Short title:

Basic Concepts of Fuzzy Graphs.

by:

Douglas R. Skuce.

Author: Douglas R. Skuce

Title: Basic Concepts of Fuzzy Graphs, with an Application to Waveform Recognition.

Dept.: Electrical Engineering.

Degree: M. Eng.

ABSTRACT

This study is an empirical investigation concerning automatic pattern recognition of "fuzzy" (not precisely quantifiable) data, in particular, waveforms such as the electroencephalogram. Emphasis is placed on determining a method for representing the data structure, both conceptually and in a computer memory, which interfaces easily between the raw feature space and any subsequent description.

A graph-structured approach is presented which is intended to apply both to functions of time or fixed images. Low complexity and low precision computations are involved, which admit the possibility of large-scale hardware implementation, while alternatively the graph structure can interface easily to heuristics of the variety found in the artificial intelligence literature.

Examples of the performance of the technique are given using actual electroencephalic data. While no theory is available as yet, the results are sufficiently promising to warrant more formal investigation, and recommendations are made in this regard.

BASIC CONCEPTS OF FUZZY GRAPHS, WITH AN APPLICATION TO WAVEFORM RECOGNITION

by

Douglas R. Skuce, B. Eng.

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of
Master of Engineering.

Department of Electrical Engineering,
McGill University,
Montreal, Quebec,
July, 1971.

ABSTRACT

This study is an empirical investigation concerning automatic pattern recognition of "fuzzy" (not precisely quantifiable) data, in particular, waveforms such as the electroencephalogram. Emphasis is placed on determining a method for representing the data structure, both conceptually and in a computer memory, which interfaces easily between the raw feature space and any subsequent description.

A graph-structured approach is presented which is intended to apply both to functions of time or fixed images. Low complexity and low precision computations are involved, which admit the possibility of large-scale hardware implementation, while alternatively the graph structure can interface easily to heuristics of the variety found in the artificial intelligence literature.

Examples of the performance of the technique are given using actual electroencephalic data. While no theory is available as yet, the results are sufficiently promising to warrant more formal investigation, and recommendations are made in this regard.

ACKNOWLEDGEMENTS

I wish to thank my thesis adviser, Dr. Martin D. Levine, for many helpful suggestions concerning the writing of this thesis.

This project was made possible by the Killam Research Scholarship granted me by the Montreal Neurological Institute, where I have had the unhindered pleasure to carry out this and other studies. In particular, I wish to thank Drs. Peter Gloor and Theodore Rasmussen for their continued encouragement and support.

My thanks are due as well to my friends Gio Coray, Stanley Selkow and Steve Nutik for many hours of discussion on various related topics.

Many hours were spent by Wendy Mackenzie battling the idiosyncrasies of the McGill Administrative Terminal System while typing the thesis. I am extremely grateful for her efforts beyond the call of duty.

Thanks are also owed to Judy Little, Cy Hatter and Neil Robbins for their fine work on the illustrations.

Finally, I am indebted to Edyie Zorychta and Chris Thompson for their careful proofreading.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Chapter 1. Introduction

1.1 Considerations Which Have Motivated This Study

The work presented here is primarily an attempt to synthesize a variety of concepts and techniques into a general approach to the problem of machine recognition and interpretation of large, complex and apparently "qualitative" classes of data. The term "fuzzy" has been used for such data. In particular, several points may be listed as the primary motivations behind this study:

1) The problem of automatic computer recognition of complex, extended, multichannel functions of time has received very little attention in the pattern recognition literature.

2) The rapidly growing field of Artificial Intelligence (AI) has given insufficient attention to the problem of representing "fuzzy" events, hierarchical relations between them, and methods for accessing them. The introduction of graph techniques is probably the most promising approach.

3) The technology for the fabrication of very large arrays of special function modules by large-scale integration techniques is progressing rapidly. This poses the question of what should these modules be, and how ought they to be connected and used.

It is felt that much simulation should be done now (via software) to explore the potential of such hardware approaches to certain pattern recognition problems, until more sophisticated theoretical methods are developed.

These very brief objectives will be elaborated upon in Chapter 3.

1.2 Choice of Data

For the reasons just described it was decided to select as an example of a fuzzy, multichannel time series a simple two-channel record of human EEG deliberately filled with "artefacts", or clinically unwanted signals. This data is of interest for at least two distinct reasons. Obviously one would like an efficient automated method for routine clinical evaluation (as is presently becoming possible for the electrocardiogram), and for research in neurophysiology. However it is felt that it is far too early to undertake the development of a program whose stated goal is to replace the human interpreter. It will be at least ten years before such a tool can be perfected, due to the difficulty of the task. The more appropriate point of view at the present is to consider the general class of problems of which the EEG is only an example, in which it is desired to relate certain events occuring in a large number of loosely defined, coarsely measurable

functions of time to situations outside this data, for example, spike-wave patterns in an EEG of a patient suffering from epilepsy. (Some other examples from this class would include geophysical and socio-economic data). Hence rather than becoming exclusively involved with the individual peculiarities of the EEG per se, one should let it be used instead as an example of data from this class, so that hopefully the techniques developed would be equally applicable throughout the class. The EEG was chosen because of the author's desire to learn more about real brains, which led him to this field, and hence his present association with the Montreal Neurological Institute.

Note that there are two other types of neurological signals which have been extensively studied by standard statistical means: the individual neuron potential or "spike" recorded from a microelectrode, and the "evoked potential" recordable from a volume of tissue. Both of these can be studied with the aid of special purpose averaging computers, although a general purpose computer is needed for involved analysis. These data, being much smaller and more precisely quantifiable, have received considerable attention from the available methods of engineering and statistics and do not belong to the class outlined above. Hence they will not be considered.

1.3 Some Important Concepts

Four concepts which will be constantly in use in what follows will be verbally defined now. More precise definitions of each will occur later as they are incorporated in a computer program.

The most fundamental notion is that of fuzziness, a term due to Zadeh (1965). A fuzzy set is one whose characteristic (membership) function is continuous onto the interval (0,1). Fuzziness is considered here to be a more appropriate representation of many real situations, when compared with binary-valued representations. Such situations have previously been handled either by statistical methods or by converting abruptly (say, by a threshold decision) to sharp (= not fuzzy) variables. This study attempts to overcome this problem. (We prefer the term "sharp" to the more popular but less descriptive "crisp").

A second important concept is that of learning. Since the program to be presented here is capable of (1) improving its performance as it is presented with more data; (2) discovering events and relations between them in the data which may not have been known to the programmer and (3) being taught names of these events in the same manner as one would teach a human student: it is felt that this program "learns". Considerable attention has been given in the artificial intelligence literature to devices

which "learn" by the administration of a reward or punishment following a correct or incorrect (respectively) response. The present program does not do this; rather it learns by either (1) clustering the data so that a maximal response is obtained from an input which in some sense is the "average of the cluster" and (2) associating certain events in the data with additional data of a conceptually different nature supplied through an additional channel.

The notion of association, introduced above, is a very fundamental one in the psychological and physiological literature. Two variables are associated to the degree that many examples of them have similar spatial-temporal relationships. Inputs to an organism, device, or program which are processed through common pathways and similar algorithms will be said to be of the same mode. (The term is borrowed from physiology, where we have sensory modes such as vision, hearing, etc.). Thus we may have inter-mode associations where for example a human student studying the EEG would learn that a certain waveform represents epileptic activity because a teacher points to it and tells him so.

The fourth essential concept is that of a hierarchy. Any process which transforms an input to an output in a series of stages, the processing being similar at each stage but in some sense more refined than the previous stage, may be termed hierarchical. It was pointed

out above that often the transition from fuzzy variables to sharp variables is made too abruptly. The solution offered here is to do this in stages, or hierarchically.

1.4 General Overview of the Thesis

In the following chapter, a brief review of various attempts at EEG analysis is given, followed by a more detailed account of the attitudes and approaches which are to be found in the "Artificial Intelligence" and "Pattern Recognition" literature.

Chapter 3 develops the notion of a "fuzzy graph", and attempts to show how it is related to the ideas reviewed in Chapter 2, and how it satisfies certain criteria which are given at the beginning of the chapter.

The details of a program written to gain experience with the fuzzy graph concept are presented in Chapter 4, and while many of the results are encouraging, they illustrate the need for a more formal approach.

A summary of the contributions of this work begins Chapter 5. The chapter concludes the thesis with a discussion of the general problem of "heuristic" versus formal methods, and suggests some avenues for more formal studies.

Chapter 2. A Review of Work Contributing to This Study

2.0 Plan of the Chapter

A review of this chapter will indicate the variety of existing work contributing to the present approach, some of it not previously considered related. This variety includes most of what has appeared under the general title "pattern recognition" (PR), as a sub-category of the more inclusive term "artificial intelligence" (AI), and certainly the various methods of EEG analysis currently in use. A more detailed exposition of our approach will be presented in the following chapter.

2.1 EEG Analysis (As It Is Reported In The EEG Journals)

This topic, mentioned last above since it is least relevant to what is to follow, will be discussed first, since this will not take long and will clear the way for the essential problems.

There have been very few serious attempts to analyse the EEG, and of these most have been the result of the interest shown by the various persons associated with MIT in the nineteen-fifties. This heritage is well documented in the books by Brazier (1961) and Rosenblith (1962). In this work, the EEG is rightfully considered

as a stochastic process and analysed as such by the methods developed in the engineering literature, that is, auto- and cross-correlation and the associated frequency domain functions. This work put on a more rigorous basis the even earlier frequency analysis attempts using various hardware bandpass filters. The more sophisticated procedures merely added additional evidence to the observation made years earlier by Grass and Gibbs (1947): "Much experimentation with the data finally leads to the conclusion that, although frequency analysis has advantages for revealing certain general features of the EEG, it is not satisfactory for clinical purposes. No index, either simple or complex, based on one or many spectra from a given case, can express the highly specific detailed information contained in the EEG".

Some early workers considered time domain and space domain analysis. Amplitude histograms were obtained by Lonsdale (1952), King (1951), and others; baseline crossing was analysed by Young (1954), Burch (1955), and Saltzberg et al. (1957). Two techniques affording visual appreciation of the spatial aspects of the EEG were those of Walter and Remond. Walter (1951) constructed a "Toposcope", an instrument consisting of an array of small oscilloscopes, one per electrode, arranged in a pattern that matched the electrode placements. The

beams were swept synchronously in circles or spirals and were brightened and/or radially deflected by the EEG signal. Motion pictures were taken (the activity being too fast to follow in real time) and eventually some aesthetically pleasing results were undoubtedly obtained. Remond (1967) produced contour maps of potential along a line of electrodes (this spatial dimension becoming the ordinate) versus time plotted horizontally. Fascinating hill and dale patterns were obtained which of course were left to visual interpretation. These attempts are all well summarized by Brazier (1961).

With the availability of computers in the late nineteen-fifties the problem of handling quantities of data of sufficient size and with adequate mathematical detail at last became tractable. The interest of Norbert Weiner at MIT spurred investigators there to apply the latest engineering developments in time series analysis. These early developments are well summarized in the book by Rosenblith (1962). The need for treating the EEG as a statistical time series is firmly established there, thus dating the non-mathematical approaches. Two techniques developed on the TX-0 computer were landmarks (Farley, (1961). The first was a pattern detecting scheme to detect alpha wave bursts by accumulating statistics on pre-assigned parameters (such as peak-to-peak amplitude and zero-crossing). Plotting burst activity as a function of the parameter showed considerably greater inter- than intra-

subject variability, opening the question of how to interpret this finding. The second technique was the computation of auto- and cross-correlation for several minutes of data, which demonstrated a high degree of variability and made visual interpretation difficult.

During the sixties, probably due to the presence of Mary Brazier, Ross Adey and D. O. Walter, The University of California at Los Angeles developed an extensive and comprehensive set of programs for the analysis of large amounts of EEG data. This work was certainly the most detailed and mathematically sophisticated to appear in the literature during this time. Representative papers include Walter (1963) and Walter and Brazier (1969). The approach is once again statistical, considering the EEG as a time series in the Weiner tradition, and relying heavily on the frequency domain. The output is nearly always a very complex looking plot (in many cases to the eye more complex than the raw data) and hopefully the mathematics have been such as to make obvious to visual inspection some aspect of the data, such as an enhanced degree of activity in some frequency band, or a coupling between two areas previously thought unrelated.

The essential point to be emphasized then is the following. All of this work, (and almost certainly the same situation exists in other areas like economics or physics), is directed solely at developing mathematical methods which will transform one complex set of curves

into another. The new set, it is hoped, will then "show" something to the eye (or even allow it to be expressed in the "cleaner" form as a number or set of numbers) which was not evident previously. Thus the problem of interpretation of the data, that is, of relating unusual events or situations to others within or external to the data, is left entirely to human judgement.

2.2 Contributions from Artificial Intelligence

We shall now turn to a completely different area, which has come to be generally known as "Artificial Intelligence" (AI). It is doubtful that there has as yet been any cross-fertilization between AI and EEG processing.

The history of AI, like other disciplines relying on computers, extends back only about fifteen years. During this time there have been a number of "vogues", and fortunately, a few well-established general principles. Among the vogues, one appears as germain to the present undertaking, namely, the much-discussed Perceptron. The essential lessons of this research will be reviewed briefly now.

2.2.1 Perceptrons

The Perceptron was originally introduced by Rosenblatt (1962) and his colleagues at Cornell in the

late nineteen-fifties. It was viewed as a class of brain models and was a direct descendent of the original formal brain model of McCulloch and Pitts (1943). It will be assumed that the reader has sufficient familiarity with the Perceptron concept to forego the lengthy discussion which would be required to introduce the subject. (A very readable summary is that of Daly et al, (1965)). It was hoped that the Perceptron would turn out to be a machine of great generality and power with which many of the problems of pattern recognition and brain modeling could be solved with one comprehensive theory. A considerable effort during the period 1958 to 1965 (including the construction of a number of hardware devices) demonstrated that in fact the Perceptron was not as general or as powerful as had been hoped. Merely increasing the size or complexity did not seem to give adequate return in performance. The methods of analysis were essentially statistical, since it was hoped that large collections of elements would, "on the average" perform arbitrarily well. The metrics of performance were those suggested by the traditional mathematics of linear spaces. Nothing which could be called "algebraic" or "topological" characterized the analysis. Practical (yet essential) estimates of the amount of memory required or the execution time were few. Hence, when it was discovered (mainly through computer simulation) that the performance left much to be desired, the underlying causes were not certain.

Toward the late nineteen-sixties, the number of Perceptron papers appearing each year began to decrease. Finally in 1969 Marvin Minsky and Seymour Papert published a book entitled: Perceptrons. An Introduction to Computational Geometry. In this book they take a fresh approach. No statistics, no multi-layer, cross-coupled machines are discussed. Instead, they take the most basic configuration of machine and examine the various algebraic and topological properties it must have. They introduce the notion of "order", which shows why earlier machines, lacking sufficient order, would be incapable of computing some of the simple predicates they were being forced to attempt. They are concerned with the problem of geometric complexity, rather than the behaviour of large networks of elements whose basic computational properties were uncertain.

The conclusion to be taken is simply this: elementary machines of the Perceptron variety do not possess enough power to perform practically useful pattern recognition, while at the same time pose considerable mathematical difficulties. (On page 102, Minsky and Papert admit that they "secretly advocate" syntactic scene analysis procedures). The main deficiences appear to be these. The Perceptron suffers excessively from the restrictions imposed by the use of Boolean variables (0 or 1) to transmit information. Since the output of each element gives no information about how near the input conditions are to

the threshold, but merely indicates on which side of the threshold the input lies, a tremendous loss occurs. This is perhaps the most succinct example of the general problem of excessively abrupt transition from fuzzy to sharp variables referred to in the introduction. A further problem, related to the problems of reliable computation considered by Winograd and Cowan (1963) for example, is that the ultimate report of the device must eventually be channeled onto a single output line, which carries too much responsibility. This is related as well to the point of view that a single decision is the appropriate response, i.e. that in the case of classifying alphabetic characters, it would be meaningless to decide a letter was say, 30% A, 45% B and 25% C. The Perceptron is not intended to handle this class of data. A third criticism, most often heard from the AI community, is that the Perceptron is not intelligent at all. By this they mean that it uses no heuristics to shorten its computational task, but simply computes all quantities whether or not a previous computation makes them irrelevant. If Perceptron popularity had stayed high, a marriage with heuristic search techniques would have undoubtedly taken place. Instead however, marriages are occurring between more powerful approaches.

2.2.2 Pattern Recognition

Of all the areas being reviewed, that most frequently referred to as "pattern recognition" (PR) has by far the greatest bulk of published literature. No comprehensive survey of this literature can be attempted here; the reviews of Nagy (1968), Levine (1969) and Nilsson (1970) are recommended. Instead, the general flavour will be indicated, and the connection between this work and the present one established.

2.2.2.1 "Classical" Pattern Recognition

What we shall term "classical" PR is well documented in the review by Nagy, and may be seen to bear the following earmarks. The problem is to design a machine such that, when an example of a certain class of data is presented to it, the machine replies with a single "name" or number, indicating to which class the example belongs. More precisely, given a set of disjoint classes, C(i), i=1,...,n, and an input vector X(k), the machine is to reply with some number i, j=1,...,n. Most often discussed is the case where the number of classes n is known in advance. Alternatively, the machine is to "discover" how many distinct classes there are, often termed "clustering". (The "classical" approach has also been termed the

"receptor/categorizer" model (RCM)), by Marill and Green, 1960).

The classification process is always broken into two stages which are quite distinct, often called "feature extraction" (the first stage) and then the "classification" stage per se. The feature extraction process usually is determined ad hoc, the bulk of attention being given to the classification. Levine (1969) has pointed out that in fact, feature extraction (the reduction of the raw data to a vector of considerably smaller dimension), is just as important as the classification algorithm to the overall performance. Some attempts (e.g., Watanabe, 1969) have been made to develop general automatic methods of feature extraction, but this process seems to require more intelligence than can be programmed, and hence much more attention has been given to the classification stage assuming that adequate features have been determined empirically.

The classification stage receives the feature vector (usually having a dimension d less than say, 20) which it is hoped adequately represents the input example. What it does with it depends on whether the machine has been "trained" yet or not. During training, the feature vectors serve to establish regions or boundaries in the d-dimensional space they inhabit. If the training is "supervised", an additional input for the class of the input vector is provided, and the coefficients defining

the regions are updated. Often the output classification (the machine's "opinion") is "fed back" so that errors may be corrected. (Most of the Perceptrons functioned according to this paradigm). In "unsupervised" learning, the machine must find the class boundaries for itself.

When the machine is considered sufficiently well trained, it is tested by inputting examples and recording the responses, right or wrong, in each case. Typically, a machine might be trained on several hundred examples of letters of the alphabet, tested on somewhat less, and with 10 classes (not all letters used, since a number are quite similar) achieve something like 95% correct response.

The mathematical flavour of the classification stage is almost exclusively that of linear spaces and statistics. The boundaries between regions are usually simple functionals (linear or simple polynomials) in the fixed space determined in advance by the features. The training consists in finding suitable values for the coefficients in these functionals. Recognition then consists in evaluating the functionals and deciding the discrete class by the sign of the result. If statistical procedures are invoked, the estimation of parameters in an assumed distribution (the so-called "parametric" approach), or the estimation of the distributions themselves ("non-parametric"), with the resulting complexity of arithmetic (matrix operations, iterative procedures) is the dominant preoccupation. Since usually the computational

load increases as the square of the dimension, large dimensionality is unfeasible.

We conclude then that these "classical" or RCM methods are applicable in the following situations only. The data must consist of a small number of sharp (in the sense defined in 1.3) disjoint classes. Any data example should be bounded in storage requirement, so as to be accomodated in core storage. The dimensionality of the feature space must be "reasonable", and probably the features will be chosen by the implementer intuitively, with some experimentation. The most likely situation is one where there can be a short training period followed by an indefinitely long operational period, in which the data is sure to retain the characteristics already learned. (Two good examples of this situation are probably the commercially available optical character readers, and the several electrocardiogram (EKG) analysis programs which are operational).

2.2.2.2 The Semantic Approach

Outside the more well-defined regions in which classical PR has had success, there lies a much larger and virtually unexplored terrain. One area is the so-called "linguistic", (or "articular" or "syntactic")

approach. This philosophy has been investigated by an increasing number of workers among whom Narasimhan (1962, 1964, 1966) is perhaps the best known. In his 1962 paper he makes the point:

> Categorization, clearly, is only one aspect of the recognition problem; not the whole of it by any means. It is our contention that the aim of any recognition procedure should not be merely to arrive at a 'Yes', 'No', 'Don't know' decision but to produce a structured description of the input picture. Perhaps a good part of this confusion about aims might have been avoided if, historically, the problem had been posed as not one of pattern <u>recognition</u> but of pattern <u>analysis</u> and <u>description</u>.

Shaw (1968) has emphasized the essential phrase in the above remark: "structured description of the input picture". This is to say that the emphasis ought to be placed on the <u>description</u> of the data in terms of simpler data types, rather than the simplest of all descriptions, an assignment to a single class. Hence Shaw proceeded to develop a phrase structure grammar (Chomsky, 1959) with graph theoretic understructure. An advantage of this degree of formalism is the theoretical results which may be called upon to assist in the development. In addition, the class of describable pictures is easily and precisely

defined. On the negative side, this degree of formalism requires strong assumptions about the nature of the data and the processor so as to insure (in Shaw's case) the recognition of sharp primitives for input to the parser. (Shaw's primitives are picture elements which allow 1 to 1 correspondence with graph edges. Thus it has a "natural" application to connected line drawings. It is difficult to see how one might select primitives in the case of more fuzzy data, such as aerial photographs or microscope images).

The work of Zahn (1970) is in a similar vein. The feature extraction must yield a set of points, which are then connected by lines according to certain rules to form a graph. This graph is then processed by mathematically justifiable algorithms to yield what hopefully is a simpler and more correct graph representing the data as clusters.

We now turn to an area of AI which at first glance does not seem very related to what we have been discussing. The writer feels however that not only is it related, but a more precise study of such relations is greatly needed. We refer to computer "comprehension" of natural language.

A sizable literature in computer processing of natural language has developed within the broad scope of AI. Earlier attempts at machine translation (between natural languages) using formal models as a common base

have given way to what appears to be a less ambitious goal, that of fact retrieval or question-answering systems. (Some recent surveys are Simmons, 1970, Pirotte, 1970, and Palme, 1970). Among these the most successful have been those which abandon mathematical formalism entirely, invoking syntactic rules as little as necessary. Rather the attention is given to the semantics of the text, where there is even less hope of a formal approach. Perhaps the best example of such a semantic network is the Teachable Language Comprehender (TLC) of Ross Quillian (1969).

TLC is a program capable of learning the meaning of simple English as it progressively experiences the text. The meaning of "meaning" is not defined in words or in mathematical formulae; rather it is a set of heuristics buried in the program for accepting each input word and making appropriate changes in the machine's memory to reflect the relation between the word and the rest of the text. The machine is under the constant supervision of a human teacher, so that both factual assertions and the capability for correctly relating the input to the existing contents of memory may be progressively taught.

This progress results in the evolution of a large and complex network of interrelated words and concepts. It is this network which is termed the "semantic memory". The details of how TLC constructs and modifies its semantic memory are once again, not important here, since we are not about to deal with English text. The salient points

for our purposes are the following: TLC uses a large and ever-growing network, arranged so that related things are connected by short paths, to represent its experience. Its knowledge is distributed fairly evenly throughout the network, though there is a heirarchical relationship between the more general and the more particular concepts. The input to the network may be thought of as its "surface", with more general concepts lying deeper. The system relies heavily on human assistance at first, though as it accumulates experience, the ability to make guesses and to generalize makes it less dependent.

TLC cannot make use of the more sophisticated methods of heuristic search due to the lack of a feasible evaluation function (meaning is hard to quantify): hence it is limited to a rather small vocabulary and slow breadth-first search. The loss of reasoning procedures such as those available in a formal underpinning (usually first-order predicate calculus) are also a price paid for the semantic power. However these criticisms will be ignored since in what is to follow, we shall not be prevented from searching, and deductive reasoning is not a very likely requirement in fuzzy data descriptions.

2.2.2.3 Graph Searching and Sequential Methods

If there is one solidly established principle which has emerged from the body of research which might

be termed "hard core" Artificial Intelligence (as opposed to the looser sense of AI in which the heading for Section 2.2 is meant) it is certainly the technique of heuristic search (often called "tree" or "graph" searching). A good overview and bibliography is that by Slagle in the book edited by Banerji and Mesarovic (1970) or the recent books by Slagle (1971), and Nilsson (1971). The essence of these methods is the reduction to a manageable size of an astronomically large number of possible combinations of situations which must be evaluated to make a useful decision in a complex environment. For example, in a game like checkers it is totally out of the question to compute all possible situations to find the best one before making a choice for a move. Hence, if all cannot be evaluated, then which?

We will not become involved in a discussion of the details of these various algorithms, since no actual graph searching will take place in what follows. This is because (1) time did not permit and (2) the data used were sufficiently simple that very small graphs were involved, making exhaustive search perfectly feasible. The point to emphasize though, is that if larger data sets are to be feasible, then any method involving computation performed at the nodes of a graph must allow an efficient application of the established search methods. That is to say, the computation must be simple, it must depend only on a concise and easily accessible number of previous

computations, and the result should be obtainable (in any particular instance) after searching only a small portion of the total graph.

A technique very closely related to the methods found in the AI literature is dynamic programming, though for some reason very few papers on graph searching methods from the AI community acknowledge this. There has recently been an increase in interest in such sequential pattern recognition methods, originating in the methods of Wald (1947), and currently studied by Fu (1967, 1968, 1970). The rationale in this approach is to reduce the expected cost of observations (feature measurements) by suitably planning the order in which they are made. The only paper from the AI community along these lines known to the writer is Slagle and Lee's, (1971). Zahn (1970) uses graph-theoretical methods (the Minimal Spanning Tree) to achieve Gestalt-like clustering, thus allowing searching as well as a graph description of the data structure. It is the opinion of this writer that such graph-based methods as these will prove to be very rewarding.

2.3 Fuzzy Sets and Concepts

It was pointed out in the introduction and again in this chapter that the notion of "fuzziness" is an essential property of many types of data, and that the abrupt transition from fuzzy data to sharp variables,

structures or concepts (without carrying along the fuzziness) is a probable source of the poor performance of methods which do this.

Zadeh has introduced the concept of a "fuzzy set" in several papers (1965, 1965a, 1966, 1968), in which he gave a number of basic fuzzy equivalents of conventional set-theoretic concepts. The central idea is that of the "membership" function, which allows an element of a set to have a continuously graded membership in the set, rather than an all-or-none (binary) membership. He points out the need of the introduction of fuzzy concepts in the areas of pattern recognition and heuristic programming ("It would be an advance of vast importance when we learn how to design machines that can understand fuzzy concepts in much the same way as human beings are capable of doing"). (1968).

Though there have been a number of papers (Goguen, 1967, C. L. Chang, 1968, Marinos, 1969, Lee, 1971) on various aspects of fuzzy set theory, to the best of the writer's knowledge there have been no attempts to directly incorporate fuzziness in a machine representation scheme, with the possible exception of the thesis of W. G. Wee (1967).

Since a proper introduction to the Zadeh fuzzy theory is best had by reading his original papers, we shall assume familiarity with them. Discussion and criticism of various aspects of the theory will be found in subsequent

chapters where certain aspects of the present work suggest alternative points of view.

Chapter 3. Fuzzy Graphs

3.0 Introduction

This chapter will introduce and discuss at length the central theme of this thesis: the notion of a "fuzzy graph". The term "fuzzy graph" has been chosen since the notions of "fuzzy" and "graph" are well known in the literature and suggest the essence of the idea. We shall attempt to present the fuzzy graph concept as a synthesis of many of the ideas reviewed in the last chapter.

3.1 A More Detailed Statement of the Objectives of This Study

In Chapter 1 a number of criteria were given briefly to indicate the various objectives of this study. These points will now be reiterated in more detail, so that the development of this chapter may be better appreciated. The order of their appearance only partially reflects the significance of the criteria.

3.1.1. The Central Problem

This thesis is directed first of all at what may be termed the "Central Problem" of many areas of pattern recognition and artificial intelligence, which we shall express thus: "to design a scheme which appropriately and efficiently represents, both conceptually and in some physical device, a large collection of fuzzy objects, and fuzzy relations between them". This statement emphasizes that fuzziness must be accomodated in some appropriate or natural way; that the representation should be amenable both to some theoretical formulation and to hardware implementation; and that some form of abstraction of the relations involving these fuzzy objects should be provided for. The function of this representation scheme should be to provide an "interface" or intermediate structural description between the raw data and some final or output description, which may range in complexity from a simple binary dichotomy to some type of "language".

3.1.2 Learning and the Problem of "Meaning"

Since we shall be concerned with the design of a machine whose function is to learn to identify certain types of patterns presented to it, the attitude

toward the machine as an "apprentice" is to be encouraged. Thus it is desirable to build into the machine as little as possible a priori information about the data it is to encounter. That is, attention should be focused on how the machine is to discover for itself the nature of the data, and how it is to ascribe some "meaning" to this data. Let us then agree to regard the machine as a mechanical student of EEG interpretation. Thus we shall call the EEG data the "visual" data, since a human student would study an EEG by eye, and any additional information about the EEG will be termed "verbal" or "auditory", since the human student would receive this information verbally.

The notion of "meaning" of the visual data then will be considered to be the following. The data, consisting of a number of fuzzy objects with fuzzy relations between them, will be input via some "visual" channel. The machine will be considered to have learned the meaning of some item of this data when it can associate a sharp name, input via some "verbal" channel, with this particular item. The name will have been learned by repeated training examples considered by the teacher to be similar, each training example being accompanied by the same name. The machine is thus able to attach names to what may be called "events" or "situations", consisting of a number of sub-events, which may in turn be named. The eventual

goal is to have the machine report not only the event and its components, but the structural relations between them. To do this it must have some representation of its experience in its memory, and this is of course the Central Problem.

The machine should possess both the ability to learn without supervision and, when supplied with verbal input, to use the verbal information to assist the learning process. In classical pattern recognition, the learning usually consists in adjusting a number of prescribed coefficients in some functions of the features. This approach allows no interplay between any "intelligence" and the visual classifier. If the machine is to be endowed with the kind of artificial intelligence found in game-playing and question-answering systems, then the visual classifying mechanisms should be of a fundamentally different nature. Hence the method of representing the visual experience and of structuring it, and of interacting with it should be posed more as a general AI problem than as a pattern classification problem.

3.1.3 Temporal Versus Non-temporal Data

For the reasons stated in Section 1.2 we would prefer to concern ourselves with "on-going" functions of time (i. e., having no previously known

segmentation such as speech has words). We shall allow a partial segmentation (into "words") but shall specify no prior structure ("phonemes") within the words, which we shall take to be non-overlapping EEG artefact signals. Since most pattern recognition work has been concerned with functions of two spatial coordinates only, and those dealing with functions of time have usually had known structures (e. g. speech or electrocardiograms), we are concerned here with a somewhat different problem. We shall attempt however to develop a method which will be applicable to other data types as well (e.g., "retina" problems).

The type of data considered here adds another restriction not found in many other more popular types. Since it is so extensive in time, any form of random access is probably uneconomical. (A full clinical EEG of 1/2 hour duration and 16 channels contains about 10^9 bits). Hence we should look for method which will allow a single scan of the data. This will have the additional advantage of allowing real-time operation of the machine, such as in a robot.

3.1.4 Implementation

Clearly an appropriate way to consider the problem from the point of view of implementation is

as a program for a general purpose digital computer. The writer is of the opinion however (cf., Aleksander, 1971) that the von Neumann machine organization is highly unsuited to this type of problem. Thus the conventional computer may be considered an experimenting ground in which one may discover the principles by which a radically different type of machine organization could be defined. This is what was alluded to in Section 3.1.1 when it was included in the "Central Problem" that a physical device was to be considered as well as conceptual devices. Too many conceptual devices are either physically unrealistic, or are cast in a form determined by some well-known physical device, such as the von Neumann machine.

Hence the following point of view on implementation will be adopted. Since there exist a large number of von Neumann machines and software for making them relatively easy to use, an approach should be taken which attempts to function usefully in this environment. In addition, since there exists a considerable development of software artificial intelligence techniques which may be employed, any advantage of them may only be had in this environment. However, it should be kept in mind that the developments in pure fabrication technology will probably continue to outpace developments in

> theoretical areas. It may be safely assumed that by the end of the decade (if not already) the capability to construct machines with billions of active elements will have advanced far beyond the theoretician's ability to take full advantage of the hardware. Thus we should not feel that the von Neumann machine is to forever be the only machine which will perform intelligent tasks, but rather suspect, as Aleksander does, that it is a poor candidate for this role. The important question then becomes: what are the alternatives? This study is concerned as well with this question.

Two additional questions (for hardware particularly) are the degree of precision required, and the reliability. The demonstration of acceptable performance using very low precision, and a method which distributes responsibility for recognition, so that errors in individual elements may be tolerated, will therefore be additional design objectives.

3.2 A Prognosis

Having reached this point, the knowledgable reader could not be blamed for feeling somewhat skeptical that all the various work mentioned and the various criteria listed are about to be combined in one successful package. He should be assured right now that this is not about to

happen. Why then has such a variety of background and objectives been mentioned? Simply because they are in fact what the writer has in mind as long-term goals, realizing that they will require much more work than what has already been accomplished, but hoping that the ideas and their feasibility could be adequately demonstrated here.

Now we will state briefly what does lie ahead, and what does not, to prevent any undue upset on the part of the reader. The program presented may be thought of as a form of clustering algorithm (for a review of clustering, see Ball, 1970), and no interaction between the visual and verbal data of the AI variety will be found. The data structure is nevertheless intended to allow such interaction, though consideration of this problem is beyond the scope of this work. Since the size of the graphs is small, no searching algorithms have been used, but the nodes compute a natural evaluation function, and searching could be easily introduced. The data is extremely simple, consisting of only four categories to be distinguished. Though no semantic reporting of the data structure is attempted, printout of the actual graph structure demonstrates that the data is accurately and efficiently represented.

The program attains a recognition rate above 90% with about 20 nodes, making about 5 nodes per category, each node requiring about 90 bits of storage. The examples on which it makes mistakes are usually poor enough in

quality that the errors may be blamed on the features, which are not intended to handle distorted or small examples.

3.3 Representing Fuzziness

In Sections 1.3 and 2.3 we have reviewed briefly the concept of fuzziness, introduced originally by Zadeh. We assume familiarity with his definitions, and have not repeated them since we can offer no theoretical extension of them. To the best of the writer's knowledge, the only significant extension of Zadeh's original papers is that of Goguen (1967), in which he generalized the unit interval J to a complete multiplicative lattice L with certain restrictions on distributivity and the group operation. He makes the important observation that J is not suitable in many instances (i.e., two fuzzy situations may not be comparable in any sense). He introduced as well hierarchies of fuzziness, so that two fuzzy sets may be compared for "degree of fuzziness", for example. Perhaps his most important addition is the use of category-theoretic language as a more general and flexible formalism for fuzziness, along with his "Principle of Fuzzification", a general rule for converting sharp mathematical situations into corresponding fuzzy ones. He makes no attempt to consider topological questions in this paper, promising them later.

The following attitudes toward theories of representing fuzziness are suggested by the works of Zadeh and Goguen. What is desired is a definition of mathematical relations between some input or feature space X, an internal (to the machine) representation scheme R, and an output space or language Y. In general, the space X is probably adequately structured as a vector space. The output space Y may range over a wide number of possibilities, from a single binary decision to the English language, or some subset thereof which is mathematically tractable, say in the manner of Chomsky (1965). Given X and Y, the problem of course is to determine a suitable R, and the mappings between them. This is where the language of category theory and topology should be most useful. Two levels of complexity may be distinguished in the possible structures for R: those R's which have a fixed vocabulary and which learn the data structure in terms of adjusting values which define functions in this vocabulary; and those which are in some sense "extensible", in that a working vocabulary is automatically developed from some simpler set of primitives and production rules. Examples of the former include simple learning of coefficients in classical pattern recognizers, and scene analysis methods which express the data structure in a sharp language such as predicate calculus (the predicates being given by the designer in advance), or in some "picture" language in the manner of Shaw or Narisimhan. TLC may be an

approximation to the latter case, though there the input is not a vector space, and the representation does not have a mathematical structure. A TLC-like program which looked at pictures as input and delivered an output in a Chomsky-like transformational grammar would be closer to the second situation.

What kinds of structures are suitable candidates for R? Remember that we wish to incorporate fuzziness, so that sharp structures per se are rejected. What about fuzzifying a given sharp mathematical structure (e.g. predicate calculus) using a Fuzzification Principle such as Goguen has proposed? The writer suspects that this is precisely what <u>not</u> to do, since as soon as sharpness is abandoned, most of the useful algebraic properties will be lost as well. It is doubtful that a fuzzy group would be of much use for example, since the structure of groups is a very sharp situation. Lee's fuzzy resolution is simply a "worst case" theory. Topological notions are more naturally fuzzified, as C. L. Chang has pointed out. It appears then that fuzzification applies naturally to some mathematical structures, and not to others, with set theory and basic point-set topology established as "good" candidates.

When this study was begun, the writer had no ideas on how to define a formal fuzzy mathematical structure adequate for R. It was decided therefore to employ the notion of a graph which has been widely applied in sharp

situations. The notion of a collection of nodes, each representing some situation and computing some function, together with relations (weighted edges), between the nodes is sufficiently general so as to be adaptable to many situations. Here we will develop a graph-like structure R which represents fuzzy clusters in a natural and hierarchical way, and which maps a feature (visual) vector space X of small dimension onto an output (verbal) space Y which is also a vector space of smaller dimension. No attention will be given to finding an optimum structure for X or Y, this question being data-dependent. Only the structure of R will be considered, since we are not concerned with the data (EEG) per se, but rather with the general problem of managing data of this type.

3.3.1 Clustering at Nodes

The clustering methods reviewed by Ball (1970) range from those which connect together every data point (e.g., Zahn's) to the more general techniques which partition the feature space into relatively large regions, such as discriminant, nearest-neighbour and mode-seeking methods. A trade-off between these two extremes would be to provide for a number of small regions where variations within the regions represent slight differences in quality, so that the regions may be considered "fuzzy sets". If we then associate each of these regions with a node in

a graph structure, we may add additional nodes to the graph whose function is to define relations between the original nodes. Thus the nodes associated directly with the fuzzy regions of feature space may be called level 1 nodes, and the function of such a node is to report, for a given input situation (i.e. a given feature vector) how close the vector is to its region, or the degree of membership in its fuzzy set.

Leaving for a moment the question of how to compute this closeness or membership function z, let us consider how we might connect such nodes in a graph. The usual method of connecting nodes which are intended to perform clustering is to connect with all-or-none (binary-valued) edges all nodes belonging to the same category. When the certainty of category is in doubt at a node, either through poor data or when the node quite properly belongs to more than one category, this method breaks down. Neither does it admit a hierarchical description of the data structure, where for example, one group of nodes represents one part of the data, another group of nodes another part, and these two groups are in turn connected to a node in a manner which represents the relation between the two parts. If we can define a means for connecting nodes on the basis of their structural relation instead of their category, we will open the possibility of a graph structure which is capable of a deeper description of the data structure than simply a binary assignment of

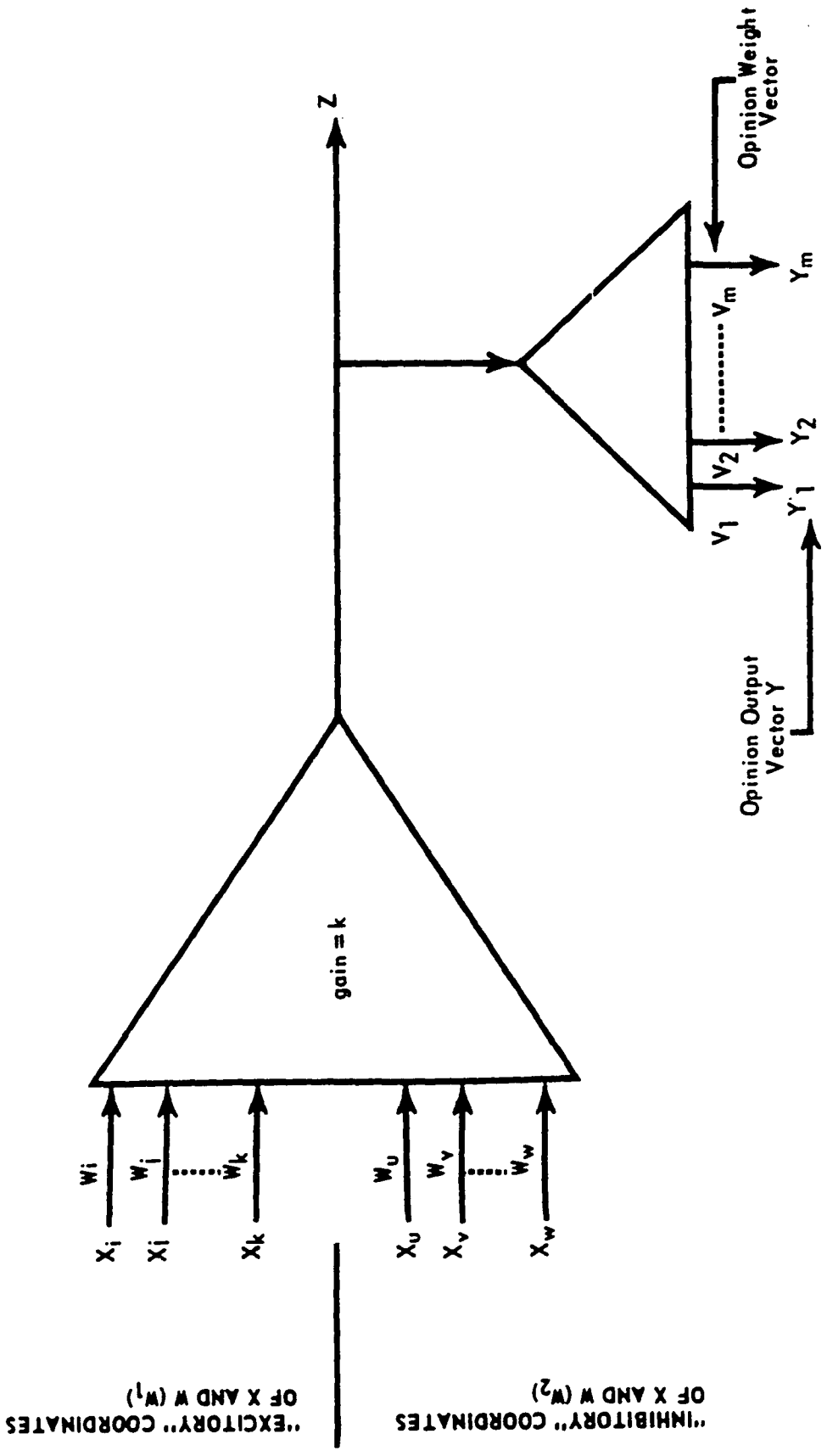


FIGURE 1. STRUCTURE OF A NODE

categories. The assignment of the category or description may then be effected by adding to the node "pointers" to the descriptions which apply to the region represented by the node. In the case of simple categories, which we consider here, the pointers may consist of a vector which we will call the "opinion" vector V, whose coordinates correspond to the categories. The node then defines a mapping between a fuzzy region of the feature space, X, and a direction in the category space Y given by the opinion vector V. By this mechanism very general transformations between situations in a feature space X and their descriptions in a space Y may be defined.

3.3.2 Closeness Function and the Node Structure

In this section we will develop the structure of a node (Figure 1), but first we must discuss membership functions for fuzzy sets. Closeness or membership functions for fuzzy sets are usually thought of as "hills", reminiscent of probability distributions, which are large over the region of interest and approach zero in all directions away from the region. Such functions tend to be very non-linear and require more computation than one would like if they are to be computed a large number of times. We may make a considerable simplification under the following conditions however. Suppose that we wish to select regions of a feature space as "interesting" or

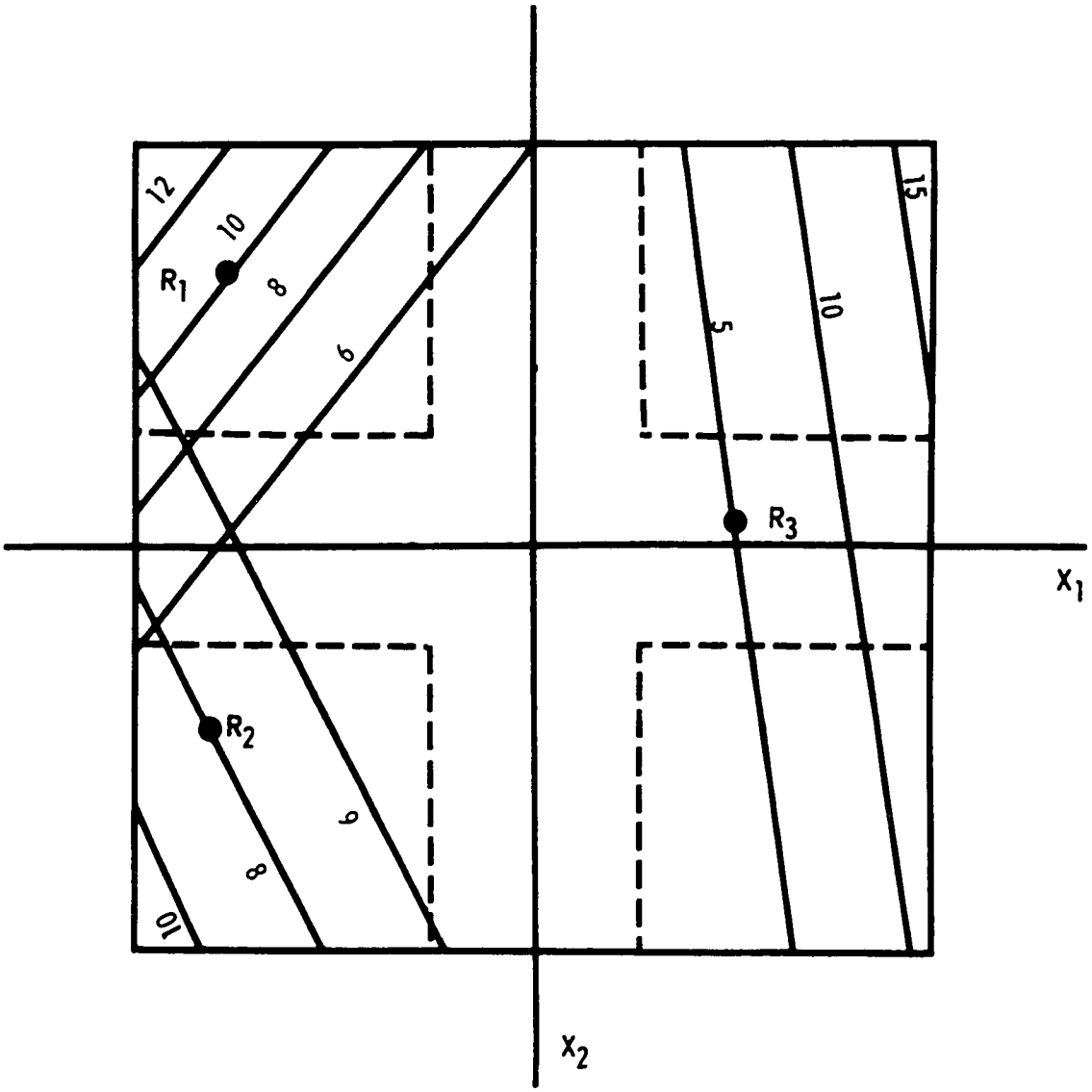


FIGURE 2. INNER PRODUCT REGIONS.

The sloping lines are loci of constant inner product, with a vector W located at the dot. It is best not to allow W inside the dotted area, such as in region R_3 which is larger than R_1 or R_2 as a result.

"important" where all the features (coordinates) have the property that their interest or importance is proportional to their magnitude. We shall call this the "magnitude-significance" property. Suppose in addition that most of the time, less than (say), half of the coordinates of an interesting region are large. Under these assumptions, we may choose a vector W somewhere in the middle of the region of interest. Then any given vector X will have a large inner product with W in the region, of magnitude proportional to X. The inner product z=X*W (lower case letters are scalars, upper case are vectors, and * indicates inner product) may then serve as a membership function for features with the magnitude-significance property, and for regions limited to areas away from the origin, bounded by the hyperplane z=0. Our initial closeness function is then:

$$z = k \max(X*W, 0) \qquad 1)$$

where W defines the region by being a "typical" vector near the middle of the region, and W should have, say, at least two coordinates greater than some threshold, and preferably not all coordinates large, since the effectiveness of the inner product in determining a region is reduced in this case. k is a normalizing "gain" factor, whose value will be given in a moment. We choose the boundary z=0 as convenient, since certainly negative inner

products are not in the region. Figure 2 indicates some inner product regions, and the advantage of keeping more than one coordinate large.

This simple z will work in some cases but not others, however. Clearly, if W = (5,5,0,0,0) for example, any vector having the first two coordinates large will appear to be in the region, regardless of the other coordinates. The inner product alone can tell nothing about X coordinates whose corresponding W coordinate is 0. To overcome this, we break W into two parts, those which are large, and those which are small. We may speak of these two parts W_1 and W_2 as though they add to form a vector W', but this is not quite appropriate, as equation 3 will show. Let t_w and h be small positive constants. Then if j is the coordinate index, then:

If $|W(j)| > t_w$, set $W_1(j) = W(j)$, and $W_2(j) = 0$

If $|W(j)| \leq t_w$, set $W_1(j) = 0$, and $W_2(j) = h(t_w - |W(j)|)$; for all j.

with $\text{sign}(W_2(j)) = \text{sign}(W(j))$ 2)

The h and t_w are selected appropriate to the scale of the W(j). (In this work, t_w=3 and h=2, since $|W(j)|<16$). W is thus broken into an "excitory" part W_1 and an "inhibitory" part W_2. We may now define the z with inhibition as:

$$z = k \max\left(X * W_1 - \sum_j |X(j) W_2(j)| \;, 0\right) \quad 3)$$

Thus a vector X which gives a large $X*W_1$ product but has components in the region defined by W_2 will give as well a large absolute product with the W_2 vector, resulting in a small z as desired.

The function of k is to insure that nodes may be assembled in levels, the outputs of one level becoming the inputs to the next (see below). For graphs of only a few levels depth, an acceptable estimate for k is

$$k = \frac{\underset{j}{\text{average}}\,|W(j)|}{W*W}$$

Thus an "average-signal" feature X will result in z values of this average value.

Reference to Figure 1 shows the visual or feature data X as a vector entering the node and resulting in the closeness value z, and in addition, an opinion vector V(j), where j is the category index which determines the opinion output vector Y. The determination of V(j) will be described in the next section. Here we note that the output opinion Y of the node will be the scalar z times the vector V, i.e., Y = zV. Thus the output opinion of a node is the estimate Y of the category to which this node corresponds times the estimate z of the membership of the input vector X in this node's region. One could also think of z as the probability that X is in the node

region R, and V(j) as the conditional probability that the category is j given a vector X in R.

3.3.3 Node Interconnections and the Graph Structure

Given a sequence of feature vectors X_i, where i is the "time" index, we wish to establish a graph structure which represents both the spatial and temporal behaviour of X_i. Recall that nodes whose inputs are feature vectors are termed level 1 nodes, and correspond to a fuzzy region of feature space. We add nodes to the graph inductively, starting with no nodes, and create or add a level 1 node at time i if X_i has at least two coordinates $X_i(j)$ and $X_i(k)$ say, greater than some threshold t and X_i is not in the region of some existing level 1 node, to avoid redundancy. If the regions belonging to the level 1 nodes are well separated, then not more than one level 1 z should be large simultaneously. However, the inner product z function often results in considerable overlap of regions, and an X_i in the intersection region of node p and node q will result in z_p and z_q being simultaneously non-zero.

Keeping this possibility of intersecting regions in mind, we ask: how are we to accomodate temporal relations between the regions, that is, if an event is characterized by the X_i vectors passing from region R_1 to R_2 to R_3, how shall the graph reflect the fact that z_1 will be large,

followed by z_2 and then z_3? (We are using subscripts to indicate node or region here, and we also use them to indicate time, as in X_i. Context should signify which is intended). We may do this as follows. As the sequence X_i leaves a region R, instead of the z values being those given by equation 3), let them have a linear decay, with a time constant greater than the time normally taken for X_i to pass between regions. That is, if i is the time index and j the node index, then:

1) Calculate z_{ij} from 2) as usual.

2) If $z_{ij} < z_{i-1,j}$ then set $z_{ij} = z_{i-1,j} - c$,

where c is the decay constant. Otherwise do not change z_{ij}. Node z values thus decay for a time sufficient for the X vector to pass into another region, and the following possibility presents itself. Form a new level 2 node n if at least two level 1 nodes have z values exceeding some threshold t. This will accomodate either the intersection case or the temporal case, although admittedly n cannot distinguish the two cases. Often such a distinction is not required, since if a vector X_i is passing from R_1 to R_2, it usually goes through the intersection, and there is a continuum of cases between the extremes: a) jumping instantly between two disjoint regions, and b) entering and then leaving the intersection of R_1 and R_2, without entering R_1 or R_2 alone. The only difficulty occurs if

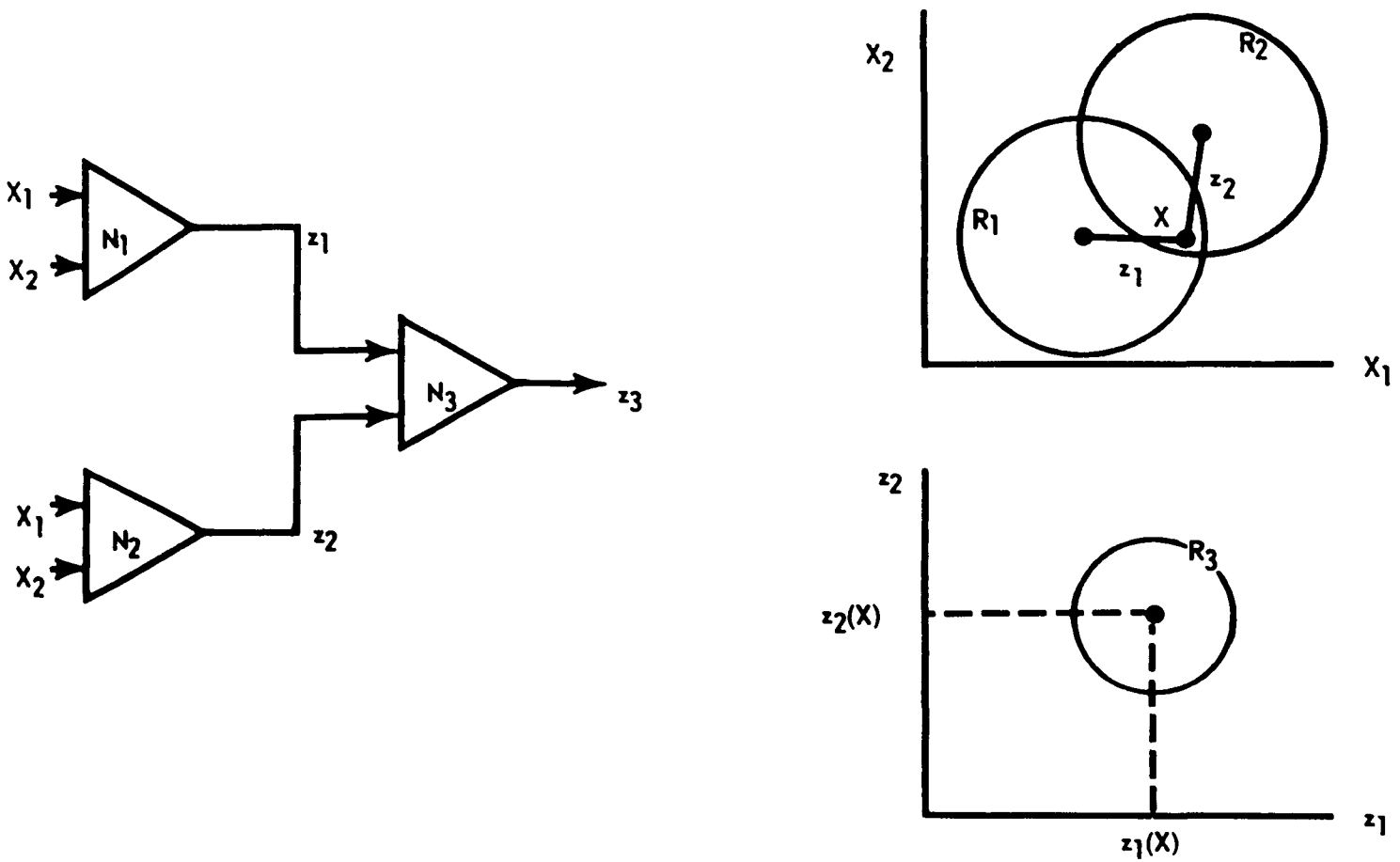


FIGURE 3. RECOGNITION OF INTERSECTION OF TWO REGIONS

To facilitate drawing in two dimensions, regions have been drawn as circles, which of course they aren't. The vector X has resulted in the formation of node N_3. Any X in both R_1 and R_2 will now be in R_3 in the z_1z_2 plane.

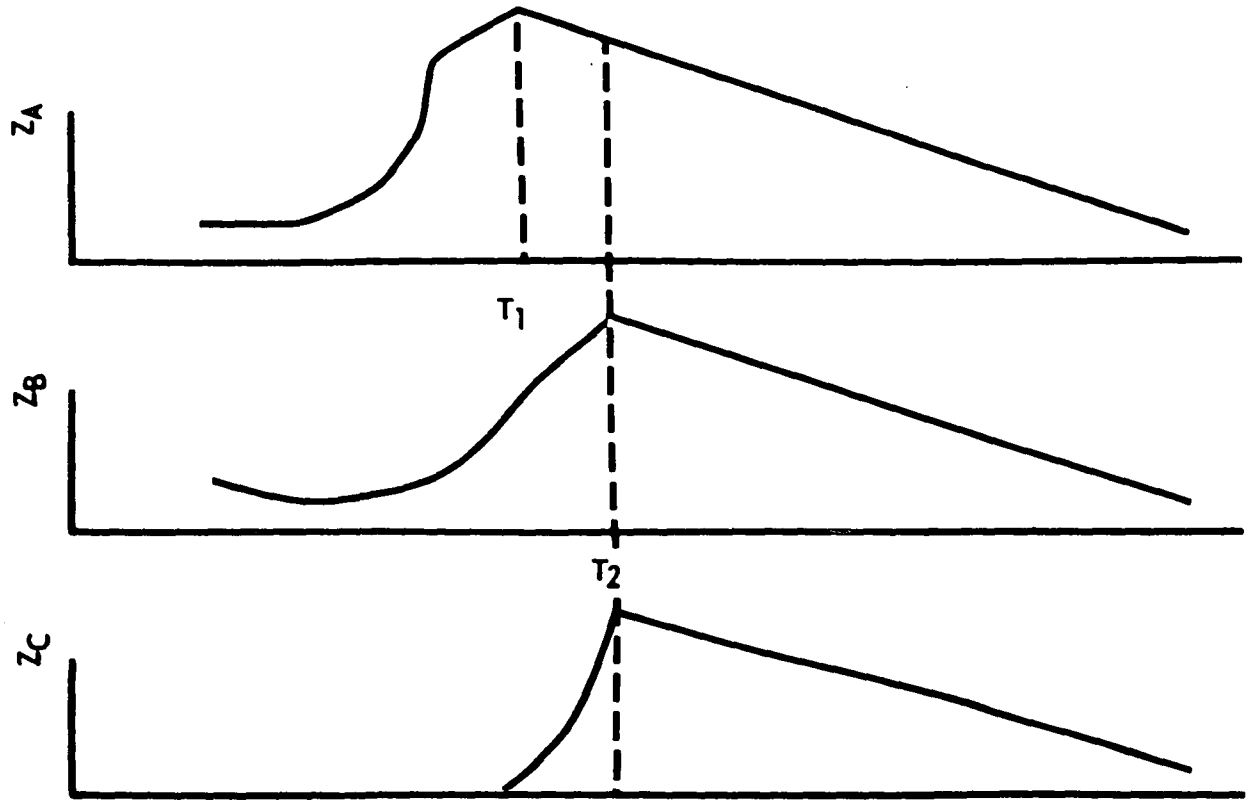


FIGURE 4. RECOGNITION OF A TEMPORAL SEQUENCE BY THREE NODES.

Nodes A and B are at some level n, while node C at level n+1 accepts their outputs Z. A is in decay at T_2 when B occurs, causing C to respond.

the data does contain such possible ambiguities in different categories, or if it is required to represent the spatio-temporal structure uniquely, for instance, when it is desired to have the machine reconstruct a waveform as it has learned it. Figures 3 and 4 indicate these two cases.

We shall endow all nodes at all levels with the same linear decay functions, and the general node formation rule becomes:

FORMATION RULE: Form a new node on level n+1 if at least two level n nodes have z values exceeding a threshold t, and no other level n+1 node has a z value exceeding some threshold t_u.

In this definition, the features themselves are considered level 0. The weights W_1 and W_2 (the "excitory" and "inhibitory" weights) are determined, as an initial estimate of the region, by taking the current X_i as the vector W and applying the rule 2).

The opinion vector V must be determined by the teacher, who provides an estimate of the category to which he wishes this X_i assigned. For example, if he is certain that X_i is category 1, and there are 4 categories and the V(j) have a range of 0 to 9, then he may set V=(9,0,0,0), etc.

Probably the first estimate W of the region R is not the best, and as subsequent examples of the same situation repeat themselves, the X_i sequence will again pass through the same region, though on varying

trajectories. When X_i again reaches a point "well in" the region of node n , we may obtain a new estimate of W and V as follows. Let the "update" threshold be t_u, and W be the vector recomputed from W_1 and W_2. If $z_i < z_{i-1}$ and $z_i > t_u$ then X_i is probably the closest to W that this current sequence will get. Recompute W and V by simply averaging the X_i and whatever V_i is available (the instantaneous teacher opinion input) with the existing W and V. That is:

UPDATE RULE: 1) Reconstruct W from W_1 and W_2 when the above conditions on z are met.

2) The new estimate of W is just:

$$W=(mW+X_i)/(m+1)$$

where the W on the right side is the reconstructed W, and m is the number of previous updates. Recompute k from the new W. Similarly:

$$V=(mV+V_i)/(m+1)$$

3) Recompute W_1 and W_2 from equation 2)

3.3.4 Obtaining an Output from the Graph

Since we have many nodes, each offering an "opinion", we must obtain some consensus. Selfridge (1959) was one of the first to anticipate this problem and suggested taking the largest opinion. This is acceptable only if one can be assured that the largest is statistically the most reliable. Frequently methods with statistical

underpinnings are seen in the literature (e.g., the so-called Bayes estimates) which tell one similarly to choose the "best" or "least risk" estimate. In practice however, one requires very large samples to obtain a good estimate of which is best, and any statistically based method requiring knowledge of distributions would undermine the objectives of the fuzzy approach. Tests using the greatest output or outputs (with the very low precision used, 4 bits, several may be equal) showed that this method was not satisfactory. Averaging the greatest two was better and the greatest three again better. Since continually (at each time step) ranking the outputs to obtain the largest n outputs begins to increase computation, an alternative method was sought.

Another method which is intuitively appealing is to average all outputs which exceed some threshold t by simply adding them. This method was found superior to the ranking method, and offers more possibilities for theoretical development. Thus

$$Y_{out} = \sum_j Y_j \qquad , Y_j > t_{op} \qquad 4)$$

where Y_j is equal to zV_j as usual, and j is an index over nodes. It will be useful in what follows to refer to this output opinion vector Y_{out} as a vector of "votes" for the various categories. If category k is given the largest vote, (i.e., Y(k) is greater than all other Y(j)) then

we will call k the opinion of the machine (the category assignment).

This output is of course, not a structural description of the data, but merely a single name assignment. The incorporation of mechanisms for more linguistic output spaces is beyond the scope of the present work; however, since one major aspect of the fuzzy graph is to obtain such descriptions, some indication of how this could be accomplished is in order. Let us consider a fuzzy predicate such as "is a member (part) of". If the predicate is considered an ordered pair of fuzzy vectors (V_1, V_2), where the V's are name vectors exactly as before, and (V_1, V_2) means "V_1 is a part of V_2 ", then the teacher may input in place of V alone, V_1 and V_2 , where V_1 is the name of the part which belongs to V_2, the parent. Thus a node carries an estimate of the part name and the parent name to be associated with a region of visual space.

We may illustrate how the structure of the graph would be interrogated to yield a part-parent description of a category (which is the ultimate parent) by assuming, for the sake of argument, the availability of an interactive graphics system. Since each node has a large number of weights associated with it, the system would not show any weight values or graph edges until asked to. We select a category k, and the system indicates the deepest level (=d) nodes whose V_2 votes are most certainly k. Note that there may be a number of these, corresponding to disjoint

regions of feature space which in various training examples have been designated as k. We then choose one of these level d nodes, and the system displays its V_1 and V_2 vectors, where the V_2 is of course the selected category, and V_1 is any part description given to this node. Since this is a deep node, it will have been formed only as a result of many earlier nodes, which are in turn, parts of other structures. Thus the deep nodes would not be expected to possess much of a part vector V_1, since part vectors should only be input on small local regions. At deep levels then, we expect large V_2 and small V_1, and analogously, at the first levels, which are relatively broad in their feature space associations and may well belong to many different categories, we expect weak V_2 vectors and sharper V_1 vectors.

We may proceed backward from level d, the system indicating which nodes are the major inputs to any selected node, obtaining both part and whole description, so that the parts, and parts-of-parts may be seen, as well as parts which are shared by other parents. Two categories could be selected, with an indication (such as increased intensity) of those nodes belonging to both, so that the common parts could be identified.

This very brief description of a possible extention of the present work is admittedly not sufficient to satisfy the reader that the graph is providing an adequate description of the data structure. It is included

to illustrate the manner in which one could extend the graph structure as it is presented here; a flexibility not enjoyed by simpler pattern recognition schemes. In Chapter 4 an example will be given of a manual recovery, by printing of the graph z and Y values during an event, of the structure of the input waveform.

3.4 Why Has This Structure Been Chosen?

At this point the reader should be wondering exactly what kind of function a graph such as we have just defined computes. Let us for the moment consider the graph without the time delays, i.e. as a function mapping a single point in feature space into a single point in the output space. The expression 3) is not very tractable from a mathematical point of view, since it contains the two non-linearities (the max function and the absolute value) which defy simplication. Why then use such a function? Why not use the more manageable and popular max and min functions from Zadeh's theory, and obtain a "Zadeh-type" fuzzy AND/OR graph? Certainly this would have the advantage of familiarity since AND/OP graphs and minimax methods are well studied in the literature.

The reason is that the writer feels that the minimax functions, while having good algebraic qualities, are not as suitable for real fuzzy sets as one would like. Max and min functions are desirable for their algebraic

properties on the fuzzy domain (termed L by Goguen, who generalized Zadeh's use of the unit interval to a completely distributive lattice with zero and infinity). However they are also "pessimistic" (see Santos (1968) and Mizumoto et al., (1969)) in that they guarantee bounds by taking worst case estimates. Such pessimistic estimates have been seen to function poorly in game playing situations when pitted against humans, since certainly humans do not make decisions based solely on the worst case (Slagle and Dixon, 1970). The problem of how much and what kind of algebraic structure to incorporate is one well familiar to those working in computer methods for handling natural language. Chomsky's formalism for representing language had to abandon the traditional structures of algebra, and still Quillian was unable to use Chomsky's methods in his TLC program. Until a theory of fuzziness can be developed which relinquishes some of the algebraic structure of L in favour of more flexible structures, we must be content with either the restrictions of formal fuzziness or the lack of rigour of the present informal fuzzy approach.

The graph structure which results from applying an algorithm of the sort described here to some real data may probably best be considered as a fuzzy <u>parse</u>, in the sense of a phrase structure grammar. A given data structure is reflected in a corresponding graph structure as a hierarchical set of relations between parts and sub-parts. Since level 1 nodes represent regions of feature space,

two can be on simultaneously only if they are not far apart and the input vector falls in their intersection, resulting in a level 2 node for the occurence of this pair. In a larger feature space, one would not want to have nodes with large numbers of inputs - with this metric their performance decreases as the number of inputs increases. The number of inputs per node would have to be restricted to a small number (say 8), and hence a number of nodes would be on simultaneously to handle situations with more than this number of features. Nodes at the next level would then play the essential role of coupling together these smaller nodes. Such large situations would be handled as indicated earlier by a search algorithm which might go something like this. The most frequent (or important) features would be computed first, then the first level nodes involving them, then the second level nodes involving these, etc. This would require a change of pointers defining the inter-node (edge) relations from the sink node, as they are now, to the source node, where the edge is from source to sink. Additional features and nodes would be computed until a sufficiently sharp output was obtained.

This degree of sharpness, or quality, is defined as follows. If the category of an input X is k, and the vote for category j is Y(j), j=1,...,n, then the quality is:

$$q = \frac{Y(k)}{\sum_{j \neq k} Y(j)+1} \qquad 5)$$

The 1 is in case the sum Y(j) is zero.

Returning to the time function case (i.e., with decaying node outputs as described above), level n+1 nodes serve the main purpose of detecting time sequential relations between level n nodes, as Figure 4 shows. Due to the non-uniqueness pointed out above, two events occuring closer in time than their prototypes will give a larger output than the prototypes, and similarly larger intervals will give smaller outputs, not an undesirable result.

In this chapter, we have tried to show how the fuzzy graph is capable of both a visual and a verbal description of an event, acting as an interface between a feature space and a description space. In the next chapter, the results of a program written to experiment with these ideas are to be presented.

Chapter 4. Program Details and Performance

This chapter will present the details of the program which was written to gain some numerical experience with the methods outlined in the last chapter. Many early results obtained during the period when the present form of the program was being evolved are not presented; rather only a sample of the more revealing aspects of the current program's performance will be given. The main interest will center on the effect of changing the various parameters, most of which are thresholds. The two effects of most interest are of course the error rate and the size (number of nodes) of the graph. We begin with a description of the data and its acquisition.

4.1 Data Acquisition

Data was recorded from five subjects, ranging from a male of 24 to a female in her sixties. Two channels of data were recorded, consisting of deliberately created eye motion artefacts. One channel was from an electrode pasted about 2 cm lateral and 2 cm above the eye, the other being 2 cm lateral and 2 cm below the eye. This arrangement gave a reasonable approximation to the four binary combinations of positive and negative-going waves in the two channels when the subjects were commanded to roll their eyes up, down, left or right. Originally it was attempted

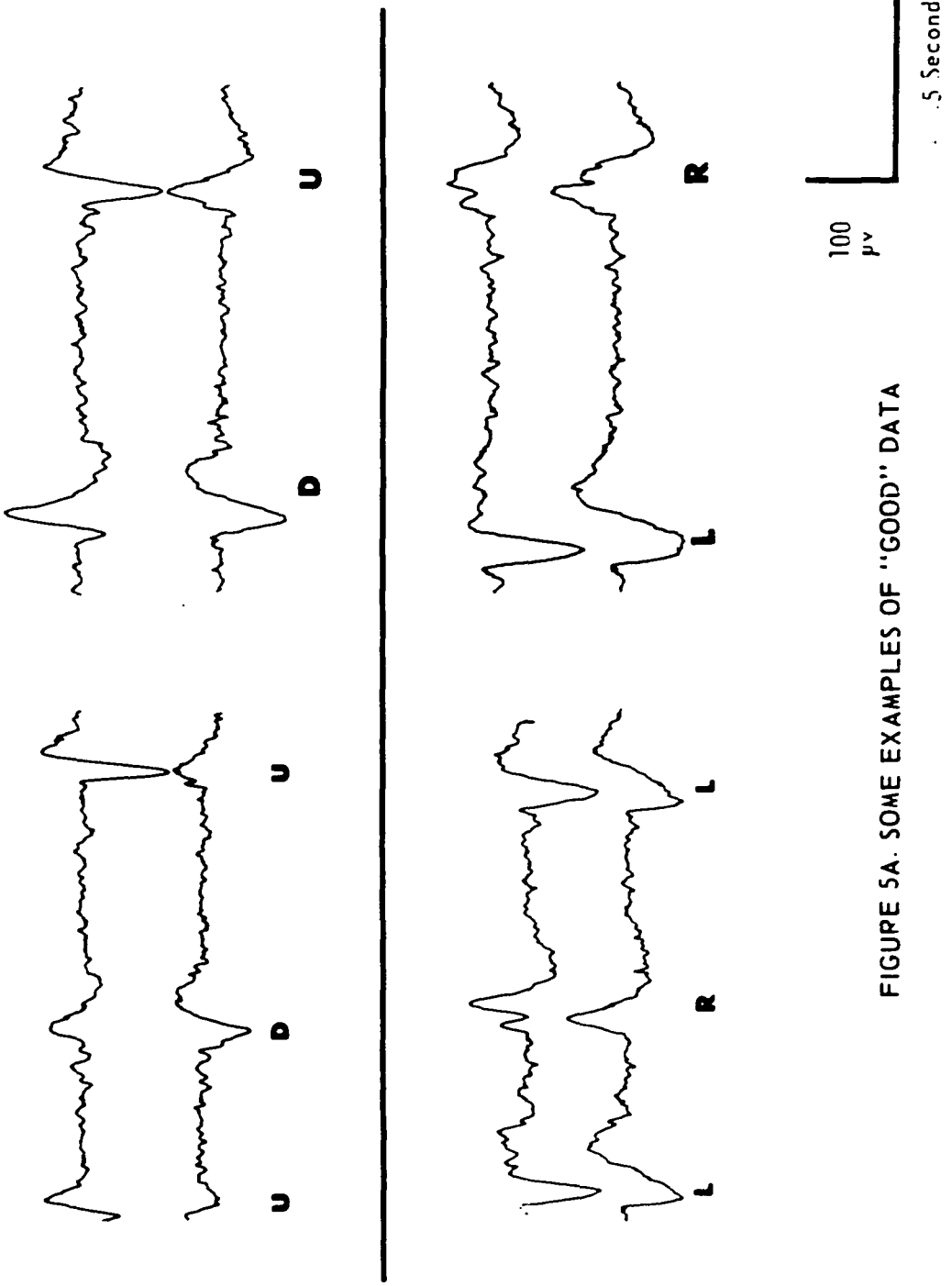


FIGURE 5A. SOME EXAMPLES OF "GOOD" DATA

FIGURE 5B. SOME EXAMPLES OF "POOR" DATA

to record more than four classes of signals by including eye blinks, squints, and the four eye rolling motions performed with the eyes shut. The experimenter however was unable to learn to classify these with any certainty; it was decided therefore not to ask a machine to do so. Even the four eye-open rolling motions showed such variability that the experimenter could not always identify the signals. Frequently one of the channels would seem to "drop out" in that only a slight wave of unrecognizable shape or polarity was present. The EEG itself, originating in underlying brain tissue, served only as a source of noise added to the artefacts. Figure 5 shows a number of examples of the four signal classes.

The recordings were made on an analog (FM) tape recorder with a bandwidth of 1 to 20 Hz., along with verbal identification of the commands to roll the eyes. They were then digitized at 150 samples/sec with a PDP-12 computer and stored on digital magnetic tape with a precision of 6 bits. This data was then edited visually using the PDP-12 display to select those examples which seemed acceptable. Many signals were unsuitable due to low amplitude, super-imposed jerks of the eye resulting in saw-tooth shaped signals, and strange waves due to the eyes taking a deviation or blinking while rolling. The edited data contained 80 up (U), 69 down (D), 52 left (L) and 76 right (R) signals. (The smaller number of lefts indicates that they were the poorest in quality). This

selection included a number considered so poor that the machine was not expected to recognize them. This data was then transferred to the 2314 disk of the McGill 360/75 on which the main programming was done in PL/1. A digital plot of the data was obtained by which each event (one of the four signal types) could be verified and correctly named by referring to the original analog tape. The events occurred in the edited data at random times, with no overlapping. The data was arranged in records of 512 sample vectors (i.e. a vector of the two channels) per record so that records could be presented to the machine in any desired order. No events spanned two records to simplify the programming. Records contained either UD or LR events only, or a mixture of all four types of events.

4.2 Feature Extraction

Since the determination of good features is a difficult problem in itself and is dependant on the particular data, and since it was not an objective of this study to analyze eye motion artefacts per se, little attention was given to feature extraction. A pre-program was prepared which accepted the raw data from disk, applied a simple four point low pass filter, of the form:

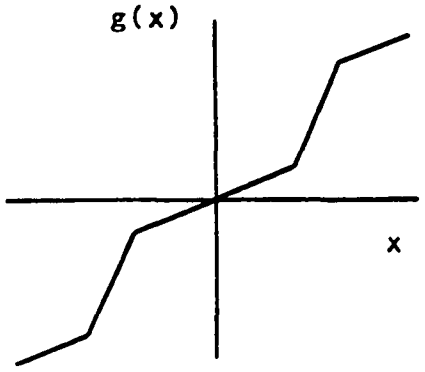
$$y_i = -\frac{5}{12}x_{i-2} + x_{i-1} + \frac{x_i}{4} + \frac{x_{i+1}}{6}$$

and wrote back on disk, for each time sample, a six component feature vector, three for each channel, as follows.

Let the smoothed signal in one channel be denoted by x_i, a scalar proportional to the electrode voltage at time i, with 6 bit precision. Compute:

$$f'_{i2} = x_{i+1} - x_{i-1} \qquad \text{and}$$
$$f'_{i3} = f'_{i-2,2} - f'_{i+2,2}$$

that is, elementary first and second derivatives. It was found that the signal and the f'_{ij} had an unsatisfactory dynamic range, since it was desired to reduce them to 5 bit features. Taking a hint from biological systems which handle wide dynamic ranges in a non-linear way so that the "physiological" range is given the greatest precision, and very small and very large signals are attenuated, the three values x_i, f'_{i2} and f'_{i3} were subjected to a memoryless piece-wise linear transformation g, of the shape shown below, to emphasize the interesting range and to suppress too small or too large values. The exact values of the function g need not concern us here. The three features for each channel which were recorded on disk then are:



$$f_{i1} = g(x_i)$$
$$f_{i2} = g(f'_{i2})$$
$$f_{i3} = g(f'_{i3})$$

where f_{i3} is given the sign of x_i, and the sign bit is in addition to 4 magnitude bits. Thus the input features have 5 bit precision, since they naturally are signed values. All subsequent z values, which are non-negative, are only 4 bit numbers. It would be interesting to try features with only 3 bit magnitude, and the author suspects they would work. Certainly the sign is the most significant bit.

It was discovered some time after having used these features for the initial development on the graph program that there were cases where the features did not distinguish categories sufficiently well. To properly test the graph performance, an improvement was added as follows. Two new features were added, one proportional to the difference between the two signal channels, and the other inversely proportional to this:

$$f_{i4} = \frac{3}{4}(x_{i1} - x_{i2})$$
$$f_{i5} = \text{sign}(f_{i4}) \frac{16}{2 + |f_{i4}|}$$

Thus the feature vector has 8 components for each time i: the f_1, f_2, and f_3 for each channel, and f_4 and f_5.

The verbal information was recorded on disk along with the visual information as a number v_i at each time step i, where v =0 if no event was occurring, and v =1,2,3 or 4 if the event was U, D, L, or R, respectively. The beginning and end of the event was a subjective decision by the author. These v_i were used to define the verbal information used to create the vector V_i (during node formation) or for updating V, with no attempt to grade the teacher's opinion of the category, as follows:

If v_i =0, V_i=(0,0,0,0)

If v_i =1, V_i=(9,0,0,0)

If v_i =2, V_i=(0,9,0,0) ... etc.

The range 0 to 9 for the opinion coordinates was arbitrarily chosen to not exceed one column printing width. Thus nodes which are always updated on the same category will retain the sharpest opinion, 9 for that category and 0 for the others. A node updated half the time on category U and the other half of the time on category R will have an opinion (4,0,0,4), the truncation being insignificant.

These features have been chosen to be mathematically simple, involving a minimum of computation, and reflect the criteria which the author found himself

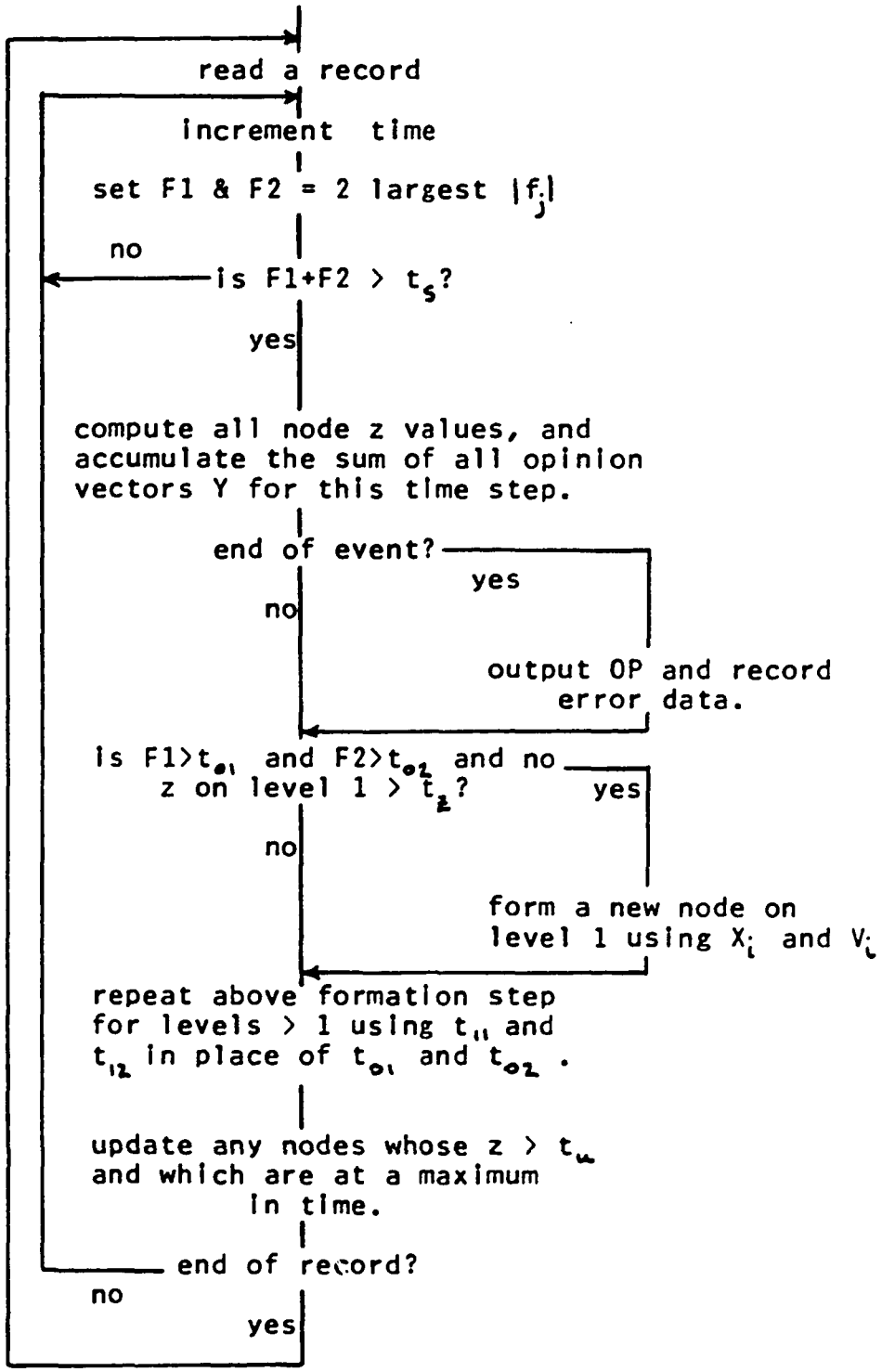


Figure 6. Flowchart of Main Program

to be using in recognizing the data types. They will be seen to suffice for the larger and less distorted examples, but fail to give an adequate measure of the poor quality waves. As we have stated earlier, this deficiency is not the direct concern of this study.

We have now specified the action of the pre-program which prepares on disks a set of features (visual input) and teacher opinions (verbal input) which the main program accepts as input with which to build the fuzzy graph.

4.3 The Main Program

The main program, which performs the recognition, reads the features and the names of the events from the disk a record at a time, so that the records may be presented in any order. Figure 6 is a flowchart of the essential program details.

We will explain Figure 6 briefly. A record is scanned until the sum of the absolute values of the two largest features F1 + F2 exceeds a threshold t_s which was fixed at 4 for all that follows. This is simply a means of skipping over noise between events. When t_s is exceeded, it is probably due to an event, and we compute the values z (closeness) and Y (opinion) for all existing nodes. Admittedly this is very wasteful, since most nodes are inactive (z=0) at any time. Since we wished to direct

attention to the node function and graph properties in particular, and since the number of nodes involved in the data at hand was economically small, the question of computing only active nodes was set aside. We shall write OP for Y_{out} for convenience. OP is the sum of all Y vectors taken over all nodes and over the duration of the event, where any components of a Y which are less than some threshold t_{op} are set to 0 in the sum. (This is a heuristic, like most of the functions found in this program, and was discovered by trial and error). The vector OP is printed at the end of each event along with an estimate of the quality of OP, q, obtained as described in Section 3.4. q is thus a large number if the OP is correct (i.e., the largest component of OP is OP(j) after event j). A 4x4 confusion matrix is maintained in which, at the end of each event, the OP vector is added to the i-th row if the event was of class i. An error matrix is also maintained, in which 1 is added to the (i,j) element, where i (the row) is the event class and j the index of the largest OP component. If more than one OP component has the same value (or all are 0) then nothing is added to the error matrix, but this is recorded in the total count of all errors.

The formation of new nodes is handled identically at all levels, except that distinct threshold values are used for level 1 (which receives the features themselves as inputs). If F1 and F2 are the absolute values of the

two largest features at any moment then if F1>t_{01} and F2>t_{02} and no level 1 node has a z value>t_3 then a new node is formed at level 1 with weight values W determined as explained in 3.3.2. The opinion or name vector V for this node is set to (9,0,0,0) if the current event is class 1 (U), (0,9,0,0) for D, etc, and V=0 if there is in fact no event in progress at the moment. Nodes on higher levels are formed in the same manner, using the thresholds t_{11} and t_{12}. These four thresholds and the fifth, t_3, control the density of nodes in the feature space. Separate thresholds were introduced for level 1 so formation at level 1 could be controlled independently of the other levels. Many of the experimental results which follow are concerned with the effects of these five thresholds.

The updating of a node occurs if the z value exceeds the threshold t_4 and is experiencing a maximum in time, as described in Section 3.3.3. It was found that the number of updates allowed any node, n , and the precision of the weight vector W, were important aspects of the updating, as will be seen in the examples which follow.

4.4 Program Performance and Discussion of Results

We now present a number of examples of the program performance, which have been selected to illustrate various aspects of the technique. To give the reader a better

Time	f_{11}	f_{12}	f_{21}	f_{22}	f_{31}	f_{32}
252U			7	-7		
253U	2	-2	10	-10		
254U	5	-5	10	-10		-3
255U	6	-7		-4	5	-7
256U	5	-8	-9	3	3	-6
257U	2	-6	-11	10		
258U		-2	-11	11		
259U	-1		-8	10	-1	
260	-2		-2	5	-1	
111			-4	9		
112D	-1	3	-7	10		
113D	-3	6	-5	9		4
114D	-4	7			-1	6
115D	-3	7	1	-6		1
116D	-2	4	5	-10		
117D	-1	1	5	-10		
118D			4	-7		
227U			7	-6		
228U	2	-2	9	-10		
229U	4	-5	6	-10	1	
230U	5	-7		-6	3	-6
231U	4	-8	-5			-7
232U	2	-6	-10	8		-2
233U		-4	-10	10		
234U		-1	-5	10		
235				6		
354			-2	6		
355D	-1	2	-5	10		
356D	-2	5	-5	10		
357D	-3	7	-1	5		5
358D	-3	7		-1		6
359D	-2	6	4	-10		
360D	-1	3	5	-10		
361D			3	-10		
362				-5		

Figure 7. Feature Outputs.

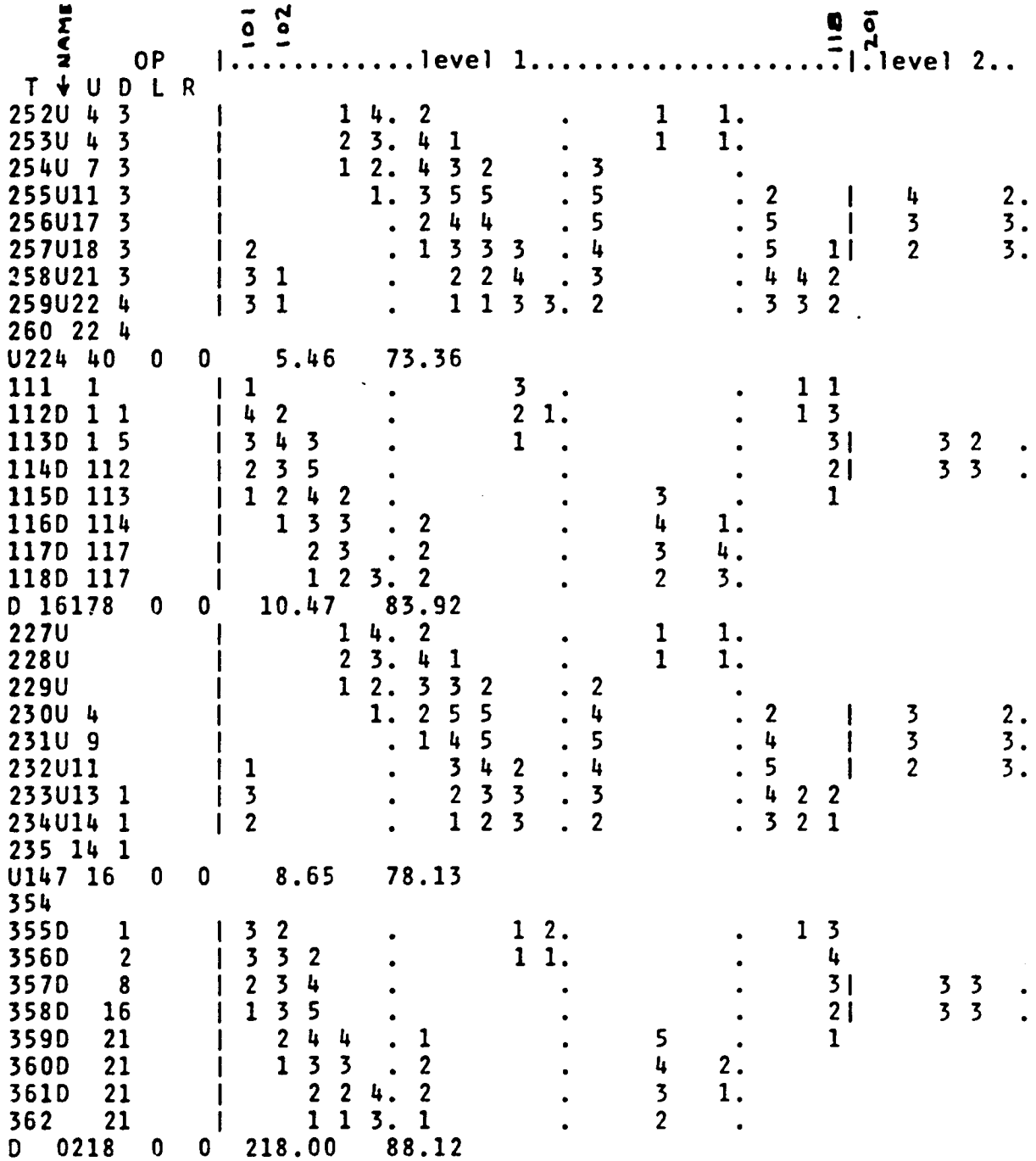


Figure 8. Graph Response to Examples From Figure 7.

"feel" for the numerous cases to follow, we first present in detail a simplified example of the program showing explicitly in Figure 7, the feature inputs as a function of time, the resulting graph response as a function of time in Figure 8, and in Figure 9 a schematic drawing of the graph at one instant, from which we will manually (by inspection) derive a fuzzy description of the data at that moment.

Recall that training consists in inputting a number of examples to the program and allowing it to construct the graph according to the rules we have discussed. Testing then consists in preventing any further formation or updates of nodes, and examining how the graph responds to additional similar cases. In our preliminary case, 16 U and 19 D examples were used for training, resulting in a graph with 18 first level and 5 second level nodes. Three training errors occured, and the training q was 61.

In Figure 7 we have selected several examples of the output from the feature program, where f_4 and f_5 do not appear, since, the reader will recall, they were added in the main program. They may be calculated from the expressions given in Section 4.2. The time appears at the left, followed by the teacher name input (U or D), followed by the six features f_{11} through f_{32}. The raw data is plotted using asterisk for channel 1 and dot for channel 2.

Figure 8 shows the main program output corresponding to the examples in Figure 7. The 8 columns following the name are the accumulating OP vector, divided by 10, 2 columns per category. Following the vertical bar are the z outputs for the first level nodes, followed by the second level nodes. The dot every five is to aid reading, and values less than 1 are not printed. At the end of each event is shown the full OP vector, the q and the average q value up to this example.

We note that at a glance, the response appears as a "flow" through the graph, as though it were a network, with different flow patterns in the U and D cases, as one would hope. The recognition is taking place on the large part of the wave only, since the features used do not measure any of the more subtle properties of the later parts of the waves. In the U example which begins at 227, we see node 202 (the hundreds digit refers to the level) and node 205 on simultaneously. The table of node weights (not shown) reveals that 202 is primarily driven by 106 and 107. By examining other U examples in the same test set, and by knowing that 106 is in decay, one may verify that 202 is the result of the temporal sequence 106, 107. Similarly, 205 is largely the simultaneity of 111 and 116, with perhaps a slight lead by 111. While the node weights themselves do not distinguish the temporal from the spatial (simultaneous) cases, as pointed out in Chapter 3, we may make the distinction for any input example by inspecting

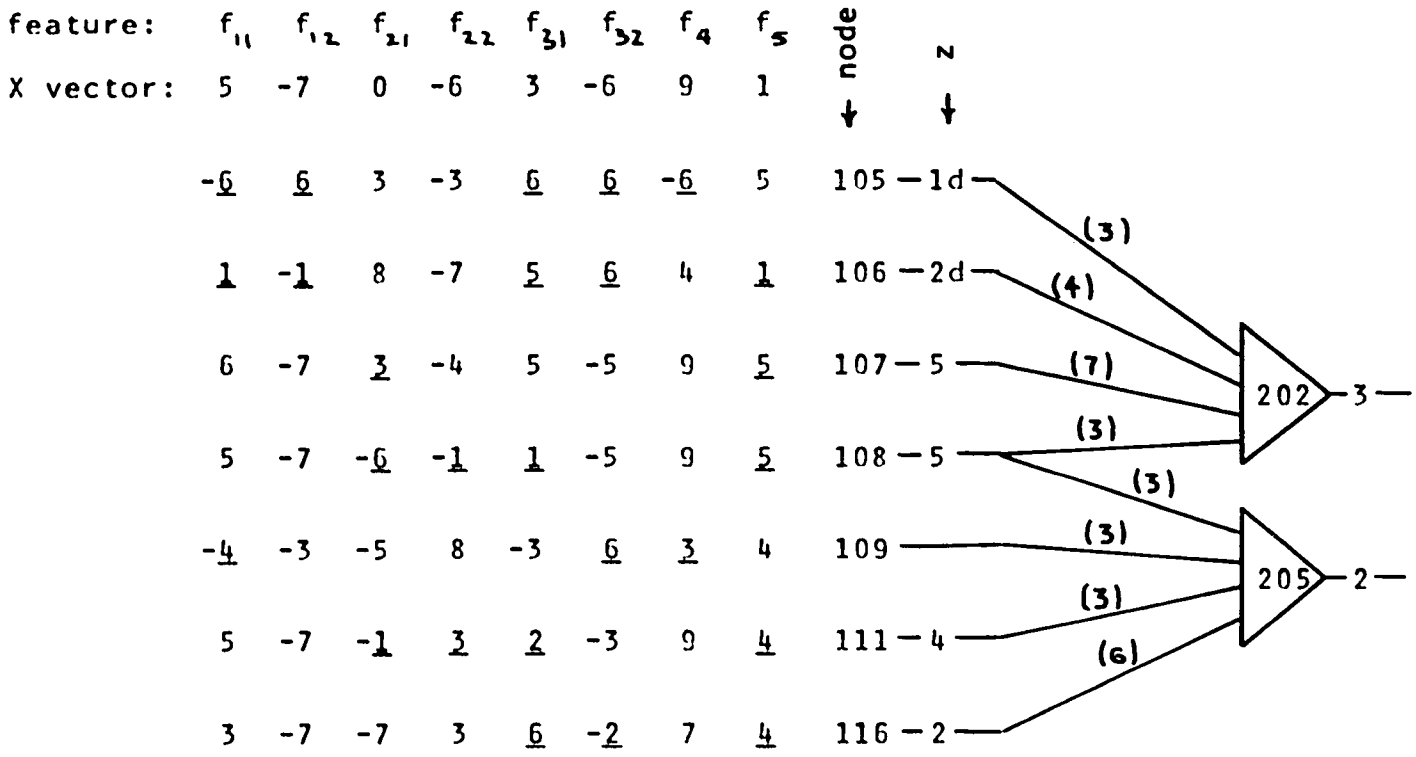


Figure 9. Active Graph Structure at One Time Instant.

The X vector at time 230 from the previous figures is shown above the weights W_1 and W_2 (underlined) for each level 1 node involved. For level 2 nodes, only the W_1 weights are shown in parentheses. d indicates a node in decay. For discussion, see text.

the z values, to note those which are in decay, as we note 106 to be here.

In Figure 9 we have selected the moment 230 from this U example and drawn the active portion of the graph, where the complete (8 component) feature vector is shown above the input weights to the level 1 nodes involved, so that one may compute the inner products if desired. The level 2 nodes being driven from these are shown with only their W_1 weights beside the graph edges, to facilitate drawing. We may make the following verbal fuzzy diagnosis from this graph at this instant, by determining the regions of feature space in which the input lies. Nodes 202 and 205 have directed our attention to the level 1 nodes shown, which tell us, by taking their weights as representative of the input situation, that channel 1 has a value of about +5 (f_{11}), with a very small slope, and a medium peak nearby. (We are making guesses about the feature vector from the weights of 107 and 108). The channel 2 signal itself (f_{12}) is about -7, with a slope between -4 and 0 (a wrong guess) , and is also near a peak. By noting that 105 and 106 are in decay, we infer that there has just occured a situation recognized by them. We leave it to the reader to describe it.

For the sake of clarity we have chosen this example to have only a few simple, small values. A careful examination of the three figures will greatly assist the

reader in understanding the examples of performance which follow.

* * * * *

In the following pages are presented a number of examples of the program performance, selected to illustrate the effect of changing various parameters. The examples are numbered for identification in the paragraphs of discussion which follow, and are summarized in Table 1, where they are called cases. The order of the cases in Table 1 is partially the order in which they were studied, with some grouping of related results where possible.

Before the discussion of cases begins, we list the various symbols used, and their meanings:

Thresholds:

t_{01} and t_{02}: If the absolute value of the largest feature F1 is $< t_{01}$, no new node is formed. If the absolute value of the second largest feature F2 is $< t_{02}$, no new node is formed.

t_{11} and t_{12}: Similar function to t_{01} and t_{02}, except for levels >1. These pairs of values were found to be more useful than the single threshold referred to in Chapter 3.

t_z: If any z value is above this value, no new nodes are formed on that level at this instant.

t_u: If a z value is above this value and is decreasing as described in Chapter 3, update that node.

OP: Refers to the method of obtaining the opinion from all nodes (see Section 3.3.4), as follows: a: OP is the sum of the 3 largest Y_j. b: OP is the sum of the 2 largest Y_j. c: OP is the sum of all $Y_j > 1$, as in 3.3.4. d: OP is the sum of all $Y_j > 0$.

A : A collection of data examples with UD pairs first, followed later by LR pairs, with some unpaired events. There are 13 U, 12 D, 11 L and 13 R.

B : Same data as A, only arranged in random order.

C : A collection of data from the same population (a selection of good to medium quality examples from all 5 subjects) as A, used for testing. Contains 20 U, 21 D, 12 L and 13 R.

D,E : See cases 26 and 27.

q : The average quality factor over all examples, computed for each example from eq. 5.

The following details pertain to the cases discussed below, unless otherwise noted. The maximum number of updates, m , counting the original (formation) W as the first update, is limited to 5. The threshold for

CASE	THRESHOLDS 01	02	11	12	z	u	OP	NO. OF NODES	TRAINING data	errors no	%	q	TEST data	errors no	%	q
1	7	5	5	5	4	4	a	36,16	A	1	2	2	C	7	10	1
2	7	5	5	5	4	4	b	36,16	A	1	2	2	C	13	20	1
3	7	5	5	5	4	4	a	37,8	B	4	8	2	C	8	12	1
4	7	5	5	5	4	4	c	36,16	A	3	6	48	C	6	9	34
5	7	5	5	3	4	4	c	36,37,3	A	5	10	70	C	7	10	38
6	7	5	9	9	4	4	c	36,0	A	8	16	27	C	10	15	19
7	7	5	5	5	4	4	c	36,16	A	9	18	38	C	26	39	32
8	7	5	5	5	4	5	d	30,10	A	9	18	13	C	6	9	3
9	7	4	5	5	4	5	d	43,10	A	12	24	14	C	8	12	2
10	7	5	5	5	4	3	d	50,10	A	12	24	18	C	9	14	4
11	7	5	5	5	4	4	d	36,16	A	4	8	20	C	6	9	3
12	7	5	5	5	4	4	c	48,15	A	9	18	29	C	14	21	16
13	7	5	5	5	4	4	c	31,12,2	A	4	8	46	C	7	10	10
14	7	5	5	5	4	4	c	37,8	B	10	20	23	C	9	14	24

Table 1. Program Performance.

the inhibiting vector W_2, t_w, is fixed at 3. The Y values from each node are divided by 10 to reduce printing space for OP, and will be referred to on this scale. Thus a typical value of z is 5, and the scaled Y values for a V=(9,0,0,0) would be 5Y/10, which truncates to Y=(4,0,0,0). We speak of this Y as a "vote" of 4 for U, since the vectors V and Y are ordered: U,D,L,R.

* * * * *

Case Discussion

1. The first few cases explore the various OP methods. Note the low values of q, compared with later values. This is because OP is the sum of only 3 Y values, so that if even all three have a single 9 in their V vector, the q will be about 15, since the typical z is about 5.

2. Like 1, only taking the 2 largest Y. The error rate becomes about 50% if only the largest Y is taken.

3. Like 1, only with random training data. Note that only half as many second level nodes were formed, though the first level is almost the same.

4. Here we use OP method c, the preferred method below. This case is the result of much experimentation with the thresholds, since formation thresholds too small result in excess numbers of nodes which respond to "noise" and essentially cover the feature space indiscriminantly. Too few nodes only respond to large, good signals. These values were found to be a reasonable compromise. Two of the 3 training errors were missed (not classified) events, being too small. Two test events were missed, and all U and R were correct. Note the much more satisfactory q value (34) from the earlier cases. This means that the average example is given a correct vote at least 34 times more certain than those in error, the average including the erroneous examples.

5. Here we drop the second level threshold slightly to see if additional nodes on this level will help. The slight loss in performance is due to excessive level 2 nodes. This optimum number of nodes seems to be a general principle: too many is as bad as too few. An unexplained phenomenon may been seen here which reoccurs in various trials: the error rate may be inversely related to the quality of recognition.

6. This case is the opposite to 5, in that no second level was allowed. We find that the first level is carrying most of the responsibility for recognition, since the waves

can be well separated by the features alone.

7. The program has been modified to function as in 4, except the OP is taken from the second level only, to better assess its contribution. One U and one L were missclassified, with no R missclassified. Nearly all D were missclassified, with the remaining errors being omissions (OP=0). The reason for the poor performance on D was not investigated, but is probably due to a spatio-temporal ambiguity.

8. Here we have reduced the degree of updating, by raising the threshold t_u slightly. We achieve an error rate of 9% with only 40 nodes. The increase in error rate during training over case 4 is not understood. The OP method has been changed to d, which results in a very low q, since allowing votes of only 1 into OP will cause votes in the wrong categories from nodes responding to "noise".

9. We attempt to improve on 8 by increasing the number of level 1 nodes, without success.

10. Another attempt to improve on 8 by increasing the update rate (decreasing t_u), again without success.

11. Here we have returned to the parameters of case 4, except continuing with OP method d (summing all Y>0). There is only a slight difference in error rate between OP method c and d.

12. We investigate the effect of the factor h, the inhibition constant from equation 2. h was 2, and now is 3, causing feature vectors which have large coordinates where they "should" have small ones (to be considered similar to a W which had these coordinates small) to have less effect. Otherwise the same as 4.

13. Like 12, only h is now 1. We conclude that h=2 is best.

14. Like 4, only with the training data rearranged in random order. Note that this causes considerably more training errors, due to nodes common to more than one class updating alternately to one, then the other, rather than many times successively to the same class. At the end of the training however, the graph has almost adjusted itself to the level of case 4, as seen by the test performance. In case 19 we will see that with a second repetition of the training data (before testing) the random training case can be made as good as the non-random.

CASE	THRESHOLDS						OP	NO. OF NODES	TRAINING				TEST			
	01	02	11	12	z	u			data	errors no	errors %	q	data	errors no	errors %	q
15	7	5	5	5	4	4	c	50+,19,3	B,B	19	19	24	C	15	23	30
16	7	5	5	5	4	5	c	38,40,6,3	B,B,B	26	17	15	C	13	20	17
17	7	5	6	4	3	4	c	27,3	B,B,B	24	16	27	C	11	16	19
18	7	5	6	4	3	4	c	21,3	B	10	20	32	C	13	20	49
19	7	5	6	4	3	4	c	27,3	B,B	18	18	28	C	13	20	13
20	7	5	6	4	3	4	c	21,3	B	10	10	33	C	11	16	51
21	7	5	6	4	3	4	c	27,3	B,B	16	16	29	C	11	16	25
22	7	5	6	4	3	4	c	30,3	B,B,B	23	15	29	C	15	23	29
21	7	5	6	4	3	4	c	17,2	B	8	16	26	C	8	12	42
22	7	5	6	4	3	4	c	18,2,1	B,B	14	14	23	C	6	9	30
23	7	5	6	4	3	4	c	19,2,1	B,B,B	21	14	24	C	9	14	35
24	7	5	6	4	3	4	c	17,2	B	9	18	26	C	8	12	41
25	7	5	5	5	4	4	c	36,10	A	10	20	31	C	13	19	6
26	7	4	5	5	3	4	c	37,8,4	D	37	29	20	E	28	20	47
27	7	4	5	5	3	4	c	45,16,16,1	D,D	76	29	20	E	28	20	61

Table 1, con't.

15. This is the first attempt to repeat the training set in the hope that it would improve the performance. We note that more than 50 nodes were required at level 1. It was decided that 50 was excessive, so no cases were allowed to exceed this limit. Apparently with thresholds too low, node formation will continue beyond what is necessary, into excessive redundancy, with a loss in performance, as seen in earlier cases.

16. An early attempt to stablize the graph so that repeated training will not form excessive level 1 nodes. The update threshold has been increased from case 15, so that three passes of the training set B result in 38 level 1 nodes, only 2 of which were added in the third pass. The second level has now become excessive, an unexplained effect.

17. A better method of restricting the number of nodes and achieving stability is to decrease t_2, the redundancy threshold. Note the improvement over 16.

18. We begin here some experiments with the updating, varying the maximum allowed number of updates, m, and the precision of the weights, which has been 4 integer bits and no fractional bits so far. This case retains precision (4,0) but m=10.

19. Like 18, only training set repeated once. Clearly not a good case.

20. Like 18, only m=20. The slight improvement is not sufficient to suggest that m is a crucial parameter.

21. In this case we add 4 fractional bits to the weights, and repeat case 20, that is, m=20. We find a considerable improvement, indicating a possible problem with the roundoff error in updating the weights with precision (4,0). This was then examined by inspecting all updates of nodes with the (4,0) precision in some reruns of earlier cases. Some of the weights showed updates where the roundoff did not take place, resulting in some loss of magnitude in those weights updated frequently. This effect was not always repeatable, and a bug in the PL/1 system is suspected. Regretfully, some of the earlier results may be affected by this problem, but it was discovered too late to repeat them. Since only 5 updates were involved, this problem should only affect the results slightly.

22. We repeat 21 with the training set passed twice. In this example we achieve the previous apparent limit of 9% error rate once again, suggesting that it is limited by the features or the data, a plausible conjecture, since examination of the examples which were missed or

misclassified showed that the features were in most cases not adequate for recognition. Note that only 21 nodes are formed, the best performance/size ratio found in this study.

23. Attempting to improve further on 22, we find that the system can be "overtrained". The precise cause of this effect was not apparent from studying the output, since it is a distributed effect over many nodes and time steps, so that the two outputs appear qualitatively the same. This is a decided problem in this type of heuristic programming.

24. We repeat 21 with m=10, and discover m to be more sensitive now that the precision is increased.

25. This case breaks the above sequence. We have repeated case 4, only have halved the magnitude of f_4, to illustrate the sensitivity to the features. Several other runs were made with similar slight changes in the features, all with similar degradation in performance. This indicates that the error rate is being limited by the features, and of course, is dependent on the particular selection of data used in the tests.

26. Finally, we put through the program the complete selection of data available on disk, consisting of 127

training examples and 142 test examples, which include many which are worse that those used above. For this and the next case, the precision was (4,4) and m=10. The size of the graph and the q were considerably better that anticipated.

27. We dare to repeat the training set, enlarging the graph somewhat but luckily improving the q without sacrificing the error rate. Of the 28 cases in error, 18 were misclassified, and 10 were too small to be classified at all, with no errors on U, which has been the cleanest category.

* * * * *

General Discussion

These results demonstrate that the fuzzy graph does indeed perform pattern recognition on noisy waveforms, with error rates of the same order as other pattern recognition programs, although this particular data type has not been studied, to the best of the author's knowledge. Not only does it assign a pattern to a category, but it provides as well an estimate of the quality of the pattern, including its degree of similarity to other patterns, and contains in the graph structure, should one wish to access it, a fuzzy description of the pattern structure.

A note on the efficiency of the graph is in order. Each node has an average of about 6 inputs (a W_1 or W_2), each W a 4 bit weight, (in those cases where the weight precision was not extended), and if the number of nodes per level is limited to 32, a 5 bit pointer per input. The V vector may be quite adequately 3 bits per category (though here we had the range 0 to 9), making 4x3 bits. The z is of course 4 bits, and the other constants may be accomodated in 20 bits. Thus 90 bits per node is sufficient for the present program (PL/1 certainly used more), and the 21 nodes of case 22 would require 1890 bits of storage, which is very little in today's computer. A graph of 10's of thousands of nodes

could be accomodated in core, with additional little-needed nodes maintained on secondary storage. The speed is fairly obvious from the arithmetic required in the node, and in an assembly language version of the program would be of the order of 50 to 100 memory cycles per node. Any reasonable search scheme should be able to reduce the number of nodes computed, in a graph of say 1000, to less than 100, so that with such a scheme, most machines today could process data 20 times as complex in 10,000 cycles per time step. This machine could use reject memories, since the occurance of errors is tolerated due to the distributed responsibility, though admittedly we haven't demonstrated this.

On the other side of the ledger, the program has several unexplained (though not as yet thoroughly investigated) quirks, such as sometimes getting worse when more highly trained (a possible human trait?), or sometimes showing lower quality with higher error rates, and being excessively sensitive to the weight precision and number of updates, an effect which could probably be overcome with a more subtle update rule. A great problem in diagnosing these phenomena has involved two of the very qualities deliberately built into the program: the fuzziness and the distributed responsibility for recognition. Printing out all the weights and all the z values of all the nodes at each time step is certainly a painful process, since there is often no obvious qualitative difference

in the appearance of the numbers in two different cases which are nevertheless performing quite differently, as exemplified by the weight precision problem. This is an inherent danger in all heuristic programming, but doubly so in fuzzy heuristic programming.

Chapter 5. Conclusions

5.1 Contributions of This Thesis

The pattern recognition literature has not given much attention to large on-going multichannel time series such as the EEG, preferring to study simpler two dimensional images. This thesis offers a possible approach to realistically handling such data, and though a full scale EEG is much larger than the data recognized here, it is felt that there are no other methods of waveform recognition which would be as naturally suited to such large-scale expansion as the present one.

However, the author prefers not to attempt this, but rather to consider the method of fuzzy graphs as presented here as an introduction via a demonstration to the general problem of bridging the considerable conceptual and practical gap between complex fuzzy data in some feature space and a sophisticated (and possibly intelligent) description of that data in some description language.

We summarize the essential aspects of this work as follows:

1. The introduction of the fuzzy graph concept as a means of mapping a feature (visual) space into some description (verbal) space is the primary contribution. The graph structure reflects the intent that the mapping between a fuzzy situation and a sharp one (such as a

syntactic description) should not occur abruptly, but should occur in stages or hierarchically, with the structure becoming sharper and more composite as the depth in the structure increases, the responsibility for any description being distributed over some part of the structure.

Apart from the fuzzy graph structure, the introduction of the inner-product membership function and its companion, the use of features with the magnitude-significance property, is believed to be new, and should be useful in other fuzzy set applications.

2. The fuzzy graph is a structure which admits the following additional possibilities, which have not been investigated here:

a) It is applicable to fixed images as well as functions of time, by removing the time decays in the node outputs.

b) It is suitable for incorporation into heuristic search algorithms to increase the speed of access in large software graphs.

c) It is potentially implementable in large-scale integrated circuit technology, since the node functions are simple, require low precision, and the distributed responsibility for recognition makes a large system failure tolerent.

5.2 Discussion and Conclusions

The most salient lesson to be had from this endeavour, at least from the author's point of view, is one already well known to more seasoned workers in the field of Artificial Intelligence, pattern recognition and heuristic programming. The point is, of course, that "heuristic programming" is in fact a euphemism for "programming in ignorance", and is at its best an unnerving experience, since like flying at night without instruments, one never knows if the runway or a mountain lies just beyond.

B. Raphael has suggested that AI is just "those problems which we don't yet know how to solve", and given this definition, the present work is certainly Artificial Intelligence. Almost all of the early work in AI and still the majority of the current research is still heuristic in nature, lacking precise theorems to guide the programming. Even the most formal area of current research, theorem proving, relies on clever guesses to improve a program's performance, and there is no formalization of the notion of the "quality" of a proof, except for the execution time. Pattern recognition based on statistical methods enjoys a considerable theoretical foundation; however the important questions in pattern recognition are now of the linguistic type, rather than statistical. Yet though certainly a step in the right direction, the

computer programs which analyse a scene or picture and report its structural relations (e.g. Guzman, 1968) are still a collection of clever heuristics. In natural language question-answering programs, there has been recourse to theorem proving methods to determine logical consequences, but the conversion from natural language to predicate calculus is only easily accomplished when the input sentences are of a kindergarten level of subtlety. Why is it that in all the programs which deal with natural language, no appearance of any of the elaborate mathematical linguistic theories can be found?

The writer has come to the conclusion, partially as a result of his experience with the present undertaking and partially as a result of the general course of research of this nature, that far more emphasis should be placed on research into basic mathematical structures and methods whose aim is directed towards real problem areas such as pattern recognition. The basic mathematical areas which the author has in mind are automata and formal languages, graph, category and topological algebraic theory, and the newer notions such as fuzziness which are not yet properly integrated into other theories. (c.f., Watanabe, 1969). Automata theory is a well developed area (relatively speaking), and should serve as a focal point. In this regard, there have appeared recently several papers on fuzzy automata (Wee, 1967, Santos, 1968 and 1970), and papers on automata methods for pattern-recognition oriented

languages (Feder, 1968, Brainerd, 1968, Montanari, 1970, Steingrandt and Yau, 1970, S. K. Chang, 1971). The important paper of Goguen (1967) which introduces the language of category theory to fuzzy set theory, has begun the incorporation of fuzziness into other mathematical domains, although the emphasis on algebraic notions is perhaps not as important as topological aspects. If fuzziness is to have any traditional mathematical property, it ought to be continuity, which is one of the glaring shortcomings of sharp theories. From continuity we could procede to limits, where the notion of abstraction (c.f. Bellman and Zadeh, 1966) would be defined as a limit of a collection of fuzzy examples, where the limit in the language describing the examples is guaranteed by continuity. The only paper merging fuzziness with set theoretic topology known to the writer is that of C. L. Chang (1968). It is based on the Zadeh assumptions that the fundamental notions of set theory are those which should be preserved in the fuzzy theory (e.g., strict containment, complement, union, De Morgan and distributivity laws). Only Watanabe (1969) has questioned this premise, suggesting instead that the notion of experimental verification and implication in the causal sense are more fundamental. The writer considers these worthwhile avenues of inquiry (which he intends to pursue) but is unable to make any concrete suggestions at present.

The author wishes to venture the following suggestions, which he intends to follow himself, on some appropriate directions for continued research on these matters. What is needed is a formal fuzzy language, for pattern recognition or for general problem description, since the latter should be considered the proper framework for any particular problem such as pattern recognition. Banerji (1968) has made some initial suggestions in this regard, although without any fuzziness. It will be probably found necessary to regard pattern recognition as a sort of game against Nature, played with fuzzy information and with costs for making observations, some of which may change the state of Nature. To this end then, it will prove useful to invent a formal fuzzy game against Nature, that is, a rigorously defined set of fuzzy situations and rules for procedure, where the two protagonists (Nature and the human who would know more about Her) have usually different sets of rules (Nature often having advantage), and neither having complete information about the other's doings at any moment. Such a fuzzy game, like existing recreational games used for research in problem solving methods, will provide a laboratory for experimentation on complex problems, and must be expressed in the fuzzy language we have referred to, along with the algorithms for playing it. Unlike existing recreational games however, this game will adapt much more naturally and usefully to

real world situations which it is our ultimate goal to understand.

REFERENCES

Aleksander, I. Non-computer machine intelligence. Letter in: AISB Newsletter, Issue 12. March, 1971.

Ball, G. Classification analysis. Stanford Research Institute Technical Note. November, 1970.

Banerji, R. B. A language for pattern recognition. Pattern Recognition. 1: 63-74. 1968.

Bellman, R. and Zadeh, L. Abstraction and pattern classification. J. Math. Anal. Appl. 13: 1-7. January, 1966.

Bellman, R. and Zadeh, L. Decision-making in a fuzzy environment. Management Science. 17: B141-B164. December, 1970.

Brainerd, W. S. The minimization of tree automata. Information and Control. 13: 484-491. 1968.

Brazier, M.A.B. (Ed.). Computer techniques in EEG analysis. Electroenceph. clin. Neurophysiol. Suppl. 20. Elsevier Publishing Company. Amsterdam. 1961.

Burch, N.R., Greiner, T.H. and Correll, E.G. Automatic analysis of the electroencephalogram as an index of minimal changes in human consciousness. Fed. Proc. 14: 1955.

Chang, C. L. Fuzzy topological spaces. J. Math. Anal. and Appl. 24: 182-190. 1968.

Chang, S. K. Automated interpretation and editing of line drawings. Proc. S. J. C. C. May, 1971.

Chang, S. K. Fuzzy programs-theory and applications. Polytechnique Institute of Brooklyn. Proc. on Computers and Automata. 21: 1971.

Chang, S. L. Fuzzy dynamic programming and the decision making process. Proc. 3rd Princeton Conference on Information Sciences and Systems. pp. 200-203. 1969.

Chomsky, A. N. On certain formal properties of grammars. Information and Control. 2: p. 137. June, 1959.

Chomsky, A. N. Aspects of the Theory of Syntax. MIT Press. Cambridge, Mass. 1965.

Daly, J. A., Joseph, R. D. and Ramsey, D. M. Perceptrons

as models of neural processes. In: Computers in Biomedical Research. Chapt. 22. 1: pp. 525-545. R. W. Stacy and B. D. Waxman (Eds). Acad. Press. New York. 1965.

Farley, B. G. Recognition of patterns in the EEG. In: Computer Techniques in EEG Analysis. Mary Brazier (Ed). Elsevier Publishing Co., Amsterdam. 1961.

Feder, J. Languages of encoded line patterns. Information and Control. 13: 230-244. 1968.

Fu, K. S., Chien, Y. T. and Cardillo, G. P. A dynamic programming approach to sequential pattern recognition. IEEE. Trans. on Electronic Computers. EC-16: 790-803. December, 1967.

Fu, K. S., Landgrebe, D. A. and Phillips, T. L. Information processing of remotely sensed agricultural data. Proc. IEEE. 57: 639-653. April, 1969.

Fu, K. S., Pyung June Min, and Li, T. J. Feature selection in pattern recognition. IEEE. Trans. Systems Science and Cybernetics. SSC-6: 33-39. January, 1970.

Gibbs, F. A. and Grass, A. M. Frequency analysis of electroencephalograms. Science. 105: 132-134. 1947.

Goguen, J. A. L-Fuzzy Sets. J. of Math. Anal. Appl. 18: 145-174. 1967.

Grass, M. A. and Gibbs, F. A. A Fourier Transform of the EEG. J. Neurophysiol. 1938.

Guzman, A. Computer Recognition of Three-Dimensional Objects in a Visual Scene. Ph.D. Thesis, Elect. Eng. Dept., MIT. December, 1968.

Lee, E. T. and Zadeh, L. A note on fuzzy languages. Information Sciences. 1: pp. 421-432. 1969.

Lee, R. C. T. Fuzzy logic and the resolution principle. (to appear). 1971.

Levine, M. D. Feature extraction: a survey. Proc. of the IEEE. 57: 1391-1407. 1969.

Lonsdale, M.E. Development of a statistical analyser for random waveforms. State Univ. of Iowa. Doctoral Dissertation. 1952.

Marinos, P. N. Fuzzy logic and its application to

switching systems. IEEE. Trans. on Computers. C-18: 343-348. April, 1969.

Marril, T. and Green, D. M. Statistical recognition functions and the design of pattern recognizers. IRE Trans. Elec. Comp. 9: 472-477. 1960.

McCulloch, W. S. and Pitts, W. A logical calculus of the ideas immanent in neural nets. Bull. Math. Biophys. 5: 115-137. 1943.

Minsky, M. and Papert, S. Perceptrons: an Introduction to Computational Geometry. The MIT Press, Cambridge, Mass. 1969.

Mizumoto, M., Toyoda, J. and Tanaka, K. Some Some considerations on fuzzy automata. J. Computer and System Sciences. 3: 409-422. 1969.

Montanari, V. and Levi, G. A grey-weighted skeleton. Information and Control. 17: 62-91. 1970.

Nagy, G. State of the art in pattern recognition. Proc. of the IEEE. 56: 836-862. 1968.

Narasimhan, R. A linguistic approach to pattern recognition. Digital Computer Laboratory. Report No. 21. Univ. of Illinois. 1962.

Narasimhan, R. Labelling schemata and syntactic descriptions of pictures. Information and Control. 7: 151-179. 1964.

Narasimhan, R. Syntax-directed interpretation of classes of pictures. Comm. ACM. 9: 166-173. March, 1966.

Nilsson, N.J. A survey of the literature on problem-solving methods in artificial intelligence. Stanford Research Institute. Artificial Intelligence Group, Technical Note 21. 1970.

Nilsson, N. Problem-solving Methods in Artificial Intelligence. McGraw Hill. New York. 1971.

Palme, J. Making computers understand natural language. Operations Research Center. S-104 50. Stockholm 80, Sweden. July, 1970.

Pirotte, A. Natural language for the computer: A survey. Report R145. MBLE Research Lab. Brussels. June, 1970.

Quillian, M. R. The teachable language comprehender: a simulation program and theory of language.

Communications of the ACM. 12: 459-476. 1969.

Rosenblatt, F. Principles of Neurodynamics. Spartan Books. Washington, D. C. 1962.

Rosenblith, W. A. Processing Neuroelectric Data. MIT Press. Cambridge, Mass. 1962.

Remond, A. The importance of topographic data in EEG phenomena, and an electrical model to reproduce them. Electroenceph. clin. Neurophysiol. Suppl. 27. 1969.

Saltzberg, B., Burch, N. R., McLennan, M.A. and Correll, E. G. A new approach to signal analysis in electroencephalography. IRE Trans. Med. Electro. 8: p. 24. 1957.

Santos, E. S. Maximin automata. Information and Control. 13: 363-377. 1968.

Santos, E. S. Fuzzy algorithms. Information and Control. 17: 326-339. 1970.

Selfridge, O.G. Pandemonium: A paradigm for learning. In: Blake, D.V. and A. M. Uttley (Eds.). Proc. of the Symp. on Mechanization of Thought Processes. Natl. Physical. Lab. Teddington, Eng. H. M. Stationary Offices. London. 1959.

Shaw, A. C. The formal description and parsing of pictures. Stanford Linear Accelerator Center. Report No. 84: 1-205. 1968.

Simmons, R. F. Natural language question-answering systems: 1969. Comm. ACM. 13: 15-30. 1970.

Slagle, J. R. Heuristic search programs. In: Theoretical Approaches to Non-numerical Problem Solving. Banerji, R. and M. D. Mesarovic (Eds). Springer-Verlag. New York. 1970.

Slagle, J. R. and Dixon J. K. Experiments with the M & N Tree-searching program. Comm. ACM. 13: 147-154. March, 1970.

Slagle, J. R. Artificial Intelligence. The Heuristic Programming Approach. McGraw Hill. New York. 1971.

Slagle, J. R. and Lee, R. C. T. Application of game tree searching techniques to sequential pattern recognition. Comm. ACM. 14: 103-110. February, 1971.

Steingrandt, W. J. and Yau, S. S. Sequential feature extraction for waveform recognition. Proc. SJCC. pp. 65-76. 1970.

Walter, D. O. Spectral analysis for electroencephalograms: mathematical determination of neurophysiological relationships from records of limited duration. Exp. Neurol. 8: 155-181. August, 1963.

Walter, D. O. and Brazier, M. A. B. Advances in EEG analysis. Electroenceph. clin. Neurophysiol. Suppl. 27. 1969.

Walter, W. G. and Shipton, H. W. A new toposcopic display system. Electroenceph. clin. Neurophysiol. 3: 281-292. 1951.

Wee, W. G. On generalizations of adaptive algorithms and application of the fuzzy sets concept to pattern classification. Ph.D. Thesis. Dept. of Electrical Engineering. Purdue Univ. 1967.

Wald, A. Sequential Analysis. Wiley. New York. 1947.

Watanabe, S. (Ed.). Methodologies of Pattern Recognition. Acad. Press. New York. 1969.

Watanabe, S. Modified concepts of logic, probability, and information based on generalized continuous characteristic function. Information and Control. 15: 1-21. 1969.

Winograd, S. and Cowan, J. D. Reliable Computation in the Presence of Noise. MIT Press. Cambridge. 1963.

Young, F. M. Probability distribution of zero-crossing intervals. MIT Acoustics Laboratory Quarterly Progress Report. p. 3. 1954.

Zadeh, L. A. Fuzzy sets. Information and Control. 8: 338-353. 1965.

Zadeh, L. Fuzzy sets and systems. Proc. Symp. System Theory. Polytechnic Institute of Brooklyn. pp. 29-37, 1965.

Zadeh, L. Shadows of fuzzy sets. In: Problems of Information Transmission. (in Russian). 2: 37-44. March, 1966.

Zadeh, L. Fuzzy algorithms. Information and Control. 12: 94-102. 1968.

Zadeh, L. Probability measures of fuzzy events. J. Math. Anal. Appl. 10: 421-427. 1968.

Zadeh, L. Similarity relations and fuzzy orderings. Information Sciences. 3: 177-200. 1971.

Zadeh, L. Quantitative fuzzy semantics. Information Sciences. 3: 159-176. 1971.

Zahn, C. T. Graph-theoretical methods for detecting and describing gestalt clusters. IEEE Trans. C-20: 68-86. 1970.