggested that the existence of a unit (neuron) in the system could be treated as a simple proposition which is true of false depending on the state of the unit. The all-or-none behaviour of a unit derives from the existence of a threshold, the unit being active or inactive depending on whether or not its excitation exceeds a threshold. They proposed a network allowing only synaptic time delays and assume that the activity of an inhibitory synapse will prevent a unit from being active. A cell will fire if and only if the number of excitatory input equals or exceeds the threshold and no inhibitor is active [012]. In Exhibit 2.2 the circles correspond to units connected with links (dendrites) which have a weight value (synaptic strength). Weight values are allowed to be either excitatory (weight = 1) or inhibitory (weight = $-\infty$).

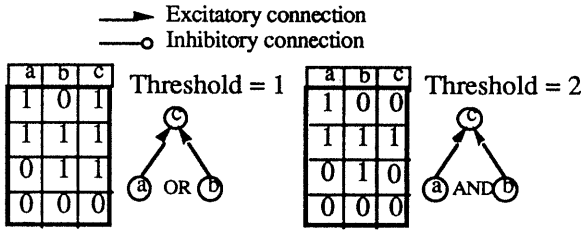


Exhibit 2.2 Three example networks capable of computing logic functions described in corresponding tables.

The proposition units (c) have a threshold value and become active (1) or inactive (0) depending on whether the sum of the input values times the weight values exceeds or equals the threshold. As shown in Exhibit 2.2 by appropriately choosing the structure, weight values and thresholds a system can be constructed to evaluate any arbitrary logic expression. These systems are not of particular interest to us because the associated engineering problem of finding the appropriate structure, weight values and thresholds to solve a given problem is extremely complex. The fundamental question of how the human brain adapts to the environment remained unanswered.

2.2.2 Introducing Learning Rules

In 1949 D.O.Hebb [013] in his book Organization of Behaviour first introduced the idea of collections of neurons that adapt. A system with varying synaptic strengths provides a suitable place for storing information if the appropriate synaptic modification rule (learning rule) can be found. The related to a particular synapse, presynaptic unit is the one which generates a signal and the postsynaptic unit is the one that receives the signal. Using this terminology Hebb proposed the following a simple learning rule:

" If the presynaptic element and the postsynaptic are both active then, the synapse should be strengthened. "

Although his rule had no definitive proof at that time, it contains a fundamental idea that both the systems investigated in Chapters 4 and 6 have in common. Weight modification is decided locally by the states of the two connected units. In addition to Hebb's rule, several other local synaptic modification rules have been suggested. Exhibit 2.3 shows the four most important such rules (an active or inactive unit is noted by + or - respectively). In [014] D.Willshaw and P.Dayan compare these rules in the context of associative memories (See

Section 2.3), using the signal to noise ratio (S/N) as a measure of recall accuracy, and suggest that the Sejnowski - Hopfield (2.3.2) rule is optimal while the Stent - Singer (2.3.3) rule is sub-optimal.

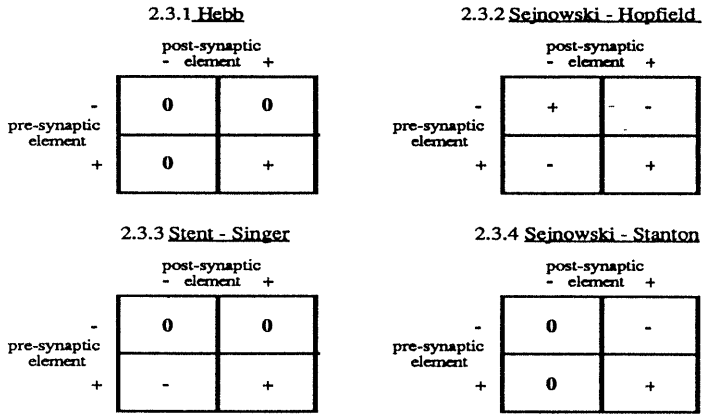


Exhibit 2.3 The four most important synaptic modification rules.

McCulloch & Pitts networks combined with Hebb's approach towards varying strength synapses were the basic ingredients for Rosenblatt's work on the Perceptrons [015][016][017][018]. In contrast to the propositional logic approach of McCulloch & Pitts, the Perceptron is a probabilistic network. Perceptron is the first model that incorporates the learning of responses to specific patterns.

A typical Perceptron consists of a sensory area (input units), an association area (association units) and a response area (response units) as shown in Exhibit 2.4. The system is provided with a pattern at the input units and the corresponding pattern at the response units, and learns to respond by means of activating the appropriate response unit when presented with an input pattern.

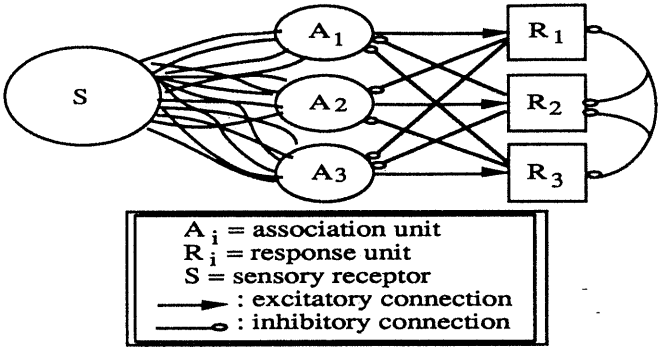


Exhibit 2.4 A typical Perceptron structure

An association unit sums all the inputs from the sensory area and if this sum exceeds a designated threshold value, emits a signal to the response units. The connections between the association units and the response units are bidirectional in order to enable feedback from the response units during learning. The systems develops in such a way that a specific association unit becomes active for a particular input pattern and in turn activates the appropriate response unit. In [016] Rosenblatt makes the following comment to the significance of the threshold of an association unit :

".... *It is possible to vary the effective threshold of an A-unit [association unit] by adding an excitatory or inhibitory component to its input signal ..... Such a condition would be hard to utilize effectively for the control of activity.*"

Rosenblatt identifies the potential usefulness of varying thresholds and suggests an external servomechanism for adjusting the threshold. This mechanism appears to be a graft on the system which do not adheres to the Perceptron formalism. In chapter 6, where we describe DAS, we provide a threshold control mechanism that is incorporated in the architecture of the system and we discuss its plausibility and effectiveness. A Perceptron

requires the existence of an input-response pair during the learning phase. The absence of the corresponding response in real-world applications, makes Perceptrons inadequate for pattern classifications purposes. Rosenblatt identified this problem and in [017] he proposed the C' class perceptron a system capable of spontaneous organization by merely observing and learning from the organization of the surrounding world.

2.2.3 Embedding Fields Theory

In 1966 S.Grossberg published a paper [019] on a class of nonlinear difference-differential equations which was appropriately interpreted as a learning theory. Later, based on these equations *embedding fields* systems were developed. Embedding fields systems were directly connected to pattern classification devices [020] and they are the predecessors of on-center off-surround systems (Section 2.3.2).

Embedding fields systems are described by the following equations:

$$\frac{dx_i}{dt} = -\alpha x_i(t) + \beta \sum_{k=1}^{n} x_k(t-\tau) y_{ki}(t) + I_i(t) \qquad (1)$$

$$y_{jk}(t) = p_{jk} z_{jk}(t) \left[\sum_{m=1}^{n} p_{jm} z_{jm}(t) \right]^{-1} \qquad (2)$$

$$\frac{dz_{jk}}{dt} = \left[-u z_{jk}(t) + \beta x_j(t-\tau) x_k(t) \right] \theta(p_{jk}) \qquad (3)$$

where: n is a positive integer ; α, u, β are positive constants ; $t \geq 0$;
$P = \| p_{ij} \|$ is a n x n matrix with $p_{ij} \geq 0$ and $\sum_{k=1}^{n} p_{ik} = 0$ or 1;

and $\theta(p) = \begin{cases} 1 \text{ if } p > 0 \\ 0 \text{ if } p \leq 0 \end{cases}$.

The values involved in equations (1) (2) and (3) have the following interpretation:

- $x_i(t)$ is the activation of a unit i at time t.

• The term $-\alpha x_i(t)$ represents a decay mechanism in the activation equation of unit i.

• $\sum_{k=1}^{n} x_k (t - \tau) y_{ki} (t)$ is the sum of the products of the activations x_k at time τ ago multiplied by the corresponding connection weight value $y_{ki}(t)$ between units i and k. $I_i(t)$ stands for an external input to unit i.

• The weight values $y_{ki}(t)$ are computed by equation (2) with p_{jk} and p_{jm} as defined above.

For $p_{1k} = \frac{1}{n-1}$ (k = 1..n) and all the other p_{ik} set to 0, a system called an *outstar* is obtained. This configuration of the weights corresponds to a system with n units, where one unit, the input unit, is connected to all the other units, hereafter called the output units. For this particular case, equation (2) has the effect of normalizing the weight values. The term $-uz_{jk}(t)$ is a decay on the weight value and the term $\beta x_j(t - \tau)\ x_k(t)$ is the product of the activation values of the units i and k at times $t-\tau$ and t respectively. This is in line to Hebb's idea that for modifying the weight values only the states of the two units at either end of a connection should be taken into consideration. It is proved in [021][019] that there exists a limit for $x_i(t)$ and $y_{jk}(t)$ for t --> ∞, thus ensuring that the system converges to a stable state. For the special case where $\alpha > \beta$ then $x_i(t) = 0$ for t -> ∞ [021]. The decay parameter of (1) forces the network to a resting (zero) state.

Embedding fields systems were originally aimed to function as input/output matching devices. They are very similar to the Perceptron in that, a output pattern, paired to an input pattern, is required to drive the system during weight modification. The presence of the input and output patterns is translated into activation at the input and output units respectively. The system modifies its weights and afterwards, probed with an input pattern, it is capable of responding with the corresponding output pattern. An extension of the

proposed system to function as a pattern classification device is presented in Section 2.2.5.

2.2.4 Introducing Thresholds and Lateral Inhibition

At the early stages of the development of embedding fields systems the need for a threshold mechanism was understood [022]. A threshold value is attached to every unit in the system, and the unit emits an output signal only if its activation value exceeds the threshold value. The addition of a threshold value Δ modifies equation (2) as follows [020]:

$$\frac{dx_i}{dt} = -\alpha x_i(t) + \beta \sum_{k=1}^{n} [x_k(t-\tau) - \Delta_{ki}] y_{ki}(t) + I_i(t) \quad (4)$$

Note that in (4) there is a different threshold value for every connection of unit i. This is different to the generally accepted idea of having only one threshold value for every unit. Increasing the number of threshold values, may increase the functionality of the system but also increases its complexity. It has been suggested by Wickelgren [023.] the use of multiple thresholds values but as suggested by Grossberg [020] the overall efficiency and miniturization of control in the system is of equivalent importance. In a recent work by Adamopoulos and Anninos [024] on the mathematical modelling of neurons, with respect to epileptic phenomena in the brain, is suggested that there are several threshold values that correspond to different activation levels of a neuron. All the systems presented in the following chapters make use of one threshold value which seems to be sufficient for pattern recognition purposes. Introducing a threshold mechanism imposes a synchronization problem with respect to the adaptation of the system. As mentioned earlier, the system requires that both an input and output pattern are present simultaneously for the system to modify its weights. Given that a threshold mechanism allows a unit to fire only when the above mentioned condition is satisfied, it necessary

to ensure that the output units and the input units are simultaneously active. For this, a control mechanism is suggested in [020] by means of diffuse arousal inputs (DAI's). A simple system with one input unit, three output units and the corresponding DAI's is shown in Exhibit 2.5.

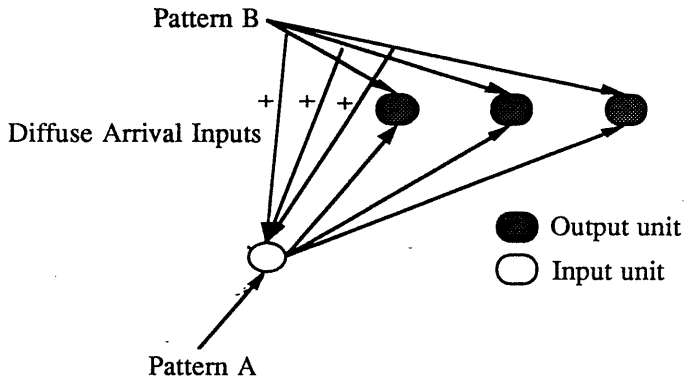


Exhibit 2.5 A simple (outstar) system with Diffuse Arrival Inputs (DAI's) DAI's secure that the arrival of pattern B combined with the presence of pattern A, will force the input units to fire.

A pattern A is made available to the input unit i and a pattern B is supplied to the output units. The system is expected to modify its weights in order to respond with pattern B when probed with pattern A. Since the input unit has a threshold mechanism it does not fire all the time and if pattern A arrives at a different time than pattern B then no learning can be achieved. The introduction of DAI's enables the input unit to emit an output signal when both patterns A and B are present in the system. DAI connections apart from their functional assistance to the particular system, can be viewed as a more general control mechanism. For example, Adaptive Resonance Theory (ART)

(Chapter 5) uses a similar control mechanism. Also it can be considered as a feedback mechanism where information from the output units enhances the quality of the input pattern.

2.2.5 A Simple System for Pattern Classification

In S.Grossberg [020] a collection of simple outstar networks is suggested as a simple system for recognizing different input patterns (see exhibit 2.6). All inputs are considered to be positive and continuous. The system is to discriminate amongst different input patterns by means of allowing precisely one unit to be active at any time. In this way the idea of local knowledge representation is introduced. Every input pattern is represented by only one unit in the system. Whether the system performs the appropriate classification is estimated by observing which of the units emits an output signal on the arrival of a specific input pattern. Clearly, if the system guarantees that a one to one correspondence exists between the input pattern and a specific unit it is suitable as a general purpose classification device. Here we are concerned with the performance and reliability of the system as described in [020]. The equation that describes the activation of a unit i is as follows:

$$\frac{dx_i}{dt} = -\alpha_i x_i(t) + \beta_{ii} I_i(t - \tau_i) - \sum_{k=1\ k\neq i}^{n} I_k (t - \tau_k)\, \beta_{ki} \qquad (5)$$

where $-\alpha_i x_i(t)$ is the familiar decay term, $\sum_{k=1\ k\neq i}^{n} I_k (t - \tau_k)$ is the sum of all the inputs other than I_i and β_{ji} the corresponding weight values. The signs in (5) indicate that only input I_i can have a positive effect on the activation of unit i while all other inputs inhibit unit i. For the simple case where input I_i is positive and all others are zero then only unit i will be active while all other units are inhibited. Therefore, for the network in Exhibit 2.6 input patterns I_1=[+,0,0], I_2=[0,+,0] and I_3=[0,0,+] will activate only the units 1,2 and 3

respectively.

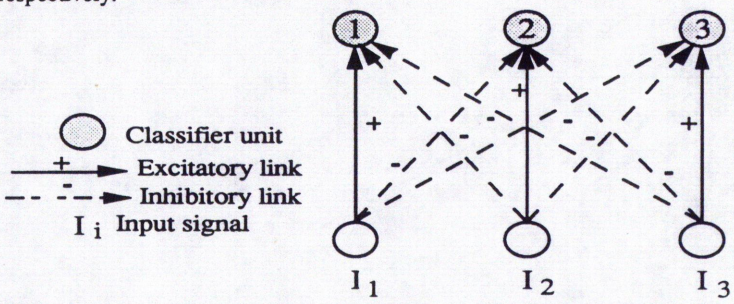


Exhibit 2.6 A simple pattern classification network.

Clearly, for all other possible input patterns, like [+,0,+], the activation of only one unit depends on the actual input and the weight values. For example if we assume all the weights to have the same absolute value with the appropriate signs, the input pattern [+r,0,+r] will result in no unit being active. In a similar way an arrangement of the weight values could lead to two units be simultaneously active (see Grossberg [020]). Nonetheless, for the appropriate weights, input values and architecture the system can classify input patterns correctly. Such a system could only become efficient if all the parameters of the system are decided automatically and ideally, by the system itself.

2.3 History - Recent Developments

In this section we briefly review some connectionist systems developed from 1970 and onwards. Since most of these systems are investigated in detail in Chapters 3 and 5, here we attempt to give a chronological account of the key developments in the field.

2.3.1 The Correlograph

In 1971 D.Willshaw [025] suggests another system with a neuronal-like structure, the correlograph. Using the principles of holography it is shown that a correlograph can act as an associative memory. This is a memory device where a binary output pattern B is associated, by means of connection strengths with a given input a binary pattern A. To recall pattern B we have only to probe the system with pattern A. Although this system is intended as an associative memory it can be easily extended to a content-addressable memory if we assume B = A. We will call such a system an auto-associative memory. This is not a trivial exercise because as Willshaw comments:

" The associative net is able to deal with incomplete and inaccurate information. When a distorted version of one of its stored patterns is presented to it, under certain conditions the correct pattern can be retrieved accurately. "

We will call this property of associative memories noise-tolerance. It is this property and the possibility of the system adapting on the basis of its input patterns that suggests the use of these systems as pattern classification devices. In Chapter 3 we investigate two slightly different systems with the same properties as the correlograph and demonstrate their performance limitations with respect to the real-world constraints outlined in Chapter 1. Another interesting aspect of auto-associative systems is their way of representing information. We mentioned earlier that Grossberg's pattern discriminators allocate one unit per pattern, thus storing information locally. Auto-associative systems provide a paradigm for distributed information storage because information resides across all the units of the network. A discussion of the

representation issue is provided in Section 2.5, where we investigate the possible implications of both approaches, local and distributed, for pattern classification devices.

2.3.2 On-Center Off-Surround Systems

In [026] a type of network called *on-center off-surround* is proposed as a pattern classification system. This is the predecessor of competitive learning type systems. Essentially, it is a continuation of embedding fields theory only this time the adaptation process is made specific and different forms of the system's behaviour are discussed in detail. The concepts of *long term* (LTM) and *short term* (STM) memory are directly associated with the architecture of the system. A typical on-center off surround network is shown in Exhibit 2.7. The synaptic strengths (weights) on the connections from layer V_1 to layer V_2 correspond to LTM while the activation of specific units in layer V_2 corresponds to STM.

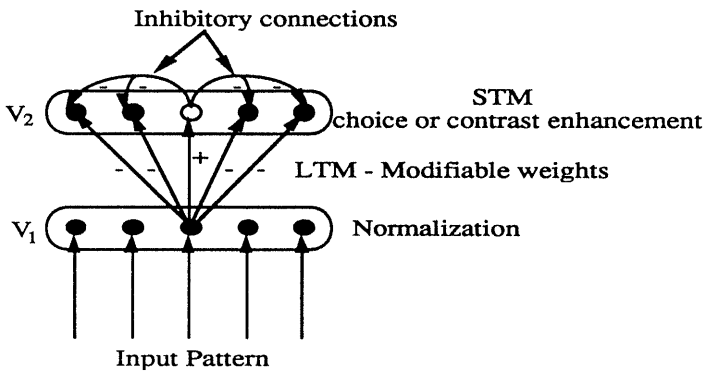


Exhibit 2.7 An example of a non-recurrent on-center off-surround network

In on-center off-surround systems V_1 performs the task of normalizing the input patterns. This means that for a given pattern I_1, the relative input

intensity $\Theta_1 = \frac{I_1}{\sum_{i=1}^{n} I_i}$ rather than the absolute intensity I_1 is the value passed to V_2.

A similar normalization process is required for auto-associative memory systems (see Chapter 3), where make use of the mechanism described here. A serious limitation of on-center off-surround systems is already apparent. In order to compute Θ_i, complete information about the patterns to be stored is required, constituting a serious violation of the constraints described in Chapter 1 imposed by a real-world environment. Since the patterns available in the environment are not known *a priori* and do not exist permanently, on-center off-surround systems are inadequate to support the kind of pattern classification device we seek. Nevertheless, the behaviour of on-center off-surround systems is quite interesting from the pattern classification point of view in that it has provided intuition about which problems to address in this research. As already mentioned, classification of a pattern is achieved by the activation of a specific unit in layer V_2. Every unit in layer V_2 has a quenching threshold (QT). Activation above this threshold will cause a unit to be active while an activation below QT will suppress the activation of the unit. By means of inhibitory connections between the units, layer V_2 will choose the unit with the maximum initial activity, the winner-unit.

2.3.3 The Cognitron

Concurrently with the development of on-center off-surround systems, K.Fukushima [003] independently proposed a new hypothesis for the weight modification phase of neural-like systems within the self-organization paradigm. In fact his system, the Cognitron, has very similar semantics to on-center off-surround systems with the exception that the weight modification scheme is defined in detail. The synaptic strength (weight) from cell x to cell y

is reinforced if and only if the following two conditions are simultaneously satisfied:

(i) Presynaptic cell x fires.

(ii) None of the postsynaptic cells situated in the vicinity of the cell y fires more strongly than y.

The basic Cognitron system consists of several layers of units and it is suggested [003] that multilayer systems enhance performance. Although this is generally accepted, there is not clear quantitative relation between these two factors. A system that uses several layers of units is suggested in chapter 7 but it has a different semantic interpretation from the Cognitron.

In the systems discussed so far, the elementary particle of a system, the unit, was made to perform a rather uniform role; to sum and threshold the incoming signals. In the Cognitron a different role is suggested for the output (out $_i$) of a unit. This is captured by the equation:

$$\text{out}_i = \phi\,[((1+e)\,(1+h)^{-1}) - 1\,] \quad (9)$$

$$\text{where } e = \sum a(v)\,u(v) \text{ and } h = \sum b(\mu)\,u(\mu)$$

$$\phi\,(x) = x \text{ if } x \geq 0 \text{ and } \phi\,(x) = 0 \text{ if } x < 0$$

Here *e* is the sum of all the excitatory (positive) weights, a(v) multiplied by the activation u(v) of the corresponding units and *h* is the corresponding sum of the inhibitory (negative) weights b(μ). All the values are taken to be non-negative in order to ensure that the output of a particular unit does not increase boundlessly and to guarantee convergence of the output [003]. Although [003] argues about the biological plausibility of the proposed unit-function, its complexity constitutes an important reason for not considering it as a candidate for our final system.

Furthermore, the basic structure (see Exhibit 2.8) of the Cognitron itself is rather complex. It requires that every layer of units has distinguishable

excitatory and inhibitory units as well as fixed synaptic connections.

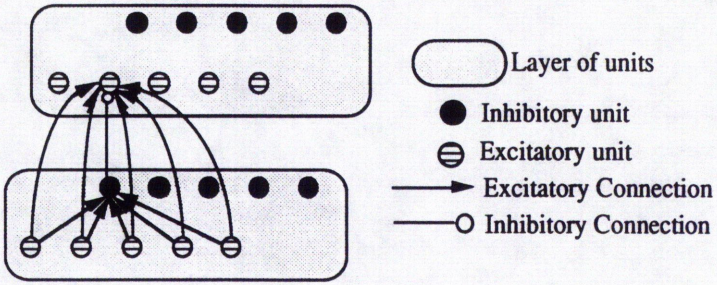


Exhibit 2.8 The basic structure of the Cognitron system.

An excitatory unit is connected via excitatory and inhibitory connections to the previous layers of units. Intra-layer connections have fixed weight values while inter-layer connections have modifiable weight values. Cognitron was designed and developed as a self-organizing system according to the definition of the term at that time. Indeed, an explicit automated process for weight adaptation is suggested and some simulation tests were made to prove its efficiency. The question remains the same as with the other systems discussed so far, how someone goes about arguing about the plausibility of its structure. Already from the early days F.Rosenblatt demonstrated that for the appropriate network structure, learning rule and type of units a range of interesting tasks(including pattern classification) can be performed. The problem of finding the appropriate network configuration has only recently been addressed. In section 2.6 we argue about a generalized paradigm of self-organization in an attempt to tackle this problem. The Cognitron was developed primarily as a pattern classification device and simulation results for the classification of several bitmap representations of letters and numbers in

order to demonstrate its capabilities is given in [003].

2.4 Architectural Issues

All the systems presented so far maintain a *fixed* (static) structure throughout the adaptation and evaluation phase. A different kind of system whose structure is required to be adaptive (dynamic) is proposed in section 2.6.

Engineering a fixed structure connectionist system requires that the following two characteristics are defined prior to adaptation:

- Pattern of connectivity
- Knowledge Representation

2.4.1 Pattern of Connectivity

Pattern of connectivity is defined by the way the units of a system are connected. In the general case one can expect that every unit in the system is connected to every other unit. We call this *full connectivity*. In this case, a system with N units will have N^2 connections and an N by N weight matrix W can be used to represent the pattern of connectivity. Given that the effect of a weight value is always multiplicative, any connectivity pattern may be represented in this way by treating non-existent connections as zero entries in the weight matrix.

A connection between two units can either support signals travelling in one direction, *unidirectional*, or signals travelling in both directions, *bidirectional*. A bidirectional connection has a single weight value which is always used regardless of the signal's direction.

We have already presented systems where the connectivity pattern groups units into layers. The term *layered architecture* is used to describe such systems. The layer of the system where input from the environment is

collected, usually the bottom layer, is called the *input* layer. In a layered architecture system, connections are only allowed between two consecutive layers of units. In most cases, it is required that the units of one layer are fully connected to the units of the previous layer. Some systems require connections between the units of a layer. It is assumed that the same pattern of intra-layer connectivity is repeated for every layer in the system. In the special case where signals are only allowed to travel from the input layer towards the top layer the system is called *feedforward* (Exhibit 2.9).

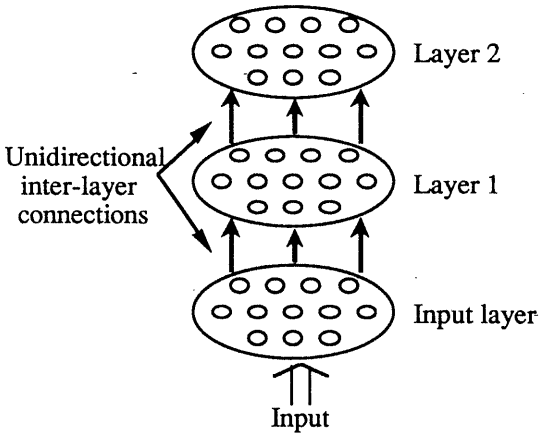


Exhibit 2.9 A feedforward layered connectionist system.

It is possible that units within the same layer are grouped into *clusters*. To our knowledge, competitive learning [009] is the only system that supports this form of organization. In this case the units of the same cluster are connected with fixed inhibitory connections, and the cluster is referred to as a *winner-take-all* cluster. Winner-take-all clusters and their function are presented in detail in Chapter 5.

2.4.2 Knowledge Representation

In a very influential paper, J.Feldman [031] presented several connectionist systems with respect to knowledge representation. His paper discusses the biological and psychological plausibility of connectionist systems and claims that a *black box* approach is unacceptable. Similarly, we claim that exact knowledge of the representation scheme is required when building an adaptive real-world pattern recognition device.

From a strictly engineering point of view, the way knowledge is represented in a connectionist system may not appear significant. A system is considered successful only on the basis of whether it is capable of performing a particular task. However, it is necessary that the potential of different knowledge representation schemes is investigated for dynamic systems, because information (patterns, categories) is stored dynamically and resources (units,connections) have to be allocated also dynamically. It follows that the designer of such a system must precisely identify the relation between information and resources in advance of the system's exposure to the environment. A try and error approach is inadequate for continuously adapting real-time response systems. Furthermore, the decision has to be problem independent because a priori knowledge about the environment is not available.

In addition to the above, to utilize contextual information requires that primary information (eg. patterns) can be identified in the system in real-time. Again this can only be achieved for representation schemes which define precisely where information is stored.

2.5 Self-Organization

Early pattern recognition connectionist systems (Section 2.2) required that,

during adaptation, the training pattern was accompanied by information about its category. This form of adaptation is called *supervised learning*. In a similar way, contemporary connectionist systems (eg. back-propagation [001]) require that the system is told about the category of every training pattern. An early attempt to design a system that is capable of forming categories of patterns without external guidance was suggested by Rosenblatt with his C' class Perceptron. In [027] Von der Malsburg proposed a new adaptation rule and designed a system capable of autonomously discovering categories in the training patterns. He introduced the term *self-organization* to describe the system. Self-organization became a key feature of systems later developed [003][028][009] Self-organization requires that the system uses, during adaptation, no other information about the environment other than the incoming patterns. The current state of affairs in self-organizing connectionist systems can be represented by a fixed structure system that adapts incrementally according to information provided by an environment (Exhibit 2.10)

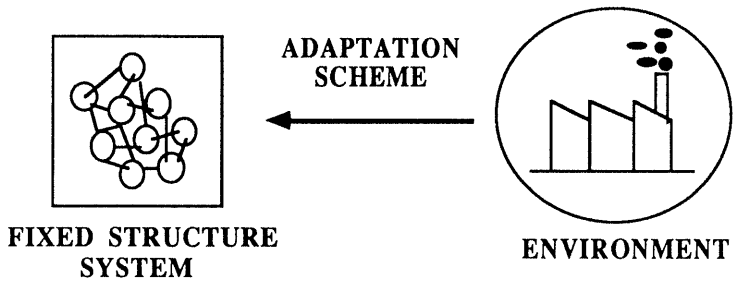


Exhibit 2.10 A fixed structure system adapts incrementally to environmental input by means of an adaptation scheme.

If f is a function representing the adaptation scheme, S is a given structure and I is a set of environmental patterns then the final system is represented by the

weight matrix:

(10) $$W = f(S,I)$$

We propose an augmented concept of self-organization represented by a function g with:

(11) $$(W,S) = g(I)$$

In other words we wish to adapt the structure as well as the weights to the given environment. The implication of (11) is that both the weights and the structure are determined solely by the environment and no other source of information is provided. Furthermore, information is provided from the environment only by means of patterns. Therefore, a system obeying (11) has to extract all the necessary information about how to adapt its structure from the incoming patterns. We only allow for a pre-existing structure that contains no information about the environment.

This augmented concept of self-organization is defined in a similar way in a recent work (SONN) presented in [029]. The SONN system has an adaptation scheme that allows dynamic development of its structure. Although the SONN system cannot be used as a pattern recognition device, under the conditions described in Chapter 1, for three reasons:

- First because it supports a supervised adaptation scheme, which for the pattern recognition case considered here is against the very notion of self-organization.
- Second because it requires a large amount of pre-existing information about the environment. Although the information required is not problem related it has to be general enough for the system to encode a wide range of environments.
- Third because its adaptation scheme is not incremental and therefore a continuous adaptation regime cannot be implemented.

In [030] Feldman addressed several problems associated with adaptive structure connectionist systems. He suggested a connectionist system with dynamic connections, but fixed number of units, which are part of a built-in switching network. In the same paper he also suggested random interconnection networks which are not considered here.

Self-organization as defined in (11) is a pre-condition for a system that satisfies the constraints presented in Chapter 1. Requiring a dynamic structure that evolves according to the environment, ensures that resources will be made available to secure integrity of information. Furthermore, an adaptive structure matches to the specification of a continuous adaptation regime in that changes in the environment will also be reflected into the structure of the system. Therefore, in the case where information has to be extracted from the system in an ad-hoc manner examination of the system's structure can be sufficient. For example in the case of a local representation system, if for some reason the number of categories in the system has to be found, it is sufficient to count the number units.

2.6 Summary

In this chapter we have presented several connectionist systems in an attempt to outline the historical developments of the field. We have only presented systems related to the work herein, rather than a complete catalogue of the field. Having identified the particular features related to pattern classification systems, some architectural issues were raised. This was followed by a brief discussion on the possible ways information is stored in connectionist systems with respect to the design of a fully adaptive system. Finally, an augmented paradigm for self-organization was proposed. It was suggested that this augmented concept of self organization is necessary to secure integrity of information within the continuous adaptation concept.

CHAPTER 3
AUTO-ASSOCIATIVE MEMORY SYSTEMS

3.1 Introduction

This chapter provides a detailed account of auto-associative memory systems with respect to pattern recall. Two basic systems are considered here, namely Hopfield networks [034] and the extended associative memory network of Hinton and Anderson [039]. A detailed account of these systems is provided with respect to their adaptation rules, structure, capacity and performance. The use of associative memory in the form of a Bidirectional Associative Memory (BAM) is considered. Finally, both systems are examined against the set of constraints described in Chapter 1, leading to Chapter 4 where a novel class of auto-associative memories is suggested.

3.2 The Data Set

As well as analysing these systems, we present simulation results for all the important cases. Because of the differences between the systems examined in the following chapters, it was necessary to choose a common set of patterns for use in our simulations in order to achieve comparative results. Moreover it was important to impose no prior constraints (eg. orthogonality) on the choice of patterns. Why this is the case becomes evident from the analysis that follows.

It was mainly the fact that some systems related to this work [008][003] were tested on the bitmap representations of the 26 capital letters of the English alphabet (see Exhibit 3.1), that lead us to use them as our simulation data.

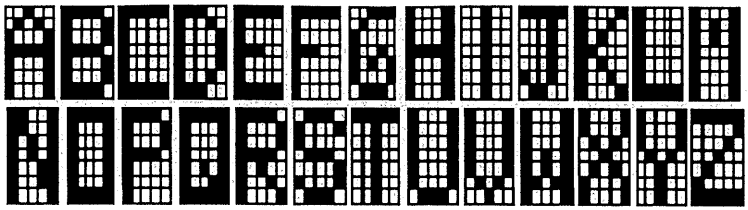


Exhibit 3.1 Bitmap representations of the patterns used for simulation purposes.

Every letter is represented as a 7 by 5 bitmap and the corresponding binary input vectors were derived from these bitmaps by assigning 1 to every black pixel and -1 to every white pixel.

3.3 Hopfield Networks

3.3.1 Introduction

The type of system Hopfield described in his original paper [034] is derived from the spin-glass model theory presented in [035]. There has been substantial research with respect to the physical properties of spin-glass models which, although interesting, lies outside the scope of the work presented here. Hopfield networks are classified as content-addressable memories. That is, a memory device where a stored pattern is recalled by probing the system with an input pattern that represents a noisy version of the stored pattern. Hopfield networks are a classic example of distributed knowledge representation. Every pattern is stored as a combination of weights over the whole network while many patterns are stored in the same physical area. Provided that an environment is described completely, by means of archetypes, the system can be used as a pattern recall device. Archetypes are represented by vectors and their respective distance is measured in Hamming

units. As explained in the following sections, the distance between two original memories is important and affects the overall behaviour of the system.

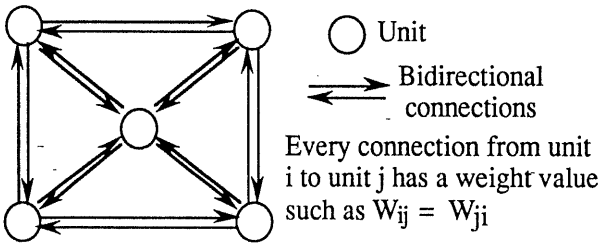


Exhibit 3.2 A Hopfield network: Every unit is connected to all the others via links that have a weight value W_{ij}.

3.3.2 Architecture

A Hopfield network (Exhibit 3.2) consists of:

- A number (N) of units
- A number of bidirectional connections between the units.

Each unit of a Hopfield network has a state value V_i. The state of a unit could be either $V_i = 1$ or $V_i = -1$[1] (binary state). Each unit has a fixed threshold U_i. Most often threshold U_i is set to zero and depending on whether the unit's excitation is positive $V_i = +1$, or $V_i = -1$ if it is negative. The link from unit i to unit j has a weight value W_{ij}. The collection of W_{ij} is treated as an N by N weight matrix W.

3.3.3 Adaptation Scheme

Hopfield networks follow the standard approach taken in connectionist systems, by separating the training phase form the evaluation phase. The

1 Hopfield [006] claims that the possible states of a unit are (1) or (0). Although, when he described the encoding of the patterns the states are transformed to (±1).

archetypes are stored as weight values on the connections of the system. Once the adaptation phase is completed we are able to retrieve the archetype individually, simply by probing the network with a probe. There is more than one way [034][036] to use a Hopfield network as a pattern recall device. In [036] a statistical approach is proposed, where statistics are gathered from the environment and they are directly translated into the weights of the system. This is an non-incremental adaptation scheme that is also computationally complex. By contrast we only consider the outer product adaptation scheme described in [034][033] which is incremental.

Consider a network with N units which is used to store k archetypes. Let V^a be the state vector of archetype a in the range 1 to k. The outer product computes W by:

$$W_{ij} = \sum_{a=1}^{k} W_{ij}^a$$

$$\text{where } W_{ij}^a = V_i^a V_j^a$$

It is immediate that W is a symmetrical matrix. Notice also that each $W_{ij}^a = \pm 1$.

We can express this in matrix notation as:

$$W = \sum_{a=1}^{k} W^a \quad (1)$$

$$\text{where } W^a = V^a V^{aT}$$

Here we are using V^a as a column vector, with V^{aT} its transpose.

Since W is a sum of terms derived from every individual pattern it can be calculated incrementally while the patterns are presented to the system sequentially by the environment. In other words, when a new pattern V^a is presented the new W can be obtained from the previous W and V^a itself.

An example of how the outer product encoding scheme is calculated is given in Exhibit 3.3. Despite the matrix-vector notation used throughout for analytical purposes, the computation involved in the encoding phase of the system can be performed locally since all the necessary information is available at the connection level. Maintaining this localized form of computation is an important design constraint for the systems suggested in chapter 4.

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

W^1

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

W^2

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

W^3

3	-3	1	-3	3	-1
-3	3	-1	3	-3	1
1	-1	3	-1	1	-3
-3	3	-1	3	-3	1
3	-3	1	-3	3	-1
-1	1	-3	1	-1	3

$W = W^1 + W^2 + W^3$

Exhibit 3.3 The intermediate W^1, W^2, W^3 and final weight matrix W using the sum of outer product scheme to store patterns $V^1 = [1,-1,1,-1,1,-1]$, $V^2 = [-1,1,-1,1,-1,1]$ and $V^3 = [1,-1,-1,-1,1,1]$.

3.3.4 Evaluation

We now consider the dynamic behaviour of Hopfield networks once W has been established. Assume we want to retrieve vector V^b we have stored previously. We clamp each of the values of the vector V^b to the corresponding units of the network. The new state of each unit **i** is given by:

$$V_i = \sum_{j=1}^{N} W_{ij} V_j^b$$

In matrix notation the output vector V of the network is:

$$V = W V^b$$

If we substitute for W with its equivalent from (1), then

$$V = \sum_{i=1}^{k} V^a V^{aT} V^b = V^b V^{bT} V^b + \sum_{i \neq k}^{k} V^a V^{aT} V^b \qquad (2)$$

If V^b is orthogonal to the rest of the archetypes V^a the summation term in equation (2) becomes equal to zero [010]. Since $V^T V = N$:

$$V = V^b V^{bT} V^b = N V^b \qquad (3)$$

$$V = \text{sign}(N V^b) = \text{sign}(V^b)$$

If V^b is not orthogonal to the rest of the archetypes then the summation term in equation (2) is $\neq 0$ and the response pattern contains noise:

$$V = \text{sign}(V^b) + \text{sign}\left(\sum_{i \neq k}^{k} V^a V^{aT} V^b\right)$$

Every probe V^b which by using (2) recalls itself, is called a *fixed point of attraction.* As shown in [010] there are fixed points of attraction that are not archetypes. For example if the system is probed with non-archetype $-V^b$ it will recall V:

$$V = \text{sign}(-V^b)$$

This means that the negation of every archetype is also a fixed point of attraction. If the system is probed with a pattern other than an archetype it can be shown that by a generalization process described in [010] it will converge to the fixed point of attraction closest to the probe. Note that every archetype is a fixed point of attraction. Therefore for a probe which lies close to an archetype the system will always recall the archetype. Exhibit 3.4 shows such an example. Consider the same archetypes as in Exhibit 3.3. Using the weight matrix W the system was probed with three vectors identical to the archetypes except for one element that had opposite sign (These elements are shown in Exhibit 3.4 using an outline font).

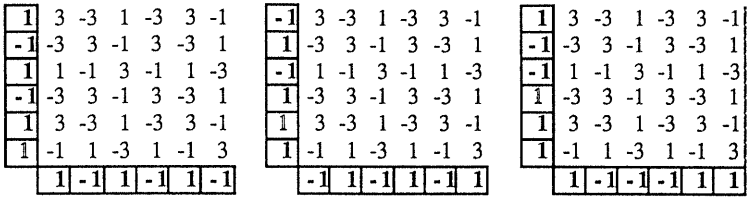


Exhibit 3.4 The system described in Exhibit 3.3 is probed with three vectors that correspond to the archetypes but with one negated element. The system responded with the corresponding archetype.

In Exhibits 3.4 and 3.6 *synchronous* evaluation of the network was used. That is the new state of a unit is not taken into consideration for the computation of the new state of the other units.

By contrast Hopfield [034] provides a different explanation about the convergence of the system into one of the archetypes by introducing an energy metric for the system.

$$E = -(\tfrac{1}{2}) \sum_{i}^{N} \sum_{j=1\, j\neq i}^{N} W_{ij}\, V_i\, V_j \qquad (4)$$

The change in energy is therefore:

$$\Delta E = -\Delta V_i \sum_{j\neq i}^{N} W_{ij}\, V_j$$

Forcing ΔV_i to take the sign of the net input ($\sum_{j\neq i}^{N} W_{ij}\, V_j$) to unit i, means that ΔE is always negative and that during recall every change in the state of a unit reduces the overall energy of the system. In effect the system performs a "discrete" gradient descent on the energy surface defined by equation (4).

Simulating such systems on a digital computer introduces a synchronization problem with respect to the update of the units' states. As it is suggested by the energy function in (4), information about the new state of a unit must be readily available to all the other units. We will call this method of updating the units *asynchronous* evaluation. This form of evaluation allows for the units to

be updated in an arbitrary order. Asynchronous update of the units is qualitatively superior to synchronous update because it comes nearer to true gradient descent. To improve the results in the synchronous case, iteration methods are used. In the simplest case, the response pattern of the network is used as new to probe the system; this cycle is repeated until a stable state is reached. In practice asynchronous update serializes the computation in the system because it requires N time steps to be completed.

A crucial aspect of the system's behaviour depends on whether a fixed point of attraction is developed for every archetype. Archetypes that lie to close to each other are fused and the system demonstrates an undesirable behaviour. As we have seen, if the archetypes are not mutually orthogonal noise is inserted during the recall process and the system may not converge to an archetype. An example of such behaviour is shown in [010]. Here we provide another example to demonstrate convergence to the wrong archetype. Consider a Hopfield network that consists of six units and the corresponding connections. Consider the following archetypes V^1,V^2 and V^3:

V^1 = [1,1,1,1,1,1] , V^2 = [1,-1,-1,1,-1,1] and V^3= [-1,1,-1,-1,-1,1]. Exhibit 3.5 shows the adaptation process of the system and the weight matrix W.

	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1

	1	-1	-1	1	-1	1
1	2	0	0	2	0	2
-1	0	2	2	0	2	0
-1	0	2	2	0	2	0
1	2	0	0	2	0	2
-1	0	2	2	0	2	0
1	2	0	0	2	0	2

	-1	1	-1	-1	-1	1
-1	3	-1	1	3	1	1
1	-1	3	1	-1	1	1
-1	1	1	3	1	3	-1
-1	3	-1	1	3	1	1
-1	1	1	3	1	3	-1
1	1	1	-1	1	-1	3

Exhibit 3.5 The intermediate and final weight matrix W using the sum of outer product scheme to store patterns V^1 = [1,1,1,1,1,1] , V^2 = [1,-1,-1,1,-1,1] and V^3 = [-1,1,-1,-1,-1,1].

Having computed the final weight matrix W, the network was tested using three probe vectors. The first probe was the original vector V^1 while the other two were X^2 = [-1,-1,-1,1,-1,1] and X^3 = [-1,1,-1,-1,-1,-1] that differ in one position from archetypes V^2 and V^3 respectively.

1	3	-1	1	3	1	1
1	-1	3	1	-1	1	1
1	1	1	3	1	3	-1
1	3	-1	1	3	1	1
1	1	1	3	1	3	-1
1	1	1	-1	1	-1	3
	1	**1**	**1**	**1**	**1**	**1**

-1	3	-1	1	3	1	1
-1	-1	3	1	-1	1	1
-1	1	1	3	1	3	-1
1	3	-1	1	3	1	1
-1	1	1	3	1	3	-1
1	1	1	-1	1	-1	3
	-1	**-1**	**-1**	**-1**	**-1**	**1**

-1	3	-1	1	3	1	1
1	-1	3	1	-1	1	1
-1	1	1	3	1	3	-1
-1	3	-1	1	3	1	1
-1	1	1	3	1	3	-1
-1	1	1	-1	1	-1	3
	-1	**1**	**-1**	**-1**	**-1**	**-1**

Exhibit 3.6 The system shown in Exhibit 3.5 is probed with three vectors V^1 = [1,1,1,1,1,1] , X^2 = [1,-1,-1,1,-1,1] and X^3 = [-1,1,-1,-1,-1,1]. For the first vector the system responded correctly but for the second and the third probe vectors the system responded with non archetypes. The changed elements are denoted as before using an outline font.

Exhibit 3.6 shows the response of the system for every probe vector. The response to probe V^1 was archetype V^1. When the system was probed with vector X^2 which lies at a distance of 1 from archetype V^2 the system responded with the non archetype [-1,-1,-1,-1,-1,1]. Similarly, X^3 reproduces itself, thereby also failing to recall an archetype. Exhibit 3.6 clearly shows that there are more fixed points of attraction than archetypes. In general as the number of archetypes stored in the system increases, so does the number of fixed points. If the weight matrix has a non-zero diagonal the number of fixed points approaches 2^N as the number of archetypes in store increases. If the weight matrix has a zero diagonal the number of fixed points is limited by N because of the orthogonality constraint (see Section 3.3.5).

A minimal requirement for a Hopfield type system, if it is to operate

successfully within a real-world environment, is to be capable of storing and recalling all the archetypes accurately. The orthogonality constraint constitutes a serious limitation for the system. It is unreasonable to expect naturally emerging data to be orthogonal. On the other hand orthogonalization methods are far too complicated, and once a set of vectors is orthogonalized it is almost impossible to transform the vectors back to their original form. The effect of this is that the response of the system becomes incomprehensible for the user. If the effects of non-orthogonality cannot be eliminated, then Hopfield networks are inadequate as real-world pattern recall devices. The character set task we described in section 3.2 is a simple example where the patterns are not mutually orthogonal. The systems presented in Chapter 4 are an attempt to reduce the effects of non-orthogonality.

3.3.5 The Capacity Issue

So far we explained the behaviour of the system and discuss how the orthogonality constraint affects its performance. We define as the capacity of a Hopfield network the number of archetypes that are fixed points when every fixed point is an archetype. A weak capacity constraint is immediate from the size of the network. For a system with N units, any set of mutually orthogonal archetypes can be no bigger than N, and therefore an upper limit for the network's capacity is N itself.

Based on the assumption that random patterns are approximately orthogonal, the use of random archetypes instead of rather strictly orthogonal archetypes is suggested in [034][010]. There arises the problem of specifying the capacity of the system for random patterns. Hopfield claimed in [034] that the network is able to store 0.15N original patterns, provided that the archetypes have mutual correlations of no more than ± 0.05. Later, other researchers

[006][037]][038][033] established different estimates which in general suggest that the capacity of the network is less than Hopfield's initial claim. Probably the most detailed analysis on the capacity of Hopfield networks can be found in [010] which suggests that a network with N units is able to store $\frac{\gamma N}{4 (\log N)}$ where γ is the minimum correlation required between the probe and the archetype to be recalled. This result assumes that the archetype elements are chosen randomly and independently, and is valid only for the synchronous update case. The systems proposed in Chapter 4 do not require an accurate capacity estimate. All we shall need is the order of magnitude for the number of units required for a particular task. For example storing 26 patterns according to the formula given above would require approximately 10730 units.

What is important for our purposes is to be able to provide a sufficient amount of resources to secure integrity of information. Most of the systems described in chapter 2 use a local representation. In these systems the number of units is independent of the size of the patterns (although it must allow for the input units). For Hopfield networks the size of the pattern must be equated to the number of units and, is therefore closely linked to the capacity of the system. The next section discusses a problem that arises from this relation.

3.4 Matching the pattern to the size of a Hopfield network

3.4.1 The Transformation Method

Consider the problem of storing 26 archetypes of size 35. The solution to this seemingly trivial problem is not as straightforward as one would expect. For the adaptation process a value for every unit is required but only 35 values are available for every vector. Therefore, a transformation of the archetypes to a higher dimensional space is necessary before they can be encoded into a

Hopfield network. We provide a solution to this problem that will enable us to perform simulations on the Hopfield network storing the characters described in 3.2. The solution also provides some insight leading to the systems presented in Chapter 4.

The general form of the problem involves patterns of size N and a Hopfield network of minimum size n, with $N < n$ (usually $N << n$). The solution requires both an encoding of N-vectors as n-vectors and a suitable corresponding decoding to allow the recall vectors to be interpreted meaningfully. Additionally the transformation mechanism should:

- Be computationally simple and as parallel as possible.
- Maintain the generalization and content-addressable properties of the Hopfield network.
- Preserve nearness.

Not every transformation mechanism satisfies these conditions. For example a conventional look-up table has no generalization. If the input is not specified exactly the memory cannot be recalled.

For simplicity we take the size of the network to be a multiple of the size of the input pattern say $n = k N$. This does not have any effect on the network's capacity as long as the number chosen is sufficiently large. In this way we view the state vector of the network as an array M with N columns each of length k. The transformation is as follows:

• Input

The value from each position of the input vector is fanned out to the corresponding column of M (Exhibit 3.7)

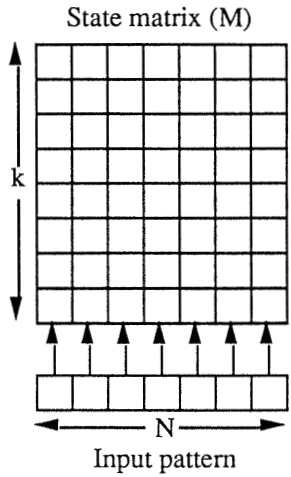


Exhibit 3.7 Mapping a N-size input pattern vector to the size vector required by the Hopfield network.

• Output

The probe vector is transformed into a n-size vector as described for input (Exhibit 3.7). The system is evaluated as described in Section 3.3.4 and responds with an M matrix. The elements of each column are summed and thresholded as shown in Exhibit 3.8 to give the value (±1) of the corresponding position of the response pattern. Ideally every element in a single column of M should respond identically. In the case the recalls are incorrect the thresholded summation allows for up to 50% error within a single column.

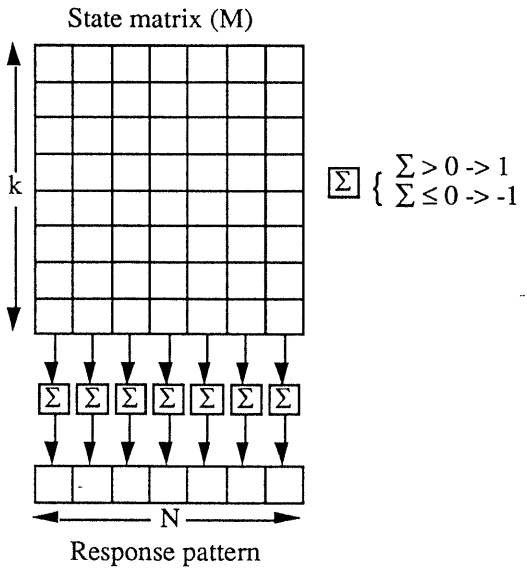


Exhibit 3.8 Mapping the response pattern from the Hopfield network to the N-size final response pattern.

This transformation mechanism satisfies all the above mentioned conditions but unfortunately it does not take advantage of the true capacity of the enlarged system. This is so, because the weights of the n size network are exact copies of the weights of an N size network. In other words the n sized network contains no more information than the smaller N sized network. This theoretical observation was verified by simulation results where the response of the system was identical for various n.

3.4.2 Simulation Results

We simulated the Hopfield network with the data set described in Section 3.2 using this transformation scheme. Unfortunately, the basic problem that we had anticipated arising from the non-orthogonality appeared. The network

was unable to recall accurately any of the original vectors (characters) even when one was used as a probe. The average recall error was 43% per character. Most often when the system was probed with one of the archetypes it responded with a non-memory. Both asynchronous and synchronous evaluation strategies were used, with iterations in the synchronous case. All our results were qualitatively unacceptable. Our next step is to look for a transformation scheme which alleviates the non-orthogonality problem.

3.5 Associative Memory Systems

3.5.1 Introduction

While Hopfield networks were being investigated attempts were made to use associative memory systems as pattern recall devices. An account of a particular associative memory system, the correlograph, was given in Chapter 2. Here we adopt a different associative memory system as described in [039] and extended in [040]. We are not concerned with the behaviour of associative memory systems in a real-world environment because in their standard form, they are inappropriate as pattern recall devices. However, associative memory forms a substantial part of the systems described in Chapter 4. The next section describes the standard system, while in sections 3.5.3 and 3.5.4 we present an extension of the system and some interesting simulation results.

3.5.2 Training and Evaluation of the Network

An associative memory system comprises of an input layer of K units and an output layer of L units (Exhibit 3.9).

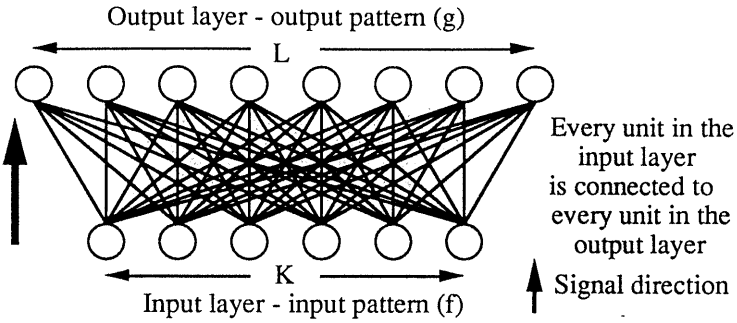


Exhibit 3.9 An associative memory system.

Unlike [039] we do not require K = L. Every input unit is connected to every output unit with a connection that has a weight value. The system is used to associate a pattern **f** at the input units with a pattern of **g** at the output units. By means of suitable weights the system is able to store a set of such associations. The weight matrix W is of size Kx L. Suppose that we want to store the association $f^i \rightarrow g^i$ (i = 1..n). When vectors f are associated with g the resulting weight matrix W is computed as follows:

$$W = \sum_{i=1}^{n} W^i \implies W = \sum_{i=1}^{n} g^i f^{iT} \quad (6)$$

Notice that W can be computed incrementally writing:

$$W^{(j)} = \sum_{i=1}^{j} W^{(i)}$$

We have:

$$W^{(0)} = 0$$

$$W^{(j)} = W^{(j-1)} + g^j f^{jT}$$

A probe vector f is applied to the input units. The system is evaluated and responds with an output vector g:

$$g = (W f) \quad (7)$$

In case $f = f^k$ we have that:

$$g = W f^k = W^k f^k + \sum_{i \neq k}^{n} W^i f^k = g^k + \sum_{i \neq k}^{n} g^i \left(f^{iT} f^k \right)$$

If the f^i vectors are mutually orthogonal $f^{iT} f^k = 0$ giving:

$$g = g^k$$

If the f vectors are not mutually orthogonal noise is inserted in the response of the system. This problem is identical to the one identified for Hopfield networks. In fact, the whole analysis of the system is very similar to that of the Hopfield system. Associative memory can be viewed as a Hopfield network with restricted connectivity. This view of associative memory is considered in Section 5.4. The maximum notional capacity, that is, the number of associations that can be stored in the system is K. It is claimed in [039] that the system is capable of associating an incomplete or noisy input pattern with the appropriate output pattern. Similarly to the Hopfield case, a probe pattern f will be associated with the output pattern g^k that corresponds to the nearest pattern f^k. Consider the example given in Exhibits 3.10 and 3.11 where three pairs of input-output vectors are associated in a network that has six input and output units.

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

	1	-1	1	-1	1	-1
1	2	-2	2	-2	2	-2
1	2	-2	2	-2	2	-2
-1	-2	2	-2	2	-2	2
-1	-2	2	-2	2	-2	2
1	2	-2	2	-2	2	-2
-1	-2	2	-2	2	-2	2

	-1	1	-1	1	1	-1
-1	3	-3	3	-3	1	-1
1	1	-1	1	-1	3	-3
-1	-1	1	-1	1	-3	3
1	-3	3	-3	3	-1	1
-1	3	-3	3	-3	1	-1
1	-3	3	-3	3	-1	1

Exhibit 3.10 Three pairs of vectors are associated and the final weight matrix is derived. Operationally and analytically the system resembles a Hopfield network (Exhibit 3.3)

The input-output pairs to be associate:

$(f^1 = [-1,-1,1,1\text{-}1,1],\ g^1 = [-1,1,-1,1,-1,1])$,

$(f^2 = [1,1,-1,-1,1,-1],\ g^2 = [1,-1,1,-1,1,-1])$

$(f^3 = [-1,1,-1,1,-1,1],\ g^3 = [-1,1,-1,1,1,-1])$

are stored in the system and the weight matrix is produced (Exhibit 3.10).

The network is evaluated using the following three probe vectors:

$X^1 = [-1,-1,1,1,-1,1]$ $(= f_1)$, $X^2 = [1,1,-1,-1,-1,1]$ and $X^3 = [1,1,-1,1,-1,1]$

Notice that X^1 is the same as f_1 while X^2 and X^3 are at a distance of 1 from f^2 and f^3 respectively.

In Exhibit 3.11 the response of the system for every probe vector is shown.

-1	3	-3	3	-3	1	-1
-1	1	-1	1	-1	3	-3
1	-1	1	-1	1	-3	3
1	-3	3	-3	3	-1	1
-1	3	-3	3	-3	1	-1
1	-3	3	-3	3	-1	1

-1	1	-1	1	-1	1

1	3	-3	3	-3	1	-1
1	1	-1	1	-1	3	-3
-1	-1	1	-1	1	-3	3
-1	-3	3	-3	3	-1	1
1	3	-3	3	-3	1	-1
1	-3	3	-3	3	-1	1

1	-1	1	-1	1	-1

1	3	-3	3	-3	1	-1
1	1	-1	1	-1	3	-3
-1	-1	1	-1	1	-3	3
1	-3	3	-3	3	-1	1
-1	3	-3	3	-3	1	-1
1	-3	3	-3	3	-1	1

-1	1	-1	1	1	-1

Exhibit 3.11 The final weight matrix (Exhibit 3.10) is used to compute the response output vector for each of the probe vectors. For every probe vector the system responded with the appropriate output vector.

As shown in Exhibit 3.11 the network responded with g^1 for probe vector X^1, with g^2 for the probe vector X^2 and with vector g^3 for the probe vector X^3.

A threshold function was applied in every output unit. Associative memories can be used as pattern recall devices by making $g^i = f^i$, that is mapping every pattern to itself. Pattern recall systems of this kind are not of particular interest to us because they still suffer from the orthogonality problem. In the

following two sections, we propose a modification of the standard associative memory system which will serve as a pattern recall device that demonstrates improved behaviour compared to the standard Hopfield system.

3.5.3 Bidirectional Associative Memory (BAM) Systems

The use of associative memories in a bidirectional fashion is suggested in [040] [054]. Based on the observation that the reverse association can be achieved by probing the network from the output units an associative memory was used to store temporally distributed patterns. Here the same idea is used to construct an auto-associative memory.

Suppose we want to store the reverse associations g^i --> f^i. In other words, we want the network to recall the f^i pattern in response to the g^i pattern. The topology of the network remains as in Exhibit 3.9 and the weight matrix definition in (3.5.2) leads to the weight matrix W ':

$$W' = \sum_{i=1}^{n} f^i g^{iT} = W^T$$

where W as in (6)

No extra computation is required in order to achieve the desired reverse association of W, because the difference between W and W^T is one of link direction; the weight value associated with the link remains the same.

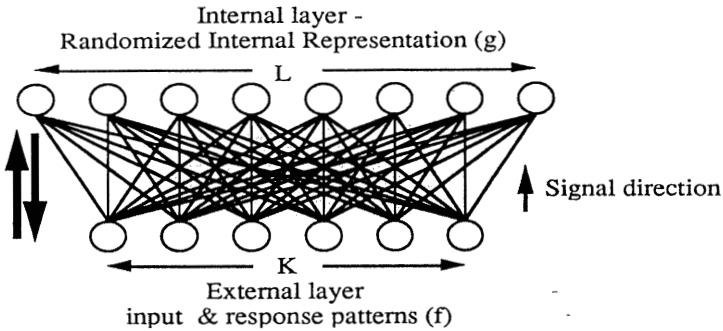


Exhibit 3.11 A BAM system.

The bidirectional use of an associative memory suggests that by probing the system twice, once from the input units to the output units and then in reverse, the original pattern f can be recalled. This observation is the basis for the use of a BAM as an auto-associative memory (Exhibit 3.11). However in this case only the f patterns, not the g patterns are available. This immediately raises the question of how to generate the weight matrix.

Our solution is to associate every pattern f with a randomly generated pattern g. We will call the pattern g the *randomized internal representation* (RIR) of pattern f. In our system, the layer for input and output is called the *external* layer while the layer for the randomized internal representations is called the *internal* layer. Simulations results with this system are presented in the next section.

3.5.4 Simulation Results

Again we use the data set described in section 3.2. We present results obtained using networks that had 35 external units and a various numbers of internal units. Note that the RIR vectors are expected to be approximately orthogonal (see Section 3.3.5). The use of orthogonal g vectors is also

suggested but as it is mentioned before the computation involved is very tedious.

The weight matrix of section 3.5.2 is computed. Each character is used in turn as a probe. First the network is probed forward and again noise is inserted at this stage because of the non-orthogonality of the characters. The RIR is then used to probe the network in reverse producing the response pattern at the external units. The response pattern should be identical to the probe if noise were not inserted during the forward phase. Note that insignificant noise is inserted in the second phase of the computation precisely because the RIR are expected to be approximately orthogonal. The experiment was conducted with different L values and Exhibit 3.10 summarizes the results.

We expected that the generalization properties of the second phase would to some extent offset the noise introduced in the forward phase. The results show that the network is not strong enough to eliminate the noise effects for the particular data (section 3.2) set used. The results are significantly improved compared to the Hopfield network (Section 3.4.2). We were able to recall one pattern 100% accurately while the average recall error was between 10.10% and 9.20% per character (Exhibit 3.12).

L	Number of 100% accurately recalled patterns	Average % error per character
1050	1	9.40
2100	1	10.10
3000	1	9.50
3500	1	9.20
4000	1	9.30

Exhibit 3.12 Results of an associative memory network for different numbers of internal units.

Although the results were not totally satisfactory, RIRs prove promising and they are implemented in the systems described in Chapter 4.

3.6 Auto-Associative Memories in a Real-World Environment

3.6.1 Introduction

So far we described Hopfield and BAM systems and demonstrated their advantages and disadvantages. Both systems were considered within a standard connectionist paradigm. The adaptation phase was separated from the evaluation phase and a fixed environment was considered. In the following sections we discuss the application of these systems in a real-world environment with respect to:

- Self-organization
- Continuous adaptation

Integrity of information has already been covered by the discussion on the effects of non-orthogonal archetypes while context sensitive recall is discussed after the introduction of RIRs (Section 4.7).

3.6.2 Self-Organization

The systems presented in the previous sections can be considered self-organized only under the *weak* definition of the term. Hopfield and BAM networks are capable of operating without a teacher, but their organization does not allow for an adaptive structure. To guarantee that their performance is not affected by their size it is necessary to provide an adequate number of units (see Section 3.3.5) to match the number (U) of archetypes in the environment. In the case where there is *a priori* information about U and the environment remains unchanged the size of the system can be defined

accurately. For a real-world environment an infinitely large number of units is necessary to guarantee that all the archetypes will be stored successfully.

3.6.3 Continuous Adaptation

Consider a system with infinitely many units. As is shown in Sections 3.3.3 and 3.5.2 the weight matrix computation is incremental, which means that archetypes are stored in the system independently from one another. It was explained in Chapter 1 that incremental adaptation allows for continuous adaptation only when the stability-plasticity problem is solved.

It is obvious that Hopfield and BAM systems, in their standard form, cannot distinguish between a novel archetype and a noisy probe. To enable Hopfield and BAM systems to make such a distinction, a *novelty detector* device (Exhibit 3.13) is used. The proposed detector device also applies to the pattern recall devices described in Chapter 4.

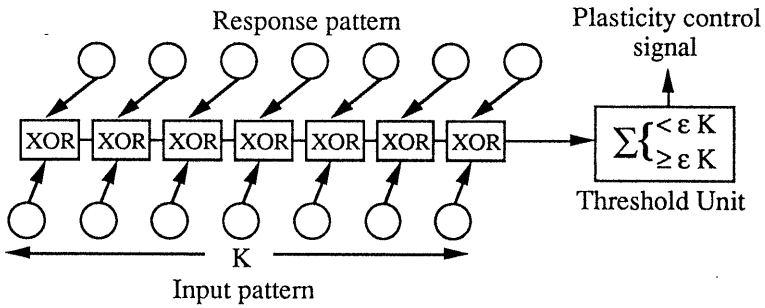


Exhibit 3.13 A novelty detector device for pattern recall systems.

The detector is very similar to the orienting subsystem in ART systems [008]. It comprises K XOR units connected to a threshold unit. The threshold unit embodies a threshold value εK , where $(0 \leq \varepsilon \leq 1)$. Note that both the XOR units and the novelty detector unit can be implemented as connectionist

networks.

Consider a system that already has some archetypes in store. A probe pattern is supplied and the system recalls the closest archetype. Note that for mutually orthogonal archetypes and for an adequate number of units, the system will always respond with an archetype. The hamming distance between the archetype and the probe is computed and compared against the value of the novelty detector unit. If the threshold unit becomes active the system is forced to change its weight values according to the probe pattern which now becomes an archetype. If the threshold unit remains inactive, the probe is considered as a noisy version of the recalled archetype and recall is completed. The value of ε determines the noise allowed in a probe for it to be considered as a noisy pattern. A case where the proposed novelty detector fails to function in the desired manner, is shown in Chapter 5.

The assumption of infinitely many units is physically unrealistic. As explained in Chapter 1 the need for optimization of the resources becomes apparent. An optimization mechanism can be implemented for the pattern recall systems by effectively deleting archetypes stored in the system. Consider a BAM system its weight matrix W and the archetype f^k. Assume that we want to remove f^k from the system. Note that:

$$W = W^k + \sum_{i \neq k} W^i$$

$$W^k = g^k f^{kT}$$

The deletion process is as follows:

Step 1: Probe the system with f^k and compute: $g^k = W f^k$

Step 2: Compute: $W^k = g^k f^{kT}$

Step 3: Delete f^k by computing the new weight matrix $W' = W - W^k$

A similar process can be used for Hopfield networks and the systems described in Chapter 4 by appropriately modifying step 2. Deleting f^k

effectively releases the space for a novel archetype to be stored. Clearly, the above process runs counter to the self-organization character of the system in the sense that a teacher is required to delete f^k. In the ideal case, the system must be able to sense which archetypes are no longer available in the environment and proceed with their deletion from the store. Unfortunately this is not possible using the above deletion operation because the non existent archetype is required for the computation. Nevertheless, this deletion operation is still extremely useful because it allows for an on-line optimization of the system's resources.

3.7 Summary

In this chapter we presented Hopfield networks and showed that they can be used as pattern recall devices with noise reduction properties. It was also shown that the quality of the recall is seriously affected by the non orthogonality of the archetypes. Associative memory systems were presented and the use of Randomized Internal Representations (RIR) in conjunction with a Bidirectional Associative Memory (BAM) was used as a pattern recall device. Finally both Hopfield and BAM systems were examined against the design constraints of a real-world environment and a novelty detector device was introduced as a solution to the stability-plasticity problem.

CHAPTER 4
RANDOMIZED INTERNAL REPRESENTATION (RIR)

4.1 Introduction

In the previous chapter Hopfield networks and BAM systems were discussed in detail. We showed that both systems suffer when the archetypes to be stored are non-orthogonal. Both systems show significant generalization properties over noisy patterns when orthogonality is guaranteed. By means of a novelty detector both systems are capable of continuous adaptation.

In this chapter we propose a system based on randomized internal representations that preserves the simplicity of the outer-product adaptation rule. The system consists of a Hopfield network and a BAM system. We shall refer to the embedded Hopfield network as the *Hopfield cluster*. The primary aim of the system is to alleviate the effects of non-orthogonality in the environmental data. The proposed system provides a single transformation mechanism that also provides a solution for the matching problem of Section 3.5. In fact the system described below is one of a general class of systems based on randomized internal representation. Two other members of this class are briefly discussed at the end of the chapter.

4.2 A combination of BAM and Hopfield Networks

4.2.1 Architecture

The architecture of the proposed system is shown in Exhibit 4.1 and comprises the two layer BAM together with a Hopfield cluster. BAM systems provide a flexibility in the size of the internal layer. The size of the external layer is constrained by the size of the environmental patterns and we assume

that this remains fixed throughout . We choose the size of the external layer to match the size of the Hopfield cluster. The size of the Hopfield cluster is determined by the capacity considerations described in Section 3.3.5. As shown in Exhibit 4.1 each of the internal units of the associative memory network is connected with a bidirectional link to one of the units of the Hopfield cluster in such a way that there is an one to one correspondence.

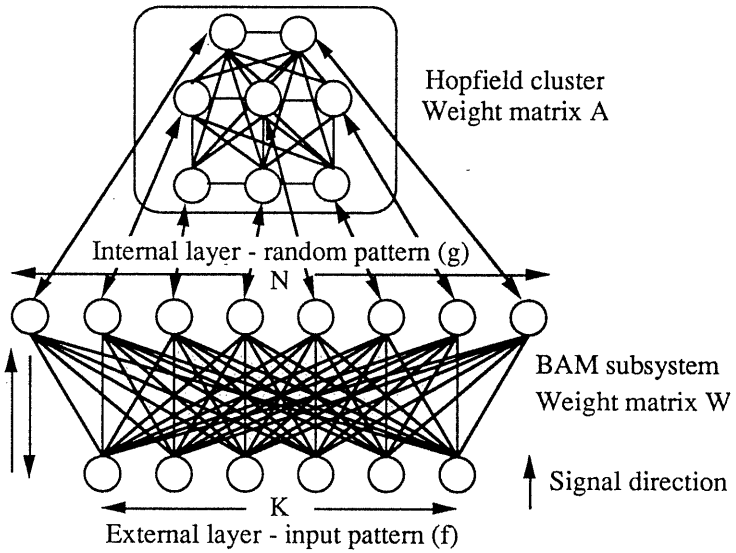


Exhibit 4.1 The proposed system is a combination of a BAM system and a Hopfield network.

Note that these links do not have weight values and their purpose is only to convey signals between the internal layer and the Hopfield cluster. Therefore, the proposed structure can be reduced to a Hopfield network connected to the external layer. In this case, the total number of units in the system is equal to the size of the Hopfield network (N) plus the size of the external layer (K) while the total number of connections is N(N+K). The relative connection

complexity of the proposed system is therefore $\frac{N(N+K)}{N^2} = 1 + \frac{K}{N}$ which is only slightly in excess of 1 because $K << N$.

Intuitively, we expected that the noise reduction properties of both subsystems would be able to eliminate this noise inserted during the recall process—the simulation results in Sections 4.2.5 and 4.4.3 show an average noise of 3% per pattern in the response of the system.

4.2.3 Adaptation Scheme

Consider a set of archetypes f^i to be encoded in the system. For every f^i a random pattern g^i is generated. The weight matrices of the BAM subsystem (W) and the Hopfield cluster (A) are computed:

$$W = \sum_{i=1}^{n} W^i$$

$$A = \sum_{i=1}^{n} A^i$$

$$\text{where } W^i = g^i f^{iT} \text{ and } A^i = g^i g^{iT}$$

The g vectors are random and therefore as explained in Section 3.3.5 the Hopfield cluster will be able to recall them accurately. Note that W and A can be computed simultaneously because they do not interact.

EXAMPLE

Consider the archetypes:

$f^1 = [1,1,1,1,1,1]$, $f^2 = [1,-1,-1,1,-1,1]$ and $f^3 = [-1,1,-1,-1,-1,1]$

for storage in a system with 6 external units, 6 internal units and 6 units in Hopfield cluster. For every archetype a random internal pattern was provided. Exhibit 4.2 shows the computation of weight matrices W and A.

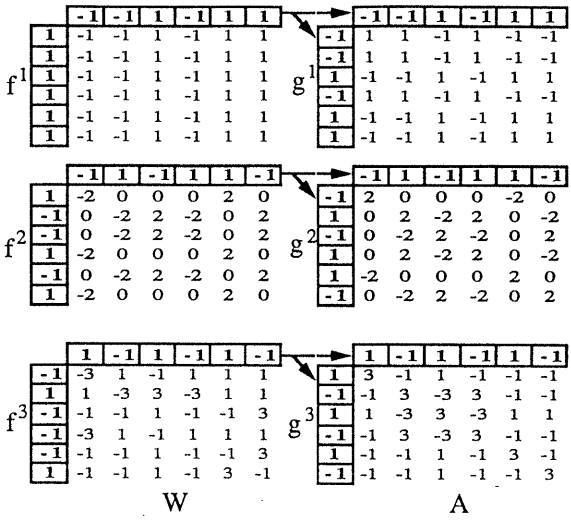


Exhibit 4.2 W and A when archetypes f^1, f^2 and f^3 are stored in the system.

Note that the patterns used in this example are those used in Section 3.3.4 where a case of erroneous recall was demonstrated for a standard Hopfield network. In the next section we demonstrate that the proposed system is capable of recalling accurately the noisy probes of exhibit 3.6.

4.2.4 Evaluation

Recalling an archetype is a three step process. A probe f^k is clamped at the external layer and the response pattern g^k is generated at the internal layer:

$$g^k = W f^k$$

This pattern is then forwarded to the Hopfield cluster. The Hopfield cluster is evaluated in the usual manner (Section 3.3.4) and the resulting pattern h^k is fed back to the internal layer.

$$h^k = A g^k$$

The BAM subsystem is then probed in reverse, and the final response f

appears at the external layer.

$$f = W^T h^k$$

This description seems to imply an explicitly synchronized behaviour, which is strictly a property of simulations and not of the system itself. However the system could certainly evaluate asynchronously but this raises certain timing problems from a physical viewpoint.

Noise is inserted into the recall process during the first phase of the network's evaluation when computing g^k in the case where the f vectors are not mutually orthogonal. As long as the noise is under 50% the Hopfield cluster should recall g^k accurately. In the case where the output from the Hopfield cluster is not noise free, the BAM subsystem is expected to eliminate the remaining noise.

EXAMPLE

In Exhibit 4.3 the system described in Exhibit 4.2 is probed with the pattern:

$$f^k = [-1,-1,-1,1,-1,1]$$

which lies at a distance of 1 from the original pattern f^2 (the erroneous element is denoted by an outline font).

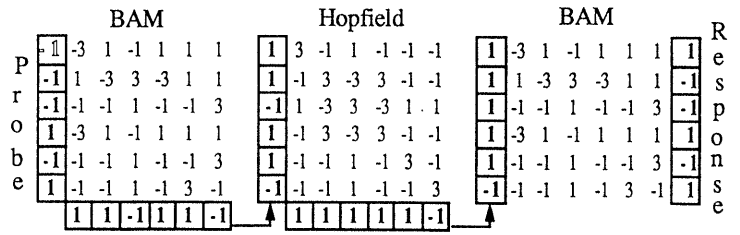


Exhibit 4.3 The system is probed with the pattern [-1,-1,-1,1,-1,1] which lies at a distance of 1 from the original pattern f^2. After the three stage evaluation the system responded appropriately with the original pattern f^2.

When the corresponding standard Hopfield system was tested with the same probe pattern it responded with an non-pattern instead of the pattern f^2 as expected (see Exhibit 3.6). As shown in Exhibit 4.3 the associative memory subsystem is evaluated from the external units to to internal units and responds with pattern [1,1,-1,1,1,-1]. This pattern is then provided to the Hopfield cluster which after synchronous evaluation, responds with pattern [1,1,1,1,1,-1]. This pattern is fed back to the internal units and the final response of the system, pattern f^2, appears at the external units.

4.2.5 Simulation Results

Again we used the set of character described in Section 3.2. Every pattern was stored and recalled from the system as described in Section 4.2.3 and 4.2.4 respectively. As mentioned above, the non-orthogonality of the archetypes inserts noise when recalling the RIR from the probe. As long as this noise is less than 50% per character a suitably sized Hopfield cluster is capable of recalling the RIR archetype. To find the size of the Hopfield cluster that will guarantee accurate recall two values are required:

(a) The number of archetypes

(b) The error introduced during the first phase of the evaluation

Within a real world environment neither (a) nor (b) is available. Nevertheless, for a wide range of applications (a) is known *a priori* and (b) is easily computed from equation (2) in Section 3.3.4. This computation can be done through time by generating random values for every individual internal unit. Exhibit 4.4 shows the error (%) introduced during recall for every character in the set.

Exhibit 4.4 The error % introduced during the first phase of the recall for every character in the data set. Note that this error in independent of the size of the internal layer.

A maximum error of 32.17% was introduced when probing with character H. Using the McEliece capacity formula (see Section 3.3.5) it was estimated that for $N \geq 7500$ accurate recall from the Hopfield cluster is guaranteed. Simulating a system with 7500 units required for computer memory which would have exceeded the capacity of our computer system.

N	Number of 100% accurately recalled patterns	Average% recall error per character
1050	8	6.8
2100	8	4.5
3000	8	4.2
3500	10	3.8
4000	10	3.5

Exhibit 4.5 Results obtained for different N and asynchronous evaluation of the Hopfield cluster.

However, we were able to simulate systems with a maximum of 4000 units in the Hopfield cluster. Exhibit 4.5 shows some of these simulation results obtained for $N \leq 4000$. These results were obtained for asynchronous evaluation of the Hopfield cluster. According to the McEliece capacity formula for asynchronous evaluation [010], 4000 units suffice for accurate recall when the probe contains noise $\leq 33\%$, a suggestion which, to a certain extent, contradicts our simulation results. This contradiction could be explained by the fact that the McEliece formulae refer to the *asymptotic* capacity of Hopfield networks. A simulation program for the above system equipped with the novelty detector of Section 3.6.3 is given in Appendix A.

4.3 A Shared Content-Addressable Memory (SCAM)

4.3.1 Pattern Classification Using Contextual Information

The system just described guarantees the recall of the closest archetype stored in the system. Quite often in a real-world environment patterns that appear simultaneously have some form of semantic relation between them. For example, the recognition (recall) of an object may depend on the presence of visual as well as sound information, and in general it is easier to recognize objects when some form of contextual information is available. Two basic ways of utilizing context information can be identified.

One way is by a hierarchical structure that represents different conceptual levels, eg letter-word-sentence. A letter which cannot be recognized alone, may be recognized successfully within the context of a word, and similarly a word is more easily recognized within the context of a sentence. In fact a significant part of our daily recognition depends heavily on contextual information.

In the absence of a hierarchical organization, contextual information may be

encoded by binding patterns together. Binding patterns requires either that patterns can be represented by winner-units between which an excitatory connection can be established; or that patterns share a common representation. Our system uses distributed representations making the encoding of a hierarchical organization extremely difficult. However the use of Randomized Internal Representations allows the encoding of contextual information by enabling patterns to share a common representation. We now describe an extension of our system which allows for dynamic binding of patterns.

4.3.2 Architecture

The extended system comprises a single Hopfield cluster and multiple BAM interfaces attached to it as shown in Exhibit 4.4. We shall assume a sufficiently large Hopfield cluster and many users attached to it with every user having their own BAM subsystem (Exhibit 4.7). The external layers can be user-specific. To each user the system appears like a dedicated copy of the basic system. Each of the users constructs their own interface according to the specifications of their respective application and store and recall archetypes using the common Hopfield cluster. In detail the operation is as follows.

There is a Random Pattern Generator (RPG) connected to all the internal layers of the BAM interfaces which provides all the random internal representations. When a pattern is to be stored by a user, the weights of the corresponding BAM subsystem are modified according to the random pattern provided by the RPG, exactly as in the single-user system. The Hopfield cluster also modifies its weights according to the same random pattern. Recall is exactly as in the basic system. The familiar three step operation takes place and the response appears at the external layer of a particular user. Notice that the user needs exclusive access to the Hopfield cluster. Although users may evaluate their

own BAM subsystems concurrently.

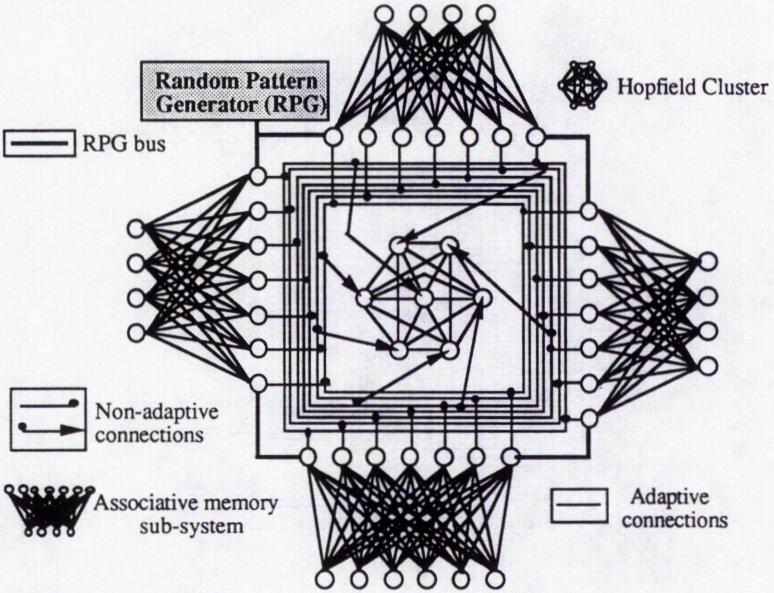


Exhibit 4.7 A shared content-addressable memory (SCAM). It allows many end users to use the same Hopfield cluster and provides dynamic binding of patterns.

4.3.3 Dynamic Binding and Synchronized Recall

In the proposed system dynamic binding of patterns is achieved by sharing the same RIR. Consider two archetypes one of which represents the letter A visually (V) and the other phonetically (P). Binding of V and P is achieved by setting $RIR^V = RIR^P$, thus forcing V and P to have a *common* representation in the Hopfield cluster. While recalling archetype V, RIR^V is also recalled which automatically allows for the recall of P. Furthermore, a synchronized recall of the two archetypes using noisy probes can be successful even in the case where neither of the probes is individually capable of recalling the

corresponding archetype. This case is demonstrated by the following example.

EXAMPLE

Consider a SCAM system with two BAM subsystems, BAM^1 and BAM^2, and a common Hopfield cluster.

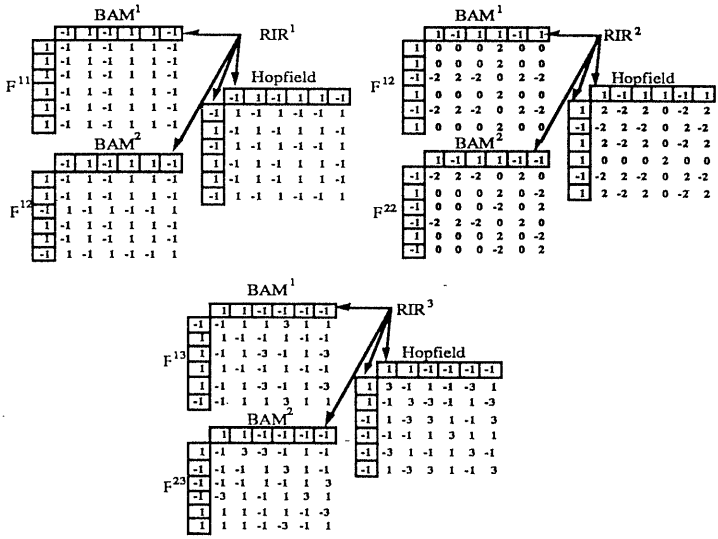


Exhibit 4.7 Patterns F^{11} and F^{12}, F^{21} and F^{22}, F^{13} and F^{23} are bind together by means of common RIRs.

Consider archetypes:

$F^{11} = [1,1,1,1,1,1]$, $F^{12} = [1,1,-1,1,-1,1]$, and $F^{13} = [\ -1,1,1,1,1,-1]$

for storage in the system via BAM^1 and archetypes:

$F^{21} = [1,1,-1,1,1,-1]$, $F^{22} = [1,1,-1,-1,1,-1]$ and $F^{23} = [1,-1,-1,-1,1,1]$

for storage via BAM^2. Storing is performed is such a way that patterns F^{11} and F^{21}, F^{12} and F^{22}, F13 and F^{23} are bind together by means of randomized common representations:

$RIR^1 = [-1,1-1,1,1-1]$, $RIR^2 = [1,-1,1,1-1,1]$ and $RIR^3 = [1,1,-1,-1,-1,-1]$ respectively. The storage operation is shown in Exhibit 4.7.

Consider:

$$X^1 = [-1,1,-1,1,-1,-1] \text{ as a probe for } BAM^1,$$

$$\text{and } X^2 = [1,-1,-1,1,-1,1] \text{ as a probe for } BAM^2$$

Note that X^1 and X^2 lie at a distance of 2 from F^{13} and F^{23} respectively. As shown in Exhibit 4.8 when probing with X^1 the system responds with a non-memory R^1. Similarly probing with X^2 the system responds with non-memory R^2.

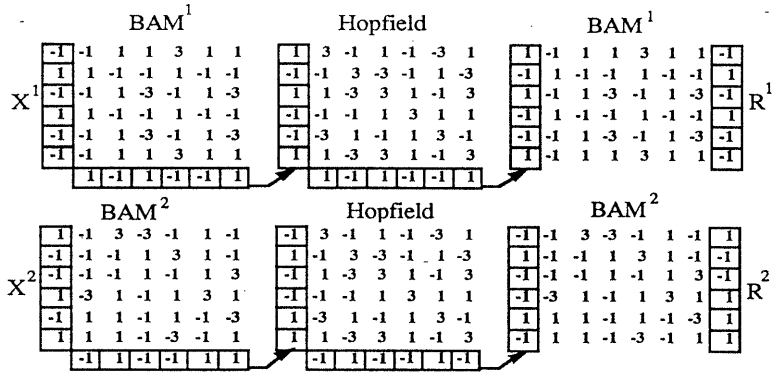


Exhibit 4.8 Probing with X^1 the system responds with a non-memory instead of archetype F^{13}. Similarly, the response for probe X^2 is a non-memory instead of archetype F^{23}.

By appropriately synchronizing the recall process it is possible to recall the correct archetypes. Consider the case where X^1 and X^2 are presented to the system simultaneously. The probe pattern for the Hopfield cluster is computed by thresholding the sum of the unthresholded responses from BAM^1 and BAM^2 as shown in Exhibit 4.8. The Hopfield cluster recalls RIR^3 which in turn produces F^{13} and F^{23} at the corresponding external layers

(Exhibit 4.9).

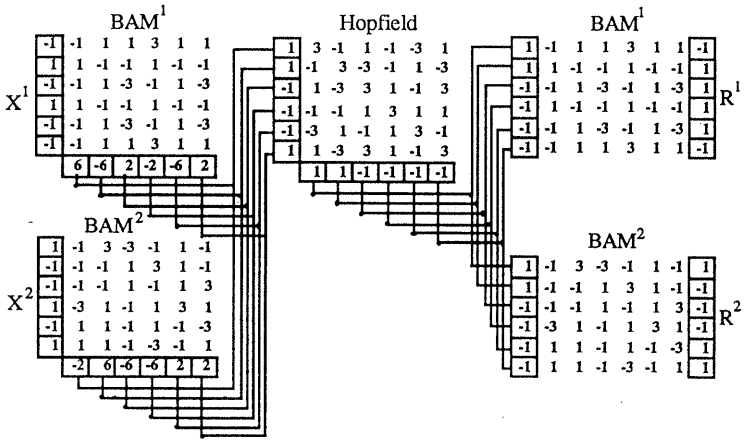


Exhibit 4.9 Synchronized recall. X^1 and X^2 are used.simultaneously and the system responds with archetypes F^{13} and F^{23} as expected.

Two different scenarios can be used to describe how the above process may be implemented in a real system. One scenario suggests that every BAM subsystem is dedicated to a one user. In this case, synchronization of binding and recall must be controlled by the individual users. Such control can be very easily achieved by appropriately using the output from the RPG unit.

The other scenario suggests that the system as a whole is exposed to a single environment and that every BAM^i (i=1..n) subsystem corresponds to a different interface. This system is operated by simply making the assumption that archetypes which appear simultaneously in the environment, are semantically related and therefore they should be bound together. This time, synchronization of binding and recall is left to the environment.

Simultaneously arriving probes are also taken to be semantically related and therefore the synchronized recall operation described above is applied

continuously.

Concluding, we would mention that providing every external layer with a novelty detector (see Section 3.6.3) enables a SCAM system to operate under a continuous adaptation regime.

4.4 A Three-Layer Bidirectional Auto-Associative Memory

4.4.1 Introduction

In this section we describe another system that serves as a pattern recall device, based on the same mechanism of Randomized Internal Representations. The proposed system is a three-layer bidirectional auto-associative memory that recalls non-orthogonal patterns with greater accuracy than the system presented in Section 4.2.

The system of Section 4.2 is modified by replacing the Hopfield cluster with another BAM. The proposed model consists of two BAM subsystems organized into a layered architecture (Exhibit 4.10).

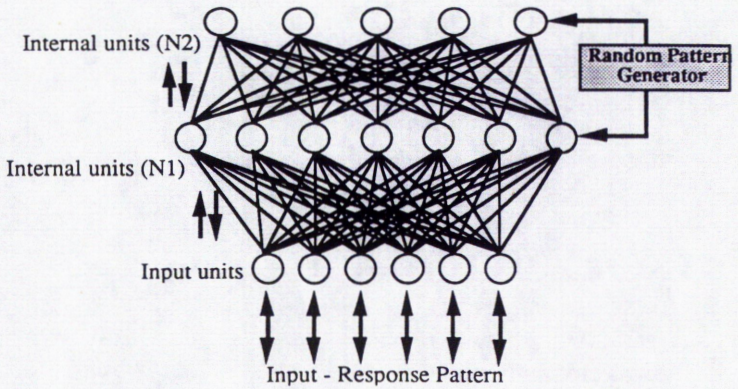


Exhibit 4.10 A three layer Bidirectional Associative Memory.

The approach proved to be fruitful, since simulation results demonstrated both

qualitative and quantitative improvement over the previous system. Tested on the same character set (Section 3.2) an average recall error of only 0.3% per character obtained while recalling the archetypes.

In Section 4.4.2 BAM's are viewed as a special case of Hopfield networks and an analysis that explains their superior performance, compared to standard Hopfield networks, is provided (Section 4.4.2). Some simulation results are presented in Section 4.4.3.

4.4.2 BAM vs. Hopfield Cluster

If the simulation of a Hopfield net is performed synchronously, every unit is updated on the basis of imperfect information about the state of its neighbourhood. This leads to unwanted interference effects that may spoil the convergence. One way of avoiding these effects is to use asynchronous simulation (Section 3.3.4). Another is to modify the net by splitting it into two components fully cross connected but with no internal connections. This approach can be implemented as a BAM system where the vector of the Hopfield cluster is the union of the input and the output vectors of a BAM. There is then no interference within a component, so that true simultaneous asynchronous update may be performed on a single component. This regime can be described as follows.

Let H be a Hopfield network with 2N units and 2N x 2N weight matrix $W = \Sigma hh^T$, where {h} is the set of patterns to be stored. Now consider H to be split as described into subsets R and S of N units each. We can then write each pattern h as h = r ++ s, where f is an R-vector and s is a R-vector. The weight matrix W can now be subdivided into four sub-matrices, thus:

	R	S
R	$\mathbf{r}\,\mathbf{r}^T$	$\mathbf{r}\,\mathbf{s}^T$
S	$\mathbf{s}\,\mathbf{r}^T$	$\mathbf{s}\,\mathbf{s}^T$

Table 1

The major diagonal quadrants represent the intraconnections within R and S. Since we are disallowing these, the weight matrix becomes W':

	R	S
R	$\mathbf{0}$	$\mathbf{r}\,\mathbf{s}^T$
S	$\mathbf{s}\,\mathbf{r}^T$	$\mathbf{0}$

Table 2.

But the two minor quadrants are precisely the weight matrices of a BAM with F and G as the two layers, while synchronous update of one component, say G, amounts to probing the BAM with F-vector f' and applying the North East quadrant of W' to recall a corresponding G-vector g'.

4.4.4 Simulation Results

Again the proposed system was tested for storing and recalling the character set of Section 3.2. Different sizes of networks were simulated. Exhibit 4.11 summarizes some of the results.

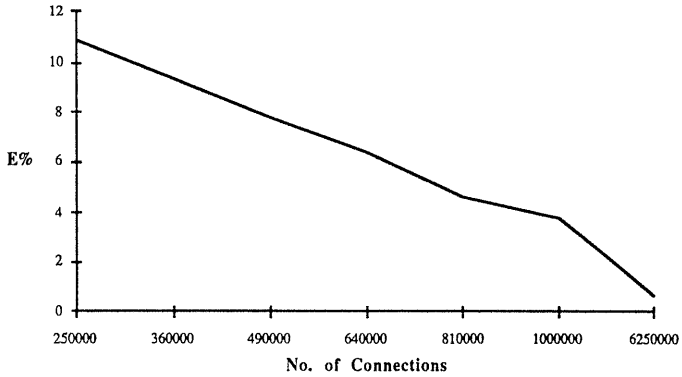


Exhibit 4.11 The average recall error per pattern (E%) decreases as the number of connections between the two internal layers increases.

Exhibit 4.11 shows the final recall error (E%) for different number of connections between the two internal layers of the proposed system. This recall error is significantly smaller than the smallest error observed for a Hopfield cluster of similar size (Section 4.2.5). For example, a three-layer BAM system with 10^5 connections between the two internal layers demonstrated a recall error of 3.9% while 16 out of 26 patterns were accurately recalled. The corresponding system with 10^5 connections in the Hopfield cluster had a recall error of 6.9% and it was only capable of recalling only 8 out of the 26 patterns.

An iterative technique implemented by computing the two internal vectors several times before computing the final output vector enhanced the results. When using 2500 units in every internal layer and iterating the vectors 12 times, 25 out of the 26 patterns were recalled accurately with only 3 hamming units recall error in the remaining pattern—an average recall error of 3% per pattern.

As it was the case with the system described in Section 4.2.1, computing resources were limited and larger size systems could not be simulated.

4.4.5 Extensions

The proposed system can be extended in two ways. First, it can be trivially extended to serve as multi-user pattern recall device in the same way the basic system was extended to serve as a SCAM system (see Section 4.3). This time the second internal layer of units (N2) will serve as the common storage place instead of a Hopfield cluster. Dynamic binding and synchronized recall can be implemented in the same way as in SCAM (Section 4.3.3).

Second, the proposed system can be extended to more than two internal layers. Although for pattern recall purposes two layers suffice, a multi-layer system can be used to represent a hierarchy of conceptual knowledge levels that cannot be otherwise implemented in distributed representation connectionist systems.

4.5 Summary

Three representatives of a new class of pattern recall devices based on randomized internal representations were described. The basic system combining a BAM and a Hopfield cluster into a single structure was presented. Simulation results demonstrated an improved performance, compared to the standard Hopfield network. The proposed system was capable of storing and recalling non-orthogonal archetypes. An extension of the basic system that serves as a SCAM system was suggested in order to utilize context. A similar three-layer version of the BAM system was then described. Simulations showed a further improvement in noise reduction during recall even though the system uses fewer connections than the equivalent system with a Hopfield cluster.

CHAPTER 5
COMPETITIVE LEARNING SYSTEMS

5.1 Introduction

In Chapters 3 and 4 auto-associative memory systems were investigated in detail and a new class of pattern recall devices based on Randomized Internal Representations was developed. Here a different class of connectionist systems, namely Competitive Learning models, is examined. We give an account of competitive learning systems with respect to their architecture, adaptation and evaluation schemes. Winner-take-all clusters, the heart of competitive learning systems, are described in detail. Three particular forms of unstable behaviour are examined and some simulation examples are presented. We then consider Adaptive Resonance Theory (ART) systems. A discussion on some of the limitations of ART systems leads to the development of Dynamic Adaptation Scheme (DAS) in Chapter 6.

5.2 Competitive Learning Models

5.2.1 Introduction

Here we give an account of the basic competitive learning system [009] with Emphasis is given on the system's operational limitations within a real-world environment. Competitive learning systems belong to the unsupervised learning category of connectionist systems. Adaptation is achieved without the use of a teacher and the system is expected to drive itself to the desired state. However in [009] four variations of the basic system are presented:

- **Auto-associator** where the system functions as an auto-associative memory.

- **Pattern associator** where the system functions as an associative memory.
- **Classification paradigm** where the system functions as a classification device.
- **Regularity (feature) detector** where there is no *a priori* set of categories and the system is expected to discover the categories of patterns in the environment according to their common features.

The auto-associator and the pattern associator variations are really extensions of the classification paradigm as explained in the following section. The regurality detector is of particular interest because it suits particularly well to the design constraints of Chapter 1. However, no clear distinction can be drawn between the classification and the regularity detector systems. The absence of control over how many categories the system develops during adaptation makes the two systems equivalent when operating in unsupervised mode. In supervised mode however, the two systems demonstrate different behaviour. Only when some form of supervision is available [004] can one distinguish between the two versions.

Competitive learning systems are standard connectionist systems in the sense that the adaptation and evaluation phase are separate. However the evaluation phase must always precede the adaptation phase. This suggests that a continuous adaptation regime can be implemented by competitive learning systems but we shall show in Section 2.4.6 that actually this is not feasible.

5.2.2 Architecture

A basic competitive learning system is shown in Exhibit 5.1. It consists of two groups of units, the input layer and the winner-take-all cluster. The input layer serves as an interface to the environment while the winner-take-all cluster is responsible for the classification.

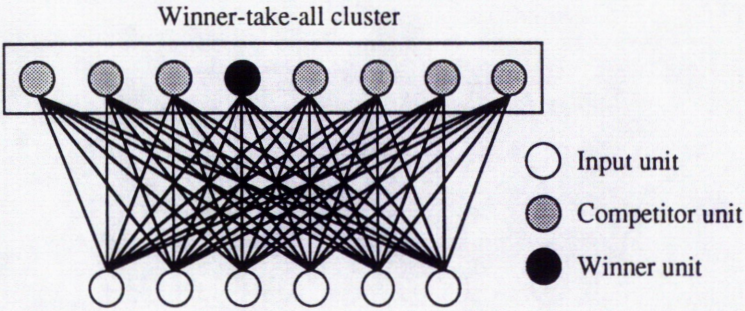


Exhibit 5.1 A standard competitive learning system architecture.

The input layer has an adequate number of units to match the size of the input patterns while the winner-take-all layer has a number of units equal to the number of categories represented in the system. Signals are allowed to travel in a feedforward manner from the input layer to the winner-take-all layer. Every input unit is connected to every output unit with a connection that has a modifiable weight value w_{ij}. Classification in the system is performed by finding a winner unit among the units in the winner-take-all cluster (see Section 5.2.4). An extended multi-layer version of the basic two layer system is presented in [009] and a similar architecture is described in Chapter 6.

Provided that there is only one winner unit for every category in the environment the basic architecture can be extended for the auto-associator and the pattern associator systems (see Exhibit 5.2).

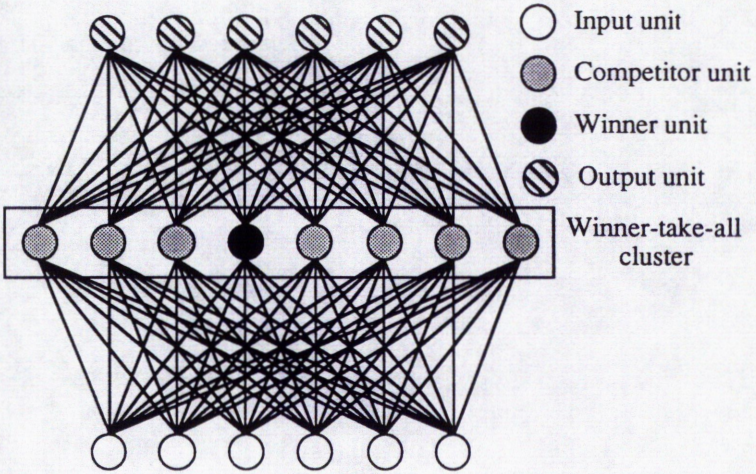


Exhibit 5.2 A competitive learning system extended to serve as auto-associator and pattern associator.

An extra layer of units, the output layer, is added to the system. Like the input layer, the output layer serves as an interface to the environment, where the pattern itself (auto-associator) or its associated pattern (pattern associator) is provided. The output layer has same number of units as the input layer for the auto-associator case or an adequate number of units to match the size of the associated pattern for the pattern associator case. For the auto-associator case, this architecture can be reduced to the basic model (see Exhibit 5.2). This is achieved simply by allowing signals to travel from the winner-take-all layer back to the input layer. Such use of a competitive learning system is very similar to bidirectional associative memories.

5.2.3 Evaluation

Consider a competitive learning system with **k** input units and **n** units in the

winner-take-all cluster. Assume that the system has completed its adaptation for a fixed set of patterns. A given pattern V is provided at the input units. The weight values on the connections between the input units and the winner-take-all cluster are always positive. Signals are allowed to travel from the input units to the winner-take-all cluster and every competitor unit computes its excitation E_i:

$$E_i = \sum_{j=1}^{k} w_{ij} V_j$$

Note that computing E_i for every competitor unit requires only locally available information, and that E_i is always positive because $w_{ij} > 0$ and $V_j = (0,1)$[1]. E_i indicates how strongly a competitor unit responds to pattern V. The competitor unit with the highest excitation becomes the winner unit and it is said that pattern V is classified into the corresponding category. The evaluation of a winner-take-all cluster is described separately in Section 5.2.4.

To construct an associator, the system is extended with the addition of an output layer and the corresponding connections and weight values w'_{ij} between the competitor units and the output units. Given the winner unit j will set $A_j = 1$, the i^{th} unit in the output layer computes its excitation O_i as:

$$O_i = w'_{ij} A_j$$

As mentioned earlier, for the case of the auto-associator system there is no need for an output layer because the input units can be used to serve as output units. In this case, the winner unit transmits a signal towards the input units and the response R_i at the i^{th} input unit is:

$$R_i = w_{ji} A_j$$

---

1 $E_i = 0$ in the extreme case where $V_j = 0$ for every j = 1..n (all-off pattern). However the all-off pattern is interpreted as total absence of input and it does not harm the analysis that follows. In this case there is no winner-unit.

We now examine the evaluation of the winner-take-all cluster in detail.

5.2.4 Winner-take-all Clusters

Perceptrons, Embedding Field theory and On-center Off-surround systems (see Chapter 2) share a common feature which is also central to competitive learning models. Pattern classification is performed by means of one active *winner unit* that represents a particular class of patterns. The winner unit is produced by a winner-take-all cluster. A winner-take-all cluster contains a group of *competitor* units which compete with each other until only the unit which receives the highest excitation from the input pattern remains active. This unit is the winner unit.

The simplest way to find the winner unit is to implement a sorting algorithm. There are two main objections to this approach:

(1) A sorting algorithm requires an external device or additional software, which violates the formalism of connectionist systems.

(2) The number of classes in an environment can be very large. Provided that a sorting algorithm serializes the computation performed by the system, an increase in response time is likely.

A connectionist winner-take-all cluster uses fixed weight inhibitory connections between the competitor units (Exhibit 5.3). The idea is that if all the competitor units inhibit each other, the unit with the highest excitation level will remain active.

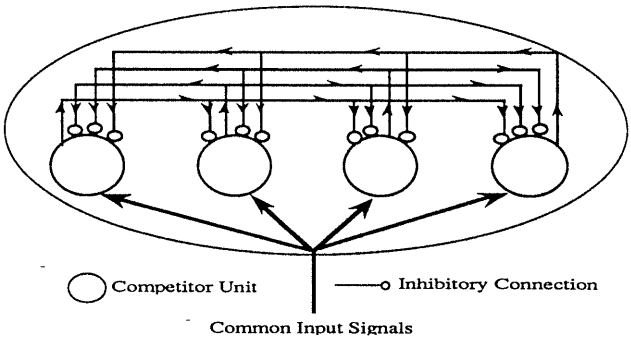


Exhibit 5.3 A winner take-all cluster with mutually inhibited competitor units.

5.2.5 Adaptation

The adaptation process described in this section applies to both the classification paradigm and the regularity detector versions of the basic system.

Consider a system with k input units and n competitor units. Assume N patterns of size k for which the system is expected to develop a number of categories. In the case where every pattern is an archetype, the system should allocate a different winner unit for every pattern. In the case where the patterns are a sample of all the possible environmental patterns the system should develop a number of categories that correspond to the categories present in the environment.

Assume that a pattern V is provided at the input units. As in the evaluation phase V is translated into a binary (1,0) vector . For every competitor unit i the respective weight values are taken to satisfy:

$$\Sigma w_{ij} = 1 \qquad (5)$$

The competitor units in the winner-take-all cluster compute their activation A_i as described in Section 5.2.3 and the winner-take-all cluster finds a winner

unit as described in Section 5.2.4. The weights in the system are modified as follows:

$$\Delta w_{ij} = \begin{cases} 0 & \text{if } i \neq \text{winner unit} \\ \varepsilon \left(\frac{V_j}{\alpha} - w_{ij} \right) & \text{if } i = \text{winner unit} \end{cases}$$

where $\varepsilon > 0$ and $\alpha = \sum_{j=1}^{n} V_j$ (= no. of active input units)

(6)

This adaptation rule was proposed by Von der Malsburg in [027]. The rule, also referred in [042] as Weber's law, has a major difference from the other rules examined so far: it violates Hebb's dictum that weight modification on a particular connection should only take into consideration the activation of the units at either end of the connection. A result of this is the need for explicit synchronization of the system. For the weights to change (6) requires knowledge of the number of active input units. In DAS (see Chapter 6) Hebb's dictum is maintained in order to avoid such synchronization problems. The adaptation rule in (6) requires that patterns are presented to the network repeatedly until the system develops a number of stable categories (slow learning). A *stable* category is the one whose winner unit and membership remain fixed over time.

Competitive learning systems do not provide any control over the number of categories the system will develop, so are inappropriate for applications where a fixed number of categories is required.

5.2.6 Stability in Competitive Learning Systems

In [043][026][002], several cases where a competitive learning system demonstrated unstable behaviour were shown. Summarizing these results, three forms of unstable behaviour can be identified:

- Unstable category representations
- Oscillatory behaviour
- Non-deterministic categorization of the data set

Unstable category representations Consider a competitive learning network and assume a fixed set of training data. Suppose that we present the system with every pattern in the data set and allow for adaptation as described in Section 5.2.5. It is expected that every pattern will correspond to a winner unit, while some pattern may correspond to the same winner unit. Suppose that pattern A from the training set is presented to the system after adaptation is completed. Also assume that when pattern A was presented during adaptation, competitor unit 1 was allocated as the winner unit. The system should now respond with competitor unit 1 as the winner unit. It is shown in [043] that this is not always the case; the system may allocate a different winner unit. To ensure that there is a fixed winner unit (stable representation), several consecutive presentations of the training pattern are required during adaptation. Consider the set of patterns shown in Exhibit 5.4. The corresponding competitive learning system with 36 input units and 12 competitor units required 20 consecutive presentations until a stable representation was obtained for every pattern. In Chapter 1 it was mentioned that competitive learning systems support an incremental adaptation rule. This is so, but to guarantee stable representations the system has to be repeatedly presented with the complete data set.

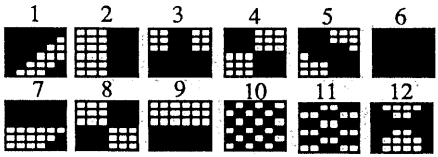


Exhibit 5.4 A set of 12 bitmap patterns. Every black pixel corresponds to an active input unit (1) while every white pixel corresponds to an inactive (0) input unit.

Furthermore, stability is guaranteed only for a given set of patterns. If the system is stable and is required to adapt to a new pattern the complete set of patterns, including the new pattern, has to be presented and the system has to adapt from scratch. This particular stability problem prevents competitive learning systems from operating under a continuous adaptation regime.

Oscillatory behaviour It is implied above that repeated consecutive presentation of a fixed set of patterns will eventually stabilize the system. It is shown in [043] that even this, is not always the case. The system may actually oscillate amongst different allocations of winner units to patterns. Consider the data set given in Exhibit 5.4. Setting ε to values less than 0.1 or greater than 0.9 the system was unable to stabilize.

Non-deterministic categorization of the data set Consider an environment with a fixed number of patterns. Assume that the system is able to stabilize. The sequence in which patterns are presented to the system has an effect on the categories developed in the system. According to Grossberg [043]:
"*Learning can also become unstable due to simple changes in an input environment. Changes in the probabilities of inputs or in the deterministic sequencing of input can readily wash away prior learning.*"

Consider the data set shown in Exhibit 5.4 and the corresponding network with 36 input unit and 12 competitor units. The patterns were presented to the network in the order given in Exhibit 5.4. For $\varepsilon = 0.4$ the system developed 6 stable categories after 20 consecutive presentations of the complete data set. Because there is no control over the number of categories the system develops, the final classification of the patterns shown in Exhibit 5.5 does not bear any significance other than the fact that the system discovered 6 categories in the data set.

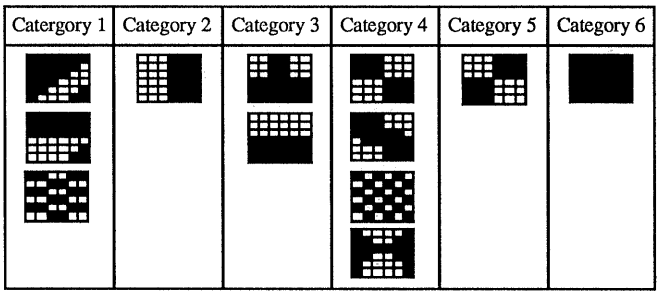


Exhibit 5.5 The pattern of Exhibit 5.4 classified in 6 categories by a competitive learning system.

Changing the order patterns were presented to the system resulted in a different classification as shown in Exhibits 5.6 & 5.7.

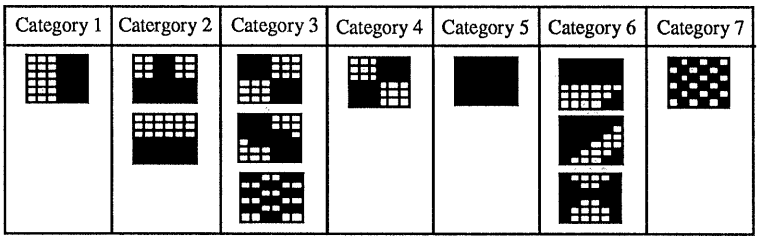


Exhibit 5.6 Changing the presentation order of patterns resulted to a different classification.

The same system with $\varepsilon = 0.4$ as before, but this time switching the order patterns 1 and 6 were presented to the system resulted to a different classification (Exhibit 5.6). This time the system developed 7 categories. Changing the initial order of patterns by switching patterns 3 and 8 also resulted to a different classification. This time the system developed 6 categories (Exhibit 5.7).

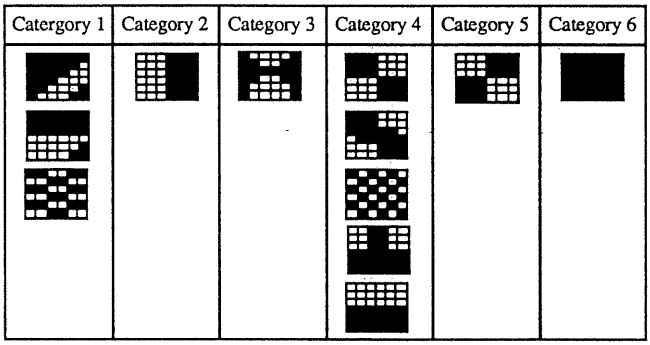


Exhibit 5.7 Again changing the sequence patterns were presented to the system resulted to yet another classification.

One may argue that this behaviour is expected in the sense that the is no relation between the patterns. However we argue that this should have resulted in the system classifying the patterns of Exhibit 5.4 into one of the two default cases:

- All the patterns classified into one category
- Every pattern forms a category of its own.

In the case where the system makes a classification other than the default cases, this classification should be stable regardless of the sequence the patterns are presented to the system.

This particular form of unstable behaviour is very important from an engineering point of view. For a real-world environment the order in which patterns appear to the system cannot be controlled. The development of the same categories of patterns, regardless of their order of appearance, is a minimum requirement for a system that attempts to obtain a consistent representation of the environment.

5.3 Adaptive Resonance Theory (ART)

5.3.1 Introduction

Adaptive Resonance Theory (ART) introduced in [002], aimed to produce a self-organized pattern recognition system that addressed the stability plasticity dilemma. Under the ART theoretical framework, ART1 systems were developed and later, ART2 [008] and ART3 [053] systems were suggested as extensions of ART1. ART2 systems are a trivial extension while ART3 systems are a very recent development which cannot be taken into account in this thesis.

5.3.2 Architecture

An ART1 system consists of three separate subsystems:

- The *attentional* subsystem which is responsible for the classification of the input patterns.
- The *orienting* subsystem which controls the plasticity of the system.
- The *gain* control subsystem which synchronizes part of the attentional subsystem

An example of an ART1 network is shown in Exhibit 5.8.

The attentional control subsystem consists of two layers of units F1 and F2 with every unit in F1 is connected to every unit in F2 (bottom-up connection). There is also a separate connection from every F2 unit to every F1 unit (top-

down connection) (not shown in Exhibit 5.8). Every connection in the attentional subsystem has a weight value and different adaptation rules apply to bottom-up and top-down connections respectively (see Section 5.3.4).

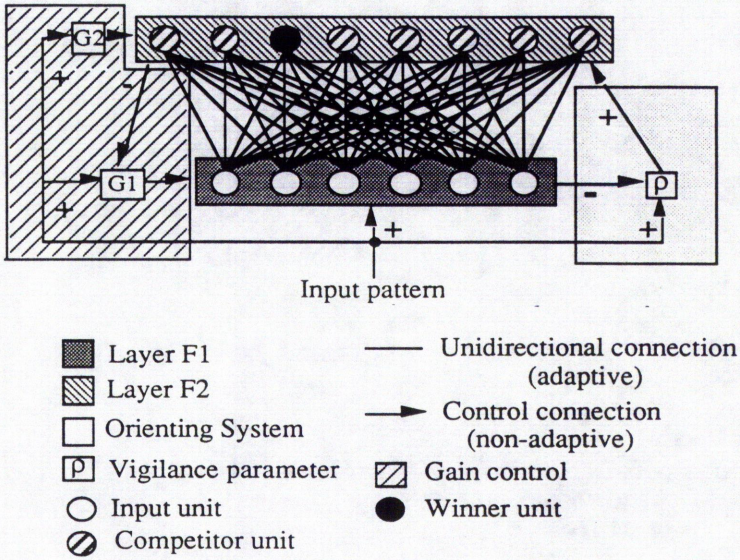


Exhibit 5.8 An ART1 system

The activation of units in F1 and F2 represents the short term memory (STM) of the system while the weights on the adaptive connections are the long term memory (LTM). Layer F2 is a winner-take-all cluster (see Section 5.2.4) and is responsible for the classification of the input pattern.

The gain control subsystem consists of two gain control units G1 and G2. Gain control unit G1 (Exhibit 5.8) has two input sources and one output. The first input source to G1 comes from an inhibitory connection with F2 while the second from an excitatory connection with the input pattern. In [008] there is a third source (intermodal inhibition) which is not considered here. The output

from the gain control unit G1 is connected to every unit in F1. An inhibitory signal from G1 disables F1 units from becoming active. Gain control unit G2 has one input and one output. G2 receives input from the input pattern and its output is connected to every unit in F2. An excitatory signal from G2 enables F2 units to become active.

The orienting subsystem consists of a control unit where the *vigilance* parameter is stored. This control unit has two inputs and one output. It receives inhibition from all the units in F1 and excitation from the input pattern. Its output, also called a STM reset wave, is connected to all the units in F2. An inhibitory signal from the orienting subsystem resets all F2 units and disables the active F2 units from becoming active.

5.3.3 Evaluation

Assume that an ART1 system has already undergone some LTM adaptation and an input pattern is provided to the system. A signal is send to G1 and G2 thus enabling units in F1 and F2 to become active. The input pattern activates the units in F1 and signals are sent to F2 using the bottom-up connections in the attentional subsystem thus activating the units in F2. Since F2 is a winner-take-all cluster, a winner unit W_1 appears, representing the category into which the input pattern is classified. The winner unit sends signals to the units in F1 using the top-down connections. An inhibitory signal is sent from the winner unit in F2 to G1 via the corresponding inhibitory connection in the gain control subsystem.

If the input pattern is still present, the inhibitory effect of F2 on G1 is eliminated by the excitatory effect of the input pattern and the units in F1 are allowed to be active.

If the input pattern is not present, G1 sends an inhibitory signal to F1 and

shuts off all its units and subsequently shuts off the system. In this sense G1 guarantees that activation in F2 alone cannot drive the system; the presence of an input pattern is necessary. This condition is reinforced by G2 which guarantees that a unit in F2 cannot be active unless an input pattern is present. For G1 and G2 to function effectively fine tuning of the system is required. For example the inhibitory signal from F2 to G1 has to be smaller than the excitatory signal from the input pattern.

Assume that a winner unit is found in F2 and the input pattern is still available. A pattern of activity is produced in F1 by the winner unit via the top-down connections. If the classification is correct the pattern at F1 is taken to be the archetype pattern of the corresponding category. In fact the use of top-down connections enables an ART1 system to function as a pattern recall device. An inhibitory signal is sent to the orienting subsystem control unit from the units in F1 while an excitatory signal is sent from the input pattern. This control unit tests whether the classification is correct. It compares the pattern at F1 with the input pattern and determines correctness using the vigilance parameter ρ. The vigilance parameter takes a value between 0 and 1. For example, if there are n units in F1, from which m units are different from the input pattern, then for $\rho n \geq m$ the classification is considered correct and the control unit sends no output signal. In this case the system is said to be *stable.*

If $\rho n < m$ then the control unit becomes active and a reset signal is sent to all the units in F2. In this case the input pattern is considered a *novel* pattern and the system will develop a new category (winner unit). A new winner unit is guaranteed because the reset signal prohibits the previous winner unit from competing any further for the particular input pattern. Since there is no activity in F2, the input pattern causes further activity in F1 and subsequently enables F2 to find another winner unit. In this case the system is said to be *plastic.* If

the new-winner unit still allows the control unit in the orienting subsystem to emit a reset signal to F2 then a new winner unit has to be found. In the case where all the units in F2 become suppressed the memory capacity of the network is exhausted.

5.3.4 Adaptation

An ART system may adapt whether it is stable or plastic. For a weight value to change it is required that some unit in F2 is active. The weight values of the top-down connections change as follows:

- If a unit in F1 is active, the weight value is increased, approaching 1.
- If a unit in F1 is inactive, the weight value is decreased, eventually approaching 0.

The weight values of the bottom-up connections are modified according to the standard competitive learning regime of Section 5.2.5. Provided that an ART system is capable of recognizing whether an input pattern belongs to a new category or not, a continuous adaptation regime can be implemented.

5.3.5 Limitations of ART Systems

Because an ART1 system has an embedded competitive learning system it can suffer from the same instability. D.G.Stork in [042] claims that this is not the case. We quote from [042]:

"*..because learning requires the F2 neuron to be active the categories associated with inactive neurons are not degraded*"

This argument is incorrect because, as shown for competitive learning systems, a winner unit currently representing one category may after some time represent another. This is also confirmed by Stork himself in the same paper:

" A particular neuron in F2 might at one instant represent category A, and at a later time (after more learning) it might represent category B... Such instabilities do not affect some neural network applications because an operator carefully controls the patterns presented (e.g., by randomizing their order, by presenting each pattern until the network has fully learned the pattern, or by presenting unusual patterns for a sufficient time so that they are learned) "

Clearly a teacher is required to ensure stability in the network. If no *a priori* information about the environment is available, then the methods suggested in the quotation above for stabilizing the system are inappropriate. ART systems were not intended to function within a real-world environment and therefore they are considered inadequate for such use.

Their main advantage over competitive learning systems is that ART1 systems provide control over the number of categories developed during adaptation. This is to be achieved by appropriate setting of the vigilance parameter ρ in the orienting subsystem. However, this control is limited, as the following example shows. Consider the two bitmap representations of capital letters C and G in Exhibit 5.9. Assume that an ART1 system has to develop a category for each capital letter of the English alphabet bitmap representation shown in exhibit 3.1. Exhibit 5.9 shows letters C and G.

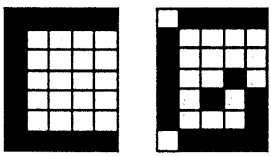


Exhibit 5.9 Bitmap representation of capital letters C and G.

In Exhibit 5.10 the same noise is added to C and G to produce noisy versions C' and G'. Notice that C' = E.

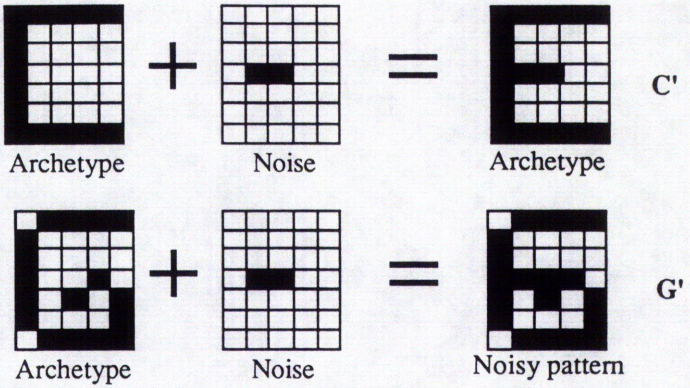


Exhibit 5.10 The same noise is added to the patterns shown in exhibit 5.9.

By setting $\rho \leq 0.95$ G' will be classified as G. The same ρ setting will therefore classify E as a noisy version of C. Conversely setting ρ in such a way that the system considers letter E to be a novel pattern also causes G' to be a novel pattern. Stork [042] has a similar example. Clearly no setting ρ enables the system to classify C' as the novel pattern, while classifying G' as G. However, [049] suggests that by suitably varying ρ such cases may be resolved. In general, one would expect that cases like this cannot be resolved by any self-organizing system, and that a supervisor is required. Even then the system must provide the appropriate localized control mechanism to enable the supervisor to exercise control. ART systems only provide a global control mechanism in the orienting subsystem.

5.4 Summary

In this chapter a detailed account of winner-take-all clusters was provided and simulation problems were identified. Competitive learning systems were discussed in detail and some examples of unstable behaviour were presented.

This was followed by a brief description of ART1 systems and we explained the reasons why ART1 systems are unable to perform successfully within a real-world environment.

CHAPTER 6
DYNAMIC ADAPTATION SCHEME (DAS)

6.1 Introduction

The Dynamic Adaptation Scheme (DAS) was developed around the idea of a winner unit and incorporates some features of Competitive Learning (CL) and ART systems. Bearing in mind the operational limitations of CL and ART systems as well as the design constraints of Chapter 1, DAS had to meet the following criteria:

- *Incremental* adaptation, to enable the system to operate under a continuous adaptation regime.
- *Stable* representation of classes, to ensure the incrementality of the adaptation rule and allows for real-time response
- *Stability-plasticity* control, to enable the system to distinguish between novel and noisy input.
- *Resources Utilization*, to allow the system to optimize its resources.
- *Self-organization.*
- *Dynamic Allocation of computing resources.* DAS belongs to the category of *constructive* algorithms [055] [029] because it modifies its structure according to the environmental input (Section 6.5.2)

The main differences between DAS systems and CL systems lie in:
- The adaptation rule
- Threshold controlled competitor units.

We discuss each of these in turn.

6.2 Adaptation Rule

Consider a network with an input layer and a winner-take-all cluster, with n competitor units. Let A be the *archetype matrix* with A_{ij} to be the j^{th} element of the i^{th} archetype, W to be the weight matrix (initially all $W_{ij} = W > 0$), and

y_i to be the excitation of competitor unit i.

The weights between the winner unit i and the input units are modified as follows:

Adaptation Rule

$$\Delta W_{ij} = \delta\,(2A_{ij} - 1) \text{ and}$$

$$W_{ij} = W + \Delta W_{ij}$$

$$\text{where } \delta < W$$

We shall:

(a) Find the condition for a competitor unit to be the winner unit

(b) Show that the proposed adaptation rule guarantees fixed representations

(a) Assume that competitor unit p is allocated as the winner unit for archetype A_p. The corresponding weights W_{pj} are modified modified as follows:

$$W_{pj} = W + \delta\,(2A_{pj} - 1)$$

When the system is probed with pattern x, the excitation y_p of competitor unit p is:

$$y_p = \sum_j W_{pj}\, x_j$$

$$y_p = \sum_j (W + \Delta W_{pj})\, x_j$$

$$y_p = \sum_j (W + \delta\,(2A_{pj} - 1))\, x_j$$

The excitation y_q of competitor unit q, which has not yet been nominated as a winner unit, is:

$$y_q = \sum_j W\, x_j$$

Therefore:

$$y_p - y_q = \sum_j (W + \delta\,(2A_{pj} - 1))\, x_j - \sum_j W\, x_j = \sum_j 2\delta A_{pj} x_j - \sum_j \delta x_j$$

$$y_p - y_q = \delta(\sum_j 2A_{pj} x_j - \sum_j x_j)$$

Let $\lambda = \sum A_{pj} x_j$ and $\kappa = \sum x_j$

Note that:

$$A_{pj} \neq x_j \rightarrow A_{pj}\, x_j = 0$$

$$A_{pj} = x_j \text{ either } \frac{A_{pj} = 0 \;\&\; x_j = 0 \rightarrow A_{pj}x_j = 0}{A_{pj} = 1 \;\&\; x_j = 1 \rightarrow A_{pj}x_j = 1}$$

Therefore, λ is the number of 1's patterns A_p and x have in common, and κ is the total number of 1's in x. Therefore:

$$y_p - y_q = 2\lambda - \kappa \text{ and either } \frac{\kappa > 2\lambda \text{ q is the winner unit}}{\kappa < 2\lambda \text{ p is the winner unit}} \quad (1)$$

If p is the winner unit x is classified as a noisy version of A_p, while if q is the winner unit x forms a new category and becomes archetype A_q.

(b) Assume that competitor unit q is the winner unit for probe x ($=A_q$). The weights W_{qj} are modified as follows:

$$W_{qj} = W + \delta\,(2A_{qj} - 1)$$

Now consider the case where the system is probed with A_p. The excitation of competitor unit p is:

$$y_p = \sum_j (W + \delta\,(2A_{pj} - 1))\, A_{pj}$$

While the excitation of competitor unit q is:

$$y_q = \sum_j (W + \delta\,(2A_{qj} - 1))\, A_{pj}$$

Therefore:

$$y_p - y_q = 2\delta(\sum_j A_{pj}A_{pj} - \sum_j A_{qj}A_{pj})$$

Note that:

$$A_{qj} \neq A_{pj} \rightarrow A_{qj}A_{pj} = 0$$

Assuming that $A_p \neq A_q$ in at least one element:

$$y_p - y_q > 0$$

which means that competitor units p is the winner unit, and therefore the representation of archetype A_p is stable.

6.3 Subsets, Supersets and the All-on pattern

Consider the patterns of Exhibit 6.1. Note that pattern A is a subset of pattern B and pattern Ω is a superset of all possible patterns. We will call pattern Ω the *all-on* pattern.

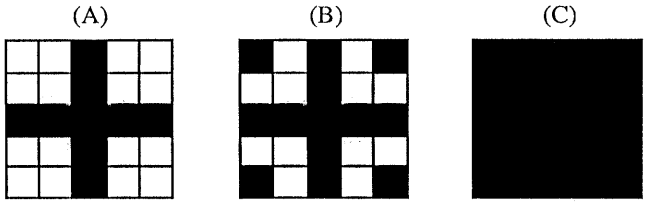


Exhibit 6.1 Pattern Ω ⊃ pattern B ⊃ pattern A.

In the general case, every pattern x which is a subset of archetype pattern A_p has:

$$\kappa = \lambda$$

According to (1) x will be classified as a noisy version of A_p. This effect is not hazardous and it can be considered as an expression of the system's fault-tolerance levels. However the effect becomes hazardous if A_P is the all-on pattern (Ω) because every pattern x is a subset of Ω and therefore:

$$\kappa = \lambda_\Omega \ \forall \ x$$

It follows from (1) that once Ω is in store, every other pattern will classified as a noisy version of Ω. Although Ω is an artificial pattern not normally found in natural data, it may arise in a real-world situation from cases such as hardware faults, floating inputs etc.

6.4 Modifiable Thresholds

The activation of a competitor unit in response to a probe denotes how close the probe is to the archetype of the corresponding class. Furthermore, it is expected that a competitor unit will reach maximum activation when the archetype of the corresponding class is used as a probe. Based on these observations we thought that a competitor unit may be capable of deciding whether the probe should be classified into the corresponding class independently of the other competitor units. This can be achieved be means of storing the maximum activation of the particular competitor unit and compare it to the activation caused by a particular probe.

We also noticed that although most connectionist systems provide a threshold mechanism for their units, CL and ART systems did not support such a mechanism. So we thought of incorporating a threshold mechanism into a unit with the purpose of storing the maximum activation of the unit. We will call this the *modifiable threshold* (MT) mechanism.

A simple implementation of MTs requires that the threshold value of a competitor unit is set to be a percentage of its maximum activation. A competitor unit is allowed to become a winner unit only when the activation exceeds the threshold value. In the general case there is only one competitor unit that becomes active. Therefore there is no need for a winner-take-all cluster as defined in Chapter 5; the active competitor unit automatically becomes the winner unit. The case where more than one unit is active is discussed in Section 6.7.

6.5.Dynamic Adaptation Scheme (DAS)

6.5.1 The Scheme

DAS consists of a set of four rules. Let $T_i > 0$ be the threshold and y_i to be the activation of competitor unit i. Let W to be the weight matrix and x to be a probe pattern:

RULE 1: A competitor unit becomes a winner unit only when its activation level exceeds its threshold value.

$$y_i \geq T_i$$

RULE 2: Using the adaptation rule of Section 6.2, weight values are modified every time a competitor unit becomes a winner unit as follows:

$$\Delta W_{ij} = \delta' (2x_j - 1)$$

$$\text{where } \delta' = ((y_i - T_i) / y_i)\, \delta,$$

$$\text{and } 1 \geq ((y_i - T_i) / yi) \geq 0$$

RULE 3: If a competitor unit is a winner unit, then its threshold value is updated as follows:

$$T_i = \varepsilon\, y_i$$

$$\text{where } 1 \geq \varepsilon \geq 0$$

RULE 4: The system is always provided with a *free competitor* unit. Every time the free competitor unit becomes the winner unit, the system's structure is modified by allocating a new free competitor unit to the system. Note that initially the system comprises only one free-competitor unit.

6.5.2 Evaluation

We shall demonstrate how a system operates under a DAS regime and the way the design criteria of Section 6.1 are satisfied. Consider a system with **k** input units and one free-competitor unit (Exhibit 6.2 (B)). Pattern X is clamped at the input units. The threshold value of the free-competitor unit is initially set to zero. The activation level of the free-competitor unit is calculated as shown in Exhibit 6.2 (b).

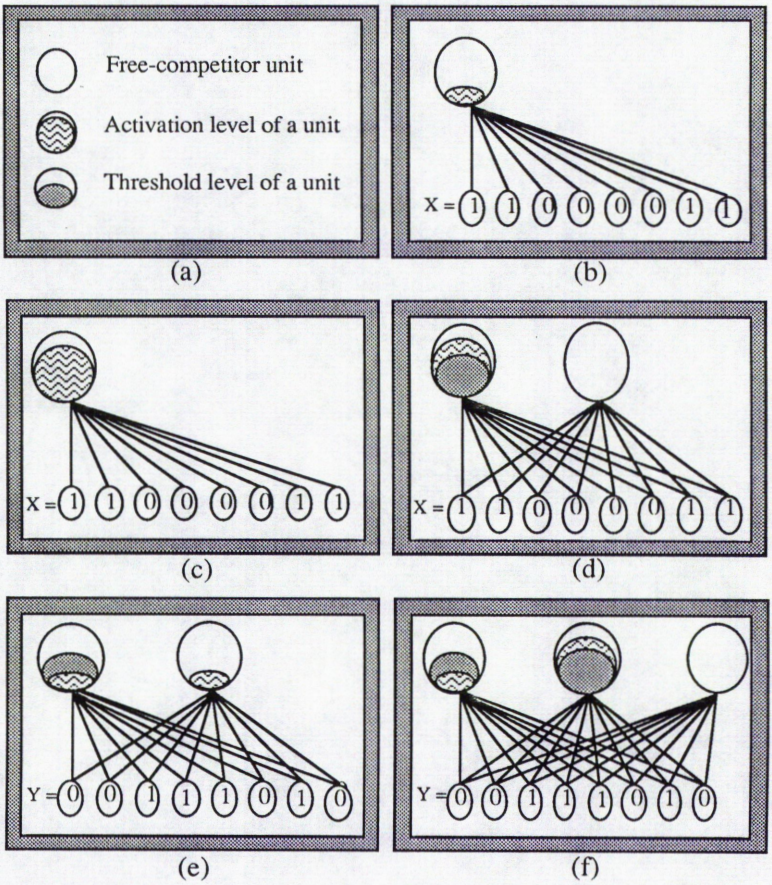


Exhibit 6.2 (a) -(f). A DAS system at different stages of operation.

In order for X to cause some initial activation to the free-competitor unit, the weight values are initially set to a positive value very close to zero. Provided that X has at least one element on, the activation of the output unit is going to be positive and therefore greater than its threshold value. According to Rule 1 the free-competitor unit becomes active and is nominated as the winner unit. The weight values on the connections between the winner unit and the input units are modified according to Rule 2. Note that in the case where a free-competitor unit is nominated as the winner unit we have:

$$\delta = \delta'$$

The new activation of the winner unit is computed (Exhibit 6.2 (c)) and the new threshold value is computed and stored (Exhibit 6.2 (d)). Another free competitor unit is made available to the system and pattern Y is presented to the system (Exhibit 6.2 (e)). The competitor unit and the free competitor unit compute their respective activation levels. If Rule 1 is valid for the competitor unit then Y is classified as a noisy version of X.

We now find the condition for a competitor unit p to become a winner unit for a given probe pattern x, and we discuss the effects of ε on the behaviour of the system. We assume that competitor unit p has already been nominated as the winner unit for archetype A_p at which point T_p was set to:

$$T_p = \varepsilon \sum_j (W_{pj} + \delta'(2A_{pj} - 1))\, A_{pj}$$

The current activation of competitor unit p due to x is:

$$y_p = \sum_j (W_{pj} + \delta'(2A_{pj} - 1))\, x_j$$

For x to be classified as a noisy version of A_p that is for p to become the winner unit Rule 1 requires:

$$y_p - T_p \geq 0$$

Therefore:

$$\sum_j (W_{pj} + \delta(2A_{pj} - 1))\, x_j - \varepsilon \sum_j (W_{pj} + \delta(2A_{pj} - 1))\, A_{pj} \geq 0$$

Initially all $W_{pj} \cong 0$, therefore:

$$\sum_j (\delta(2A_{pj}x_j - x_j)) - \varepsilon \sum_j (\delta(2A_{pj}\, A_{pj} - A_{pj}) \geq 0 \qquad (2)$$

If λ is the number of 1's common in both A_p and x, α is the total number of 1's in A_{pj} and κ is the total number of 1's in x_j. From (2):

$$2\lambda - \kappa - \varepsilon\alpha \geq 0 \qquad (3)$$

$$\frac{2\lambda - \kappa}{\alpha} \geq \varepsilon \qquad (4)$$

Consider the case where $\varepsilon = 1$. From (3):

$$2\lambda - \kappa - \alpha \geq 0 \qquad (5)$$

But:

$$\lambda \leq \mu \;\&\; \lambda \leq \alpha$$

Therefore (5) can only be valid for:

$$\lambda = \kappa = \alpha \;\text{->}\; A_p = x$$

which shows that setting $\varepsilon = 1$ forces the system to consider every probe as a novel archetype and also shows that a DAS system can be tuned in such a way that it does not suffer from the all-on pattern effect (Section 6.3). In general, as ε increases the number of new categories increases but the fault tolerance of the system decreases. Given a specific set of archetypes (4) can be used to calculate the value of ε for which every archetype is allocated a different winner unit while the system maintains an optimal fault-tolerance capability. Another consequence of (4) is a lower limit to the fault tolerance of the system. This is because we require that $2\lambda - \kappa > 0$.

However, as was the case for ART systems (Section 5.3.5), using the same ε for every winner unit can sometimes lead to noisy patterns being considered as archetypes. The difference in DAS is that the stability-plasticity control is localized at the unit level, thus allowing for fine tuning of ε. Of course fine

tuning requires supervision which violates both the self-organization and real-world constraints.

Alternatively, one may view ε as an environmental parameter, in which case supervision becomes an implicit feature of the environment. For example, ε can be used to express how strongly the environment "believes" that a specific probe is a novel archetype.

If (4) is not true the free competitor unit is nominated as the winner unit for pattern Y. Weight modifications occur (Rule 2), the new threshold value is computed (Rule 3) and another free competitor unit is provided to the system (Rule 4) (Exhibit 6.2 (f)).

Assume that another pattern (Z) is clamped on the input units and the first competitor unit is nominated as the winner unit. According to Rule 2 the weight values will change but this time δ' will be significantly smaller than δ. If pattern Z persists over a long period of time, Rule 4 ensures that it will be considered as the archetype of the class while pattern X will be treated as a noisy version of Z. The *first impression* (pattern X) of a particular class is the one that has the most significant effect on the weight values while any changes within the class have a marginal effect. However, provided that these changes persist, changes in the weight values will become significant, thus allowing the system to adapt dynamically. This particular feature of DAS enables the system to be plastic at all times. In this way a continuous adaptation regime is fully implemented. This operation enables DAS systems to adapt according to both *global* and *local* changes in the environment:

- The presence of a new class of patterns (global) is encoded dynamically by means of the free-competitor unit becoming a winner unit.
- Changes within an existent class of patterns (local) are encoded by allowing changes on the weight values every time a competitor unit becomes a winner unit.

Resource Utilization can be achieved by using a decay mechanism for both the threshold values and the weights in the system. This decay mechanism forces weight and threshold values to decrease in time except for the case where the corresponding competitor unit is active. The decay mechanism can help the system to optimize its resources as follows. Consider the case where archetype Y or a noisy version of it does not appear for a long period of time. The decay mechanism forces the threshold and the weight values of the second competitor unit to decrease; eventually reaching zero. When this happens the second competitor unit becomes effectively a free-competitor unit. Provided that a new class of pattern appears, the second competitor unit will gradually change its weights to represent the new class of patterns. In this way re-use of resources is achieved. Note that a decay mechanism is optional and that it can be omitted if sufficient resources are available.

DAS vs. ART From the above it becomes clear that DAS satisfies all the design constraints of Section 6.1. It provides the means for dynamic allocation of computing resources, it allows for the system to adapt continuously according to global and local changes in the environment, it guarantees stable representation of classes, it provides a stability-plasticity control mechanism integrated in the system, it provides a realistic way for re-use of the computing and finally all the above are supported under a self-organization regime.

Specifically DAS has the following advantages compared to ART systems:

- Simple architecture that only requires for a modifiable control mechanism built in the competitor units.
- Localized stability-plasticity control.

- One step adaptation-evaluation.
- Dynamic allocation of computer resources.
- Automated re-use of resources.
- Localized (Hebbian) adaptation rule which ensures stable representations.

Unlike ART systems, DAS assesses the novelty of a probe without recalling the archetype. Like any other classification device, the basic DAS system can be trivially extended to function as a pattern recall device. In the next section we provide some simulation results obtained using DAS to store the patterns of Section 3.2 as well as the patterns of Exhibit 5.4.

6.6 Simulation Results

As we have already mentioned, the value of ε decides the number of categories and the fault tolerance levels of the system. Exhibit 6.3 is a diagrammatic representation of the effects of changing ε.

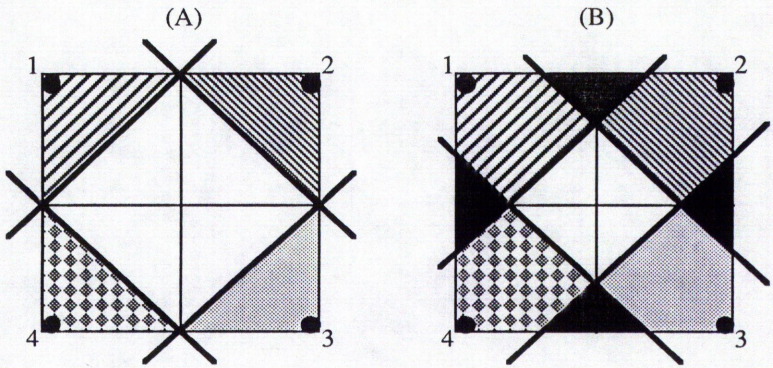


Exhibit 6.3 Changing the value of ε modifies the fault-tolerance of the system.

The corners (1,2,3,4) of the square correspond to the archetypes of a different category. The four lines across the square define the *area of attraction* for each

corner in the sense that probes within the area are classified as the pattern of corresponding corner. Each line is defined by the threshold value of the corresponding competitor unit. The symmetry of the areas is due to the fact the ε is the same for all the competitor units. Exhibit 6.3 (B) shows how the areas of attraction change when ε is decreased. The fault-tolerance of the system is increased but probes which lie within the black areas of the square are classified as noisy versions of both the adjacent corners.

We experimented using different ε values, for the patterns of Exhibit 5.4. Simulation showed that setting $\varepsilon > 0.70$ causes to every pattern considered as an archetype of a different category. Exhibit 6.4 shows the classification of the patterns for some $\varepsilon \leq 0.70$.

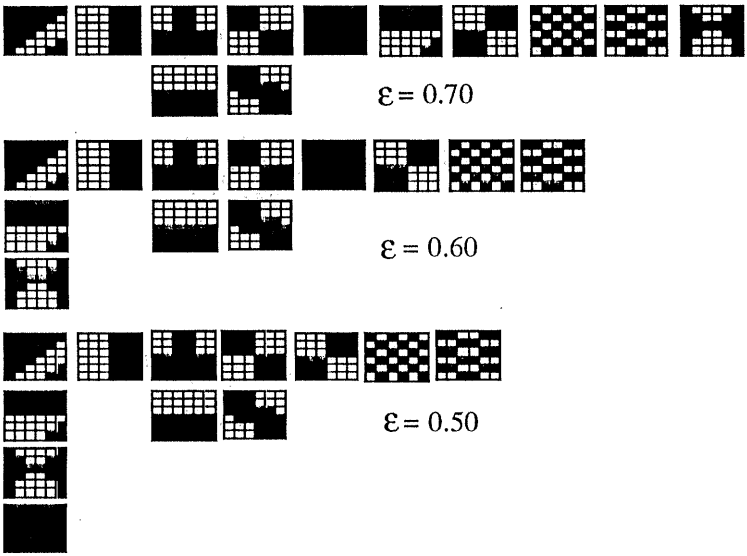


Exhibit 6.4 Classification of the patterns in Exhibit 5.4 for different e values. Every column corresponds to a different class of patterns. For $\varepsilon > 0.7$ the system allocated a different winner unit for every pattern.

The order patterns were presented to the system had no effect on the simulation results and the performance of the system did not depend on the values of δ.

DAS was also tested against the data set described in Section 3.2 extended by the all-on pattern. Two different transformations of the bitmaps into input vectors were used. One was the standard binary (0,1) transformation. The other used the real interval 0,1 to allow for grey levels. Exhibit 6.5 shows some of the simulation results obtained for different ε values.

ε	C
0.90	27
0.85	27
0.83	27
0.80	24
0.75	19
0.70	14
0.65	13
0.60	11
0.50	7

Exhibit 6.5 The number C of categories is inversely proportional to the value of ε.

For $\varepsilon \geq 0.83$ all 27 patterns were classified into different category. Exhibit 6.6 shows some of the probes that where classified into the corresponding classes A,D,F,H .

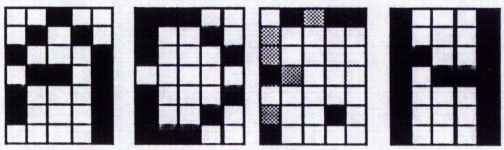


Exhibit 6.6 Some examples of noisy patterns which DAS classified appropriately.

The third pattern in Exhibit 6.6 is an example of a grey-scale probe that was appropriately classified by the system.

After a number of consecutive presentations of the noisy H in Exhibit 6.6, a shift on the weight values of the corresponding winner unit was observed. The system was then probed with H and it was observed that the activation level of the corresponding winner unit was lower than the activation caused by the noisy H. The system was now responding to the noisy version stronger than to the archetype, indicating that the noisy pattern became the archetype of the class.

To test the effects of the decay mechanism the following experiment was made. After presenting the system with all the characters, it was probed using the archetypes themselves. All the patterns were presented to the system for several times except for character O while the decay rate was kept constant. Then a new pattern was presented to the network and the winner unit which corresponded to character O was nominated as the winner unit for the new pattern.

Simulations showed that the decay rate can be adjusted according to the resources left available in the system in such a way that when the system detects a shortage of resources, an increase of the decay rate forces the system to *forget* seldom seen categories and allocate the corresponding winner units to new classes. Alternatively, the decay rate can be set externally according to the expected rate of change in the environment. If categories appear and disappear frequently, a relatively higher decay rate may improve resource utilization. A simulation computer program for DAS is given in Appendix B.

6.7 Context Sensitive Classification

As with the single user auto-associative memory systems of Chapter 4, there are cases where a DAS system fails to classify a probe correctly. Three cases of classification failure can be identified:

(a) Two or more winner units are allocated

(b) No winner unit is allocated

(c) An incorrect winner unit is allocated

Case (a) is probably the most common form of failure. Multiple winner units may be allocated to the same probe in the case where the ε setting allows overlapping areas of attraction (Exhibit 6.3 (B)).

Case (b) is only possible when the system runs out of free-competitor units or it is artificially stopped from storing any further global changes. Case (c) can only happen when the noise in the probe exceeds the prescribed fault tolerance of the system for the particular class. In all three cases classification can be improved by enabling the system to use contextual information. Case (c) is different from (a) and (b) in that identification of failure involves interpretation of the system's response by the user.

As in a SCAM system (Section 4.3.1) context sensitive recognition can be achieved by appropriately representing semantically related patterns. Semantic relations between different patterns can be established either by the system (eg. simultaneous arrival at input) or made explicit by the user. Two possible ways of storing semantic relations between patterns were identified (Sections 4.3.1 and 1.5):

(1) *Binding* (eg. binding the phonetic and visual representations of a letter)
(2) *Hierarchical organization* (eg. letter-word-sentence)

DAS can be extended to operate under either the binding regime or the

hierarchical regime.

(1) *Binding*

Consider the system of Exhibit 6.7. It comprises two DAS systems (DAS^1 and DAS^2), in their initial state and an extra layer of units with one free-competitor unit.

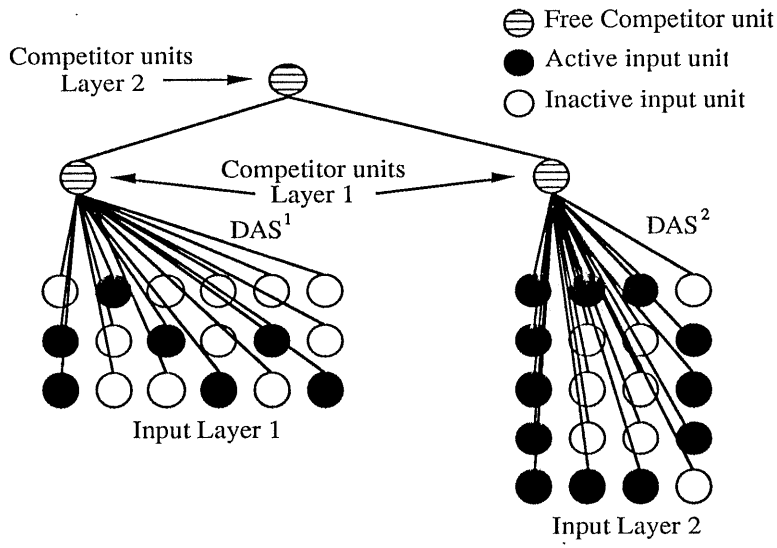


Exhibit 6.7 Multi-layer DAS system which allows for binding of semantically related patterns and context sensitive classification.

Assume that the pattern at the input layer of DAS^1 is the phonetic form of the pattern at the input layer of DAS^2. Weight values are set according to the DAS regime. Each DAS subsystem is evaluated and the corresponding free-competitor unit becomes a winner unit (Exhibit 6.8). Subsequently two free-competitor units are added to DAS^1 and DAS^2 respectively (Exhibit 6.8). The winner units now serve as input units for the free-competitor unit in the next

layer. This part of the system is treated as a standard DAS system.

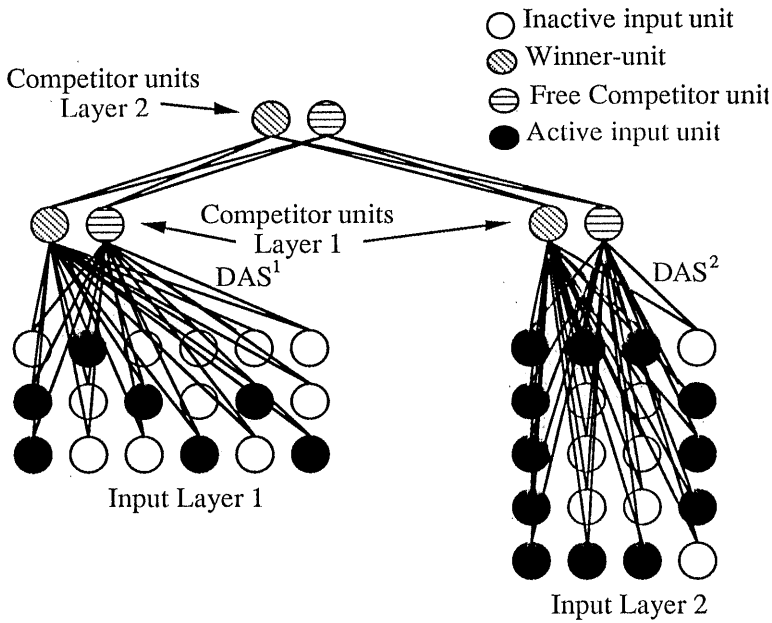


Exhibit 6.8 The system of Exhibit 6.11 after the two input patterns are in store and bind together. Three free-competitor units are added to the system.

The free competitor unit becomes a winner unit representing both input patterns. Subsequently a new free-competitor unit is added. Weight values are modified under the standard DAS regime. Note that because full connectivity is always applied between two consecutive layers of competitor units, adding free competitor units at layer 1 requires that new connections are established not only to the free competitor unit of layer 2 but also to the competitor units of this layer. There arises the problem of deciding the weight values for the newly added connections between free-competitor units at layer 1 and existing competitor units at layer 2. A competitor unit at layer 2 will only become winner unit because of activation at the competitor units of layer

1. Therefore we can safely set the connections between the competitor units at layer 2 and the free-competitor units at layer 1 to be inhibitory.

Let us now describe how context sensitive classification improves cases (a), (b) and (c). Assume that two patterns arrive at input layers 1 and 2 simultaneously. Assume that after evaluation one winner unit is found at DAS^1 and two winner units are found at DAS^2. Using the winner units as input to layer 2, an overall winner unit is allocated. The winner unit at layer 2 will be the one that corresponds to the right combination of winner units at layer 1. Probing the system in reverse, from the winner unit of layer 2 towards layer 1, results in only one winner unit one for each DAS subsystem. Similarly, if there is only one winner unit at layer 1 during the feedforward evaluation of the system, finding a winner unit at layer 2 again results in one winner unit for each layer 1 subsystem. The case of an incorrect winner unit at layer 1, is resolved in exactly the same way.

(2) Hierarchical organization

Consider a system which comprises several DAS subsystems (Exhibit 6.9) For this particular example, every subsystem is a copy of a basic DAS system with the data set of Section 3.2 already in store. The bitmaps of letters H, E, L, L, O are presented to the system as shown in Exhibit 6.9.

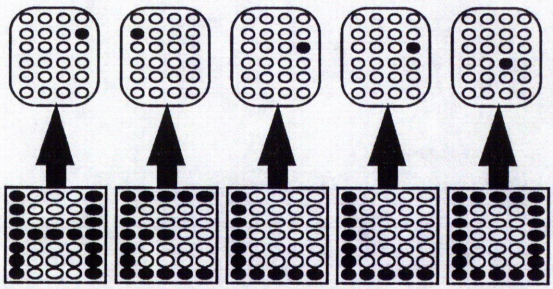


Exhibit 6.9 Multiple copies of a basic DAS system are simultaneously probed with semantically related patterns.

The system can be further extended by adding more basic subsystems and layers of competitor units. In the general case every two consecutive layers of units are treated as a separate DAS subsystem. Note that the properties of the basic DAS system apply to the multi-layer version.
The system of Exhibit 6.10 is subject to two constraints:

- It requires the number of letter classification DAS subsystems to be equal to the length of the longest possible word. This effectively restricts dynamic allocation of resources at a macro level. However dynamic allocation is still available at a micro level.

- The particular example shown here (letter-word) is a special case of a context hierarchy because of the significance in the order of letters. Such representation require that patters are treated *spatially*. For example assume that the maximum length of a word is 16 and the maximum number of words in a sentence is 8. In this case a system with three layers of competitor units, representing a letter-word-sentence hierarchy, requires 16x8 copies of the basic letter classification subsystem. Although resources may not be a real problem, the architecture of the system does not allow for one input layer where letters are presented sequentially and treated temporally.

However, if the order of patterns does not affect their semantic relationship patterns can be treated temporally. Assume that any combination of the letters H, E, L, L, O is a valid representation of the word "HELLO". A system with only one copy of the basic letter classification DAS subsystem and a layer of competitor units for word representation is sufficient. Exhibit 6.11 shows this system at times t =1,2,3,4,5.

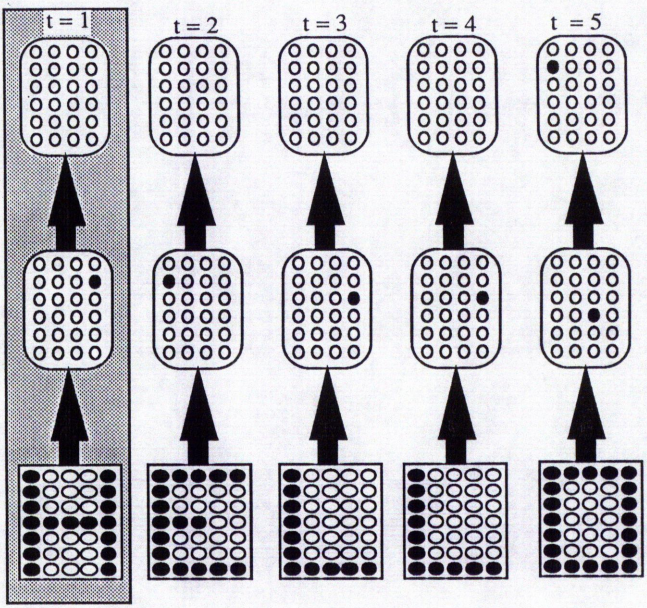


Exhibit 6.11 The architecture of Exhibit 6.11 is reduced to a two layer system with only one cluster of competitor units at each layer.

Every time a letter is presented to the system, a winner unit is allocated at the letter layer. The competitor units at layer 2 are different from the standard units in DAS in that they have to store their activation until all the letters of the word have been presented to the system. Only then they are allowed to check against their threshold settings and decide whether to become a winner unit or not.

The system of Exhibit 6.11 is effectively performing word classification instead of letter classification. However it can be trivially extended to a pattern recall system (Section 5.2.2) which will produce letter output.

6.8 Summary

A Hebbian type adaptation rule was suggested and although it guarantees fixed representations it is unable to handle the all-on pattern effectively. Modifiable thresholds were introduced as the means for a competitor unit to decide whether or not the probe is close to the corresponding archetype. DAS and its associated rules were described. Simulation results were presented showing that the initial design specifications were fully satisfied. Finally, three variations of a multi-layer DAS system capable of context sensitive pattern classification were presented.

CHAPTER 7
CONCLUSION

7.1 Introduction

We have introduced auto-associative memory systems based on Randomized Internal Representations (RIR) and Dynamic Adaptation Scheme (DAS) and shown that both systems are capable of pattern recognition within a real-world environment. We conclude by providing a comparative overview of RIR and DAS systems.

7.2 DAS vs. RIR

Table 7.1 summarizes all the important features of DAS and RIR systems with respect to the design constraints of Chapter 1.

Continuous Adaptation Both RIR and DAS systems have an incremental adaptation rule and provide a stability-plasticity (SP) control which allow for a continuous adaptation regime to be implemented. However the SP control for RIR systems is very different from that for DAS systems. On the one hand RIR are capable of distinguishing between novel and noisy probes by means of a novelty detector device which is graft to the system and is partly implemented in a non-connectionist fashion. On the other hand DAS systems have an implicit mechanism which is built into the competitor units and forms an active part of the system.

	DAS	RIR
Continuous Adaptation (Stability -plasticity control)	*YES*	*YES*
Integrity of Information	*YES*	*YES*
Pattern Deletion (optimization of resources)	*YES*	*NO* [1]
Self-organization	*YES*	*NO* [2]
Pattern Recall	*YES*	*YES*
Pattern Classification	*YES*	*NO*
Context Sensitive Pattern Recognition	*YES*	*YES*
Hebbian type adaptation rule	*YES*	*YES*
Knowledge Representation	*Local*	*Distributed*
Key Feature	*Modifiable Thresholds*	*Randomized Internal Representations*

Table 7.1 An overview of RIR and DAS systems.

Integrity of Information Hopfield and BAM networks suffered loss of information because of the orthogonality constraint imposed over the archetypes. RIR systems address this problem successfully by means of random vectors associated to every archetype pattern. However the generation of the random vectors requires for a Random Vector Generator which is external to the system. Depending on the implementation RIR systems may

---

[1] Only available under supervision.
[2] Under the strong definition of the term.

also require explicit synchronization of the recall process. Competitive Learning (CL) and ART systems guarantee integrity of information only over a fixed set of patterns. It was shown that under a continuous adaptation regime loss of information is inevitable. DAS systems guarantee integrity of information under continuous adaptation but careful control over their decay mechanism operation is required.

RIR and DAS systems are subject to a general capacity constraint with respect to integrity of information. It is assumed that an infinite supply of resource is available to guarantee storage of all the archetypes in the environment. However an archetype deletion operation is available to enable both systems to optimize their resources. In general DAS systems are more efficient than RIR systems because they require fewer units per pattern.

Pattern Deletion DAS systems do provide an automated operation for deleting categories from the system by means of a decay mechanism. In this way an effective control of the system's resources is achieved. DAS systems are capable of sensing whether a category exists in the surrounding environment by internally registering the appearance of probes that belong to this category. If a category does not appear for a substantial amount of time the corresponding resources (units, connections) are gradually reallocated to a new category. RIR systems do provide a deletion operation which is not automated. To engage this operation, an RIR system requires the assistance of a supervisor who explicitly *tells* the system to delete a specific archetype. The deletion of an archetype creates extra storage space which can be immediately occupied by a new archetype. Obviously, in cases where an RIR system is asked to operate under a self-organization regime this deletion operation is not available. Nevertheless, this deletion operation is quite useful for a wide range

of applications because it allows for *on-line* interaction with system.

Self-Organization Both DAS and RIR systems are self-organized under the weak definition of the term. DAS and RIR systems do not need a supervisor to direct their operation. Weight changes are totally self-controlled, enabling them to operate autonomously in any real-world environment. However there are certain *system* parameters which have to be initialized. DAS systems have two system parameters:

- δ for their adaptation rule
- ε for the modifiable threshold control mechanism

RIR systems have only one system parameter:

- ε for the novelty detector device

The system parameters of both DAS and RIR are defined independently of the surrounding environment. However DAS system parameters allow for user intervention if information about the environment is known.

Only DAS systems are considered self-organized under the strong definition of the term. In addition to self-controlled weight changes there is a self-controlled adaptive structure. Although structural adaptation only affects the size and not the basic two layer basic design of the system. RIR systems have a fixed structure which has to be defined prior to the system's exposure to the environment. However this is not a severe limitation for RIR systems because structure is defined independently from the environment. In this sense structure can be viewed as another system parameter.

Pattern Recall and Pattern Classification At the beginning of this thesis it was suggested that pattern recognition connectionist systems have to be subdivided into two categories:

• Pattern Recall systems

• Pattern Classification systems

RIR systems are purely pattern recall devices. The world is described by means of archetypes and recognition of a probe is the recall of the closest archetype. In this sense RIR systems are limited to recognition-assistant roles. DAS systems however, are primarily pattern classification devices. They can be directly applied as pattern recognition devices with the additional responsibility of controlling external devices solely on incoming information. Such an external device can be another layer of units where the archetype of the category is recalled. In this way DAS systems are trivially extended to pattern recall devices.

Context Sensitive Pattern Recognition A number of cases where recognition may be unsuccessful were shown for both DAS and RIR systems. It was suggested that recognition may improve in cases where some form of contextual information is used. Two possible ways of encoding contextual information were suggested:

• Binding

• Hierarchical organization

A Shared Content-Addressable Memory (SCAM) system based on RIR was suggested. SCAM systems support dynamic binding of archetypes and allow for context sensitive recall by means of a synchronized recall operation. At this stage it seems unlike that any distributed representation pattern recall device will be capable of supporting hierarchical organizations. However the suggested three-layer BAM RIR system provides a good basis for investigating whether a multi-layer version of it might be capable of representing some form of pattern hierarchy.

Two variations of a multi-layer DAS system support binding and hierarchical organization respectively. The binding case is effectively a grandmother cell approach while the hierarchical organization is along the lines of similar models suggested for standard CL systems. In both cases context sensitive recall improves on the recognition capabilities of the standard DAS systems.

Hebbian adaptation rule Both RIR and DAS systems support a Hebbian-type adaptation rule. Weight changes depend solely on the state of the two units at either end of a connection. In this way two goals have been achieved. Reduced implementation costs, but more significantly having both RIR and DAS operating under the same adaptation rule makes a possible unification of the two systems under a common analytical framework more feasible.

In summary DAS systems have the following advantages over RIR systems:

- Purely self-organized
- Lower units per category ratio (capacity)
- Support for both binding and hierarchical organization context sensitive recognition
- Pattern classification and recall
- All operations are integrated into the system and no external devices are required

APPENDIX A

RIRs single-user system with a Hopfield Cluster and a Novelty Detector

Simulation Program

```
/* NOTE : The Hopfield cluster is under asynchronous update and has a non-
          zero diagonal weight matrix
*/

/* GENERAL NOTES ON THE NOTATION OF THE PROGRAM
   A stands for activation
   C stands for competitor
   E stands for external
   I stands for internal
   R stands for recall
   no stands for number
   W stands for weight
   B stands for BAM subsystem
   H stands for Hopfield Cluster
Examples : A_of_I_U stands for activation of internal unit,
           R_pattern stands for recall pattern,
           no_of_E_U stands for number of external units,
           BW_values stands for BAM subsystem weight values
           HW_values stands for Hopfield cluster weight values etc..
/*


#include <stdio.h>
#include <math.h>
#define no_of_E_U #VALUE#
#define no_of_I_U #VALUE#
#define no_of_patterns #VALUE#

int BW_values[no_of_E_U][no_of_I_U];
int HW_values[no_of_I_U][no_of_I_U];
int R_pattern[no_of_E_U];
int A_of_E_U[no_of_E_U];
int A_of_I_U[no_of_I_U];

int i,j,k,counter;
float e_value; /* FOR THE NOVELTY DETECTOR */
```

```
main()
{
   /* READ ε (e_value) VALUE */
   scanf("%f",&e_value);

   /* INITIALIZE THE BAM WEIGHT VALUES */
   for (i = 0 ; i < no_of_I_U ; i++)
   {
      for (j = 0 ; j < no_of_E_U ; j++)
      {
         BW_values[i][j] = 0;
      }
   }

   /* INITIALIZE THE HOPFIELD CLUSTER WEIGHT VALUES */
   for (i = 0 ; i < no_of_I_units ; i++)
   {
      for (j = 0 ; j < no_of_I_U ; j++)
      {
         HW_values[i][j] = 0;
      }
   }

   /* MAIN LOOP STARTS HERE */
   for (i = 0 ; i < no_of_patterns ; i++)
   {
      /* INITIALIZE THE ACTIVATIONS OF THE INTERNAL LAYER */
      for (i = 0 ; i < no_of_I_U ; i++)
      {
         A_of_I_U[j] = 0;
      }

      /* INITIALIZE THE ACTIVATIONS OF THE EXTERNAL LAYER */
      for (i = 0 ; i < no_of_E_U ; i++)
      {
         A_of_E_U[j] = 0;
         R_pattern[j] = 0;
      }


      /* READ A PROBE PATTERN */
      for (j = 0 ; j < no_of_E_U ; j++)
      {
         scanf("%f",&A_of_E_U[j]);
      }

      /* COMPUTE ACTIVATION AT THE INTERNAL LAYER */
      for (j = 0 ; j < no_of_I_U ; j++)
      {
         for (k = 0 ; k < no_of_E_U ; k++)
         {
            A_of_I_U[j] = A_of_I_U[j] + A_E_U[k] * BW_values[k][j];
         }
         /* THRESHOLD THE UNITS */
         if (A_of_I_U[j] >= 0)
```

```
    {
        A_of_I_U[j] = 1;
    }
    else
    {
        A_OF_I_U[j] = -1;
    }
}

/* COMPUTE THE RESPONSE OF THE HOPFIELD CLUSTER */
for (j = 0 ; j < no_of_I_U ; j++)
{
    for (k = 0 ; k < no_of_I_U ; k++)
    {
        A_of_I_U[j] = A_of_I_U[j] + A_I_U[k] * HW_values[k][j];
    }
    /* THRESHOLD THE UNITS */
    if (A_of_I_U[j] >= 0)
    {
        A_of_I_U[j] = 1;
    }
    else
    {
        A_OF_I_U[j] = -1;
    }
}

/* COMPUTE THE RECALL AT THE EXTERNAL LAYER */
for (j = 0 ; j < no_of_E_U ; j++)
{
    for (k = 0 ; k < no_of_I_U ; k++)
    {
        R_pattern[j] = R_pattern[j] + A_of_I_U[k] * BW_values[k][j];
    }
    /* THRESHOLD THE UNITS */
    if (R_pattern[j] >= 0)
    {
        R_pattern[j] = 1;
    }
    else
    {
        R_pattern[j] = -1;
    }
}

/* INITIALIZE counter */
counter = 0;

/* USE THE NOVELTY DETECTOR TO COMPARE THE RECALL
   PATTERN TO THE PROBE */
for (j = 0 ; j < no_of_E_U ; j++)
{
    if (R_pattern[j] * A_of_E_U[j] > 0)
    {
        counter = counter + 1;
```

```
            }
        }

        /* CHECK TO SEE WHETHER THE PROBE IS A NOVEL
           ARCHETYPE */
        if (counter < no_of_E_U * e_value)
        /* AND MODIFY THE WEIGHTS */
        {
            /* GENERATE A RANDOM PATTERN */
            for (j = 0 ; j < no_of_I_U ; j++)
            {
                A_of_I_U[j] = (random() % 2) * 2 ) - 1;
            }

            /* MODIFY THE WEIGHTS AT THE BAM */
            for (j = 0 ; j < no_of_I_U ; j++)
            {
                for (k = 0 ; k < no_of_E_U ; k++)
                {
                    BW_values[k][j] = BW_values + A_of_E_U[k]*A_of_I_U[j];
                }
            }

            /* MODIFY THE WEIGHTS AT THE HOPFIELD CLUSTER */
            for (j = 0 ; j < no_of_I_U ; j++)
            {
                for (k = 0 ; k < no_of_I_U ; k++)
                {
                    HW_values[k][j] = BW_values + A_of_I_U[k]*A_of_I_U[j];
                }
            }
        }
        else
        { /* RECALL IS COMPLETED */
            for (j = 0 ; j < no_of_E_U ; j++)
            {
                printf("%d",R_pattern);
            }
        }
    }
}

/*
NOTE : THIS C PROGRAM SHOULD COMPILE AND RUN ON ANY
       UNIX BASED SYSTEM

*/
```

APPENDIX B

Dynamic Adaptation Scheme (DAS)
Simulation Program

```
/* GENERAL NOTES ON THE NOTATION OF THE PROGRAM
   A stands for activation
   C stands for competitor
   U stands for unit
   I stands for input
   T stands for thershold
   no stands for number
   max stands for maximum
   W stands for weight
Examples : A_of_I_U stands for activation_of_input_unit,
           A_of_C_U stands for activation_of_competitor_unit,
           no_of_C_U stands for number_of_competitor_units,
           T_of_C_U stands for threshold of competitor unit etc..
/*


#include <stdio.h>
#include <math.h>
#define max_no_of_C_U  #VALUE#
#define no_of_I _U #VALUE #
#define no_of_patterns #VALUE#

float W_values[no_input_units][max_no_competitor_units];
float A_of_C_U[max_no_of_C_U];
float A_of_I_U[no_of_I_U];
float T_of_C_U[max_no_of_C_U];

int i,j,k;
float initial_W_value, e_value, d_value;
int no_of_C_U, winner_U;

main()
{
   /* READ INITIAL WEIGHT, ε (e_value) and δ (d_value)VALUES
   scanf("%f",&initial_W_value);
   scanf("%f",&e_value);
   scanf("%f",&d_value);
   scanf("%f",%delta);

   /* INITIALIZE THE WEIGHT VALUES */
   for (i = 0 ; i < max_no_of_C_U ; i++)
```

```
{
   for (j = 0 ; j < no_of_I_U ; j++)
   {
      W_values[i][j] = initial_W_value;
   }
}

/* INITIALIZE ACTIVATION AND THRESHOLD VALUES */
for (j = 0 ; j < max_no_of_C_U ; j++)
{
   A_of_C_U[j] = 0;
   T_of_C_U[j] = 0;
}

/* INITIALIZE NUMBER OF COMPETITOR UNITS */
no_of C_U = 0;

/* MAIN LOOP STARTS HERE */
for (i = 0 ; i < no_of_patterns ; i++)
{
   winner_U = 0; /* STATE THAT A NON WINNER UNIT EXISTS */
   /* READ A PATTERN */
   for (j = 0 ; j < no_of_I_U ; j++)
   {
      scanf("%f",&A_of_I_U[j]);
   }

   for (j = 0 ; j < no_of_C_U + 1 ; j++)
   {
      /* COMPUTE THE ACTIVATION OF COMPETITOR UNIT j */
      for (k = 0 ; k < no_of_I_U ; k++)
      {
         A_of_C_U[j] = A_of_C_U[j] +A_of_I_U[k] * W_values[k][j]
      }

      /* CHECK ACTIVATION AGAINST THRESHOLD AND
         IF COMPETITOR UNIT j IS A WINNER OR
         THE FREE-COMPETITOR UNIT IS THE WINNER BUT
         NO OTHER WINNER UNIT EXISTS */
      if (A_of_C_U[j] >= T_of_C_U[j] && j < no_of_C_U ||
         A_of_C_U[j] >= T_of_C_U[j] && winner_U== 0)
      {
      /* MODIFY WEIGHTS */
         for (k = 0 ; k < no_of_I_U ; k++)
         {
            if ( A_of_I_U[k] > 0)  /* ACTIVE INPUT UNIT */
            {
               weight_values[k][j] = W_values[k][j] +
               ((A_of_C_U[j] - T_of_C_U[j]) / A_of_C_U[j]) * delta;
            }
            else  /* INACTIVE INPUT UNIT */
            {
               W_values[k][j] = W_values[k][j] -
               ((A_of_C_U[j] - T_of_C_U[j]) / A_of_C_U[j]) * delta;
```

```
                }
            }

            /* COMPUTE NEW ACTIVATION FOR WINNER UNIT /*
            for ( k = 0 ; k < no_of_I_U ; k++)
            {
                A_of_C_U[j] = A_of_C_U[j]+ A_of_I_U[k]*W_values[k][j];
            }

            /* SET NEW THRESHOLD FOR WINNER_UNIT */
            T_of_C_U[j] = A_of_C_U[j]*e_value;
            A_of_C_U[j] = 0; /* REINITIALIZE ACTIVATION */
            winner_U = 1; /* STATE THAT A WINNER UNIT EXISTS */
            /* IF THE WINNER UNIT IS THE FREE COMPETITOR UNIT
                ALLOCATE A NEW FREE COMPETITOR UNIT */
            if (j == no_of_C_U)
            {
                no_of_C_U = no_of_C_U + 1;
            }
        }

        /* IF UNIT j IS NOT A WINNER */
        else
        {
            T_of_C_U[j] = T_of_C_U[j] - d_value /* OPTIONAL  DECAY */
            A_of_C_U[j] = 0; /* REINITIALIZE ACTIVATION */
        }
    }
  }
}


/*
NOTE : THIS C PROGRAM SHOULD COMPILE AND RUN ON ANY
       UNIX BASED SYSTEM

*/
```

REFERENCES

[001] Rumelhart, D.E., Hinton, G.E. and Williams R.J., *Learning Representations by back-propagating errors*, Nature, Vol. 323, 9 October 1986.

[002] Carpenter, G.A. and Grossberg, S., *A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine*, Computer Vision, Graphics and Image Processing, Vol. 37, pp. 54-115, 1987

[003] Fukushima, K., *Cognitron: A self-organizing Multilayered Neural Network*, Biological Cybernetics, Vol. 29, pp. 121-136, 1975.

[004] Fukushima, K., Miyake, S and Takayuki, I., *Neocognitron: A neural Network Model for a Mechanism of Visual Pattern Recognition*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-13, No. 5, September/October 1983.

[005] Cross, J.F. and Longstaff I.D., *A pattern recognition approach to understand the multi-layer perceptron*, Pattern Recognition Letters, Vol. 5, pp. 315-319, 1987

[006] Aleksander, I., *Adaptive pattern recognition systems and Boltzmann machines: A rapprochement*, Pattern Recognition Letters, Vol. 6, pp. 113-120, 1987

[007] Giebel, H. and Marko, H. , *Recognition of Handwritten Characters with a System of Homogeneous Layers*, Kybernetik, Heft 9, pp. 455-459, 1970

[008] Carpenter, G.A., and Grossberg S., *The ART of adaptive Pattern Recognition by a Self-Organizing Neural Network*, IEEE Computer, March 1988, pp. 77-88

[009] Rumelhart ,D.E. and Zisper, D., *Feature Discovery by Competitive Learning*, in Parallel Distributed Processing McClelland, Rumelhart and the PDP Research Group, Vol. 1, Chapter 5, MIT Press, 1986

[010] McEliece, J.R, Posner C.E., Rodemich, R.E and Venkatesh S.S., *The Capacity of the Hopfield Associative Memory*, IEEE Transaction on Information Theory, Vol. IT-33, No. 4, July 1987.

[011] McCulloch, W.S. and Pitts, W., *A logical calculus of the ideas immanent in nervous activity*, Bulletin of Mathematical Biophysics, Vol. 5, pp. 115-133, 1943

[012] Minsky, M., *Neural Networks*, in Computation Finite and Infinite Machines, Prentice-Hall Inc., Series in Automatic Computation, pp 32 -

66, 1967

[013] Hebb, D.O., Organization of Behaviour, Wiley, New York, 1949

[014] Willshaw, D. and Dayan, P., *Optimal Plasticity from Matrix Memories: What goes up MUST come down*, Technical Report, University of Edinburgh, Centre for Cognitive Science and Department of Physics, 19 December 1989

[015] Rosenblatt, F., *Principles of Neurodynamics*, Spartan Books, pp. 79 - 94, 1962

[016] Rosenblatt, F., *Principles of Neurodynamics*, Spartan Books, pp. 489 - 493, 1962

[017] Rosenblatt, F., *Two Theorems of Statistical Separability in the Perceptron*

[018] Rosenblatt, F., *The Perceptron: Aprobabilistic Model for Information Storage and Organization in the Brain,* Psychological Review, Vol. 65, No. 6, 1958

[019] Grossberg, S., *Nonlinear difference-differential equations in prediction and learning theory*, Proc. Natl. Acad. Sci. USA, Vol. 58, pp. 1329-1334, 1967

[020] Grossberg, S., *Neural Pattern Discrimination*, Journal of Theoretical Biology, Vol. 27, pp. 2391-337, 1970

[021] Grossberg, S., *Global Ratio Limit Theorems for some Nonlinear Functional-Differential Equations II*, Bulletin American Mathematical Society, pp. 101-105, 1968

[022] Grossberg, S., *Some physiological and biochemical consequences of psychological postulates*, Applied Mathematics, Vol. 60, pp. 758-765, 1968

[023] Wickelgren, W.A., Bulletin of Mathematical Biophysics, Vol. 31, 123.

[024] Adamopoulos, A. and Anninos, P., *Computer simulations of neural networks with chemical markers*, Parallel Processing in Neural Systems and Computers, North Holland, pp. 49 - 52, 1990

[025] Willshaw, D., *Holgraphy, Associative Memory, and Inductive Generalization*, in Hinton G.E. and Anderson, J.A, Parallel Model of Associative Memory, Lawrence Erlbaum Associates Inc., 1981

[026] Grossberg, S., *Adaptive Pattern Classification and Universal Recoding: I Parallel Development and Coding of Neural Feature Detectors*, Biological Cybernetics, Vol. 23, pp. 121-134, 1976

[027] Malsburg, von der C., *Self-Organization of Orientation Sensititve Cells*

in the Striate Cortex, Kybernetik, Vol. 14, 85-100, 1973

[028] Grossberg, S., *On the Development of Feature Detectors in the Visual Cortex with Applications to Learning and Reaction-Diffusion Systems, Biological Cybernetics*, Vol. 21, 145-159, 1976

[029]Tenorio, M.F.M., *The Self-Organizing Neural Network Algorithm: Adapting Structure for Optimum Supervised Learning*, Proc. of the 23rd Annual Hawaii International Conference on System Sciences, Vol. 1, pp. 187-195, 1990

[030] Feldman, J.A., *Dynamic Connections in Neural Networks*, Biological Cybernetics, Vol. 46, pp. 27-39, 1982

[031] Feldman, J.A., *Neural Representations of Conceptual Knowledge*, Technical Report 189, University of Rochester, Department of Computer Science, June 1986

[033] Amit, J.D., Gutfreund H. and Sompolonsky H., *Storing Infinite Numbers of PAtterns in a Spin-Glass Model of Neural Networks*, Physical Review Letters, Vol. 55, No. 14, 30 Septemeber 1985

[034] Hopfield, J.J., *Neural Networks and physical systems with emergent collective computational abilities*, Proc. Natl. Acad. Sci. USA, Vol. 79, pp. 2554-2558, April 1982, Biophysics

[035] Kirkpatrick, S. and Sherrington D., *Infinte-ranged models of spin-glasses*, Physical Review B, Vol. 17, No. 11, 1 June 1987

[036] Rumelhart ,D.E., Smolensky, P., McClelland J.L, and Hinton, G.E., *Schemata and Sequential Thought Processes in PDP Models*, in Parallel Distributed Processing McClelland, Rumelhart and the PDP Research Group, Vol. 2, Chapter 14, MIT Press, 1986

[037] Abu-Mustafa, S. and St.Jacques J., *Information Capacity of the Hopfield Model*, IEEE Transactions on Information Theory, Vol. IT-31, No. 4, July 1985

[038] Cowan, J.D. and Sharp, D.H., *Neural Nets*, Technical Report, Los Alamos National Laboratory, LA-UR-87-4098, Los Alamos, 1987

[039] Anderson, J.A. and Hinton G.E., *Models of Information Processing in the Brain*, in Hinton G.E. and Anderson, J.A, Parallel Model of Associative Memory, Lawrence Erlbaum Associates Inc., 1981

[040] Kosko, B., *Bidirectional Associative Memories*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 18, No. 1, January/February 1988.

[041] Tank, D.W. and Hopfield, J.J., *Computing with Neural Circuits*: A

Model, Science 233

[042] Stork, D.G., *Self-Organization, and Adatpive Resonance Networks*, Journal of Neural Network Computing, pp. 26-42, Summer 1989

[043] Grossberg, S., *Competitive Learning: From interactive activation to adaptive resonance*, Cognitive Science, Vol. 11, pp. 23-63, 1987

[044] Rumelhart, D.E. and McClelland, L., *An interactive Activation Model of Context Effects in Letter Perception: Part . The Contextual Enhancement Effect and Some Test and Extensions of the Model*, Psychological Review, Vol. 89, No. 1, pp. 60-94, 1982

[045] Rumelhart ,D.E., McClelland J.L, and Hinton, G.E., *The appeal of PDP*, in Parallel Distributed Processing McClelland, Rumelhart and the PDP Research Group, Vol. 1, Chapter 1, MIT Press, 1986

[046] McClelland, J.L., *Connections between Levels of Description of Perception*, Proc. IJCNN '90, Vol. I, pp. 743-747, 1990

[047] Bairaktaris, D., *A connectionist model for character recognition, Proc. Parallel Computing* '89, pp. 359-364, 1989

[048] Hopfield, J.J., and Tank, D.W., *Computing with Neural Circuits: A Model*, Science, Vol. 233, pp. 625-633

[049] Galindo, P.L. and Michaux, T., *An improved Competitive Learning Algorithm Appied to High Level Speech Processing*, Proc. of International Joint Conference on Neural Networks, Vol. I, pp. 142-146, 1990

[050] Grossberg, S. & Stone, G., *Neural Dynamics of Word Recognition and Recall: Attentional Priming, Learning, and Resonance,* Psychological Review, Vol. 93, No. 1, pp. 46 -74, 1986

[051] Reicher, G.M., *Perceptual recognition as a function of meaningfulness of stimulus material*, Journal of Experimental Psychology, Vol. 81, pp. 274-280, 1969

[052] Anderson, J.R., *Spreading Activation*, in Tutorials in Learning and Memory, Anderson, J.R and Kosslyn, S.M., W.H. Freman Company, New York, pp. 61-91, 1984

[053] Carpenter, G.A., and Grossberg S., *Adaptive Resonance Theory: Neural Network Architectures for Self-Organizing Pattern Recognition*, Parallel Processing in Neural Systems and Computers, North Holland, pp. 383 - 389, 1990

[054] Simpson, P., *Artificial Neural Systems*, Pergamon Press, pp. 58-63, 1990.

[055] Mezard, M. and J-P. Nadal J-P, *Learning in Feedforward Layered Networks: the Tling Algorithm*, Journal of Physics General A, 22, pp.2191-2204, 1989.