"If you want to learn how to train networks yourself, you take your network and you go to where the road crosses that way, where a crossroads is. Get there be sure to get there just a little 'fore 12 that night so you know you'll be there. You have your network and be training a problem there by yourself ... A big man will walk up there and take your network and he'll tune it. And then he'll train a problem and hand it back to you. That's the way I learned to train anything I want."

with apologies to Tommy Johnson

University of Alberta

The Crossroads of Connectionism: Where Do We Go from Here?

by

David Alexander Medler ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of *Doctor of Philosophy*

Department of Psychology

Edmonton, Alberta

Spring 1998

0-612-29077-8

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and research for acceptance, a thesis entitled *The Crossroads of Connectionism: Where Do We Go from Here?* by *David Alexander Medler* in partial fulfillment of the requirements for the degree of *Doctor of Philosophy*.

[illegible]
Dr. Michael R. W. Dawson

[illegible]
Dr. Alan Kingstone

[illegible]
Dr. C. Donald Heth

[illegible]
Dr. Peter van Beek

AK
Dr. Paul Thagard

3 - 26 - 98
Date

Abstract

Connectionist research has reached a crossroads: while some view connectionism as a Kuhnian-like paradigm shift (e.g., Schneider, 1987), others view it as no more than a mere implementational account of classical cognitive theories (e.g., Fodor & Pylyshyn, 1988). Consequently, connectionism is in need of reevaluation if it is to become an effective tool within cognitive science. To this end, it will first be argued that to be of import to science overall, connectionism must fulfill three functions: (1) the collection of data, (2) the direct comparison of data to theory, and (3) the articulation of theory. Furthermore, within cognitive science, connectionism must be able to explain information processing at three different levels of analysis: computational, algorithmic (including the functional architecture), and implementational. Thus, to begin our reevaluation, a framework for connectionism is developed wherein the essentials of cognitive science, information processing, and the basic tenets of connectionism are presented. This is followed by a brief review of the history of connectionism, from its origins with the Greek philosopher, Aristotle, to the neural network architectures in use today. With this framework established, connectionism's contributions to the tri-level hypothesis are explored. It is shown that from a computational perspective, both computability and complexity theory are fundamental to connectionism and therefore connectionism is able to answer those questions that are of interest to cognitive science. Furthermore, connectionism also aids in developing explanatory theories based on natural computation. From the algorithmic perspective, it is shown that analysis of the internal structure of connectionist models is a necessary step if connectionism is to contribute to cognitive science. Such analyses reveal that connectionist networks can produce novel cognitive theories, yet—when constrained—can also learn existing theories. Moreover, studies of the functional architecture show that lesioned connectionist networks can be used to inform cognitive neuroscience, especially in terms of semantic networks and the locality assumption. Finally, at the implementational level it is shown how connectionism shares a two-way interaction with neuroscience. It is therefore concluded that connectionism represents neither a paradigm shift nor a mere implementational account of classical theories. Connectionism is, however, an effective tool within cognitive science and has the potential to unite the field.

Acknowledgments

First and foremost I would like to thank my supervisor, Mike Dawson, for inspiration, motivation, and perspiration (*the fear is real!*). Many times have you gone above and beyond the call of a supervisor to keep me in this game. I would also like to thank the other members of my supervisory committee, Alan Kingstone and Don Heth, for keeping me honest and pointing me in directions that I might not otherwise have considered. And to all the denizens skulking about in the basement with me, keep the coffee brewing. In other words, I would like to thank the members of the Biological Computation Project (past, present, and honourary) who have provided for many "interesting" discussions while consuming far too much coffee. Special acknowledgments go out to Istvàn Berkeley, Stefan Kremer, and David McCaughan who helped me refine many of the ideas presented in this thesis, and to Michael Snyder who politely listened to me ranting about one thing or another. Finally, the research in this thesis has been supported in part by a Province of Alberta Graduate Fellowship, an Andrew Stewart Memorial Graduate Prize, and an Izaak Walton Killam Memorial Scholarship.

Table of Contents

List of Tables

List of Figures

Chapter 1

The Crossroads of Connectionism

Connectionist research has reached a crossroads. While some hail connectionism as a Kuhnian-like paradigm shift (e.g., Schneider, 1987), others dismiss connectionist models as mere implementational accounts of classical cognitive architectures (e.g., Broadbent, 1985; Fodor & Pylyshyn, 1988). Although these original arguments are now a decade old, the debate over connectionism's place in cognitive science still continues today (e.g., Jackendoff, 1992; Burr, 1994; Gallistel, 1995; Horgan & Tienson, 1996). Even those researchers who view connectionist research as a valid enterprise have challenged its ability to contribute significantly to cognitive science (e.g., McCloskey, 1991; Dawson, Shamanski, & Medler, 1993; Dawson & Shamanski, 1994). Clearly, connectionism is in need of reevaluation if it is to become an effective tool within cognitive science.

This use of the word 'tool' is intentional. In this incarnation, a tool is defined as an instrument useful in the practice of a vocation or profession (although the more cynical scientist may find the alternative definition of 'an unwitting or compliant agent of another' more appropriate in these circumstances). Tools are devised when a science requires more exact observations (Hull, 1943); that is, when the previous tools of science no longer provide suitable answers to the questions being posed, new tools must be fashioned. Consequently, the purpose of a tool within science is threefold: (i) to determine significant facts pertinent to the field of study; (ii) to compare fact with theory directly; and (iii) to aid in the articulation of theory itself (Kuhn, 1970). Therefore, we must show that connectionism can fulfill these three goals if it is to contribute significantly to science in general and cognitive science in particular.

Moreover, once committed to this three-pronged approach, we are necessarily committing ourselves to both empirical and theoretical work—the "two essential elements of modern science" (Hull, 1943, p. 1). The definition of empirical work is relatively easy[1]: it is the collection and analysis of data. We can collect data through self-report (e.g., protocol analysis; Newell & Simon, 1963) or through direct observation (e.g., neural responses to stimuli; Hubel & Weisel, 1959). Furthermore, in terms of analysis, data can be simply described (e.g., knowledge from analogy; Lorenz, 1974), correlated with other data (e.g., functional localization of the brain and structural damage; Kolb & Whishaw, 1996), or experimentally manipulated to produce a cause-and-effect relationship (e.g., learning as a function of repetition; Thorndike, 1932).

Theoretical work, on the other hand, comprises three main areas: (i) the explanation of known data, (ii) the prediction of new data, and (iii) inter-theoretic reduction (e.g., Fodor, 1968; Churchland, 1989; McCloskey, 1991; Seidenberg, 1993). Furthermore, the theories we construct often constrain the way in which we perceive the world (Pylyshyn, 1984). In other words, by adopting one theory over another, we are in fact committing ourselves to a specific interpretation of the world, which consequently influences our empirical work (Nagel, 1961). More formally, the purpose of a theory is to:

1. organize and interpret the known data so generalizations beyond the original data may be stated;
2. clearly state which elements of the theory can be credited when the theory correctly accounts for data (i.e., identify those aspects of the theory responsible for generating the correct prediction);
3. clearly state which elements of the theory can be blamed when the theory incorrectly accounts for data (i.e., relate the failure to specific assumptions of the theory, and determine if the failure is fundamental to the theory);
4. explain its important similarities and dissimilarities to alternative theories in the same domain;

[1] Nagel (1961), however, states that empirical work and the resulting experimental laws are not only dependent upon a loose definition of the word "observable" but are also often couched within the language of some theory. Consequently, in Nagel's opinion, a precise definition of empirical work is open to interpretation and far from easy.

5. potentially account for other phenomena beyond the scope of the original theory;

In addition to these five basic elements of theory building, Seidenberg (1993) describes two more principles that are important for cognitive theorizing.

6. explain phenomena in terms of independently motivated principles;
7. show how phenomena previously thought to be unrelated actually derive from a common underlying source.

Therefore, to be effective, connectionist researchers should not only be able to provide empirical data from their simulations, but they should also be able to describe how their models contribute to the theoretical knowledge of the domain in question. It should be noted that empirical data constrains the formulation of our theories, and the theories themselves are required not only to explain the data we have collected, but also to guide our research questions. Consequently, connectionists must engage in both theoretical and empirical work to be productive researchers within the realm of science.

Empirical vs. Theoretical Work

The preceding statement about engaging in both theoretical and empirical work has proved contentious to some researchers. This is especially true (if not confusing!) within connectionist research. One side of the argument claims that connectionist research can be pursued within a purely theoretical framework, much like theoretical physics is often studied separately from experimental physics (Kukla, 1989, 1990). Others take a neutral stance, stating that connectionist research is both theoretical and empirical in nature (Dawson, 1990; Seidenberg, 1993). Finally, there are those that argue that connectionism is not a theory in and of itself, but nevertheless can be used to empirically test theories (McCloskey, 1991).

For example, Kukla (1989, 1990) maintains that all computer simulations fall into the classification of purely theoretical work[2]: "AI is armchair psychology made respectable again" (1989, p. 786) and even the "(r)ecent departures from classical AI, such as connectionist theories,

[2] This is in direct opposition to Newell & Simon's (1981) assessment of computer science and Artificial Intelligence given during the tenth Turing Lecture: "Computer science is an empirical discipline" (p. 35). In fact, this view is still common today, " AI is currently an experimental discipline as opposed to a theoretical one" (Ginsberg, 1993, p. 395).

have continued to meet these new standards of formal rigor" (1990, p. 780). Kukla's arguments are based on his belief that computer programs are simply extremely well formulated theories. If we had enough time and resources, we could sit down, think the programs through in our mind, and evaluate their outcome—computers simply make this step easier. Hence, one could conclude from Kukla that connectionist researchers who use computers to simulate their models are necessarily engaged in purely theoretical work.

Conversely, McCloskey (1991) argues that "connectionist networks should not be viewed as theories of human cognitive functions, or as simulations of theories, or even as demonstrations of specific theoretical points" (p. 387). Instead, connectionist models should be viewed as similar to animal models of human behaviour; that is, not a theory of human cognition *per se*, but rather an object of study:

> . . . one can observe the effects of varying the network architecture while holding the training process constant; one can subject a network to several different forms of damage, and restore it to its "premorbid" state; one can inspect connection weights and activation patterns across hidden units; and so forth. (p. 393)

In other words, connectionist research should be considered simply as an empirical endeavor that can be used nonetheless to contribute to the development of theories. "Modeling is an aid to, but not a substitute for, theoretical work" (p. 393).

The middle ground, and the one taken here, is that connectionist researchers engage in both empirical and theoretical work. Seidenberg (1993) describes connectionism as a body of tools that can be used not only to simulate preexisting theories (functioning like a statistical tool analyzing a complex set of data), but also to develop theories that are explanatory rather than merely descriptive. Similarly, Dawson (1990) states that although many properties of connectionism are analyzed formally, the underlying motivation for connectionist research is empirical. Consequently, the position adopted in this thesis is that connectionism is both theoretical—it constrains our view of cognitive science—and empirical in that the data collected from connectionist models are used not only to support these theories, but also to modify and elaborate these theories. As Thagard (1996, p. 8) notes ". . . theory without experiment is empty, experiment without theory is blind."

There are many examples of connectionism contributing to both the empirical and theoretical aspects of science in general (e.g., medical diagnosis, Baxt, 1992; recognition of text on engineering documents, Gouin & Scofield, 1994; image processing, Hecht-Nielsen, 1992; predicting globular protein sequences, Qian & Sejnowski, 1988; stock market prediction, White, 1988). Why, then, are researchers at a crossroads when it comes to cognitive science in particular? The answer to this question lies in understanding the questions that cognitive science is interested in, how these questions have traditionally been answered, and how connectionists are attempting to answer these questions today.

A Guide to the Crossroads

To understand why we are now at the crossroads of connectionism, we first have to understand how we arrived here. Consequently, the first half of the thesis will focus on the fundamental assumptions underlying connectionism, the history of connectionism, and current connectionist architectures.

To begin our reevaluation, Chapter 2 will provide a brief introduction to cognitive science and the underlying principles of connectionism. This will include the main assumption that the mind is an information processor, and that to explain any information processor fully, a researcher must appeal to a tri-level hypothesis. That is, a researcher must be able to describe an information processor at a computational, algorithmic (including the functional architecture), and implementational level (e.g., Marr, 1982). Connectionism fills a niche in cognitive science that has been created by a breakdown in the commitment to the tri-level hypothesis by the traditional researchers within the area.

Chapter 3 will trace how connectionism has evolved as a tool within science. Contrary to popular belief[3], connectionism has a long and varied past. The roots of connectionism can be traced back to the early Greek philosopher, Aristotle, while some of its more modern philosophical themes are derived from the British empiricist tradition. Early psychologists, such as William James and Edward Thorndike, played quite a large role in the development of connectionist ideas, as did neuropsychologists like Karl Lashley and Donald Hebb. Connectionism really began to flourish, however, with the invention of the digital computer and the work of researchers like Frank Rosenblatt and Oliver Selfridge. Ironically, it was the computational analysis of these early computer models by Marvin Minsky and Seymour Papert in their book *Perceptrons* that caused the near

[3] For example, Horgan and Tienson (1996, p. 173) allude that, for all intents and purposes, connectionism emerged (or at least reemerged in any substantial form) in the 1980's as "attested by the pre-publication sellout of the relatively expensive, two-volume 'Bible of Connectionism,' *Parallel Distributed Processing*" edited by Rumelhart, McClelland & the PDP Group.

demise of connectionist research. The ideas and models in this chapter comprise what is known as Old Connectionism.

The fourth chapter will address New Connectionism; that is, connectionist research in the post-*Perceptrons* era. It will begin by describing McClelland's *Interactive Activation and Competition* (IAC) model of information storage and retrieval and how it relates to the notion of information processing in cognitive science. Then, the most popular learning algorithm in use today—Rumelhart, Hinton, and Williams' *Generalized Delta Rule* (a.k.a., backpropagation of error)—will be presented. The chapter will also present variations on this algorithm, specifically Dawson and Schopflocher's *value unit* architecture. Finally, other connectionist architectures, such as radial basis function networks, self-organizing networks and recurrent networks, will also be introduced.

With the introduction to connectionism completed, the remainder of the thesis will focus on answering some of the questions that cognitive science is interested in addressing. Specifically, connectionism's contribution to addressing questions posed at Marr's (1982) computational, algorithmic (including Pylyshyn's functional architecture), and implementational level will be covered.

In Chapter 5, connectionism's contribution to the analysis of the computational level will be discussed. First, the two types of information processing problems—function approximation and pattern classification—addressed by cognitive science will be defined in terms of computability, complexity, and information processing theory. Second, empirical results will be presented to show the *in practice* power of connectionist networks on a limited set of function approximation and pattern classification problems. Finally, it will be concluded that not only can connectionism answer questions posed at the computational level of analysis, but connectionism also provides a method of exploring natural computation and developing explanatory theories.

Chapter 6 will elaborate a newly developed technique of analyzing how connectionist networks can contribute to the algorithmic description of information processors (Berkeley, Dawson, Medler, Schopflocher, & Hornsby, 1995; Dawson, Medler, & Berkeley, 1997). This technique, dubbed "banding analysis", allows for an easy interpretation of the internal representations adopted by the hidden units within connectionist networks. Chapter 6 will focus on a particular pattern classification problem—the mushroom problem—and compare a "classical" machine learning algorithm with some connectionist models. Furthermore, a new way of inserting "classical rules" into the network structure by elaborating the output of the networks will be introduced. The results of this section will be discussed in terms of connectionism's contribution to the algorithmic level of analysis.

Two recent connectionist approaches to examining the functional architecture will be discussed in Chapter 7. To answer questions at this level, both approaches lesion network structure to produce behavioural deficits. The first approach will investigate lesioning an IAC network and comparing the performance of the network to patients suffering from Alzheimer's disease. The results from this experiment provide an alternative explanation to the semantic networks proposed by classical cognitive science. The second approach evaluates the locality assumption as used in cognitive neuroscience. It was previously concluded (Farah, 1994) that results from lesioned connectionist networks gave reason to dismiss the locality assumption. By applying the banding analysis developed in Chapter 6 to the internal structure of a network, and correlating these analyses with observed behavioural dissociations of the lesioned network, it will be shown that this dismissal may have been premature. Consequently, connectionism can be used to address concerns about the functional architecture of cognition.

In Chapter 8, it will be shown how connectionist research has filled the niche at the implementational level often missing from classical accounts within cognitive science. Two different lines of research will be reviewed. First, it will be shown how biological constraints are being applied to connectionist networks to produce models that are more neuromorphic. Second, it will shown how connectionism has contributed to new knowledge within neuroscience. To end the chapter, new research will be presented on using redundancy in connectionist networks, and how this implementational design decision not only improves the performance of neural networks, but also allows us to answer questions about the possible relevance of redundancy to biological systems.

Finally, Chapter 9 will provide us some direction as to where we should be headed. Specifically, the question of whether or not connectionism is a paradigm shift for cognitive science or merely an implementation of classical theories will be addressed. It will be concluded that, to be effective, connectionist researchers must define their problems more clearly, interpret the internal structure of their networks, and pay more attention to biological constraints when designing their networks. When these constraints are met, then connectionism is able to fulfill the three agendas of science and becomes an effective tool within cognitive science. More importantly, connectionism possesses the ability to unite the field of cognitive science.

Chapter 2

Connectionism and Cognitive Science

In this chapter, we will investigate the basic relationship between connectionism and cognitive science in order to establish a working framework for connectionism. To accomplish this, we will briefly introduce the field of cognitive science. This introduction will include the underlying assumption that the mind is an information processor, and that to explain any information processor requires an appeal to the tri-level hypothesis. Finally, we will define the three basic tenets of connectionism.

But, before we proceed with our investigation, we need to distinguish between the two different approaches towards connectionist research. As stated in Chapter 1, connectionism is used in many different fields of science. For example, connectionist networks have been used for aiding astronomical work (Storrie-Lombardi & Lahav, 1995), assisting medical diagnosis (Dawson, Dobbs, Hooper, McEwan, Triscott, & Cooney, 1994), regulating investment management (Zapranis & Refenes, 1995), and controlling robotic limb movement (Walter & Schulten, 1993). Many of these systems, however, are approached from an engineering perspective; that is, the designers are only interested in making the networks as efficient as possible (in terms of network topology, correct responses, and generalization). Consequently, this attitude towards connectionism could be characterized as the "engineering" approach. In fact, it may just be this approach that Reeke and

Edelman (1988, p. 144) had in mind when they offered this blunt assessment of connectionist research:

> These new approaches, the misleading label 'neural network computing' notwithstanding, draw their inspiration from statistical physics and engineering, not from biology.

Although the engineering approach to connectionist research is of interest and demands much attention, it is not the strategy taken here. The second approach—the cognitive science approach—is to use connectionism to answer questions pertaining to human cognition, from perceptual processes to "higher level" processes like attention and reasoning. That is, we are interested in drawing our inspiration from biology, not technology. Consequently, the goals of the engineering approach (e.g., minimizing network structure, improving generalization, etc.) are not necessarily those of the cognitive science approach to connectionism. To understand what these goals are, however, we need to understand what cognitive science is.

What is Cognitive Science?

> Cognitive science is defined principally by the set of problems it addresses and the set of tools it uses. The most immediate problem areas are representation of knowledge, language understanding, image understanding, question answering, inference, learning, problem solving, and planning . . . The tools of cognitive science consist of a set of analysis techniques and a set of theoretical formalisms. (Collins, 1977; p. 1).

The "birth" of cognitive science is often traced back to the Symposium on Information Theory held on September 10-12, 1956 at M.I.T. (Gardner, 1985). There, researchers from various disciplines gathered to exchange ideas on communication and the human sciences. Three talks in particular, Miller's *The magical number seven*, Chomsky's *Three models of language*, and Newell and Simon's *Logic theory machine*, have been singled out as instrumental in seeding the cognitive science movement. Following these talks, a perception began to emerge that "human experimental psychology, theoretical linguistics, and computer simulations of cognitive processes were all pieces of a larger whole" (Miller, 1979; p. 9; cited in Gardner, 1985, p. 29). That is, there arose a belief that to understand the functioning of human cognition, one had to combine the efforts of several different disciplines. In fact, similar sentiments had been expressed previously in the literature by such researchers as Hebb (1949) and Wiener (1948).

. . . It draws primarily on the component disciplines of Computer Science, Linguistics, Philosophy and Psychology and aims to provide both a sound theoretical understanding of human knowledge acquisition, representation and deployment, and practical experience in the interdisciplinary approach to contemporary topics such as expert systems, human computer interaction, natural language processing, machine vision, and parallel distributed processing. . .
University of Birmingham.

Cognitive science is an exciting and rapidly evolving field that deals with complex cognition, intelligent systems, and the emergent behavior of large-scale computational systems. It synthesizes aspects of a wide variety of disciplines, including psychology, computer science, linguistics, philosophy, and neuroscience.
Indiana University.

Cognitive Science is a rapidly expanding field of study aimed at understanding the mental processes that underlie cognitive abilities. . . Philosophers, Psychologists, Linguists, Neuroscientists and Computer Scientists have all approached the basic questions posed by the nature of mental processes in their own ways as part of the broader endeavours of their respective fields. Cognitive Science is distinguished from these traditional disciplines by its highly interdisciplinary approach. . .
University of Queensland.

Box 2.1 Sample descriptions of cognitive science programs.

. . . a proper explanation of these blank spaces on the map of science (can) only be made by a team of scientists, each a specialist in his own field but each possessing a thoroughly sound and trained acquaintance with the fields of his neighbors . . . (Wiener, 1948; p. 9)

As a formal discipline, cognitive science (originally termed *cognitive simulation*) first appeared in the latter half of the 1950's at what is now Carnegie Mellon University (Collins & Smith, 1988). Since then, cognitive science programs have been developed at many different universities throughout the world (e.g., University of Queensland, Australia; Queen's University, Canada; Roskilde University; Denmark; University of Hamburg, Germany; University of Birmingham, U. K.; Indiana University, U. S. A.). A survey of program descriptions (see Box 2.1) reveals both the interdisciplinary nature of the discipline and the types of problems studied.

So, then, what is cognitive science? Cognitive science can be defined as the interdisciplinary study of mind; It draws upon such diverse fields as Computing Science and Artificial Intelligence (e.g., Collins & Smith, 1988), Linguistics (e.g., Osherson & Lasnik, 1990), Neuroscience (e.g., Posner, 1993), Philosophy (e.g., Leiber, 1991), and Psychology (e.g., Gardner, 1985), to name but a few. Although each discipline has its own unique interpretation of cognitive science, they are bound into a cohesive whole by a central tenet. This tenet states that the mind is an information processor; that is, it "receives, stores, retrieves, transforms, and transmits information" (Stillings et al., 1987, p. 1). This information and the corresponding information processes can be studied as patterns and manipulations of patterns. Furthermore, these processes posit representational or semantic states that are fully realized within the physical constraints of the brain:

> Modern psychology takes completely for granted that behavior and neural function are perfectly correlated, that one is completely caused by the other. There is no separate soul or life-force to stick a finger into the brain now and then and make neural cells do what they would not otherwise. (Hebb, 1949, p. xiii)

Finally, cognitive science is governed by scientific methodology, specifically, the agendas of collecting data, comparing data with theory, and evaluating and formulating theory.

The Information Processing Approach

The information processing approach to cognition developed during World War II as an amalgamation of human factors work and information theory[1] (Anderson, 1985). Its popularity as a theory of cognition arose for two reasons. First, there was a revolt against the hard-line behavioural view (e.g., Watson, 1913) that appealing to mental states when describing observable behavior was unnecessary and undesirable. In contrast, the theory of feed-back control being developed in cybernetics (e.g., Wiener, 1948) once again made it acceptable for scientists to assign teleological explanations to internal mental states. Second, the advent of the modern computer gave researchers a working metaphor for human cognition. Thus, theories of cognition based on the information processing approach could be implemented easily on computers, thereby providing existence proofs that were previously unattainable.

The principles underlying the information processing approach today can be characterized as follows (Best, 1995):

1. *Informational Description*—both environment and mental processes can be described by the amounts and types of information they contain.

[1] Human factors work is concerned with human/machine interactions while information theory focuses on how information is best sent from a source to a receiver. Early work in information theory is attributed to Claude Shannon who proved the *Noisy Coding Theorem* in 1948 which states that as long as we are willing to settle for a rate of transmission below channel capacity, there is an encoding scheme that will reduce the probability of an error to any desired level (Roman, 1992).

2. *Recursive Decomposition*—cognitive processes can be broken into simpler cognitive processes (which can also be broken into simpler processes, etc.) to produce an hierarchy of cognitive processes.
3. *Flow Continuity*—information goes forward in time; whatever inputs are needed for a certain cognitive process can be found in the outputs of the cognitive processes that feed into it.
4. *Flow Dynamics*—mental processes take time (they do not occur instantaneously) because they are coexistent with the chemical and electrical properties of the brain.
5. *Physical Embodiment*—all cognitive processes take place in a physical system (e.g., neural in the case of humans; silicon in the case of computers). This implies that the knowledge we have must be stored in the physical system in the form of "representations".

This "classical" approach to cognitive science focuses on the notion that patterns of information can be represented as symbols, and that these symbols can be manipulated. This approach is typified by what Newell and Simon (1981, p.41) call the physical symbol system hypothesis:

> *The Physical Symbol System Hypothesis.* A physical symbol system has the necessary and sufficient means for general intelligent action.

Operations on these symbols take place in a serial manner, proceeding from one unit (e.g., Miller, Galanter, & Pribram, 1960), agent (e.g., Minsky, 1985), or module (e.g., Fodor, 1983) to another. Adopting this classical approach to cognition leads researchers to develop cognitive models that often resemble flowcharts (Anderson, 1985); that is, the models are comprised of "black boxes" that compute some function, and these boxes are connected to each other through a series of arrows which represent flow of control. This approach can be seen in Figure 2.1 which illustrates the Test-Operate-Test-Exit (TOTE) unit of Miller, Galanter, and Pribram (1960).

How does a TOTE unit (Miller et al., 1960) fit into the classical information processing approach to cognition? The unit receives as its input information about the internal or external world. This information is compared to an Image, which "is all the accumulated, organized knowledge that the organism has about itself and its world" (p. 17). If the information is congruent with the Image, then the unit exits. If, however, the information is incongruent with the Image, then

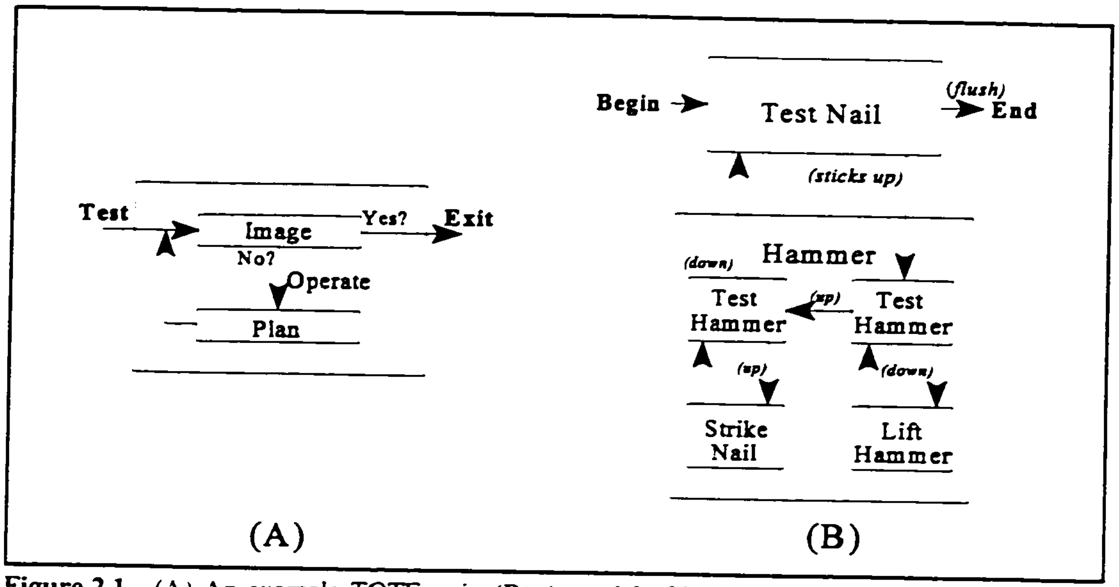

Figure 2.1 (A) An example TOTE unit. (B) A model of hammering a nail using TOTE units (Adapted from Miller, Galanter, & Pribram, 1960).

the information is operated on by a Plan. "*A Plan is any hierarchical process in the organism that can control the order in which a sequence of operations is to be performed*" (p. 16; their italics). This new information produced by the Plan is then tested against the Image again, and the process repeats, with each cycle taking a fixed amount of time. Therefore, a TOTE unit (i) receives information from the internal and external world, (ii) has a hierarchy of processes, (iii) processes information in a serial manner, (iv) requires time to complete a task, and (v) represents its knowledge of the world in an Image.

Although the classical approach to human information processing is widely accepted today, it is sometimes regarded as too strict and formal when compared with actual cognitive performance. For example, Norman (1986) lists several properties of human cognition that are handled poorly by traditional information processing theories. Humans are constantly recalling past experiences, learning from new experiences, making decisions based upon incomplete data, and committing (and recovering from) errors. Therefore, in order to model human cognition, information processing systems must be (i) robust yet flexible and creative, (ii) relatively insensitive to missing or erroneous data, and (iii) capable of sustaining damage to its parts. Thus, in addition to the five properties cited by Best (1995), an information processing system should also have these essential properties:

1. *Graceful Degradation*—performance should decline gradually with both impoverished data and damage to the system,

2. *Content-Addressable Memory*—information should be retrieved through (partial) descriptions of the data as opposed to absolute addressing or random access often assumed in classical models,
3. *Output Availability*—output of the system should be available continuously (i.e., the current state of a system provides continuous information about the current evaluation of an hypothesis) as opposed to stage models of information processing, and
4. *Iterative Retrieval*—retrieval should be an iterative process based on description where there is competition among alternative interpretations rather than the more traditional search techniques.

Norman (1986) has noted that connectionism can account for these essential properties of human information processing. Hence, connectionism has often been touted as an alternative to the classical approach to information processing (e.g., Bechtel & Abrahamsen, 1991; Best, 1995; but see Boden, 1988). Although both classical and connectionist researchers agree that information is being processed, they disagree on *how* it is processed. For example, classical systems use explicit, often logical, rules arranged in an hierarchy to manipulate symbols in a serial manner. Connectionist systems, on the other hand, rely on parallel processing of sub-symbols, using statistical properties instead of logical rules to transform information.

But, are these distinctions valid? Many of the *perceived* differences between classical and connectionist models are due to the working assumptions of both approaches. If, however, we use the classical working definition of an information processor, the only major difference between the two approaches is in the recursive decomposition of processes into a hierarchy (although a multi-layered network could be interpreted as a hierarchy of processes—albeit a potentially difficult hierarchy to interpret). Classical researchers adopt what Braitenberg (1984) has termed the *analytical* approach to information processing; that is, start at the highest concept (e.g., memory) and break it into smaller concepts (e.g., episodic and semantic memory; Tulving, 1972). Connectionist researchers, however, typically adopt a *synthetic* approach to information processing; that is, start with the smallest units (e.g., artificial neurons) and attempt to build a larger system from them (e.g., distributed memory model; McClelland & Rumelhart, 1985, 1986). In the end, however, both approaches describe memory from an information processing vantage. Thus, this distinction between the working assumptions of classical and connectionist researchers does not mean that they are speaking different languages, just different dialects.

As an aside, it must be recognized that there is some opposition to the information processing approach, whether classical or connectionist in nature (Bunge, & Ardila, 1987). Most major objections centre on the fact that the information processing approach reduces humans to mere computers, responding along fixed guidelines with no room for self-determination. Similar to the behaviourist position of avoiding teleological explanations for behaviour, the information processing account of cognition cannot easily account for such things as emotions, dreams, and motives (e.g., Gray, 1994[2]).

The Turing Machine Metaphor

The above description of an information processor is a rather loose set of guidelines in determining what sorts of properties are needed for a system to be characterized as engaging in information processing. In more formal terms, an information processor can be fully described in terms of a Turing Machine (Turing, 1936). Like many mathematicians of the time, Alan Turing was interested in trying to find an answer to Hilbert's *Entscheidungsproblem*—the problem of mathematical decidability. In order to determine if there was a mechanism for determining the truth or falsehood of any mathematical statement, Turing required a mechanism that was both powerful enough to handle any mathematical statement, and yet simple enough to leave no question as to its claims of decidability. Therefore, Turing developed a hypothetical machine that was functionally simple in configuration, yet still capable of performing complicated information processing tasks. A Turing Machine (TM) is the simplest mechanism that characterizes those aspects of an information processor that are both necessary and sufficient. In fact, all attempts so far to develop other information processing devices have proved reducible to some form of TM (Rogers, 1987; Johnson-Laird, 1988; Bridges, 1994).

> *By means of detailed combinatorial studies . . . the proposed characterizations of Turing and of Kleene, as well as those of Church, Post, Markov, and certain others, were all shown to be equivalent.* (Rogers, 1987, p. 18; his italics).

[2] But see LeDoux and Fellous (1995) for a computational perspective on emotions, Antrobus (1993) for a connectionist model of dreaming, and Smith (1996) for a connectionist account of motivation.

There is a quick caveat to make here. TM's can be enumerated only in correspondence with the integers. Therefore, there is a countably infinite number of computable functions that can be carried out by a TM. The set of all functions, however, can be put in correspondence with the real numbers—an uncountable infinity. Since a TM cannot enumerate the set of reals, there is an uncountably infinite number of functions that cannot be computed (Ballard, 1997). Because it is assumed that the output units of artificial neural networks can adopt any real number value (although this is often transformed to fall within a specific range), it could be argued that some types of neural networks are in fact more powerful than a TM[3].

A Turing Machine consists of two main parts: an infinitely long ticker tape, and a machine capable of manipulating the contents of the tape (see Figure 2.2). The ticker tape is divided into cells, with each cell containing either a symbol or a blank. These cells can then be grouped into areas, such as questions for the machine to solve, scratch pads for doing work, and even instructions

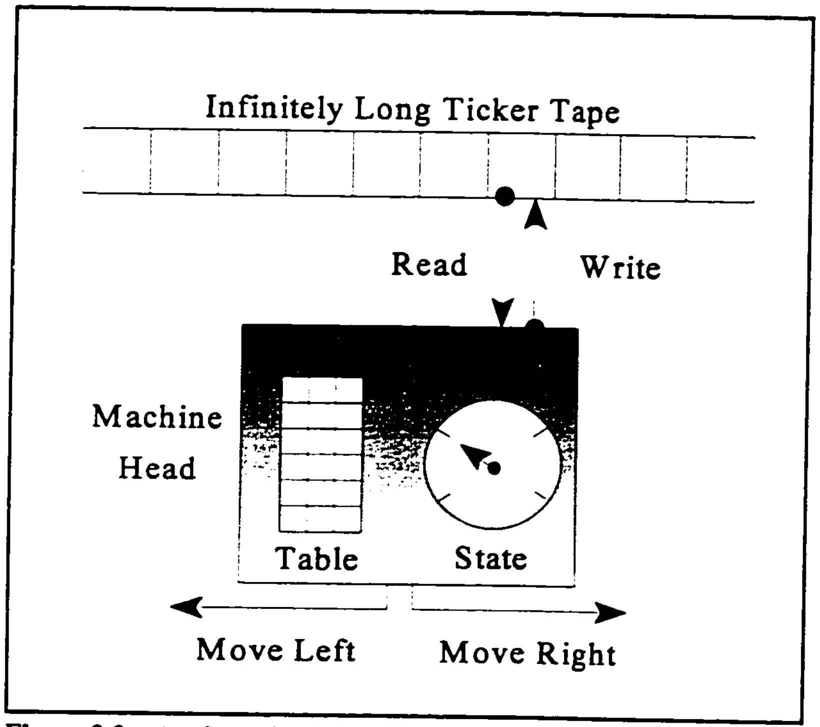

Figure 2.2 A schematic illustration of a Turing Machine.

[3] In fact, this statement (or variations on it) has been used to argue that connectionist networks are too powerful and therefore tell us nothing interesting about cognitive processes. If, however, connectionist networks are truly modeled on brain functioning, then they are not any more powerful than a TM. For example, if we take the firing of neurons to be an all-or-none process *à la* McCulloch & Pitts, then neurons are binary and the real-valued activations adopted by connectionist networks are only approximations to this binary encoding (e.g., approximating neural spike frequency or number of synaptic connections).

for instantiating other TMs. The ticker-tape therefore contains all the information that the machine processes.

The machine, itself, is composed of four components: the machine head, the machine state, the machine table, and a mechanism for moving along the tape one cell either left or right. Symbols on the tape are manipulated by the machine head—it both reads from the tape and writes to the tape. Once a symbol is read from the tape, the current machine state is ascertained. The machine can only be in one of a number of finite states at any time. Both the symbol read from the tape and the current machine state are then used as indices to access the machine table. The table contains simple instructions (such as write a blank to the tape, move one space left, and adopt state 5) for the machine to carry out. At the completion of the instructions, the machine reads the next symbol on the tape and the process repeats itself until the machine comes to an end state (i.e., solves the problem posed). If a problem is computable, then the TM is guaranteed to stop.

Although different Turing Machines can be built for different information processing problems, the real power behind the TM as a metaphor for information processing comes from the Universal Turing Machine (UTM). The UTM is a machine capable of reading the description of another TM from the ticker tape (in effect, reading the machine table of a particular TM), and then carrying out those instructions as if it were that TM. What this means is that a UTM can solve *any* information processing problem that is computable! Furthermore, the machine table of a UTM will be those base instructions required for building all other machine tables.

A Turing Machine—and therefore an information processor—can be described at three different levels. First, it can be described in terms of what problem the machine is solving (i.e., the question on the tape). Second, the steps that a particular instantiation of a TM requires to solve a problem (i.e., the machine state and table) can be described. Related to this, we can also ask what basic instructions are required to build a particular instantiation of a machine (i.e., the machine table of a UTM). Third, we can describe the physical characteristics of the (hypothetical) machine. This may seem like an odd move since it has been argued that the TM embodies all characteristics of an information processor without appealing to the underlying physical mechanisms (e.g., Johnson-Laird, 1988). But, if it is impossible to physically construct a portion of a TM (such as the read/write heads), then that particular TM is moot no matter how well the tape, machine state and table are constructed.

The Tri-Level Hypothesis

We have stated that information processing can be formally characterized in terms of a Turing Machine, and that the operations of a TM can be described at three different autonomous levels. It is not surprising, therefore, that it has been argued (e.g., Marr, 1982; Pylyshyn, 1984; Rumelhart & McClelland, 1985; Johnson-Laird, 1988; Horgan & Tienson, 1996; Dawson, 1998) that for any information processing system to be understood completely, it must be described at three different levels of analysis.

> Almost never can a complex system of any kind be understood as a simple extrapolation from the properties of its elementary components . . . If one hopes to achieve a full understanding of a system . . . then one must be prepared to contemplate different kinds of explanation at different levels of description that are linked, at least in principle, into a cohesive whole, even if linking the levels in complete detail is impractical. (Marr, 1982, pp. 19-20)

Marr (1982) has defined these three levels as computational, algorithmic, and implementational (see Figure 2.3). The computational level is a description of what information processing problem is being solved by the system. The algorithmic level is a description of what steps are being carried out to solve the problem. The implementational level is a description of the physical characteristics of the information processing system. There is a one-to-many mapping from the computational level to the algorithmic level, and a one-to-many mapping from the algorithmic

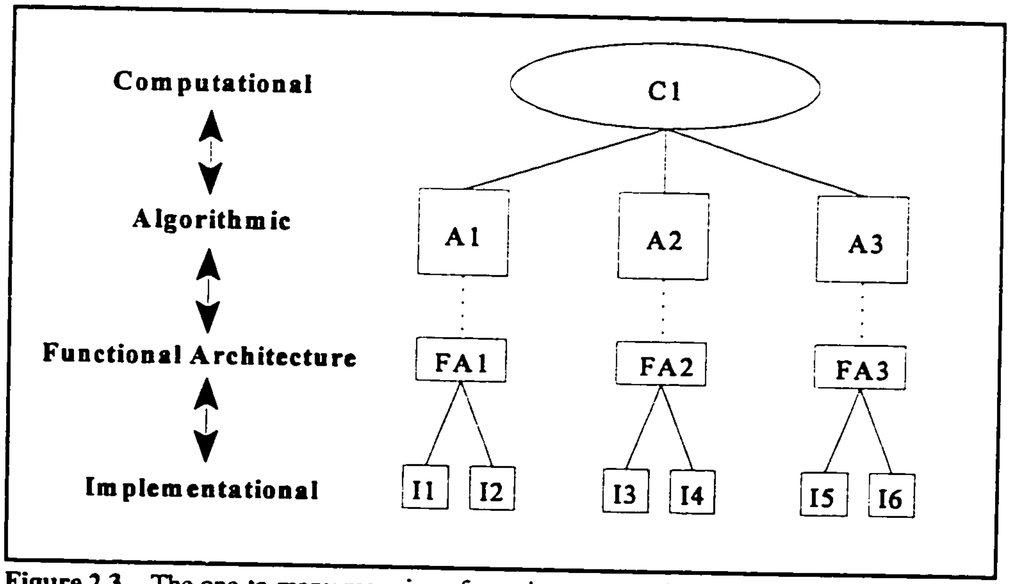

Figure 2.3 The one-to-many mappings from the computational level to the implementational level of the tri-level hypothesis. The double-headed arrows indicate that the levels influence each other.

level to the implementational level. In other words, there is one computational description of a particular information processing problem, many different algorithms or steps for solving that problem, and many different ways in which a particular algorithm can be physically implemented. Furthermore, the inter-level relations are transitive; that is, if the computational level description is realized by some algorithmic level description (C–A), and that algorithmic level description is in turn realized by some implementational level description (A–I), then the computational level description is realized by the implementational level description (C–I). Note that this does not mean that the algorithmic level description can be disregarded. What it does mean, however, is that the mind has some physical realization; that is, the mind and brain are equivalent and we can consequently dispose of Descartes' mind-body dualism.

Although agreeing with Marr's general framework, others have defined the three levels in slightly different terms. For example, Schneider (1987) talks of the computational, cognitive, and physiological levels. Best (1995) describes them as the mental, cognitive, and neural levels. Johnson-Laird (1988; p. 58) maintains that we need a theory of "*what* is computed", "*how* the system carries out the computations", and the "underlying neurophysiology." Green (1996), refers to the three levels as behavioral (i.e., a person's performance), cognitive (i.e., cognitive system underlying behavior), and biological (i.e., nature of the brain systems mediating cognition). Similarly, Horgan and Tienson (1996) distinguish between the levels of cognitive-transition functions[4], representation and algorithm, and hardware realization or implementation. Regardless of the terminology used, the underlying concepts of a computational, algorithmic, and implementational description remain the same.

Finally, Pylyshyn (1984) also characterizes an information processor as having three different levels of description. Each level has considerable autonomy so that regularities at one level can be described to a first approximation without appealing to the regularities expressed at other levels. Furthermore, the three levels are tied together in an instantiation hierarchy, with each level instantiating the one above. According to Pylyshyn, a cognitive system should first be described at the biological or physical level—if all behavioural regularities can be accounted for here, then we need go no further in our explanation. The next step is to explain a system in terms of the symbols

[4] Horgan and Tienson (1996) take exception with Marr's term of "computational" as they suggest that it refers to 'how' something is computed (which is actually described at the algorithmic level). This is a needless distinction as Marr clearly indicates that the computational level is a description of the competence of the system, not how the system is computing.

and syntax that it uses; in other words, in terms of what program the system is executing. Finally, if regularities still exist that cannot be explained at either the physical level or the symbol level, then we must appeal to a semantic level. The semantic level deals with the content of thought (our beliefs, goals, etc.). Although the syntactic and semantic levels roughly correspond to the algorithmic and computational levels respectively, Pylyshyn's equivalent to the implementational level does not place as much emphasis on actual biology. If there is no known biological mechanism to account for a certain cognitive phenomenon, Pylyshyn is comfortable in proposing some hypothetical construct to fill the void. Pylyshyn calls this final level the functional architecture of a system. It is the basic building blocks from which the symbols and syntax are constructed. For our purposes, however, we will use the functional architecture as a bridge between Marr's algorithmic level and implementational level. In the formal terms of a Turing Machine, the functional architecture could be thought of as the machine table of an UTM.

The working assumption that connectionist models are information processors therefore requires them to be explained at these three different levels, including the functional architecture. In fact, it has been argued (e.g., Rumelhart & McClelland, 1985; Dawson & Shamanski, 1994; Dawson, 1998) that for connectionism to contribute significantly to cognitive science, it must be described at these different levels. Furthermore, Schneider (1987) suggests that connectionism may be the very tool required for relating the three levels of analysis.

> The connectionist framework suggests that we might be able to connect the computational, cognitive, and physiological levels of analysis and to do so with a conceptually very simple system. (p. 74)

The Computational Level

The computational level describes both what problem is being solved by the information processing system, and why the problem is being solved. It is a statement of the system's competence and defines the types of functions (e.g., $f{:}X \rightarrow Y$) the system can compute. In other words, the computational level is simply a description of the input-output behaviour of a particular system. Traditionally, analysis at this level has been the domain of philosophers, behavioural psychologists, linguists, and computer scientists.

For example, a generalized question at the computational level might take the form "What information processing problem P is being solved by the mapping of function $f{:}X \rightarrow Y$, and why can it not be characterized by the function $g{:}X \rightarrow Y$?" In answering these types of questions, we are seeking the goals and constraints that underlie the solutions to information processing problems. We are therefore developing a theory of competence of a particular system (Richards, 1988). By considering the goals of an information processing system—specifically how to capture the properties and events of the world that are significant to the information processing system at hand—we develop theories that show how a reliable and accurate representation of the world can be computed.

But, is a computational description by itself sufficient for describing an information processing system? When Thorndike (1932; p. 1) states "We are concerned . . . with the fundamental facts of learning whereby a situation which first evokes response A later evokes response B, different from A" he is concerned with finding those aspects of the environment, such as repetition and belongingness, that are crucial for learning. Thorndike describes learning as the change in the connection strength between S_1-R_1, and S_2-R_1, where connection strength is determined only by the probability of S_1 or S_2 evoking R_1. There is no appeal to underlying processes in Thorndike's description, whether at an abstract algorithmic level or at a more concrete physiological level.

It is therefore possible that two or more information processing systems could have the same computational level description. That is, in some respect the behavioural repertoire of system **A** is identical to the behavioural repertoire of system **B** in that same respect. When taken by itself, this type of equivalence has been termed weak equivalence (Fodor, 1968), or "Turing" equivalence (Pylyshyn, 1984) after the imitation game proposed by Turing (1950). The problem with weak equivalence, though, is that while it describes what problem the system is solving, it offers no explanation of how the system is solving the problem. Therefore, one cannot be justified in using system **A** as a model of "how" system **B** is solving a specific problem. Although some may argue that weak equivalence is sufficient (e.g., Penrose, 1989), the goal of cognitive science is not only to describe intelligent behaviour, but also to explain how it comes about. Therefore, cognitive science requires more than weak equivalence!

The Algorithmic Level

The algorithmic level of analysis focuses on the specific steps (or algorithms) employed to solve the problem under consideration. In particular, the algorithmic level is concerned with how the input and output of the system are represented, and how input is transformed into output. The algorithmic level is most often associated with cognitive psychology and psycholinguistics.

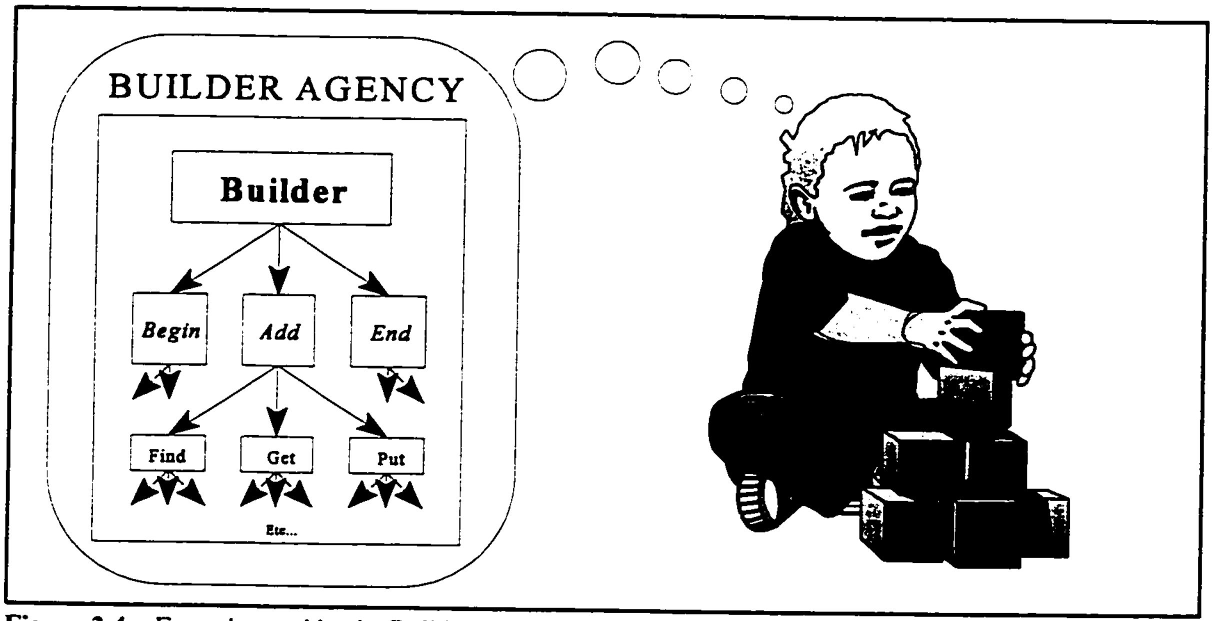

Figure 2.4 From the outside, the Builder Agency is a computational description of building a tower. From the inside, the individual agents describe the algorithmic account of how to build a tower.

How does one go about describing an information processor at the algorithmic level? The most common approach from a cognitive psychology perspective is to identify the overall problem and then break this into subgoals, which in turn can be broken into subgoals, and so forth. Cummins (1983) has described this process as *functional analysis*. For example, Minsky (1985) describes the problem of a child playing with blocks. In order to build a tower from the blocks, the child employs an agent known as *Builder*. But, building a tower is too complicated for any one agent, so *Builder* employs other agents such as *Begin*, *Add*, and *End*. *Add* also employs other agents such as *Find*, *Get*, *Put*, and so on (see Figure 2.4). From the outside, *Builder* is an agency that can be described at the computational level of analysis—it builds a tower. Taken from the inside, however, *Builder* is an agent that calls upon simpler and simpler agents to solve a particular problem. By following these simple agents and the links between them, we can describe how the task of building a tower is accomplished.

According to Fodor (1968), once we have a description of how a particular system is solving a problem, we are in a position to make claims about strong equivalence. We can say that system **A** is strongly equivalent to system **B** in some respect when **A** is weakly equivalent to **B** in that respect *and* the processes upon which the behaviour of A are contingent are of the same type as the processes upon which the behaviour of **B** are contingent. One is now justified in using system **A** as

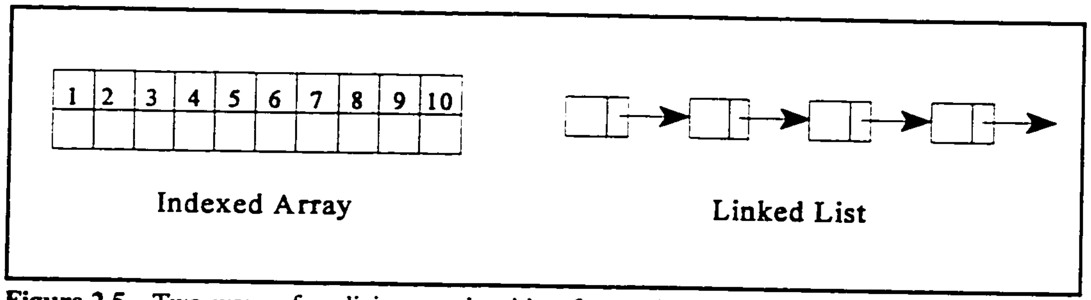

Figure 2.5 Two ways of realizing an algorithm for storing items in memory: an indexed array and a linked list (see Tremblay & Sorenson, 1984).

a model of how system **B** is solving a specific problem. Although Fodor's description of strong equivalence is sufficient for describing systems in general, I believe that it is inadequate for explaining cognitive systems in particular. I am more inclined to call this relationship medium equivalence. Two reasons for this belief will be elaborated.

First, imagine that you had to store a list of items in memory for later serial recall. The algorithm for this would be simple. Initially, define an index which accesses some memory location. To store memory, use the index to locate an empty memory slot, place the item within the memory location, then increase the index to store the next item. To recall memory, start the index at the beginning of the list, retrieve the item from the memory location, and then increment the index to recall the next item. This simply describes a general procedure for accomplishing the serial recall task[5], it does not state the specifics of carrying out the algorithm. For example, two ways of realizing this algorithm would be to use either an indexed array or a linked list—as long as the problem was fairly simple and straightforward, there would be little difference in using one approach over another (see Figure 2.5). Both approaches allow the easy storage and retrieval of serial information (a computational level description) and, consequently, Fodor would call these strongly equivalent systems.

Whereas the algorithm is a procedural description, the choice of an indexed array or a linked list is an architectural distinction. In terms of the TM metaphor, the procedure is the program on the tape while the architecture is the machine table. It is important to make this distinction because the choice of architecture will often constrain the competence of the system as defined at the computational level description. Although both architectures described above are equivalent in terms

[5] Although the serial recall of a list of items is a classical cognitive psychology experiment (see Norman, 1982; Best, 1995), in no way do I actually want to imply that cognitive systems use this particular procedure for performing serial recall.

of this simple algorithm, the architectural decisions may very well pose problems if we extended the task slightly. For example, if we required random access to the list without any increase in access time, then the indexed array would be the correct architectural move. If, however, we required dynamic memory storage then the linked list would be the correct choice. Consequently, our description at the algorithmic level requires not only a description of the procedure, but also the architecture. A description of the procedure alone only results in medium equivalence.

Second—and in a slightly different vein—the recursive decomposition at the algorithmic level leads to a rather unique problem. How does one know when to stop breaking goals into sub-goals? In other words, when does one stop appealing to simpler and simpler agents? In the above example, the *Builder* agency has the intention of building a block tower, but it cannot do this without the *Builder* agent which calls upon other agents. These sub-agents themselves can be seen as agencies that have the intention of performing a certain task such as *Add*. This problem is akin to the homonculus problem (Ryle, 1949) and, hence, has been termed *Ryle's regress* by Stillings et al. (1987): to explain the intentionality of one agent, you have to appeal to the intentionality of another agent, ad infinitum.

Where, then, does this appeal to intentionality end? In other words, how do we banish the ghost from the machine? One possible move to make is to stop the recursive decomposition when (i) the function of an agent can be realized physically or, (ii) further decomposition is impossible even when physical constraints are unknown—thus, the primitive level would possess the basis of intentionality[6] (Pylyshyn, 1991). Making this move leads us to the functional architecture of an information processing system.

The Functional Architecture

At the basis of the algorithmic level is the *functional architecture*; it is composed of those aspects of an information processing system that cannot be broken into further functional units. Pylyshyn (1984) defines the functional architecture as

> those basic information-processing mechanisms of the system for which a nonrepresentational or nonsemantic account is sufficient. The operation of the functional architecture might be explained in physical or biological terms, or it

[6] This basis of intentionality is similar to what Jackendoff (1992) refers to as "I-concepts"—those base concepts that all of thought is generated from.

might simply be characterized in functional terms when the relevant biological mechanisms are not known. (p. xvi)

In other words, the functional architecture is the programming language—or architecture—from which the algorithmic level is built. It can be viewed as equivalent to the machine table of a UTM, as it contains those base instructions from which all other TMs are built.

Consequently, if we have equivalent computational descriptions and equivalent algorithmic descriptions *including the functional architecture*, then—and only then—can we make claims of strong equivalence. And, cognitive science is in the business of providing strong equivalence!

Beyond making claims of strong equivalence, why do we actually need the functional architecture? As stated earlier, at the computational level and even the algorithmic level we often refer to states of intentionality, appealing to conscious decisions to guide our behaviour. Consequently, if we were to leave our description here before moving onto the implementational level, it would be too tempting to make Descartes' error of separating the conscious mind from the body—sticking the proverbial finger into the brain and making a neural cell do what it otherwise would not. Moving from the algorithmic level to the implementational level would thus become some magical step that endowed intentionality to the system. The functional architecture, being composed of those processes not dependent on a representational or semantic interpretation, allows us to banish the ghost from the machine.

In fact, the functional architecture may be the most important step in describing an information processor in that it provides the needed link between the algorithmic level and the implementational level. Not only must algorithms be built from the functional architecture, but the functional architecture must be implemented in some biological or mechanical form. Consequently, if we can identify the functional architecture of some biological system (say a human), and we can identify the same functional architecture in some artificial system (say a computer), then we can make claims of equivalent competency with regards to information processing—independent of implementation! Furthermore, if some functional architecture leads to intentionality in a biological system, then that same functional architecture will lead to intentionality in an artificial system as well (but see Searle, 1980, 1990).

What makes the functional architecture so powerful in comparing information processing systems is Fodor's (1968) notion of functional equivalence. Although two systems may be radically different in their composition (e.g., one based in carbon and the other in silicon), we can equate the two functionally if, in the long run, they produce the same state of affairs and their effects are indistinguishable. Formally, Fodor (1968) defines functional equivalence as follows:

> if m_1 and m_2 are functionally equivalent mechanisms, and if $e_1, e_2, \ldots e_n$ are the series of effects of m_1, then somewhere in that series there must be an e_i such that $e_i \ldots e_n$ are effects of m_2. (p. xviii)

It is important to note that the above definition does not require that m_1 and m_2 share each and every property, as that would imply that m_1 and m_2 were the same mechanism. Instead, all that is required is that m_1 and m_2 share those functional properties pertinent to the information processing task at hand.

Today, the most promising research at the functional architecture level is being conducted within the emerging field of cognitive neuroscience (e.g., Gazzaniga, 1995). A survey of this literature shows that cognitive neuroscientists use a variety of techniques such as brain imaging (e.g., Roland, Kawashima, Gulyás, & O'Sullivan, 1995), lesioning studies (e.g., Rapp & Caramazza, 1995), and even neural network simulations (e.g., Andersen, 1995) to answer questions at this level.

The Implementational Level

The implementational level of analysis is an account of the physical structure of an information processing system. It attempts to relate the representational or semantic states of an information processing system to the causal laws governing its physical structure. In other words, it describes how an algorithm can be realized physically. Research at this level is conducted mainly by neuroscientists.

As stated earlier, the implementational level is an integral part of the tri-level hypothesis and, thus, is required for a complete explanation of any information processor. No matter how well thought out and investigated a researcher believes the computational and algorithmic levels are, if the underlying physiology cannot support the other two levels, then the beliefs of the researcher must be in error. Unfortunately, it is at this level that traditional cognitive science breaks down.

For example, in their attack on connectionism, Fodor and Pylyshyn (1988) assert that classical theories are committed to the fact that "neurons implement *all* cognitive processes . . . by supporting the basic operations that are required for symbol processing" (p. 11; their italics). Furthermore, they state that

Clearly it is pointless to ask whether one should or shouldn't do cognitive science by studying "the interaction of lower levels" as opposed to studying processes at the cognitive level since we surely have to do *both*. (p. 66; their italics).

Despite these assertions, Fodor and Pylyshyn are never specific about how neurons or neural processes may implement cognitive processes. In fact, they claim that implementation is irrelevant to psychological theory! As long as one has a theory of the origins of some empirical phenomenon (e.g., graceful degradation) then implementational accounts can be ignored in regards to cognition.

Fortunately, not all cognitive scientists hold this view. Clark (1989, p. 61) contends that "no serious study of the mind (including philosophical ones) can, I believe, be conducted in the kind of biological vacuum to which cognitive scientists have become accustomed." Furthermore, the appeal of *Parallel Distributed Processing* (PDP) models is based on their obvious "physiological" flavour: Connectionist models are more closely tied to the physiology of the brain than any other information processing models (McClelland, Rumelhart, & Hinton, 1986).

Describing an information processor at the implementational level is a necessary but not sufficient step. It does not nullify or replace the other two levels, it simply rounds out the explanation of the system in question. As stated repeatedly, all three levels of analysis are required to understand any information processor—the implementational level constrains the types of algorithms that can be executed, which in turn constrains our computational descriptions.

The Logic behind Computer Simulations

Computers and the human brain are radically different. Nevertheless, the information processing approach to cognition has developed and taken hold as a direct result of computational theory (Turing, 1936) and the invention of computers (von Neumann, 1958). In fact, classical theories are not only likened to computer programs, but it has been stated that "theories of mind should be expressed in a form that can be modeled in a computer program" (Johnson-Laird, 1988, p. 52). Although connectionist theories of computation have their basis in neurophysiology (and not engineering principles as classical theories do), connectionist models are still implemented on computers. Hence, some justification for using computers to simulate human cognition is required.

If computers and the human brain are so radically different, how, then, can we relate the functioning of a computer to the functioning of the human mind? Lorenz (1974) outlines two different scientific methods of description when the task of explaining the original system (in our case, the mind) is problematic. The first method is explanation by homology. Homologous systems are those systems that have similarities caused by common descent. For example, one can compare the fifth digit in the human hand, the bat wing, and the whale flipper (see figure 3 in Lorenz, 1974). All digits are homologous because they share a common evolutionary link; unfortunately, understanding the function of the bat's or the whale's digit does not explain the function of the human's digit. Therefore, comparison by homology runs the risk of misrepresenting the function of the analyzed system.

The second method is explanation by analogy. Analogous systems are systems that perform the same function, but have developed through independent means, or parallel adaptation. Hence, understanding the function of the bird's wing allows us to understand the function of the bat's wing, even though they both developed under differing circumstances. Furthermore, we can apply our understanding about the principles of aerodynamics and lift from the bird and bat wing to create the artificial "wings" of aircraft. Therefore, analogy allows us to explain the similarity of functions of two or more independent systems, regardless of how the systems developed.

For years, homologous explanations via animal models (e.g., Watson, 1913; Kupfermann, Castelluci, Pinsker, & Kandell, 1970; Gallistel, 1995) have been used under the assumption that the knowledge gained from the simpler animal would generalize to the more complex human. But, not only are these models constrained to the very simplest aspects of human information processing, there is also the risk of misrepresenting functional descriptions.

With the advent of modern computing devices, however, more scientists are using explanation by analogy via computer models to understand human information processing (e.g., von Neumann, 1958; Newell & Simon, 1963; Quillian, 1968; Minsky, 1975; Marr, 1982; Quinlan, 1986). Such an approach has advantages over animal models because we can (i) model (hypothetical) internal states often precluded from animal research, (ii) simplify, via functional equivalence, certain elements of the brain (i.e., neurons and connections), (iii) study higher level cognitive processes such as language that appear uniquely human, and (iv) perform precise lesioning experiments to understand functional roles of large groups of processing elements.

Of course, computer models of cognition have always met with resistance from various sources. In fact, there appears to be some confusion over the exact nature of computer simulations. On the one hand, Kukla (1989) argues that computer simulations of cognition are purely theoretical. Programs are merely formal statements of a theory, and computers are just a method of implementing our theories quickly and efficiently. On the other hand, McCloskey (1991) argues that although a computational device (e.g., a connectionist network) may reproduce certain aspects of human performance, the ability to do so does not qualify the program as a theory, nor does it amount to explaining the performance.

Furthermore, Lewandowsky (1993) warns of several hazards inherent to computer simulations. First, there may by inadvertent discrepancies between the intended specifications of a verbal theory and the pragmatic decisions concerning computer code resulting in a simulation that no longer speaks to the properties of the theory. Second, simulation results may not be due to the fundamental properties of a theory, but to mismatches between real-world situations and the simulated environments (for example, see Pinker & Prince's [1988] critique of Rumelhart & McClelland [1986]). Third, the simulations themselves may be no more easier to understand than the real-world processes they purport to explain[7]. Although Lewandowsky's hazards may be considered more as warnings for prospective modelers, others have been more damning in their attacks against computer simulations.

For example, Searle (1980, 1990) has argued that computer programs lack intentionality and therefore have little to tell us about thinking. Only machines (and not programs!) with the equivalent internal causal powers of the brain can be considered minds. In response to this, Churchland and Churchland (1990, p. 37) state:

> We presume that Searle is not claiming that a successful artificial mind must have all the causal properties of the brain, such as the power to smell bad when rotting, to harbor slow viruses such as kuru, to stain yellow with horseradish peroxidase and so forth. Requiring perfect parity would be like requiring that an artificial flying device lay eggs.

Despite their apparent disagreement over the ability to create a thinking machine, both Searle and the Churchland's agree that the key to any successful model of cognition is going to be dependent on correctly stating the functional architecture of a system. The Churchland's believe that the correct functional architecture does not lie in the traditional symbol manipulation machines of classical cognitive science, but in machines with a more brainlike architecture; that is, connectionist machines.

[7] Dutton and Starbuck (1971) call this *Bonini's paradox* after Charles Bonini (e.g., 1963) who found that his computer models of business organizations were sometimes more difficult to understand than the real-world problems that they were suppose to model.

Therefore, if the elements within our computer simulation are functionally equivalent to the description of the nervous system (i.e., have the same functional architecture), then we are justified in using a computer as a modelling tool. To borrow from Fodor (1968):

> Our intention is that, in this machine, the relays will play the role of neurons: . . . the relays should fulfill whatever functions the neuron fulfills *in vivo*—that the machine's relay should be functionally equivalent to the organism's neuron. (p. xviii, his italics)

Connectionism Defined

Finally, we are left with the definition of connectionism itself as studied within the field of cognitive science. Connectionism is a theory of information processing. It is based on the known neurophysiology of the brain, and attempts to incorporate those functional properties thought to be required for cognition. This is in contrast to the classical approach which assumes that the architecture of the mind *is* the architecture of von Neumann style computers (Pylyshyn, 1984).

What, then, are the functional properties of the brain that are required for information processing? Connectionists adopt the view that the basic building block of the brain is the neuron. The neuron has six basic functional properties (Dudai, 1989). It is an input device receiving signals from the environment or other neurons. It is an integrative device integrating and manipulating the input. It is a conductive device conducting the integrated information over distances. It is an output device sending information to other neurons or cells. It is a computational device mapping one type of information into another. And, it is a representational device subserving the formation of internal representations. Consequently, we would expect to find these functional properties within our artificial neural networks.

As an example, Rumelhart, Hinton, and McClelland (1986, p. 46) list eight properties that are essential to *Parallel Distributed Processing* (PDP) models.

- A *set of processing units*
- A *state of activation*
- An *output function* for each unit
- A *pattern of connectivity* among units
- A *propagation rule* for propagating patterns of activities through the network of connectivities

- An *activation rule* for combining the inputs impinging on a unit with the current state of that unit to produce a new level of activation for the unit.
- A *learning rule* whereby patterns of connectivity are modified by experience
- An *environment* within which the system must operate

These eight properties of PDP models map easily onto the six functional properties of the neuron. The processing unit is the neuron itself. The state of activation and the activation rule are part of the input and intergrative device of the neuron and the output function is simply the output of the neuron. The pattern of connectivity and propagation rule map onto the conductive function of the neuron. And, the learning rule and environment are part of the computational and representational functions of the neuron.

To be fair, though, PDP models are simply a subclass of connectionist models. Therefore, Bechtel and Abrahamsen (1991) have reduced the above list to four properties that distinguish the different types of connectionist architectures. These four properties are (i) the connectivity of units, (ii) the activation function of units, (iii) the nature of the learning procedure that modifies the connections between units, and (iv) how the network is interpreted semantically.

The above properties of connectionist models can be summarized in three basic tenets. First, signals are processed by elementary units. Second, processing units are connected *in parallel* to other processing units. Third, connections between processing units are weighted. These three tenets are necessarily broad in their descriptions so as to accommodate all aspects of connectionism; however, further elaboration is given below.

For example, the processing of signals encompasses the receiving, transformation, and transmission of information. The signals themselves may be carried by electrical, chemical, or mechanical means. Furthermore, signals could be supplied from an external stimulus (such as light impinging on the retina) or from other processing units. The processing units (see Figure 2.6) may refer to neurons, mathematical functions, or even demons *à la* Selfridge (1959). Lastly, information may be encoded in the units either locally or in a distributed manner.

The connections between units may or may not be massively parallel in the sense that every unit is connected to every other unit. Moreover, connections may be "feed-forward" (i.e., signals being passed in one direction only; Rumelhart, Hinton, & Williams, 1986a, 1986b), or "interactive" (i.e., bidirectional passing of signals; McClelland, 1981).

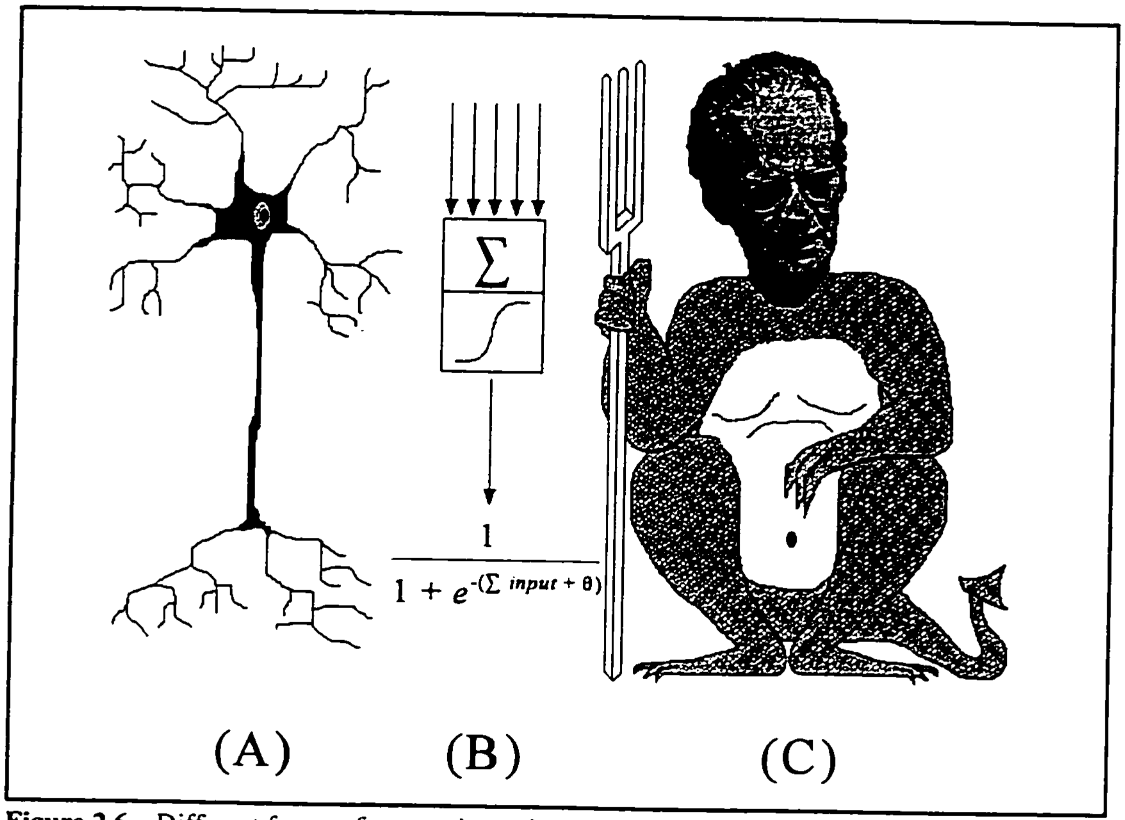

Figure 2.6 Different forms of processing units: (A) stylized sympathetic ganglion, (B) mathematical equation, (C) cantankerous *Pandemonium* demon.

Finally, the weights associated with the connections may be "hardwired," learned, or both. The weights represent the strength of connection (either excitatory or inhibitory) between two units. These three tenets allow a large spectrum of models (e.g., Selfridge's *Pandemonium*, 1959; Rumelhart & McClelland's *Past-Tense Acquisition Model*, 1986; Dawson's *Motion Correspondence Model*, 1991) to fall within the classification of connectionism. It should be noted that throughout this thesis, the terms connectionist network, *Artificial Neural Network* (ANN), and *Parallel Distributed Processing* (PDP) model will be used synonymously.

With our framework established, we can now begin our reevaluation of connectionism. We will start at the beginning.

Chapter 3

A Brief History of Connectionism

This solution takes the form of a new associationism, or better, since it differs deeply and widely from that older British associationism, of a new connectionism. (Thorndike, 1932, p. 4)

It has been remarked that psychology has a long past but only a short history; the very same could be said about connectionism. Over the past decade, there has been a literal explosion in connectionist research. As Hanson and Olson (1991, p. 332) state "The neural network revolution has happened. We are living in the aftermath." As a result, one view popular with current researchers is that connectionism really emerged in the 1980's—there is only brief mention of research before that time (e.g., Bechtel & Abrahamsen, 1991; Horgan & Tienson, 1996).

Connectionism, however, has a very long past. In fact, one can trace the origin of connectionist ideas to the early Greek philosopher, Aristotle, and his ideas on mental associations. These ideas were elaborated by the British empiricists and then naturally extended by the founders of psychology. Neuropsychologists contributed to the growth of connectionism by attempting to relate the processes of learning and memory to the underlying properties of the brain. But, this only half of the picture. The other half of the picture is filled in by those researchers engaged in mathematical research and early computing science.

Although it might be argued that these past researchers were not true "connectionists" in today's terms, the ideas they put forth in the disciplines of philosophy, psychology, neuropsychology, mathematics, and computing science are fully embodied within today's connectionism. And, it is only through a review of the contributions made by each of these disciplines that we can place connectionism in its proper context today.

Philosophical Roots

Although the popularity of connectionist research has grown considerably over the past decade, it is certainly not a new phenomenon. Aristotle (ca. 400 B.C.) has been cited by Anderson, Pellionisz, and Rosenfeld (1990) as the first to propose some of the basic concepts of connectionism; that is, memory is composed of simple elements linked or connected to each other via a number of different mechanisms (such as temporal succession, object similarity, and spatial proximity). Furthermore, these associative structures could be combined into more complex structures to perform reasoning and memory access. Thus, a "well-specified outline of a perfectly viable computational theory of memory" (Anderson et al., 1990, p. 3) based on the interconnection of simple elements existed at least 2,400 years ago.

Moreover, many of the underlying assumptions of connectionism can be traced back to the ideas eminent in the philosophical school of materialism (e.g., la Mattrie, Hobbes), and the resulting school of British empiricism (e.g., Berkeley, Locke, Hume). Materialists held the view that nothing existed except for matter and energy, and that all human behaviour—including conscious thought—could be explained solely by appealing to the physical processes of the body, especially the brain (cf., Descartes' dualism). This led to the empiricist view that human knowledge is derived ultimately from sensory experiences, and it is the association of these experiences that lead to thought (Aune, 1970; Leiber, 1991). Therefore, human cognition is governed by physical laws and can by studied empirically.

Within the empiricist tradition, accounting for psychological processes is known as associationism. The basic concepts of associationism are

i. mental elements or ideas become associated with one another through experience,
ii. experience consists of such things as spatial contiguity, temporal contiguity, similarity, and dissimilarity of ideas,
iii. complex ideas can be reduced to a set of simple ideas,
iv. simple ideas are sensations, and
v. simple additive rules are sufficient to predict complex ideas composed from simple ideas.

Although many associationist concepts are evident in the behaviourist movement in psychology, the cognitivist movement within psychology has dismissed associationism as inadequate to account for cognitive phenomenon such as recursive grammars (e.g., Bever, Fodor, & Garret, 1968).

Not surprisingly, with assumptions founded in associationist theories, connectionism has often been mistaken for associationism (e.g., Fodor & Pylyshyn, 1988, footnote 29), and subsequently dismissed as a viable theory of cognition. As pointed out by Thorndike (1932), however, connectionism should not be confused for associationism. Rather, connectionism has borrowed concepts from associationism and has expanded them. For example, connectionism employs such concepts as distributed representations, hidden units, and supervised learning—concepts foreign to associationism (Bechtel & Abrahamsen, 1991).

In fact, Bechtel (1985) points out that connectionism embodies a very distinctive characteristic that distinguishes cognitivism from behaviourism and associationism; specifically, connectionist modelers postulate that the connections between units provide structure in which mental activity occurs, and this structure is important for mediating future behaviour. Hence, connectionists are not repudiating cognitivism, they are simply providing an alternative to the standard rules and representation view of cognition. On the other hand, connectionism does embrace one very important aspect of associationism often missing from classical cognitive models; connectionism focuses on learning as a natural activity of the system being modeled. Consequently, Bechtel (1985, p. 60) concludes that connectionism may provide "a basis to draw together aspects of the two traditions that have generally been viewed as incommensurable."

Psychological Manifestations

With the emergence of psychology as a distinct field from philosophy, the ideas underlying connectionism became more refined and based on the neurophysiology of the day. In fact, as architects of a new field, early psychologists called for the integration of the mind and body, insisting that "a certain amount of brain-physiology must be presupposed or included in Psychology" (James, 1890/1950, vol. 1, p. 5). Consequently, founding psychologists such as Spencer (1855/1910) and James (1890/1950) are often cited for early examples of networks embodying connectionist principles; that is, networks that combined associationist principles with neurology.

The appearance of the hardline behaviourist movement (e.g., Watson, 1913; Skinner, 1938), by all accounts, should have signaled the demise of connectionist ideas in psychology[1]. Surprisingly, however, it was behavioural psychologists (e.g., Thorndike, 1932, 1949; Hull, 1943), that finally made the distinction between associationism and connectionism (Walker, 1990). Following the demise of behaviourism and the rise of cognitivism and symbolic processing, connectionist research all but disappeared from psychological literature. It has only recently become vogue once again.

But, for now, let us concentrate on psychology's contribution to connectionism.

Spencer's Connexions

In his two volume series entitled *The Principles of Psychology* (1855/1910), Herbert Spencer laid out the foundations of what was then the emerging field of psychology. One of his central tenets was that a description of the nervous system was essential for the understanding of psychology. Thus, he devoted several sections of his text to describing neural structures and their functions. Part of this description included describing how connections may be formed—not only the connections between one neuron and another (see Figure 3.1), but also the connections between ideas and concepts. He even went so far as to state that "there is a fundamental connection between nervous changes and psychical states" (vol. I, p. 129).

Using the growth of intelligence as an example, Spencer first identified those psychical aspects that define intelligence—the correspondence of internal relations with external relations. Intelligence grew as a function of the differentiation of external events into ordered states of consciousness. Thus, changes in the psychical states could be linked directly to changes in the external order. As an infinite number of correspondences between internal and external events could exist over time, Spencer concluded that no general law could be stated for such a series of changes. Instead, a law of changes had to be sought in the small, immediately connected changes:

> When any state *a* occurs, the tendency of some other state *d* to follow it, must be strong or weak according to the degree of persistence with which A and D (the objects or attributes that produce *a* and *d*) occur together in the environment. (vol. 1, pp. 408)

[1] Watson proposed that psychology should only be interested in objective, observable behaviour. "The consideration of the mind-body problem affects neither the type of problem selected not the formulation of the solution of that problem." (Watson, 1913, p. 166). This is exemplified by Skinner's (1938) view that behaviour could be studied without any appeal to the brain.

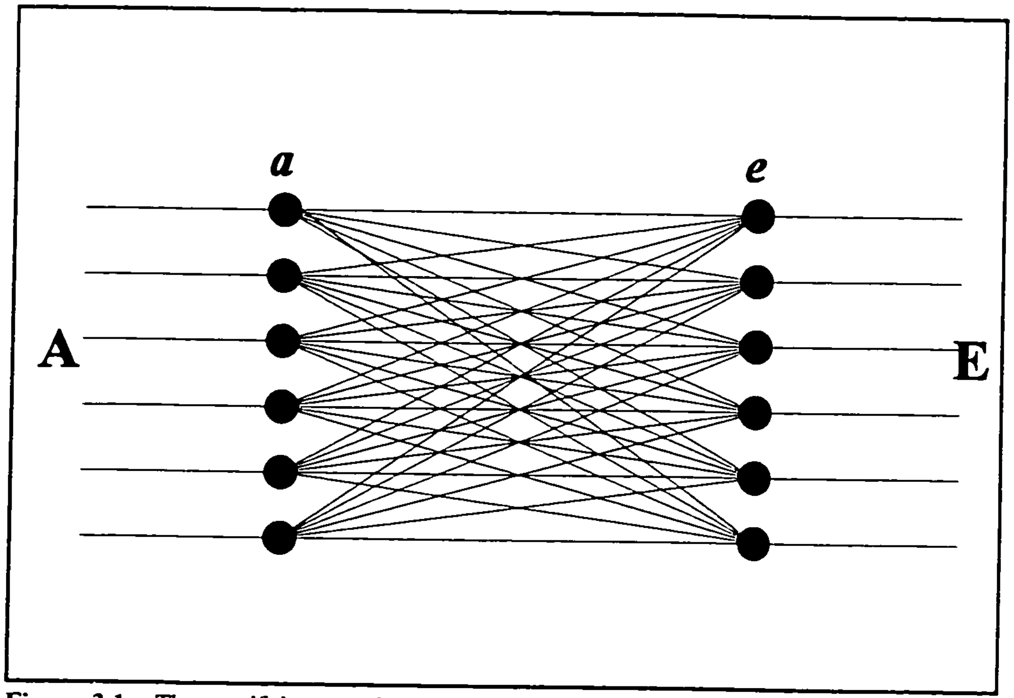

Figure 3.1 The needful connexions between afferent (A) and efferent (E) fibres to allow efficient transmission of a signal to move a muscle. Points *a* and *e* are where the afferent and efferent fibres diverge respectively (adapted from Spencer's [1855/1910] Figure 7)

This law of connection also holds for events that co-exist in the world. If events A and B habitually coexist in the environment, then conscious states *a* and *b* must coexist as well. As neither A or B is antecedent or consequent, then state *a* is just as likely to induce state *b* as state *b* is to induce state *a*. Thus, the networks of connections could either be "feed-forward" (as in the case of *a* and *d*) or "interactive" (as in the case of *a* and *b*).

As one last note, Spencer states that it is "the strengths of the connexions" (vol. 1, p. 409) between the internal states and external events that is important. In other words, correct knowledge of the world is encoded within the connections of the brain.

James' Associative Memory

Further examples of early connectionist theory are also evident in William James' (1890/1950) treatment of psychology (interestingly enough, also a two volume set entitled *The Principles of Psychology*). James, like Spencer, was committed to the fact that psychological phenomenon could be explained in terms of brain activity—"*no mental modification ever occurs*

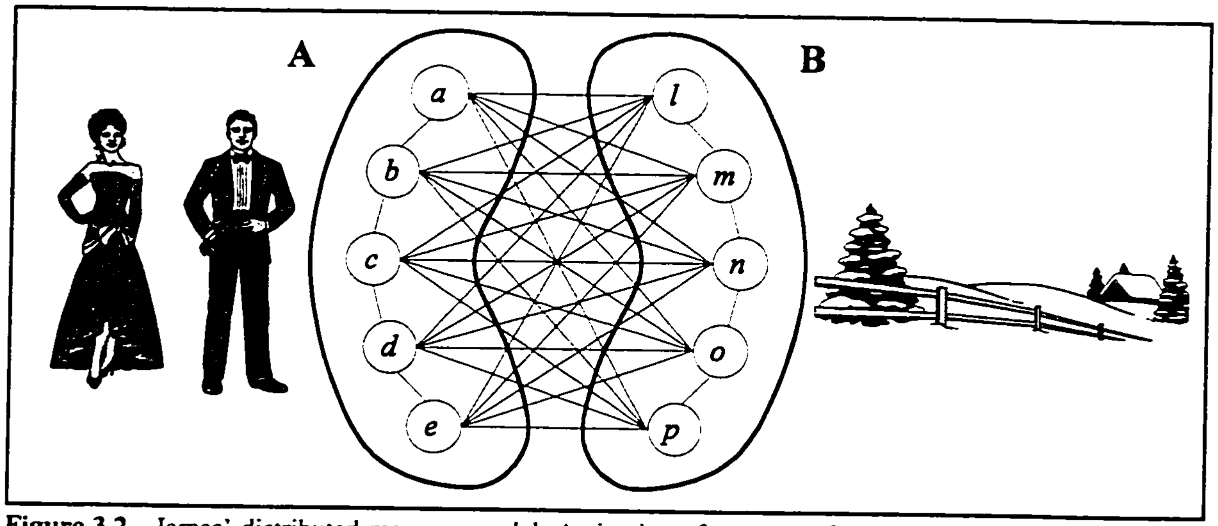

Figure 3.2 James' distributed memory model. Activation of event A (formal dinner party) causes activation of event B (walking through a frosty night) through weighted parallel connections.

which is not accompanied or followed by a bodily change" (vol. 1, p. 5; his italics). In fact, James equated the analysis of neural functioning with the analysis of mental ideas.

> There is a complete parallelism between the two analyses, the same diagram of little dots, circles, or triangles joined by lines symbolizes equally well the cerebral and mental processes: the dots stand for cells or ideas, the lines for fibres or associations. (James, 1890/1950, vol. 1, p. 30)

The most obvious example of connectionist principles is James' (1890/1950) associative memory model; the model consists of individual ideas that are connected in parallel such that recall of one idea is likely to cause the recall of related ideas. Thus, within this model, activation of event A with its component parts *a, b, c, d,* and *e* (e.g., attending a dinner party) caused activation of event B with its component parts *l, m, n, o,* and *p* (e.g., walking home through the frosty night) since all aspects of A were connected, or redintegrated, with all aspects of B (see Figure 3.2).

James recognized that, all things being equal, any activation in such a network would unfortunately result in "the reinstatement in thought of the *entire* content of large trains of past experience"[2] (vol. 1, p. 570). To counter this type of total recall, James proposed the law of interest: "some one brain-process is always prepotent above its concomitants in arousing action elsewhere"

[2] James quickly points out that only the minor personages within Dickens' and Eliot's novels possess this type of memory system.

(vol. 1, p. 572). Hence, not all connections in the brain are created equal. But, how are these connections modified, and hence the associations between memories learned? In order to account for learning, James formulated the *Law of Neural Habit*:

> *When two elementary brain-processes have been active together or in immediate succession, one of them, on reoccurring, tends to propagate its excitement into the other.* (vol 1., p. 566; his italics)

In other words, when two events occur repeatedly, the connection between the relevant brain-processes is strengthened. Note that James is talking about modifying brain-processes *physically* and not simply strengthening the associations between ideas. Even now, we begin to see the borrowing and modification of associationist ideas to account for cognitive processes and learning in biological systems.

More importantly, these very simple concepts—weighted, modifiable, parallel connections—laid down over a century ago form the cornerstone of connectionism today.

Thorndike's Connectionism

Edward Lee Thorndike was a student of James; therefore, it is not surprising that he carried over some of the principles inherent in James' work. Although considered one of the founding behaviourists—especially in terms of founding the study of operant behaviour (Pierce & Epling, 1995)—Thorndike was concerned with states of mind (cf., Watson, 1913), and how they changed with experience. More importantly, however, Thorndike can be considered one of the first true connectionists.

In his book, *The Fundamentals of Learning* (1932), he differentiates between the principles of British associationism and "new connectionism." Furthermore, in 1949, he summarized what he considered his most important contributions to psychology under the title *Selected Writings from a Connectionist's Psychology* so that students may "know something of connectionist psychology" (p. v).

Thorndike's connectionism can be viewed as a turning point where theories of neural association became sub-symbolic and graduated from merely implementational accounts to accounts of the functional architecture (Walker, 1990). In other words, the neural connections became a substitute for, instead of a mechanism of, ideational processes. Thus, his computational descriptions of the fundamentals of learning were couched in the language of connectionist principles.

For example, Thorndike believed that the most prevalent questions within learning theory were:

1. What happens when the same situation or stimulus acts repeatedly upon an organism—does the mere frequency of an experience cause useful modifications?
2. What happens when the same connection occurs repeatedly in a mind?
3. What effect do rewards and punishments have on connections, and how do they exert this effect?

In order to answer these questions, Thorndike proposed two different laws. The first law, the "*Law of Exercise or Use or Frequency*", states that all things being equal, the more often a situation connects with or evokes or leads to or is followed by a certain response, the stronger becomes the tendency for it to do so in the future. The second law, the "*Law of Effect*", states that what happens as an effect or consequence or accompaniment or close sequel to a situation-response, works back upon the connection to strengthen or weaken it. Thus, if an event was followed by a reinforcing stimulus, then the connection was strengthened. If, however, an event was followed by a punishing stimulus, then the connection was weakened. The principles underlying this law are very similar to the supervised learning techniques (such as error backpropagation) used in today's neural networks.

Finally, Thorndike anticipated the backlash against the principles of connectionism:

> Many psychologists would indeed deny that any system of connections was adequate to explain his behaviour, and would invoke powers of analysis, insight, purpose, and the like to supplement or replace the simple process of connection-forming by repetition and reward. (Thorndike, 1932, p. 355)

Through a series of experiments, however, Thorndike (1932) shows that there is "no sufficient reasons for ascribing any power over and above that of repetition and reward to any 'higher powers' or 'forms of thought' or 'transcendent systems'" (p. 382) and thus "justif[ies] the connectionist's faith" (p. 4).

Hull's Learning Rule

In 1943, Clark L. Hull set for himself the task of elaborating the laws of behaviour from a molar level description of neural activity (since he considered the results of molecular neurophysiology inadequate). As part of this elaboration, he described several functional properties of neural activity that he deemed important for organism survival. These include:

i. the afferent neural impulse (s_1) which is a non-linear function of the input it receives,

ii. interactions between two or more afferent neural impulses (s_2 & s_3) which implies that behaviour to the same stimulus is not constant under all conditions, and

iii. the spontaneous generation of nerve impulses which may account for the variability of behaviour to identical environments.

With these functional properties identified, Hull stated that the "supremely important biological process" of learning could be expressed in terms of modifying receptor-effector connections:

> The essential nature of the learning process may, however, be stated quite simply . . . the process of learning consists in the strengthening of certain of these connections as contrasted with others, or in the setting up of quite new connections. (pp. 68-69)

The process of learning is wholly automatic—it occurs as the result of the interaction of the organism with its environment, both external and internal. Furthermore, the rules of learning must be capable of being stated in a clear and explicit manner without recourse to a guiding agent. Thus, Hull developed several empirically testable equations to describe the learning process. The one that concerns us the most for historical reasons is his formula for the growth of stimulus-response habits. This is simply the increase in the strength of connection (to a physiological maximum) between a stimulus and a response as a function of the number of reinforcing trials.

The growth of habit strength is dependent on three factors (p. 114):

1. The physiological limit or maximum (M)
2. The ordinal number (N) of the reinforcement producing a given increment to the habit strength ($\Delta_S^{\circ}H_R$)
3. The constant factor (F) according to which a portion ($\Delta_S H_R$) of the unrealized potentiality is transferred to the actual habit strength at a given reinforcement.

Thus, habit strength as a function of the number of reinforcement repetitions can be computed as follows

$$ {}_S^N H_R = M - Me^{-N \log \frac{1}{1-F}} \tag{3.1}$$

which generalizes over trials to

$$\Delta H = f(M - H) \tag{3.2}$$

Hull is quick to point out that habit strength cannot be determined by direct observation; the strength of the receptor-effector connection can only be measured and observed indirectly. This is because the organization of the processes underlying habit formation are "hidden within the complex structure of the nervous system" (p. 102). Consequently, the only way of inferring habit strength is to note the associations between the antecedent conditions which lead to habit formation and the behaviour which is the consequence of these same conditions. It would be tempting, therefore, to conclude that Hull's concept of habit formation could be classified as unsupervised learning as there is no recourse to an external guiding agent. But, the fact that the amount of learning is dependent on a portion of unrealized potential makes this a supervised learning algorithm.

In fact, it has been noted by Walker (1990) that Hull's habit strength equation is a forerunner to the Rescorla-Wagner rule (1972), which has been shown by Sutton and Barto (1981) to be essentially identical to the Widrow-Hoff (1960) rule for training *Adaline* units (see Equation 3.6). Furthermore, this equation can be seen as a primitive form of the generalized delta rule for backpropagation in neural networks (see Chapter 4, Equation 4.3).

The Neuropsychological Influence

Connectionist models derive their inspiration from neurophysiology. Consequently, it is appropriate to touch briefly on the neuropsychological influence exerted on connectionism. Following the pioneering work of such researchers as Sherrington and Cajal[3], researchers began to

[3] Sherrington was responsible for coining the term *synapse* to denote the structural and functional loci of interaction between neurons while Cajal was responsible for introducing the *neuron theory*—the nervous system is composed of neurons which are individual functional units.

seek the neural correlates of learning and memory. From this research paradigm emerged two prominent figures in regards to the history of connectionism: Karl Lashley and Donald Hebb.

Lashley's Search for the Engram

One of the most intensive searches to localize memory traces—or engrams—within the brain was initiated by Karl Lashley in the 1920's. Lashley's studies involved training an animal to perform some specific task, such as brightness discrimination or maze orientation, and lesioning a specific area of the cortex either before or after training. Lashley then recorded the behavioural effects of cortical lesions on retention and acquisition of knowledge. In 1950, he summarized 30 years of research into two principles:

- The *Equipotentiality* Principle: all cortical areas can substitute for each other as far as learning is concerned.
- The *Mass Action* Principle: the reduction in learning is proportional to the amount of tissue destroyed, and the more complex the learning task, the more disruptive lesions are.

In other words, Lashley believed that learning and memory was a distributed process that could not be isolated within any particular area of the brain. Furthermore, it was not the location of the lesion that was important (within reason[4]), but the amount of tissue destroyed that determined the degree of behavioural dissociation. His failure to isolate the engram to any specific group of cells or neural pathways led Lashley to proclaim

> This series of experiments has yielded a good bit of information about what and where the memory trace is not. It has discovered nothing directly of the real nature of the engram. I sometimes feel, in reviewing the evidence on the localization of the memory trace, that the necessary conclusion is that learning just is not possible... Nevertheless, in spite of such evidence against it, learning does sometimes occur. (1950, p. 478).

[4] Lashley recognized that removing large portions of the visual cortex would prevent such things as brightness discrimination, but that this was due to the animal not being able to see, not to any deficit in learning or memory processes per se.

Although the principles of equipotentiality and mass action have been controversial since their publication, they do contribute to the field of connectionist research; specifically, to the ideas of distributed representations, multiple internal representations, and emergent network properties. In fact, recent lesioning experiments performed by connectionists (e.g., Farah, 1994; Plaut & Shallice, 1993) would tend to agree with Lashley in terms of network processing being distributed and nonlocalized. Just as neuropsychologists have questioned Lashley's conclusions (e.g., Hunter, 1930), however, the conclusions derived from experiments on lesioned connectionist networks are also being challenged (e.g., Medler, Dawson, Kingstone, & Panasiuk, 1997).

Hebb's Neuronal Learning

The most influential work in connectionism's history is undoubtably the contribution of Canadian neuropsychologist, Donald O. Hebb (a student of Lashley). In his book, *The Organization of Behaviour* (1949), Hebb presented a theory of behaviour based as much as possible on the physiology of the nervous system. Hebb reduced the types of physiological evidence into two main categories: (i) the existence and properties of continuous cerebral activity, and (ii) the nature of synaptic transmission in the central nervous system. Hebb combined these two principles to develop a theory of how learning occurs within an organism. He proposed that repeated stimulation of specific receptors leads slowly to the formation of "cell-assemblies" which can act as a closed system after stimulation has ceased. This continuous cerebral activity serves not only as a prolonged time for structural changes to occur during learning, but also as the simplest instance of a representative process (i.e., images or ideas).

The most important concept to emerge from Hebb's work was his formal statement of how learning could occur (cf., James' *Law of Neural Habit*). Learning was based on the modification of synaptic connections between neurons. Specifically,

> *When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.* (Hebb, 1949, p.62; his italics)

The principles underlying this statement have become known as Hebbian learning. From a neurophysiological perspective, Hebbian learning can be described as a time-dependent, local, highly interactive mechanism that increases synaptic efficacy as a function of pre- and post-synaptic activity. Although the neurophysiology in Hebb's day was inadequate to support or deny Hebb's postulate, more recent research has shown that Long-Term Potentiation (e.g., Bliss & Gardner-Medwin, 1973; Bliss & Lomo, 1973) and Long-Term Depression (e.g, Crepel, Hemart, Jaillard, & Daniel, 1995) have those putative mechanisms required of Hebbian learning.

Within connectionism, Hebbian learning is an unsupervised training algorithm in which the synaptic strength (weight) is increased if both the source neuron and target neuron are active at the same time. A natural extension of this (alluded to by Hebb as the decay of unused connections) is to decrease the synaptic strength when the source and target neurons are not active at the same time. Hence, Hebbian learning can be formulated as:

$$w_{ij}(t+1) = w_{ij}(t) + NET_i \; NET_j \tag{3.3}$$

where

$w_{ij}(t)$ = the synaptic strength from neuron i to neuron j at time t

NET_i = the excitation of the source neuron.

NET_j = the excitation of the destination neuron.

There are serious limitations with Hebbian learning as stated (e.g., the inability to learn certain patterns), but variations of this simple algorithm exist today (e.g., Signal Hebbian Learning, Differential Hebbian Learning; see Wasserman, 1989).

The Mathematical Influence

The next major formulation of connectionist theories can be attributed to McCulloch and Pitts (1943). In their seminal paper *A logical calculus of the ideas immanent in nervous activity*, they explicitly laid out the foundations of neural modelling in terms of propositional logic. To accomplish this, they simplified the activity of neurons into five functional states (p. 118):

1. The activity of the neuron is an "all-or-none" process.
2. A certain fixed number of synapses must be excited within the period of latent addition in order to excite a neuron at any time, and this number is independent of previous activity and position on the neuron.
3. The only significant delay within the nervous system is synaptic delay.
4. The activity of an inhibitory synapse absolutely prevents excitation of the neuron at that time.
5. The structure of the net does not change with time.

Using these principles, McCulloch and Pitts were able to show that *any* statement within propositional logic could be represented by a network of simple processing units. Furthermore, such nets have the *in principle* computational power of a Universal Turing Machine. "If any number can be computed by an organism, it is computable by these definitions, and conversely" (p. 128). Therefore, networks are able to describe information processing at the computational level of analysis.

> To psychology, however defined, specification of the net would contribute all that could be achieved in that field— even if the analysis were pushed to the ultimate psychic units or "psychons," for a psychon can be no less than the activity of a single neuron. (p. 131)

McCulloch and Pitts also proved that there is always an indefinite number of topologically different nets realizing any temporal propositional expression, although time discrepancies might exist between the different realizations. What this states is that there exists many different algorithms to compute the same function.

Early Computer Models of Connectionism

> Logically, if it were possible to construct non-living devices—perhaps even of inorganic materials—which would perform the essential functions of the conditioned reflex, we should be able to organize these units into systems which would show true trial-and-error learning with intelligent selection and the elimination of errors, as well as other behavior ordinarily classed as psychic. Thus emerges in a perfectly natural manner a direct implication of the mechanistic tendency of modern psychology. Learning and thought are here conceived as by no means necessarily a function of living protoplasm any more than is aerial locomotion. (Hull, C. L., & Baernstein, H. D., 1929, pp. 14-15).

Perhaps the most influential event in the development of connectionism was the invention of the modern computer. Theories that could only be tested previously by observing the behaviour of animals or humans (e.g., Thorndike, 1932; Hull, 1943) could now be stated more formally and investigated on artificial computation devices. Hence, theory generation and refinement could now be accomplished faster and more precisely by using the empirical results generated by the computer simulations.

The computer and its influence on learning theory can be credited with producing both positive and negative press for connectionism. Selfridge's *Pandemonium* (1959) and Rosenblatt's

Perceptrons (1958, 1962) did much to further the concepts of connectionism. The proofs on the limitations of simple perceptrons by Minsky and Papert (1969/1988), however, nearly caused the complete abandonment of connectionism.

Pandemonium

Recognizing that previous attempts to get machines to imitate human data had all but failed, Selfridge (1959) proposed a new paradigm for machine learning. *Pandemonium* was introduced as a learning model that adaptively improved itself to handle pattern classification problems that could not be adequately specified in advance. Furthermore, whereas previous computer models relied on serial processing, Selfridge proposed a novel architecture to deal with the problem, parallel processing. The move to parallel processing was not an arbitrary one, but one motivated by two factors: (1) it is easier, and more "natural" to handle data in a parallel manner[5], and (2) it is easier to modify an assembly of quasi-independent modules than a machine whose parts interact immediately and in a complex way.

Pandemonium consists of four separate layers: each layer is composed of "demons" specialized for specific tasks (see Figure 3.3). The bottom layer consists of data or image demons that store and pass on the data. The third layer is composed of computational demons that perform complicated computations on the data and then pass the results up to the next level. The second layer is composed of cognitive demons who weight the evidence from the computational demons and "shriek" the amount of evidence up to the top layer of the network. The more evidence that is accumulated, the louder the shriek. At the top layer of the network is the decision demon, who simply listens for the loudest "shriek" from the cognitive demons (a winner-take-all strategy), and then decides what was presented to the network.

The initial network structure is determined *a priori* by the task, except for the computational level which is changed by two different learning mechanisms. The first mechanism changes the connection weights between the cognitive demons and the computational demons via supervised learning (all other connections within the network being fixed a priori). The weights are trained using a hill-climbing procedure in order to optimize the performance of the network. After supervised learning has run long enough to produce approximately optimal behaviour, the second learning mechanism is employed.

[5] "parallel processing seems to be the human way of handling pattern recognition" (Selfridge & Neisser, 1960, p.66).

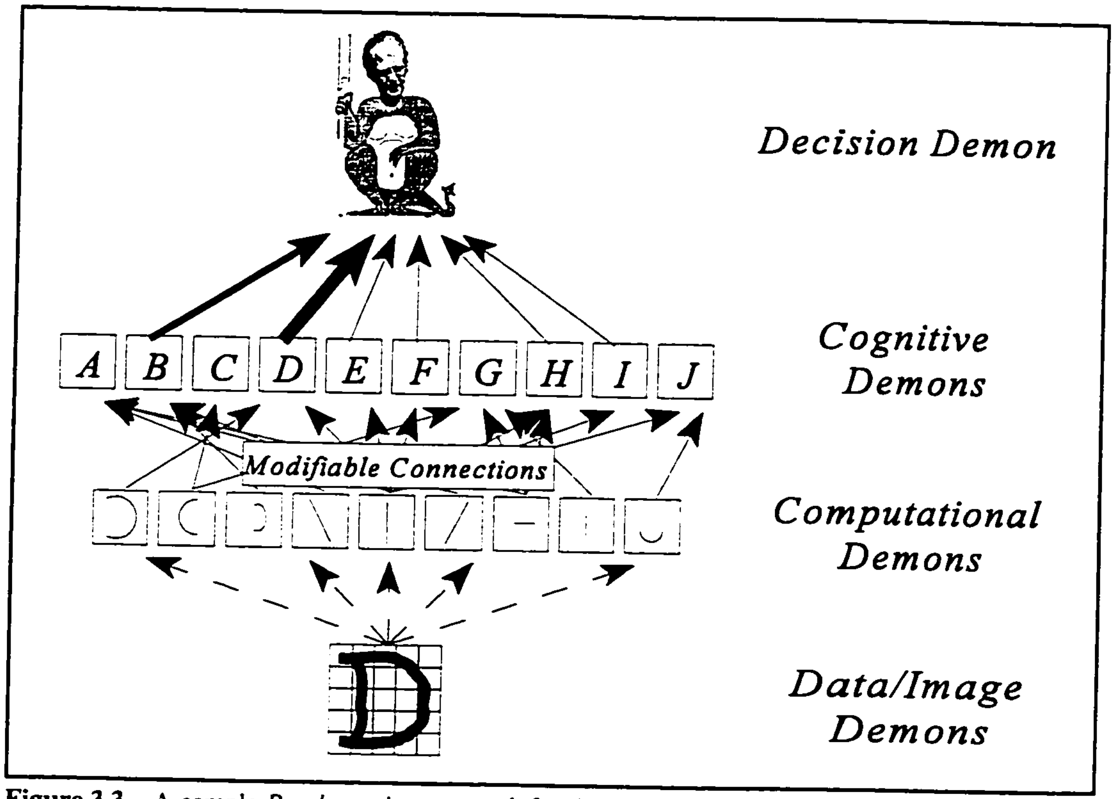

Figure 3.3 A sample *Pandemonium* network for detecting hand-written characters. The Decision Demon listens for the loudest "shriek" from the Cognitive Demons.

The second learning mechanism selects those computational demons that have a high worth (based on how likely they are to influence a decision), eliminates those demons that have a low worth, and generates new demons from the remaining good demons. Generation can be accomplished by either mutating a demon, or conjoining two successful demons into a continuous analogue of one of the ten nontrivial binary two-variable functions. In fact, this second training mechanism may be one of the first genetic machine learning algorithms.

Selfridge demonstrated the effectiveness of *Pandemonium* on two different tasks: distinguishing dots and dashes in manually keyed Morse code (1959), as well as recognizing 10 different hand-printed characters (Selfridge & Neisser, 1960). Thus, a practical application of connectionist principles has been applied to pattern recognition. In fact, *Pandemonium* has been so successful as a model of human pattern recognition that it has been adopted and converted into a more traditional symbolic model (with connectionist principles conveniently ignored) by cognitive psychologists (e.g., Lindsay & Norman, 1972).

Rosenblatt's Perceptron

The perceptron, more precisely, the theory of statistical separability, seems to come closer to meeting the requirements of a functional explanation of the nervous system than any system previously proposed. (Rosenblatt, 1958, p. 449).

Although originally intended as a genotypic model of brain functioning (Rosenblatt, 1958, 1962), the *perceptron* has come to represent the genesis of machine pattern recognition. Basically, the perceptron is a theoretically parallel computation device composed of:

i. a layer of sensory units (S-unit) which transduce physical energy (e.g. light, sound, etc.) into a signal based on some transformation of the input energy,
ii. any number of layers of association units (A-unit) which have both input and output connections, and
iii. a final layer of response units (R-unit) which emit a signal that is transmitted to the outside world.

Figure 3.4 shows different graphical representations of a perceptron system. Note that the same perceptron system can be expressed in terms of a network diagram, a set diagram, or even a symbolic diagram. Rosenblatt was not concerned with how the perceptron was interpreted, but was concerned with what it could tell us about statistical separability.

An elementary α-perceptron is defined, then, as a network in which S-units are connected

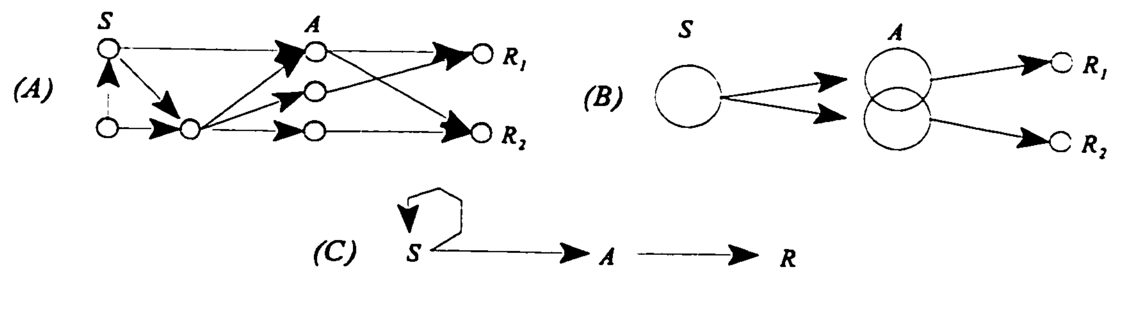

Figure 3.4 Different diagrams representing the same perceptron system. (A) Network diagram. (B) Set diagram. (C) Symbolic diagram. (Adapted from Figure 2, Rosenblatt, 1962, p. 86)

to A-units (although not necessarily massively parallel), and all A-units are connected to a single R-unit, with no other connections being permitted. Furthermore, all connections are considered to have equal transmission rates, τ. The transfer function between units *i* and *j* at time *t* is expressed as

$$c_{ij}^{*}(t-\tau) = u_i^{*}(t-\tau)\ v_{ij}(t-\tau) \tag{3.4}$$

where $u_i^{*}(t-\tau)$ is the output of unit *i* at time *t*, and $v_{ij}(t-\tau)$ is the connection strength between units *i* and *j* at time *t*. Connection strengths from S- to A-units are fixed, while connection strengths from A- to R-units vary with the reinforcement history applied to the perceptron. Both A- and R-units have a threshold, θ, and emit a signal whenever the input signal, α, is equal to or greater than θ. We can assume that τ=0 without loss of generality, and thus, the reinforcement rule is

$$\Delta v_{ij} = u_i^{*}(t)\cdot\eta = \begin{cases} \eta \text{ if } \alpha_i(t) \geq \theta \\ 0 \text{ otherwise} \end{cases} \tag{3.5}$$

where η is of constant magnitude.

The theoretical importance of the elementary α-perceptron lies in the fact that, for binary inputs, a solution exists for every classification, *C(W)*, of all possible environments *W*. In other words, an elementary α-perceptron is capable of solving *any* pattern classification problem expressed in binary notation. The proof is rather trivial:

i. For every possible pattern, S_i, in *W*, let there be a corresponding A-unit, a_i.
ii. Make the connection, v_{ij}, between a_i and the corresponding sensory unit, s_j, excitatory (i.e., value equal to +1) if the pattern on that s_j is "on"; otherwise make the connection inhibitory (i.e., value equal to -1).
iii. Set the threshold of a_i, θ, equal to the number of excitatory connections. Thus, a_i responds to one and only one pattern in *W*.
iv. If S_i is a positive instance of *C(W)* then make the connection from a_i to the r-unit positive (i.e. value equal to +1); otherwise make the connection negative (e.g., value equal to -1). With the threshold of the R-unit equal to zero, the network correctly classifies all S_i in *W*.

The above proof shows theoretically that any pattern classification problem expressed in binary notation can be solved by a perceptron network. As a concrete example, Figure 3.5 shows the network configuration for solving the XOR problem. The problem with this proof, however, is that it produces the final network structure, but does not indicate if the network could be trained to such a configuration. Consequently, Rosenblatt developed the *Perceptron Convergence Theorem* to show that an elementary α-perceptron using an error correction procedure is guaranteed to converge on a solution in finite time, providing that (i) a solution exists, (ii) each pattern is presented to the network at least twice, and (iii) the connections between the S-units and A-units are fixed.

Although theoretically very powerful, the practical problem with perceptrons was that there was no reliable method of adjusting the connections between the sensory (input) units and the association (internal) units. Hence, as a true learning network, perceptrons were limited to just a layer of sensory units connected directly to a layer of response units, with no intervening layers. With the output of the R-unit being monotonic (i.e., $u_i^*(t) = f(\alpha_i(t))$, where $\alpha_i(t)$ is the algebraic sum of all the inputs into unit u_i), the resulting networks were very limited in their computational power. Rosenblatt was quick to point this limitation out, although he left the proof up to the reader.

> It is left to the reader to satisfy himself that a system with less "depth" than an elementary perceptron (i.e., one in which S-units are connected directly to the R-unit, with no intervening A-units) is incapable if representing *C(W)*, no matter how the values of the connections are distributed. (Rosenblatt, 1962, p. 101).

S_i	*A-Unit*	*Response*
off off	a_0	*off*
on off	a_1	*on*
off on	a_2	*on*
on on	a_3	*off*

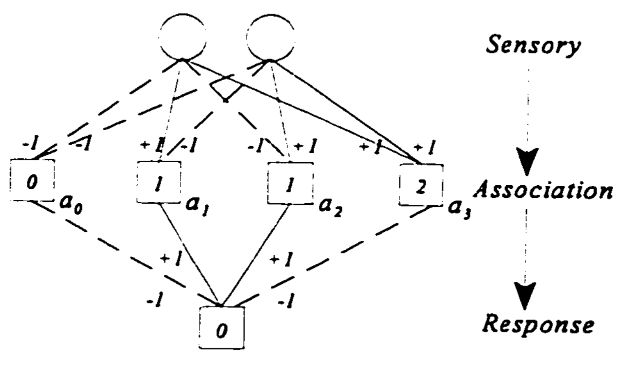

Figure 3.5 The solution for an elementary α-perceptron solving the XOR problem. Weights are indicated by the number next to the connection, whereas biases are indicated by the number within the unit.

Adaline

The next major formulation in learning rules for networks came from Widrow and Hoff (1960). They developed "*Adaline*" (first for *ada*ptive *line*ar, then *ada*ptive *li*near *ne*uron, and later *ada*ptive *line*ar element as neural models became less popular) as an adaptive pattern classification machine to illustrate principles of adaptive behaviour and learning. The learning procedure was based on an iterative search process, where performance feedback was used to guide the search process. In other words, a designer "trains" the system by "showing" it examples of inputs and the respective desired outputs. In this way, system competence was directly and quantitatively related to the amount of experience the system was given.

The typical Adaline unit, also called a "neuron element," is illustrated in Figure 3.6. It is a combinatorial logic circuit that sums the signals from weighted connections (gains), a_i, and then sends an output signal based on whether or not the internal signal exceeded some threshold. The threshold was determined by a modifiable gain, a_0, which was connected to a constant +1 source. As opposed to the usual convention of using signals of 0 and 1, the Adaline used input signals of -1 and +1 which meant a signal was *always* passed along a connection (unless the gain on the line was

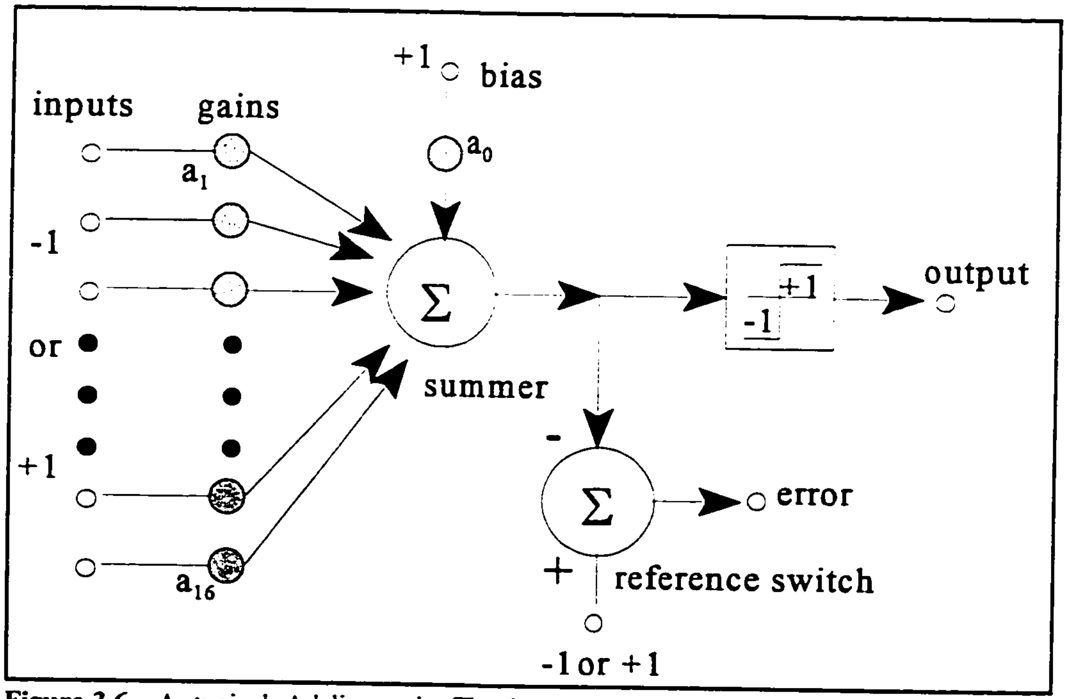

Figure 3.6 A typical Adaline unit. The inputs are sent along weighted connections (gains) to a summer which performs a linear combination of the signals. The output of the summer is compared to the value of the reference switch and the gains are adjusted by the same absolute value to produce an output of exactly -1 or +1.

zero). Similarly, the gains on the connections were adjusted so that the output signals were exactly -1 or +1; therefore, classification was not simply correct, but *exactly* correct. This restriction on the outputs meant that learning continued even if the classification was simply correct as the summed inputs may not be exactly -1 or +1. This continued learning was an improvement over the simple perceptron which did not change its weights if the gross classification was correct.

The learning procedure is based on the error signal generated by comparing the network's response with the optimal (correct) response. For example, consider an Adaline unit with 16 input lines and a bias threshold. A pattern is presented over the 16 input lines, and the desired output is set into the reference switch (see Figure 3.6). If the error (computed as the difference between the summer and the reference switch) is greater than zero, then all gains including the bias are modified in the direction that will reduce the error magnitude by 1/17. Upon immediate representation of the pattern an error signal of zero would be produced. Another pattern can now be presented to the network and the connections modified[6]. Convergence is achieved when the error (before adaptation) on any given pattern is small and there are small fluctuations about a stable root mean-square value. Consequently, the error correction algorithm is also known as the LMS (Least Mean Squares) algorithm in signal processing (Anderson & Rosenfeld, 1988).

The Widrow-Hoff rule (Sutton & Barto, 1981) is formulated as:

$$r_i(t) = z[(t) - y(t)]\,x_i(t) \tag{3.6}$$

where t is the target pattern, $y(t)$ is the network's output, and $x_i(t)$ is the input to the network. Because this rule is dependent on an external teacher it is termed *supervised* learning. Within neural network research, the Widrow-Hoff rule is often referred to as the *delta rule* because the amount of learning is proportional to the difference between the output and the target (Rumelhart, Hinton, & McClelland, 1986).

Perceptrons Revisited (Minsky & Papert)

Although it was known for a decade that simple perceptrons were limited in their ability to classify some patterns, it was not until Minsky and Papert published *Perceptrons* in 1969 that the extent of these limitations were fully realized. In fact, it was with this publication that the

[6] Note that at this point, presenting the first pattern to the network would produce an error that was small but not necessarily zero.

connectionist tide was stemmed[7] (at least for a while). Instead of asking if neural networks were good, Minsky and Papert asked the question "what are neural networks *good for*?" This is clearly a computational level question aimed at identifying the limitations of the representational abilities of perceptron-like networks. As Minsky and Papert point out in their prologue to the 1988 edition of *Perceptrons*, "No machine can learn to recognize X unless it possesses, at least potentially, some scheme for *representing* X." (p. xiii; their italics).

Hence, their approach to the study of neural networks was based on studying the types of problems that were being proposed at the time—mainly visual pattern recognition (Papert, 1988). In doing so, they discovered that some pattern recognition problems (e.g., distinguishing triangles from squares) were relatively easy and could be computed by simple networks. Conversely, some problems (e.g., determining if a figure was connected or not) were extremely difficult and required large networks to solve them. The main distinction between these two types of problems was not the size of the pattern space, but the concept of **order** (Minsky & Papert, 1988/1969, p. 30).

In general, the order of some function $\psi(X)$ is the smallest number k for which we can find a set Φ of predicates satisfying

$$\begin{cases} |S(\varphi_p)| \le k \textit{ for all } \varphi \textit{ in } \Phi, \\ \psi \in L(\Phi). \end{cases} \tag{3.7}$$

where φ is a simple predicate, and $L(\Phi)$ is the set of all predicates that are linear threshold functions. It should be noted that the order of ψ is a property of ψ alone, and not relative to any particular Φ. Functions that have an order of 1 are called "linearly separable" and can be solved by a single layer perceptron.

The types of pattern recognition problems that gave simple perceptrons trouble were those whose order was greater than 1. These types of problems are termed "linearly inseparable" and require a layer of processing units between the input and output units. At the time, however, there was no reliable method of training this intermediate level, and therefore perceptrons were limited to being trained on linearly separable problems only.

Minsky and Papert (1988/1969) used a very simple and elegant example to show the practical limitations of perceptrons. The exclusive-or (XOR) problem (see Figure 3.5) contains four

[7] Neural network research did not wane due to lack of interest, but because of lack of funding (Papert, 1988).

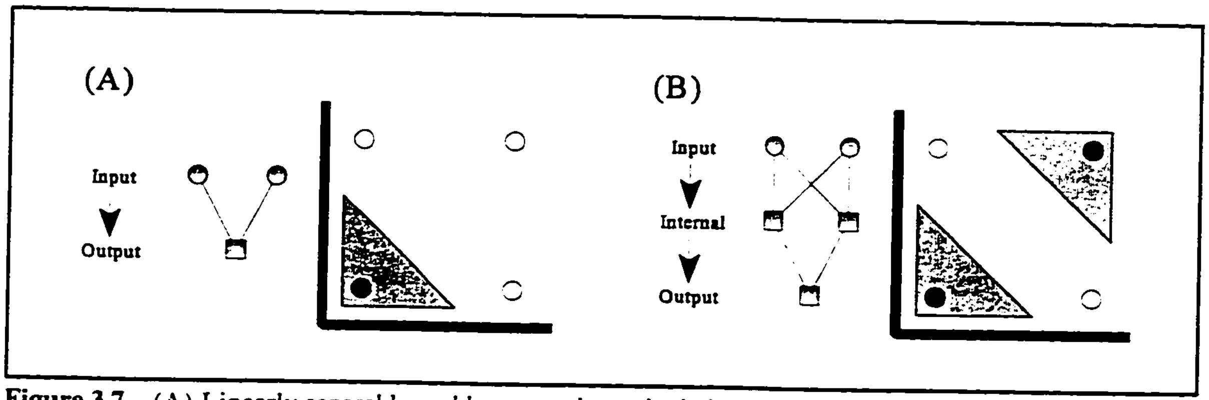

Figure 3.7 (A) Linearly separable problems require a single hyperplane to make the proper classification and, therefore, require no internal units. (B) Linearly inseparable problems necessitate two (or more) hyperplanes to make the correct classification. Therefore, a network requires a layer of internal units.

on, but not both. Thus, changing the input pattern by one bit changes the classification of the pattern. This is the most simple example of a linearly inseparable problem (see Figure 3.7). A perceptron using linear threshold functions requires a layer of internal units to solve this problem, and since the connections between the input and internal units could not be trained, a perceptron could not *learn* this classification. And, if perceptrons failed on this small pattern set, what hope was there for larger pattern sets that were also linearly inseparable?

Furthermore, Minsky and Papert lay out other limitations of networks. For example, if a network is to solve a problem with order R, then at least one partial predicate φ must have as its support the *whole space R*. In other words, at least one internal unit must be connected to each and every input unit. This network configuration violates what is known as the "limited order" constraint. Another limitation that Minsky and Papert discuss is the growth of coefficients. For linearly inseparable problems, the coefficients (i.e., weights) can increase much faster than exponentially with $|R|$. This leads to both conceptual and practical limitations. Conceptually, although the behaviour of a network may be "good" on small problems, this behaviour may become profoundly "bad" when the problem is scaled up. Practically, for very large $|R|$, the amount of storage space required for the weights would overshadow the space required to simply represent the problem.

Although advances in neural network research have produced methods for training multiple layers of units (e.g., Rumelhart, Hinton, & Williams, 1986a), many of Minsky and Papert's concerns remain unanswered. Networks using linear threshold units still violate the limited order constraint when faced with linearly inseparable problems (but see Chapter 5). Furthermore, the scaling of weights as the size of the problem space increases is still an issue (Feldman-Stewart & Mewhort, 1994). Nevertheless, connectionist research remains a thriving endeavour within cognitive science today.

The Importance of Old Connectionism

The publication of *Perceptrons* by Minsky and Papert in 1969 has taken on almost a mythical aura—it has been likened to the huntsman being sent out to bring back the heart of Snow White (Papert, 1988). Regardless of whether or not the work precipitated or merely coincided with the decline of connectionist research, it serves as a useful delineation between the "Old" and "New" connectionism.

The examples of connectionist networks provided in this chapter are often classified under the term "Old Connectionism". Old Connectionism is characterized by two different types of networks. The first are small, trainable networks, such as single layer perceptrons, that are computationally limited (i.e., cannot be trained to solve linearly inseparable problems). The second type of networks are large, computationally powerful networks that are mainly hardwired (although they could have a trainable layer of weights such as *Pandemonium*), and thus are limited in their learning ability. The problem with Old Connectionism was that it had no reliable way of combining these two different types of network architectures. To be an effective tool within cognitive science, researchers had to find a way of combining these two different types of networks.

Consequently, we are left with the question "Why should we be interested in Old Connectionism?" The first reason is purely academic. To understand the role of connectionism today and why we are at a crossroads, we have to understand how the field has developed. Knowing the history of connectionism, not only are we in a position to counter the arguments against connectionism from the classical camp (e.g., knowing why connectionism is not associationism), but also we are in a position to evaluate claims from the connectionist camp that it may represent a paradigm shift (see Chapter 9). To be effective researchers, we need to know both sides of the argument.

The second—and more important—reason is that by studying the development of connectionism we can appraise the strengths and weaknesses of the connectionist approach to information processing and adjust our course of inquiry accordingly. For example, we know that connectionist networks have the *in principle* power of a UTM (McCulloch & Pitts, 1943), but we also know that *perceptron*-like single layer networks are limited in their computational power (Minsky & Papert, 1969). Thus, we should focus current research on multilayer networks. We know that there are guaranteed algorithms—based very much on early behaviourist theorizing—for training single layer networks (Rosenblatt, 1962), yet no such algorithm exists for multiple layer networks. Can the same be said of biological learning? Finally, we should stop working in the "biological vacuum" and heed the echoing call for models of learning to be based more on the known physiology of the brain.

With the introduction to connectionism's interdisciplinary background—from its philosophical roots to its computational apex—completed, the current state of connectionism can now be evaluated.

Chapter 4

Modern Connectionism

Whilst standing at the crossroads of connectionism, we have discerned several things. First, we have learned that one of the working assumptions of cognitive science is that the mind is an information processor, and as such must be described at three different levels of analysis; the computational, algorithmic, and implementational. Second, we have traced the history of connectionism in order to understand how it has developed as a tool within science (in general terms of describing possible learning mechanisms) and how it has contributed to the growing field of cognitive science (in specific terms of pattern recognition processes). It should be clear at this point that connectionist networks can be classified as information processors.

But, a fairly substantial question remains: Do connectionist networks perform the same type of information processing as humans (in other words, are they of interest to cognitive science)? If we were to leave our description of connectionism at this point, we would be obligated to conclude that connectionist models were *not* equivalent to the information processors studied by cognitive scientists. This is because we have seen how a computational level analysis of the early connectionist networks (e.g., Minsky & Papert, 1988/1969) revealed a severe limitation underlying the network architectures in use. Therefore, if connectionism is to be of value to cognitive science, researchers must seek a method of overcoming these limitations.

Hence, this chapter is concerned with describing connectionist systems in the post-*Perceptrons* era; that is, networks falling under the classification of "New Connectionism". New Connectionism is characterized by computationally powerful networks that can be fully trained. Such networks have often been hailed as providing a simple universal learning mechanism for cognition (but see Gallistel, 1995). Furthermore, because of the computational processing power of these new networks—they are both universal function approximators and arbitrary pattern classifiers—they may be just the type of information processors that are interesting to cognitive science.

As stated earlier, we are living in the aftermath of the neural network revolution. As a consequence, the number of different connectionist architectures available to researchers today is immense; to discuss them all is beyond the scope of this thesis. Instead, this chapter will focus on three specific architectures and provide a cursory examination of four other connectionist architectures. By the end of this chapter, it should be clear that connectionist networks are able to answer the questions regarding information processing systems posed by cognitive scientists.

It should be noted, however, that the demarcation between "Old" and "New" is somewhat tenuous. Following the publication of *Perceptrons*, there was a decrease in the number of researchers actively engaged in connectionist research; but, research did not cease. In certain respects, however, there was a change in the focus of connectionist research. Whereas previous researchers were interested in a connectionist theory of mind, the focus of research during the 1970's and early 1980's was more directed towards a connectionist theory of memory. This is exemplified by the work on associative memory models reported in Hinton and Anderson, 1981. The models described in *Parallel Models of Associative Memory* were seen as a departure from standard memory models of the time for three distinct reasons (e.g., Rumelhart & Norman, 1981):

1. The systems were assumed to have a neurophysiological foundation,
2. The systems offered an alternative to the "spatial" metaphor of memory and retrieval,
3. The systems assumed a parallel, distributed-processing system that did not require a central executive to coordinate processing.

These researchers were aware of the limitations of connectionist models for learning linearly inseparable pattern classification tasks; consequently, the focus of their research was directed more towards how memory was stored and retrieved. In many ways, the work presented in Hinton and Anderson (1981) serves an important role by bridging the gap between *Perceptrons* and *Parallel Distributed Processing*.

Modern Connectionist Architectures

In this chapter, three different network architectures will be described in detail. These architectures were chosen because in subsequent chapters they will play a large role in our analysis of connectionism's contribution to information processing as dictated by the tri-level hypothesis. After each of the main architectures is described, related network architectures will also be reviewed. These reviews will provide somewhat less detail as they are meant to provide only a cursory examination of comparable architectures.

The first architecture to be described in detail is James McClelland's (1981) *Interactive Activation and Competition* (IAC) model of information retrieval from stored knowledge. Although early versions of the IAC architecture did not learn—hence, it could rightly be considered within the class of Old Connectionism as defined earlier—the model displays many characteristics of human cognition that are missing from classical symbolic models (e.g., see Chapter 2). Furthermore, a new learning rule for IAC networks is proposed within this chapter. Thus, it is included here in our description of modern connectionism.

Following the description of the IAC network, the work of two pioneering researchers in the field of neural network learning immediately following the *Perceptrons* era will be briefly reviewed. First, Stephen Grossberg's (1974) instar and outstar configurations and his Adaptive Resonance Theory (ART) networks (1976) will be introduced. Second, we will cover Teuvo Kohonen's (1982) self-organizing maps (which are now commonly referred to as Kohonen networks). Coupled with the new learning rule for the IAC networks, these architectures provide a link from the "Old" to the "New" Connectionism.

The second architecture to be covered in detail is the *generic* PDP architecture; that is, a multi-layered network trained with Rumelhart, Hinton and Williams' (1986a, 1986b) backpropagation algorithm. The generic PDP network is probably the most well known and most widely used architecture today—it is estimated that about 70% of real-world network applications use the backpropagation learning algorithm (Werbos, 1995). Furthermore, the algorithm is suitable for both function approximation tasks and pattern classification problems. Consequently, it has the computational power and competence required by cognitive science.

One criticism leveled against the generic PDP architecture, however, is that it is only capable of a static mapping of the input vectors. The brain, on the other hand, is not stateless but rather a high-dimensional nonlinear dynamical system (Doya, 1995). Consequently, the recurrent network architecture pioneered by John Hopfield (1982) will be briefly discussed. The basic characteristic of recurrent networks is that some processing activation (usually the output) at time *t* is re-used (usually as an input) at time *t*+1. Thus, a fully connected recurrent network is potentially a very powerful architecture for temporal processing; however, more efficient heuristics and algorithms for reliable learning are required.

The third architecture to be discussed specifically is a variation on the generic PDP architecture developed by Dawson and Schopflocher (1992). These *value unit* networks use the same basic learning algorithm as the generic PDP architecture, but use a non-monotonic activation function—the *Gaussian*—in their processing units. This new activation function has been shown to have certain theoretical and practical advantages over standard backpropagation networks. For example, value units are able to solve linearly inseparable problems much easier and with fewer hidden units than standard networks. Also, the hidden unit activations adopted by value unit networks often fall into distinct "bands", allowing for easier interpretation of the algorithms being carried out by the network.

Finally, the last architecture to be briefly covered is the Radial Basis Function (RBF) network (e.g., Moody & Darken, 1989). The reason for covering the RBF architecture is that it and the value unit architecture are often confused. This is because both networks use a Gaussian activation function in their processing units. As the section will show, however, the networks are not equivalent.

By reviewing these different architectures, it will become clear how connectionism has grown into a viable tool for cognitive science. More specifically, in this and following chapters, it will be shown how the IAC, generic PDP, and the value unit architectures contribute to understanding connectionism's new role in cognitive science. These three architectures provide novel ways of approaching classical problems at each level of the tri-level hypothesis.

The Interactive Activation and Competition Model

McClelland's (1981) *Interactive Activation and Competition* (IAC) model illustrates the power of a large network for retrieving general and specific information from stored knowledge of specifics. Although the network rightly falls into the category of Old Connectionism as defined earlier (i.e., the network is hardwired and cannot "learn" new information), it is included in this section because it nicely illustrates those properties of an information processing system that are often overlooked in classical theories. These include graceful degradation, content-addressable memory, output availability, and iterative retrieval. Furthermore, as will be illustrated in Chapter 7, the IAC architecture can be used to answer questions concerning the effects of neurological damage—such as that caused by Alzheimer's Disease—on semantic knowledge.

The basic IAC network consists of processing units that are organized into competitive pools. Connections within pools are inhibitory; this produces *competition* within the pools as strong activations tend to drive down weaker activations within the same pool. Connections between pools, however, are normally excitatory and bi-directional; thus, we have *interactive* processing. Units within the network take on continuous *activation* values between a minimum and a maximum, with their output normally equal to the activation value minus some threshold (although this can be set to zero without loss of generality). The basic mathematics of network functioning are fairly straightforward (e.g., Grossberg, 1978; McClelland & Rumelhart, 1988).

Units within the IAC network compute their activation, a_i, based upon the unit's current activation and the net input. The net input arriving into a unit (Equation 4.1) is calculated by summing the weighted activations sent from all other internal units connected to it plus any external activation supplied by the environment. Thus, the net input to unit i that is connected to j other units is

$$net_i = \sum_j w_{ij}\ output_j + extinput_i \tag{4.1}$$

where w_{ij} is the weight (positive or negative) of the connection between unit i and unit j, and

$$output_j = [a_j]^+ = \begin{cases} a_j \ \textit{if}\ a_j > 0, \\ 0\ \textit{otherwise}. \end{cases}$$

Once the net input to a unit has been calculated, the change in that unit's activation can be computed as follows:

$$\begin{aligned} &\textit{If}\ (net_i > 0),\\ &\qquad \Delta a_i = (max - a_i)net_i - decay(a_i - rest).\\ &\textit{Otherwise},\\ &\qquad \Delta a_i = (a_i - min)net_i - decay(a_i - rest). \end{aligned} \tag{4.2}$$

where *max*, *min*, *decay*, and *rest* are parameters supplied by the modeler. Normally, the parameters are set to $max = 1$, $min \leq rest \leq 0$, and $0 \leq decay \leq 1$. It is also assumed that a_i is initialized and remains within the range $[min, max]$.

With these equations in place, we can evaluate how Δa_i changes over time. For example, imagine that the input, net_i, to a unit is fixed at some positive value. As the activation, a_i, of a unit becomes greater and greater, Δa_i becomes less and less—when a_i reaches *max* then $\Delta a_i = -decay(a_i - rest) = -decay(max - rest)$. When a_i is equal to the resting level, then $\Delta a_i = (max - rest)net_i$. If we assume that $max = 1$ and $rest = 0$, then these equations reduce to $\Delta a_i = -decay$ when a_i is maximal and $\Delta a_i = net_i$ when a_i is minimal. Between these two extremes is the ***equilibrium*** point, where $\Delta a_i = 0$; that is, we can calculate the value of a_i such that given a constant net input, the unit's activation does not change with time. To determine the equilibrium point (assuming $max = 1$ and $rest = 0$), we simply set Δa_i to zero and solve for a_i which gives:

$$\begin{aligned}
0 &= (max - a_i)net_i - decay(a_i - rest) \\
0 &= net_i - (a_i)(net_i) - (a_i)(decay) = net_i - a_i(net_i + decay) \\
a_i &= \frac{net_i}{net_i + decay}
\end{aligned} \tag{4.3}$$

This means equilibrium is reached when the activation equals the ratio of the net input divided by the net input plus the decay. Analogous results to Equation 4.3 are obtained when the net input is negative and constant. It should be noted that equilibrium is only reached when the net input to the unit is constant—if the net input changes with time, then equilibrium is not guaranteed (in practice, however, equilibrium is often achieved).

Having analyzed the mathematical basis of the network, we can now turn our attention to a more specific example of the IAC architecture. McClelland's (1981) network is based on the bi-directional interconnection of nodes. A node is a simple processing device that accumulates excitatory and inhibitory signals from other nodes via weighted connections and then adjusts its output to other nodes accordingly. There are two different types of nodes in the network: *instance* and *property* nodes[1]. There is one instance node for each individual encoded in the network. The

[1] In reality, both of the nodes have the same physical characteristics and therefore only represent different types of information. It is often assumed, however, that instance nodes are 'hidden' from direct access whereas property nodes are not.

instance node has inhibitory connections to other instance nodes and excitatory connections to the relevant property nodes. The property nodes encode the specific characteristics of an individual. Property nodes are collected into cohorts of mutually exclusive values; nodes within a cohort have mutually inhibitory connections. Knowledge is extracted from the network by activating one or more of the nodes and then allowing the excitation and inhibition processes to reach equilibrium.

All information processing takes time, and the IAC model is no exception. Time is measured in "cycles", where a cycle consists of a node computing its activation level (dependent on previous activation, current excitatory and inhibitory signals being received, and a decay function) and then sending a signal to all connected nodes. During a cycle, all nodes are computing their activation levels in parallel. Furthermore, at the completion of any cycle, we can evaluate the current state of the network. This means that we can wait for the network to reach equilibrium and a definite answer, or we can ask what the network's best "guess" to a question is before equilibrium is reached.

The specific network reported by McClelland (1981) encodes information about the members of two gangs called the "Jets" and the "Sharks". Property cohorts include NAME, GANG AFFILIATION, AGE, EDUCATION LEVEL, MARITAL STATUS, and OCCUPATION. Figure 4.1 illustrates the network's architecture and the individual properties within each cohort. Note that McClelland's original network had 27 individuals encoded, while Figure 4.1 only encodes the properties of three individuals.

To illustrate how the network retrieves specific information, we will use Lance as an example. First, the name node "Lance" is activated by an external signal. The node then sends an inhibitory signal to all other name nodes, and an excitatory signal to the instance node for *Lance*. When the instance node receives enough stimulation, it sends an inhibitory signal to all other instance nodes, and an excitatory signal to the properties of Lance, specifically the nodes for "Jets", "20's", "J.H.", "Married", "Burglar" and the name node "Lance". These property nodes send out inhibitory signals to the other nodes within their cohorts and an excitatory signal back to the instance nodes to which they are connected (which means instance nodes other than *Lance* may be activated). Eventually, the network will settle into a state of equilibrium where the properties of Lance will be activated at a high level and all other properties will be relatively inhibited. This is an example of content-addressable memory.

To retrieve general information from the network, we can activate one of the other property nodes. For example, if we wished to find out the average characteristics of members in the Jets Gang, we would simply activate the "Jets" unit and look for the property nodes with the highest

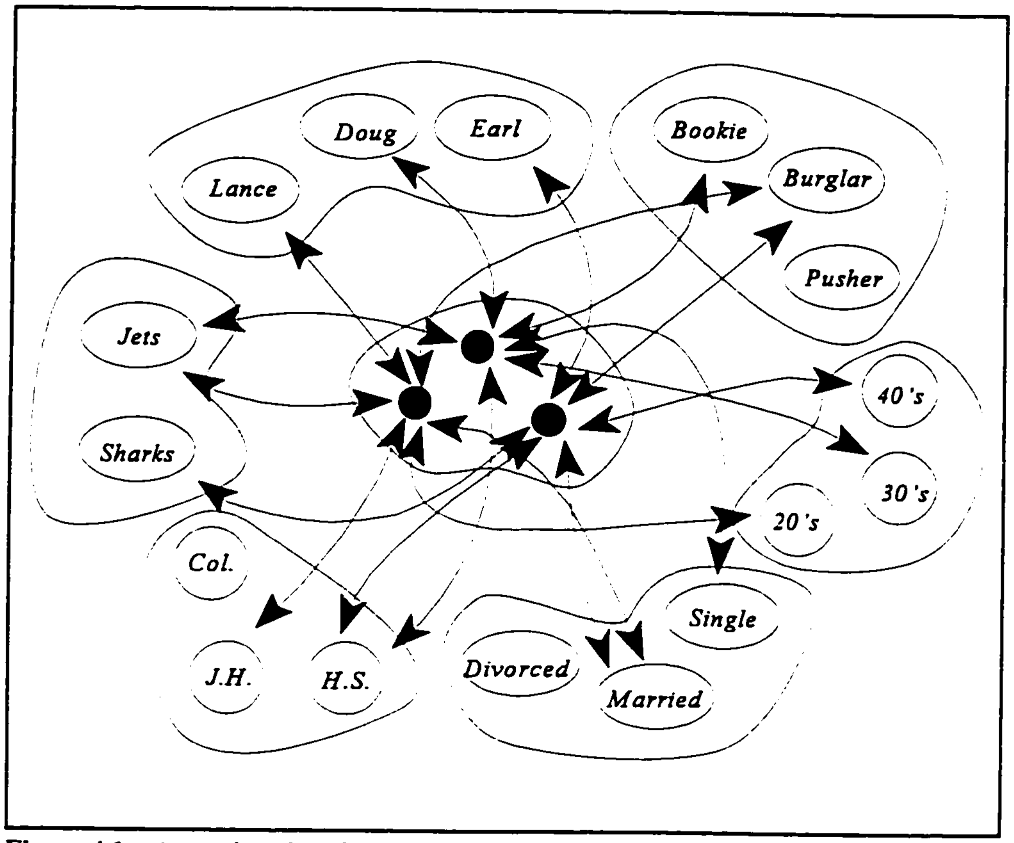

Figure 4.1 A much reduced version of McClelland's (1981) "Jets" and "Sharks" network for illustrative purposes. The solid nodes are "instance units" (one for each gang member) while the hollow nodes are "property units" that encode specific characteristics. Inhibitory connections are not shown.

amount of activation in each cohort. It turns out that the average member of the Jets is in his 20's, has a Junior High education, is single, and is equally likely to be a pusher, a bookie, or a burglar. Furthermore, the network will also tell us who the gang members of the Jets are. This general information is not encoded specifically anywhere within the network; therefore, the model "has no explicit representation that the Jets tend to have these properties" (McClelland, 1981, p. 171) and yet this information is available.

Finally, the IAC model is able to handle incomplete or missing data and even perform when the network has been damaged. If the network is given incorrect information (e.g., inquire about who was a Shark, in their 20's, single, a burglar, and had a Junior High education) it will return with the best match. In this case, the network returns the individual Ken who fits all the characteristics except for education level (McClelland & Rumelhart, 1988). Furthermore, if we sever the connection between the instance node *Lance* and the property "Burglar", the network is still able to return a value of "Burglar" when the property node "Lance" is activated even though there is no direct connection. In effect, Lance will activate other individuals who are similar to him, and thus the network "guesses" at Lance's profession by default assignment.

The IAC network illustrates quite effectively content-addressable memory (e.g., retrieval of the properties of Lance by supplying his name only), output availability (e.g., assessing the state of the network at the end of a cycle), iterative retrieval (e.g., finding the average property of a Jets member from all other possible properties), and graceful degradation (e.g., retrieval of information when connections are severed or incorrect information is given)—properties required in a model of human information processing (Norman, 1986). This model also questions the classical view that we explicitly store generalizations.

Furthermore, the IAC model is similar to the semantic networks of classical cognitive science (e.g., Quillian, 1968; Collins & Loftus, 1975). Both the IAC model and semantic networks store semantic information in a complex network of serial and/or parallel associations (compare Figure 4.1 with Figure 1 of Collins & Loftus, 1975, p. 412). One difference between the two approaches, however, is that the IAC model uses both excitatory and inhibitory connections whereas a semantic network uses excitatory connections solely[2]. This is an important distinction as inhibitory connections can be just as important as excitatory connections; in fact (and quite contrary to what one might intuitively believe), inhibitory connections are necessary for keeping semantically grouped concepts together. For example, in Chapter 7 we will see how recent results from a lesioned IAC network can account for the distortions in semantic networks in patients with Alzheimer's disease (e.g., Chan et al., 1993).

A Possible Learning Mechanism for IAC Networks

Although IAC models are an important contribution to the connectionist's tool bag, the initial models still suffered from the inability to learn and, hence, rightly fall into the classification of Old Connectionism. The ability of the IAC networks to incorporate so many of the characteristics of human information processing, however, make it difficult to dismiss the architecture for lack of a learning mechanism. Consequently, a possible learning mechanism for the IAC networks is proposed here.

In devising a new learning mechanism, we would want to incorporate as many of the known neurophysiological properties (both theoretical and empirical) of learning as possible. The first

[2] For other differences between semantic networks and neural networks, see Barnden (1995).

modification would be to add a Hebbian-like learning mechanism to increase or decrease the weighted connections between nodes, so that those nodes that are active together become more strongly connected (either inhibitory or excitatory), and those nodes that are seldom active together weaken their connection (Hebb, 1949). The second modification would be to limit the maximum possible weight of any connection. The idea behind this restriction comes from Hull's (1943) growth of habit strength and Minsky and Papert's (1988/1969) observation that network weights often grow without bound and there is no evidence that biological neural networks behave in this manner. A third property to be incorporated into a possible learning mechanism would be to prevent weights from shifting sign; that is, weights that are positive remain positive, and weights that are negative remain negative. Finally, a decay process should be added to the learning mechanism to account for the "use it or lose it" property evident in real neural circuits (e.g., Dudai, 1989). Consequently, learning in an IAC network could be accomplished by adding the following equation to the mathematics of the architecture:

$$\begin{aligned}&\mathit{If}\ (w_{ij} > 0),\\ &\qquad \Delta w_{ij} = \eta(wmax - w_{ij})a_i a_j - wdecay(w_{ij}).\\ &\mathit{Otherwise},\\ &\qquad \Delta w_{ij} = \eta(wmin + w_{ij})a_i a_j - wdecay(w_{ij}).\end{aligned} \tag{4.4}$$

where $wmax \geq max$, $wmin \leq min$, and $wdecay \geq 0$ are parameters specific to the network weights, η is a learning parameter, and the unit activations a_i and a_j are assumed to fall into the range $[min, max]$ as before. What Equation 4.4 states is that the change in weight is equal to some proportion of the unrealized weight potential (cf., Hull's growth of habit strength) minus some decay process. Note that this equation guarantees that as the inactivity between nodes persists, weights will approach but never cross zero; in other words, weights that are inhibitory remain inhibitory, and weights that are excitatory remain excitatory. Thus, the network is an unsupervised learning algorithm based on self-organizing principles. A similar learning rule—although less encompassing—for IAC networks has been proposed and tested by Burton (1994) to train a face recognition network.

Therefore, with these modifications, the IAC architecture could be said to bridge the gap between Old and New Connectionism. But, the IAC network is still somewhat limited in the knowledge it can represent; for example, while the architecture represents semantic knowledge quite well, the architecture probably is not suitable for controlling limb movement. Consequently, we need to explore other architectures and learning methods, and the best place to start is with the forerunners to the most common network and learning algorithm today.

Grossberg's Instars and Outstars

Many of the ideas commonly used in artificial neural networks today can be attributed to Stephen Grossberg (1974). One such contribution is the instar and outstar configurations, which were originally proposed as models of certain biological functions. Basically, instars are neurons fed by a set of inputs through synaptic weights, while outstars are neurons driving a set of weights. Instar neurons and outstar neurons are capable of being interconnected to form arbitrarily complex networks.

The purpose of instar neurons is to perform pattern recognition. Each instar is trained to respond to a specific input vector **X** and to no other. This is accomplished by adjusting the weight vector **W** to be like the input vector. The output of the instar is the sum of its weighted connections (see Equation 4.7). This calculation can be seen as the dot product of the input vector and the weight vector, which produces a measure of similarity for normalized vectors. Therefore, the neuron will respond most strongly to the pattern for which it was trained.

An instar is trained using the formula ,

$$w_i(t+1) = w_i(t)+\alpha[x_i-w_i(t)] \tag{4.5}$$

where,

$w_i(t)$ = the weight from input x_i

x_i = ith input

α = training rate coefficient which should be set to 0.1 and then gradually reduced during the training process.

Once trained, the instar will respond optimally to the input vector **X**, and respond to other vectors that are similar to **X**. In fact, if you train it over a set of vectors representing normal variations of the desired vector, the instar develops the ability to respond to any member of that class.

The outstar works on a complementary basis to the instar. It produces a desired excitation pattern for other neurons whenever it fires. To train the outstar, its weights are adjusted to be like a desired target vector

$$w_i(t+1) = w_i(t)+\beta[y_i-w_i(t)] \tag{4.6}$$

where β is the training rate coefficient which should start at 1 be slowly reduced to 0 during training. Ideally, the outstar neuron should be trained on vectors that represent the normal variation of the desired vector. Thus, the output excitation pattern from the neuron represents a statistical measure of the training set and can converge to the ideal vector even if it has only seen distorted versions of the vector.

Grossberg's Adaptive Resonance Theory

It would be hard to mention Grossberg without making a least a brief mention about *Adaptive Resonance Theory* (ART). ART was initially introduced by Grossberg in 1976 as a theory of human information processing: it has since evolved into a series of real-time neural network models that perform supervised and unsupervised category learning, pattern classification, and prediction (Carpenter & Grossberg, 1995).

The simplest ART network is a vector classifier—it accepts as input a vector and classifies it into a category depending on the stored pattern it most closely resembles. Once a pattern is found, it is modified (trained) to resemble the input vector. If the input vector does not match any stored pattern within a certain tolerance, then a new category is created by storing a new pattern similar to the input vector. Consequently, no stored pattern is ever modified unless it matches the input vector within a certain tolerance. This means that an ART network has both plasticity and stability; new categories can be formed when the environment does not match any of the stored patterns, but the environment cannot change stored patterns unless they are sufficiently similar.

There are many different variations of ART available today. For example, ART1 performs unsupervised learning for binary input patterns, ART2 is modified to handle both analog and binary input patterns, and ART3 performs parallel searches of distributed recognition codes in a multilevel network hierarchy. ARTMAP combines two ART modules to perform supervised learning while fuzzy ARTMAP represents a synthesis of elements from neural networks, expert systems, and fuzzy logic (e.g., Carpenter & Grossberg, 1995). Other systems have been developed to suit individual researcher's needs; for example, Hussain & Browse (1994) developed ARTSTAR which uses a layer of INSTAR nodes to supervise and integrate multiple ART2 modules. The new architecture provides more robust classification performance by combining the output of several ART2 modules trained by supervision under different conditions.

Kohonen Networks

A Kohonen network (Kohonen, 1982) can be characterized as a self-organizing map used for pattern recognition. It differs from the generic PDP architecture in several ways (see the following section). First, application of an input vector to the network will cause activation in all output neurons: the neuron with the highest value represents the classification. Second, the network is trained via a non-supervised learning technique. This poses a rather interesting problem. As the training is done with no target vector, it is impossible to tell *a priori* which output neuron will be associated with a given class of input vectors. Once training is completed, however, this mapping can easily be done by testing the network with the input vectors. A typical Kohonen network is illustrated in Figure 4.2.

The *n* connection weights into a neuron are treated as a vector in *n*-dimensional space. Before training, the vector is initialized with random values, and then the values are normalized to make the vector of unit length in weight space. The input vectors in the training set are likewise normalized.

The algorithm for training a Kohonen network can be summarized as follows:

4. Apply an input vector **X** to the network.
5. Calculate the distance D_j (in *n* dimensional space) between **X** and the weight vectors W_j of each neuron.
6. The neuron that has the weight vector closest to **X** is declared the winner. Use this weight vector W_c as the center of a group of weight vectors that lie within a distance of d from W_c
7. Train this group of vectors according to
$$W_j(t+1) = W_j(t) + \alpha[\mathbf{X} - W_j(t)]$$
for all weight vectors within a distance d of W_c. Note the similarity between this equation and Equation (3.12).
8. Perform steps 1 through 4 for each input vector.

As training proceeds, the values of d and α are gradually reduced. It is recommended by Kohonen that α start near 1and reduce to 0.1, whereas d can start as large the greatest distance between neurons and reduce to a single neuron. Furthermore, the number of training cycles should be approximately 500 times the number of output neurons to ensure statistical accuracy.

Because the input and weight vectors are normalized they can be viewed as points on the surface of a unit hypersphere. The training algorithm therefore adjusts the weight vectors surrounding the winning neuron to be more like the input vector. In other words, the algorithm tends to cluster weight vectors around the input vector.

Such adaptive units can be organized into a layer to produce a feature map. A feature map is a nonlinear method of representing the original signal space and resembles the topographic maps found in many areas of the brain (Ritter, 1995). The feature map is produced by the unsupervised training of the adaptive units which gradually develop into a spatially organized array of feature detectors whence the position of the excited units signal statistically important features of the input signal. Consequently, more frequently occurring stimuli will be represented by larger areas in the map than infrequently occurring stimuli.

Kohonen maps and unsupervised learning are but one way of training connectionist networks. But, if both the input and corresponding output patterns are known *a priori*, then supervised learning can be used. The most common supervised learning algorithm is the backpropagation algorithm used to train generic PDP networks.

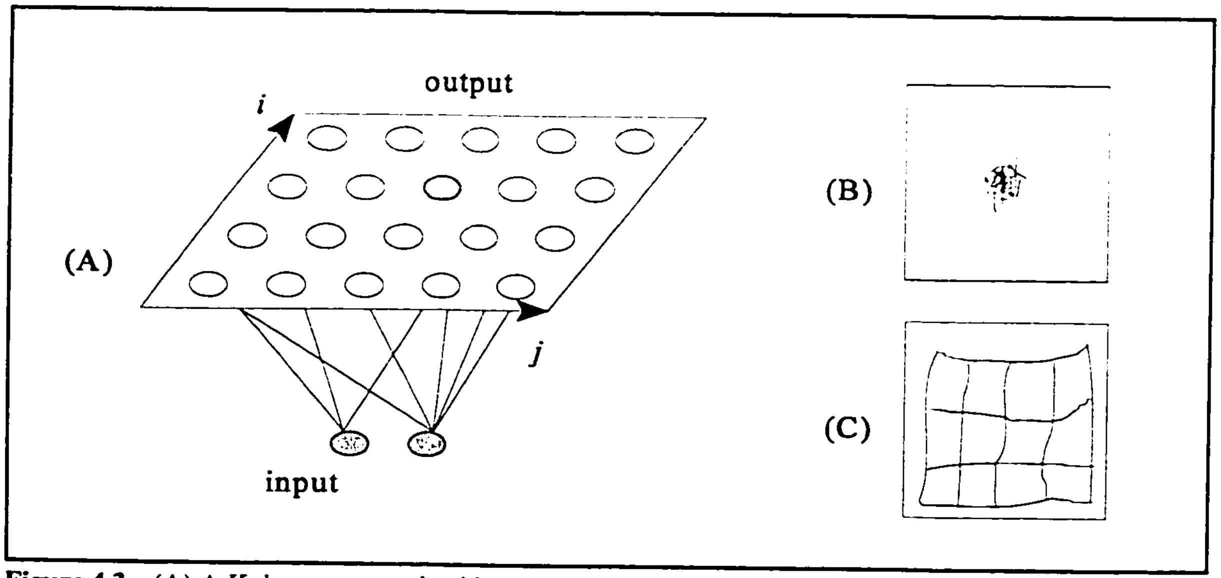

Figure 4.3 (A) A Kohonen network with two inputs mapping onto a 4 x 5 output field. (B) Randomized weight structure before training. (C) Typical weight structure following training.

The Generic PDP Network

As we saw in Chapter 3, an elementary α-perceptron has the *in principle* power to solve any pattern classification problem expressed in binary notation, whereas a network with less depth is limited in its computational power. This increase in computational ability derives from the fact that a multilayer network can theoretically carve a pattern space into an arbitrary number of decision regions (Lippman, 1987). Furthermore, it can be shown that multilayer networks are also universal function approximators—that is, they are able to solve any function approximation problem to an arbitrary degree of precision (e.g., Cybenko, 1989; Hartman, Keeler, & Kowalski, 1989; Hornik, Stinchcombe, & White, 1989). These results are specific to the network architecture alone, and not to the learning rule used to train the networks.

Thus, we need to make a distinction between the network architecture and the learning rule. This move serves a dual purpose. First, it allows us to make claims about the computational power of networks regardless of the training procedure used. Second, we can evaluate the learning rule independent of the network architecture. The consequence of making this distinction between architecture and learning rule is that it allows us to (i) address concerns about the "biological plausibility" of certain learning algorithms (e.g., backpropagation) without compromising the interpretation and final results of the trained network, and (ii) determine if differences in network performance are due to architectural discrepancies or modifications of the learning algorithm. Therefore, we will first define the generic connectionist architecture, and then define the learning rule.

The Generic Connectionist Architecture

The building block for the generic connectionist architecture is the artificial neuron (see Figure 2.6). The functional properties of the artificial neuron mimic those of actual neurons; that is, the neuron receives and integrates information, processes this information, and transmits this new information (e.g., Dudai, 1989; Levitan & Kaczmarek, 1991). Mathematically, the input function to the neuron is expressed in Equation 4.7; net_{pj} is a linear function of the output signals, o_{pi}, from units feeding into j with weighted connections, w_{ij}, for pattern p.

$$net_{pj} = \sum_i w_{ij} o_{pi} \qquad (4.7)$$

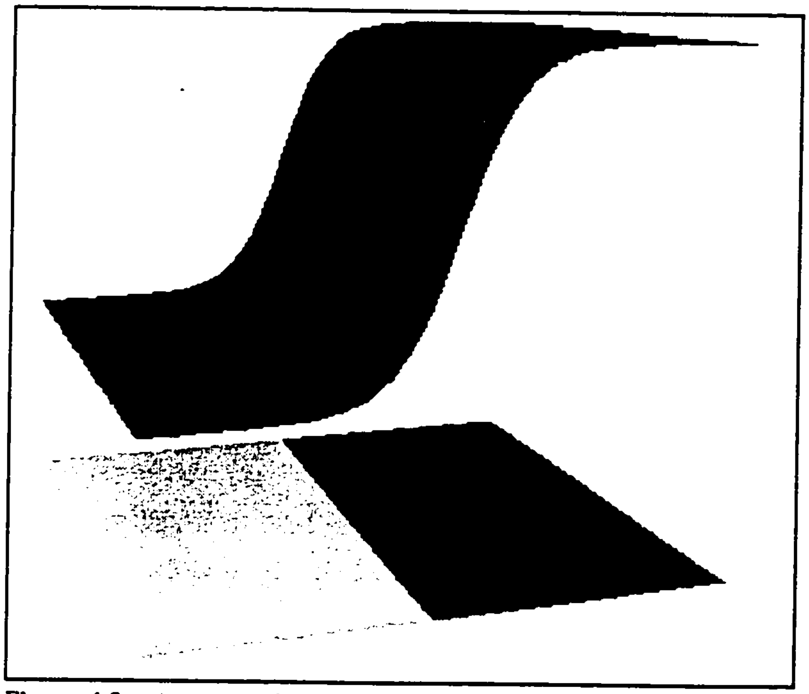

Figure 4.3 A monotonic activation function (such as the logistic) divides a pattern space into two distinct regions.

The output function of the neuron is a non-linear function of its input and is expressed in Equation 4.8. Most learning rules require that the function be both differentiable and monotonic. Consequently, the most common activation function used is the logistic or sigmoid function which compresses the range of the net input so that the output signal lies between 0 and 1. This function allows the network to process large signals without saturation and small signals without excessive attenuation. Thus, in Equation 4.8, o_{pj} is the output of the neuron, net_j is the input, and θ_j is the "bias" of the unit which is similar in function to a threshold.

$$o_{pj} = f(net_{pj}) = (1 + e^{-net_{pj} + \theta_j})^{-1} \qquad (4.8)$$

Units that use a function such as the logistic have an **order** of 1 (Minsky & Papert, 1988/1969) and are able to carve a pattern space into two distinct regions (see Figure 4.3). Thus, networks using this form of activation function can solve linearly separable problems without any hidden units. These networks have been termed *Integration Devices* by Ballard (1986), and generic connectionist networks by Anderson and Rosenfeld (1988).

The power of these simple units emerges when they are connected together to form a network, or *multi-layer perceptron* (MLP). The most common MLP is a feed-forward architecture consisting of an input layer, an internal or hidden layer, and an output layer (see Figure 3.7B); such networks are often referred to as three-layer networks, although this nomenclature is not always agreed upon[3]. Units in one layer propagate their signals to units in the next layer through uni-directional, weighted connections. Normally, connections do not exist within layers, nor do they transcend more than one layer (i.e., from the input layer directly to the output layer); however, exceptions do exist.

Furthermore, it is assumed for simplicity that processing within the network occurs in discrete time intervals. It is further assumed that all processing is done in parallel; that is, all signals pass through the connections from one layer to the next at the same time and all units in a layer process their activations at the same time. Thus, the speed of processing—in terms of how long it takes the network to solve a problem—is directly proportional to the number of layers in the network, not the number of processing units. A three layer network with 5,000 units theoretically takes the same number of time steps to compute its function as a three layer network with five units (practically, this is not the case as networks are often modeled using serial computers). Consequently, parallel processing in neural networks is often hailed as a solution to the 100-step constraint[4] plaguing classical models.

Although MLP's have the required computational competence for cognitive scientists to find them interesting, their real allure lies in their ability to learn. Various training techniques have been proposed previously (e.g., Selfridge's supervised and genetic learning; Rosenblatt's reinforcement rule; Widrow and Hoff's Delta Rule), but they have all been limited to training only one layer of weights while keeping the other layers constant. What connectionism needed to move into the mainstream was a general learning rule for networks of arbitrary depth.

The Generalized Delta Rule

Papert's (1988) likening of *Perceptrons* to the huntsman being sent out to bring back Snow White's heart is appropriate, for the huntsman did not return with the heart of Snow White, but the

[3] For example, Wasserman (1989) argues that since input units do not compute any function, they should not be counted as a layer; therefore, he calls these two-layer networks.

[4] The 100-step constraint is based on the processing speed of neurons (e.g., Feldman & Ballard, 1982). Most complex behaviours occur in a few hundred milliseconds—this means entire behaviours are executed in less than a hundred time steps as opposed to the millions of time steps required by classical models.

heart of a deer. Similarly, connectionism was not slain by *Perceptrons*, it was just quietly minding its time until its prince came. And, for connectionism, Prince Charming turned out to be the *Generalized Delta Rule* (GDR).

The GDR (Rumelhart, Hinton, and Williams, 1986a, 1986b) can be considered one of the most significant contributions to connectionist research: It has allowed the training of generic multilayer networks. In fact, the work of Rumelhart et al. is often cited as the catalyst for the strong resurgence of connectionist research in the latter half of the 1980's (e.g., Bechtel & Abrahamsen, 1991; Horgan & Tienson, 1996). As the name implies, the GDR is a generalization of the Widrow-Hoff Delta Rule for training networks of Adaline units (Widrow & Lehr, 1995). The training procedure, however, is commonly referred to as backpropagation of error, or backpropagation (backprop) for short.

Although Rumelhart et al. are often credited with popularizing the GDR, the backpropagation procedure itself was formulated previously on at least three separate independent occasions: first by Werbos in 1974[5], then by Parker in 1982[6] and finally by LeCun in 1986[7]. In fact, the GDR is simply a basic form of backpropagation. In its more general form (e.g., Werbos, 1995), backpropagation contributes to the *prediction* and *control* of large systems (in terms of optimal planning and reinforcement learning), and not simply to supervised learning as is often assumed. Consequently, backpropagation can be applied to any differentiable, sparse, nonlinear system—it is not restricted to any specific form of MLP, nor is it restricted to artificial systems. The main advantage of backpropagation over traditional methods of error minimization is that it reduces the cost of computing derivatives by a factor of N, where N is the number of derivatives to be calculated. Furthermore, it allows higher degrees of nonlinearity and precision to be applied to problems.

Werbos (1995) notes that since backpropagation is used in so many different applications, its actual definition has often become muddled and confused. Therefore, he offers these two standard definitions (p. 135, his italics):

[5] *Beyond regression: New tools for prediction and analysis in the behavioral sciences.* Masters thesis, Harvard University.

[6] *Learning logic*, Invention Report S81-64, File 1, Office of Technology Licensing, Stanford University, Stanford, CA.

[7] Learning processes in an asymmetric threshold network. In E. Bienenstock, F. Fogelman Souli, & G. Weisbuch (Eds.) *Disordered systems and biological organization*. Berlin: Springer.

1. Backpropagation is a procedure for *efficiently* calculating the derivatives of some output quantity of a nonlinear system, with respect to all inputs and parameters of that system, through calculations proceeding *backwards* from outputs to inputs. It permits "local" implementation on parallel hardware (or wetware).
2. Backpropagation is any technique for adapting the weights of parameters of a nonlinear system by somehow using such derivatives or the equivalent.

What we are concerned with, however, is the special form of backpropagation for training neural networks. Werbos (1995) calls this the *basic* form of backpropagation, although most researchers today simply refer to it as backprop. The GDR, as applied to neural networks then, is a supervised learning algorithm (cf., Widrow & Hoff's delta rule—Equation 3.6) used to adjust the weights in an MLP in accordance with the *Principle of Minimal Disturbance*[8]. To begin, a training vector is presented to the network via the input units and the activations are then passed through weighted connections to the hidden units. The net input function to the hidden units is computed (Equation 4.7), the activation function is applied, then the output signal is generated (Equation 4.8) and propagated to the output units. The output units then use Equations 4.7 and 4.8 to produce a final output signal, o_{pj}, which is compared to the desired target output, t_{pj}. The total error, E, is defined in Equation 4.9, where p is an index over the patterns being presented, j is an index over output units, o is the actual state of the output, and t is the desired (target) state of the output.

$$E = \frac{1}{2}\sum_{p}\sum_{j}(t_{pj} - o_{pj})^2 \qquad \textbf{(4.9)}$$

Learning is defined therefore as the minimization of this error term by gradient descent through an error surface in weight space. Gradient descent is described by the relation $\mathbf{W}_{k+1} = \mathbf{W}_k + \mu(-\nabla_k)$ where $\mathbf{W}_k$ is a weight vector, μ is a parameter that controls stability and rate of convergence and ∇_k is the value of the gradient of the sum squared error (SSE) surface at $\mathbf{W}_k$. Consequently, to begin gradient descent, an initial weight vector, $\mathbf{W}_0$, is defined and the gradient of the error surface at this point is measured. Weights are then altered in the direction opposite to the measured gradient,

[8] *Principle of Minimum Disturbance*: Adapt to reduce the output error for the current training pattern, with minimal disturbance to responses already learned. (Widrow & Lehr, 1995, p. 719). It is noted that unless this principle is followed it is difficult to store the required pattern responses simultaneously; hence, learning becomes problematic.

producing a new weight vector based upon the above relation. Every time this procedure is repeated with a newly calculated weight vector, $\mathbf{W}_k$, the SSE is caused to be reduced on average and moves towards a minimum. Because the true gradient is often impractical and inefficient to obtain, the instantaneous gradient is often computed based on the square of the instantaneous error. The instantaneous gradient is used because it is an unbiased estimate of the true gradient and is easily computed from single data samples (Widrow & Lehr, 1995).

Therefore, to minimize E by gradient descent, the partial derivative of E with respect to each weight within the network needs to be computed. For a given pattern, p, this partial derivative is computed in two passes: a forward pass using Equations 4.7 and 4.8, and a backward pass which propagates the derivatives back through the layers; hence, backpropagation of error.

The backward pass begins by first differentiating Equation 4.9 which gives

$$\partial E_p / \partial o_{pj} = t_{pj} - o_{pj}$$

and then applying the chain rule to compute

$$\partial E_p / \partial net_{pj} = \partial E_p / \partial o_{pj} \cdot \partial o_{pj} / \partial net_{pj}.$$

The second term of the above equation is produced by differentiating Equation 4.8 which produces

$$\partial o_{pj} / \partial net_{pj} = f'_j(net_{pj}) = o_{pj}\ (1 - o_{pj}).$$

Therefore, the effect on the error due to a change in the total input to an output unit is known. But, as the total input is simply a linear function of the output from previous layers and the related connection weights, the effect on the error due to a change in the previous outputs and weights can be computed. For a weight w_{ij} from unit i to unit j, the derivative is

$$\partial E_p / \partial w_{ij} = \partial E_p / \partial net_{pj} \cdot o_{pi}$$

and the effect of all connections emanating from unit i is simply

$$\partial E_p / \partial o_{pi} = \sum \partial E_p / \partial net_{pj} \cdot w_{ij}.$$

Thus, two different error signals can be defined depending on if the unit is an output unit or an internal unit. For an output unit, the error signal is

$$\delta_{pj} = (t_{pj} - o_{pj}) f'_j(net_{pj}) \qquad \textbf{(4.10)}$$

whereas for an internal unit, the error signal becomes

$$\delta_{pj} = f'_j(net_{pj}) \sum_k \delta_{pk} w_{kj} \qquad \textbf{(4.11)}$$

Hence, weights in the network are changed by

$$\Delta_p w_{ij} = \eta \delta_{pj} o_{pi} \tag{4.12}$$

where η is a learning parameter to scale the weight change, and Equation 4.10 is used for output units and Equation 4.11 for internal units. Finally, Equation 4.13 shows how learning can be improved by adding a momentum term, α, which uses the previous weight changes to influence the current changes.

$$\Delta_p w_{ij}(t) = \eta \delta_{pj} o_{pi}(t) - \alpha(\Delta_p w_{ij}(t-1)) \tag{4.13}$$

The weights of the network can be updated after every pattern presentation, or after the entire pattern set has been presented. Typically, training of the network continues until convergence is reached. For function approximation problems, convergence is measured by a sufficiently small total sum of squared errors (SSE) as computed by Equation 4.9. For pattern classification problems, convergence is attained when the network correctly classifies all input patterns (or at least a significant portion thereof). The performance of the network is normally assessed by the number of "sweeps" or "epochs" the network uses to solve the problem, where a sweep is defined by the single presentation of the entire training set.

Thus, the GDR overcomes the earlier limitations of Old Connectionism by allowing multilayer networks to be trained on any information processing problem. As Minsky and Papert (1988) point out in their Epilogue to *Perceptrons*, however, many problems still exist with the GDR and the generic PDP architecture. One problem is that the GDR searches through an error space using gradient descent; although gradient descent on average moves towards a minimum it is not guaranteed to move towards a global minimum. In other words, it is neither dependable nor efficient, though there are techniques for trying to improve this (e.g., Werbos, 1995). Another problem is that networks with only one layer of hidden units trained with the GDR still must violate the limited order constraint to solve linearly inseparable problems.

Although basic backpropagation is powerful enough to solve a wide variety of problems, much work is done on improving the performance of artificial neural networks especially in regards to three specific characteristics:

i. *Generalization*: the ability to predict data outside the original training set,

ii. *Learning Speed*: increasing the convergence rate, especially for systems learning from real-time experience, and

iii. *Fault Tolerance*: the ability to perform despite noise or breakage.

The general rule of thumb—at least from an engineering perspective—for the first two characteristics is to make the networks as simple as possible: use fewer connections and smaller weights. The third characteristic, on the other hand, is trickier to pin down. For example, the ability to perform despite noise can be seen as the ability to generalize; thus, smaller network structure would seem to be the answer. Performance despite breakage, however, requires larger network structures with some form of redundancy built in. We will see in Chapter 8 how this redundancy may actually help with the first two characteristics as well.

As mentioned earlier, one problem with generic PDP networks is that they are static: that is, previous inputs have no effect on new inputs (except during the training period). Consequently, the standard generic PDP architecture may not be appropriate for modeling some time dependent tasks, such as recognizing a pattern of sounds as forming a word. Therefore, the recurrent network architecture will be introduced.

Recurrent Networks

A recurrent network is defined as one in which the network's output is fed back into the network. Figure 4.4 shows the structure of one type of recurrent network. In this network, inputs are received from an external source, passed to a hidden layer, and then on to the output layer. The signal from the output layer is passed to an external source, as well as back to a state layer which then acts as an input layer (along with the actual input layer) to the hidden layer on the next pass.

As the output of the network at time (t) is used along with a new input to compute the output of the network at time $(t+1)$, the response of the network is dynamic. That is, the network's response can be stable (successive iterations produce smaller and smaller output changes until the outputs become constant) or unstable (the network's outputs never cease changing). This stability issue proved a problem for early researchers, but Cohen and Grossberg (1983) devised a theorem showing that at least a subset of recurrent networks were guaranteed to produce outputs with stable states. Stable networks are typified by weight matrices that are symmetrical along the main diagonal, with diagonal weights of zero (i.e., $w_{ij} = w_{ji}$, $w_{ii} = 0$).

Much of the early work on recurrent networks was pioneered by John Hopfield (1982). In fact, some have argued that it was because of Hopfield's stature as a well-known physicist that neural network research was made respectable again (Anderson & Rosenfeld, 1988). Hence, certain configurations of recurrent networks are referred to as Hopfield nets. One problem that plagued earlier versions of Hopfield networks, though, was that the networks tended to settle into local minimum instead of the global minimum. To combat this problem, one can change the weights statistically instead of deterministically. This technique is known as simulated annealing, and networks trained using this method are known as Boltzmann machines (e.g., Hinton & Sejnowski, 1986).

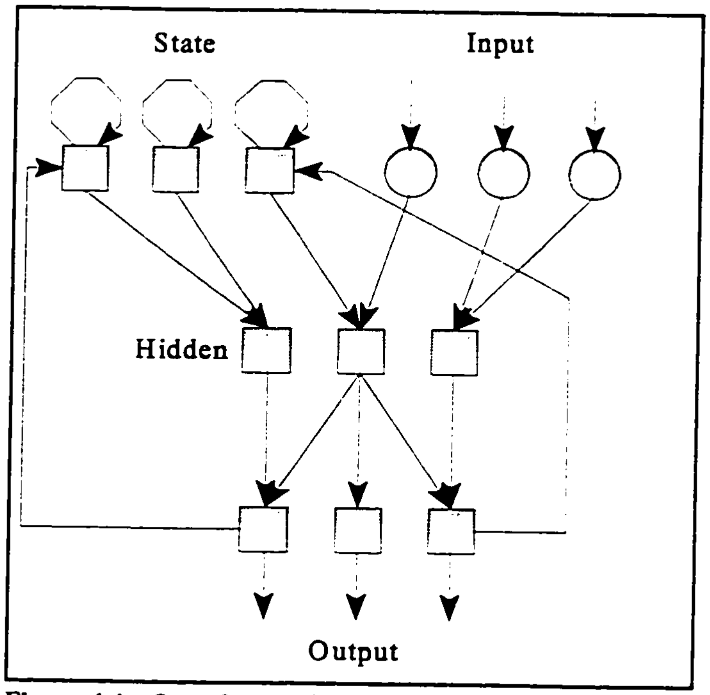

Figure 4.4 One form of recurrent network architecture in which connections from output to state units are one-for-one. Note that not all the connections are shown.

It has been shown that recurrent networks can simulate finite state automata (e.g., Cleermans, Servan-Schreiber, & McClelland, 1989) and that one can construct a second-order recurrent network such that internal deterministic finite-state automata state representations remain stable (Omlin & Giles, 1996). Furthermore, it has been proven that finite size recurrent networks can simulate any multi-stack Turing Machine in real time and non-deterministic rational nets can simulate non-deterministic Turing Machines (Siegelmann & Sontag, 1994).

Adding recurrent connections to the generic PDP architecture is but one way of improving the performance of such networks. Another way is to use different activation functions within the processing units. Such an approach was taken by Dawson and Schopflocher (1992) when developing the value unit architecture.

Value Unit Networks

Despite their immense theoretical power as universal function approximators and arbitrary pattern classifiers, networks trained with the GDR suffer from severe practical training problems. Networks are prone to local minima and notoriously slow if they do find a solution. One reason for this behaviour is the limitations imposed on the processing units by the GDR—processing units must have a function that is both differentiable and monotonic. Consequently, the most commonly used activation function for processing units is the *logistic* (see Equation 4.6). This choice of activation function is normally motivated by engineering principles; for example, the *logistic* function is chosen because it fulfills the requirements of the learning rule, while similar functions—such as *tanh*—are chosen simply for their ability to improve performance in terms of learning speed over the *logistic.*

But, we could also adopt a different perspective and choose an activation function based upon neurophysiological evidence. Evidence from single-unit recordings (that is, record the output of the neuron with respect to its input) suggests that there are at least two functionally different types of neurons in the brain in regards to their output encodings (e.g., Ballard, 1986). This can be illustrated by comparing the recordings from neurons that function as a basic part of the oculomotor system to neurons in the visual areas of the cortex.

The first type of neurons—for example, those in the servo system controlling eye movement—have linear outputs whose *firing rate* is proportional to a scalar parameter such as the rate of eye rotation. These neurons could be characterized as *summation* or *integration* devices (Ballard, 1986) and have the equivalent activation function as the *logistic* used in artificial neurons. The outputs of such neurons have two features—larger values mean more frequent pulses, and the output is one dimensional. From a physiological perspective, these neurons use *frequency* encoding. In other words, neurons using a monotonic activation function could be viewed as encoding *variables.*

In contrast, neurons in visual areas of cortex use fundamentally different encodings for their output. These neurons have multidimensional receptive fields[9]; that is, if the input stimulus is within a receptive field, the neuron will increase its firing rate, otherwise it remains at its baseline firing rate. The firing rate is specifically determined by the degree of match between the stimulus and

[9] In this respect, a receptive field of a neuron is defined in terms of *all* the neuron's inputs, including possible feedback connections from neurons in other parts of the cortex. This is in contrast to the normal interpretation of receptive field which limits itself to the inputs from a specific stimulus.

receptive field—the stronger the match, the stronger the firing rate. From a physiological perspective, neurons with this type of firing pattern use *spatial* or *place* encoding. In other words, neurons using a nonmonotonic activation function could be viewed as encoding *values*. Consequently, Ballard (1986) terms these neurons *value units*.

As "the value unit way of representing information seems to be a property of most cortical cells" (Ballard, 1986, p. 68), the logical move from a cognitive science perspective would be to incorporate this type of activation function into a connectionist network.

The Value Unit Architecture

In considering a nonmonotonic activation function for artificial neurons, the most likely choice would be the *Gaussian*. Such an activation function is readily apparent not only within the cones of the eye (e.g., see Davidoff, 1995), but also within the tuned neurons in the visual cortex (Hubel & Wiesel, 1959). From a computational perspective, the *Gaussian*

$$o_{pj} = G(net_{pj}) = e^{-\pi(net_{pj} - \theta_j)^2} \qquad \textbf{(4.14)}$$

where net_{pj} is the same as in Equation 4.7 and θ_j is the bias of the activation function, has the advantage of being able to carve a pattern space into three decision regions (see Figure 4.5). Consequently, such an activation function could be said to be *limited* **order** 2; it is of limited order because the planes are restricted to parallel cuts in the pattern space.

The nonmonotonicity of the activation function buys the value unit networks certain theoretical and practical advantages over standard integration device networks. For example, the fact that a single value unit can subdivide a pattern space into three regions by placing two parallel hyperplanes within the pattern space means that the processing power of the unit is increased. Whereas standard integration device networks require a hidden unit for every **order** of the problem (e.g., a 4-parity problem is **order** 4 and therefore requires four hidden units), networks with value units in both the hidden and output layers require considerably fewer. In fact, for problems such as parity which require parallel cuts of the pattern space, the number of hidden units needed is

(**order** div 2) - 1

where the operation *div* returns the quotient. Therefore, a solution to the XOR problem can be represented in a network without any hidden units, and a solution to the 4-parity problem can be

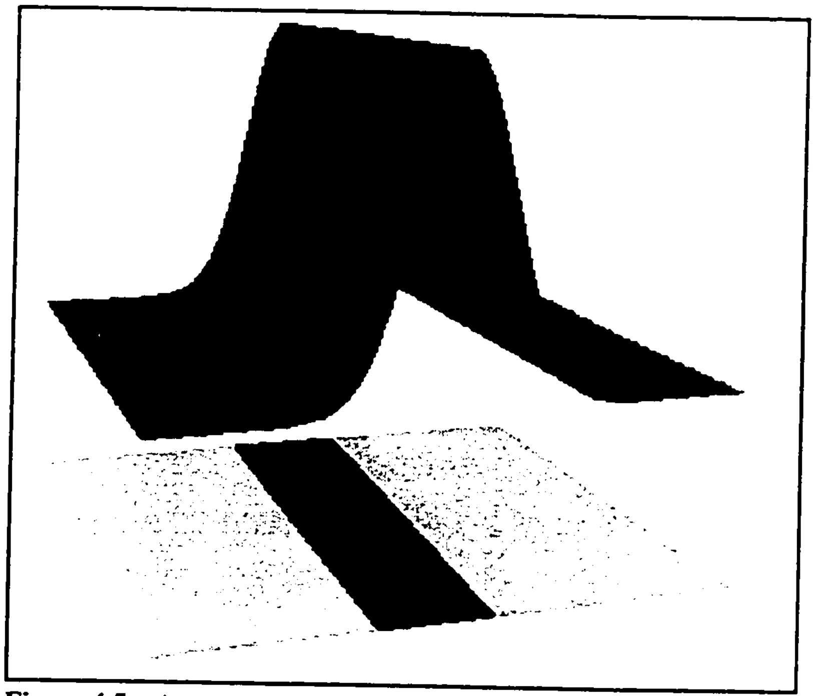

Figure 4.5 A non-monotonic activation function (such as the Gaussian) carves a pattern space into three regions.

represented in a network with only one hidden unit! Moreover, the added processing power of the value unit means that the limited order constraint need not be violated; that is, it is possible to solve the parity problem without any hidden unit connected to every input unit.

On the other hand, the nonmonotonicity of the activation function does limit the value unit architecture. Because the *Gaussian* is not invertible, a value unit network cannot perform function approximation tasks in the traditional sense—it is limited to pattern classification tasks only (but see Chapter 5). In all other respects, however, the value unit architecture is the same as the generic PDP architecture, including its ability to be fully trained by a variation of the GDR.

Modifying the Generalized Delta Rule

Normally, replacing $f'(net_{pj})$ in Equations 4.10 and 4.11 with the first derivative of the *Gaussian* causes a network to fall into a local minimum which asserts that some property of the pattern space p is not true, but fails to assert the some property of p is true. To avoid these local minima, Dawson and Schopflocher (1992) modified the GDR by adding a second term to the standard GDR's error function to produce a new cost function, C_p. The first component of Equation 4.15 is the standard cost function used in the backpropagation algorithm and measures the failure of the network to match the observed output o_{pj} with the target output t_{pj}. The second component of

C_p measures the network's failure to set $net_{pj} = \theta_j$ (basically, the mean of the *Gaussian*) when the desired output is equal to 1: It essentially prevents the unit's activation from falling towards either negative or positive infinity.

$$C_p = \frac{1}{2}\sum_{j=1}^{n}(t_{pj} - o_{pj})^2 + \frac{1}{2}\sum_{j=1}^{n} o_{pj}(net_{pj} - \theta_j)^2 \tag{4.15}$$

The new learning rule is therefore based on changing the weights such that the cost function in Equation 4.15 is minimized. Consequently, the desired weight change for the connection originating from unit i and terminating at unit j for given pattern p can be computed as

$$\Delta_p w_{ij} = \eta(\delta_{pj} - \varepsilon_{pj})o_{pi} \tag{4.16}$$

where $\delta_{pj} = (-t_{pj} - o_{pj})G'(net_{pj})$ is equivalent to δ_{pj} in Equation 4.8 with the exception that the derivative is of the Gaussian and not the logistic. The ε_{pj} term is equal to $T_{pj}(net_{pj} - \theta_j)$ and is the augmented error-minimization function from Equation 4.11. Similarly, the unit's bias term can be modified using the equation

$$\Delta_p \mu_j = -\eta(\delta_{pj} - \varepsilon_{pj}) \tag{4.17}$$

by treating the parameter μ_j as a connection weight between output unit j and some other unit whose activation is always 1. Furthermore, it can be shown that, at the end of training, the error function minimized is equivalent to that of the GDR's. These modifications allow a network trained with the backpropagation algorithm to use nonmonotonic activation functions.

Value Unit Performance

Value unit networks have been applied to a wide variety of pattern classification tasks, from "toy" problems (Dawson & Schopflocher, 1992; Dawson, Schopflocher, Kidd, & Shamanski, 1992; Medler & Dawson, 1994a, 1994b), to diagnosing Alzheimer's patients from SPECT data (Dawson et al., 1994), to identifying logical problems (Berkeley, Dawson, Medler, Schopflocher, & Hornsby, 1995), to classifying mushrooms (Dawson & Medler, 1996). One of the surprising aspects of the value unit architecture is that, from an engineering perspective, they have been shown to converge faster and more reliably on linearly inseparable problems than the more traditional MLPs that use monotonic activation functions. Furthermore, value unit networks show better generalization, and better ability to be "scaled up" from toy problems.

To quickly show the processing power of value units over standard integration devices on linearly inseparable problems, a small experiment was conducted using four different network architectures to solve the 4-parity problem (see Figure 4.6). The first network (A) was a standard integration device network with four hidden units. The second network (B) had the same structure (i.e., 4 hidden units), but used value units instead. The third network (C) used the minimal value unit architecture of one hidden unit to solve the problem. Finally, the fourth network (D) used two value

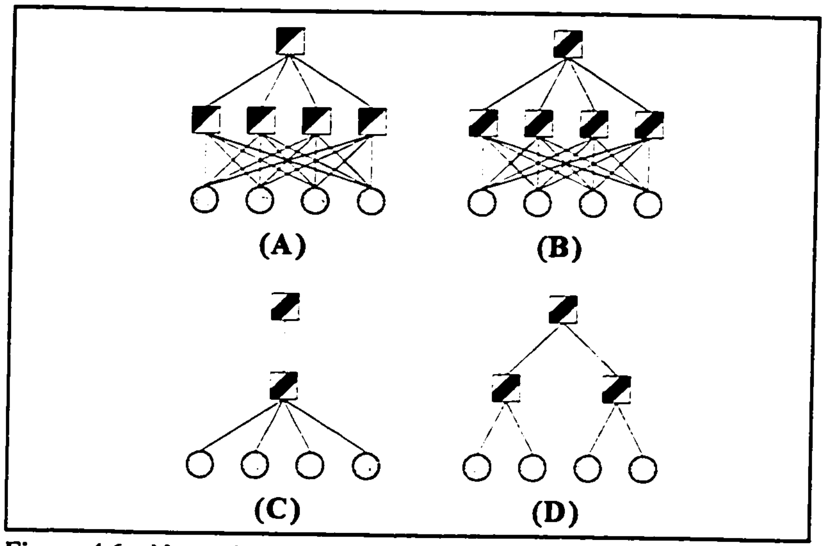

Figure 4.6 Network architectures for solving the 4-parity problem. Network (A) is an integration device, while networks (B), (C), and (D) are value unit networks. Note that (A) and (B) have identical network structure (except for the processing unit) and that (D) does not violate the limited order constraint.

units, but did not violate the limited order constraint; that is, no hidden unit was connected to every input unit. All networks were trained to a criterion of 0.0025 and had an upper limit of 10,000 sweeps imposed. The integration device network was trained with momentum $\eta = 0.9$, and learning rate $\alpha = 0.5$. The value unit networks were trained with $\eta = 0$, and $\alpha = 0.01$. For all networks, biases were started at 0, and weights were randomized from a square distribution between [-1, 1]. Table 4.1 reports the results of the experiment; specifically the convergence percentage (50 trials each) and the minimum, median, and maximum number of sweeps to reach convergence.

Table 4.1

Performance of Networks Trained on the Four Parity Problem

		Sweeps		
Architecture	**% Convergence**	**Min**	**Med**	**Max**
A	24	1929	3607	9246
B	100	46	213	621
C	88	98	267	620
D	64	133	134	209

The first thing to note is that all value unit networks, regardless of network topology, outperformed the integration device network both in terms of convergence rate and speed of convergence. In fact, when the integration device network did converge—which was only approximately one quarter of the time—it took an order of magnitude longer to do so. At this point, it would be easy to be lulled into the engineering approach to connectionism; that is, extolling the virtues of one architecture over another based solely on performance. As stated earlier, however, we are interested in connectionism from a cognitive science perspective. Therefore, we should approach the performance of the value unit architecture from a tri-level hypothesis perspective.

Consequently, from a computational level analysis, value unit networks show more competence than integration device networks in solving linearly inseparable problems such as parity. This competence will be more fully explored in Chapter 5. At the algorithmic level of analysis, value unit networks are able to solve the parity problem using a number of different network topologies. As it has been argued that networks are algorithms (Rumelhart & McClelland, 1985), this means that the different network topologies are different algorithmic descriptions for solving the 4-parity problem. Choosing the correct algorithm (network architecture) simply becomes a matter of comparing the computational competence between systems we are modeling. More importantly, however, the value unit architecture has other algorithmic advantages as will be shown in Chapters 6 and 7. Finally, at the implementational level, the fourth value unit network architecture (D) satisfies Minsky and Papert's (1988/1969) limited order constraint, effectively addressing one of their concerns about neural networks.

The Radial Basis Function Network

One of the misconceptions surrounding the value unit architecture is based upon its use of a *Gaussian* activation function. This is because another network architecture, the *Radial Basis Function* (RBF) network (e.g., Moody & Darken, 1989) uses a similar activation function. That, however, is where the similarities end.

The RBF network is a three-layer feedforward network that uses a linear transfer function for the output units and a nonlinear transfer function (normally the *Gaussian*) for the hidden units. The input layer simply consists of *n* units connected by weighted connections $\{\mu_{ij}\}$ to the hidden layer and a possible smoothing factor matrix $\{\Sigma_j\}$. A hidden unit can be described as representing a point **x** in *n*-dimensional pattern space. Consequently, the net input to a hidden unit is a distance measure between some input, $\mathbf{x}_p$, presented at the input layer and the point represented by the hidden unit; that is, $net_j = \|\mathbf{x} - \mathbf{x}_p\|$. This means that the net input to a unit is a monotonic function as opposed to the nonmonotonic activation function of the value unit. The *Gaussian* is then applied to the net input to produce a *radial* function of the distance between each pattern vector and each hidden unit weight vector. Hence, a RBF unit carves a hypersphere within a pattern space whereas a value unit carves a hyperbar.

In general, an RBF network can be described as constructing global approximations to functions using combinations of basis functions centered around weight vectors. In fact, it has been shown that RBF networks are universal function approximators (Girosi & Poggio, 1990). Practically, however, the approximated function must be smooth and piecewise continuous. Consequently, although RBF networks can be used for discrimination and classification tasks (see Lowe, 1995, for some examples), binary pattern classification functions that are not piecewise continuous (e.g., parity) pose problems for RBF networks (Moody & Darken, 1989). Thus, RBF networks and value unit networks are not equivalent.

Modern Connectionism: Conclusions

The major turning point in connectionist research occurred with the discovery of methods for training multilayer networks. With this discovery, connectionist models not only had the computational power to answer those questions interesting to cognitive science, but also had a method of *learning* how to answer those questions. Thus, there is an explicit distinction between network architectures and the learning rules used to train them in modern connectionism. This distinction will be further elaborated in the following chapters.

Specifically, Chapter 5 will consider the computational power of the integration device and value unit network architectures. Chapters 6 and 7 will elaborate on the algorithmic and functional architecture descriptions of neural network architectures. Finally, Chapter 8 will look at the implementational issues surrounding some of the learning rules (e.g., the biological plausibility of backprop), as well as architectural design decisions for neural networks.

Chapter 5

The Computational Analysis of Connectionism

I was left with a deep respect for the extraordinary difficulty of being sure of what a computational system can or cannot do. (Papert, 1988, p. 11)

Our working assumption that the mind is an information processor requires us to adopt the tri-level hypothesis in order to explain cognition. Thus, if connectionism is to be an effective tool within cognitive science, it has to contribute to explanations at all three levels of analysis—including the functional architecture. In the first half of this thesis, the relationship between connectionism and the tri-level hypothesis has only been alluded to. Consequently, the next four chapters of this thesis will explicitly address the tri-level hypothesis from a connectionist perspective, ultimately showing that connectionism may be just the tool required by cognitive science. This chapter begins this by focusing our attention on the first level of the hypothesis: the computational analysis of an information processor.

The computational analysis of an information processor is dependent on many factors: the problem being solved, the available data, and the natural constraints of the world to name but a few (e.g., Richards, 1988). Consequently, it is often difficult to separate what is and is not appropriate for this level of analysis. Therefore, as a guide, we will use Marr's (1982) definition of the computational level of analysis

> . . . the abstract computational theory of the device, in which the performance of the device is characterized as a mapping from one kind of information to another, the abstract properties of this mapping are defined precisely, and its appropriateness and adequacy for the task are demonstrated. (p. 24)

Thus, guided by the above definition, this chapter will examine how connectionism contributes to the computational analyses of information processors. It will be shown both theoretically and empirically that connectionist networks possess the computational power to be of interest to cognitive science. That is, they can solve problems in both function approximation and pattern classification domains.

These results alone, however, beg the question: "What does connectionism contribute to the computational level of analysis that the classical approach does not?" Indeed, it has been shown that all computations that can be produced by a local system (e.g., a classical model) can also be produced by a distributed system (e.g., a connectionist model), yet the distributed system has no more computational power than the local system (Winograd & Cowan, 1963). Similarly, Hopcroft and Ullman (1979) have stated that the distinction of distributed versus specific representations has no importance at the computational level dealt with by psychology.

Following these statements, it would be tempting to assume that adopting a connectionist approach does not add anything to the computational level analysis—so, why bother adopting it? From a computational perspective, however, other important distinctions besides distributed processing exist between the classical and connectionist approaches. Indeed, there are two reasons why the connectionist approach to computational analyses should be adopted.

First, it has been argued that it is tractable computability (i.e., computations that can be performed on a physical device), and not computability in the mathematical sense, that is assumed in cognitive science (Horgan & Tienson, 1996). Nevertheless, this assertion does not alter our general analysis of computation as tractably computable functions are merely a subset of all computable functions. What this does do, on the other hand, is lead us to the theory of natural computation instead of formal computation. Natural computation is concerned with not only the computational abilities of an information processor, but also the biological basis behind computation. Connectionism offers an analogous medium in which to explore natural computation, something that is lacking in classical approaches.

Second, connectionist systems are based on very simple processing elements. This is an important fact for establishing explanatory versus merely descriptive theories of computational systems (Seidenberg, 1993). In short, explanatory theories provide a reason *why* a computation is the way it is instead of simply describing the computation. Furthermore, the underlying principles governing the explanatory theory in one domain will transcend to other domains as well. Thus, it is argued that the proper analysis of connectionist models (which ultimately will include all levels of the tri-level hypothesis) has the potential of not only explaining previous cognitive theories but also formulating new cognitive theories as well.

To begin our analysis of the computational level, however, we will start with the very basics: the mapping from one kind of information to another. In terms of information processing, this transformation of information may come in two general forms. First, one stream of continuous information (e.g., the position of an object in space) may be mapped into another stream of continuous information (e.g., moving the eye muscles in order to track the object). This type of transformation is analogous to computing a function. Second, relatively large quantities of information (e.g., size, shape, colour) may be mapped into discrete information (e.g., ball). This can be characterized as classifying patterns of information. These two transformations form the basis of the computational analysis of an information processor.

Inputs, Outputs, and Computations

At the most basic level, every information processor can be described as computing some function. In other words, a computational description of an information processor will be the analysis of some function approximation problem; that is, given some set of inputs, what is the function that produces the observed outputs? The inputs and outputs of such a function can be bounded or infinite, discrete or continuous. A special case of function approximation occurs if the outputs of the function are *bounded and discrete*—this is normally referred to as pattern classification.

Given this definition, it is tempting to relate function approximation to stimulus-response theory. A stimulus-response table could be built from the observed inputs and outputs, and the function would then simply be the corresponding stimulus-response model[1]. In fact, Suppes (1969) has suggested that stimulus-response models are theoretically isomorphic to *finite automata*, where a finite automaton can be regarded as a black box that receives a finite number of inputs, adopts a

[1] The fundamental axiom of stimulus-response theory is that for any stimulus-response table, there is a stimulus-response model that becomes isomorphic to it as $n \rightarrow \infty$.

finite number of internal states, and transmits a finite number of outputs (see also Bever, Fodor, & Garrett, 1968). Arbib (1964) mathematically defines a finite automaton, A, as the quintuple

$$A = (I, O, S, \lambda, \delta)$$

where

I is a finite set of inputs,

O is a finite set of outputs,

S is a finite set of internal states,

$\lambda : S \times I \rightarrow S$ is the next-state function,

$\delta : S \times I \rightarrow O$ is the next-output function.

A is assumed to work on a discrete time scale, so if at time (t) it is in state q and receives as its input a, then at time (t+1) it will change to state $\lambda(q, a)$ and transmit signal $\delta(q, a)$.

The importance of Suppes' claim is apparent when it is recognized that both classical (e.g., TOTE hierarchies) and connectionist (e.g., modular networks) information processing models can be shown to be equivalent to finite automata. This implies that concepts embodied in both classical and connectionist approaches can be subsumed into stimulus-response models. Here, it would seem, our quest for a definition of function approximation ends.

Arbib (1969) has shown, however, that although stimulus-response models may be theoretically isomorphic to finite automata, they are, in practice, less powerful. The reason for this discrepancy between theory and practice arises because of two powerful practical issues. First, the actual time required for the stimulus-response models to reach asymptote becomes prohibitive with large models. Second, the number of conditioning states grows exponentially with the number of internal states being subsumed (e.g., with m internal states and p inputs, the number of conditioning states is $p^{2(m+1)p}$). When these issues are considered, it becomes clear that simply recording the performance of the organism does not mean that we have described it at a computational level. As Richards (1988) states

> We cannot claim to understand its function simply by recording its performance. We must also understand the concepts that are to be embodied in the perceptual machine—our brains. (p. 4)

In other words, not only do we need to be aware of the mapping of input onto output, we also need to precisely define the abstract properties of this mapping (see also Fodor, 1968). To do this, we shall return to the Turing machine (see Chapter 2) and the formal aspects of computational theory. A TM is composed of a finite automaton, A, an infinitely long ticker tape, and a device for manipulating symbols on the tape. The ticker tape allows for an infinite memory, and the read-write device allows manipulation of the environment. Hence, a TM is more powerful than a finite automaton and precisely describes the types of computations that cognitive science is interested in. To fully understand the computational analysis of a TM as it pertains to information processing, we will turn to computational theory.

A Formal Definition of Computation

Computational theory can be formally divided into two areas: *computability* (you cannot possibly compute all the things you want) and *complexity* (that which you can compute is too expensive). In other words, computability seeks to characterize those entities that can be computed and complexity measures the efficiency of those computations (e.g., Bridges, 1994; Ballard, 1997). In the next two subsections, a formal analysis of both of these concepts will be developed. These concepts will then be related to information processing theory by way of function approximation and pattern classification.

Computability

The modern study of computability originated in the year 1936 with the introduction of three mathematically precise notions (the Turing machine, Kleene's partial recursive functions, Church's lambda calculus) that tried to capture the informal aspects of computable partial functions (Bridges, 1994). It has been proven that these three notions and all subsequent formalizations characterize the same computable partial functions from $\mathbb{N}^n$ to $\mathbb{N}$; therefore, all notions of computability can be stated in terms of a Turing machine (e.g., Rogers, 1987; Johnson-Laird, 1988; Bridges, 1994).

Before we proceed with the introduction to computability, however, the basics of set theory is required (e.g., Halmos, 1974). A set, X, is defined as a collection of entities (also called a string), $x_1, x_2, \ldots x_n$, such that $x_i \in X$ and x_i is the i^{th} term. It is assumed that there is a unique empty string, Λ, with no terms over X. A nonempty finite set X is called an *alphabet*, and the set of all strings over X is denoted by X^*. The sets that are of most interest to computability theory are the natural, rational, and real number sets which are defined as:

i. the set of natural numbers: $\mathbb{N} \equiv \{0, 1, 2, \ldots\}$,

ii. the set of rational numbers: $\mathbb{Q} \equiv \{\pm m/n: m, n \in \mathbb{N}, n \neq 0\}$, and

iii. the set of real numbers: $\mathbb{R}$

It should be noted that set theory naturally extends to finite-dimensional vector spaces, where $x \in X$ is a *scalar*, and a *vector* is defined as set of scalars $(x_1, x_2, \ldots x_n)$. The n-fold Cartesian product of X (i.e., $X \times X \times \ldots \times X$; n factors) is known as a *vector space* and is symbolized by X^n, where $n \in \mathbb{N}$ and n is known as the dimensionality of the space (Halmos, 1987).

The most basic relationship between two sets, A and B, is that of equality. This is characterized by the following axiom:

> **Axiom of extension**: Two sets are equal ($A = B$) if and only if they have the same elements, otherwise they are not equal ($A \neq B$).

Furthermore, if A and B are sets and if every element of A is an element of B, then A is called a *subset* of B and is indicated by $A \subset B$. If $A \subset B$ and $A \neq B$ then A is called a proper subset of B. The creation of subsets is the basic principle behind set theory. In other words, given a set B and a condition $S(x)$, A is a subset of B defined by $A = \{x \in B\colon S(x)\}$. This symbolism is summed up by the

> **Axiom of specificity**: To every set B and every condition $S(x)$ there corresponds a set A whose elements are exactly those elements x of B for which $S(x)$ holds true.

This axiom is important because in computability theory, we are often forced to deal with subsets as opposed to entire sets.

This is especially the case when dealing with functions. A function can be defined as the relation f such that, given sets X and Y, the domain of f is X, the range of f is Y, and for each $x \in X$ there is a unique element $y \in Y$ with $(x, y) \in f$. Note that this definition implies that every element within X maps onto one and only one element in Y, but does not preclude the fact that many x may map onto several—or only one—y, nor that every $y \in Y$ has a corresponding $x \in X$. When describing a function, we shall use the notation $f{:}X \to Y$ which is an abbreviation for "f is a function from X to Y". An alternative notation often used in the literature is $f(x) = y$. Finally, It should be noted that the term *function* is reserved for the undefined object that is somehow active, and the ordered pairs (x, y) are called the *graph* of the function.

In computability theory, we are most often concerned with functions that are defined on subsets of X^n instead of the entire set X^n. Consequently, this leads to the definition of a *partial function* φ from a set A to a set B. The domain of the partial function $\varphi{:}A \to B$ is a subset of A and its range is a subset of B. The partial function $\varphi(x)$ is *defined* if $x \in$ domain (φ) and *undefined* if x

$\in A$ and $x \notin$ domain (φ). Partial functions can be operated on in the normal sense; That is, given partial functions φ and ψ from $\mathbb{N}$ to $\mathbb{N}$ the operations

sum: $(\varphi + \psi)(n) = \varphi(n) + \psi(n)$,

product: $(\varphi \cdot \psi)(n) = \varphi(n) \cdot \psi(n)$, and

composite: $\varphi \circ \psi(n) = \varphi(\psi(n))$

hold true as long as $\varphi(n)$ and $\psi(n)$ are both defined for sum and product, and $\psi(n)$ is defined and belongs to domain (φ) for composite.

Using these basics of set theory, we can now define computability in terms of the Turing Machine; more specifically, a deterministic binary TM. Thus, given a finite input alphabet $X = \{0, 1\}$ and a finite tape alphabet $Y = \{0, 1, \not{b}\}$ where $\not{b}$ is a distinguished blank element, we define a Turing Machine as the quadruple

$$\text{TM} \equiv (Q, \delta, q_0, q_F)$$

where

Q is a finite set of states,

δ is a partial function $Q \times Y \rightarrow Q \times Y \times \{L, R, \Lambda\}$,

q_0 is the start state $(q_0 \in Q)$, and

q_F is the halt state $(q_F \in Q)$.

The partial function δ: $Q \times Y \rightarrow Q \times Y \times \{L, R, \Lambda\}$ is a state transition function where L, R, and Λ are interpreted as *left move*, *right move*, and *no move* respectively. Hence, at any given instance, the TM will be in state q with the string $u \in Y^*$ to the left of the read/write head, the string $v \in Y^*$ under and to the right of the read/write head, and blanks to the right of v. Consequently, the instantaneous configuration of the TM is fully described by the triple (u, q, v) and the state transitions of the TM are simply the successive configurations $C \vdash C'$ given by the sequence of triples.

Thus, a computation is defined in terms of a TM as a finite sequence of admissible configurations $(C_0, C_1, \ldots, C_n)$ such that

$C_0 = (\Lambda, q_0, v)$ for some $v \in X^*$,

$C_i \vdash C_{i+1}$ for each i, and

C_n is of the form (Λ, q_F, v') for some $v' \in X^*$

where v is the input, v' is the output, C_0 the initial configuration and C_F is the final configuration of the computation. If C_F is reached, then we say that TM completes the computation $(C_0, C_1, \ldots, C_n)$ on the input v.

Using this definition of a computation, we can define the partial function computed by a Turing machine. That is, given TM $\equiv(Q, \delta, q_0, q_F)$, with input alphabet X, tape alphabet Y, and $S \subset X^*$, the partial function $\varphi: S \rightarrow X^*$ is said to be computed by TM if TM completes a computation on the input $s \in S$, where $\varphi(s)$ is the output of that computation; otherwise, $\varphi(s)$ is undefined. In other words, we say that a partial function $\varphi: \mathbb{N}^n \rightarrow \mathbb{N}$ is Turing machine computable if it is a partial function from $\mathbb{N}^n$ to $\mathbb{N}$computed by some binary TM. Furthermore, if $m > 1$, then we say that a partial function $\varphi: \mathbb{N}^n \rightarrow \mathbb{N}^m$ is Turing machine computable if the functions $P_k^m \circ \varphi: \mathbb{N}^n \rightarrow \mathbb{N} (k = 1, \ldots, m)$ are Turing machine computable.

Following from this definition comes perhaps the most significant thesis in computational theory, the Church-Turing thesis:

> A partial function $\varphi: \mathbb{N}^n \rightarrow \mathbb{N}$ is computable (in any accepted informal sense) if and only if it is computable by some binary Turing machine—that is, if and only if $\varphi = \varphi_k$ for some k. (Bridges, 1994, p. 32).

It should be noted that the Church-Turing thesis cannot be proven: it is an unsubstantial claim that all notions, either formal or informal, of a computable partial function from $\mathbb{N}^n$ to $\mathbb{N}$ are equivalent to the formal notion of a Turing machine computable partial function. It is an accepted thesis, however, because (1) all attempts to formalize the intuitive notion of a computable partial function have led to the same class of functions, and (2) the lack of any convincing example of a computable partial function that is not also Turing machine computable.

As stated earlier, this definition of a function means that there is a countably infinite number of computable functions, but it also leaves an uncountably infinite number of noncomputable functions. For example, you can enumerate all the natural numbers, but you cannot enumerate all the reals (however, you can compute some of the reals). Hence, a TM must operate within these bounds and any definition of a computable function within information processing theory must also operate within these bounds. Knowing what we can or cannot compute is only half the battle, however, as we must also know what practical "cost" is associated with each computation.

Complexity

Complexity is a measure of the efficiency of a computation. In practice, complexity is measured in terms of some appropriate resource—time, space, or memory—used in the computation (Bridges, 1994). Although complexity is often dependent on assumptions ranging from the architecture of the machine, to the size of the problem, to the criterion used in evaluating solutions, the abstract theory of complexity is machine independent. Nevertheless, the basic models of complexity are often related to TM computations such as the number of steps required to solve the problem or the number of distinct cells visited by the read/write head. Hence, the TM will figure prominently in our formal analysis of the computational level.

Consequently, a complexity measure can be defined such that $\Gamma \equiv \gamma_0, \gamma_1, \gamma_2, \ldots$ is an infinite sequence of computable partial functions γ_i: $\mathbf{N} \rightarrow \mathbf{N}$ that satisfies Blum's axioms:

B1: for each i, domain (γ_i) = domain (φ_i).

B2: the function COSTS: $\mathbf{N}^3 \rightarrow \{0, 1\}$ defined by

$$\text{COSTS}(i, n, k) \begin{array}{ll} = 1 & \text{if } \gamma_i(n) = k; \\ = 0 & \text{otherwise} \end{array}$$

is computable.

The computable partial function γ_i is called the complexity function or cost function associated with φ_i. If the function $\varphi_i(n)$ is undefined, then the corresponding cost is considered to be undefined as well. Furthermore, both of these axioms are independent of one another; that is, one cannot be deduced from the other. These axioms form the basis of complexity theory and relates the concept of complexity to functions and Turing Machines.

Many more theoretical proofs exist about the formal nature of complexity (see Bridges, 1994); however, for our purposes we are more interested in some of the more practical aspects concerning complexity. For instance, in evaluating the complexity of an algorithm, cost is measured as a function of the size of the input. Thus, "easy" problems—in terms of computational steps required—are bounded by a polynomial function of their input whereas "hard" problems are based on an exponential function of their input. Unfortunately, most interesting problems turn out to be hard and the best solutions for all but the smallest problem sizes are prohibitive to compute (Ballard, 1997).

With the more formal aspects of computability and complexity dealt with, we can now turn towards the crux of the computational level of analysis: trying to figure out what function a system is computing. In other words, we want to be able to describe the function approximation task the information processor is engaged in.

Function Approximation

Using the notation of both set theory and computability theory, function approximation is the process of defining the undefined object (i.e., the function) to an arbitrary degree of precision given a set of ordered pairs (i.e., the graph). In other words, given a set of input-output pairings, we want to approximate the function that reproduces the output when given the input.

Thus, in terms of our previous definition, the graph of the function would be the inputs and outputs encoded on the tape of the TM, and the undefined process or function is the TM in its entirety (including the portion of tape that it has used for potential calculations). Therefore, when McCulloch and Pitts (1943, p. 131) state that "specification of the net would contribute all that could be achieved in that field" they are talking about defining the function of cognition. A function is now defined in terms of information processing theory.

A distinction—albeit a picky one—needs to be made between *function approximation* and *function calculation*. In function calculation, the exact values of a function are computed. Function calculation requires the precise function to be known *a priori* and is really only useful in mathematical analyses. What cognitive science is interested in, however, is how an information processing system approximates a function. That is, the systems of interest are not infallible and therefore are not calculating functions, but rather are approximating functions. Furthermore, information processors can learn new functions (e.g., an infant learning to reach) and modify old functions (e.g., an adult relearning to reach to a point in space after a prism has been placed in front of the eyes) through a series of approximations.

There are many formal proofs that connectionist networks have the *in principle* computational power of an UTM (e.g., McCulloch & Pitts, 1943). It has even been argued that, theoretically, connectionist networks are more powerful than TM's. In truth, however, function approximation in neural networks has practical limitations.

Function Approximation—Empirical Results

To this point, we have discussed only the theoretical aspects of connectionist function approximation. In Chapter 1, however, it was argued that connectionist must engage in both theoretical and empirical work to be effective researchers as theories will dictate the types of experiments to run and experiments will constrain our theories. Another reason for this claim is to distinguish between the *in principle* power and the *in practice* power of our models. In other words, a theory only works if it can be put into practice.

Consequently, the results from several different experiments will be reported to illustrate the computational abilities—from a more formal perspective—of connectionist networks in function approximation. In total, six different experiments were conducted to test not only function approximation but also interpolation and extrapolation of both integration device networks and value unit networks.

Approximating a Function

All networks were trained on the function

$$h(x) = 0.8 * \sin(\pi x) \tag{5.1}$$

using 21 evenly-spaced data points over the interval [-1, 1]. Thus, domain(h) = [-1, 1] and range(h) = [-0.8, 0.8]. It should be noted, however, that the output function of most standard processing units is limited to the range [0, 1]. Consequently, for the first three experiments, Equation 5.1 was normalized so that it fell within the range [0, 1].

Training a network normally proceeds as follows. A training pattern is presented at the input units and these values are propagated through the network with appropriate modifications to the output units. For supervised learning, the actual output is then compared to the desired output. At this point, there is a slight divergence in training practices. One approach that is commonly used is to create a running total of the *Sum Squared Error* (SSE), and then make the appropriate weight changes once all patterns in the training set have been presented (this is referred to as a training epoch or sweep). This approach has the advantage that training algorithms using gradient descent will modify the weights such that the total SSE after weight changes will be less than—or at least equal to—the previous SSE. Furthermore, the presentation order of the training patterns does not matter as it is the final SSE that weight modifications are based upon. Consequently, the performance of the network, as measured in terms of either a sufficiently small total SSE or the number of correct outputs, can be evaluated during the training sweep. If the performance criterion is reached, then weight modifications are not required.

A second approach—and the one used throughout the experiments in this thesis—is to make the weight changes after each presentation of a training pattern. This approach actually has several disadvantages. First, it is not guaranteed that the overall change in total SSE will be less as the weight changes made after presentation of pattern p may be reversed by the changes made after presentation of pattern $p+1$. As a result of this, a test sweep must be run after each training sweep to effectively evaluate the performance of the network. Second, the presentation order of the training data becomes important; consequently, the most effective training method is to randomize the presentation order on each training sweep. In practice, this approach requires much more 'bookkeeping' during training which has the potential to increase the practical time required to train the network to criterion. This approach, however, does have one major advantage from a cognitive science perspective, that of biological plausibility. All indications from neuropsychological experiments (e.g., Dudai, 1989) point to the fact that neurological changes occur more or less immediately after presentation of a stimulus, and not after every single stimulus from a training set has been presented. Indeed, how would an organism ever know when a specific training set had ever been completely presented? Therefore, this approach sacrifices engineering for biology.

For function approximation problems, a stringent "hit" criterion is required. One way to measure this is to require a sufficiently small total SSE. This method leaves the possibility that one pattern could produce a grossly inaccurate response and yet the total network SSE be sufficiently small enough to reach criterion. Another method for evaluating performance is to require the difference between actual and desired output for each training pattern to be within a certain range. For these experiments, a very strict "hit" criterion of 0.000025 was used. This means that a hit was only recorded if the absolute difference between the actual and desired output was 0.005 or less. Using such a stringent hit criterion makes it highly unlikely that a network will correctly classify all outputs; therefore, all networks were trained to a maximum number of 10,000 sweeps. Overall performance was then evaluated against the actual function.

Experiment 1: Integration Device

In this first experiment, the practical ability of integration device networks to function approximate was evaluated. As most theoretical proofs of function approximation in networks are based upon some arbitrarily high number of hidden units (e.g., Cybenko, 1989; Hartman, Keeler, & Kowalski, 1989; Hornik, Stinchcombe, & White, 1989), this experiment examined the relationship between the number of hidden units in the network architecture and network accuracy in function approximation.

Network Architecture: Five different network architectures were constructed for this experiment. All networks had one input unit and one output unit, but differed in the number of hidden units (0, 1, 2, 5, or 10). The network was trained with the backpropagation algorithm presented in Chapter 4, with the modification on the updating rule as described above (i.e., updating

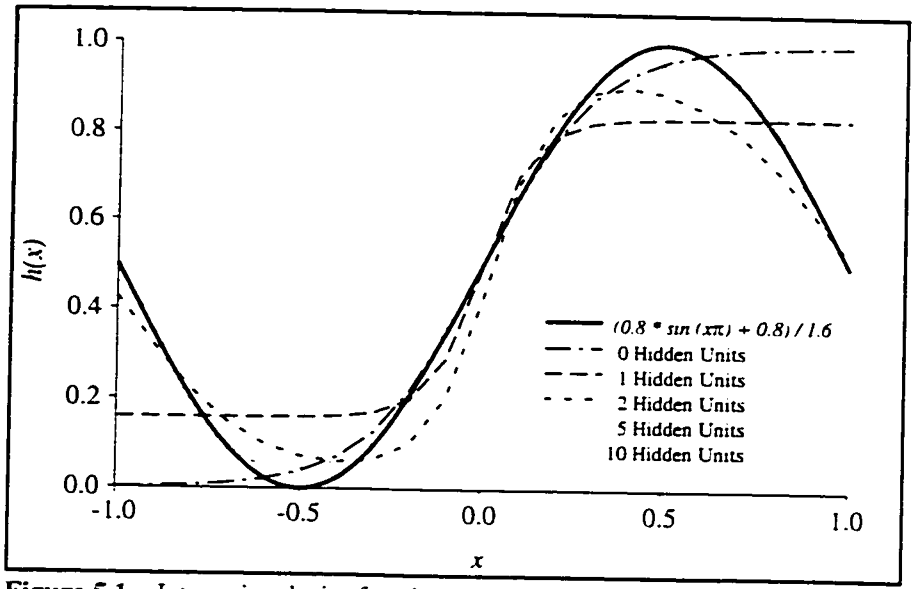

Figure 5.1 Integration device function approximation as a function of number of hidden units.

after each pattern presentation). All weights and biases in the network were randomly initialized between -1.0 to 1.0. The network learning rate was set to 0.1 and the momentum was set at 0.9.

Network Performance: Figure 5.1 shows the performance of the five different networks trained to approximate the normalized function. As can be seen, the network without any hidden units could not approximate the function. This is not surprising as a single integration device unit is limited to producing a sigmoidal function (which a network without hidden units approximates quite well). A similar looking function is produced by the network with only one hidden unit, although the function appears to be limited in its range. Again, this is to be expected because a single hidden unit is limited to placing a single hyperplane within the problem space, which limits the network to approximating a monotonic function only.

Adding a second integration device unit to the hidden unit layer should theoretically allow the network to approximate a basic nonmonotonic function as the second unit allows a second hyperplane to be positioned in an orientation opposite to the first hyperplane within the problem space. As can be seen in Figure 5.1, empirical results support this theory: an integration device network with two hidden units can indeed approximate a nonmonotonic function. Although the basic shape of the function is correct, the network has some difficulty approximating the function at the extreme values (i.e., 0.0 and 1.0) of the function. This would suggest that more hidden units are required.

In fact, with five hidden units the network approximates the function almost perfectly. There is some discrepancies at the extreme values once again, but this is relatively minor when compared to the network with two hidden units. This discrepancy suggests nevertheless that more hidden units may be required. As can be seen from Figure 5.1, however, a network with ten hidden units still has problems with the extreme values of the function and is not that much more accurate than the network with five hidden units. Again, this suggests that either more hidden units are required, or more sweeps through the pattern set are required. For now, however, we can conclude that an integration device network with at least five hidden units can successfully approximate the normalized version of the function given in Equation 5.1.

Experiments 2 & 3: Value Unit

In the next two experiments, the value unit architecture was evaluated for its potential to approximate functions. Because the value unit uses a nonmonotonic activation function, it is not uniquely invertible, and therefore it has been suggested that value units are not suitable for function approximation (Dawson & Schopflocher, 1992). On the other hand, the RBF unit—which also uses a nonmonotonic activation function—is routinely used for function approximation (Moody & Darken, 1989; Lowe, 1995). It must be remembered, however, that the basis underlying the RBF network is different than the value unit (see Chapter 4). Consequently, empirical trials were run to see if value unit networks are able to approximate functions *in practice*.

Network Architecture: Both Experiments 2 and 3 used the same network layout (i.e., 0, 1, 2, 5, and 10 hidden units) and input/output values as Experiment 1, the only difference being the Gaussian activation function used in the processing units. Networks were initialized with a bias of 0.0 and weights between -1.0 to 1.0. Learning rate was set to 0.001 and momentum was equal to 0.0.

In Experiment 2 the modified generalized delta rule for training networks of Gaussian processing units was used (see Chapter 4). It was noted, however, that the second component of the modified cost function (Equation 4.15) actually serves to penalize any value that is not zero or one with more of a penalty assessed to those values closer to zero. That is, if o_{pj} is 0 or net_{pj} is 1, then the second component drops out of the cost function. If these values are not 0 or 1, then the second component acts to pull the unit's activation away from zero and towards one. Although this modification of the cost function works well for pattern classification tasks that require outputs of zero or one, it was hypothesized that the modification will necessarily penalize any function approximation problem requiring continuous values between zero and one. Thus, Experiment 3 dropped the modification to the cost function from the training procedure.

Network Performance: Figure 5.2 shows the results of standard value unit function approximation (Experiment 2) while Figure 5.3 shows the results of value unit function approximation with the modification to the cost function removed (Experiment 3). The first thing to note in Figure 5.2 is that a value unit networks with at least one hidden unit can approximate the shape of the function quite well. It should also be noted however, that none of the actual values of the function are approximated. In fact, regardless of the number of hidden units, the network tends to produce values that are pulled very much towards one and away from zero. When compared to Figure 5.1, Experiment 2 shows that the standard value unit networks do not approximate this function as well as the integration device networks.

When the second term of the modified cost function for the value unit architecture was removed in Experiment 3, the function approximation abilities of the value unit networks was increased. As can be seen in Figure 5.3, all five networks were able to approximate at least a portion of the function. Specifically, the networks with zero and one hidden units could approximate the function between 0.0 and 1.0 while the network with two hidden units could approximate the function between -1.0 and 0.0. When compared to Figure 5.1, the value unit networks with five and ten hidden units were actually able to approximate the function as well as or better than the integration device networks. It is therefore concluded that if the modified cost function is removed

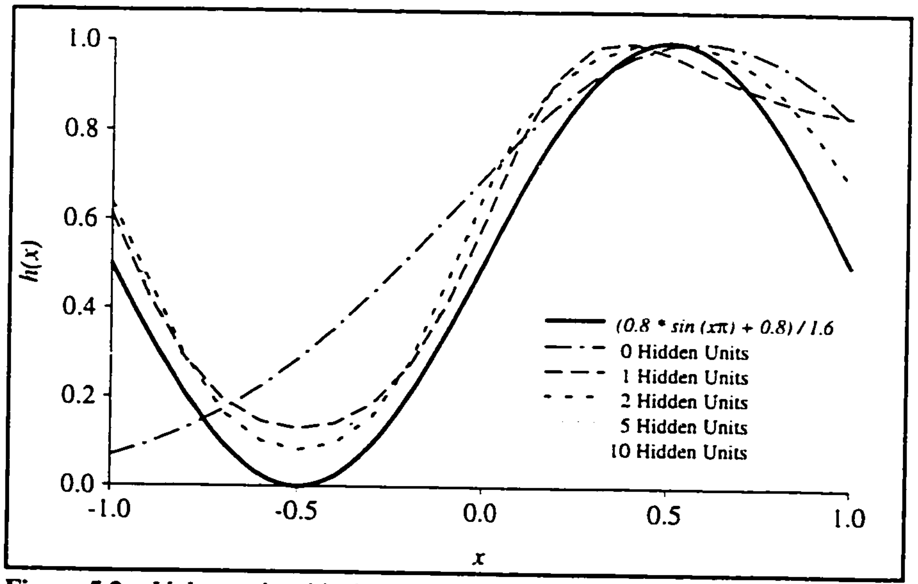

Figure 5.2 Value unit with the modified learning term function approximation with different numbers of hidden units.

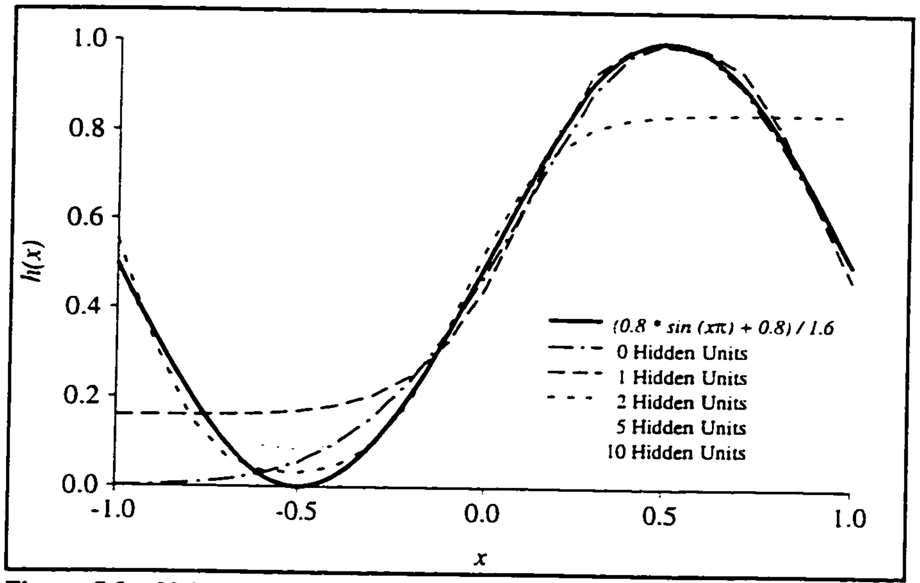

Figure 5.3 Value unit function approximation without the modified learning term

from the value unit architecture, the networks are able to perform function approximation *in practice* given enough hidden units.

Experiments 4 & 5: A Different Approach

As the output of artificial neurons is normally restricted to the range [0, 1], the original function in Equation 5.1 had to be normalized; consequently, the network was not truly computing the function. An alternative approach to solving function approximation problems is to use one input/output unit to encode the sign of the function and another input/output unit to encode the value of the function. Thus, Experiments 1 and 3 were re-run with the new network architecture and input/output mappings.

Network Architecture: Both the integration device network (Experiment 4) and the value unit network (Experiment 5) consisted of 2 input units (one for the value, one for the sign), 0, 1, 2, 5, or 10 hidden units, and 2 output units. Network parameters (i.e., weights, biases, learning rate, and momentum) were initialized as in Experiments 1 and 3 for the integration device networks and value unit networks respectively. Negative or positive inputs were indicated by a -1 or +1 on the first input unit. To indicate whether an output value was negative or positive, the first output unit was rounded either up or down; that is, if the value was less than 0.5 it was converted to -1 and if the value was 0.5 or greater, then it was converted to +1.

Network Performance: Figure 5.4 shows the function approximating abilities of the integration device network. As can be seen, networks with two or fewer hidden units could not

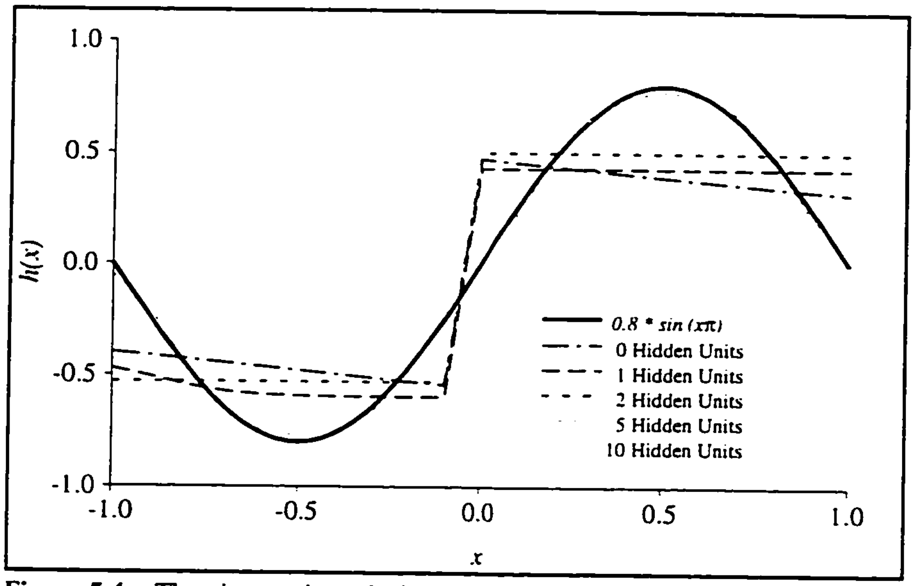

Figure 5.4 The integration device networks doing function approximation with the modified input and output.

successfully approximate the function. On the other hand, networks with five and ten hidden units were able to approximate the function quite well, with only slight disturbances near the value of 0.0 and 0.8. Again, it appears as if the ten hidden unit network did not perform any better than the five hidden unit network.

The results from the value unit network are shown in Figure 5.5. Similar to the integration device networks, the value unit networks with zero and one hidden unit could not solve the function

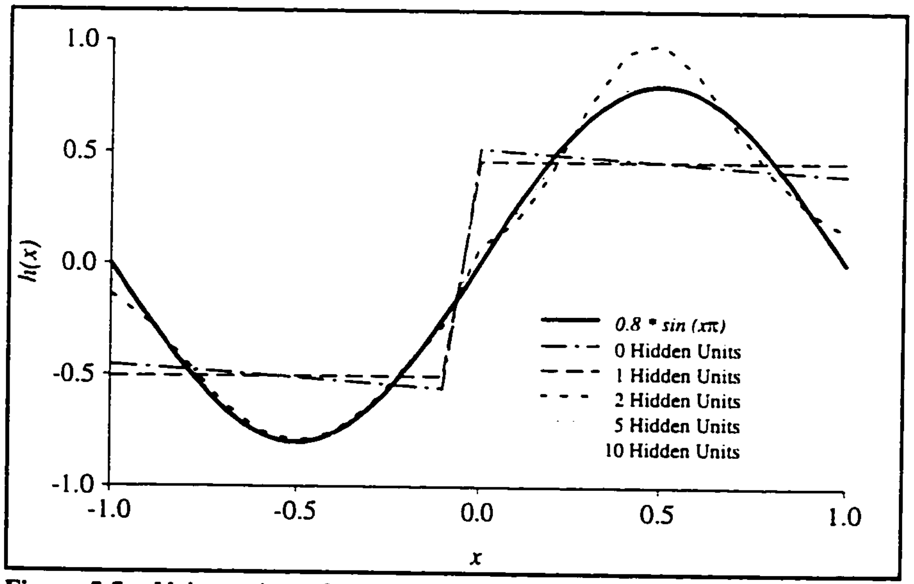

Figure 5.5 Value unit performance with the modified input and output.

approximation task. In contrast, however, the two hidden unit network was able to approximate most of the function, only producing slight disturbances at the maximum value of the function where it tended to pull the values towards 1.0. The value unit networks with five and ten hidden units were able to function approximate as well as the integration device networks in Experiment 4.

Experiment 6: Interpolation & Extrapolation

Although the results from Experiments 1 to 5 are encouraging, the ability of a network to perform function approximation is best measured in terms of its ability to interpolate and extrapolate. Interpolation is the ability of the network to produce correct responses within the range it was trained on while extrapolation is the ability to correctly predict the function beyond the original training range. From a theoretical point, proofs of function approximation are limited to a finite range; therefore, it is not expected that either type of network will be able to successfully extrapolate.

To test the interpolation and extrapolation abilities of the networks, the networks were trained on the 21 data points as before. For interpolation, the networks were tested on 20 evenly spaced points between -0.95 and 0.95. For extrapolation using the network architectures in Experiments 1 and 3, the networks were tested on 20 evenly spaced points between [-2.0, -1.1] and [1.1, 2.0]. It was noted that using the network architecture from Experiments 4 and 5 could lead to a possible confound as the negative and positive output sign values could be related directly to the sign values at the input layer. Consequently, the training of the network was extended over the range [-1.5, 1.5] and extrapolation was tested between the values [-2.0, -1.6] and [1.6, 2.0]. The network trained over the extended range was able to perform function approximation correctly.

Network Architecture: As Experiments 1 through 5 showed little difference between networks using five hidden units as opposed to ten hidden units, all networks run in Experiment 6 used five hidden units only. Furthermore, as the value unit architecture with the second term removed from the cost function performed better than the standard value unit network, interpolation and extrapolation were only tested on the modified value unit architecture. Networks were initialized as before.

Network Performance: Figure 5.6 gives the results for the interpolation and extrapolation abilities of the networks. As can be seen in Figure 5.6 (a) and (b), both the integration device network and the value unit network are able to perform interpolation on both the normalized version of Equation 5.1 and on the equation proper. A slightly different result is seen for the extrapolation experiments. Figures 5.6 (c) and (d) show that extrapolation is minimal at best for most of the

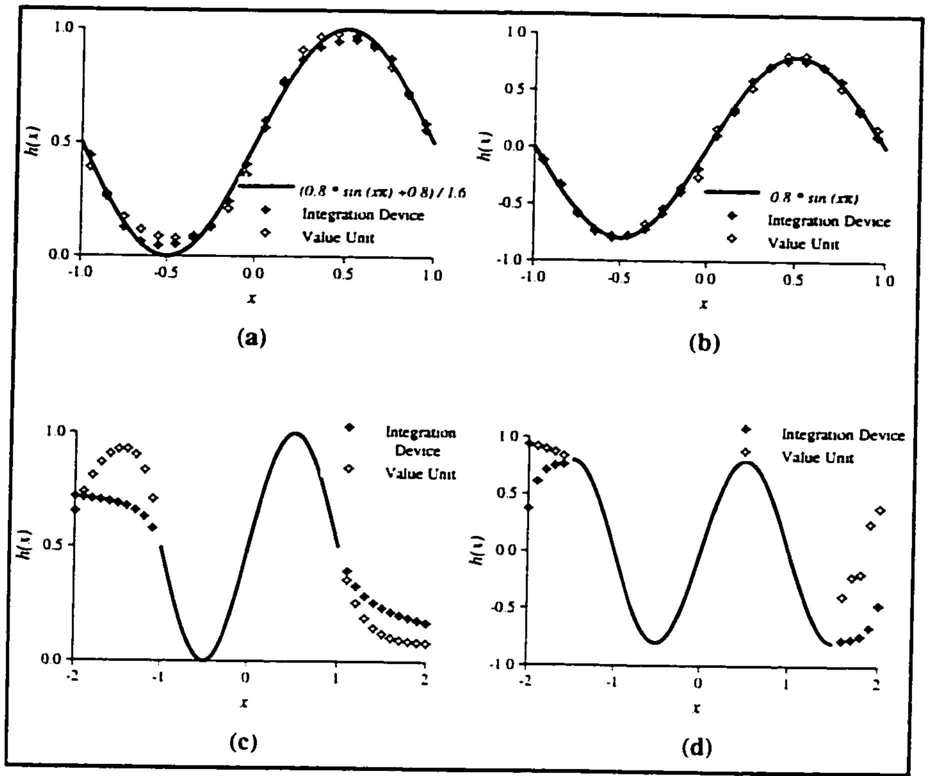

Figure 5.6 Interpolation and extrapolation results

networks. One possible exception is the integration device network trained with the added input and output unit—it follows the basic shape of the function, but the values generated by the network depart from the actual values as the range is extended.

In conclusion, the representativeness of the training data for this particular function is only valid for points within the outer bounds of the training set. Once the outer bounds are violated, the networks cease their ability to function approximate.

Function Approximation Conclusions

Both integration device networks and value unit networks (with the second term of the cost function removed) are able to perform function approximation. Although most theoretical proofs are based on an arbitrarily high number of hidden units, empirical results show that once the network is able to function approximate adding more hidden units does little to improve performance, at least for this relatively simple function. Furthermore, both network architectures are able to interpolate the function effectively, but extrapolation is extremely poor.

While connectionist models are very good at function approximation tasks (they are universal function approximators; Cybenko, 1989), there is the whole other computational problem of pattern classification. Whereas function approximation may be ideally suited for motor control, pattern classification is required for such things as understanding language (e.g., phoneme to morpheme conversion) and recognizing objects (e.g., visual recognition of letters). Consequently, we will now focus on connectionist pattern classification.

Pattern Classification

The goal of pattern classification is to map a specific pattern onto a more general pattern: In other words, given a probability density function $P(x)$, where situations x appear randomly and independently, we want to map these situations into one of k classes. We can assume that $k = 2$ without loss of generality[2] and therefore we can define a conditional probability distribution function $P(\omega \mid x)$, where $\omega = \{0, 1\}$ (x is the observed situation and ω the classification). It is possible that neither the properties of the environment $P(x)$ nor the properties of the decision rule $P(\omega \mid x)$ are known *a priori*; however, it is known that the two functions exist (Vapnik, 1982). Consequently, we wish to approximate the decision rule $P(\omega \mid x)$ by observing l pairings of random samplings of the environment $P(x)$ with their corresponding classifications $\omega = \{0, 1\}$ .Therefore, if we define a class of possible decision rules $F(x, \alpha)$, the problem of pattern classification reduces to choosing a function from $F(x, \alpha)$ such that the probability of misclassification is minimized. Hence, from Vapnik (1982), we want to find the minimum of the functional

$$I(\alpha) = P(\alpha) = \int_{X,\omega} (\omega - F(x,\alpha))^2 P(x,\omega)\,dx\,d\omega \tag{5.2}$$

where the function $P(x, \omega) = P(\omega \mid x)P(x)$ is called the joint density of the pair (x, ω) defined over the space (X, ω). Based on empirical data, Vapnik has defined the functional for the frequency of incorrect classification for each decision rule as

$$I_{emp}(\alpha) = \nu(\alpha) = \frac{1}{l}\sum_{i=1}^{l} (\omega_i - F(x_i,\alpha))^2 \tag{5.3}$$

[2] We can obtain a subdivision of k classes by making subsequent subdivisions of situations into two classes.

which is computed from the means of the sample $x_1, \omega_1;...;x_l, \omega_l$. The significance of Equations 5.2 and 5.3 is realized when classical theorems of probability theory are applied: As the number of trials increases indefinitely, the frequency of occurrences of an event converges towards the probability of that event. Formally, for any fixed α and error tolerance η, the equation

$$\lim_{l \to \infty} P\{|P(\alpha) - \nu(\alpha)| > \eta\} = 0 \qquad \textbf{(5.4)}$$

holds true (see also Vapnik & Chervonenkis, 1971). It should be noted that although Equation 5.4 guarantees that a solution to the pattern classification problem will be found, it does not imply that the minimization of 5.3 will produce the minimum functional of 5.2. Much work has been done on finding better minimizations of 5.4 and its variations (e.g. Wenocur & Dudley, 1981; Baum & Haussler, 1989; Gallant, 1993), but for the purposes of this paper it is sufficient to show that convergence is guaranteed.

Our next step is to relate the above findings to pattern classification in ANNs. Much work has been done on convergence theorem proofs for connectionist networks. Most are based on variations of Rosenblatt's (1962) *Perceptron Convergence Theorem*, which showed that convergence was guaranteed for a network in finite time as long as there were as many hidden units as there were patterns. The problem with this proof is that the hidden units come to represent one pattern only and therefore generalization is poor. More modern proofs (e.g., Lippmann, 1987; Baldi & Hornik, 1995) have shown that any pattern classification problem can be solved by a network of monotonic threshold devices with at least one layer of hidden units. The importance of the hidden units in a network is related to the computational description of a pattern classification task. The more computationally challenging a pattern classification task is, the more hidden units will be required to solve it.

One way of describing a pattern classification problem is in terms of its **order** of complexity (Minsky and Papert, 1988/1969; see Chapter 3). A problem that is linearly separable is said to have order 1 while a problem that is linearly inseparable has order greater than 1. In general, the minimum number of hidden units required to correctly solve a pattern classification task is based on the order of both the problem space and the activation function used in the processing units. For example, consider the 5-Parity problem that is illustrated in Figure 5.7. To correctly solve the problem requires five, four-dimensional hyperplanes to subdivide the pattern space; the 5-Parity problem can be described as having order 5. Hence, for a network using monotonic activation

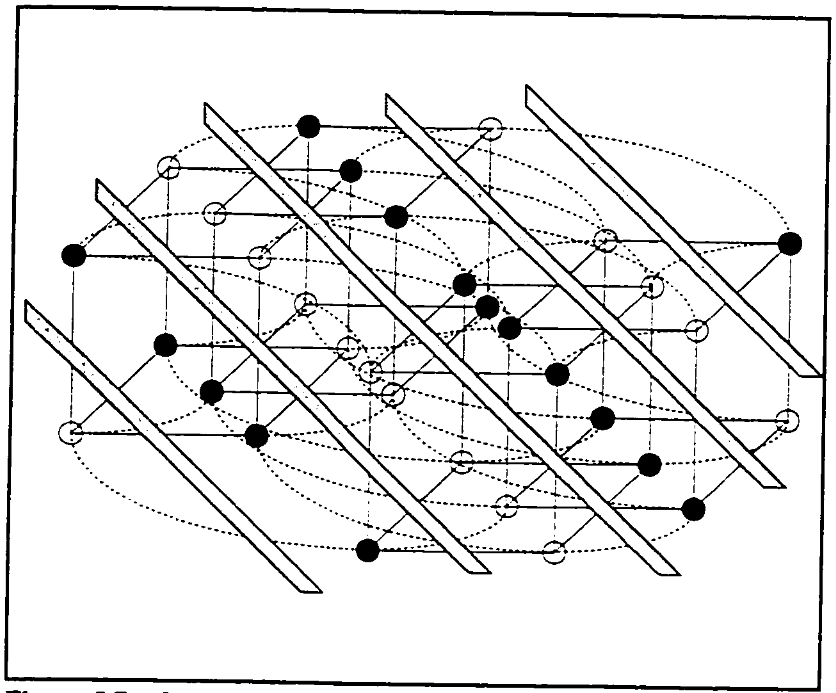

Figure 5.7 5-Parity illustrated. Each point is connected to four other points of opposite value. The 5-Parity problem is of order 5 and therefore can be solved by five parallel hyperplanes, each one being defined by one hidden unit.

functions (which have order 1, see Figure 4.3), the minimum number of hidden units required for solving the problem will be five. Conversely, networks that use a non-monotonic activation function which have order greater than 1 will require fewer hidden units. For example, a value unit network (activation function with order 2, see Figure 4.5) will require 2 hidden units, while a network using a sinusoidal activation function (unlimited order) requires no hidden units (McCaughan, 1997).

Although the analysis of the order of complexity of a problem is useful for a computational description of a problem, it does have its limitations. For example, theoretically it is possible to solve the 5-parity problem with 5 hidden units using a monotonic activation function; however, empirically it is very difficult to train a network with only five hidden units (see Table 4.1 for an example of the difficulty training a standard integration device network on the simpler 4-parity problem). Furthermore, it is often difficult to pre-define the order of complexity of a pattern classification problem. This raises an interesting question: Just because a system (whether classical or connectionist) has the *in principle* computational power to solve some form of information processing problem, how do we know that it has the practical ability to solve the problem?

Pattern Classification—Empirical Results

The problems presented in this section can be characterized as "toy" problems and are often used as benchmarks for evaluating different architectures and learning rules (e.g., Minsky & Papert, 1988/1969; Rumelhart, Hinton, & Williams, 1986a). All problems are based on binary input and output patterns. For the first three reported experiments—*Majority*, *Symmetry*, and *Parity*—multiple input units mapped onto a single output unit. The Majority problem is a linearly separable problem regardless of input size and consequently is order 1. Symmetry is a linearly inseparable problem that has constant order 2 regardless of the size of input. Parity, on the other hand, is a linearly inseparable problem whose order is equal to the size of its input. Therefore, these three experiments manipulate input size with order. The final experiment—the *Encoder* problem—has equal numbers of input and output units and maps the pattern on the input units through a reduced set of hidden units onto the output units. It should be noted that each result is based on the average of 50 different runs for each problem type and size by network architecture.

For each of the experiments, both integration device and value unit networks were tested. Because integration device networks use a monotonic function in their processing units, they are highly suited for solving linearly separable problems. In fact, integration device networks do not require any hidden units to solve such problems. The nonmonotonic activation function used by the value unit networks, on the other hand, makes them highly suited to solve linearly inseparable problems. Previous results (Shamanski & Dawson, 1994) have shown that integration device networks are better at solving the Majority problem and that value unit networks are better at solving the Parity problem. These current experiments seek to expand the previous results and to test the two different architectures on other problem types.

Experiment 7: Majority Problem

The Majority problem is one of the simplest pattern classification tasks: it is a linearly separable problem and therefore has order 1. It is based on turning the output unit on whenever there is a "majority" of input units turned on. That is, given n input units, if $(n/2 + 1)$ units are active, then turn on the output unit; otherwise, turn the output unit off.

In this experiment, eight different sizes (2, 3, 4, 5, 6, 7, 8, & 9 input units) of the Majority problem were tested. Each network was presented with all 2^n training patterns, where n is the number of input units. Networks were trained to a criterion of 0.01 which means that the network had to produce a value of 0.9 or higher when a 1 was required, and a value of 0.1 or lower when a value of 0 was required. An upper limit of 10,000 sweeps was imposed on the networks.

Network Architecture: Three different types of networks were tested in this experiment. The first architecture was an integration device network with n input units, no hidden units, and one output unit. The network weights and bias were initialized from the range [-1, 1]. The learning rate was set at 0.5 and the momentum was set to 0.9.

The second architecture was a value unit network with n input units, zero, one, or two hidden units, and one output units. Because the value unit architecture uses a nonmonotonic processing unit, it is difficult for the processing unit to make a linearly separable cut in a pattern space. Pilot studies showed that value unit network could solve the 2-Majority problem without any hidden units, but that as the problem size increased, hidden units were required. For the 3-, 4-, and 5-Majority problems, one hidden unit was used, while the 6-, 7-, 8-, and 9-Majority problems required two hidden units[3]. Network weights and biases were randomly initialized over the range [-1, 1], the learning rate was set to 0.1, and no momentum was used.

To make a fair comparison of the performance differences between the integration device networks and the value unit networks, a third architecture was also tested. This was an integration device network with n input units, zero, one, or two hidden units as above, and one output unit. All network parameters were initialized to the same values as the first architecture.

Network Performance: The results of this experiment are reported in Figure 5.8 which shows the mean number of sweeps to convergence and the standard error for each of the three architectures. All networks were able to solve the Majority problem in less than 100 sweeps through the pattern set. In fact, as the problem size increased, the integration device networks required fewer sweeps to converge on a solution. Adding the hidden units to the integration device network improved its performance for the larger Majority problems ($n \geq 5$) and in fact produced better results than the value unit networks. For the smaller size Majority problems ($n < 5$), the value unit networks were able to solve the problem in fewer sweeps, even when hidden units were used in the integration device networks.

[3] The number of hidden units was chosen to maximize performance. Shamanski and Dawson (1994) found that 6- and 7-Majority could be solved with one hidden unit; however, in these current experiments we endeavored to keep all other parameters constant which produced erratic results for 6- and 7-Majority using only one hidden unit. On a slightly different note, 3-Majority can be solved without any hidden units, but only if the hit criterion is seriously relaxed.

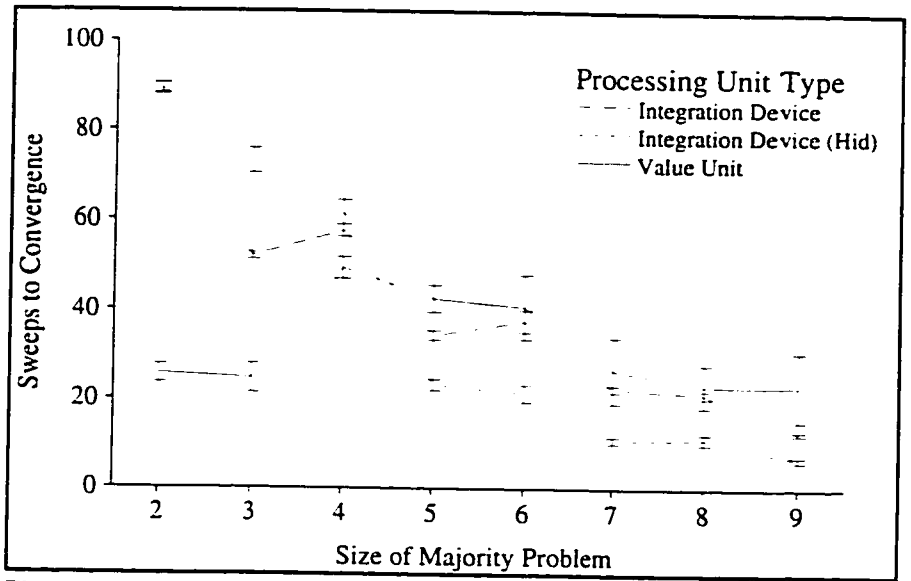

Figure 5.8 Performance of integration device and value unit networks on the Majority problem.

Experiment 8: Symmetry Problem

The Symmetry problem is a unique problem in that it is an linearly inseparable problem that has a constant order 2. Consequently, it is possible to manipulate the size of the problem without affecting the "difficulty" of the problem. Symmetry is defined by the input bits being symmetric; for example, the input patter [1 0 0 1] is a positive example of symmetry while [1 0 1 0] is a negative example (for input sizes that are odd, the middle bit is ignored).

Again, eight different sizes (2, 3, 4, 5, 6, 7, 8, & 9 input units) of the Symmetry problem were tested with all possible training patterns. For this experiment, networks were trained to a criterion of 0.0025, meaning that the network had to produce a value of 0.95 or higher when a 1 was required, and a value of 0.05 or lower when a value of 0 was required. An upper limit of 10,000 sweeps was imposed on the networks.

Network Architecture: Three different network architectures were used in this experiment. The first architecture was an integration device network with n input units, two hidden units, and one output unit. The network weights and biases were randomly initialized between the values of [-1, 1] and the learning rate was set to 0.1 with a momentum of 0.9. The second architecture was a value unit network with n input units, no hidden units, and one output unit. Because the Symmetry problem has order 2 regardless of input size and a value unit also has order 2, no hidden units are required to solve this problem. Consequently, the second architecture is the minimum network size required to solve the problem. The third architecture equalized the number of hidden units between the integration device network and the value unit network (much like in Experiment 8). Thus the third architecture was a value unit network with n input units, two hidden units, and one output unit. Both value unit architectures had their weights and biases randomly initialized from the range [-1, 1], their learning rate set at 0.01, and no momentum.

Network Performance: As can be seen in Figure 5.9—which shows the mean sweeps to convergence and standard error—the value unit architecture with two hidden units produced the best overall performance. In fact, the number of sweeps to convergence is more or less constant for all sizes of the Symmetry problem. For the value unit architecture with no hidden units, the mean number of sweeps to convergence increases with the size of the problem. On the other hand, the

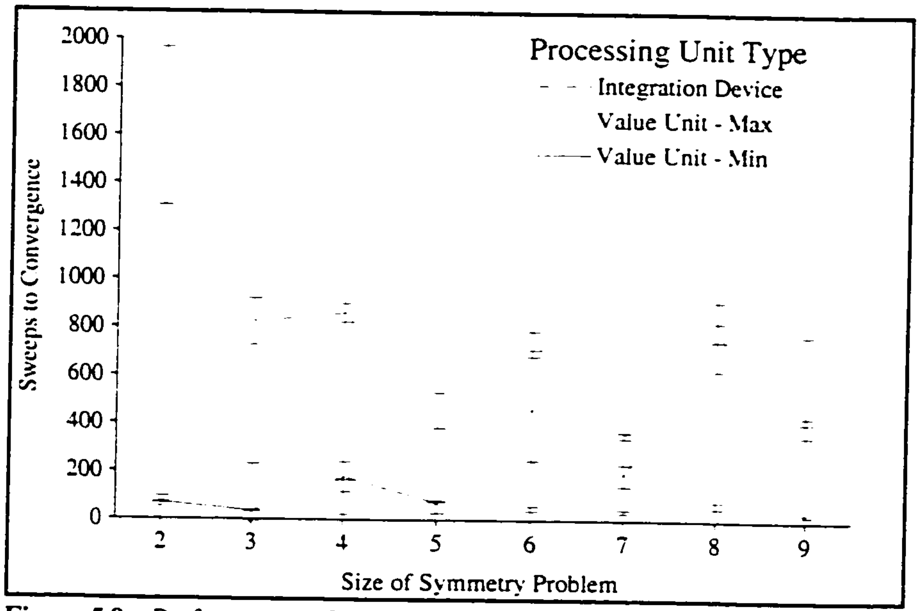

Figure 5.9 Performance of integration device and value unit networks on the symmetry problem.

integration device architecture decreases sweeps to convergence with an increase in problem size, although it never performs as well as the value unit networks with hidden units. In fact, on 8- and 9-Symmetry, the integration device network is just as good as the value unit network with no hidden units. This is not surprising, however, as a value unit network with no hidden units is basically equivalent to an integration device network with two hidden units that are constrained to cutting parallel slices in a pattern space.

Experiment 9: Parity Problem

The Parity problem is often considered the most difficult toy pattern classification task. This is because changing only one bit in the input pattern changes the network's response to the pattern.

In other words, moving from one vertex to an adjacent vertex on a binary hypercube causes a change in pattern classification. Consequently, Parity is defined by the number of active bits in the input pattern. If there is an odd number of input bits on, then the output unit should be turned on; otherwise, the output should be off.

Eight different sizes (2, 3, 4, 5, 6, 7, 8, & 9 input units) of the Parity problem were tested with all possible training patterns. For the Parity problem, there are equal numbers of positive and negative examples of classification for each size of problem. As with the Symmetry problem, networks were trained to a criterion of 0.0025 and an upper limit of 10,000 sweeps was imposed on the networks.

Network Architecture: Three different network architectures were tested on the Parity problem. The first network was an integration device network with *n* input units, *n* hidden units, and one output unit. Because the order of the Parity problem is equal to the size of the problem in bits, then theoretically one needs a one-to-one correspondence between the number of input units and the number of hidden units. Network weights and biases were randomized between [-1, 1], momentum was set to 0.9, and the learning rate was 0.1 for 2- and 3-Parity and 0.05 for 4-Parity and beyond. A different learning rate was required for the larger problems to produce an acceptable rate of convergence.

The second and third network architectures were value unit networks. The second architecture had the minimum number of hidden units required to theoretically solve the Parity problem as stated in Chapter 4. That is, for 2-, 3-, 4-, 5-, 6-, 7-, 8- and 9-Parity problems the number of hidden units required were 0, 1, 1, 2, 2, 3, 3, 4, and 4 respectively. The third architecture had equal numbers of input and hidden units. Again, both value unit architectures had their weights and biases randomly initialized from the range [-1, 1], their learning rate set at 0.01 for the smaller problems and 0.001 for the larger problems, and no momentum.

Network Performance: Figure 5.10 shows the mean number of sweeps to convergence as well as standard error for each of the network architectures over the different sizes of Parity problem. The first thing to note is that the integration device network completely failed to converge on any solution for 5-Parity and above. This result is actually quite common (see Shamanski & Dawson, 1994). The second thing to note is that both value unit architectures required more sweeps to converge as the size of the parity problem increased. For the 2- to 6-Parity problems, the value unit

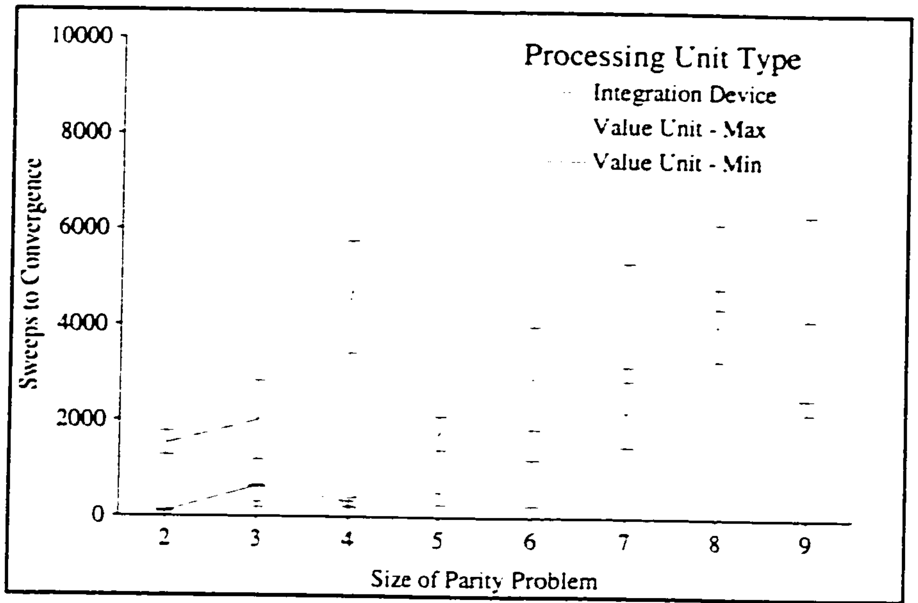

Figure 5.10 Performance of integration device and value unit networks on the parity problem.

architecture with the maximum number of hidden units performed better than the value unit architecture with the minimum number of hidden units. This reversed for the 7- to 9-Parity problems. It should be noted, however, that this reversal in Figure 5.10 is somewhat misleading as the number of networks to actually reach convergence was less for the minimum structure than the maximum structure (e.g., for 9-Parity, only 30% of the minimum structure networks converged whereas 68% of the networks with the maximum number of hidden units converged on a solution). In conclusion, the value unit architecture is clearly better for solving the Parity problem than the integration device architecture.

Experiment 10: Encoder Problem

The encoder problem consists of replicating the input pattern at the output layer by passing it through a constrained hidden layer . More specifically, a network with *n* input units has $\log_2 n$ hidden units, and *n* output units. For the Encoder problem, the number of training patterns is equal to the number of input units. Furthermore, there is the restriction that the training patterns must be orthogonal to each other. Networks were trained to a criterion of 0.0025 and an upper limit of 10,000 sweeps was imposed on the networks.

Network Architecture: Seven different sizes of networks, 2-1-2, 4-2-4, 8-3-8, 16-4-16, 32-5-32, 64-6-64, 128-7-128 were trained on the Encoder problem. Two different types of network architectures were trained: an integration device network and a value unit network. For the integration device architecture, weights and biases were randomly initialized between the values of [-1, 1] and the learning rate was set to 0.1 with a momentum of 0.9. For the value unit architecture, weights and biases were randomly initialized from the range [-1, 1], and the learning rate set at 0.01 with no momentum.

Network Performance: Network performance on the Encoder problem is illustrated in Figure 5.12 which shows mean number of sweeps to convergence and the standard error. First, it should be noted that the maximum number of sweeps was increased to 12,000 to produce a 100% rate of convergence in both architecture types. As can be seen, the value unit networks clearly outperformed the integration device network. In fact, the mean number of sweeps to converge on

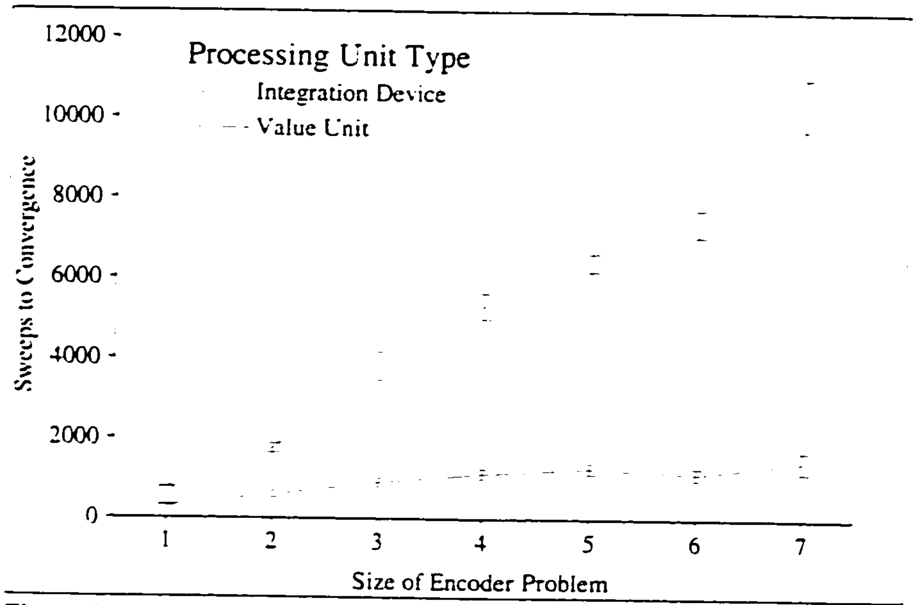

Figure 5.11 Performance of integration device and value unit networks on the Encoder problem.

a solution for the value unit architecture remains fairly constant regardless of the size of the problem. This is in contrast to the integration device network which clearly increases in the number of sweeps required to solve the larger problems. As a quick note to end this section, pilot studies have shown that the value unit architecture is able to solve the 16-2-16 encoder problem, albeit with a huge increase in the number of sweeps to reach convergence (≈ 100,000 sweeps).

Pattern Classification Conclusions

From these four experiments, it is clearly the case that the value unit architecture is superior for solving linearly inseparable problems while the integration device architecture should be preferred for problems that are linearly separable. Furthermore, the value unit architecture seems to "scale up" better than the integration device architecture. This can especially be seen in the Symmetry problem and the Encoder problem: An increase in the problem size does not increase the number of sweeps to convergence. This is an important result because one criticism often leveled against connectionist networks is that they do not scale up very well (e.g., Minsky & Papert, 1988/1969; Feldman-Stewart & Mewhort, 1994). In other words, the value unit architecture may provide a means for effectively increasing the size of computable problems.

Connectionist Conclusions

To this point, we have approached the computational level analysis from a formal perspective, as would be dictated by the traditional view on information processing. Consequently, we have described the abstract computational theory in terms of mapping one type of information to another and have endeavored to describe how the abstract properties of this mapping are defined precisely. It has been concluded that connectionism fulfills the formal requirements of computational theory, and empirical results have been presented to support computational theories. The questions remains, however, as to what connectionism adds to the computational level of analysis.

The chapter will conclude with two possible answers to this question. The first is to make the move away from formal computation towards natural computation (e.g., Horgan & Tienson, 1996; Ballard, 1997) which connectionism seems to accomplish quite easily. The second answer is to look at the explanatory power of connectionist approaches to computational analysis. In other words, connectionism may be just the tool for showing how a specific computation is both appropriate and adequate for the task at hand.

Natural Computation

Although a formal analysis of computational theory in terms of both computability and complexity is necessary for understanding the computational analysis of an information processor, it has been argued that what we should really be studying is practical computational theory (Horgan & Tienson, 1996). As stated earlier, however, the results of computability theory are based on the notion of the Turing Machine and are therefore fundamental to cognitive science. Furthermore, the study of practical computation is merely a subset of formal computation, which means that all results from our study of formal computation will apply to our study of practical computation.

The formal study of complexity, however, is dependent on many assumptions including the size of the problem, the criterion used to evaluate a solution, and machine architecture. Complexity can also be framed in terms of natural computation (Ballard, 1997). The natural computation approach views complexity from a slightly different perspective, producing models that are effective while also providing insights into the possible mechanisms of brain computation. The two main elements governing this perspective are Minimum Description Length (MDL) analysis and learning. The goal, then, of understanding complexity in terms of natural computation would be to find an underlying principle that addressed each of these elements. And, connectionism fits that bill nicely. To illustrate this, these two elements will be elaborated below.

Minimum Description Length Analysis

Consider a system that is defined by the sets of input/output data it produces. If the sets are random, then there is no smaller description of the data than the data itself. Practically, however, many physical systems—including the brain—incorporate many regularities and can produce much more economical descriptions of the data sets. One way of describing such data sets is to use Minimum Description Length (MDL) analysis. MDL can be seen as a principled version of Occam's razor[4] where the goal is to find a simple yet accurate description of the data.

In MDL, when accounting for the size of the encoded data, you have to include the size of the encoder as well. In the case of a biological system, the encoder is simply the behavioral program. Consequently, the succinctness of a specific computational theory can be measured in terms of the data it accounts for and the complexity of the theory/encoder. Of course, the ideal result from such an analysis would not only account for the present data, but it would also generalize to new data.

Recent research (e.g., Zemel, 1995) has shown that MDL analysis can be successfully applied to neural networks. In such an analysis, the network is a generative model of the output given the input where the model includes the network architecture and weights and the data are the residuals. Consequently, by applying the MDL technique to neural networks, the optimal network structure for generalization can be found, and such structures can then be compared to actual neural structures such as the receptive fields of neurons (Ballard, 1997). Hence, MDL demonstrates the appropriateness and adequacy of a computational theory for a specific task.

[4] "*Entia non sunt mutiplicandu, prœter necessitatem*" or "Entities should not be multiplied unnecessarily" is credited to the medieval philosopher William of Ockham (1285-1347), although there is some question as to whether he used that exact phrase (see Thorburn, 1915; Burns, 1915; Thorburn, 1918). Within science today, Occam's razor is often stated as "When you have two competing theories which make exactly the same predictions, choose the one that is simpler" or—in the spirit of Occam—"Make it simple".

Learning

The interesting contribution that connectionism has to offer to computational analysis of an information processing system is the fact that connectionist systems can learn. What this means is that although the input and output parameters must be specified for the model *a priori* (just as they must be for a classical approach) the actual function that computes the input/output mapping does not need to be specified *a priori* (unlike a classical system). As an example of the different approaches to a computational description of function approximation, we will consider the problem of sensori-motor maps; specifically, reaching towards a fixated object. The classical approach is typified by researchers in artificial intelligence (e.g., Horn, 1979). Traditional robotic limb manipulation is achieved through a series of programmed end effector movements based on either forward or inverse kinematics. Using this approach, Churchland (1989) has designed a crablike robot that can successfully reach towards an object that has been placed in front of it. Churchland has described the task in terms of trigonometric functions expressly calculated for each of the rotatable eyes and the shoulder and elbow joints. Using a method of projection from the sensory to the motor map, Churchland is able to show correct performance of the crab to reaching in two-dimensional space.

Although this traditional method of robotic limb manipulation effectively mimics sensori-motor behaviour, there is no indication that the nervous system carries forth such complex trigonometric functions in the step-by-step fashion that is required. In fact, analyses of biological systems indicate that the planning and execution of limb and eye-movements may be based in discrete modules distributed in a number of cortical and subcortical regions (e.g., Bizzi & Mussa-Ivaldi, 1995; Georgopoulos, 1995; Jordan, 1995).The connectionist approach to the computational problem of visually-guided reaching is truly based on function approximation. Whereas the traditionai approach is based on very rigid pre-defined trigonometric functions, connectionist networks approximate the desired joint angles given angles subtended by the eyes (e.g., Medler & Dawson, 1994a; Medler & Dawson, 1994b). Therefore, the computational description is limited to learning the proper input/output mappings required for the task. Employing connectionist models to approximate the inverse kinematics function circumvents the computational complexity of the numerical solution while providing a learning mechanism for adaptation to environmental changes such as obstacles, loads, and friction (Eckmiller, 1989; Kuperstein, 1988).

Computational Explanation

To conclude our computational analysis of connectionism, we will engage in a little "fictional science" courtesy of Braitenberg (1984). Imagine watching and recording the behaviour of a small "vehicle" swimming around in a pond; over the long run, its movement seems to follow a complicated trajectory as suggested in Figure 5.12. From an analytical approach (see Chapter 2, p. 20), the movement of the vehicle could be described at the computational level as something approaching "Brownian motion" with some sort of drive added. Following this analysis, we could predict with some degree of certainty the future movement of the vehicle by analyzing the mathematical components of the vehicle's previous movements—say by using Fourier transforms—and creating a mathematical model of the vehicle's path. This could potentially suffice as a computational *description* of the vehicle's behaviour, both accounting for previous data and predicting new data. Yet this type of characterization only describes the behaviour, it does not explain the behaviour, and, cognitive science requires explanations, not descriptions.

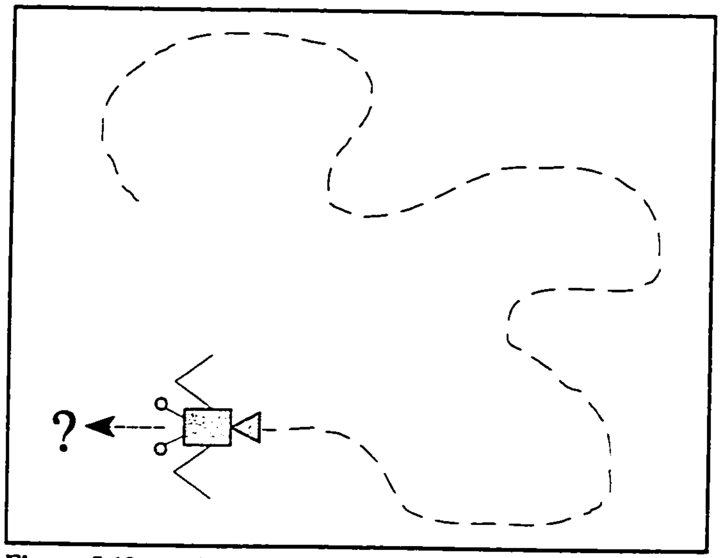

Figure 5.12 Analyzing the computational level description of the movement of a simple "vehicle" through water.

How might we construct a computational *explanation* then? Braitenberg (1984) adopts the view that we should look for the simplest explanation for behaviour—very much in accordance with Occam's razor—and that this often lies within the simple mechanisms of the organism itself. In other words, Braitenberg proposes that we adopt a *synthetic* approach to psychology; Identify the simplest mechanisms governing the behaviour of an organism and observe what happens when they interact with each other and the environment. Thus, we are not only describing the computational properties of an organism's behaviour, but also *explaining* those same computational properties.

Consequently, to produce a computational explanation of our vehicle swimming around in the pond, we might postulate some very simple mechanisms. One mechanism might be a simple temperature sensor that drives a motor in proportion to the amount of heat detected. In other words, the vehicle would tend to speed up in warm water and slow down in cold water. Hence, the proper computational explanation for the vehicle's movement in this case is very simple: stay in cold water and avoid warm water. The "erratic" behaviour of the vehicle only emerges because of outside forces, such as friction and currents, acting upon the vehicle. These outside forces have no role in the computational function being computed by the vehicle, although they are necessary for understanding the behaviour of the vehicle at that specific time. Of course, we would have to confirm this explanation by testing the vehicle in other environmental situations, controlling as many outside forces as possible.

A similar idea is espoused by Seidenberg (1993) in his Chomskian-inspired evaluation of different approaches to cognitive theorizing. One approach to cognitive theorizing is based on examining a large range of data and attempting to produce generalizations about the patterns of data; this approach produces theories with "descriptive adequacy." Although descriptive theories can describe the phenomena that do occur and generate novel predictions, Seidenberg argues that they cannot explain why other and equally plausible phenomena do *not* occur. Furthermore, he continues, it is not enough for us simply to describe the kinds of things that are in the world, but we also need to understand *why* things are the way they are and not any other way. This type of cognitive theorizing is equivalent to the analytical approach to analyzing an information processor.

Another—more fruitful—approach to cognitive theorizing is to show how the phenomena in question derive from deeper principles; such an approach produces explanatory theories. Whereas theories with descriptive adequacy are based on task- or phenomenon-specific principles, a condition on explanatory theories is that they appeal to a small set of concepts that are independently motivated. Furthermore, if these underlying principles are explanatory, then they will also contribute to the understanding of phenomena in different domains. Explanatory theorizing is equivalent to the synthetic approach to cognition. As a result, it allows us to understand *why* things are a certain way and not another. In other words, whereas descriptive theories only describe possible computational functions, explanatory theories define the computational competence of an information processor. Seidenberg (1993) states that connectionism is just the tool to contribute to the development of theories that are explanatory and not merely descriptive.

Seidenberg's view is not without controversy. For example, Massaro (1988, 1990) contends that certain assumptions within connectionist models are unnecessary and inconsistent and that the models themselves are too powerful to be of any theoretical importance to cognitive science. Seidenberg argues, however, that connectionist models acquire their explanatory power when constraints are applied in systematic ways. One important constraint is their appeal to a small yet general set of computational principles (see Chapter 2, pp. 33-36). The second constraint lies in the further requirement that they be neurobiologically relevant. When these constraints are met, Seidenberg argues that connectionism contributes to the development of explanatory theories by providing a candidate set of independently motivated theoretical principles (cf., McCloskey, 1991).

Thus, connectionism can be viewed as a way of not only modeling cognitive functions, but also offering new explanatory cognitive theories. Although connectionism can and often produces computational theories that are appropriate and adequate for particular tasks in specific domains, their advantage over classical theorizing is that connectionism's underlying principles remain the same regardless of domain.

> . . . one starts with a set of principles concerning learning and the representation of knowledge. If the principles are identified correctly, modeling should merely involve incorporating domain-specific variables such as different types os stimulus inputs, motoric responses, and learning experiences . . . The relevant generalizations about the domain in question should then fall out of the model. (Seidenberg, 1993, p. 231)

Computational Level Conclusions

Connectionist networks have all the computational power of classical models—they are both universal function approximators and arbitrary pattern classifiers. Therefore, they are able to answer the types of computational questions that are of interest to cognitive scientists. In other words, connectionism has all the properties that a traditional approach to computational analysis possesses.

What, then, do we have to gain from a connectionist approach? One gain is moving from a formal notion of computation to a more natural mode of computation. Another is that connectionism may offer computational explanations as opposed to merely descriptions. Further answers to this question will become apparent when we continue the analysis of the other levels within the tri-level hypothesis.

Chapter 6

The Algorithmic Analysis of Connectionism

The deepest rift between connectionist and classical researchers exists at the algorithmic level of analysis. On the one hand, researchers such as Fodor and Pylysyhn (1988) maintain that connectionist models should be considered as mere implementations of classical cognitive architectures. In other words, although connectionist models may be interesting in their own right and may be of value to such areas as neuroscience, they have nothing new to contribute to cognitive psychology. On the other hand, there are those researchers who state that connectionist models provide a completely novel approach to understanding cognition, and therefore could very well signal the start of a Kuhnian-like paradigm shift in the study of cognition (e.g., Schneider, 1987).

This rift is typified by the exchange between Broadbent (1985) and Rumelhart and McClelland (1985) in the *Journal of Experimental Psychology: General.* Immediately on the heels of McClelland and Rumelhart's (1985) connectionist account of distributed memory and representation of knowledge, Broadbent took issue with their approach, claiming that considering its implications at anything other than a physiological level was inappropriate.

> [McClelland and Rumelhart, 1985] believe that their approach has implications at the psychological and not merely at the physiological level. . . . These claims are not appropriate and might in some circumstances damage the acceptance of the distributed theory at its proper level. (Broadbent, 1985, p. 189)

Rumelhart and McClelland (1985) respond to Broadbent with their own views concerning the proper level of analysis for connectionist networks. Their main point was that their model is fully relevant to psychology as it is stated at the algorithmic level. Furthermore, they contended that no particular level of analysis is independent of the others and thus their model appeals to an implicit computational theory as well as certain implementational considerations. (Interestingly, Broadbent ignores the algorithmic level and bases his criticisms on the distinction between the computational and implementational levels.)

> We believe that our proposal is stated primarily at the algorithmic level and is primarily aimed at specifying the representation of information and the processes or procedures involved in storing and retrieving information. (Rumelhart and McClelland, 1985, p. 193)

Hence, Rumelhart and McClelland claim that connectionist models and classical models are competitors at the same level of description.

Why are connectionist and classical accounts of psychological processing presumed to be competitors? The main reason is because the two approaches have different views about how information is processed. The classical approach sees information processing as the manipulation of symbols using logical rules (which may be either explicit or implicit). This leads classical researchers to conclude that "Only the algorithm and the representations on which it operates are intended as a psychological hypothesis" (Fodor & Pylyshyn, 1988, p. 65). In contrast, the connectionist approach sees information processing as the manipulation of sub-symbolic elements by statistical means. This leads many classical researchers argue that the level of cognition is not connectionist in nature and therefore that connectionist models have no role in informing cognitive science at the algorithmic level. For example, Fodor and Pylyshyn (1988) conclude that cognitive learning is not based on statistical inferences realized by adjusting parameters (i.e., the connectionist approach), but is based on theory construction effected by framing hypotheses and evaluating them against empirical evidence (i.e., the classical approach).

As illustrated by the comments of Rumelhart and McClelland (1985), however, some connectionists firmly believe that the algorithmic level of analysis is applicable to their models and, therefore, that they are relevant to psychological theory. Thus, it is claimed that connectionist models offer an alternative approach to cognitive processing. This is because it is assumed by both classical and connectionist researchers that the underlying principles of connectionist networks (i.e., the statistical manipulation of sub-symbols) necessarily means that they offer an alternative account of cognitive processing.

> There is no doubt that these models have a different feel than standard symbol-processing models. The units, the topology and weights of the connections among them, the functions by which activation levels are transformed in units and connections, and the learning (i.e., weight-adjustment) function are all that are "in" these models; one cannot easily point to rules, algorithms, expressions, and the like insides them. (Pinker & Prince, 1988, p. 76)

Hence, much of the debate between connectionist and classical researchers is centred on this misconception about the assumed nature of connectionist networks. It seems that neither side is willing to substantiate their claims, but rather take the "correctness" of their assumptions on faith. This reluctance is characterized by Adams (1979, p. 50) "I refuse to prove that I exist . . . for proof denies faith, and without faith I am nothing." In other words, they want the connectionist approach to cognitive processing to be radically different—if, for no other reason, than to say that the other side is wrong.

One simple way of substantiating their claims, however, would be to analyse the internal structure of trained networks. This analysis could reveal how the networks are solving problems and may show that the connectionist and classical approaches are not so far removed from each other. "Analysing what the net computes after learning provides a simple empirical way of exploring the intimate relationship between learning and representation" (Hanson & Burr, p. 482).

But, the analysis of connectionist networks can be rather intimidating—especially for large networks. And, if this move is not undertaken, then connectionism is reduced to no more than "black-box" theorizing. Therefore, when connectionist models are offered as algorithmic theories of cognition *sans* analysis, classical researchers are justifiably skeptical. This skepticism is clearly illustrated by Pylyshyn's frank assessment of connectionism:

> 'Voodoo,' remarks Zenon Pylyshyn . . . 'People are fascinated by the prospect of getting intelligence by mysterious Frankenstein-like means—by voodoo! And there have been few attempts to do this as successful as neural nets!' (Stix, 1994, p.44)

Even connectionist researchers freely admit that it is extremely difficult to determine how connectionist networks accomplish the task that they have been taught. "One thing that Connectionist networks have in common with brains is that if you open them up and peer inside, all you can see is a big pile of goo" (Mozer & Smolensky, 1989, p.3). Similarly, Seidenberg (1993, p.229) states that "if the purpose of simulation modeling is to clarify existing theoretical constructs, Connectionism looks like exactly the wrong way to go. Connectionist models do not clarify theoretical ideas, they obscure them." As a result, connectionists are reluctant to analyse the internal structure of their networks in an attempt to determine what algorithm lies within.

Unfortunately, this reluctance has raised serious doubts concerning the ability of connectionists to provide fruitful theories about cognitive processing. Because researchers rarely understand the internal workings of PDP models, McCloskey (1991) suggested that "Connectionist networks should not be viewed as theories of human cognitive functions, or as simulations of theories, or even as demonstrations of specific theoretical points" (p.387).

Thus, if we are to address the concerns of McCloskey (1991) and Fodor and Pylyshyn (1988), connectionists must take the time and effort to analyse the internal structure of their networks. The importance of analysing the internal structure of connectionist networks is highlighted by Hanson and Burr (1990):

> Post hoc analyses of the way a network computes and represents information over subsets of hidden units . . . can clarify imprecise or incomplete models of psychological phenomena and can help reveal important relations between learning and representation that are not taken into account in the rule-based approach. (p. 476)

This importance has not been overlooked by connectionists. Several different techniques for interpreting the internal structure of connectionist networks have been proposed. These analyses are roughly divided into two categories: (i) analysis of network weights, and (ii) analysis of hidden unit activities.

Two main approaches to the analysis of network weights exist. This first approach—known as *Hinton diagrams*—is perhaps the simplest way of examining the internal functioning of networks. Following learning, the weight values from the input to the hidden units, and from the hidden units to the output units are encoded in a diagram as a series of white (positive) and black (negative) rectangles. The area of each rectangle signifies the relative strength of that connection normalized to the largest weight (e.g., Hinton, McClelland, & Rumelhart, 1986; Hinton, 1987). Such an analysis indicates the relative importance of each input unit to any hidden unit and thus the function of the hidden unit can be interpreted in terms of the relevant input units. Furthermore, the relative importance of each hidden unit to any output unit can also be stated. Hinton diagrams do have drawbacks; they are limited to relatively small networks where the input and output representations are local.

The second approach to analysing weights is to examine the frequency distribution of the weights after training. The rationale behind this technique is that such an analysis will reveal the distributed or local nature of the hidden units. It is assumed that distributions with a large, symmetric, unimodal concentration of weights near zero represent units that have adopted a distributed representation while distributions that are skewed or bimodal are more likely to represent local units (Hanson & Burr, 1990). Unfortunately, such a technique does not tell us what the unit is actually computing—it only tells us if a unit is likely to have a local or distributed interpretation, without any appeal to what that interpretation might be. To accomplish this last yet crucial step, Hanson and Burr perform a cluster analysis on the connection weights from the input to the hidden units and then isolate those units with the strongest connections. A star diagram is then created which identifies patterns of connectivity visually. These diagrams convey the same information as Hinton diagrams and are therefore subject to the same drawbacks.

An alternative approach to analysing network structure is to perform cluster analysis on the hidden unit activations. Again, two main methods exist for performing this type of analysis on trained networks: *in vivo* clustering and *in vitro* clustering. In vivo clustering involves *re*-presenting the training set to the network and recording the activities of the hidden units to each pattern. Hierarchical clustering is then performed on the stimulus-by-hidden-unit-activity matrix to recover functional groups of units that the network has organized during learning (e.g., Elman, 1990). In vitro clustering involves transforming the weights into activation space via the original activation function—this essentially computes the maximal activation of each hidden unit give all inputs. The distance between the transformed weights is then measured (say, by Hamming distance) and clustering analyses can be applied to determine either what input or output features are close together in hidden unit activation space or what hidden unit groups exist in input/output activation space. Both in vivo and in vitro analyses only indicate some of the underlying structure of the network, however, and further analyses must be performed before any meaningful interpretation can be extracted from the network.

Although the interpretation techniques described above have been successfully applied to a variety of different networks, the clustering analyses these techniques are dependent on means that the role of each individual hidden unit is not always clear. Furthermore, such analyses do not address the issue of how the network is explicitly solving the problem. In other words, the algorithm is still obscured. Thus, we cannot address the issue of whether or not classical and connectionist accounts of cognitive processing are radically different. To do this, we need to know what each individual unit is doing, and how the network uses that unit to solve the problem.

Consequently, in this chapter we will review and expand a network interpretation technique first described by Berkeley, Dawson, Medler, Schopflocher, and Hornsby (1995). This technique allows interpretation of individual hidden units while also providing a mechanism for analysing the pattern of activities across the hidden units. Thus, we will be able to explicitly address the assumed distinction between classical and connectionist models of cognitive processing.

The Banding of Hidden Unit Activation Values

Consider using a set of patterns to train a network of value units. After training, one could again present each pattern to the network and record the activity that each pattern produced in each hidden unit (cf., *in vivo* analysis; Hanson & Burr, 1990). This amounts to "wiretapping" each hidden unit while the stimulus set is being presented. The recorded activations could then be used to create a *jittered density plot*[1] for each hidden unit (e.g., Figure 6.1). A jittered density plot is basically a one dimensional scatter plot in which the x-axis represents the unit's activation between 0.0 and 1.0, and the y-axis represents a random jittering introduced to prevent points from overlapping (Chambers, Cleveland, Kleiner, & Tukey, 1983, pp. 19-21). Consequently, such a plot not only presents the activation values of the hidden units, but also presents the relative distribution of values. If the activation values are distributed across the full range of values between 0.0 and 1.0, then the jittered density plots will appear "smeared". If, on the other hand, activation values fall into distinct clusters, these will appear as vertical "bands" within the plots.

The jittered density plots in Figure 6.1 illustrate the results of wiretapping three different networks trained on (a) 3-Parity, (b) 4-Parity, and (c) 5-Parity (see Chapter 5). In each of the density plots, a single dot is used to represent the activity in that hidden unit produced by presenting one pattern to the network. The density plots in Figure 6.1 were produced in networks trained on 8, 16, and 32 different input patterns for 3-, 4-, and 5-Parity problems; consequently, each density plot is composed of 8, 16, and 32 different dots respectively.

[1] A histogram would be inappropriate for this type of analysis for two reasons. First, the real-valued activations of the units would have to be quantized and important information could be lost that way. Second, for very large data sets, a few patterns (say, eight) that comprise a distinct group may be "graphically lost" when plotted next to a larger group of patterns (say, eight thousand).

As can be seen in Figure 6.1, the density plots for hidden value units are highly structured (see also Berkeley, et al., 1995). That is, the activation values are revealed as distinct "bands" within each plot. Furthermore, analyses of patterns within each band show that they are related in some manner. For example, in Figure 6.1(a), the one input pattern falling into Band-A has two input bits on, the four patterns falling into Band-B all have an odd number of bits on (i.e., either one or three bits on) and the three patterns falling into Band-C have either zero or two bits on. Exactly the same interpretation can be given to Figure 6.1(b).

In fact, this same banding structure is always seen in 3- and 4-Parity networks trained with one hidden unit. This is because there is only one optimal solution for each problem. This solution requires that a hyperplane be positioned within the input space such that it bisects the points in the hypercube defined by the input. This requires that the weights from the input units to the hidden unit be ±0.5. Furthermore, the weights must be balanced; that is, for every positive weight, there is a negative weight (for odd Parity, the extra weight could be positive or negative). Thus, the same

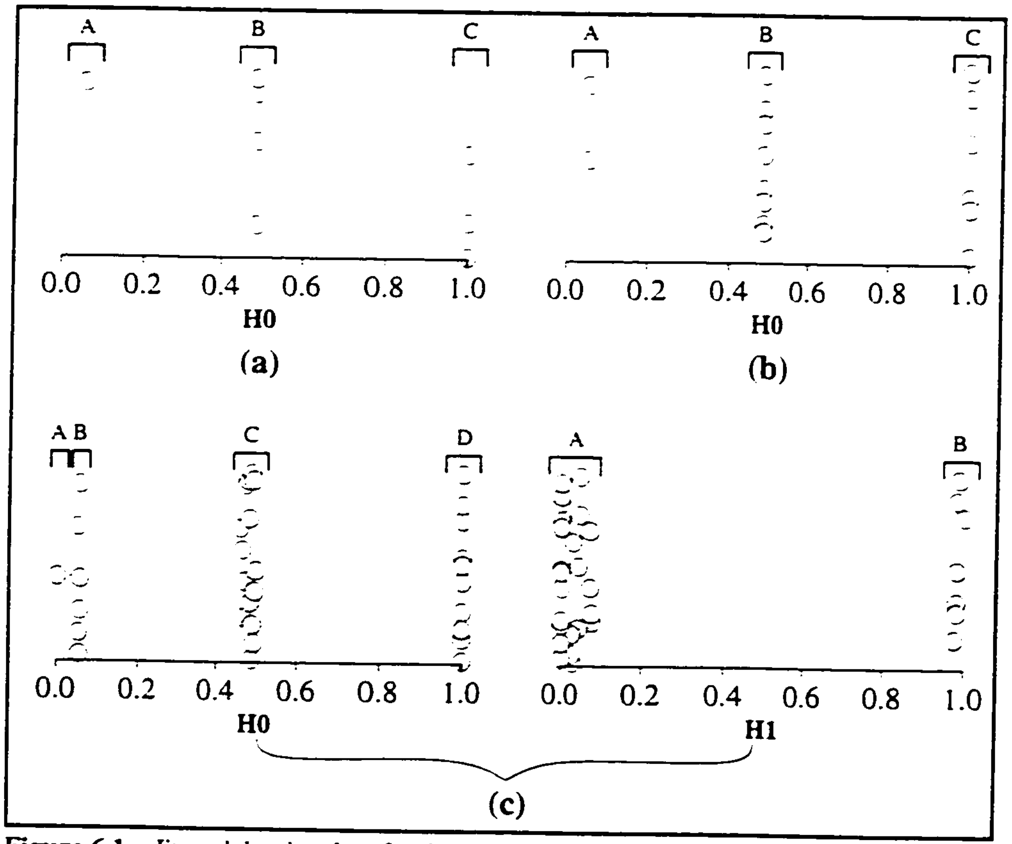

Figure 6.1 Jittered density plots for the (a) 3-Parity, (b) 4-Parity, and (c) 5-Parity problems. Note that (a) and (b) only require one hidden unit while (c) requires two hidden units. See text for analysis of the bands.

banding pattern will always be produced. It should be noted, however, that because any weight may be positive or negative, the patterns falling into Band-A and Band-C may be different, although the patterns in Band-A will always have two bits on. Moreover, Band-B will always contain those input patterns with an odd number of bits on. Consequently, although the banding structure will remain constant between networks, the exact interpretation will vary.

The hidden unit activations for the 5-Parity problem—shown in Figure 6.1(c)—have a slightly different interpretation. Band-A of Hidden Unit 0 (hereafter 0-A) has one pattern with three input bits on, while 0-C contains all other patterns with either one, three, or five bits on. The other two bands contain only those patterns with an even number of bits on; 0-B contains patterns with two or four bits on, and 0-D contains patterns with zero, two, or four bits on. At first glance, the banding structure for the second hidden unit appears less structured. For example, 1-A contains a mixture of patterns; some have an even number of bits and some have an odd number of bits turn on. Even those patterns falling into 1-B have either zero, two, three, or four bits turned on. It turns out, however, that the one odd pattern falling into 1-B is the one pattern in 0-A.

Consequently, the structure of Hidden Unit 1 becomes apparent when it is paired with Hidden Unit 0 to produce the correct outputs. Of the eight possible hidden unit activation combinations, only five are actually ever produced. The combinations [0-A, 1-B] (read "activation produced in Band-A of Hidden Unit 0 and Band-B of Hidden Unit 1 concurrently") and [0-C, 1-A] produce an output of 1.0 (odd parity). Even parity, on the other hand, is signified by the combinations [0-B, 1-A], [0-D, 1-A], and [0-D, 1-B]. Thus low to medium activation in Hidden Unit 0 paired with high activation in Hidden Unit 1 signals odd parity while low activation paired with low activation or high activation paired with either low or high activation denotes even parity.

It should be noted that Berkeley et al., 1995, did not find that this banding structure was typical of density plots for standard backpropagation networks using the logistic activation function (a monotonic function). On the other hand, McCaughan (1997) did find that networks using hidden units with a sinusoidal activation function (again, a nonmonotonic function) produced interpretable bands.

Consequently, it appears that the "banding phenomenon" may be dependent upon the nonmonotonicity of the activation function. This conclusion is based on two properties of nonmonotonic functions. First, a value of 1.0 produced by a nonmonotonic function (say, a value unit) carries much more information than a value of approximately 1.0 produced by monotonic function (say, an integration device). This is because the number of input patterns that are capable of producing a value of 1.0 in a hidden value unit is limited to those input patterns that, when multiplied by the weight vector and subtracted from the bias of the unit, produce a value equal to the mean of the Gaussian. That is,

$$[i_1 .. i_n]\begin{bmatrix} w_1 \\ \vdots \\ w_n \end{bmatrix} - \theta = \mu$$

where $[i_1 .. i_n]$ is the input pattern, $[w_1 .. w_n]$ is the weight vector, θ is the bias of the unit, and μ is the mean of the Gaussian. If the bias is equal to mean of the Gaussian (as is the case for value unit networks), then all input patterns that produce a value of 1.0 will lie in a plane orthogonal to the weight vector. Consequently, there are only a limited number of input patterns that are capable of producing a value of 1.0 in a hidden value unit. Furthermore, all of these patterns are related in the sense that they lie within the same hyperplane with respect to the weight vector and input space.

As an example, Figure 6.2a shows a value unit superimposed over the input space for the 3-Parity problem. There are three co-planar points that will produce the maximal activity within the value unit. Furthermore, as can be seen, these three points lie equidistant from the two hyperplanes that are carved by the value unit (as represented by the dashed lines). If the unit is thresholded by

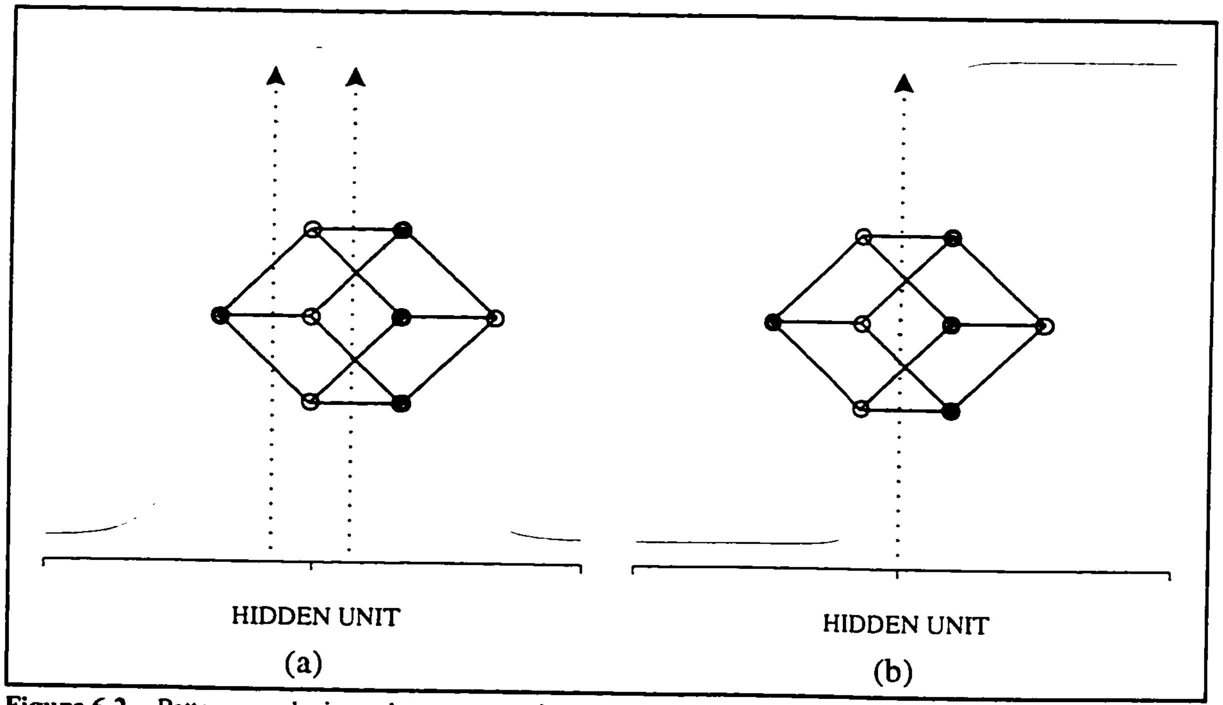

Figure 6.2 Patterns producing values near one in (a) a value unit, and (b) an integration device.

forcing values to either zero or one, then a hyperbar (or closed halfspace) is defined by these two planes. This means that there is only a relatively limited number of patterns that can produce a value of one. As the goal of information theory is to use as few bits a possible to convey as much information as possible (e.g., Roman, 1992), a value unit maximizes the amount of information by picking out a very distinct set of input patterns (i.e., those patterns that are co-planar).

In contrast, Figure 6.2b illustrates the partitioning of the 3-Parity problem by a single integration device unit. Although there is one pattern that will produce maximal activity of approximately 1.0 within the unit[2], there are three other points that will also produce values near one. As can be seen in the figure, no planar relationship exists between the four points that produce values close to one. In fact, if the unit is thresholded, then an open halfspace is defined (by the dashed line) which means there are an infinite number of input patterns that have no planar relationships that could potentially produce a value of one. Consequently, a value of one in an integration device unit does not convey as much information as a one in a value unit.

This finding actually has implications for the neuron doctrine within perceptual psychology (e.g., Barlow, 1972). The doctrine is based on the assumption that there is one stimulus pattern that will produce maximal activity within the neuron. Although it is often difficult to apply the neuron doctrine to biological systems, it can be readily applied to integration device networks. For example, Dawson, Kremer, and Gannon (1994) applied the neuron doctrine to define the *trigger feature* of the hidden unit as that input pattern which produces maximal activity in the unit. The trigger feature is determined by setting the input bit to 1.0 for all positive valued weights leading into the hidden unit and 0.0 for all negative valued weights (this produces the maximum net input possible for a specific hidden unit). Thus, the sensitivity of the hidden unit can be determined (c.f., in vitro analysis; Hanson & Burr, 1990).

Practically, however, many more input patterns will produce a value of near one than just the trigger feature. In fact, the number of input patterns producing a value near one in a hidden integration device unit will be dependent on the steepness of the logistic function; for example, thresholding the unit by using the step function—a logistic function with maximum steepness—will produce values of either zero or one. Therefore, it is possible for numerous (unrelated) patterns to produce maximal activity (within a certain threshold) within the hidden unit. Consequently, the range

[2] In theory, an activation function such as the logisitic only produces a value of 1.0 as the net input $\rightarrow \infty$ and a value of 0.0 when the net input $\rightarrow -\infty$.

of inputs the unit (or neuron) is actually sensitive to may be quite extensive and the assumption of the neuron doctrine may be inappropriate and misleading.

When these results are applied to the planar description of banding analysis, this means that patterns not lying in a plane orthogonal to the weight vector may still produce a value near one. Consequently, there is not the same relationship between input patterns producing maximal responding for logistic functions as there is for Gaussian functions.

A second reason that banding appears dependent on nonmonotonic functions relies on the fact that the weight vector defines an orthogonal hyperplane within the input space. This means that other parallel hyperplanes are automatically defined within the input space. Consequently, all points lying in a parallel hyperplane some distance δ from the original hyperplane will produce some activation $0 \leq \lambda < 1$ in the hidden unit. Furthermore, because of the nonmonotonicity of the activation function, all points lying in the hyperplane some distance $-\delta$ from the original will also produce the same activation value λ. In other words, all input patterns that lie in parallel planes a set distance below and above the plane bisecting the space as defined by the weight vector will produce the same activation in the hidden unit. Consequently, such input patterns will fall into "bands" when the activation values are plotted. This, of course, holds true for any nonmonotonic function. This is not to say, however, that banding is guaranteed in networks with nonmonotonic functions, only that it is possible. In the same breath, it is not to say that banding cannot occur in networks with monotonic functions, just that it is highly unlikely.

This aspect of the banding phenomenon is illustrated in Figure 6.3 which shows the solution for solving the 3-Parity problem using the value unit structure. In this case, the hidden unit has positioned itself within the input space such that it creates a hyperplane defined by the points [0, 0, 0], [1, 1, 0], and [0, 1, 1]. Consequently, when the Gaussian is applied to these patterns, it produces a value of 1.0. As can be seen in the figure, the points [1, 0, 0], [0, 0, 1], and [1, 1, 1] all lie a distance δ from the hyperplane while the point [0, 1, 0] lies a distance $-\delta$ from the hyperplane. When the Gaussian is applied, it produces a value of around 0.5. Finally, the point [1, 0, 1] lies a distance of 2δ from the hyperplane and produces a value near zero when the Gaussian is applied. Thus, when these value are passed to the output unit and its Gaussian is applied, the values of zero and one are produced, solving the problem.

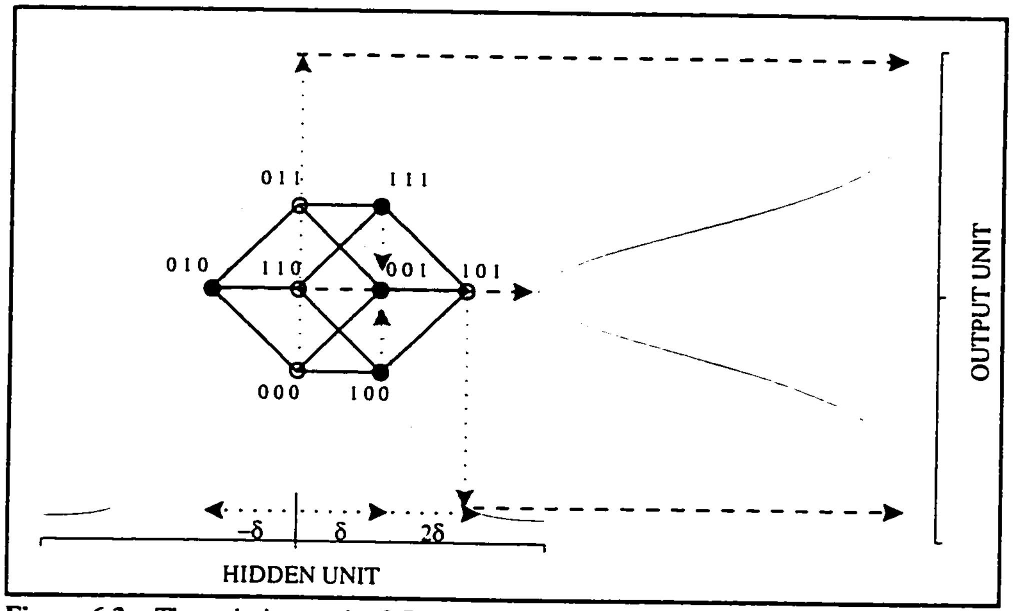

Figure 6.3 The solution to the 3-Parity Problem given one hidden value unit. Note how the circles (indicating points in the pattern space) correspond to the banding structure in Figure 6.1(a) when the hidden unit is correctly positioned in the pattern space.

Although this suffices as a mathematical basis of the observed banding in value units, it is a rather trivial point in terms of the algorithmic analysis of an information processor. All this tells us is how points in a pattern space are converted into hidden unit activations. The more interesting question hinges on what these points in the pattern space actually represent.

Berkeley, et al. (1995) found that in many cases, the density plots for hidden value units are highly structured. In other words, this "banding" provides an important method for interpreting the kinds of features to which a hidden unit is sensitive. Each band in such a density plot supports a coherent interpretation: that is, each pattern falling into a band is characterized by *definite features*—either a specific feature or set of features. These definite features can be quickly identified by calculating simple descriptive statistics such as means, standard deviations, and correlations. To illustrate the utility of identifying definite features, let us consider an example problem.

The Mushroom Problem

Previous research (e.g., Berkeley et al., 1995) has shown that banding occurs for problems such as logical relationships, parity, majority, and determining kinship. These problems, however, were relatively limited in the number of the training patterns used (e.g., the largest data set used was 576 patterns for the logic problem). Therefore, a larger training set was sought to see if banding would scale up. One such data set is the mushroom data set defined by Schlimmer, 1987[3]. The data set offers two improvements over previous training sets—it is an order of magnitude larger than previous data sets used for banding analysis and it has been used as a benchmark for testing traditional machine learning algorithms.

The data set consisted of the hypothetical description of 23 different mushrooms in the *Agaricus* and *Lepiota* family (see Lincoff, 1981, pp. 500-525). Each mushroom was described as a set of 21 different features (see Appendix A). Multiple featural descriptions of one species of mushroom were possible because one species might be found in several different habitats, have more than one possible odour, etc. The total data set consisted of 8,124 different instances. 4,208 of these patterns corresponded to edible mushrooms; the remaining 3,916 training patterns corresponded to inedible mushrooms (i.e., mushrooms that were definitely poisonous, or were of unknown edibility and therefore not recommended).

A Classical Algorithm

Classical accounts of cognitive theories within psychology have their counterparts in the algorithms of traditional Artificial Intelligence (AI). Both approaches are based on the manipulation of symbols by logical rules. Furthermore, both fields have relative difficulty accounting for learning—as opposed to say planning or search (Ginsberg, 1993). In terms of AI, learning can be characterized as either discovery (what could be characterized as insight within psychology) or generalization learning (the extrapolation of knowledge from existing information). An example of generalization learning is concluding that a large, yellow mushroom is poisonous because all large, yellow mushrooms you have seen in the past have been poisonous. This type of generalization learning is classified as inductive learning.

One successful inductive learning technique used in AI research to induce general rules from a set of observed instances is Quinlan's (1986) ID3 algorithm. The ID3 algorithm creates a decision tree by splitting the training instances at each node of the tree into positive and negative instances. In other words, the algorithm makes a series of binary decisions about the input pattern. Thus, any decision tree produced by the ID3 algorithm is equivalent to a series of inference rules (e.g., $\neg$married(m) $\wedge$ man(m) $\rightarrow$ bachelor(m)). The problem with the original algorithm, however, is that

[3] The mushroom database can be retrieved via ftp from ftp.ics.uci.edu: pub/machine-learning-databases, or through the WWW at http://www.ics.uci.edu/~mlearn/MLRepository.html

it is limited to creating trees with at most two children at each node. Therefore, it is inappropriate for data sets that have more than two characteristics within each feature.

Consequently, a modified version of the ID3 algorithm (see Box 6.1) was created to handle multiple characteristics (i.e., more than two) within each feature set. That is, instead of a binary decision at each node, multiple branching from each node was possible. When the new algorithm was applied to the mushroom data set, it produced the decision tree shown in Figure 6.4. All 8,124 different instances of the mushrooms can be correctly identified as either poisonous or edible in no more than five questions. This decision tree is rather unique in that each discriminating element f_i has only one characteristic that is non-exclusive. It should also be noted that there are no mushrooms with Purple Spore Print Colour that could not be classified by Odour first.

The generalization abilities of the decision tree were tested by splitting the data set into a training set and a test set, each containing 2,104 examples of edible mushrooms and 1,958 examples of poisonous mushrooms for a total of 4,062 examples. Because the original data set is laid out in an orderly fashion, simply splitting the data set in half produces a training set that contains characteristics that are not present in the test set, and vice versa. When the new algorithm was applied to the training set, it created a much more complicated decision tree. For example, the first discriminating feature identified was Gill Colour; however, there were four non-exclusive characteristics present within this feature, and each one required a different follow-up question. In total, nine different questions were required to completely classify the training data. Furthermore, when the decision tree was applied to the test data, 464 mushrooms were misclassified and another 224 mushrooms were left unclassified (an overall error rate of 17%).

Given as inputs a set F of features, an overall set S, and a target concept T:

1. If every element of the set S is in T, return "yes". If no element of S is in T, return "no".
2. Otherwise, let f_i be the most discriminating element* of F. If no features remain, return "failure".
3. Starting with the left-most characteristic of f_i and working towards the right-most characteristic, if f_i is such that a subset of S neither satisfies $f_i(S \cap f_i)$ exclusively nor fails to satisfy $f_i(S - f_i)$ exclusively (e.g., the feature f_i does not discriminate T), then recursively return a tree such that all the children finally satisfy T.

* To find the most discriminating element f_i of F, tally the number of positive and negative examples for each characteristic within f_i. Add the total number of non-exclusive characteristics together—noting which ones were not exclusively classified. The most discriminating element f_i is the one with the smallest total. If two elements have the same total, then choose the one with the smallest number of characteristics.

Box 6.1 Modified ID3 algorithm for multiple featural decisions.

Two more generalization tests were conducted. The second generalization test reversed the training and test set, so the decision tree was actually trained on the former test set. This produced a much simpler decision tree based on only two questions, Odour and Spore Print Colour. When tested with the training set, however, the decision tree misclassified 576 mushrooms and left 228 mushrooms as unclassified (an overall error rate of 20%). The third generalization test randomly assigned patterns from the original data set to either the training set or the test set, keeping the same proportion of edible to poisonous mushrooms. This new training set produced a decision tree identical to the one presented in Figure 6.4; consequently, the decision tree classified all examples within the testing set perfectly.

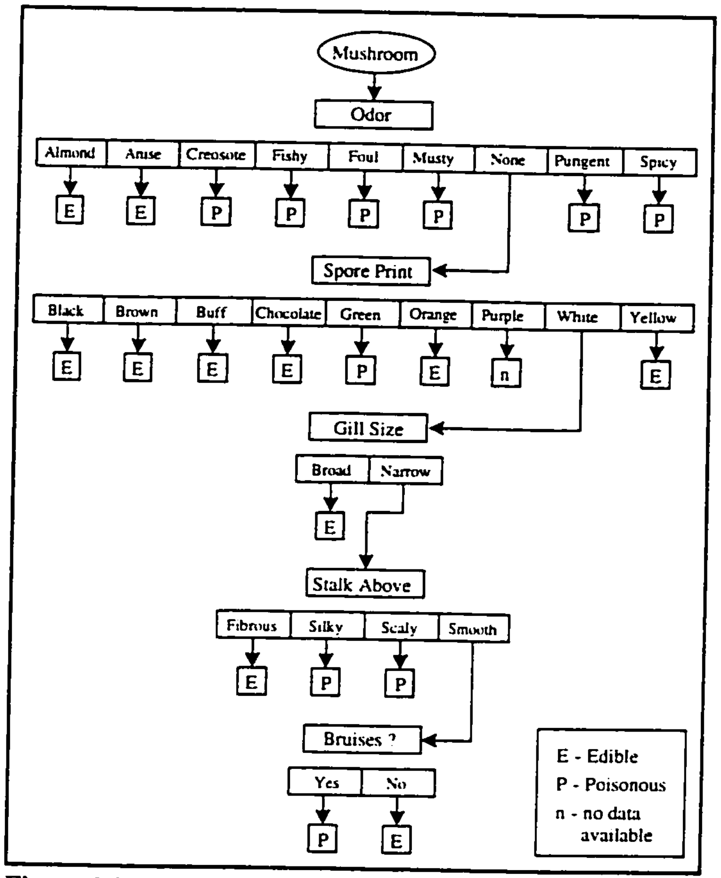

Figure 6.4 Decision tree produced by the modified ID3 algorithm.

As stated earlier, decision trees are equivalent to a series of inference rules. Therefore, we can translate the tree in Figure 6.4 into a set of rules. When asked in order, the five rules presented in Box 6.2 will correctly identify all mushrooms as either edible or poisonous. Given that these rules were produced by a classical system, the question that now arises is whether or not a connectionist network can learn to solve this problem, and if it can, whether or not it has a "different feel" than the classical algorithm.

A Connectionist Algorithm

Given that classification of the data set could be learned by a classical system, the purpose of this experiment was to answer three different questions. First, could a connectionist network learn to accomplish this task? Second, if a network could learn to classify the mushrooms, then would its hidden units reveal density plots that were banded and interpretable? Third, if interpretable structure was discovered in the trained network, then what would the relationship be between this structure and the nested IF-THEN statements in the algorithm described in Box 6.2? In other words, would the network discover new rules (a novel cognitive architecture), or would it simply be a mere implementation of the classical rules discovered by the modified ID3 algorithm.

Step 1 *What is the odour of the mushroom?*
If it is almond or anise then it is edible.
If it is creosote or fishy or foul or musty or pungent or spicy then it is poisonous.
If it has no odour then proceed to Step 2.

Step 2 *Obtain the spore print of the mushroom.*
If the spore print is black or brown or buff or chocolate or orange or yellow then it is edible.
If the spore print is green then it is poisonous.
If the spore print is white then proceed to Step 3.

Step 3 *Examine the gill size of the mushroom.*
If the gill size is broad, then it is edible.
If the gill size is narrow, then proceed to Step 4.

Step 4 *Examine the stalk surface above the mushroom's ring.*
If the surface is fibrous then it is edible.
If the surface is silky or scaly then it is poisonous.
If the surface is smooth the proceed to Step 5.

Step 5 *Examine the mushroom for bruises.*
If it has no bruises then it is edible.
If it has bruises then it is poisonous.

Box 6.2 Algorithm for identifying mushrooms

Network Input

The multiple characteristics within each feature of the data set poses an interesting problem for encoding input data for the network. Basically, one has three choices as to how to represent the data; a unary encoding (one unit for each characteristic which requires 119 input units), a binary encoding (characteristics are represented across sets of input units—this requires 54 input units), or a real value encoding (21 input units are used and characteristics are represented by discrete real numbers). In the version of the mushroom problem described here[4], features were coded as discrete activation values between 0.0 and 1.0 in the 21 input units. Each different activation value corresponded to a different value of the particular feature encoded in that input unit. For example, if there were four different characteristics of a feature, the input values would be 0.0, 0.33, 0.66, and 1.0.

This method of coding the input allowed another question to be answered. Mainly, is banding dependent on binary input patterns, or would discrete but real activation values produce banding as well? Although the mathematical analysis of the banding phenomenon given above should generalize to any discrete input values, this particular input coding allowed an empirical test of this banding theory.

[4] Pilot studies have shown that networks using either unary encoding or binary encoding can successfully learn this problem.

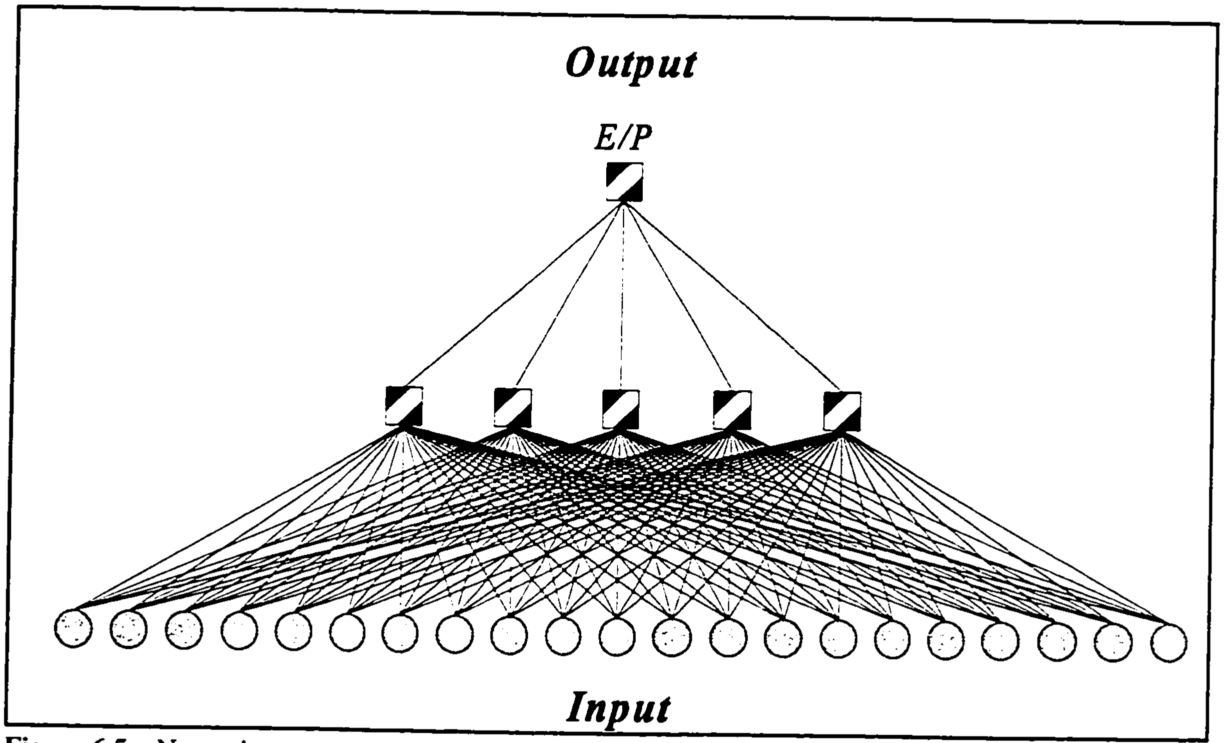

Figure 6.5 Network structure used to solve the mushroom problem.

Network Architecture

The network had 21 input units, five hidden value units, and one output value unit (see Figure 6.5). The output value unit was trained to generate a response of "1" to an edible mushroom, and a response of "0" to an inedible mushroom. Initial connection weights for the network were randomly selected from the range [-1.0, 1.0]. The biases of each hidden unit and of the output unit were set to 0, and were not modified during training. The network was trained with the Dawson and Schopflocher (1992) learning rule, with a learning rate of 0.01 and with no momentum. The network converged (i.e., achieved a hit on every pattern) after only 189 sweeps through the training set. The fact that the network converged provided the answer to the first question—a value unit network could learn to classify mushrooms as being edible or inedible.

The next step was to see if the network solution provided quantitatively different performance in its generalization ability. If the network was solving the problem in some different way than the classical algorithm, then different generalization results should be apparent. (It should be noted that because of the nature of network responses—either zero or one—unclassified responses cannot be explicitly made and are therefore scored as misclassified). To assess its generalization ability, the network was retrained using the generalization training set defined for the classical algorithm and tested on the test set. The network converged on a solution in 572 sweeps. When tested for generalization, the network misclassified 647 patterns or 16% of the test patterns (compare with a 17% error rate for the modified ID3 algorithm). The network was then retrained with the data sets reversed, and the network converged in 162 sweeps. This time, generalization was slightly poorer—the network misclassified 992 patterns (a 24% error rate as compared to 20%). Finally, the network was retrained a third time with the randomly assigned training and test sets. The network converged in 468 sweeps and showed perfect generalization (0% error rate). Consequently, it can be concluded that the (classical) decision tree produced by the modified ID3 algorithm and the connectionist network produce quantitatively similar generalization results.

If the network and the decision tree are producing quantitatively similar results, does this mean that they have qualitatively similar algorithms? To answer this question, banding analysis was applied to the internal structure of the fully trained network.

Interpretation of the Network

The density plots for the five hidden units in the trained network are shown in Figure 6.6. As can be seen, three of the five hidden units reveal distinct banding while the other two show strong activations near zero with scattered patterns representing small subsets of mushrooms. Therefore, we can conclude that—at least from a mathematical basis—banding can occur with discrete, non-binary inputs. But, are these bands interpretable?

Following the practice of Berkeley et al. (1995), descriptive statistics (i.e., means, standard deviations, and correlations) were computed for the set of features that defined the mushrooms that fell into each band identified in Figure 6.6. A *definite unary feature* is defined as a constant characteristic of an input feature for all patterns within the same band. It is determined by calculating the mean and standard deviation for each input feature within the band; a mean of x and a standard deviation of 0.0 for any feature means that it has the constant characteristic x and therefore represents a definite unary feature. *Definite binary features* are determined by pair-wise correlations; a perfect positive correlation means that two input features are always present together while a perfect negative correlation means that two input features are never present together. It should be noted that computing definite binary features requires binary inputs; therefore, for the purposes of computing correlations only, the discrete real inputs used to train the network were converted into their binary counterparts following training. Analysis of both unary and binary definite features reveals that almost all of the distinct bands in Figure 6.6 contain definite features.

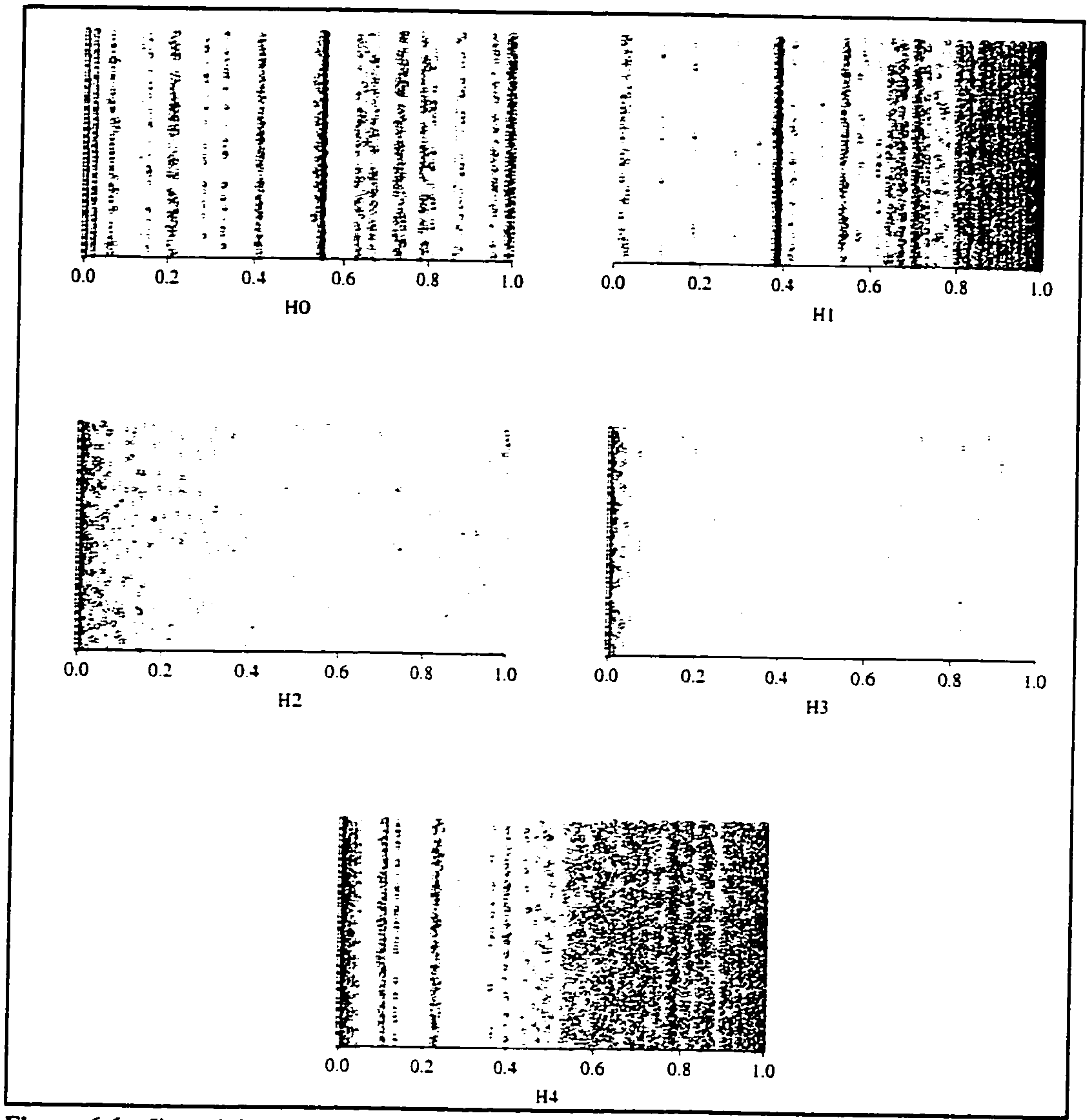

Figure 6.6 Jittered density plots for the five hidden units used to solve the mushroom problem.

A full interpretation of the network is not provided here, however, as a complete interpretation of a network will be provided shortly. Suffice it to say that although many of the bands contained definite unary features, many more of the bands were characterized by definite binary features. For example, if a band had the characteristic "No Odour" it would also have the characteristic "Green Spore Print Colour" whereas if it had the characteristics "Almond or Anise Odour" there were no perfect correlations to spore print colour, although spore print was never green. Consequently, it appears that the network was detecting similar rules to those listed in Box 6.2; that is, the network keyed in to the same features that were detected by the classical rules.

It should be noted that similar findings on network interpretability using the mushroom problem were reported by Dawson and Medler (1996). In contrast to the current network, however, their network used a reduced architecture—the number of input units was reduced from 21 to 16 as determined by pilot studies[5], and only four hidden units were used. The banding analysis of their network revealed bands that contained very specific unary and binary definite features.

For example, all mushrooms falling within Band H of Hidden Unit 2 had the following description:

> Free gill attachment, narrow gill size, tapering stalk shape, stalk surface above ring silky or smooth, stalk surface below ring silky or smooth, stalk color above ring is pink or white, stalk color below ring is pink or white, white veil color, one ring, several population. If it has bruises, then odor is almond or anise, crowded gill spacing, and pendant ring type. If it does not have bruises, then odor is foul or musty, close gill spacing, and evanescent ring type. If odor is almond or foul, then spore print color is brown or green or orange. If odor is anise or musty, then spore print color is buff or chocolate or purple or white. Leaves or path or urban or woods habitat.

This level of detail is typical of the definite features for all of the bands identified in the network reported by Dawson and Medler.

In fact, by analysing the bands produced across units for each pattern, it could easily be determined how the network classified whether a mushroom was edible or not. For the most part, three of the hidden units in the network (hidden units 1, 2, and 3) detected features that were characteristic of poisonous mushrooms. As a result, the network defined a very large class of edible mushrooms as those that failed to produce activity in its hidden units; 4064 (96.58%) of the edible mushrooms produce bands of [0-A, 1-A, 2-A, 3-A]; these mushrooms are classified as edible because the network fails to find any poisonous features in them.

While this approach worked for most of the mushrooms, it failed for a minority of them. A small percentage of edible mushrooms fell into the same (high activity) bands in Hidden Unit 2 as a number of poisonous mushrooms. As a result, a different approach was used to identify the edibility of these mushrooms. The network pooled the activation of hidden units 0 and 2 to solve this problem, essentially by intersecting the features detected by the two units with an AND operation.

[5] During initial pilot studies, it was observed that certain input units always had weights very near zero because they were either constant across all mushrooms or had no correlation on determining edibility. Therefore, those input features were removed from the original training set and the input units were reduced accordingly.

The high activity bands of Hidden Unit 2 captured groups of related mushrooms, but included both poisonous and edible mushrooms. The intersection of the members of the high activity bands from both units defined a set of edible mushrooms—this intersection picked out the edible mushrooms from the bands in Hidden Unit 2 that also contained poisonous mushrooms. Consequently, the three other "rules" needed to identify the remaining 144 edible mushrooms were: [0-C, 1-A, 2-H, 3-A], [0-D, 1-A, 2-J, 3-A], and [0-D, 1-A, 2-K, 3-A]—each of these rules uniquely identifies 48 or 1.14% of the remaining edible mushrooms.

Again, this analysis showed that the network was using similar rules to the ones generated by the classical algorithm, although the exact rules were not duplicated. Indeed, some of the bands discovered in the network reported by Dawson and Medler represent classes that emerge very late in the Box 6.2 algorithm. However, because the mushroom network is processing many other features of mushrooms—many of which may be correlated with features that the Box 6.2 algorithm ignores—many of the classes revealed in the network appear to be more complex than those that could be derived from the algorithm.

Consequently, the analysis of the network structure shows that although the network produces quantitatively similar results to the classical algorithm, it has in fact discovered another algorithm for correctly classifying the mushroom data set. In other words, the network is not merely implementing a classical algorithm, but has produced a novel algorithm. Some classical cognitive scientists may argue, however, that the network is only implementing a classical algorithm that has yet to be discovered (say, by some other classical learning paradigm besides the modified ID3 algorithm). But, this type of argument is circular and amounts to no more than moving the goal posts.

Classical Rules and Connectionist Networks

Although the above results are interesting in their own right, they lead us to a new question. Can a connectionist network be taught explicit rules, and if so, what would the banding structure be like?

Normally, in pattern classification, a network is only informed of what the correct classification should be. Consequently, the number of output units is limited to the number of possible output classifications. But, it is often the case that more information than just the classification is actually available; that is, information about why an input pattern is one class or another may be available. In certain respects, this could be viewed as preprocessing of the data, although not in the normal sense. Usually, preprocessing involves identifying those features in the input that are relevant for classification and those features that are redundant and therefore can be removed. In other words, preprocessing is used to *reduce* the amount of information at the input level (this type of preprocessing was done in the original mushroom network).

Another way of preprocessing the data would be to *add* information at the output level. In terms of a neural network, this would involve adding extra output units that could be trained to express some prior knowledge about the problem. This has been termed 'injection of hints' or 'extra output learning' (e.g., Suddarth, Sutton, & Holden, 1988; Yu & Simmons, 1990; Gällmo & Carlström, 1995). Thus, preprocessing in this sense becomes a powerful method for incorporating prior knowledge. Furthermore, once trained, the extra output units can be removed without affecting the classification performance of the network.

Inserting Rules

To see if rules could be inserted into the network via elaborated outputs, an experiment was conducted in which the classical rules presented in Box 6.2 were used to elaborate the output. Although previous research into output elaboration has focused on improving the learning speed and generalization ability of networks (Gällmo & Carlström, 1995), the current research was more interested in how inserting rules would affect the internal structure of the networks as determined by the banding analysis.

Network Architecture

The network used 21 input units, five hidden units and ten output units. Again, input activation values were discrete real numbers between 0.0 and 1.0. It was conjectured that because there were five hidden units and five rules in Box 6.2 algorithm, each hidden unit may be tuned to a specific rule. Ten different output units were used in the network. One unit encoded for the edible/poisonous classification, and the other 9 units encoded whether or not the mushroom was classified by Rule 1 edible (R1e), Rule 1 poisonous (R1p), etc. There was only one unit for the third rule as Rule 3 only produces an edible or unclassified response. Furthermore, only one "rule" output unit was activated at any time. The network structure is shown in Figure 6.7.

Network parameters were slightly changed from the previously reported experiment. Weights were randomized between ± 1.0, and all biases were initialized to zero; this time, however, biases were modified during training. The hit criterion was set to 0.0025, learning rate was set to $\alpha = 0.005$, and no momentum was used.

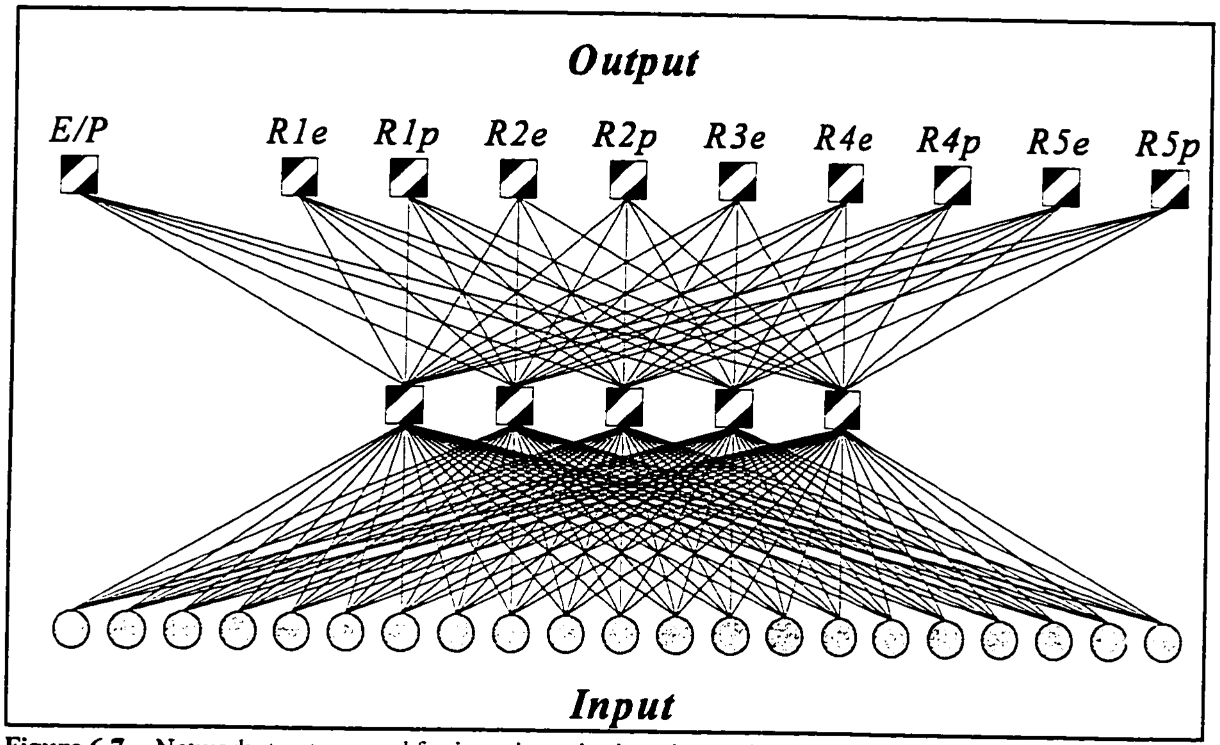

Figure 6.7 Network structure used for inserting rules into the mushroom network. One output unit is used for classification (E/P) and the other nine are used to indicate the rule necessary for classification (e.g., Rule 1 edible ≡ R1e).

Network Interpretation

First, it should be noted that it took longer to train the network with the elaborated output than the original mushroom network; convergence was reached after 8,699 sweeps through the pattern set. Second, and more importantly, the network showed very marked banding. As can be seen in Figure 6.8, injecting rules into the network via extra output units cleaned up the hidden unit banding structure as compared to the standard network illustrated in Figure 6.6. Definite feature analysis of the patterns falling into each band showed that all bands had highly interpretable structure.

Interpretation of Hidden Unit 0 shows that there are 7,020 pattern falling into 0-A and that all the patterns in this band share the one constant feature of a white veil colour and an odour that is *not* almond or anise (a characteristic of edible mushrooms). The 18 patterns that appear separated from the main cluster are included in 0-A because they have a white veil colour and have no ring while all the other patterns in 0-B and 0-C have one ring. The 280 patterns falling into 0-B all have no bruises, no odour, an enlarging stalk shape, and one ring. Band 0-C contains 824 patterns that have a free gill attachment, a smooth stalk surface above ring, a white stalk colour above ring, and

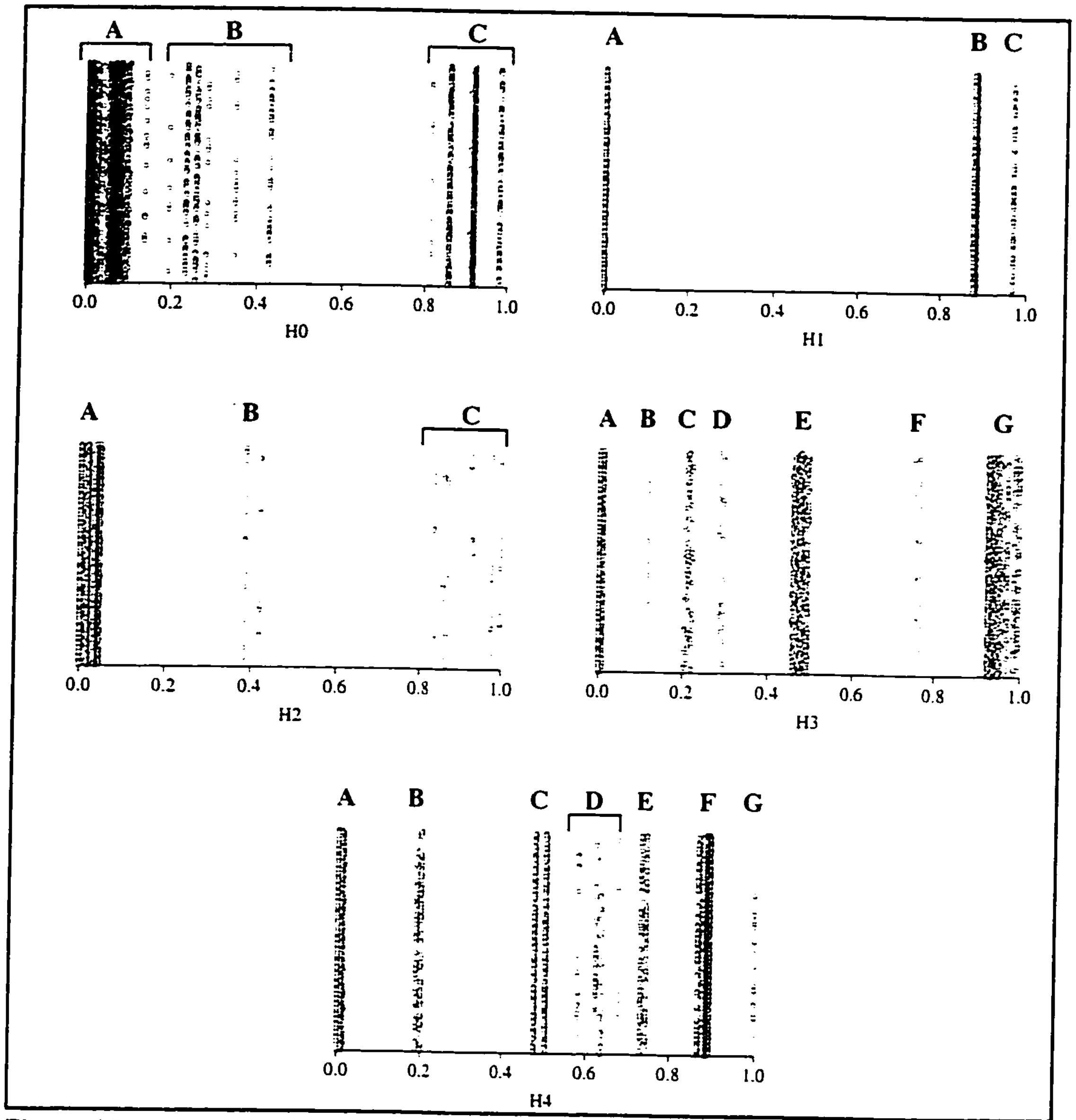

Figure 6.8 Jittered density plots for the five hidden units used in the network trained with the elaborated output.

a white veil colour in addition to one ring. Furthermore, both 0-B and 0-C show extreme forms of microbanding. For example, in addition to the above characteristics, the 12 patterns forming the lower bound of 0-C have no bruises, no odour, an enlarging stalk shape, brown stalk colour below ring, an evanescent ring type, a white spore print colour, and are found living in leaves. The 96 patterns forming the upper bound of 0-C, on the other hand, have bruises, an almond or anise odour, a tapering stalk shape, white stalk colour below ring, a pendant ring type, a brown or purple spore print colour, and are found in woods. Both microbands share crowded gill spacing, narrow gill size, a smooth stalk surface below ring, and a population of several. The remaining microbands in 0-B and 0-C have similarly rich interpretations.

Hidden Unit 1 has three very distinct bands. The 7, 348 patterns that are in 1-A are delineated by not having certain characteristics; specifically, both the gill colour and the spore prints colour is not green (remember that a green spore print indicates a poisonous mushroom by Rule 2). Band 1-B contains 704 patterns that all have bruises, have an almond or anise odour, free gill attachment, close spacing, broad gill size, enlarging stalk shape, smooth stalk surface above ring, white stalk colour both above and below ring, white veil colour, one pendant ring, and a black or brown spore print colour. All patterns falling into 1-C, also have bruises, but no odour, free gill attachment, close spacing, broad gill size enlarging stalk shape, smooth stalk surface both above and below ring, white stalk colour both above and below ring, white veil colour, two pendant rings, and—importantly—green spore print colour. In terms of the rules, the important distinction between 1-B and 1-C is that 1-B has almond or anise odour, while 1-C has no odour and green spore print colour.

Hidden Unit 2 is characterized by three bands that again contain microbanding. The majority of patterns (8028) are in 2-A which contains no definite features. Band 2-B has 24 patterns with the important characteristics of no bruises, no odour, narrow gill size, a fibrous stalk surface above the ring, and a white spore print colour. In other words, all the patterns in 2-B are classified as edible because they satisfy Rule 4e. The final 72 patterns are in 2-C which contains all mushrooms with no odour, white spore print colour, narrow gill size, and either a scaly, silky, or smooth stalk surface above the ring. Furthermore, definite binary feature analysis revealed that if the mushroom had a scaly or silky stalk surface above ring, then the mushroom always had no bruises. This means that if the mushroom had a smooth stalk surface above the ring, then the mushroom may or may not have had bruises which is an important characteristic for determining if a mushroom is edible or poisonous (Rule 5).

Both Hidden Unit 3 and Hidden Unit 4 revealed seven different bands. Again, each band has very rich interpretation. For the sake of brevity, however, only the highlights of the units will be discussed. Basically, Hidden Unit 3 detects mushrooms primarily based on odour. Band 3-A does not contain any mushrooms with either almond or anise odour. All other bands in Hidden Unit 3 contain mushrooms with almond or anise odour or mushrooms with no odour. Those bands containing mushrooms with no odour all have white spore print and narrow gill spacing and therefore are differentiated by either the stalk surface above the ring, or if the stalk surface above ring is smooth, by the presence or absence of bruises.

In Hidden Unit 4, all mushrooms falling into the higher bands have no odour, but can be differentiated because of their spore print colour. The exceptions to this are 4-D which contains mushrooms with both chocolate and white spore print colours and 4-G which contains mushrooms with only white spore print colour.

With the interpretation of the internal structure of the hidden units completed, we can now turn to how the network is actually solving the problem. Of the possible 1,323 band combinations (3 x 3 x 3 x 7 x 7), the network only produced 15 different combinations. Moreover, each of these combinations uniquely identified one of the classical rules as laid out in Box 6.1. These different combinations are presented in Table 6.1 along with the rule that the bands encode.

Table 6.1

Band combinations and the rules they encode

Band Combination	Rule	Band Combination	Rule
[0-C, 1-A, 2-A, 3-C, 4-A] [0-C, 1-B, 2-A, 3-E, 4-C] [0-C, 1-B, 2-A, 3-G, 4-B]	Rule 1e	[0-A, 1-A, 2-A, 3-A, 4-A]	Rule 1p
[0-A, 1-A, 2-A, 3-A, 4-F] [0-B, 1-A, 2-A, 3-A, 4-D] [0-B, 1-A, 2-A, 3-A, 4-E]	Rule 2e	[0-A, 1-C, 2-A, 3-G, 4-A]	Rule 2p
[0-A, 1-A, 2-A, 3-G, 4-A]	Rule 3e		
[0-A, 1-A, 2-B, 3-D, 4-G] [0-A, 1-A, 2-B, 3-F, 4-D]	Rule 4e	[0-B, 1-A, 2-C, 3-A, 4-D]	Rule 4p
[0-C, 1-A, 2-C, 3-D, 4-G] [0-C, 1-A, 2-C, 3-F, 4-D]	Rule 5e	[0-A, 1-A, 2-C, 3-B, 4-A]	Rule 5p

As can be seen, only four band combinations are required to classify all poisonous mushrooms. For example, the combination [0-A, 1-A, 2-A, 3-A, 4-A] contains all 3,796 patterns that fail Rule 1 and therefore are poisonous. Rule 2p is represented by combining 1-C—which has contains mushrooms with a green spore print colour—with 3-G (black, brown, green, or white spore print colour). Combining 0-B, 2-C, and 4-D picks out those mushrooms with no odour, white spore print colour, narrow gill spacing, and silky or scaly stalk surface above the ring. Finally, the eight mushrooms that are classified as poisonous because of Rule 5 are picked out by the band combination [0-A, 1-A, 2-C, 3-B, 4-A].

Identifying edible mushrooms, on the other hand, requires either one (Rule 3e), two (Rules 4e & 5e), or three (Rules 1e & 2e) different band combinations. The interpretation of the band combinations for the edible rules are similar to those for the poisonous rules and therefore are not elaborated. It should be noted, however, that these different combinations often contain bands that are also used to identify poisonous mushrooms (e.g., 0-B, 2-C, 3-G, 4-D). It is the combination of bands, nevertheless, that distinguish the rules; for example, Rule 4p and 2e are distinguished because 4p has Band 2-C whereas 2e does not.

Consequently, these results show that classical rules can be inserted into the hidden unit structure of a connectionist network. In other words, prior knowledge can be used to constrain how a network learns to solve a problem. Contrary to the original hypothesis, however, the network did not adopt a "one rule - one hidden unit" type of structure. Instead, rules were uniquely encoded across hidden units in a distributed manner. The banding analysis, nevertheless, makes it fairly straight-forward to interpret the distributed nature of the network and, therefore, makes the goo a little less intimidating.

Are They Really Rules?

From the above results, it is evident that rules can be inserted into the structure of a connectionist network. But, the skeptic might ask, would any elaborated output suffice for creating the highly banded structure, or was there something special about the outputs chosen? In other words, were the extra output units contributing any useful structure to the network, or was the banding due to having a higher dimensionality of the output space and the rules a mere side-effect of this?

To answer these questions, a control network was trained with randomly elaborated outputs. The network had 21 input units, five hidden units, and ten output units as before. One output unit was used for classifying the mushroom as edible or poisonous, while the other nine output units were assigned random values. That is, for each input pattern, one of the nine extra output units was randomly assigned to be turned on whenever that pattern was presented. Therefore, "random rules" were inserted into the network.

Network Interpretation

The control network trained with the random rules completely failed to converge on a solution within 10,000 sweeps through the training set. To ensure this was not a local minimum, the maximum number of sweeps was increased to 20,000 and ten different networks were trained. Again, none of the networks reached convergence. In fact, of the 81,240 possible hits (10 output units × 8,124 patterns), the networks typically produced around 7,000 hits and 74,000 misses. Clearly, the insertion of random rules disrupts network learning.

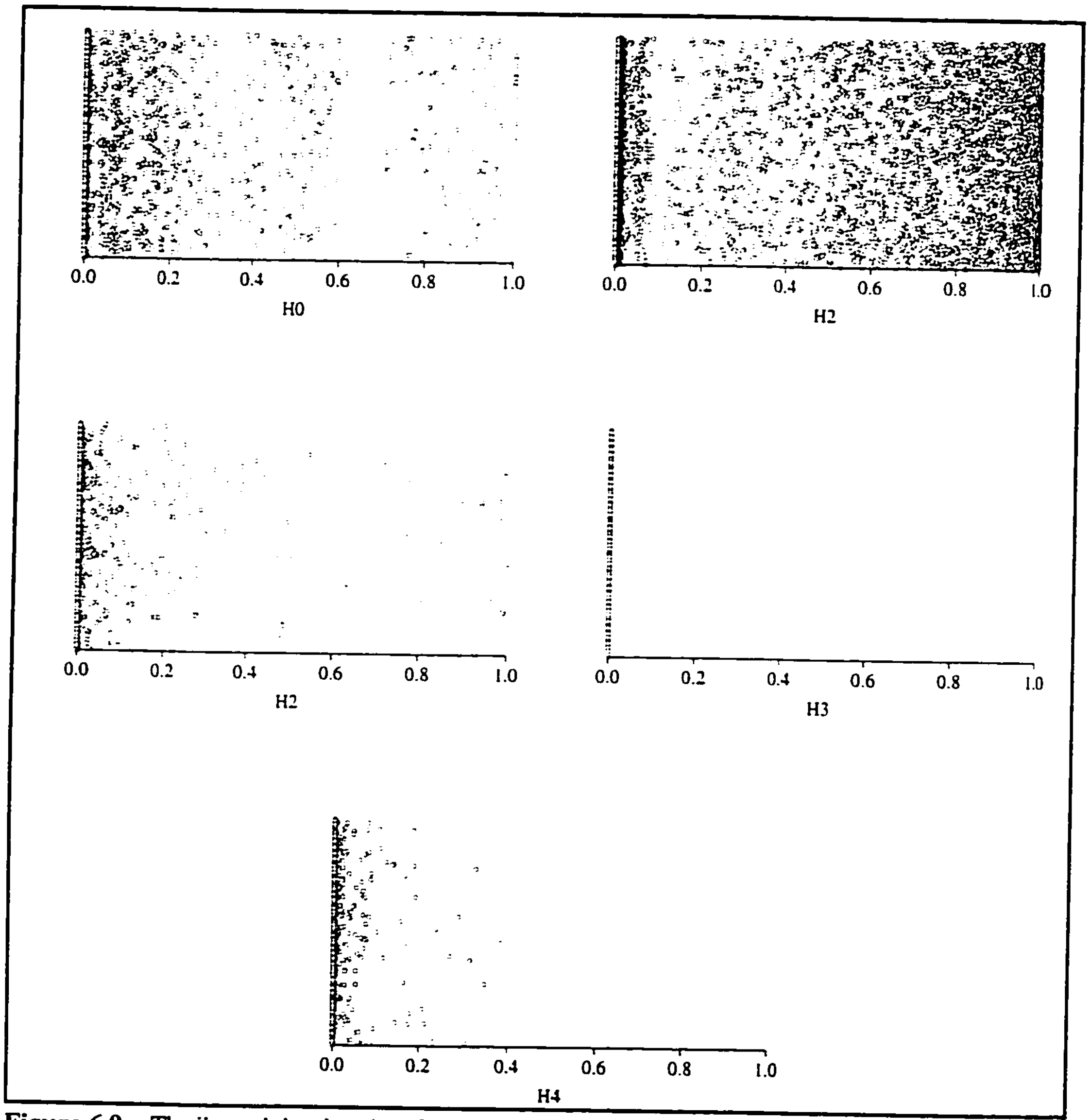

Figure 6.9 The jittered density plots for a network trained with "random rules" in the elaborated outputs. Convergence was never attained, and no banding structure is evident in any of the plots.

Although convergence was not attained, the density plots of the hidden units were graphed to see if any banding like that shown in Figure 6.6 or 6.8was evident. As can be seen in Figure 6.9, however, no banding structure is evident within the hidden units at all. In fact, one hidden unit produces absolutely no activation at all. Thus, we can conclude that random rules at the output level do not produce banding in the hidden layer.

Algorithmic Level Conclusions

It has been argued that connectionist models are no more than mere implementations of classical cognitive theories (e.g., Fodor & Pylyshyn, 1988). Furthermore, while some connectionists maintain that their networks are in fact algorithms (Rumelhart & McClelland, 1985), the network structure is often so complex that it defies traditional analysis techniques and thus has been likened to a pile of goo. Consequently, although both connectionist and classical researchers would like to be able to say something about the algorithmic nature of networks (one for and one against), the opacity of network structures make this a difficult task indeed.

This chapter reviewed and expanded a new interpretation technique for a specific type of connectionist network that makes this task more tractable. An overview of the banding analysis of hidden value unit structure was provided. Briefly, following training, a network is presented with the input patterns once again and the activities produced in the hidden units are recorded and plotted in jittered density plots. It turns out that these plots are highly structured and form "bands"; furthermore, all the patterns that fall into a certain band usually share definite unary or binary features. Consequently, the banding patterns are interpretable and, when combined across units, form an easily identified algorithm for solving the pattern classification problem.

Although sharing some characteristics, the algorithms provided by the banding analysis are not the same as the classical rules produced by more traditional techniques. In other words, connectionist network do not provide merely implementational accounts of classical cognitive algorithms, but produce novel cognitive theories (for a more in depth philosophical discussion of this point, see Dawson, Medler, & Berkeley, 1997). At the same time, however, it was shown that classical rules could be inserted into a network's structure by elaborating the output responses of the network. This implies that using prior knowledge during training produces network structures that may not be that dissimilar from classical theories of cognitive algorithms.

In conclusion, the results from this chapter indicate that this banding analysis allows cognitive scientists to generate detailed algorithmic accounts of PDP models of cognitive processes.

Hence, connectionist models should be able to inform other empirical investigations of cognition (e.g., by suggesting experiments to be performed by cognitive psychology). Furthermore, although connectionist networks—if left on their own—can discover novel cognitive theories, classical cognitive theories can also be recognized within the network structure. Thus, it would seem that the rift separating classical and connectionist researchers is not as large as we have been led to believe.

Chapter 7

The Functional Architecture & Connectionism

Our analysis of the algorithmic level has shown that connectionism is not a mere implementation of existing classical models, but in fact can produce novel cognitive theories. This result is a direct consequence of interpreting the internal structure of hidden units via banding analysis. Although this result is important in and of itself, our analysis of the algorithmic level is not quite complete. As stated in Chapter 2, we must also study the functional architecture in order to make claims of strong equivalence.

The functional architecture is the foundation of the algorithmic level; it consists of those basic processing steps from which all algorithms are built. Furthermore, the functional architecture acts as the bridge between analyses at the algorithmic and implementational levels. In other words, the functional architecture links the "cognitive" with the "neural" level—it is where mind becomes fully instantiated within brain.

Although it is tempting to assign the functional architecture its own distinct level (the quad-level hypothesis?), it rightly belongs at the base of the algorithmic level (e.g., see Figure 2.3). A quick analogy will make this clear. Computing science makes the distinction between software (the program) and hardware (the physical machine). Regardless of the actual programming language used—C, Pascal, Fortran, Lisp, etc.—software must somehow be implemented on the hardware. To accomplish this, all languages are derived from the same building blocks; that is, machine code.

Machine code is the base set of instructions (programs) that can be directly implemented in a physical device. The study of the functional architecture, then, is equivalent to the study of machine code.

This analogy also highlights a very real problem in the study of the functional architecture. Although it would seem to be a fairly simple exercise to identify the machine code and write programs, it is in fact a very difficult—if not near impossible—problem when the instructions are missing. This exact problem is aptly described by Arbib (1972). In 1966, a new computer was installed at Stanford University; however, the instruction manuals failed to arrive. A group of determined graduate students declared that the instructions were not needed as they would be able to deduce the machine code (and write meaningful programs) simply by studying the structure of the physical machine. After several hours of experimentation, however, the students were unable to deduce anything meaningful about the machine code. Consequently, the graduate students were unable to write even the simplest of programs and were lead to proclaim that the computer was quite perverse. When the instructions finally arrived, however, it was revealed that the new machine was in fact very similar to the machines previously used by the students.

Studying the functional architecture is akin to the graduate students trying to discern the machine code of the computer—it would be a relatively simple exercise, but someone forgot to give us the instructions to the brain. Unfortunately, this does not bode well for cognitive scientists as it seems they have a nearly impossible task ahead of themselves. Exactly how, then, does one go about studying the functional architecture of the mind?

Initial studies of the functional architecture were limited to interpreting results from cognitive psychology experiments. One of the best examples of this line of research is the long running imagery debate between Stephen Kosslyn and Zenon Pylyshyn (e.g., Kosslyn, 1980; Pylyshyn, 1973, 1981). The debate started in 1973 with Pylyshyn's critique of mental imagery; he began by intimating that the very study of imagery was paradoxical and muddled (How can one look at a mental image? Why can't we count the stripes of an imagined tiger?). Instead, Pylyshyn proposed that propositional structures are used for all forms of cognition, including imagery.

Kosslyn, however, believed that images are themselves a component of the functional architecture. This conclusion was based on a series of experiments in which subjects were instructed to study a map, and then asked to find certain landmarks on the map from memory (e.g., Kosslyn, Ball, & Reiser, 1978). Results from these experiments showed that the time required for the subjects to respond was directly related to the distance between the landmarks. Kosslyn interpreted this to mean that subjects must store and manipulate mental images.

Pylyshyn (1981) has pointed out, however, that if the instructions are changed to remove any explicit or implicit implications of time (i.e., from pushing the button when you "arrive" at the second location, to what is the compass bearing of the second location) then the relation between reaction time and distance disappears. In other words, by changing the belief of the subject, the subject's behaviour is changed. Pylyshyn (1984) argues that the functional architecture must be cognitively impenetrable; that is, changing belief does not change performance. Because mental images are subject to belief, then they must not be part of the functional architecture.

But, Kosslyn (1990) cites several cognitive studies that show that changing the belief of subjects does not, in fact, change their performance on imagery tasks. Furthermore, he maintains that the study of mental imagery is the study of the functional architecture and may be one of the first case studies of how the brain creates the mind.

> The study of mental imagery is interesting in part as a bridge between perception and mental activity. As such, it is the cognitive faculty "closest to the neurology" because so much is now known about the neural mechanisms of perception. (Kosslyn, 1990, p. 94)

The imagery debate has recently culminated in Kosslyn (1994) declaring the debate over in favour of mental imagery (*Image and Brain: The Resolution of the Imagery Debate*). Part of the reason for this bold claim is that Kosslyn has moved from using purely cognitive tasks to using brain imaging techniques. Behavioural data—in Kosslyn's view—has proven inadequate to answer the question of whether or not visual mental images rely on a representation that depicts information. Thus, the only way to answer this question is to image what the brain is doing during mental imagery tasks. For example, Kosslyn et al. (1996) report that activation in area 17 is correlated with response times for subjects visualizing letters. Furthermore, Tagaris et al. (1997), have shown occipital lobe activation during mental rotation studies is associated with error performance.

These two experiments highlight one of the approaches to studying the functional architecture today; that is, recording the activity of the brain during mental processing. Cognitive neuroscientists have an arsenal of different analysis techniques available to them today to aid in this endeavour. Examples of these include invasive procedures such as single cell recording (e.g., Földiák & Young, 1995) and neural circuit analysis (e.g., Byrne & Crow, 1995), and various non-invasive imaging procedures including Event-Related Potentials (ERPs), functional Magnetic Resonance

Imaging (fMRI), Positron Emission Tomography (PET), and Single Photon Emission Computed Tomography (SPECT) (e.g.,Arbib, 1995).

Another method often employed to study the functional architecture is to note behavioural deficits associated with some form of brain disturbance. In other words, function is inferred from dysfunction. One such example is the dual-route model of reading advanced by Coltheart (e.g., 1980, 1983). The different routes were proposed after noting that patients suffering from various injuries manifested different forms of acquired dyslexia depending on the location of the injury. Although inferring function from dysfunction is a useful technique, it is limited to correlational and not causal inferences. This is because patients are studied only following trauma, and therefore, there is no measure of their pre-trauma baseline performance. Consequently, their behaviour can only be correlated with the behaviour of normal subjects.

Connectionism is another approach often used to address questions about the functional architecture. In fact, it may be in a very unique position to contribute to analyses of the functional architecture—specifically, in inferring function from dysfunction. This is because connectionism allows a causal link to be made between behavioural deficits and lesioned structure; that is, we can measure the performance of a network both pre- and post-trauma and directly relate any differences in performance to disturbances in the network structure. Consequently, there is a large and growing literature on using connectionist networks to model neuropsychology deficits (for a review of numerous models and their implications, see Farah, 1994; Plaut, 1995).

In this chapter, the results of lesioning two connectionist networks are used to specifically address issues about the functional architecture. The first experiment examines how semantic knowledge may be organized within memory, and how this organization may breakdown in patients with Alzheimer's Disease (AD). One assumption is that knowledge is organized in a semantic network (e.g., Collins & Quillian, 1969; Collins & Loftus, 1975) and that the performance of AD patients is characterized by a breakdown in the structure of semantic knowledge, which includes the formation of new abnormal associations and clusterings (e.g., Chan et al., 1993). The results from the first experiment challenges both of these assumptions and suggests an alternative architecture—specifically, a modification of the IAC architecture—to account for normal and patient data.

The second experiment addresses the locality assumption (see Farah, 1994) that is often adopted within cognitive neuroscience. Farah has argued against the locality assumption based upon the results from lesioned connectionist networks, using the tact that the "nonlocal" nature of PDP networks precludes the locality assumption. Current results, especially when combined with the results from Chapter 6, indicate that the assumption that PDP models are nonlocal may be incorrect, and therefore that the locality assumption should not be dismissed simply based on the performance of lesioned networks.

Both sets of experiments will be used to address fundamental questions about the functional architecture—the first about how semantic knowledge may be stored, the second about how local systems may develop. In both cases, lesioning the networks is an essential component of studying the functional architecture. This is because if our assumptions about the functional architecture are correct, then intact networks should produce results in line with the behaviour of normal subjects and lesioned networks should produce results in line with patient data. If the results are different, then our assumptions about the functional architecture will be incorrect.

Semantic Disturbances in AD Patients

Patients suffering from Alzheimer's disease (AD) often show deterioration in the hierarchical structure of semantic memory. These disruptions manifest themselves in the poor performance of AD patients on tests of verbal fluency. For example, Chan et al., (1993) have reported that AD patients produce significantly fewer animal names than age-matched normal controls. Furthermore, the semantic maps of AD patients are abnormal in that the general semantic structure appears to be breaking down; there is an increase in the distance between concepts and abnormal clusters are formed.

To account for these data, Chan et al., (1993) assume that semantic knowledge is represented within a semantic network of the type proposed by Collins and Loftus (1975). Consequently, extended distances between animal names within AD patients' cognitive maps are interpreted as a weakening of the associations between concepts. Furthermore, the abnormal clusters are seen as a disruption in semantic organization due in part to the "establishment of new, albeit abnormal, associations" (p. 259). Unfortunately, given the etiology of AD, the active reorganization of semantic memory seems unlikely. Thus, the assumption that semantic knowledge is encoded within a classical semantic network may be incorrect. In other words, the functional architecture underlying semantic memory may not be a semantic network of the Collins and Loftus type.

If a classical semantic network fails to account for the data, then what are the alternatives? We have already been introduced to one alternative in Chapter 4—the Interactive Activation and Competition (IAC) model of McClelland (1981). Such a network encodes semantic knowledge in nodes that are connected to each other via both positively (activation) and negatively (competition) weighted connections. Thus, it differs from classical semantic networks. The question arises as to whether or not these positive and negative connections are required for the functional architecture of semantic knowledge. If it can be shown that a lesioned IAC network (i.e., one not dependent on actively forming new connections) can produce qualitatively similar results to those obtained from AD patients, then an alternative account of the functional architecture will be provided.

A Connectionist Account of Semantic Disturbances

Chan et al.'s (1993) results were based on subjects' responses during an Animal Fluency Test that is administered as part of a standard neuropsychological battery used at the Alzheimer's Disease Research Center at the University of California, San Diego. Subjects were requested to name as many animals as possible within 60 seconds; researchers recorded the responses verbatim and in the order in which they were generated.

To simulate these results, an IAC network was created to encode both animal names and a subset of their semantic properties. Network responses (i.e., animal names) were recorded over a set number of cycles, and compared to results reported by Chan et al. In other words, the network was asked to recall as many names as possible over a set amount of time.

Normally, the IAC architecture is used to recall information by allowing the network to settle into a steady state. For example, to find the properties of Lance, one would activate the node for "Lance" and then let the network settle into a state in which the property nodes "Jets", "20's", "J.H.", "Married", and "Burglar" were all active at the same time. For the recall of animal names, however, a steady state needs to be avoided. Thus, the basic IAC architecture was modified to include a suppression cohort. Basically, the suppression nodes within this cohort act to drive down the activation values of recently activated name nodes and associated instance nodes. In other words, once an animal's name is recalled, it is suppressed to prevent repetitious recall.

Method

Stimuli: The animal names encoded in the network were chosen from Table 8 (A FOUR-FOOTED ANIMAL) of Battig and Montague (1969). This table lists both the most commonly recalled animals and their frequency of occurrence (maximum 442). Only those animals recalled at least ten times were included in the network. Thus, a total of 42 animals were chosen, with frequencies ranging from 426 for "dog" to 11 for "elk."

In the data reported by Chan et al. (1993), three different semantic categories were identified: size (small, large), diet (herbivore, carnivore), and domesticity (domestic, wild). The current study also uses these three categories; however, these categories have been expanded and two more categories have been added. Therefore, knowledge is encoded into the network using five different semantic categories. These categories are size (small, medium, large), domesticity (domestic, wild), diet (omnivore, carnivore, herbivore), range (Americas, Africa), and type (canine, feline, equine, bovine, ovine/porcine, cervidae, rodent, other). It should be noted these characteristics are only a small subset of all possible characteristics and that the characteristics assigned to certain animals are sometimes spurious and not meant to be fully accurate. The 42 animal names and their assigned semantic characteristics are reported in Appendix B.

Network Architecture: The network architecture is based on a slight modification of the standard Interactive Activation and Competition (IAC) model of McClelland (1981). The basic network structure is illustrated in Figure 7.1. There are nine different cohorts within the network: one instance cohort, five property cohorts (size, domesticity, diet, location, type), a names cohort, a priming cohort, and a suppression cohort. Positive connections exist between cohorts, and negative connections exist within cohorts (the suppression cohort is the one exception to this).

There are 42 different instance nodes—one for each animal name—within the instance cohort (represented by the filled circles in Figure 7.1). Connections to other nodes within the cohort have a value of -0.5, and all connections from instance nodes to property nodes (represented by open circles in Figure 7.1) have a value of 1.0. All property nodes have inhibitory connections of -1.0 to other nodes within their cohort, and have a self-excitatory connection of 0.5. The excitatory connections from the property nodes to the instance nodes were randomized between [0.5, 1.0] to represent different degrees of knowledge. In other words, given the instance of "lion", it is easy to produce the characteristic of "feline"; but, it may be more difficult to produce "lion" as an instance of "feline".

The priming cohort had one node—"animals"—which was used to prime the network in order to recall the animal names (if the network were expanded to include other semantic categories, other nodes, such as "clothing" could be added to this cohort). As the priming node was clamped "on" for recall, a self-excitatory connection was not required. The priming node was connected to the instance nodes by randomized excitatory connections. To compute these weights, the recall frequency of animal names (see Appendix B) was normalized and used to scale a random value

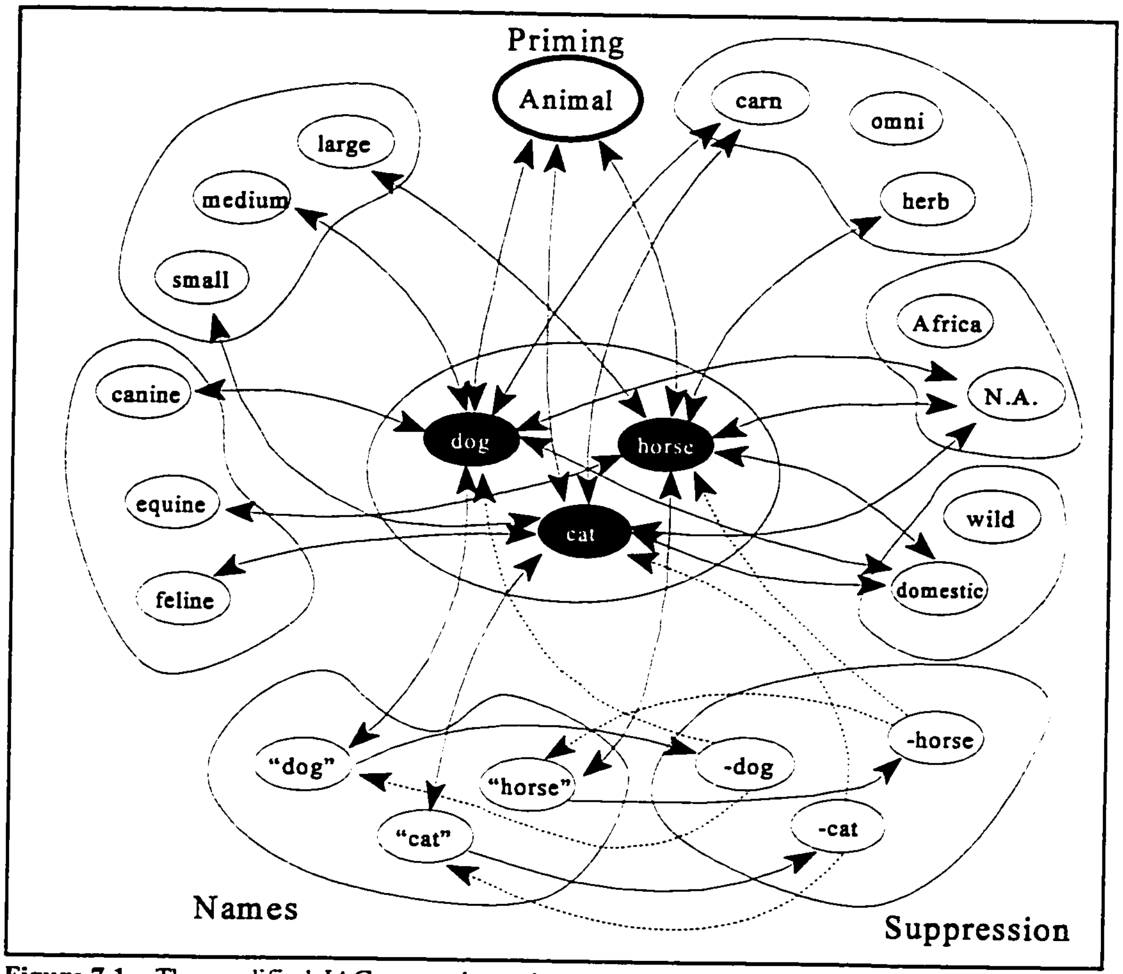

Figure 7.1 The modified IAC network used to recall animal names. Solid lines indicated excitatory connections, broken lines are inhibitory connections, and arrows indicate direction of signal. Note that not all units are shown nor are inhibitory connections within each cohort.

between [0.0, 0.5] which was then added to a base weight of 0.5. Hence, weights from the priming cohort to the instance cohort fell within the range [0.5, 1.0]. Again, this randomization was performed to represent differing amounts of semantic knowledge.

Nodes within the names cohort had self-excitatory connections of 0.5 and inhibitory connections to other name nodes of -0.5. Each name node had an excitatory connection of 1.0 to the appropriate instance node, and an excitatory connection of 2.0 to the appropriate suppression node. Nodes within the suppression cohort were rather unique, in that they only received excitatory connections from the name nodes (instead of the instance nodes) and a self-excitatory signal of 2.0. Inhibitory connections to other nodes within the suppression cohort were set to -0.01 and the inhibitory connections to the appropriate nodes with the names cohort and the instance cohort were set to value of -3.0.

Recording Network Behaviour: To assess the recall abilities of the network, ten different "subjects" were created by randomly setting the weights between the property cohorts and the instance cohort as described above. In effect, each network represented a subject with differing degrees of semantic knowledge. These networks were considered as "normal" or control subjects.

Each network was primed by clamping the 'Animals' node to a value of 1.0. The network was then allowed to process the signal for 180 cycles. At each cycle, the activation values for each of the 42 names were recorded. Following processing, the activation values were plotted (see Figure 7.2) and an animal's name was considered recalled at the peak of an activation. A "winner-take-all" strategy was adopted for the processing signal; that is, if the peak of one activation curve was overshadowed be the processing of another signal, then the overshadowed animal was considered **not** recalled.

Normal Performance

Results show that the modified IAC network successfully recalls animal names. In fact, the networks recalled on average 14.9 (s.d. = 2.84) animal names within the 180 processing cycles. The

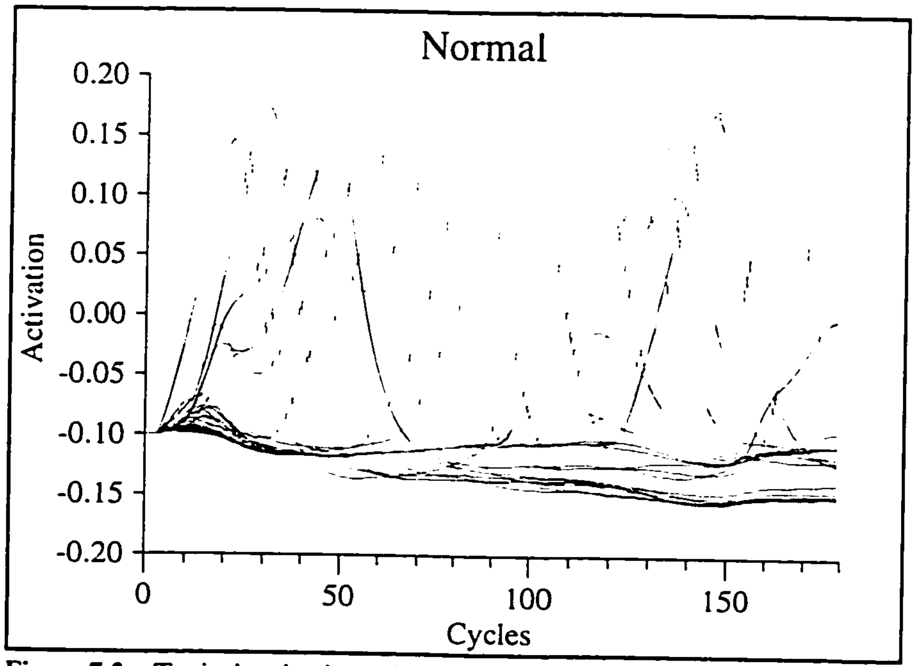

Figure 7.2 Typical activation values for a normal subject. Peaks represent an animal name being recalled.

animal names and the number of processing cycles required for recall for each of the ten normal subjects are reported in Appendix C. The first thing to note is that some of the subjects repeated animal names; however, no subject repeated a name more than once. When repeated names were removed from the recall list, subjects reported 13.4 (s.d. = 1.95) unique names on average.

Furthermore, the recalled list of names for each subject clearly exhibits clustering of animals. For example, the response pattern of Subject 7 starts with "dog" and then shows a rapid recall (10 processing cycles or less between names) of animals that generally fall into the categories large, North American, domestic, and herbivore. There is a slight pause (21 cycles) and then the subject generates animals that roughly fall into the categories large, African, wild, and herbivore. Note, also, that recall of names tends to slow down as the number of processing cycles increases. This type of response patterning is typical of actual human performance on the same task. (see Gruenewald & Lockhead, 1980).

The results from the ten normal subjects in this simulation are qualitatively similar to the responding of humans. The next step to take is to lesion the architecture to see if the damaged networks would produce similar results to AD patients. This is where connectionist models have the advantage over clinical studies. We can damage our normal networks and directly relate any behavioural deficits to the lesions. In other words, we can directly compare pre- and post-trauma behaviour.

AD Performance

There are many different approaches that researchers have taken to lesioning PDP networks: for example, adding noise to existing connection weights, cutting specific connections between processing units, and removing entire processing units from the network (for a review see Harley, 1993). For this experiment, lesions were created by reducing the connection weight by a random amount. This technique was adopted to mimic the reduction in dendritic arborization and neurotransmitter release (which causes a reduction in signal transmission) that is prevalent in AD (e.g., Kolb & Wishaw, 1996).

"Patient" networks were created by multiplying each connection weight by a random number between [0.0, 1.0], multiplying this number by 25%, 50%, 75%, or 100% to represent varying degrees of damage, and then subtracting the amount of damage from the original connection weight. Thus, damaged connection weights were driven towards zero, and weights that were positive remained positive while weights that were negative remained negative.

The results of lesioning subject 1—in terms of activation values—are shown in Figure 7.3 for networks with 25%, 50%, 75% or 100% damage while Table 7.1 reports the processing cycles and animal names recalled for the normal network and the damaged networks.

Table 7.1

Number of cycles required and names recalled as a function of damage

Normal		25%		50%		75%		100%	
Cycle	Name	Cycle	Name	Cycle	Name	Cycle	Name	Cycle	Name
23	horse	32	donkey	37	donkey	48	donkey	65	donkey
30	donkey	41	bull	63	buffalo	92	camel	126	camel
39	bull	52	buffalo	73	llama	125	donkey		
47	buffalo	63	horse	83	mule	158	camel		
57	cow	75	bull	87	bull				
64	zebra	80	zebra	103	cow				
72	lamb	84	mule	119	buffalo				
85	moose	95	llama	129	mule				
100	sheep	102	horse	150	cow				
117	cat	108	cow	167	buffalo				
131	giraffe	118	buffalo						
137	camel	136	mule						
145	rhino	141	horse						
147	elephant	151	llama						
164	bear	161	bull						
		168	cow						

As can be seen, the processing of the network with 25% damage appears to be more convoluted than the normal network (cf., Figure 7.2). Closer inspection of the plot, however, reveals that many of the peaks are repeats. In fact, Table 7.1 shows that of the 16 names recalled by the damaged network (one more than the normal network) half of the names are repeats, with some of the names being repeated a total of three times.

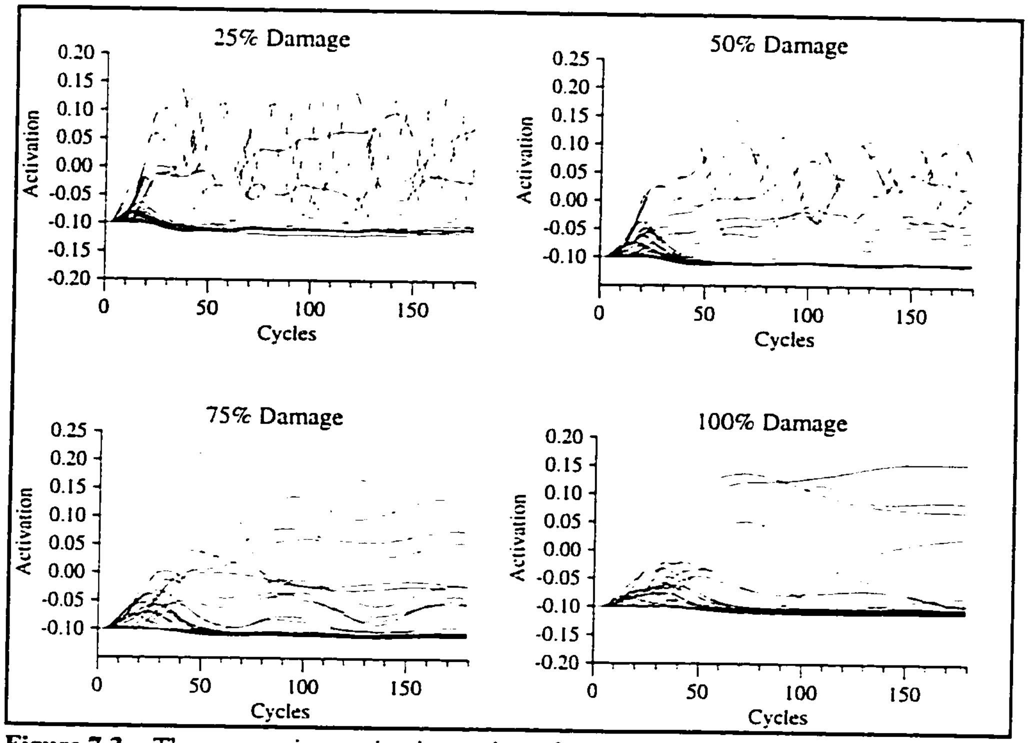

Figure 7.3 The processing activation values for the recall of names in networks damaged 25%, 50%, 75% and 100% for Subject 1.

The plots of the 50%, 75%, and 100% damaged networks indicate that increasing the amount of damage decreases the number of activation peaks, while increasing the amount of cyclic processing. In other words, fewer animal names are recalled, and those that are recalled are repeated more often. Consequently, for the remainder of the patient networks, a moderate amount of damage (i.e., 50%) was assumed.

Results from the patient networks are presented in Appendix D. Compared to normal networks, patient networks recalled on average significantly fewer animal names in terms of both total number of names recalled ($\bar{x}$ = 11.6, s.d. = 1.776; $t(18) = 3.129$, $p < .05$) and uniquely recalled names ($\bar{x}$ = 5.7, s.d. = 1.636; $t(18) = 8.134$, $p < .01$). Furthermore, although some semantic relationships appear to be preserved, when patient responses are compared to their normal counterpart, disruptions in the organization of semantic knowledge are clearly evident. For example, compare the patterning of responses for Subject 1 pre- and post-trauma. These results are qualitatively similar to the results given by AD patients when tested on the same task.

Conclusions and Implications

The results from these studies indicate that disturbances in the organization of semantic memory can be accounted for by a simple weakening of connections between processing elements instead of the active reorganization of semantic knowledge as proposed by Chan et al (1993). Consequently, the modified IAC network may be a step towards identifying the functional architecture of semantic memory. Not only can it qualitatively account for the behaviour of normal subjects, but it also accounts for the data of patients suffering from AD.

This implication about the functional architecture of semantic memory is further supported by an unpredicted result of the damaged networks' performance. Although normal networks sometimes recalled names that had previously been reported, damaged networks invariably repeated names already reported. In fact, with 50% damage, half of the total names recalled by the networks were repeated. Within the cognitive neuroscience literature, the inappropriate repetition of previous responses is known as perseveration (e.g., Plaut & Shallice, 1993).

The current network was not designed as a model of perseveration; however, perservation is a characteristic of AD (e.g., Miller, 1989; Girling & Berrios, 1990; Perry et al., 1996). As stated earlier, the power of theory is measured not only in terms of how well it accounts for the collected data, but also in terms of how well it predicts future data. Clearly, this theory of the functional architecture for semantic memory both accounts for previous data and has successfully predicted new data.

While these results show how semantic memory may be organized, the next experiment will address a fundamental assumption used within cognitive neuroscience when interpreting the behaviour of patients suffering from some form of brain trauma.

The Locality Assumption

One problem of considerable interest within cognitive neuroscience is the issue of assigning specific behavioral functions to specific brain regions; that is, the localization of function[1]. Initially, such assignments were based on patients who had suffered some form of brain lesion and

[1] This research has been reported in Medler, D. A., Dawson, M. R. W., Kingstone, A., & Panasiuk, E. (1998). The locality assumption revisited: The relation between internal structure and behavioral dissociations. *Under review*.

consequently manifested some behavioral deficit (e.g., damage to the posterior superior temporal lobe near the superior temporal gyrus and the resulting inability to understand language and speak in coherent sentences; a.k.a., Wernicke's aphasia). It is assumed, then, that the damaged brain region is critical for the behavior in question. Although recent technological advances in brain imaging techniques (e.g., fMRI, PET, SPECT, ERPs) have been able to shed more light on the problem, lesioning studies are still one of the major contributors to this area (e.g., Tranel, Damasio, & Damasio, 1995; Franz, Ivry, & Helmuth, 1996).

To aid in this endeavor, cognitive neuroscientists find it useful to distinguish between two qualitatively different types of behavioral deficits. A single dissociation consists of a patient performing task I extremely poorly and task II at a normal level, or at least very much better than task I. In contrast, a double dissociation occurs when one patient performs task I significantly poorer than task II, and another patient performs task II significantly poorer than task I (e.g., Shallice, 1988).

Cognitive neuroscientists have spent a great deal of time examining the logic underlying the kinds of valid inferences that can be drawn from dissociation data (e.g., Caramazza, 1986; Shallice, 1988). For example, Shallice argues that there is no evidence that a double dissociation can be observed as the result of two different lesions to a "properly distributed network" (p. 257); that is, a system that is not local cannot manifest a double dissociation. The corollary of this view is that "the existence of a neuropsychological double dissociation signifies that at least part of an overall system is functionally specialized" (p. 258).

The view that dissociation data can be used to conclude that internal structures are functionally specialized or local in nature is strongly related to what Farah (1994) calls the locality assumption. According to the locality assumption, when one component of the functional architecture is damaged, the effects of this damage will be exclusively local, with the nondamaged components of the architecture functioning normally in the absence of the damaged component: "the patient's behavior will therefore manifest the underlying impairment in a relatively direct and straightforward way" (Farah, 1994, p. 43).

Farah (1994) hypothesized that the locality assumption may be unwarranted for two reasons. First, its validity depends upon the additional assumption that the brain is organized into a set of functionally distinct modules, in the sense of Fodor (1983). Farah pointed out that whether or not the brain is modular is an unresolved empirical issue. Second, Farah noted that it is possible for nonlocal or distributed architectures, such as parallel distributed processing (PDP) networks, to produce single or double dissociations when lesioned. As the interactive nature of PDP networks is "directly incompatible with the locality assumption" (p. 46), the locality assumption may not be an indispensable tool for cognitive neuroscientists.

In support of her second claim, Farah (1994) reviewed three areas in which neuropsychological dissociations had been used previously to make inferences about the underlying architecture via the locality assumption. For each of these content areas (visual attention, semantic memory, and face recognition), Farah described an alternative, interactive architecture—a PDP network. In each of the networks reviewed, local damage produced (local) behavioral deficits analogous to the neuropsychological dissociations of interest. In two of these cases a single dissociation was produced, while the third produced a double dissociation (but see Butterworth, 1994 regarding this latter claim).

These results led Farah (1994) to conclude that one cannot infer that a specific behavioral deficit is associated with the loss of a local function. This is because of the prevailing view that PDP networks are, by definition, distributed and therefore nonlocal in structure. As a result, any local behavioral dysfunction observed in the network could not be due to damage to local internal structures, because no such structures are thought to exist within the network. Moreover, because PDP models putatively embody theories that are more parsimonious and more consistent with other information about brain structure and function, the strong conclusion to be drawn is that the locality assumption is unnecessary and false.

A potential fatal flaw to Farah's (1994) approach is that it is not correct to assume that PDP networks are, by definition, nonlocal. There are many examples in the literature of researchers who have trained a PDP network to perform a task of interest, have interpreted the internal structure of the trained network, and have found strong evidence of internal local structure (e.g., Berkeley, Dawson, Medler, Schopflocher, & Hornsby, 1995; Dawson & Medler, 1996; Elman, 1990, 1991; Hanson & Burr, 1990; Hinton, 1986; Sejnowski & Rosenberg, 1986). Indeed, Chapter 6 shows that networks can have very specific interpretations that support local internal structure. In most cases, this internal local structure manifested itself as a number of internal processing units that each had learned to respond to very specific stimulus features.

In light of these data, and the results reported in Chapter 6, it is premature to conclude that lesioning a PDP network produces local deficits without compromising local structure. One must first independently determine whether the internal structure of the lesioned network is local or not before drawing conclusions from the network's behavior.

The research described in the remainder of this chapter illustrates this more rigorous approach to using PDP networks to investigate the locality assumption. We trained a PDP network to solve a series of logic problems; following training, the network was lesioned. We then completed three separate analyses on the lesioned network. First, we assessed the qualitative behavioral changes in the network's output. Second, we analyzed the internal structure of the intact network using the banding technique described in Chapter 6 to determine precisely what role each hidden unit had in the performance of the network. This enables us to relate observed behavioral deficits with any functional specificity of the ablated structures. Third, we analyzed the lesioned network by correlating the number of errors with damage to local internal structure.

It is important to note, at the outset, that one might argue that this problem domain, and hence this experiment, is of little importance to cognitive neuroscience because there is no reported relation between brain lesions and performance on logic tasks. Not only is this position incorrect (e.g., see Damasio, 1994, p. 18; McCloskey, 1993; Robertson, Knight, Rafal, & Shimamura, 1993), but it also misses the crucial point that the present study is designed to test the rationale underlying Farah (1994); that behavioral dissociations produced by local lesions within a PDP network result from ablating local structure within that network. Thus, the choice of problem domain is not particularly relevant to the issue at hand and to the conclusions to be drawn.

Method

The Logic Problem

Bullinaria (1994) has argued that in order for researchers to inform cognitive neuroscience by lesioning PDP networks, several properties should be true of these models. For example, the model should be trained (instead of handwired), and the model should learn to solve complex problems. One instance of such a problem was originally proposed by Bechtel and Abrahamsen (1991), and requires a network to identify four different types of logical arguments, and to judge the validity of the arguments. This problem is both psychologically relevant and composed of a large enough set of stimulus patterns to make it challenging. Furthermore, interpretations of networks trained on this problem have produced results with important theoretical implications (Dawson, Medler & Berkeley, 1997).

Problem Definition. Bechtel and Abrahamsen's (1991) original stimulus set was used to train the network of value units. Each pattern in the training set was a logical argument consisting of two sentences and a conclusion. The first sentence was composed of a connective and two variables; the second sentence and the conclusion were each composed of a single variable.

The problem set consisted of four classes of problem: *modus ponens* (MP), *modus tollens* (MT), *alternative syllogism* (AS), and *disjunctive syllogism* (DS). There were two different versions of each AS and DS problem type. Table 7.2 illustrates examples of valid arguments for each problem type, and also introduces the descriptive notation adopted to aid network interpretation.

Table 7.2

Examples of valid and invalid instances of each of the argument types for the logic problem.

PROBLEM TYPE	EXAMPLE VALID PROBLEM	DESCRIPTIVE NOTATION FOR VALID PROBLEM	EXAMPLE INVALID PROBLEM	DESCRIPTIVE NOTATION FOR INVALID PROBLEM
Modus Ponens (MP)	If A then B A ——— Therefore B	Connective: IF..THEN S1(V1) = A, S1(V2) = B S2 = A CONCLUSION = B	If A then B B ——— Therefore A	Connective: IF..THEN S1(V1) = A, S1(V2) = B S2 = B CONCLUSION = A
Modus Tollens (MT)	If A then C Not C ——— Therefore not A	Connective: IF..THEN S1(V1) = A, S1(V2) = C S2 = C, S2 is negated CONCLUSION = A, conclusion is negated	If A then C Not A ——— Therefore not C	Connective: IF..THEN S1(V1) = A, S1(V2) = C S2 = A, S2 is negated CONCLUSION = C, conclusion is negated
Alternative Syllogism (AS) Type I	D or A Not D ——— Therefore A	Connective: OR S1(V1) = D, S1(V2) = A S2 = D, S2 is negated CONCLUSION = A	D or A D ——— Therefore not A	Connective: OR S1(V1) = D, S1(V2) = A S2 = D CONCLUSION = A, conclusion is negated
Alternative Syllogism (AS) Type II	B or C Not C ——— Therefore B	Connective: OR S1(V1) = B, S1(V2) = C S2 = C, S2 is negated CONCLUSION = B	B or C C ——— Therefore not B	Connective: OR S1(V1) = B, S1(V2) = C S2 = C CONCLUSION = B, conclusion is negated
Disjunctive Syllogism (DS) Type I	Not both C and D C ——— Therefore not D	Connective: NOT BOTH...AND S1(V1) = D, S1(V2) = C S2 = C CONCLUSION = D, conclusion is negated	Not both C and D Not C ——— Therefore D	Connective: NOT BOTH...AND S1(V1) = D, S1(V2) = C S2 = C, S2 is negated CONCLUSION = D
Disjunctive Syllogism (DS) Type II	Not both A and D D ——— Therefore not A	Connective: NOT BOTH...AND S1(V1) = A, S1(V2) = D S2 = D CONCLUSION = A, conclusion is negated	Not both A and D Not D ——— Therefore A	Connective: NOT BOTH...AND S1(V1) = A, S1(V2) = D S2 = D, S2 is negated CONCLUSION = A

Note. In the training set, valid MP and MT arguments were turned into invalid arguments by interchanging S2 and the conclusion. For the remaining problem types, valid arguments were turned into invalid ones by interchanging the sign of S2 and the conclusion.

Each argument was represented as a pattern of on/off activity in a set of 14 input units (see Figure 7.4) using the representational scheme adopted by Bechtel and Abrahamsen (1991). Different examples of each argument type were constructed by selecting two variables from a set of four letters (A, B, C, D). A variable letter could also be negated (e.g., Not A). For each type of argument, 48 different valid instances (the conclusion follows from the two sentences) and 48 different invalid instances (the conclusion does not follow from the two sentences) were used, creating a total training set of 576 patterns.

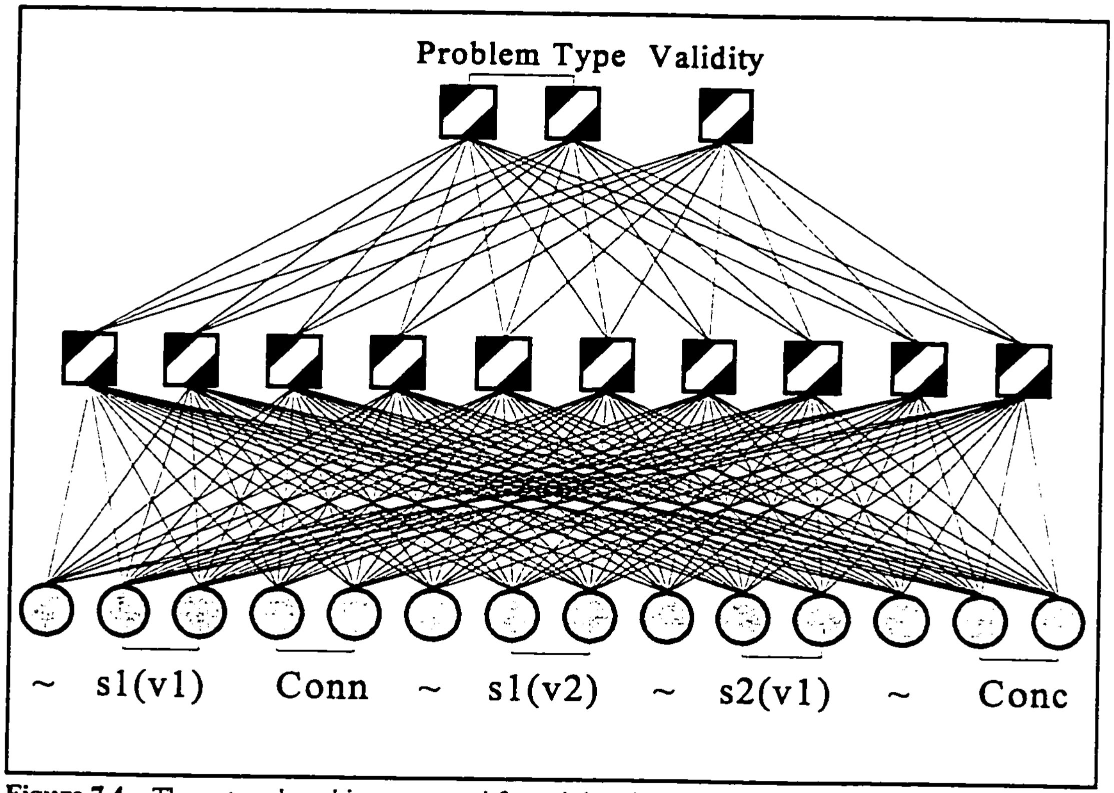

Figure 7.4 The network architecture used for solving the Logic Problem. Note that ~ ≡ Not, s1(v1) ≡ Sentence 1 (Variable 1), etc., Conn ≡ connective, Conc ≡ conclusion.

Network Architecture. A network of value units with 14 input units was trained on the problem set described above. The network had three output units. Two of the output units were used to represent one of four argument types (AS, DS, MP, MT); the third was used to indicate argument validity (see Figure 7.4). In contrast to Bechtel and Abrahamsen's (1991) original network which used two layers of 10 hidden integration devices, the value unit network had a single layer of 10 hidden units (in general, because of the nonmonotonic nature of the value unit's activation function, fewer hidden units are required to solve problems than are required by the standard architecture). Pilot studies established that a network with 10 hidden value units would reliably converge to a solution for the logic problem, while networks with smaller numbers of hidden units would converge on a solution, but would not do so reliably .

The network of value units was trained with the Dawson and Schopflocher (1992) modification of the GDR, using a learning rate of $\alpha = 0.01$ and a momentum of $\eta = 0.0$. Network weights were randomly set in the range [-0.01, 0.01]. Network biases were initialized to 0 and then modified during training. Network connections were updated after every pattern presentation, and pattern presentation was randomized every sweep. The network was trained until a "hit" was recorded for every output unit for every pattern in the training set. A hit was defined as being an activation of 0.9 or greater when the desired output was 1, and as being an activation of 0.1 or less when the desired output was 0. Convergence (i.e., a hit on every pattern) was achieved in 1362 sweeps.

Lesioning the Logic Network

As mentioned earlier, there are several different approaches to lesioning PDP networks. Whereas in the previous experiment the connection weights were randomly decreased (i.e., noise was added to the weights), in this experiment, entire processing units were removed from the network. Specifically, in these simulations we took our intact PDP network and selected a single processing unit to be ablated from it. This was accomplished by forcing the ablated units to always generate an internal activation that was equal to zero, regardless of what stimulus was being presented to the network. In other words, we lesioned the intact network by turning processing units off.

There were two major reasons for our decision to adopt this particular method of lesioning networks. First, our approach to interpreting the internal structure of PDP networks focuses upon the functional role of individual processing units, a point that will be detailed later. By restricting ourselves to the removal of entire processing units from the intact network, we placed ourselves to take maximum advantage of our knowledge about the internal structure when interpreting behavioral deficits (i.e., qualitative changes in network outputs). Second, the ablation of individual processing units provides a simple and useful context for asking questions about the locality assumption.

The intact network of value units trained on the logic problem had 10 hidden units. Consequently, we produced 10 different "patients" by ablating each of these hidden units in turn (i.e., selecting one of the hidden units, and preventing it from adopting an activation value different from zero). In other words, each lesioned network, or patient, was missing one hidden processing unit, and each lesioned network differed from all others in terms of which hidden unit had been destroyed.

In order to determine the effect of each lesion, we re-presented all 576 logic problems to each lesioned network, and recorded the resulting behavior as determined by the activity produced in the three output units. Our primary interest was in determining whether removing a single hidden unit produced qualitative changes in network behavior. A qualitative change is defined as a change in the classification of problem type (e.g., misclassify an MP as an AS) and/or a change in classification of problem validity (e.g., misclassify a valid problem as invalid). In order to examine such qualitative changes in network behavior, we applied a thresholding rule to convert output activations in the lesioned networks from continuous values to binary values. Specifically, if an output unit's activity was 0.5 or greater, we converted it into a 1; otherwise, the activity was converted into a 0[2]. By converting output activations from continuous into binary values, we were able to interpret network outputs by applying the binary encoding scheme that Bechtel and Abrahamsen (1991) used to define desired network outputs (see also Figure 7.4).

Analyzing the Lesioned Networks

Three different analyses are performed on the lesioned networks. First, we adopt the standard approach of analyzing the qualitative change in the networks' behavior as determined by differences between the desired and actual outputs. Second, we perform a "banding analysis" on the internal structure of each ablated hidden unit to independently assess if there is local structure within the network. Third, we correlate the number of errors produced by the lesioned networks as determined in the first analysis with the local internal structure of the hidden units as revealed by the second analysis. Taken together, these three analyses show that local behavioral deficits in the lesioned logic network are due to the ablation of local internal structure.

Analysis I: Qualitative Changes in Network Behavior

In this first analysis, we assess the qualitative changes in network behavior by comparing the actual performance of a lesioned network with the desired network performance; in effect, comparing a "patient" with a normal. This is the approach taken by Farah (1994). Our analysis will

[2] It could be argued that this thresholding rule is cheating as the network was originally trained to producing values between [0.0, 0.1] and [0.9, 1.0]. Thresholding the output, however, is no different than having a subject participate in a forced-choice paradigm.

show that local damage can produce specific local behavioral impairments, ranging from misclassifying almost half of the problem set to no change in behavior. Additionally, two of our lesioned networks will reveal the presence of a double dissociation, providing converging evidence for the conclusion that local behavioral impairments are due to damage to local internal structure (cf., Shallice, 1988).

General Network Behavior

The Bechtel and Abrahamsen (1991) training set for the logic problem can be described as being comprised of many examples of eight different basic problem types: there are 96 invalid alternative syllogisms (ASI), 96 valid alternative syllogisms (ASV), 96 invalid disjunctive syllogisms (DSI), 96 valid disjunctive syllogisms (DSV), 48 invalid modus ponens (MPI), 48 valid modus ponens (MPV), 48 invalid modus tollens (MTI), and 48 valid modus tollens (MTV). By applying the thresholding rule to the output units of the lesioned network, any response of a lesioned network must also fall into one of these eight different problem types.

Because both the input and output of a lesioned network are restricted to fall into one of eight different categories, it is useful to represent network responses in terms of a confusion matrix such as the three matrices illustrated in Tables 7.3, 7.4, and 7.5. Each row of a confusion matrix for the logic problem represents the type of problem presented as input to a network. Each column of this matrix represents the type of problem that was indicated as a response by the network. The number in each cell of this matrix indicates the frequency with which a particular input problem produced a particular network response. For example, in Table 7.3, the number 96 in the DSI row and the DSI column indicates that on 96 different occasions, a DSI problem was correctly classified as a DSI problem by one of the networks. Similarly, in Table 7.4, the number 6 in the DSI row and the ASV column indicates that on 6 different occasions a DSI problem was incorrectly classified as an ASV problem by one of the networks.

In order to determine whether removing a single processing unit from the intact network produced local deficits, we examined each of the 10 confusion matrices for the lesioned networks. First, we simply counted the number of off-diagonal cells that were not equal to zero; this measurement indicates the number of different types of errors produced by the network. The maximum number of error types that could be made for any hidden unit was 56. The average of this measure was 5.4; it ranged from 0 (when hidden unit 4 was ablated) to 10 (when hidden unit 6 was ablated). This small number indicates that the effect of any single processor lesion is to produce local behavioral deficits as opposed to global behavioral deficits (all 56 error types).

Table 7.3

Confusion Matrix for Hidden Unit 3

	ASI	ASV	DSI	DSV	MPI	MPV	MTI	MTV
ASI	96							
ASV		96						
DSI			96					
DSV				96				
MPI			46		0		2	
MPV				47		1		
MTI	6		38		4		0	
MTV				48				0

Table 7.4

Confusion Matrix for Hidden Unit 7

	ASI	ASV	DSI	DSV	MPI	MPV	MTI	MTV
ASI	96							
ASV		96						
DSI		6	1	11	12	12	35	19
DSV			17	0	67		12	
MPI					48			
MPV						48		
MTI							48	
MTV								48

Table 7.5

Confusion Matrix for Hidden Unit 4

	ASI	ASV	DSI	DSV	MPI	MPV	MTI	MTV
ASI	96							
ASV		96						
DSI			96					
DSV				96				
MPI					48			
MPV						48		
MTI							48	
MTV								48

We also measured the sum of the values within the off-diagonal cells for each of the 10 confusion matrices, which indicated the total number of mistakes made by the network (as opposed to the number of mistake types). On average, 128 errors were produced from a single lesion, accounting for a large 22.2% error rate. The total number of errors ranged from 0 (when hidden unit 4 was ablated) to 210 (when hidden unit 6 was ablated).

In addition to these general statistical descriptions of the type and number of errors observed when hidden units are ablated, the confusion matrices can be used to provide a detailed qualitative account of the effect of a particular lesion. A quick analysis of the confusion matrices revealed that nine of the ten confusion matrices showed local damage producing very local behavioral deficits, while the remaining confusion matrix showed no behavioral deficit at all! This cursory examination of the confusion matrices will be expanded in Analysis III. For now, however, we will focus on two confusion matrices (Hidden Units 3 and 7) that show a double dissociation, and the one confusion matrix (Hidden Unit 4) that shows no behavioral deficits.

A Double Dissociation

Table 7.3 presents the confusion matrix that was produced after ablation of Hidden Unit 3 (HU3) from the intact network. This matrix reveals that the network is performing perfectly on all AS and DS problems. However, it fails to correctly classify either valid or invalid MP and MT problems. Of the 192 MPI, MPV, MTI, and MTV problems, the network classified one problem correctly.

In contrast, Table 7.4 is the confusion matrix produced after removing Hidden Unit 7 (HU7) from the intact network. This matrix reveals that the network is performing perfectly on all AS, MP, and MT problems. However, its ability to correctly classify either valid or invalid DS problems disappears, with only one out of the 192 DSI and DSV problems being correctly classified.

Upon closer inspection, it can be seen that the confusion matrices in Tables 7.3 and 7.4 represent a double dissociation—they suggest that the two hidden units involved in these lesions provide the network with the ability to process two different types of logical connectives. Recall from Table 7.2 that all of the different logic problems that are presented to the network are constructed from only three different connectives. All ASI and ASV problems are based on the connective "Or", all DSI and DSV problems are based on the connective "Not both...and", and all MPI, MPV, MTI and MTV problems are based on the connective "If...then". With this in mind, a natural interpretation of Table 7.4 is that HU7 is a "Not both ... and" detector, which when ablated produces dramatic deficits in the ability to classify DS problems, but has no effect on classification of AS, MP, and MT problems. Similarly, for Table 7.3, it is natural to infer that HU3 is an "If .. then" detector, which when ablated prevents the network from responding to MP and MT problems but does not affect responding to AS and DS problems.

<u>*A Null Lesion*</u>

The confusion matrices in Tables 7.3 and 7.4 indicate that some of the lesions to the intact network produce highly specific dysfunctions in network performance. However, not all lesions to the network produce such effects. For example, Table 7.5 is the confusion matrix obtained from the network after ablation of Hidden Unit 4 (HU4). As can be seen from this confusion matrix, there was no qualitative change in network performance.

Why does a lesion to HU4 produce no appreciable affect on network behavior, while lesions to HU3 and HU7 produce a double dissociation? In order to answer this question, we must turn to an independent analysis of the role played by each of these hidden units in the network.

Analysis II: The Interpretation of Internal Structure

In Analysis II, we perform a "banding analysis" on the internal structure of each ablated hidden unit to assess independently if local structure exists within the logic network. This is a crucial step if one wants to make conclusions about the locality assumption based on the performance of lesioned PDP networks. Our analysis will show that the hidden units associated with producing the double dissociation in the previous analysis also have highly interpretable internal local structure. Conversely, the hidden unit that produced no change in network performance when ablated shows little internal structure. Thus, our analysis shows that changes in network performance are due to the ablation of local internal structure; therefore, we cannot dismiss the locality assumption.

Banding Analysis

In our use of this graphing technique, each hidden unit is represented with its own jittered density plot. Each dot within a unit's density plot represents the activity produced in that unit by one of the patterns that were presented to the network. For example, the three density plots illustrated in Figure 7.5 represent three of the hidden units in the intact logic problem network, and as a result each is composed of 576 different dots (one for each of the 576 logic problems). Refer to Chapter 6 for a complete description of the banding analysis technique.

The Internal Structure of Individual Units within the Logic Network

To illustrate the utility of this approach, let us consider the double dissociation described previously. When HU7 is lesioned, the network loses the ability to process DS problems, which are the only problems defined with the "Not both...and" connective. With this in mind, it is natural to conclude that HU7 is a "Not both...and" detector.

The density plot in Figure 7.5a was obtained for HU7 in the intact network. This density plot is divided into two distinct bands, one of which falls at zero activity, the other of which falls at very high activity. When this unit generates high activity, it is signaling that a pattern that falls into this high activity band has been detected. This band contains 192 patterns, all of which share a single property—they all contain the "Not both...and" connective. Hence, the banding analysis confirmed the behavioral inference (from the confusion matrix) that HU7 is a "Not both...and" detector.

Let us now consider HU3. From the confusion matrix in Table 7.3, it is natural to infer that HU3 is an "If...then" detector, which, when ablated, prevents the network from correctly classifying MP and MT problems, but which does not affect its ability to classify AS and DS problems.

The density plot of Figure 7.5b density plot was obtained for HU3 in the intact network. It is divided into two distinct bands, one of which falls at low activity, the other of which falls at activities greater than 0.6. When this unit generates high activity, it is signaling that a pattern that falls into this high activity band has been detected. Surprisingly—given the behavioral evidence from the lesioned network—the 384 patterns in the high activity band not only contain problems constructed with the "If...then" connective, but also patterns with the "Not both...and" connective.

In other words, the only feature shared by all of the patterns in this high activity band is that they do not possess the "Or" connective that defines AS problems. The banding analysis of HU3 reveals that it is not simply an "If...then" detector. Note, however, that this does not compromise the double dissociation that was revealed earlier by the behavioral analysis of the lesioned networks.

Finally, let us turn to a consideration of HU4. As was shown in Table 7.5, when this

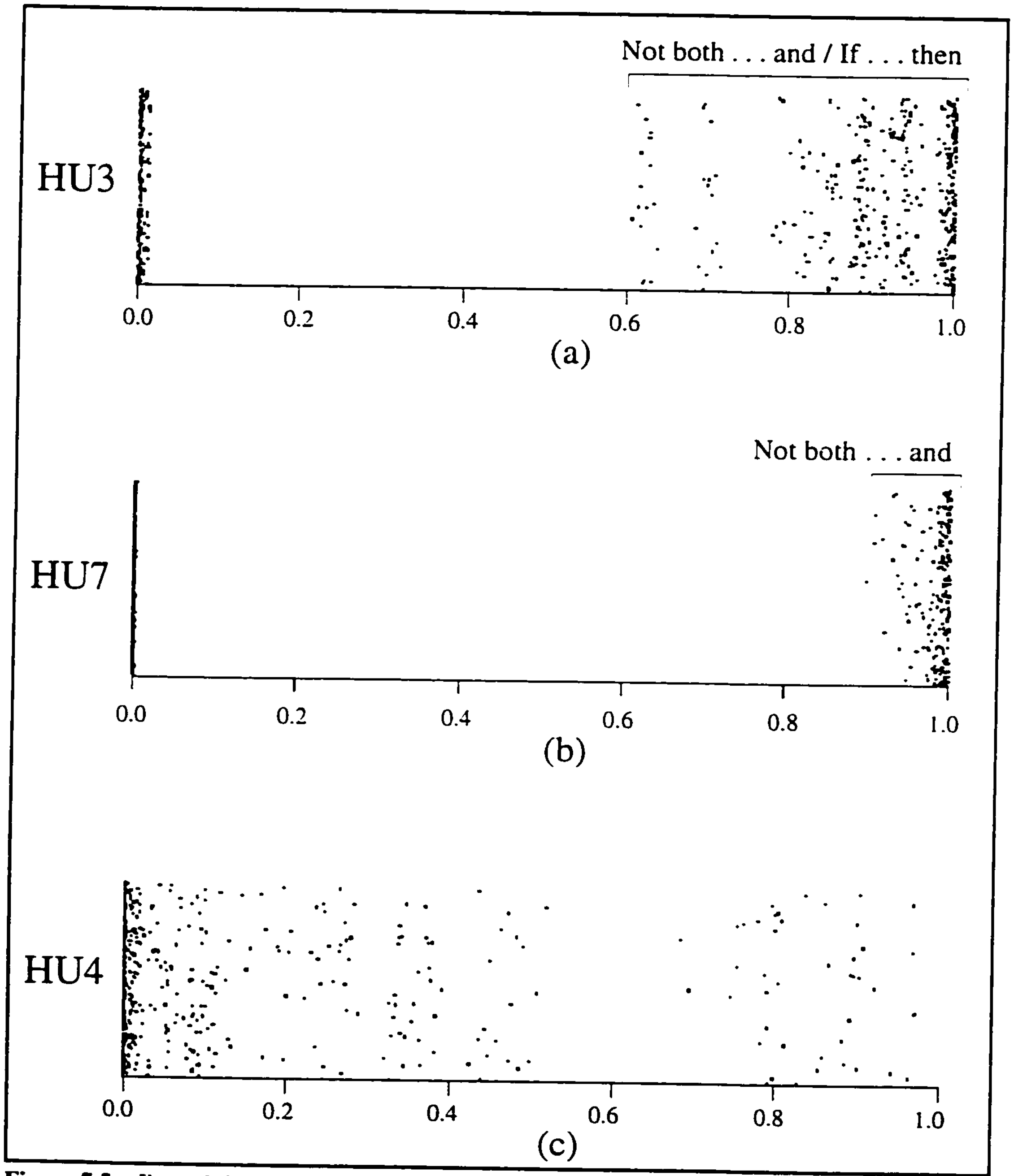

Figure 7.5 Jittered density plots for (a) Hidden Unit 3, (b) Hidden Unit 7, and (c) Hidden Unit 4. Note the (a) and (b) have a relatively high degree of banding whereas (c) shows relatively little banding.

processor was ablated, network performance was not appreciably affected. Figure 7.5c shows the jittered density plot for HU4. This density plot is quite different in appearance from those presented in Figure 7.5a and 7.5b. In Figure 7.5c, 6,532 of the patterns fall in a broad smear that ranges from an activation of 0.0 to approximately 0.5. A second smeared band, containing the remaining 44 patterns, ranges between activation values of approximately 0.65 to 1.0. These latter 44 problems are associated with the interpretation "S1(V1) = S2 and S1(V2) = CONCLUSION", which can be found in examples of all of the problem types, and thus cannot be used to distinguish one type of problem from another.

Summary and Implications

Tables 7.3, 7.4, and 7.5 and Figure 7.5 provide qualitative evidence that is at odds with Farah's (1994) use of PDP networks to test the locality assumption. These figures show that when highly specific local deficits in network behavior are observed (Tables 7.3 and 7.4), these deficits are associated with the loss of hidden units that have highly local function within the network. This local function is revealed in the banding analysis of these units (Figures 7.5a and 7.5b)—their local nature is reflected in strikingly banded density plots, as well as very specific interpretations that emerge from these density plots. In contrast, when no local deficits in network behavior are observed (Table 7.5), this is associated with the loss of a hidden unit that has no local function within the network—its nonlocal nature is reflected in a density plot that has little banding (Figure 7.5c), and which does not lend itself to a specific interpretation.

These results are consistent with Shallice's (1988) assessment of double dissociations: namely, that an observation of a double dissociation in the behaviours of lesioned networks corresponded with highly localized internal structure as assessed by independent means. It is interesting to note, however, that the internal structure of the network as indicated by the "banding" analysis is not the same as the local structure implied by the behavioural performance of the network. This has direct implication for the modularity assumption that Farah (1994) assumes within her definition of locality.

Farah (1994) argued that the locality assumption is a consequence of the view that the architecture for cognition is organized into independent modules (e.g., Fodor, 1983). Each of these modules are assumed to be designed to solve specific information processing problems, and are therefore informationally encapsulated. This means that each module has available to it only the information that is required to solve the problem at hand. Because of this, different modules need not interact with one another during processing. Interaction is limited instead to when processing by one module is finished, and the result of this processing is passed along as input to another module. "Even these interactions are limited, so that a given component receives input from relatively few (perhaps just one) of the other components" (Farah, 1994, p. 43).

Hence, the behaviour of the lesioned networks would imply a module for detecting "Not both...and" connectives and another module for detecting "If...then" connectives. The analysis of the internal structure, however, indicates that there are not functionally separate modules for computing these connectives. In fact, the architecture revealed by the analysis is similar to Shallice's (1988) description of overlapping processing regions that are local and yet not modular.

For example, taken at a compuational level description, it appears that the role of Hidden Units 3 and 7 is connective detection (as illustrated by the double dissociation). It is possible that Hidden Units 3 and 7 could be acting as a module in the Fodorian sense. When both units are lesioned, however, the network is still able to respond, indicating that the processing units function as parts of a greater whole

Analysis III: Analyzing Quantitative Relationships

The previous two analyses provide qualitative evidence suggesting that dissociations in network behaviors result when local structure is ablated from a PDP model. These results, however, are based on three of the ten "patients" that were created; therefore, they are only suggestive. Further empirical support is required.

The purpose of our third analysis is to quantitatively analyze the relationship between deficits in network performance caused by lesioning a hidden unit and an objective measure of that unit's local structure (i.e., the bandedness or specificity of its jittered density plot). Results will show that the deficit in network performance following a lesion is highly correlated with the local internal structure of the ablated hidden unit. Large behavioral deficits are associated with the ablation of units that exhibit a high degree of functional locality.

Defining a Quantitative Measure of Locality

Factor analysis provides an elegant way to quantify local structure in density plots. In mathematical terms, factor analysis is a technique for compressing a data matrix by expressing it as a smaller number of factors and factor weights. In practice, researchers who use factor analysis are concerned with finding a set of factors whose loadings are maximally interpretable; Thurstone (1932) states that one way to accomplish this is to rotate a factor structure until its simple structure is optimized. Simple structure is characterized by a number of data points having high factor loadings-that is, they posses the "feature" that the factor captures—while the remaining data points have zero factor loadings because they do not possess the "feature" captured by the factor. The degree of simple structure can be analytically described by the amount of variance within the factor loadings (Kaiser, 1958); factor loadings with high degrees of simple structure will have more variance than factor loadings with little simple structure.

If the jittered density plots described earlier are viewed as being analogous to factors, then it becomes apparent that density plots that are easily interpreted can be described as having simple structure. Specifically, the two density plots in Figures 7.5a and 7.5b demonstrate a high degree of simple structure (note the clustering of data points at either end of the activation axis) while the density plot in Figure 7.5c shows very little clustering of data points and therefore little simple structure. Moreover, we can quantify this simple structure by calculating the variance of the activation values within each hidden unit.

The qualitative claim that was suggested by the previous study was that the amount of behavioral deficit produced by a lesion was proportional to the (functional) locality of the lesioned unit. If a hidden unit had a highly specific function, then its ablation would produce more dramatic dissociations—as measured by network error—than if it served a much less specific function. By employing Kaiser's (1958) use of variance to define simple structure, we are now in a position to investigate this claim in a much more objective and quantitative fashion by examining the correlation between network errors and the variance of density plots associated with ablated hidden units.

Local Structure, Lesions, and the Logic Network

As was described earlier, one quantitative measure of network error is the number of different types of errors produced when a unit is lesioned (i.e., the number of nonzero off-diagonal cells in the confusion matrix). Figure 7.6(a) provides the scatterplot between this error measure and the variance of the density plot of the ablated unit. A strong positive linear relationship is revealed. Indeed, the correlation between these two measures is 0.930. A second measure of network error is the total number of errors generated (i.e., the sum of the off-diagonal cells in the confusion matrix). Figure 7.6(b) illustrates the scatterplot between this error measure and density plot variance, revealing another strong positive linear relationship. The correlation between these two measures is 0.929.

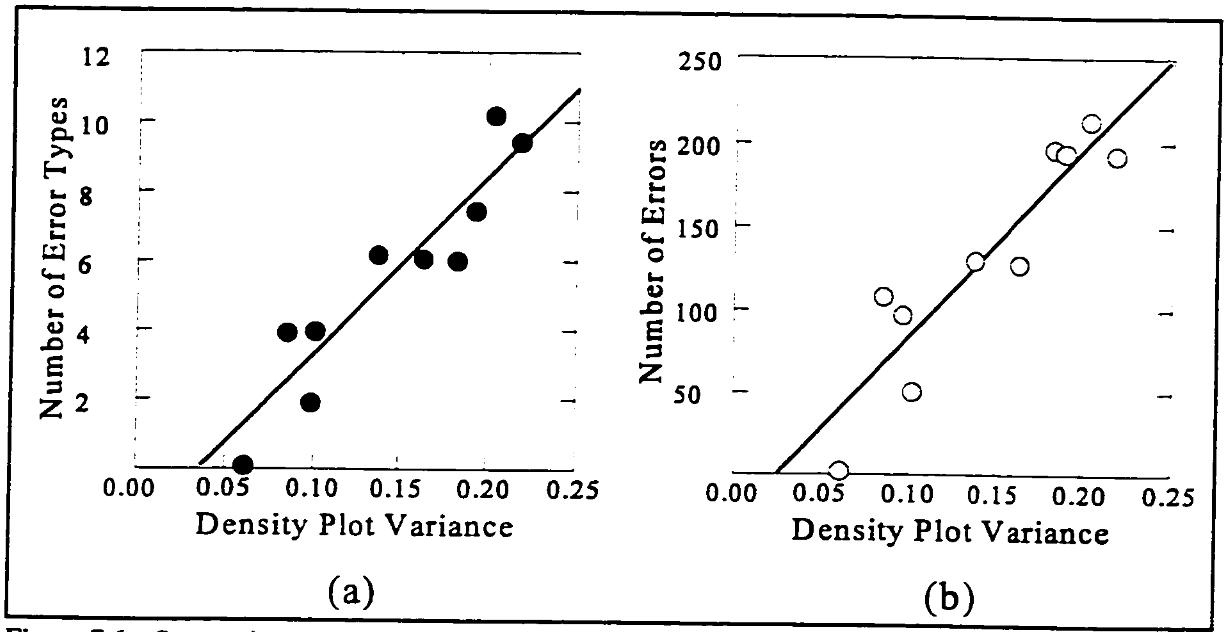

Figure 7.6 Scatterplots correlating (a) the type of errors, and (b) the total number of errors made by the lesioned networks with a measure of the simple structure of the ablated units.

<u>*Summary*</u>

Following Kaiser (1958), the variance of a density plot can be used as a measure of its simple structure. There is a very strong linear relationship between density plot variance and network errors (specifically, the number of types of errors, and the total number of errors). This relationship indicates that when hidden units that have high-variance density plots are ablated, the result is a much higher degree of network errors than when hidden units that have low-variance density plots are ablated. This provides quantitative support for the claim that in the logic network, large behavioral deficits are associated with the removal of units that exhibit a high degree of functional locality.

General Discussion

The results from the preceding study suggest not only that PDP networks can have a highly local structure, but the more local this structure is, the greater the effect of its ablation. This result disconfirms the view that models consistent with generic connectionism necessarily represent a nonlocal architecture (cf., Farah, 1994); and, it supports the view that when strongly local behavioral deficits are observed, these are likely due to injuries to localized internal functions (cf., Shallice, 1988).

Relationship To Other Results

One question that immediately arises is whether the conclusions drawn from our simulation results are atypical. It is possible that our results are idiosyncratic, and that local behavioral deficits in PDP networks are generally due to injury to nonlocal structures, as is assumed by Farah (1994). However, even a brief survey of the literature concerning lesions to connectionist networks reveals many additional examples of local dissociations in network behaviors that appear to be due to disruptions in very specific network functions:

(1) Mozer and Behrmann (1990) described a connectionist model of spatial attention that consisted of several distinct layers of units devoted to specific functional tasks (i.e., a network for building locationally invariant representations of letters and words, a network that serves as an attentional mechanism, and a network for cleaning up the interpretation constructed by other functional components). Mozer and Behrmann modeled neglect dyslexia by performing a local lesion to this network—damaging the connections between the network's input feature maps and the attentional mechanism.

(2) Humphreys, Freeman, and Muller (1992) described a connectionist model of visual search that consisted of many layers of processors, with early layers corresponding to a retinal array and maps of single features, intermediate layers representing combined feature maps, and late layers instantiating decision mechanisms for detecting targets. In one set of simulations, they found that ablating units in early layers produced the equivalent of a blind spot for the network. They also found that ablating units in later layers disrupted search for targets in homogeneous arrays of distractors, but had little effect on search for targets in heterogeneous arrays of distractors. This kind of result mirrored the performance of a visual agnosic patient described by Humphreys, Riddoch, Quinlan, Price and Donnelly (1992). The deficits exhibited by both the network and the patient were attributed to damage to a particular local function in the system—the grouping between simple form disjunctions.

(3) Variations of a particular connectionist network have been used to study the relationships between network damage and different forms of dyslexia (e.g., Hinton, Plaut & Shallice, 1993; Hinton & Shallice, 1991; Plaut & Shallice, 1993; Plaut, 1995). In general, each of these networks consisted of an architecture in which each layer of processing units served a particular functional role. For example, one layer of units represented input features, a second layer of networks represented higher-order associative features, a third layer of units represented network output (typically in terms of semantic features), and a fourth set of units had the function of

"cleaning up" the activity in the semantic units. (This architecture can be chained together to create a more complex network, as illustrated by Hinton, Plaut, and Shallice, 1993, p. 82). Lesions to connections between different layers of processors lead to qualitatively different kinds of errors (e.g., disruption of the naming of abstract words, but not of concrete words) because of damage to the interactions between specific local functions in the network (e.g., Plaut & Shallice, 1993, Figure 2, Figure 6).

(4) Indeed, a closer examination of the networks reported by Farah (1994) suggest that the local behavioral deficits that she observed were in fact caused by the ablation of local structure. For example, the face recognition network was constructed so that some units only process "face" information, while other units only process "name" information. Similarly, the visual attention network described by Farah includes sets of units that are devoted to processing information coming from specific spatial locations. Finally, the semantic network has some units devoted to functional properties, and other units devoted to visual properties.

The Possibility Of Nonlocal Lesions Is Still Present

Of course, the evidence to date is not sufficient to make the strong claim that when any local behavioral deficits are observed in a network, this is because of an injury to local internal structure. There exists the possibility that lesions to highly distributed networks can produce local behavioral deficits.

For example, Wood (1978) performed lesions to an associative memory model that is highly distributed in nature, the "brainstate-in-a-box" classifier originally described by Anderson, Silverstein, Ritz, and Jones (1977). In general, this kind of model is used to remember different "category names", which are represented as patterns of activity in a set of processing units. Furthermore, this kind of model operates best when each to-be-learned category name is uncorrelated with all of the others. However, Wood was able to show that in cases where there were high degrees of correlation between distinct category names (i.e., two names that differ only with respect to values encoded in 2 of 8 processing units), lesioning these two critical processors produced a marked disruption for the two similar category names, but had much less effect on other (uncorrelated) names that the network had also learned.

Unfortunately, in many cases it is impossible to determine from research reports whether local deficits in the performance of lesioned networks is due to the ablation of local, internal structure. This is because researchers typically neglect to interpret the internal structure of their PDP networks in order to determine the functional role of the units that they remove.

Summary and Conclusion

The localization of functions within the brain is a dominant problem within cognitive neuroscience. One key approach is to identify a specific cognitive or behavioral deficit following a brain lesion, and to speculate whether that specific brain region is essential for the computation in question. This area of research, however, is highly dependent on the locality assumption. Farah (1994) rejected the locality assumption based on the performance of PDP networks that had been lesioned. This rejection was based on the assumption that PDP networks are by definition nonlocal.

Our starting point was that the assumption that PDP networks are nonlocal and that lesions to nonlocal structures produce specific deficits are two assumptions that need to be assessed independently. We revisited the locality assumption using the performance of a lesioned PDP network. It was found that local lesions produce very local and severe impairments; that is, a single lesion to the network produces an average of 5 (out of 56) different error types, yet produces an average error rate of 22.2%.

A qualitative assessment of individual units found that one unit (HU4) produced no qualitative change in behavior, while two others (HU3 and HU7) produced a double dissociation. From a purely behavioral assessment, it would appear that HU4 possessed no local structure, while HU3 and HU7 possessed highly local structure; specifically, for detecting the connectives "If ... then" and "Not both ... and" respectively.

To confirm the suggestion that damage to local structure produces local deficits, the internal structure of the intact network was analyzed. The "banding" analyses of the internal structure of the intact network confirmed that HU4 had very little local structure, and that HU7 had highly local structure consistent with detecting the connective "Not both ... and." The banding analysis of HU3 also showed that it had highly local structure consistent with detecting connectives; however, as opposed to the behavioral analysis, the internal analysis showed that it was not simply an "If .. then" detector—it also detected the connective "Not both ... and." Furthermore, quantitative analyses showed that lesioning hidden units with a high degree of local structure—as measured by the variance of the banding plots—produced more errors in network responding than lesioning units with very little local structure.

These results indicate that the locality assumption as defined by Farah (1994) cannot be rejected simply based on the overt behavior of PDP networks. Moreover, these results are consistent with Shallice's (1988) assessment of double dissociations; namely, that an observation of a double dissociation corresponds with highly localized internal structure as assessed by independent means.

More importantly, however, these results have a strong implication for cognitive neuroscientists studying the localization of function via behavioral and lesion data. In our study, we found that the internal local structure of the network as indicated by the "banding" analysis was not the same as the local structure implied by the behavioral performance of the network. Based on the behavioral data alone, our conclusions about the internal local structure would have been incorrect. Therefore, this implies that cognitive neuroscientists must be very careful when assigning functions to local structure based on behavioral data alone.

Functional Architecture Conclusions

This chapter presented the results from two different lesioned connectionist networks in order to address two different aspects of the functional architecture. The first network examined how semantic memory may be organized by looking at the disruptions of the semantic network in AD patients. The second network addressed the locality assumption as used in cognitive neuroscience.

Results from the first experiment indicate that the semantic disturbances prevalent in AD patients can be accounted for by a simple weakening of connections between processing elements instead of the active reorganization of semantic knowledge. Therefore, the modified IAC network provides an alternative account of the functional architecture of semantic memory (as opposed to the classical semantic network). This alternative account is further strengthened by the fact that the IAC model also predicted that perseveration would occur in AD patients. Although this was an unexpected finding, a brief review of the literature indicates that perservation is indeed a characteristic of Alzheimer's. This result shows the power of the synthetic approach to cognitive science. By starting with very simple ideas, we have been able to account for what would otherwise be very complex emergent behaviour.

Furthermore, the results from the locality study suggest not only that PDP networks can have a highly local structure, but also that lesioning this structure can have very local effects on the behaviour of the network. This result contradicts Farah's (1994) position that models consistent with generic connectionism necessarily represent a nonlocal architecture. Thus, we have used a PDP network to cast doubt on Farah's critique of the locality assumption and support Shallice's cautious move of abandoning the modularity assumption. The discovery of a double dissociation in a PDP network is not sufficient to make a claim about modularity; however, it does appear to be sufficient to infer the existence of internal local structure.

The results from both of these experiments show how connectionism can contribute to studies of the functional architecture. Thus, where Chapter 6 has shown that connectionist models can produce novel cognitive theories that are not mere implementations of classical accounts, Chapter 7 has now shown how the very same connectionist principles used to produce cognitive algorithms can be applied to answer questions about the very building blocks of cognition.

As stated at the beginning of the chapter, however, the functional architecture also acts as the bridge between the algorithmic level and the implementational level—it shows how the algorithm can be implemented in the physical device. Fortunately, the connectionist principles developed so far offer a natural extension into the final level of the tri-level hypothesis: the study of the physical device—our brain.

Chapter 8

The Implementational Analysis of Connectionism

We have seen how connectionism can be used as a tool for exploring questions posed at the computational, algorithmic, and functional architecture levels. We will now conclude with connectionism's contribution to the implementational level description of an information processor. The implementational level is concerned with describing those physical properties of the system that are essential for carrying out the functions relevant for information processing. Therefore, the implementational level can be attacked from two different angles. The first is to look at the biological plausibility of connectionist networks. Although connectionists like to say that their networks are biologically inspired, many of the assumptions held by researchers need to be re-evaluated. The second, and more productive angle, is to ask what connectionism has contributed to the field of neuroscience. If connectionism cannot be explained at an implementational level, then it should not be able to contribute to neuroscience.

To conclude our look at the implementational level, connectionism's contribution to the study of biological redundancy will be revisited. This research will address not only the issue of applying biological constraints to connectionist networks, but also the issue of using connectionism to answer questions about the structure of the brain.

Biological Plausibility of Connectionism

Before we can start asking in earnest what types of contributions connectionism has made to neuroscience, we must ensure that the *Artificial Neural Networks* (ANNs) in question are biologically plausible. In other words, just because a network is able to mimic a certain biological phenomenon does not mean that the network is a valid model of brain functioning. For example, there are several properties of ANNs that are in direct conflict with known biological properties (e.g., Crick & Asanuma, 1986; Smolensky, 1988); consequently, unless these areas are addressed, ANNs will be no more than rough analogies to the brain and their contributions to the analysis of the implementational level will be limited at best.

To address this problem, Gardner (1993) has called for a new generation of connectionist networks to be developed. This third generation of networks should be neuromorphic, transcending the simplified components, layered architectures, and limited scale of first and second generation networks[1]. Consequently, third generation connectionism should be a hybrid of neurobiology and neural networks. In other words, we should be using knowledge from neurobiology to guide the development of neural networks. We must keep in mind, of course, that we only want to capture those aspects of neurobiology that are functionally important to information processing (Churchland & Churchland, 1990). We do not want our networks to smell bad when they rot (or are lesioned), if smelling bad is not a function of information processing.

This section will briefly provide three different examples of how this problem of functional biological plausibility is currently being addressed. These areas are (i) the learning rules used to train ANNs (especially the GDR), (ii) the assumption of monotonic and homogeneous networks, and (iii) massive parallelism (see also Crick, 1989). Each area shall be discussed in terms of their conflict with known biological properties and their recent advancement towards biological plausibility.

The first problem encountered when equating the generic PDP architecture to the known properties of biological networks is the learning rules used to train the artificial networks. For example, consider the GDR. As stated earlier, the GDR relies on propagating the error between network output and an externally supplied desired output back through the network. The idea of comparing the network output to an externally supplied output is not our main concern, as one can imagine reaching towards an object (network output), comparing how far your hand is from the

[1] Gardner defines first generation networks as those typified by Rosenblatt's single layer perceptron (see Chapter 3) and second generation networks as fully trainable multilayered perceptrons (see Chapter 4).

intended target (desired output), and then using the difference to adjust your next reaching response. The question that does concern us, however, is how that error difference is processed by the network.

Although there is evidence that dendrites both receive and transmit signals (e.g., Levitan & Kaczmarek, 1991), and that substances can pass backwards through a chain of neurons (e.g. HRP tracing; Kandel, Schwartz, & Jessel, 1991), it is often contended that back-propagation is biologically implausible. For example, Churchland and Sejnowski (1989, p. 41) state that "back-propagation is biologically implausible, inasmuch as error signals cannot literally be propagated back down the very same axon the signal came up." Hanson (1990) points out, however, that most arguments of this nature usually mean "that the 'C-code' implementation is biologically implausible, not the implementation one might find in brains." (p. 514).

Backpropagation aside, many of the learning rules do have biological plausibility. Hebbian learning necessarily has its basis in the neurophysiology of the brain (Baxter & Byrne, 1993). Furthermore, there is evidence in the brain of recurrent networks of neurons, so certain types of recurrent ANNs may have biological plausibility (e.g. Kandel et al., 1991; Lisberger & Sejnowski, 1992). Nevertheless, problems still exist with the training algorithms as most of them require that the activation function be monotonic.

Although most ANN training algorithms require their activation functions to be monotonic, there is considerable evidence (e.g., Ballard, 1986) to suggest that nonmonotonicity is prevalent in certain biological networks; for example, colour perception in the retina and simple cell receptive fields. As much of this thesis has shown, when a nonmontonic function such as the Gaussian is used within a network's processing units (i.e., the value units of Dawson and Schopflocher, 1992), then network performance is generally enhanced—especially for pattern classification. Therefore, adopting a biological constraint actually improves the performance of ANNs on certain tasks. Furthermore, if value units are used in the hidden layer of a network while integration devices are used as the output units (creating a nonhomogeneous network), then the resulting networks train faster, are easier to interpret because of fewer units and connections, and can generalize their performance to new instances better than standard networks (e.g., Dawson, Schopflocher, Kidd, & Shamanski, 1992).

Another contention between biology and ANNs is the massively parallel assumption governing connections between units within ANNs. This assumption means that the number of connections can be overwhelming. This, of course, can lead to difficulties in interpreting network structure and function. Although the brain has been called "massively parallel", the term does not carry the same meaning, proportionally speaking, as a massively parallel ANN. In a massively parallel ANN, each unit in one layer projects to each and every unit in the layer above it as well as to the units in the layer below it: There is no indication that the brain is wired in such a fashion. Current research indicates that when spatial constraints are applied to connections in ANNs, then biologically plausible "receptive fields" have evolved in pattern recognition networks (Dawson, Kremer, & Gannon, 1994). By limiting the number of connections within an ANN, the functional aspects of mammalian pattern recognition emerge (cf., Moorhead, Haig, & Clement, 1989).

As a last note, there will always be those researchers who strictly equate connectionism with computing science and mathematics. Consequently, they argue that connectionism is not biological and will never be of interest to cognitive science (e.g., Searle, 1980, 1990). If, however, we keep in mind that we are only interested in those properties of the nervous system that are functionally important to information processing (e.g., Churchland & Churchland, 1990) then connectionism has the potential to contribute to cognitive neuroscience.

Connectionism and Neuroscience

Although connectionism has drawn heavily from neuroscience, its contributions back to the discipline are often viewed as lacking. The silicon chip is definitely not living and bares no resemblance to the brain, so how could it contribute to something inherently biological? Connectionist models are at best "stick and ball models" (Douglas & Martin, 1991, p. 292) according to some neurophysiologists, and therefore have little to contribute to neuroscience based on that fact alone. If, however, one adopts a functionalist approach to brain and mind, then the potential contribution of connectionism to neuroscience is substantial.

Two areas in neuroscience that are often passed over are the localization and the redundancy of functions in the brain. Although much has been done to identify the different areas and their related functions within the brain (for review see Kandel et al., 1991; Kolb & Wishaw, 1996), little attention has been paid to why these functional areas are localized in the brain. Ballard (1986) suggests, however, that this type of organization is to be expected if a connectionist network evolves to solve the "packing problem". Given a finite number of processors (neurons), and an enormous variety of functions (vision, movement, etc.), the only way for the network (brain) to successfully code all functions is to group corresponding processes into functional areas. Although Farah (1994) has questioned the locality assumption based on results from connectionist networks, the results reported in Chapter 7 indicate that connectionist networks do organize into local structures. Consequently, connectionism has the potential to address the assumptions used by cognitive neuroscientists.

An interesting contribution that connectionism has made to neuroscience is a fully functional neuron made of silicon (Mahowald & Douglas, 1992). The artificial neuron combines neurophysiological principles with silicon engineering to produce an analog integrated circuit with the functional characteristics of real nerve cells. Ion current flow is emulated efficiently because the physics underlying the conductivity of silicon and of biological membranes is similar. Consequently, excitatory and inhibitory responses can be mimicked. The value of this research is apparent: Large artificial networks can be built with the effect of each addition or subtraction to the network immediately recognizable. We can see the effects of selectively destroying "neurons" without having to worry about costly animals and ethical considerations.

Perhaps a more practical contribution to neuroscience comes from Servan-Schreiber, Printz, and Cohen (1990) who modelled catecholamine effects on a neural network. Release of catecholamine increases the responsitivity of cells to excitatory and inhibitory inputs, producing many behavioural consequences, including attention, learning, memory, and motor behaviour. There is no adequate account, however, of how the effect of catecholamines on individual cells relates to overall behavioural changes.

Using a connectionist network, Servan-Schreiber, Printz, and Cohen determined that changes in the responsitivity of individual elements do not affect their ability to detect signals and ignore noise, but it is the combination of such changes throughout the network that leads to better signal detection. Their model was then pitted against actual human data from a signal detection task with central nervous system stimulants, and was found to be in line with the data (e.g. human subjects: 5.5% misses and 0.5% false alarms; simulation: 6.6% misses and 0.78% false alarms). Clearly, the connectionist network has added to our knowledge of the effect of catecholamines on individual neurons and how these effects are combined in a neural network to give us better signal detection.

Another example of connectionism's contribution to neuroscience is evident from motor learning in a recurrent network based on the vestibulo-ocular reflex (VOR) (Lisberger & Sejnowski, 1992). "The performance of the VOR is established and maintained by a learning mechanism that uses the association of visual and vestibular inputs to guide adaptive changes in the gain of the VOR" (Lisberger & Sejnowski, 1992, p. 160). Although it is normally agreed that learning is mediated by changes in the gain of steady-state transmission at different synaptic locations, the site of the learning has often been debated (see Judge, 1992) as there is contradictory data from neural recordings taken under different conditions. Lisberger & Sejnowski's network was able to convert long-term modulation of neural responses into changes in the amplitude of the steady output of the system. This conversion led them to propose a third alternative to the learning site which actually incorporates the otherwise contradictory data.

The final example of connectionism's contribution to neuroscience comes from Lynch, Granger, Larson, and Baudry (1989). Lynch et al. modeled the olfactory memory of the rat, giving a fairly detailed description of the relationships among patterns of axonal activity, NMDA receptors, and the production of LTP (see Lynch et al., 1989, Table 1). In programming their model into a connectionist network, however, they discovered that their description was not fully complete. Problems arose because individual neurons were not synchronized, fired with different frequencies and spikes, and longer lasting synaptic changes due to LTP had to interact with temporary changes in synaptic strengths brought on by particular patterns of stimulation. Consequently, Lynch et al. were forced to deal with "the precise formalization of necessary and sufficient conditions for LTP induction, which in turn forced us to ask many more specific physiological questions about these events than had been tested so far" (p. 196). In this case, the ANN served as the basis for seeking further information about LTP coordination in actual biological networks.

Redundancy Revisited

To conclude the implementational analysis of connectionism, the effects of applying the biological constraint of redundancy on ANNs will be evaluated. It will be shown how applying this constraint not only improves the performance of connectionist networks (and satisfies the move towards biological plausibility) but also contributes to the understanding the biological role of redundancy.

One biological characteristic often overlooked in the design of ANNs is redundancy, where redundancy is defined as the replication of functional processes within the brain. Redundancy in biological systems has been documented since the nineteenth century when it was proposed that the left and right hemispheres of the brain were basically mirror images of themselves and this therefore facilitated functional recovery following unilateral brain lesions (Gall & Spurzheim, 1810-1819; cited in Almli & Finger, 1992). Although we now know that the two hemispheres of the brain perform vastly different functions, redundancy within the two hemispheres is still held as a viable theory of functional recovery (Almli & Finger, 1992; Marshall, 1984).

Further neurophysiological evidence for redundancy in biological systems comes from recent studies of animal physiology. For example, Kovac, Davis, Matera, and Croll (1983) found several physiological systems within the nervous system of *Pleurobranchaea californica* that produced essentially the same behavior; when combined, though, these systems greatly enhanced the precision of simple and complex movements.

This results is consistent with Calvin's (1983) consideration of why redundancy may have developed from an evolutionary perspective. As an example, Calvin speculated on the evolutionary pressures exerted on the neural timing systems required by early hominids for hunting with thrown objects. At short distances, a single timing neuron is sufficient to allow the proper release time needed for a hit; however, the timing precision required for a strike increases eight-fold with a mere doubling of throwing distance. Thus, the brain needed to evolve a more accurate timing mechanism.

From an evolutionary perspective, the faster a system can evolve, the greater the chance of survival. Therefore, it would be more advantageous to evolve several small systems that work in parallel to achieve a goal, than a large and highly specialized system. In other words, instead of evolving a more accurate timing neuron, the brain combined the efforts of several timing neurons to increase the precision above that of any single neuron. Consequently, redundancy may have evolved because it was easier to replicate, and thus improve, what was already present than to develop a single system beyond reproach.

Connectionism offers an excellent opportunity to explore these implementational issues of redundancy. What happens if the simple processing elements of a connectionist network are replicated? Will accuracy increase beyond the performance of any single neuron? Taking a cue from Calvin (1983), a redundant network was created to simulate the motor activity of reaching to a point in two-dimensional space.

The Reaching Network

In 1994, Medler and Dawson presented the results of applying redundancy to networks trained on either pattern classification (e.g., parity) or function approximation (e.g., a simulated robotic arm). These results showed that redundancy produced faster convergence, more accurate results, and more stable networks than comparable standard non-redundant networks. The following experiment expands their procedure for assessing the effects of redundancy on the function approximation task.

Definition of Problem Space

The problem space used in this simulation was first described by Churchland (1992). Basically, a "crab" is trained to reach to a point in two-dimensional space. The crab has two rotatable eyes which are able to fixate on an object, and a two-joint arm (shoulder and elbow) which is able to move in the two-dimensions directly in front of the crab. Thus the inputs to the network are the two angles (α, β) subtended by the eyes while fixated on the object (Figure 8.1a), and the outputs are the two angles (θ, ϕ) required by the shoulder and elbow joints to reach the object (Figure 8.1b). All angles were normalized to fall within the range of 0 to 1. The inputs could be considered two-dimensional sensory-state space coordinates, and the outputs would then be considered as separate two-dimensional motor-state space coordinates. The network, therefore, learns the appropriate mapping between the two state spaces (see also Zipser and Anderson, 1988).

To train the network, 50 random points within the problem space were chosen and the appropriate angles for the input and output were computed. If a point fell within an unreachable area (the shaded areas in Figure 8.1) then a new point was chosen. These 50 points were then presented to the network, and the total sum squared error (SSE) of the network was calculated. This process comprised one training sweep. After each training sweep, another 50 random points were generated—this is in opposition to Medler and Dawson (1994) who used the same 50 training examples for each sweep. It is possible that the redundant networks reported in Medler and Dawson (1994) simply learned the 50 training examples and would not be able to generalize. Consequently, the new training paradigm will allow the generalization ability of redundant networks to be assessed.

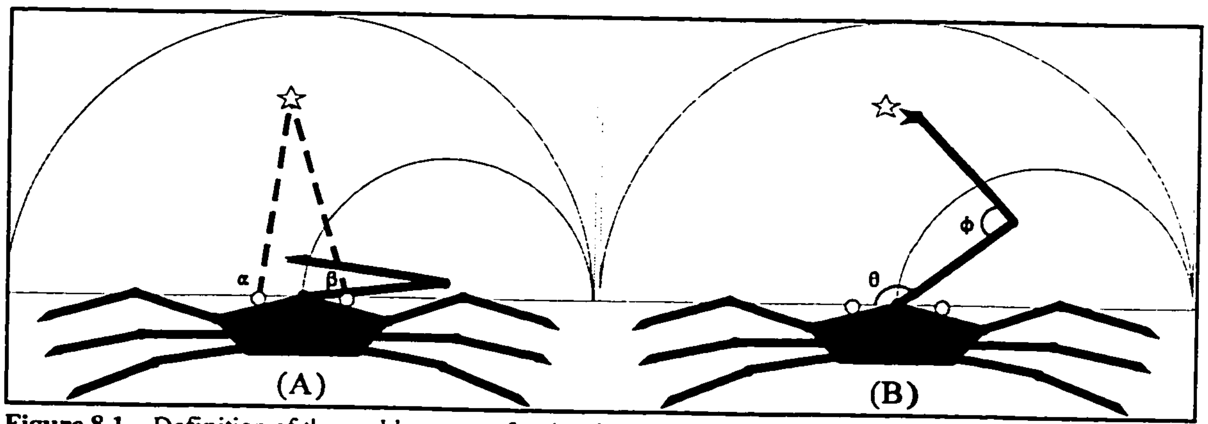

Figure 8.1 Definition of the problem space for the simulated robotic arm. (A) Inputs are the two angles (α, β) subtended by the eyes. (B) Outputs are the two angles (θ, ϕ) subtended by the shoulder and elbow joints respectively. The shaded areas are unreachable with the current configuration.

Network Architecture

The standard network architecture was a three-layer integration device network with two input units, two hidden units, and two output units. The connections between the input and hidden layers, and the hidden and output layers were fully parallel. Both connection weights and biases were randomized between [-1, 1]. The network was trained with the basic backpropagation algorithm, with the learning rate set to $\alpha = 0.1$, and momentum $\mu = 0.9$. As the network was engaged in function

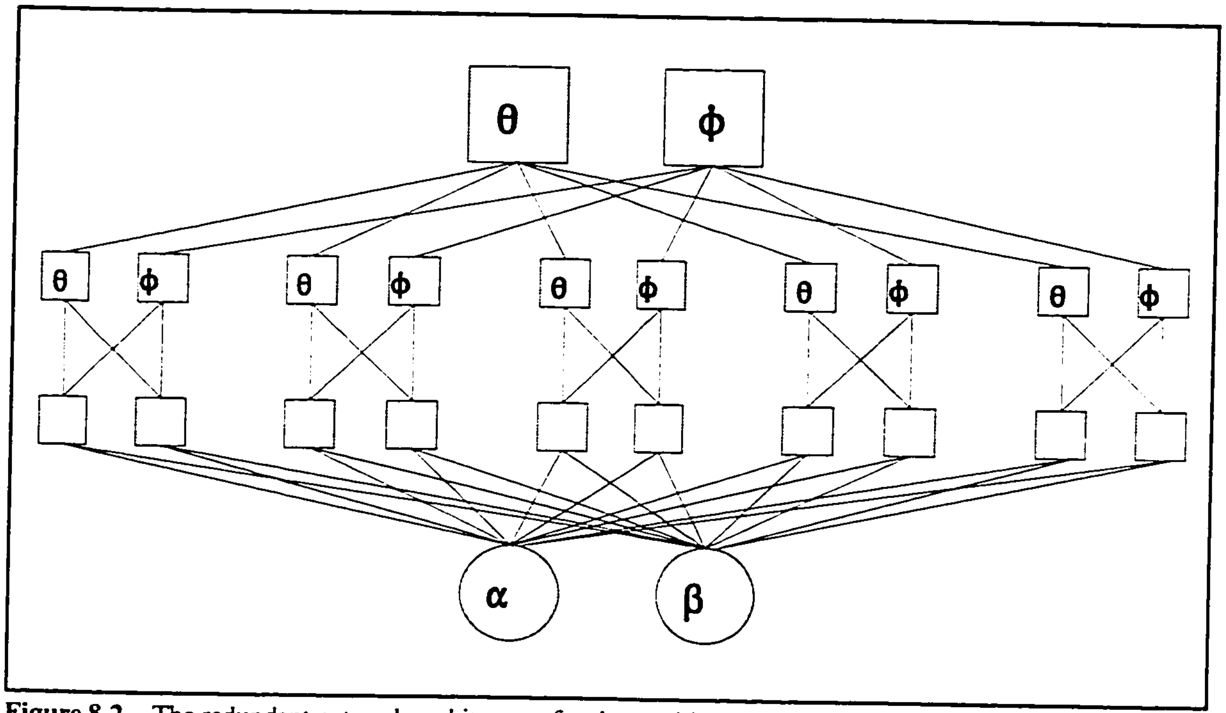

Figure 8.2 The redundant network architecture for the reaching network. Note the individual redundant networks are not directly connected to one another, and the redundant outputs are only connected to the appropriate decisions units.

approximation, the hit criterion was set to a stringent 0.000001 (requiring a output to be within 0.001 of the target value), and the maximum number of processing sweeps was set to 50,000.

The redundant network architecture was created by replicating the hidden unit layer and the output unit layer five times[2]. Each of the replicated output units was then connected to the appropriate Decision Unit, which acts as the redundant network's output units. All connections leading into the Decision Unit are modifiable; therefore, the Decision Unit's response is a weighted

[2] Medler & Dawson (1994b) showed that there was an optimal level of redundancy in terms of accuracy versus amount of processing. No advantage was gained from increasing the level of redundancy above five.

sum of the replicated output units. Figure 8.2 shows the redundant network structure for an ANN with five levels of redundancy trained on the reaching problem. It should be noted that, as opposed to a network with two layers of hidden units, no connections exist directly between each of replicated networks. Furthermore, each of the replicated networks was initialized independent of the others. Connection weights and biases were randomly distributed over the range [-1, 1], and the learning rate and momentum were set to $\alpha = 0.1$ and $\mu = 0.9$ respectively.

As the function approximation results from Chapter 5 suggest that accuracy increases when the number of hidden units is increased, the standard and redundant networks were also tested with five hidden units. Furthermore, one question that was not addressed in Medler and Dawson is considered in this experiment. Is the performance increase in redundant networks simply due to the increase in the number of hidden units? For example, a network with five hidden units and five levels of redundancy will actually have 25 hidden units. Consequently, a third standard network with 25 hidden units was also tested.

Results and Discussion

As the randomization process during weight assignment can introduce great variability in performance, ten different networks were created for each architecture to properly assess performance. Furthermore, in this experiment, each network was trained to the maximum number of 50,000 sweeps, with the network's performance being evaluated at each $\log_{10}$ processing step (i.e., 100, 200, ... , 900, 1000, 2000, etc.)[3]. Thus, network performance was evaluated by computing the mean total SSE and standard deviation of the ten networks at 23 different points. These values were then graphed so performance differences between the standard and redundant networks could be observed.

The results for the standard (S2) and redundant networks (R2) with two hidden units are presented in Figure 8.3. Table 8.1 presents the means and standard deviations of the SSE performance for all networks and Table 8.2 presents the results of the significance tests. As can be seen, within the first 1,000 processing sweeps, the redundant network performs as well as or worse than the standard network. This is reflected in the fact that the average SSE (taken over all processing sweeps) for the standard network was not significantly different than the mean SSE of

[3] This is in contrast to Medler and Dawson where the maximum network sweeps were increased from 100 to 50000 in $\log_{10}$ steps and SSE was recorded at each maximum sweep step. Thus, the current results show the averages of continuous training.

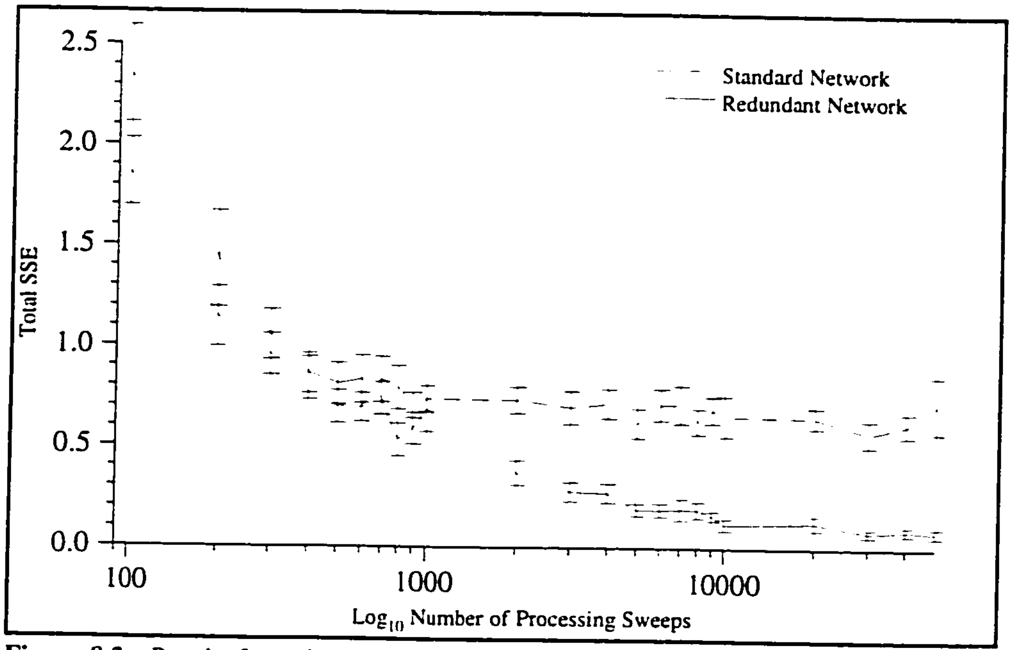

Figure 8.3 Results from the standard and redundant networks trained with two hidden units.

the redundant network. By 2,000 processing sweeps, however, there is a clear difference in the performance of the two networks. In fact, by the time 50,000 processing sweeps are completed, there is a significant difference between the mean SSE for the redundant network and the standard network. Furthermore, it can be seen from Figure 8.3 that as the number of processing sweeps increases, the standard error bars for the redundant network are smaller than for the standard networks. Thus, it is concluded that redundancy improves the performance of networks with two hidden units in terms of decreasing both SSE and the amount of variance in responding.

Figure 8.4 presents the results from the standard (S5) and redundant networks (R5) trained with five hidden units. The standard network with 25 hidden units (S25) is also presented for comparison. Again, when compared to both of the standard networks, the redundant network actually produces equivalent or worse behaviour during the initial training phase. The mean value of the average SSE for the redundant network is not significantly different from the average SSE for the standard networks. In fact, there is no significant difference in the average SSE between the standard network with five hidden units and the standard network with 25 hidden units.

Table 8.1

Means and standard deviations for SSE of the different networks

	Average SSE		50,000 SSE	
Architecture	$\bar{x}$	s.d.	$\bar{x}$	s.d.
Standard				
(S2)	0.782	0.26	0.716	0.28
(S5)	0.430	0.29	0.164	0.08
(S25)	0.427	0.32	0.095	0.04
Redundant				
(R2)	0.532	0.53	0.081	0.05
(R25)	0.495	0.52	0.040	0.03

Table 8.2

Results of statistical tests

Test ($\bar{x}_1$ - $\bar{x}_2$)	t	d.f.	p
Average SSE			
S2 - R2	1.34	18	n.s.
S5 - R5	-0.35	18	n.s.
S25 - R5	-0.35	18	n.s.
S25 - S5	0.02	18	n.s.
50,000 SSE			
S2 - R2	7.05	18	$p < .001$
S5 - R5	4.96	18	$p < .001$
S25 - R5	3.66	18	$p < .005$
S25 - S5	2.65	18	$p < .05$

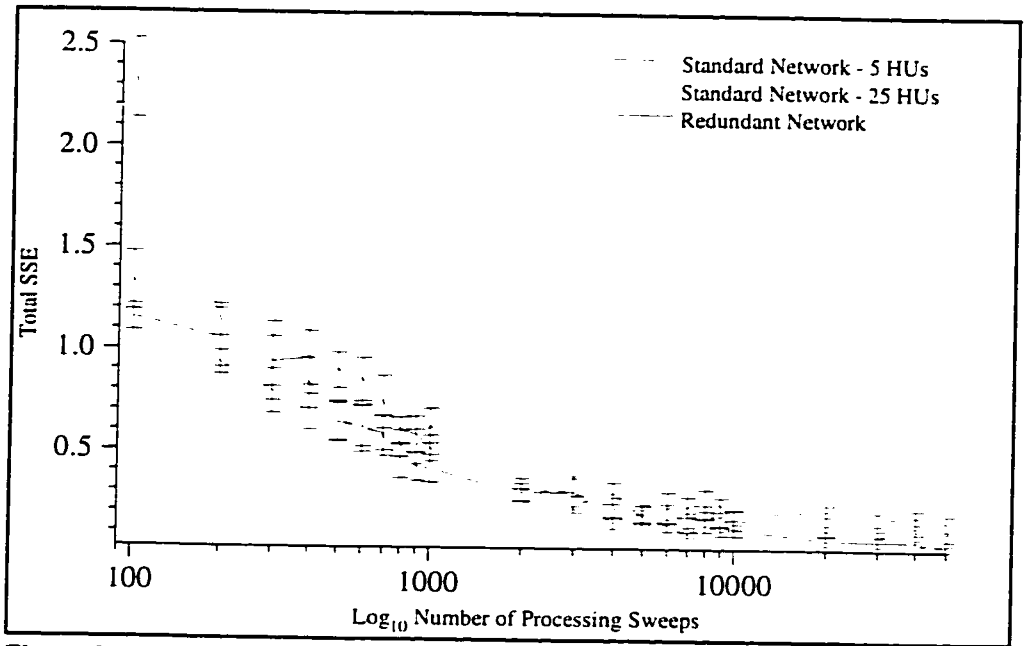

Figure 8.4 Results from the standard and redundant networks trained with five hidden units. A standard network with 25 hidden units is included for comparison purposes.

After 10,000 processing sweeps, however, significant differences in SSE begin to emerge. Looking at Figure 8.4, this is difficult to see because of the scale. But, statistical tests reveal that at 50,000 sweeps, the redundant network has a significantly smaller mean SSE than both of the standard networks. When the standard networks are compared at 50,000 sweeps, the network with 25 hidden units actually has a significantly smaller mean SSE. It is unclear from Figure 8.4 whether or not the standard error bars for the redundant networks are smaller than the standard error bars for the standard networks.

Thus, it can be concluded from these studies that redundancy significantly improves performance—measured by total SSE—as the number of processing sweeps is increased. Furthermore, with limited hidden units, the redundant networks show less variability in performance than standard networks. In other words, they are more stable. Also, the modified training procedure (i.e., training the network on continuously new data) shows the redundant networks were not keying in on specific input-output pairings, but could learn any relevant pairing. This is important because it has been suggested that increasing the number of processing elements over a certain point would actually decrease the generalization ability of the network (e.g., Werbos, 1995).

Interestingly, the difference between the standard and redundant networks became less as the number of hidden units was increased. This difference, however, remained significant. More importantly, by comparing the redundant network with five hidden units to the standard network with 25 hidden units, it has been shown that the increase in performance is not due simply to more processing units; this suggests that the structure imposed by redundancy is important.

From one side of the coin, adding the biological constraint of redundancy to the structure of ANNs has increased their performance. Although this is an important result in and of itself, the more important conclusion is that redundant networks can successfully incorporate neuromorphic properties. For example, redundant networks abandon the massively parallel assumption adopted by most networks, while increasing the number of simple processing elements. Therefore, we are moving towards a more biologically plausible neural network.

On the flip side of the coin, using redundancy in connectionist networks allows us to speculate on the relevance of redundancy in biological systems. The fact that redundancy increases the motor accuracy of the simulated reaching response is definitely in line with biological evidence from certain crustaceans whose movement is regulated via a set of redundant command neurons, each specialized for a specific range of motion (Kovac, Davis, Matera, & Croll, 1983).

This result also supports the evolutionary theories of redundancy. For example, the increased precision of the redundant network over the standard network lends credence to Calvin's (1983) hypothesis about redundancy evolving to increase the precision of a system. Also, the very distinct difference in accuracy between the redundant and standard networks with two hidden units suggests that it is easier to evolve several crude mechanisms working in parallel than one extremely effective mechanism.

Implementational Level Conclusions

In conclusion, not only can connectionist models be described at an implementational level, but they can also be used to explain implementational issues within cognitive science and neuroscience. Recent advancements have applied biological constraints to ANNs, creating what Gardner (1993) would term neuromorphic networks. Interestingly, making networks more biologically plausible has resulted in an (often unanticipated) improvement in network performance. This improvement in performance is clearly evident when the biological constraint of redundancy is applied to connectionist networks.

Furthermore, as long as researchers keep the biological basis of connectionism in mind, connectionism will continue to contribute theoretically and practically to neuroscience. In fact, the relationship between connectionism and neuroscience is one of give and take. As Wasserman (1989, p. 200) so aptly points out,

> Successful models, based upon speculations about the brain's structure, lead neuroanatomists and neurophysiologists to reexamine their observations, looking for corresponding structures and functions. Conversely, advances in the biological sciences have led to modified and elaborated artificial models.

Chapter 9

Where Do We Go from Here?

What have we learned while standing at the crossroads of connectionism? Is connectionism a Kuhnian-like paradigm shift as Schneider (1987) would have us believe, or is it simply an implementational account of classical cognitive architectures (e.g., Fodor & Pylyshyn, 1988)? Furthermore, can connectionism contribute significantly to the study of cognitive science? In other words, does connectionism have what it takes to be an effective tool within cognitive science?

In order to answer these questions, we began by defining the study of cognitive science. Cognitive science is the interdisciplinary study of the mind. It is based on the assumption that the mind is an information processor governed by scientific principles, and therefore open to scientific discovery. Thus, it was argued that if connectionism was to contribute significantly to cognitive science, it would have to be able to (i) generate data, (ii) compare data to theory, (iii) and articulate theory.

The best way to accomplish these three goals is to study information processing systems at three different levels: computational, algorithmic (including the functional architecture), and implementational. It was claimed that connectionism may be the very tool to provide explanations at all three levels of the tri-level hypothesis. To substantiate this claim, we presented both theoretical and empirical results from our analyses of connectionism within the tri-level hypothesis framework. What have these results led us to conclude?

A Paradigm Shift?

Is connectionism a Kuhnian-like paradigm shift?

No.

Although on the surface it may be appealing for connectionists to regard their field as a paradigm shift—especially within cognitive psychology—when the facts are considered, it is clear that this is not the case. Kuhn (1970) defines a paradigm[1] shift as consisting of three phases: (i) the discovery of novel facts or theories or at least the awareness of an anomaly within the previous framework, (ii) a more or less extended exploration of the anomaly, and (iii) the formulation of a new paradigm in which the anomalous becomes the expected. In other words, the paradigm shift emerges from the inadequacies of the previous paradigm and subsequently replaces it:

> That is what fundamental novelties of fact and theory do. Produced inadvertently by a game played under one set of rules, their assimilation requires the elaboration of another set. After they have become parts of science, the enterprise, at least of those specialists in whose particular field the novelties lie, is never quite the same again. (Kuhn, 1970, p. 52).

Consequently, to label connectionism as a paradigm shift in cognitive science—of which cognitive psychology is a subdiscipline—it should follow these three phases (or at least be in the process of these phases). But, connectionism does not even fall into the first phase.

Within cognitive science, the underlying tenet is that the mind is an information processor. Connectionism does not dispute this fact, but embraces it. In fact, our analysis of the computational level showed that, from a formal perspective, connectionist networks have the computational power to represent those information processing problems studied by cognitive science. Thus, the results of computability theory and complexity theory are fundamental to cognitive science and connectionism. There is no anomaly here.

To further strengthen this theoretical claim, empirical results were presented to show the *in practice* power of connectionist networks on function approximation and pattern classification tasks. Results from the function approximation experiments showed that integration device networks performed function approximation within a certain tolerance, and were able to interpolate to new

[1] It should be noted that the definition of a "paradigm" is far from clear. For example, Masterman (1967) points out that Kuhn (1962) defines "paradigm" no less than 22 times. Kuhn (1970) therefore refined his definition of paradigm into two separate usages: the first usage is for a group of shared theories (although Kuhn is more comfortable with the term *disciplinary matrix*) and the second is the group of shared examples.

values; extapolation, however, was poor. In other words, networks are very sensitive to the range of data they are trained on.

Normally, function approximation is limited to networks using standard monotonic activation functions. In fact, it had been previously proposed that value units would be inappropriate for function approximation due to the noninvertable nature of the Gaussian (Dawson & Schopflocher, 1992). Initial studies showed that this was indeed the case—although value units could approximate functions near values of one, they could not produce values near zero. If the modification to the cost function is removed from the training algorithm, however, then value unit networks are able to perform function approximation in practice.

In terms of pattern classification, it was shown that value units are superior to integration device networks for problems that are linearly inseparable. In fact, value unit networks converge on solutions not only faster, but also more reliably. Furthermore, value unit networks show better scalability; that is, increasing the size of the problem does not necessarily affect network performance. For linearly separable problems, however, integration device networks should be preferred as they do not require hidden units to successfully partition the pattern space (value units, on the other hand, do require hidden units). Consequently, each architecture is specialized for differnt types of problems.

These current empirical results and the past theoretical results of other researchers have shown that connectionist networks have the *in principle* power of a Universal Turing Machine and are thus able to answer the questions that are of interest to cognitive science. This necessarily begs the question, "what does connectionism contribute to the computational level analysis that cannot also be accounted for by classical approaches to cognitive science?" It was proposed that the answer to this question is actually two-fold.

First, it was argued that it is natural computation and not mathematical computation in the formal sense that is of interest to cognitive science. Connectionism is in a unique position to answer questions posed by natural computation. These include such aspects as minimum description length analysis and learning.

Second, connectionism is in an unique position to provide computational analysis that are not only descriptive, but are also explanatory. Consequently, connectionism is able to contribute computational level analyses of information processors based on and constrained by natural computation. In other words, connectionism is capable of explaining cognition, and not merely describing it.

But, do these two contributions of connectionism represent an anomaly within information processing theory? No, they merely offer a refinement. In other words, connectionism acts as a tool in which computational theories may be honed.

If connectionism is not a paradigm shift for cognitive science, is it a paradigm shift within cognitive psychology? One of the reasons that it may be considered a paradigm shift is that it is assumed that connectionist networks perform a qualitatively different type of information processing; one that is based on the interconnection of highly simplified processing elements. Again, though, this move does not change the nature of information processing, it merely refines it to capture more of the known "facts" (e.g., graceful degradation) that classical modelers are well aware of. In fact, as shown in Chapters 6 and 7, connectionist models can appear very "classical" in nature.

A Mere Implementation?

> If Connectionism is considered simply as a theory of how cognition is neurally implemented, it may constrain cognitive models no more than theories in biophysics, biochemistry, or, for that matter, quantum mechanics do. (Fodor & Pylyshyn, 1988, p. 68)

Does this mean that connectionism is merely an implementational account of classical theories within cognitive science then?

No.

Although classical researchers may find it appealing to consider connectionism as a mere implementation of classical theories, when facts are considered, again, this is not the case. As results have shown, connectionism can constrain cognitive models.

For example, at the algorithmic level, it was shown that the internal structure of a certain type of connectionist model—specifically, the value unit architecture—could be interpreted and analysed. This analysis technique has been termed "banding analysis" (Berkeley, et al., 1995) because it is based upon the examination of bands that emerge in the jittered density plots of the activations of hidden value units. It was shown that banding is a function of the nonmonotonicity of the Gaussian, and hence, should be applicable to any type of backpropagation network using a nonmonotonic activation function (e.g., McCaughan, 1997). Although the mathematical analysis of banding is important, the more interesting aspect of banding is that all patterns falling into a band share *definite* features. That is, all patterns within a band share some common interpretation.

Previous studies have shown that banding occurs in a number of different problems and with both binary and discrete non-binary data (e.g., Berkeley et al., 1995; Dawson & Medler, 1996), and that analysis of the banding reveals solutions that are not mere implementations of of classical theories (Dawson, Medler, & Berkeley, 1997). In Chapter 7, the banding analysis technique was used to answer a different question. Could "classical" rules be inserted into the structure of a connectionist network? In other words, could a connectionist network be taught classical rules?

To answer this question, a network was trained on the mushroom data set used in Dawson and Medler (1996) as a clear set of "classical" rules for correctly classify the data existed. Rules were presented to the network via elaborated output—the network had one output to represent the edibility of the mushroom, and nine other output units, one for each rule. Thus, the network was trained not only on the correct classification, but also on the reason for the classification.

Following successful training, banding analysis was performed on the hidden unit activations. The banding structure within the hidden units was very distinct and "cleaned up" in comparison to a control network and previous results (compare with Figure 1 of Dawson & Medler, 1996). Furthermore, each band was highly interpretable, having both definite unary and binary features. Importantly, when bands were combined across hidden units, it was revealed that the classical rules could be extracted from the network structure. Specifically, the four rules classifying poisonous mushrooms were represented by four different band combinations. The five rules classifying edible mushrooms were represented by 11 different band combinations. No other band combinations existed—the different rules were explicitly represented by very specific combinations of bands.

Furthermore, another control network showed that the elaborated output had to carry information to be useful. If the elaborated outputs were random, the network did not learn the problem and no banding was evident within the hidden units. Consequently, it has been shown that connectionist networks can be taught explicit classical rules for pattern classification by elaborating their output. Although these results may look to support Fodor and Pylyshyn's (1988) claim that networks are merely implementations of classical algorithms, all these results actually show is both classical and connectionist methods can produce equivalent theories. This new interpretation technique allows researchers to be able to state their algorithms explicitly and compare and contrast them with classical models. This step is necessary if connectionists want to show that their networks are not mere implementations of classical cognitive architectures.

Further evidence that connectionist networks are nor mere implementations of classical architectures was provided in Chapter 7. At the functional architecture level, it was shown how lesioning experiments on two different types of connectionist networks could be used to answer questions about the underlying structure of cognition. The first experiment used a modified Interactive Activation and Competition (IAC) model to address the breakdown in the semantic knowledge in patients with Alzheimer's disease (AD). In the second experiment, the results from a lesioned network were used to assess the locality assumption as used in cognitive neuroscience.

It has been previously noted that the semantic networks of AD patients are disturbed when compared to those of normals. It had been proposed by Chan et al. (1993) that part of this breakdown in semantic knowledge is the formation of new, albeit abnormal, associations. In the first experiment, however, it was shown how semantic disturbances can be accounted for by a simple weakening of connections between processing elements instead of the active reorganization of semantic knowledge.

Furthermore, the IAC model also predicted that perseveration would occur in AD patients. The network was not designed as a model of perseveration (in truth, we were unaware that AD patients showed perseveration). A brief review of the literature, however, indicates that perservation is indeed a characteristic of Alzheimer's. This result shows the power of the synthetic approach to cognitive science. By starting with very simple ideas, we have been able to account for what would otherwise be very complex emergent behaviour.

The second network combined the banding technique described in Chapter 6 with lesioning to address questions about the locality assumption as used in cognitive neuroscience. It had been argued by Farah (1994) that the locality assumption (defined by local structural damage producing local cognitive deficits) may very well be incorrect. This conclusion was based on the fact that Farah could produce local behavioural deficits in a PDP network that had been lesioned. Her conclusions, however, are based on the assumption that PDP networks are nonlocal by definition. But, she did not take the time to confirm her assumptions. If the time is taken to analyse the internal structure of the network, then it can be shown that PDP models do possess local structure, and that ablating this local structure produces local behavioural deficits. This is clearly established in the three different analyses of the lesioned logic network reported in Chapter 7. Therefore, the locality assumption may not be incorrect as Farah has stated.

Thus, the results from both the algorithmic level and functional architecture reveals that connectionism can inform and constrain cognitive models. Furthermore, connectionism is not a paradigm shift as the cognitive models produced clearly have a classical flavour, from the rules in the mushroom network, to the rules in the logic network. In other words, the theories produced by the connectionist networks can be translated into classical theories.

A New Direction

So, where do we go from here? If connectionism is neither a paradigm shift nor an implementational account of classical theories, then we may have to accept that the underlying distinctions often made between the "classical" approach and the "connectionist" approach may not be as valid as once thought. What is the advantage, then, of adopting a connectionist viewpoint?

Connectionism is as much a theory of information processing as it is a tool for collecting data. Connectionists use one common language whether giving descriptions at the computational level (e.g., Thorndike, 1932), the algorithmic level (e.g., Dawson, Medler, & Berkeley, 1997), or the implementational level (e.g., Zipser & Andersen, 1988).

> So long as the tools a paradigm supplies continue to prove capable of solving the problems it defines, science moves fastest and penetrates most deeply through confident employment of those tools. (Kuhn, 1970, p. 76)

This is in contrast to the classical approach which often must adopt different vocabularies for each level at the risk of losing important information in the translations. In all the experiments reported in this thesis, the common language of connectionism was used—the same principles applied to each level of analysis. Therefore, connectionism possesses the ability to unite the field of cognitive science.

References

Adams, D. (1979). *The hitch hiker's guide to the galaxy*. London: Pan Books.

Almli, C. R., & Finger, S. (1992). Brain injury and recovery of function: Theories and mechanisms of functional reorganization. *Journal of Head Trauma Rehabilitation, 7*, 70-77.

Andersen, R. A. (1995). Coordinate transformations and motor planning in posterior parietal cortex. In M. S. Gazzaniga (Ed.), *The cognitive neurosciences* (pp. 519-532). Cambridge, MA: MIT Press.

Anderson, J. R. (1985). *Cognitive psychology and its implications* (2nd ed.). New York: W. H. Freeman.

Anderson, J. A., & Rosenfeld, E. (Eds.). (1988). *Neurocomputing: Foundations of research.* Cambridge, MA: MIT Press.

Anderson, J. A., Pellionisz, A., & Rosenfeld, E. (Eds.). (1990). *Neurocomputing 2: Directions for research.* Cambridge, MA: MIT Press.

Anderson, J. A., Silverstein, J. W., Ritz, S. A., & Jones, R. S. (1977). Distinctive features, categorical perception, and probability learning: Some applications of a neural model. *Psychological Review, 84*, 413-451.

Antrobus, J. S. (1993). The dreaming mind/brain: Understanding its processes with connectionist models. In C. Cavallero and D. Foulkes (Eds.), *Dreaming as cognition* (pp. 77-92). London: Harvester Wheatsheaf.

Arbib, M. A. (1964). *Brains, machines, and mathematics.* New York: McGraw-Hill.

Arbib, M. A. (1969). Memory limitations of stimulus-response models. *Psychological Review, 76*, 507-510.

Arbib, M. A. (1972). *The metaphorical brain.* New York: Wiley.

Arbib, M. A. (Ed.) (1995). *The handbook of brain theory and neural networks.* Cambridge, MA: MIT Press.

Aune, B. (1970). *Rationalism, Empiricism, and Pragmatism: An introduction.* New York: Random House.

Baldi, P. F., & Hornik, K. (1995). Learning in linear neural networks: A survey. *IEEE Transactions on Neural Networks, 6,* 837-858.

Ballard, D. H. (1986). Cortical connections and parallel processing: Structure and function. *Behavioral and Brain Sciences, 9,* 67-120.

Ballard, D. H. (1997). *An introduction to natural computation.* Cambridge, MA: MIT Press.

Barlow, H. B. (1972). Single units and sensation: A neuron doctrine for perceptual psychology? *Perception, 1,* 371-394.

Barnden, J. A. (1995). Semantic networks. In M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks* (pp. 854-857). Cambridge, MA: MIT Press.

Battig, W. F., & Montague, W. E. (1969). Category norms for verbal items in 56 categories: A replication and extension of the Connecticut category norms. *Journal of Experimental Psychology Monograph, 80,* 1-46.

Baum, E. B., & Haussler, D. (1989). What size net gives valid generalization? *Neural Computation, 1,* 151-160.

Baxt, W. G. (1992). Improving the accuracy of an artificial neural network using multiple differently trained networks. *Neural Computation, 4,* 772-780.

Baxter, D. A., & Byrne, J. H. (1993). Learning rules from neurobiology. In D. Gardner (Ed.), *The neurobiology of neural networks* (pp. 71 - 105). Cambridge, MA: MIT Press.

Bechtel, W. (1985). Contemporary connectionism: Are the new parallel distributed processing models cognitive or associationist? *Behaviorism, 13*, 53-61.

Bechtel, W., & Abrahamsen, A. (1991). *Connectionism and the mind: An introduction to parallel processing in networks*. Cambridge, MA: Blackwell.

Berkeley, I. S. N., Dawson, M. R. W., Medler, D. A., Schopflocher, D. P., & Hornsby, L. (1995). Density plots of hidden value units activations reveal interpretable bands. *Connection Science, 7*, 167-186.

Best, J. B. (1995). *Cognitive psychology* (4th ed.). New York: West.

Bever, T. G., Fodor, J. A., & Garrett, M. (1968). A formal limitation of associationism. In T. R. Dixon and D. L. Horton (Eds.), *Verbal Behavior and General Behavior Theory* (pp. 582-585). Englewood Cliffs, NJ: Prentice-Hall.

Bizzi, E., & Mussa-Ivaldi, F. A. (1995). Toward a neurobiology of coordinate transformations. In M. S. Gazzaniga (Ed.), *The cognitive neurosciences* (pp. 495-506). Cambridge, MA: MIT Press.

Bliss, T. V. P., & Gardner-Medwin, A. R. (1973). Long-lasting potentiation of synaptic transmission in the dentate area of the unanaesthetized rabbit following stimulation of the perforant path. *Journal of Physiology, 232*, 357-374.

Bliss, T. V. P., & Lomo, T. (1973). Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path. *Journal of Physiology, 232*, 331-356.

Boden, M. A. (1988). *Computer models of mind*. Cambridge, UK: Cambridge University Press.

Braitenberg, V. (1984). *Vehicles*. Cambridge, MA: MIT Press.

Bridges, D. S. (1994). *Computability: A mathematical sketchbook*. New York: Springer-Verlag.

Broadbent, D. (1985). A question of levels: Comment on McClelland and Rumelhart. *Journal of Experimental Psychology: General, 114*, 189-192.

Bullinaria, J. A. (1994). Simulating nonlocal systems: Rules of the game. *Behavioral and Brain Sciences, 17*, 61-62.

Bunge, M., & Ardila, R. (1987). *Philosophy of psychology*. New York: Springer-Verlag.

Burns, C. D. (1915). Occam's razor. *Mind, 24*, 592.

Burr, D. J. (1994). Artificial neural networks: a decade of progress. In R. J. Mammone (Ed.), *Artificial neural networks for speech and vision* (pp. 3-10). London: Chapman & Hall.

Burton, A. M. (1994). Learning new faces in an interactive activation and competition model. In V. Bruce & G. W. Humphreys (Eds.), *Object and face recognition* (pp. 313-348).Hillsdale, NJ: Lawrence Earlbaum Associates.

Byrne, J. H., & Crow, T. (1995). Invertebrate models of learning: *Aplysia* and *Hermissenda*. In M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks* (pp. 487-491). Cambridge, MA: MIT Press.

Calvin, W. H. (1983). A stone's throw and its launch window: Timing precision and its implications for language and hominid brains. *Journal of Theoretical Biology, 104*, 121-135.

Caramazza, A. (1986). On drawing inferences about the structure of normal cognitive systems from the analysis of patterns of impaired performance: The case for single-patient studies. *Brain and Cognition, 5*, 41-66.

Carpenter, G. A., & Grossberg, S. (1995). Adaptive resonance theory (ART). In M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks* (pp. 79-82). Cambridge, MA: MIT Press.

Chambers, J. M., Cleveland, W. S., Kleiner, B., & Tukey, P. A. (1983). *Graphical methods for data analysis*. Belmont, CA: Wadsworth International Group.

Chan, A. S., Butters, N., Paulsen, J. S., Salmon, D. P., Swenson, M. R., & Maloney, L. T. (1993). An assessment of the semantic network in patients with Alzheimer's disease. *Journal of Cognitive Neuroscience, 5*, 254-261.

Churchland, P. M. (1989). *A neurocomputational perspective*. Cambridge, MA: MIT Press.

Churchland, P. M., & Churchland, P. S. (1990). Could a machine think? *Scientific American, 262*(1), 32-37.

Churchland, P. S., & Sejnowski, T. J. (1989). Neural representation and neural computation. In L. Nadel, L. Cooper, P. Culicover, & M. Harnish (Eds.), *Neural connections, mental computation* (pp. 15-48). Cambridge, MA: MIT Press.

Clark, A. (1989). *Microcognition*. Cambridge, MA: MIT Press.

Cleermans, A., Servan-Schreiber, D., & McClelland, J. L. (1989). Finite state automata and simple recurrent networks. *Neural Computation, 1*, 372-381.

Cohen, M. A., & Grossberg, S. G. (1983). Absolute stability of global pattern formation and parallel memory storage by competitive neural networks. *IEEE Transactions on Systems, Man, and Cybernetics, 13*, 815-826.

Collins, A. (1977). Why cognitive science. *Cognitive Science, 1*, 1-2.

Collins, A. M., & Loftus, E. F. (1975). A spreading activation theory of semantic processing. *Psychological Review, 82*, 407-428.

Collins, A. M., & Quillian, M. R. (1969). Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behavior, 11*, 671-684.

Collins, A., & Smith, E. E. (Eds.) (1988). *Readings in cognitive science: A perspective from psychology and artificial intelligence*. San Mateo, CA: Morgan Kaufmann.

Coltheart, M. (1980). Iconic memory and visible persistence. *Perception & Psychophysics, 27*, 183-228.

Crepel, F., Hemart, N., Jaillard, D., & Daniel, H. (1995). Long-term depression in the cerebellum. In M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks* (pp. 560-563). Cambridge, MA: MIT Press.

Crick, F., & Asanuma, C. (1986). Certain aspects of the anatomy and physiology of the cerebral cortex. In J. L. McClelland, D. E. Rumelhart, & the PDP Group (Eds.), *Parallel Distributed Processing* (vol. 2). Cambridge, MA: MIT Press.

Cummins, R. (1983). *The nature of psychological explanation*. Cambridge, MA: MIT Press.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems, 2*, 303-314.

Damasio, A. R. (1994). *Descartes' error*. New York: Avon Books.

Davidoff, J. (1995). Color perception. In M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks* (pp. 210-215). Cambridge, MA: MIT Press.

Dawson, M. R. W. (1990). Empirical issues in theoretical psychology: Comment on Kukla. *American Psychologist, 45*, 778-780.

Dawson, M. R. W. (1991). The how and why of what went where in apparent motion: Modeling solutions to the motion correspondence problem. *Psychological Review, 98*, 569-603.

Dawson, M. R. W. (1998). *Understanding cognitive science*. Cambridge, MA: Blackwell.

Dawson, M.R.W., Dobbs, A., Hooper, H.R., McEwan, A.J.B., Triscott, J., & Cooney, J. (1994). Artificial neural networks that use SPECT to identify patients with probable Alzheimers disease. *European Journal of Nuclear Medicine, 21*, 1303-1311.

Dawson, M. R. W., Kremer, S. C., & Gannon, T. N. (1994). Identifying the trigger features for hidden units in a PDP model of the early visual pathway. In R. Elio (Ed.), *Proceedings of the tenth Canadian conference on artificial intelligence* (pp. 115-120). Palo Alto, CA: Morgan Kaufman.

Dawson, M. R. W., & Medler, D. A. (1996). Of mushrooms and machine learning: Identifying algorithms in a PDP network. *Canadian Artificial Intelligence, 38*, 14-17.

Dawson, M. R. W., Medler, D. A., & Berkeley, I. S. N. (1997). PDP networks can provide symbolic models that are not mere implementations of classical theories. *Philosophical Psychology, 10*, 25-40.

Dawson, M. R. W., & Schopflocher, D. P. (1992). Modifying the generalized delta rule to train networks of non-monotonic processors for pattern classification. *Connection Science, 4*,19-31.

Dawson, M. R. W., Schopflocher, D. P., Kidd, J., & Shamanski, K. S. (1992). Training networks of value units. *Proceedings of the Ninth Canadian Artificial Intelligence Conference*, pp. 244-250.

Dawson, M. R. W., & Shamanski, K. S. (1994). Connectionism, confusion, and cognitive science. *Journal of Intelligent Systems, 4*, 215-262.

Dawson, M. R. W., Shamanski, K. S., & Medler, D. A., (1993). From connectionism to cognitive science. In L. Goldfarb (Ed.) *Proceedings of the Fifth University of New Brunswick Artificial Intelligence Symposium. pp. 295 - 305.* Fredericton, NB: UNB Press.

Douglas, R. J., & Martin, K. A. C. (1991). Opening the grey box. *Trends in Neuroscience, 14*, 286-293.

Doya, K. (1995). Recurrent networks: Supervised learning. In M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks* (pp. 796-800). Cambridge, MA: MIT Press.

Dudai, Y. (1989). The neurobiology of memory. Oxford: Oxford University Press.

Dutton, J. M., & Starbuck, W. H. (1971). *Computer simulation of human behavior*. New York: John Wiley & Sons.

Eckmiller, R. (1989). Generation of movement trajectories in primates and robots. In I. Aleksander (Ed.). *Neural computing architectures: The design of brain-like machines*. Cambridge, MA: MIT Press.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science, 14*, 179-211.

Elman, J. L. (1991). Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning, 7*, 195-225.

Farah, M. J. (1994). Neuropsychological inference with an interactive brain: A critique of the "locality" assumption. *Behavioral and Brain Sciences, 17*, 43-104.

Feldman, J. A., & Ballard, D. H. (1982). Connectionist models and their properties. *Cognitive Science, 6*, 205-254.

Feldman-Stewart, D., & Mewhort, D. J. K. (1994). Learning in small connectionist networks does not generalize to large networks. *Psychological Research, 56*, 99-103.

Fodor, J. A. (1968). *Psychological explanation: An introduction to the philosophy of psychology*. New York: Random House.

Fodor, J. A. (1983). *The modularity of mind*. Cambridge, MA: MIT Press.

Fodor, J. A., & Pylyshyn, Z. W. (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition, 28*, 3-71.

Földiák, P., & Young, M. P. (1995). Sparse coding in the primate cortex. In M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks* (pp. 895-898). Cambridge, MA: MIT Press.

Franz, E. A., Ivry, R. B., & Helmuth, L. L. (1996). Reduced timing variability in patients with unilateral cerebellar lesions during bimanual movements. *Journal of Cognitive Neuroscience, 8*, 107-188.

Gallant, S. I. (1993). *Neural network learning and expert systems*. Cambridge, MA: MIT Press.

Gallistel, C. R. (1995). The replacement of general-purpose theories with adaptive specializations. In M. S. Gazzaniga (Ed.), *The cognitive neurosciences* (pp. 1255-1267). Cambridge, MA: MIT Press.

Gällmo, O., & Carlström, J. (1995). Some experiments using extra output learning to hint multi layer perceptrons. In L.F. Niklasson & M.B. Bodén (Eds.), *Current trends in connectionism - Proceedings of the 1995 Swedish Conference on Connectionism* (pp. 179-190). Hillsdale, NJ: Lawrence Earlbaum Associates.

Gardner, D. (Ed.) (1993). *The neurobiology of neural networks*. Cambridge, MA: MIT Press.

Gardner, H. (1985). *The mind's new science*. New York: Basic Books.

Gazzaniga, M. S. (Ed.) (1995). *The cognitive neurosciences*. Cambridge, MA: MIT Press.

Georgopoulos, A. P. (1995). Motor cortex and cognitive processing. In M. S. Gazzaniga (Ed.), *The cognitive neurosciences* (pp. 507-518). Cambridge, MA: MIT Press.

Ginsberg, M. (1993). *Essentials of artificial intelligence*. San Francisco: Morgan Kaufmann.

Girling, D. M., & Berrios, G. E. (1990). Extrapyramidal signs, primitive reflexes, and frontal lobe function in senile dementia of the Alzheimer type. *British Journal of Psychiatry, 157*, 888-893.

Girosi, F., & Poggio, T. (1990). Networks and the best approximation property. *Biological Cybernetics, 63*, 169-176.

Gouin, P., & Scofield, C. (1994). Neural network segmentation and recognition of text data on engineering documents. In R. J. Mammone (Ed.), *Artificial neural networks for speech and vision* (pp. 562-573). London: Chapman & Hall.

Gray, P. (1994). *Psychology* (2nd ed.). New York: Worth.

Green, D. W. (Ed.) (1996). *Cognitive science: An introduction*. Cambridge, MA: Blackwell.

Grossberg, S. (1974). Classical and instrumental learning by neural networks. *Progress in Theoretical Biology, 3*, 51-141.

Grossberg, S. (1976). Adaptive pattern classification and universal recording: I. Parallel development and coding of neural feature detectors. II. Feedback, expectation, olfaction, and illusions. *Biological Cybernetics, 23*, 121-134, 187-202.

Grossberg, S. (1978). A theory of visual coding, memory, and development. In E. L. J. Leeuwnberg & H. F. J. M. Buffart (Eds.), *Formal theories of visual perception*. New York: Wiley.

Gruenewald, P. J., & Lockhead, G. R. (1980). The free recall of category examples. *Journal of Experimental Psychology: Human Learning and Memory, 6*, 225-240.

Halmos, P. R. (1974). *Naive set theory*. New York: Springer-Verlag.

Halmos, P. R. (1987). *Finite-dimensional vector spaces*. New York: Springer-Verlag.

Hanson, S. J. (1990). Learning and representation: Tensions at the interface. *Behavioral and Brain Sciences, 13*, 511-518.

Hanson, S. J., & Burr, D. J. (1990). What connectionist models learn: Learning and representation in connectionist networks. *Behavioral and Brain Sciences, 13*, 471-489.

Hanson, S. J., & Olson, C. R. (1991). Neural networks and natural intelligence: Notes from Mudville. *Connection Science, 3*, 332-335.

Harley, T. A. (1993). Connectionist approaches to language disorders. *Aphasiology, 7*, 221-249.

Hartman, E., Keeler, J. D., & Kowalski, J. M. (1990). Layered neural networks with Gaussian hidden units as universal approximations. *Neural Computation, 2*, 210-215.

Hecht-Nielson, R. (1992). Neural networks for image analysis. In G. A. Carpenter & S. Grossberg (Eds.), *Neural networks for vision and image processing* (pp. 449-460). Cambridge, MA: MIT Press.

Hebb, D. O. (1949). *The organization of behavior*. New York: John Wiley & Sons.

Hinton, G. E. (1986). Learning distributed representations of concepts. *Eighth Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Lawrence Earlbaum Associates.

Hinton, G. E., & Anderson, J. A. (Eds.) (1981). *Parallel models of associative memory*. Hillsdale, NJ: Lawrence Earlbaum Associates.

Hinton, G. E., McClelland, J. L., & Rumelhart, D. E. (1986). Distributed representations. In D. E. Rumelhart, J. L. McClelland, & the PDP Research Group (Eds.), *Parallel Distributed Processing* (vol. 1; pp 77-109). Cambridge, MA: MIT Press.

Hinton, G. E., Plaut, D. C., & Shallice, T. (1993). Simulating brain damage. *Scientific American, 269*(4), 76-82.

Hinton, G. E., & Sejnowski, T. J. (1986). Learning and relearning in Boltzman machines. In D. E. Rumelhart, J. L. McClelland, & the PDP Research Group (Eds.), *Parallel Distributed Processing* (vol. 1; pp 282-317). Cambridge, MA: MIT Press.

Hinton, G. E., & Shallice, T. (1991). Lesioning an attractor network: Investigations of acquired dyslexia. *Psychological Review, 98,* 74-95.

Hopcroft, J. E., & Ullman, J. D. (1979). *Introduction of automata theory, languages, and computation.* Reading, MA: Addison-Wesley Publishing.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences, 79,* 2554-2558.

Horgan, T., & Tienson, J. (1996). *Connectionism and the philosophy of psychology.* Cambridge, MA: MIT Press.

Horn, B. K. P. (1979). Kinematics, statics, and dynamics of two-dimensional manipulators. In P. H. Winston, & R. H. Brown (Eds.). *Artificial intelligence: An MIT perspective* (vol. 2) (pp. 273-308). Cambridge, MA: MIT Press.

Hornik, M., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks,* 2, 359-366.

Hubel, D. H., & Wiesel, T. N. (1959). Receptive fields of single neurons in the cat's striate cortex. *Journal of Physiology (London), 148,* 574-591.

Hull, C. L. (1943). *Principles of Behavior.* New York: Appleton-Century-Crofts.

Hull, C. L., & Baernstein, H. D. (1929). A mechanical parallel to the conditioned reflex. *Science, 70,* 14-15.

Humphreys, G. W., Freeman, T. A. C., & Muller, H. J. (1992). Lesioning a connectionist model of visual search: Selective effects on distractor grouping. *Canadian Journal of Psychology, 46*, 417-460.

Humphreys, G. W., Riddoch, M. J., Quinlan, P. T., Price, C. N., & Donnelly, N. (1992). Parallel pattern processing and visual agnosia. *Canadian Journal of Psychology, 46*, 377-416.

Hunter, W. S. (1930). A consideration of Lashley's theory of the equipotentiality of cerebral action. *Journal of General Psychology, 3*, 455-468.

Hussain. T. S., & Browse, R. A. (1994). ARTSTAR: A supervised adaptive resonance classifier. In R. Elio (Ed.) *Proceedings of the Tenth Biennial Conference of the Canadian Society for Computational Studies of Intelligence. pp. 121-130.* Palo Alto, CA: Morgan Kauffman.

Izui, Y., & Pentland, A. (1990). Analysis of neural networks with redundancy. *Neural Computation, 2*, 226-238.

Jakendoff, R. (1992). *Languages of the mind.* Cambridge, MA: MIT Press.

James, W. (1890/1950). *The principles of psychology.* New York: Dover.

Johnson-Laird, P. N. (1988). *The computer and the mind: An introduction to cognitive science.* Cambridge, MA: Harvard University Press.

Jordan, M. I. (1995). Computational motor control. In M. S. Gazzaniga (Ed.), *The cognitive neurosciences* (pp. 597-609). Cambridge, MA: MIT Press.

Judge, S. J. (1992). Stitch in time saves design. *Nature, 360*, 104.

Kaiser, H. R. (1958). The VARIMAX criterion for analytic rotation in factor analysis. *Psychometrika, 23*, 187-200.

Kandel, E. R., Schwartz, J. H. & Jessel, T. M. (1991). *Principles of neuroscience* (3rd Ed.). New York: Elsevier.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics, 43,* 59-69.

Kolb, B., & Whishaw, I. Q. (1996). *Fundamentals of human neuropsychology* (4th ed.). New York: W. H. Freeman.

Kosslyn, S. M. (1980). *Image and mind.* Cambridge, MA: Harvard University Press.

Kosslyn, S. M. (1990). Mental imagery. In D. N. Osherson, S. M. Kosslyn, & J. M. Hollerbach (Eds.), *visual cognition and action* (pp. 73-97). Cambridge, MA: MIT Press.

Kosslyn, S. M. (1994). *Image and brain: The resolution of the imagery debate.* Cambridge, MA: MIT Press.

Kosslyn, S. M., Ball, T. M., & Reiser, B. J. (1978). Visual images preserve metric spatial information: Evidence from studies of image scanning. *Journal of Experimental Psychology: Human Perception and Performance, 4,* 47-60.

Kosslyn, S. M., Thompson, W. L., Kim, I. J., Rauch, S. L., & Alpert, N. M. (1996). Individual differences in cerebral blood flow in area 17 predict the time to evaluate visualized letters. *Journal of Cognitive Neuroscience, 8,* 78-82.

Kovac, M. P., Davis, W. J., Matera, E. M., & Croll, R. P. (1983). Organization of synaptic inputs to paracerebral feeding command interneurons of *Pleurobranchaea californica.* I. Excitatory inputs. *Journal of Neurophysiology, 49,* 1517-1538.

Kuhn, T. S. (1970). *The structure of scientific revolutions* (2nd ed.). Chicago: University of Chicago Press.

Kukla, A. (1989). Nonempirical issues in psychology. *American Psychologist, 44,* 785-794.

Kukla, A. (1990). Theoretical psychology, artificial intelligence, and empirical research. *American Psychologist, 45,* 780-781.

Kuperstein, M. (1988). Neural model of adaptive hand-eye coordination for single postures. *Science, 239,* 1308-1311.

Kupfermann, I., Castelluci, U., Pinsker, H., & Kandel, E. R. (1970). Neuronal correlates of habituation and dishabituation of the gill-withdrawl reflex in *Aplysia. Science, 167,* 1743-1745.

Lashley, K. S. (1950). In search of the engram. In *Society of Experimental Biology Symposium No. 4: Psychological mechanisms in animal behavior* (pp. 478-505). London: Cambridge University Press.

LeDoux, J. E., & Fellous, J.-M. (1995). Emotion and computational neuroscience. In M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks* (pp. 356-359). Cambridge, MA: MIT Press.

Leiber, J. (1991). *An invitation to cognitive science.* Cambridge, MA: Basil Blackwell.

Levitan, I. B., & Kaczmarek, L. K. (1991). *The neuron.* Oxford: Oxford University Press.

Lewandowsky, S. (1993). The rewards and hazards of computer simulations. *Psychological Science, 4,* 236-243.

Lincoff, G. H. (1981). *National Audubon Society field guide to North American mushrooms.* New York: Alfred A. Knopf Publishers.

Lindsay, P. H., & Norman, D. A. (1972). *Human information processing.* New York: Academic Press.

Lippman, R. P. (1987). An introduction to computing with neural nets. *IEEE ASSP Magazine, April,* 4-22.

Lisberger, S. G., & Sejnowski, T. J. (1992). Motor learning in a recurrent network model based on the vestibulo-ocular reflex. *Nature, 360,* 159-161.

Lorenz, K. Z. (1974). Analogy as a source of knowledge. *Science, 185,* 229-234.

Lowe, D. (1995). Radial basis function networks. In M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks* (pp. 779-782). Cambridge, MA: MIT Press.

Lynch, G., Granger, R., Larson, J., & Baudry, M. (1989). Cortical encoding of memory: Hypotheses derived from analysis and simulation of physiological learning rules in anatomical structures. In L. Nadel, L. A. Cooper, P. Culicover, & R. M. Harnish (Eds.), *Neural connections, mental computation*. Cambridge, MA: MIT Press.

Mahowald, M., & Douglas, R. (1991). A silicon neuron. *Nature, 354*, 515-518.

Marr, D. (1982). *Vision*. San Francisco: W. H. Freeman.

Marshall, J. F. (1984). Brain function: Neural adaptations and recovery from injury. *Annual Review of Psychology, 35*, 277-308.

Massaro, D. W. (1988). Some criticisms of connectionist models of human performance. *Journal of Memory & Learning, 27*, 213-234.

Massaro, D. W. (1990). The psychology of connectionism. *Behavioral and Brain Sciences, 13*, 403-406.

Masterman, M. (1967). The nature of a paradigm. In M. Kochen (Ed.), *The growth of knowledge; readings on organization and retrieval of information*. New York: Wiley.

McCaughan, D. B. (1997). On the properties of periodic perceptrons. *Proceedings of the 1997 International Conference on Neural Networks*, 188-193.

McClelland, J. L. (1981). Retrieving general and specific information from stored knowledge of specifics. *Proceedings of the Third Annual Meeting of the Cognitive Science Society*, 170-172.

McClelland, J. L., & Rumelhart, D. E. (1985). Distributed memory and the representation of general and specific information. *Journal of Experimental Psychology: General, 114*, 159-188.

McClelland, J. L., & Rumelhart, D. E. (1986). A distributed model of human learning and memory. In J. L. McClelland, D. E. Rumelhart, & the PDP Research Group (Eds.), *Parallel Distributed Processing* (vol. 2; pp 170-215). Cambridge, MA: MIT Press.

McClelland, J. L., & Rumelhart, D. E. (1988). *Explorations in parallel distributed processing*. Cambridge, MA: MIT Press.

McClelland, J. L., Rumelhart, D. E., & Hinton, G. E. (1986). The appeal of Parallel Distributed Processing. In J. L. McClelland, D. E. Rumelhart, & the PDP Research Group (Eds.), *Parallel Distributed Processing* (vol. 1; pp 3-40). Cambridge, MA: MIT Press.

McCloskey, M. (1991). Networks and theories: The place of connectionism in cognitive science. *Psychological Science, 2*, 387-395.

McCloskey, M. (1993). Theory and evidence in cognitive neuropsychology: A "radical" response to Robertson, Knight, Rafal, and Shimamura (1993). *Journal of Experimental Psychology: Learning, Memory, and Cognition, 19*, 718-734.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics, 5*, 115-133.

Medler, D. A., & Dawson, M. R. W. (1994a). Training redundant artificial neural networks: Imposing biology on technology. *Psychological Research, 57*, 54-62.

Medler, D. A., & Dawson, M. R. W. (1994b). Using redundancy to improve the performance of artificial neural networks. In R. Elio (Ed.) *Proceedings of the Tenth Biennial Conference of the Canadian Society for Computational Studies of Intelligence. pp. 131-138.* Palo Alto, CA: Morgan Kauffman.

Miller, E. (1989). Language impairment in Alzheimer type dementia. *Clinical Psychology Review, 9*, 181-195.

Miller, G. A., Galanter, E., & Pribram, K. H. (1960). *Plans and the structure of behavior.* New York: Henry Holt.

Minsky, M. (1975). A framework for representing knowledge. In P. H. Winston (Ed.), *The psychology of computer vision* (pp. 211-277). New York: McGraw Hill.

Minsky, M. (1985). *The society of mind.* New York: Simon & Schuster.

Minsky, M. L., & Papert, S. A. (1988/1969). *Perceptrons* (expanded ed.). Cambridge, MA: MIT Press.

Moody, J., & Darken, C. (1989). Fast learning in networks of locally tuned processing units. *Neural Computation, 1*, 281-294.

Moorhead, I. R., Haig, N. D., & Clement, R. A. (1989). An investigation of trained neural networks from a neurophysiological perspective. *Perception, 18*, 793-803.

Mozer, M. C., & Behrmann, M. (1990). On the interaction of selective attention and lexical knowledge: A connectionist account of neglect dyslexia. *Journal of Cognitive Neuroscience, 2*, 96-123.

Mozer, M. C., & Smolensky, P. (1989). Using relevance to reduce network size automatically. *Connection Science, 1*, 3-16.

Newell, A., & Simon, H. A. (1963). GPS: A program that simulates human thought. In E. A. Feigenbaum & J. Feldman (Eds.), *Computers and thought* (pp. 279-293). R. Oldenbourg KG.

Newell, A., & Simon, H. A. (1981). Computer science as empirical inquiry: Symbols and search. In J. Haugeland (Ed.), *Mind Design* (pp. 35-66). Cambridge, MA: MIT Press.

Norman, D. A. (1986). Reflections on cognition and parallel distributed processing. In J. L. McClelland, D. E. Rumelhart, & the PDP Research Group (Eds.), *Parallel Distributed Processing* (vol. 2; pp 531-546). Cambridge, MA: MIT Press.

Omlin, C., & Giles, C. L. (1996). Constructing deterministic finite-state automata in recurrent neural networks. *Journal of the ACM, 43*, 937-972.

Osherson, D. N., & Lasnik, H. (Eds.) (1990). *Language: An invitation to cognitive science* (vol. 1). Cambridge, MA: MIT Press.

Papert, S. (1988). One AI or many? *Dædalus, 117*, 1-14.

Penrose, R. (1989). *The emperor's new mind.* London: Oxford University Press.

Perry, W., Potterat, E., Auslander, L., & Kaplan, E., et al. (1996). A neuropsychological approach to the Rorschach in patients with dementia of the Alzheimer type. *Assessment, 3*, 351-363.

Pierce, W. D., & Epling, W. F. (1995). *Behavior analysis and learning*. Englewood Cliffs, NJ: Prentice Hall.

Pinker, S., & Prince, A. (1988). On language and connectionism: Analysis of a parallel distributed processing model of language acquisition. *Cognition, 28*, 73-193.

Plaut, D. C. (1995). Lesioned attractor networks as models of neuropsychological deficits. In M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks* (pp. 540-543). Cambridge, MA: MIT Press.

Plaut, D. C., & Shallice, T. (1993). Perseverative and semantic influences on visual object naming errors in optic aphasia: A connectionist account. *Journal of Cognitive Neuroscience, 5*, 89-117.

Posner, M. I. (Ed.) (1989). *Foundations of cognitive science*. Cambridge, MA: MIT Press.

Pylyshyn, Z. W. (1973). What the mind's eye tells the mind's brain: A critique of mental imagery. *Psychological Bulletin, 80*, 1-24.

Pylyshyn, Z. W. (1981). The imagery debate: Analogue media versus tacit knowledge. *Psychological Review, 87*, 16-45.

Pylyshyn, Z. W. (1984). *Computation and cognition*. Cambridge, MA: MIT Press.

Pylyshyn, Z. W. (1991). The role of cognitive architecture in theories of cognition. In K. VanLehn (Ed.), *Architectures for intelligence*. Hillsdale, NJ: Lawrence Earlbaum Associates.

Qian, N., & Sejnowski, T. J. (1988). Predicting the secondary structure of globular proteins using neural network models. *Journal of Molecular Biology, 202*, 865-884.

Quillian, M. R. (1968). Semantic memory. In M. Minsky (Ed.), *Semantic information processing* (pp. 216-260). Cambridge, MA: MIT Press.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1*, 81-106.

Rapp, B. C., & Caramazza, A. (1995). Disorders of lexical processing and the lexicon. In M. S. Gazzaniga (Ed.), *The cognitive neurosciences* (pp. 901-914). Cambridge, MA: MIT Press.

Reeke, G.N. Jr., & Edelman, G. M. (1988). Real brains and artificial intelligence. *Dædalus, 117,* 143-173.

Rescorla, R. A., & Wagner, A. R. (1972). A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. In A. H. Black and W. F. Prokasy (Eds.), *Classical conditioning II: Current research and theory* (pp. 64-69). New York: Appleton-Century-Crofts.

Richards, W. (Ed.) (1988). *Natural computation*. Cambridge, MA: MIT Press.

Ritter, H. (1995). Self-organizing feature maps: Kohonen maps. In M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks* (pp. 846-851). Cambridge, MA: MIT Press.

Robertson, L. C., Knight, R. T., Rafal, R., & Shimamura, A. P. (1993). Cognitive neuropsychology is more than single-case studies. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 19,* 710-717.

Rogers, H., Jr. (1987). Theory of recursive functions and effective computability. Cambridge, MA: MIT Press.

Roland, P. E., Kawashima, R., Gulyás, B., & O'Sullivan, B. (1995). Positron emission tomography in cognitive neuroscience: Methodological constraints, strategies, and examples from learning and memory. In M. S. Gazzaniga (Ed.), *The cognitive neurosciences* (pp. 781-788). Cambridge, MA: MIT Press.

Roman, S. (1992). *Coding and information theory*. New York: Springer-Verlag.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review, 65,* 386-408.

Rosenblatt, F. (1962). *Principles of neurodynamics*. Washington, DC: Spartan.

Rumelhart, D. E., Hinton, G. E., & McClelland, J. L. (1986). A general framework for parallel distributed processing. In J. L. McClelland, D. E. Rumelhart, & the PDP Research Group (Eds.), *Parallel Distributed Processing* (vol. 1; pp 45-76). Cambridge, MA: MIT Press.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986a). Learning internal representations by error propagation. In J. L. McClelland, D. E. Rumelhart, & the PDP Research Group (Eds.), *Parallel Distributed Processing* (vol. 1; pp 318-362). Cambridge, MA: MIT Press.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986b). Learning representations by back-propagating errors. *Nature, 323*, 533-536.

Rumelhart, D. E., & McClelland, J. L. (1985). Levels indeed! A response to Broadbent. *Journal of Experimental Psychology: General, 114*, 193-197.

Rumelhart, D. E., & McClelland, J. L. (1986). On learning the past tense of English verbs. In J. L. McClelland, D. E. Rumelhart, & the PDP Research Group (Eds.), *Parallel Distributed Processing* (vol. 2; pp 216-271). Cambridge, MA: MIT Press.

Rumelhart, D. E., & Norman, D. A. (1981). Introduction. In G. E. Hinton & J. A. Anderson (Eds.), *Parallel models of associative memory* (pp 1-7). Hillsdale, NJ: Lawrence Earlbaum Associates.

Ryle, G. (1949). *The concept of mind*. London: Hutchinson & Company.

Schlimmer, J. C. (1987). *Concept acquisition through representational adjustment* (Tech. Rep. No. 87-19). Doctoral dissertation, Department of Information and Computer Science, University of California Irvine.

Schneider, W. (1987). Connectionism: Is it a paradigm shift for psychology? *Behavior Research Methods, Instruments, & Computers, 19*, 73-83.

Searle, J. R. (1980). Minds, brains, and programs. *Behavioral and Brain Sciences, 3*, 417-424.

Searle, J. R. (1990). Is the brain's mind a computer program?. *Scientific American, 262*(1), 26-31.

Seidenberg, M. S. (1993). Connectionist models and cognitive theory. *Psychological Science, 4*, 228-235.

Sejnowski, T. J., & Rosenberg, C. R. (1986). NETtalk: A parallel network that learns to read aloud. *The John Hopkins University Electrical Engineering and Computer Science Technical Report, JHU/EECS-86/01.*

Selfridge, O. G. (1959). Pandemonium: A paradigm for learning. In D. V. Blake & A. M. Uttley (Eds.), *The mechanisation of thought processes* (pp. 511-529). London: HMSO.

Selfridge, O. G., & Neisser, U. (1960). Pattern recognition by machine. *Scientific American, 203,* 60-67.

Servan-Schreiber, D., Printz, H., & Cohen, J. D. (1990). A network model of catecholamine effects: Gain, signal-to-noise ratio, and behavior. *Science, 249,* 892-895.

Shallice, T. (1988). *From neuropsychology to mental structure.* New York: Cambridge University Press.

Shamanski, K. S., & Dawson, M. R. W. (1994). Problem type by network type interactions in the speed and transfer of connectionist learning. In B. MacDonald, R. Holte, and C. Ling (Eds.) *Proceedings of the Machine Learning Workshop at AI/GI/VI'94. pp.iv-1 - iv-7.* Calgary, AB: University of Calgary Press.

Siegelmann, H. T., & Sontag, E. D. (1994). Analog computation via neural networks. *Theoretical Computer Science, 131,* 331-360.

Skinner, B. F. (1938). *The behavior of organisms.* New York: Appleton-Century-Crofts.

Smith, E. R. (1996). *What do connectionism and social psychology offer each other?* Journal of Personality & Social Psychology, *70, 893-912.*

Smolensky, P. (1988). On the proper treatment of connectionism. *Behavioral and Brain Sciences, 11,* 1-74.

Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artiifical Intelligence, 46,* 159-216.

Spencer, H. (1910/1855). *The principles of psychology* (3rd ed.). New York: D. Appleton and Company.

Stillings, N., Feinstein, M. H., Garfield, J. L., Rissland, E. L., Rosenbaum, D. A., Weisler, S. E., & Baker-Ward, L. (1987). *Cognitive science: An introduction*. Cambridge, MA: MIT Press.

Stix, G. (1994). Bad apple picker: Can a neural network help find problem cops? *Scientific American, 271*(6), 44-46.

Storrie-Lombardi, M. C., & Lahav, O. (1995). Astronomy. In M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks* (pp. 107-110). Cambridge, MA: MIT Press.

Suddarth, S. C., & Kergosien, Y. L. (1990). Rule-injection hints as a means of improving network performance and learning time. In L. B. Almeida & C. J. Wellekens (Eds.), *Neural Networks*, Lecture Notes in Computer Science (vol. 412, pp. 120-129). New York: Springer-Verlag.

Suppes, P. (1969). Stimulus-response theory of finite automata. *Journal of Mathematical Psychology, 6*, 327-355.

Sutton, R. S., & Barto, A. G. (1981). Toward a modern theory of adaptive networks: Expectation and prediction. *Psychological Review, 88*, 135-171.

Tagaris, G. A., Kim, S-G., Strupp, J. P., Andersen, P., Uğurbil, K, & Georgopolous, A. P. (1997). Mental rotation studied by functional magnetic resonance imaging at high field (4 Tesla): Performance and cortical activation. *Journal of Cognitive Neuroscience, 9*, 419-432.

Thagard, P. (1996). *Mind: Introduction to cognitive science*. Cambridge, MA: MIT Press.

Thorburn, W. M. (1915). Occam's razor. *Mind, 24*, 287-288.

Thorburn, W. M. (1918). The myth of Occam's razor. *Mind, 27*, 345-353.

Thorndike, E. L. (1932). *The fundamentals of learning*. New York: Teachers College, Columbia University.

Thorndike, E. L. (1949). *Selected writings from a connectionist's psychology*. New York: Greenwood Press.

Tranel, D., Damasio, H., & Damasio, A. R. (1995). Double dissociation between overt and covert face recognition. *Journal of Cognitive Neuroscience, 7*, 425-432.

Tremblay, J.-P., & Sorenson, P. G. (1984). *An introduction to data structures with applications* (2nd ed.). New York: McGraw-Hill.

Tulving, E. (1972). Episodic and semantic memory. In E. Tulving & W. Donaldson (Eds.), *Organization of memory* (pp. 385-397). New York: Academic.

Turing, A. M. (1936). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, Series 2, 42*, 230-265.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind, 59*, 433-460.

Vapnik, V. N. (1982). *Estimation of dependencies based on empirical data*. Berlin: Springer-Verlag.

Vapnik, V. N. , & Chervonenkis, A. Y. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of probability and its applications, 16*, 264-280.

von Neumann, J. (1958). *The computer and the brain*. New Haven: Yale University Press.

Walker, S. F. (1990). A brief history of connectionism and its psychological implications. *AI & Society, 4*, 17-38.

Walter, J. A., & Schulten, K. (1993). Implementation of self-organizing neural networks for visuo-motor control of an industrial robot. *IEEE Transactions on Neural Networks, 4*, 86-95.

Wasserman, P. D. (1989). *Neural computing*. New York: Van Nostrand Reinhold.

Watson, J. B. (1913). Psychology as the Behaviorist views it. *Psychological Review, 20*, 158-177.

Werbos, P. J. (1995). Backpropagation: Basics and new developments. In M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks* (pp. 134-139). Cambridge, MA: MIT Press.

Wenocur, R. S., & Dudley, R. M. (1981). Some special Vapnik-Chervonenkis classes. *Discrete Mathematics, 33*, 313-318.

White, H. (1988). Economic prediction using neural networks: The case of IBM daily stock returns. *Proceedings of the IEEE International Conference on Neural Networks, San Diego 1988*, II-451 - II-459.

Widrow, B., & Hoff, M. E. (1960). Adaptive switching circuits. *1960 IRE WESCON Convention Record*, 96-104.

Widrow, B., & Lehr, M. A. (1995). Perceptrons, Adalines, and backpropagation. In M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks* (pp. 719-724). Cambridge, MA: MIT Press.

Wiener, N. (1948). *Cybernetics*. New York: John Wiley & Sons.

Winograd, S., & Cowan, J. D. (1963). *Reliable computation in the presence of noise*. Cambridge, MA: MIT Press.

Wood, C. C. (1978). Variations on a theme by Lashley: Lesion experiments on the neural model of Anderson, Silverstein, Ritz, and Jones. *Psychological Review, 85*, 582-591.

Yu, Y.-H., & Simmons, R. F. (1990). Extra output biased learning. *Proceedings of the International Joint Conference on Neural Networks (IJCNN-90)* (vol. 3, 161-166).

Zapranis, A. D., & Refenes, A. N. (1995). Investment management: Tactical asset allocation. In M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks* (pp. 491-495). Cambridge, MA: MIT Press.

Zemel, R. S. (1995). Minimum description length analysis. In M. A. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks* (pp. 572-575). Cambridge, MA: MIT Press.

Zipser, D., & Andersen, R. A. (1988). A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons. *Nature, 331*, 679-684.

Appendix A

Mushroom Traits and Attributes

Trait	Attributes
cap shape	bell, conical, convex, flat, knobbed, sunken
cap surface	fibrous, grooves, scaly, smooth
cap colour	brown, buff, cinnamon, gray, green, pink, purple, red, white, yellow
bruises?	no, yes
odour	almond, anise creosote, fishy, foul, musty, none, pungent, spicy
gill attachment	attached, descending, notched
gill spacing	close, crowded, distant
gill size	broad, narrow
gill colour	black, brown, buff, chocolate, gray, green, orange, pink, purple, red, white, yellow
stalk shape	enlarging, tapering
stalk surface above ring	fibrous, scaly, silky, smooth
stalk surface below ring	fibrous, scaly, silky, smooth
stalk colour above ring	brown, buff, cinnamon, gray, orange, pink, red, white, yellow
stalk colour below ring	brown, buff, cinnamon, gray, orange, pink, red, white, yellow
veil type	partial, universal
veil colour	brown, orange, white, yellow
ring number	none, one, two
spore print colour	black, brown, buff, chocolate, green, orange, purple, white, yellow
population	scattered, several, solitary
habitat	urban, wastes, woods

Note: Class Distribution
Edible = 4208 (51.8%),
Poisonous = 3916 (48.2%),
Total = 8124

Appendix B

Animal Names and Semantic Categories

Name	S	D^1	D^2	R	T
dog	s	d	c	n	c
cat	s	d	c	n	f
horse	l	d	h	n	e
cow	l	d	h	n	b
lion	m	w	c	a	f
tiger	m	w	c	a	f
elephant	l	w	h	a	o
pig	m	d	o	n	p
bear	l	w	o	n	o
mouse	s	d	o	n	r
rat	s	d	o	n	r
deer	m	w	h	n	C
sheep	m	d	h	n	p
giraffe	l	w	h	a	o
goat	m	d	o	n	p
zebra	l	w	h	a	e
squirrel	s	w	h	n	r
wolf	m	w	c	n	c
donkey	l	d	h	n	e
rabbit	s	d	h	n	r
leopard	m	w	c	a	f

Name	S	D^1	D^2	R	T
mule	l	d	h	n	e
fox	s	w	c	n	c
bull	l	d	h	n	b
buffalo	l	w	h	n	b
moose	l	w	h	n	C
rhinoceros	l	w	h	a	o
camel	l	w	h	a	o
antelope	m	w	h	a	C
hippopotamus	l	w	h	a	o
lamb	s	d	h	n	p
monkey	s	w	o	a	o
raccoon	s	w	o	n	r
panther	m	w	c	n	f
llama	l	d	h	n	o
skunk	s	w	o	n	r
cheetah	m	w	c	a	f
jaguar	m	w	c	n	f
beaver	s	w	h	n	r
gazelle	m	w	h	a	C
turtle	s	w	o	n	o
elk	m	w	h	n	C

Note: (S)= (s)mall, (m)edium, (l)arge; (D^1)omesticity = (d)omestic, (w)ild; (D^2)iet = (o)mnivore, (c)arnivore, (h)erbivore; (R)ange = (a)frica, (n)orth america; (T)ype = (c)anine, (f)eline, (e)quine, (b)ovine, (p)orcine/ovine, (r)odent, (C)ervidea, (o)ther.

Appendix C

Animal Names Recalled For Each Normal Subject

Sub 1		Sub 2		Sub 3		Sub 4		Sub 5	
Cycle	Name	Cycle	Name	Cycle	Name	Cycle	Name	Cycle	Name
23	horse	21	horse	24	dog	22	dog	24	cow
30	donkey	31	bull	28	cat	32	rabbit	28	horse
39	bull	40	donkey	33	mouse	35	lamb	32	donkey
47	buffalo	47	buffalo	41	rabbit	39	mouse	39	mule
57	cow	63	sheep	44	rat	49	beaver	42	bull
64	zebra	86	zebra	50	bull	54	llama	46	llama
72	lamb	105	lamb	61	mule	59	donkey	62	buffalo
85	moose	123	moose	69	donkey	63	horse	73	moose
100	sheep	141	rabbit	77	llama	69	cow	89	rabbit
117	cat	159	sheep	90	cow	81	bull	95	sheep
131	giraffe			96	buffalo	91	sheep	112	lamb
137	camel			106	sheep	99	mule	125	dog
145	rhino			156	moose	110	donkey	136	zebra
147	elephant			166	zebra	117	llama	142	moose
164	bear					125	moose	164	beaver
						130	buffalo	176	camel
						141	horse		
						154	zebra		
						166	sheep		

Sub 6		Sub 7		Sub 8		Sub 9		Sub 10	
Cycle	Name	Cycle	Name	Cycle	Name	Cycle	Name	Cycle	Name
23	horse	9	dog	22	cow	23	dog	24	horse
34	bull	22	horse	34	mule	32	beaver	36	llama
40	donkey	29	mule	38	donkey	35	mule	42	bull
42	buffalo	35	bull	40	buffalo	41	donkey	49	donkey
60	llama	45	cow	47	bull	50	rabbit	68	moose
69	lamb	48	buffalo	58	sheep	60	rat	79	buffalo
89	moose	73	zebra	79	zebra	69	cow	84	lamb
110	sheep	79	hippo	93	llama	79	llama	98	sheep
123	llama	92	moose	110	lamb	84	bull	111	cat
137	lamb	106	elephant	126	rabbit	93	lamb	123	rabbit
151	rabbit	113	giraffe	141	zebra	97	sheep	133	lamb
159	zebra	116	rhino	173	pig	105	beaver	138	rat
175	antelope	119	hippo			115	donkey	146	beaver
		139	monkey			126	horse	157	mouse
		160	gazelle			132	zebra	161	squirrel
						146	cow	173	raccoon
						157	buffalo		
						165	llama		
						169	sheep		

Appendix D

Animal Names Recalled For AD Subjects With 50% Damage

Sub 1		Sub 2		Sub 3		Sub 4		Sub 5	
Cycle	Name	Cycle	Name	Cycle	Name	Cycle	Name	Cycle	Name
31	cat	33	elephant	34	dog	32	cow	32	cow
39	bull	48	squirrel	53	rat	36	horse	54	camel
54	lamb	55	camel	67	mouse	46	llama	65	horse
67	donkey	60	giraffe	78	dog	65	cow	75	cow
74	buffalo	71	elephant	91	rat	79	donkey	96	camel
80	bull	88	camel	107	rabbit	90	llama	116	cow
98	horse	100	hippo	121	dog	100	horse	137	camel
108	donkey	109	rhino	128	rat	109	cow	157	cow
121	bull	118	elephant	144	skunk	119	donkey		
151	donkey	123	camel	154	rabbit	127	llama		
162	bull	133	zebra	167	rat	132	horse		
		140	rhino			149	cow		
		152	hippo			161	horse		
		163	elephant			167	llama		

Sub 6		Sub 7		Sub 8		Sub 9		Sub 10	
Cycle	Name	Cycle	Name	Cycle	Name	Cycle	Name	Cycle	Name
27	horse	30	horse	34	horse	34	cow	31	cat
47	buffalo	42	mule	42	sheep	49	bull	43	sheep
57	elk	59	cow	56	lamb	69	mule	70	donkey
66	llama	68	buffalo	76	llama	93	cow	89	llama
70	horse	86	mule	87	horse	105	bull	97	sheep
86	elephant	100	cow	101	sheep	116	mule	113	horse
92	giraffe	107	buffalo	119	mule	128	horse	121	llama
112	hippo	125	mule	132	llama	140	cow	126	bull
127	buffalo	140	buffalo	142	horse	153	bull	129	cow
148	hippo	146	donkey	164	rabbit	161	mule	147	sheep
167	giraffe	167	mule			176	llama	163	llama
175	moose	176	buffalo			178	horse	173	cow

IMAGE EVALUATION TEST TARGET (QA-3)