Connectionist Models Applied to Automatic Speech Recognition

Yoshua Bengio
Department of Computer Science,
McGill University, Montreal

November 1987

A thesis submitted to
the Faculty of Graduate Studies
in partial fulfillment of the
requirements for the degree of M.Sc.

ISBN 0-315-45992-1

Connectionist Models Applied to Automatic Speech Recognition

Abstract

Connectionist models and their application to Automatic Speech Recognition are studied in this thesis. In particular, the Boltzmann machine algorithm and the error back propagation algorithm are presented in detail, and are used in a specific application to speaker normalization and the recognition of the place of articulation for sonorant sounds. Spectral lines are used to represent static speech information into the connectionist networks. The networks learn to classify vowels into three categories: back position, central position and front position. The recognition rate is shown to be dependent on how speech data is coded at the input of the networks. The best results are obtained using coarse and multi-level coding. The recognition rates are significantly better than results on similar data, also using spectral lines information, but with Hidden Markov Models alone.

Resumé

Les modèles connexionistes et leur application à la reconnaissance automatique de la parole sont étudiés dans ce mémoire. En particulier, l'algorithme de la machine de Boltzmann et l'algorithme de rétro-propagation de l'erreur sont présentés en detail. Ceux-ci sont utilisés dans une application à la normalisation des locuteurs, pour la classification des voyelles en fonction de leur lieu d'articulation en trois catégories: position arrière, centrale et frontale. Les lignes spectrales du signal auditif sont extraites puis fournies à l'entrée des réseaux connexionistes. Il est montré que le taux de reconnaissance dépend de la manière dont les lignes spectrales sont codées à l'entrée des réseaux. Les meilleurs resultats sont obtenus avec un codage à multiple niveaux et l'activation d'unités voisines. Les taux d'erreur obtenus sont significativement meilleurs que ceux obtenus avec des données similaires, aussi representées par leur lignes spectrales, mais utilisant les seuls modèles de Markov cachés.

Table of Contents

page

1. Introduction 1
2. Overview of Connectionist Models 3
   - 2.1 From Neurobiology 4
     - 2.1.1 Operation of the Neuron 4
     - 2.1.2 About Learning 5
   - 2.2 Historical Perspective 6
     - 2.2.1 Formal Neurons 6
     - 2.2.2 The Perceptron 9
     - 2.2.3 NETL 12
   - 2.3 Taxonomy 12
   - 2.4 Current Models 15
     - 2.4.1 The Hopfield Net 15
     - 2.4.2 A Regularity and Correlation Detector 17
     - 2.4.3 Competitive Learning 21
3. The Boltzmann Machine Algorithm 25
   - 3.1 Constraint Satisfaction 25
   - 3.2 Learning 28
   - 3.3 Derivation of Gradient Descent Rule 31
   - 3.4 Implementation of the Boltzmann Machine for the Experiments 34
     - 3.4.1 Temperature Cooling Schedule 36
     - 3.4.2 Other Parameters 37
4. The Error Back Propagation Algorithm 39
   - 4.1 Description and Derivation 40
   - 4.2 Parameters and Implementation for the Experiments 43
5. Essential Problems in Automatic Speech Recognition 48
   - 5.1 Physiology of the Human Auditory System 49
     - 5.1.1 The Ear 49
     - 5.1.2 Auditory Nervous System 49
   - 5.2 Automatic Speech Recognition 52
     - 5.2.1 Pattern Classification 52
     - 5.2.2 Fast Fourier Transform 54
     - 5.2.3 Dynamic Time Warping 55
     - 5.2.4 Hidden Markov Models 56

6. Experiments in Automatic Speech Recognition Performed with Connectionist Models 58
6.1 Spectral Lines Extraction 59
6.2 Coding 62
6.3 Experimental Results 67
6.4 Comparison with Results Using Hidden Markov Models 69
7. Direction for Future Work 72
7.1 Larger Neural Nets and Integration of Several Modules 72
7.2 Incorporating the Time Dimension of Speech in Connectionist Models 73
7.2.1 Window in the Input 73
7.2.2 Recurrent Links with Delays 76
7.2.3 Context Feedback 77
7.3 Realization of Connectionist models in Hardware 79
8. Conclusion 81
References 83

List of Figures and Tables

page

Figure 2.1 Example of a McCulloch-Pitts formal neuron.....7
Figure 2.2 A simple network of formal neurons.....8
Figure 2.3 Decision regions of a perceptron network.....11
Figure 2.4 Structure of the Hopfield net.....16
Figure 2.5 Architecture of a competitive learning mechanism.....23
Figure 3.1 Input, output and hidden layers for the Boltzmann machine.....26
Figure 3.2 Flow diagram of high level structure of the implemented Boltzmann machine algorithm.....35
Figure 4.1 Feedforward layers of the error back propagation algorithm.....40
Figure 4.2 Flow chart diagram of high level structure of implemented back propagation algorithm.....46
Figure 5.1 Model of auditory sensitivity.....51
Figure 6.1 Spectrogram of pronunciation of the letter 'a' before spectral lines were extracted.....60
Figure 6.2 Spectral lines extracted from pronunciation of the letter 'a'.....61
Figure 6.3 Multi-level energy coding.....63
Figure 6.4 Coarse coding.....64
Figure 6.5 Learning; error rate for the training set *vs* number of full learning cycles for the BMA.....69
Figure 7.1 Window in the input stream.....75
Figure 7.2 Context feedback with carry units.....78

Table I : Error on test set for various coding schemes.....66
Table II : Speed and best results for the two algorithms.....67
Table III : Comparison with HMM algorithm, same data.....70

1. Introduction

The human auditory system is able to solve the very difficult problem of speech perception. Speech of poor quality, distorted by noise, with inter-speaker and intra-speaker variabilities is easily recognized by humans, using the numerous phonemic, lexical, syntactic and semantic constraints built into speech. Since phonemes cannot be described by an invariant set of acoustic features, and cannot be unambiguously segmented, automatic recognition of connected speech (especially for multiple speakers) is a very complex task which remains to be satisfactorily solved.

Artificial and adaptable neural networks (also known as connectionist models, or parallel distributed processing models) have recently attracted considerable attention in the field of Artificial Intelligence (AI). Connectionist models seem particularly well suited to problems in automatic vision and speech perception. Connectionist models were studied in the 50's and 60's but research in this field was discouraged by a pessimistic evaluation of the potential of the perceptron by Minsky and Papert in 1969. However, in recent years connectionist models have been proposed which solve the problems and limitations of the perceptrons pointed out by Minsky and Papert. In addition, neural networks can very easily be implemented through massively parallel architectures in hardware. Available VLSI technology and the trend towards parallel computers thus makes neural nets technologically very attractive.

Despite their name, neural nets are generally not plausible models of neuron operation. Instead, they are inspired from neurobiological cues. The first of these indications is the massively parallel architecture, made up of very simple computational elements, each controlling very little memory. In

contrast to the Von Neumann architecture, where the computational power (CPU) is physically separate from the memory, in the connectionist models processing elements and memory elements are distributed and close to each other. This results in the capacity for the connectionist models to bring to bear a very large amount of knowledge simultaneously when solving a problem. This is especially useful in perception, where huge amounts of data must be examined in a very short time (using very few machine cycles) if real time recognition is desired.

This thesis will study connectionist models, including a brief neurobiological and historical overview and the description of various models (chapter 2). The general problem of Automatic Speech Recognition will be considered along with a short description of the human auditory system (chapter 5). For this thesis, experiments were performed with neural networks to classify sonorant portions of connected speech in order to achieve a speaker-independent recognition of the place of articulation. These experiments were based upon two popular connectionist models models: the Boltzmann machine algorithm (described in detail in chapter 3) and the error back propagation algorithm, or multi-layer perceptron (chapter 4). In these experiments the speech spectral information was computed and represented at the input of the connectionist models using spectral lines. The results of these experiments as well as their methodology, including the spectral line extraction technique, the coding of data into the neural nets and a comparison of the results with experiments performed on similar data using Hidden Markov Models are presented in chapter 6. In chapter 7 the possible direction of future research is indicated and additional aspects of the application of connectionist models to automated speech recognition are considered. In particular, we consider the problem of the representation of the sequential nature of speech in neural nets.

2. Overview of Connectionist Models

Current symbolic, heuristic, and logical AI works well for high level mental functions but is not so successful for problems such as automatic visual or speech perception. Connectionist models (also known as artificial neural networks, or parallel distributed processing models) have the potential of achieving better performance in these fields. Recent interest in connectionist models results from the new VLSI techniques available, new more powerful connectionist models, and also from the huge computing requirements of image or speech recognition. The connectionist models can be implemented in massively parallel architectures, with computational elements (or nodes) connected via "weights". In a neural net, memory and processing power are both distributed and close to each other. In opposition to the Von Neumann machine CPU which has to look at each passive piece of information of memory in sequence, the neural net architecture can bring a large amount of knowledge to bear simultaneously. Units of connectionist systems are densely interconnected but send signals consisting of very few bits (typically a single bit marker or a continuous scalar value). This should be contrasted with the parallel architectures which use message passing. The latter machines usually require more complex processing units with more storage and longer messages. Because connectionist models generally use a distributed representation (each concept is represented not by one unit but by the pattern of activation of many units, cf. sections 2.3 and 7.3), they are fault-tolerant. If a small random set of units fails, the performance of the network is only slightly affected (graceful degradation)(Rumelhart & al., 1986). However, this distributed representation makes the internal representation used by these networks hard to understand and modify manually. Therefore these networks need an automatic

learning mechanism to incorporate new knowledge. As such, the locus of learning in connectionist models is the pattern and strength of connections, hence the name connectionism.

2.1 From Neurobiology

The human brain is an organ made of 10^{10} to 10^{12} neurons, that uses a small number of stereotyped signals. It is far less homogeneous (and thus far more complex) than other organs: there are different types of neurons and they are connected to each other in a very complex way. These connections are called **synapses**. At chemical synapses, the specialized ending of a neuron ("presynaptic terminal") is very close (about 50 nm) to a postsynaptic **dendrite** of another neuron. Through these dendrites, the neuron collects signals coming from other neurons which are of two electrical types: one for short distances and one for long distances. These signals have the same form among neurons of the body, as well as in most animals with a nervous system.

2.1.1 Operation of the Neuron

The signals arriving at a synapse (action potentials) travel along the axon (which may be very long, up to about 1 m for example in nerve fibers). They are series of impulses of constant amplitude, travelling at constant speed from the body of the originating neuron to the presynaptic terminals at the junction with other neurons.

When an impulse arrives at a synapse, a certain quantity of neurotransmitter is released. On the postsynaptic terminal, these transmitter molecules are detected and this results in a local postsynaptic potential (PSP). This is translated into an increase

in electrical potential if the synapse is **excitatory**, a decrease if it is **inhibitory**.

The postsynaptic potentials from various dendrites travel towards the body of the neuron (usually close to the dendrites) and sum up with other postsynaptic potentials. This summation is called integration, it involves both a spatial (from the simultaneous action of several synapses) and temporal component (sequence of impulses coming from a single synapse).

If the postsynaptic potential increases above a critical level called threshold, an enormous increase in potential occurs very rapidly, creating an impulse (action potential) which can spread through its axon towards other neurons, without loss of amplitude (unlike the PSP, since it is continuously regenerated). After an impulse, there is a period of low potential called the refractory period (of the order of the millisecond) during which a second impulse cannot be initiated, therefore limiting the maximum frequency of firing.

Note that the PSP is a continuously valued and local quantity whereas the action potential is of a binary nature (although the frequency of firing may vary continuously) and can travel long distances.

2.1.2 About Learning

Neurobiologists now believe that the mechanisms of memory must be seeked at the level of the synapses. In 1949, Hebb proposed a synaptic modification mechanism. His ideas are generally accepted but have yet to be definitely proven experimentally. Specifically, he proposed that

"when an axon of cell A is near enough to excite a cell B and repeatedly and persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells, such that A's efficiency, as one of the cells firing B, increases."

This means that repeated coactivation of two cells increases the synaptic strength ; a synapse growth would be related to the correlation of the pre-synaptic and post-synaptic cells activities.

Experiments show that activity of neurons result in ultrastructural changes in synapses. For example, in kittens reared in the dark and then exposed to vision for several hours, the number and the density of synaptic vesicles (discrete units containing neurotransmitters) increases significantly (Weisel & Hubel, 1963). In another experiment, the brains of rats experiencing either an enriched environment (A) or an impoverished environment (B) are compared. The cerebral cortex of rats "A" is thicker and weighs more than the cerebral cortex of rats "B". However, the enriched brain does not contain more neurons, only more dendrites, and therefore more synapses (Watson, 1976, pp.191-192),(Volkmar & Greenough, 1972).

2.2 Historical Perspective

2.2.1 Formal Neurons

In 1943, McCulloch and Pitts (McCulloch & Pitts, 1943) presented a highly simplified neuron model. An example of the McCulloch and Pitts formal neuron is presented in figure 2.1. The formal neuron is a boolean processing element with m inputs $x_1,..,x_m$ and one output y. It is characterized by its bias *b* and the m weights $w_1,...,w_m$. The module determines its output in function of its inputs on a discrete time scale t=1,2,3...

$$y(t+1) = 1 \text{ iff } \sum_i w_i x_i(t) > b$$
$$\text{otherwise } y(t+1)=0. \qquad (2.1)$$

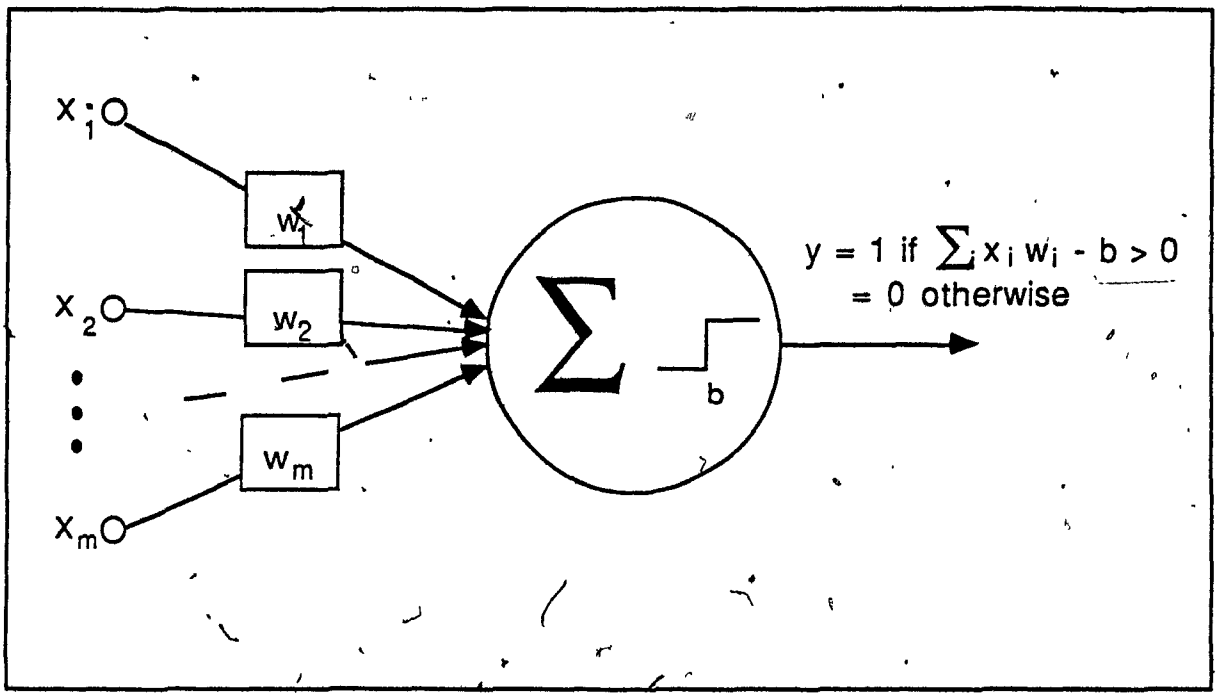

Figure 2.1 : *Example of a McCulloch-Pitts formal neuron. The activation function is a hard limiting threshold.*

In correspondence to real neurons, the positive weights will be said to be excitatory, and the negative weights inhibitory (cf. excitatory and inhibitory synapses, section 2.1.1)

In terms of this definition of the formal neuron, a formal neural net can be defined as a collection of formal neurons with the same time scale, interconnected by sending the output of each formal neuron to the inputs of several other formal neurons. These networks can be represented by a digraph (directed graph), as shown in figure 2.2.

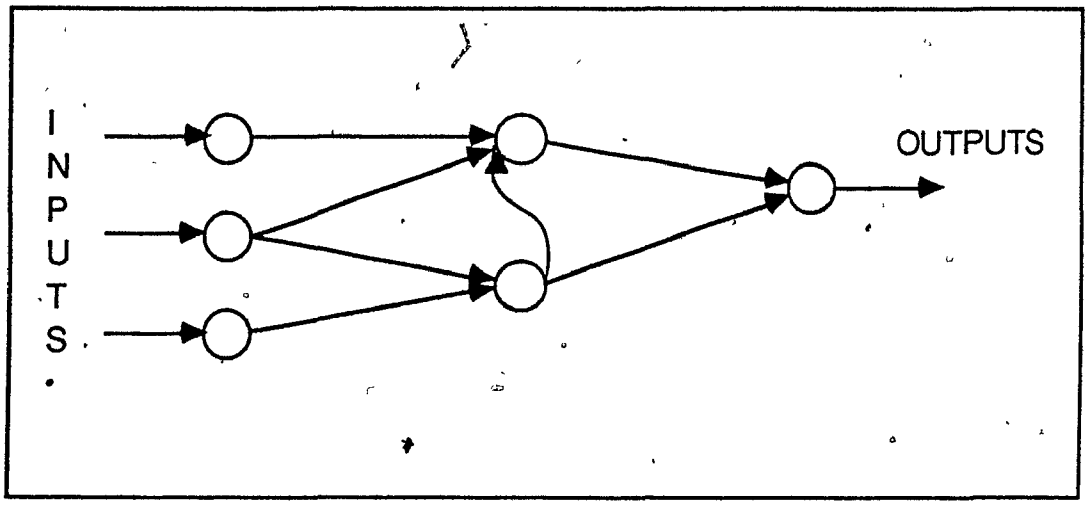

figure 2.2 : *A simple network of formal neurons*, where each neuron computes its boolean output in function of its boolean inputs as defined in equation 2.1. Note that this network has no loop but connectionist networks in general can have some.

Formal neural nets were shown to be finite automata, and conversely, it was shown that it is always possible to construct a formal neural net that has the input-output behavior of a given finite automaton (for example see Arbib, 1964). A corollary to this argument is that formal neural nets are equivalent to digital computers (by using some formal neurons for storage). Thus formal neural nets can compute any digital function. Formal neural nets can be "programmed", by filling in the values of weights and the description of the structure of the network. However, they would be much more useful if they could learn (adapt) instead of having to be programmed. The first ideas in this direction came from Donald Hebb's *Organization of Behavior* (1949). He proposed the first learning rule for synapses: synapses should be strengthened when both the presynaptic and postsynaptic units were active simultaneously (section 2.1.2). His idea that the weights of the connections should depend on the correlated activity of the connected nodes still persists today in current learning rules, even though the details of the rules may differ. The next step in the development of a learning rule came with the work of Frank Rosenblatt on the learning procedure for the perceptron.

2.2.2 The Perceptron

Rosenblatt introduced the perceptron in 1959, and presented a learning rule with a proof of convergence in his book, *Principles of Neurodynamics* (1962). The perceptron is very much like the formal neuron except that it has modifiable weights. The perceptron was thought as a pattern recognition device that has some ability to "learn" to recognize patterns of a set after a finite number of trials. The main result of Rosenblatt's work is summarized in his learning rule and its convergence theorem :

> "Given an elementary alpha-perceptron, a stimulus world W, and any classification C(W) for which a solution exists; let all stimuli in W occur in any sequence, provided that each stimulus must reoccur in finite time; then beginning from an arbitrary initial state, an error correction procedure will always yield a solution to C(W) in finite time,..."(p.596)

The learning rule is called an "error-correction" or "reinforcement rule", and is defined as follows: if the response of a perceptron to a given stimulus is correct, then nothing is changed. If the response is incorrect, the weights are changed. I.e. :

$$\Delta w_i = \text{constant} \times (\text{target\_output} - \text{actual\_output}) \times \text{actual\_input}_i \quad (2.2)$$

For units which accept continuous inputs, the learning rule in (2.2) can still be used.

Obviously, this learning rule is only applicable to networks with one layer of perceptrons, since both desired output and input of each unit must be known to apply the rule. In addition to that problem, such a single-layer perceptron system can only learn linearly separable functions (this was an important argument used by Minsky and Papert (1969) when showing the limitations and

problems of perceptrons). Indeed, it cannot learn non-linearly separable functions because there exists no set of weights that will enable the perceptron to perform that type of classification using a single layer of perceptrons. The perceptron convergence theorem is still true since it mentioned that the learning rule converged for classification problems for which there existed a solution with the perceptron architecture. An example of functions that are not linearly separable is the exclusive-or (XOR) function, or the parity function.

Minsky and Papert 's book (*Perceptrons*, 1969) presented a thorough analysis of the perceptron and of its limitations. Its pessimistic evaluation of the perceptron discouraged further development of perceptron-like models until the early 80's. The non-linearly separable functions could be computed on a multi-layer perceptron, but it was not known how to train the nodes not directly connected to the input or the output of the network. Such nodes are called **hidden** nodes. The discovery of learning algorithms for multi-layer perceptron-like networks (see chapters 3 and 4) gave rise to a new research interest in neural networks. Figure 2.3, derived from Lippman (1987), shows the types of decision regions that can be formed by single-layer, two-layer and three layer perceptrons with two inputs so that the input space could be represented on the plane. One can see in this figure why the single layer perceptron, which can form a decision region for a class defined by a half space bounded by a hyperplane, cannot produce the XOR function. One can also see in this figure that to solve most general classification problems, a perceptron with three layers of weights (i.e. two hidden layers) can be used (although the number of necessary hidden nodes depends on the complexity of the classification).

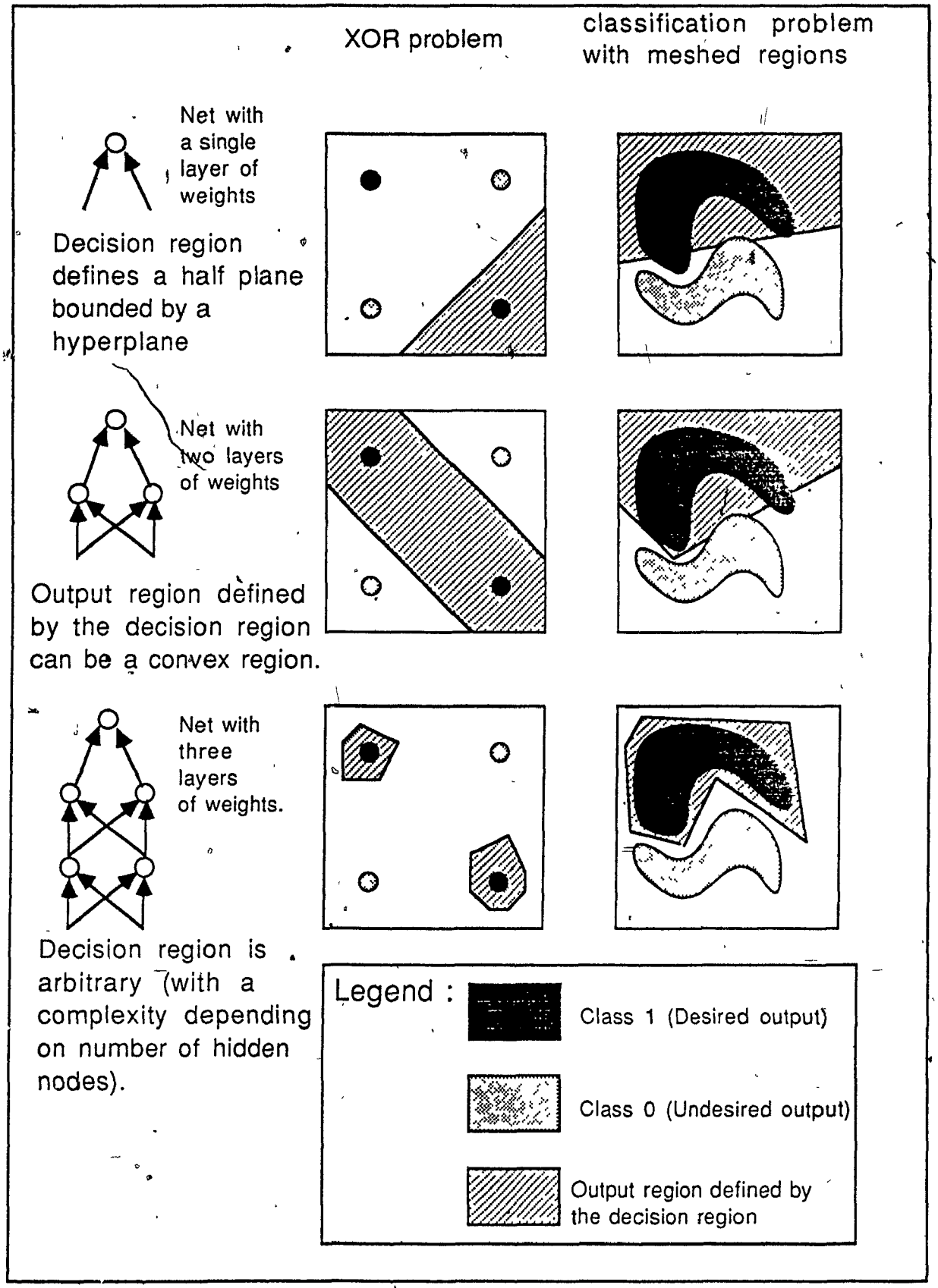

Figure 2.3 : *Decision regions of a perceptron network, with one, two , and three layers of weights (i.e. zero, one and two hidden layers respectively).* Derived from (Lippman, 1987).

2.2.3 NETL

The NETL system (Fahlman, 1979) is a type of connectionist system for manipulating symbolic knowledge. It uses local representation: each node represents a particular concept, defined by the user; connections represent relations or assertions. It is an implementation of a semantic network. The architecture is made of a large number of simple processing units to represent the concepts and the relations. Each one is capable of storing only a few single-bit markers and of performing simple boolean operations, all in parallel, in response to the broadcast command of a system controller (a serial machine).

The NETL system can be used as an "intelligent" knowledge base. It can perform inheritance, handle exceptions and perform set intersections, all in parallel (constant time). It works well for symbolic processing, where every feature is either active or inactive, but is not appropriate for the noisy and very variable sensory information of speech and image recognition.

2.3 Taxonomy

Connectionist models can be classified along the following axes:

- local *vs* distributed representation :

Local representation consists in the assignment of specific concepts to individual units. The 'grandmother cell' theory says that there exists one particular cell in your brain that gets activated for one concept. For example, if someone talks to you about his grandmother, or if you see your grandmother coming into the room, the grandmother cell will be activated. The cell will be

used only to represent that concept. On the other hand a distributed representation means that a certain pattern of activation of units will represent the activation of a concept, and furthermore, that each unit may be used in the activation of many concepts. At the extreme, we find the **hologram model** : each tiny element of a hologram contains information about all parts of the picture. Conversely, any subset of the hologram contains the whole picture, albeit with less details. Most probably, the brain uses an intermediate representation in which parts of the cortex are organized in modules, themselves using a relatively distributed representation. The disadvantage of distributed representations is the difficulty to interpret the activation of the units. It is also impossible to add macroscopic knowledge to those nets from the outside by changing the microscopic weights since each weight influences many concepts. One must absolutely use a learning rule which relies on training and examples. An advantage of distributed representations is their tolerance to failures or errors of the units (**graceful degradation**). This is an important feature of connectionist models, which has been studied by a number of researchers e.g. McClelland (1986), Hinton & Sejnowski (1986) and is similar to graceful degradation in the brain (Rumelhart & al., 1986),(Schwartz, Marin, & Saffran, 1979).

- binary *vs* continuous inputs and outputs :

Some networks may accept continuous input but produce binary output (e.g. the perceptron). Others may have both inputs and outputs that are continuous (e.g. the error back propagation algorithm). Others may be binary in nature but can be converted to continuous input and output (e.g. the perceptron can become the Widrow-Hoff perceptron (Widrow & Hoff, 1960)). Others can be binary in nature, but continuous inputs or outputs can be used by changing or looking at internal variables of the units (e.g. the Boltzmann machine algorithm cf. chapter 3 and section 6.2.2).

- stochastic *vs* deterministic operation

It is believed that biological neurons fire according to a stochastic rather than a deterministic process. The Boltzmann machine (chapter 3) and other models (e.g. Barto et al. 1983) use a stochastic activation function (or simply add random noise to the deterministic part of the function). Of course, a simpler approach is to use a deterministic network (e.g. the error back propagation, chapter 4, the Hopfield net, section 2.4.1).

- supervised *vs* unsupervised learning

This is a fundamental axis of categorization of connectionist models, although it can admit intermediate cases. A supervised adaptive network is one in which both inputs and desired outputs (or classification) are specified. The network has a "teacher" that provides not only an environment but also a measure of the errors that the network makes. In the unsupervised case, the network is simply immersed in an environment, and it tries to construct an internal model or representation of that environment that will extract the structure, or the statistical regularities present in this environment. Such a network may thus "discover" a set of statistically salient features of the environment. The single-layer or multi-layer perceptrons are supervised neural networks. The Boltzmann machine used for classification is also one. The Competition network presented in section 2.4.3 (Rumelhart & Zipser, 1985), the Regularity Detector in section 2.4.2 (Geman, 1981) and Kohonen's self-organizing feature maps (Kohonen, 1982,1984) are examples of unsupervised learning paradigms.

- auto-association or pattern completion *vs* pattern association or classification

There are a lot of associative memory models (or content addressable memory)(e.g. Kohonen, 1977). A certain number of patterns are stored in the network. In the retrieval phase, a noisy or incomplete version of the pattern is presented, and the network converges to the closest stored pattern (e.g. Hopfield Net, section 2.4.1), or the network tries to complete the missing or unspecified parts of the input (e.g. the Boltzmann machine). Another type of network operates by computing an output when an input is given. They can thus be used for pattern association or classification, or to realize any input/output function.

2.4 Current Models

The most popular current models of neural nets are the error back propagation algorithm and the Boltzmann machine algorithm. These two models have been used in the experiments described in chapter 6, so they are thoroughly described in chapters 3 and 4. In this section, other models of neural nets are therefore presented.

2.4.1 The Hopfield Net

The Hopfield net was introduced in 1982 by Hopfield. It is a network with binary inputs that can be used as an associative memory or to solve optimization problems. Hopfield worked on several versions of this network (Hopfield, 1982),(Hopfield, 1984),(Hopfield, 1986). A version of this network used as associative memory will be described here (see figure 2.4 for the structure of the network).

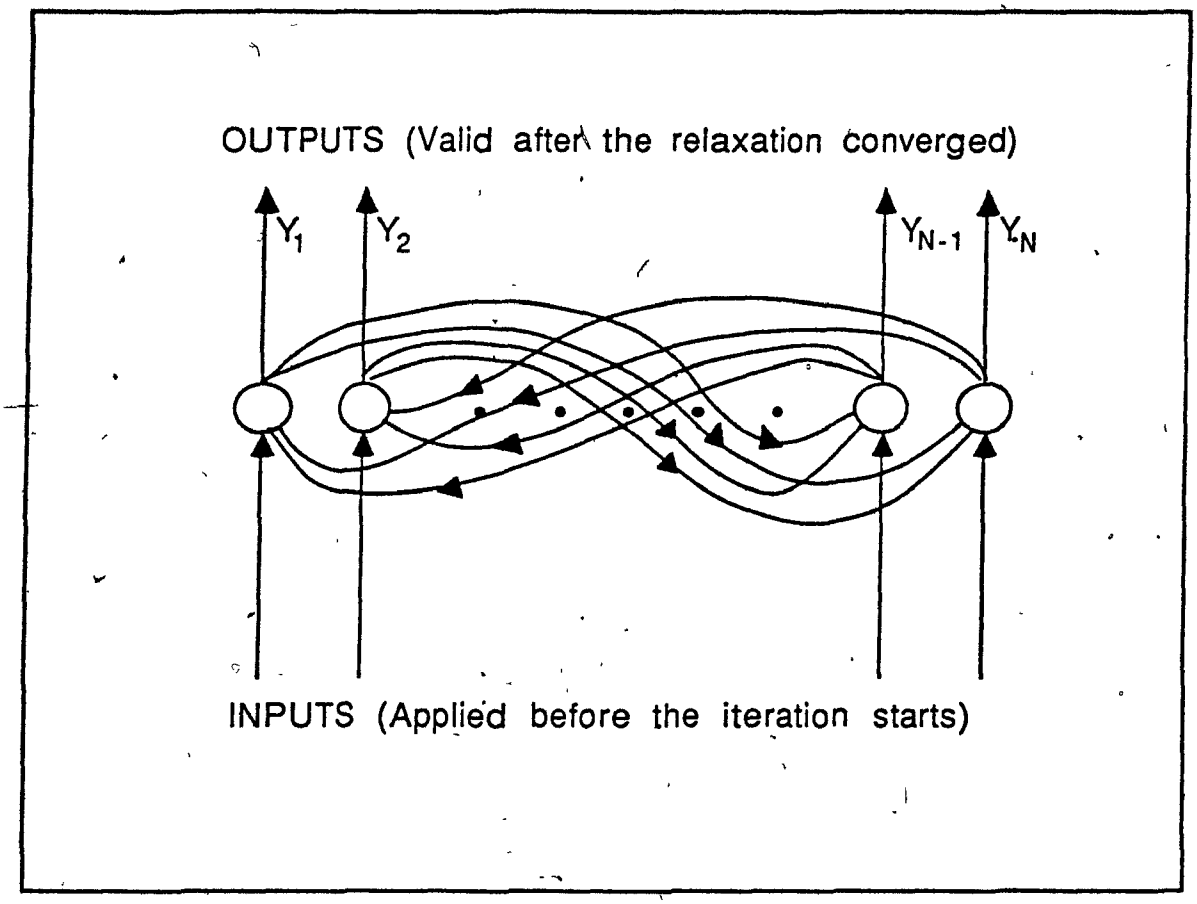

Figure 2.4 : *Structure of the Hopfield net.*

The N nodes are all connected to each other. They are used both as input and output. Each node computes its output, which can be either 1 or -1, as follows :

$$Y_i \leftarrow f\left(\sum_j W_{ij}\, Y_j\right) \qquad (2.3)$$

where $f(x) = 1$ if $x>0$ and -1 otherwise, and Y_i is the output of node **i**, W_{ij} is the connection weight between node **i** and **j**. The Hopfield net works by applying a value to the set of nodes and then iterating by successively updating the output of the nodes. Hopfield proved (Hopfield, 1982) that this relaxation process would converge if the weights were **symmetrical** ($W_{ij} = W_{ji}$) and the node outputs are updated **asynchronously**.

The weights are determined not by a learning process but by assigning them the following value, depending on the set of examplar vectors $(x_{1s}, x_{2s}, \ldots x_{Ns})$ for the M classes s :

$$W_{ij} = \sum_{s=0 \text{ to } M-1} x_{is}\, x_{js}, \quad \text{for } (i \neq j)$$
$$= 0 \quad \text{for } (i=j) \qquad (2.4)$$

x_{is} can only take values +1 or -1. The **relaxation** process is stopped when the network outputs have ceased changing.

The number of patterns that can be stored in the Hopfield net is limited to less than 0.15 x N. If more classes are stored, the network may start converging to some spurious patterns which were not stored. Another problem with the Hopfield net is that a stored pattern will be unstable if it shares many bits with another pattern : when the pattern is applied at time zero, the network converges to another, close pattern instead of staying in this configuration. This problem can be eliminated by an orthogonalization procedure (Grant & Sage,1986).

2.4.2 A Regularity and Correlation Detector

In (Geman, 1981) the author attempts to "design ... a system whose purpose is to discover temporal and spatial regularities in a high-dimensional environment". Like the other connectionists he proposes to look to the neural and cognitive sciences for clues about the proper architecture for an intelligent system. One of these clues is that the strength of a synaptic connection is influenced by the activities of the two neurons that communicate through this connection. Another one is that some cells of the visual cortex (and probably of other sensory areas) are initially (in the newborn animal) not specific in their activities, and they learn to signal different selective events (or features) of

the environment. This specialization is dependent on the environment and the experiences of the animal.

At a given time, we observe only a fraction of the features of the environment (e.g. part of an object is hidden from our view, etc...). We use the observed features to predict the value of some of the unobserved features. Geman defines the state of a feature as observed or unobserved. The value of a feature is only available in the observed state. He assumes that the state (observed or unobserved) of a feature carries no information about the value of this feature; what the neural network does is to guess or estimate the value of unobserved features. He calls a feature "decided" if it has been observed or its value has been estimated. The decided features thus represent current hypotheses about the external environment. To estimate the value of an unobserved feature, the neural networks forms an opinion concerning its value which is based upon the value of other decided features. If that opinion is strong enough (i.e. reaches above a certain threshold) then this feature becomes decided and its value available to other units.

Geman defines features f_i having two possible values, + and -. He then asks the following question : given all joint distributions among pairs of feature values, how could we obtain a local opinion Oj for the value of an unobserved feature f_j ? For this purpose he defines the following quantities for each feature :

$y_i = 1$ if $f_i = +$ and 0 otherwise
$n_i = 1$ if $f_i = -$ and 0 otherwise

$r^+_{ij} = P(y_j|y_i)$ {probability that f_j=+ given that f_i=+}
$r^+_{ij} = P(n_j|y_i)$

Thus $y_i=0, n_i=0$ means that feature f_i is unobserved. A possible form for O_j is then the following :

$$O_j = (1/M)\ (\ \sum_i y_i\ \log(1/(1-r^+_{ij})) - \sum_i y_i\ \log(1/(1-r^-_{ij}))\ ) \qquad (2.5)$$

where $M = \sum_i y_i$ = the number of observed features with value +. The two sums in the equation represent the opinions that f_j is + or -, respectively, given that decided f_i's are +.

However, a real system has only a finite experience and thus the conditional probabilities can only be estimated. The problem is that in summing the "opinions" of the various connections the fact that some of them have more or less "experience" (i.e. that their opinion should be more or less valued) is not taken into account. To solve that problem he looks for new functions $r^+_{ij}(t)$ and $r^-_{ij}(t)$ which tend to the conditional probability as $t \rightarrow \infty$ but are small when the number of simultaneous observations is small. Consequently "inexperienced" features would not much influence the opinion O_j. He derives the following definition of $r^+_{ij}(t)$ and $r^-_{ij}(t)$:

$$r^+_{ij}(t) = P(y_j|y_i)[1-\exp(-\ \varepsilon\ p\ \ P(y_i)\ t]$$
$$r^-_{ij}(t) = P(n_j|y_i)[1-\exp(-\ \varepsilon\ p\ \ P(y_i)\ t] \qquad (2.6)$$

where $p = P(f_i$ is observed and f_j is observed), and ε is a small constant.

Up to now only the first layer of cells were considered, i.e. those that are directly tied to the external world. To derive an opinion on the value of unobserved features only pairwise statistical information were used i.e. only first-order statistics. However these first-order statistics don't represent the whole available statistical information about the perceived input. Adding statistical variables of higher order would significantly improve the decision taking (as adding hidden layers to the perceptron enhances its capabilities). In fact it would be necessary to add to the network some hidden units with the following definition :

$z = y_{i1}\ y_{i2} \ldots y_{ik}$, k=1 to N,

where N is the number of input cells

These hidden units are ON when k specific input cells are ON. k represents the order of a particular statistic. Obviously the number of such cells grows much too rapidly with N (exponentially) and it will not be possible to have all of the possible high order statistics in the network. Thus Geman proposes that only those "that are, by some measure most frequent and most important" be kept. He defines the importance of a statistic as its "correlation to innately important events". He calls this choice of relevant high order statistics the commitment of cells. He proposes, based on studies of neural development, that the commitment should be hierarchical. Layers of increasing statistical order draw their inputs from a few preceding layers. Initially the first order statistics are evaluated, for the first layer. Then, based on these statistics and correlation to innately important events, some second order cells are chosen (representing the conjunction of pairs of features). As in the first layer, statistics are then accumulated : the connection strengths mature. After a while we are ready to commit cells in the third layer, based on the first two, etc...

The system just described analyses the spatial structure of the input, i.e. it is not concerned with the correlation between succeeding frames of inputs. However our brain handles both spatial and temporal analysis and intelligent machines should be able to do so. Geman proposes a second system, called the temporal coding module, which takes its first level inputs (primitives) from various stages of the spatial coding module. The only difference with the design of the spatial module is that the definition of high level statistics is now the following :

$z = 1$ if y_{i1} follows y_{i2} follows y_{i3} ... follows y_{ik}.

Thus hidden units detect sequences of events rather than their simultaneous activity. "Follow" is defined in a rigorous discrete manner : activity in a level **p** unit is said to follow activity in a level **q** unit if it occurs during the **p**th period of time after activity in the level **q** unit. In other words, the sequence of primitives associated with the level **q** unit must immediately precede the sequence of primitives associated with the level **p** unit.

Finally Geman argues about the need for a third module, which would provide some definition of "appropriate action" and "innately important events". The primitives of this module consists of the inherently good and bad feedback (e.g. pain and pleasure in the animal). Higher level cells represent events in the spatial module and the temporal module that are correlated with the good and bad primitives. They influence the commitment of high level cells in the spatial and temporal modules.

2.4.3 Competitive Learning

Rumelhart and Zipser report (1985) on studies of an unsupervised learning paradigm called "competitive learning", applied to parallel networks of neuron-like elements. This method provides to a neural net a way to discover regularities and general features which can be used to classify a set of patterns. These feature detectors can be formed on the basis of a multi-layer system that can learn categorizations which are not linearly separable. Other workers proposed competitive learning algorithms such as Kohonen 's feature maps (Kohonen, 1984) and Fukushima's neocognitron (Fukushima, 1980).

The basic characteristics of the competitive learning scheme are the following, as outlined by Rumelhart & Zipser (1985,:

1. The units of the network are identical except for a randomly distributed parameter. This initially makes each unit respond differently to the input.

2. Each unit can distribute a fixed amount of positive "strength" to its input connections ($\sum_i w_{ij}$ is fixed to 1).

3. The units of each inhibitory cluster compete to respond to the input.

The consequence is that units adapt in order to respond to sets of similar patterns (they put their weight strength in connections that are often used by a set of similar input patterns).

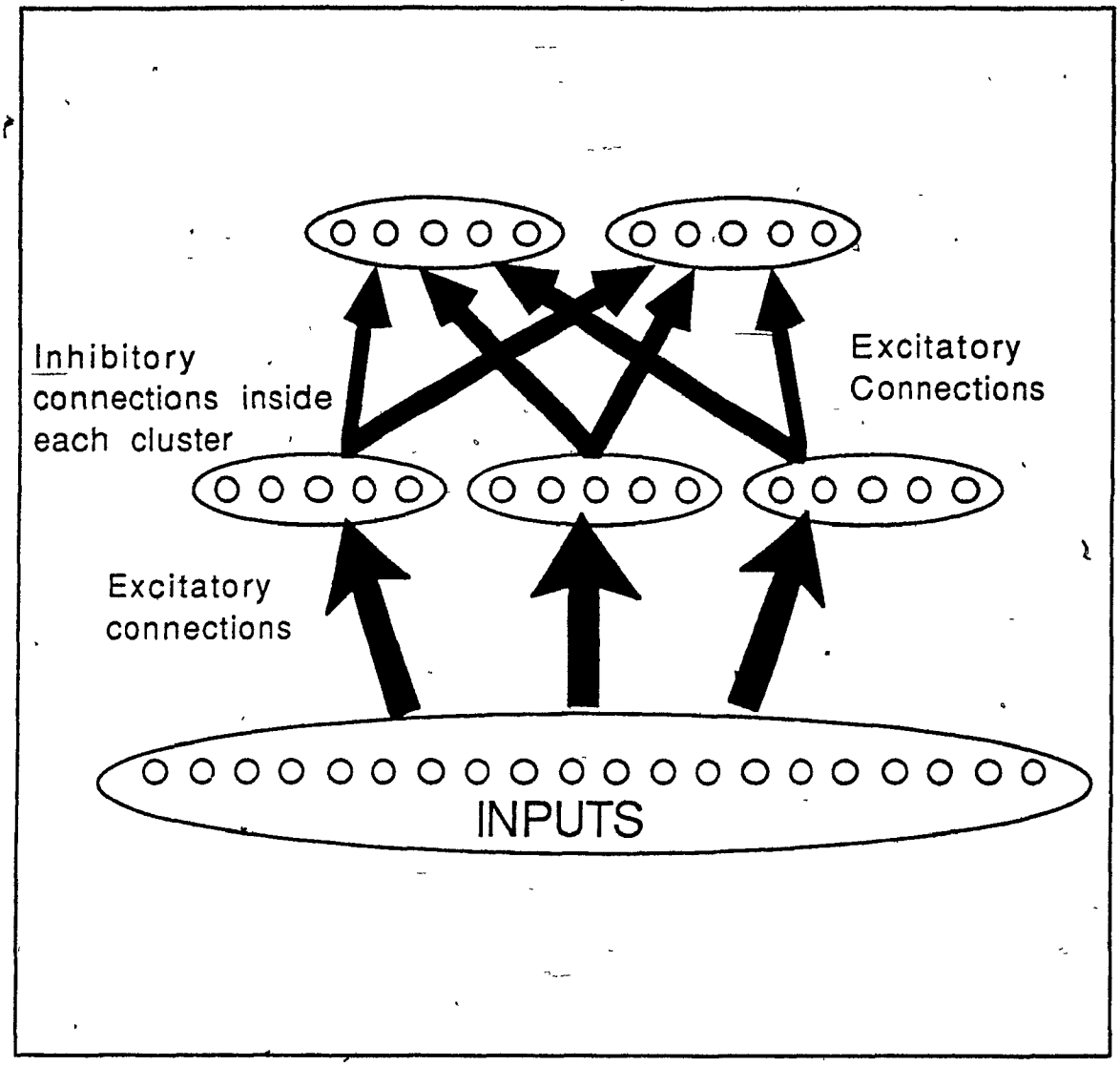

Figure 2.5 : *Architecture of a competitive learning mechanism.*

The architecture of a competitive learning system consists of a set of hierarchically layered units in which each layer connects with the layer above it. Within a layer, the units are separated into inhibitory clusters in which all elements inhibit each other, competing with one another to react to the pattern of the preceding layer. The unit receiving the largest input in a cluster attains its maximum value (state) while the others are forced to a minimum value. All units in an inhibitory cluster receive excitatory inputs from all the units in the lower layer. Each unit distributes a fixed amount of weight among its

excitatory input connections. Thus for a unit, learning means "shifting weight from its inactive to its active input lines", when it wins the competition :

$$\Delta w_{ji} = 0 \qquad \text{if unit j loses on stimulus k}$$
$$= \text{constant} \times [\, (c_{ik}/\, n_k) - w_{ji}\,] \qquad \text{if it wins on stimulus k.} \tag{2.7}$$

where c_{ik}=1 if unit i in the lower layer is active for stimulus k and 0 otherwise. n_k is the number of active units in the lower layer for stimulus k :

$$n_k = \sum_i c_{ik} \tag{2.8}$$

As a result of this learning algorithm, experiments show that each cluster of size M classifies the stimulus set into M groups. Each unit responds approximately to an equal number of input patterns. The units categorize the patterns in a structurally relevant way if there is structure (redundancy) in the environment. A large number of independent clusters receiving inputs from the same input units can classify the inputs according to a variety of independent features present in the stimulus, although it is possible that two clusters choose the same classification. The authors show analytically that after it has been learning for a period of time, the system will spend most of its time in the most highly stable of the equilibrium states. An equilibrium state is defined as a classification state (of the weights) in which the weights are not changing in the average (providing a stable set of features).

As suggested by a formal analysis, asymptotically the weights were experimentally found to be proportional to the probability that the "presynaptic" unit is active when the "postsynaptic" unit wins. That is,

$$w_{ji} \rightarrow P(s_i = 1 \mid \text{unit j wins}). \tag{2.9}$$

3. The Boltzmann Machine Algorithm

This model was introduced in 1983 by Falhman, Hinton & Sejnowski. A basic idea of this model is that of using parallel networks to perform relaxation searches that simultaneously satisfy different weak constraints represented by the clamped values of input or output units. The computation is performed by iteratively decreasing a cost function representing the extent to which the current state of the network (which means the output values of all the units in the network) violates the input constraints. These "weak" constraints are not necessarily absolute constraints : they have a cost associated to their violation (i.e. there may be some exceptions to these constraints, as for example there may be exceptions to some grammar rules).

This model (as well as several others) has received much of its inspiration from concepts of statistical physics. Like the back-propagation model it can also be seen as an extension of the perceptron.

3.1 Constraint Satisfaction

Hopfield introduced in 1982 a neural net with a relaxation scheme involving a cost function, and where the network converges to a local minima of the cost function (see section 2.4.1). He showed that if the connections among the units are symmetrical (i.e. the weight W_{ji} from unit **i** to unit **j** equals the weight W_{ij} from unit **j** to unit **i**), and if the units are updated asynchronously, then repeated iterations are guaranteed to find an energy minimum (the cost function never decreases). He called the cost function "Energy" and defined it as :

$$E = - \sum_{i<j} W_{ij} s_i s_j + \sum_i B_i s_i \qquad (3.1)$$

where W_{ij} is the weight on the connection from unit **j** to unit **i**, s_i is the state (0 or 1) of the ith unit, and B_i is a bias for unit **i**. The Hopfield net contains only visible units, i.e. units which are directly connected (influenced) by the environment (input/output). The Boltzmann machine model has **visible** and **hidden units**. The hidden units are only connected to each other and to the visible units.

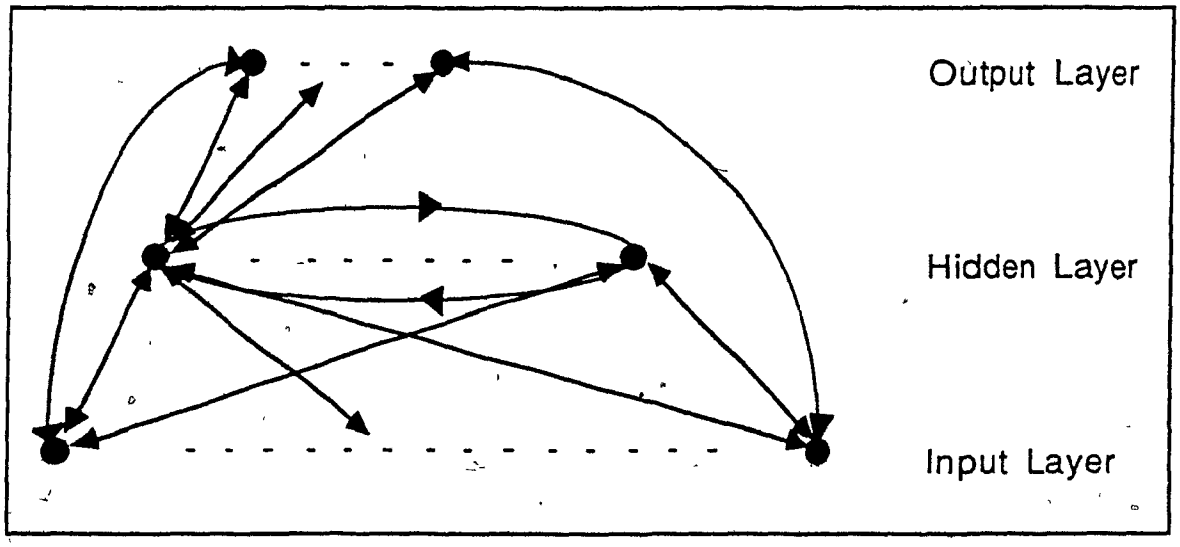

Figure 3.1 : *Input, Output and Hidden Layers for the Boltzmann Machine.*

To ensure that the cost function never decreases, Hopfield's updating rule chooses for every unit **i** the state for which the energy is lower. Since the connections are symmetrical, the unit should be ON (1) if

$$\Delta E_k = \sum_i W_{ki} s_i - B_k > 0 \qquad (3.2)$$

and OFF (0) otherwise. Thus the unit is a **binary threshold unit.**

The main problem with this rule is that the network will get trapped in local minima of the Energy function (if there are any local minima). This was a desirable feature for Hopfield nets since these networks are used as associative memories where local minima represent stored items. However, for more complicated problems where the energy landscape has undesirable local minima, in particular in the case of constraint satisfaction, it is desirable to reach an absolute minimum of the Energy function.

To solve this problem, the authors of the Boltzmann machine looked at the work of Kirkpatrick (1983) who introduced a search technique for solving hard optimization problems. Instead of allowing only reductions of the energy, some upward moves must be allowed in order to get out of local minima. Kirkpatrick used a physical analogy, that of finding a very orderly (low energy) state of a metal by melting it and then slowly allowing it to cool. This is called annealing and thus a search procedure based on this procedure was called **"simulated annealing"**.

The intuitive explanation of why this principle functions (Hinton & Sejnowski, 1986) can be understood by considering the state of the network as a ball-bearing on an energy landscape. When the temperature is very high, (i.e. imagine that we shake a lot the system) the ball can be anywhere with almost equal probability. This is because when the temperature is very high the energy barrier between low energy minima and higher energy minima is very small. On the other hand, if the system is gently shaken (i.e. at lower temperatures), the ball will have a higher probability of being in a low energy minima, but it will not make transitions between one minima and another very often. Annealing starts by applying a high temperature to the network and gradually reduces temperature. As a result, the system will go through a temperature at which the thermal noise makes the best compromise between the absolute probability of transitions (escaping a minimum) and the ratio of probabilities of settling into

minima of different energy (see equation (3.4))(cf. Hinton & Sejnowski, 1986). At the end of annealing the ball-bearing will be at the bottom of the global minimum. The proof of this phenomenon is of the domain of statistical physics but the result applies to the Boltzmann machine.

Thus, the idea of Hinton and Sejnowski was to apply the simulated annealing procedure to the Hopfield net. This can be done by modifying the update rule for each unit, specifically using a **sigmoidal** function of the energy barrier over the temperature instead of the deterministic threshold function of Hopfield nets :

Unit **k** chooses the state 1 with probability

$$p_k = 1 / (1 + \exp (- \Delta E_k / T) \qquad (3.3)$$

where T represents temperature in the simulated annealing process.

As a result of this update rule, in thermal equilibrium the relative probability of two global states is determined by their energy difference and follows a Boltzmann distribution (hence the name Boltzmann machine)(see Hinton, Sejnowski & Ackley, 1984):

$$P_a / P_b = \exp (- (E_a - E_b) / T) \qquad (3.4)$$

where E_a is the energy of state **a**, and P_a the probability of finding the network in that state at equilibrium.

3.2 Learning

The result of each relaxation is, with a certain probability, a global state of low energy. Thus we could control what the network will compute by "assigning" an energy value (and thus a

probability) to each global state. However, in general the environment or teacher will not be able to specify the probabilities of global states since that would include visible as well as hidden units. It may only provide information about the visible units. Why are hidden units desirable in a connectionist model ?

The old perceptron had only visible units, and thus it was easy to find a learning rule for the weights. However, since there were connections only between pairs of visible units, the network could only learn the first order statistical structure (with the v thresholds) and the second order statistical structure (with the $v^2/2$ pairwise connections) implicit in the environmental examples.

When faced with a problem of a higher structure (for example the XOR problem, see section 2.2.2) the perceptron convergence procedure (learning rule) fails miserably.

There are only pairwise connections in the Boltzmann Machine but high level statistical information can be represented using hidden units (that are not directly influenced by the environment). For example, in the case of the XOR problem, if we introduce only one hidden unit, that will have weights to detect an AND of the two input units then it is possible to find the other weights to get the output unit to behave like an XOR of the two input units.

However how does one decide what the hidden unit(s) should recognize (i.e. what feature should they detect)? What weights should be assigned to their connections from the input units so that the hidden units become useful feature detectors that represent high order statistical regularities of the environment?

One answer to these questions is the Boltzmann Machine learning algorithm. Making certain assumptions, the authors of the Boltzmann machine algorithm, derived an information theoretic

measure of how well the weights model the structure of the environment, and thus how to modify the weights to improve this measure. The first assumption is that when the environment clamps some of the visible units, they stay clamped until the network reaches thermal equilibrium (not like for the Hopfield net). The second assumption is that there is no statistical structure in the sequence of input patterns (the neural network is only going to learn about the static structure of these patterns, not about their sequence).

There are 2^v possible environment vectors over the v visible units. The complete structure of the environment is thus defined by the probabilities $P^+(V_a)$ of the 2^v vectors V_a. (The + means that the visible units are clamped). The idea is to compare this distribution with the $P^-(V_a)$ (the probability of vector V_a over the visible units when the network is running freely, no clamped units). Thus a perfect set of weights is one for which $P^+(V_a)=P^-(V_a)$ for all environment (input/output) vectors V_a. The information theoretic measure (Kullback, 1959) of the distance between two distributions is given by :

$$G = \sum_a P^+(V_a) \ln (P^+(V_a) / P^-(V_a)) \qquad (3.5)$$

It is called asymmetric divergence or information gain. It is not symmetrical because it is more important to get the probabilities correct for input patterns that occur frequently than for rare ones (hence the first factor).

It is possible to do gradient descent in G : changing the weights so as to reduce G. We show in the next section (3.3) that

$$\partial G/\partial W_{ij} = - 1/T [p_{ij}^+ - p_{ij}^-] \qquad (3.6)$$

where p_{ij}^+ and p_{ij}^- are the probabilities, averaged over all environmental inputs and measured at equilibrium that the units i

and j are ON when the visible units are clamped and not clamped, respectively.

Two interesting features of this equation is that it involves only local information and that it is the same for visible and hidden units.

However, the algorithm is still not completely determined. One still has to decide of a temperature cooling schedule, of how many weights to change at a time, of how much to modify each weight given the p_{ij}^+ and p_{ij}^-, of how long to collect the statistics for the p_{ij}^+ and p_{ij}^-. These aspects will be discussed in section 3.4 on the implementation.

3.3 Derivation of Gradient Descent Rule

An equivalent of the following demonstration can be found in works by Hinton & Sejnowski (1986) and Hinton, Sejnowski & Ackley (1984). This section is not absolutely necessary to the understanding of the rest of the thesis, and can thus be optionally skipped by the reader. We consider here the case where visible units are either fixed or clamped, and the network is used for pattern completion.

Let us derive the gradient descent rule, presented in (3.6). The G-measure is defined in (3.5). We want to find its derivative w.r.t. the connection weight W_{ij}. However, $P^+(V_a)$ (the probability of applying vector V_a over the visible units, i.e. when they are fixed, at equilibrium) is independent of W_{ij} (since the visible units are fixed from the outside, they do not depend on the network activity and connections). Hence the derivative of G can be written as follows :

$$\partial G/\partial W_{ij} = \sum_a [\, \partial P^+(V_a)/\partial W_{ij} \ln(P^+(V_a)/P^-(V_a)) + P^+(V_a)(1/P^+(V_a)\ \partial P^+(V_a)/\partial W_{ij} - 1/P^-(V_a)\ \partial P^-(V_a)/\partial W_{ij})]$$

$$= - \sum_a P^+(V_a)/P^-(V_a)\ \partial P^-(V_a)/\partial W_{ij} \qquad (3.7)$$

Now let us consider the last factor of (3.7), i.e. consider the network when the visible units are not fixed. The probability distribution over the visible units at equilibrium is the following :

$$P^-(V_a) = \sum_b P^-(V_a \wedge H_b) = (\sum_b \exp(-E_{ab}/T))/(\sum_{I,u} \exp(-E_{Iu}/T) \qquad (3.8)$$

where H_b is a vector of states of the hidden units, V_a a vector of states of the visible units and E_{ab} the energy of the system in state $(V_a \wedge H_b)$:

$$E_{ab} = - \sum_{i<j} W_{ij}\ s_i^{ab}\ s_j^{ab} \qquad (3.9)$$

as in (3.1), where W_{ij} is the connection weight between unit **i** and unit **j**, and s_i^{ab} is the state (0 or 1) of unit **i** when the vectors V_a and H_b are applied to the visible and hidden units respectively. Note that the bias has been eliminated from the equations by considering it as the weight of a connection from an input unit which is always ON. From (3.9), we obtain:

$$\partial \exp(-E_{ab}/T)/\partial W_{ij} = 1/T\ s_i^{ab}\ s_j^{ab}\ \exp(-E_{ab}/T)$$

Hence we can differentiate (3.8) :

$$\partial P^-(V_a)/\partial W_{ij} = (1/T \sum_b \exp(-E_{ab}/T)\ s_i^{ab}\ s_j^{ab})/\sum_{I,u} \exp(-E_{Iu}/T) \cdot -(\sum_b \exp(-E_{ab}/T)\ 1/T \sum_{I,u} \exp(-E_{Iu}/T)\ s_i^{Iu}\ s_j^{Iu})/(\sum_{I,u} \exp(-E_{Iu}/T))^2$$

$$= 1/T\, [\, \sum_b P^-(V_a \wedge H_b)\ s_i^{ab}\ s_j^{ab} - P^-(V_a) \sum_{I,u} P^-(V_I \wedge H_u)\ s_i^{Iu}\ s_j^{Iu}] \qquad (3.10)$$

and rewrite $\partial G/\partial W_{ij}$ as follows :

$$\partial G/\partial W_{ij} = -1/T \left[\sum_a P^+(V_a)/P^-(V_a) \sum_b P^-(V_a \wedge H_b)\, s_i^{ab}\, s_j^{ab} - \sum_a P^+(V_a) \sum_{l,u} P^-(V_l \wedge H_u)\, s_i^{lu}\, s_j^{lu} \right] \quad (3.11)$$

Now we can simplify a little as follows :

$\sum_a P^+(V_a) = 1$, by definition of a probability.

The probability of a conjunction can be written in terms of the conditional probability :

$$P^+(V_a \wedge H_b) = P^+(H_b \mid V_a)\, P_+(V_a)$$
$$P^-(V_a \wedge H_b) = P^-(H_b \mid V_a)\, P^-(V_a)$$

also,

$$P^-(H_b \mid V_a) = P^+(H_b \mid V_a) \quad (3.12)$$

since the probability of a hidden state vector given a visible state vector is the same wether or not the visible state vector has been clamped or arrived to this value by iterations of the network. Thus we can write

$$P^-(V_a \wedge H_b)\, P^+(V_a)/P^-(V_a) = P^+(V_a \wedge H_b) \quad (3.13)$$

and simplify as follows :

$$\partial G/\partial W_{ij} = -1/T \left[\sum_{a,b} P^+(V_a \wedge H_b)\, s_i^{ab}\, s_j^{ab} - \sum_{l,u} P^-(V_a \wedge H_u)\, s_i^{lu}\, s_j^{lu} \right] \quad (3.14)$$

Let us define p_{ij}^+ and p_{ij}^- (the probabilities averaged over all possible input vectors that unit i and unit j are both ON, at equilibrium) as follows :

$$p_{ij}^+ = \sum_{a,b} P^+(V_a \wedge H_b)\, s_i^{ab}\, s_j^{ab} \quad (3.15)$$

$$p_{ij}^- = \sum_{l,u} P^-(V_l \wedge H_u)\, s_i^{lu}\, s_j^{lu} \qquad (3.16)$$

so that the derivative of G w.r.t. to W_{ij} can be written as in (3.6) :

$$\partial G/\partial W_{ij} = -\, 1/T\, [\, p_{ij}^+ - p_{ij}^- \,]. \qquad (3.6)$$

3.4 Implementation of the Boltzmann Machine for the Experiments

The program to implement the Boltzmann machine algorithm and the various experiments conducted with the speech data was written in C on a VAX-8600 with the VMS operating system. There are several versions of the program, for the different experiments conducted. The program itself is divided in 8 modules. A flow diagram of the high level structure of the implementation of the Bolzmann machine algorithm is presented in figure 3.2. Let us now consider some aspects of the algorithmic implementation.

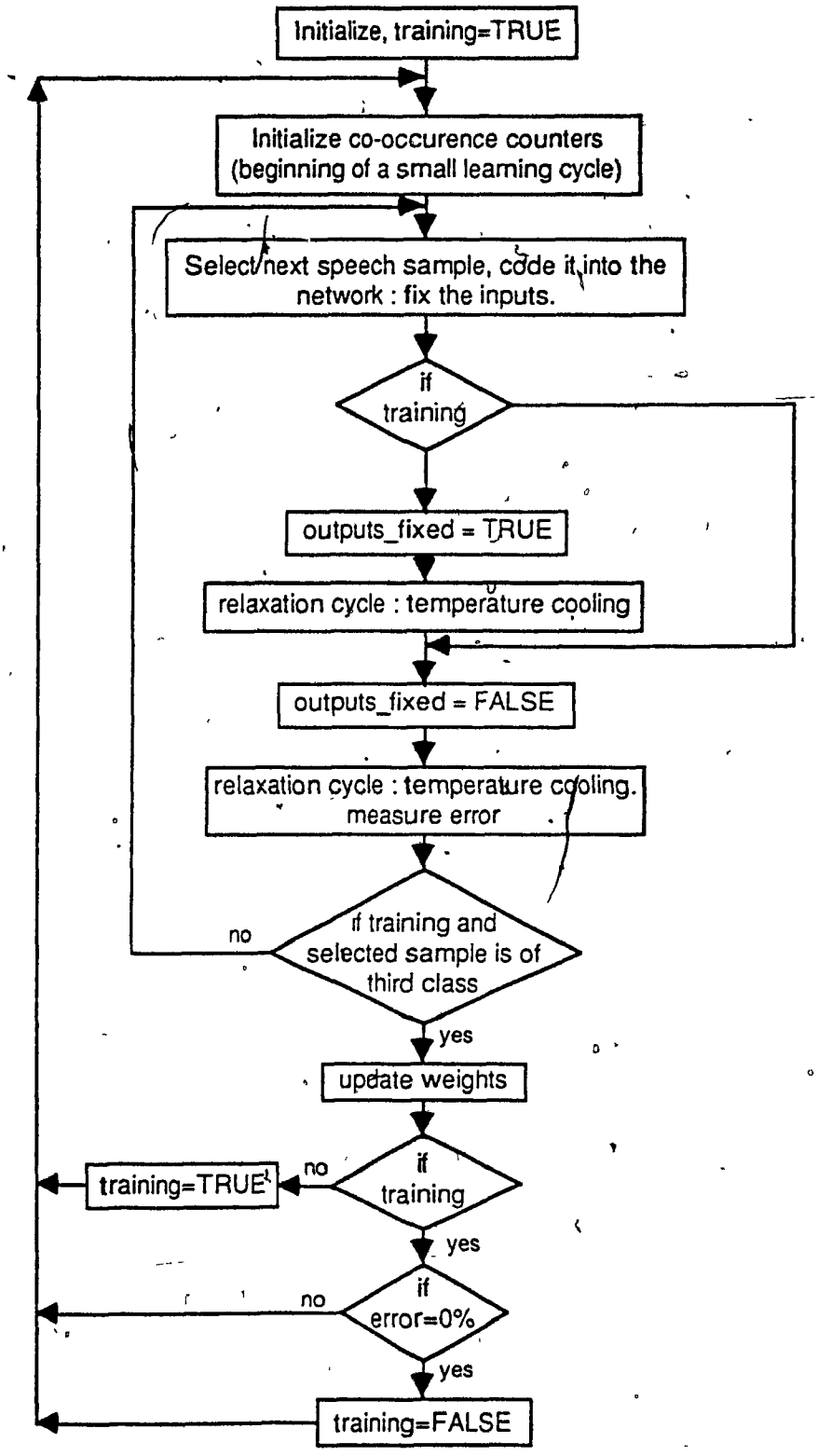

Figure 3.2 : *Flow chart diagram of high level structure of the implemented Boltzmann machine algorithm.* The infinite loop is stopped when the error on the testing set doesn't improve anymore. Note that the speech samples are presented with a first, second, and third class examplars always in sequence.

3.4.1 Temperature Cooling Schedule

The cooling schedule which is guaranteed to make the system reach the best solution is the following (see Geman & Geman, 1984):

$$T(k) \geq c / \log (1 + k) \qquad (3.17)$$

where c is a constant, k stands for the kth cooling iteration and T(k) is the temperature at the kth iteration. In those conditions, with probability converging to one as $k \rightarrow \infty$, the configurations of the network will be those of minimal energy.

However such a cooling schedule is very slow, because of the logarithmic function. In the first experiments that were performed, such a cooling schedule was used.

For the cooling rate, after trying the slow logarithmic cooling the following schedule was used :

$$T(k) = \text{starting\_temperature} / (k + 1) \qquad (3.18)$$

where log(k+1) in (3.7) has been replaced by (k+1). Identical experimental results were obtained with this cooling, but with a gain in speed greater than 10.

In addition to the cooling rate, one must determine the temperature at which to start and to stop. To decide of a starting temperature, the **average energy barrier** ΔE_k that each unit has to jump to make an upward move was computed. The absolute values of local energies of each unit (as defined in equation 3.2) were added for all the iterations to compute a total energy, which is then averaged over all those cycles and units. The chosen starting temperature is this average absolute energy divided by 3. Thus at the beginning of the cooling, the average unit will have

either about 42% or 58% probability of being ON depending on the sign of its local energy. This average energy measures the average height of the energy barriers that each unit has to climb to make an upward jump in the energy landscape. Choosing a starting temperature which depends on this value ensures that if the network constructs very high energy barriers it will be able to jump above them at least at the beginning of the cooling. Most other researchers experimenting with the Boltzmann machine chose a fixed schedule starting at a given temperature and ending at another one, independently of the value of the weights.

To stop the cooling one could choose to wait for a fixed number of iterations, or a fixed finishing temperature, or one could wait for the network to settle (approximately reaching thermal equilibrium). For the experiments of this thesis, we chose to wait for the network to stabilize. At each cooling cycle, the number of units which have switched from one state to another is counted. If the network doesn't change state for a certain number **m** of consecutive cycles, the cooling is stopped. **m** was chosen to be two cycles (by trial and error and looking at the behavior of the network at low temperatures).

3.4.2 Other Parameters

After thermal equilibrium has been reached, for a certain number of cycles, statistical informations are gathered about the number of occurrences and co-occurrences of ON states. These are to be used to update the weights. The number of cycles during which statistics are collected was computed as:

$$\text{len\_stat} = 4 \,/\, \text{sqrt}(\text{average\_error} + 0.1) \qquad (3.19)$$

Thus, as the error gets smaller, more time is spent accumulating statistics. This is because the learning procedure is driven by the

amount of error (discrepancy between the behavior of the units when the visible units are clamped and not clamped).

For the update rule, as suggested in several experiments with the Boltzmann machine (see Ackley, Hinton & Sejnowski 1985) the weights were increased or decreased by a constant multiplied by the SIGN of the difference between the two probabilities p_{ij}^+ and p_{ij}^-, rather than by the difference itself. The advantage of this method is that it deals better with wide variations in the first and second derivative of G (the distance measure between clamped and not clamped network). Thus when G falls rapidly and then rises rapidly the learning doesn't take the large divergent steps that the original definition (definition (3.6)) of ΔW takes.

$$\Delta W_{ij} = \text{learning\_speed} \times \text{SIGN} (p_{ij}^+ - p_{ij}^-) \qquad (3.20)$$

To allow slower variations when approaching the solution (convergence), the learning speed was also changed in function of the average error, as follows :

$$\text{learning\_speed} = 10 \times \text{average\_error} + 1 \qquad (3.21)$$

4. The Error Back Propagation Algorithm

The error back propagation algorithm is also called the **multi-layer perceptron** (Rumelhart & al., 1986). It is a simple extension of the perceptron to several layers, and its learning rule looks similar to the learning procedure of the perceptron (see section 2.2.2)(Rosenblatt 1959, 1962) and its variation (Widrow and Hoff, 1960). As outlined in section 2.2.2 (and shown by the pessimistic evaluation of Minsky and Papert, 1969), the single layer perceptron is limited to learning linearly separable functions, which means that most interesting functions, including such a simple one as the XOR function, cannot be learnt by the perceptron. If we add only one "hidden unit" in addition to the two input units and the output unit, and we set its bias and its connection weights from the input units so that it computes the AND function of the inputs, then the network will be able to learn the other weights, using the perceptron learning procedure, to compute the XOR function.

The "hard learning" problem is to decide how to set the weights and connections of the hidden units. This is what the Boltzmann machine algorithm and the error back propagation algorithm try to do. For the error back propagation model, like for the perceptron, the units are deterministic. They compute their output in function of the weighted sum of their inputs. Like in the model of Widrow and Hoff (1960), the units compute a continuous (not binary) output, and the learning procedure changes the weights so as to implement a gradient descent in a cost or error measure. Although the output units give out a continuous value, the multi-layer perceptron can be used as a classifier, with each output unit standing for a class. The selected class is the one corresponding to the output unit with maximum value.

4.1 Description and Derivation

Let us consider the basic model : it is organized in feedforward layers. There is one input layer, a certain number of hidden layers and an output layer, as shown in figure 4.1. The outputs of units at a layer only go to the input of units on the layer immediately "above". Each unit sends its output to all the units in the superior layer.

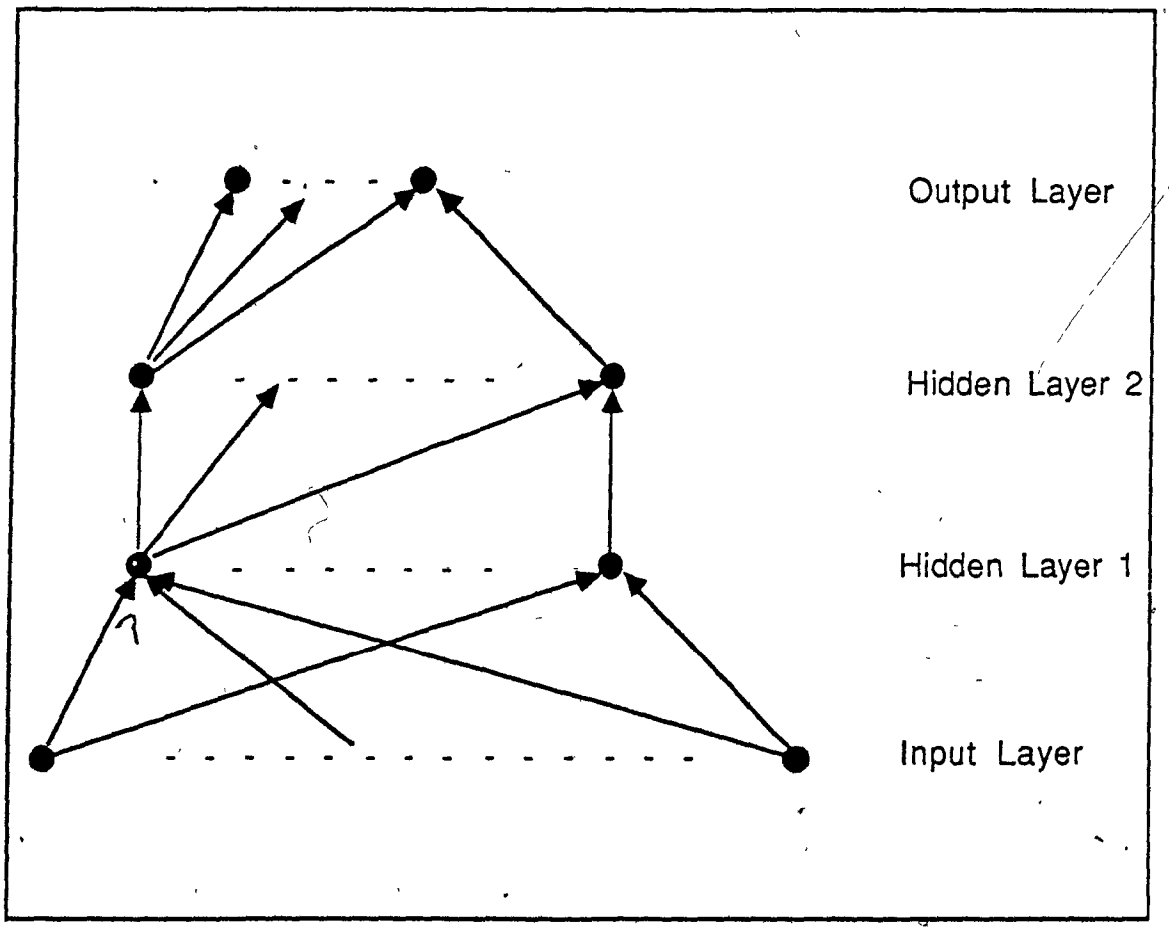

Figure 4.1 : *Feedforward Layers of the Error Back Propagation Algorithm.*

Each unit Y_i computes its output in function of the output of units Y_j on the previous layer as follows :

$$X_i = \sum_{\text{units j on the layer below unit i}} W_{ij} Y_j \qquad (4.1)$$

$$Y_i = f (X_i) \qquad (4.2)$$

where W_{ij} is the weight of the connection from the unit **j** on the previous layer to unit **i**. The function f() is called the activation function. For the perceptron, it is a threshold (or step) function so that f(x) is 1 if x>0 and 0 otherwise. In the case of the error back propagation algorithm a semilinear activation function will be necessary (as defined in Rumelhart, Hinton & Williams 1986) : one which is nondecreasing and differentiable.

Let us define the following squared error measure, when pattern **p** is presented :

$$E_p = 1/2 \times \sum_j (target_{pj} - Y_{pj})^2 \qquad (4.3)$$

where $target_{pj}$ is the desired value for the output unit Y_j (Y_{pj} is the actual output), when pattern **p** is presented. Let us also define the global error as:

$$E = \sum_p E_p \qquad (4.4)$$

To implement gradient descent in E, we want ΔW_{ji} to be proportional to $- \partial E / \partial W_{ji}$. Using the chain rule, we have :

$$\partial E_p / \partial W_{ji} = (E_p / \partial X_{pj})(X_{pj} / W_{ji}) \qquad (4.5)$$

where X_{pj} is the weighted sum of the inputs of unit **j**, as defined in definition 4.1, at the presentation of pattern **p**. Substituting definition 4.1 in equation 4.5, we obtain :

$$\partial X_{pj} / W_{ji} = \partial/\partial W_{ji} (\Sigma_k W_{jk} Y_{pk}) = Y_{pi} \quad (4.6)$$

For the first factor of equation 4.5, let us define the error measure at unit **j**, when pattern **p** is presented :

$$D_{pj} = - \partial E_p / \partial X_{pj} \quad (4.7)$$

Thus we can write

$$- \partial E_p / \partial W_{ji} = D_{pj} \; Y_{pi} \quad (4.8)$$

and to implement gradient descent in E we want to set the change in weights as follows :

$$\Delta_p W_{ji} = \text{learning\_rate} \times D_{pj} \times Y_{pi} \quad (4.9)$$

Let us now determine the value of D_{pj}. Applying the chain rule :

$$D_{pj} = - (\partial E_p / \partial Y_{pj})(\partial Y_{pj} / \partial X_{pj}) \quad (4.10)$$

The second factor is simply

$$\partial Y_{pj} / \partial X_{pj} = f'(X_{pj}) \quad (4.11)$$

the derivative of the activation function. For the first factor, let us first consider the case of output units. By differentiating the definition of E_p (4.3) and assuming that there are no connections from an output unit to another output unit, we obtain :

$$\partial E_p / \partial Y_{pj} = - (target_{pj} - Y_{pj}) \quad (4.12)$$

thus,

$$D_{pj} = f'(X_{pj}) \; (target_{pj} - Y_{pj}) \quad (4.13)$$

for units on the output layer. For hidden units let us again use the chain rule to write D_{pj} in function of D_{pk} of units **k** in the layer above unit **j** :

$$\sum_{\text{units k on the layer above unit j}} (\partial E_p / \partial X_{pk})(\partial X_{pj} / \partial Y_{pj})$$
$$= \sum_{\text{units k on the layer above unit j}} (\partial E_p / \partial X_{pk})\, \partial/\partial Y_{pj} \left(\sum_i W_{ki} Y_{pi}\right)$$
$$= \sum_{\text{units k on the layer above unit j}} (\partial E_p / \partial X_{pk})\, W_{kj}$$
$$= - \sum_{\text{units k on the layer above unit j}} D_{pk} W_{kj} \qquad (4.14)$$

thus for hidden units :

$$D_{pj} = f'(X_{pj}) \sum_{\text{units k on the layer above unit j}} D_{pk} W_{kj} \qquad (4.15)$$

To summarize, definitions (4.1),(4.2),(4.9),(4.13) and (4.15) define how to set the output of units as well as how to modify the weights to implement a gradient descent in the squared error measure defined in (4.3) and (4.4). For each pattern, the input pattern is presented to the input units and derived signals are propagated in a feedforward pass towards the output units. In a second phase the error is computed at the output units and propagated with the error signals (D_j) in a backward pass towards the input units. In both cases, the weights are used in the propagation. Hence the name "error back propagation" algorithm.

4.2 Parameters and Implementation for the Experiments

Let us now determine the activation function f(x). Since its derivative f'(x) is to be computed, the derivative of the activation function must exist. Thus the step function of the perceptron is not appropriate because it is discontinuous. The function suggested by the authors of the algorithm, and found in many other algorithms,

(such as the Boltzmann Machine algorithm) is the sigmoid (or logistic) function :

$$Y_j = f(X_j) = 1 / (1 + \exp(- X_j)) \qquad (4.16)$$

and its derivative can be computed and simplified as follows:

$$\partial Y_j / \partial X_j = \exp(-X_j) / (1+\exp(-X_j))^2$$
$$f'(X_j) = Y_j (1 - Y_j) \qquad (4.17)$$

thus the error signal D_j can be computed as follows :
For output units :

$$D_j = (target_j - Y_j) Y_j (1 - Y_j) \qquad (4.18)$$

For hidden units :

$$D_j = Y_j (1 - Y_j) \sum_{\text{units k on the layer above unit j}} D_k W_{kj} \qquad (4.19)$$

The activation function $f(X_j)$ does not permit a value of 0 or 1 for the output Y_j unless X_j is infinite, i.e. some weights are infinite. Thus desired output values should be assigned a value close but not equal to 1 and 0, for example 0.9 and 0.1.

A possible additional feature of the activation function is a bias or threshold to the weighted sum of inputs :

$$X_j = B_j + \sum_{\text{units i in layer below unit j}} W_{ji} Y_i \qquad (4.20)$$

In that case, that threshold can be learnt like any other weights by assuming that it is a weight from a unit which is always set to 1.

Note that because all units in the last hidden layer are connected to all units of the output layer, and because error signal are propagated in proportion to the weights, if the weights are

initialized at 0, all weights would adapt in the same way. This problem is simply solved by starting the learning system with small random weights.

If the learning rate is sufficiently small, the network will converge to a minimum of the error measure in weight space, however, this may be very slow. On the other hand if the learning rate is too large, the network might oscillate when it approaches a strong curve in the weight space landscape. A way to get fast convergence (i.e. using a high learning rate) without leading to oscillations is to include a momentum term (Rumelhart, Hinton & Williams, 1986) in the learning rule:

$$\Delta W_{ji}(t+1) = a \times \Delta W_{ji}(t) + \text{learning\_rate} \times D_j \times X_j \qquad (4.21)$$

where **a** is a constant which determines the influence of past movements of W_{ji} on the current change; it was set to 0.95 in the experiments described in chapter 6. This procedure is equivalent to applying a low pass frequency filter to the landscape of the weight space, thus filtering out high curvatures which might have caused very slow progress of convergence, or oscillations if the learning rate was too large.

Since this network is updated deterministically and it is not desirable that the network learns the exact input/output pairs by rote, some noise was introduced at the input units. A uniform random variable in U[-0.05,0.05] was added to the input values. Comparing results with or without the input noise, slightly better performance was observed when the inputs are noisy (actually best results are the same, but without noise, the error on the test set gets worse with more learning, as expected).

The coding of the input information is done like for the Boltzmann Machine algorithm and is explained in section 6.2 on coding.

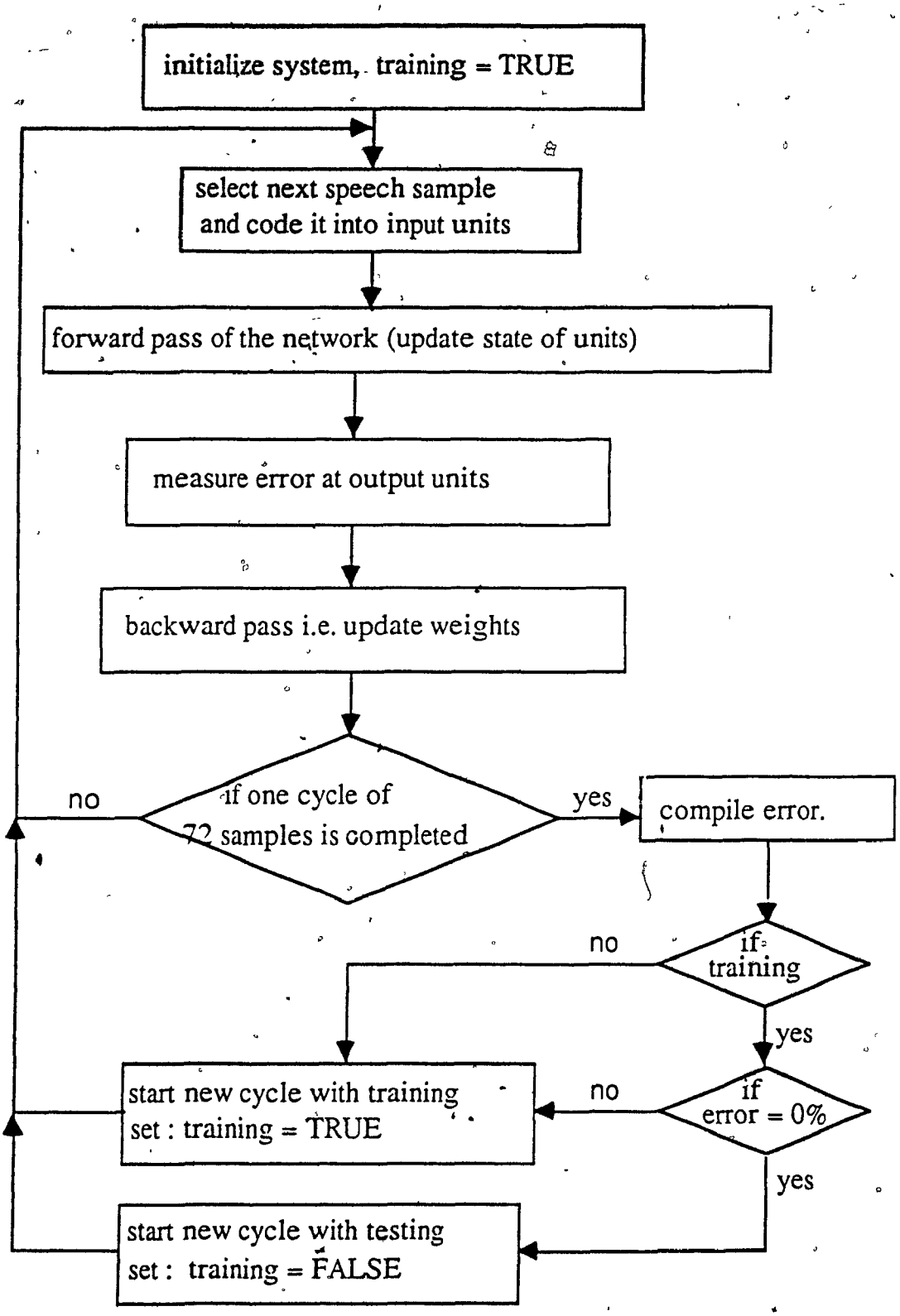

Figure 4.2 : *Flow chart diagram of high level structure of implemented back propagation algorithm.* The infinite loop is stopped when the error on the training set doesn't improve anymore (it oscillates, though).

The program which implements the Error Back Propagation algorithm is much simpler and shorter than the one for the Boltzmann machine. It also runs faster since the output corresponding to an input vector is obtained in one pass over the network (there is no relaxation). It was written in C on a VAX 8650 running under VMS.

5. Essential Problems in Automatic Speech Recognition

Speech perception is different from the other areas of perceptual research because of its role in language, thought and communication. Furthermore, the sequential nature of speech makes the problem of automatic speech recognition (ASR) distinctive from the static recognition of visual patterns.

Humans are able with little conscious effort to recognize speech of poor quality, distorted by noise or even obliterated. This is due to a great amount of redundancy and structure in the speech signal. The semantic, syntactic, lexical and phonological constraints enable us to recognize continuous speech even with high distortions and noise.

A central problem in speech perception is the fact that each phoneme does not correspond to an invariant set of acoustic features in a particular stretch of sound, for all contexts (problem of the lack of acoustic-phonetic invariance).

Because of coarticulation, stretches of sound associated to different phonemes overlap and the acoustic description of a phoneme depends on the surrounding phonetic context. This phenomenon also results in a great difficulty to segment the speech signal into separate phonemic or word units. It is possible to segment the speech signal into acoustic units, but they won't necessarily correspond to linguistic segments (phonemes).

5.1 Physiology of the Human Auditory System

5.1.1 The Ear

The auditory system comprises the auditory nervous system and the ear, itself divided into outer, middle and inner ear. The sound goes through the external auditory canal and makes the tympanic membrane vibrate. These vibrations from the outer ear are transmitted to the inner ear through the three small bones of the middle ear. Sound is transmitted through fluid in the inner ear: the fluid in the cochlea is set into vibratory motion. Sensory cells are located on the organ of Corti, inside the snail shaped cochlea. The vibrations of the basilar membrane in the cochlea induce an excitation of the sensory neurons (hair cells). The neural code which is generated is modified on its ascending path to the brain. The basilar membrane is thus the interface between the external acoustic environment and the internal neural representation. Experiments showed that each point along the basilar membrane vibrates maximally at one specific stimulus frequency, and this vibration decreases gradually for higher or lower frequencies. Similarly, each sensory neuron is maximally excited at a particular frequency, called characteristic frequency. However, sensory neurons display a finer frequency selectivity (resolution) than the basilar membrane.

5.1.2 Auditory Nervous System

There are two types of nerve fibers involved in the auditory nervous system. Information travels from the cochlear membrane to the brain through the afferent fibers and from the brain towards the ear through the efferent fibers. The efferent fibers are probably used to modulate the incoming signal, using past

information, to enhance variations in the sound rather than steady-state signals.

A nerve fiber in the ascending pathway responds to sounds within a certain frequency range. That range gradually increases with increasing intensity of the signal. In the ascending pathway, nerve fibers are arranged according to a tonotopic organization, i.e. fibers close to each other have close characteristic frequency and characteristic frequency regularly increases with distance along a cross section of the nerve. A similar organization is found in the auditory cortex, although the sound is not coded like at the sensory neurons. This frequency coding can thus be called a place code. The auditory system performs a kind of spectral analysis of the incoming sounds, although the output of each fiber is not a linear function of input power. However, there is also a temporal coding of low frequencies. Indeed, the discharge pattern of neurons on the auditory ascending pathway is phase-locked (synchronized) to the periodicity of low frequency sounds (below 4-5 kHz). Researchers in the field of auditory physiology presently assume that the auditory system uses both temporal and spectral analysis in parallel, with one representation more important for the analysis of certain sounds and the other more important for other types of sounds.

The ascending auditory nervous system modifies the information gathered at the receptors, with a complexity increasing as one moves towards the auditory cortex. A certain number of nerve fibers (efferent system) bring feedback from the auditory cortex to the ascending neural pathways, and from these pathways to the neural receptors. It is not known today how the signal emitted for the case of complex sounds at the sensory neurons is transformed as it travels to the cortex. It is thus not possible to predict, except for simple steady-state sounds, what will be the response at the end of the ascending nervous pathway.

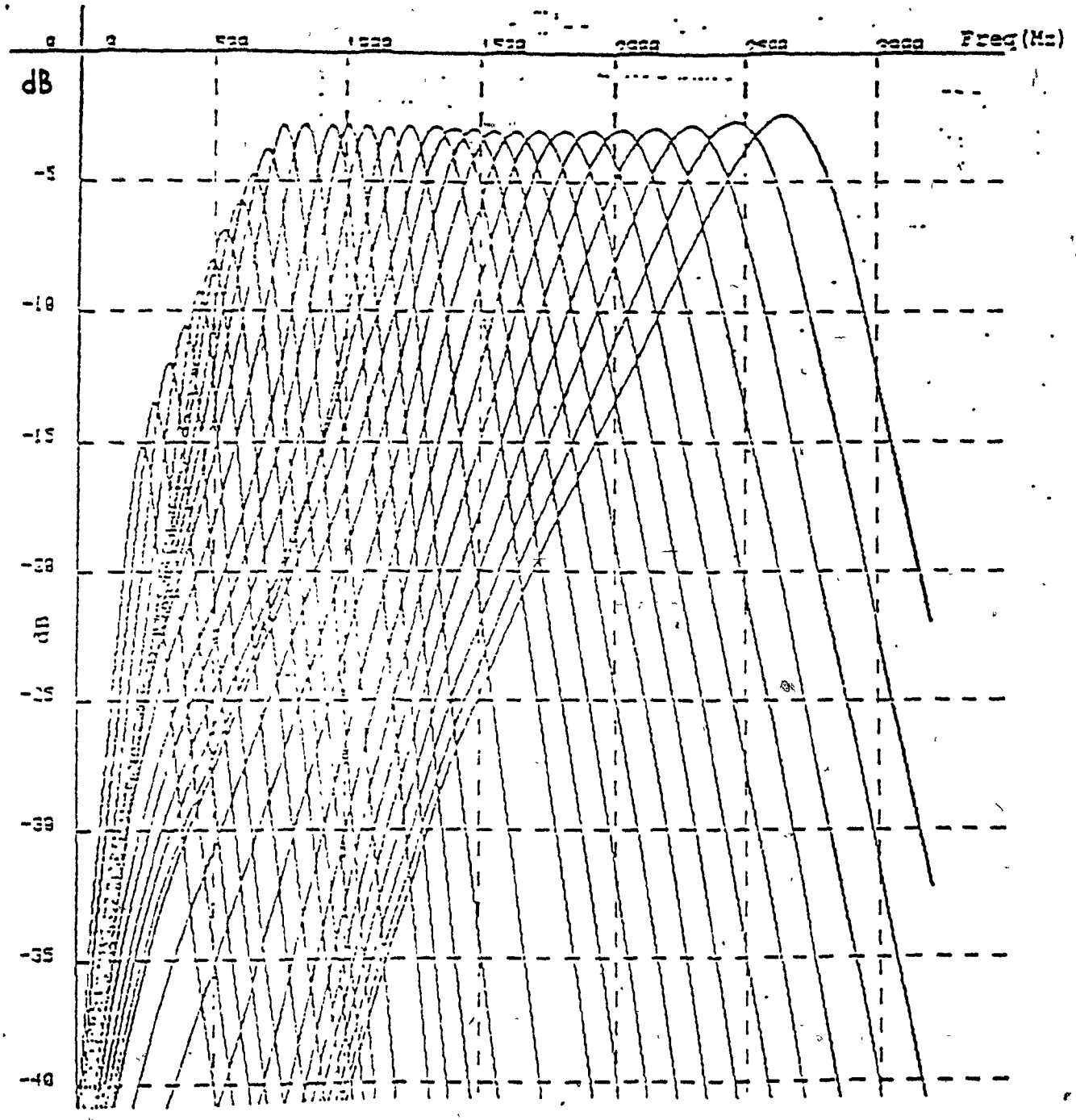

Figure 5.1 : *Model of auditory sensitivity.* From Seneff, (1986). Each curve represents the frequency response of a filter, in a filter bank modeling the auditory nerve sensitivity.

For individual neurons of the auditory pathway, a frequency response curve to single tones can be experimentally determined. These curves are not symmetrical in frequency, especially for characteristic frequencies above 2 kHz. The high frequency skirt of these frequency threshold curves is steeper than the low frequency skirt. The width of these curves (measured at a certain energy level below the threshold, such as 10 dB) is approximately constant for characteristic frequencies less than 500 Hz to 1 kHz. Above 1 kHz, the bandwidths increase about linearly with frequencies (Moller, 1983). This can be modeled by indexed filter banks with a characteristic frequency that is a logarithmic function of the index (number) of the filter bank, as shown on figure 5.1 from (Seneff, 86), (see section 6.1 on the non-linear coding of frequency for the experiments with connectionist models).

5.2 Automatic Speech Recognition

Today's ASR real-time application systems impose serious constraints on the end-user, for example, a limited vocabulary, a noise-free environment, a speaker-dependant recognition (and training), or the obligation to pause between words. These limitations reflect the inadequacy of current theoretical models of speech perception.

5.2.1 Pattern Classification

The simplest (and still used today) method in ASR relies on pattern classification and template-matching. The speech signal is first preprocessed to be described by a certain pattern, or a vector of features. In the training phase the descriptions

(templates) for the set of training samples representing the different classes (e.g. words, or phonemes) to be recognized are stored in memory. For each class, one or more 'average' patterns are computed, i.e. clusters are formed in feature space (with possibly several clusters per class), and an average pattern is computed for each cluster. In the recognition phase, the input feature vector (template) is compared to all the stored templates. The class of the stored template which is at the shortest distance in feature space from the input template is selected.

A basic requirement of this method is the proper selection of invariant features describing each speech sample, which may be difficult. Indeed, the acoustic signal is very variable (of variable duration, noisy, depending on speaker, context, position of word in the sentence, etc...).

A popular classification method in feature space is the linear discriminant function (discriminative distance). Given d classes, each class C_i is associated with a discriminant function D_i such that:

$$(C_i \ni \mathbf{X}) \text{ iff } (D_i(\mathbf{X}) > D_j(\mathbf{X})) \quad \forall\ j \neq i, \tag{5.1}$$

where $\mathbf{X}$ is the acoustic feature vector $(x_1, x_2, \ldots x_f)$, and

$$D_i(X) = W_i \,.\, X' \tag{5.2}$$

where $\mathbf{X}' = (x_1, x_2, \ldots, x_f, 1)$. The boundaries between classes are thus defined by hyperplanes in the feature space. The use of $\mathbf{X}'$ instead of $\mathbf{X}$ (with the addition of a fixed element, 1, at the end of the vector) is to permit these hyperplanes to avoid going through the origin.

This linear discriminant function exactly matches the description of the perceptron. The additional element in $\mathbf{W}_i$ can be seen as a bias :

$$D_i(X) = \sum_{i=1 \text{ to } f} w_i x_i + w_{f+1} \qquad (5.3)$$

As a consequence, the linear discriminant function will have the same limitation as the perceptron, i.e. the inability to classify sets of vectors which are not linearly separable.

5.2.2 Fast Fourier Transform

Spectral analysis is very often used in the preliminary phases of signal preprocessing. According to the Fourier Theorem, any periodic signal (or signal of finite duration) can be separated into the sum of an infinite but discrete set of sinusoid signals of various amplitudes. A signal which has a limited frequency bandwidth can be sampled to obtain a discrete sequence without losing any information. The signal can be represented as a finite sum of discrete sinusoids : the discrete Fourier transform. A time sequence of N consecutive real numbers can be mapped to a spectral description, made of the amplitudes of N discrete frequency components. An algorithm exists to compute the discrete Fourier transform in time O(NlogN) : the Fast Fourier Transform (FFT) algorithm. In general the FFT is computed for subsequences of the discrete time signal, to obtain a sequence of discrete spectral descriptions of the signal.

It must be remarked that the choice of the time window to be analyzed through the FFT is very important. A short window provides a good time resolution but a poor frequency resolution (N small). On the other hand, a very large window (e.g. the whole speech signal) provides very precise spectral information and no temporal information at all. This trade-off can be expressed in the following uncertainty relation :

$$\Delta T \times \Delta F = c \quad \text{where c is a constant close to 1} \qquad (5.4)$$

where ΔT and ΔF are the time and frequency resolutions.

In speech recognition, this window will be chosen according to knowledge of the necessary time and frequency resolution for the perception of speech, and of our knowledge of auditory physiology (in general around 10 ms windows are used).

5.2.3 Dynamic Time Warping

As was mentioned earlier in the introduction of this chapter and in section 5.2.1, different pronunciations of the same sound may result in different temporal distortions : it will not be possible to match exactly the two sequences of patterns. We can imagine the signal as a rubber band with some parts more stretched than others depending on the pronunciation. It was observed that the rate of speech variations mainly affects the steady-state parts (e.g. vowels, fricatives) of the signal.

The Dynamic Time Warping (DTW) method of measuring 'distance' between two different templates is based on a dynamic programming algorithm. Each template consists of a sequence of vectors. The distance between every vector of one template and the second template are computed and placed in a matrix. For example if we compute this matrix for the comparison of a template *vs* itself, we find a null diagonal. The objective of the algorithm is to find a path in that matrix, (in the same direction as that diagonal, starting and finishing at the same corners of the distance matrix) that will associate each vector of the first template with one or more of the second one. At each point in the path, the local distances are added. The desired path is that one which minimizes the total distance. The path is constrained to obey chronological order. The resulting total distance is used as

the distance measure between the two templates, and is used for the classification decision.

5.2.4 Hidden Markov Models

A Markov chain is a stochastic process describing a sequence of trials in which

1) the outcome of each trial belongs to a finite set of states $(S_1, S_2, \ldots S_m)$,

2) the outcome of any trial only depends upon the outcome of the preceding trial : to each pair of states S_i and S_j is associated a probability p_{ij} that S_j will occur immediately after S_i.

In a Hidden Markov Model (HMM), the states are, as the name suggests, hidden. One can only make an observation, which is generated by a random function of the current hidden state. Then according to the transition probabilities p_{ij}, the underlying Markov chain changes states. The observer observes the output (e.g. spectra) a_k when the process is in state S_j with probability $q_{jk}(S_j, a_k)$.

In a Continuous Parameters Markov Model (cf. De Mori, Merlo, & Palakal, (1986) with the same data and spectral lines technique as the experiments of this thesis, section 6.4), the observation function $q(S_i, S_j, y_k)$ is the probability that the spectral line y_k is observed in the transition from the state S_i to the state S_j.

One of the main advantages of Hidden Markov Models (HMM) for ASR is that they take into account the sequential nature of the speech signal and can include the time warping process (see section 5.2.3) and learn a parametric and statistic description of the speech through the presentation of a large set of examples.

The most popular training algorithms are the Backward-Forward (Baum, 1972) and the Viterbi (Viterbi, 1967) algorithms. However, HMMs can be used only after the chosen speech process has been modeled with a state topology, allowable transitions and this means including explicit knowledge on speech production and recognition in the HMMs. In fact, each class to be recognized must be modeled differently.

6. Experiments in Automatic Speech Recognition Performed with Connectionist Models

Connectionist models were applied to perform **speaker-independent** recognition of place of articulation for vowels. Although speaker normalization is a difficult task, it is known from speech analysis that sonorant portions of speech spectrograms exhibit similar images when different speakers pronounce the same sound or the same sequence of sounds. The experiments reported in this thesis investigate the application of neural network models for performing speaker normalization, using spectral lines characterized by their frequency and amplitude to represent the speech samples.

Sounds are classified into three categories according to their place of articulation in the mouth : back position, central position or front position. The speech data consisted of 144 speech samples, 72 used for the training of the networks, 72 used only for testing. These samples were extracted from the continuous speech of 38 speakers (24 males and 14 females) pronouncing connectedly spoken letters and digits. Details of segmentation can be found in De Mori, Laface, & Mong (1985).

Static representation of speech data is based on spectral lines, already used by Merlo, De Mori & Palakal (1986) to perform speech recognition tasks. The original time signal is sampled at 20 kHz over 12 bits and its power spectrum (FFT every 10 ms) and zero crossings are computed. This information is used to identify the sonorant portions of the signal that exhibit resonances. This segmentation is based on rough spectral features that eliminate segments containing frication noise, silences and buzz-bars. The remaining spectrogram (time- frequency-energy pattern) is sent to the spectral line extraction program.

The networks consist of a certain number of
1) input nodes, coding the energy/frequency information from the spectral lines,
2) a certain number of hidden nodes that learn through examples an internal representation of the input/output environment and
3) three output nodes whose activation level represent the three classes of vowels (back, center or front position). The data was initially classified (labeled) using an algorithm (De Mori, Laface, Mong, 1985).

6.1 Spectral Lines Extraction

To extract the spectral lines, the spectrogram is treated as an image. This image is processed by a thinning and a skeletonization algorithms, and then by a line tracing algorithm. These algorithms are described in Palakal & De Mori (1985). Segments were extracted where spectral lines were quasi stationary, allowing the use of simple static data as the input to the connectionist networks, rather than time-varying sequences of descriptions. The spectral lines extraction program was provided by Mathew Palakal (cf. Palakal; De Mori 1985).

The thinning algorithm used is the Safe-Point Thinning Algorithm, described in Naccache & Shinghal (1984). This algorithm was chosen because connectivity of lines are maintained by keeping the points at the junctions, and excess erosion is not allowed. Figure 6.1 shows the spectrogram for a pronunciation of the letter 'a', before the spectral line extraction algorithm is applied. Figure 6.2 shows the spectral lines extracted from the spectrogram of the letter 'a'. Time increases on the vertical axis, with each step corresponding to 10 msec. Frequency increases on the horizontal axis; energy is coded by letters and digits. For example the letter A represents half the energy of the letter B, the letter Z half the energy of digit 0.

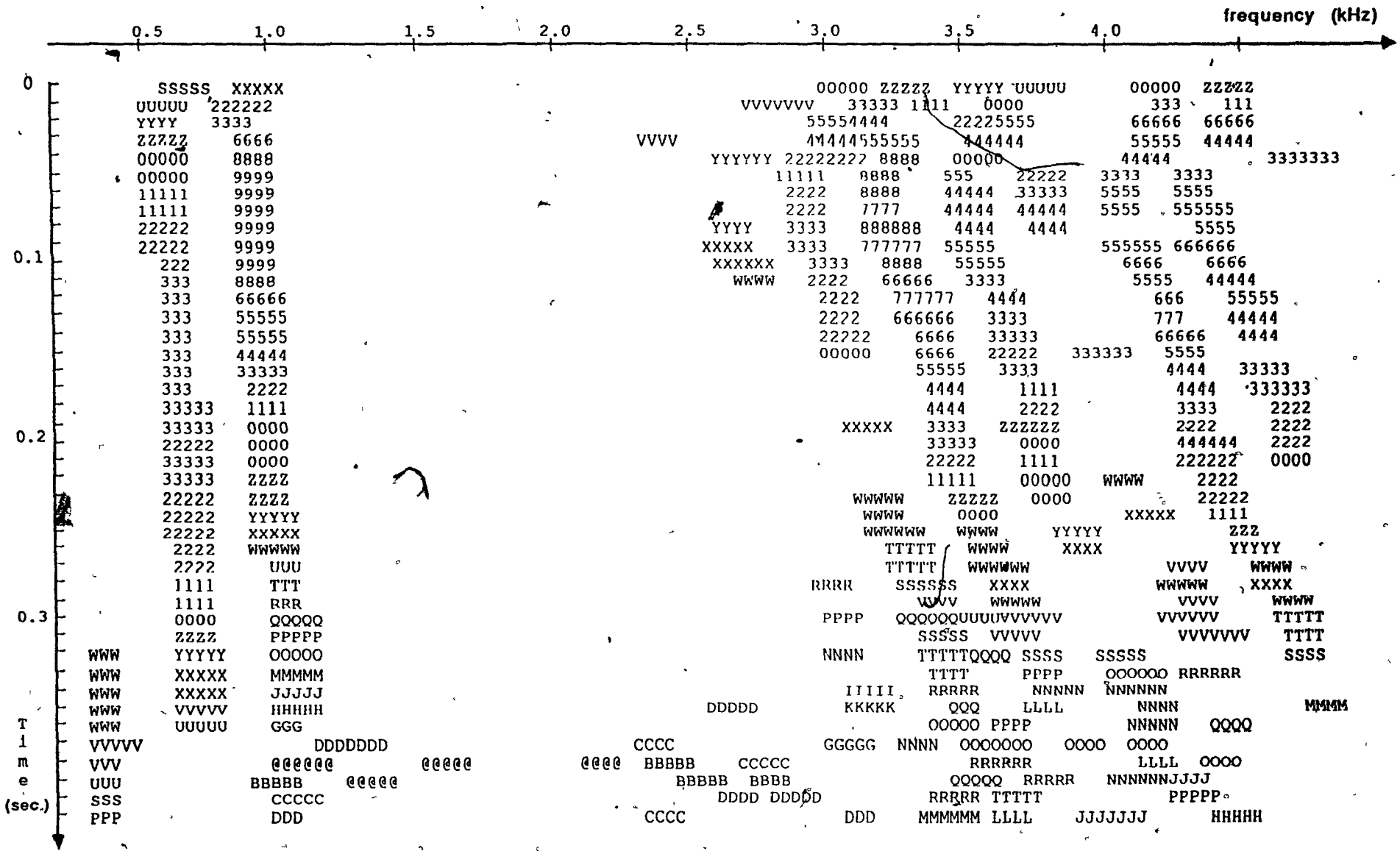

Figure 6.1 : Spectrogram of a pronunciation of the letter 'a', before spectral lines were extracted. Letters and digits represent energy levels (@<A<B<...<Z<0<1<..<8<9).

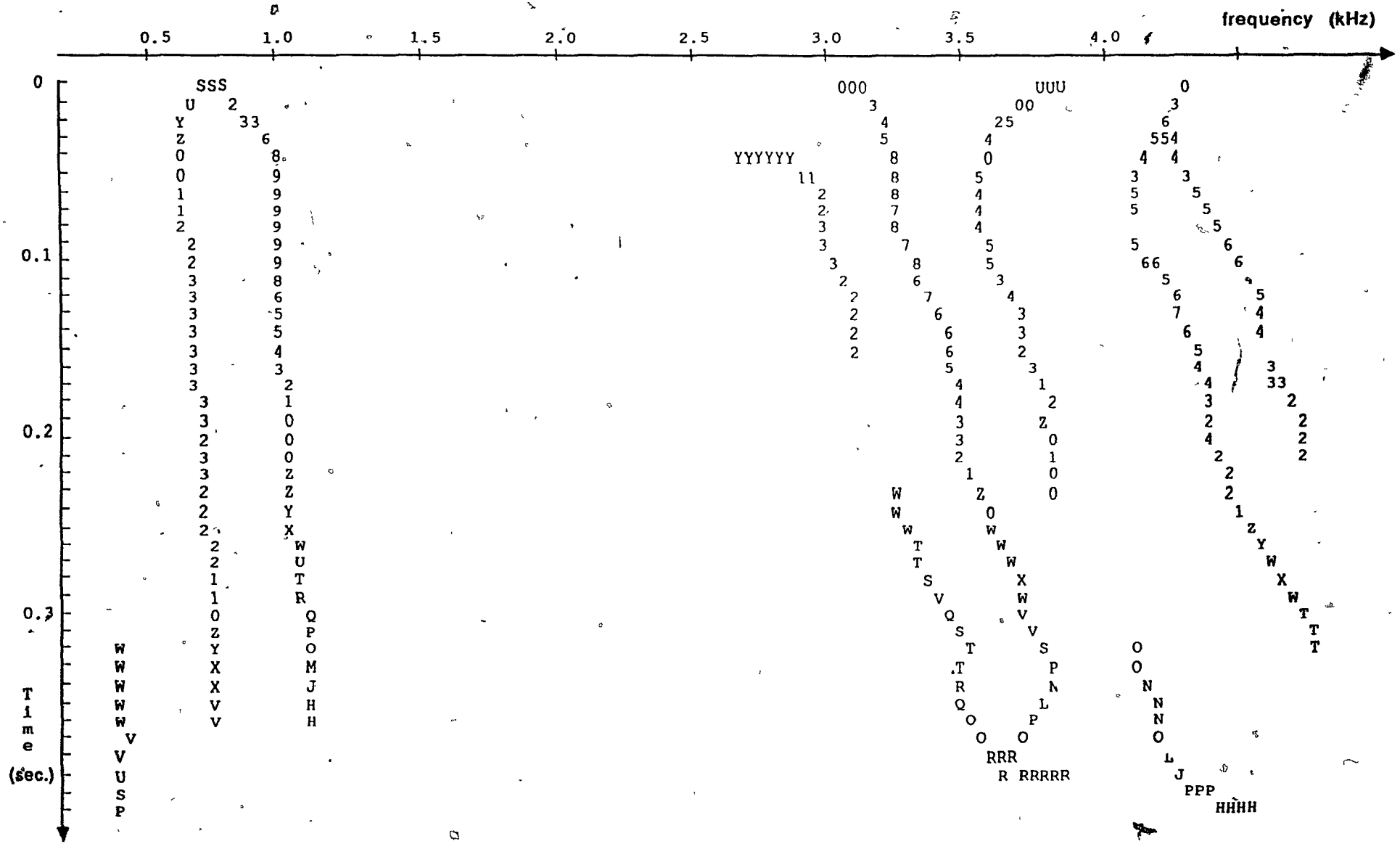

Figure 6.2 : Spectral lines extracted from a pronounciation of the letter 'a'. Time is represented on the vertical axis, frequency on the horizontal axis and energy by letters and digits (9>8>..>0>Z>Y>..>B>A).

The Line Tracing Algorithm (ALTRACE) is applied to the result of the thinning algorithm. This step discards all scattered points, keeps all the lines and smooths the pattern. The algorithm retains collinearity, continuity, curvelinearity as well as other properties present in the pattern.

The output of the Spectral Line Extraction algorithm is a description of n lines, each with its energy and frequency. The first line is called a base line and is selected as the line of highest energy in the low frequencies (< 1 kHz). The others may be described relative to the base (or anchor) line. The difference between their frequency and energy and those of the anchor line are provided by the spectral lines extraction program. The base line frequency is absolute and its energy is absolute but was not provided to the network. Thus the network received an energy-normalized description of the spectral lines.

6.2 Coding

Given the set of lines, each with its energy/frequency values, continuous or binary activation values were assigned to the input nodes of the neural nets. The way this coding is done has an impact on the efficiency of the network (error rate) as well as on the efficiency of the learning (number of learning cycles necessary). It also has an impact on the number of input nodes, and thus on the speed of the simulation.

The input nodes are first assigned to frequency intervals, based on a model of auditory sensitivity (see figure 5.1 in chapter 5). One node or a group of input nodes represent a range of frequencies. The distribution of frequencies was inspired from an approximation of the ear model : under 1 kHz (low frequencies), the characteristic frequencies grow linearly as follows :

$$\text{frequency\_index} = \text{INT}[(\text{frequency} - \text{minimum\_frequency}) \times \text{low\_freq\_index\_range}/\text{low\_freq\_range}] \quad (6.1)$$

where INT[x] represents the integer part of the real number x. Above 1 kHz characteristic frequencies grow logarithmically (so that higher characteristic frequencies are farther apart and their bandwidth is wider) :

$$\text{frequency\_index} = \text{INT}[c1 \times \log(\text{frequency} - 1000) - c2] \quad (6.2)$$

where c1 and c2 are chosen to make the minimum and maximum high frequencies fall on the boundaries of the high frequency indices. Thus for high frequencies, the frequency index (on the grid) is a logarithmic function of the frequency.

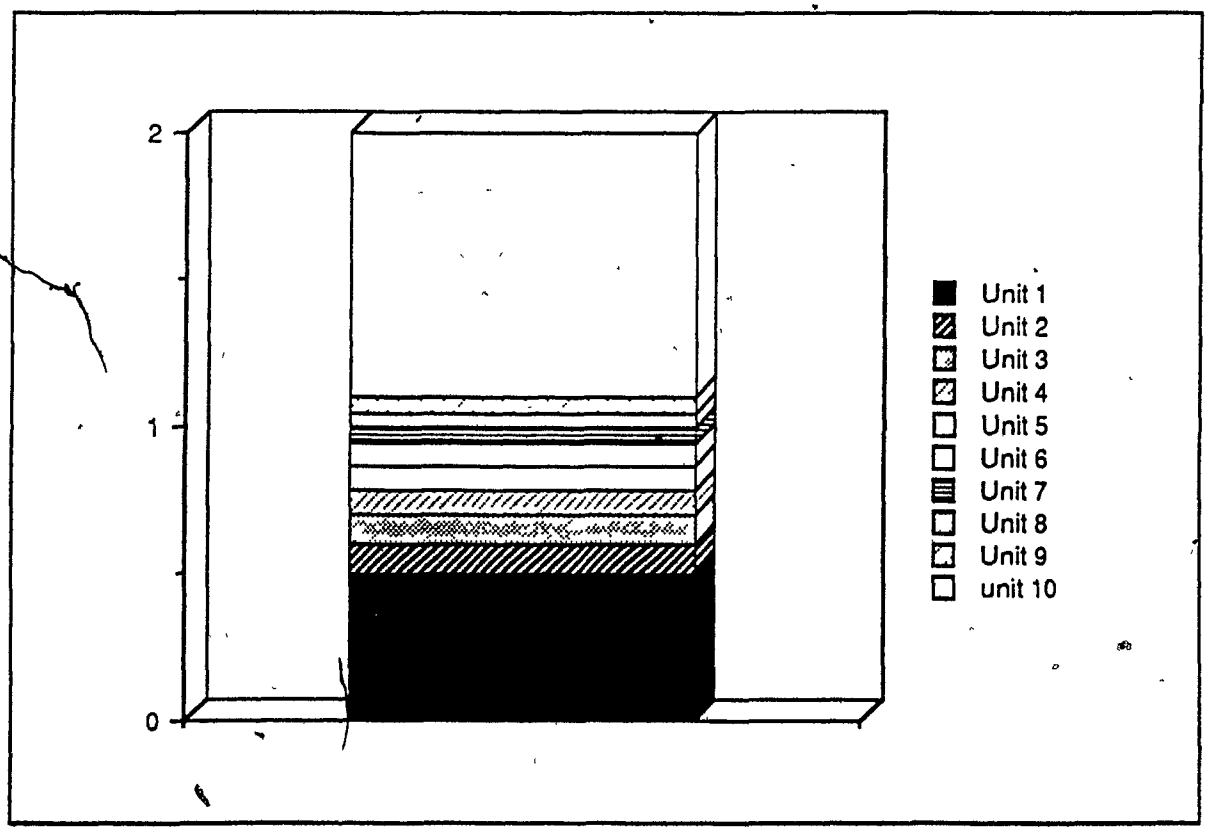

Figure 6.3 : *Multi-level energy coding.* Relative (w.r.t. base line which has energy 1.0) energy range represented by each of ten units coding a particular frequency range.

To represent energy, several nodes were used within each frequency range, each representing a certain energy level (or range). Thus the input nodes can be seen as points on a 2-dimensional grid of frequency and energy (see figure 6.4). The distribution of energy was chosen empirically (see figure 6.3) so as to make the number of samples falling in each range about equal. A typical number of ranges chosen was 10. Better results are obtained with this multi-level coding of the energy than with only one node per frequency range, coding the energy continuously between 0 and 1. The system converges slower to a solution and makes a larger error, as shown in table I. Note that in those experiments with only one node per frequency range, more frequency resolution as well as more hidden units were provided, so as to obtain a similar number of connections in the network.

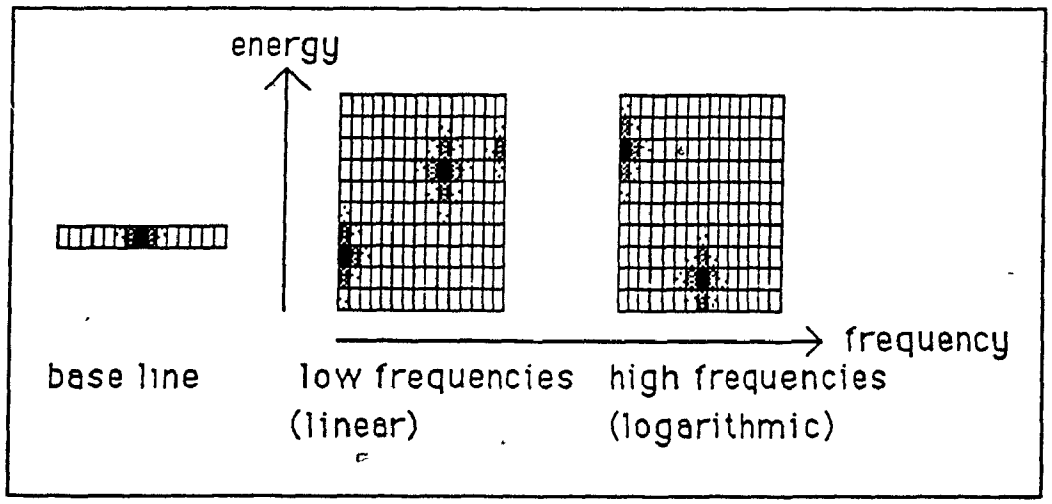

Figure 6.4 : *Coarse coding (neighbors get activated), showing base line units, low frequency units and high frequency units.*

In addition to exciting one node for each energy/frequency input, neighboring nodes also received an input (with intensity decreasing with distance on the 2-dimensional energy/frequency grid). Thus for each input spectral line, there were 12 additional

nodes on the grid which were activated, as shown on figure 6.4. This strategy called **coarse coding** (or neighborhood code, see Prager, Harrison & Fallside (1986) for a discussion about it) gives much better results than the simple excitation of one energy/frequency node per spectral line (see table I). One should recall from section 5.1 about the human auditory system that nerve fibers are not only excited for a small "individual" frequency range but also display some activity (decreasing for farther neighbors) when their neighbors are active. Indeed, the frequency response (threshold curve) of adjacent fibers overlap.

The energy level coded into the network is always relative to the energy of the base line, and the network does not receive information about the absolute energy of the base line. In fact a separate set of nodes (coding only for low frequencies, and no energy levels) is provided to represent the base line. Typically there are

- 15 nodes to represent (low frequencies) the base line,
- 15 (frequencies) x 10 (energy levels) = 150 nodes to represent low frequencies and
- 15 (frequencies) x 10 (energy levels) = 150 nodes to represent high frequencies.

This makes a total of 315 input nodes, as shown on figure 6.4.

	Boltzmann Machine	Error Back Propagation
Experiment 1 (coarse coding, multi-level energy coding, and relative frequencies)	4.2%	6.9%
Experiment 2 As before but No Coarse coding	9.7%	9.7%
Experiment 3 No multi-level energy coding	8.3%	15.3%
Experiment 4 Absolute instead of relative frequencies	5.6%	6.9%

TABLE I : Error on Test Set for Various Coding Schemes

The output value of nodes on the input layer is continuous (between 0 and 1) for the error back propagation algorithm. However, the Boltzmann machine units normally have binary inputs and outputs. To code a continuous value at the output of the input layer units, the following stratagem was employed. Since the output of Boltzmann machine unit k is chosen to be 1 (instead of 0) with the continuous probability p_k,

$$p_k = 1 / (1 + \exp (- \Delta E_k / T) \qquad (3.3)$$

the desired continuous input can be assigned to p_k. Afterwards the unit will choose a sequence of 1's and 0's each according to probability p_k, in the relaxation process. This procedure will be effective since the Boltzmann machine operates with a long relaxation cycle (often 10 to 50 cycles in our experiments). This method also has the advantage of automatically providing some noise to the input (something that had to be implemented with the

error back propagation algorithm, see section 4.3). Note that small noise in the input is desirable because the training consists in repeatedly presenting the same set of examples, rather than presenting new examples each time.

6.3 Experimental Results

The experiments described in this chapter show that the error back propagation algorithm was faster but less accurate than the Boltzmann machine algorithm. The results obtained with the two methods are shown in table II. Table II shows a comparison of speeds for the best performances of the two algorithms. These results were obtained with 2x200=400 hidden nodes (2 hidden layers, 103600 connections) for the error back propagation algorithm, and 100 hidden nodes (75490 connections) for the Boltzmann machine algorithm. The running time shown in table II is for the presentation of one speech sample (from the test set), on the VAX 8650 CPU.

	Boltzmann Machine	Error Back Propagation
Speed (CPU time for one sample)	3 sec.	0.21 sec.
Error on Test Set	4.2 %	6.9%

TABLE II : Speed and Best Results for the 2 Algorithms.

The program measured the error on the training set and repeated the presentation of the set of examples until this error reached 0%. At this point the test set was presented and the error measured. Afterwards one can still present training examples. For the Boltzmann machine (stochastic) this may result in a few errors that slightly modify the weights. After the error rate again reached 0% the program presented the test set to the neural network. The error on the test set was thus slightly oscillating from one presentation of the test set to another (after some more learning, with the training set).

Similarly for the error back propagation, there continues to be learning after 0% error on the training set has been measured, since what drives the learning is the difference between target and actual continuous outputs (whereas a successful output is observed if the correct class has a larger output value than the others).

Typically, the Boltzmann machine converged to 0% error on the training set after 15 to 20 presentations of the training set. The shape of the learning curve (error rate *vs* number of presentations of the training set) is shown in figure 6.5. Afterwards, further training for a certain number of presentations (about 10) could provide slightly better results on the testing set. The error back propagation network converged to 0% error on the training set after about 10 presentations of the training set.

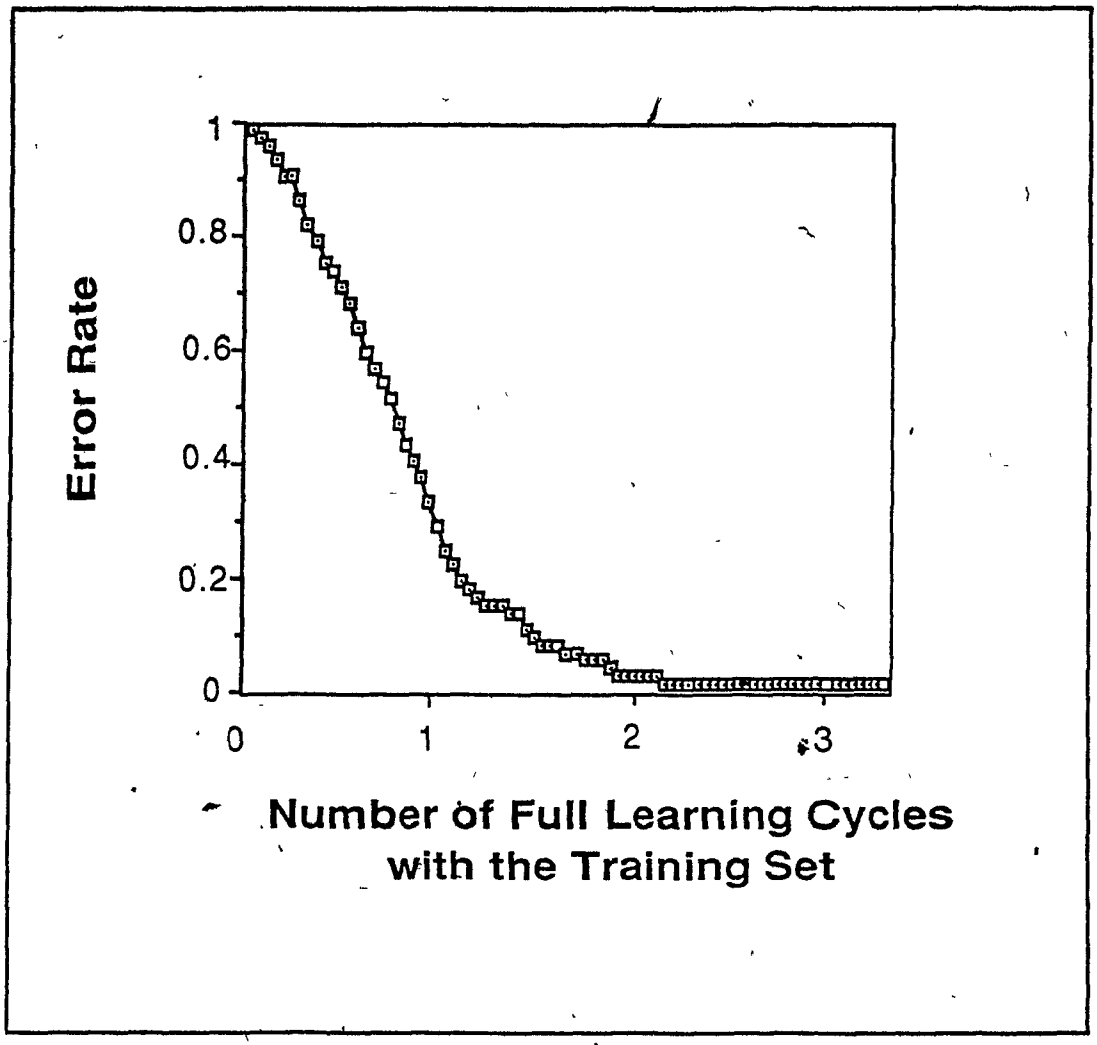

Figure 6.5 : *LEARNING ; Error Rate for the Training Set* vs *Number of Full Learning Cycles, for the BMA.* Note that the error stayed at 1.4% until the 20th cycle, when it reached 0% error.

6.4 Comparison with Results Using Hidden Markov Models

Experiments were performed with similar data, using the more traditional Hidden Markov Models (using the Forward-Backward algorithm, Bahl, Jelinek and Mercer 1983), and using the same spectral line extraction technique to preprocess the speech

signal. These results were reported in (Merlo, De Mori & Palakal, 1986), (cf. table III).

Exactly the same algorithm was used to compute spectral lines information for vowels (sonorant sounds) extracted from random sequences of connectedly spoken letters and digits. The training set consisted 14 female and 24 male speakers. The test set consisted of 9 new female and 9 new male speakers. The error was consistently different for the three classes of output (whereas with the connectionist models differences were observed but not consistently from experiment to experiment).

place of articulation	HMM alg.	HMM alg. + rules	
Back	5%	3%	error on
Central	2%	2%	training set
Front	2%	1%	
average	**3%**	2%	
Back	16%	6%	error on
Central	2%	2%	test set
Front	4%	4%	
average	**7.3%**	4%	

TABLE III : Comparison with HMM Algorithm, Same Data

Note that in addition to the Hidden Markov Model procedure, some knowledge about the expected frequencies of spectral lines was used to improve performance. If the *a-priori* probabilities computed by the Hidden Markov Model procedure for the different classes were close, rules were executed to take the decision. However this usage of empirical rules makes the method less generalizable and not automatic in nature, since the rules were defined in function of the observed weaknesses of the HMM algorithm with the speech data.

With the experiments conducted using the connectionist models it was always possible to reach 0% error on the training set after sufficient learning. This was in fact the criteria to start measuring error on the testing set. This suggests that much better results on the testing set could be obtained with the connectionist models if a much larger training set was used. Indeed, with a very large training test, the testing samples would be very close to samples which would have been already seen in the training set.

7. Direction for Future Work

7.1 Larger Neural Nets and Integration of Several Modules

Good results were obtained with neural nets of a size of a few tens of thousands of connections to solve a qualitatively difficult problem such as speaker normalization; however, they are of a limited bearing (i.e. classification of sonorant sounds in three categories). Will it be possible to extend the size of these models for more complex problems such as the recognition of phonemes in non-segmented continuous speech signal ? I.e. how will their size and rate of learning grow with the complexity of the problem? Will their performance (as measured by the error rate) get better or worse?

To answer these questions, a primary avenue of research is the consideration of the time dimension of the speech signal, since it is necessary to consider time when dealing with speech units longer than the vowel. In the experiments reported in this thesis, a static description of the sounds analyzed was considered . For a true automatic speech recognition system, the sequential nature of speech will obviously have to be considered in the design of a connectionist model. This question is treated in more details in section 7.2.

For the performance of the networks, better recognition rates should be expected in large systems that will take into account the different contexts (acoustic, phonological, lexical and semantic). One should recall (chapter 5) that humans perform better when recognizing a phoneme when it is in a word rather than isolated, and perform better to recognize a word when it is in a sentence rather than when it is presented alone.

If the length of the training phase of connectionist models appears to grow too much for larger problems, it will be possible to consider a solution that involves different modules that can be trained separately (or hierarchically) for subproblems of the global ASR task. For instance, the network developed in the experiments described in this thesis could constitute one such module.

7.2 Incorporating the Time Dimension of Speech in Connectionist Models

Attempts at segmenting the speech signal by acoustically-defined criteria have not proven to be very successful, especially for connected speech. On the other hand, the speech signal is sequential in nature and the human auditory system uses parts of it to recognize other parts. The determination of the phonetic classification of an acoustic segment depends on the acoustic segments preceding and following it. Psycholinguistic experiments have proven (e.g. see in Sawusch, 1986) that we use not only the left context but also the right context in the identification of a phoneme or of even of a word.

Consequently, future research should tackle the question of how to represent and code into connectionist models the time dimension and the various acoustic, phonetic, lexical, syntactic and semantic contexts that influence the perception of a small segment of speech.

7.2.1 Window in the Input

The following solution was proposed by Sejnowski with NETTalk (Sejnowski & al., 1986). He uses the error back propagation algorithm to convert written English text into speech.

The input text is presented to the network through a **window** that shows not only the current input vector to be processed but also a certain number of precedent and subsequent input vectors (see figure 7.1). They constitute the left and right context. The neural net was trained to generate the correct sound (or phoneme) associated with each letter in the different possible contexts (other letters around it). The same architecture was used by (Bourlard & Wellekens, 1987) for speech recognition, to map acoustic vectors to phonemes. The algorithm was applied to a speech sentence with a known phonemic segmentation. This can be a weakness of the previous approach, since the training data has to be phonemically segmented (because the neural net needs the target output when it is learning).

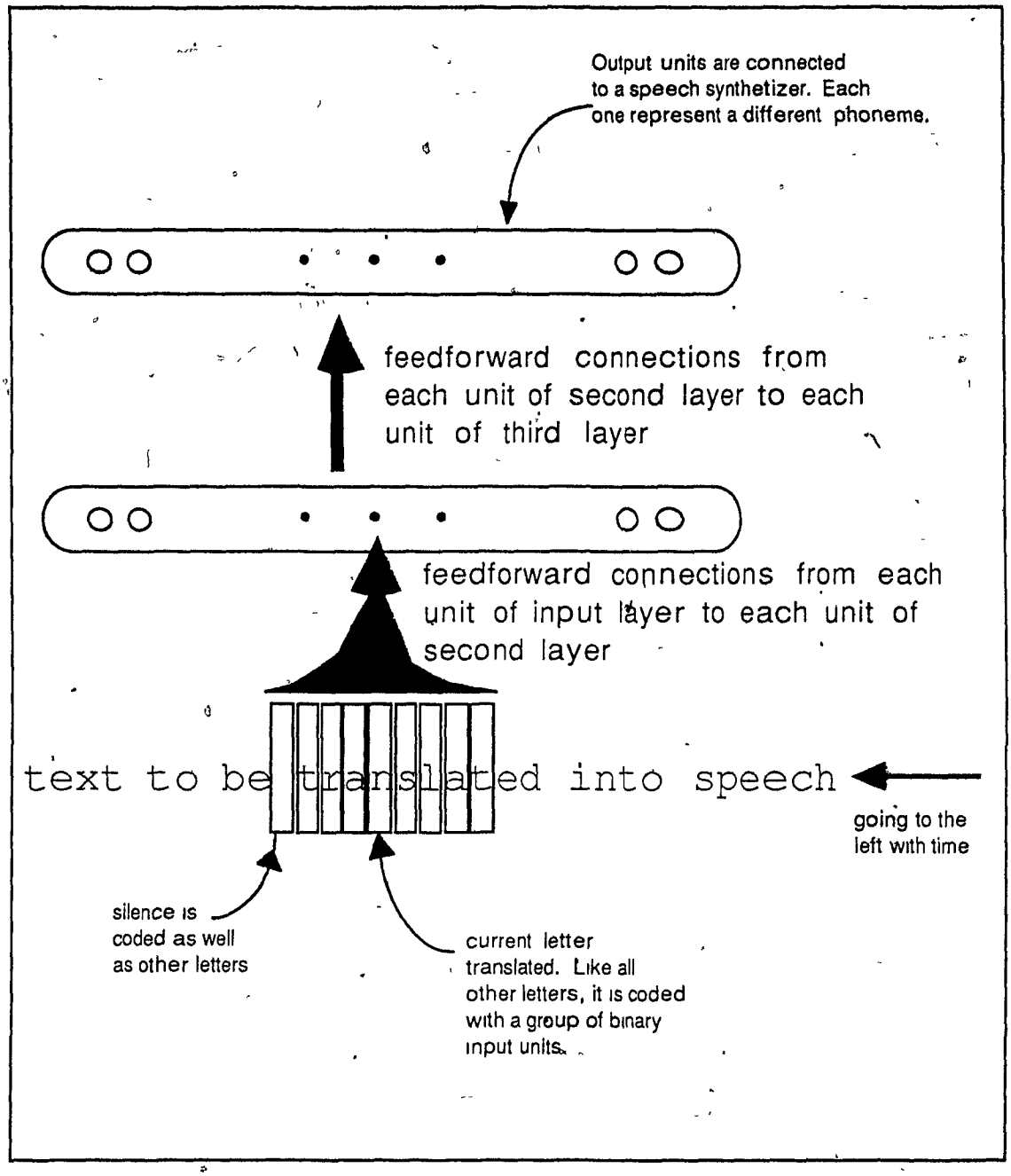

Figure 7.1 : *Window on the input stream.* Here as used for NETtalk (Sejnowski & al., 1986) to map text to speech.

A drawback of this method is that the context is deliberately chosen by the user and this imposes a fixed limit on the context. For example, in the above mentioned example

(Bourlard & Welleken, 1987), the network uses only acoustic context.

Another example of a window solution is the TRACE model (McClelland & Elman, 1986). It consists of a hierarchical network, organized in three levels for features, phonemes and words. In this model of local representation, each unit represents a hypothesis about a particular feature, phoneme or word, at a particular time relative to the beginning of the utterance. Old portions of the input continue to be processed as new input arrives. This allows right (as well as left) context to influence the recognition of speech. Connections are bidirectional and excitatory for units of different layers which are mutually consistent, inhibitory for units on the same layer which are inconsistent. Words and phoneme units are centered every three time-slices, thus adjacent units overlap. There are several features, organized in banks, with 9 units to represent 9 levels of activation of these features, repeated for every time slice. Input is presented sequentially at the feature level.

7.2.2 Recurrent Links with Delays

Another solution is to incorporate the contextual effects and time integration in the network itself by modifying the way each unit operates. For example, the connections between units can impose a delay (which is neurally plausible). Eventually, these delays can be modified with a learning rule. The same effect can be obtained by permitting recurrent links, since each unit takes at least one time unit to compute its output from its input. The Boltzmann machine has recurrent links, but the state (output value of all nodes, including hidden nodes) of the network is initialized for each input frame, in order to perform the simulated annealing procedure.

Watrous and Shastri proposed a generalization of the multi-layer perceptron and error back propagation algorithm which admits recurrent links and delays (Watrous & Shastri, 1986). Their model includes D connections with D different delays (0 to D-1) where there was only one connection in the basic multi-layer perceptron :

$$X_i(t) = \sum_{j,d} W_{jid}\, Y_j(t-d) \qquad (7.1)$$

where Y_i is computed as before (definition (4.16) in chapter 4).

The weight change rule is modified as follows :

$$\Delta W_{ijd} = \text{constant} \times \sum_s D_j(t-s)\; Y_i(t-s-d) \qquad (7.2)$$

where $D_j(t-s)$ is the error signal at time t-s for unit **j**, given by :

$$D_j(t) = \sum_{a,k} W_{jka}\, D_k(t+a)\, f'_j(t) \qquad (7.3)$$

for hidden units, where f'(x) is the differential of f(x), as in equation (4.17) of chapter 4. For output units,

$$D_j(t) = (Y_j(t) - target_j(t))\; f'_j(t). \qquad (7.4)$$

7.2.3 Context Feedback

This method assumes that the speech signal can be considered as the manifestation of an underlying Markov process, with the property that its state at time t depends only on the state at time (t-1) and new information (new speech frame) :

$$S(t+1) = F\,(\,S(t),\, I(t)\,) \qquad (7.5)$$

where S(t) is a state which summarize all available knowledge about past speech, and I(t) is the spectrogram frame input at time t. This strategy was implemented with the Boltzmann machine (Prager, Harrison, & Fallside, 1986) by using **carry units.**

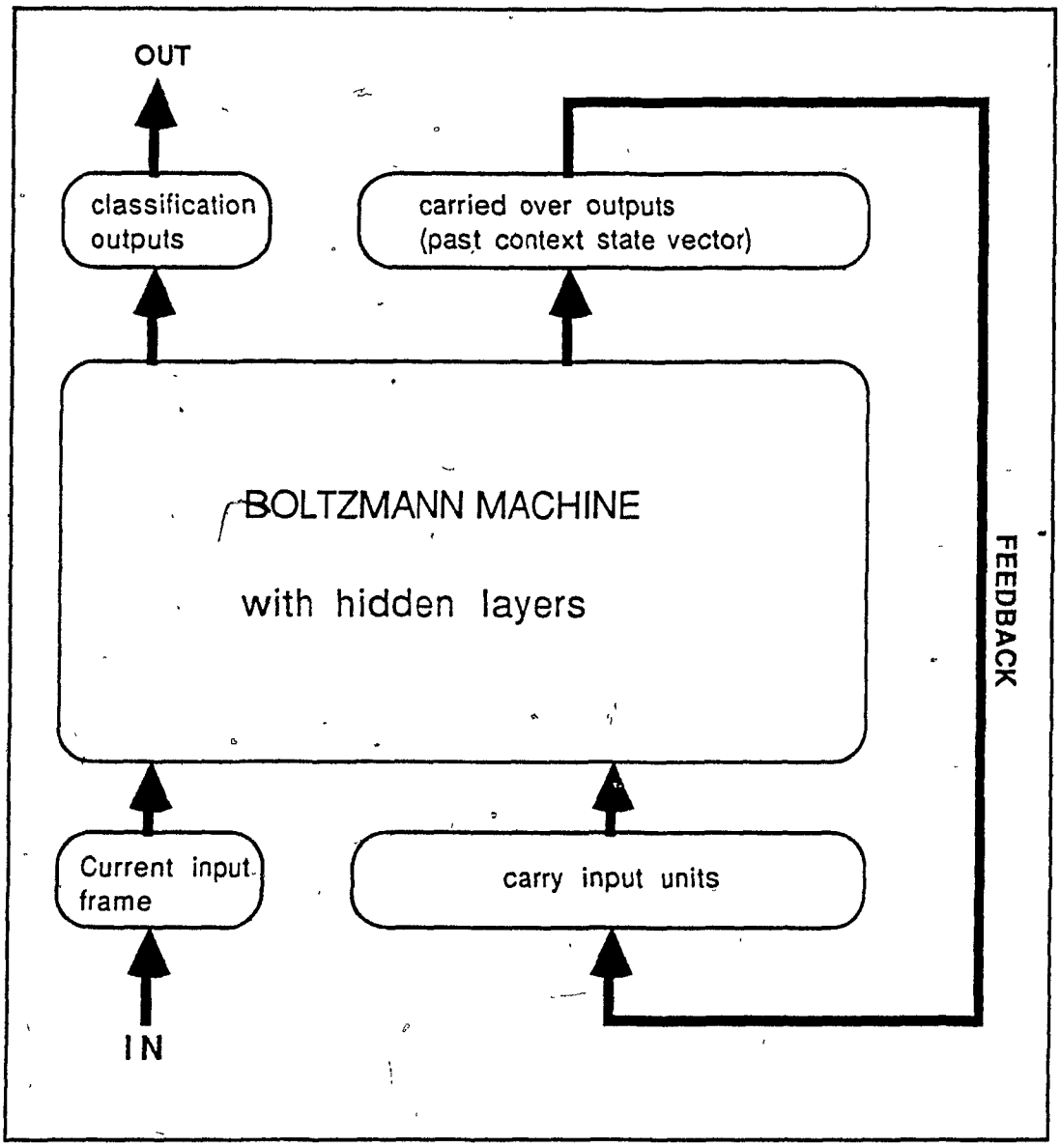

Figure 7.2 : *Context feedback with carry units, on the Boltzmann machine* as proposed by Prager & al. (1986).

As shown in figure 7.2, a part of the output is fed back to the input of the machine, via the carry input units. The machine

learns to utilize this information about the values of units on the carry output state vector to perform better in the recognition of frame I(t).

Experiments performed by Prager & al. (1986) tend to show that the Boltzmann machine indeed used the carried over information in cases which were ambiguous if the context was not given.

A weakness of this method is that it considers only left context information.

7.3 Realization of Connectionist models in Hardware

The full potential of connectionist models will be available only when experiments will be performed on massively parallel hardware that simulates or directly implements neural networks models. The simulations performed for this thesis were executed on a Von Neumann traditional computer. They needed a lot of CPU time. Since connectionist models are still being developed and experiments with varying models are being conducted, it would be more profitable for the present time to use massively parallel computers flexible enough to program the individual nodes. For this purpose, it would be much more efficient to use **SIMD** (Single Instruction Multiple Data) machines such as Hillis' Connection Machine (Hillis, 1985). Indeed, they use smaller processing units than MIMD (Multiple Instruction Multiple Data) computers, each with less memory but allowing for much more massive parallelism (65,536 = 64K units in the Connection Machine). As such, connectionist models are ideally suited to massive parallelism. No special algorithm is needed to segment the problem, since the connectionist models were initially designed as parallel models.

In addition to the advantage of being easily transportable to parallel computers, connectionist models also offer the possibility of readily carrying out **wafer-scale integration** because of their inherent fault tolerance (see Rumelhart & McClelland, 1986). A few malfunctioning units in a wafer would simply be ignored and this would not significantly alter the performance of the neural net. In fact this might allow for much cheaper VLSI production, since it is much easier to build a million-transistor system with 95% working elements, than to build a perfect system. Yield in production decreases exponentially with area of the circuitry, since it is a question of conjunction of probabilities :

probability that circuit of size 2A is faultless
= (probability that circuit of size A is faultless)^2

8. Conclusion

In this thesis, the development, theory and applications of connectionist models were studied. In particular, the Boltzmann machine algorithm and the error back propagation algorithm were examined in detail and were implemented in an application to automatic speech recognition. Connectionist models were shown to present a potentially very useful solution to problems in automatic perception.

The Boltzmann machine model and the error back propagation algorithm were used to perform speaker normalization and classify sonorant portions of continuous speech according to place of articulation : back position, central position and front position.

An important conclusion of this thesis is that experimental results, especially concerning the error rate, have been shown to be dependent on how spectral lines are coded. The best results were obtained using coarse and multi-level coding, as well as relative frequency and amplitude in a non-linear frequency scale derived from knowledge about ear sensitivity. The results obtained with connectionist models were significantly better than the ones obtained with Hidden Markov Models (alone), using similar data and preprocessed with the same technique of spectral lines extraction.

Some important issues in Automatic Speech Recognition were considered. In particular, the thesis stressed the difficulty of the ASR problem for multiple speakers and connected speech, because of the sequential nature of speech, the lack of acoustic-phonetic invariance, and the strong influence of the acoustic, phonetic, lexical and semantic contexts.

Although the models used in the experiments of this thesis assumed a static representation of speech, some methods to include the time dimension into the connectionist models were also examined. Knowledge of speech perception permits us to predict that the error rate obtained in our experiments would be significantly reduced if the sequential nature of speech and the influence of context were taken into account in the connectionist models. Considering the 0% error rate obtained on the training set, we can also suppose that better recognition rates would be obtained if the training set was significantly larger.

References

Ackley, D.H., Hinton, G.E., & Sejnowski, T.J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science, 9,* pp. 147-169.

Amari, S., and Takeuchi, A., (1978). Mathematical theory on formation of category detecting nerve cells. *Biological Cybernetics, 29,* pp.127-136.

Anderson, J.A., (1983). Cognitive and psychological computation with neural models. *IEEE Transactions on Systems, Man, and Cybernetics, vol. 13, no. 5,* pp.799-815.

Arbib, M.A., (1964). *Brains, Machines and Mathematics.* McGraw-Hill.

Barry, R.D., (1985). An associative hierarchical Self-Organizing System. *IEEE Transactions on Systems, Man, and Cybernetics, vol. 15, no. 4,* pp.570-579.

Barto, A.G., (1985). Learning by statistical cooperation of self-interested neuron-like computing elements. *Human Neurobiology, 4;* pp.229-256.

Barto, A.G., and Anandan, P., (1985). Pattern-recognizing stochastic learning automata. *IEEE Transactions on Systems, Man, and Cybernetics, vol. 15, no. 3,* pp.360-375.

Barto, A.G., Sutton, R.S., and Anderson, C.W., (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics, vol. 13, no. 5,* p.834.

Barto, A.G., Sutton, R.S., and Brouwer, P.S., (1981). Associative search network : a reinforcement learning associative memory. *Biological Cybernetics, 40,* pp.201-211.

Baum, L.E., (1972). An inequality and associated maximization technique in the statistical estimation for probabilistic functions of Markov processes. *Inequalities, vol. 3,* pp.1-8.

Beroule, D., (1985). *Un modele de memoire adaptive, dynamique et associative pour le traitement automatique de la parole.* PhD thesis, Paris-Sud University.

Bourlard, H., and Wellekens, C.J., (1987). Multilayer perceptrons and automatic speech recognition. *Proceedings of IEEE International Conference on Neural Networks,* San Diego, California.

Bridle, J.S., and Moore, R.K., (1984). Boltzmann machines for speech pattern processing. *Proceedings of the Institute for Acoustics Automn Meeting,* November 1984.

Changeux, J-P., (1985). *Neuronal Man, the Biology of Mind.* Pantheon Books, NY.

De Mori, R., Laface, P., & Mong, Y. (1985). Parallel algorithms for syllable recognition in continuous speech. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 7,* pp.56-69.

De Mori, R., & Palakal, M. (1985). On the use of taxonomy of time-frequency morphologies for automatic speech recognition. *Proceedings of the International Joint Conference on Artificial Intelligence,* Los Angeles, CA, pp.877-879.

Fahlman, S.E. (1979). *NETL, a system for representing and using real-world knowledge.* MIT Press.

Fahlman, S.E., Hinton, G.E., & Sejnowski, T.J. (1983). Massively parallel architectures for AI : NETL, Thistle, and Boltzmann machines. *Proceedings of the National Conference on Artificial Intelligence AAAI-83.*

Fahlman, S.E. and Hinton, G.E., (1987). Connectionist architectures for artificial intelligence. *Computer (USA), vol.20, no.1,* pp.101-108.

Feldman, J.A., (1985). Connections : massive parallelism in natural and artificial intelligence. *Byte,* April 1985, pp.277- 284.

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

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics, 20,* pp.121-136.

Geman, S., (1981). Notes on a self-organizing machine. In G.E. Hinton & J.A. Anderson (Eds.), *Parallel Models of Associative Memory,* Erlbaum, Hillsdale, NJ. pp.237-263.

Geman S., and Geman, D., (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 6, no. 6,* pp.721-741.

Grant, P.M., & Sage, J.P., (1986). A comparison of neural networks and matched filter processing for detecting lines in images. *AIP Conference Proceedings 151, Neural Networks for Computing,* Snowbird Utah.

Hebb, D.O. (1949). *The organization of Behavior.* New York : Wiley.

Hillis, W.D., (1985). *The Connection Machine.* MIT Press, Cambridge.

Hinton, G.E., (1985). Learning in parallel networks. *Byte,* April 1985, pp. 265-273.

Hinton, G.E., (1986). Learning distributed representations of concepts. *Proceedings of the Eighth Annual Conference of the Cognitive Science Society,* Amherst MA.

Hinton, G.E., & Sejnowski, T.J., (1986). *Learning and Relearning in Boltzmann Machines.* Parallel Distributed Processing : Explorations in the Microstructure of Cognition. Vol 1 : Foundations. (pp.282-317) Cambridge, MA : MIT Press.

Hinton, G.E., Sejnowski, T.J., & Ackley, D.H. (1984). *Boltzmann machines: Constraint satisfaction networks that learn* (Tech. Rep. No. CMU-CS-84-119). Pittsburgh, PA : Carnegie-Mellon University, Department of Computer Science.

Hogg, T., and Huberman, B.A., (1985). Parallel computing structures capable of flexible associations and recognition of fuzzy inputs. *Journal of Statistical Physics, 41,* pp.115-123.

Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences,* USA, 79, pp. 2554-2558.

Hopfield, J.J., and Tank, D.W., (1985). Neural computation of decisions in optimization problems. *Biological Cybernetics, 52,* pp.141-152.

Huberman, B.A., and Hogg, T., (1984). Adaptation and self-repair in parallel computing structures. *Physical Review Letters, vol. 52, no. 12,* pp.1048-1051.

Jackson, L.B., (1986). *Digital Filters and Signal Processing.* Kluwer Academic Publishers, MA.

Kienker, P.K., Sejnowski, T.J., Hinton, G.E., Schumacher, L.E., (1985). *Separating figure from ground with a parallel network.* Unpublished.

Kirkpatrick, S., Gelatt, C.D.Jr., & Vecchi, M.P. (1983). Optimization by simulated annealing. *Science, 220,* pp.671-680.

Klopf, A.F., (1982). *The Hedonistic Neuron, a theory of memory, learning, and intelligence.* Hemisphere Publishing.

Kohonen, T., (1977). *Associative memory : a system theoretical approach.* New York : Springer.

Kohonen, T., (1982). Clustering, taxonomy, and topological maps of patterns. *Proceedings of the Sixth International Conference on Pattern Recognition , Silver Spring, MD.* (pp.114-125).

Kohonen, T., (1984). *Self-organization and associative memory.* Berlin : Springer-Verlag.

Kuffler, S.W., Nicholls, J.G., and Martin, A.R., (1984). *From Neuron to Brain. a Cellular Approach to the Function of the Nervous System.* Sinauer Publishers, MA.

Levinson, S.E., Rabiner, L.R., and Sondhi, M.M., (1983). An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition. *The Bell System Technical Journal, Vol. 62, No. 4* (April 1983), pp.1035-1074.

Lippmann, R.P., (1987). An introduction to computing with neural nets. *IEEE ASSP Magazine,* April 1987, pp.5-22.

Lipschutz, S., (1974). *Theory and Problems of Probability.* McGraw-Hill.

Loeb, E.P., and Lyon, R.F., (1987). Experiments in isolated Digit Recognition with a Cochlear Model. *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing,* 1987., pp.27.3.1-27.3.4.

McCulloch,W.S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics, 5,* pp.115-133.

Merlo, E., De Mori, R., Palakal, M., & Mercier, G. (1986). A continuous parameter and frequency domain based Markov model. *International Conference on Acoustics, Speech and Signal Processing,* pp. 1597-1600.

Minsky, M. & Papert, S. (1969). *Perceptrons.* Cambridge, MA : MIT Press.

Moller, A.R., (1983). *Auditory Physiology.* Academic Press, NY.

Naccache N.J., & Shinghal, R. (1984). SPTA : A proposed algorithm for thinning binary patterns. *IEEE Transactions on Systems, Man and Cybernetics, 14 (3),* pp. 409-419.

Neumann von, J., (1958). *The computer and the brain.* Yale University Press, New Haven.

Pisoni, B., and Luce, P.A., (1986). Speech perception : research, theory, and the principal issues. *Pattern Recognition by Humans and Machines, vol. 1 : Speech Perception.* Academic Press, Florida. pp.1-50.

Pollack, J., and Waltz, D.L., (1986). Interpretation of natural language. *Byte,* February 1986, pp. 189-199.

Pucknell, D.A., and Eshraghian, K., (1985). *Basic VLSI Design Principles and Applications.* Prentice-Hall of Australia.

Prager, R.W., Harrison, T.D., & Fallside, F. (1986). Boltzmann machines for speech recognition. *Computer Speech and Language, 1,* pp. 3-27.

Rosenblatt, F. (1962). *Principles of neurodynamics.* New York : Spartan.

Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986). Learning internal representation by error propagation. *Parallel Distributed Processing : Exploration in the Microstructure of Cognition. Vol. 1 : Foundations.* (pp. 318-362) Cambridge, MA : MIT Press.

Rumelhart, D.E., McClelland, J.L. & the PDP Research Group, (1986). *Parallel Distributed Processing : Exploration in the Microstructure of Cognition.* Cambridge, MA: MIT Press.

Rumelhart, D.E., and Zipser, D., (1985). Feature discovery by competitive learning. *Cognitive Science, 9,* pp.75-112.

Sawusch, J.R., (1986). *Auditory and phonetic coding of speech. Pattern Recognition by Humans and Machines, vol. 1 : Speech Perception.* Academic Press, Florida. pp.51-88.

Sejnowski, T.J., Kienker, P.K., and Hinton, G.E., (1986). Learning symmetry groups with hidden units : beyond the perceptron. *Physica 22D,* pp.260-275.

Sejnowski, T.J., and Rosenberg, C.R., (1986). *NETtalk : a Parallel Network that Learns to Read Aloud.* Technical Report JHU/EEC-86/01, Johns Hopkins University.

Seneff, S., (1986). A computational model for the peripheral auditory system : application to speech recognition research. *International Conference on Speech and Signal Processing,* 1986, Tokyo, pp.1983-1986.

Shastri, L., and Feldman, J.A., (1985). Evidential reasoning in semantic networks : a formal theory. *Proceedings of the International Joint Conference on Artificial Intelligence,* pp.465-474.

Ullman, J.D., (1984). *Computational Aspects of VLSI.* Computer Science Press.

Viterbi, A.J., (1967). Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory, 13* (April 1967), pp.260-269.

Volkmar, F.R. & Greenough W.T. , (1972). Rearing complexity affects branching of dendrites in the visual cortex of the rat. *Science, 176,*pp.1445-1447.

Waltz, D.L., (1987). Applications of the Connection Machine. *IEEE Computer,* January 1987, pp.85-97.

Watrous, R.L., and Shastri, L., (1986). *Learning phonetic features using connectionist networks : an experiment in speech recognition.* Technical Report MS-CIS-86-78, University of Pennsylvania, October 1986.

Watson, W.E., (1976). *Cell Biology of Brain.* John Wiley & Sons, NY.

Weisel, T.N., & Hubel D.H., (1963). Effects of visual deprivation on morphology and physiology of cells in the cat's lateral geniculate body. *Journal of Neurophysiology, 26,*pp.978- 993.

Widrow, G., & Hoff, M.E. (1960). Adaptive switching circuits. *Institute of Radio Engineers, Western Electronic Show and Convention, Convention Record, Part 4,* pp.96-104.