Functionality in Neural Nets*

L.G. Valiant
Aiken Computation Laboratory
Harvard University
Cambridge, MA 02138

Abstract

We investigate the functional capabilities of sparse networks of computing elements in accumulating knowledge through successive learning experiences. As experiences, we consider various combinations of episodic and concept learning, in supervised or unsupervised mode, of conjunctions and of disjunctions. For these we exhibit algorithms for learning in well defined senses. Each concept or episode is expressible in terms of concepts or episodes already known, and is thus learned hierarchically, without disturbing previous knowledge. Minimal assumptions are made about the computing elements, which are assumed to be classical threshold elements with states. Also we adhere to severe resource constraints. Each new concept or episode requires storage linear in the relevant parameters, and the algorithms take very few steps. We hypothesise that in our context functionality is limited more by the communication bottlenecks in the networks than by the computing capabilities of the elements and hence that this approach may prove useful in understanding biological systems even in the absence of accurate neurophysiological models.

1 Introduction

Knowledge acquisition by learning is a cognitive phenomenon that has proved difficult both to define and to reproduce computationally. The fact that the biological systems that manifest this phenomenon are composed of neurons that are both slow and sparsely connected compounds the mystery. Fortunately these computational constraints are so severe and rule out so many computational mechanisms, that it is quite possible that consideration of them will yield incisive clues into the basic functions underlying learning. In this paper we pursue just this line of enquiry with particular reference to learning discrete knowledge.

The proposed approach is the following. We describe a model of a neuron that is essentially the threshold element of McCulloch and Pitts [1943] but with additional states and adaptive capability. The intention of the model is to have it simple enough that there be little question of it being too powerful for biological plausibility. We then study the basic learning functions that can be implemented on networks of such *neuroids*. Each function implements an *interaction*, the response to a single experience of communication with the outside world. To maintain plausibility we restrict ourselves to interactions that are constrained in three ways: (i) Since biological neurons appear to have response times not much less than 10^{-2} seconds, the basic algorithms have execution times no more than about ten neural updates. (ii) Each algorithm is a sequence of a few steps each of which is asynchronous in that its outcome does not depend on the order of execution of the neuroids. (iii) Since the number of neurons in the human brain is modest, conventionally estimated at about 10^{10}, we restrict ourselves to algorithms that use storage economically.

We conjecture that the learning capabilities under the above resource disciplines of models such as ours upper bound those of corresponding biological systems. This is based on the hypothesis that if substantial long distance associations are to be realised in a sparse network with a very low bit rate in the connections, then the capabilities of the network are governed more by the communication limitations than by the computing power of individual neuroids. (This kind of statement may be amenable to mathematical analysis.)

We consider several modes of learning and show that they can be supported simultaneously by compatible mechanisms. In each case the learning task can be regarded as that of establishing a circuit in the network for computing a Boolean expression. Each mode of learning can be defined formally. The intention behind the definitions is the following. *Episodic* learning concerns the memorization of a single instance of an input. In contrast *concept* learning involves several input instances and aims at deriving a rule that has good inductive behaviour in the sense of being able to classify further unseen inputs reliably (c.f.[Valiant, 1984]). Learning is *supervised* if some classification information about an input is provided by, say, a teacher. Otherwise it is *unsupervised*. A special case of the latter is *correlational* learning which aims at identifying statistically correlated groups of attributes. We shall consider the most basic forms of Boolean expressions, namely simple *conjunctions* and *disjunctions*. It is known that enriching these slightly in certain directions leads to computational intractability [Kearns *et al.*, 1987]. In all cases the attributes can be either propositional, or, in a certain restricted sense, relational.

Also central to our concerns are two aspects of learning that are apparently severely constrained in neural imple-

*Part of this work was done while the author was visiting Oxford University. Support from the SERC and from grants NSF-DCR-86-00379, ONR-N00014-85-K-0445 and DAAL03-86-K-0171 is gratefully acknowledged

mentations but have received little attention. We require the learning mechanisms to be *hierarchical* in that an attribute in an expression to be learned may itself be the value of a previously learned expression. Equally important we insist that learning be *cumulative*. The learning of a new expression should not interfere with existing circuits.

Each learning experience potentially involves all the information in memory, in the sense that a new input may relate to any previously learned knowledge. For this reason we regard learning tasks (as well as memory references) as being among the most resource intensive cognitive tasks , and hence the most appropriate for our approach. In contrast, low level vision, for example, operates on less information, the image on a retina. We therefore envisage the overall learning system as being composed of a separate *neural tabula rasa* (NTR) which is the main instrument of memory and learning, together with several *peripherals* that realise communication with the outside world. The NTR is a mathematically simple network of neuroids or nodes with essentially no pre-programmed knowledge except for some pre-randomization. Some of the nodes can be controlled directly by the peripherals and are thus given real-world meanings by them (e.g. they may fire if the input has certain attributes of colour, shape, etc.)

To show our positive results we use some general mechanisms that support the several learning modes. We assume that each node of an N node network has a little under $\sqrt{N}$ bi-directed connections. Since in real neurons the number of dendrites has been estimated as up to 4×10^4 this degree assumption is not unreasonable for a unit of a 10^{10} node system. We assume connectivity properties that are possessed by most graphs (i.e. random graphs), and also by some well-studied easily constructed families of them. The representation of concepts we use is local (cf. [Feldman, 1982], [Feldman and Ballard, 1982]), essentially corresponding to 'grandmother cells', with redundancy ensured by replication. To implement relations we assume the existence of a *relator peripheral* (RP) which can identify some pre-programmed relations, and distributes in time the flow of this information into the NTR to break symmetries (e.g. distinguish 'x above y 'from 'y above x ').

We emphasize strongly that while we claim that our approach is a valid one for understanding the *functionality* of a biological system being modelled, no claim is being made about the particular algorithms or mechanisms we use. Indeed, once one way of realizing a functionality has been discovered, numerous others may exist also. An immediate question that presents itself is whether the functions realized by our mechanisms can be achieved also if the representation used is distributed or holographic (e.g. [Hopfield, 1982; Ackley *et al.*, 1985; Rumelhart *et al.*, 1986]).

2 The Model and Some Mechanisms

The NTR is modelled as a sparse network of identical neuroids or nodes and can be described formally as a sextuple $(G, W, \sum, I, \lambda, \delta)$. Here G is a directed graph with nodes $V = \{1, 2 \ldots, N\}$ and edges $E \subseteq V \times V$. Each edge $(i, j) \in E$ has *weight* $w_{ij} \in W$, where W is a set of real numbers. A description of the instantaneous condition of neuroid $i \in V$ consists of specifications of the weights of all the edges directed into i (the dendrites) together with the specification of the *state triple* $s \in \sum$. $\sum$ is a subset of $Q \times T \times \{F, \overline{F}\}$ where Q is a set of *states*, T is a set of real number *thresholds*, and the choice of $\{F, \overline{F}\}$ denotes whether the node is *firing*. The *transition function* δ defines how the state triple is updated and the *learning function* λ describes how the weights are updated. Formally, if s_j is the state triple of node j and the value of f_j denotes whether it is firing, then

$s_j := \delta(s_j, \sum\{w_{ij} \mid i \text{ is firing }\})$, and

$w_{ij} := \lambda(s_j, w_{ij}, f_i)$

The intention of the definition is to restrict communication between nodes entirely to firings. A firing of a node can effect directly the weight of any edge incident to it. The firing of neighbours can effect the state triple of a node only via the value of the sum of the weights of the edges coming from them. If T_j is the threshold of j then δ is assumed to be such that j certainly fires if:

$\sum\{w_{ij} \mid i \text{ is firing}\} \geq T_j$.

Nodes may be forced to fire, independently of this, by peripherals.

The initial condition of the network is described by I. We assume that it contains no substantial programmed information, except possibly for some pre-randomization. For example, while we discuss five different kinds of neuroids they can be regarded as all of the same kind, but with random initialization into distinct states that control their subsequent history.

The updating of each neuron is regarded as atomic: for neuron j the value of the state triple s_j and the weights of the incoming edges w_{ij} are all computed instantaneously by the functions δ and λ from the previous values of them as well as from the firing states f_i of the neighbouring nodes at that instant.

The NTR has no global clock. We assume, however, a basic time unit called a *cycle*. Each neuroid updates itself at most once in each cycle. Our algorithms consist of a sequence of *steps*. In each step the peripherals force a set of neuroids to fire simultaneously, and this may cause a cascade of other firings in the NTR. We assume that the algorithms and total state of the NTR are such that all such cascades terminate in a condition that is stable, where no more neuroids can fire, and that stability is reached before a cycle has elapsed. If a node fires in state F it continues to do so until stability is reached. If a node is in the persistent firing state F^* then it will continue to fire until stability is reached in the following step also. In each step we restrict ourselves to algorithms that are *asynchronous*, in the sense that the order in which the neuroids update themselves is immaterial to the outcome.

An *interaction* corresponds to a new input from the outside world. The peripheral processes the input and presents it to the NTR as a sequence of steps in time. For each step enough time is allowed for the spread of activation of the firings to reach stability before the next step. Except for firings, all other state and weight information persists between successive steps and interactions.

There is no notion of node address. All storage allocation and communication is achieved implicitly using certain connectivity properties of the graph. The graph representing the connections consists of N nodes each connected

by edges in both directions to about $\sqrt{N}$ other nodes. The weights of such a pair of bi-directed edges need not be related.

The connectivity property we need is, in its simplest form, the following: For any two nodes i and j there should be a third node adjacent to both of them. This kind of property is used both for allocating previously unused storage as well as for establishing communication between pairs of used function nodes.

Graphs having such degree and such connectivity properties can be constructed, related as they are to finite projective planes [Hall, 1986]. More relevant here is the fact that "most" graphs of this degree approximate these properties. This adds to evolutionary plausibility. We shall take this latter approach here and treat the graphs as random.

In fact we need slight variants of the above property. For fault tolerance we hypothesize that there are about c nodes representing each concept where c is a moderate constant (e.g. $c = 10$). This degree of replication has to be preserved when new nodes are allocated. Degree $\sqrt{(N/c)}$ suffices for this.

The above description of our model can be completed to a complexity-theoretic model parametrized by N, the number of nodes, by imposing appropriate quantitative restrictions on Q, W, δ and λ. Insisting that $|Q|$ be a constant independent of N seems natural. The main question is how to deal with the numerical weights. It appears generous to allow these O(log N) bits in some agreed representation. This would imply that δ and λ have circuit complexity polynomial in N, even if no other restrictions are placed on them. In fact the δ and λ actually used in our algorithms are very restricted. All Boolean tests on numerical values are thresholds and all numerical functions are monotonic and continuous.

3 Results

For each learning task we need to describe an algorithm for achieving it. The algorithms need to be such that they do not interfere with each other, even if, for example, several concepts are being learned simultaneously from examples interleaved in time. Also there need to be mechanisms for supporting hierarchical learning.

In this section we shall outline an implementation informally in as much detail as space allows. For simplicity we omit some simple mechanisms that would make the algorithms more fault tolerant. First we describe the state set Q. Each node starts in a state that pre-destines its purpose. Most basically a node is either a *relay* (R) node or a *function* node. The latter splits into the various categories of supervised learning that are supported, in our case *episodic* (E), *conjunctive* concepts (C), *disjunctive* concepts (D) as well as the *correlational* (L) category. Each non-correlational node is initially *available* (A) but once it has been designated a purpose it becomes *busy* (B). A function node that is available first becomes busy in *unsupervised* (U) mode (i.e. when allocated) and can subsequently remain busy in *supervised* (S) mode, (but transitions in the reverse direction are not allowed). Thus in the notation of regular languages we denote a state by a word from $\{A,B\}R \cup \{A,U,S\}\{E,C,D\} \cup L$. In the descriptions of the algorithms we use further states that exist only during the execution of single interactions, and do not persist after their completion. We append a description with $\bar{F}$ if it is not firing, and F or F^* if it is. By omitting a letter from a state description we denote the set of states in which the conditions denoted by the remaining letters hold. Also we identify a state description at any instant with the set of neuroids having corresponding states. For example SEF signifies the set of all episodic nodes currently in supervised mode and firing.

The algorithms could be expressed formally by specifying the update functions δ and λ. For the sake of clarity we describe each step of each algorithm as a set of conditional rules $\{\ldots\} \Rightarrow \ldots$ that we expect the relevant neurons to be executing at that step. Here $\{\ldots\}$ describes the conditions required for the update described. We note that F and F^* are both firing states indistinguishable by δ and λ. Hence in the conditions we will abbreviate "F or F^*" by "F".

As initial conditions we choose the available relay and function nodes to have threshold 2 and the correlation nodes threshold 3/2. Also any edge directed away from an available relay node has weight 0. The nodes controlled directly by peripherals are initialized as function nodes, and all edges directed away from these and other function nodes have weight 1 initially. The nodes are distributed randomly in appropriate proportions among the functionalities $\{E,C,D,L,R\}$

To describe each algorithm we describe (a) the *triggering set* of nodes that the peripherals cause to fire during the interaction, (b) the *desired consequence* (i.e. the functionality of the new circuit established), (c) the *side-effects* (non-interference with other circuits, and resource bounds), and (d) the *algorithm* itself which acts on each neuroid locally. For brevity we shall not detail (c) here.

3.1 Supervised Episodic Conjunctions

In supervised learning the task is to set up a circuit that makes a chosen target node i fire whenever a certain function of a set J of function nodes is satisfied. Learning is episodic if it uses just one instance or example, and consists of memorizing the conjunction of attributes that are true for that example. The procedure is as follows:

Triggering Set: $J \cup \{i\}$ where $i \in UE$

Desired Consequence: At future interactions $i \in SE$ and whenever $J \subseteq F$, $i \in F$ also.

Algorithm:

Step 1 : Triggered set is $J \cup \{i\}$.
$\{k \in ARF\} \Rightarrow k \in AR_1F^*$.
$\{l \in UEF, k \in F, j \in \bar{F}\} \Rightarrow w_{kl} := 1, w_{jl} := 0,$
$l \in UE_1F^*$.

Step 2 : Triggered set is $\{i\}$
$\{k \in AR_1F, \sum\{w_{jk} \mid j \in F\} \geq 1, j \in F\}$
$\Rightarrow k \in AR_2F^*, w_{jk} := 0.$
$\{l \in UE_1F\} \Rightarrow l \in UE_2F^*$.

Step 3 : Triggered set is $J \cup \{i\}$.
$\{k \in AR_2F, s \in \bar{F}\} \Rightarrow w_{sk} := 0, k \in BR, T_k := 1.$
$\{k \in AR_1\bar{F}\} \Rightarrow k \in AR.$
$\{l \in UE_2F\} \Rightarrow T_l := \sum\{w_{kl} \mid k \in F\}, l \in SE.$

Side effects: The state triples of only $O(|J|)$ nodes are affected. The weights of edges not adjacent to these are not affected.

The idea behind the algorithm is the following. Since initially all available relay nodes (AR) have threshold two, and the edges to them from function nodes have weight one, the relay nodes k that fire in step 1 are those adjacent to at least two of $\{i\} \cup J$. For those of these that are adjacent to i, the weight w_{ki} will be changed from zero to one. Via the use of persistent firings it is ensured that node i will enter step 3 in state UE_2F, where its threshold will be updated to the sum of the weights of the incoming edges from nodes that are firing. Meanwhile in step 2 the triggered set is reduced to the singleton i and among the relay nodes still firing those will be selected to go into state AR_2 that are adjacent to i. (The remainder, those connected only to J, will cease firing in step 2 and revert to state AR in step 3.) In steps 2 and 3 the nodes in state AR_2 will update themselves correctly in order to achieve the desired consequence.

To complete a proof of validity we have to show that in the graph chosen there will be at least one distinct path of length two via a relay node from each $j \in J$ to i.

3.2 Supervised Conjunctive Concepts

Learning concepts inductively takes place over a number of interactions, each one involving the presentation of an example. We implement the simple elimination algorithm for learning conjunctions from positive examples alone, that is shown to have convergent inductive properties in [Valiant, 1984]. The following describes the interaction in which the r^{th} example is presented and has attributes corresponding to the set J_r of nodes.

Triggering Set: $J_r \cup \{i\}$ where $i \in UC$ for $r = 1$ and $i \in SC$ for $r > 1$.

Desired Consequence: After s interactions whenever $\bigcap\{J_r \mid 1 \leq r \leq s\} \subseteq F$ then $i \in F$.

Algorithm: $r = 1$ is treated exactly as episodic conjunctions except that in step 3 i gets state SC.

For $r \geq 2$:

Step: $\{i \in SCF, l \in \bar{F} \Rightarrow w_{li} := 0, T_i := \sum\{w_{ki} \mid k \in F\}$.

In other words the first example is treated as an episodic conjunction. Any attribute in it that fails to appear in a subsequent positive example is eliminated.

3.3 Supervised Disjunctive Concepts

Disjunctions cannot be learned from positive examples alone (see [Kearns *et al.*, 1987]) but in principle they can be learned from negative examples alone [Valiant, 1985]. In the current context we shall indicate the positivity of the example being presented in an interaction by causing the target node i to fire. Negativity is indicated by the absence of firing. This formulation has the advantage that learning from negative examples can take place as background activity for numerous concepts simultaneously. To keep a lid on the resources needed, however, it is advantageous to have positive examples also.

The simple solution we describe has to see the positive examples first in order to obtain the universe of possible disjuncts. The algorithm first forms the disjunction of all the attributes occurring in them. It then goes on to see negative examples. By a simple elimination algorithm, varieties of which are analysed in [Valiant, 1985], it deletes all attributes that occur in any negative example.

To construct the initial disjunction we use a variation on the algorithm for episodic conjunctions. This constructs essentially the same circuit as there except that the threshold of node i is unity and its state is SD. In the second phase when the r^{th} negative example is seen (i.e. nodes J_r fire but $i \in SD$ does not), the elimination rule is

$$\{l \in SD\bar{F}, k \in F\} \Rightarrow w_{ki} := 0$$

The algorithm can be adapted so that if it sees a positive example following some negative ones and the circuit fails to recognise it as positive then it appends the new attributes to the disjunction.

3.4 Perceptron Algorithms

Learning algorithms for linear discriminant functions that are in the classical perceptron style [Rosenblatt, 1958], [Minsky and Papert, 1969], [Littlestone, 1987] can be implemented in the same manner as the above elimination algorithm for disjunctions. In neuroidal implementations it is important, however, that the algorithms be *self-checking*. In other words if the examples of the concept are inconsistent with the assumed concept class then the algorithm should discover this. Elimination algorithms do this simply by eliminating all possibilities and hence producing null circuits. Perceptron algorithms of the classical variety appear to need additional mechanisms. One possibility is to do mistake counting [Littlestone, 1987].

3.5 Unsupervised Episodic Learning

In unsupervised learning there is no indication given to the learning system as to the label to be attached to a new item learned. In our framework we identify unsupervised learning with tasks where a new available function node has to be allocated to realise the output of the newly created circuit. We consider *episodic* and *correlational* learning in unsupervised mode. The former corresponds to memorizing a new combination of previously learned attributes, given one instance of it (e.g. memorizing a new word). The latter is concerned with spotting combinations of attributes that occur in statistical correlation with each other. Episodic unsupervised learning can be used to allocate storage in preparation for supervised learning. Hence the progression $A \rightarrow U \rightarrow S$ in the states. Correlational learning can be used to learn conjunctions that are to be the constituent monomials for subsequent supervised learning of disjunctions (so as to realise limited learning of disjunctive normal form). We shall consider only the case of learning conjunctions of length two, since longer ones can be made up from a sequence of these, time-stepped by the peripherals.

The allocation of new storage is effected by similar principles to the ones used for finding relay nodes in supervised learning. For any pair of function nodes i, j let A_i and A_j be the sets of about c nodes that are functionally equivalent to i and j respectively. We need that for all i, j there exist about c nodes that are connected to both something in A_i and something in A_j. For this a random graph (now restricted just to the function nodes) of degree about $\sqrt{N/c}$ suffices. The basic algorithm is as follows:

Triggering Set: $\{i, j\}$

Desired Consequence: For some previously available episodic function node k to construct circuit such that subsequently $i, j \in F \Rightarrow k \in F$.

Algorithm:
Triggered Set: $\{i, j\}$
Step: $\{k \in AEF,\ l \in \bar{F}\} \Rightarrow k \in UE, w_{lk} := 0$.

The idea of the algorithm is that since the threshold of k is two, it will be fired only if it is connected to both i and j.

Mechanisms are also needed for arresting unintended cascades of memory allocations. It is necessary and possible, for example, to ensure that a newly allocated neuroid does not immediately cause further allocations, unless explicitly requested by the peripherals.

3.6 Correlational Learning

Next we describe how correlations are detected. In general the question of detecting correlated pairs is computationally problematic even for sequential models. This can be seen by considering what we have called the light-bulb problem: There are n synchronized sources (bulbs). Each one is on (off) during each time interval with probability p ($(1-p)$ respectively) independent of previous intervals. Also one pair of bulbs is correlated (e.g. the probability that they are both on in any one interval is $q \gg p^2$) while all other pairs are pairwise independent. The problem is to detect the correlated pair. If $p = 1/2$ then after $0(logn)$ intervals, the n sequences of bits for the n sources will contain sufficient information, the pair having minimal Hamming distance corresponding to the desired pair. The computational problem is to discover this pair efficiently. The first sequential algorithm achieving this in less than n^2 steps is due to Paturi [1988], and requires n^α steps where α is a constant ($1 < \alpha < 2$) depending on the correlation. It remains an open problem whether this bound can be improved to near linear.

For a plausible neural implementation we need more stringent requirements. Perhaps surprisingly, these can be satisfied in the case when p is small ($\sim 1/N$) which is just the case of most practical interest. For in any interaction we expect just a minute fraction of the episodes or concepts in memory to be relevant.

The basic algorithm for learning pairwise correlations is taken from [Valiant, 1985] and is of the Hebb variety [Hebb, 1949]. The rules that update the correlation neuroids are the following

$\{k \in LF, i \in F\} \Rightarrow w_{ik} := \sigma(w_{ik})$
$\{k \in LF, i \in \bar{F}\} \Rightarrow w_{ik} := \sigma^{-1}(w_{ik})$

Here σ is an increasing function such that $\sigma^r(1) \to 5/4$ and $\sigma^{-r}(1) \to 0$ as $r \to \infty$, both processes taking place in suitably small steps. The idea of the algorithm is that $k \in L$ will fire only if at least two of its inputs fire. Only those edges w_{ik} will be reinforced often enough that do correspond to elements of correlated pairs. Network connectivity properties are used to ensure that for any correlated pair, there will be a correlation neuroid adjacent to both.

The learning of large numbers of correlated pairs can be shown to be supportable as a background activitiy simultaneously with other processes.

3.7 Relators

We think of the firing of a function node as indicating, in the first instance, the truth of a proposition regarding the input examined by the peripherals. In order to be able to express relational information about the input we will need to designate some of the function nodes controlled by the peripherals as indicative of relations. We shall use a restricted notion of a relation that we call a *relator*. If x, y are propositional variables and R a relator then the statement xRy denotes that in the input examined by the peripherals there are objects X, Y satisfying x and y respectively that are in relationship R.

We assume that there is a *relator peripheral* (RP) containing a fixed set of pre-programmed relators that it is capable of interpreting. For each binary relation R it controls two nodes (or sets of nodes) in NTR called $R^{(1)}$ and $R^{(2)}$ to correspond to the two arguments. If a conjunction of relations $(x_1 R_1 y_1)(x_2 R_2 y_2) \ldots (x_r R y_r)$ is detected by the RP in any mode of learning or recognition, the RP breaks this into $2r$ steps for presentation to the NTR. For $1 \le s \le r$ at step $2s-1$ x_s and $R_s^{(1)}$ fire, while at step $2s$ y_s and $R_s^{(2)}$ fire. Thus timing is used to resolve the symmetry between each pair (x_s, y_s), and also the symmetries among the relations.

Learning Boolean expressions that contain relators can be done in the following way. First each pair $(x, R^{(i)})$ is learned in unsupervised episodic mode at a different step. Finally the UE nodes so created are used as if they were propositional, to learn the expression in some standard way.

While relators cannot express identity among objects explicitly, this can be simulated using certain special additional relators or propositional predicates. For example the unary uniqueness relator R, where Rx denotes that there is only one object satisfying x, is clearly useful. In a sense we can simulate arbitrary equivalence classes among the objects if we hypothesize predicates z_i where z_i denotes "this is the i^{th} object in the scene" for some ordering imposed by the peripherals. In that case whenever RP fires x_s and $R_s^{(1)}$ (or $R_s^{(2)}$ and y_s), it also fires the z_i corresponding to the object concerned. The triggering sets then become triples.

3.8 Learning Hierarchically

In a typical learning situation we have a triggering set $J \cup \{i\}$ where i is the intended output neuroid of a circuit to be constructed. If learning is hierarchical then the nodes in the triggering sets may be higher level concepts not controlled directly by the peripherals. To fire them a cascade of firings of neuroids corresponding to lower level concepts would have to be initiated. The unintended firings may, however, cause unwanted interference. This can be avoided by introducing a mechanism that detects the highest level firing in any such cascade.

4 Conclusion

It seems unreasonable to ascribe a complex function to a computational system unless one has a plausible candidate for the computational mechanism that might be supporting the function. In this paper we have described a neural model of computation and demonstrated some mechanisms for realising some very basic functions that appear to be relevant to cognition. A more complete exposition of these results will appear in a forthcoming paper [Valiant, 1988].

It would be desirable clearly to extend the results to other functions, such as richer learnable classes [Blumer et al., 1986; Haussler, 1987; Kearns et. al., 1987]. Also there seems to be substantial scope for improving the quality or robustness of the algorithms given. One issue is resilience to errors in the data. Also there are numerous other questions, which we have not addressed, concerning the behaviour of the system during long sequences of interactions.

References

[Ackley *et al.*, 1985] D. H. Ackley, G. E. Hinton and T. J. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9 (1985) 147.

[Blumer *et al.*, 1986] A. Blumer, A. Ehrenfeucht, D. Haussler and M. Warmuth. Classifying learnable geometric concepts with the Vapnik-Chervonenkis dimension. *Proc. 18th ACM Symp. on Theory of Computing*, (1986) 273-282.

[Feldman and Ballard, 1982] J. A. Feldman and D. H. Ballard. Connectionist models and their properties. *Cognitive Science*, 6 (1982) 205-254.

[Feldman, 1982] J.A. Feldman Dynamic connections in neural networks. *Biol. Cybern.* 46 (1982) 27-39.

[Hall, 1986] M. Hall, Jr. *Combinatorial Theory.* Wiley, New York, (2nd edition) 1986.

[Haussler, 1987] D. Haussler Learning Conjunctive Concepts in Structural Domains. Proc. AAAI (1987), Morgan Kaufmann, Los Altos, C.A. 466-470.

[Hebb, 1949] D. O. Hebb. *The Organisation of Behaviour.* Wiley, New York, 1949.

[Hopfield, 1982] J. J. Hopfield. Neural networks and physical systems with emergent computational abilities. *Proc. Nat. Acad. Science*, 79 (1982) 2554.

[Kearns *et al.*, 1987] M. Kearns, M. Li, L. Pitt and L. G. Valiant. Recent results on Boolean concept learning. *Proc. 4th Int. Workshop on Machine Learning*, Morgan Kaufmann, Los Altos, CA (1987) 337-352.

[Littlestone, 1987] N. Littlestone. Learning quickly when irrelevant attributes abound. In *Proc. 28th IEEE Symp. on Foundations of Computer Science*, (1987) 68-77.

[McCulloch and Pitts, 1943] W. S. McCulloch and W. H. Pitts. A logical calculus of ideas imminent in nervous activity. *Bull. of Math. Biophysics*, 5 (1943) 115.

[Minsky and Papert, 1969] M. Minsky and S. Papert. *Perceptrons.* MIT Press, Cambridge, MA (1969).

[Paturi, 1988] R. Paturi. The light bulb problem. Technical Report CS88-129, UC San Diego, 1988.

[Rosenblatt, 1958] F. Rosenblatt. The perceptron, a probabilistic model of information storage and organization in the brain. *Psychological Review*, 62 (1958) 386.

[Rumelhart *et al.*, 1986] D. E. Rumelhart, G. E. Hinton and R. J. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing*, Vol 1, (eds D. E. Rumelhart and J. L. McClelland), MIT Press, Cambridge (1986).

[Valiant, 1984] L. G. Valiant. A theory of the learnable. *CACM*, 27 (1984), 1134-1142.

[Valiant, 1985] L. G. Valiant. Learning disjunctions of conjunctions. *Proc. of 9th Int. Joint Conf. on Artificial Intelligence*, (ed A. Joshi), Morgan Kaufmann, Los Altos, CA (1985), 560-566.

[Valiant, 1988] L. G. Valiant. Functional capabilities of neural nets. To appear.