UNIVERSITY OF CALIFORNIA

Irvine

A Neural Model of Adaptive Behavior

A dissertation submitted in partial satisfaction of the

requirements for the degree Doctor of Philosophy

in Information and Computer Science

by

Steven Edward Hampson

Committee in charge:

Professor Dennis Kibler, Chair

Professor Mary-Louise Kean

Professor Scott Huddleston

1983

The dissertation of Steven Edward Hampson is approved,

and is acceptable in quality and form for

publication on microfilm:

Mary Louise Kean

Scott Huddleston

Dennis F. Kibler

Committee Chair

University of California, Irvine

1983

ACKNOWLEDGMENTS

I gratefully acknowledge the suggestions, encouragement and tolerance of my committee members Dennis Kibler, Mary-Louise Kean and Scott Huddleston. In addition, Dennis Volper, Ira Baxter and John Justeson took the time to read early drafts. Their comments and suggestions were also very useful. Final proof reading by Pat Harris and my mother, Luella Hampson, was equally indispensable and greatly appreciated. And certainly, words cannot express my gratitude to Kathy Harper for drawing the figures used in the text.

Finally, thanks are still due to Dr. Robert Loomis for providing an ideal introduction to research some years ago.

TABLE OF CONTENTS

LIST OF FIGURES

1.0 WHY NEURAL MODELS?

The obvious success of the modern computer has made it an attractive metaphor for general information processing. In particular, a symbol manipulation capability is often viewed as the logical underpinning of intelligent behavior (Newell 80). The symbol manipulation model is in close agreement with the actual mechanism of computer processing, but its distinction between an active processor and passive data may make it an inappropriate model for biological information processing (Anderson and Hinton 81). Though logically adequate, it does not reflect the pragmatic constraints on information processing, such as limitations in time or space. Symbol manipulation is an abstract model, and "does not attempt to be precise enough to deal with such quibbles" (Newell 80). Only when dealing directly with such constraints are specific biological analogies apt to be recognized.

The purpose of this model is not the simulation of specific biological systems, but implementation of general processes underlying intelligent behavior using a neuron-like element as the basic building block. Biologically motivated mechanisms for pattern storage, recognition and learning are explored, with emphasis on achieving maximum parallelism. Certain aspects of goal-directed behavior are also investigated. Recent neurophysiological research has demonstrated the existence of neural circuits that can be interpreted as performing such functions. These biological models provide important insight concerning possible approaches.

1.1 Parallel Processing

The field of artificial intelligence (AI) has had its successes, but it has also had its obvious failures. The sensory-motor skills of the common laboratory rat are still unapproached despite the immediate need for such capabilities in industrial robots. In such cases, careful attention to what is known of biological intelligence may be useful in designing similar artificially intelligent systems. Although the physical mechanisms of information processing are different (e.g., transmitter modulated ion channels vs. transistor flip-flops), higher level functional similarities may be appropriate. In particular, neural systems may prove very useful as models of large scale parallelism (Kent 81).

Most useful applications of AI require large amounts of computation. This may be because AI programs are uniformly inefficient, but it more likely reflects the true amount of computation needed for AI problems. Fahlman (79) has argued that intersection of partially specified sets is an important process of intelligence which, in general, cannot be implemented efficiently. Because partial or best match pattern matching subsumes subgraph monomorphism, which is an NP-complete problem, Hayes-Roth (78) has concluded that many AI applications which rely on pattern-matching are inherently computationally expensive.

Since programs are executed sequentially, efficiency is generally measured in units proportional to CPU run time. However, advances in LSI technology have made possible a radical change in computer architecture through the utilization of parallel processors. Given this capability, it is possible to explore techniques that minimize reaction time (input to output) rather than the total amount of computation. This should permit qualitatively different, and more biologically oriented approaches to many problems.

For example, in detecting members of a particular category, (e.g., a plant species), key features may not exist which distinguish individuals of that group from members of closely related groups. In the case of plant species, they may actually interbreed, producing intermediate progeny. The only way to describe these categories is as a probabilistic distribution of all features rather than the presence or absence of any particular one. Thus, at a minimum, sequential processing time is proportional to the number of features. As an alternative to a sequential decision tree approach, the input might be exhaustively compared to all group descriptions in one parallel step, and the best match selected (e.g., Selfridge 59). Obviously this is computationally expensive, but if the processes of comparison and best match selection can be done in unit time, it can be done very quickly. Such an approach can be used to implement many of the desirable characteristics of biological memory such as associative or content addressable access (Kohonen 77,80).

In fact, there may be fundamental limitations to the utility of sequential computation since the use of a single CPU places a practical upper limit on the rate of computation possible, a limit which may prove fatal to many real-time AI applications. For example, the human retina contains about 100 million cells, each of which can integrate hundreds of inputs (in parallel) and respond hundreds of times a second. This amount of raw computing power may be forever beyond the reach of a single CPU operating within the laws of physics. Even a single Purkinje cell has about 10 ** 5 inputs and can respond about 10 ** 2 times a second, giving on the order of 10 ** 7 operations (multiplications) per second, a number above the range of most computers.

To use parallel processing effectively, the necessary computations must be decomposable. Unfortunately, programs written in current computer languages for current computers seldom lend themselves to large-scale decomposition. It is likely that the computational medium has shaped the programming methods and the domains of application. An alternative to attempting to decompose existing programs is to reapproach the problems with the specific intention of exploiting the computational potential of parallel processing. In particular, pattern manipulation is an easily decomposable process. Knowledge representation, utilization and learning can be viewed as forms of pattern processing, so AI may be an ideal area in which to intentionally explore the applications of parallel computation. Many problems in AI might be effectively

approached in this manner if:

"The fundamental problem of understanding intelligence is not the identification of a few powerful techniques, but rather the question of how to represent large amounts of knowledge in a fashion that permits their effective use and interaction." (Goldstein and Papert 77)

It is possible that an over emphasis on sequential, high level processes has obscured the importance of low level, parallel processing. This point has been made by James Albus:

"An obvious but seldom recognized fact is that planning is not characteristic of the behavior of most biological organisms. ... The rarity and late arrival of the ability to plan suggests that a highly developed precursor, or substrate was required from which planning capabilities evolved. ... The implication is that a sensory-interactive, goal-directed motor system is not simply an appendage to the intellect, but is rather the substrate in which intelligence evolved." (Albus 79)

High level processes such as natural language processing are attractive problems, but it should not be overlooked that the underlying mechanisms of a rat's behavior are equally enigmatic.

1.2 Intelligent Behavior

AI production systems capture the spirit of distributed (though not necessarily parallel) processing, and have had reasonable success in modeling intelligent behavior (Waterman and Hayes-Roth 78). Specifically, behavior is broken down into a sequence of operator applications. On the basis of the system's current goals, the current state of the environment, and past experience, a heuristically best guess of the correct operator can be made. Though this information is usually not all represented explicitly in production systems, it could

be without changing the nature of their operation. Because all information can be brought to bear on the selection of each operator, behavior can be viewed as a network of alternative actions, rather than a collection of precanned sequences. This multiplicity of decision points is important in avoiding rigid, stereotypic behavior.

Stimulus-response behavior in biological organisms can be viewed from the same perspective. All organisms have receptors to sense current conditions (analogous to antecedents in production rules), and mechanisms by which they can effect changes in themselves and their surroundings (analogous to production rule consequences). In primitive organisms, a sensory cell may connect directly to a motor cell. This results in a rigid behavioral system such that the particular stimulus is always associated with a particular response. If such a response is adaptive, as in withdrawal from a noxious stimulus, the organism might be said to have made an intelligent response to its environment. In more advanced organisms, one or two interneurons may be inserted into the pathway. If these cells receive information from other sensors, such as a food detector, the advantages of leaving can interact with the advantages of staying before the final motor cell is triggered. This response strategy is more flexible and clearly capable of more "intelligent" responses to the environment.

The evolution of biological intelligence can be viewed as the progressive growth and specialization of a net of interneurons between sensory and motor cells; in general, the thicker the net, the greater

the opportunity for analysis of the environment before a response is triggered. Such a network is an example of distributed pattern storage and recognition. The system is a completely distributed intelligence since there is no "central executive" in it, and "memory" is not something to be searched but is simply the modifiable pathway between input and output. The behavioral systems of many lower organisms provide reasonably well understood examples of such distributed control (Kandel 76, 79a).

This neural structure can be applied to production systems. If the left-hand sides of rules were implemented in a network of pattern detectors, and right-hand sides as operators, the analogy would be very close. If the production system displayed intelligent behavior, an explicitly distributed version of it also would, assuming best fit rule selection was an acceptable strategy. This would be an example of a meaningfully intelligent system made up of individual elements, each of which has little more processing power than a neuron.

1.3 Learning

The capacity for learning is an important aspect of any intelligent system which can be separated from the processes of overt behavior. This is illustrated biologically by people with certain amnestic syndromes; they are unable to commit new information to permanent memory, yet they can display intelligent behavior (Milner 70). Most production systems are in a comparable condition because

their capacity for intelligent behavior is fixed at the time of creation. For this reason, a great deal of effort is often expended in discovering the right set of rules. This has generated considerable interest in constructing systems that can create their own productions, or at least add to a long-term store of useful facts. In current AI models, any learning capabilities are typically dependent on the action of an omnipotent central executive (e.g., Lenat 77) rather than low level, local adjustments (e.g., Holland and Reitman 78).

The tunability of neural models is well established (Nilsson 65), which demonstrates that at least some aspects of learning can be implemented in low level, neural-like structures. In the invertebrate Aplysia, such distributed learning has been extensively investigated (Kandel 79ab). At the cellular level, several physiologically distinct processes of learning have been described, and all have direct expression in the overt behavior. Neural learning processes have been observed in a wide range of organisms (Woody 82c).

Most formal neural learning algorithms are proposed because they are simple and/or biologically plausible. It is then necessary to discover and prove what can be learned using that process. However, the effects of even very simple rules are often complex and difficult to analyze (e.g., Amari 77a, Amari and Takeuchi 78). (Even without learning, the analysis of neural nets can be quite complex. e.g., Amari 77b, Kishimoto and Amari 79). While these models can be mathematically challenging, they are of little biological value if the

initial assumptions are biologically incorrect, and of questionable value to AI if the behaviors produced are not powerful enough to address AI problems.

There are a large number of learning algorithm variations, and there is as yet no comprehensive theory of biological learning, so a specifically targeted approach is taken here. Desired system behavior is identified, and mechanisms proposed to implement it. The proposed model has the general characteristics of a production system, so it should be capable of interesting behavior, and though it is not developed as a strictly biological model, its central principles are intended to be consistent with biological capabilities.

The view taken here is that the brain is an inherently structured system, and that intelligence is not simply an emergent property of large groups of neurons. It is apparent that there are specific functions a brain must perform, and there are specific, hard-wired structures to perform them. Consequently, an important aspect of this study is an attempt to identify primitive processes underlying intelligent behavior and to implement them with neural networks, assuming specific, hard-wired systems as necessary. For example, it has been suggested that there are specialized plasticity controlling systems which control the modification of synaptic connections (Krasne 78, Feldman 81, Kety 82, Kasamatsu 83).

In the development of the model, a fairly general characterization of intelligence has been adopted: goal-directed behavior which takes into consideration current conditions and past experience. This is sufficiently general so that it does not conflict with other definitions or examples of intelligence, but suggests the important processes which must be considered: goal-seeking, knowledge representation, pattern-matching and learning. Although enormous amounts of effort have been expended tackling these problems with the hope of achieving human-like performance, such an optimistic goal need not be set immediately. It is often easier to analyze highly complex systems when similar but simpler ones are already understood.

This characterization of intelligence is equally applicable to all animate behavior, so it is possible to look for the most general principles in the simplest organisms. This can be pursued to the cellular level, since observable behavior in many simple organisms can be traced to the firing of individual neurons. In a simple invertebrate, an identifiable cell may trigger ingestion, another might measure gut distension, and an array of others serve as environmental sensors. More complex behavior may be organized along similar lines, though with orders of magnitude increase in the number of cells.

1.4 Expected Benefits

The increased computational power of parallel processing may be necessary for many real-time AI applications, such as those which must continually respond or adjust to a rapidly changing, complex environment. A fundamental problem is that external reality can be much more complex and variable than its internal representation. Ideally, the internal representation would constantly reflect all pertinent external conditions, and the decision making process would continuously adjust to this information.

Distributed memory models based on neuron-like elements display many of the desirable characteristics of biological memory, being content addressable and potentially reconstructive (Hinton and Anderson 81). Another potential benefit of distributed representation and processing is a fail-soft capability. Biological organisms are remarkable in their ability to function despite considerable damage (Lashley 29), and the human brain is no exception (Lorber 80).

Another fundamental reason for this type of simulation is that analytic description of behavior resulting from the interaction of large numbers of variables over time can be effectively intractable. It is clear that macroscopically sophisticated behavior can result from the interaction of microscopically simple processes, but the relationship may be too complex for analytic description. In many cases simulation is necessary to demonstrate the relationship between the two.

Finally, the processes being explored are biologically motivated and implemented using a neuron-like element, so there should be similarities in implementation in the artificial and natural systems. By observing how microscopic dysfunction affects macroscopic behavior in a controllable artificial system, it may be possible to gain insight into pathologies of biological information processing.

1.5 The Model

A neural model of adaptive behavior is developed. It is intended to have the general properties of a production system. In particular, it is intended to support arbitrary input-output (stimulus-response) mappings in the Boolean domain. Behavioral completeness requires that any mapping be possible, and learning completeness requires that any mapping be learnable.

In chapter II, the gill withdrawal reflex of Aplysia is described and analyzed as a neural model of adaptive operator application. In chapter III, a model neuron is developed and a useful family of computable functions is described. In chapter IV, a learning algorithm is developed to train a single node (model neuron). Assembly structure and learning controls appropriate for a single operator are introduced in chapter V. Functional completeness in this case is the ability of an operator to compute, and learn to compute, any Boolean function. In chapter VI multiple operators are combined into a Boolean behavioral system. A common memory with appropriate

learning characteristics is introduced. Completeness for multiple operators requires that any input be potentially able to trigger any set of operators. Simple sensory-motor behavior is analyzed in terms of the model and some psychological aspects of goal directed behavior. In chapter VII, an evaluation system is developed. Evaluation is used to train the application of operators and to identify behaviorally relevant input patterns. Learned evaluation is introduced as a method of learning sequences of actions. Chapter VIII contains a summary and discussion of the overall model.

Terms such as teaching, evaluation, reward, motivation, reinforcement and feedback are often confused in the literature, a tradition which has been observed and perhaps advanced here. However, evaluation, learning and behavioral situations will be mechanistically defined, hopefully avoiding some of the confusion caused by the imprecise connotations of these terms.

2.0 GILL WITHDRAWAL

The gill withdrawal reflex of Aplysia is an example of purposeful, adaptive behavior (Kandel 79ab). It is useful as a simple neural model of intelligent operator application since the basic purpose can be inferred and the mechanisms of implementation examined. The computer model developed here is more elaborate, but is based on observations and interpretations of gill withdrawal discussed in this chapter.

2.1 The Biological Model

Behaviorally, the gill withdrawal reflex is quite simple. If the animal is poked, the gill is withdrawn. This response can be habituated by continual poking (jets of water from a water pic in the actual experiments) so that the reflex is completely suppressed. The response returns, however, if the animal is "hurt" (electrical stimulation). The response is especially facilitated if the animal is poked prior to being hurt (Carew et al. 83). Thus the response displays both associative and non-associative conditioning.

At the risk of being teleological, the purpose of this system can be interpreted as a protective withdrawal from possible trauma. That is, since the animal may be hurt after initial contact with a foreign object, it is adaptive to withdraw the delicate gill at the first signs of danger. If the animal is continually poked with no deleterious results, perhaps by a piece of seaweed, the response

should be suppressed, (habituated), so that the animal isn't continually holding its breath. If the animal is suddenly hurt, a reasonable strategy would be to reinstate (sensitize) the response until it could be safely habituated again. Sensitization is especially reasonable if the painful experience was immediately preceded by being poked.

This may be taking liberties in identifying the purpose of Aplysia behavior, but the observed functioning of the system is consistent with this interpretation. In any case, it is not unreasonable to investigate it as a mechanism which fulfills this purpose, even if the true biological significance of the system is debatable.

Similar behavior might be desired in a hypothetical crash protector for a computer system. If a system goes down, it should have done a number of things in order to minimize the damage. A "core withdrawal reflex" might write crucial information from core to disk at the first signs of impending danger. Adaptive behavior would include habituation and associative sensitization.

In Aplysia, the neural processes responsible for this behavior are reasonably well understood. The actual biochemical mechanisms aren't particularly enlightening (except to biochemists), but the abstract functions of the various elements of the system are of general interest. As shown in Figure 2.1, this system can be reduced to 4 elements: a poke detector, a pain detector, a motor neuron and a

muscle cell. (The actual system is somewhat more complex). Poking the animal fires the poke detector. This in turn fires the motor neuron which causes the muscle to contract, withdrawing the gill. In the absence of any activity in the pain detector, the synapse between the poke detector and the motor cell progressively loses its effectiveness in firing the motor neuron. Eventually it becomes totally ineffective. This is the physiological basis of gill withdrawal habituation. If the pain detector fires, the synapse is reactivated, restoring the reflex. Thus the action of the entire system can be explained by the variable strength of a single synapse.

Gill withdrawal also displays what might be called long and short term memory (Kandel 77, 79ac). The terminology is for convenience, not to suggest a particular relationship with memory characteristics of higher animals. The effects of both habituation and sensitization may be either short term or long term. If training is for a short period, its effects wear off and the synapse returns to its previous state. If training is for a longer period, the effects become fixed. This gives the system a capability for short term, episodic adaptation, and a certain amount of noise rejection.

2.2 Principles And Interpretations

A number of important principles can be observed in this simple system:

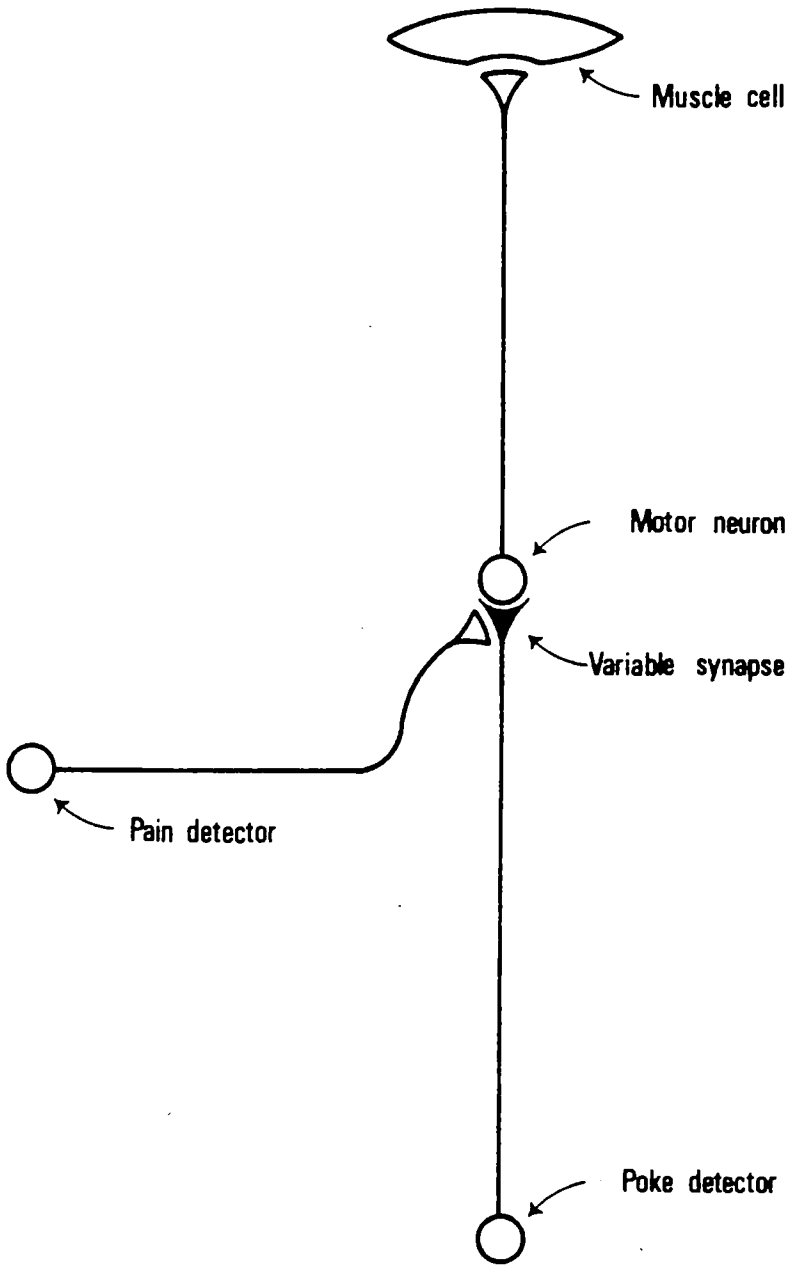


Figure 2.1 Aplysia gill withdrawal system (simplified).

* The medium is the message. When a cell fires, it sends a pulse down its axon. The pulse is indistinguishable from any other pulse on any other axon. What gives it its meaning is which axon it is on. There are no informative tokens passed. Such "labeled line" or "place coding" of information underlies most physiological theories of mental and behavioral processes (Kandel 76, 77, 79c, Gallistel 80, Hebb 49, 80, Barlow 72, Feldman 81), though it is questioned by others (John 76, 80, John and Schwartz 78).

* The functioning of the system is goal oriented, being designed to correctly apply the "withdraw gill" operator. It may be difficult to precisely identify the purpose of a complex biological system, but the goals of simple ones can be guessed at, and the goals of artificial systems can be precisely formulated. Only by identifying the purpose of a system can its performance be evaluated.

* The gill withdrawal operator is a pattern detector. In this simplified system there is only one synapse onto the motor neuron, which corresponds to the output of the poke detector. However, one could add a vision detector's output to the motor neuron's input so the animal would have two input features on which to base its decision. Their effects could both be regulated by the pain detector. The pattern the withdrawal operator should respond to is one of impending danger.

* The "purpose" of individual neurons is determined by the system(s) they function in. Since survival of the organism is a crucial constraint, specialized systems have evolved to maximize that property. Just as successful metabolism relies on specialized structures (heart, lungs, kidneys etc.), successful behavior requires specific information processing systems. The nervous system is highly structured both in terms of visible anatomy and connections (Shepherd 79), and less observable transmitter systems (Cotman and McGaugh 80 ch. 6, Shepherd 83 ch. 9, 25). Since a system's function constrains the properties of its parts, it makes sense to view individual neurons as parts of larger systems. In the development of the model, desired system behavior is identified and implemented. The properties of the component parts are constrained by these system goals.

* The information a neuron receives is not homogeneous. In the gill withdrawal reflex, the motor neuron receives information about the current state of things from the poke detector. The pain detector provides a specialized type of instructive input which tells the motor neuron when it should have fired. The two types of input are distinguished both functionally and physiologically. Systems utilizing current conditions and instruction are sometimes called feedback and feedforward systems (Cotman and McGaugh 80 ch. 11). Feedback systems are essentially servomechanisms which compare a desired value with an actual value and adjust their output appropriately if the two are different. No learning takes place. A furnace thermostat is an example of this. A feedforward system

receives information indicating how correct its output was and adjusts its output function accordingly.

* The poke detector does not "control" the motor neuron. Many models of biological motor activity rely on metaphors such as "module A outputs a command to do X", or "system A controls system B". This implies that a cell or module has selective control over what effects its output produces. A tempting explanation of this is that module A somehow notices that module B effects something it wants done. Module A therefore establishes control over that system so it can be fired (or inhibited) when appropriate. The result is that output from A "controls" B. An alternative view is that B receives information as to when it should have fired. It then notices that A has predictive potential of this and makes the appropriate adjustments. The final result is the same, but the learning process is conceptually different.

2.3 Improved Aplysia

With these interpretations, the Aplysia model can be formalized somewhat. Instead of the poke detector, there can be an array of size, shape and smell sensors, and in place of gill withdrawal, any avoidance operator can be substituted. Pain still seems a good measure of things to avoid, but it could be elaborated with an array of detectors for various unpleasant conditions. A small modification in the pathway will also be made, which makes it much easier to model

(and draw). The pain detector will directly inform the motor neuron when it should have fired. The motor neuron can then adjust its own input synapses rather than having the pain detector do it. This reduces the model's biological validity somewhat since it confounds pre- and post synaptic adjustment, but if the purpose of the biological system is correctly interpreted, the functional result of synaptic modification should be similar in the model and natural system.

The purpose of this more general system (Fig. 2.2) is still the same: learning to avoid damage to the organism. As Thorndike put it, "He who learns and runs away, lives to learn another day" (Thorndike 99). The pain detector tells the motor neuron when it should have fired, and in the absence of that signal the motor neuron assumes it shouldn't have. A problem with this arrangement is that completely avoiding a predator produces no pain, so the response habituates. However, the alternative of always bolting and never waiting to see what is really dangerous has its own drawbacks.

The behavior of this improved Aplysia can be quite flexible. If a certain organism with shape Sh1, size Sz3, and smells Sm4 and Sm7 were to bite the Aplysia, the pain detector would tell the motor neuron it should have fired. The motor neuron should then adjust its synaptic weights so that any future occurrence of Sh1, Sz3, Sm4 and Sm7 would cause it to fire. The motor neuron would act as an AND gate for those features. It might also be that there is some variability in that type of predator so that a family resemblance should be

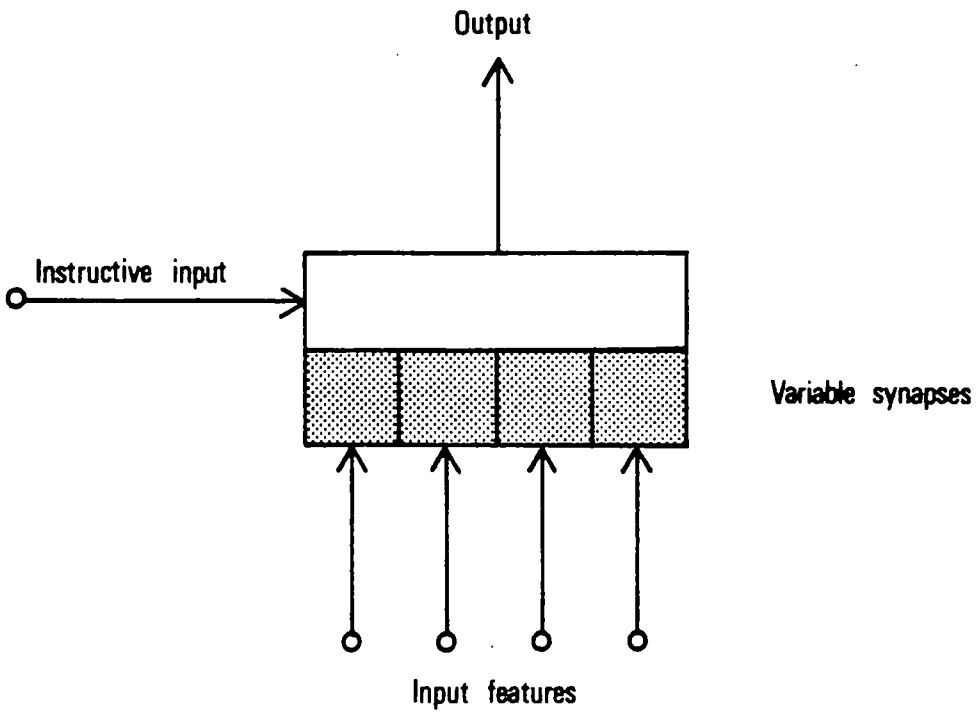


Figure 2.2 Formalized motor neuron.

sufficient reason to trigger avoidance, say at least 2 of the 4 features. At the other extreme, there may be several distinct types of predators with smells Sm9, Sm10 and Sm11. The motor neuron should then function as an OR gate for those input features.

Empirical evidence as to the actual computational capabilities of neurons is scarce, so it may be informative to investigate the theoretical characteristics of model neurons. The models may be inaccurate, but their capabilities and limitations are usually well defined. By attempting to build functional systems, some constraints on the necessary (or sufficient) characteristics of biological neurons may be surmised.

3.0 A MODEL NEURON

3.1 The Goal

A model neuron is developed in this chapter. It is compatible with the pattern detection capabilities attributed to the motor neuron in the gill withdrawal reflex. Whether this emphasis is biologically correct may be debatable, but it is consistent with the observed behavior of gill withdrawal and permits nodes to be assembled into larger and more powerful networks.

The type of output a neuron theoretically produces depends on the type of neural model used. Although most formal models focus on neurons as binary pattern classifiers (Nilsson 65), it is possible to describe more general capabilities. Since a neuron varies its output in response to its inputs, it can be thought of as computing a function. The inclusion of memory permits more elaborate models (e.g., the neuron as a finite-state machine), but the model developed here is designed to implement a trainable function.

For parsimony, neural assemblies (chapters 5, 6 and 7) will be constructed with a single type of trainable node - a sort of abstract pyramidal cell. The constraints on its behavior are straightforward: if a single node cannot compute a necessary function, then an assembly of nodes must be able to. For example, model neurons capable of computing OR and AND can be assembled into a network capable of computing any Boolean function, since all Boolean functions can be represented in disjunctive (OR of ANDs) or conjunctive (AND of ORs)

normal form. If the desired output is limited to Boolean functions, such a model is formally complete, though not necessarily efficient. Intuitively, the number of nodes needed to compute a complex function is inversely proportional to the complexity of the individual nodes.

3.2 The General Function

The McCulloch-Pitts "formal neuron" was proposed in 1943 (McCulloch and Pitts 43) and is still in many ways the generic neuron model. Many of its assumptions and limitations are present in current models. One such limitation common to almost all neural models (which is perpetuated here) is that the function computed by a node is of the form:

$$\text{Out} := F(\sum Fi(Xi))$$

That is, the output, Out, is computed as a function, F, of the sum of the functions, Fi, of the individual inputs, Xi. While this is a rather strong theoretical limitation, it works reasonably well in practice since it can compute a wide range of useful functions (including OR and AND) and is easy to implement.

3.2.1 Summation Of Inputs - Simple summation is justifiable to some extent as an application of conditional probability, and also has some biological validity. Summation corresponds precisely to a Bayesian statistical model if the probability that a node should fire can be conditioned on a set of exhaustive and mutually exclusive occurrences

(Hunt 75). If [ABi] is the probability (frequency) that A and Bi occur together, A = the node, Bi = an input feature, then [ABi]/[Bi] is the probability that A will occur, given that Bi has. This probability times the current input probability of Bi can be summed over all Bi to give the total probability that A should fire. However, since a given collection of input features is most likely neither exhaustive nor mutually exclusive, a rigorous justification on these grounds is difficult.

The biological justification is that the decision to fire a neuron is made (ideally) in one place only, the base of the axon, where all dendritic inputs are summed. This is a common simplification of the process of neural integration. The actual process is more complex (Shepherd 79 ch. 3, 83 ch. 8, Kandel and Schwartz 81) and is still poorly understood, but at a minimum is presumably capable of (something like) simple summation.

One very desirable characteristic of summation is that it can be implemented as a one step operation. For example, multiple current sources can add their contribution to a final summed current, independent of other current sources. Thus it is possible to sum the contributions of any number of features in a constant amount of time.

3.2.2 Input Functions - Limitations on the input functions, Fi(Xi), also constrain the functional capabilities of the node. A common limitation is that the input functions are restricted to a single

weight times the input.

$$Fi(Xi) := Wi * Xi$$

The weight, Wi, is meant to model the strength of a single synaptic connection. Most proofs of neural model behavior are for functions of this form. For an input of 0, the result is necessarily 0. If Xi is limited to between 0 and 1, the input function describes a line from the origin (0,0) to (1,Wi). In the proposed model this limitation is modified somewhat. Input is limited to between -1 and 1, and the input function describes a line from (-1,Ni) to (0,0) and from (0,0) to (1,Pi). Both weights Ni and Pi are trainable. (The reason for those particular limits will be discussed shortly.)

While it can be useful for conceptual purposes to view this as a single function, it is computed as two separate, single weight functions.

$$Fi(Xi) := \tilde{}Xi * Ni + Xi * Pi$$

Thus this more complex input function can be computed with single weight functions, provided that both the signal Xi and its complement ~Xi are available as inputs. (The relationship of ~Xi to Xi will also be considered shortly.) Among other things, this allows patterns to be defined on the basis of missing features as well as present ones. This is a simple extension of the "formal neuron" and all proofs involving single-weight functions are still applicable.

Biological synapses may be either excitatory or inhibitory, although a single synapse is most likely fixed as to its type. Since it is unnecessary for an input to be both excitatory and inhibitory

simultaneously, in this model the range of individual synaptic weights is allowed to vary from negative (inhibitory) to positive (excitatory) so that one synapse can do the work of two.

A further increase in capabilities could be made if the input functions were not required to be linear. For binary input features which are necessarily either present or absent, a linear function is reasonable since the extremes of the function are associated with the extremes of the input value. Intermediate input represents the probability of the feature's presence, and a linear function gives intermediate output. However, for non-binary features, intermediate input need not represent probability, but may represent the true magnitude of the stimulus intensity. Successful behavior might require different responses to different intensities. Linear input functions allow a node to respond maximally to the extreme values of an input, but not to intermediate ones. For that case, a parabola or any inverted U-shaped function would be appropriate. Obviously, more complex input functions, Fi, allow a node to compute a more complex output function.

3.2.3 Unknown - The ability to distinguish the value "unknown" from other values is of considerable importance. If an object is described as having feature A, this does not imply that all other features are missing, they are only unspecified. If input (and output) values are limited to between 0 and 1, and interpreted as 0 and 100% probability, there is no place on the scale which represents unknown. This is a

problem inherent in any implementation of the law of the excluded middle. Many logic applications confound false and unknown (e.g., Prolog).

This could be avoided by using two nodes, one to represent Xi and another to represent ~Xi. Unknown would be represented by a summed contribution of 0 (treating ~Xi as a negative number). An alternative arrangement using only a single node is to establish a baseline floating level of output and express the probability of presence or absence as variations above and below that level. This appears to be biologically common (Sejnowsi 81), and because it requires half the number of nodes, that approach is the one developed here. Input and output are constrained to be between -1 (certainly not present) and 1 (certainly present). Unknown is represented as the non-excluded middle value of 0. A similar logic strategy is used in the Mycin system (Shortliffe 75).

With this representation, "no information" and "conflicting information" are confounded since both can produce a zero output. If Xi and ~Xi were represented separately, that problem could be avoided, but such a capability didn't seem especially useful, so single node representation was acceptable.

3.2.4 The Final Function F - The nature of the final function, F, of the summed input signals depends on the type of output desired. If a threshold pattern detector is needed, output may be limited to 0 or 1.

If probabilistic output is desired, output may be restricted to between 0 and 1. For a completely general function, an unbounded range would be necessary. In addition, there is no reason output has to vary linearly with summed input. Biologically, neural response seems to vary with the logarithm (or as a power function) of input intensity (Shepherd 83 pg. 198), permitting a wider range of intensity to be represented within fixed limits. In the current model, a linear range of -1 to 1 is implemented. Values outside of that range are simply interpreted as the appropriate limit.

Most pattern detectors have a single threshold value which the summed inputs must exceed in order to fire the node at all, and above which it fires at full intensity. Besides any computational advantages this strategy may have, it is a reasonable approximation of neural behavior. Textbook neurons do respond in an all-or-nothing fashion. However, this does not mean that magnitude information is not conveyed. Input is also temporally summed, so that a strong input produces a higher frequency of firing than a weaker input. Firing frequency is commonly interpreted as a magnitude measure (Barlow 72). This style of expressing magnitude information may have some theoretical advantages, but it may also simply reflect biological limitations inherent to neurons. There did not appear to be any obvious benefits in transmitting magnitude by frequency modulation, so in the proposed model the magnitude of output is expressed directly in the amplitude, which is considerably easier to model.

In the original McCulloch-Pitts neuron, there was one inhibitory input which was absolutely inhibitory rather than additive. This feature may have some biological relevance. In the central nervous system, synapses are sometimes classified as type 1 and type 2, which appear to represent the extremes of a continuum. (Cotman and McGaugh 80 ch. 3). Type 1 synapses are generally excitatory and on the dendrites of the neuron, which is the "normal" location for input. Type 2 synapses are generally inhibitory and on the cell body. They can have longer term and more general inhibitory effects than inhibitory inputs on the dendrites. Certainly, different neuro-transmitters have different effects and different time courses of action (Shepherd 83 ch. 9). Though non-additive input is not included in the current model, it is likely that it could serve a useful purpose, to adjust overall sensitivity, for example. The standard binary classifier (Nilsson 65) has one constant input used to adjust the firing threshold. However, a variable threshold is not necessary in the present model, so that capability is not utilized.

3.3 The Neural Function

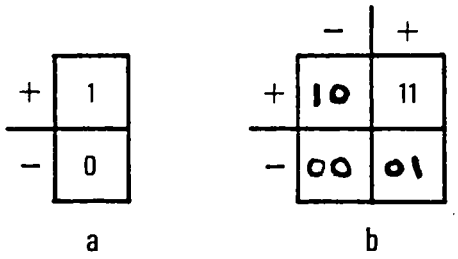
The final result is a node (model neuron) that has two values (synaptic weights) associated with each of its inputs, Fi. One is referred to as Pi for "present" input, and the other as Ni for "not present". Both weights may be either positive (excitatory) or negative (inhibitory). There may be any number of inputs. Output is calculated as:

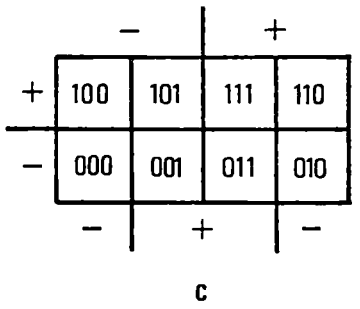
$$\text{Out} := \sum_{Fi > 0} Fi * Pi \quad + \sum_{Fi < 0} - Fi * Ni$$

3.4 Some Useful Functions

Karnaugh maps (Fig. 3.1) provide a useful technique for representing functions of Boolean features (Dietmeyer 78, Mano 79). Each box represents a specific combination of input features, and all possible combinations are represented. A Boolean function can be specified by shading or placing 1's in the boxes corresponding to the desired "true" conditions. The remaining "false" boxes can be filled with 0's or left blank. If appropriate, continuously valued output, or a "don't care" response could also be indicated. The topological properties of such a representation are convenient for designing circuits to compute Boolean functions, since a minimal OR of ANDs representation can be determined by grouping blocks of 1s. Visually, Karnaugh maps are not very useful for more than 4 features, but since the size of the input space increases exponentially with the number of features, it proved impractical to exhaustively model large feature spaces anyway. Consequently, examples will be for functions of 4 or fewer features, though any number can be simulated. The number of features is specified as a runtime variable.

Choosing synaptic weights for Boolean functions is reasonably straightforward (Fig. 3.2). With inputs A, B and C, if the desired function is (A OR B), then appropriate weights are Pi = (3/2, 3/2, 0)





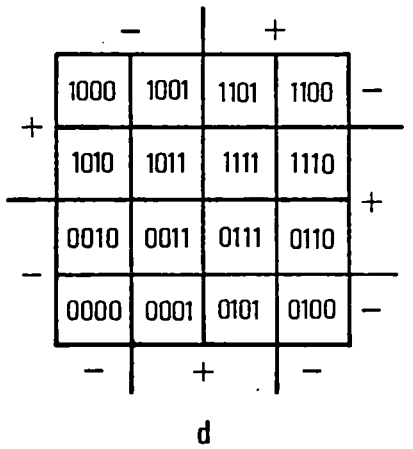
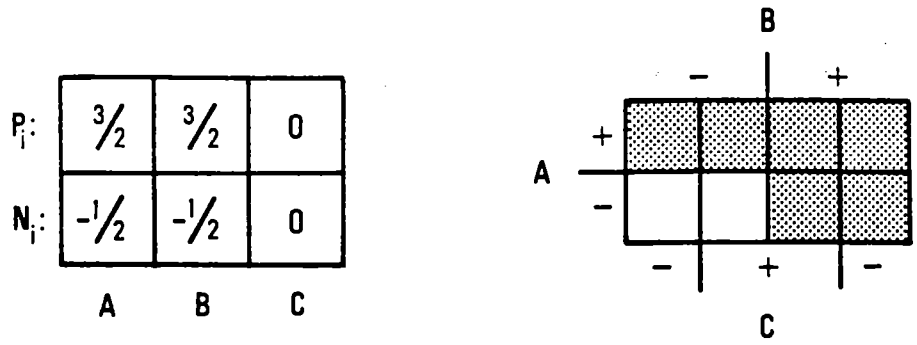


Figure 3.1 Karnaugh maps. a) 1 feature b) 2 features
c) 3 features d) 4 features

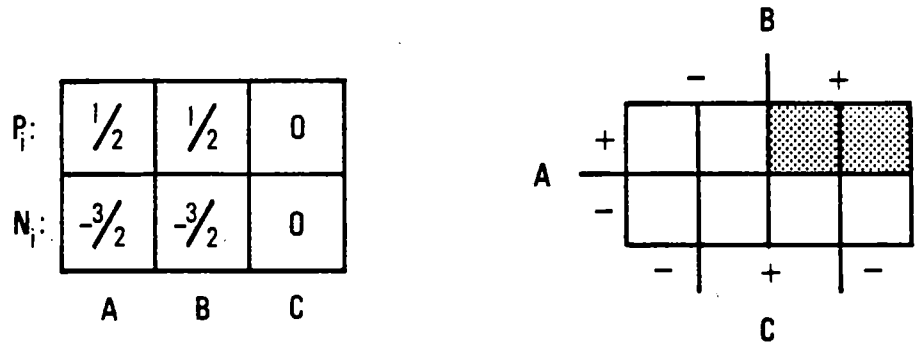
and Ni = (-1/2, -1/2, 0) (Fig. 3.2a). If either A or B are present (=1) input will sum to 1 or greater. If both inputs are absent (= -1), input sums to -1. If one input is .5 and the other is -1, the node's output is .25 rather than the probabilistically correct answer of .5. If both inputs are .5, output is 1.25 rather than the correct value of .75. This inability to compute exact probabilities is an inherent limitation of the functional form, but probabilistically correct calculations are more susceptible to inaccuracies due to unknown inter-feature dependencies (Adams 76, Shortliffe 75).

The AND relationship can also be expressed (Fig. 3.2b). If the function desired is (A AND B), (written (AB) for convenience), appropriate weights are Pi = (1/2, 1/2, 0) and Ni = (-3/2, -3/2, 0).

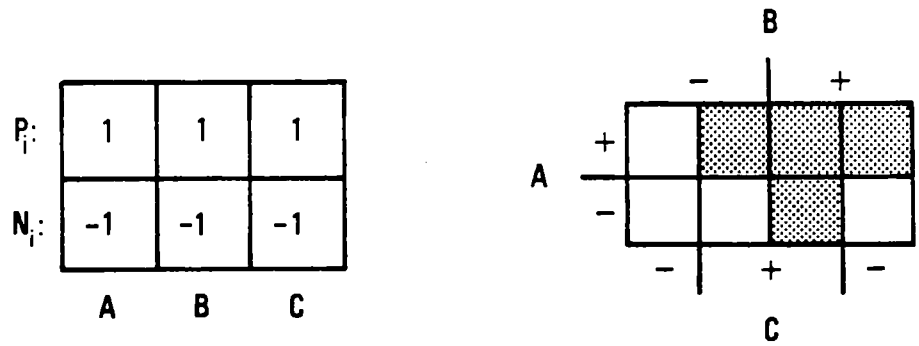
It is also possible to represent functions such as (at least 2 of 3) features, Pi = (1,1,1), Ni = (-1,-1,-1), (Fig. 3.2c), or in general (at least X of N) features. This is a formalization of what Kent has called the "ALMOST" gate (Kent 81). It is interesting to note that OR and AND are simply the extremes of this function (Shapiro 79). (At least 1 of N) is equivalent to OR and (at least N of N) is equivalent to AND. A dramatic advantage over specialized AND/OR nodes is realized for intermediate functions. (At least 10 of 20) can be detected by one general node but would require about 10 ** 5 strictly AND/OR nodes.



a.(A OR B)



b.(A AND B)



c.(At least two of A,B,C)

Figure 3.2 Synaptic weights and Karnaugh maps of 3 Boolean functions.
a) OR b) AND c) at least 2 of 3

The above examples apply equally well to patterns involving the absence of a feature. The Pi and Ni values of that feature are simply exchanged.

3.5 Weight Limits

For the general case of (at least X of N), it is straightforward to solve for values of Pi and Ni such that exactly X features yield a sum of 1 and exactly X-1 features yield a sum of -1. More than X features give a sum > 1 and less than X-1 features give a sum < -1.

$$X * Pi + (N - X) * Ni = 1$$
$$(X - 1) * Pi + (N - X + 1) * Ni = -1$$

Solving for Pi and Ni gives values of

$$Pi = (2(N - X) + 1)/N$$
$$Ni = -(2X - 1)/N)$$

At the extreme of X = N (= AND)

$$Pi = 1/N$$
$$Ni = -2 + 1/N$$

At the other extreme of X = 1 (= OR)

$$Pi = 2 - 1/N$$
$$Ni = -1/N$$

As N varies from 1 to infinity, these weights vary between 2 and -2. Thus the (at least X of N) function can discriminate on the basis of a single feature within a weight limit of 2. Similar calculations for output limits of 0 and 1 (or -.5 and .5) yield a weight limit of 1. In general, the distance between output limits determines the necessary weight limits.

With fixed output limits, the computational power of a node can be increased by increasing the weight limits. For example, allowing weights to vary between -4 and 4 rather than -2 and 2 allows a node to compute new functions such as ((A OR B) AND C). (Pi = (3/2, 3/2, 0) Ni = (-1/2, -1/2, -4)). In general, larger weight limits allow increasing depth of parenthesization. The same effect can be achieved with fixed weight limits by reducing the distance between the output limits.

Many models allow unbounded synaptic weights. This allows a wider range of functions to be computed, but has a problem in that the possible error of a weight is also unbounded. By limiting weights to a maximum of + or - 2, a weight can never be further than 4 from the correct value. The unbounded weights of the standard binary classifier could be scaled to within fixed limits, but in practice seldom are.

Even without a weight limit, some simple Boolean functions cannot be computed by a single node (e.g., Exclusive Or). This is another limitation of the functional form. However, it is unlikely that biological neurons can compute all functions of their inputs either. In fact, the limitations of a linear input function may parallel important biological limitations. Of the 16 possible Boolean functions of 2 features, only Exclusive Or and Equivalence cannot be expressed as a linear function, and those appear to be the most difficult to learn (Bourne 70, Hunt et al. 66, Neisser and Weene 62).

3.6 Prototype Pattern Detection

The (at least X of N) function can be interpreted as a prototype detector. It gives its strongest response to the central prototype (all N of N) and decreasing output for decreasing numbers of features (N-1 of N), (N-2 of N), ... etc. As was shown, it can give perfect discrimination on the basis of a single feature. Prototype, or family resemblance detection, is a useful capability for the description of natural categories (Mervis and Rosch 81).

With output thresholds of -1 and 1 and synaptic weights between -2 and 2, a single node can represent prototypes of the form:

at most X1 features give output = -1 (= false)
at least X2 features give output = 1 (= true)
where X2 - X1 >= 1

This function can also be implemented with only positive weights and variable thresholds. Which form is preferable is problematic since the use of this alternative representation was not extensively explored. Both inhibitory synapses and variable thresholds exist in the nervous system, so neither form is inherently unrealistic.

Although it is possible to implement pattern classifiers with a single threshold value, this introduces difficulty in representing magnitude information. With a single threshold, all output above (and below) the threshold is roughly equivalent. With two thresholds, the magnitude of values between the thresholds can be meaningful.

3.7 Results

The ability for one type of neuron to describe many types of functions is of considerable importance. In the generalized Aplysia example, the motor neuron is trained to detect the proper function. There was no reason, a priori, for putting any particular type of functional node (AND/OR etc.) in that position.

Prototype pattern detection is a general function that the present model can compute. Because this function contains OR and AND, it is sufficiently powerful for Boolean completeness of assemblies. The intermediate input intensity example is a type of function which the proposed model node cannot describe. There are presumably many useful properties which are beyond the capabilities of the present model. As they are observed they may suggest modifications. Immediately implementing the most powerful capabilities might be argued for, but more elaborate models are more difficult to deal with. The present one is viewed as an acceptable trade off between power and simplicity.

4.0 TRAINING A NODE

4.1 The Goal

A learning algorithm for the model neuron is developed in this chapter. The general method of training and the type of output desired are consistent with the gill withdrawal reflex. Specifically, a trainable neuron (the motor neuron) is viewed as a pattern detector. It receives instruction (pain) indicating the correct value of its output which it can increase (by sensitization), or decrease (by habituation). This is viewed as a general learning paradigm which gill withdrawal is a simple instance of. It is equivalent to the formal constraints of perceptron training (Nilsson 65), so there is nothing uniquely biological about gill withdrawal if it is formalized in this way. The basic learning process developed in this chapter is equivalent to perceptron training, but a number of modifications and extensions will be introduced.

4.2 Neural Plasticity Mechanisms

There are many organizing processes that shape the nervous system. During early development, cell division, migration and differentiation take place, and specific patterns of interconnection develop. Later, selective cell death and synaptic degeneration occur, producing a highly structured system before it is exposed to the environment (Cowan 79, Lund 78, Shepherd 83 ch. 10). Some aspects of neural growth are present throughout the life span of most organisms.

At the synaptic level, learning-related structural changes are often observed (Thompson et al. 83).

Even in a statically wired organism a considerable amount of neural, and therefore behavioral plasticity remains, due primarily to the existence of variable synapses (Tsukahara 81). The mechanisms of synaptic modification can be classified as either homosynaptic or heterosynaptic (Kandel 79c). Homosynaptic processes are intrinsic to the synapse and include habituation and its converse of facilitation. The synapse spontaneously becomes either more or less effective based on its previous activity. Heterosynaptic processes are dependent on the activity of other pathways, and include dishabituation (sensitization) and its converse of inhibition. In this case an external signal causes a change in synaptic strength. There are additional processes in both categories (Krasne 78). In addition to modification of specific synapses, the sensitivity of the neuron as a whole to its synaptic inputs can be modifiable (Lynch et al. 77, Woody et al. 76, Brons and Woody 80, Kandel 77, Woody 82 pg. 151, 195, Brons et al. 82).

Since there are a number of biological learning mechanisms besides habituation and sensitization, it is obvious that the gill withdrawal model of biological learning is not complete. However, it provides a good starting point, and the other processes may be best understood by exploring their capabilities and limitations individually. By modeling processes without identifying their purpose, there is the risk of merely simulating biological

limitations.

The learning process implemented here is conceptually consistent with gill withdrawal, but the actual method of implementing the process departs from the biological model. At some point it is important to notice that computers and neurons work on different physical principles, so that increasingly detailed simulation of biological processes is simply increasingly inefficient implementation of the desired behavior.

4.3 Learning Models

Formal learning models have been categorized in different ways (Duda and Hart 73, Hunt 75, Nilsson 65). One common distinction is between parametric and non-parametric training. In general, parametric training implies a certain amount of pre-knowledge of the correct functional form and appropriate features, so that only certain population variables require measurement. In non-parametric training, the functional form and the appropriate features are not predetermined, so a general form is used, such as linear or quadratic. The function is tuned by adjusting its coefficients during training. Since an "unbiased" linear form is probably a reasonable approximation of the actual neural function, this distinction is rather indistinct in the case of neural learning.

A second major division of learning is between training with and without a teacher. The Hebbian model of synaptic modification (Hebb 49) is the pre-eminent example of learning without a specialized teacher input. In the Hebb model, a synaptic weight is increased if there is input on that line when, or just before, the node fires. Something like this has been demonstrated in several neural systems (e.g., O'Brien and Quinn 82). Teacherless learning is appealing since the question of who generates the teacher signal doesn't arise, and with the addition of some whole network constraints it can lead to the self-organization of a number of interesting types of pattern detectors (Amari and Takeuchi 78). However, there are limits to the types of functions that can be learned this way. Learning is essentially limited to picking out statistical associations from background noise. If input is unpatterned (e.g., random sequences of random combinations of features), there is nothing to be learned. To learn useful output, some input information must be interpreted as instruction or evaluation. In addition, the basic Hebbian model is susceptible to a number of saturation and stability problems (Sutton and Barto 81).

Learning with a teacher implies specialized input information with instructive properties. Threshold pattern classifiers are a common example (Nilsson 65). In this case, a teacher input specifies which side of the threshold input should sum to, so the weights, and perhaps the threshold, can be adjusted accordingly. A teacher signal can describe appropriate output with arbitrary precision by indicating

for each input whether the current output is right or wrong, too high or low, or by specifying the correct output directly. If teaching information is used to fire the node, Hebbian learning can be utilized. In general, instructive input might be specialized in time (the period immediately after a neuron fires), synapse type (as in gill withdrawal), signal pattern or transmitter type, so long as it is functionally identified to the receiving neuron. In addition, learning may be equally selective for the preceding inputs and activity of the receiving neuron.

Without belaboring the biological validity of the basic Hebb model, it can be observed that most implementations are inherently input oriented. That is, a network is organized solely by its input, and little attention is given to desired output. Since it was suggested that an important aspect of network organization is goal directed behavior, development of this model will concentrate on that process. This requires specialized instructive/evaluative input to indicate appropriate output.

The current model assumes the existence of a perfect teacher (i.e., the correct answer is always available to adjust the function). However, with multiple operators, a global evaluation of behavior may not provide an unambiguous teacher for each operator. For example, knowing that you did something wrong may not tell you what would have been right. This problem will be avoided in greater detail when evaluation is discussed (chapter 7).

4.4 Adjusting Weights: The Basic Algorithm

The basic learning process in the model can be summarized as four conditions.

A weight should be increased if:

1) The node should have fired
2) There was input on that line
3) The node's output was < 1
   (i.e., less than the upper threshold)
4) The weight is not already its maximum value

Number 1 uses the teacher input(s) to indicate correct output. In gill withdrawal, this corresponds to the pain input. Number 2 makes learning associative. Gill withdrawal displays both associate and non-associative components. Number 3 prevents needless learning and a host of problems resulting from unneeded modifications. Gill withdrawal has not been reported to display this characteristic, but the resulting phenomena of "blocking" (section 4.6) is displayed by other molluscs (Sahley et al. 81). Number 4 enforces the fact that synaptic weights cannot be unbounded. Except for 4, these conditions are the same as perceptron training (Nilsson 65). They are also similar to what has been called Rescorla-Wagner learning (Rescorla 72, Rescorla and Wagner 72). One important characteristic is that learning takes place only when the node did not respond correctly, that is when output < 1. More succinctly, "organisms only learn when events violate their expectations" (Rescorla and Wagner 72).

One possible learning function for achieving the above constraints is:

```
Wi := Wi +
TS * In(i) * (1 - Out) * (2 - Wi) * wt_rt
```

Where:

Wi	= the synaptic weight (-2 to 2) associated with input from node(i)
TS	= a teaching signal (0 to 1) expressing the probability that the node should have fired
In(i)	= the output (0 to 1) of node(i)
Out	= the current output (-1 to 1) of this node
wt_rt	= a constant determining the maximum rate of weight change

If TS = 1, In = 1, Out = -1 and Wi = -2 the weight will increase by the maximum amount, 8 * wt_rt. In(i) is between 0 and 1 rather than -1 and 1 because Pi and Ni are adjusted separately. Positive values are used to adjust Pi and negative values to adjust Ni. Synaptic weakening takes place by a complementary process.

By applying the function to each weight, (either Pi or Ni for each input) a node can be trained, by appropriate selection of input and TS, to compute any of the previously discussed functions. For proof that such algorithms converge on the correct output, see Nilsson (65).

4.5 Modifications

The preceding function is more for illustration of the basic constraints than for serious implementation. It suffers from a number of practical limitations which effect the rate of convergence. However, since the basic approach is convergent, it is possible to consider ways of improving it so that the correct output is reached

more efficiently. Rather than formulating a simple, elegant learning algorithm and discovering its capabilities, the specific goal of training a node's output rapidly and efficiently will be pursued. A number of modifications will be suggested to that end.

4.5.1 The Rate Constant Wt_rt - An obvious candidate to increase learning speed is the value of wt_rt. Though a large value produces rapid learning, it also results in a short memory and greater noise sensitivity. This particular constant could be tuned to the noise level of its environment. However, a limit to the maximum rate of output change has subsequently made this value largely irrelevant since the resulting weight changes are usually much less than the limit.

4.5.2 Approaching Limits - A more useful method of increasing learning speed is to modify the rate at which limits are approached. The correct values can be approached (and achieved) much more rapidly if they are not approached asymptotically. Two limits are asymptotically approached in the learning function, the extreme values of the weights and the correct value of output of the node. For the weight limit, this modification is straightforward. The adjustment is calculated without regard to the limit, and if the adjusted weight is beyond a limit, it is set back to it.

Approaching the output limit is treated in a similar manner. It is desirable to approach the limit rapidly, but without overshoot. The limiting term (1 - Out) reduces the rate of increase as output approaches 1, but does not take into consideration the number of features being adjusted. For patterns with few features the rate of approach will be slow, and for patterns with many features overshoot will occur. The solution to this problem was to calculate the weight adjustments without regard to the output limit, then recalculate the current output on the basis of the new weights. If output is greater than a limit, the weight changes are scaled back so that output equals the limit exactly.

This technique is also used to limit the rate of output change for the node as a whole, nd_rt, usually to .3. This makes the maximum rate of individual weight change largely irrelevant, but the same tradeoffs between speed of learning and noise sensitivity occur at the whole node level. Actually, a variable nd_rt which increases with the error magnitude proved to be useful and is currently employed:

```
nd_rt := .3 + Abs(CO - Out) * .2
```

where CO is the correct output, and Abs(CO - Out) is the error of Out. The constants were empirically adjusted during network simulation.

4.5.3 Choosing Relevant Features - Another process which strongly affects the speed of learning is identification of the appropriate features to adjust. In most learning processes, the relevant features are identified by the simple requirement that they are present when

weights are being modified. This strategy is adequate for small feature spaces, but becomes progressively worse with increasing numbers of irrelevant features. (Irrelevant in the sense that they do not affect the value of the function being learned). The problem is obvious in the case of a constant feature. For learning the function (A OR B) when C is a constant 1, it is clear that C is irrelevant, though it is always present. A constant (or in general, independently distributed) feature should not be considered an important part of any pattern. Unfortunately the learning algorithm is helpless to prevent this, and expends a considerable amount of time and effort increasing and decreasing the weight on C. Taken to extremes, this problem is fatal to the learning process. If there is one key feature and a great deal of irrelevant background activity, the learning process will be unusably slow since the relevant feature cannot be separated from the background. The perceptron training process requires one constant feature which corresponds to an adjustable threshold, but increasing the number of constant features beyond the minimum of one decreases the rate of convergence.

A remedy for this problem is suggested by the predictive nature of input for correct output. The value of [X|Bi] (the probability of X given Bi) is equal to [X] (the probability of X) if Bi has no predictive value for X, as is the case for a constant feature. If [X|Bi] is greater than [X], Bi is predictive of X's occurrence. If [X|Bi] is less than [X], Bi is predictive of X's non-occurrence. This permits a modification of the learning algorithm so that weights are

made more positive for only those features which have positive predictive value. Likewise, weights are made more negative for features having negative predictive potential (e.g., a missing feature in an AND). Related conditional or predictive processes have been considered in other models (Sutton and Barto 81, Klopf 82, Rosch 78, Uttley 79, Bindra 75, Sejnowski 81).

To implement this, the increase (decrease) in weights is limited to the ratio of the predictive potential of each feature to the maximum (minimum) value of all the features currently present. Since this ratio is always between 0 and 1, its effects can be tuned to some extent by taking it to an adjustable exponent. For example, to sharpen the focus on only the most predictive features rather than spreading the responsibility around, the ratio could be squared. It is interesting to note that for many Boolean functions, a weight vector exactly proportional to conditional probability will correctly categorize the input space.

A conditional probability "trace" can be iteratively calculated as:

Method I

```
[XBi]    := [XBi] + (X * Bi - [XBi]) * frq_rt
[Bi]     := [Bi] + (Bi - [Bi]) * frq_rt
[X|Bi]   := [XBi]/[Bi]
[X]      := [X] + (X - [X]) * frq_rt
```

or as:

Method II

```
[X|Bi]   := [X|Bi] + (X - [X|Bi]) * Bi * frq_rt
[X]      := [X] + (X - [X]) * frq_rt
```

where frq_rt is a constant determining the maximum rate of change.

The first method has the seemingly desirable property of adjusting for the amount (frequency and intensity) of input. However, it has several drawbacks. In the absence of input, Bi, the stored information decays into the background noise. Reducing the rate of change delays the process but doesn't eliminate the problem. A related problem is that the ratio [XBi]/[Bi] can vary widely with small changes in [XBi] or [Bi]. Because of this, the computed conditional probability can wander around quite a bit depending on the distribution of the input feature's occurrences and the value of frq_rt. The second method has neither of these problems, but has its own drawback. Since it doesn't adjust for input amount, low level or infrequent features take longer to reach their proper value.

Both methods also suffer from a theoretical limitation of all single "trace" theories. Unless X is Boolean, frequency and intensity are confounded, so that an intense but infrequent signal can be equivalent to a weaker but more frequent one (e.g., a trace value of .8 can result from 1.0 80% of the time, .8 100% of the time, or anything in between). Both methods also suffer from the usual rate of change trade off. A long memory (small rate of change) is appropriate for a reasonably stable system, and a short memory is needed for rapid adaptation. The current learning rate (frq_rt = .1) was selected to give reasonable results when cycling through the complete input space of 4 or 5 features. Both methods were implemented and work adequately, though with the above defects. For lack of a clear cut preference, either method can be chosen as a run time option, though

method II is generally used.

Another variation on the calculation of conditional probability is to adjust the trace values only when output is incorrect. In this case, the trace is predictive of when the node's output should increase or decrease, rather than when it should be on or off. This has some theoretical appeal, and often gives superior results. It is implemented as a run time option.

4.5.4 Learning Vs Unlearning - The use of conditional probability works well in identifying weights which should be increased in magnitude, but selectively identifying those which are too large and should be decreased in magnitude proved more difficult. Fortunately it also proved to be less crucial than selective increase (if in fact it is desirable at all), so for convenience, decreases are currently calculated without regard to predictive potential. This still leaves the problem of how much they should be decreased. The solution was to make a distinction between learning (increasing in magnitude) and unlearning (decreasing in magnitude). Their relative contributions depend on your faith in the accuracy of past learning. Arbitrarily, the contributions of unlearning and learning were set at .1 and .9 of the total change. Unlearning only works exactly for the Boolean extremes of input, but hasn't been noticeably troublesome in subsequent development when intermediate values also occur.

Weight limits of -2 and 2 are adequate for the (at least X of N) function, but the model learns faster if it has extra room to maneuver in. However, if weights go beyond -2 and 2, it is because other unnecessary weights are accumulating. In order to give the model some extra room, but to discourage its abuse, the weight limits were doubled to -4 and 4, and the unlearning fraction was coupled to the weight of the most predictive feature in the current input.

```
un_frac := un_frac + (1 - un_frac) * ((Wt - 2)/2)
```

It also proved useful to adjust the unlearning rate with the magnitude of error. A large error is indicative of inappropriate weight accumulation, so unlearning is increased with increasing error.

```
un_frac := un_frac + Abs(CO - Out) * .2
if un_frac > .8 then un_frac := .8
```

The constants were empirically adjusted.

In certain biological systems it has been suggested that unlearning does not occur at all, and that as a result, memory becomes saturated with increasing age (Barnes 79, Baudry et al. 81). On the surface, this appears to be more of a limitation than a feature. Though perhaps unbiological, these adjustments of unlearning proved quite effective in avoiding weight saturation, and keep a node near the center of its plasticity range.

4.5.5 Rate And Direction - Finally, it should be noticed that there are basically two variables controlling the change in output of a node: the direction (increase or decrease), and the rate of change.

The correct output, CO, indicates direction, and Rt controls the rate.

```
New_out := Old_out + (CO - Old_out) * nd_rt * Rt
```

In the original algorithm, the teaching signal, TS, controlled both rate and direction.

```
New_out := Old_out + (1 - Old_out) * nd_rt * TS
                   - (1 + Old_out) * nd_rt * (1 - TS)
```

An alternative is to control direction and rate separately. TS can be used to indicate correct output, and a separate value, Rt, can be used to adjust the rate.

```
New_out := Old_out + (TS - Old_out) * nd_rt * Rt
```

In assembly training (next chapter) it is useful to control rate and direction separately, so the latter form is utilized.

4.5.6 Memory Time And Noise Rejection - Typically the environment does not provide instruction which is 100% accurate. The fact that things get better or worse may or may not be contingent on the preceding activity of the organism. Uncontingent reinforcement is essentially noise in the teaching signal. Since there is a great deal of uncontrolled environmental feedback to any organism, any learning process should be as insensitive to teaching noise as possible. The logical extreme of this problem is what a neuron, or organism, should do in response to a totally uncontrolled reinforcement signal. This appears to be a significant problem for a wide range of organisms (Seligman 75, Maier and Seligman 76, Maier and Jackson 79).

As previously observed, noise sensitivity can be reduced by decreasing the learning speed, but slow learning has its own drawbacks. The memory characteristics of Aplysia (Kandel 77, 79ac) can be interpreted as a response to this problem. Variable synapses may display both long and short term memory. Simplified a bit, the important characteristic of this phenomena is that if a synapse is held at one value (say W = 1) long enough, it becomes fixed. Otherwise it decays back to the previously fixed value. This allows a certain amount of teaching signal noise rejection and the capability for a short term episodic memory.

At a minimum, two weights are required, a short term value (ST) and a long term value (LT). ST is the weight that is actually used in output calculations, and it is trained the same as before except:

1) it can move faster toward LT than away from it
2) it may spontaneously decay back to LT with some slow rate constant.

Number 1 rejects noise by keeping ST from "random walking" away from LT. Number 2 makes some specific assumptions about useful episode times, since effects decay with a specific time constant. The other process to consider is the training of LT. It can be trained with the same teaching signal as ST but with a slower rate constant, or may move toward ST at some slow rate. This arrangement is different than a single memory with a slower rate of change, since noise sensitivity can be reduced without a concurrent reduction in the maximum rate of noiseless learning.

Theoretically this process could be extended with as many levels of memory as desired, giving a wide spectrum of adaptation times. Each weight could be coupled with its immediate neighbors as ST and LT were coupled to each other, or it might move toward some weighted average of other weights. As before, only the shortest term value need be used in actual output calculations. The current model applications are not intended to be either episodic or noisy, so this is not of much immediate value, but a dual trace memory is implemented as a runtime option.

4.6 Neural Learning And Classical Conditioning

It has been demonstrated that perceptron-like learning algorithms can display many characteristics of classical conditioning (Sutton and Barto 81, Rescorla and Wagner 72, Rescorla 72). The phenomena of blocking is such an effect. As an example of this, if a node is completely trained to detect some feature A before a new feature B is paired with A, further training will not increase the input weight of B. Since the node is already responding correctly, there is no need for any weights to be adjusted.

More generally, the Rescorla-Wagner learning model emphasizes the predictive nature of the inputs for the correct output (Rescorla 67), a feature which is specifically addressed in the Bayesian selection of appropriate features to strengthen. However, Rescorla-Wagner learning shows a contingency effect only after a period of training. In the

proposed model, conditional probability is stored as a separate input trace which can produce differential conditioning on a single trial.

This two-stage model of learning is similar to one proposed by Mackintosh (Mackintosh 75, Dickinson and Mackintosh 78, Mackintosh and Reese 79). In that model, the "salience" constant in Rescorla-Wagner learning is replaced by a variable reflecting a feature's current predictiveness. This modification explains a number of "latent learning" phenomena that are not adequately captured by the Rescorla-Wagner model (Mackintosh 75, Lubow 73, Pearce and Hall 79, Rescorla and Holland 82, Gormezano et al. 83). In the current model, conditional probability is adjusted to predict when a node should be on, or alternatively when output should be increased. Roughly equivalent acceleration of learning is possible, though the resulting "latent learning" effects are somewhat different. There are other computational variations, each of which produces slightly different effects. The important commonality is that a two-stage model permits a significant acceleration of learning by explicitly utilizing feature salience.

Perceptron training does have some differences from classical conditioning however. Rescorla reports that if two separately conditioned stimuli are paired together for a time, they are no longer as effective individually (Rescorla 72, Rescorla and Holland 76). This is the same as saying that if input sums beyond 1, the weights are reduced. At the moment this doesn't seem advantageous, so it is not implemented in the model.

Another feature of classical conditioning that is not displayed by this model is that novel stimuli are more apt to be utilized as relevant features. In current applications of the model, novelty is not a useful characteristic to detect. Consequently, that process is not addressed here, though it clearly is an aspect that should be considered for more general conditions.

Actually, the entire concept of classical conditioning may be something of a red herring. As observed by Woody (Woody 82) and many others, classical conditioning is not a unitary phenomena, but is in fact the result of several distinct learning processes. It is a style of behavior under particular conditions, not a specific mechanism. Consequently it may be misleading to debate whether an organism displays "real" classical conditioning. For example, Aplysia displays associative learning but has not been reported to display blocking (though other molluscs have). It may be more productive to explore the individual processes of learning rather than classify their combination as "classical" or not.

From a functional point of view, an important characteristic of classical conditioning is that its best formal model (Rescorla and Wagner 72) is similar to the mathematically useful perceptron training process (Nilsson 65). The particular mechanisms are immaterial if their results are adequately captured at the functional level. Aplysia learning (as described here) is not sufficiently powerful to completely implement the perceptron training process, but it seems reasonably safe to assume that such mechanisms do exist in simple

organisms.

4.7 Results

The final result is an algorithm which is several pages long, and considerably more complex than the Hebb hypothesis. It is, however, reasonably efficient in training a node's output. By having a complex process produce a simple result rather than vice versa, it is possible to proceed to larger neural assemblies since the action of the component parts is reasonably well defined.

Because it is a linear function, learning convergence in the proposed model could presumably be proved using something like the perceptron convergence theorem. Conditional probability, weight limits and unlearning complicate the issue, but the basic process of training a linear function as a pattern detector is well established.

In Figure 4.1, the function (at least 3 of 4) is learned. For convenience, patterns are cycled in numeric order (i.e., -1-1-1-1 = 0 and 1111 = 15), though the actual order of presentation is largely immaterial. The output for representative inputs with N, N-1, N-2, N-3, and N-4 features is shown. Positive instances are learned in order of similarity to the central prototype (all 4 of 4), and negative instances are learned in order of similarity to the inverse of the prototype (0 of 4). This is consistent with human learning of prototypes (Mervis and Rosch 81). As a corollary, it's clear that borderline hits and misses (X and X-1 features) are the best training

examples (Winston 75). Though cyclic repetition of all input possibilities is an unnatural situation, it seems reasonable that highly specific resolution (separating (2 of 4) from (3 of 4)) would generally take longer to learn than low resolution (separating (0 of 4) from (4 of 4)).

As an example of a node's ability to learn AND's, a node was sequentially trained to detect each letter of the alphabet using a 14-stroke representation scheme (Fig. 4.2) (Rumelhart and Siple 74). For each letter, the complete alphabet was cycled through until discrimination was perfect. All letters were successfully learned in an average of 9 cycles.

Two styles of input presentation are used in testing the model. Either the input patterns are cycled through in numeric order, or they are presented in randomized blocks of one cycle. Totally random pattern selection also works, but the results are difficult to assess. For individual functions, input order can effect training speed, but on the average it had little effect.

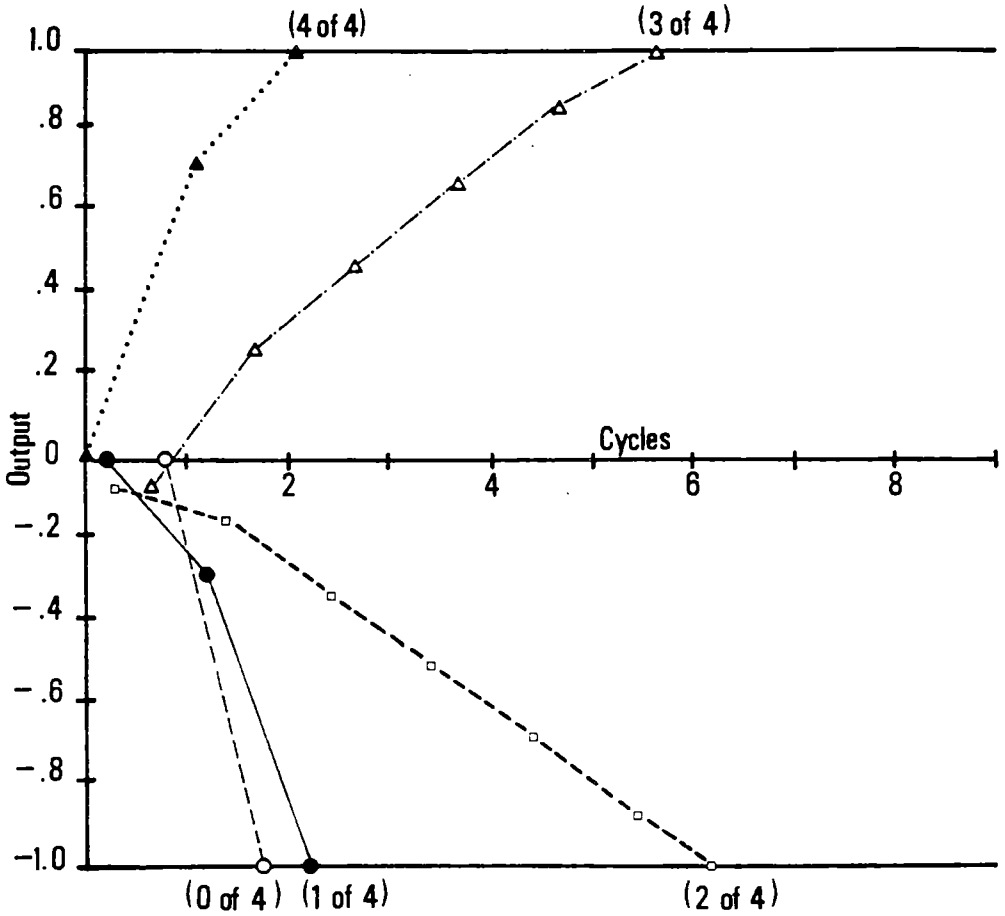


Figure 4.1 Prototype learning. (at least 3 of 4)

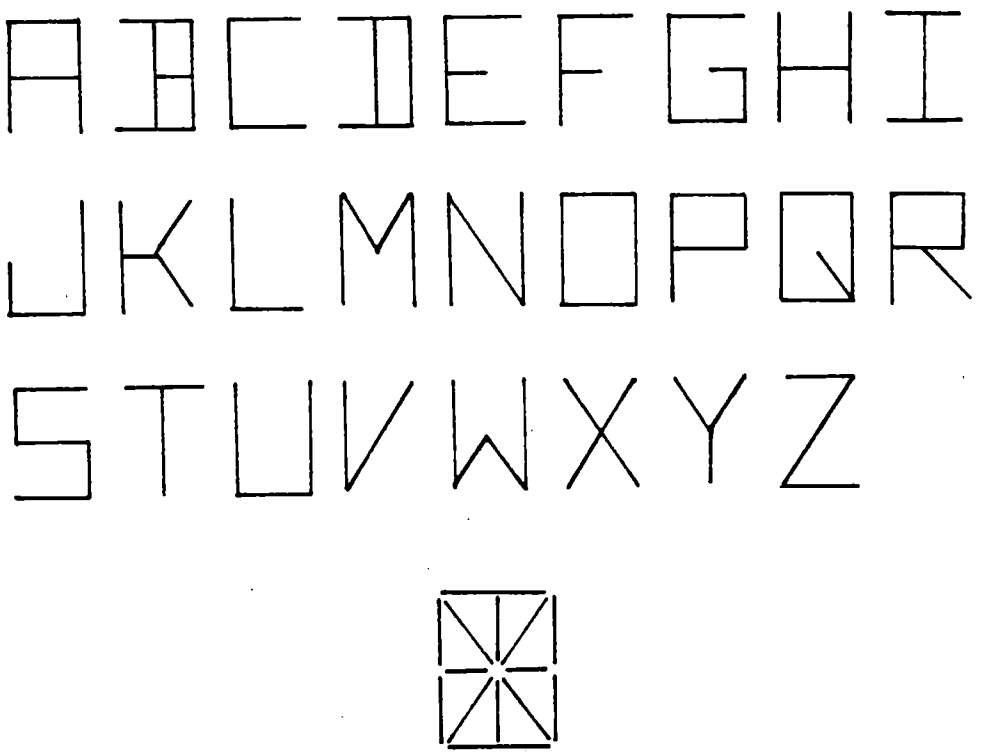


Figure 4.2 14 stroke alphabetic representation.
(Rumelhart and Siple 74)

5.0 A MODEL OPERATOR

5.1 The Goal

If behavior is viewed as a series of operator applications, the only requirement for intelligent behavior is that each input pattern results in the appropriate response, or conversely that each operator is a pattern detector for those conditions under which it should fire. Since the features an operator detects can be the output of any other neuron, this is consistent with both externally and internally generated behavior. The general purpose of this chapter is to investigate network controls which can train a collection of nodes to act together in a cooperative manner so as to implement an abstract operator.

As previously observed, if there are desirable functions a single node cannot detect, then an assembly of nodes should be able to. The current domain is restricted to Boolean inputs, so for completeness an operator should be able to detect any Boolean function. There are Boolean functions a single node (linear function) cannot detect, (e.g., Exclusive Or), so the need to train an assembly to detect them arises. The specific goal of this chapter is to develop network structures and associated learning processes appropriate for learning arbitrary Boolean functions.

5.2 Operator Structure

The structure of a Boolean operator can be quite simple. Since all Boolean functions can be expressed as the OR of ANDs, this determines the minimum computation needed for completeness. A two-level structure is a sufficient, though not necessarily efficient, representation (Fig. 5.1). A single OR node on top is required for the final output, and since an unbounded (<= 2 ** N for N features) number of AND nodes may be needed for arbitrarily complex Boolean functions, a plane of arbitrarily large size is used for the lower nodes. The minimum connections required are that the plane nodes see the input features and that the top node sees the plane nodes. Thus the desirable simplicity of linear functions can be maintained for individual nodes while implementing a completely general Boolean function. Reaction time (input to output) is two node delays.

This is essentially the structure of a perceptron (Rosenblatt 58, 62). However, both the lower nodes and the top node are trainable. In the standard perceptron only the top node is trainable; the lower plane is a collection of pre-wired functions. Another important difference in the current model is that each plane node can always see all the input features. No attempt is made to minimize interconnections as has been emphasized in perceptron analysis (Minsky and Papert 72). With increasing network size, that problem must ultimately be addressed, but any approach which does not work with optimal interconnection is not apt to improve with limited connections.

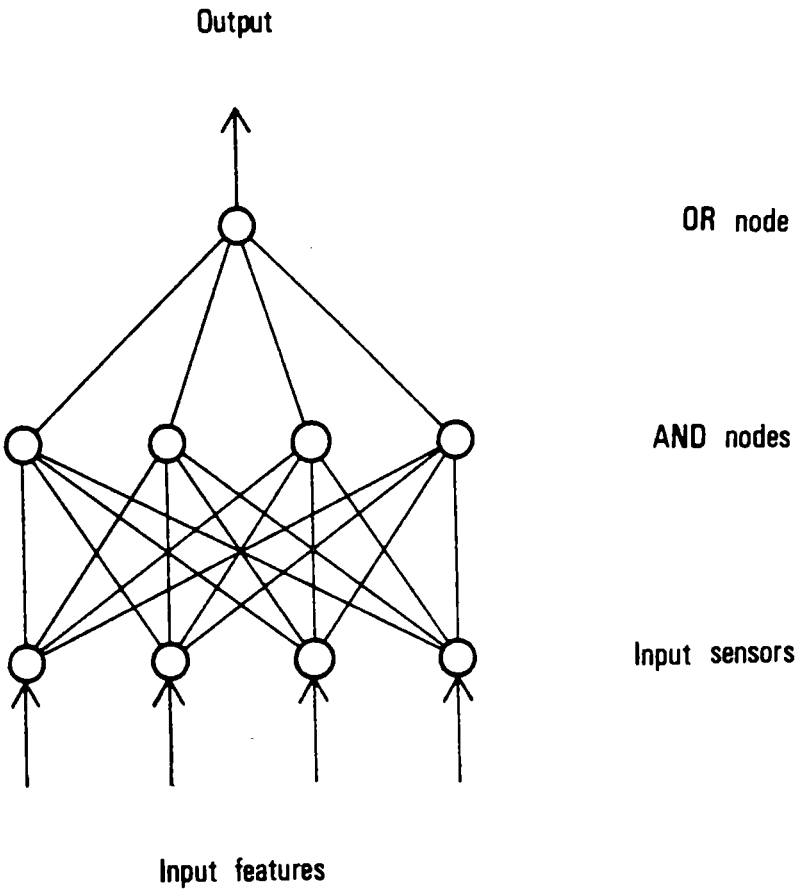


Figure 5.1 Minimal structure capable of Boolean completeness using linear function nodes.

Even in very simple systems a variety of distinct types of associations can be identified. For example, in Figure 5.2a, the co-occurrence of features A and B produces response R. Direct stimulus-response (or CS-US) associations (A -> R, B -> R) are the basis of most formal models of instrumental or classical conditioning (e.g., Rescorla and Wagner 72).

For Boolean completeness, the model requires at least one intervening layer between stimulus and response. It seems reasonable that biological S-R connections also possess intermediate levels of analysis. Consequently, intermediate associations are possible (Fig. 5.2b). The existence of such associations (A -> C, B -> C, C -> R) has been demonstrated and incorporated into the Rescorla-Wagner model (Rescorla 73).

Finally, the introduction of recurrent connections (Fig. 5.2c) produces a new set of associations. Sideways connections (A -> B, B -> A), are the most direct interpretation of within-stimulus learning phenomena (Rescorla and Durlach 81). Top-down connections (C -> A, C -> B), provide a form of "cognitively" driven expectation.

5.3 Lateral Inhibition

Lateral inhibition between nodes is often used to implement specific network properties. Two common approaches are inhibition of output and inhibition of learning. If the firing of a node is inhibitory to the firing of other nodes, there is a maximum number of

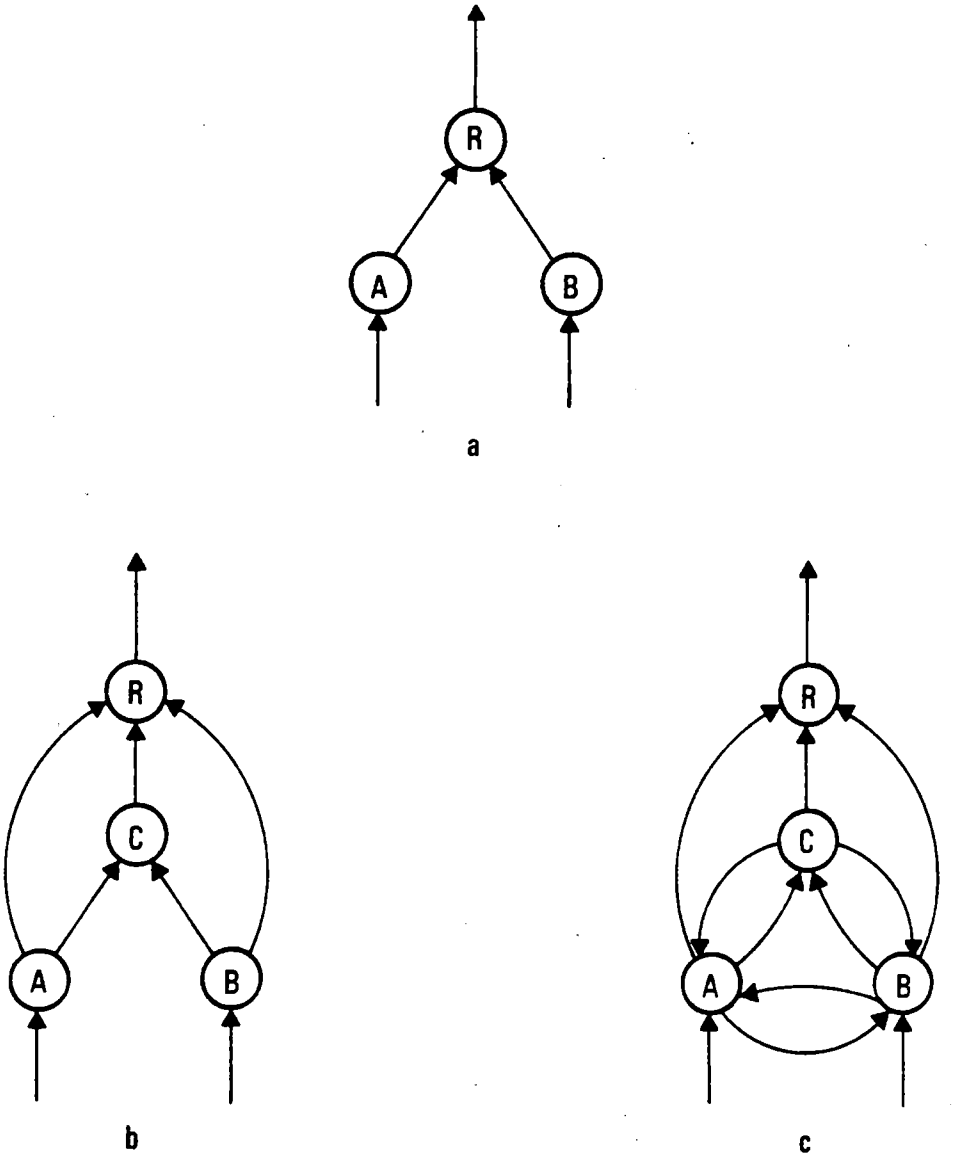


Figure 5.2 Increasing complexity of input-output associations.
a) direct b) indirect c) recurrent

nodes that can be on at any one time. Many neural models incorporate this principle as diffuse or random inhibitory connections within a network (Amari 77a), and it is a common mechanism for sharpening discrimination in natural systems (Linsay and Norman 77). Lateral inhibition of learning is similar, except that the firing of a node reduces the ability of other nodes to increase their output (Fukushima 75, 80, Fukushima and Miyake 78, 82). An obvious difference is that there need not be any limit to the number of nodes that are on at one time.

Both of these processes can be structured so as to support assemblies more complex than a single pool of nodes. A layered network can be constructed by limiting interaction to planes, and a topographic effect can be achieved by limiting the effects to nearby nodes (Amari 80, Kohonen 82ab, Overton and Arbib 82).

5.4 Operator Training

Training an operator can be viewed as two separate processes: training the top node and training the plane. Training the top node to be an OR is trivial if the bottom nodes are already correctly trained, and impossible if they aren't, since it uses their output. The pain to train falls mainly in the plane. Constraints on the plane's output are relatively simple:

1) For all incorrect input patterns
   all nodes should be off.
2) For all correct input patterns
   at least one node should be on.

The first constraint is easy to implement. The desired value of TS = -1 is simply broadcast to all nodes in the plane. All nodes with output greater than -1 learn and adjust their output downward. The second constraint is more complex. If TS = 1, then at least one node should have output >= 1. If this is the case, no changes are necessary. If this is not the case, learning should take place and nodes should adjust their output upward. This can be implemented by controlling either the value to be approached, TS, or the rate at which it is approached, Rt. As before:

```
New_out := Old_out + (TS - Old_out) * Rt
```

5.4.1 Varying TS - A form of lateral inhibition is achieved if TS for all nodes in the plane except the one with maximum output is calculated as:

```
Plane_TS := TS - Max_out * 2
If TS = -1 then Plane_TS := -1
Truncate(Plane_TS, -1, 1)
```

As the node with maximum output (Max_out) approaches the desired limit (Max_out = TS = 1) all other nodes are forced back to -1. Truncate limits Plane_TS to between -1 and 1. The node with maximum output is trained with TS directly. For convenience the value of Max_out is determined by sequentially searching all nodes in the plane, but biologically motivated parallel networks displaying appropriate behavior have been demonstrated (Amari and Arbib 77). When run to completion, the plane specializes so that exactly one node is on for all correct inputs, and all are off for incorrect ones. The top node

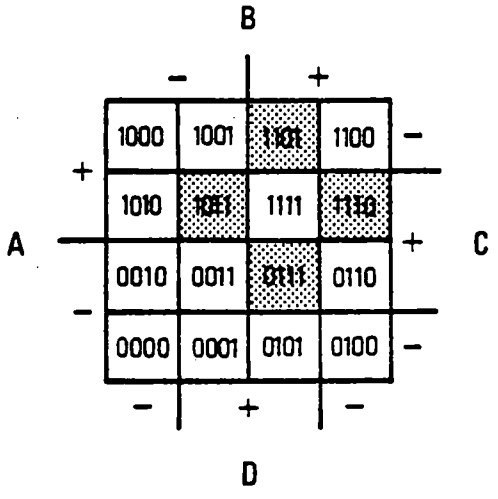
can now be trained as an OR of the plane's output.

Actually, the top node need not wait, nor receive output only from the plane. It can be connected to the input features as well, and be trained with TS while the plane is being trained. If it needs the plane's output it simply won't do very well until the plane is correctly trained and will then converge on the desired output. Of course if the function can be learned by a single node, the output of the plane isn't necessary, and the top node can learn the pattern directly.

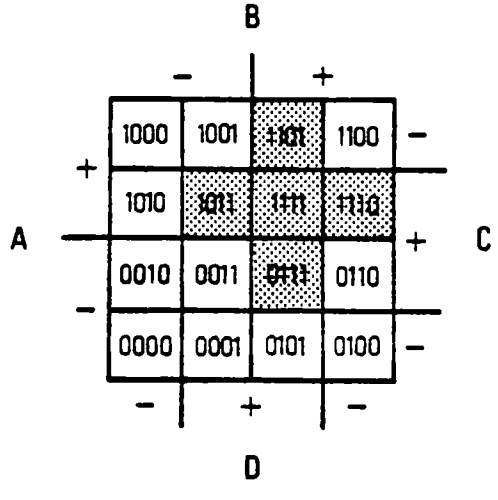
Unfortunately, there is a potential problem in connecting the output node directly with the input. The plane's output will produce a decodable representation (the ANDs of an OR) while the raw input may not be decodable by a single node. There are some cases (e.g., Fig. 5.3a, which will be considered in greater detail shortly) such that the output node does better without raw input. Empirically, the raw input was not especially useful to the output node anyway. Rather than completely disconnecting them, the rate of change for those links was reduced.

5.4.2 Varying Rt - An alternative approach is to not demand that correct inputs trigger exactly one node in the plane (as varying TS does) but only that at least one node fires. This can be implemented as a limitation of the rate of change:

```
Plane_Rt := TS - Max_out
If TS = -1 then Plane_Rt := 1
Truncate(Plane_Rt,0,1)
```



a.(3 of 4 features)



b.(At least 3 of 4 features)

Figure 5.3 a) Prototypic category without central exemplar, exactly 3 of 4

b) Prototypic category with central exemplar, at least 3 of 4

Rather than being forced back to -1 as Max_out approaches 1, the output of other nodes in the plane simply stops going up. In effect, as Max_out approaches 1 the learning process is turned off in the rest of the plane. As before, when TS = -1 all nodes are trained to be off. This is considerably more efficient since it allows patterns to overlap, and it is easier (faster) for the plane to learn more inclusive, general patterns than highly specific ones. In addition, fewer nodes are required. Under conditions where overlap is possible, this algorithm converges much more rapidly on the correct output. Because of this advantage, controlling Rt was the method chosen for further development.

5.4.3 Group Dispersal - There is one modification that is necessary for either of the preceding algorithms to work very effectively (i.e., at all). The problem is that Max_out has to be rather large before it begins to effectively inhibit other nodes. Consequently, in the early phases of training there is considerable crosstalk and confusion as to just which node(s) are responding to which pattern. This could be remedied by making inhibition effective at lower levels of Max_out:

```
Plane_Rt := (TS - Max_out) ** 2
```

This technique would be acceptable if the Max_node were always the correct node to learn the pattern. Though this is often the case, it is not always true and the algorithm might be unusably slow or fail to converge. For example, in Figure 5.3a an especially pathological case was shown. It is a prototypic category (at least 3 of 4

features) (Fig. 5.3b), without the central prototype itself (4 of 4 features). This is clearly an unnatural situation; if all the Smith children differ from their father in exactly one (different) feature, and all look like Smiths, then so should the father who has all the common features. The initial response of the plane is to over-generalize and learn the pattern (at least 3 of 4), which is easy to learn since it requires only one node. This is not unreasonable, and people display similar behavior by strongly categorizing central examples without any prior exposure to them (Mervis and Rosch 81).

What the plane needs to do is break up the single pattern, which is wrong in the center, into 4 separate cases. (A different strategy is considered in the next chapter). If Max_out = 1, it is still necessary to turn off learning in the plane, but it is clearly disadvantageous to shut off learning in other nodes well before Max_out reaches 1, since it may be an incorrect over-generalization. The extreme form of this is to make Plane_Rt (or Plane_TS) a step function which is 1 when Max_out < 1 and 0 (or -1) when Max_out = 1. This is conceptually simpler and is currently implemented. Unfortunately this results in an even greater amount of mutual interference.

What is needed is a method of limiting the training effects to those nodes most likely to correctly learn the pattern, and minimize the effects on those which are least likely to be able to. Since it is generally true that nodes with a large preexisting response to an input are more apt to learn it correctly, a rather Hebbian technique

is used which limits a node's rate of learning to its output as compared with the other nodes in the plane. The node(s) with the smallest output do not learn at all, and those with the largest output learn at the maximum rate. All other nodes learn at intermediate rates.

```
Rank := (Out - Min_out)/(Max_out - Min_out)
Rt := Plane_Rt * Rank
```

The function has subsequently been made more sensitive to the distribution of output values, but this simple form was also adequate for the functions tested. This is consistent with reports linking a neuron's likelihood of learning a pattern to its predisposition to respond to it. (Weinberger 82 pg. 65, 83, Woody 82a pg. 135).

The algorithm utilizes a global teaching signal, TS, that indicates when a node should be on, but not which one. By including a node's current output in the calculation of Rt, an element of reinforcement is introduced into the training process. Uniform reinforcement is reasonable if there is little information as to the correct node to reinforce; by spreading the credit around, the best node is apt to be reinforced some at least (for any reasonable definition of best). As the system develops, the group of potentially correct nodes becomes smaller, and reinforcement can be more selective. Only when Max_out equals 1 does that node get exclusive credit for detecting the input pattern.

The basic tradeoff is between minimizing training cross talk (which is disruptive training noise to all but the correct node) and maximizing the chance that an appropriate node is affected by the training signal. This unavoidable disruption of preexisting weights is consistent with standard theories of forgetting. In particular, intervening activity (new learning) causes disruption, and similar patterns disrupt each other more than dissimilar ones.

5.4.4 Error Driven Learning - The amount of change required of the plane can be reduced a bit. If the output of the top node is correct (equal to TS), it is not necessary that the lower nodes be adjusted, no matter what their output. The purpose of the assembly, after all, is that the top node respond correctly. To implement this, a global rate variable can be controlled with the error of the top node:

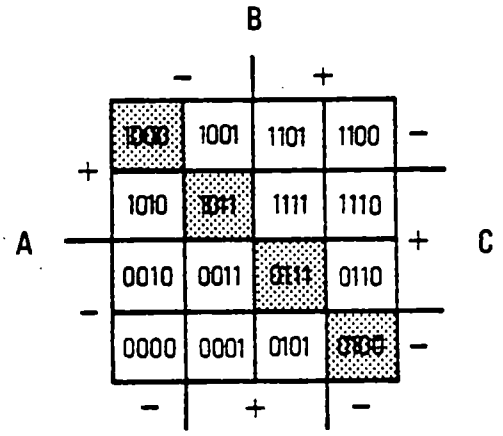
```
GRt := abs(TS - Top_out)
truncate (GRt,0,1)
Plane_Rt := Plane_Rt * GRt
```

This was generally effective in accelerating convergence, and is implemented as a runtime option. The concept of error controlled learning is central to the multi-operator system developed in the next chapter.

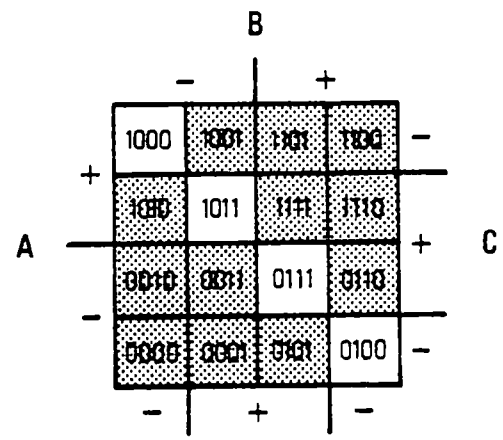
5.4.5 Detecting Negative Instances - Given almost any learning scheme, some functions will be harder to learn than others. For example, the function in Figure 5.4a can most simply be detected as ((~AB~C~D) OR (~ABCD) OR (A~BCD) OR (A~B~C~D)), which is a top level 4

way OR of 4 way ANDs. The complement of the function, (Fig. 5.4b), can be represented as ((~A~B) OR (AB) OR (~CD) OR (C~D)), which is a top level 4 way OR of 2 way ANDS. Since patterns with more features take longer to learn, it would be expected that the first would take longer. In fact the first takes 20 cycles to learn while the second takes only 10.

The current model is designed to detect when an operator should be applied. As previously discussed, this corresponds to circling the 1s on a Karnaugh map. An alternative technique is to describe the complement of the function by circling the 0s. This describes the conditions under which the operator should not be applied, and it is a simple matter to negate this function. One of the two may be much easier to learn. This suggests a possible improvement. The two processes can be run in parallel, with a final node to utilize the output of the two networks. The top node will learn to use whichever is the more useful and can combine the two to capitalize on the strengths of both. Simulations were run with this structure, and it behaves as expected. However, the increase in learning speed was usually not spectacular, and since it doubled the number of nodes needed, this course was not pursued.



a.($\bar{A}B\bar{C}\bar{D}$) OR ($\bar{A}BCD$) OR ($A\bar{B}CD$) OR ($A\bar{B}\bar{C}\bar{D}$))



b.($\bar{A}\bar{B}$) OR (AB) OR ($\bar{C}D$) OR ($C\bar{D}$))

Figure 5.4 A Boolean function and its complement.

5.5 Pattern Encoding And Decoding

Allowing more than one node in the plane to be on at a time has several theoretical advantages. In a fully interconnected net of N nodes, there are N ** 2 pieces of information stored as the synaptic weights. (The present network is not fully interconnected for several reasons. That aspect will be discussed later). If output of the plane is restricted to at most one node, at most N states can be expressed, but if any subset of nodes can be on, the plane can represent 2 ** N possible states. Restricting output to N values when N ** 2 can be stored and 2 ** N represented is a considerable waste of information, obviously a bad characteristic for a knowledge representation system.

An analogous situation occurs in other models where output is identified as distinct patterns of oscillation (self-sustaining firing patterns). Under certain assumptions of net characteristics, it has been demonstrated that only N distinct oscillation patterns are possible in a fully interconnected net of N neurons (Shaw 78).

This technique of encoding a single value (state, category, concept, feature) with a set of general nodes rather than a single, highly specialized one is used in the nervous system. For example, rather than a series of narrowly tuned color (wavelength) detectors, the color continuum is initially encoded as the relative output of 3 broadly tuned, overlapping detectors (Mollon 82). This representation technique is also used in the other senses, and has been identified as

a general principle of biological information representation (Erickson 74, Kent 81). Albus's cerebellum model explicitly incorporates this principle (Albus 79, 81). Susceptibility to damage is greatly reduced since information is distributed across a large number of nodes. This is a commonly cited "holographic" property of biological memory.

Node specificity is inversely related to the size of the encoding set, forming a continuum from high specificity (one concept, one node) to broad tuning (one concept, many nodes). This point has been made by Wickelgren:

> "... the conflict between specific-neuron theories and more holistic (distributed holographic) theories of encoding comes down to the following: When we think of an idea, is the subset of neurons that are activated above (or below) baseline (spontaneous firing rate) a very small or a very large subset of all the neurons in the cortex." (Wickelgren 79)

Distributed representation has many advantages, but it would be useless if the information could not be decoded when narrow resolution is needed, for example, to say "yellow" under appropriate conditions. (For an argument that this isn't obvious see Carlson 80 pg. 222). In Figure 5.5 a 2 to 1 encoding and decoding network is shown. It encodes N mutually exclusive input features into simultaneous activity of N/2 features. A final output node decodes the value of the original input C. If the plane is interpreted topographically, the output node's weights are analogous to the center surround weightings found in the retina (Hubel and Wiesel 79, Kuffler and Nicholls 76, Linsay and Norman 77). The Pi values are excitatory in the center and inhibitory at the sides, and the Ni weights are the inverse of this. Both weight patterns are found in the retina.

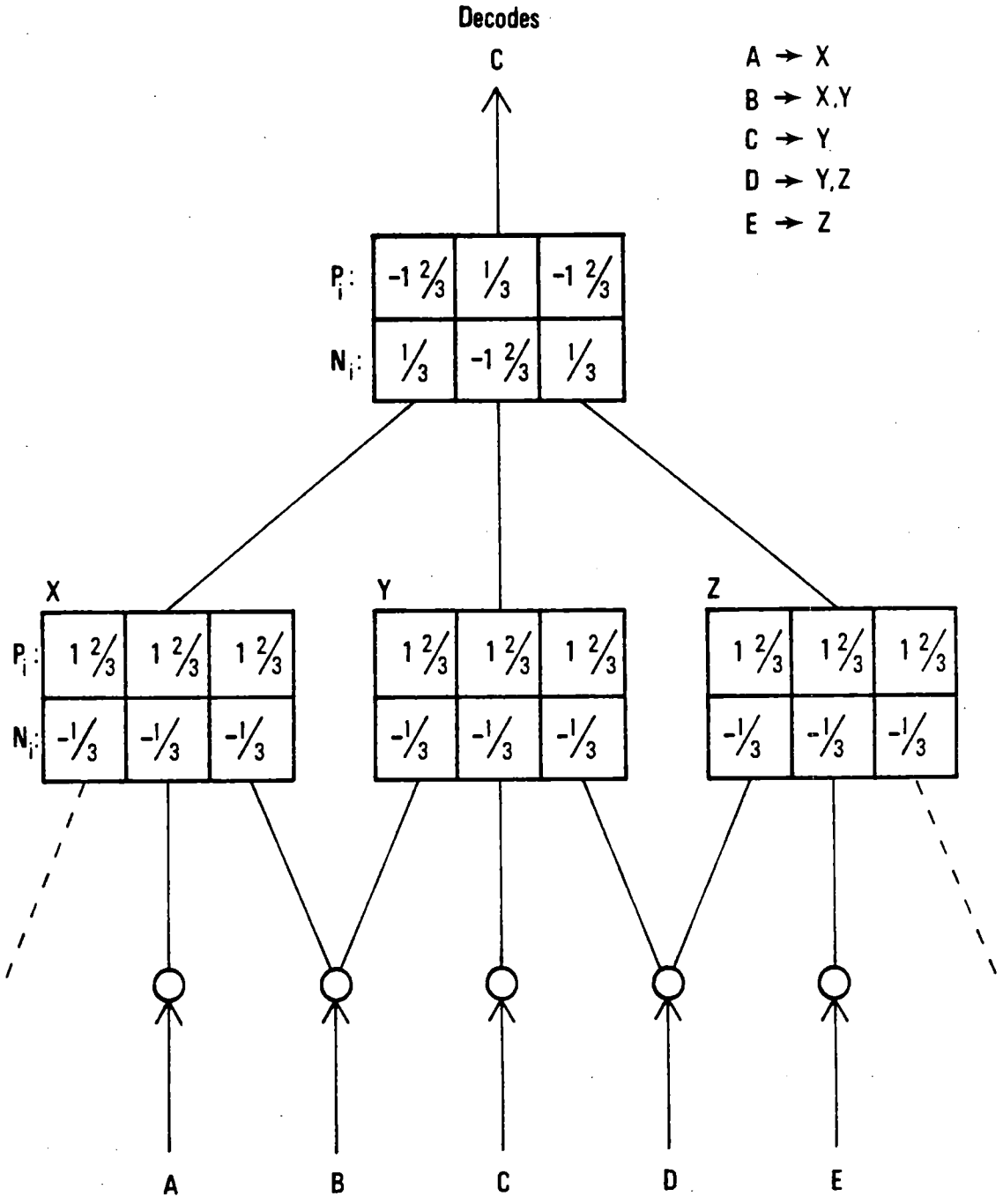


Figure 5.5 2 to 1 encoding and decoding of mutually exclusive input features.

Rather than completely decoding a distributed representation, specialized detectors need only be trained to extract the specific features which are currently useful. This way an organism can have the advantages of both extremes in information representation. In general, any encoding scheme which encodes into sets of nodes can be decoded by a single AND node. It just takes longer to learn the proper decoding for larger sets. Lateral inhibition of output is used in many neural systems to limit the size of encoding sets (Linsay and Norman 77).

While it is possible to encode up to 2 ** N mutually exclusive features (both quality and quantity, e.g., hue and intensity) as Boolean sets of N nodes, it is theoretically possible to encode an unbounded (more or less) number of features in only 2 nodes if their ratios of activity are used (Fig. 5.6). Output ratio determines quality, and total output conveys quantity. A two-level assembly of nodes is capable of decoding ratios, but the necessary weight size varies inversely with resolution, which places a practical limit on the number of distinct features that can be decoded. Though the model is computationally capable of decoding ratios, the current learning algorithm is designed for Boolean inputs and does not converge on those structures. A different approach which does will be discussed in the next chapter.

Mathematically, it is possible to encode an unbounded number of features in a single value, but that does not appear to be biologically realistic.

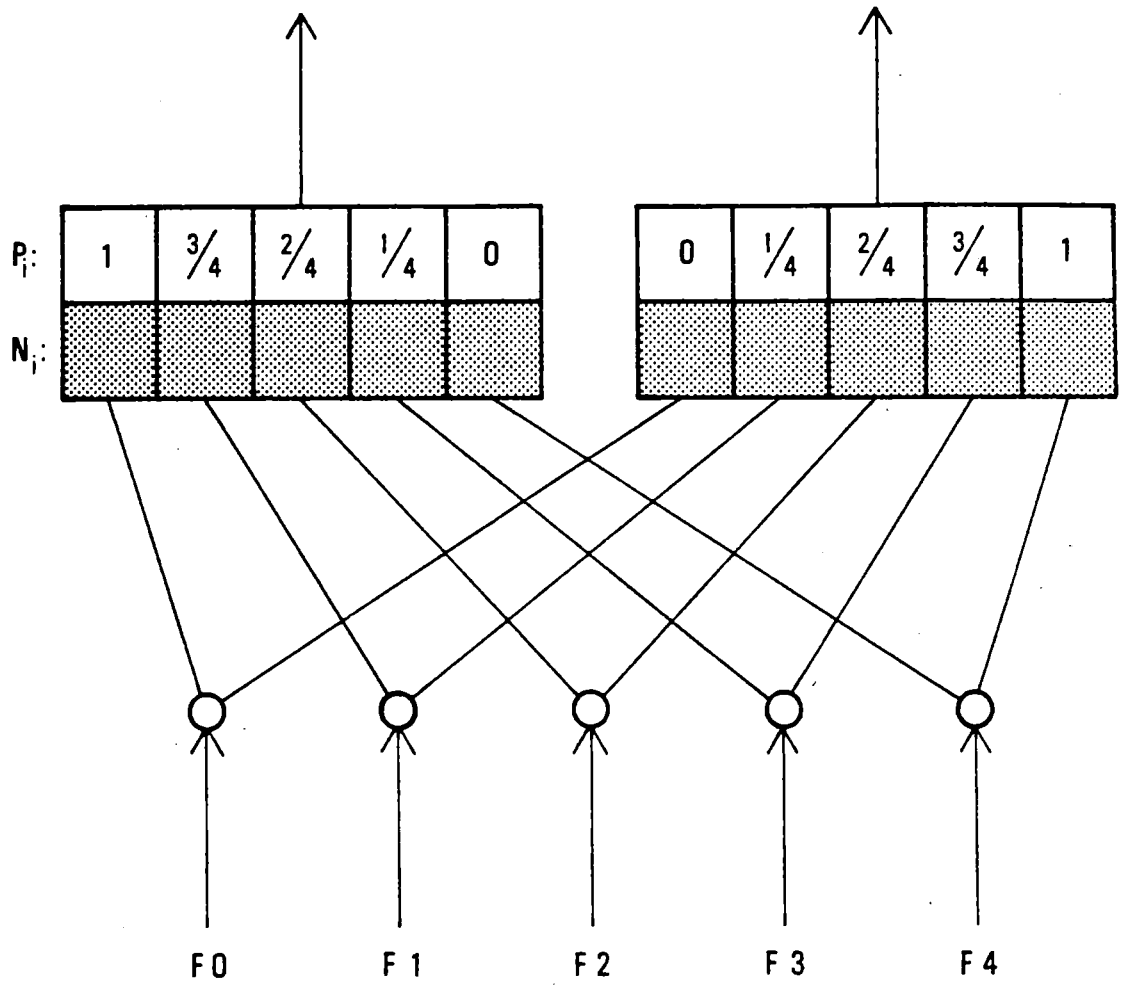


Figure 5.6 Ratio encoding of mutually exclusive input features by 2 nodes.

Using set and/or ratio encoding, single features can be uniquely represented, but simultaneous occurrence is represented ambiguously. Increasing the number of nodes in the representation decreases the ambiguity, and when the number of nodes equals the number of features, ambiguity can be reduced to zero. Having 3 color detectors in the retina (rather than 2 or 4 for instance) can be viewed as a practical compromise between economy of representation (2 detectors) and the reduction of ambiguity in color detection (4 or more).

5.6 Number Of Nodes

System capacity and learning speed can be improved by increasing the number of nodes. This was accomplished in three ways: increasing the number of nodes in the plane, increasing the number of planes to form a stack, and increasing the number of stacks.

As a general rule, it is advantageous to have more nodes in a plane than the minimum needed for the final structure. A node does best if it is taught only the correct pattern and is not confused by the training of other nodes. With only a small pool of nodes, it is difficult for all the patterns to be covered adequately and rapidly, so there is more confusion as to which node gets what pattern while they sort themselves out.

Unfortunately it is sometimes difficult to determine the minimum number of nodes needed to represent an arbitrary function in linear or prototypic normal form. One upper bound is the number of nodes needed

for an OR of ANDs representation. This is useful, but it can be misleading since an OR of ANDs representation of the function (at least 10 of 20) would suggest about 10 ** 5 nodes as appropriate, when in fact only one is needed. Despite the fact that it is not a tight upper bound, it is often a reasonable approximation. For representational completeness, the plane must have on the order of 2 ** N nodes for N features, and empirically, increasing the number of nodes per plane beyond that was not beneficial.

Under many circumstances, system performance can also be improved by increasing the number of planes. They can be stacked so that each plane sees the output of the planes below it, or they can be arrayed in parallel stacks of one or more planes. All planes can be trained independently with the same teaching signal.

There are a number of ways multiple planes can improve the top node's output. Primarily, this allows more nodes to be specialized for correct patterns, since the planes will represent the correct patterns in duplicate. Although all planes will eventually converge, they may take some time to perfectly converge. Different initial states may converge at different rates and in different ways. By providing redundant representation, the output node's learning can be accelerated. A noisy message can be decoded if it is repeated often enough.

Another reason for multiple planes is that information can be shared. For example, if one node in a plane detects ((ABC) AND D), another detects ((ABC) AND E), and a third is to be trained to detect ((ABC) AND F), it would be advantageous if (ABC) were pre-detected by a node in a lower plane (Fig. 5.7). Since the new pattern would require only 2 features rather than 4, it would be easier to learn. Conceptually, it is easier to learn complex patterns if they are formed of precomputed chunks. The present learning algorithm doesn't effectively exploit this capability, but a different one which does will be developed in the next chapter.

Since there is a statistical advantage in multiple representation, a number of potential problems can be side stepped by the introduction of statistical diversity. Increasing either the number of planes or stacks can achieve this. However, it is presumably preferable to specifically address problems of convergence rather than statistically swamp them, so further development of the model will be limited to a single stack, and usually a single plane. In practice, multi-plane systems are rather expensive to simulate anyway. The number of nodes per plane, and the number of planes are entered as runtime variables.

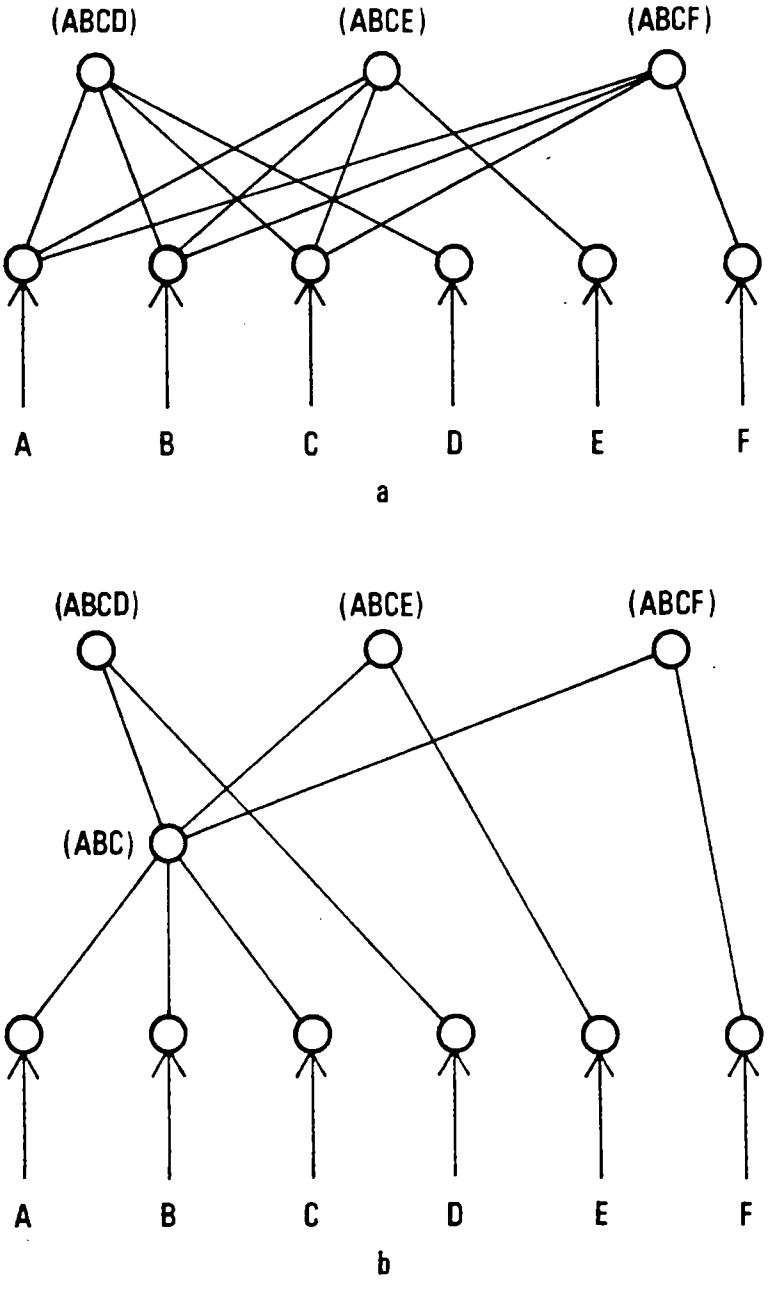


Figure 5.7 Shared concept in a multi-level system.
a) without sharing b) with sharing

5.7 Interconnection

An important concept that has not been considered in any detail is the level of interconnection within the network. The only possibilities that have been described so far are connecting a node to only the plane below it, or to all nodes below it. It is cheaper to simulate if the connections are restricted, but more powerful if each node has more information available to it. A high degree of interconnection can be advantageous, but is not without drawbacks (Ashby 60). For example, a node with a large number of inputs may be less dependent on any one of them, but is dependent on the functional constancy of a larger set of features. In a constantly adapting system, this may be a significant liability. If a small set of reliable features is available, it can only be detrimental to dilute them with less reliable ones.

Connecting the output node directly with the input as well as intermediate planes has some biological justification. The neocortex is a layered structure, and its principle output (pyramidal) cells have inputs which connect with the raw input as well as the output of other layers (Shepherd 79). In addition, different layers may receive different inputs. Deep connections are a significant advantage in layered systems. If each layer were connected only to its nearest neighbors, the same information might have to be sequentially learned by each layer before it could be utilized at the top level. By connecting a node with all nodes below it (or conversely making its output available to all nodes above it), the top level can use

information as soon as it becomes available anywhere in the network.

The logical extreme of this principle is to connect each node with all other nodes. This is obviously more expensive since the number of weights grows as N ** 2, but the number of distinct communication channels is only N since a node's output is essentially broadcast. A fully interconnected net has the advantage that every node has access to all information represented in the network.

An interesting result of this is that a plane can learn lateral inhibition of output, a property that is hard-wired in many model and neural systems. This is most obvious when the output of a plane is restricted to exactly one node. During simulation, nodes rapidly learned that any activity in the same plane is inhibitory, though there was no structural predisposition toward this. Such an ability can be advantageous, since mutually incompatible interpretations develop mutual inhibition. Mutual inhibition gives rise to the phenomena of relaxation, in which different interpretations of an input compete with one another until only one (hopefully the best) is left. This has been identified as an important aspect of biological information processing (Anderson and Hinton 81, Feldman 81).

Conceptually, recurrent connections also allow reconstruction of unknown features (e.g., McClelland and Rumelhart 80, Rumelhart and McClelland 80, Kohonen el al. 81). For example, if 4 of 5 features are present for a 5 way AND and the last feature is unknown (Output = 0), the AND may decide that the pattern it detects is most likely

present. If the unknown feature is able to look upward, it will notice that a highly predictive node is on (a node detecting a pattern that the feature is a part of). The unknown feature may then conclude that it is probably present also. If features can see each other, the parts of a pattern may also learn to recognize each other. The relative effects are determined by the connection weights which reflect the network's training experiences.

5.8 Feedback

An important property of many recurrently connected neural networks is the potential for self-sustaining activation or oscillation (Oguztoreli 79, Amari 72). Such structures can exist in the proposed network (e.g., cycles of Pi = 1 or odd length cycles of NOTs), but they are not stable in the long run. This results from the nature of the reinforcement system. The system is designed to reinforce useful structures, and self-sustaining activity is not useful in this model. Ideally, if the learning process is correctly designed, the information processing system won't develop pathological structures.

With recurrent connections, the development of some positive feedback is inevitable due to a basic feature of the learning algorithm: learning takes place after several passes of output calculation. By failing to distinguish between input values that appear before a node's own output is introduced into the network and

those that appear afterward, pre and postdictive features are confounded. Postdictive features may reflect useful information, but may also simply reflect a node's own output, which is not useful and can in fact be dangerous. For example, in a partially trained system, a node's own output can be more correlated with its correct output than any of its "real" input features. (In a perfectly trained system it's 100% correlated). Consequently the node will begin to selectively listen to itself after it has reached some minimal level of competence.

This is similar to the problem of a director surrounded by "yes men". Only their opinion prior to the stating of his own is of value, since afterward it is simply a restatement of his own. If the director were to believe this postdictive input (assign positive input weights to it), a cycle of self-sustaining opinion could develop totally decoupled from external reality.

The most direct feedback loop can be avoided by not connecting a node with itself. Though useful, this does not solve the problem in general. Consequently, in fully interconnected networks, learning was limited to those features which were present before a node's own output could significantly affect the network, usually before it reached a magnitude of .3. This is an effective, but rather heavy-handed approach.

A possible biological example of such a function has been described in the crayfish tail flip response (Krasne 78). The crayfish tail flip is similar to gill withdrawal in that the triggering feature (poking the abdomen) can be habituated. An important difference is that a tail flip has the effect of massively stimulating the poke sensors. Thus one tail flip would cause a series of flips and habituate the response. Similar to the problem encountered with recurrent connections, a cell's output can influence its own input in a detrimental way. The crayfish's solution to this problem is presynaptic inhibition of the sensor's input to the flip motor neuron when that cell fires. By briefly ignoring certain inputs after it fires, the motor neuron is not misled by the effects of its own output. Krasne suggests that this may be an important technique for controlling plasticity in the nervous system.

With bottom-to-top, unidirectional information flow, a network's output can be calculated in one pass from input to final output. With recurrent connections, several passes must be allowed as information circulates in the network. Output levels usually stabilize after 4 or 5 passes, though it is not crucial that they do so. The system is given a fixed number of iterations to make up its mind, and as long as output is correct after the last pass, system behavior is correct. The number of iterations is entered as a runtime variable.

The large number of weights and the extra passes of output calculation required for complete interconnection take a heavy toll on the speed of simulation, so while it is implemented as a runtime

option, full interconnection is seldom used in practice. An intermediate compromise might be to connect each node with all nodes below it and all nodes in the same plane. If output in the plane were computed synchronously, all nodes would have one cycle to use the output of other nodes in the same plane without danger of feedback.

5.9 Weight Patterns

Controlling either TS or Rt will train a network to detect Boolean functions. The result is roughly equivalent to an OR of ANDs circuit designed by circling groups of 1s on a Karnaugh map. If TS is controlled the groups do not overlap, but by controlling Rt, overlap is possible (and desirable). For some Boolean functions the algorithm converges on exactly an OR of ANDs, but since nodes are not limited to being OR and AND, other final states are possible. If the top node is connected only to the plane, the first approach of controlling TS forces the top node to be an OR, though the lower nodes are still not necessarily trained to be ANDs. If the top node also receives input from the feature detectors, it is no longer constrained to be an OR since it can detect patterns of simultaneous activity in the detectors and the plane. The second approach of adjusting Rt can result in somewhat more complex weight patterns. Error driven learning and recurrent connections further complicate interpretation of the weight pattern the nodes develop. If ratios were decoded, the results would presumably be more opaque yet.

This progress from clarity to obscurity results from the continual relaxation of constraints on correct network structures. By relaxing constraints, the number of correct structures increases and the amount of change needed to reach one decreases. Thus capacity and learning speed are bought at the price of decreasing perspicuity of the resulting answer.

5.10 Results

Using a small network (one plane, 20 nodes in the plane), the operator training algorithm was tested on a set of 40 Boolean functions of 4 features (Fig. 5.8). For each function, each of the 16 input patterns was presented in constant rotation until the accumulated error of the output node (its difference from TS) reached zero for a complete cycle. All functions were successfully learned in an average of 12 cycles. In Figures 5.9abc, 3 of these functions and their learning curves are shown. In Figure 5.10 a single learning curve is shown for the network as it is sequentially retrained to detect those three examples. Examples 5.9b and 5.9c are about the hardest and easiest functions of 4 features for the network to learn. As can be seen in Figure 5.9b, error does not necessarily decrease monotonically. This results from training the output node while the plane is still incompletely trained.

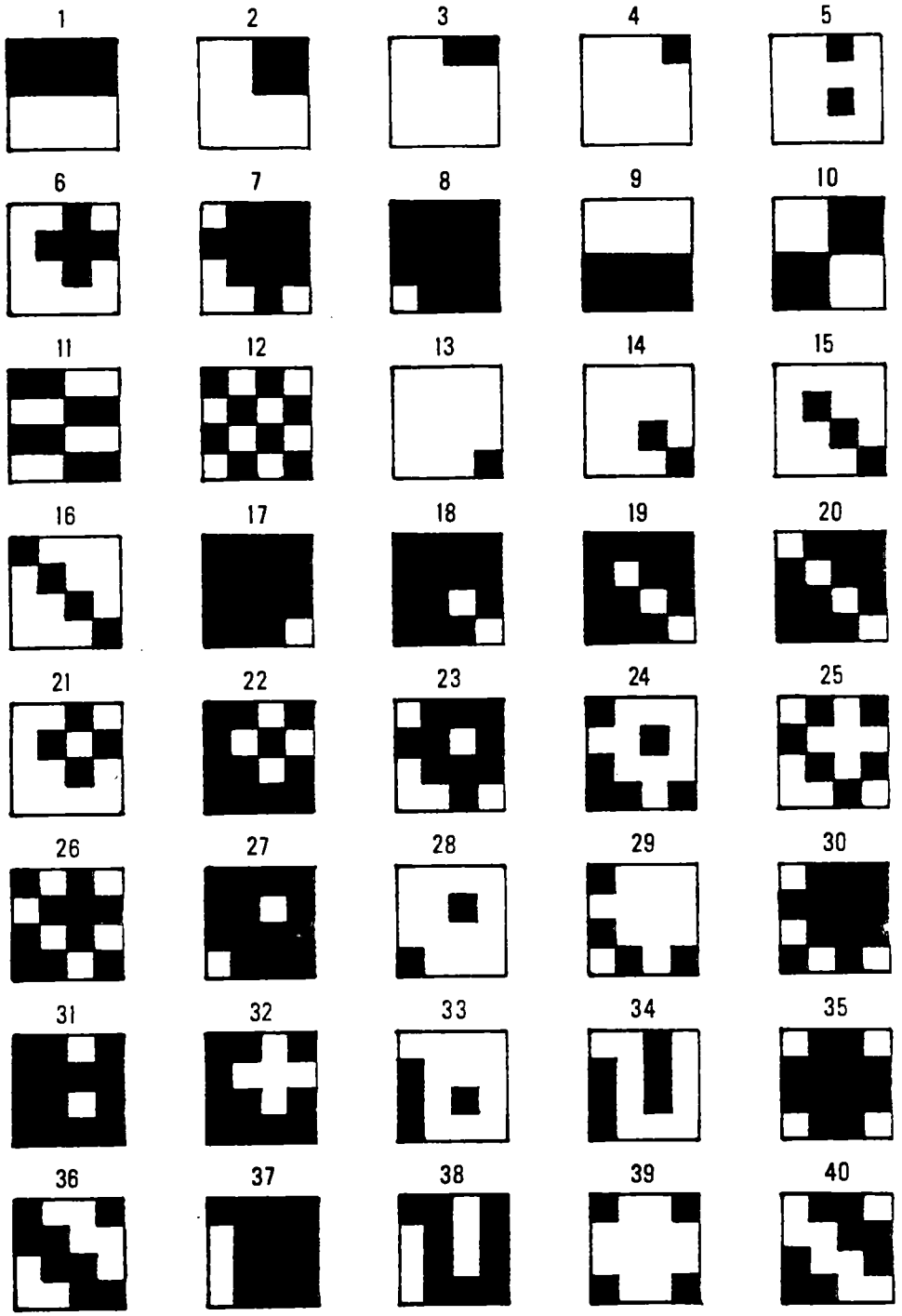


Figure 5.8 40 Boolean functions used for program development.

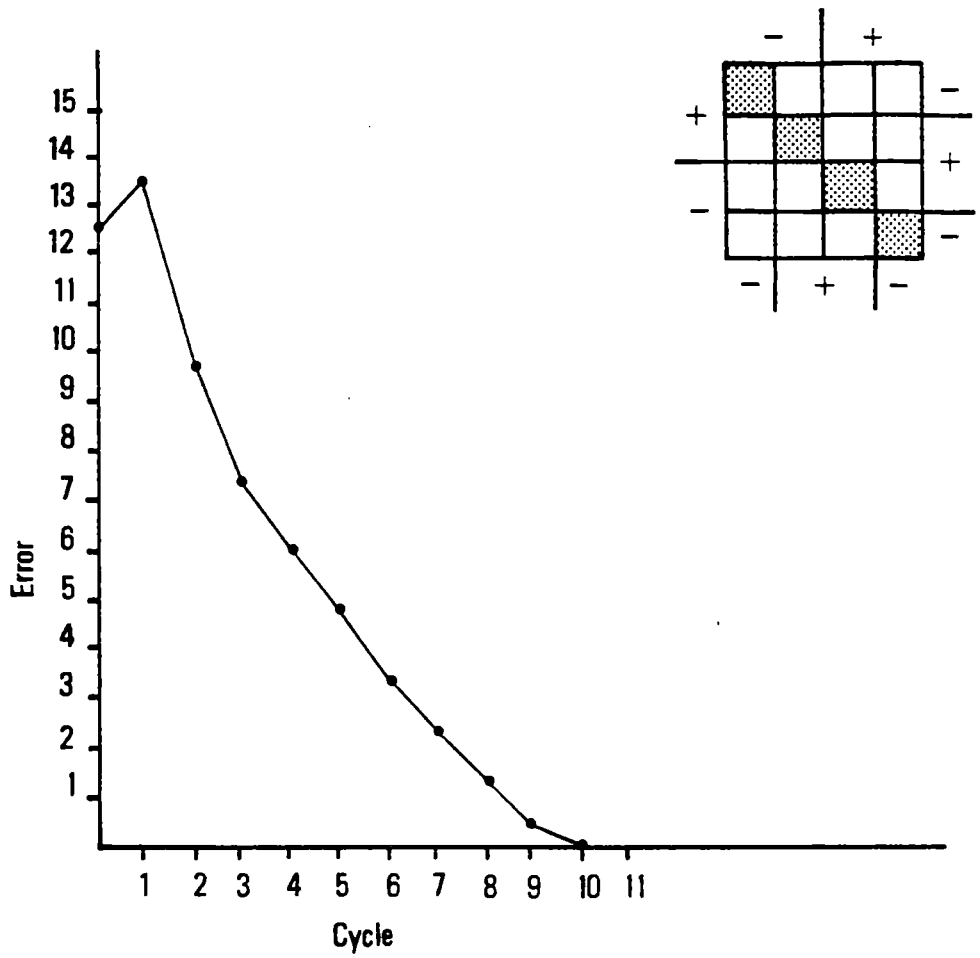


Figure 5.9a A Boolean function and its learning curve.

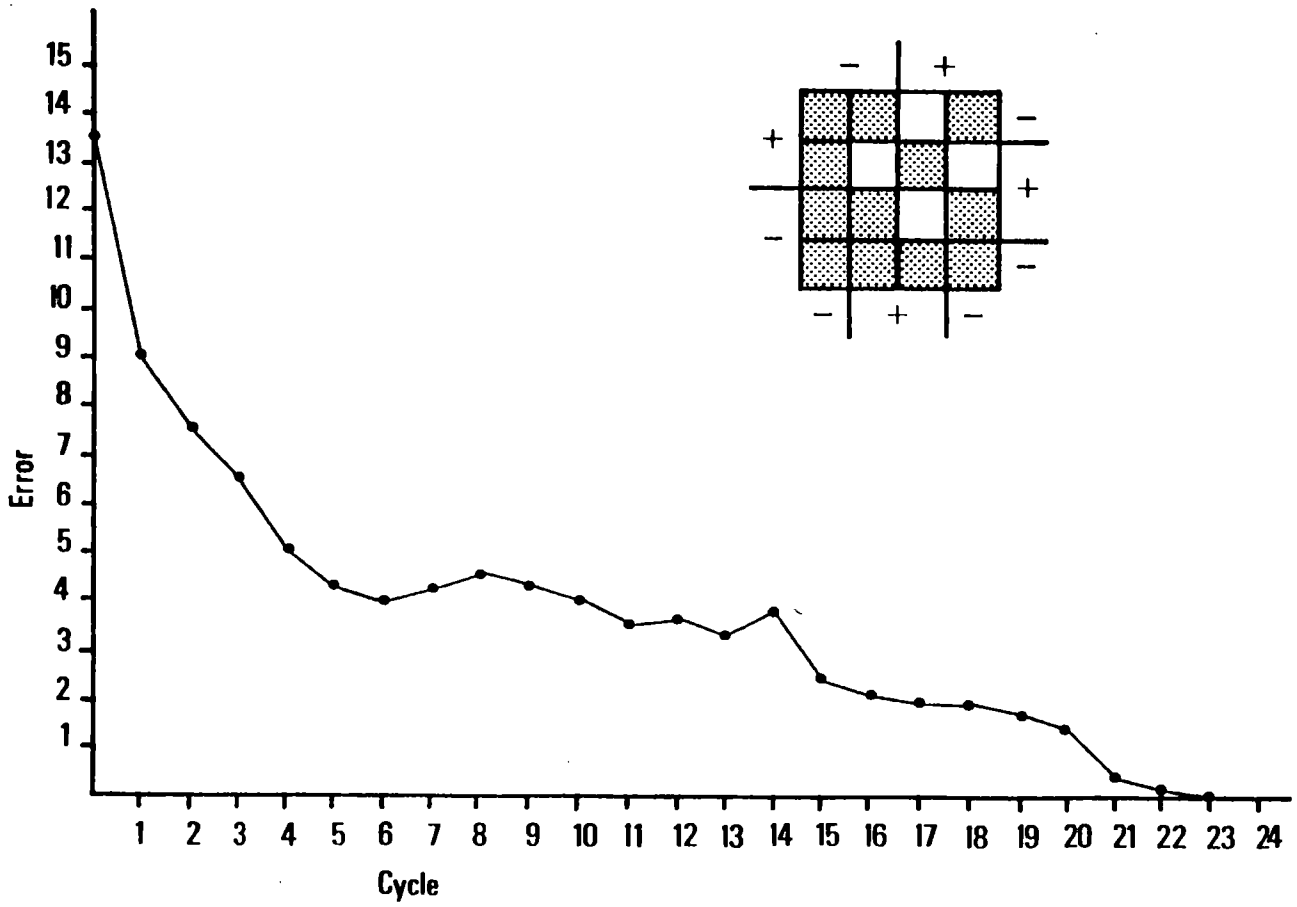


Figure 5.9b A Boolean function and its learning curve.

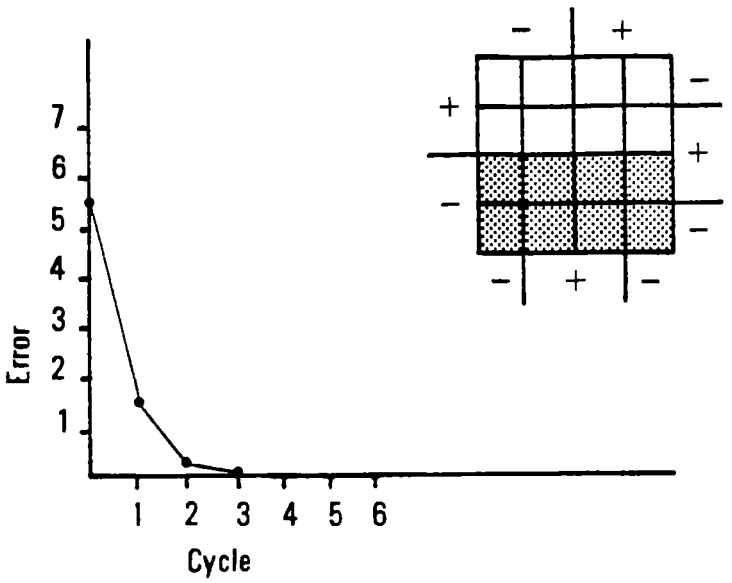


Figure 5.9c A Boolean function and its learning curve.

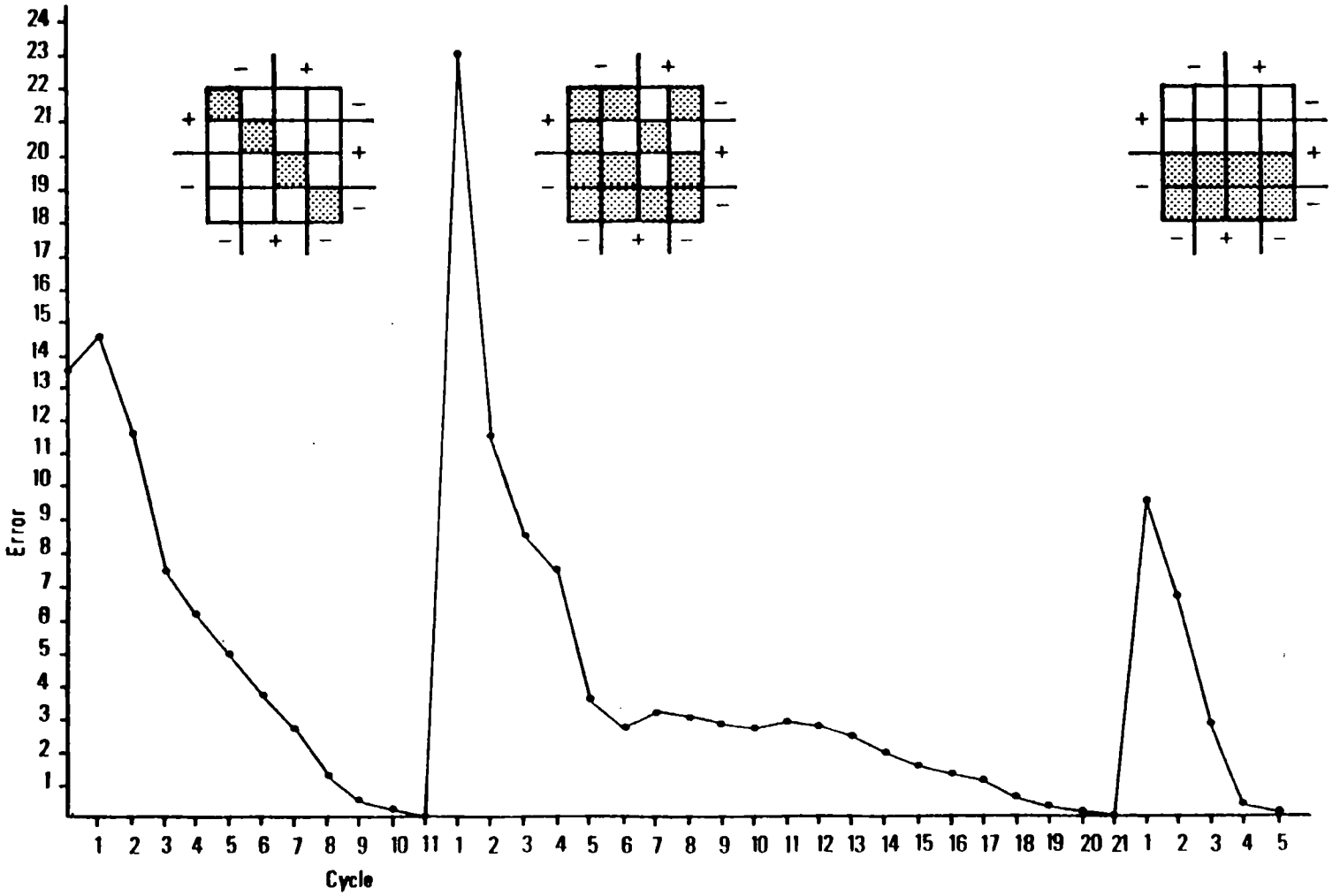


Figure 5.10 Sequentially retraining a network to detect the previous 3 Boolean functions.

With only 4 features, the current algorithm appears to be complete. However, the "missing prototype" function (Fig. 5.3a) becomes increasingly hard to learn as the number of relevant features is increased. With cyclic presentation, the plane may fail to converge. Using a random presentation order the plane eventually converges, but still with considerable difficulty. The missing prototype function points out a potential problem with the algorithm; it can be overly eager to generalize.

Rather than address that issue immediately, a different learning system will be introduced in the next chapter. Once developed, it can be superimposed on the operator training process to guarantee arbitrarily rapid convergence, though with a parallel reduction in the ability to generalize.

6.0 A TRAINABLE BEHAVIOR SYSTEM

6.1 The Goal

A logical extension of a single Boolean operator is a Boolean behavioral system, mapping inputs to sets of output actions. If only one operator is applied at a time, the result is a production system. Many AI domains, such as game playing, can be appropriately modeled as the sequential application of single operators, but competent sensory-motor behavior is apt to require simultaneous operator application. Biologically this is certainly the case. The model developed in this chapter is appropriate for either single or simultaneous operator application, though it is trained only as a production system.

6.2 Shared Memory

A trivial but effective approach is to individually train a collection of separate operators. Although it would produce the desired behavior, this technique has serious drawbacks. The most obvious is that providing a separate "brain" for each operator is impractically extravagant. Another objection is that totally separate operators cannot share information. If the same pattern is important to several operators, it must be learned and represented separately by each of them.

The latter objection can be overcome by interconnecting the nodes of the various operator networks, but keeping their reinforcement systems separate. The number of nodes is not reduced, but information can be shared. (Henceforth only nodes in the final output level will be considered as operators; the lower level(s) function as common memory). Unfortunately, the first problem can't be solved in such a straightforward manner. The individual teaching signals can't simply be merged to train a shared pool of nodes. The learning algorithm was designed for a single operator so that any activity when the operator should be on was good, and any activity when it should be off was bad. With multiple operators, some will be on and some off most of the time. Only when all operators should be off can all nodes in the shared memory be trained to be off.

Actually, even that point is debatable. It has been questioned whether a state of behavioral quiescence is a state of operator inactivity, or a state of actively applying the "do nothing" operator (Seligman 75). The psychological implications are different, and the model differs as to whether a "no-op" operator is implemented or not. One implication of having such an operator is that there are now no inputs for which TS = -1 can be broadcast to all nodes. For an active organism, those situations may be vanishingly small and of little relevance to the learning process anyway.

Conceptually, the two alternatives are not mutually exclusive. Inactivity may result from indecision (nothing comes to mind), or from highly specific knowledge (if you don't move, it won't bite). The

model is usually run without a no-op operator, but the blank spots on a Karnaugh map can just as easily be filled in with its specific application. This is implemented as a runtime option. When inactivity is appropriate, purposefully doing nothing is a more reliable response than indecisively doing nothing, so it would make sense to convert the latter to the former.

The operator training algorithm is simply inappropriate for a shared memory. Because of this, a different learning strategy is utilized. It has two components:

1) Input driven categorization
2) Goal driven focusing

These two learning processes will be considered separately. The approach is similar to one utilized in (Reilly et al. 82). For simplicity, operators are limited to single nodes and the shared memory to a single plane since that structure is minimally complete in the Boolean domain (Fig. 6.1).

6.3 Input Driven Categorization

Input driven categorization can be summarized as:

At least one node in the common memory should be on for any input.

This introduces an intrinsic learning process in the plane. Independent of the system's output, learning takes place until input is categorized one way or another. Learning is identical to that in the previous chapter, except that TS is equal to 1 for all inputs. There is good biological evidence for input driven learning (Spinelli

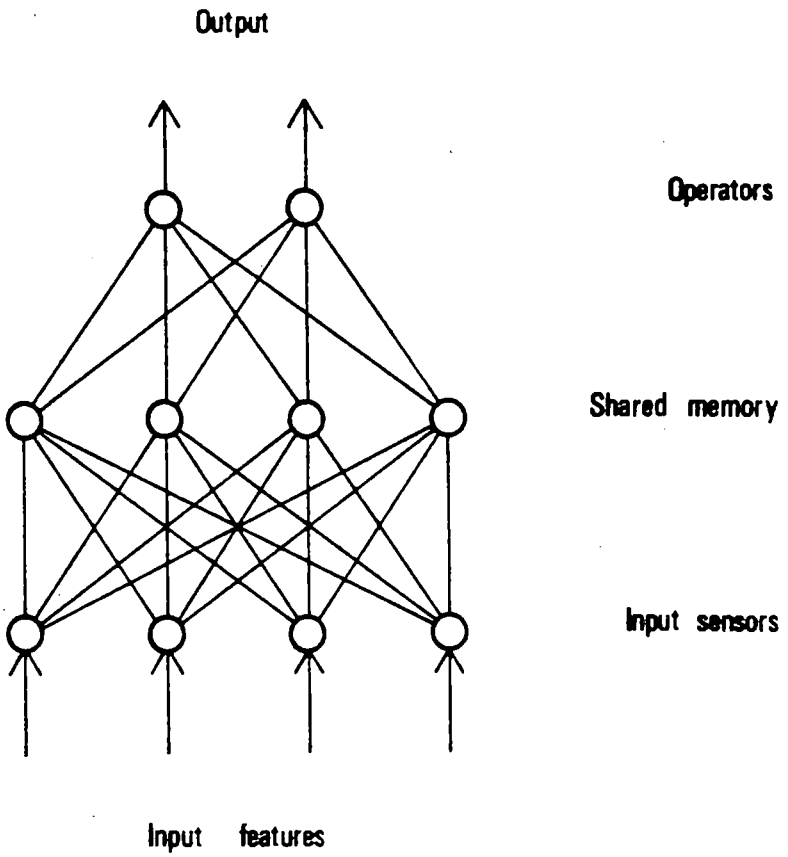


Figure 6.1 Minimal structure capable of arbitrary input-output mappings in the Boolean domain using linear function nodes.

et al. 72).

Useful categories tend to reflect the natural categories in environmental input (Mervis and Rosch 81), so input driven learning should capture those natural associations. Many neural learning models based on the Hebb hypothesis are designed to model this process. Given the specific needs of a particular type of organism, the structure could also be innately prebiased toward detecting those categories which will most likely prove useful.

6.4 Goal Driven Focusing

Goal driven focusing can be summarized as:

If an output error occurs, the common memory does not represent the current input pattern in a specific enough form for the operators to use. Therefore the state of the common memory should be adjusted to produce a more specific representation.

The same experimental approach demonstrating input driven learning (Spinelli et al. 72) has also been used to demonstrate goal driven modification (Spinelli and Jenson 79,82).

With a single operator, output error can be computed as the difference between the operator's actual and desired output. For multiple operators this has to be modified, but if at most one operator is applied at a time, only minimal modification is necessary. Only the output (Op_out) of the applied operator (Op_node) is used:

```
1) If Op_node is the Correct_op then
       error := 1 - Op_out
2) If Op_node is not the Correct_op then
       error := 1
```

A more sensitive measure might include the output level of other operators. Memory should be adjusted when error > 0. This can be implemented as a step function, but works more smoothly if the amount of adjustment varies continuously with error. In general, the amount of adjustment should be proportional to the probability that a more specific representation would improve behavior.

In the current model, operators are single nodes detecting (at least X of N) features, so any memory modification should converge on a representation which is decodable by that function. An obviously decodable extreme results if each input pattern is uniquely detected by a node in the lower plane. The operators could then pick and choose, functioning only as ORs. Any learning process which converges on this state will eventually converge on correct behavior, although it may need up to 2 ** N nodes to completely cover an N feature input space.

In order to implement this process, large categories must be subdivided into smaller ones. One mechanism for achieving this is to focus the most specific current category more narrowly on the current input whenever an operator error occurs. If the error persists, the category should ultimately be focused down to a single input pattern. Intuitively, focusing a node means that it fires more selectively for inputs resembling the current input. This requires:

1) Shifting its central prototype (all N of N) toward the current input
2) Sharpening its discrimination by increasing X in (at least X of N)

If carried to completion, the combination of these two processes will eventually focus a node on a single input pattern. Input driven learnings fills in any conceptual gaps left by the focusing process.

The specific learning characteristics of focusing may be important when observing neural learning processes. In operator training the effects of learning are immediately obvious; a node increases or decreases its response to the current input. With focusing, a node might modify (reduce) its output for everything except the current input. If no immediate output changes are observed, this could lead to the mistaken conclusion that no learning took place. Such decreases in background firing rate may be quite common (Weinberger personal communication).

The general process of training the common memory to detect increasingly specific input categories has a number of appealing features. Most important, it is a (potentially) complete technique for learning Boolean functions. It is also compatible with biological evidence for both input and goal driven learning phenomena. In addition, focusing is a reasonably efficient process, since network modification takes place only when behavior is apt to be improved. If no output errors occur, the representation must be specific enough for perfect behavior, so no learning is required. If an error does occur, the representation is not specific enough at that point. Through successive focusing, the area of error is identified and can then be used by the operators to correct their output. Thus any detectable deviation from optimal behavior creates a change which shifts behavior

toward optimality. Since the general purpose of biological learning is presumably to improve behavior, the system is a fairly general model of adaptation.

With more powerful decoding capabilities, the operators could detect a wider range of functions, and the shared plane would have to change less to reach a decodable state. The current operators are single nodes, but if an operator had a small private network (as it did in the previous chapter), it could decode an additional range of patterns. Clearly there is a tradeoff between the resources allocated to individual operators vs the common memory. As usual, the current model is a sufficient, but not necessarily efficient implementation.

6.5 The Focusing Mechanism

A neural system displaying what might be interpreted as focusing has been described in the hippocampus (Alger and Teyler 76, Dunwiddie and Lynch 78, Anderson et al. 80). In that system, the current inputs to a neuron become more effective in firing it, and the unused inputs become less effective. Its firing function is thus modified to look more like the current input. This is apparently achieved by simultaneously strengthening the synapses of the active inputs and raising the firing threshold of the cell as a whole. This learning process has been extensively studied only in the hippocampus, but may be functional throughout the cortex (Lee 82).

The proposed model neuron does not have an adjustable threshold, but a similar effect can be achieved by increasing the negative weights on the inverse or "not" of the current input features, (making their absence more inhibitory). For biological realism, (and some practical reasons) the model was also modified to use an adjustable threshold, but since the desired behavior can be implemented without structural modification, the threshold technique was not extensively developed.

6.5.1 Invert Focusing - The primary method of adjusting weights by "invert" focusing is:

A) Center prototype
   1) reduce all positive weights
   2) reallocate weight based on current input
B) Sharpen discrimination
   1) shrink current output toward 1
   2) invert current input
   3) increase negative weights

This has the desired effect of driving the node's output function toward (all N of N) of the current input features. Focusing is activated only when Max_out >= 1, and only one node (the most specific current category) need be focused. This keeps the computational burden of focusing to a minimum, while still assuring that the plane converges on a decodable representation (or on a state containing a decodable representation to be more precise). This is not to say that single node focusing is an optimal adjustment process, just that it doesn't appear to be too bad.

6.5.2 Threshold Focusing - The threshold focusing process utilizes a variable threshold and focuses by:

A) Center prototype
   1) reduce positive weights
   2) reallocate weight based on current input
B) Sharpen discrimination
   1) shrink current output toward 1
   2) increase threshold toward 1

This is more realistic, but still contains significant modification used to avoid weight saturation. In the biological model, weight saturation apparently can occur (Barnes 79, Baudry et al. 81), though the learning capability is probably designed to last somewhat longer than the organism. Rather than permit unbounded weights or build inevitable senility into the model, a form of unlearning is included. Weights are redistributed while output is reduced toward 1, and the threshold varies between 0 and 1. The biological concern of how best to distribute plasticity over a specific life span was also not addressed.

By eliminating the "invert input" step, the independence of input signals and their complements is restored. In fact, this reintroduces the technique of representing a category and its complement with different nodes. A greater number of nodes are needed, and learning is a bit slower, but the technique appears to be equally viable.

One advantage of using a variable threshold rather than negative weights, is that the threshold value provides a single direct measure of how "focused" a node is. This suggested an interesting learning strategy. Since unused, unfocused nodes are ideal for filling gaps in

the Karnaugh map (starting new concepts), it is advantageous to have a ready supply. Consequently, whenever the pool of unfocused nodes drops below 10% of the total pool, all nodes are defocused slightly by reducing their thresholds. That value was rather arbitrarily based on an observation that about 10% of the neurons in the brain may be measurably plastic (Bures and Buresova 70). This background process of neural "garbage collection" was effective in recycling old, underutilized nodes to satisfy current learning needs.

Another advantage of threshold focusing is that it can train a network to decode ratios. As previously observed, ratio encoding is biologically common and has several theoretical advantages, suggesting that threshold focusing may be the more promising for future development. However, to avoid complicating the original structure of a node, invert focusing will be used in subsequent examples. Threshold focusing is implemented as a runtime option.

6.6 Rate Of Focusing

The speed of operator convergence can be adjusted by changing the rate of focusing. The faster the plane is focused, the sooner the operator level can converge on correct behavior. Unfortunately, a greater number of nodes are required. Ideally, memory modification would not occur beyond the minimum required for correct operator application. Some overshoot is inevitable since the operators require time to utilize the plane's output, but the slower the rate of

focusing, the smaller the overshoot. Somewhat arbitrarily, the focusing algorithm was adjusted so that the 4-feature test set was learned at about the same rate as with the operator training algorithm developed in the previous chapter.

People sometimes display what is sometimes called one-shot learning. After only a single presentation of a particular pattern (a picture for example), it can be reliably recognized for weeks. If desired, the focusing process could be run to completion on a single presentation in order to produce such instantaneous learning. However, as previously observed, representation by broadly tuned, overlapping categories has several advantages over narrowly tuned, highly specific categorization. In addition, one-shot learning runs the risk of being too specific. By inadvertently including a number of irrelevant features as necessary, that specific pattern may never be encountered again. By incrementally focusing, the relevant features can be identified with conditional probability. Sensory habituation, selective attention and novelty detection are all methods of identifying relevant features for an organism, and so can presumably reduce the hazards of rapid learning.

6.7 Common Memory Focusing Vs Operator Training

Biologically, there appears to be a functional distinction between behavioral (procedural) learning and the encoding of specific, behaviorally uncommitted (declarative) information (Squire 82, Squire

et al. 83, Kent 81). The difference between operator and common memory training is consistent with this functional dichotomy. Operator training is inherently output oriented and is inappropriate for learning specific instances. Focusing on the other hand, is behaviorally uncommitted and is quite capable of learning specific instances, in one shot if necessary. It has been suggested that hippocampus-type learning (focusing) is a unique invention of the higher vertebrates (Lynch and Baudry 83).

This functional difference results directly from the basic processes of learning. An operator learns to be on when presented with positive instances and to be off when presented with negative ones. Consequently, it requires a number of input presentations to completely distinguish a particular pattern from related ones. Focusing learns both when to be on (for the current input), and when to be off (anything else) when presented with a positive instance. The resulting one-shot learning capability is necessary for the encoding of unique events.

6.8 Convergence

The operator training process only empirically demonstrated convergence with random order input, but an intuitive argument can be made for convergence of the shared memory. First, it should be noticed that focusing a single node is trivially convergent; a single presentation is sufficient to learn the AND of a collection of input

features. Incremental focusing simply does it in smaller steps. Secondly, focusing always shrinks the category a node describes. Because a node is focused only if its output is greater or equal to 1, it can't be continuously focused on more than one pattern. As its discrimination sharpens, competing focusing points drop out when their output drops below 1. The only other possible source of interference is input driven learning. This does cause some defocusing of useful nodes, but unused nodes are also pulled into gaps in the Karnaugh map. If useful nodes are later refocused, only unused nodes are left.

Focusing can be superimposed on the operator training process. This reduces its ability to generalize, but generally increases the rate of convergence.

6.9 Number Of Nodes

The shared memory learning process requires more nodes than operator training since it has only a minimal tendency to generalize (effectively none), and at least one node must be on for all inputs, rather than correct ones only. As before, the system generally does better with a surplus of nodes, though the minimum can be hard to determine. Pathological "checkerboard" cases (e.g., Fig. 6.2) may require on the order of 2 ** N nodes, but most cases are not so bad. (Interestingly, Figure 6.2 requires only 4 nodes). Of course, in real life learning, only a small percentage of possible feature combinations are ever encountered or need to be learned in perfect

detail. The possibility of set and ratio encoding reduce the average number of nodes needed, but since optimum encoding is not guaranteed, the contribution is difficult to quantify.

As in the single operator system, the number of nodes in the shared memory can be increased by increasing the number of nodes per plane, the number of planes per stack and the number of stacks. Multi-plane systems can be trained by applying the focusing process to each level. Multi-stack systems were not implemented for the same theoretical and practical reasons they were not pursued in the single operator system. The number of nodes per plane and the number of planes are entered as runtime options.

If only a single node in a plane is focused, the "signal to noise" ratio decreases as the number of nodes per plane increases. "Noise" in this case is slightly predictive categories which can share the learning credit. The model is usually run with 20 nodes per plane, but it has been run with up to 200 without noticeable degradation. However, with increasing node number, some modification would eventually be necessary. Hard-wired lateral inhibition appears to be a viable approach for limiting the size of encoding sets.

A possible objection to uniform training of a multi-plane stack is that changes in the top level affect only the appropriate weight patterns of the operator level, while changes in the bottom level potentially disrupt weight patterns in all other levels. A straight forward modification addressing this problem is to reduce the rate of

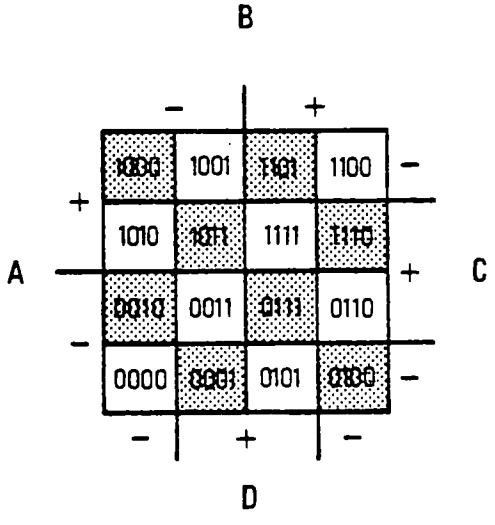


Figure 6.2 4 feature "checkerboard" Boolean function.

change in the lower levels. This was implemented by halving the error signal at each level. In multi-level systems, this often increased the rate of convergence on the desired output. If an output error can be rapidly corrected in the upper levels, the lower levels are modified very little. If errors persist, modifications are pushed deeper into the memory network. By learning the more general patterns in the deeper levels, sharing of information is facilitated. Over a period of time, the most abstract categories are learning in the deepest, most "visible" levels. Realistically, multi-modal sensory information doesn't come together until the "middle" of the nervous system, making that the conceptual bottom of the current model (or top, depending on your point of view). Tapered learning is implemented as a runtime option.

This arrangement produces another form of long and short term memory. Like the dual (or multiple) weight mechanism previously proposed, tapered learning provides some noise rejection by reducing the rate of change in the lower levels without sacrificing speed of adaptation in the upper levels. For the same reason, short term, episodic adjustment is also possible. Unlike the previous mechanism in which long and short term memory reside in the same synapse, long and short term representations are in different nodes in different planes. Tapered learning could also have been implemented in the operator training process, giving it an episodic memory capability. A range of memory spans appears to be a generally useful property for noise rejection, and could probably be implemented to protect any

trainable values in the system.

6.10 Interconnection And Temporal Encoding

The nodes of the common memory can be interconnected in a variety of ways, as they were in the single operator system. For the extreme case of full interconnection, a number of test runs were made. Because of the different learning strategy, the danger of positive feedback loops of more than one node was considerably reduced, though a direct connection of a node to itself was still troublesome. Convergence was not accelerated, but the network did converge.

In a fully trained, fully interconnected system it is interesting to note that different input patterns produce different temporal firing patterns in the same node. Only the final output value represents a node's decision on categorization, but a post-stimulus trace of its output can be much more informative than its final output alone. This is in keeping with John's observations that post-stimulus firing patterns can accurately distinguish an animal's categorization of input (John 76, 80, John and Schwartz 78).

A fully interconnected network (one operator, one plane, 20 nodes, 4 iterations) was trained to detect the function in Figure 6.3. After the network was completely trained, the post-stimulus firing pattern of the operator node was recorded for each of the 16 inputs. Many input patterns can be distinguished on the basis of shape alone, and if output levels are considered, most inputs can be easily

identified. Nodes in the common memory showed different, but often equally distinctive firing patterns.

This demonstrates an alternative to the "labeled line" theory of information representation. Rather than representing a concept as a particular state of neural activation, it can be represented as a temporal firing pattern. Not only the final state of activation, but the pattern of spreading activation (Ratcliff and McKoon 81, Quillian 67, Collins and Loftus 75, Collins and Quillian 72, Anderson and Hinton 81) conveys useful information. The temporal pattern might vary from cell to cell as it does in the current model, or may be relatively location independent as in John's observations.

Utilization of temporal information could have important consequences for the appropriate interconnections of an assembly. For example, in order to use all information in the common memory, an operator currently must be connected to all nodes in it. If an operator could decode temporal sequences, it could tap in to a highly interconnected memory anywhere and still have access to its entire store of information. Complete interconnection within the common memory is expensive, but the necessary connections between it and the operators are drastically reduced. Actually, complete interconnection is not an absolute necessity, since even unidirectional information flow can produce distinctive firing patterns as successive waves of analysis reach the higher levels. Increasing interconnection simply increases the potential complexity of firing patterns.

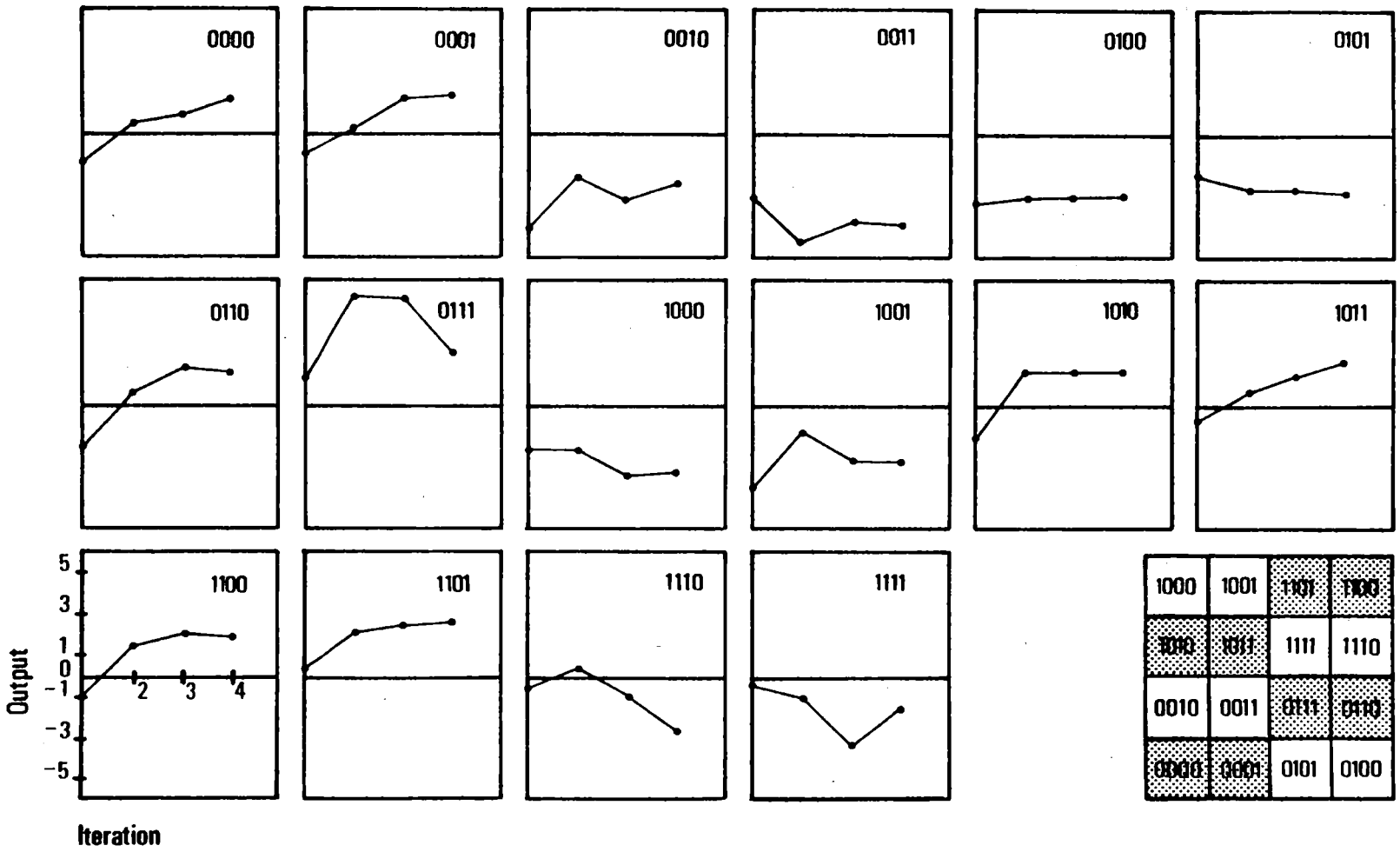


Figure 6.3 Post stimulas trace of operator output for each input of a 4 feature Boolean function.

While these model results are similar to John's neural observations, they do not entirely support his conclusions. In particular, this is not evidence against the "pattern detector" theory of neuron function, since the nodes are trained exclusively as pattern detectors. The fact that categories can be reliably distinguished on the basis of temporal firing patterns does not necessarily mean this information is used. Temporal encoding may well be used in the nervous system, but in the present model, post-stimulus firing patterns are simply a by-product of information flow, not an optimized representation in their own right.

It is also interesting to note that the learning processes can form functional pathways between indirectly connected nodes in a sparsely connected network. With probabilistic interconnection, increasing the number of planes increases the chances that a pathway exists between any specific input feature and output node. Under such conditions, interconnection complexity and network depth may be traded off against each other.

6.11 Sensory-Motor Behavior

Useful output can be produced in a strictly Boolean domain (e.g., poker playing (Waterman 70)), so the model is capable of potentially interesting behavior. In addition, although the input-output mappings are explicitly defined only for Boolean inputs, non-Boolean values can also be handled. Sensory-motor behavior is a natural area of

application. In particular, a great deal of intelligent behavior can be modeled in terms of continuously valued servomechanisms (Albus 81, Gallistel 80, Miles and Everts 79, Powers 73, Robinson 81).

Simple goal-seeking systems can be built quite easily. If an operator is considered to be off when its output is zero or below, a single valued servomechanism can be built using two operators (Fig. 6.4). (For illustrative purposes, the reference and actual values are between 0 and 1 although the "invert" focusing algorithm expects input values between -1 and 1). The system is defined at the Boolean extremes, but also works appropriately for intermediate values; operator output is proportional to the difference between the reference and actual value. Separate "motor neurons" can be provided for each operator (Fig. 6.4a), or a single one can be shared if both its positive and negative output are used (Fig. 6.4b). Two servomechanisms can be used to control two values, such as x and y coordinates for movement in a plane (Fig. 6.5). Simultaneous movement along both axes will occur when appropriate.

In a servomechanism, the actual and desired values are both input with the same type of signal. Neither signal has any special properties. Albus analyzes the cerebellum in servomechanistic terms (Albus 81). He observes:

"[input] fibers can be categorized into at least two classes based on their point of origin: those carrying information that may include commands from higher levels in the motor system, and those carrying feedback information about the results of motor outputs. Once those two sets of fibers enter the cerebellum, however, they intermingle and become virtually indistinguishable."

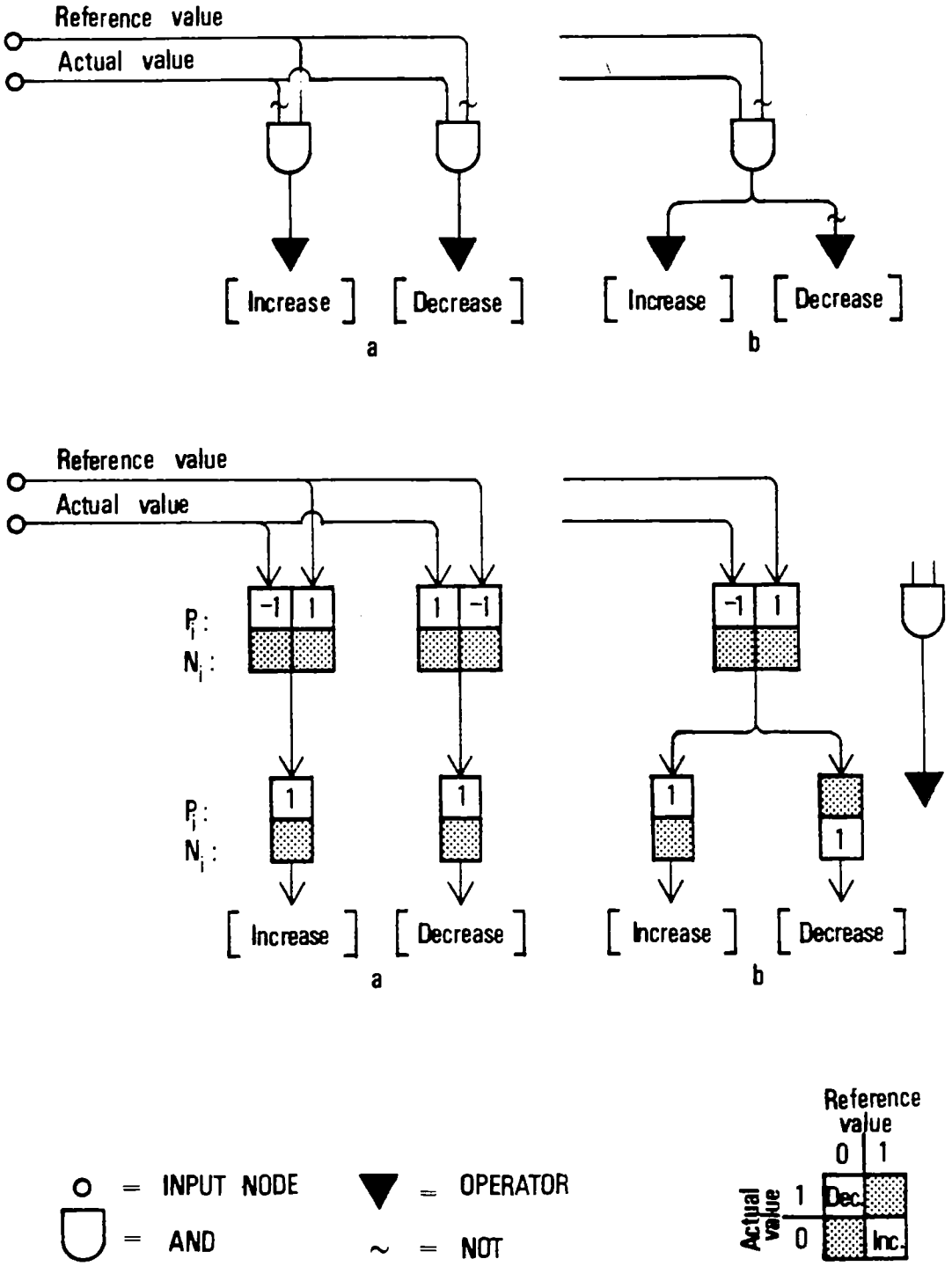
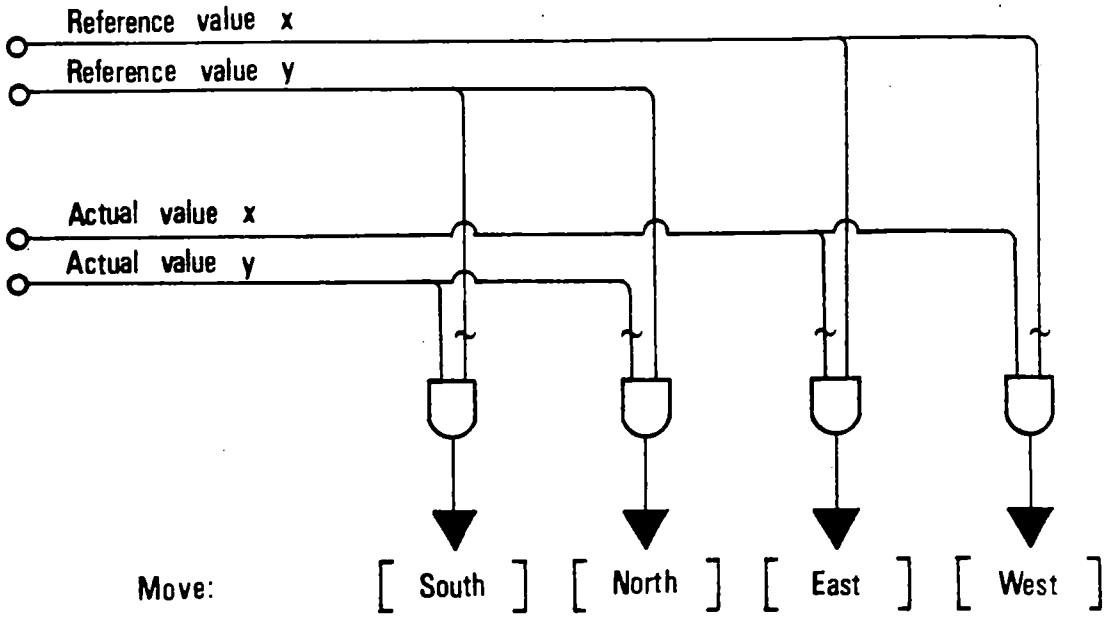


Figure 6.4 A servomechanism consisting of 2 Boolean operators.



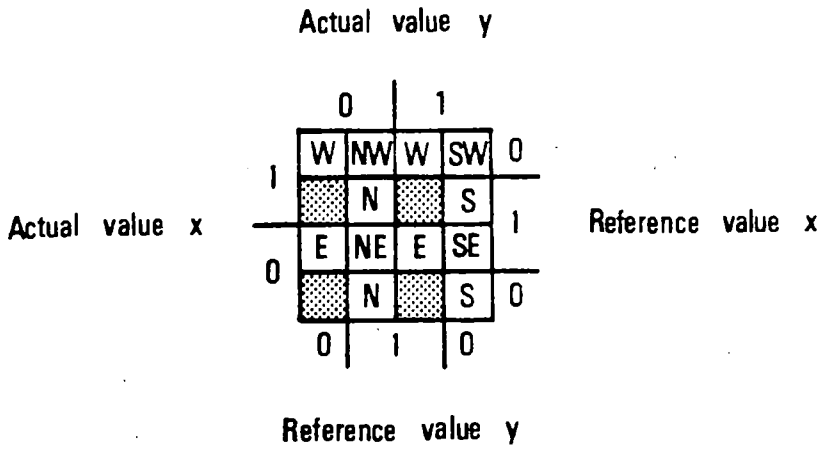


Figure 6.5 A 2-directional servomechanism.

In the cerebellum, specialized (climbing) fibers uniquely address each output (Purkinje) cell. These are commonly thought to adjust the Purkinje cell's output function, though few researchers have reported direct physiological evidence for this (Ito 82ab). Whatever the mechanism, the cerebellum does appear to have important learning capabilities (Thompson 83, Thompson et al. 82, 83).

The previous examples of servomechanisms cannot be turned off. The reference value is always between 0 and 1, and the actual value is continuously adjusted to match it. It is useful to provide an enable/disable input (Fig. 6.6). In this case a disable input is provided to directly inhibit an operator whenever it should not be enabled. Inhibition (disabling, depotentiation) of lower level systems by higher level ones is common in the nervous system (Kandel and Schwartz 81 ch. 24, Kent 81, Gallistel 80).

It is also useful to gate (multiplex) one of a number of reference values into a servomechanism (Fig. 6.7). By making the reference value an "input parameter", the process of goal selection can be conceptually separated from the process of goal achievement. If appropriate lower level systems already exist, all that is needed to modify behavior is to gate in a different sequence of goals. Biologically, selective gating is implemented with either direct inhibition of excitatory input neurons, or as presynaptic inhibition of their output synapses (Kandel and Schwartz 81 pg. 274). The current model does not include presynaptic inhibition, so direct inhibition is used. This example can be represented as a 5 feature

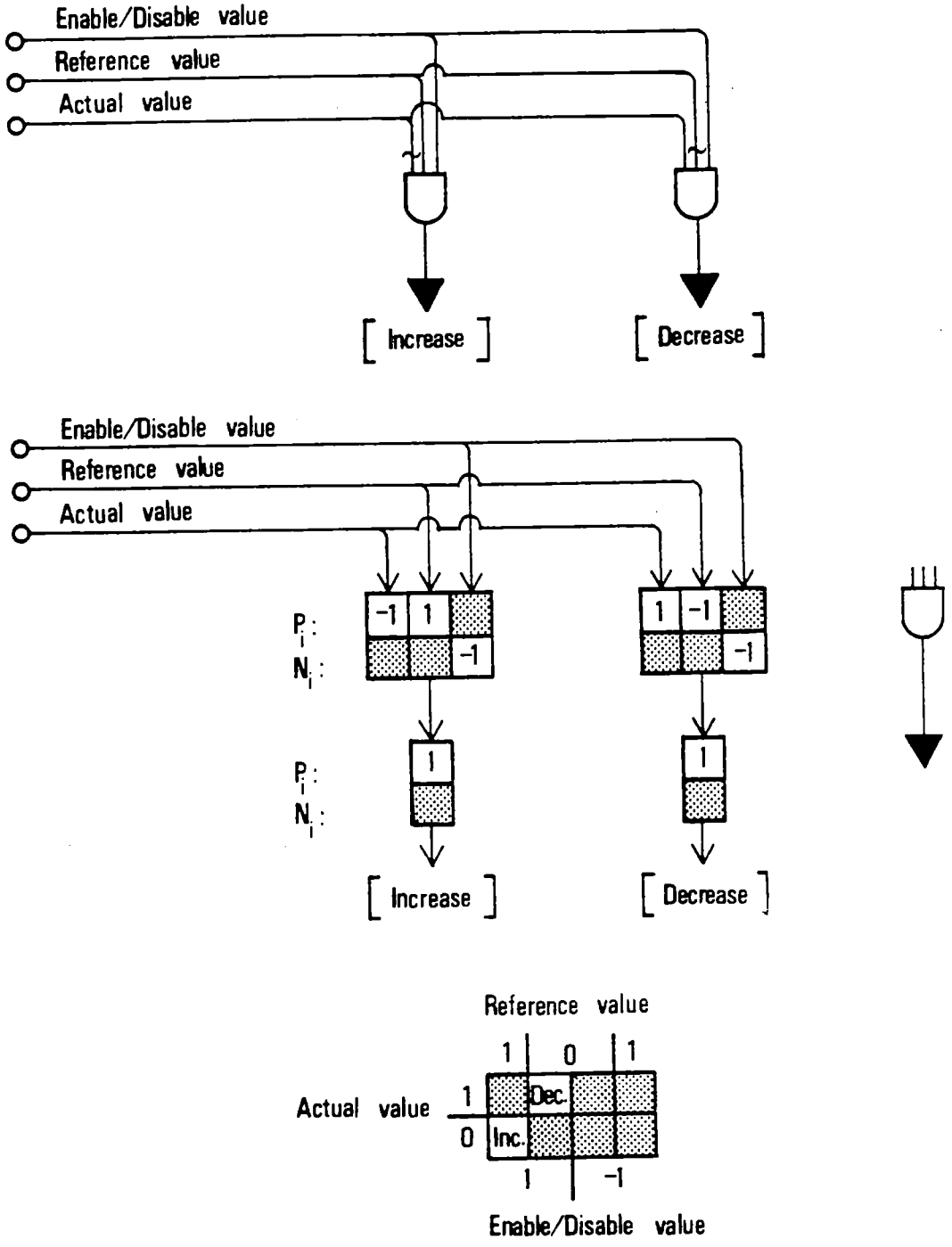


Figure 6.6 Enable/disable input to a servomechanism.

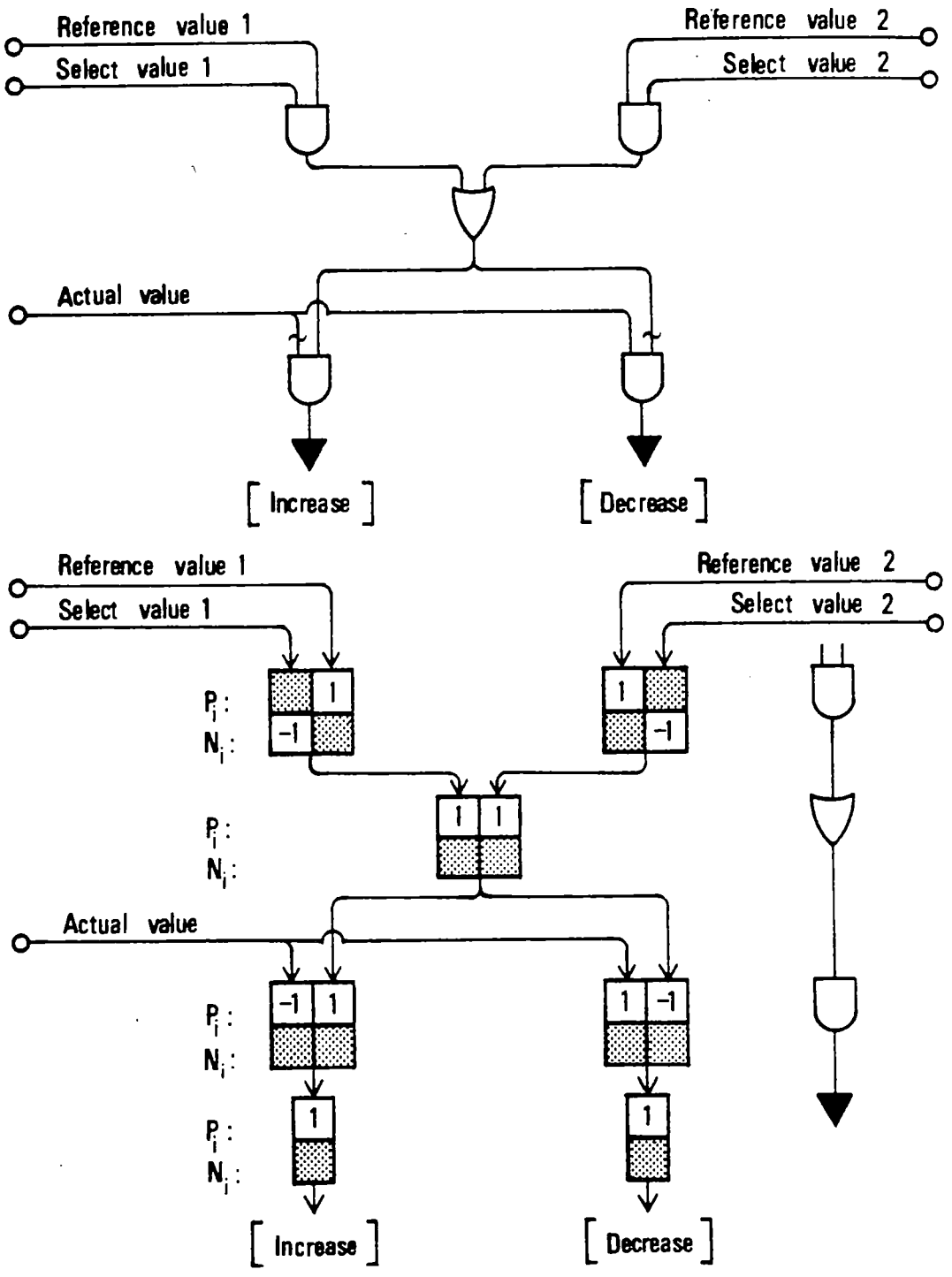


Figure 6.7 Multiplexed input goals to a servomechanism.

Boolean function, but a Karnaugh map representation would be of questionable value.

A more elaborate network combining the previous examples is shown in Figure 6.8. This network is capable of putting a peg in a hole, given:

Boolean values

Want peg in hole
If peg is in hole
If peg is in hand

location values

x and y coordinates of hand
x and y coordinates of peg
x and y coordinates of hole

and hand operators

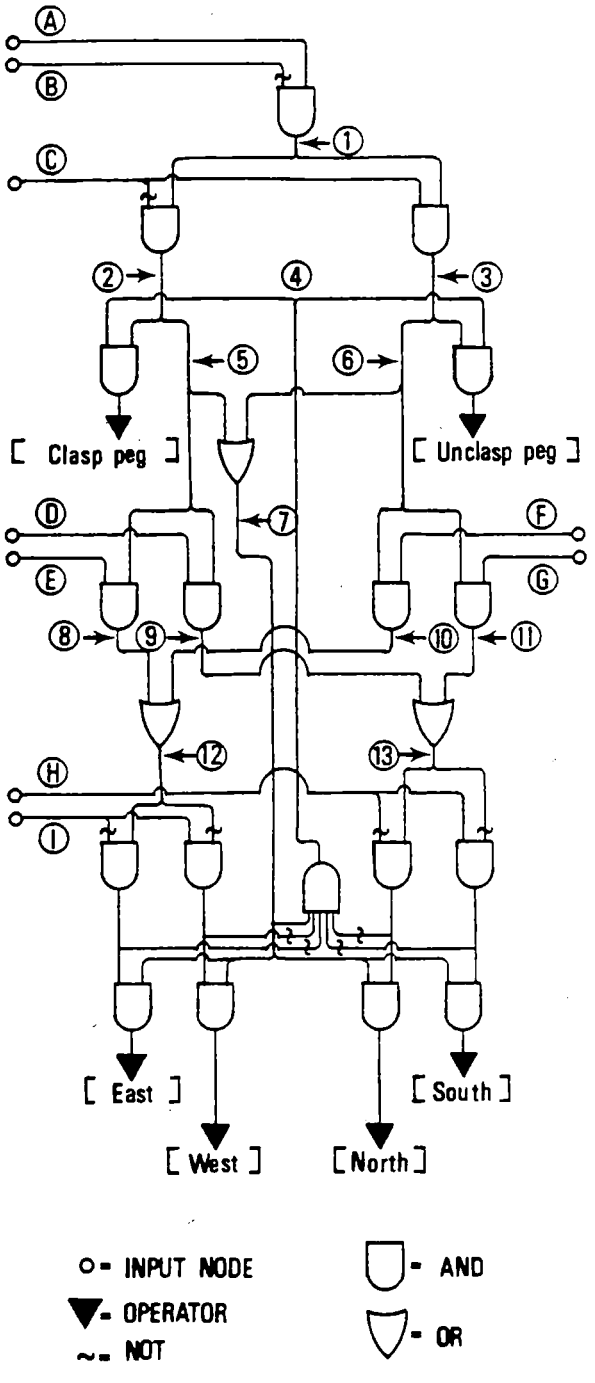
grasp
ungrasp
move north, east, south, west

The appropriate input-output connections can be described as a Boolean function mapping 9 features to 6 operators.

In simulated networks, the resulting weight patterns are usually too complex for easy interpretation, but this hand-designed system demonstrates some interesting features. The highest level goal is successively refined through a 4 level hierarchy to low level, executable goals. For example:

goal: peg in hole =>
goal: hold peg =>
goal: hand at peg =>
goal: move hand east

Because it describes a Boolean function, this behavioral system could



Inputs:

Ⓐ Want peg to be in hole
Ⓑ Is peg in hole?
Ⓒ Is peg in hand?
Ⓓ Peg location, y
Ⓔ Peg location, x
Ⓕ Hole location, x
Ⓖ Hole location, y
Ⓗ Hand location, y
Ⓘ Hand location, x

Internally computed signals:

① Goal: Achieve state of peg in hole
② Goal: Clasp peg
③ Goal: Put peg in hole
④ Is hand at goal?
⑤ Goal: Move hand to peg
⑥ Goal: Move hand to hole
⑦ Enable hand
⑧ Goal: Move hand to peg x location
⑨ Goal: Move hand to peg y location
⑩ Goal: Move hand to hole x location
⑪ Goal: Move hand to hole y location
⑫ Goal: Move hand to location x
⑬ Goal: Move hand to location y

Figure 6.8 Servomechanism hierarchy to put a peg in a hole.

also be expressed as a two-level network, but a hierarchal approach permits greater sharing of information. The complexity of the network is increased somewhat, but the complexity of the individual nodes is decreased.

It is interesting to note that many of the basic "meanings" of neurons are represented. There are input driven feature detectors (peg in hole), output generating motor nodes (grasp), and a bunch of "intentional" nodes (goal: get peg). Depending on your point of view, these internal signals can be interpreted as goals, commands, drives or needs to achieve something, or simply as shared patterns which happen to be predictive of correct output. They can also be viewed as means-ends selected operators since they are triggered by current and desired conditions (Newell and Simon 72), or as a hierarchy of servomechanisms (Albus 81). The network could also be explicitly represented as a production system, since any node can be expressed as an if-then rule.

The top level input "want peg in hole" still seems a bit mentalistic, but if a person were instructed to put the peg in the hole when a green light goes on, that input could be replaced with sensory input "light is green". The system would then be completely mechanistic, though still teleologically interpretable in terms of internal goals and intentions. If the top level input were an internal "thirst" sensor and "put peg in hole" were interpreted as "empty glass in mouth", the system could also be interpreted as modeling a homeostatic activity.

The preceding example suggests that a number of interesting psychological aspects of behavior can be addressed in the context of a Boolean stimulus-response model. One potentially interesting result is that behavioral terms such as trigger features, drives, commands, needs, intentions and goals can find natural expression in a homogeneous neural network whose design criteria can be mechanistically specified in terms of efficient input-output connections. This is in keeping with Gallistel's observation that "the problem of motor control coordination becomes the problem of motivation as one ascends the action hierarchy" (Gallistel 80).

6.12 Results

A 9 feature input space is too large to actually simulate, but a one dimensional version (movement along one axis only) is of reasonable size. The resulting 6 feature, 4 operator function was learned by a one plane network in 18 cycles. Location extremes of -1 and 1 were used since that is the expected range for the learning algorithm. Only 25 nodes were used in the common memory plane, well short of the 64 different input patterns.

In general, development and testing was limited to 4-feature functions. The same set of 40 functions used to test operator training (Fig. 5.8) was also used to test focusing. Overall learning speed was adjusted to be about the same for the two processes, but speed on individual functions was often quite different. As expected,

operator training is better when generalization is possible (e.g., pattern 6) and focusing is generally superior when specific instances are important (e.g., pattern 12).

To investigate the limits of system capabilities, an 8-feature checkerboard was learned (the 8-feature counterpart to pattern 12). This is one of the hardest functions to learn with any number of features. The system was run with 350 nodes in the memory plane (2 ** 8 = 256). The function was successfully learned in 32 cycles - not greatly in excess of the 22 cycles required to learn the 4-feature version. However, it took 25 cpu hrs, so larger input spaces were not investigated. Though the number of cycles needed to learn the function didn't increase greatly, with sequential simulation each additional feature requires about 4 times the previous runtime.

7.0 EVALUATION AND CREDIT ASSIGNMENT

7.1 The Goal

The model, as developed so far, is capable of learning arbitrary S-R mappings, provided it is explicitly taught the correct function. The need for such an omnipotent instructor is a common criticism of neural learning "with a teacher". The purpose of this chapter is to develop a plausible model of neural instruction.

The standard argument that learning is unnecessary if the correct response is already known ignores the fact that instruction may occur only after a response is made. The correct response is not known beforehand, but may be at least partially deduced from succeeding state changes. This is the essence of trial-and-error learning, a common feature of biological behavior. Post-response instruction may completely specify the correct output, "learning with a teacher", or may only evaluate the correctness of the preceding action (sometimes referred to as "learning with a critic" (Barto et al. 81, Widrow et al. 73)). Although a critic may not directly indicate the correct output, it conveys useful information for modifying output, which from a local point of view is explicit teaching. The specific problems addressed in this chapter are the evaluation of behavior and the generation of specific teaching signals based on that evaluation.

7.2 Evaluation

Implementation of a specialized evaluation process is consistent with what is known of biological reinforcement systems. In particular, limbic structures such as the hypothalamus and amygdala seem to be involved in evaluation (Gallistel 73, Gallistel et al. 81, Olds and Forbes 81, Rolls et al. 80, Rolls 81, Pugh 77, Kent 81, Kapp et al. 82, Albus 81). As Albus observes:

"The ability to discriminate good from bad and to evaluate whether a particular activity is rewarding or punishing is critical to the selection and control of behavior. The emotional centers of the limbic system provide this capacity for evaluation. These regions of the brain provide the value judgements as to whether the results reported by the sensory-processing regions of the brain are good or bad. These are the centers that tell us whether what we are doing (or are thinking of doing) is rewarding or punishing." (Albus 81 pg. 95)

This hypothesis is consistent with reports that direct electrical stimulation of the hypothalamus or amygdala can modulate learning speed in other parts of the brain (Woody 82 pg. 159, Woody et al. 83, Kim et al. 83, McGaugh 83).

The functional distinction between data and evaluation type information is sometimes anatomically distinguishable.

"The operant conditioning process may then be the result of classical conditioning of certain cells, such as those of the basal ganglia, by inputs from the reward [evaluation] system immediately after they are successfully fired by input activity from the cortical action scheme generators or perceptual analyzers [data]. ... The distribution of these two input systems to the cells of the basal ganglia is consistent with this idea. The cortical inputs carrying activation patterns to generate specific outputs are restricted to selected cells. The reward system's input, relevant to any just completed action is diffuse and widespread." (Kent 81 pg. 186)

Biologically oriented models often use terms such as reinforcement or pleasure to explain the learning process. However, mechanistic implementation of those evaluative terms must eventually result in the alteration of specific values, requiring an implicit teaching process. For example, if it is pleasurable to do something when you should have (output = 1, teacher = 1) and painful to do something when you shouldn't have (output = 1, teacher = -1), then maximizing pleasure and minimizing pain result in matching output to the implicit teacher value. (Biologically, pleasure and pain do not appear to be as symmetric as this example would suggest). If instruction is in response to the node's output, the term evaluation may be more appropriate, but if instruction is not dependent on the output of the node, teaching seems the better term. The pain detector in gill withdrawal is an example of a teaching signal.

7.3 Credit Assignment

The model must address two basic problems of credit assignment if a single, global evaluation of behavior is used to train multiple outputs. If more than one operator is applied simultaneously, it is not clear which ones are responsible for desirable or undesirable changes (Barto et al. 81). Similarly, if a sequence of operators is applied before a goal is achieved, it is difficult to determine which operators contributed to the final achievement (Minsky 63). It should be noticed, however, that these problems of credit assignment occur only in training the operators, not in training the common memory.

Whenever an error occurs, the shared representation is simply made more specific.

In addition, there is what Feldman (82) has called the "deferred outcome" problem, in which the results of behavior may not be immediately observed. This is one step further back in the learning sequence:

stimulus => behavior => results => evaluation =>
credit assignment => teaching/learning

and so compounds the previous two problems. It was once thought that reinforcement must be delivered almost immediately for effective conditioning. However, the Garcia effect (specific food aversion due to delayed sickness) has shown that behavior-result delays of an hour or more are sometimes acceptable (Garcia and Koelling 66, Garcia et al. 82, Dickinson and Mackintosh 78). This may be a specialized system, but similar effects have now been shown in other situations (D'amato et al. 81). In any event it is a significant limitation in biological learning. That problem will not be addressed in the current model. Any relevant effects of behavior are assumed to be immediately observable.

7.3.1 Simultaneous Application - The first problem of simultaneous operator application can be avoided by training the model as a production system. If only one operator is applied at a time, it is clear which should get credit or blame:

1) If an operator fired and things get worse,
   then it was wrong and should be off.
2) If an operator fired and things get better,

then it was right and should be on
and everybody else should be off.
3) If nobody fired and things get worse,
then somebody should have been on so all move up.
4) If nobody fired and things get better,
then nobody should be on (and they weren't)
and perhaps learn to apply the no-op operator.

If the current evaluation is less than optimal, staying the same can be treated the same as getting worse. This trial-and-error strategy will cycle through all the operators (repeating some) until the correct operator is applied. The system will then stabilize for that input pattern. Providing the amount of mutual interference isn't excessive, the system will eventually stabilize on correct output for all inputs.

This is essentially the same as Thorndike's Law of Effect (Thorndike 13). As summarized by Hilgard and Bower, this law states: "responses to a situation which are followed by a rewarding state of affairs will be strengthened or stamped in as habitual responses to that situation; responses which are unsuccessful will be weakened or stamped out as responses to that situation" (Hilgard and Bower 75 ch. 2). The basic process is the formation of associations between stimulus and response as controlled by evaluation.

7.3.2 Sequential Application - The second problem of sequential credit assignment can be addressed with the introduction of learned secondary evaluation. Primary (innate) evaluation identifies a specific goal state, and secondary (learned) evaluation indicates the likelihood that any other state is on a path to that goal. This

learning rule was implemented as:

Eval := Eval + (Next_eval - Eval) * r

In effect this says the secondary evaluation of a state should predict the evaluations of succeeding states.

For example, in an n state sequence, state n-1 is as good as the goal state, n (Eval = 1), if the correct operator for that transition is known. The learned evaluation of state n-1 then makes the transition from n-2 to n-1 rewarding. Action sequences are learned backwards, producing a gradient of positive evaluation leading to the final goal state. Primary evaluation identifies innate goal states, while secondary evaluation provides immediate feedback for transitions leading toward those states.

Samuel implemented a similar learning process in his checkerboard evaluation function:

> "We are attempting to make the score [evaluation], calculated for the current board position, look like that calculated for the terminal board position of the chain of moves which most probably will occur during actual play. Of course, if one could develop a perfect system of this sort it would be the equivalent of always looking ahead to the end of the game. The nearer this ideal is approached, the better would be the play" (Samuel 63).

By simulating succeeding states, Samuel used this information to choose the next transition. Because the current model doesn't anticipate future states, the correctness of a transition can be determined only after it has actually been made. A similar approach has also been used to learn pole balancing (Barto et al. 82).

One characteristic of this learning scheme is that the evaluation gradient disappears as behavior stabilizes. That is, when behavior is well learned, secondary evaluation predicts succeeding evaluation very closely. A similar effect has been observed in biological learning:

"Once organized, goal gradients and anticipatory goal responses become cumbersome and unnecessary ... there is some evidence that the goal gradient does, in fact, occur primarily during the early stages of learning or exploration ... I might add that organization probably precedes from the "goal" backwards ..." (Mandler 75 pg. 36).

Because the evaluation gradient saturates, it is necessary to temporarily habituate positive evaluation in order to avoid the possibility of looping action sequences.

This explicit separation of evaluation learning from behavior learning is consistent with the observation that aversive conditioning involves two separate processes, the learning of "conditioned fear" (evaluation) and the learned behavioral response to it (Thompson et al. 83, Thompson 83, Rescorla and Solomon 67, Weinberger 82).

Since the evaluation system can use the output of the common memory, single node evaluation is possible. As with single node operators, the logical extreme of completely decoding the input space is trivially adequate since the correct evaluation can be uniquely attached to each input pattern. However, like output error, a large change in evaluation may now indicate that input categorization is insufficiently specific, in this case for accurate evaluation. Consequently, the magnitude of evaluation change was included in the common memory focusing rate. Thus evaluation is also logically complete and reasonably efficient since memory modification is

proportional to evaluation error. As in the case of single node operators, the burden on the common memory could be significantly reduced by providing a small dedicated network for evaluation.

Training an evaluation node is slightly different than other nodes in the system, since it is the only type which is trained with a non-Boolean teaching signal. The weight adjustment algorithm is appropriate for training continuous output, but the teaching value for conditional probability was kept Boolean by predicting whether a node's output should go up or down, rather than whether it should be on or off. If correct output is approached cautiously (no more than a 50% correction on each input) this is quite adequate. It is also possible to generalize the conditional probability trace to predict intermediate output. This permits faster convergence, but is theoretically at risk because it confounds teaching frequency and intensity. For lack of a clear preference, both techniques are implemented as runtime options.

The implementation of specialized evaluation and sensory-motor systems introduces the most general learning problem the model has to address: the formation of an acceptable mapping between initially goalless behavior and initially behaviorless goals. Learned behavior can be more elaborate than genetically hardwired action, but the learning process itself may be quite complex. Initial exploration of this problem does not suggest that there is a simple solution. On the contrary, though the problem can be simply specified, implementation of an efficient, general solution appears to be quite difficult.

7.4 Learned Homeostatic Behavior

Simple homeostatic (servomechanistic) behavior can be learned with primary evaluation alone. For example, if it is rewarding to drink when thirsty or eat when hungry, then behavior will converge on those homeostatic actions. Taking a specific example, it is innately rewarding to taste sugar when hungry. The appropriate primary value for this state can be defined as (hungry AND sweet_taste) => (Eval = 1). Any behavior which results in sugar ingestion will be reinforced if the organism is hungry. For instance the Eat operator might learn to fire in the state (hungry and looks_fruit_like). The internal feature "hungry" is a necessary part of both the reward and behavior system. Neuron firing which is dependent on both hunger and the sight of food has in fact been observed in the hypothalamus (Rolls 81). These neurons were closely associated with the internal reward system. It is interesting to note that the neurons responded to the sight of learned food objects.

In order to learn homeostatic behavior, two internal signals must be provided: reinforcement for correct behavior, and a motivational/drive signal to indicate when the goal is active (i.e., will be rewarded). It is reasonable to assume that many such homeostatic teachers exist in an organism, corresponding to the observed homeostatic behaviors (eating, drinking, etc.). This is consistent with the known involvement of the hypothalamus in homeostatic function. Direct electrical stimulation of the hypothalamus produces results which can be described as both

motivational and rewarding (Deutsch and Howarth 63, Gallistel 73). In that theory, stimulation

> "activated two systems, a drive or motivational system that was the energizing factor responsible for the initiation of behavior, and a satisfying or reinforcing system that was responsible for the establishment of the connection between the response and the brain stimulus" (Olds and Forbes 81 pg. 529).

This two-system theory is not universally accepted, but its functional distinction between motivation and reward would seem to be justified in the context of the proposed model.

As previously observed, a drive signal is treated the same as other data in the system. Any "motivational" properties it has are due simply to its inclusion as a relevant feature in the production of behavior.

It is generally accepted that some homeostatic processes are regulated by both drive and satiety (anti-drive) signals (Gormezano et al. 83). That is, a homeostatic activity (eating for example) is not only activated by positive instances (hungry), but is actively inhibited by negative ones (full). This provides an ability to distinguish the absence of hunger from actually being stuffed, which can be of considerable practical significance. Satiety can be modeled as the inverse of a 2-signal drive-reward system. A complementary satiety-punishment system is logically adequate and biologically parsimonious.

7.5 Models Of Learning And Memory

As demonstrated by the well known amnestic patient H.M., bilateral temporal lobe damage in humans produces a dramatic reduction in long term memory formation (Milner 70). Consequently, models of human learning and memory often speculate on the function of that region of the brain, and the hippocampus and amygdala in particular. It is thought that this region is dynamically involved with cortical centers in the consolidation of memory (Squire 82, Squire et al. 83). Based on the processes implemented in this model it is possible to hypothesize a specific effect: the identification of useful short term (ST) values so long term (LT) values can move toward (consolidate) them. The hippocampus and other medial temporal structures are thus seen as an important link between the evaluation and sensory-motor systems of the brain.

The period of time before an LT value moves completely to a new, correct ST value constitutes a period of vulnerability for newly established memories. Loss of the ST value before "consolidation" is complete would produce a permanent memory deficit. If LT values move incrementally toward ST values, permanent loss would be inversely proportional to memory age. Such effects are observed in humans after electroconvulsive therapy (Squire 82), and in animals with shock or protein synthesis inhibitors (Agranoff 82). If there was no consolidation, ST values would eventually return to their previous LT states. ST and LT memory would both be functional, but there would be no transfer between them. This appears to be the case for H.M.

To effectively control consolidation, a general learning rule can be adopted that LT moves toward ST when ST is useful, otherwise ST moves to LT. Thus a possible scenario for long term memory formation is:

1) short term modification learns a new input pattern.

2) Subsequent evaluation (and usefulness detection in general) determines that some or all of it is worth remembering.

3) The hippocampus selectively identifies (points to) those short term values so they can be consolidated.

Albus has suggested a similar function:

"Evaluations are also useful in the control of memory storage. Some events are very important to remember; others are not. The emotions tell us what is worth remembering. ... The hippocampus is believed to make the emotional judgements as to what is worth remembering. This allows the brain to be selective in what it stores. ... Destruction of this selection center would therefore result in everything being forgotten as if unimportant" (Albus 81 pg.97)

Observations that H.M. is more impaired in the acquisition of "facts" than behavior (Milner 70, Squire 82, Squire et al. 83), suggests that the hippocampus is selectively involved with (something like) the behaviorally uncommitted focusing process, rather than the output oriented operator training process. It has also been reported that if a rat is preexposed to a new environment prior to training, subsequent electroconvulsive shock (ECS) does not disrupt learning (Miller 82). This suggests that ECS is more disruptive of situation learning (focusing) than behavioral adjustment (operator training). Prior environmental exposure permits undisturbed situation learning, and the subsequent behavior adjustment is resistant to ECS.

As proposed in the model, the evaluative system must also learn. Thompson (Thompson et al. 80) has observed learning in the hippocampus and has commented on the central importance of evaluation:

> ... the learning-dependent increase in hippocampal neuron activity that we have described is a general phenomenon whenever training involves the pairing of a signal and a reinforcing stimulus.

More specifically, it has been suggested that the hippocampus and amygdala are involved in the learning of contingencies between stimulus and reinforcement (Gabriel et al. 80, Rolls 82). The temporal parameters of hippocampal learning are in a behaviorally relevant range. Learning (long term potentiation) can be induced in seconds, and may persist briefly or for weeks (Bliss and Lomo 73, Lynch and Baudry 83, Douglas and Goddard 75, Lynch and Schubert 80, Barrionuevo and Brown 83). In addition, individual hippocampal cells may display complex forms of associative conditioning (McNaughton and Barns 77, McNaughton et al. 78, Levy and Steward 79).

The preceding model of short term modification is not intended as a general model of short term memory, since it is only one aspect of a collection of possible short term memory phenomena. In examples of neural "short term memory", it is useful to clearly distinguish between reverberating circuits (Carlson 80 ch. 18), prolonged activity in individual neurons (Roberts and Grant 76), the delayed activation of different neurons (Bindra 76), physically distinct input traces (Sutton and Barto 81), and short term weight modification (gill withdrawal Kandel 77, 79ac). In addition, there are probably more biological levels of consolidation than the two (ST and LT)

implemented in the model.

On the other hand, the implemented LT/ST distinction is not too far from the functional LT/ST distinction used in animal learning.

"short-term is defined as mediating acquisition -- the trial-to-trial improvement in responses within a single session. Long-term memory mediates the improvement in performance maintained from one session to another over intervals of hours to days, or even longer" (Agranoff 82).

As observed in the case of H.M., the model is also compatible with some important aspects of human LT/ST memory.

7.6 Results

The standard set of 40 4-feature functions was used to test sequential application. Using the no-op option, a 2 operator production system was produced. A single input pattern, (1111), was chosen as the primary goal state (Eval = 1), and other states were learned as a sequence leading to it. Thus a 4-feature Boolean function can be treated as a sequence of 15 operator selections. Appropriate operator action received reinforcement resulting from a transition to the next state in the sequence, and inappropriate action resulted in a transition to a neutral, unreinforced state. As before, the system was trained with cyclic presentation of all input patterns. Early behavior is always random since training information is available for only the final transition. The large number of errors (things don't get better) causes the common memory to learn the input space. With the input patterns identified, correct behavior and the

resulting evaluation gradient move back from the final state. Since correct output must be learned sequentially, learning is much slower, but all functions were learned in an average of 51 cycles.

As a more concrete example of the model's ability to assemble action sequences, it can learn to run Karnaugh map mazes. A single state is selected as the goal, and 4 operators are provided to move between adjacent states. As before, behavioral success is determined by the resulting change in evaluation. Rather than a 15 step sequence of 2 operators, 4 operators are used in sequences which need not be longer than 4 (the farthest apart 2 states can be on a 4 feature map). Learning proceeds much the same as in the previous example. Early behavior is random since all states are evaluated as 0 except for the goal state. This unsuccessful early behavior causes the common memory to learn the input space. Correct action and the resulting evaluation gradient then move out from the goal state, except that they can now spread in 4 directions rather than 1. After 18 cycles, behavior is stable with all state transitions leading toward the final goal (Fig. 7.1a).

While demonstrating the model's ability to assemble sequences, the previous example isn't a particularly challenging maze. In order to implement walls, the model was extended to include negative evaluation. Walls were modeled as states with negative primary evaluation (Eval = -1). Learning is the same as before, except that the evaluation of a state is the sum of several, independent evaluations. A positive gradient spreads out from the goal state, and

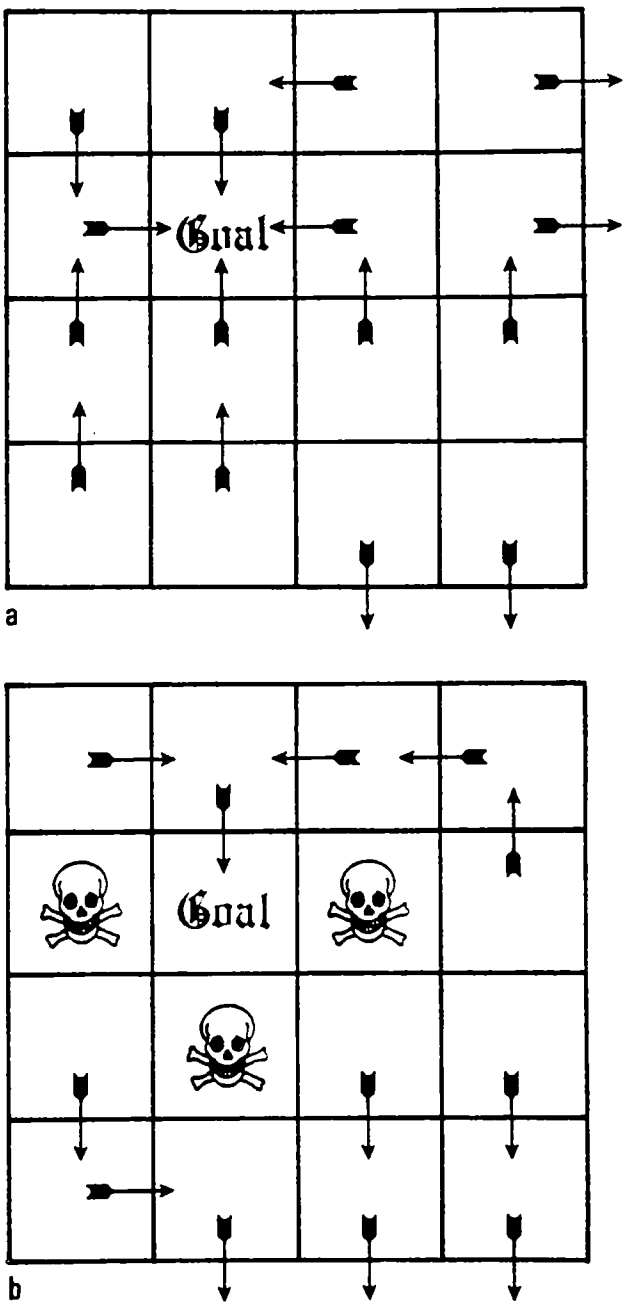


Figure 7.1 Learned goal-seeking behavior on a Karnaugh map maze.
a) no bad states b) 3 bad states

negative evaluation spreads out from the "bad" states. The learned evaluation of any state is simply the sum of these overlapping gradients. The number, type and location of the primary values are entered at runtime. An example with 3 negative states and one positive state is shown in Figure 7.1b. As can be seen in the figure, behavior is appropriate since all paths lead to the goal, and none pass through bad states. This example was learned in 32 cycles.

8.0 SUMMARY AND DISCUSSION

8.1 The Goal

An attempt has been made to define and implement a minimal neural system capable of adaptive behavior in a completely defined environment. Behavior is modeled as the application of specific operators in response to patterns of internal and external features. Behavioral completeness requires that any stimulus (feature pattern) potentially be able to trigger any response (set of operators). Learning completeness requires that any stimulus-response (SR) mapping be learnable. Goal based behavior evaluation determines what the appropriate SR mapping is. Thus the necessary functioning of the model system can be precisely defined. The resulting problem is simple enough to be formally approached, but general enough to address a number of interesting issues.

8.2 SR Behavior

The proposed model is behaviorally neutral in the sense that any response can be made to any stimulus in order to achieve any goal. However, it is clear that all stimulus-response-reinforcer combinations are not biologically equivalent (Lolordo 79, Manning 76). A completely general SR model may not be biologically realistic, but there seems little value in immediately modeling species-specific idiosyncrasies. A general model can be specialized for constrained circumstances, while the reverse process may be considerably more

difficult. Of course the underlying assumption is that there are general processes and that intelligent behavior is not just a collection of special purpose tricks.

Though the model may be too general in some ways, it is very limited in others. One conspicuous limitation is the almost complete exclusion of temporal information. For example, information about the recent past and expected future could be maintained or computed to supplement input from the current state. In a noisy environment, this would allow an organism's world model to "flywheel" through higher levels of input noise (Albus 81). Including such internal features may be stretching the SR model a bit, but it is clearly a useful capability. Certainly the ability to project expected future states is necessary for the development of any planning capacity. It has been suggested that the necessary action-result associations are an important aspect of biological behavior (e.g., Adams and Dickinson 81).

Autonomous activity is another limitation of strict SR models. Some aspects of autonomous behavior can be adequately modeled as chained sequences of internal stimulus-response actions, but it is also true that even individual neurons are capable of autonomous behavior. Autonomous central programs or pattern generators appear to be important in the behavior of all organisms (Gallistel 80, Shepherd 83 ch. 20, 21, Bentley and Konishi 78). As Gallistel observed:

"It is now recognized by neurobiologists, if not yet by psychologists, that endogenously active oscillators constitute a second kind of elementary functional unit in behavior" (Gallistel 80).

To suggest internal SR loops within individual neurons is clearly stretching the SR model beyond its usual application.

There is another important problem with SR models. While SR associations are commonly thought of as connecting input features with output actions, it can easily be shown that learned associations are in fact between mental concepts and behavioral goals (McGaugh 81). For example, if a learned stimulus is the visual presentation of a cube, it may be recognized with varying orientation, lighting, size etc. Certainly the raw visual features have very little in common; it is the invariant mental representation of a cube that is the salient feature. A similar argument can be made on the output side. A rat trained to run a maze will swim through it if necessary. The motor output has nothing in common other than the same goal.

The preceding objection is not to SR theory in general, but only to peripheral SR explanations. SR associations exist, but it must be accepted that they are between mental concepts and behavior goals. Specifying exactly what those are and how they are acquired complicates the SR model considerably. In simplified situations the distinction may not be important, but for real-world learning the issue must be addressed.

By applying only one operator at a time, the problem of credit assignment for simultaneous actions was avoided. However, simultaneous application permits more sophisticated behavior, so the problem must eventually be considered. The "reinforcement learning"

work of Barto, Sutton and Klopf does address this problem.

Despite these problems, a considerable amount of interesting behavior can be produced within the SR domain. Useful SR behavior is demonstrated by AI production systems, and even simple Boolean production systems are capable of producing interesting behavior. By implementing the capabilities of a production system using a neuron-like element, the gap between neural level processes and useful, macroscopic behavior can be bridged.

8.3 Summary

The model was developed in five steps:

1) analysis of gill withdrawal
2) structure and training of a model neuron
3) structure and training of a single operator
4) structure and training of multiple operators
5) behavior evaluation and credit assignment

8.3.1 Gill Withdrawal - The behavior and mechanism of Aplysia gill withdrawal (Kandel 79ab) was analyzed as a neural model of adaptive operator application. An operator was formalized as a trainable category detector. Gill withdrawal can be viewed as a trainable operator which is "taught" its correct output by a pain detector. A combination of associative sensitization and habituation is sufficient to produce the observed behavior.

8.3.2 Structure Of A Model Neuron - In a system requiring Boolean completeness, the necessary properties of a node's output are well defined. Either a node must be able to individually compute any Boolean function, or it must have sufficient power so that an assembly of nodes can. The complexity required for complete "decoding" of an input space grows exponentially with the number of inputs, providing an upper bound on the functional complexity necessary for Boolean completeness. Minimum complexity is simply proportional to the number of inputs.

A linear function was chosen since it is sufficiently powerful for Boolean completeness of assemblies, is relatively simple to implement, and is probably within neural capabilities. A similar function was originally proposed as the McCulloch-Pitts "formal neuron" (McCulloch and Pitts 43) and is the most common functional form used to model neural computation. Many aspects of biological learning can be described with a simple linear function (Sutton and Barto 81) and its limitations may also parallel biological limitations (Bourne 70, Hunt et al. 66, Neisser and Weene 62)

It was observed that a linear function can implement the (at least X of N features) function, formalizing the "ALMOST" gate suggested by Kent (81). Significantly, this function can be treated as a prototype description which includes OR (at least 1 of N) and AND (at least N of N) as its extremes. Prototypes appear to play an important role in both the process of biological learning and the actual structure of natural categories (Mervis and Rosch 81).

Rather than single threshold, binary pattern classification, a continuous, three-valued logic was implemented. Output above and below the resting "unknown" output value of 0 represents increasing certainty in the presence or absence of the category detected by the node. This appears to be biologically common (Sejnowski 81). Output beyond the limits of 1 and -1 is interpreted as absolute certainty. With output thresholds of -1 and 1 and synaptic weights between -2 and 2, the range of computable prototypes is:

```
at most  X1 features give output <= -1
at least X2 features give output >=  1
where X2 - X1 >= 1
```

8.3.3 Training A Node - The ability to convergently train a linear function as a binary pattern classifier is well known as the perceptron convergence theorem (Nilsson 65). This algorithmic process is similar to formal theories of classical conditioning (Rescorla and Wagner 72), and is consistent with biological learning in the gill withdrawal reflex of Aplysia. In particular, a node is told both when it should be on and when it should be off, and its input weights are adjusted accordingly. This standard process was modified in several ways. Most importantly, the adjustment of weights was based on a Bayesian selection of appropriate features to strengthen. This was effective in excluding irrelevant activity from weight modification. A similar two-stage conditioning process has been suggested as an extension to the Rescorla-Wagner model (Mackintosh and Reese 79).

Unlearning was introduced and used to avoid weight saturation. Reversible learning appears to exist in the gill withdrawal system, but may not in hippocampal learning (Barnes 79, Baudry et al. 81). As a consequence, the learning capacity of that system might eventually saturate if the organism were to live long enough.

Since rapid adaptation results in greater noise sensitivity, a dual trace memory was implemented to minimize the effects of learning noise. Several of its characteristics parallel human memory properties.

8.3.4 A Single Operator - Using nodes that can be trained to compute OR and AND, it is possible to compute any Boolean function. Based on disjunctive normal form, a minimal two-level structure is adequate. This is similar to the structure of a perceptron (Rosenblatt 59, 62). Unlike the standard perceptron, both levels are trainable. The problems of restricted interconnection (Minsky and Papert 72) were not addressed.

A learning algorithm was developed which encodes "correct" input patterns as potentially overlapping categories in the lower plane. Though perfect generalization is not guaranteed, the algorithm tends to produce the largest categories possible, thus requiring the minimum number of nodes. The "holographic" advantages of representation by large, overlapping categories rather than highly specific ones were discussed. Several network structures and interconnection schemes

were explored, though for simulation efficiency the minimal network was the most extensively investigated.

The introduction of recurrent connections produces the possibility of within-stimulus learning. Though not necessary for completeness, this appears to be an important aspect of biological learning (Rescorla and Durlach 81). The possibility of self-sustaining activity or oscillation is also produced. The formation of inappropriate positive feedback was observed and inadequately dealt with.

8.3.5 Multiple Operators - Multiple operators were combined into a single behavioral system. Completeness in this system requires mapping arbitrary inputs to arbitrary sets of outputs. While the desired behavior could be produced by training a set of totally independent operators, it seems unavoidable that shared memory is required for systems of any size.

The operator training algorithm was inappropriate for shared memory, so another process was developed, loosely based on hippocampal plasticity (Dunwiddie and Lynch 78). Unlike the operator training process which learned on both positive and negative instances, the common memory learns only on positive instances, when nodes are "focused" on the current input. Input driven learning complements the focusing process by defocusing the existing concepts sufficiently to categorize all input patterns. This adjustment of the common memory

is compatible with both input (data) and output (goal) driven neural learning phenomena (Spinelli et al. 72, Spinelli and Jensen 79, 82).

Tapered focusing was introduced as a mechanism to facilitate the sharing of information in a hierarchically layered system. By reducing the speed of learning in the lower levels, only the most abstract concepts are formed there over an extended period of time. This appears to be an effective approach for the self-organization of hierarchical systems.

With increasing interconnection, an alternative method of information representation was made possible. Rather than representing a category as a static state of neural activation, categories can be identified as temporal firing patterns. Though only the final state represents a node's decision on categorization, a post-stimulus trace of its activity can reliably distinguish many input patterns. This is similar to some biological observations (John 76, 80, John and Schwartz 78). The current model doesn't decode temporal patterns, so although such information is available, it isn't utilized.

It was observed that the operator training process is good at learning generalizations, but inappropriate for learning specific instances. Focusing on the other hand, is capable of learning specific patterns in "one shot", but poor at generalization. This is consistent with a biological dichotomy observed between behavioral adjustment and the acquisition of specific, behaviorally uncommitted

information (Squire 82, Squire et al. 83, Kent 81).

8.3.6 Evaluation And Credit Assignment - A specialized evaluation system was developed and used to train the behavioral system implemented in the preceding chapters. Two problems of credit assignment were addressed. The problem of training simultaneous operators with a single evaluation signal (Barto et al. 81) was avoided by restricting the system to single operator application. The sequential credit assignment problem (Minsky 63) was addressed with the introduction of secondary evaluation. Primary evaluation identifies innate state values, while secondary (learned) evaluation provides immediate feedback for any state change. This approach was utilized by Samuel to learn checkers (Samuel 63), and appears capable of organizing arbitrarily long sequences of actions to achieve a single goal. It is generally compatible with what is known of biological "reward" systems (Gallistel 73, Pugh 77).

8.4 Discussion

The complete model is shown in Figure 8.1. Behavior is the result of information flow between input and output. Learning occurs in three places. Operators are trained to be on and off, the common memory is trained to categorize the input space, and evaluation learns to predict future evaluation. Change detection compares succeeding evaluations to see if things get better or worse. Appropriate

operator application is deduced from this change. Learning in the common memory is triggered by operator error or by large changes in evaluation. Candidate behavior is produced by trial and error and selected by evaluation.

This system demonstrates a biologically plausible model of adaptive behavior. It is constructed with linear function elements which are probably within neural capabilities. The various forms of learning are proposed in order to organize appropriate action. They are not implemented to model specific biological phenomena, but are required by the basic constraints on adaptive behavior. The fact that similar processes have been observed in biological behavior and have been tentatively identified in neural systems suggests that this is a useful conceptual division of function.

By modeling specific functions of neural systems rather than detailed physiology, it should be easier to investigate the relationship between neural processes and intelligent behavior. In addition, a more functional approach avoids the problems created by nature's tendency to implement the same function in a variety of ways. For example, the presynaptic process proposed for Aplysia learning is not a unique mechanism for associative conditioning. It is reassuring to have specific neural mechanisms, but there is no reason to assume that they are the same in all organisms.

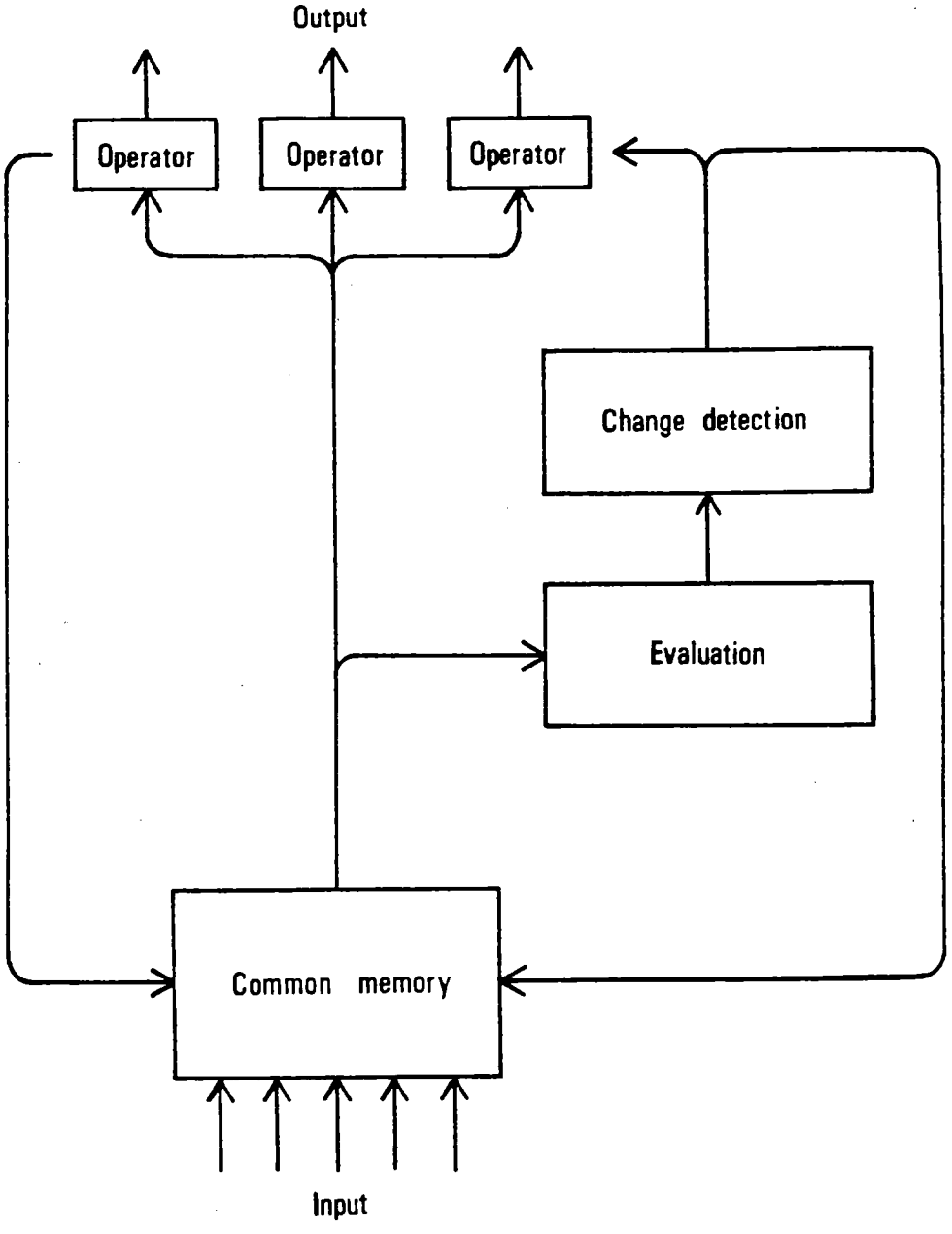


Figure 8.1 The complete model.

8.4.1 Completeness And Efficiency - The combination of individually trainable operators plus a trainable common memory is viewed as a minimal, adaptive sensory-motor system. Because of its ability to represent disjunctive normal form, (and linear or prototypic normal form in general), the system is computationally complete in the Boolean domain. The structure can be elaborated with multiple planes and stacks, and various interconnection schemes, but the minimal system (single node operators, one common memory plane) is logically sufficient.

Learning completeness is equally desirable, but is not as easily demonstrated. Because it implements a linear function, the model node could presumably be shown to learn pattern classification by use of the perceptron convergence proof (Nilsson 65). Unfortunately, considering the current complexity of the program, a formal proof of assembly behavior would be difficult. With common memory focusing there appears to be some hope for formally demonstrating learning completeness. If so, operator training can be modified to include sufficient focusing to guarantee its convergence as well. Because of simulation expense, the system was developed on an input space of 4 features, and empirically it appears to be complete for that size feature space. The programming details have been adjusted for a small number of features, but the conceptual approach appears to be extendable.

Efficiency in time and space are important, though often contradictory constraints on network characteristics. For example, rapid focusing produces rapid (efficient) learning, but is very inefficient in resource utilization because it requires many more nodes. Efficiency is a matter of trade-offs determined by the particular situation. A number of modifications were introduced as ways of improving efficiency, but the issue was only addressed in a very general fashion.

In terms of node utilization, representation in prototypic form is significantly more efficient than disjunctive or conjunctive normal representation. For real world categories, prototypic representation may be especially advantageous. Because of its strong tendency to generalize, the operator training process produces representations that are quite efficient in the use of nodes.

With sequential simulation, the model is, or course, very time inefficient. However, a node's output can theoretically be calculated in one parallel step, and the output of multiple nodes can be computed in parallel. The identification of Max_out can also be done in one parallel step. Consequently, an actual parallel implementation would have a very short reaction time. With one-pass output calculation (no recurrent connections), reaction time is determined by the depth of the network. A deeper network permits an advantageous sharing of information but sacrifices reaction speed. Biological systems are presumably structured in order to optimize behavior within their particular time and space constraints. Human reaction time suggests

an upper bound of about 100 cell delays between input and output.

8.4.2 AI Concerns - In the introduction, a general characterization of intelligence was given as "goal-directed behavior which takes into consideration current conditions and past experience". Mechanistic implementation of this in the context of a production system requires some activity in a number of standard fields in AI such as knowledge representation, pattern matching, learning and problem solving. However, the use of neuron-like elements results in a number of interesting differences from more standard AI approaches to these problems.

Knowledge representation is an interesting area in which to compare neural and more standard AI approaches. While a linear function of input features is natural for neural concept representation, its power to describe prototypes, and the ability of prototypes to describe real world categories, are seldom used in AI systems. Categorization in AI is typically based on identifying a minimal set of necessary and/or sufficient features, rather than a probabilistic distribution of all features. Since necessary (AND) and sufficient (OR) features can be identified as the ends of the prototypic continuum, it is not surprising that features of intermediate value are more the rule than the exception. With sequential computation, it is more economical to utilize such key features when they exist, but it should be recognized that such an approach is a highly restrictive technique for dealing with natural

categories.

Continuous values are also natural in neural models, but seldom found in AI knowledge representation (e.g., Michalski et al. 83). For real world situations, continuous representation may be quite useful.

Several of the more philosophical issues of knowledge representation can have precise expression in neural networks. In a hierarchically connected system, the "primitives" of representation are the raw input sensors. The "extension" of a concept is the set of input patterns a node responds to, and its "intention" is defined by its connection weights to higher level nodes.

If recurrent connections are permitted, the network is still well defined, but its semantic interpretation is complicated somewhat. For example, feedback can lead to a logical paradox if a NOT node is connected with itself. (A => ~A). This can be an annoying problem in logic, often causing "all hell to break loose" (McCawley 81), but (perhaps unfortunately) poses no problem at all in network implementation. It is a simple oscillator. In general, recurrent connections complicate the logical definition of a concept since it may be partly cyclic. Since positive and negative feedback are both useful information processing techniques, this points out a limitation of standard logic for modeling some neural processes.

In addition, "unknown" is not effectively handled in standard logic. The logical values in the network are better described as a continuous three-valued logic than the discrete two-valued logic commonly used. Various forms of multi-valued and fuzzy logic have been proposed (McCawley 81 ch. 12), but none is in common use. It appears that some form of extended logic is necessary to adequately describe neural computation.

Another important difference between neural intelligence and standard AI is in the appropriateness of symbol manipulation as the basic model of information processing. Much of AI is in close agreement with the symbol manipulation paradigm. This has probably resulted from general adoption of the Von Neuman computer as a metaphor for abstract information processing. Computer architecture is in close agreement with the symbol manipulation model, but neural hardware is not.

In particular, the distinction between an active processor and passive data is inappropriate for neural computation. Memory is not a collection of passive facts to be searched by a central executive, but comprises the entire active network of neurons between input and output. Biological memory is inherently associative and content addressable, and potentially reconstructive, producing a fundamental distinction between biological and computer-like memory.

Likewise, the concept of a central processor is inappropriate for the description of neural assemblies. There is no "little man in the network" corresponding to the CPU of a computer. Computation is a distributed, parallel process which Feldman calls "computing with connections" (Feldman 81). The characteristics of such distributed computation can be quite different than computation based on a single central processor. For example, with parallel processing, techniques which minimize reaction time rather than total computation can be utilized.

The use of a single processor produces a sequential bottleneck in computation. This computational bottleneck is a significantly limiting factor to the complexity of problems which can be effectively addressed. Consequently, AI problems and approaches have been strongly selected for those which can be efficiently dealt with using sequential computation. Appropriately parallel domains such as sensory-motor behavior have been effectively defined out of the field of AI. Symbol manipulation is useful as a conceptual model of high level processing, but it is possible that over-reliance on this general model has obscured the possibilities of more specialized approaches.

It is hard to contrast learning techniques in neural and artificial intelligence because they are only partially characterized in both cases. However, it seems unavoidable that the "what" of learning will affect the "how", so it is not surprising that AI learning techniques tend toward identifying necessary and sufficient

conditions, while neural learning models employ a more probabilistic approach. Which approach is more appropriate is determined by the domain of application, but as previously discussed, neural models are sometimes more appropriate for representing real world categories.

Another difference is that neural learning tends to be incremental and thus potentially noise resistant. AI learning techniques are often very sensitive to noise. In general, learning speed and noise resistance are inversely related. The proposed model contains a dual (short term and long term) memory which improves noise resistance without as great a sacrifice in learning speed.

Operator training and focusing seem to represent two fundamentally different learning paradigms. Operator training is equivalent to many AI learning situations. A category description is progressively refined by the presentation of positive and negative examples (Hunt 75, Cohen and Feigenbaum 82 ch. D3, Michalski et al. 83). Focusing is driven by positive instances only: a prototype category is made more specific by focusing on the current input. In the common memory, the two learning processes complement each other. Focusing makes large, general categories more specific, and gap filling (the operator training algorithm without negative instances) attempts to broaden (generalize) existing categories to include new instances.

In a system with finite memory, learning and forgetting are directly related. Forgetting can be due to loss of access (e.g., if a concept is too specific to be triggered), but a more interesting case is the modification of old concepts by new learning. When a node is focused, it may leave gaps in the Karnaugh map. Input driven learning fills these gaps by broadening old categories. Nodes may also be refocused on new categories. Because of this, old concepts are progressively modified until they are entirely forgotten. If the rate of learning is large compared to system capacity, forgetting will be rapid and noticeable. Conversely, when capacity is large compared to the learning rate, forgetting may be imperceptible. Machine learning algorithms typically do not explicitly worry about memory requirements.

The learning and memory characteristics of the model are consistent with general biological characteristics: new learning causes forgetting, similar concepts interfere with each other more than dissimilar ones, and loss can be progressive rather than all or nothing. In addition, unlearning of a particular overt behavior is largely a process of over-writing the old connections with new correcting ones. The old connections are not eliminated, the system is simply adjusted to correct for them. Because the model does incorporate a certain amount of explicit unlearning at the node level, it does not have to deal with the biological problems of weight saturation and the appropriate distribution of plasticity over a finite life span.

A useful model which is seldom used in AI is the servomechanism concept. The servomechanism has proved useful in psychology as a mechanistic model of goal seeking, and neural models are quite compatible with a servomechanistic interpretation of behavior. One interesting characteristic of servomechanisms is that desired (goal) states and the actual, current state are treated as the same type of information. Goals and current conditions can be freely intermixed in a servo/neural behavioral system. Some production systems use this capability by explicitly including goals as left-hand side features (e.g., Lenat 77).

8.4.3 Biological Relevance - A number of biological parallels were observed in the development of the model. Since the model was strongly constrained by completeness and efficiency issues, these parallels suggest that some neural characteristics may be similarly shaped. Besides the general issue of Boolean completeness, a number of specific constraints were considered. Some of these were node and network complexity, parallelism, interconnection, information sharing, learning and reaction speed, noise rejection, network capacity and resistance to damage. Different emphasis on any of these issues would produce different behavior in the overall model, but all are significant constraints on adaptive, intelligent behavior, and have presumably shaped neural characteristics.

Perhaps the most interesting possibility is the central importance of focusing as a learning process. At present the biological evidence for this is only suggestive, but model results demonstrate that prototype focusing is at least a theoretically viable process. It would be equally gratifying if neural characteristics similar to the conditional probability trace method II were to be demonstrated. Animals display similar conditional learning, but it has not been determined whether this is a property of individual neurons or of larger systems.

It is always dangerous to infer the purpose of biological processes, but a useful understanding must ultimately be based on function rather than an exhaustively detailed description of mechanisms which does not distinguish between unavoidable limitations and selectively optimized capabilities. Of course, if one looks hard enough, there is usually sufficient biological evidence to selectively support almost any hypothesis, but on the whole, the good agreement between model results and general biological characteristics suggests that many neural mechanisms can be usefully viewed in terms of the desired capabilities and the resulting structures and processes which have been developed in the model.

8.4.4 Future Work - The domain of Boolean input patterns was chosen because of its well defined, mathematically tractable nature. This is a significant restriction since spatial, temporal and relational inputs are not explicitly modeled, but it may be possible to extend

the Boolean formalism to include those domains. For instance, the temporal firing pattern of a single input can be represented and detected in the same manner that simultaneous firing of multiple inputs is detected. Relational patterns can be expressed as temporal sequences, suggesting that the temporal domain may be a profitable area of future development. Spatial or topographic effects are achieved in many models simply by limiting connections and interaction to neighboring nodes. These possible extensions suggest that progress in the Boolean domain may provide a useful foundation for investigation into other domains.

9.0 BIBLIOGRAPHY

Adams, C., Dickinson, A.: Actions and habits: Variations in associative representations during instrumental learning. In: Information processing in animals. Spear, N. E., Miller, R. R. (eds.). Hillsdale, NJ: Lawrence Erlbaum Associates (1981)

Adams, J. B.: A probability model of medical reasoning and the Mycin model. Math. Biosci. 32, 177-186 (1976)

Albus, J. S.: Mechanisms of planning and problem solving in the brain. Math. Biosci. 45, 247-293 (1979)

Albus, J. S.: Brains, behavior and robotics. Peterborough, NH: BYTE/McGraw-Hill (1981)

Alger, B. E., Teyler, T. J.: Long-term and short-term plasticity in CA1, CA3 and dentate regions of the rat hippocampal slice. Brain Res. 110, 463-480 (1976)

Amari, S.: Characteristics of random nets of analog neuron-like elements. IEEE Trans. Sys. Man Cyber. smc-2 643-657 (1972)

Amari, S.: Neural theory of association and concept formation. Biol. Cyber. 26, 175-185 (1977a)

Amari, S.: Dynamics of pattern formation in lateral-inhibition type neural fields. Biol. Cyber. 27, 77-78 (1977b)

Amari, S.: Topographic organization of nerve fields. Bull. Math. Biol. 42, 339-364 (1980)

Amari, S., Arbib, M. A.: Competition and cooperation in neural nets. In: Systems Neuroscience. Metzler, J. (ed.). New York, NY: Academic Press (1977)

Amari, S., Arbib, M. A. (eds.): Competition and cooperation in neural nets. New York, NY: Springer-Verlag (1982)

Amari, S., Takeuchi, A.: Mathematical theory on formation of category detecting nerve cells. Biol. Cyber. 29, 127-136 (1978)

an der Heiden, U.: Analysis of neural networks, lecture notes in biomathematics 35.
New York, NY: Springer-Verlag (1980)

Anderson, J. A., Hinton, G. E.: Models of information processing in the brain.
In: Parallel models of associative memory.
Hinton, G. E., Anderson, J. A. (eds.).
Hillsdale, NJ: Lawrence Erlbaum Associates (1981)

Anderson, P., Sunberg S. H., Sveen, O., Swann, J. W., Wigstrom, H.: Possible mechanisms for long-lasting potentiation of synaptic transmission in hippocampal slices from guinea-pigs.
Jour. Physiol. 302, 463-482 (1980)

Ashby, W. R.: Design for a brain.
London: Chapman and Hall (1960)

Barlow, H. B.: Single units and sensation: A neuron doctrin for perceptual psychology. Perception 1, 371-394 (1972)

Barnes, C. A.: Memory deficits associated with senescence: A neurophysiological and behavioral study in the rat.
Jour. of Comp. and Physiol. Psych. 93, 74-104 (1979)

Barrionuevo, G., Brown, T. H.: Associative long-term potentiation in hippocampal slices.
Proc. Natl. Acad. Sci. 80, (in press) (1983)

Barto, A. G., Sutton, R. S., Anderson C. W.: Neuron-like adaptive elements that can solve difficult learning control problems. Comp. and Info. Sci. Dept.
U. of Mass. Amherst, Mass. tech report 82-20 (1982)

Barto, A. G., Sutton, R. S., Brouwer, P. S.: Associative search network: a reinforcement learning associative memory. Biol. Cyber. 40, 201-211 (1981)

Baudry, M., Arst, D. S., Lynch, G.: Increased glutamate receptor binding in aged rats.
Brain Research 223, 195-198 (1981)

Bently, D., Konishi, M.: Neural control of behavior.
Ann. Rev. Neurosci. 1, 35-99 (1978)

Bindra, D.: A theory of intelligent behavior.
New York, NY: John Wiley and Sons (1976)

Bliss, T. V. P., Lomo, T.: Long-lasting potentiation of synaptic transmissions in the dentate area of the anaesthetized rabbit following stimulation of the perforant path.
Jour. Physiol. London 232, 331-356 (1973)

Bourne, L. E.: Knowing and using concepts.
Psych. Rev. 77, 546-556 (1970)

Brons, J. F., Woody, C. D.: Long-term changes in excitability of cortical neurons after Pavlovian conditioning and extinction.
Jour. Neurophys. 44, 605-615 (1980)

Brons, J. F., Woody, C. D., Allon, N.: Changes in the excitability of weak intensity electrical stimulation of units of the pericruciate cortex in cats.
Jour. Neurophys. 47, 377-388 (1982)

Bures, J., Buresova, O.: Plasticity in single neurons and neural populations. In: Short-term changes in neural activity and behavior. Horn, G., Hinde, R. A. (eds.).
Cambridge, MA: Univ. Press (1970)

Carew, T. J., Hawkins, R. D., Kandel, E. R.: Differential classical conditioning of a defensive withdrawal reflex in Aplysia californica. Science 219, 397-400 (1983)

Carey, S.: The child as word learner.
In: Linguistic theory and psychological reality.
Halle, M., Bresnan, J., Miller, G. (eds.).
Cambridge, MA: MIT Press (1978)

Carlson, N. R.: Physiology of behavior.
Boston, MA: Allyn and Bacon, Inc. (1980)

Cohen, P. R., Feigenbaum E. A.: The handbook of artificial intelligence vol. 3.
Stanford, CA: Heuristech Press (1982)

Collins, A. M., Loftus, E. F.: A spreading-activation theory of semantic processing.
Psych. Rev. 82, 407-429 (1975)

Collins, A., Quillian, M. R.: Experiments on semantic memory and language comprehension. In: Cognition in learning and memory. Gregg, L. W. (ed.). New York, NY: Wiley (1972)

Cotman, C. W., McGaugh, J. L.: Behavioral neuroscience. New York, NY: Academic Press (1980)

Cowan, W. M.: The development of the brain. Sci. Amer. 241, 112-133 (1979)

D'Amato, M. R., Safarjan, W. R., Salmon, D.: Long-delay conditioning and instrumental learning: Some new findings. In: Information processing in animals. Spear, N. E., Miller, R. R. (eds.). Hillsdale, NJ: Lawrence Erlbaum Associates (1981)

Deutsch, J. A., Howarth, C. I.: Some tests of a theory of intercranial self-stimulation. Psych. Rev. 70, 444-460 (1963)

Dickinson, A., Mackintosh, N. J.: Classical conditioning in animals. Ann. Rev. Psych. 29: 587-612 (1978)

Dietmeyer, D. L.: Logic design of digital systems. Boston, MA: Allyn and Bacon, Inc. (1978)

Douglas, R. M., Goddard, G. V.: Long-term potentiation of perforant path granule cell synapse in the rat hippocampus. Brain Res. 86, 205-215 (1975)

Duda, R. D., Hart, P. E.: Pattern classification and scene analysis. New York, NY: Wiley (1973)

Dunwiddie, T., Lynch, G.: Long-term potentiation and depression of synaptic responses in the rat hippocampus: Localization and frequency dependency. Jour. Physiol. 276, 353-367 (1978)

Erickson, R. P.: Parallel "population" neural coding in feature extraction. In: The neurosciences. Schmitt, F. O., Worden, F. G. (eds.). Cambridge, MA: MIT Press (1974)

Fahlman, S. E.: Net1: A system for representing and using real-world knowledge. Cambridge, Mass: MIT Press (1979)

Feldman, J. A.: A connectionist model of visual memory.
In: Parallel models of associative memory.
Hinton, G. E., Anderson, J. A. (eds.).
Hillsdale, NJ: Lawrence Erlbaum Associates (1981)

Feldman, J. A.: Dynamic connections in neural networks.
Biol. Cyber. 46, 27-39 (1982)

Fukushima, K.: Cognitron: A self-organizing multilayered neural network. Biol. Cyber. 20, 121-136 (1975)

Fukushima, K.: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position.
Biol. Cyber. 36, 193-202 (1980)

Fukushima, K., Miyake, S.: A self-organizing neural network with a function of associative memory: Feedback-type cognitron. Biol. Cyber. 28, 201-208 (1978)

Fukushima, K., Miyake, S.: Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In: Competition and cooperation in neural nets. Amari, S., Arbib, M. A. (eds.).
New York, NY: Springer-Verlag (1982)

Gabriel, M., Foster, K., Orona, E., Saltwick, S. E., Stanton, M.: Neuronal activity of cingulate cortex, anteroventral thalamus, and hippocampal formation in discriminative conditioning: Encoding and extraction of the significance of conditional stimuli.
In: Progress in psychobio. and physiol. psych. vol. 9.
Sprague, J. M., Epstein, A. N. (eds.).
New York, NY: Academic Press (1980)

Gallistel, C. R.: Self-stimulation: The neurophysiology of reward and motivation. In: The physiological basis of memory. Deutsch, J. A. (ed.).
New York, NY: Academic Press (1973)

Gallistel, C. R.: The organization of action.
Hillsdale, NJ: Lawrence Erlbaum Associates (1980)

Gallistel, C. R., Shizgal, P., Yeomans, J. S.: A portrait of the substrate for self-stimulation.
Psych. Rev. 88, 228-273 (1981)

Garcia, J., Koelling, R. A.: Relation of cue to consequence in avoidance learning. Psychon. Sci. 4, 123-124 (1966)

Garcia, J., Rusiniak, K. W., Kiefer, S. W., Bermudez-Rattoni, F.: The neural integration of feeding and drinking habits. In: Advances in behavioral biology. vol 26. Woody, C. D. (ed.). New York, NY: Plenum Press (1982)

Goldstein, I., Papert, S.: Artificial intelligence, language and the study of knowledge. Cog. Sci. 1, 84-123 (1977)

Gormezano, I., Kehoe, E. J., Marshall, B. S.: Twenty years of classical conditioning research with the rabbit. In: Progress in psychobiology and physiological psychology. vol 10. Sprague, J. M., Epstein, A. N. (eds.). New York, NY: Academic Press (1983)

Hayes-Roth, F.: The role of partial and best matches in knowledge systems. In: Pattern directed inference systems. Waterman, D. A., Hayes-Roth, F. (eds.). New York, NY: Academic Press (1978)

Hebb, D. O.: The organization of behavior: A neuropsychological theory. New York, NY: Wiley (1949)

Hebb, D. O.: Essay on mind. Hillsdale, NJ: Lawrence Erlbaum Associates (1980)

Hilgard, E. R., Bower, G. H.: Theories of learning. Englewood Cliffs, NJ: Prentice Hall (1975)

Hinton, G. E., Anderson, J. A. (eds.): Parallel models of associative memory. Hillsdale, NJ: Lawrence Erlbaum Associates (1981)

Holland, J. H., Reitman, J. S.: Cognitive systems based on adaptive algorithms. In: Pattern directed inference systems. Waterman, D. A., Hayes-Roth, F. (eds.). New York, NY: Academic Press (1978)

Hubel, D. H., Wiesel, T. N.: Brain mechanisms of vision. Sci. Amer. 241, 150-162 (1979)

Hunt, E. B.: Artificial Intelligence. New York, NY: Academic Press (1975)

Hunt, E., Martin, J., Store, P.: Experiments in induction. New York, NY: Academic Press (1966)

Ito, M.: Mechanisms of motor learning. In: Competition and cooperation in neural nets. Amari, S., Arbib, M. A. (eds.). New York, NY: Springer-Verlag (1982a)

Ito, M.: Synaptic plasticity underlying the cerebellar motor learning investigated in rabbit's flocculus. In: Advances in behavioral biology. vol. 26. Woody, C. D. (ed.). New York, NY: Plenum Press (1982b)

John, E. R.: A model of consciousness. In: Consciousness and self-regulation. Schwartz, G. E., Shapiro, D. (eds.). New York, NY: Plenum Press (1976)

John, E. R.: A neurophysiological model of purposive behavior. In: Neural mechanism of goal-directed behavior and learning. Thompson, R. F., Hicks, L. H., Shvyrkow, V. B. (eds.). New York, NY: Academic Press (1980)

John, E. R., Schwartz, E. I.: The neurophysiology of information processing and cognition. Ann. Rev. Psych. 29, 1-29 (1978)

Kandel, E. R.: Cellular basis of behavior. San Francisco, CA: W.H. Freeman and Company (1976)

Kandel, E. R.: Neuronal plasticity and the modification of behavior. In: Handbook of physiology, the nervous system. Kandel, E. R. (ed.). Baltimore, MD: Waverly Press Inc. (1977)

Kandel, E. R.: Behavioral biology of Aplysia. San Francisco, CA: W.H. Freeman and Company (1979a)

Kandel, E. R.: Small systems of neurons. Sci. Amer. 241, 66-76 (1979b)

Kandel, E. R.: Cellular insights into behavior and learning. In: The Harvey lectures. 73, 19-92 (1979c)

Kandel, E. R., Schwartz, J. (eds.): Principles of neural science. New York, NY: Elsevier North-Holland (1981)

Kasamatsu, T.: Neuronal plasticity maintained by the central norepinephrine system in the cat visual cortex. In: Progress in psychobiology and physiological psychology vol. 10. Sprague, J. M., Epstein, A. N. (eds.). New York, NY: Academic Press (1983)

Kent, E. W.: The brains of men and machines. Peterborough, NH: BYTE/McGraw-Hill (1981)

Kety, S. S.: The evolution of concepts of memory. In: The neural basis of behavior. Beckman, A. L. (ed.). New York, NY: SP Medical and Scientific Books (1982)

Kim, E. H.-J., Woody, C. D., Berthier, N. E.: Rapid acquisition of conditioned eye blink responses in cats following pairing of an auditory CS with glabella tap US and hypothalamic stimulation. Jour. Neurophysiol. 49, 767-779 (1983)

Kishimoto, K, Amari, S.: Existence and stability of local excitations in neural fields. Jour. Math. Biol. 7, 303-318 (1979)

Klopf, A. H.: The hedonistic neuron. Washington DC: Hemisphere Publishing Corp. (1982)

Kohonen, T.: Associative memory. New York, NY: Springer-Verlag (1977)

Kohonen, T.: Content-addressable memories. New York, NY: Springer-Verlag (1980)

Kohonen, T.: Self-organized formation of topologically correct feature maps. Biol. Cyber. 43, 59-69 (1982a)

Kohonen, T.: Analysis of a simple self-organizing process. Biol. Cyber. 44, 135-140 (1982b)

Krasne, F. B.: Extrinsic control of intrinsic neural plasticity. Brain Res. 140, 197-216 (1978)

Kuffler, S. W., Nicholls, J. G.: From neuron to brain: A cellular approach to the function of the nervous system. Sunderland, MA: Sinaver Associates (1976)

Lashley, K. S.: Brain mechanisms and intelligence. Chicago, IL: Chicago Univ. Press (1929)

Lee, K. S.: Sustained enhancement of evoked potentials following brief, high-frequency simulation of the cerebral cortex in vitro.
Brain Res. 239, 617-623 (1982)

Lenat, D. B.: Automated theory formation in mathematics.
IJCAI5, 833-842 (1977)

Levy, W. B., Steward, O.: Synapses as associative memory elements in the hippocampal formation.
Brain Res. 175, 233-245 (1979)

Lindsay, P. H., Norman, D. A.: Human information processing. New York, NY: Academic Press (1977)

Lolordo, V. M.: Constraints on learning. In: Animal learning: Survey and Analysis. Bitterman, M. E., Lolordo, V. M. (eds.).
New York, NY: Plenum Press (1979)

Lorber, J.: Is your brain really necessary?
Science 210, 1232-1234 (1980)

Lubow, R. E.: Latent inhibition.
Psych. Bull. 79, 398-407 (1973)

Lund, R. D.: Development and plasticity of the brain.
New York, NY: Oxford Univ. Press (1978)

Lynch, G., Baudry,: The biochemistry of memory:
A new and specific hypothesis.
Nature (in press) (1983)

Lynch. G. S., Dunwiddie, T., Gribkoff, V.: Heterosynaptic depression: a postsynaptic correlate of long-term potentiation. Nature 266, 737-739 (1977)

Lynch, G., Schubert, P.: The use of in vitro brain slices for multidisciplinary studies of synaptic function.
Ann. Rev. Neurosci. 3, 1-22 (1980)

Mackintosh, N. J.: A theory of attention: variations in the associability of stimuli with reinforcements.
Psych. Rev. 82, 276-298 (1975)

Maier, S. F., Seligman, M. E. P.: Learned helplessness: Theory and evidence.
Jour. Exp. Psych. General 105, 3-46 (1976)

Mandler, G.: Mind and emotion.
New York, NY: Wiley (1975)

Manning, A.: Animal learning: Ethological approaches.
In: Neural mechanisms of learning and memory.
Rosenzweig, M. R., Bennett, E. L. (eds.).
Cambridge, MA: MIT Press (1976)

Mano, M. M.: Digital logic and computer design.
Englewood Cliffs, NJ: Prentice-Hall, Inc. (1979)

Marslen-Wilson, W. D., Teuber, H. L.: Memory for remote events in anterograde amnesia: recognition of public figures from news photographs.
Neuropsychologia 13, 353-364 (1975)

McCawley, J. D.: Everything that linguists have always wanted to know about logic.
Chicago, IL: Univ. Chicago Press (1981)

McClelland, J. L., Rumelhart, D. E.: An interactive activation model of the effect of context in perception. Part I. Center for human information processing. Univ. of Calif., San Diego. CHIP 91 (1980)

McCulloch, W. S., Pitts, W.: A logical calculus of the ideas immanent in nervous activity.
Bull. Math. Biophys. 5, 115-133 (1943)

McGaugh, J. L.: Behaviorism in psychology and biology.
In: Encyclopedia of the twentieth century.
Cappeletti, V., (ed.). Rome, Istituto deila enciclopedia Italiana (1981)

McGaugh, J. L.: Hormonal influences on memory.
Ann. Rev. Psych. 34, 297-323 (1983)

McNaughton, B. L., Barns, C. A.: Physiological identification and analysis of dentate granule cell responses to stimulation of the medial and lateral perforant pathways in the rat.
Jour. Comp. Neurol. 175, 439-453 (1977)

McNaughton, B. L., Douglas, R. M., Goddard, G. V.: Synaptic enhancement in fascia dentata: Cooperativity among coactive afferents. Brain. Res. 157, 277-293 (1978)

Mervis, C. B., Rosch, E.: Categorization of natural objects. Ann. Rev. Psych. 32, 89-115 (1981)

Michalski, R. S., Carbonell, J. G., Mitchell, T. M. (eds.): Machine learning. Palo Alto, CA: Tioga Publishing (1983)

Miles, F. A., Evarts, E. V.: Concepts of motor organization. Ann. Rev. Psych. 30, 327-362 (1979)

Miller, R. R.: Behavioral constraints on biochemical and physiological models of memory. In: Changing concepts of the nervous system. Morrison, A. R., Strick, P. L. (eds.). New York, NY: Academic Press (1982)

Milner, B.: Memory and the medial temporal regions of the brain. In: Biology of memory. Pribram, K. H., Broadbent, D. E. (eds.). New York, NY: Academic Press (1970)

Minsky, M.: Steps toward artificial intelligence. In: Computers and thought. Feigenbaum, E. A., Feldman, J. (eds.). New York, NY: McGraw-Hill (1963)

Minsky, M., Papert, S.: Perceptrons. Cambridge, MA: MIT Press (1972)

Mollon, J. D.: Color vision. Ann. Rev. Psych. 33, 41-85 (1982)

Neisser, U., Weene, P.: Hierarchies in concept attainment. Jour. Exp. Psych. 64, 640-645 (1962)

Newell, A.: Physical symbol systems. Cog. Sci. 4, 135-183 (1980)

Newell, A, Simon, H. A.: Human problem solving. Englewood Cliffs, NJ: Prentice-Hall (1972)

Nilsson, N. J.: Learning machines. New York, NY: McGraw-Hill (1965)

O'Brien, J. H., Quinn, K. J.: Central mechanisms responsible for classical conditioned changes in neuronal activity. In: Advances in behavioral biology. vol. 26. Woody, C. D. (ed.). New York, NY: Plenum Press (1982)

Oguztoreli, M. N.: Activity analysis of neural networks. Biol. Cyber. 34, 159-169 (1979)

Olds, M. E., Forbes, J. L.: The central basis of motivation: intracranial self-stimulation studies. Ann. Rev. Psych. 32, 523-574 (1981)

Overton, K. L., Arbib, M. A.: The extended branch-arrow model of the formation of retino-tectal connections. Biol. Cyber. 45, 157-175 (1982)

Pearce, J. M., Hall, G.: Loss of associability by a compound stimulus comprising excitatory and inhibitory elements. Jour. Exp. Psych: Anim. Behav. Proc. 5, 19-30 (1979)

Powers, W. T.: Behavior: The control of perception. New York, NY: Aldine Publishing Co. (1973)

Pugh, G. E.: The biological origin of human values. New York, NY: Basic Books Inc. (1977)

Quillian, M. R.: Word concepts: a theory and simulation of some basic semantic capabilities. Behav. Sci. 12, 410-430 (1967)

Ratcliff, R., McKoon, G.: Does activation really spread? Psych. Rev. 88, 454-462 (1981)

Reilly, D. L., Cooper, L. N., Elbaum, C.: A neural model for category learning. Biol. Cyber. 45, 35-41 (1982)

Rescorla, R. A.: Pavlovian conditioning and its proper control procedures. Psych. Rev. 74, 71-80 (1967)

Rescorla, R. A.: Informational variables in Pavlovian conditioning. In: Psychology of learning and motivation Vol. 6. Bower, G. H. (ed.). New York, NY: Academic Press (1972)

Rescorla, R. A.: Evidence for a "unique stimulus" account of configural conditioning. Jour. Comp. Phys. Psych. 85, 331-338 (1973)

Rescorla, R. A., Durlach, P. J.: Within-event learning in Pavlovian conditioning. In: Information processing in animals. Spear, N. E., Miller, R. R. (eds.). Hillsdale, NJ: Lawrence Erlbaum Associates (1981)

Rescorla, R. A., Holland, P. C.: Some behavioral approaches to the study of learning. In: Neural mechanisms of learning and memory. Rosenzweig, M. R., Bennet E. L. (eds.). Cambridge, MA: MIT Press (1976)

Rescorla, R. A., Solomon, R.: Two-process learning theory: relationships between Pavlovian conditioning and instrumental learning. Psych. Rev. 74, 151-182 (1967)

Rescorla, R. A., Wagner, A. R.: A theory of Pavlovian conditioning. In: Classical conditioning II. Black, A. H., Prokasy, W. F. (eds.). New York, NY: Appleton-Century-Crofts (1972)

Roberts, W. A., Grant, D. S.: Studies in short-term memory in the pigeon using the delayed matching-to-sample procedure. In: Processes of animal memory. Medin, D. L., Roberts, W. A., Davis, R. T. (eds.). Hillsdale, NJ: Lawrence Erlbaum Associates (1976)

Robertson, P: Non-temporal prediction - A distributed system for concept acquisition. CSCSI/SCEIO Conf. Pro. (1982)

Robinson, D. A.: The use of control systems analysis in the neurophysiology of eye movements. Ann. Rev. Neurosci. 4, 463-503 (1981)

Rolls, E. T., Bortan, H. J., Mora, F.: Neurophysiological analysis of brain-simulation reward in the monkey. Brain Res. 194, 339-357 (1980)

Rolls, E. T.: Neuronal mechanisms underlying the formation and disconnectin of associations between visual stimuli and reinforcement in primates. In: Advances in behavioral biology. vol. 26. Woody, C. D. (ed.). New York, NY: Plenum Press (1982)

Rosenblatt, F.: Principles of neurodynamics: Perceptrons and the theory of brain mechanisms. Washington, DC: Spartan Books (1962)

Rosenblatt, F.: The perceptron: A probabilistic model for information storage and organization in the brain. Psych. Rev. 65, 386-408 (1958)

Rosch, E.: Principles of categorization. In: Cognition and categorization. Rosch, E., LLoyd, B. B. (eds.). Hillsdale, NJ: Lawrence Erlbaum Associates (1978)

Rumelhart, D. E., McClelland, J. L.: An interactive activation model of the effect of context in perception. Part II. Center for human information processing. Univ. of Calif., San Diego. CHIP 95 (1980)

Rumelhart, D. E., Siple, P.: The process of recognizing tachistoscopically presented words. Psych. Rev. 81, 99-118 (1974)

Sahley, C., Rudy, J. W., Gelperin, A.: An analysis of associative learning in a terrestrial mollusc. Jour. Comp. Physiol. 144, 1-8 (1981)

Samuel, A. L.: Some studies in machine learning using the game of checkers. In: Computers and thought. Feigenbaum, E. A., Feldman, J. (eds.). New York, NY: McGraw-Hill (1963)

Sejnowski, T. J.: Skeleton filters in the brain. In: Parallel models of associative memory. Hinton, G. E., Anderson, J. A. (eds.). Hillsdale, NJ: Lawrence Erlbaum Associates (1981)

Selfridge, O.: Pandemonium, a paradigm for learning. In: Proc. symp. on the mechanization of thought processes. Blake, D., Uttley, A. (eds.). London, H. M. Stationery office (1959)

Seligman, M. E. P.: Helplessness. San Francisco, CA: W.H. Freeman and Company (1975)

Shapiro, S. C.: The SNePs semantic network processing system. In: Associative networks. Findler, N. V. (ed.). New York, NY: Academic Press (1979)

Shaw, G. L.: Space-time correlations of neuronal firing related to memory storage capacity. Brain Res. Bull. 3, 107-113 (1978)

Shepherd, G. M.: The synaptic organization of the brain. New York, NY: Oxford Univ. Press (1979)

Shortliffe, E. H.: A model of inexact reasoning in medicine. Math. Biosci. 23, 351-379 (1975)

Spinelli, D. N., Hirsch, H. V. B., Phelps, R. W., Metzler, J.: Visual experience as a determinant of the response characteristics of cortical receptive fields in cats. Exp. Brain Res. 15, 289-304 (1972)

Spinelli, D. N., Jensen, F. E.: Plasticity: the mirror of experience. Science 203, 75-78 (1979)

Spinelli, D. N., Jensen, F. E.: Plasticity, experience and resource allocation in motor cortex and hypothalamus. In: Advances in behavioral biology. vol. 26. Woody, C. D. (ed.). New York, NY: Plenum Press (1982)

Squire, L. R.: The neuropsychology of human memory. Ann. Rev. Neuro. 5, 241-273 (1982)

Squire, L. R., Cohen, N. J., Nadel, L.: The medial temporal region and memory consolidation: a new hypothesis. In: Memory consolidation. Weingartner, H., Parker, E. (eds.). Hillsdale, NJ: Lawrence Erlbaum Associates (1983)

Sutton, R. S., Barto, A. G.: Toward a modern theory of adaptive networks: Expectation and prediction. Psych. Rev. 88, 135-170 (1981)

Thompson, R. F.: The engram found? Initial localization of the memory trace for a basic form of associative memory. In: Progress in psychobiology and physiological psychology. vol. 10. Sprague, J. M., Epstein, A. N. (eds.). New York, NY: Academic press (1983)

Thompson, R. F., Berger, T. W., Berry, S. D., Clark, G. A., Kettner, R. N., Lauond D. G., Mauk, M. D., McCormick, D. A., Solomon P. R., Weisz, D. J.: Neuronal substrates of learning and memory: Hippocampus and other structures. In: Advances in behavioral biology. vol 26. Woody, C. D. (ed.). New York, NY: Plenum Press (1982).

Thompson, R. F., Berger, T. W., Berry, S. D., Hoehler, F. K., Kettner, R. E., Weisz, D. J.: Hippocampal substrate of classical conditioning. Physiol. Psych. 8, 262-279 (1980)

Thompson, R. F., Berger, T. W., Madden IV, J.:
Cellular processes of learning and memory in the mammalian CNS. Ann Rev. Neurosci. 6, 447-491 (1983)

Thorndike, E. L.: The associative process in animals. Biological lectures from the marine biological laboratory at Woods Hole. Boston, MA: Antheneum (1899)

Thorndike, E. L.: The psychology of learning. New York, NY: Columbia Univ. Press (1913)

Tsukahara, N.: Synaptic plasticity in the mammalian central nervous system. Ann. Rev. Neurosci. 4, 351-379 (1981)

Tunturi, A. R.: A difference in the representation of auditory signals for the left and right ears in the isofrequency contours of right middle ecosylvian auditory cortex in the dog.
Amer. Jour. Physiol. 168, 712-727 (1952)

Uttley, A. M.: Information transmission in the nervous system. New York, NY: Academic Press (1979)

Waterman, D. A.: Generalization learning techniques for automating the learning of heuristics.
Artificial Intelligence 1, 121-170 (1970)

Waterman, D. A., Hayes-Roth, F. (eds.): Pattern directed inference systems. New York, NY: Academic Press (1978)

Weinberger, N. M.: Effects of conditioned arousal on the auditory system. In: The neural basis of behavior. Beckman, A. (ed.). New York, NY: Spectrum Publishing (1982)

Wickelgren, W. A.: Chunking and consolidation.
Psych. Rev. 86, 44-60 (1979)

Widrow, B., Gupta, N. K., Maitra, S.: Punish/reward: learning with a critic in adaptive threshold systems.
IEEE Trans. Syst. Man Cyber. 3, 455-465 (1973)

Winston, P. H.: The psychology of computer vision. New York, NY: McGraw-Hill (1975)

Woody, C. D.: Memory, learning and higher mental function. New York, NY: Springer-Verlag (1982a)

Woody, C. D.: Neurophysiological correlates of latent facilitation. In: Advances in behavioral biology. vol 26. Woody, C. D. (ed.). New York, NY: Plenum Press (1982b).

Woody, C. D. (ed.): Advances is behavioral biology. vol. 26. New York, NY: Plenum Press (1982c)

Woody, C. D., Buerger, A. A., Ungar, R. A., Levine D. S.: Modeling aspects of learning by altering biophysical properties of a simulated neuron. Biol. Cyber. 23, 73-82 (1976)

Woody, C. D., Kim, E. H.-J., Berthier, N. E.: Effects of hypothalamic stimulation on unit responses recorded from neurons of sensorimotor cortex of awake cats during conditioning. Jour. Neurophysiol. 49, 780-791 (1983)