Masters Thesis

Switching in Boolean Circuits and Modelling Cognition through Neuroids

Author:
Pritam Bhattacharya

Supervisor:
Dr. Manoj Gopalkrishnan

A thesis submitted in fulfilment of the requirements for the degree of Master of Science

in the

School of Technology and Computer Science

Tata Institute of Fundamental Research

January 2014

Abstract

In computational complexity theory, Håstad's switching lemma is a vital analytical tool for proving lower bounds on the size of constant-depth Boolean circuits. Using the switching lemma, Håstad showed in 1986 that Boolean circuits of depth k, in which only AND, OR, and NOT gates are allowed, require size $\exp\left(\Omega\left(n^{\frac{1}{k-1}}\right)\right)$ for computing the PARITY function on n variables. In essence, the switching lemma says that, given an arbitrary formula in disjunctive normal form, if we set some fraction of the variables randomly, then with high probability, the restricted function can be computed by a decision tree of small depth.

In 2012, Impagliazzo, Matthews and Paturi formulated an extended switching lemma which says that, given a sequence of formulas in conjunctive and/or disjunctive normal form on the same set of variables, if all of them are hit by the same random restriction, then it is exponentially unlikely that there is a large subset of formulas where each formula contributes a large number of variables to their joint decision tree. In the first chapter of this thesis, we begin by discussing Håstad's switching lemma, and how it is useful in proving that PARITY does not belong to AC^0. We then move on to discussing the extended switching lemma and its proof.

In the second chapter, we critique the neuroidal model for cognition proposed by Valiant. We start off by discussing the motivation behind Valiant's work. We then explore the physiology of the brain and some insights from cognitive psychology that aided Valiant's formulation of the neuroidal model. Next, we describe the actual model in extensive detail and present the algorithm put forward by Valiant for implementing unsupervised memorization within his model as a case study. Finally, we conclude by discussing the relevance of the neuroidal model to the current attempts by researchers to build cognitive computing systems.

Contributions

The main contributions of this thesis are as follows:

1. In section 1.4.3, we break down the proof of Håstad's switching lemma due to Razborov and present it in a manner that we hope will aid comprehension, without sacrificing mathematical rigour. We provide the big picture first, by stating the general strategy used, and then zoom in to explicitly describe the encoding and decoding procedures that lie at the heart of the proof.

2. In section 1.5.3, we devise an illustrative example to help the reader understand better the encoding and decoding algorithms appearing in the proof of the extended switching lemma introduced by Impagliazzo, Matthews and Paturi[20]. We do so in the hope of making the original proof much more accessible to a reader not having expertise in the field.

3. Chapter 2 is concerned with the neuroidal model proposed by Valiant in his book "Circuits of the Mind". In section 2.1.8, we devise a use-case to illustrate how a simple strategy suggested by Valiant captures the cognitive ability of learning to recognize newly encountered items. We revisit this strategy in section 2.4.6 and provide the more formal algorithmic description given by Valiant.

4. In section 2.4.8, we collect and present the discussions by Valiant regarding the flexibility and redundancy inherent to the neuroidal model, which provide evidence of its considerable robustness.

5. In section 2.5, we pinpoint the characteristics of the knowledge representation scheme used by Valiant and discuss the assumptions that he makes regarding the topology of the underlying network in his model.

6. In section 2.6, we present the algorithmic realization of unsupervised memorization as a case study, in the hope that it will be instructive in building intuition and conveying the essence of similar algorithms implementing various other cognitive tasks within the model.

7. In section 2.7, we explore the relevance of Valiant's model to the development of computing architectures that emulate the brain.

Contents

List of Figures

Chapter 1

Switching in Boolean Circuits

1.1 Introduction

In theoretical computer science, circuit complexity is a branch of computational complexity theory in which Boolean functions are classified according to the size or depth of Boolean circuits that compute them.

Circuit complexity goes as far back as 1949, when Claude Shannon proved that almost all Boolean functions on n variables require circuits of size $\Theta(2^n/n)$. Despite this fact, complexity theorists have not been able to prove satisfactory circuit lower bounds for specific Boolean functions. On the other hand, superpolynomial lower bounds have been proved, although under certain restrictions, on the family of circuits used.

The first function for which superpolynomial circuit lower bounds were shown was the PARITY function, which computes the sum of its input bits modulo 2. In 1984, Furst, Saxe and Sipser first proved the fact that PARITY is not contained in AC^0. Later improvements by Håstad (1986) in fact established that any family of constant-depth circuits computing the PARITY function requires exponential size. It was in order to establish this fact that Håstad formulated his famous switching lemma.

Recently, in 2012, Impagliazzo, Matthews and Paturi formulated exponential time algorithms with improved savings for AC^0 circuit satisfiability and for counting solutions. In addition, they also provided an improved bound on the correlation of AC^0 circuits with PARITY. As an important component of their analysis, they extended Håstad's switching lemma to handle multiple k-CNFs and k-DNFs.

1.2 Boolean Functions and Boolean Circuits

1.2.1 Boolean Functions and their Families

Definition 1.1. A Boolean function is a function of the form $f : \{0,1\}^n \to \{0,1\}$, where n is a non-negative integer called the arity of the function.

There are 2^{2^n} n-ary Boolean functions for every n. Every n-ary Boolean function can be expressed as a propositional formula in n variables x_1,x_2,...,x_n. Any two propositional formulas are logically equivalent if and only if they express the same Boolean function. Alternatively, any Boolean function can be represented as a multivariate polynomial over the finite field $\mathbb{F}_2$, a binary decision diagram (BDD), a negation normal form (NNF), or a propositional directed acyclic graph (PDAG).

Typically, in circuit complexity, we speak of not just Boolean functions, but rather families that consist of Boolean functions for each different input length. Examples include families like PARITY^n, OR^n and AND^n, where the superscript n refers to the arity of the function.

1.2.2 Conjunctive and Disjunctive Normal Forms

Definition 1.2. A conjunctive normal form (CNF) is an AND (conjunction) of clauses, where each clause is an OR (disjunction) of literals.
Eg. - $(x_1 \vee x_3 \vee \neg x_6) \wedge (x_2 \vee \neg x_7) \wedge (\neg x_4 \vee x_5 \vee x_6)$

Definition 1.3. A disjunctive normal form (DNF) is an OR (disjunction) of terms, where each term is an AND (conjunction) of literals.
Eg. - $(x_1 \wedge x_2 \wedge \neg x_4) \vee (x_3 \wedge \vee \neg x_6) \vee (\neg x_1 \wedge x_5 \wedge x_6)$

A DNF (or CNF) is a syntactic object, but we also think of it as computing a function $f : \{0,1\}^n \to \{0,1\}$ in the obvious way. The size of a DNF (or CNF) is the number of terms (or clauses) it has, while the width of a DNF (or CNF) is the maximum number of literals in a term (or clause).

1.2.3 Boolean Circuits and Computation

Definition 1.4. A Boolean circuit C over a collection of Boolean functions Ψ is a directed acyclic graph with the following properties:

(i) There is a set of special nodes with in-degree 0, called the input nodes.

(ii) There is a special node with out-degree 0, called the output node.

(iii) Every non-input node v is labelled by a function $f_v \in \Psi$.

A Boolean circuit C computes a Boolean function f as follows –
Consider any non-input node v in the circuit C, and let the directed edges (u_1, v),(u_2, v), $\ldots$,(u_r, v) form the set of all incoming edges to v. Then, the vertex v computes the function C_v, defined recursively as:

$$C_v(x) = f_v(C_{u_1}(x), C_{u_2}(x), \ldots, C_{u_r}(x))$$

Moreover, for each input node u in C, we have $C_u(x) = x_i$, where x_i is one of the bits in the input x to the function f.

Let z denote the output node of a circuit C. Then, C is said to compute the n-ary Boolean function f iff:

$$\forall x \in \{0,1\}^n : \; C_z(x) = f(x)$$

Figure 1.1 below shows a Boolean circuit computing the XOR function on two input variables x_1 and x_2.

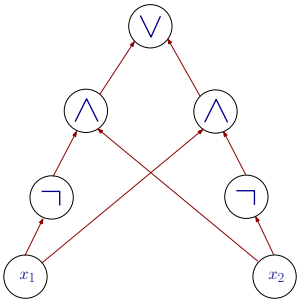

FIGURE 1.1: Boolean circuit computing $x_1 \bigoplus x_2$

1.2.4 Uniformity

Boolean circuits are a prime example of so-called *non-uniform* models of computation in the sense that inputs of different lengths are processed by different circuits, in contrast with *uniform* models such as Turing machines where the same computational device is used for all possible input lengths. An individual computational problem is thus associated with a particular family of Boolean circuits $C_1, C_2, \ldots$, where each C_n is the circuit handling inputs having n bits. A uniformity condition is often imposed on these families, requiring the existence of some resource-bounded Turing machine which, on input n, produces a description of the individual circuit C_n.

Definition 1.5. A family of Boolean circuits $\{C_n : n \in \mathbb{N}\}$ is said to be *polynomial-time uniform* if there exists a deterministic Turing machine M, such that:

- M runs in polynomial time
- $\forall n \in \mathbb{N}$, M outputs a description of C_n on input 1^n

Definition 1.6. A family of Boolean circuits $\{C_n : n \in \mathbb{N}\}$ is said to be *logspace uniform* if there exists a deterministic Turing machine M, such that:

- M runs in logarithmic space
- $\forall n \in \mathbb{N}$, M outputs a description of C_n on input 1^n

1.2.5 Complexity Measures for Boolean Circuits

We define two complexity measures for Boolean circuits as follows-

Definition 1.7. The size of a circuit C, denoted by $SIZE(C)$, is the number of non-input nodes in C.

Definition 1.8. The depth of a circuit C, denoted by $DEPTH(C)$, is the length of the longest path from any input node to the output node in C.

We say that a circuit C computes an n-ary Boolean function f efficiently if $SIZE(C) = O(n^k)$, for some constant k. We say that the circuit C exhibits a high degree of parallelism if $DEPTH(C) = (\log n)^{O(1)}$.

1.2.6 Complexity Classes for Boolean Circuits

Definition 1.9. NC^i is the class of decision problems decidable by uniform boolean circuits with a polynomial number of gates having at most two inputs and depth $O(\log^i n)$, or the class of decision problems solvable in time $O(\log^i n)$ on a parallel computer with a polynomial number of processors.

Definition 1.10. The total hierarchy of NC classes is defined as:

$$\mathrm{NC} = \bigcup_{i \geq 0} \mathrm{NC}^i$$

Clearly, we have $\mathrm{NC}^1 \subseteq \mathrm{NC}^2 \subseteq \cdots \subseteq \mathrm{NC}^i \subseteq \cdots \subseteq \mathrm{NC}$.

Definition 1.11. For each i, the complexity class AC^i consists of the languages recognized by Boolean circuits with depth $O(\log^i n)$ and a polynomial number of AND and OR gates with unlimited fan-in.

Definition 1.12. The total hierarchy of AC classes is defined as:

$$\mathrm{AC} = \bigcup_{i \geq 0} \mathrm{AC}^i$$

The AC classes are related to the NC classes as follows:

$$\forall i \in \mathbb{N} : \ \mathrm{NC}^i \subseteq \mathrm{AC}^i \subseteq \mathrm{NC}^{i+1}$$

As an immediate consequence of this, we have that NC = AC. Moreover, it is known that both inclusions are strict for $i = 0$.

1.2.7 Random Restrictions

Definition 1.13. For some Boolean function f over n variables $x_1, x_2, \ldots, x_n$, a restriction (or partial assignment) α is defined to be the fixing of some of the n variables to 0 or 1, while leaving the remaining variables free. More formally, a restriction is a function $\rho : \{x_1, x_2, \ldots, x_n\} \to \{0, 1, *\}$.

Usually, we say that the free variables have been set to 'star'($*$).The set of variables which a restriction α leaves free is usually denoted by stars(α). We also denote by $f|_\alpha$ the restricted function from $\{0,1\}^{|\mathrm{stars}(\alpha)|} \to \{0,1\}$. We write $\mathcal{R}_s$ to represent the set of all restrictions α with exactly s stars. Since the construction of any unique restriction $\alpha \in \mathcal{R}_s$ involves choosing the variables to be included in stars(α) and also setting each of the other $(n - s)$ variables to either 0 or 1, we have $|\mathcal{R}_s| = \binom{n}{s} 2^{n-s}$.

Definition 1.14. A random restriction with s stars is defined to be any restriction α chosen uniformly at random from $\mathcal{R}_s$. Equivalently, a random restriction can be said to be any restriction $\alpha \in \mathcal{R}_s$, where the set stars(α) is chosen uniformly at random from the $\binom{n}{s}$ possibilities, and the unstarred variables $[n] \backslash \mathrm{stars}(\alpha)$ are also fixed uniformly at random from the 2^{n-s} possibilities.

The switching lemma is primarily concerned with how applying a random restriction simplifies a DNF f.

1.3 Decision Trees and their Construction

1.3.1 Decision Trees as a Model of Computation

Definition 1.15. A decision tree is a rooted binary tree with each internal node labelled with a variable and each of its leaves labelled with an element from the set $\{0,1\}^n$.

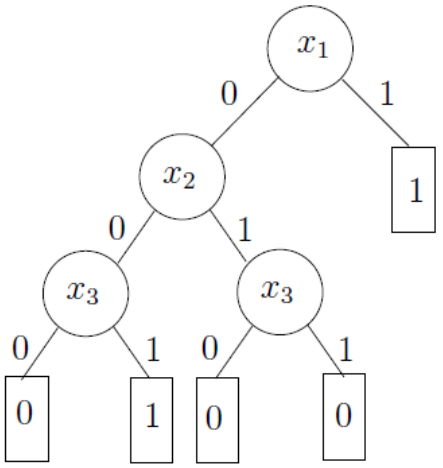

FIGURE 1.2: Decision tree computing $f(x_1, x_2, x_3) = (x_1 \vee \neg x_2 \wedge x_3)$

A decision tree can be seen as a model of computation. In particular, a decision tree which has each of its leaves labelled by a single bit can compute a Boolean function $f : \{0,1\}^n \to \{0,1\}$ in the following manner –

- On any input $x \in \{0,1\}^n$, start with the root node v.
- Query the variable representing v, say x_i, to check whether it is 0 or 1.
- If it is 0, then go to its left child.
- Otherwise, if it is 1, then go to its right child.
- Repeatedly query variables in this manner until a leaf node is reached.
- Then, the label on the leaf gives us the value of $f(x)$.

In general, a decision tree can also simultaneously compute several Boolean functions on the same set of variables. In that case, its leaves are labelled by as many bits as the number of functions it computes. In section 1.3.5, we will be seeing examples of such decision trees.

1.3.2 Complexity Measures for Decision Trees

Just as with circuits, we typically define two measures for describing the complexity of a decision tree as follows –

Definition 1.16. The size of a decision tree D, usually denoted by $SIZE(D)$, is defined to be the total number of nodes (or variables) present in the tree.

Definition 1.17. The depth of a decision tree D, usually denoted by $DEPTH(D)$, is defined to be the length of the longest path (in terms of the number of nodes/variables present on the path) from its root to a leaf.

Given a Boolean function f, we write $DT_{depth}(f)$ to denote the least depth of a decision tree that computes f.

Definition 1.18. A function f that requires every decision tree computing it to have depth n is called an evasive function. Eg. - AND, OR, PARITY etc

1.3.3 Depth of Decision Trees Computing PARITY^n

Definition 1.19. PARITY^n is defined to be an n-ary Boolean function $f : \{0,1\}^n \to \{0,1\}$ such that:

$$\forall x \in \{0,1\}^n : \qquad f(x) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

PARITY^n depends only on the number of 1s in the input, and is therefore a symmetric Boolean function. Not only is PARITY^n an evasive function, we can actually prove something stronger about decision trees computing PARITY^n.

Lemma 1.20. *In a decision tree computing PARITY^n, every path from the root to a leaf must have length n.*

Proof. Suppose, in a decision tree D computing PARITY^n, there exists a path P from the root to a leaf having length less than n. This implies that there exists a variable x_i not being queried at any point in the path P. Now, let us consider a pair of inputs $y, z \in \{0,1\}^n$ such that they differ only in the bit x_i and all the variables appearing in the path P are set according to the path P. Therefore, both the inputs y and z will be evaluated along the same path P, and hence the decision tree D will output the same value for both. However, from the definition of PARITY^n, we know that PARITY$^n(y)$ and PARITY$^n(z)$ cannot have the same value, since the number of 1s in the inputs y and z differ by exactly 1. Thus, the decision tree D can not be correctly computing PARITY^n, which contradicts our assumption. Hence proved. □

1.3.4 Canonical Decision Trees for a DNF

Note that, once a decision tree for a function f is presented to us, one can construct a DNF corresponding to the function by looking at all the paths that end in a leaf with label 1, writing down each of these paths as conjunctions of literals to form clauses, and then taking an OR of all the clauses to obtain a DNF. Since there is a unique path between any two vertices in a tree, and in particular between the root and any leaf, the resultant formula will be unique. A unique CNF can also be obtained in a similar way.

However, given a DNF corresponding to a function f, it may not be possible to obtain a unique decision tree. For example, we can have two different decision trees computing the DNF $(x_1 \wedge x_2) \vee (\neg x_1 \wedge x_2)$, one with x_1 appearing as the root and the other with x_2 appearing as the root. Hence, it becomes convenient to define a canonical representation of decision trees computing f. The idea is to fix an ordering on the clauses as well as an ordering on the variables, which in turn enables us to induce a fixed ordering on the literals appearing in each clause. This forms a total order which enforces the uniqueness of the canonical representation.

The canonical decision tree of a boolean function $f : \{0,1\}^n \to \{0,1\}$ can be constructed recursively as follows –

- Let $c_1 \vee c_2 \vee \cdots \vee c_m$ be the DNF representation of a function f, having width r.
- Let $c_i = x_{i_1} \wedge x_{i_2} \wedge \cdots \wedge x_{i_k}$ be the first non-trivial clause, where $k \leq r$.
- Let us assume that the canonical ordering induces the ordering $i_1 < i_2 < \cdots < i_k$ on the literals in the clause c_i.
- Construct a full binary decision tree with all nodes at level j getting the label x_{i_j}, for $1 \leq j \leq k$.
- Note that, exactly one assignment to $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$ would have caused c_i to evaluate to 1. In that case, simply attach a leaf with label 1 at the end of the path corresponding to this particular assignment.
- For all other assignments ρ to $x_{i_1}, x_{i_2}, \ldots, x_{i_k}$, we recursively construct the joint decision tree of the restricted function $f|_\rho$ and attach it to the end of the path corresponding to the assignment ρ.
- Finally, if the restricted function $f|_\rho$ has become a constant, then just attach the appropriate leaf node to the end of the path corresponding to the assignment ρ.

Since each clause can have at most r literals, and since there are m clauses, the depth of the canonical decision tree can be at most rm.

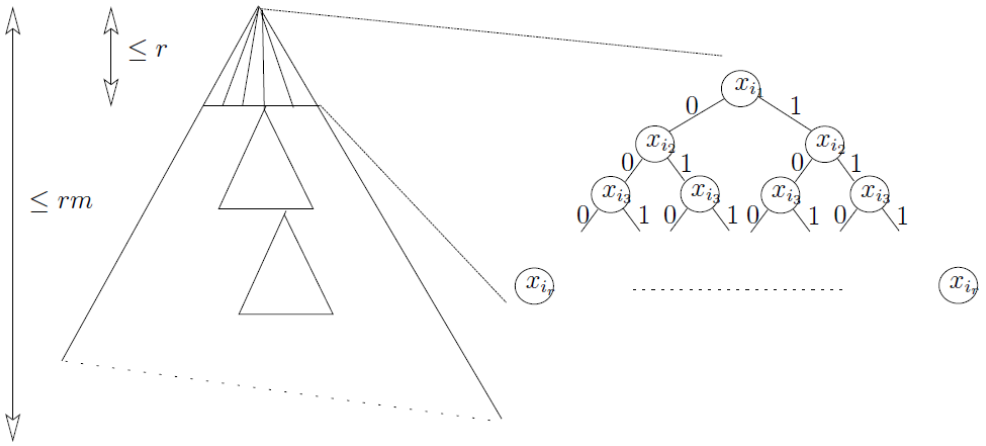

FIGURE 1.3: Construction of a canonical decision tree for a DNF

1.3.5 Canonical Decision Trees for a Sequence of CNFs or DNFs

We can also construct the canonical decision tree for a sequence of CNFs and/or DNFs $\Phi = (\phi_1, \phi_2, \ldots, \phi_l)$ recursively as follows –

- First construct the canonical decision tree D_1 for ϕ_1.
- Along each path in D_1, restrict $\phi_2, \ldots, \phi_l$ by the assignment ρ that corresponds to the path and recurse.
- Finally, label the leaves of the resulting tree with the l-tuples of leaf labels from the original trees.

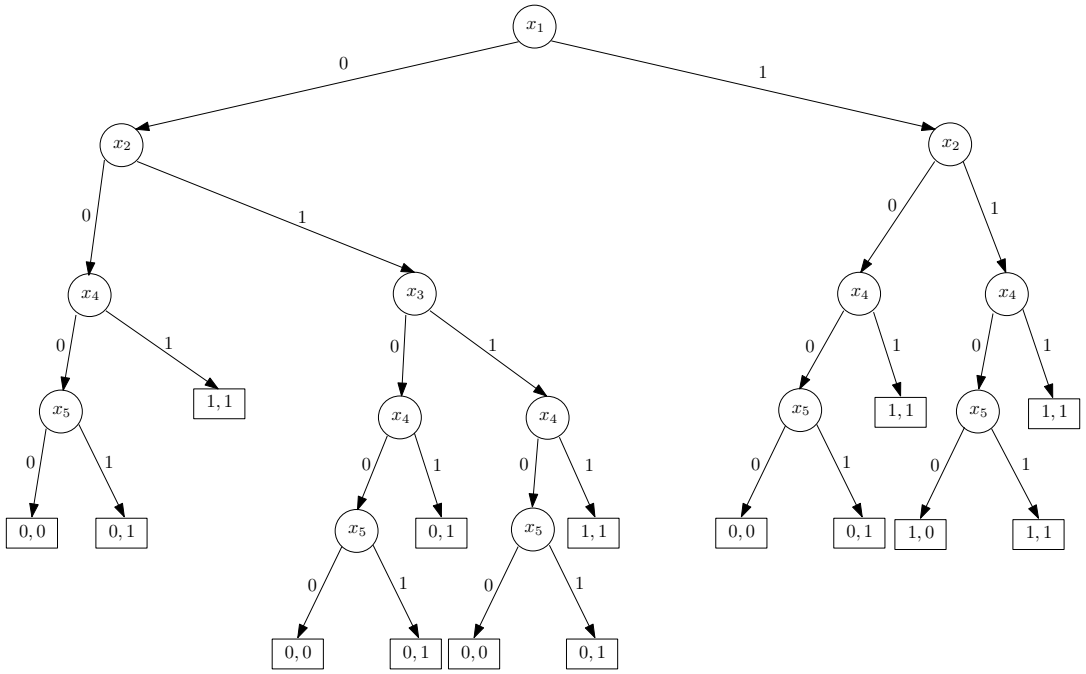

FIGURE 1.4: Canonical decision tree on a sequence of CNFs

Figure 1.4 shows the canonical decision tree for the sequence of CNFs $\Phi = (\phi_1, \phi_2)$, where $\phi_1 = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee x_4)$ and $\phi_2 = (x_4 \vee x_5)$, with the variables queried in order of their indices. When we construct the canonical decision tree for a CNF/DNF ϕ, we group variables by the clause we are at when we query them, and say the clause contributes the corresponding set to the path. Similarly, for any subcircuit ψ of ϕ, we regard the set of nodes contributed by all the occurrences of variables in ψ as the set of nodes contributed by ψ.

1.4 Håstad's switching lemma

1.4.1 Background

The fact that PARITY $\notin \mathrm{AC}^0$ was first established by Furst, Saxe and Sipser[12] (and independently by Ajtai[1] in a different guise) when they proved an exponential lower bound on the size of depth-d circuits for PARITY. For the simple case of depth-2 circuits for PARITY, it is quite straightforward to verify this lower bound.

Claim 1.21. *If a CNF (or a DNF) computes PARITY^n, then:*

i) Each term in the CNF (or DNF) includes all n variables.

ii) There are at least 2^{n-1} terms in the CNF (or DNF).

Proof. Let C be some CNF circuit computing PARITY on n variables $x_1, x_2, \ldots, x_n$. Suppose, by way of contradiction, that C has some term T which does not depend on some variable x_i. Then, when all the inputs to T are 0, the OR gate representing T outputs 0, and so the single AND gate on the next level also outputs 0, which becomes the output of the whole circuit. Now, if we flip the value of x_i, the output of T still remains 0, and thus the output of C does not change. But since we've flipped only one variable, the PARITY has to flip. However, this leads to a clear contradiction since we've assumed that C computes PARITY. Hence, we have proved that every term must depend on all variables.

To compute PARITY, C must output 0 on 2^{n-1} different settings of the input variables. However, C can output 0 only when one of the OR gates outputs 0. But then, each OR gate, since it depends on all variables, can output 0 on exactly one unique setting of the input variables. Hence, there has to be at least 2^{n-1} terms in C. □

However, for circuits computing PARITY having depth 3 and above, the analytical tool that needs to be invoked is Håstad's switching lemma.

1.4.2 Statement

The switching lemma tells us that, on applying a random restriction to a CNF/DNF, we end up with a Boolean function that depends only on few variables, i.e. it can be computed by a decision tree of some small depth d. This allows us to write the restricted function as a small formula in disjunctive normal form. A formula in conjunctive normal form hit by a random restriction of the variables can therefore be 'switched' to a small formula in disjunctive normal form. The formal statement of Håstad's switching lemma is as follows –

Theorem 1.22 (Håstad's Switching Lemma)**.** *Let f be a DNF (or CNF) of width at most w over n variables. Let α be a random restriction with $s = pn$ stars, where $p \leq \frac{1}{5}$. Then, for each $d \geq 0$ (and $d \leq s$), we have:*

$$\Pr[DT_{depth}(f|_\alpha) > d] \leq (10pw)^d$$

1.4.3 Proof

The original proof of Håstad's switching lemma (1986)[15] involves an argument with conditional probabilities. It uses the probabilistic method to argue that the probability that a restriction from the family fails to have the desired properties is strictly less than 1. In the argument one considers, for example, an OR of small ANDs (DNF formula with short terms) and considers each term in turn. The basic idea is that a term that is falsified by a restriction does not contribute any variables to the AND of small ORs and for each term that is not falsified it is more likely that the term is satisfied (and thus the whole formula is fixed to a constant) than that any variable is contributed in the AND of small ORs. One complication in Håstad's argument is that one must deal with the bias induced on the probability space by the observations of terms that are falsified by the restriction. To handle this in as simple a manner as possible, Håstad in fact proves the switching lemma conditioned on the event that some arbitrary function's value is fixed to 0. He then argues that such conditioning can only bias the outcome in his favour. For the fully independent restrictions that Håstad first considers this argument is fairly easy. However, with increased complexity of the families of restrictions, such arguments become increasingly non-trivial. Furthermore, because the argument is recursive in nature, in some cases there are other conditionings that complicate things further.

Simpler and more elegant proofs have subsequently been given by Razborov (1993)[32] and Beame (1994)[4]. The proof due to Razborov uses an unusual combinatorial strategy. The full argument of that proof is presented here.

Proof of Theorem 1.22. Let us say a restriction β is *bad* if $DT_{depth}(f|_\beta) > d$, and let $\mathcal{B}$ be the set of all *bad* restrictions. Then, our goal reduces to showing that:

$$\frac{|\mathcal{B}|}{|\mathcal{R}_s|} \leq (10pw)^d$$

In order to do this, we will define an encoding $\text{Enc}(\beta)$ of each bad restriction β. This encoding will consist of a restriction B_1 which is basically an extension of β with exactly d more variables fixed, plus a few extra bits of auxillary information. We will then go on to show a 'decoding' procedure which recovers β from $\text{Enc}(\beta)$. This implies that the encoding maps each *bad* restriction $\beta \in \mathcal{B}$ to something unique, which means that we have an injective mapping $\mathcal{B} \hookrightarrow \mathcal{R}_{s-d} \times A$, where A is a small auxillary set. Thus, we conclude that $|\mathcal{B}| \leq |\mathcal{R}_{s-d}| \times |A|$. Hence, an upper bound on the size of the auxillary set A will enable us to prove the required probability bound.

Recall that we have a DNF f of width at most w over n variables with an ordered set of terms $T_1, T_2, T_3, \ldots$, along with some bad restriction $\beta \in \mathcal{B}$. Since β is a bad restriction, it does not fix any terms T_i to 1, since that would make the $f|_\beta$ constantly 1, which would imply that $DT_{depth}(f|_\beta) = 0$. However, β will probably 'kill' many terms, ie. fix them to 0. This is because β has to fix just one literal to 0 in order to kill the whole term. Also, in case β does not kill the term T_i, it leaves a non-trivial term $T_i|_\beta$ over the variables set to $*$. Thus, when we apply β to f, we know that it kills most terms, fixes no terms to 1, and restricts the terms that it keeps alive to fewer variables. Usually, we denote by T_{i_1} the first term in the ordering $T_1, T_2, T_3, \ldots$ which β does not kill. Also, we write $U_1 = T_{i_1}|_\beta$ for the restricted version of that term, which is basically a conjunction on literals set to $*$.

We will construct a canonical decision tree for $f|_\beta$, denoted by $C(f|_\beta)$, as follows –

- Consider the first term T_{i_1} in order which is not killed by β. Say it reduces to the term U_1 on d_1 variables. Construct a complete depth-d_1 decision tree over the variables in U_1 by querying them in order of their indices.
- Note that there will be exactly one path, which we will denote by σ_1, which forces $T_{i_1}|_\beta$ to 1. Put a leaf with label 1 at the end of this path.
- For all the other paths ρ, recursively construct the canonical decision tree $C(f|_{\beta\rho})$ and attach it at the end of the path. If $f|_{\beta\rho}$ is constant, simply attach a leaf with that constant as the label.

Now, β being a bad restriction implies that $DT_{depth}(f|_\beta) > d$. In particular, the depth of the canonical decision tree $C(f|_\beta)$ exceeds d. Therefore, we may define π' to be the

lexicographically leftmost path of depth exceeding d in $C(f|_\beta)$, and π to be the trimmed version of the path π' which fixes exactly d variables. So, we have $\beta\pi \in \mathcal{R}_{s-d}$. Moreover, the restriction $\beta\pi$ is such that $f|_{\beta\pi}$ is not a constant function.

Next, we will define the encoding of β as follows-

- Let T_{i_1} be the first term in order which is not killed by β, and let $U_1 = T_{i_1}|_\beta$ be its restriction under β. Let the number of variables in U_1 be d_1. Now, let σ_1 be the assignment of the variables in U_1 which sets U_1 to 1.
- Let π_1 be the part of the path π which sets these variables.
- Note that, if π_1 is not all of π, then $\beta\pi_1$ must kill U_1. In that case, let T_{i_2} be the first term in order which is not killed by $\beta\pi_1$, and let $U_2 = T_{i_1}|_{\beta\pi_1}$ be its restriction under $\beta\pi_1$. Let the number of variables in U_2 be d_2. Now, let σ_2 be the assignment of the variables in U_2 which sets U_2 to 1, and π_2 be the part of the path π which sets these variables.
- Continue in this manner until eventually π_l finishes all of π. At this point, truncate σ_l to set just the variables that π_l sets.
- Our encoding of β will be given by:

$$Enc(\beta) = \beta\sigma_1\sigma_2 \dots \sigma_l + (\text{some auxillary bits})$$

Note that the restriction $B_1 = \beta\sigma_1\sigma_2 \dots \sigma_l$ has $s - (d_1 + d_2 + \cdots + d_l) = s - d$ variables set to $*$. Thus, $B_1 \in \mathcal{R}_{s-d}$.

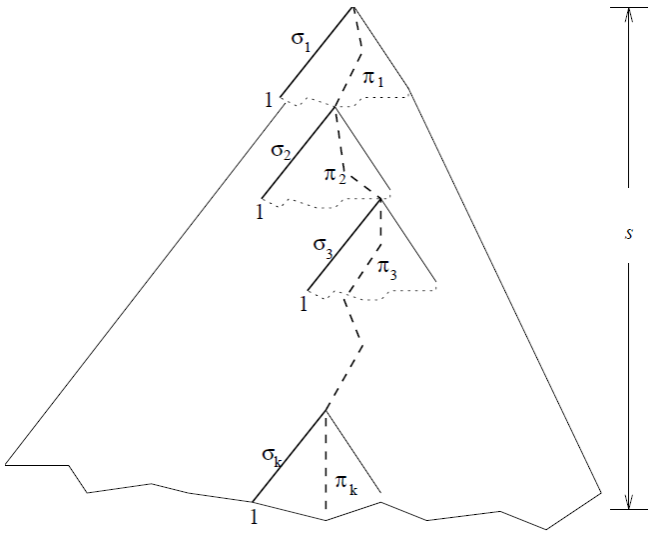

FIGURE 1.5: Encoding of a bad restriction β (taken from Beame[4])

Finally, we will specify the auxillary information that we need to decode B_1 back to β. We already know that T_{i_1} is the first term in f which $\beta\sigma_1$ sets to 1. So, the first term in f which is fixed to 1 by B_1 must also be T_{i_1}. Now, we can add $d_1 \log w$ bits of auxillary information to specify which variables in T_{i_1} are the ones which σ_1 fixes, and an additional d_1 auxillary bits to specify how π_1 sets these variables. Actually, since the decoder does not know the value of d_1, we need to use an additional sentinel symbol so that it can parse the input string correctly, which means that the total number of bits we need to use is $(d_1\lceil\log(w+1)\rceil + d_1) \leq (d_1 \log w + 2d_1)$. Thus, by adding at most $d_1 \log w + 2d_1$ auxillary information bits, the decoder can determine T_{i_1}, σ_1 and π_1. Now, once the decoder knows σ_1 and π_1, it can consider the restriction $B_2 = \beta\pi_1\sigma_2 \ldots \sigma_l$. By construction, T_{i_2} is the first term in f which $\beta\pi_1\sigma_2$ sets to 1. So, the first term in f which is fixed to 1 by B_2 must also be T_{i_2}.Again, by adding at most $d_2 \log w + 2d_2$ auxillary information bits, the decoder can determine T_{i_2}, σ_2 and π_2. The decoder can keep on proceeding in this manner until it has $B_l = \beta\pi_1\pi_2 \ldots \pi_{l-1}\sigma_l$. The only difference now is that we might be looking for a first term which is still undetermined on applying the restriction B_l rather than the first term which is fixed to 1. Irrespective of that, by adding at most $d_l \log w + 2d_l$ auxillary information bits, the decoder can determine σ_l and π_l. So, at this point, the decoder has completely determined the $\sigma_1\sigma_2 \ldots \sigma_l$ part of the encoded restriction $Enc(\beta)$. Moreover, the decoder knows that this part fixes exactly d variables, and so it can figure out that its done. Thus, we conclude that, by using at most $(d_1 \log w + 2d_1) + (d_2 \log w + 2d_2) + \cdots + (d_l \log w + 2d_l) = d \log w + 2d$ bits of auxillary information, a decoder can uniquely recover β from $B_1 = \beta\sigma_1\sigma_2 \ldots \sigma_l$.

In effect, we have shown the existence of an injective mapping from the set $\mathcal{B}$ of bad restrictions into the set $\mathcal{R}_{s-d} \times \{0,1\}^{d \log w + 2d}$. This enables us to conclude that:

$$
\begin{aligned}
\Pr_{\alpha \in \mathcal{R}_s}[\alpha \text{ is a bad restriction}] &\leq \frac{|\mathcal{R}_{s-d} \times \{0,1\}^{d \log w + 2d}|}{|\mathcal{R}_s|} \\
&= \frac{\binom{n}{s-d} 2^{n-(s-d)} (4w)^d}{\binom{n}{s} 2^{n-s}} \\
&= \frac{s(s-1)(s-2)\ldots(s-d+1)}{(n-s+d)(n-s+d-1)\ldots(n-s+1)} (8w)^d \\
&\leq \left(\frac{s}{n-s+d}\right)^d (8w)^d \\
&\leq \left(\frac{p}{1-p}\right)^d (8w)^d \\
&\leq (10pw)^d \qquad \text{since } p \leq \tfrac{1}{5}
\end{aligned}
$$

Hence proved. □

1.4.4 Håstad's switching lemma $\Rightarrow$ PARITY $\notin$ AC^0

The smallest AC class AC^0 consists of constant-depth circuits having a polynomial number of AND and OR gates with unlimited fan-in. Invoking Håstad's switching lemma enables us to prove that PARITY $\notin$ AC^0. More specifically, we get –

Theorem 1.23. *If a circuit C of size S with unbounded fan-in and constant depth k is computing PARITY^n, then:*

$$S \geq 2^{\Omega\left(n^{\frac{1}{k-1}}\right)}$$

Proof of Theorem 1.23. Let C be an AC^0 circuit of size S and depth k computing the boolean function $f : \{0,1\}^n \to \{0,1\}$. It can be shown that C can be converted into a *leveled* depth-k circuit, where the levels alternate AND and OR gates, the input wires are the $2n$ literals, and each gate has fan-out 1, by increasing the size to at most $(2kS)^2 \leq O(S^4)$. Since this increase in size changes only the constant hidden in the Ω-notation in the original statement, we can assume without loss of generality that the circuit is of this form. We can also assume without loss of generality that the bottom layer of C is made up of AND gates.

Let us first hit the circuit with a random restriction α_0, which sets 1/100 fraction of the variables to $*$. Suppose we have an AND gate in the bottom layer with fan-in exceeding $w = 20 \log S$. A Chernoff-type bound shows that, except with probability exponentially small in w (and hence $\ll S$), the gate gets at least $\frac{3}{4}w$ non-$*$variables. Now, each such variable can immediately kill this AND gate with a probability of $\frac{1}{2}$. Thus, we conclude that, except with probability exponentially small in w (and hence $\ll S$), the gate gets killed immediately. Finally, a union bound over all gates in the bottom layer enables us to conclude that there is a high probability that this initial restriction α_0 kills all gates having fan-in exceeding w. Thus, without loss of generality, we can assume that every gate at the bottom level of the circuit has fan-in at most $w = 20 \log S$.

Under the current set of assumptions, the bottom two layers of C consist of DNFs of width at most $w = 20 \log S$. So, if we apply a random restriction α_1 with $*$-fraction $p = 1/(20w)$ to the circuit C. Then, for each such DNF f, the switching lemma tells us that:

$$\Pr_{\alpha_1}[DT_{depth}(f|_{\alpha_1}) > w] \leq (10pw)^w = \left(\frac{1}{2}\right)^w = \left(\frac{1}{2}\right)^{20 \log S} \ll \frac{1}{S}$$

So, by a union bound over at most S such DNFs, we get:

$$\Pr_{\alpha_1}[\exists f : DT_{depth}(f|_{\alpha_1}) > w] \ll 1 \Rightarrow \Pr_{\alpha_1}[\forall f : DT_{depth}(f|_{\alpha_1}) < w] \gg 0$$

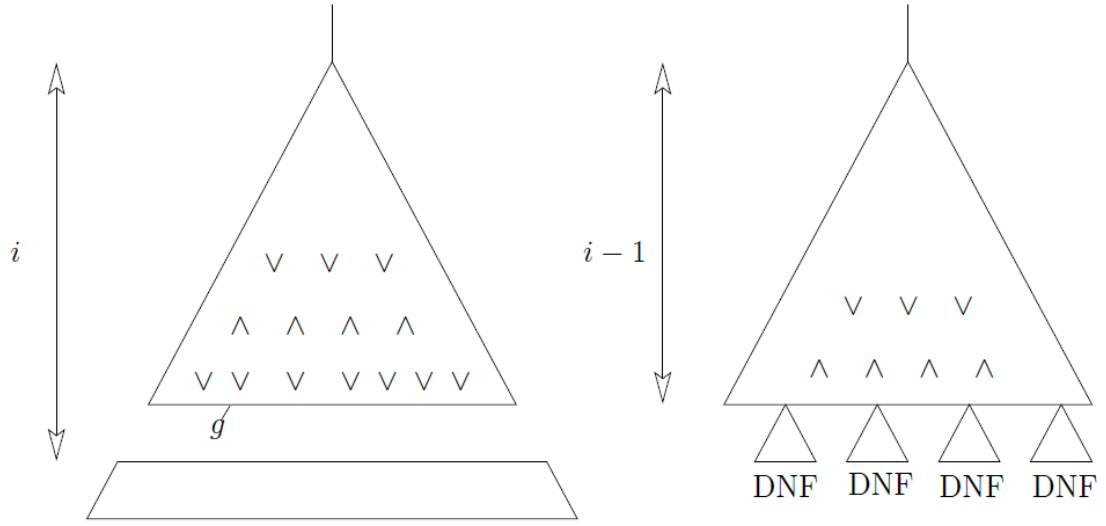

FIGURE 1.6: Collapsing the bottom layers of an AC^0 circuit by application of Håstad's switching lemma

Thus, we see that there exists a *good* restriction α_1 with $*$-fraction $p = 1/(20w)$ whose application simplifies every DNF present in the bottom two layers of the circuit C to functions representable by a depth-w decision tree. But we know that such functions are also representable by a CNF of width w. So, if we fix such a *good* restriction α_1 and plug in these CNFs to the circuit, then we can collapse layers 2 and 3 from the bottom (as shown in Figure 1.6) to get a new circuit of depth $k-1$, which has bottom level fan-in at most w.

Now, we recursively keep applying *good* restrictions $\alpha_2, \alpha_3, \ldots$, each with $*$-fraction $p = 1/(20w)$, and yielding circuits of depth $k-2, k-3, \ldots$. After repeating this $k-2$ times, we obtain a circuit of depth 2. At this point, the number of variables set to $*$ is given by:

$$m = n.p^{k-2} = \frac{n}{(400 \log S)^{k-2}}$$

But every restriction of the PARITY function has to be either PARITY or its negation. Moreover, we know that, to compute PARITY on m variables, we require a DNF of size 2^{m-1} and also a CNF of size 2^{m-1}. Thus, we must have:

$$\begin{aligned} S \geq 2^{m-1} &\Rightarrow \log S \geq \Omega(m) \\ &\Rightarrow O(\log S)^{k-1} \geq n \\ &\Rightarrow \log S \geq \Omega\left(n^{\frac{1}{k-1}}\right) \\ &\Rightarrow S \geq 2^{\Omega\left(n^{\frac{1}{k-1}}\right)} \end{aligned}$$

Hence proved. □

1.5 Extended switching lemma

1.5.1 Background

The Circuit Satisfiability problem, deciding whether a Boolean circuit has an assignment where it evaluates to true, is in many ways the canonical NP-complete problem. It is not only important from a theoretical point of view, but pragmatically, since search problems directly reduce to Circuit Satisfiability without increasing the size of the search space. However, very few results are known either theoretically or empirically about the difficulty of Circuit Satisfiability.

For an algorithm that decides satisfiability for circuits in a class $\mathcal{C}$, we express its worst-case running time as $|C|2^{n(1-\mu)}$, where C is a circuit with n inputs. We say that μ is the savings over exhaustive search and write it in terms of n and the parameters of the class $\mathcal{C}$. The larger μ is, the more non-trivial the algorithm can be considered. Some typical questions that arise in this regard are as follows –

- When can we exploit the structural properties of a circuit class $\mathcal{C}$ to obtain savings over exhaustive search?
- What are the best savings for various circuit classes?
- How is the expressive power of a circuit class $\mathcal{C}$ related to the amount of savings for its satisfiability problem?

Recently, Impagliazzo, Matthews and Paturi (2012)[20] came up with an improvement in the savings for the satisfiability algorithm for AC^0 circuits of size cn and depth d. The exact theorem they obtained was the following –

Theorem 1.24. *There is a Las Vegas algorithm for deciding the satisfiability of AC^0 circuits with cn gates and depth d whose expected time has savings at least $\frac{1}{O(\log c+d\log d)^{d-1}}$.*

The above algorithm immediately follows from the existence of an algorithm that enumerates all satisfying assignments by partitioning the space into sub-cubes where the circuit is constant.

Theorem 1.25. *There exists a Las Vegas algorithm which, on input a size cn and depth d circuit C in n variables, produces a set of restrictions $\{\rho_i\}_i$ which partition $\{0,1\}^n$ and such that for each i $C|_{\rho_i}$ is constant. The expected time and number of restrictions are both $poly(n)|C|2^{n(1-\mu_{c,d})}$, where the savings $\mu_{c,d}$ is at least $\frac{1}{O(\log c+d\log d)^{d-1}}$.*

As another consequence of the above theorem, they also obtained the following bounds on correlation between AC^0 circuits and the PARITY function.

Theorem 1.26. *The correlation of PARITY with any AC^0 circuit of size cn and depth d is at most $2^{-\mu_{c,d}n} = 2^{-n/O(\log c + d \log d)^{d-1}}$.*

As a key ingredient in their analysis, they extended Håstad's switching lemma to handle multiple k-CNFs and k-DNFs.

1.5.2 Statement

The extended switching lemma introduced by Impagliazzo, Matthews and Paturi[20] is stated as follows –

Theorem 1.27 (Extended Switching Lemma)**.** *Let $\phi_1,...,\phi_m$ be a sequence of k-CNFs and/or k-DNFs in the same n variables. For any $p \leq \frac{1}{13}$, let ρ be a random restriction which leaves pn variables unset. The probability that the decision tree for $(\phi_1,...,\phi_m)|_\rho$ has a path of length $\geq t$, where each ϕ_i contributes at least one node to the path, is at most $(13pk)^t$.*

Note that, for any non-empty subset S of $\{\phi_1,...,\phi_m\}$, the probability that the decision tree for that subset of k-CNFs or k-DNFs restricted to ρ has a path of length $\geq t$ where each $\phi_i \in S$ contributes at least one node to the path is also at most $(13pk)^t$. Moreover, observe that every path P in the decision tree for $(\phi_1,...,\phi_m)|_\rho$ is also a path in the decision tree for some subset S of $\{\phi_1,...,\phi_m\}$ restricted to ρ, where each formula in S contributes at least one node to P. Now, since the number of non-empty subsets of $\{\phi_1,...,\phi_m\}$ is equal to $(2^m - 1)$, using a simple union bound immediately leads us to the following corollary –

Corollary 1.28. *Let $\phi_1,...,\phi_m$ be a sequence of k-CNFs and/or k-DNFs in the same n variables. For any $p \leq \frac{1}{13}$, let ρ be a random restriction which leaves pn variables unset. The probability that the decision tree for $(\phi_1,...,\phi_m)|_\rho$ has a path of length $\geq t$ is at most $(2^m - 1)(13pk)^t$.*

1.5.3 An Illustrative Build-up to the Proof

The proof of the extended switching lemma is quite similar to the proof of Hastad's switching lemma presented earlier. The core idea is to encode a 'bad' restriction and a 'bad' path in the resulting decision tree using a different restriction which sets additional variables and a few extra bits. The total size needed for this encoding will be much less

than the size needed to encode the original restriction, and this ratio will give us the required probability bound. In the rest of this section, we will try to build intuition for the proof by illustrating how the encoding and decoding algorithms work.

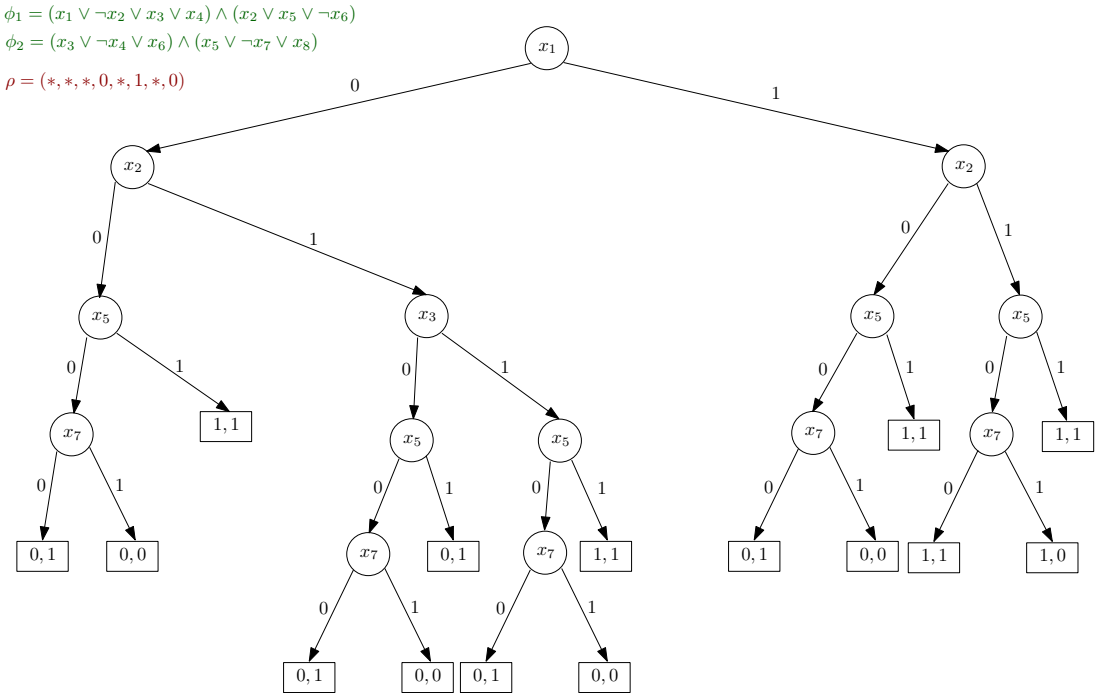

FIGURE 1.7: Canonical decision tree for $(\phi_1, \phi_2)|_\rho$

Let $\Phi = (\phi_1, \phi_2)$ be a sequence of CNFs, where:

$$\phi_1 = (x_1 \vee \neg x_2 \vee x_3 \vee x_4) \wedge (x_2 \vee x_5 \vee \neg x_6)$$

$$\phi_2 = (x_3 \vee \neg x_4 \vee x_6) \wedge (x_5 \vee \neg x_7 \vee x_8)$$

Suppose that the restriction $\rho = \{x_4 = 0, x_6 = 1, x_8 = 0\}$ is applied to Φ. Then, Figure 1.7 shows the canonical decision tree constructed for $(\phi_1, \phi_2)|_\rho$ by querying the variables in order of their indices. Now, let us fix the target depth to be 4, and choose the lexicographically leftmost path $P = \{x_1 = 0, x_2 = 0, x_5 = 0, x_7 = 0\}$ of length at least 4 in this canonical decision tree. Note that both ϕ_1 and ϕ_2 contribute at least one node to P.

Let X_1, X_2, X_3, X_4 denote the variables queried along the path P and let $\vec{p} = (p_1, p_2, p_3, p_4)$ denote the values that P assigns to these variables. Now, let C_1, C_2, C_3, C_4 and F_1, F_2, F_3, F_4 denote the clauses and formulae respectively which contribute X_1, X_2, X_3, X_4 to the path P. Also, let $\vec{index} = (index_1, index_2, index_3, index_4)$ denote the indices of X_1, X_2, X_3, X_4 in the corresponding clauses according to the canonical ordering on the variables, and

let $\vec{last} = (last_1, last_2, last_3, last_4)$ be such that, for $1 \leq i \leq 4$, we have:

$$last_i = \begin{cases} 2 & \text{if } X_i \text{ is the last variable contributed by } F_i \text{ along } P \\ 1 & \text{if } X_i \text{ is the last variable contributed by } C_i \text{ (but not by } F_i\text{) along } P \\ 0 & \text{otherwise} \end{cases}$$

Moreover, we denote by σ_i the restriction which sets X_i in such a manner so as not to satisfy C_i, and sets all other variables to $*$; while we denote by π_i the restriction which sets X_i to p_i, and sets all other variables to $*$.

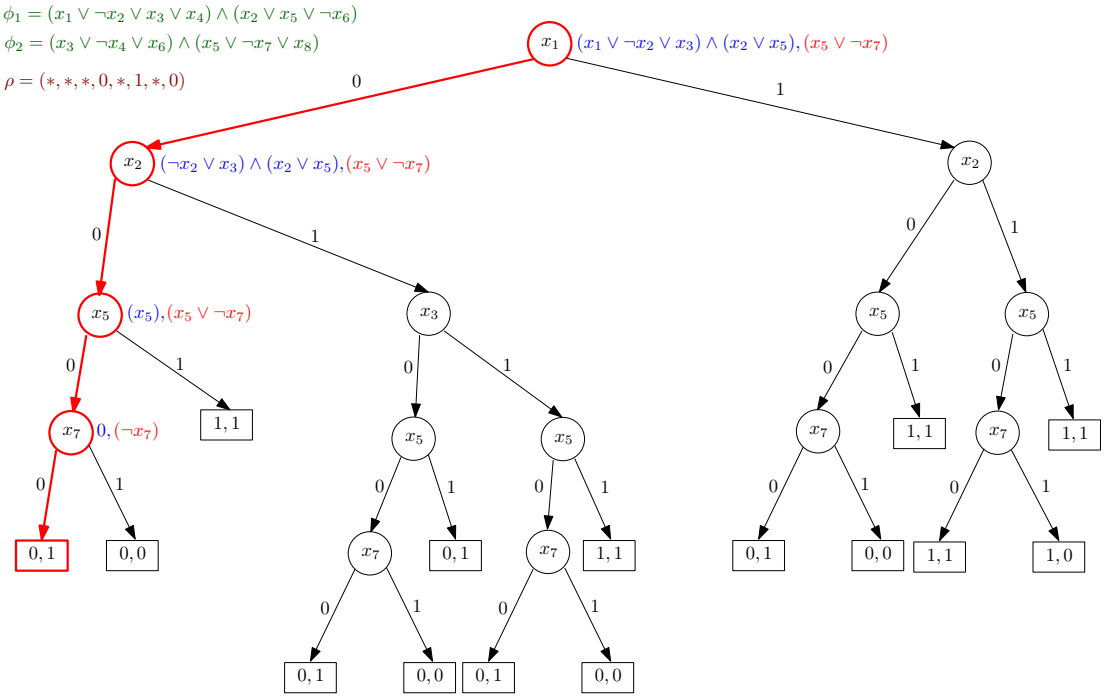

FIGURE 1.8: Path P in canonical decision tree for $(\phi_1, \phi_2)|_\rho$

Figure 1.8 highlights the path P in the canonical decision tree for $(\phi_1, \phi_2)|_\rho$ and also mentions the corresponding restrictions of ϕ_1 and ϕ_2 alongside each node in the path. By examining Figure 1.8 closely, it becomes quite easy for us to fill in the values of X_i, p_i, F_i, C_i, index$_i$, last$_i$ and σ_i in the table shown below –

i	X	p	F	C	index	last	σ
1	x_1	0	ϕ_1	$(x_1 \vee \neg x_2 \vee x_3 \vee x_4)$	1	0	$x_1 = 0$
2	x_2	0	ϕ_1	$(x_1 \vee \neg x_2 \vee x_3 \vee x_4)$	2	1	$x_2 = 1$
3	x_5	0	ϕ_1	$(x_2 \vee x_5 \vee \neg x_6)$	2	2	$x_5 = 0$
4	x_7	0	ϕ_2	$(x_5 \vee \neg x_7 \vee x_8)$	2	2	$x_7 = 1$

Now, by looking at the table above, we will encode the restriction ρ to:
$\rho' = \rho\sigma_1\sigma_2\sigma_3\sigma_4 = \{x_1 = 0, x_2 = 1, x_4 = 0, x_5 = 0, x_6 = 1, x_7 = 1, x_8 = 0\}$

Note that the restriction ρ' sets all the variables (and in the same way) that ρ sets; but it also sets 4 extra variables x_1, x_2, x_5, and x_7, in the way they were set by σ_1, σ_2, σ_3 and σ_4 respectively. Therefore, given ρ', if we can somehow manage to figure out which are the extra variables it sets, then we can essentially recover ρ from its encoding ρ'.

An algorithmic procedure for decoding ρ from ρ' does exist if we provide $\vec{index} = (1, 2, 2, 3)$, $\vec{last} = (0, 1, 2, 2)$ and $\vec{p} = (0, 0, 0, 0)$ to the decoder as auxillary information. The algorithm starts off by initializing $last_0 = 1$, $F_0 = \phi_1$, and $\rho_0 = \rho'$. Then, at every iteration, the algorithm determines the values of F_i, C_i, X_i, π_i and ρ_i in the following manner –

- If $last_{i-1} = 2$, then F_i is set to ϕ_{j+1}, where j is the index such that the value of F_{i-1} was ϕ_j. Otherwise, if $last_{i-1} = 1$ or $last_{i-1} = 0$, then F_i is set to be equal to F_{i-1}.
- If $last_{i-1} = 0$, then C_i is set to be equal to C_{i-1}. Otherwise, if $last_{i-1} = 1$ or $last_{i-1} = 2$, then C_i is set to be the first clause in F_i not satisfied by the restriction ρ_{i-1}.
- Once F_i and C_i have been obtained, then X_i is determined by looking at the value of $index_i$.
- Finally, the restriction ρ_i is constructed by setting the value of X_i to p_i in the restriction ρ_{i-1}.

For our current example, the values of F_i, C_i, X_i, π_i and ρ_i obtained by the algorithm in each iteration are presented in the following table –

i	last	F	C	index	X	p	ρ
0	1	ϕ_1	-	-	-	-	$(0, 1, *, 0, 0, 1, 1, 0)$
1	0	ϕ_1	$(x_1 \vee \neg x_2 \vee x_3 \vee x_4)$	1	x_1	0	$(\mathbf{0}, 1, *, 0, 0, 1, 1, 0)$
2	1	ϕ_1	$(x_1 \vee \neg x_2 \vee x_3 \vee x_4)$	2	x_2	0	$(\mathbf{0}, \mathbf{0}, *, 0, 0, 1, 1, 0)$
3	2	ϕ_1	$(x_2 \vee x_5 \vee \neg x_6)$	2	x_5	0	$(\mathbf{0}, \mathbf{0}, *, 0, \mathbf{0}, 1, 1, 0)$
4	2	ϕ_2	$(x_5 \vee \neg x_7 \vee x_8)$	2	x_7	0	$(\mathbf{0}, \mathbf{0}, *, 0, \mathbf{0}, 1, \mathbf{0}, 0)$

The above table clearly tells us that the variables x_1, x_2, x_5, and x_7 were set by the path P and not by the original restriction ρ. Hence, by setting these variables to $*$, we can easily recover the original restriction as:

$$\rho = \{x_1 = *, x_2 = *, x_3 = *, x_4 = 0, x_5 = *, x_6 = 1, x_7 = *, x_8 = 0\}$$

In order to extend the intuitions gleaned from this section into a rigorous proof of the extended switching lemma, we need to formalize the algorithmic decoding procedure presented above and calculate the bound on the size of the encoding set so as to obtain the required probability bound.

1.5.4 Formal Proof of the Extended Switching Lemma

Proof of Theorem 1.27. Let P' be a path in the joint decision tree for $(\phi_1,...,\phi_m)|_\rho$ of length at least t where each ϕ_i contributes at least one node to the path. Let P denote the prefix of P' of length exactly t. Let $X_1, X_2, \ldots, X_t$ denote the variables queried along the path P and let $p_1, p_2, \ldots, p_t$ denote the values that P assigns to $X_1, X_2, \ldots, X_t$.

Let $C_1, C_2, \ldots, C_t$ and $F_1, F_2, \ldots, F_t$ denote the clauses and formulae respectively which contribute $X_1, X_2, \ldots, X_t$. Note that some C_is and F_is may refer to the same clauses or formulae if they contribute more than one variable to P. Now, let $index_1, index_2, \ldots, index_t$ denote the indices of $X_1, X_2, \ldots, X_t$ respectively in the corresponding clauses $C_1, C_2, \ldots, C_t$. Also, for $1 \leq i \leq t$, let $last_i$ be 2 if X_i is the last variable contributed by F_i along P; be 1 if x_i is the last variable contributed by C_i (but not by F_i) along P; and be 0 otherwise.

Let $\sigma = \sigma_1\sigma_2 \ldots \sigma_t$, where σ_i is a restriction where we have:

$$\forall y \neq X_i : \sigma_i(y) = * \text{ and } \sigma_i(X_i) = \begin{cases} 0 & \text{if the literal } X_i \text{ appears in the clause } C_i \\ 1 & \text{if the literal } \neg X_i \text{ appears in the clause } C_i \end{cases}$$

Note that σ is constructed to not satisfy the clauses $C_1, C_2, \ldots, C_t$.

Let π_i be the restriction where $\pi_i(X_i) = p_i$ and $\pi_i(y) = *$ for all $y \neq X_i$.
Note that $P = \pi_1\pi_2 \ldots \pi_t$.

Let $\rho_i = \rho\pi_1 \ldots \pi_i\sigma_{i+1} \ldots \sigma_t$ for $0 \leq i \leq t$. Note that $\rho_0 = \rho\sigma = \rho'$ and $\rho_t = \rho P$.

We map (ρ, P) to $\rho' = \rho\sigma \in \mathcal{R}_{pn-t}$, $\vec{index} = (index_1, index_2, \ldots, index_t) \in [k]^t$, $\vec{last} = (last_1, last_2, \ldots, last_t) \in [3]^t$ and $\vec{p} = (p_1, p_2, \ldots, p_t) \in [2]^t$.

Observe that, when X_{i+1} is queried along the path P in the decision tree for $(\phi_1,...,\phi_m)|_\rho$, C_{i+1} is actually the first clause in F_{i+1} not satisfied by the restriction $\rho\pi_1 \ldots \pi_i$, because otherwise a variable in an earlier clause would have been queried instead. Since setting more variables in a restriction cannot change a clause from satisfied to not satisfied, all the clauses in ϕ upto and including C_i must be satisfied by ρ_i. Now, let j be the largest index such that $C_j = C_{i+1}$, or equivalently X_j is the last variable from C_{i+1} along P.

By the construction of $\sigma_{i+1}, \ldots, \sigma_j$, $\sigma_{i+1} \ldots \sigma_j$ does not satisfy C_{i+1}. Morevoer, due to the canonical ordering enforced while constructing the decision tree, either we have $j = t$ or $\rho\pi_1 \ldots \pi_i\sigma_{i+1} \ldots \sigma j$ sets all the variables in C_{i+1}. In either case, no σ_l for $l > j$ can satisfy C_{i+1}, and so we conclude that ρ_i does not satisfy C_{i+1}. Hence, we have proved that, when $last_i = 1$, then C_{i+1} is the first clause in ϕ not satisfied by $\rho_i = \rho\pi_1 \ldots \pi_i\sigma_{i+1} \ldots \sigma_t$.

Keeping this in mind, we can now provide an algorithm for decoding (ρ, P) from ρ', $\vec{index}$, $\vec{last}$ and $\vec{p}$. The algorithm, which recovers ρ by figuring out the variables in ρ' that were set by σ, is as follows –

Algorithm 1.5.1 Algorithm to decode (ρ, P) from ρ', $\vec{index}$, $\vec{last}$ and $\vec{p}$

$last_0 \leftarrow 1$
$F_0 \leftarrow \phi_1$
for $i = 0$ to $t - 1$ **do**
  if $last_i = 2$ **then**
    $F_{i+1} \leftarrow \phi_{q+1}$, where q is the index such that $F_i = \phi_q$
  else
    $F_{i+1} \leftarrow F_i$
  end if
  if $last_i = 0$ **then**
    $C_{i+1} \leftarrow C_i$
  else
    $C_{i+1} \leftarrow$ first clause in F_{i+1} that is not satisfied by ρ_i
  end if
  $X_{i+1} \leftarrow$ variable in C_{i+1} having index $index_{i+1}$
  obtain π_{i+1} by setting X_{i+1} to p_{i+1}
  obtain ρ_{i+1} from ρ_i by using π_{i+1}
end for

Let us call any path in the decision tree for $(\phi_1,\ldots,\phi_m)|_\rho$ a *bad* path if it has length greater than t, and each ϕ_i contributes at least one node to the path. Also, let $\mathcal{B}$ denote the set of all *bad* restrictions in $\mathcal{R}_{pn}$, for which at least one bad path exists in $(\phi_1,\ldots,\phi_m)|_\rho$. Then, the fact that we can uniquely decode (ρ, P) from ρ', $\vec{index}$, $\vec{last}$ and $\vec{p}$ for any bad path P actually proves the existence of an injective mapping from the set $(\mathcal{B} \times \mathcal{P})$ into $(\mathcal{R}_{pn-t} \times [k]^t \times [3]^t \times [2]^t)$, where $\mathcal{P}$ denotes the set of all bad paths under the restriction ρ. Therefore, we can immediately conclude that:

$$|\mathcal{B}| \times |\mathcal{P}| \leq |\mathcal{R}_{pn-t} \times [k]^t \times [3]^t \times [2]^t|$$

Since at least one bad path exists in $(\phi_1,\ldots,\phi_m)|_\rho$ if ρ is a bad restriction, therefore the probability that ρ indeed belongs to the set $\mathcal{B}$ of bad restrictions can be bounded as

follows –

$$\begin{aligned}
\Pr[\rho \in \mathcal{B}] &\leq \frac{|\mathcal{R}_{pn-t} \times [k]^t \times [3]^t \times [2]^t|}{|\mathcal{R}_{pn}|} \\
&= \frac{\binom{n}{pn-t} 2^{n-pn+t} (6k)^t}{\binom{n}{pn} 2^{n-pn}} \\
&= \frac{(pn)!(n-pn)!}{(pn-t)!(n-pn+t)!} (12k)^t \\
&= \left(\frac{pn}{n-pn+t}\right)\left(\frac{pn-1}{n-pn+t-1}\right)\cdots\left(\frac{pn-t+1}{n-pn+1}\right)(12k)^t \\
&\leq \left(\frac{pn}{n-pn+t}\right)^t (12k)^t \\
&\leq \left(\frac{12pk}{1-p}\right)^t \qquad \leq (13pk)^t \qquad \text{since } p \leq \tfrac{1}{13}
\end{aligned}$$

Hence, the proof is complete. □

Chapter 2

Valiant's Neuroidal Model

2.1 Introduction and Problem Specification

2.1.1 Motivation

Let us look at an experience which is quite common in our day-to-day lives. Suppose, while browsing the weekly list of bestsellers in the newspaper, you come across a title which presents a rather surprising juxtaposition of words. The first time you come across this unusual title, the experience presumably causes some adjustments in your brain, since the presentation of the same sequence of words at a later time usually elicits some kind of recognition. The phrase that you learnt to recognize may have been any combination of intelligible words, and your attention may have been drawn to it for only a very brief period of time. Moreover, your memorization of the new book title does not appear to interfere in any way with unrelated knowledge you had previously memorized.

Unfortunately, it is still beyond us to provide a satisfactory explanation for an experience as mundane and commonplace as this. Till date, hardly any theories exist that can satisfactorily explain how any mechanism that even remotely resembles the brain in structure and quantitative parameters could give rise to the variety of such basic phenomenon that human brains appear to exhibit. However, by formulating this problem in a concrete manner, it may be possible to attempt plausible specifications of both the cognitive functions that are to be explained, as well as of some computational models that capture the basic capabilities of the human brain. If the suggested candidates for these cognitive functions and computational models turn out to accurately capture the essential aspects of all the basic phenomenon displayed by the human brain, then an algorithmic explanation will exist for how these functions are supported in the brain.

2.1.2 The Computational Approach

The computational approach proposed by Leslie Valiant[41] in order to study this problem has been described very nicely, by Valiant himself, in terms of the following analogy. Suppose that, one day, on a distant planet, we find some robots that have some very interesting and complex behaviour that we wish to understand better. We could possibly take one of the three different approaches mentioned below in order to achieve this –

(1) We study their components and architecture much as a neurobiologist studies the brain, and construct theories of their general computational capabilities.

(2) We perform experiments on their behaviour like a psychologist might, and construct theories of that behaviour.

(3) We attempt to figure out how one might build systems that resemble the robots in behaviour, using components that resemble those that they are built from.

The third approach mentioned is basically the computational approach, in which one constructs a theory that accommodates both of the above viewpoints simultaneously and also accounts for the computational resources such as time and hardware. It is generally more successful at yielding valuable insights, for which pursuing the first two approaches separately may not be sufficient.

2.1.3 Problem Specification

According to Valiant[41], a computational account of the workings of the brain would essentially consist of three parts. They are –

(1) A precise specification of the cognitive functions or tasks for which an explanation is sought. These cognitive functions could include rote learning or memorization, recall, supervised or unsupervised inductive learning, and any other task that is regarded as basic to cognition.

(2) A description of the basic model of computation. Such a model would include definitions of the individual components that correspond to neurons, as well as of the connections through which they communicate with each other. Also, such a model would need to incorporate a certain degree of parallelism, and hence differ from the standard computational architecture of von Neumann, known as the *stored program* model.

(3) A specification of the computational mechanisms or *algorithms* that enable the underlying model to realize the claimed functions or tasks.

2.1.4 Constraints on the Model

Putting together such a computational account for even the superficially simplest tasks (such as memorization) becomes problematic as soon as we try to take into consideration the gross quantitative parameters that the brain is known to have. Valiant realized that, for any proposed computational model to be considered valid, it has got to respect these quantitative parameters, and the task is further complicated by the issue of *learning*.

As Valiant[41] points out, the major constraints encountered are as follows:

(1) The first major constraint is speed. Individual neurons in the cortex have switching times (time for which a generated action potential lasts) between 1-10 milliseconds. In contrast, humans can perform significant tasks of scene recognition in 100-200 milliseconds. This implies that in a few hundred steps of relay firings at most the brain can do tasks that are currently beyond our imagination to even specify.

(2) A second constraint is the finiteness of the number of neurons in the brain. Typically, the number is around 10^{10}.

(3) The sparseness of the interconnections ($\sim 10^{14}$) relative to the total number of neurons is another important constraint, since this limits the means by which neurons can communicate with each other.

The issue of learning further complicates things. Valiant[41] argues that many recognitions tasks, such as identifying a chair, must have a large learning component, since it is implausible that humans have these capabilities entirely preprogrammed at birth. Thus, the problem becomes much more challenging than simply trying to explain how such recognition tasks can be implemented on a fixed number of slow, sparsely connected neurons. According to him, the correct question that should be asked is: "How can slow, sparsely connected neurons program themselves to do these tasks using knowledge derived from interacting with the world?"

2.1.5 Sources of Complexity

Valiant[41] makes us aware of the three distinct sources of computational difficulty that the mechanisms of the brain have to overcome. These are –

(1) **Computational Complexity:**
Any computational problem may have substantial intrinsic computational difficulty,

which means that a minimum amount of resources, such as time or storage space, will be required for the computation. The resource bounds in the brain, as currently understood, are seriously limited. A fixed number ($\sim 10^{10}$) of neurons need to suffice for a lifetime. Significant recognition tasks can be performed in about 100-200 milliseconds, which allows for relatively few successive steps of cortical firings, when we consider the switching times of individual neurons, which vary between 1 to 10 milliseconds. Clearly, any functionalities attributed to the brain have to be such that these computational resources are demonstrably sufficient.

(2) **Descriptional Complexity:**

Descriptional complexity is concerned with the fact that for any function, while there may be several different programs for computing it, these programs will require some minimum length of description. Once we fix an appropriate language for specifying a program (perhaps a standard programming language, or perhaps a theoretical model such as a Turing machine), and agree on how to measure program length in terms of it, then empirical evidence suggests that, to achieve any significant functionality appropriate for complex situations, one needs long programs. Now, when we seek to find out exactly what the brain does, the target of the search should be of the order of the complexity of a set of programs rather than a single equation as found in physics. Even if the programs have some unifying underlying principles, in their totality they will be long. Furthermore, the interactions among them during execution may be even more difficult to describe, since timing issues in distributed and parallel systems may be very complex in general.

The large descriptional complexity does not arise only because much new information is acquired by the brain through learning. Valiant believes that the computational mechanisms that underly the basic functions of memory, learning, and recall are already of substantial complexity. The evidence from biology, he points out, is certainly consistent with this view, since it has been estimated that fully a third of the mammalian genome is dedicated exclusively to the functions of the brain[36].

(3) **Learning:**

A third source of difficulty is the inherent complexity of *learning*. The various programs in our brain are either present at birth, as a result of evolution, or are learned during life, as a result of interactions with the world. Some may even be the result of some combination of the two. In order to have any chance of uncovering their nature, Valiant feels it is essential that we adopt an intellectual view of how these programs got into the brain in the first place. The phenomenon for which he seeks a quantitative explanation is: how can a system, with limited computational resources and exposure only to a moderate number of situations, acquire programs that are effective and robust in dealing with new situations not previously seen?

Valiant finds it necessary to draw our attention to these sources of complexity, since therein lie the impediments which any computational theory of the brain will need to confront. Thus, he argues, there is some virtue in exhibiting mechanisms that overcome them, regardless of whether or not these mechanisms are actually the same as those used in the brain.

2.1.6 Random Access Tasks

The tension between the substantial functionality of the brain and its restricted computational resources may be interpreted as evidence that the problem is too difficult in the present state of knowledge and cannot be pursued fruitfully in the foreseeable future. However, Valiant[41] chooses to interpret it in the opposite direction. He argues that, when a problem is severely constrained, it may actually be easier to investigate merely because the number of plausible avenues to search are so few that even the first one found may yield valuable insights. Since the severity of the constraints is the leverage that Valiant hopes to exploit, he seeks to maximize it by focusing on a class of tasks to which sparsely connected, slow neurons seem to be the least suited. He calls them *random access tasks* because the attribute that they all share is that the execution of any one of them has the potential to involve any part of memory. For example, the task of memorizing a new book title that consists of the juxtaposition of an almost arbitrary pair of words potentially requires access to any of the words that the reader already knows. For each such random access task, Valiant tries to give the simplest formulation that, he believes, captures at least some fundamental computational hurdle that must be overcome.

2.1.7 Neuroidal Tabula Rasa (NTR) and Peripheral Devices

Since Valiant decides to focus specifically on random access tasks, he finds it convenient for the sake of conceptual simplicity to separate the device that performs them from the devices that do not. Therefore, he hypothesizes a device called *neuroidal tabula rasa* (NTR) which is capable of computing the relevant random access tasks. Moreover, he hypothesizes the existence of some *peripheral devices* that are needed to mediate between the NTR and the senses and chooses to delegate all other tasks to various peripheral devices. In his model, Valiant considers the NTR to be free of preprogrammed knowledge, except for a few generic algorithms needed to realize the basic random access tasks.

2.1.8 A Simple Recipe for Recognition in the NTR

Let us look back at the problem of learning to recognize a book title, say "Lord of the Rings", which presents a previously unseen juxtaposition of words. Since the words appearing in the title were memorized previously, Valiant[41] assumes the presence of specific *neuroids* that represent each of these words in memory. A *neuroid* is basically a linear threshold element with an associated state space that acts as an abstraction for a biological neuron in the NTR. Moreover, the detection of each of these words by the visual sensory organs induces the peripherals to cause a *firing* of the particular neuroids representing it, where the *firing* of a neuroid in the NTR is an event that corresponds to the generation of an action potential by a real neuron. Also, Valiant wishes to model a synaptic connection between the axon of one neuron and the dendritic tree of another by directed edges between 2 neuroids in the NTR, and he assigns a weight to each such directed edge in order to capture the synaptic strength.

Now, in order to be able to recognize the book title "Lord of the Rings" on subsequent presentation, the NTR first needs to select a neuroid, whose current state indicates that it does not represent any memory item at present, that has incoming edges from at least one neuroid representing each of the words in the title, namely "Lord", "of", "the", and "Rings". Once such a neuroid has been selected, the NTR needs to run an algorithm that updates its threshold to make it equal to the sum of the weights of all the incoming edges from neuroids that represent any one of the 4 words in the title. For example, let us suppose that the NTR selects a neuroid that has exactly one incoming edge of weight 1 from each of the set of neuroids representing one of the 4 words in the title. Then, the algorithm will end up setting the threshold of this neuroid to 4. In addition, the algorithm needs to update the neuroid's state to a new one that indicates that it is no longer available to represent any new item, and also set the weights of all the incoming edges from neuroids not representing any of the words in the title to 0.

The above algorithm, suggested by Valiant[41], selects a particular neuroid satisfying certain conditions and makes it *represent* the book title in the sense that, every subsequent time the title is encountered, the selected neuroid ends up firing spontaneously. Notice that, only when *all* the 4 words in the title are encountered again, the sum of the weights on the incoming edges exceeds the new threshold of the neuroid set by the algorithm, causing it to fire spontaneously. Moreover, the firing of any neuroids not representing one of the words in the title fail to exert any influence on the firing of this neuroid in the future, since the weights of all the edges coming from them have been set

to 0 by the algorithm. Another interesting point to note here is that the algorithm always chooses a neuroid which does not already represent some item in memory, and once it is made to represent the encountered book title, its state is changed in such a manner so as to ensure that it does not get picked for representing another newly encountered item in the future, which is consistent with the cumulative nature of memorization. Thus, even this simple algorithm shows some very promising characteristics and should pique one's interest considerably to study and comprehend Valiant's model in its entirety.

The algorithm outlined in this section will be revisited and formulated more clearly in section 2.4.6, once a proper formal description of the neuroidal model has been provided. Before that however, we explore the physiology of the brain (in section 2.2) and some insights from cognitive psychology (in section 2.3), so that the reader can better appreciate the intuitions behind Valiant's model when we finally describe it in full detail.

2.2 The Brain and its Inner Workings

2.2.1 A History of our Understanding of the Brain

The idea that understanding the biological brain may lead to a better understanding of ourselves is a tantalizing one indeed. It is no doubt responsible for the substantial research efforts that have been devoted to the brain over the last century, and that continue with increasing momentum. The history of this science has been punctuated with a series of striking discoveries and the development of some very powerful experimental techniques. Some of the landmark events that stand out in the timeline are as follows –

- In the 1880s, Spanish neuroanatomist Santiago Ramón y Cajal proposed the so-called 'neuron doctrine', which asserted that the brain consisted of cells called *neurons*, discrete and physically separate from each other, that communicated with each other via specialized junctions, or spaces, between cells.
- Subsequently, in 1897, Sir Charles Scott Sherrington suggested the term *synapse* for the gaps between neurons at the points at which they came close to touching.
- In the 1920s, Adrian and Zotterman recorded electrical impulses from single nerve fibers. This finally confirmed the view that long-range communication along nerve fibers was electrical in nature.
- Around 1950, John Eccles first provided experimental evidence that communication between neurons at synapses was chemical in nature.

- A detailed theory of how axons produced electrical impulses, or *action potentials*, was finally offered by Hodgkin and Huxley in 1952.

Thus, by the middle of the 20^{th} century, substantial progress had been made toward understanding the basic nature of the constituents of the brain. Although at the more macroscopic or systems level an equally great amount is known, relatively less is understood. Associations between specific parts of the brain and their functionalities can be made by electric recordings and by recently developed non-intrusive techniques for measuring the distribution of blood flow in the brain while a subject is performing various activities. Through anatomical studies, one can investigate which parts of the brain are connected to each other directly and which are not. In spite of the wealth of knowledge which has now accumulated, the main question of how the brain represents and processes information remains unresolved.

2.2.2 Neurons and Synapses

The core component of the nervous system, which includes the brain, spinal cord, and peripheral ganglia, is the *neuron*, which is an electrically excitable cell that processes and transmits information through electrochemical signals. A typical neuron (shown in Figure 2.1) possesses a cell body (often called the soma), dendrites, and an axon. Dendrites are thin structures that arise from the cell body, often extending for hundreds of micrometres and branching multiple times, giving rise to a complex 'dendritic tree'. An axon is a special cellular extension that arises from the cell body at a site called the axon hillock and travels for a long distance, as far as 1 meter in humans. The cell body of a neuron frequently gives rise to multiple dendrites, but never to more than one axon, although the axon may branch hundreds of times before it terminates. An axonal branch of one cell sometimes comes in very close proximity to a point in the dendritic tree of another cell to form a specialized connection called a *synapse*. Neurons communicate with one another via synapses. Electrical activity in the axon of the *presynaptic* neuron can cause chemical changes at the synapse and thereby influence electrical activity in a dendrite of the *postsynaptic* neuron. Synapses can be excitatory or inhibitory, i.e. they can either increase or decrease activity in the target neuron. Some neurons also communicate via electrical synapses, which are direct, electrically conductive junctions between cells.

All neurons are electrically excitable, maintaining voltage gradients across their membranes by means of metabolically driven ion pumps, which combine with ion channels embedded in the membrane to generate intracellular-versus-extracellular concentration

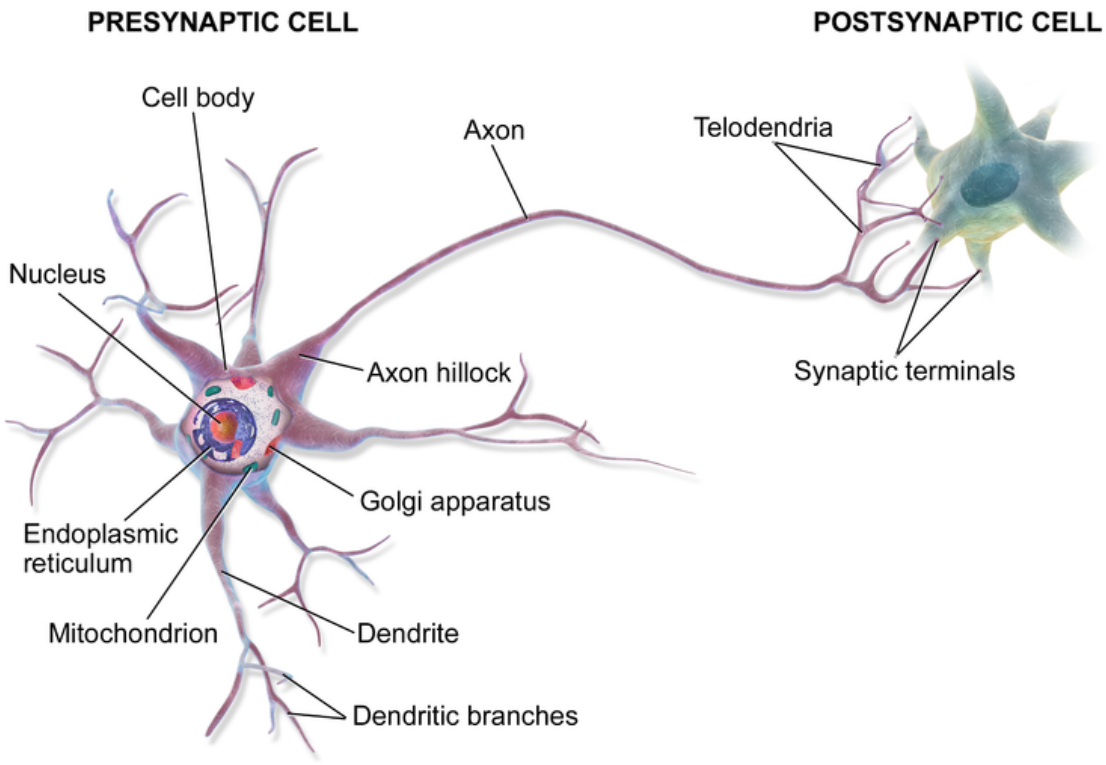

FIGURE 2.1: Anatomy of a Typical Neuron (*Source: Wikimedia Commons*)

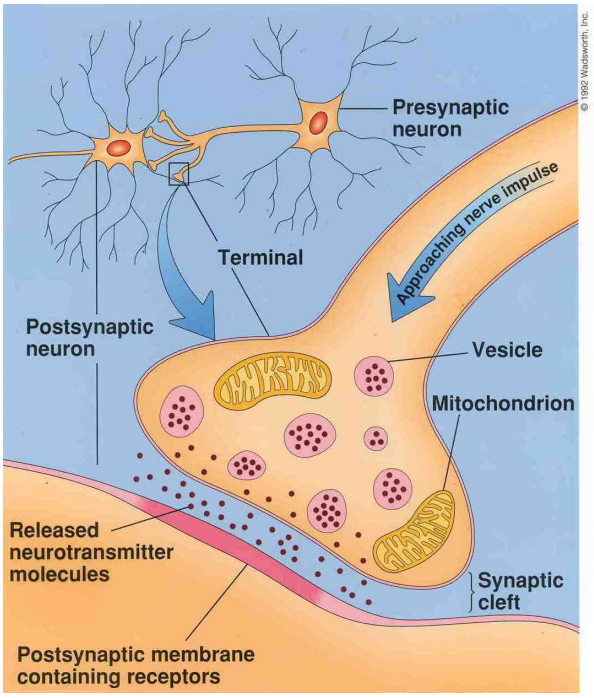

FIGURE 2.2: Synaptic Transmission via Neurotransmitters in a Chemical Synapse (*Source: Wikimedia Commons*)

differences of ions such as sodium, calcium, potassium, and chloride. Changes in the cross-membrane voltage can alter the function of voltage-dependent ion channels. If the voltage changes by a large enough amount, an all-or-none electrochemical pulse called an *action potential* is generated, which travels rapidly along the cell's axon, and activates synaptic connections with other cells when it arrives.

In a chemical synapse, the process of synaptic transmission is as follows. When an action potential reaches the axon terminal, it opens voltage-gated calcium channels, allowing calcium ions to enter the terminal. Calcium causes synaptic vesicles filled with neurotransmitter molecules to fuse with the membrane, thereby releasing their contents into the synaptic cleft. As shown in Figure 2.2, the neurotransmitters diffuse across the synaptic cleft and activate chemical receptors on the postsynaptic neuron. The most well-known neurotransmitters are glutamate, GABA (gamma aminobutyric acid), acetylcholine, dopamine and serotonin. The effect upon the postsynaptic neuron is determined not by the presynaptic neuron or by the neurotransmitter, but by the type of receptor that is activated. Receptors can be classified broadly as excitatory (causing an increase in firing rate), inhibitory (causing a decrease in firing rate), or modulatory (causing long-lasting effects not directly related to firing rate).

2.2.3 Generation of Action Potentials

Action potentials result from the presence of special types of voltage-gated ion channels in the cell membrane of neurons[21]. A voltage-gated ion channel is a cluster of proteins embedded in the membrane that has three key properties:

(i) It has the capability to assume more than one conformation.

(ii) At least one of the conformations creates a channel through the membrane that is permeable to specific types of ions.

(iii) The transition between the conformations is influenced by the membrane potential.

Thus, a voltage-gated ion channel tends to be open for some values of the membrane potential, and closed for others. However, the relationship between membrane potential and channel state is probabilistic. Moreover, a time delay is involved. Therefore, the accurate thing to say is that the membrane potential determines the rate of transitions and the probability per unit time of each type of transition.

Voltage-gated ion channels are capable of producing action potentials because they can give rise to positive feedback loops. The membrane potential controls the state of the ion channels, but the state of the ion channels controls the membrane potential. Thus, in some situations, a rise in the membrane potential can cause ion channels to open, thereby causing a further rise in the membrane potential. An action potential occurs when this positive feedback cycle proceeds explosively. The variation in amplitude of an action potential with time is determined by the biophysical properties of the specific voltage-gated ion channels that produce it.

The course of the action potential can be divided into five parts as follows –

(1) **Rising Phase:** For a neuron at rest, there is a high concentration of sodium and chlorine ions in the extracellular fluid compared to the intracellular fluid while there is a high concentration of potassium ions in the intracellular fluid compared to the extracellular fluid. This concentration gradient along with potassium leak channels present on the membrane of the neuron causes an efflux of potassium ions making the resting potential E_K close to 70 mV. A typical action potential is initiated at the axon hillock by a depolarization; like, for example, a stimulus that increases the membrane voltage V_m. The depolarization opens both the sodium and potassium channels in the membrane, allowing the ions to flow into and out of the axon, respectively. If the depolarization is small, the outward potassium current overwhelms the inward sodium current and the membrane repolarizes back to its normal resting potential around 70 mV. However, if the depolarization is large enough, the inward sodium current increases more than the outward potassium current and a runaway condition (positive feedback) results: the more inward current there is, the more V_m increases, which in turn further increases the inward current. A sufficiently strong depolarization causes the voltage-sensitive sodium channels to open, and the increasing permeability to sodium drives V_m closer to the sodium equilibrium voltage $E_{Na} \approx +55mV$. The increasing voltage in turn causes even more sodium channels to open, which pushes V_m still further towards E_{Na}. This positive feedback continues until the sodium channels are fully open and V_m is close to E_{Na}. The critical threshold voltage for this runaway condition is usually around 45 mV, but it depends on the recent activity of the axon. The sharp rise in V_m and sodium permeability correspond to the rising phase of the action potential.

(2) **Peak Phase:** The positive feedback of the rising phase slows and comes to a halt as the sodium ion channels become maximally open. At the peak phase of the action potential, the sodium permeability is maximized and the membrane voltage V_m is nearly equal to the sodium equilibrium voltage E_{Na}.

(3) **Falling Phase:** The same raised voltage that opened the sodium channels initially also slowly shuts them off, by closing their pores. Thus, the sodium channels become inactivated. This lowers the membrane's permeability to sodium relative to potassium, driving the membrane voltage back towards the resting value. At the same time, the raised voltage opens voltage-sensitive potassium channels; the increase in the membrane's potassium permeability drives V_m towards E_K. Combined, these changes in sodium and potassium permeability cause V_m to drop quickly, repolarizing the membrane and producing the falling phase of the action potential.

(4) **Undershoot Phase:** The raised voltage opens up many more potassium channels than usual, and some of these do not close right away when the membrane returns to its normal resting voltage E_K. In addition, further potassium channels open in response to the influx of calcium ions during the action potential. The potassium permeability of the membrane is transiently unusually high, driving the membrane voltage V_m even lower than the potassium equilibrium voltage E_K. Hence, there is a hyperpolarization, that persists until the membrane potassium permeability returns to its usual value, and this corresponds to the undershoot phase of the action potential.

(5) **Refractory Period:** Each action potential is followed by a refractory period, which can be divided into an absolute refractory period, during which it is impossible to evoke another action potential, and then a relative refractory period, during which the stimulus required to evoke another action potential has to be stronger than usual. These two refractory periods are caused by changes in the state of sodium and potassium channel molecules. When closing after an action potential, sodium channels enter an *inactivated* state, in which they cannot be made to open regardless of the membrane potential, and this gives rise to the absolute refractory period. Even after a sufficient number of sodium channels have transitioned back to their resting state, it frequently happens that a fraction of potassium channels remains open, making it difficult for the membrane potential to depolarize, and thereby giving rise to the relative refractory period. Because the density and subtypes of potassium channels may differ greatly between different types of neurons, the duration of the relative refractory period is highly variable.

In Figure 2.3 below, we have shown a diagrammatic representation of the typical phases in the generation of an action potential via voltage-gated ion channels.

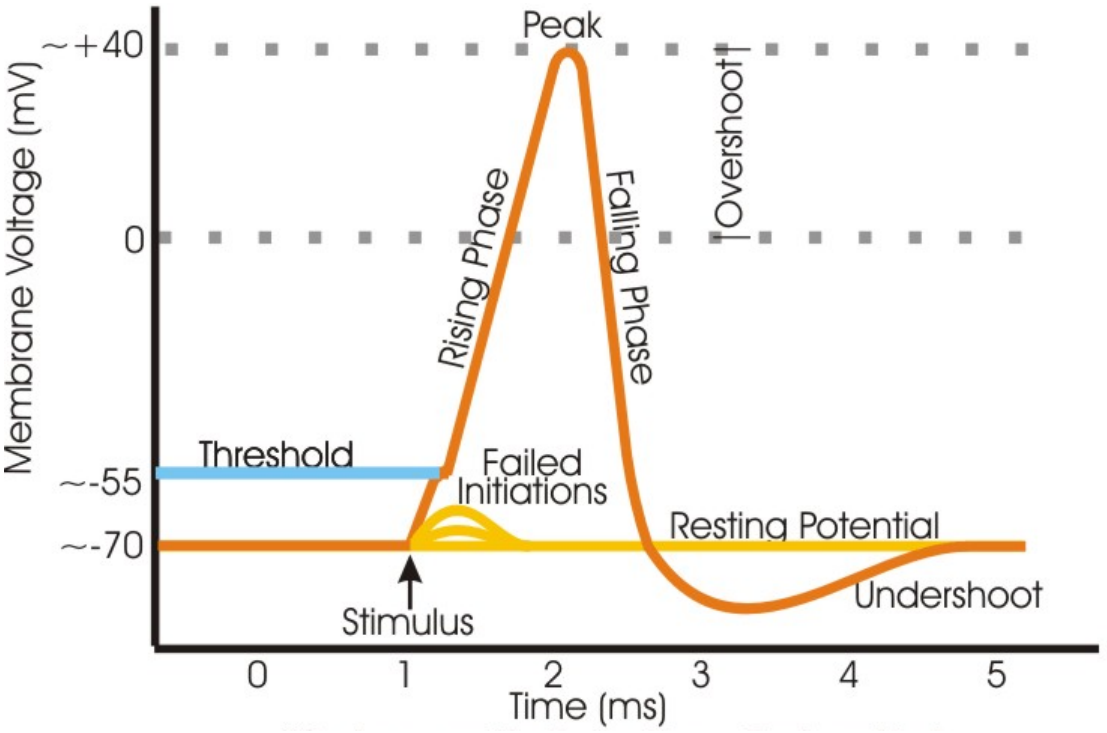

FIGURE 2.3: Typical Phases in the Generation of an Action Potential via Voltage-Gated Ion Channels (*Source: Wikimedia Commons*)

2.2.4 Conduction of Action Potentials

The part of the axon where it emerges from the soma is called the axon hillock. Besides being an anatomical structure, the axon hillock is also the part of the neuron that has the greatest density of voltage-dependent sodium channels. This makes it the most easily excited part of the neuron and the spike initiation zone for the axon. The action potential generated at the axon hillock propagates as a wave along the axon. The currents flowing inwards at a point on the axon during an action potential spread out along the axon, and depolarize the adjacent sections of its membrane. If sufficiently strong, this depolarization provokes a similar action potential at the neighboring membrane patches.

Once an action potential has occurred at a patch of membrane, the membrane patch needs time to recover before it can fire again. At the molecular level, this absolute refractory period corresponds to the time required for the voltage-activated sodium channels to recover from inactivation. There are many types of voltage-activated potassium channels in neurons, some of which inactivate fast and some of which inactivate slowly or do not inactivate at all. This variability always guarantees the availability of a source of current for repolarization, even if some of the potassium channels are inactivated because of preceding depolarization. On the other hand, all neuronal voltage-activated sodium

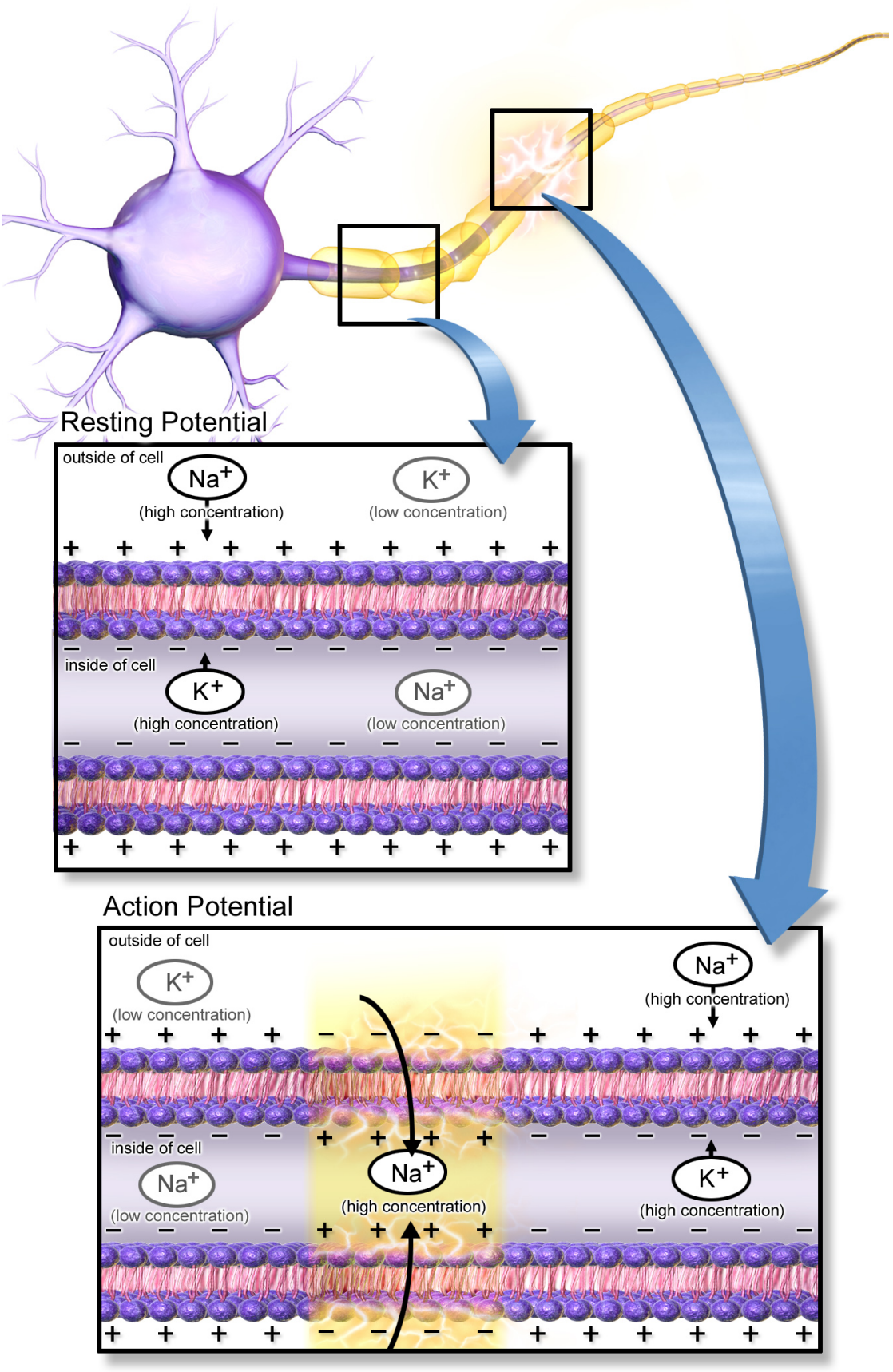

FIGURE 2.4: Conduction of an Action Potential towards an Axon Terminal (*Source: Wikimedia Commons*)

channels inactivate within several milliseconds during strong depolarization, thus making following depolarization impossible until a substantial fraction of sodium channels have returned to their closed state.

Although it limits the frequency of firing, the absolute refractory period ensures that the action potential moves in only one direction along an axon. The currents flowing in due to an action potential spread out in both directions along the axon. However, only the unfired part of the axon can respond with an action potential; the part that has just fired is unresponsive until the action potential is safely out of range and cannot restimulate that part. In the usual *orthodromic conduction*, the action potential propagates from the axon hillock towards the synaptic knobs. Propagation in the opposite direction, known as *antidromic conduction*, is very rare.

2.2.5 The Neocortex and Pyramidal Neurons

The average adult human brain weighs about 1.4 kilograms and most of it is covered with a fairly uniform outer layer called the *cortex*. With the exception of a small part of it that is older in terms of evolutionary history, the majority of the cortex is believed to have evolved at the time of the appearance of mammals. For this reason, this larger part is called the *neocortex*. Valiant chooses to primarily focus on the neocortex in his quest to model cognition due to the following encouraging factors –

(1) All the evidence points to the neocortex as being the main seat of memory and higher brain functions.

(2) Long distance connections are realized in the neocortex by one class of cells called *pyramidal neurons*, thus facilitating random access tasks that may need information from any part of memory.

(3) The neocortex is a part of the brain that has grown explosively in relation to most other parts since humans evolved from early primates, suggesting that it is organized along principles that scale well with size.

The human neocortex (shown in Figure 2.5), sometimes called the *gray matter*, is a layer of tissue having many convoluted folds, typically a little more than 2000 square centimeters in total area and a little more than 2 millimeters in average thickness. The cell bodies of the pyramidal neurons reside in this thin layer. Each of these cells has a very long axon that is about 0.0003 millimeters wide but up to several centimeters long. The axon typically leaves the gray matter near the cell body, passes through the

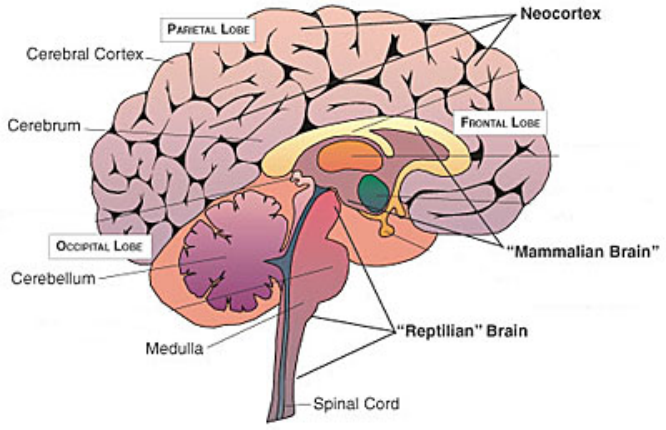

Figure 2.5: The Human Brain and its Regions (*Source: Wikimedia Commons*)

so-called *white matter* to a distant part of the neocortex (sometimes in the opposite hemisphere), re-renters the gray matter there and then splits into numerous branches to form an axonal branching. Thus, the white matter is best viewed as a cable box through which a vast amount of long distance communication is realized. The axonal branches carry the output of the computation performed by the neuron and form synapses with the dendritic trees of other neurons.

2.3 Cognitive Functions in the Brain

2.3.1 Cognitive Substrate

While providing a computational account of the working of the brain, the problem of specifying the cognitive functions presents the most formidable difficulties. For a complicated device such as the brain, it is difficult both to describe the total behaviour in its full complexity, as well as to decompose it into simpler constituents. There exist theoretical results[22, 23] in computational learning theory that show that even relatively simple computational mechanisms can result in behaviours that are so complex that a description of the mechanism or of the constituent parts of the behaviour cannot be recovered feasibly from observations of the behaviour itself. However, Valiant believes that there is some *substrate* of simple functions on which human cognition is built, which he hopes to discover by inspired guesswork.

In order to find candidates for the functions of the this cognitive substrate, one approach could be to study the results of experiments in cognitive psychology and to use them to identify the basic functionalities that underly behaviour. However, this appears to be

difficult in practice. Thus, Valiant[41] adopts a reverse approach that starts with some simple functionalities and then traces the implications that follow from them.

2.3.2 Boolean Functions, Conjunctions and DNFs

In 1854, Boole[5] constructed a mathematical system, now called Boolean algebra, that was explicitly motivated by questions of cognition. In this section, we discuss his system and explain why Valiant feels it to be a useful starting point.

A variable x in Boolean algebra can take values 'true' or 'false' rather than numerical values. This is because the intention of a Boolean variable is to represent a proposition. Thus, x can stand for "it is raining", and y for "it is cold". The intention of a Boolean operation is to create new propositions from old ones. For example, the operation 'and', denoted by $\wedge$, can be used to create the new proposition '$x \wedge y$' which in this instance would represent the proposition "it is raining and it is cold". In a similar way, the operation 'or', denoted by $\vee$, can be used to create the new proposition '$x \vee y$' which would represent the proposition "it is raining or it is cold or possibly both". A third operation is 'not', and it is denoted by $\neg$. So, '$\neg y$' would represent the proposition "it is not cold". While $\wedge$ and $\vee$ have two arguments, and are hence binary operators, $\neg$ is unary and has just one argument. Boolean algebra is concerned with the laws under which expressions formed by Boolean operations can be manipulated.

Perhaps the most important aspect of Boole's work is that it provides a model of cognition in which variables can take on only a discrete choice of values, in particular 'true' and 'false', rather than an unbounded choice. Though the merits of this central idea, that cognition should be modeled in terms of discrete mathematics, can be debated, it has withstood the test of time remarkably well and permeates current thinking. Valiant also readily adopts this view that discrete representations play a fundamental part in the substrate of principles on which human cognition is built, though he notes that by itself Boolean algebra is clearly an incomplete theory of cognition. However, he is reluctant to reject it simply for that reason, because he feels that the true "laws of thought" are of significant descriptional complexity and Boolean algebra captures only a part of it.

The most basic cognitive tasks to be considered are those of *recognition*. In Valiant's neuroidal model, these are implemented by circuits that evaluate the corresponding Boolean function or *predicate*. For some set of input variables, say x, y, z, the circuit will output 'true' or 'false' according to the value the function takes when supplied with

the truth values for x, y and z. The input variables may themselves be the outputs of Boolean functions and may express something complicated. In general, whenever a *scene* is presented to the system, the truth of the predicates are evaluated by the system for that scene.

A central question is to determine which classes of Boolean functions are most appropriate for modelling how knowledge is represented in the brain. In particular, an incremental version of this question that should be of particular interest is: if one new piece of knowledge is learned, what is the appropriate class of representations from which the act of learning selects? Classes that are too general become unrealistic if no plausible mechanism can be found for learning instances of them. On the other hand, classes that are too restrictive are unrealistic if they cannot express significant fragments of real world knowledge.

A simple but unavoidable class is that of *conjunctions*. A conjunction is simply the 'and' of a set of Boolean variables, such as $x_3 \wedge x_5 \wedge x_7$. The truth of every variable that appears in a conjunction is both necessary and sufficient for the conjunction to be true. Valiant[41] notes that conjunctions seem to be well-suited for representing single instances of objects or events, since they express the conjunction of all the relevant attributes. The notion of "yesterday's dinnertime guests" contains simultaneously the attributes 'yesterday', 'dinnertime' and 'guest'. Thus, it seems difficult to avoid expressing it as some kind of conjunction. For this reason, Valiant accommodates conjunctions centrally in his computational model for cognition.

On the other hand, it has been extensively argued that more general concepts require richer representation classes. Wittgenstein argued that the notion of a "game" has no attributes that are both necessary and sufficient. For example, not every game is won or lost, not every game is played by two people, and so on. For these reasons we need to go to more general representations. A most attractive generalization is *disjunctive normal form*, usually abbreviated to DNF. A DNF expression is a disjunction of conjunctions, such as:

$$(x_1 \wedge x_5 \wedge x_7) \vee (x_2 \wedge x_4 \wedge x_8) \vee (x_3 \wedge x_5 \wedge x_6)$$

DNFs can express concepts that have several distinct varieties of typical members. As an illustration, if we are allowed to write one conjunction to characterize one-person games and another one to characterize two-person games, then we could get a far better characterization of games with a DNF than is possible with a single conjunction.

One crucial point is that, though the conventional definitions of conjunctions and DNF allow negations of variables, Valiant tries to avoid using negations as much as possible. In Valiant's neuroidal model, the firing of a neuroid usually corresponds to a variable z being true. Now, if the need to represent its negation arises, then a separate neuroid is associated with a new variable, say y, which is logically equivalent to $\neg z$ and which will be true when the neuroid fires. However, on a few occasions, it becomes necessary to face the issue of negation or inhibition explicitly.

2.3.3 Learning Phenomenon

Learning phenomena have been classified and categorized in numerous ways. However, Valiant[41] specifically wishes to draw our attention to two dichotomies that appear to be fundamental in any context in which we wish to describe explicit mechanisms of learning.

The first dichotomy is between *memorization* and *inductive learning*. Memorization is simply the storage of some information that is explicitly provided or internally deduced. The information memorized may be, for example, the spelling of a word, or it may relate to the appearance of a person, the description of an event, or the result of a logical deduction. Inductive learning, on the other hand, can be defined as any kind of information gathering where the information acquired is not explicitly given or necessarily implied by that which is explicitly given. The common characteristic that the phenomena of inductive learning have is that some form of generalization is involved that is not dictated unquestionably by the evidence. When learning to recognize chairs from some examples, we acquire a capability that is somehow more general than the ability to merely recognize the particular examples of chairs that we have seen.

In conventional computers, memorization is a trivial operation. In the neural context, however, it raises challenging computational problems that can easily be underestimated. Of course, the challenge of modeling inductive learning is even greater, since learning involving generalization poses fundamental philosophical questions as to its very nature that memorization does not.

The second dichotomy is between *supervised* and *unsupervised* learning. Consider the process in which the learner is presented with some examples. In the case of supervised learning, the information describing each example is accompanied by information of a second kind called the *labeling*. The labeling could be provided by the teacher or deduced internally by the learner. Like, when learning the concept of a chair, a child is presented

with a sequence of examples each labeled as "chair" or "not chair". In unsupervised learning, on the other hand, information describing the examples alone is presented, with no additional commentary.

We can use these two dichotomies to describe four modes of learning and characterize each of them with examples as follows–

- *Unsupervised memorization* is appropriate for memorizing the sound or spelling of a word about which we have no previous knowledge.
- *Supervised memorization* is appropriate for associating the face of a person with a name.
- *Unsupervised inductive learning* is typically to do with spotting combinations of events or attributes that occur together unexpectedly often.
- *Supervised inductive learning* is appropriate for learning a concept or category such as 'chair' or 'honesty' from examples labeled as positive or negative instances of the given category.

In his computational model of the brain, Valiant aims to set up a neuroidal circuit for each of the learning processes that, when given inputs subsequent to the learning experience, will recognize the input according to the required function.

2.3.4 The Nature of Concepts

The nature of a concept, sometimes called a category or a universal, has always been a central subject of philosophical speculation. When we describe an object as a 'chair', we essentially mean that it belongs to a certain class. Valiant[41] feels that one should attempt to delimit more carefully the nature of the classes that humans employ as categories. While saying that concepts correspond to Boolean functions is definitely a step toward taking a position on this issue, Valiant argues that it is only a small one. Further, he thinks that a better question to ask would be whether there exist more detailed models of concepts that are more useful.

One major problem area is to determine which classes of computationally tractable knowledge representations are sufficient to represent human concepts. Is it Boolean conjunctions, or disjunctive normal form, or something else? Psychological experiments yield some clues here. For example, it has been found[37] that artificial concepts made up

by combining elementary concepts by "and" or "or" are much easier to learn by humans than those composed by the "exclusive-or" connective. Further clues are provided by the attempts of researchers in artificial intelligence to describe natural concepts formally.

Boolean concepts may have internal structure in several additional ways. The *exemplar* theory suggests that our representation of a chair consists of descriptions of particular chairs that we have seen. On seeing a new object, we compare it with these exemplars and see whether it is sufficiently similar to at least one of them. Of course, one big unanswered problem here is to describe measures of similarity that work for the whole range of human concepts. Related theories suggest that concepts are *graded*. Some chairs are classified by human subjects as more typical than others. Furthermore, this subjective meausure of typicality correlates with more objective ones, such as reaction times measured when subjects are asked to categorize objects. Yet another theory claims that there are a number of attributes that are each positive indicators of chairhood, and a chair is anything which satisfies at least a certain number of these.

Perhaps, it is not quite correct to view each concept as unitary. For example, although our notion of "India" is at some level a single Boolean function, it is more useful to view it as the interaction of maybe a large number of functions variously acquired by memorization or inductive learning. Thus, we may have in our mind, as a prototype, a map from an atlas viewed in childhood. In addition, we can clearly recognize outlines that are close enough, which is an ability acquired by inductive learning. Moreover, there may be many items of information acquired by memorization that are associated with these inductively learned functions; in this case, perhaps, the names and locations of cities. Thus, if we see an outline map that resembles India and a dot inside labeled Mumbai, the immediate computational reaction is probably not usefully viewed as the evaluation of the unitary concept of India, but rather as the recognition of a number of distinct predicates. On occasion, these may turn out to be inconsistent with each other, in which case we would need to resolve amongst them by other means.

2.3.5 Cognitive Psychology

Over the last century, a large body of experimental data has been collected regarding the cognitive performance of humans under various laboratory conditions. The phenomena studied have included memory, learning, attention, as well as numerous others. Though many of these results are robust and reproducible in the same way as are experimental results in the physical sciences, one glaring difference is that no global theories have

emerged that account for the broad ranges of phenomena and can make comparable predictions. To Valiant, it seems that a successful search for such global theories will need to be theory driven. One needs to start with theories that have the potential to be global. Then, experiments can resolve among the candidates. The neuroidal formulation about to be described in the next section is intended to facilitate a range of such theories. In developing the neuroidal model, Valiant[41] found to his surprise that several mechanisms that were introduced to overcome computational impediments corresponded closely to notions that already had wide acceptance in cognitive psychology. This section explores in detail each of these connections, which are as follows:

1. **Multi-Object Scenes:** The propositional predicates of Boolean algebra can be applied to a whole scene, but the issue we need to consider is how they are applied to parts of a multi-object scene. For example, in the normal interpretation when a picture is being described, the predicates blue or green could describe the whole picture. However, the question of how humans deal with the situation where the picture contains several distinguishable parts, one of which is blue and another green, has been the subject of careful investigation[38]. In a typical experiment, a human subject is presented with pictures that each contain a number of figures, such as green triangles, blue squares, etc. The subject is asked various questions and the time required for answering them is reliably measured. A typical finding is that when asked whether the picture contains an object having a single attribute, such as being green or being a triangle, the time taken is independent of the number of objects. The interpretation of this is that processing of all the objects is carried out in parallel, and perhaps a global propositional predicate, such as "there is a triangular object", is evaluated for the whole scene, in time independent of the complexity of the scene. In contrast, if an object having a conjunction of two attributes is sought, such as a green triangle, then the time taken to answer increases linearly with the number of objects, suggesting that the subject is, at some level, processing the objects in sequence. This sequential strategy is exactly the solution which is adopted in the neuroidal model to deal with multi-object scenes, chiefly because it appears to be the simplest general computational mechanism for this task.

2. **Illusory conjunctions:** Conjunctions of attributes in a part of a scene are typically not noticed if they have not been attended to. In experiments of the kind just described, subjects sometimes report having seen a green triangle and a red square when they have in fact been presented with a red triangle and a green square, but for too short a time to be able to attend to them separately. This phenomenon is called *illusory conjunctions*. Even in our everyday lives, we fail to

recall conjunctions that we have no motivation to have noticed. For example, very few people can recall which are the letters associated with a particular digit on a mobile keypad, although they have been exposed to these inputs innumerable times. Even single attributes need to be attended to to be remembered. That is precisely why people often find it difficult to recall the direction in which the head faces in particular denominations of currency notes. Such attentional mechanisms were incorporated into the neuroidal model by Valiant because they solve several computational problems effectively, and not in order to fit psychological data. In particular, Valiant worked under the assumption that the attentional system can identify meaningful constituent parts of the scene and attend to each of them in turn[31].

3. **Imagery:** In the neuroidal model, the view is taken that the sensory areas of the cortex process the perceptual inputs, and transform them to more and more abstract representations as the information is passed up to higher levels. Suppose, at some point, neurons corresponding to 'chair' fire. If by some internal deductions or associations the neurons corresponding to 'table' are caused to fire as a result, the firing of these 'table' neurons may cause some activity in the lower level sensory areas, similar to the activity induced by the sight of the table. This reverse flow of information corresponds to the act of *imagining* a table. Thus, while certain activity in the sensory areas would normally cause the 'table' neurons in the higher area to fire, the act of human imagination suggests that exactly the reverse is also possible and typical. Hence, that is also the viewpoint that has been adopted in the neuroidal model. The existence and location of such imagery areas have been the subject of detailed investigation by means of psychological experiments as well as brain scans[24, 25], and the results are not inconsistent with this viewpoint.

4. **Working Memory:** Psychologists have consistently differentiated *short term* or *working* memory[39] from long term memory. In the neuroidal model we have peripherals that correspond to imagery and working memory in order to empower the NTR to perform random access tasks and to store long term memory.

5. **Memory Capacity:** Experiments have been performed to explore the limits of memory capacity. In one such experiment, Standing[34] presented 10,000 pictures to a group of people over five days. Right afterwards, these people were shown a combination of new pictures and original pictures, and astonishingly, they correctly recognized the pictures they had previously seen with an accuracy rate of 83%. Such experiments, aimed at determining the quantitative limits of cognitive performance, are clearly of considerable relevance for resolving among detailed computational theories.

6. **Concept Formation and Learning Relations:** The issue of concept formation goes beyond simple memorization and is quite difficult to investigate. Inspite of that, several impressive studies have been carried out with pigeons. There is substantial evidence that, after seeing many examples of a concept, such as pictures that depict water in some form, the pigeons succeeded in generalizing appropriately, and they could classify previously unseen pictures according to whether the generalization learned holds or not[17]. Though it is difficult to isolate in laboratory experiments simple learning phenomenon that distinguish humans from other species, one area in which non-humans appear to have much more difficulty is that of learning relations[18]. Correspondingly, Valiant found that the handling of relations in the neuroidal model also introduces an extra level of difficulty.

7. **Learning Algorithms:** A prudent approach to identifying the knowledge representations and learning algorithms that are used by humans is through the study of language learning in children. Let us consider the question of whether children tend to overgeneralize of undergeneralize when using a word recently learned. Since many learning algorithms have a tendency to do one or the other, observations on humans can be used to rule out learning algorithms that are clearly inconsistent with observation. Current evidence[6, 26] suggests that overgeneralization is more prevalent in humans than undergeneralization.

8. **Pavlovian Conditioning:** Pavlovian conditioning is a striking phenomenon about which a large amount of experimental data has been accumulated[27, 30]. In a typical experiment demonstrating it, a subject has an air puff blown into an eye causing it to blink (or is prompted to perform some other reflex action), and at about the same time is also presented with a one of a wide variety of stimuli, such as the sight of a yellow square. It is observed that, if this procedure is repeated enough times, the subject becomes *conditioned*, so that at later times the presentation of only the yellow square causes blinking even in the absence of the air puff. This can easily be regarded as a random access phenomenon, since the range of stimuli one can substitute for the yellow square is quite large. Furthermore, instead of having just one ordinary stimulus such as the yellow square, one can have several. The conditioned response can then be made dependent on more than one such variable. It is tempting to speculate that the learning phenomenona associated with Pavlovian conditioning reflect some basic learning mechanisms at the neural level. Therefore, it is of considerable interest to try and relate them to learning algorithms.

9. **Priming:** Another example of a well-studied area of robust phenomena, is *priming*[40]. In a typical experiment, a human subject is given a list of fifty words to

read. Some time later, the subject is presented with some word fragments and asked to complete each one to make a word. Interestingly, it is found that the subject is more likely to reconstruct a word recently seen in the list, rather than an equally valid alternative, even when the subject cannot consciously recollect having seen that word. In the neuroidal model, there can be several alternative ways of incorporating priming effects. For example, Valiant suggests allowing recently increased synaptic weights to attenuate with time, so that the most recent changes always have greater relative influence when compared to earlier ones. This may be allowed only at low levels, or only at certain other levels of processing. Thus, various alternative models of priming can be constructed by experimentation and compared.

10. **Mixed Mode Representations:** Another question to ponder over is whether an item in memory is represented accurately in each mode separately. For example, we can ask: do there exist neurons for recognizing dogs within the vision area and each of the other sensory areas separately, or are the indicators that confirm doghood in the various modalities mixed together at a level preliminary to recognition by any one? More particularly, if we present a word fragment and a picture fragment, which separately are not enough to remind a person of the object to which they both refer, can the presentation of the fragments together do so? To Valiant, our innate ability to solve crossword puzzles suggests an affirmative answer to this question, which in turn makes him believe that objects do have mixed mode representations. Mixed mode representations also seem to be more economical in terms of representation and computation, and this would appear to compensate for the accompanying loss in precision.

2.4 Valiant's Neuroidal Model

2.4.1 The History of Artificial Neural Networks

An artificial neuron is a mathematical abstraction of biological neurons. Artificial neurons are the constitutive units in an artificial neural network. An artificial neuron receives one or more inputs (representing one or more dendrites) and sums them to produce an output (representing the axon in a biological neuron). Usually the sums of each node are weighted, and the sum is passed through a non-linear function known as an activation function or transfer function.

The first artificial neuron was the Threshold Logic Unit (shown in Figure 2.6), which was proposed by Warren McCulloch and Walter Pitts in 1943. As a transfer function, it employed the Heaviside step function, equivalent to applying a simple threshold. Since the beginning it was noticed that any Boolean function could be implemented by networks of such devices. This is easily seen from the fact that one can implement the AND and OR functions, and use them in the disjunctive or the conjunctive normal form. Initially, only a simple model was considered, with binary inputs and outputs, and some restrictions on the possible weights. Researchers also realized soon that cyclic networks, with feedbacks through neurons, could define dynamical systems with memory, but most of the research concentrated on strictly feed-forward networks because of the smaller difficulty they present.

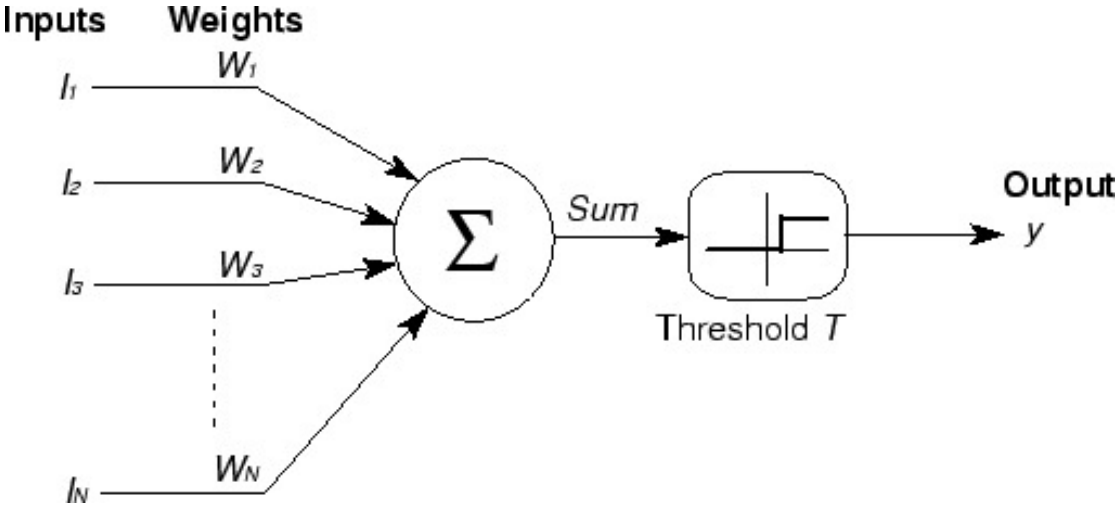

FIGURE 2.6: Threshold Logic Unit (*Source: Wikimedia Commons*)

One important and pioneering artificial neural network that used the linear threshold function was the perceptron[33], developed by Frank Rosenblatt in 1957. This model already considered more flexible weight values in the neurons, and was used in machines with adaptive capabilities. In the late 1980s, when research on neural networks regained strength, neurons with more continuous transfer functions such as the sigmoid function started to be considered. The possibility of differentiating the activation function allows the direct use of the gradient descent and other optimization algorithms for the adjustment of the weights while learning. Neural networks also started to be used as a general function approximation model.

2.4.2 Neuroidal Nets

In order to model the *neuroidal tabula rasa* (NTR), Valiant[41] defines an idealized model of a network of artificial neurons which he calls a *neuroidal net*. Each *neuroid*

in the net is defined to be a *linear threshold element*, as formalized by McCulloch and Pitts in 1943, but is augmented by states and a simple timing mechanism. However, as it turns out, these apparently innocuous augmentations render the model a lot more programmable and computationally powerful.

A *neuroidal net* is specified by a quintuple $(G, W, X, \delta, \lambda)$, where:

- G is the graph describing the topology of the underlying network
- W is the set of possible weights for the edges of the graph
- X is the set of modes for the neuroids
- δ is the update function for the modes
- λ is the update function for the weights

The subsequent few subsections will elaborate on each of the notions described above. Such a quintuple essentially provides us with a complete description of the net. In addition, if the *initial conditions*, i.e. the initial weights and modes of the neuroids, and *input sequence*, i.e. the timing and firing of those neuroids controlled directly by the peripherals, are specified, then the behaviour of the net is also determined.

2.4.3 Topology of the Network

In any neuroidal net, the topology of the underlying network is described by a directed graph $G = (V, E)$, where V is a finite set of N nodes labelled by distinct integers in $[N]$ and E is a set of directed edges between the nodes. Nodes i and j are said to be *adjacent* or *neighbours* if at least one of the directed edges (i, j) or (j, i) belongs to E. A neuroid corresponds to a node j together with all the edges directed *towards* it. An edge (i, j) models a synapse between two neurons where i is the presynaptic neuron and j is the postsynaptic neuron. By saying that a neuroid corresponds both to a node j as well as the incoming edges to j, Valiant essentially associates each synapse with its postsynaptic neuron.

2.4.4 Modes of a Neuroid

The *mode* of a neuroid describes every aspect of its instantaneous condition other than the weights on its edges. It models properties that are global for the neuron rather than relating only to particular synapses. It is specified as a pair of values $(q, \vec{T})$, where q

belongs to a finite set of states Q and $\vec{T}$ is a vector of γ numbers $T^{(1)}$, $T^{(2)}$,...,$T^{(\gamma)}$ for some fixed integer γ. The first component $T^{(1)}$ of $\vec{T}$, called the *threshold* of the neuron and denoted in short by T, models the electrical potential required to be overcome to cause the neuron to fire. Valiant allows for the possibility of $\gamma > 1$ in his model. For example, we may wish to include a second number that expresses the confidence in a generalization that has been learned inductively by a neuroid. However, when not stated otherwise, we shall assume that $\gamma = 1$. At any instant, the mode of neuroid i is denoted by s_i, its state by q_i, and its threshold by T_i.

The states bear names that are mnemonics for their function, such as:

- AM $\rightarrow$ Available Memory
- SM $\rightarrow$ Supervised Memory
- AR $\rightarrow$ Available Relay

The state space Q is partitioned into two kinds of states called *firing* and *queiscent* states. If a neuroid goes to a firing state, then it is a convention to update its state by appending the letter 'F' at the end of the mnemonic for its previous state. For each neuroid i, the Boolean variable f_i is defined as follows:

$$f_i = \begin{cases} 1 & \text{if neuroid } i \text{ is in a firing state} \\ 0 & \text{otherwise} \end{cases}$$

2.4.5 The Mode and Weight Update Functions

The action of the update functions δ and λ on neuroid i depend, among other things, on the quantity w_i, which is defined to be the sum of those weights w_{ki} of neuroid i that are on edges (k, i) that come from neuroids that are currently firing. More formally, we can write:

$$w_i = \sum_{(k,i) \in E, f_k = 1} w_{ki}$$

The mode update function δ defines, for each combination (s_i, w_i) that holds at time t, the mode $s_i' \in X$ that neuroid i will transit to at time $t + 1$, and is therefore written as:

$$\delta(s_i, w_i) = s_i'$$

The weight update function λ defines, for each combination (s_i, w_i, w_{ji}, f_j) that holds at time t, the weight w'_{ji} that edge (j, i) will bear at time $t+1$, and is therefore written as:

$$\lambda(s_i, w_i, w_{ji}, f_j) = w'_{ji}$$

The definitions of δ and λ express the following basic intentions–

(a) The updates to a neuroid should depend only on its own condition and that of neuroids from which it has incoming edges.

(b) The firing condition of these neighbours should be the only aspect of their instantaneous description that has any direct effect.

(c) The actual dependence on these neighbours should be through a linear sum of the form in which w_i is defined above.

Valiant allows updates to a neuroid to occur even when it is not firing itself, which helps to prevent unwanted cascades of firings. If every neuron that needed to be updated was forced to fire, then the effects of this additional activity would also have to be controlled while designing algorithms. Notice that, once G, X and W are defined for the NTR, an algorithm for a functionality is simply a specification of the update functions δ and λ as well as a description of the input sequence that has to be realized.

2.4.6 An Illustrative Example

As an example, suppose that we have a neuroid i that at time $t = 0$ is in a state we call A1, and has 5 incoming edges with different weights from its neighbouring nodes, as shown in Figure 2.7. Suppose that, independent of the initial value of the threshold T_i, we want that at time $t = 1$ the state of neuroid i be A2, the weights of the incoming edges from neighbours not firing at time 0 be set to 0, and that the new threshold T_i equal the sum of the weights of the incoming edges from all the neighbours that did fire at time 0. Figure 2.8 illustrates the updates that are required in one particular instance when the peripherals prompt neuroids 2,4 and 6 to fire at time 0.

The transitions below show how the algorithm that realizes these updates would be expressed within the model:

$$\forall T_i, w_i : \delta([A1, T_i], w_i) = [A2, w_i]$$

$$\forall T_i, w_i, w_{ji} : \lambda([A1, T_i], w_i, w_{ji}, 0) = 0$$

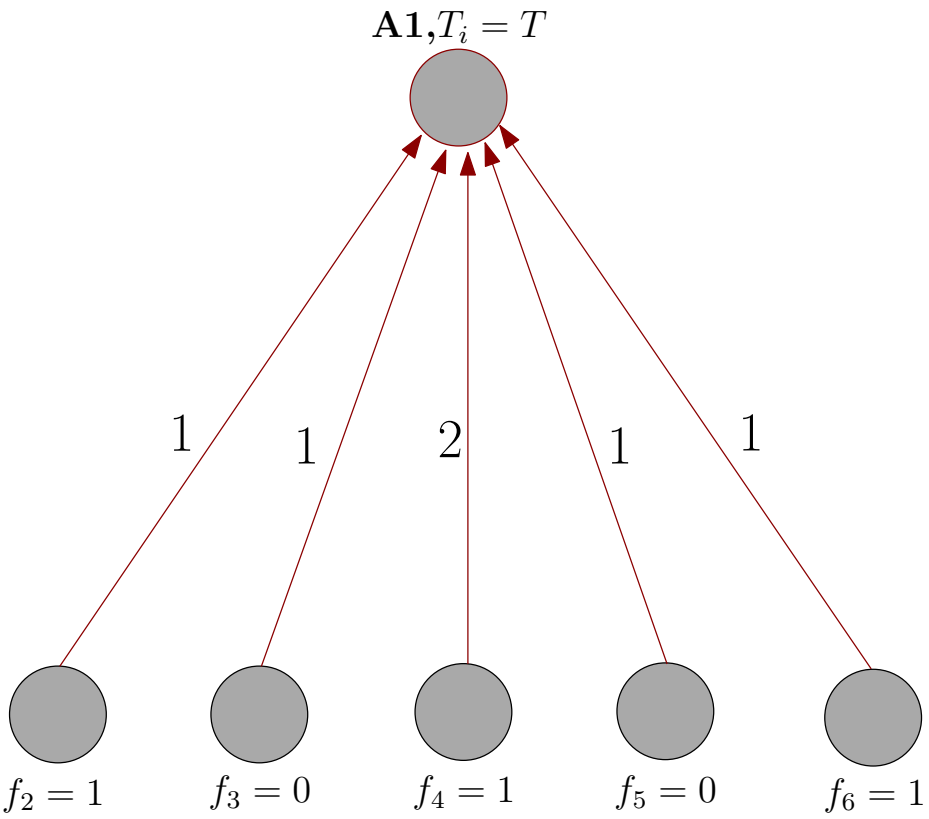

FIGURE 2.7: Initial Weights and Modes (at T=0) in Example Algorithm

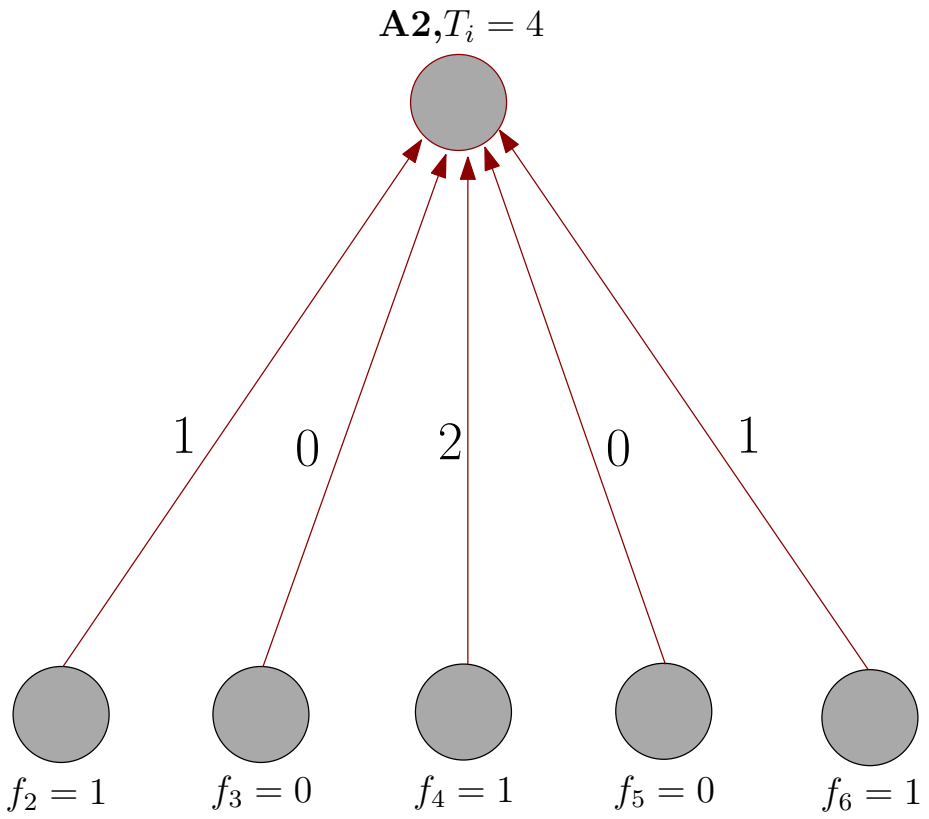

FIGURE 2.8: Updated Weights and Modes (at T=1) in Example Algorithm

When writing more elaborate algorithms, Valiant prefers, for the sake of clarity, a more succinct notation that groups together the transitions that are to occur simultaneously at a node. In this notation, the above transitions would be written as:

$$\{q_i = \text{A1}\} \Rightarrow \{q_i := \text{A2},\ T_i := w_i,\ \text{if } f_j = 0 \text{ then } w_{ji} := 0\}$$

The above algorithm enables node i to recognize at later times the same pattern of inputs that it was exposed to at time 0. If all the inputs that were firing at time 0 fire together again at a later time, then the same value of w_i will be achieved as at time 0, and the value of the threshold set at time 1 will be reached, which will cause the node to fire. The firing of additional inputs will not affect this outcome since the weights from these have been set to 0 at time 1. Also, once the node is in state A2, no further opportunities for weight changes are possible, unless more transitions are added.

2.4.7 Timing Mechanism and Neuroidal Algorithms

Timing is crucially important to Valiant's model. The peripherals have the power to cause various sets of neuroids in the NTR to fire simultaneously at various times. The actual choices of the sets and the times, which are called *prompts*, determine the input sequence. There is a substantial body of experimental evidence that suggests that synchronized rhythmic behaviour is a pervasive characteristic of the cortex[3, 11], and hence the hypothesized power of causing synchronous firings ascribed to the peripherals is not unreasonable. Between successive prompts from the peripherals, an algorithm is expected to perform only a few basic steps. The assumption of any global synchronization mechanism is not required, but Valiant[41] does suppose that the neuroids share common notions of a time unit, and hence can keep in synchrony for such short sequences of basic steps by following their own clocks. Even if their clocks keep slightly different times it is not problematic as long as they need to keep in step for only short periods. However, for simplicity, Valiant assumes in his model that the neuroids have exactly identical clocks.

In order to make programming manageable, neuroidal systems actually work with two very different time scales. The individual transitions δ and λ work on a scale of very short intervals called *microunits*, which roughly correspond to the switching times of biological neurons. For orchestrating computations in the NTR, the peripherals work on a longer time scale, measured in terms of *macrounits*. The peripherals are able to *prompt* the NTR by simultaneously causing to fire some subset of the neuroids in the NTR that are directly controllable by the peripherals. A cascade of computations on

the microunit time scale then ensues in the NTR, and we assume that this terminates in a stable situation before a full macrounit of time has elapsed. When this macrounit has elapsed, the peripherals may prompt the NTR again with the same or a different subset of neuroids.

In order to ensure that the cascade of firings in the NTR initiated by a prompt does indeed terminate in a stable situation, several alternative approaches can be taken. One major issue is that the computation performed in one cascade of the NTR needs to have an interpretation that is robust to the various demands that may be made on the NTR. For example, the peripherals may prompt low level neuroids, and the algorithm being executed may need to modify neuroids representing higher level concepts that are separated in the network from the prompted ones by several intermediate neuroids. We need to avoid situations in which an algorithm is executed incorrectly because the various input signals that were to arrive simultaneously had traversed paths of different lengths, and for that reason failed to arrive when needed. In order to keep the algorithms as simple as possible, Valiant wants us to assume that the implementation is equivalent to one in which a neuroid undergoes cascade transitions only when all the signals that need to arrive there have arrived. Clearly, a sufficient condition for this is that the graph formed by the neuroids that are actively involved in any cascade is acyclic, and all paths from the inputs to any one node are of the same length.

A *neuroidal algorithm* for a task is defined by Valiant to be a sequence of steps, each initiated by a prompt from the peripherals, and resulting in some updates to some neuroids. These prompts will be separated in time by an integral number of macrounits. Now it turns out that a convenient high-level description can be provided for neuroidal algorithms of interest to us, that convey their essence. In these descriptions, it is assumed that any neuroids that need to be prompted by the peripherals can be prompted directly by the peripherals, thereby suppressing the possibility that the prompted neuroids influence the ones taking part in the algorithm through a cascade of threshold firings. As a result, a macrounit can be equated with a microunit in the description of these algorithms. Thus, we can very conveniently describe all algorithms at this level on just one time scale. A more detailed level of neuroidal implementation, of the kind described in the previous paragraph, is assumed to support these high-level algorithms. Perhaps the simplest way to interpret such algorithms is to assume that the threshold transitions that occur in cascades take infinitesimal or zero time, while all other transitions take unit time, corresponding to a macrounit. In order to maintain consistency, Valiant therefore adopts the following conventions in the high-level descriptions of all neuroidal algorithms to be discussed in later sections: all transitions are assumed to

take unit time, with the exception of threshold transitions and prompts, both of which cause the corresponding nodes to fire instantaneously.

2.4.8 Discussions on the Model

Some points discussing the salient features of the model and some possible extensions to it are as follows–

(1) Since each synapse of a real neuron appears to be either excitatory or inhibitory, the weights in the model are assumed to have fixed sign, i.e. each w_{ji} is predetermined to be either non-negative or non-positive.

(2) In Valiant's model, the graph G describing the topology of the network can be generalized to a *multigraph*. So then, instead of there being at most one edge from node i to node j, there may be several, say k. In that case, we distinguish between them by $(i,j)^1, (i,j)^2, \ldots, (i,j)^k$, and their weights by $w_{ij}^1, w_{ij}^2, \ldots, w_{ij}^k$. Such a multiplicity of edges from node i to node j corresponds to the axonal branching of neuron i having k synapses with the dendritic tree of neuron j. Note that while two synapses of weight one may have the same effect as one synapse of weight two as far as the conditions that would make j fire, the update function λ may treat them differently.

(3) Another aspect of the model is that it can even allow randomized transitions δ and λ. This means that for each combination of argument values of δ and λ there may not be just one outcome at the next time unit, but several. A neuroid will choose randomly among these according to predetermined probabilities. While randomization has found applications in many areas of computation, it is not known whether it plays any role in the brain.

(4) The model can also allow each state $q \in Q$ to have a *latency* $l(q)$ that is a positive integer. If node i arrives in state q at time t, then neither its mode nor its weights can be changed until time $t+l(q)$ at the earliest. The latency is a timing mechanism corresponding roughly to the refractory period in real neurons. The updates to the mode and weights of a neuroid at time t is determined entirely by the description of the net at time $(t-1)$, unless it happens that at time t a neuroid has not completed the latency period it entered most recently, in which case no update will occur. A state that has latency l can be easily simulated by a sequence of l states each of latency one, which have the property that they each transition in unit time to the next state in the sequence, independent of all other conditions. Hence, allowing states to have differing rather than the same latencies does not increase the power of the model, but may allow for more succinct descriptions of some algorithms.

(5) For economy of descriptions, *threshold transitions* are assumed by default. Such a transition occurs whenever all three of the following conditions hold: $w_i \geq T_i$, there is no other explicitly stated transition which is available, and neuroid i is not within a latency period. In such a threshold transition, only the state of a neuroid is updated and the update changes the state to one with the same mnemonic but with an 'F' appended, indicating that it is a firing state. Threshold transitions typify the process by which circuits recognize inputs, while the other transitions typically modify weights or thresholds for the purpose of learning.

(6) The definitions provided by Valiant are not the minimal ones that achieve a certain expressive power. They incorporate some redundancy in order to provide for ease of expression, which is useful if the model is to be used as a programming language. For example, the dependence of λ on w_i is redundant in the sense that the value of w_i can be stored as a $T^{(k)}$ component of the mode s_i, and updated by means of δ transitions in a subsequent step. Transformations such as this, that leave the expressive power of the model invariant, provide evidence of the robustness of the model, since we know that, if a model is robust, then its expressive power should be preserved under reasonably wide ranges of mutations to the model.

2.5 Knowledge Representation

2.5.1 Representation of Items

Since a neural system can cope in a complex external world, one must presume that its behaviour can be described in terms of the various semantic items that are meaningful in that world. By the term *item* we mean just about any aspect of the world that may be useful in describing it. Individual objects, events, properties, relationships, concepts, and categories are all examples of items. Valiant[41] finds it plausible that the items that are appropriate for describing the world of experience of the NTR provide the right vocabulary for describing its behaviour when it interacts with that world. However, a harder problem is to determine whether these same items also provide the most appropriate internal vocabulary for the neural system.

There exist some theories that posit localist representations of items[2], which claim that each item in the world that can be recognized by an individual correspond to only some particular neurons that recognize it. At the other extreme, there exist theories that posit global (or holographic) representations of items[19], which claim that the representation of each item is spread over many or even all of the neurons. Various positions that are

intermediate have also been taken[9].

Valiant emphasizes that the choice of knowledge representation is a most central issue in analyzing models of neural computation. Instances of the representation have to be expressible succinctly in the neuroidal model. Also, there has to be a plausible account of how these structures can be learned.

2.5.2 Positive Knowledge Representations

The representations that Valiant[41] adopts in his model, called *positive* representations, can be characterized in terms of the following five features –

(1) Each neuron corresponds to a single semantic item.

(2) There are typically several neurons representing each item, if it is represented at all.

(3) Only those new items are added to memory that are experienced and noticed by the attentional mechanisms. For example, we do not have neurons to represent all possible combinations of car makes and colours that we are already familiar with. Suppose we notice a black Volkswagen on the road one day. Unless a black Volkswagen has figured prominently in our life in the past, neurons for both the colour black and Volkswagen will fire, but there may not be neurons previously allocated for the combination.

(4) The representation is *hierarchical*. Once some items have been assigned to neurons, new items expressible in terms of the items already represented can be assigned to previously unused neurons. However, the hierarchy need not be strict in the sense that cyclical relationships are forbidden. It is possible that once two related items have been assigned, their meaning is refined in terms of each other. The emphasis is more on the idea that some items are high level, being satisfied for very specific inputs, while others are lower level and of greater generality, being satisfied for wider ranges of input. High level items are typically represented in terms of combinations of lower level items.

(5) The reality of the neuroidal implementations makes it *graded*, so that only approximations of any idealized Boolean functions are represented. One reason for this is that the physical neural connections necessary to realize the ideal function will be present only with high probability, and not with certainty. A second reason is that sometimes there may not be any simple idealized function to realize. For example, a circuit for recognizing a chair may involve subcircuits for various types of chairs so that the overall circuit has no simple characterization.

In the neuroidal model, the set of neuroids representing an item z is usually denoted by $\tilde{z}$. Typically there are about r neuroids corresponding to each item, where r is called the *replication factor*. The basic intention is that, when the system is presented from the outside with an input corresponding to item z, then all the r neuroids in the set $\tilde{z}$ should fire.

Replication ensures robustness, as well as a slightly lower connectivity requirement on the network than would be needed otherwise. Sets or assemblies of cells to represent a concept has been used by several authors[7, 16] before, usually with the implication that within each such set or assembly each member has an excitatory influence on the others. Such assemblies would have a self-imposed tendency toward an all-or-none firing behaviour. However, Valiant makes no such assumption in his representation.

Valiant[41] envisages that, in practice, any grading in the individual circuits is compensated for by having multiple representations for important items, so as to improve the overall system reliability. For example, the nodes corresponding to the item 'chair' may be associated with several circuits, each of which attempts to find confirmation of chairhood in a different way. Now, if a reasonable fraction of these succeed, then it can be assumed that overwhelming evidence of chairhood has been found.

2.5.3 Vicinal Algorithms

All of the algorithms that Valiant describes as part of his model can be considered, at a suitable level of abstraction, as vicinal or neighbourly. The most basic feature of a *vicinal* algorithm is that, whenever some communication has to be established between two items not directly connected, the algorithm establishes the necessary communication via neuroids that are each common neighbours of some pair of neuroids that represent the two items respectively. Thus, if $\tilde{x}$ and $\tilde{y}$ correspond to two items, and $E(\tilde{x})$ and $E(\tilde{y})$ are the sets of neighbouring nodes of $\tilde{x}$ and $\tilde{y}$ respectively, then $E(\tilde{x}) \cup E(\tilde{y})$, known as the *undirected frontier* of $\tilde{x}$ and $\tilde{y}$ (shown in Figure 2.9), is the set through which communication takes place. Because of the primacy they give to communication via common neighbours we call these algorithms vicinal.

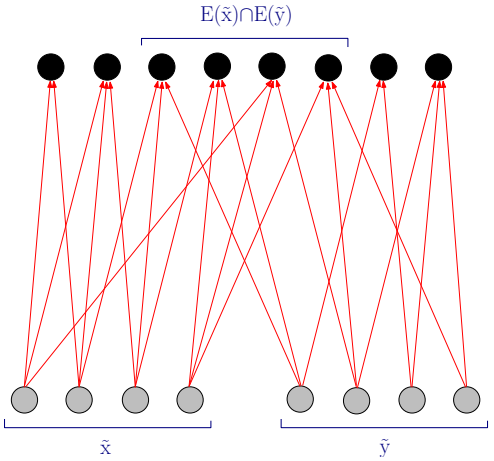

FIGURE 2.9: The Undirected Frontier of $\tilde{x}$ and $\tilde{y}$

Some algorithms may require that a two-way channel of communication be established, if temporarily, between pairs of nodes. This can be done most easily by requiring that the edges be *bidirectional*, in the sense that a directed edge $(i, j) \in E$ has an associated reverse edge (j, i) also belonging to E. The algorithms that exploit bidirectionality can be interpreted as demonstrating the computational efficacy of point-to-point feedback. Whether such bidirectionality between individual neurons is pervasive in the cortex is currently unknown. However, it is well established that for most pathways linking one cortical area to another there are reciprocal pathways going in the opposite direction. What is unknown is whether the reciprocity is precise enough to realize bidirectional edges directly.

2.5.4 Random Graphs

The simplest model that supports vicinal algorithms directly are *random graphs*. Random graphs have two important properties that are necessary for supporting these algorithms, namely a certain *frontier* property, and a certain *hashing* property. *Random mulipartite graphs* also have very similar frontier and hashing properties, and hence are equally good for supporting vicinal algorithms. However, for the sake of greater simplicity, Valiant assumes the former model.

The main difference between the models is that while the first treats all nodes as equal, the second splits them into sets, each of which corresponds to a different area of the cortex. In both cases the edges model long distance communication between pyramidal neurons in the cortex. In the multipartite case we assume that certain pairs of areas are connected, and those that are have random connections between them. It is interesting

to note that random multipartite graphs resemble some of the hierarchical structures that have been found in visual cortex[10].

2.5.5 Frontier Property

Since vicinal algorithms establish communication between the sets representing two items through the frontier of these two sets, this frontier should be non-empty in general. Furthermore, since it is the aim of some of the algorithms to store a new item at this frontier, so that the new item becomes an equal citizen with the others, one also needs that this frontier be of size about r, where r is the replication factor. The operation of allocating frontier nodes for storing a conjunction $x \wedge y$ is called JOIN by Valiant.

A graph $G = (V, E)$ is said to have the (r, l, m)-frontier property if, when $\tilde{x}, \tilde{y} \subseteq V$ are randomly chosen disjoint subsets of size r, the size of the frontier $E(\tilde{x}) \cup E(\tilde{y})$ has expectation l and variance m. A graph that is ideal for executing vicinal algorithms should have $l = r$ and $m = 0$.

Let us consider a random graph on N nodes such that for each pair of nodes (i, j), a directed edge joining i to j is present with probability p, independent of all other pairs. Then the expected indegree of each node is the same as the expected outdegree and equals pN. We regard r as a fixed constant, N as varying and large, and p as diminishing as N grows. Let $\tilde{x}$,$\tilde{y}$ be disjoint sets of r nodes. Then the probability that any one fixed node i, not belonging to $\tilde{x}$ or $\tilde{y}$, lies in the frontier of $\tilde{x}$ and $\tilde{y}$ is $(1-(1-p)^r)^2$. To see this, first observe that the probability that $i \notin E(\tilde{x})$ is $(1-p)^r$, since $(1-p)$ is the probability that it fails to be connected to any fixed member of $\tilde{x}$, and there are r such members. The probability that it is connected to at least one node in $\tilde{x}$ is, therefore, $(1-(1-p)^r)$. Since the same statement holds for $\tilde{y}$, the square of this expression gives us the desired probability. Note that we are assuming here that the edges coming to i from different nodes are either present or absent independent of each other.

Both in this section and the subsequent section, we shall make the following assumptions, which we will call the *pristine conditions assumptions*, while estimating probabilities–

(i) The number of neuroids representing any one item already stored is exactly r.

(ii) The edges directed toward neuroids not yet allocated are present with equal probability independently.

Regarding r as fixed, if $p \to 0$ as $N \to \infty$, then by application of the binomial theorem to $(1-(1-p)^r)^2$, we get:

$$\begin{aligned}
&(1-(1-p)^r)^2 \\
&= 1 - 2(1-p)^r + (1-p)^{2r} \\
&= 1 - 2(1 - rp + r(r-1)p^2/2 + O(p^3)) + (1 - 2rp + 2r(2r-1)p^2/2 + O(p^3)) \\
&= r^2p^2 + O(p^3)
\end{aligned}$$

Thus, for each node i not belonging to $\tilde{x}$ or $\tilde{y}$, the probability of being in their frontier $E(\tilde{x}) \cup E(\tilde{y})$ is given by $p^* = r^2p^2 + O(p^3)$, and is independent for different choices of i. Hence, the size of the frontier is governed by a binomial distribution consisting of $N^* = N - 2r$ trials, each with probability of success p^*. If we choose $p = (Nr)^{-\frac{1}{2}}$, then we obtain that the number of nodes in the frontier has:

$$\begin{aligned}
\text{Expectation} &= N^*p^* \\
&= (N-2r)(r/N - O(N^{-\frac{3}{2}})) \\
&= r - O(N^{-\frac{1}{2}})
\end{aligned}$$

$$\begin{aligned}
\text{Variance} &= N^*p^*(1-p^*) \\
&= (N-2r)(r/N - O(N^{-\frac{3}{2}}))(1 - r/N + O(N^{-\frac{3}{2}})) \\
&= r - O(N^{-\frac{1}{2}})
\end{aligned}$$

Thus, we conclude that a random graph on N nodes with $p = (Nr)^{-\frac{1}{2}}$, or expected degree $(N/r)^{\frac{1}{2}}$, has the $(r, r - O(N^{-\frac{1}{2}}), r - O(N^{-\frac{1}{2}}))$-frontier property. For implementing vicinal algorithms, random graphs with expected degree $(N/r)^{\frac{1}{2}}$ appear therefore to be well-suited.

Since we intend to make the NTR capable of learning hierarchically, it is pertinent to ask what happens when the process of creation of new frontiers from sets that were previously frontiers themselves is repeated to arbitrary depth. Unfortunately, it turns out that the variance of this process is too large to maintian stability over a large number of iterations. Experiments suggest, however, that for $r = 50$, this process is maintainable in such random graphs to depth 4 or 5. It is not clear whether there exist graphs that have frontier properties that have lower variance and hence are more stable. It is known, however, that for $r \geq 2$, the ideal $(r, r, 0)$-frontier property is not possessed by any bidirected graph with more than $3r$ nodes[13].

2.5.6 Frontier Property and Associations

Besides memory allocation, the common neighbours of pairs of neuroids also play an important role in vicinal algorithms in establishing *associations* between arbitrary pairs of items already stored. If node sets $\tilde{x}$ and $\tilde{z}$ are already allocated, we may wish to update the network so that at later times, whenever $\tilde{x}$ fires, so will $\tilde{z}$. Valiant calls this update operation LINK.

If we suppose that $\mu \geq 0$ is some constant, then in a random graph with edge probability $p = (\mu/(rN))^{\frac{1}{2}}$, the expected number of members of $\tilde{z}$ that have a common neighbour with at least one member of $\tilde{x}$ is about $r(1 - e^{-\mu})$, assuming that $\tilde{x}$ and $\tilde{z}$ have size r. For example, if $\mu = 1$, then about 63% of the $\tilde{z}$ nodes will be connected to some member of $\tilde{x}$ via a common neighbour, while if $\mu = 4$, then the fraction is as large as 98%.

To prove the claim about the number of members of $\tilde{z}$ that have a common neighbour with $\tilde{x}$, let us first consider the probability that there is a path of length two to one fixed member, say $i \in \tilde{z}$, from at least one member of $\tilde{x}$ via a fixed node j. Since the probability that at least one member of $\tilde{x}$ is connected to j is $(1 - (1 - p)^r)$, and the probability that j is connected to i is p, the probability of both happening is $p(1 - (1 - p)^r)$. Using the binomial theorem, this can be written as $p(rp + O(p^2))$, which on substituting for p gives $\mu/N + O(N^{-\frac{3}{2}})$. Hence, the probability that there is such a path via at least one of the $N - 2r$ possible choices of j is:

$$1 - (1 - \mu/N + O(N^{-\frac{3}{2}}))^{N-2r}$$

Using the fact that $(1 - \frac{1}{x})^x \to e^{-1}$ as $x \to \infty$ we obtain that this probability is $1 - e^{-\mu}(1 + O(N^{-\frac{1}{2}})$. Since this probability holds for each $i \in \tilde{z}$, the expected number of members of $\tilde{z}$ having the required property is r times this quantity, which approaches $r(1 - e^{-\mu})$ asymptotically as $N \to \infty$.

Therefore, when establishing an association from the representatives of one item $\tilde{x}$ to the representatives of another $\tilde{z}$, a greater value of μ implies that, on expectation, a greater fraction of the nodes of $\tilde{z}$ will be reached successfully. However, values for μ greater than one perturbs the frontier node allocation process described in the last section, since we will end up allocating about μr rather than r nodes to each new item. This can be counteracted by having the node allocation process reject each node provisionally allocated with probability μ^{-1} uniformly at random. In this way, both the processes of JOIN and LINK can be supported on the same network.

2.5.7 Hashing Property

The *hashing* property is required to ensure that the nodes chosen for representing a new item will be, with high probability, among those not previously chosen. Since the only mechanism used for allocating new storage is that of assigning a frontier $E(\tilde{x}) \cup E(\tilde{y})$ to a new item that is associated with the conjunction $x \wedge y$, the property that needs to be ensured is that for any choice of x and y, this frontier contains a sufficient number of previously unallocated neuroids which will be able to represent the new item effectively.

As in conventional hashing, Valiant assumes that only a certain constant fraction of all the neuroids are ever assigned. To justify the plausibility of this assumption, he observes that a human living for 100 years and having 10^{10} available neurons will be able to allocate up to 10^3 new neurons each hour without more than 10% of the memory ever filling up. This holds even in the absence of any provisions for freeing memory or forgetting.

Let us consider the situation in which a set of items is already stored and a new one is to be allocated. Now, we allocate the frontier of some appropriate pair of sets $\tilde{x}$ and $\tilde{y}$ to store the new item. This pair is quite arbitrary, though perhaps restricted to a certain depth in the network. We need to show that if the graph was randomly chosen then with high likelihood it can accommodate the new item in the sense that the frontier will contain a significant number of nodes that are still available for allocation.

Valiant[41] does a very approximate calculation using the pristine conditions assumptions (mentioned in section 2.5.5) and assuming $\mu = 1$ for simplicity. Let us consider $\tilde{x}$ and $\tilde{y}$ both to have size r, assume that there are $N(1-u)$ nodes still unallocated for some constant u (where $0 \leq u \leq 1$), and assume that each potential edge to any fixed unallocated node i, from any fixed $\tilde{x}$ or $\tilde{y}$ node, is present with probablility $p = (Nr)^{-\frac{1}{2}}$ independent of the presence of other edges. Then, exactly as before, the probability that node i is adjacent to some node in $\tilde{x}$ and some node in $\tilde{y}$ is $(1-(1-p)^r)^2 = r^2p^2 + O(p^3)$. Hence, if there are $N(1-u)$ choices of i, then the expected size of the frontier of $\tilde{x}$ and $\tilde{y}$ that is unallocated is given by:

$$\begin{aligned} & N(1-u)(r^2p^2 + O(p^3)) \\ = \; & N(1-u)(\frac{r^2}{Nr} + O((Nr)^{-\frac{3}{2}})) \\ = \; & r(1-u) + O(N^{-\frac{1}{2}}) \end{aligned}$$

The advantages achieved by using such a hashed memory can be summarized as follows. Regarding r as a constant, suppose that we have $(u/2)N/r$ items already stored, for some small enough constant $u < 1$, and some sequence of another $(u/2)N/r$ new items come along, each one expressible as the conjunction of a pair previously memorized. Although the new pairs may be viewed as chosen from many more pairs potentially, and although only N neuroids are available, we can nevertheless allocate memory for this arbitrary sequence of new items that present themselves, at least with probability. To reiterate, the main point is that arbitrary new items drawn from quadratically many possibilities can be allocated without needing quadratically many nodes.

2.6 Unsupervised Memorization - A Case Study

Valiant[41] describes several algorithms within the neuroidal model in which to tries to capture the essence of various cognitive tasks such as supervised and unsupervised memorization, supervised inductive learning, correlational learning etc. This section attempts to convey the general flavour of these algorithms by selecting and presenting Valiant's algorithm for unsupervised memorization as a case study.

2.6.1 Problem Specification

Unsupervised memorization appears on the surface to be a simple task. Following one presentation of an input, changes need to take place in the neuroidal system so that if an identical or similar enough input is presented in the future, the neuroidal system will recognize this repetition.

Despite its apparent simplicity, this task poses the very fundamental problem of storage allocation. Since the instance to be memorized may be unanticipated and arbitrary a mechanism is required for allocating storage space to essentially arbitrary new items. Since Valiant's model employs a positive knowledge representation, the process of storage allocation can be treated as one of identifying some previously unused neuroids and committing them to the purpose of representing the newly encountered item.

Since memorization can be related to forming Boolean conjunctions, Valiant considers the following idealization of the task –

- An input has a number of attributes that correspond to items $x_1, x_2, ..., x_n$ that are represented in the NTR by neuroid sets $\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_n$. We regard these sets as all having size r, and regard the neuroids within any one of these sets as having identical behaviour, i.e. at any time either all fire or none do.

- When this input is presented, the nodes in all the sets $\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_n$ will be caused to fire by the peripherals.
- The objective is to allocate a new set of neuroids $\tilde{z}$ to represent the new item z that corresponds to this input, and to update the net in such a way that in future interactions the nodes $\tilde{z}$ will fire if and only if $\tilde{x}_1, \tilde{x}_2, ..., \tilde{x}_n$ all fire simultaneously.
- Further, the underlying network is assumed to be a random graph having the $(r, r - O(N^{-\frac{1}{2}}), r - O(N^{-\frac{1}{2}}))$-frontier property.

2.6.2 Algorithmic Approach

Valiant[41] attempts to solve the problem of learning 2-conjunctions first, since the more general case of learning n-conjunctions can easily be reduced to this case. Thus, he starts off by specifying an algorithm that will allocate to the item $z = x \wedge y$ the frontier $\tilde{z} = E(\tilde{x}) \cup E(\tilde{y})$ and enable the neuroids in $\tilde{z}$ to modify themselves so that in future they will all fire whenever both $\tilde{x}$ and $\tilde{y}$ fire simultaneously.

The initial conditions for the algorithm about to specified (as shown in Figure 2.10) –

(i) The neuroids that implement unsupervised memorization are all initially in *available memory* (**AM**) state.

(ii) All the incoming edges to these neurons have weight 1 initially.

(iii) The threshold of all the neuroids in the state (**AM**) is effectively infinite, or in other words, higher than the number of incoming edges. Thus, the neuroids cannot undergo threshold firing until after they are allocated and have changed to *unsupervised memory* (**UM**) state.

A formal description of the algorithm follows:-

Step 0:

Prompt $\tilde{x}$.

$\{q_i = \text{AM}, w_i \geq 1\} \Rightarrow \{q_i \leftarrow \text{AM1}, T_i \leftarrow w_i, \text{if } f_j = 1 \text{ then } w_{ji} \leftarrow 2\}$.

Step 1:

Prompt $\tilde{y}$.

$$\{q_i = \text{AM1}, w_i \geq 1\} \Rightarrow \{q_i \leftarrow \text{UM}, T_i \leftarrow T_i + w_i,$$
$$\text{if } f_j = 0 \wedge w_{ji} = 1 \text{ then } w_{ji} \leftarrow 0,$$
$$\text{if } f_j = 0 \wedge w_{ji} = 2 \text{ then } w_{ji} \leftarrow 1\}.$$
$$\{q_i = \text{AM1}, w_i < 1\} \Rightarrow \{q_i \leftarrow \text{AM}, T_i \leftarrow \infty, \forall j \; w_{ji} \leftarrow 1\}.$$

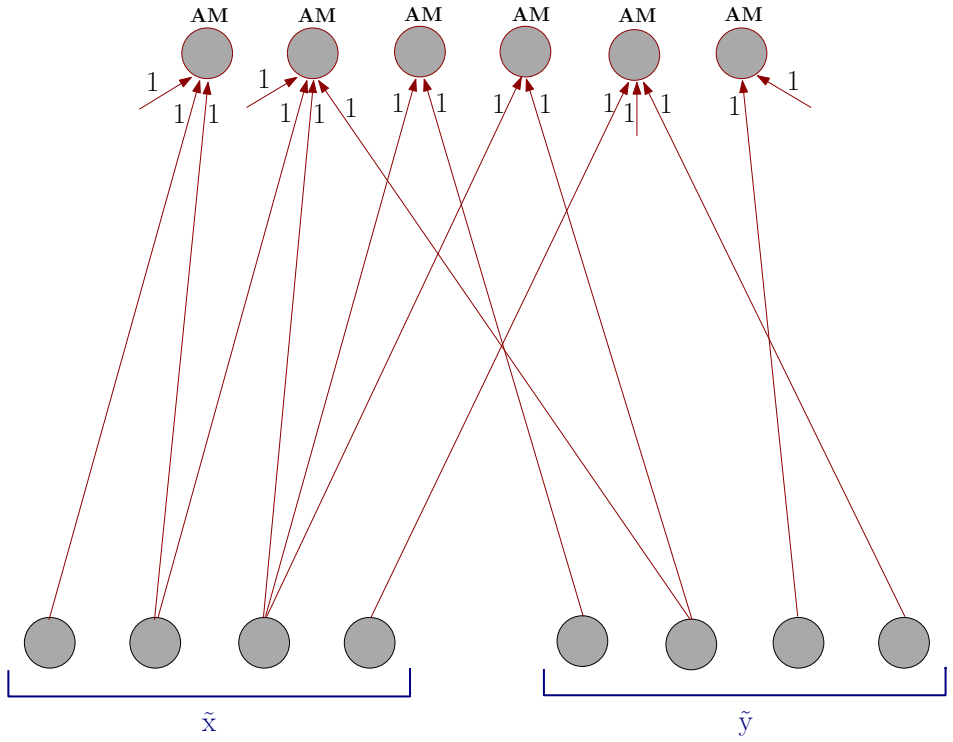

FIGURE 2.10: Initial Conditions at T=0 for Unsupervised Memorization

While analyzing the above algorithm, keep in mind that –

(i) The conditions in a rule at step t always refer to time t, but the updates mentioned happen only at time $t+1$.

(ii) Default threshold transitions are instantaneous but are implied to exist only for nodes that do not satisfy the precondition in any explicitly stated rule.

(iii) Any node which is firing state at a particular instant spontaneously ceases to fire after one unit of time has elapsed.

In the above algorithm, the $\tilde{x}$ nodes fire in the first step. As shown in Figure 2.11, all **AM** nodes adjacent to a member of $\tilde{x}$ go into state **AM1**, record the value of w_i in T_i, and update the weights on the incoming edges from $\tilde{x}$ to 2, leaving the values of the others as 1. In the second step the $\tilde{y}$ nodes fire. By this time, the $\tilde{x}$ nodes have ceased firing by default. As shown in Figure 2.12, those **AM1** nodes that are adjacent also to some $\tilde{y}$ node now go into unsupervised memory state (**UM**), update their threshold to equal the total number of $\tilde{x}$ and $\tilde{y}$ nodes to which they are adjacent, and adjust their weights so as to be 1 on edges coming from the $\tilde{x}$ and $\tilde{y}$ nodes, and to be 0 on all other incoming edges. The remaining **AM1** nodes, those not adjacent to any of the $\tilde{y}$ nodes, revert to the original mode and weights of available memory neuroids. Therefore, this

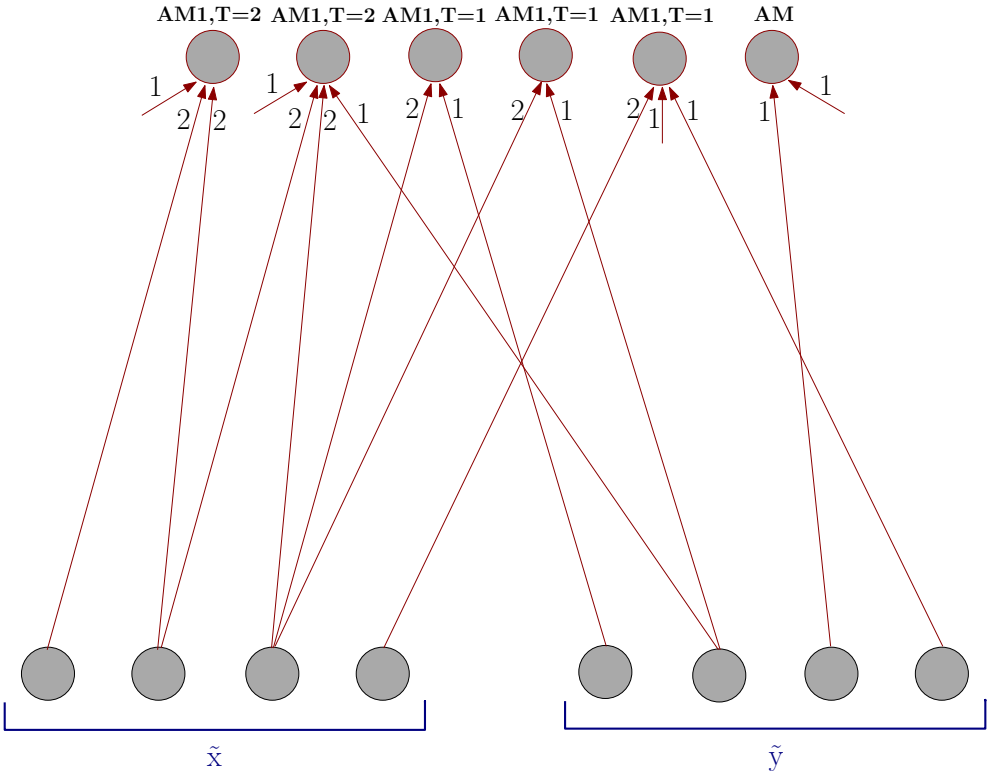

FIGURE 2.11: Weights and Modes at T=1 for Unsupervised Memorization

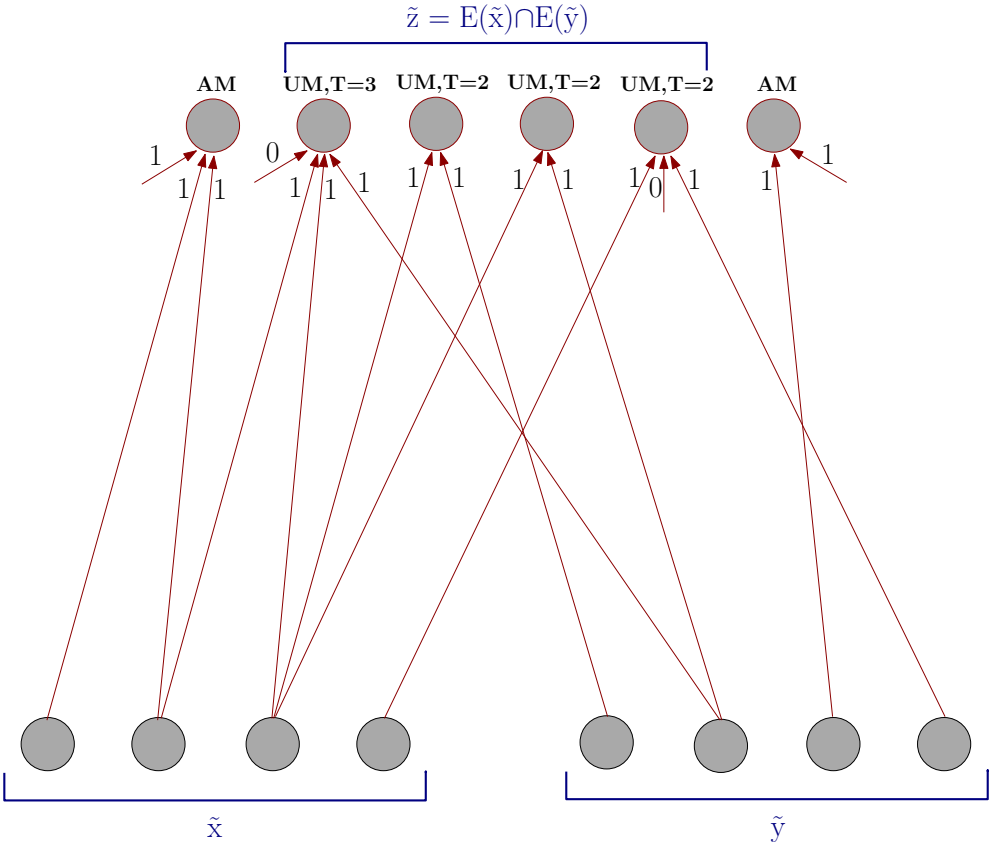

FIGURE 2.12: Weights and Modes at T=2 for Unsupervised Memorization

algorithm does indeed perform unsupervised memorization of 2-conjunctions.

For learning conjunctions of arbitrary length, the previous algorithm can be used iteratively as a subroutine, as long as the peripherals do appropriately more work. To learn $x_1 \wedge x_2 \wedge x_3$, for example, the peripherals would find some ordering on these 3 items, such as x_1, x_2, x_3. They would then supply x_1 and x_2 in a call of the algorithm for learning 2-conjunctions so that a new item $z_1 = x_1 \wedge x_2$ is memorized. Finally, they would supply z_1 and x_3 in a second call of that algorithm so that $z = z_1 \wedge x_3 = x_1 \wedge x_2 \wedge x_3$ is memorized. By supplying z_1, we mean that $\tilde{z}_1$ will be made to fire, which will be achieved by causing $\tilde{x}_1$ and $\tilde{x}_2$ to be fired simultaneously. It is easy to see that this strategy can be iterated so that conjunctions of any length n can be learned, though the time needed for learning would increase linearly with n.

2.6.3 Discussions

A few points that merit discussion are as follows –

- In the algorithm described above, the **AM1** nodes do not undergo threshold firings since, when alternative transitions are available, those are invoked instead. This allows T_i to be used to memorize a number rather than to represent a real threshold when a node is in state **AM1**.
- The avoidance of inessential firings can be important for an algorithm, since this also minimizes the side-effects. In the above algorithm, for example, if the **AM1** nodes had been allowed to fire then these would have caused a cascade of unintended storage allocations of a nature similar to the intended one.
- The strategy of allocating only previously unused neuroids to represent a new item makes it possible for learning to be cumulative, in the sense that unrelated items in memory will be left alone by any one execution of unsupervised memorization.
- All items, once allocated, are treated as equal citizens. Items represented by nodes that are allocated by the mechanisms described here play the same role as items represented by nodes that are controlled directly by the peripherals.
- The hierarchical memory allocation process implies a hierarchical view of the knowledge that can be stored, where items are defined relative to each other rather than being defined absolutely. However, any item already stored can be learned in supervised mode in terms of any of the others. Hence circular relationships among the items may develop that do not respect the hierarchy of the original allocation process.

2.7 Concluding Remarks

The brain is remarkably energy efficient and can carry out computations that challenge the world's largest supercomputers, even though it relies on decidedly imperfect components: neurons that are a slow, variable, organic mess. Comprehending language, conducting abstract reasoning, controlling movement — the brain does all this and more in a package that is smaller than a shoebox and consumes only around 20W of power[8, 14], less than a household light bulb. Therefore, in recent times, researchers have been directing a lot of effort towards ushering in a new computing paradigm called cognitive computing, whose aim is the development of computer systems modeled after the human brain. Cognitive computing[29] integrates technology and biology in an attempt to re-engineer the brain, one of the most efficient and effective computers on earth. An early instantiation of a cognitive computing chip has been developed under the Darpa SyNAPSE program at IBM directed by Dharmendra Modha[28, 35]. The neuroidal model lays down a robust theoretical framework which acts as a very good guideline for building cognitive computing systems in the future.

Valiant feels that the enormous descriptional complexity of the brain may be due as much to the variety of mechanisms incorporated as to the intricacy of any one of them. Valiant knew that in order to describe such a variety of algorithms one needed a suitably expressive language. Therefore, he purposefully designed a highly *programmable* model, so that it could support a greater variety of tasks than any previous modelers appear to have attempted within a single system. Moreover, the broad flexibility of the model allows us to investigate wide classes of hypotheses together.

In any model such as the one proposed by Valiant, two opposing constraints need to be reconciled. First, the model needs to be simple enough so that there is little question that real cortical neurons are at least as powerful computationally. Then, any algorithm devised for neuroids can be construed as an existence proof that the corresponding functionality can indeed be supported by cortical neurons. The second constraint on the model is that it has to capture the essence of the computational capabilities of the brain, at least for implementing random access tasks. The direct relevance of Valiant's model to real neural computations rests on the fact that it does seem to satisfy these two opposing constraints simultaneously. In fact, Valiant provides us with a computational scheme that supports cognitive functions such as memorization, association, inductive learning and is feasible on networks of model neurons that respect the widely observed values of the 3 quantitative parameters of the neocortex: the neuron number, the synapse number and the switching times. Moreover, the algorithms are simple and most of them require just one step of vicinal or neighbourly influence.

Bibliography

[1] Miklós Ajtai. $\sum_1^1$-formulae on finite structures. *Annals of Pure and Applied Logic*, 24:1–48, 1983.

[2] H.B. Barlow. Single units and sensation: a neuron doctrine for perceptual psychology. *Perception*, 1:371–394, 1972.

[3] Erol Başar and Theodore Holmes Bullock. *Induced Rhythms in the Brain.* Birkhäuser, 1992.

[4] Paul Beame. A switching lemma primer. *Technical Report 95-07-01, Department of Computer Science and Engineering, University of Washington*, 1994.

[5] George Boole. *An Investigation of the Laws of Thought.* 1854.

[6] M. Bowermann. The acquisition of word meaning: an investigation of some current concepts. In *Thinking: Readings in Cognitive Science.* Cambridge University Press, 1977.

[7] V. Braitenberg. Cell assemblies in the cerebral cortex. In *Theoretical Approaches to Complex Systems, Lecture Notes in Biomathematics 21*, pages 171–188, 1978.

[8] Guy C. Brown. *The Energy of Life: The Science of what Makes Our Minds and Bodies Work.* Free Press, 1999.

[9] J.A. Feldman. Computational constraints on higher neural representations. In *Computational Neuroscience.* MIT Press, 1990.

[10] D.J. Felleman and D.C. van Essen. Distributed hierarchical processing in primate cerebral cortex. *Cerebral Cortex*, 1:1–48, 1991.

[11] W.J. Freeman. *Mass Action in the Nervous System.* Academic Press, 1975.

[12] Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.

[13] A. Gerbessiotis. *Topics in Parallel and Distributed Computation.* PhD thesis, Division of Applied Sciences, Harvard University, Cambridge, MA, 1993.

[14] Leslie Hart. *How the Brain Works*. Basic Books, New York, 1975.

[15] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *STOC*, pages 6–20, 1986.

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

[17] R.J. Herrnstein. Riddles of natural categorization. *Philosophical Transactions of the Royal Society of London B*, 308:129–144, 1985.

[18] R.J. Herrnstein. Levels of stimulus control: A functional approach. *Cognition*, 37:133–166, 1990.

[19] J.J. Hopfield. Neural networks and physical systems with emergent computational abilities. *Proceedings of the National Academy of Sciences*, 79:2554, 1982.

[20] Russell Impagliazzo, William Matthews, and Ramamohan Paturi. A satisfiability algorithm for AC^0. In *SODA*, pages 961–972, 2012.

[21] Eric R. Kandel, James H. Schwartz, and Thomas M. Jessell. *Principles of Neural Science*. Elsevier, 1981.

[22] Michael J. Kearns and Leslie G. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *Journal of the ACM*, 41(1):67–95, 1994.

[23] Michael Kharitonov. Cryptographic hardness of distribution-specific learning. In *STOC*, pages 372–381, 1993.

[24] Stephen Michael Kosslyn. *Image and Mind*. Harvard University Press, 1980.

[25] Stephen Michael Kosslyn and Olivier Koenig. *Wet Mind*. Free Press, New York, 1992.

[26] Yonata Levy, Izchak M. Schlesinger, and Martin D.S. Braine. *Categories and Processes in Language Acquisition*. Erlbaum, Hillsdale, NJ.

[27] James E. Mazur. *Learning and Behavior*. Prentice Hall, Englewood Cliffs, NJ, 1990.

[28] Paul Merolla, John V. Arthur, Filipp Akopyan, Nabil Imam, Rajit Manohar, and Dharmendra S. Modha. A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm. In *CICC*, pages 1–4, 2011.

[29] Dharmendra S. Modha, Rajagopal Ananthanarayanan, Steven K. Esser, Anthony Ndirango, Anthony Sherbondy, and Raghavendra Singh. Cognitive computing. *Communications of the ACM*, 54(8):62–71, 2011.

[30] Ivan P. Pavlov. *Conditioned Reflexes*. Oxford University Press, 1927.

[31] M.I. Posner and S.E. Peterson. The attention system of the human brain. *Annual Review of Neuroscience*, 13:25–42, 1990.

[32] Alexander A. Razborov. An equivalence between second order bounded domain bounded arithmetic and first order bounded arithmetic. In *Arithmetic, Proof Theory and Computational Complexity*, page 247–277. Oxford University Press, 1993.

[33] Frank Rosenblatt. The perceptron – a perceiving and recognizing automaton. *Report 85-460-1, Cornell Aeronautical Laboratory*, 1957.

[34] L. Standing. Learning 10,000 pictures. *The Quarterly Journal of Experimental Psychology*, 25:207–222, 1973.

[35] Jae sun Seo, Bernard Brezzo, Yong Liu, Benjamin D. Parker, Steven K. Esser, Robert K. Montoye, Bipin Rajendran, José A. Tierno, Leland Chang, Dharmendra S. Modha, and Daniel J. Friedman. A 45nm CMOS neuromorphic chip with a scalable architecture for learning in networks of spiking neurons. In *CICC*, pages 1–4, 2011.

[36] J.G. Sutcliffe. mRNA in the mammalian nervous system. *Annual Review of Neuroscience*, 11:157–198, 1988.

[37] S.J. Thorpe, K. O'Regan, and A. Pouget. Humans fail on XOR pattern classification problems. In *Neural Networks from Models to Applications*. IDSET, Paris, 1989.

[38] A. Treisman and G. Gelade. A feature-integration theory of attention. *Cognitive Psychology*, 12:97–136, 1980.

[39] Endel Tulving. *Elements of Episodic Memory*. Oxford: Clarendon Press, 1983.

[40] Endel Tulving and D.L. Schacter. Priming and human memory systems. *Science*, 247:301–306, 1990.

[41] Leslie G. Valiant. *Circuits of the mind*. Oxford University Press, 1994.