A BOOLEAN COMPLETE NEURAL MODEL OF ADAPTIVE BEHAVIOR

by

Steve Hampson
and
Dennis Kibler

Technical Report #190a

Department of Information & Computer Science
University of California, Irvine
Irvine, CA 92717

November 1982

TABLE OF CONTENTS

A BOOLEAN COMPLETE NEURAL MODEL OF ADAPTIVE BEHAVIOR

1.0 ABSTRACT

A multi-layered neural assembly is developed which has the capability of learning arbitrary Boolean functions. Though the model neuron is more powerful than those previously considered, assemblies of neurons are needed to detect non-linearly separable patterns. Algorithms for learning at the neuron and assembly level are described. The model permits multiple output systems to share a common memory. Learned evaluation allows sequences of actions to be organized. Computer simulations demonstrate the capabilities of the model.

2.0 INTRODUCTION

We define and implement a neural system capable of adaptive behavior in a completely defined environment. Behavior is modeled as the application of specific operators in response to specific inputs. Behavioral completeness requires that any stimulus (input pattern) potentially be able to trigger any response (set of operators). Learning adjusts the input-output connections so that behavior converges on the correct stimulus response (SR) mapping. Learning completeness requires that any SR mapping be learnable. 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.

The model's ability to learn arbitrary Boolean functions is an important capability, since the domain of neural models is often limited to linearly separable functions. This is a serious limitation since the percentage of possible Boolean functions that are linearly separable rapidly approach zero as the number of features increases (Robertson 82). The proposed model is more general than others, but is still significantly restricted since spatial, temporal and relational inputs cannot be explicitly modeled. However, it may be possible to extend the Boolean formalism to include those domains. We defer discussion of these possibilities until the basic model has been developed.

The system is built using a single type of neuron-like element. This model neuron was developed on the basis of biological evidence and theoretical constraints on its necessary properties. It is a formalization of the "almost" gate suggested by Kent (Kent 81), and is more powerful than the standard binary pattern classifier (Nilsson 65).

Emphasis is also placed on network controlling processes in order to effectively control synaptic plasticity. 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. The general form of desired behavior is identified, and mechanisms are proposed to implement it.

The model has the general characteristics of a production system, so it is capable of potentially interesting behavior, and though it is not developed as a strictly biological model, the central principles are 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 to control the network as necessary. In particular, it has been suggested that there are specific plasticity controlling systems which control the modification of synaptic connections (Krasne 78, Feldman 81, Kety 82).

3.0 RELATED WORK

A common division of neural learning processes is between learning 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.

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, learning is essentially limited to picking out statistical association 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 behavior, some input information must be interpreted as instructive, and used to adjust the function. In addition, the Hebbian model is susceptible to a number of stability and saturation problems (Sutton and Barto 81).

Learning with a teacher implies that there are inputs to a node which have the special properties of a teaching signal. Threshold pattern classifiers are a common example (Nilsson 65). In this case, the 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 an arbitrarily complex function by indicating for each input whether the current output is too high or low, or by specifying the correct output directly.

Without belaboring the biological validity of Hebb's model (there is little evidence for it), 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. However, goal directed behavior is an important aspect of neural functioning. Development of the present model concentrates on that process. This requires instructive feedback to indicate correct output. Biologically, both data and goal driven learning appear to contribute (Spinelli et al. 72, Spinelli and Jensen 79).

At the system level, 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 nodes that can be on at any one time. Many model networks incorporate this principle as diffuse or random inhibitory connections within the net (Amari 77), and it is a common process in biological systems (Linsay and Norman 77). Lateral inhibition of learning is similar, except that the firing of a node prevents learning in other nodes (Fukushima 75). The latter process is utilized in this model.

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).

4.0 MODEL DEVELOPMENT

The proposed model is developed in four stages:

1) structure and training of a model neuron
2) structure and training of a single operator
3) structure and training of multiple operators
4) evaluation and credit assignment

4.1 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 adopted since it is sufficiently powerful for Boolean completeness of assemblies, and is relatively simple to implement. Such a function is probably well within neural capabilities, and is the most common functional form used to model neural computation. A linear function can be used to implement the "at least X of N features" function. This includes OR (at least 1 of N), and AND (at least N of N) as its extremes. Significantly, this function can be viewed as a prototypic category description.

Prototypic categorization is generally defined in terms of similarity to a central (prototypic) example. 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 the standard single threshold, binary output function, a continuous, three-valued logic is utilized. Output above and below a 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 common biologically (Sejnowsi 81). Output beyond the limits of 1 and -1 is interpreted as absolute certainty.

The neural output function has two values (synaptic weights) associated with each input feature, 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). The explicit representation of feature absence permits categories to be defined on the basis of missing features, and avoids confounding "unknown" with "not present". There may be any number of inputs. Output is calculated as:

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

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

A node that is capable of behaving as the (at least X of N) function can be achieved by setting Pi to (2*(N-X)+1)/N and Ni to -(2*X-1)/N. The presence of X features yields an output of 1, and X-1 features give an output of -1 for values of N >= 1. If X equals N we have, as a special case, the "AND" function. In this case Pi is 1/N and Ni is -(2*N-1)/N. If X equals 1 we have, as another special case, the "OR" function. In this case Pi is (2*N-1)/N and Ni is -1/N.

4.2 Training A Node

A neuron must be able to adjust its output in order to improve its performance. The ability to train a linear function as a binary pattern classifier is well known as the perceptron convergence theorem (Nilsson 65). This process is similar to biological learning in the gill withdrawal reflex of Aplysia (Kandel 79). 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.

A similar learning process was adopted in this model. A synaptic weight is increased if:

1) the node should be on
2) the node's output was less than 1
3) there was input on that line
4) the weight was not already its maximum value

The second constraint is consistent with the observation that "organisms only learn when events violate their expectations" (Rescorla and Wagner 72). Weights are reduced by a complimentary process.

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. Weights are modified in proportion to the feature's predictive potential. This was effective in excluding irrelevant activity from weight modification. It is also consistent with contingency theories of learning (Rescorla 72, Rescorla and Wagner 72, Sutton and Barto 81).

A "trace" of conditional probability is iteratively computed with one of two methods:

Method I

```
[TFi]  := [TFi] + (T * Fi - [TFi]) * r
[Fi]   := [Fi] + (Fi - [Fi]) * r
[T|Fi] := [TFi]/[Fi]
[T]    := [T] + (T - [T]) * r
```

Method II

```
[T|Fi] := [T|Fi] + (T - [T|Fi]) * Fi * r
[T]    := [T] + (T - [T]) * r
```

Fi is the input value for feature i, T is a teacher signal (-1 or 1) indicating whether the node should have been off or on, [Fi] is an estimate of the probability (frequency) of

Fi, [T] is the probability of T, [TFi] is the probability of co-occurrence of T and Fi, and [T|Fi] is the probability of T given Fi. The "memory length" of the function is determined by the rate constant r. As in standard conditional probability, the predictiveness of a feature can be determined by comparing [T|Fi] to [T].

The use of conditional probability works well in identifying those input weights which should be increased in magnitude, but selectively identifying those which are too large and should be decreased in magnitude proved more difficult. The solution was to make a distinction between learning (increasing in magnitude) and unlearning (decreasing in magnitude). Their relative contributions depend on one's faith in the accuracy of past learning. Arbitrarily, the contributions of unlearning and learning were set at .1 and .9 of the total change.

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, weights only go beyond -2 and 2 if 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 := .1 + .9 * Wt/4

This adjustment of un_frac proved quite effective in

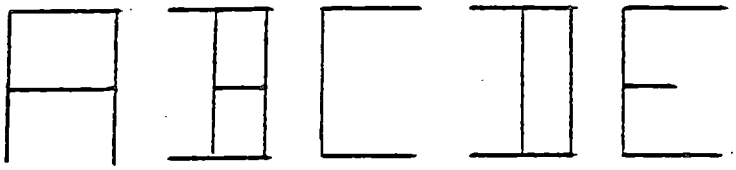
controlling inappropriate weight accumulation.

As an example of a node's ability to learn AND's, a node was sequentially trained to uniquely detect each letter of the alphabet using a 14-stroke representation scheme (Fig. 1, 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.

In Figure 2, the function (at least 3 of 4) is learned. The complete input space (16 patterns) was continuously cycled through, and the node's output for representative inputs with N, N-1, N-2, N-3, and N-4 features is shown. The 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 acquisition order is consistent with human learning of prototypes (Mervis and Rosch 81).

4.3 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 2-level structure is adequate (Fig. 3). This is similar to the structure of a perceptron (Rosenblatt 62). Unlike the standard perceptron, where only the top node is trained and the lower plane is



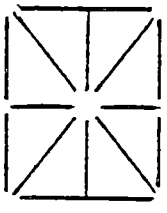


Figure 1. Fourteen stroke alphabetic representation

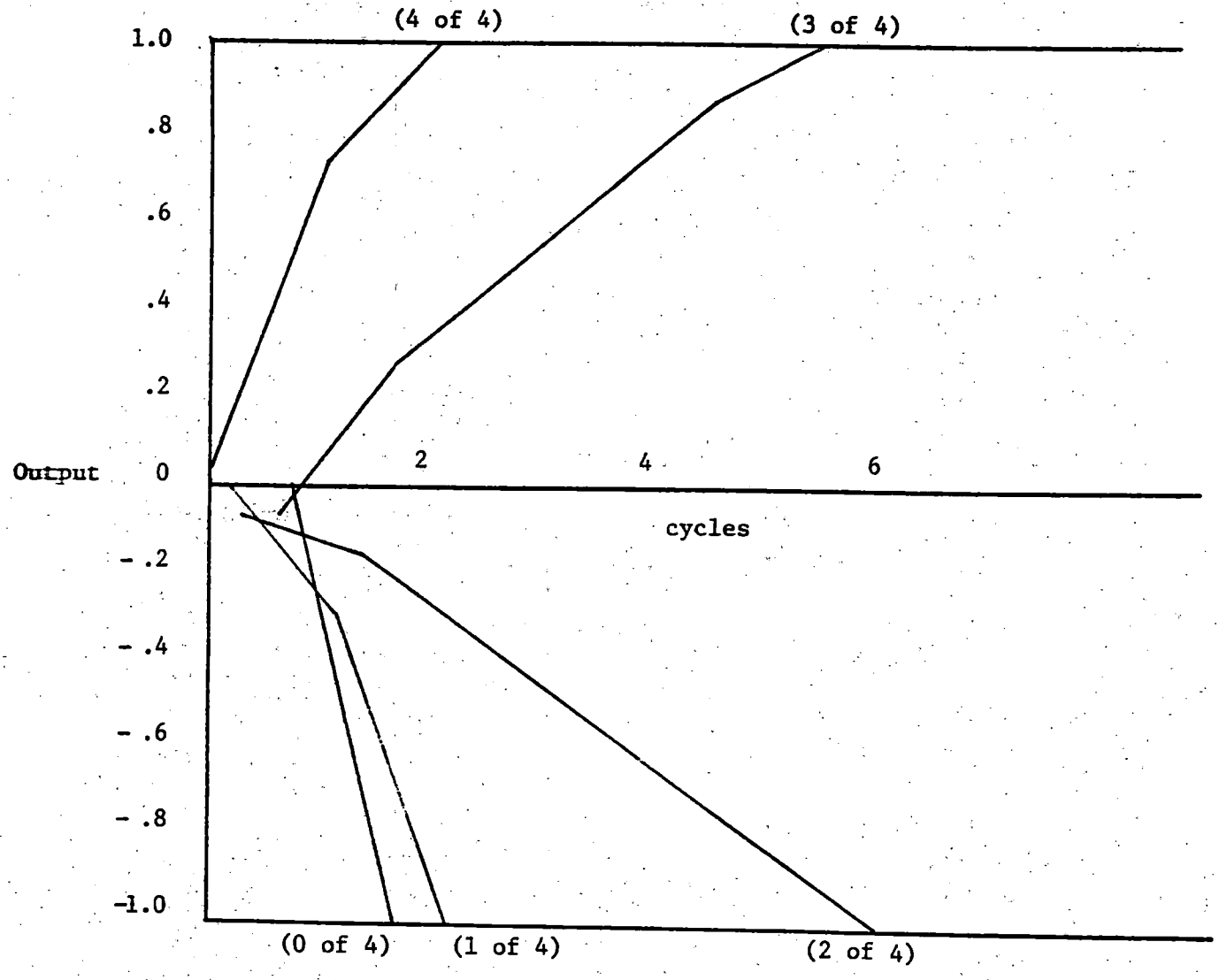


Figure 2. Prototype learning of function "at least 3 or 4"

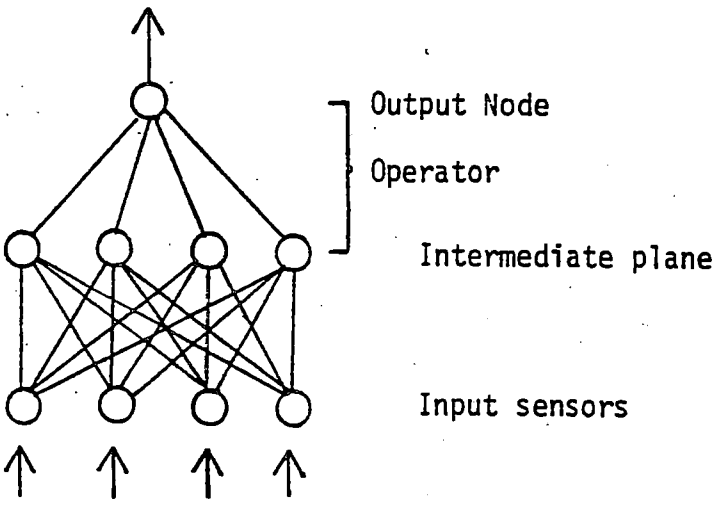


Figure 3. A minimum structure capable of detecting Boolean functions

"hard-wired", both levels are trainable. The problems of restricted interconnection (Minsky and Papert 72) are not addressed.

Training an operator can be viewed as two separate processes: training the top node and training the plane. Training the top node is trivial if the lower nodes are already trained, and impossible if they aren't since it uses their output. The problem is to train the bottom nodes. The constraints are relatively simple:

1) For all "false" input patterns (T = -1)
   all nodes should be off
2) For all "true" input patterns (T = 1)
   at least one node should be on

The first constraint is easy to implement. The desired value of T = -1 is simply broadcast to all nodes. The second is accomplished by broadcasting T = 1 plus lateral inhibition of learning. As the node with maximum output (Max_out) approaches 1, the learning rate of all other nodes is reduced to 0. This is implemented by controlling Rt in the rate equation:

```
    New_out := Old_out + (1 - Old_out) * c *  Rt
where:
    Rt := 1 - Max_out
    If Rt > 1 then Rt := 1
```

The constant c determines the maximum learning rate. When at least one node detects each "true" input, no further learning is necessary.

This process was modified by adjusting learning to include the relative output of nodes in the plane.

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

This recognizes the fact that the nodes most likely to learn a pattern are those with the greatest predisposition to respond to it. 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 learning algorithm encodes "true" input patterns as potentially overlapping categories in the lower plane. The algorithm tends to produce the largest categories possible, thus requiring the minimum number of nodes, though the absolute minimum (perfect generalization) is not guaranteed. The output node can be trained simultaneously with the plane. It simply doesn't do very well until the plane converges.

System capacity and learning speed can be increased 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. In addition, various interconnection schemes are possible. For example, nodes can be connected with only the plane below them, all planes below them, or all other nodes in the system. Any of these structures can be trained by applying the learning process independently to each plane. These variations suggest a

number of interesting biological analogies (Hampson 83), but for simulation efficiency the minimal network was the most extensively investigated.

Using a small network (one plane, 20 nodes), the operator training algorithm was tested on a number of 4-feature Boolean functions. 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 T) reached zero for a complete cycle. All functions were successfully learned, in an average of 12 cycles.

A standard set of 40 4-feature functions was always run to determine the effects of program modification. In Figures 4abc, 3 of these functions and their learning curves are shown. In Figure 5 a single learning curve is shown for the network as it is sequentially retrained to detect those three examples. Examples 4b and 4c are among the hardest and easiest functions of 4 features for the network to learn. As can be seen in Fig 4b, error does not necessarily decrease monotonically. This results from training the output node while the plane is still incompletely trained.

4.4 Multiple Operators

Finally, multiple operators are combined into a single behavioral system. Completeness in this system requires mapping arbitrary inputs to arbitrary sets of outputs. A

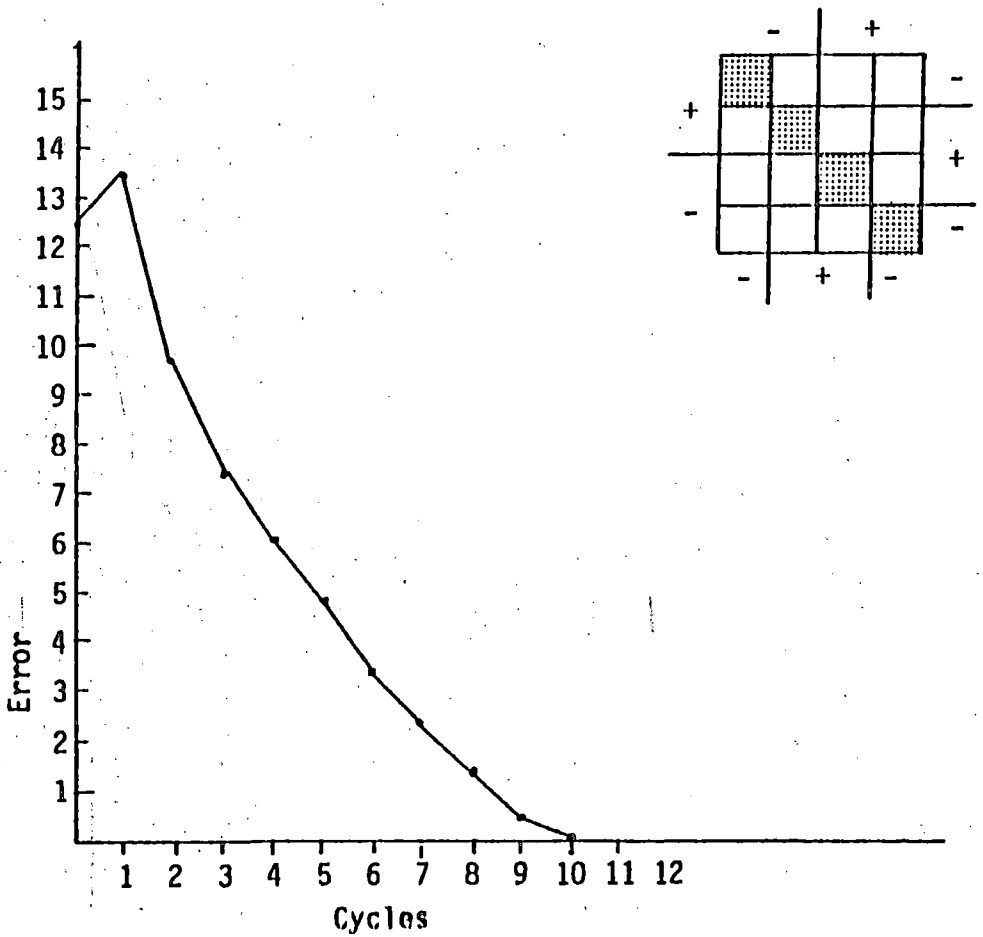


Figure 4a. Accumulated error per cycle while learning a Boolean function

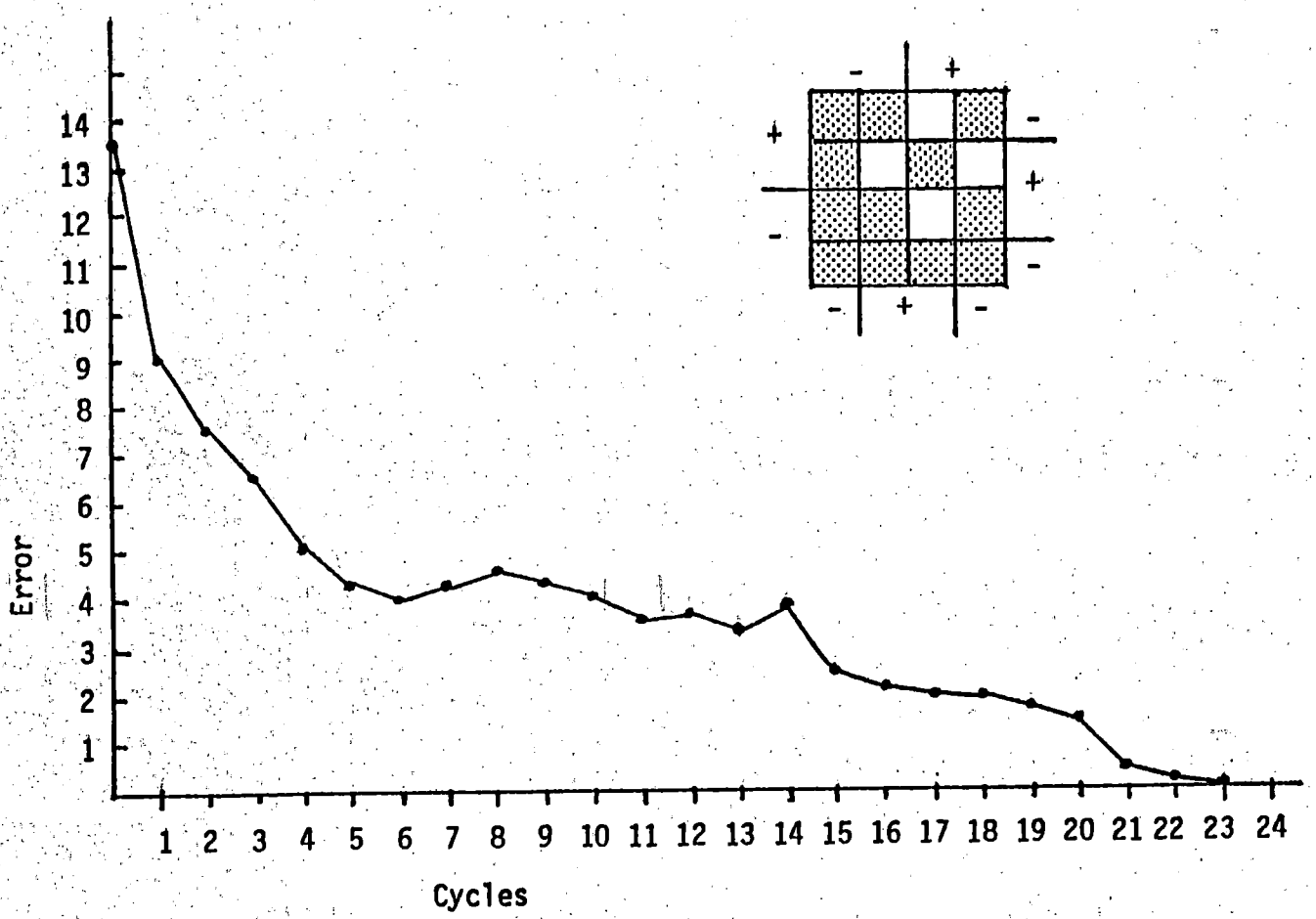


Figure 4b. Accumulated error per cycle while learning a Boolean function

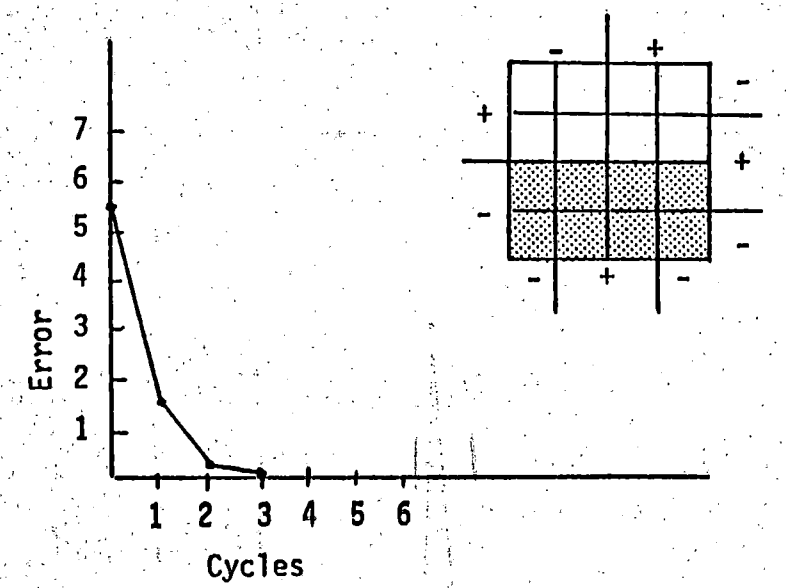


Figure 4c. Accumulated error per cycle while learning a Boolean function

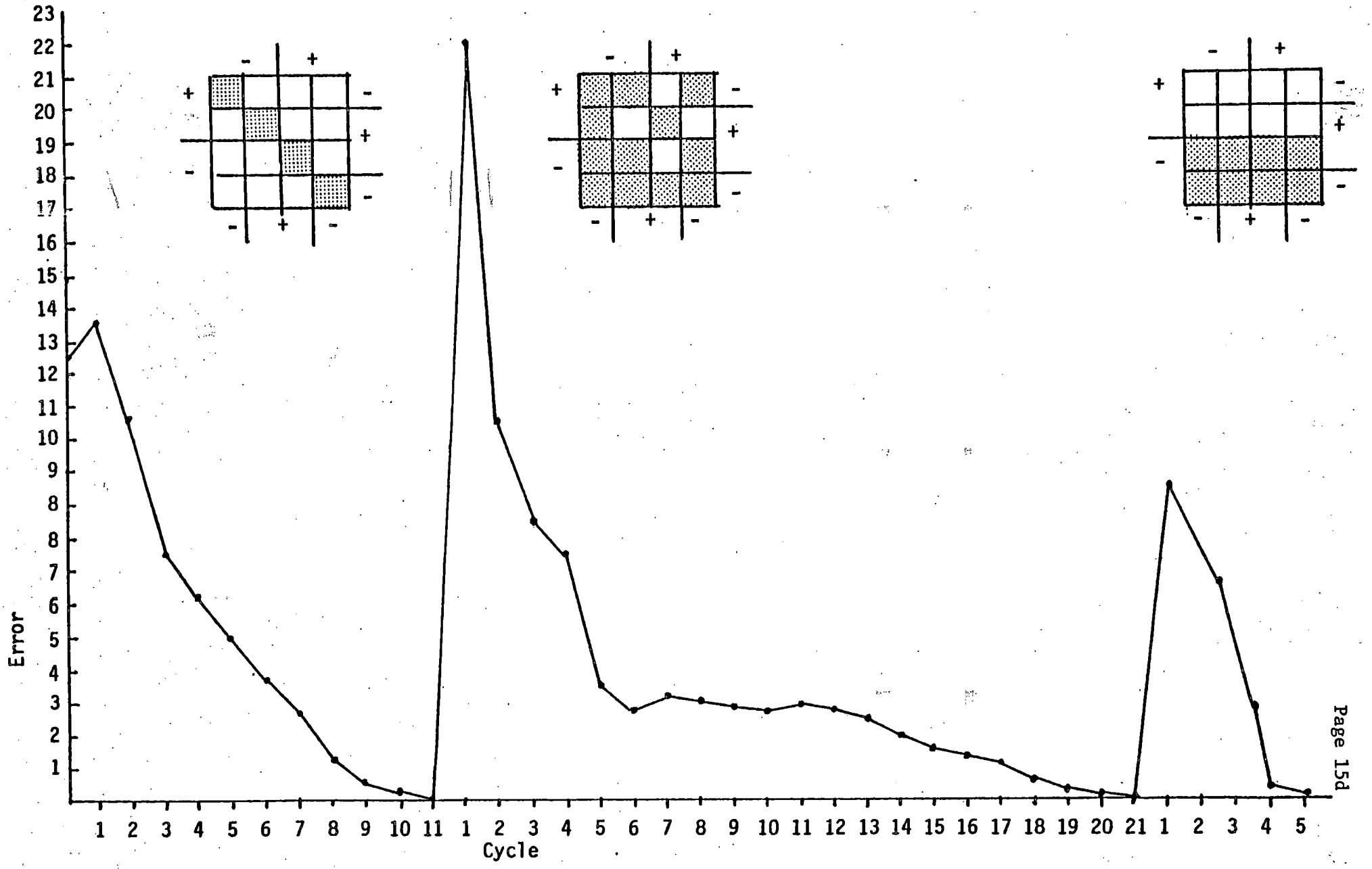


Figure 5. Retraining a network

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. It seems unavoidable that shared memory is required for systems of any size.

What is needed is a way to augment (or replace) the separate, dedicated memory planes of each operator with a single, shared pool of nodes. Since a two level system is logically adequate for separate operators, it is also computationally complete in a shared memory system. Operators can be reduced to single nodes sharing a single lower plane. The previous learning algorithm is inappropriate for shared memory, so another procedure was developed. It consists of two separate processes:

1) Input driven categorization
2) Goal driven focusing

These two processes will be considered separately. The approach is similar to one utilized in (Reilly et al. 82).

4.5 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 constraint provides the plane with an intrinsic function, learning until every input is categorized one way or another. Even in the absence of operator (goal driven) information, learning will take place until input is successfully categorized. Learning is identical to that in the previous section, except that T is equal to 1 for all inputs.

Useful categories tend to reflect the natural categories in environmental input (Mervis and Rosch 81), so input driven learning should capture those natural associations. Most neural 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 categories which will most likely prove useful.

4.6 Goal Driven Focusing

Goal driven focusing can be summarized as:

If a behavioral (operator) error occurs, the common memory does not represent the current input pattern in a specific enough form for the operators to use. Therefore the common memory should be adjusted in a direction that produces a more specific representation.

With a single operator, output error can be computed as the difference between the operator's actual and correct 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. Memory is 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.

If operators are limited to single nodes detecting (at least X of N) features, any adjustment process should converge on a representation which is decodeable by that function. An obviously decodeable extreme results if each input pattern is uniquely represented by one 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 broken up into smaller ones. One mechanism for achieving this is to focus the 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. Input

driven learning fills in any conceptual gaps left by the focusing process.

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.

A neural system displaying what might be interpreted as focusing has been described in the hippocampus (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.

The 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, the model was also modified to use an adjustable threshold, but since the desired behavior can

also be implemented without structural modification, the threshold technique is not developed here.

The primary method of adjusting weights for "invert" focusing is:

A) Center prototype
   1) reduce 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.

Error driven focusing is a reasonably efficient process since memory modification takes place only when behavior is apt to be improved. If no output errors occur, the representation is 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. The operators can be continuously trained, since the common memory represents, or will soon represent, the input in a decodeable form.

Multiple plane systems are also possible for the common memory. As before, each plane can be trained independently. Tapered focusing can also be used as a mechanism to facilitate the sharing of information in layered systems.

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 a simple but effective approach for the self-organization of hierarchical systems.

With complete interconnection between nodes, an alternative method of information representation is 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 is not utilized.

In general, development and testing of the common memory was limited to 4-feature functions. The same set of 40 functions used to test operator training 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, and focusing is generally superior when specific instances are important.

4.7 Evaluation And Credit Assignment

There are two basic problems of credit assignment when a single evaluation signal is used to instruct multiple operators. 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).

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)

If the current evaluation is less than optimal, staying the same can be treated the same as getting worse. Thus the resulting change in evaluation indicates the correctness of the preceding behavior. 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.

The second problem of sequential credit assignment can be solved 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 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 final state, n, 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 evaluation leading to the final goal state. Primary evaluation identifies goal states, and secondary evaluation provides immediate feedback for transitions leading toward those states. Since the evaluative gradient eventually saturates, positive evaluation must temporarily habituate to prevent the possibility of looping action sequences. This approach is consistent with what is known of biological reinforcement systems (Gallistel 73, Pugh 77).

Samuel implemented a similar learning process in his checkerboard evaluation function (Samuel 63). By simulating succeeding states, the learned evaluation can be used to

choose the next transition. However, 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).

The evaluation system can use the output of the common memory, so single node evaluation is computationally complete. However, a large change in evaluation may indicate that common memory categorization is insufficiently specific for accurate evaluation. To guarantee convergence on correct evaluation, the magnitude of evaluation change is included in the focusing error signal. As before, the logical extreme of completely decoding the input space is trivially adequate since the correct evaluation can be uniquely attached to each input pattern. Thus evaluation is logically complete and potentially efficient since memory modification is proportional to evaluation error.

The standard set of 40 4-feature, single operator Boolean functions was used to test this learning process. A single input pattern 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.

5.0 COMPLETENESS AND EFFICIENCY

Because of its ability to represent disjunctive normal form, (and "prototypic" normal form in general), the system is representationally 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 and evaluators, 1 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. Because of the expense of

simulating parallel process on a sequential machine, 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 tuned for a small number of features, but the conceptual approach appears to be extendable. With common memory focusing there is promise for proving learning completeness (Hampson 83). If so, operator training could be modified to include sufficient focusing to guarantee its convergence as well.

Efficiency in time and space are important, though often contradictory constraints on network characteristics. For example, rapid focusing produces rapid (efficient) learning, but requires many more nodes. The present model was developed primarily with an eye on completeness since that is a well defined goal. Efficiency is a matter of trade-offs determined by the particular situation. However, in terms of node utilization, representation in prototypic normal form is significantly more efficient than disjunctive or conjunctive normal representation. In addition, the operator training process tends toward maximum generalization, thus requiring the minimum number of nodes. Such issues will be of increasing importance as neural models are scaled up for practical application.

6.0 DISCUSSION

The complete model is shown in Figure 6. 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 provides behavioral evaluation/instruction by comparing succeeding evaluations to see if things get better or worse.

This system demonstrates a biologically plausible model of adaptive behavior. It is constructed with linear function elements which are probably well within neural capabilities. Various forms of learning are proposed in order to organize appropriate action. These learning processes are also suggested by known neurophysiology. Thus, though not explicitly physiological, the system is intended to be consistent with biological capabilities. By modeling the more abstract 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 (e.g., vision) in a variety of ways.

Operator training and focusing seem to represent two fundamentally different learning paradigms. Operator training is equivalent to many artificial intelligence (AI)

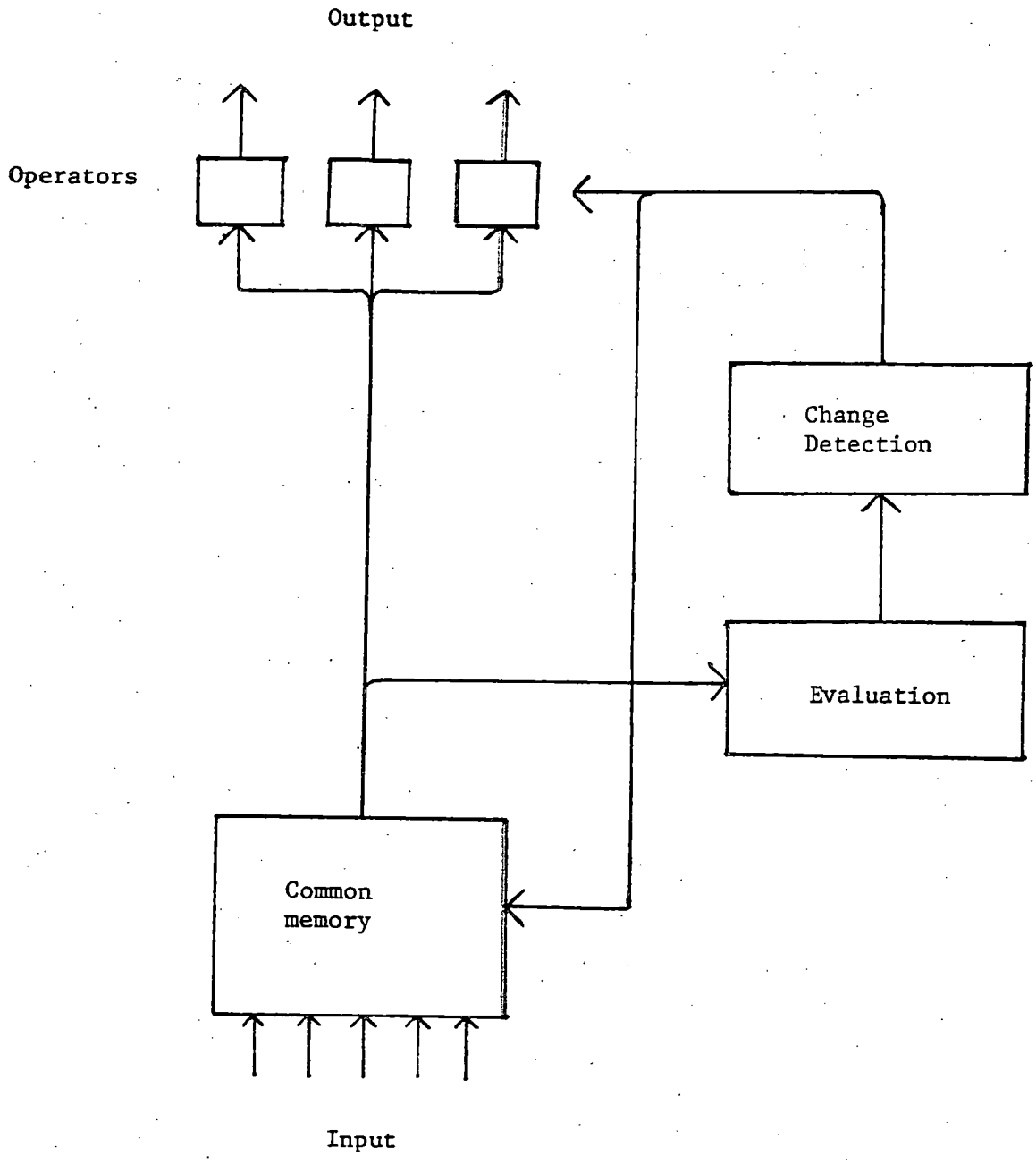


Figure 6. The model system

learning situations. That is, a category description is produced from presentation of positive and negative examples (Cohen and Feigenbaum 82 ch. D3). Focusing is driven by positive instances only. The operator training process is good at learning categoric generalization, but inappropriate for learning specific instances. Common memory focusing, on the other hand, is capable of learning unique occurrences in "one shot" if necessary, but is poor at generalization. This is consistent with a biological dichotomy observed between behavioral adjustment and the acquisition of specific, behaviorally uncommitted knowledge (Squire 82, Kent 81).

Secondary evaluation appears to be a simple but effective means of coping with the sequential credit assignment problem. It does not guarantee formation of optimal action sequences, but does appear to be complete in its ability to assemble arbitrarily long sequences in order to achieve a single goal.

Neural knowledge representation has significant differences from more traditional 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 key features when they exist, but it should be recognized that such an approach is a highly restrictive technique for dealing with natural categories.

Another 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 (Albus 81, Gallistel 80). Hierarchical servomechanisms can be easily built within the Boolean domain (Hampson 83). One interesting characteristic of servomechanisms is that desired (goal) states and the actual, current state are treated as the same type of data. Goals and current conditions can be freely intermixed in a servo/neural behavioral system.

A number of unanticipated biological parallels were observed during development of the model. Since the model was strongly constrained by completeness and efficiency, these parallels suggest that some neural characteristics may be shaped by similar issues. Perhaps the most interesting possibility is the central importance of prototype 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 observed. Finally, the inclusion of learned evaluation proved to be effective in organizing sequences of actions. Such a process has been suggested to be important in biological behavior (Pugh 77). An important difference may be in the explicit inclusion of unlearning. Some biological systems may learn by an irreversible process, which can approach saturation within the life span of the organism (Barnes 79).

7.0 FUTURE WORK

It is hoped that other input domains can be approached as logical extensions of the Boolean domain. 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. In addition, the evaluation system should be extended to support simultaneous operator application.

8.0 CONCLUSIONS

We have developed a multi-layer neural network that appears to be capable of learning arbitrary Boolean functions. Individual neurons are capable of computing the (at least X of N) function, permitting efficient representation in prototypic normal form. Synaptic weights are modified to reflect conditional probability. Multiple operators can share a common memory. The operators and the shared memory learn at the same time, though by different processes. Operators learn through sequential presentation of positive and negative instances. Common memory categorizes each input pattern, and operator feedback causes that categorization to become more specific. Secondary evaluation permits sequences of actions to be learned. These processes have distinct characteristics which appear to be reflected in biological learning.

LITERATURE CITED

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

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

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

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

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)

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)

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)

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

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)

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)

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

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)

Hampson, S. E.: A neural model of adaptive behavior.
Dissertation: in preparation (1983)

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

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. of Psych. 29: 1-29 (1978)

Kandel, E. R.: Small systems of neurons.
Scientific American 241, 66-76 (1979)

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)

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)

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

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

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

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

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

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

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

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

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., 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)

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

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

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

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)

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)

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

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